T.C.
DÜZCE ÜNİVERSİTESİ
SAĞLIK BİLİMLERİ ENSTİTÜSÜ

SAĞLIK ALANINDA YAPILAN ARAŞTIRMALARDA KÜMELEME ALGORİTMALARININ KULLANIMI: BİR UYGULAMA

Özge PASİN

YÜKSEK LİSANS TEZİ

BİYOİSTATİSTİK VE TIBBİ BİLİŞİM ANABİLİM DALI

DANIŞMAN

Prof. Dr. Handan ANKARALI

Düzce, 2015

TEZ ONAYI

Biyoistatistik ve Tıbbi Bilişim Anabilim Dalı Yüksek Lisans Programı Çerçevesinde yürütülmüş olan "**Sağlık Alanında Yapılan Araştırmalarda Kümeleme Algoritmalarının Kullanımı: Bir Uygulama"** adlı çalışma, aşağıdaki jüri tarafından Oybirliği / Oy çokluğu ile Yüksek Lisans Tezi olarak kabul edilmiştir.

Tarih : / / 2015

TEZ SINAV JÜRİSİ

Prof.Dr.Handan ANKARALI
Düzce Üniversitesi
Jüri Başkanı

Yrd. Doç. Dr. Şengül CANGÜR
Düzce Üniversitesi
Üye

Yrd. Doç. Dr. Ünal Erkorkmaz
Sakarya Üniversitesi
Üye

Yukarıdaki Tez, Yönetim Kurulunun / / 2015 sayılı kararı ile kabul edilmiştir.

Prof.Dr. Recep ÖZMERDİVENLİ
Sağlık Bilimleri Enstitü Müdürü

BEYAN

Bu tez çalışmasının kendi çalışmam olduğunu, tezin planlanması aşamasından yazım aşamasına kadar bütün aşamalarda etik dışı davranışımın olmadığını, bu tezdeki bütün bilgileri akademik ve etik kurallar içinde elde ettiğimi, elde edilen bütün bilgi ve yorumlara kaynak gösterdiğimi ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını beyan ederim.

28 / 08 /2015

Özge PASİN

TEŞEKKÜR

Biyoistatistik mesleğini öğrenmede en önemli ve en temel basamaklardan biri olan yüksek lisans tezimin sonuna gelmiş bulunmaktayım. Bu aşamada her zaman yanımda olan ve beni destekleyen, manevi desteğini hep hissettiğim, beni kırmadan her zaman yardım eden yoğun çalışma zamanında bile benden yardımını esirgemeyen çok sevdiğim hocam Düzce Üniversitesi Biyoistatistik Anabilim Dalı Bölüm Başkanı Sevgili Prof. Handan Ankaralı' ya çok teşekkür ederim.

Beni hep destekleyen yardımlarını esirgemeyen, anlayış gösteren sayın hocalarım Yrd. Doç. Dr. Şengül Cangür ve Yrd. Doç. Dr. Mehmet Ali hocaya, tez zamanı boyunca hep yanımda olan desteğini asla esirgemeyen, benimle birlikte uykusuz kalan ve her zaman manevi destekçim olan canım arkadaşım Dr. Merve Alpay'a teşekkür ederim.

En zor zamanlarımda yanımda olan ve varlığı ile tez zamanımda sıkıntıya girmemi engelleyen ve beni hep neşelendiren canım ablam Dr. Tuğçe Pasin'e ve bugünlere gelmemde büyük emeği olan çok değerli olan annem Neşe Pasin ve babam Fatih Pasin'e çok teşekkür eder, Saygılarımı Sunarım.

Özge Pasin

İÇİNDEKİLER

TABLOLAR LİSTESİ

ŞEKİLLER LİSTESİ

KISALTMALAR VE SİMGELER

ACM : Association for Computing Machinery

ADCC : Adaptive Deflect and Conquer Clustering

ADULT : Adult Treatment Panel

AGNES : Agglomerative Nesting

AIC : Akaike Information Criterion

ASGC : Axis Shifted Grid Clustering

BIC : Bayesian Information Criterion

BIRCH : Balanced Iterative Reducing and Clustering Using Hierarchies

BSS : Kümeler arası kareler toplamı

CACTUS : Clutering Categorical Data Using Summaries

CDU : Yoğun aday birimi

CF : Clustering Feature

CF-tree : Clustering Feature Tree

CH : Calinski Harabasz

CHAMELEON : Hierarchical Clustering Algorithm Using Dynamic Modeling

CLARA : Clustering Large Applications

CLARANS : Clustering Large Applications Based on Randomized Search

CLATIN : Clustering Large Application with Triangular Irregular Network

CLICKS : Mining Subspace Clusters in Categorical Data Via K-Partite Maximal Cliques

CLIQUE : Clustering in Quest

CLOPE : A Fast and Effective Clustering Algorithm for Transactional Data

Cm : Santimetre

CoeffRT, CoeffDT : Ayarlanabilir katsayılar

COOLCAT : An Entropy Based Algorithm for Categorical Clustering

CS : Cluster Structure

CURE : Clustering Using Representatives

DBSCAN : A Density Based Spatial Clustering of Application With Noise

DCF : Distribution Cluster Features

DIANA : Divisive Hierarchical Clustering

DILCA : Distance Learning of Categorical Attribute

Dist : Uzaklık Matrisi

DNA : Timokinon, thymoquinone

DT : Yoğunluk eşik değeri

ECLARANS : Enhanced CLARANS

EM : Expectation Maximization

ENCLUS : Entropy Based Approach

Eps : Yakınlık mesafesi

ESS : Error Sum of Squares

GDILC : A Grid Based Density Isoline Clustering Algorithm

GKM : Global *K* modes clustering

GMM : Gaussian Mixed Model

HCAD : Hierarchical Clustering Based on Attribute Dependency Algorithm

HDL : High Density Lipoprotein

HIERDENC : Hierarchical Density Based Clustering of Categorical Data

I : Input

IB : Information Bottleneck

KDD : Knowledge Discovery and Data Mining

Kg : Kilogram

Kg/m^2 : Kilogram/metrekare

LIMBO : Scalable Information Bottleneck

MAFIA : Merging of Adaptive Finite Intervals

Mg/dl : Miligram/desilitre

MinPts : Bölgenin yoğun olarak belirlenebilmesi için ε komşuluğundaki minimum nokta sayısı

MPFD : Modified Partition Fuzzy Degree

MULIC : Multiple Layer Clustering

NSGC : New Shifting Grid Clustering Algorithm

O : Output

OPTICS : Ordering Points to İdentify Clustering Structure

PAM : Partitioning Around Medoids

PFD : Partition Fuzzy Degree

QROCK : Quick Robust Clustering Using Links

RC : Nispi Yakınlık

RI : Nispi Bağlanabilirlik

ROCK : Robust Clustering Using Links

RT : Komşu bölge genişliği

SIAM : Society for Industrial and Applied Mathematics

SIGKDD : Special Interest Group on Knowledge Discovery and Data Mining

SIGMOD : Special Interest Group on Management of Data

Sim : Similarity, Benzerlik

SPSS : Statistical Package for the Social Sciences

STING : Statistical Information Grid-Based Clustering Algorithm

STIRR : Sieving Through Iterated Relational Reinforcement

TP-PAM : Tree Pruning PAM Algorithm

TSS : Total Sum Of Squares

UPGMA : Unweighted Pair Group Method With Averaging

UPGMC : Unweighted Pair Group Centroid

WCD : Within Cluster Distances

WPGMA : Weighted pair group method with arithmetic mean

WPGMC : Weighted pair group centroid method

WSS : Kümeler içi kareler toplamı

ÖZET

SAĞLIK ALANINDA YAPILAN ARAŞTIRMALARDA KÜMELEME ALGORİTMALARININ KULLANIMI: BİR UYGULAMA

Özge PASİN

Yüksek Lisans Tezi, Biyoistatistik ve Tıbbi Bilişim Anabilim Dalı

Tez Danışmanı: Prof. Dr. Handan ANKARALI

EYLÜL 2015, 240 Sayfa

Kümeleme yöntemleri ile benzer özelliklere sahip değişken ve bireyler bir grupta toplanabilmektedir. Birçok uygulama alanına sahip olmasına rağmen kümeleme yöntemi ülkemizde sağlık araştırmalarında nadir olarak kullanılmaktadır. Bu tez çalışmasının amacı, farklı kümeleme algoritmalarını tanıtmak ve bu algoritmaların nasıl ve hangi durumlarda doğru bir şekilde kullanılabileceğini göstermektir. Aynı zamanda sağlık alanından elde edilmiş gerçek bir veri seti üzerinde uygulanabilir olan farklı kümeleme algoritmalarının sonuçlarını karşılaştırmaktır. Yapılan değerlendirmeler sonucunda kullanılan iki farklı veri seti için hesaplanan kappa katsayıları istatistiksel olarak orta düzeyde anlamlı bulundu. Gerçekleştirilen uygulama sonucunda her iki veri seti için de kappa katsayısı bakımından en uygun ve en hızlı sonuçlar üreten algoritmanın *En Uzak İlk Kümeleme Yöntemi* olduğu sonucuna varıldı. Framingham risk grupları ile oluşturulan kümeler arasında çapraz tablolar oluşturularak grupların dağılımı incelendiğinde ise, en isabetli kararların *Make Density Based* ve *EM algoritmalarıyla* elde edilen kümeleme sonuçları olduğu görüldü. Sonuç olarak kümeleme yöntemlerinin hastalıklara ait risk faktörlerinin incelenmesinde, klinik bilgileri de dikkate alarak hastalık gruplarının oluşturulmasında ve buna bağlı olarak da doğru hastalık teşhislerinin konulmasında önemli bir rol oynacağı düşünülmektedir. Ayrıca veri dağılımı ve özellikleri dikkate alınarak kullanıldığında kümeleme algoritmalarının, sağlık alanında her türlü planlama ve hastalık teşhisi için bir tanı aracı olarak kullanılabileceği kanısındayız.

Anahtar Sözcükler: Algoritma, Framingham risk skoru, Kümeleme Analizi, Veri Madenciliği

ABSTRACT

USAGE OF CLUSTER ALGORITHMS IN HEALTH STUDIES: AN APPLICATION

Özge PASİN

Master of Science Thesis, Biostatistics and Medical Informatics Department

Supervisor: Prof. Dr. Handan ANKARALI

SEPTEMBER 2015, 240 Pages

With clustering methods variable and individuals which have similar characteristics may be collected in a group. Although clustering methods have many applications, there are limited studies in health researches in our country. While the purpose of this study is to introduce different clustering algorithms and show how and which cases should be correctly used. At the same time, different clustering algorithms results which can be applied on a real data set were compared. According to the evaluations, for two different data sets the kappa coefficients were statististically significant and its degree are intermediate. In terms of both data sets the most convenient and fastest algorithm is *Farthest* clustering algorithm. The results obtained by *Make Density Based* and *EM* algorithms gave the most accurate desicions in terms of the distribution of the groups among Framingham risk groups crosstables. As a result, with taking into account the criterion of clinical information it is thought that the examination of clustering of risk factors of the disease, will be played an important role for introduction of accurate disease diagnosis. In addition we believe that when considering data distribution and characteristics of data sets clustering algorithms can be used as a diagnostic tool for the plannings and diagnosis of diseases in the field of health.

Key Words: Algorithm, Framingham Risk Score, Cluster Analysis, Data Mining

1. GİRİŞ

Günümüz bilimi kanıta dayalı bilgiyi kabul etmektedir. Bilimde kanıt doğru elde edilmiş veri olarak kabul edilmektedir. Verinin varlığı birtakım hesaplamaları birlikte getirir ki bu hesaplamalarda istatistik yöntem ve prensiplerden yararlanılır. Bilgi ve teknolojinin takip edilemez hızda arttığı ve ilerlediği herkes tarafından kabul edilmektedir. Yoğun bilgi kümesinden faydalı ve yararlı sonuçlar elde edebilmek için daha kapsamlı ve daha ileri istatistik yöntemlerin kullanımı nerdeyse zorunlu hale gelmiştir. Teknolojinin özellikle internet sonrası hızla gelişmesi, teorisi ortaya atılmış istatistik yöntemlerin uygulama alanına hızla girmesine neden olmuştur. Bu yöntemlerin kullanılması ise karmaşık bilgiyi daha iyi anlamamıza ve gerçek dünyayı daha iyi yorumlamamıza neden olmaktadır.

Çok sayıda kişinin çok sayıda özelliğine ait bilgilerin yer aldığı durumlarda verileri daha iyi değerlendirmek amacıyla geliştirilen yöntemler, "Veri Madenciliği" genel başlığı altında toplanmıştır. Her alanda olduğu gibi sağlık alanında da kayıtlar artık bilgisayar ortamına yani veri tabanlarına aktarılmaktadır. Böylece veriler uzun süre saklanabilmekte ve kullanışlılık özelliği kazanmaktadır. Ayrıca zaman içerisinde çok sayıda bilgiye kolayca ulşma imkânı doğmaktadır. Veri tabanları planlı oluşturulduğunda birçok yeni hipotezin doğmasına ve test edilmesine olanak sağlamaktadır.

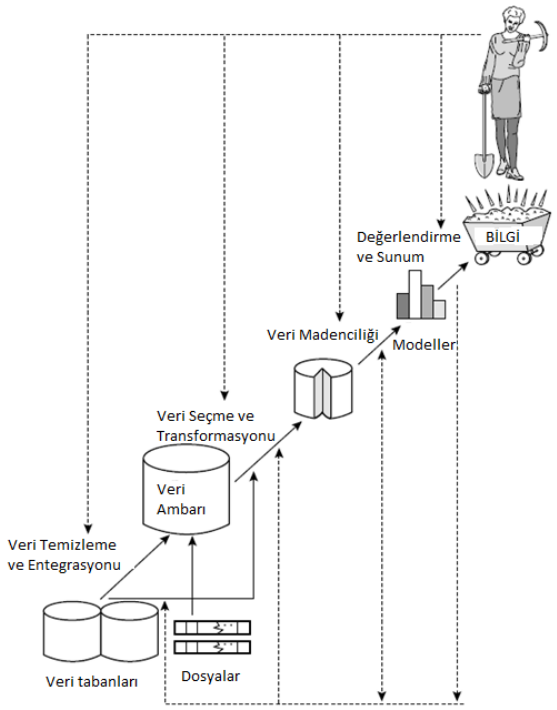
Veri Madenciliği başlığı altında yer alan ve özellikle son 10 yılda yaygın kullanım alanı bulan tanı koyma, sınıflama, gruplama ve tahmin etme amaçlarıyla geliştirilen çok sayıda algoritma mevcuttur. Sağlık alanı araştırmalarında da birçok hipotezin temel amacı bunlardan biri veya birkaçıdır. Tanı koyma, sınıflama veya tahmin modellerinin, sağlık alanında olduğu gibi diğer alanlarda da yaygın bir şekilde kullanılmasına karşın bireyleri veya özellikleri gruplama veya kümeleme amacıyla geliştirilen yöntemlerin uygulamada çok yaygın kullanılmadığı görülmektedir. Bu sonuç bize, araştırmacıların kümeleme yöntemleri yardımıyla elde edecekleri bilgiyi nerede ve nasıl kullanılacakları ve sonuçlarını nasıl yorumlayacakları konularında bilgi eksiği olduğunu düşündürmüştür. Ayrıca yaygın kullanılan istatistik paket programlarında var olan birkaç kümeleme algoritmasının dışında özellikle son yıllarda geliştirilmiş yeni kümeleme algoritmalarının literatürde kullanımı sınırlı sayıdadır. Söz konusu eksikliklerden yola çıkarak bu tez çalışmasında, teorisi literatüre geçmiş farklı kümeleme algoritmalarının karşılaştırmalı olarak tanıtılması amaçlanmış ayrıca uygulamada

yaygın kullanılmamış bazı algoritmaların sağlık alanından elde edilen bir veri seti üzerinde uygulaması yapılarak elde edilen sonuçların nasıl yorumlanacağı gösterilmiştir. Bu tez çalışması sonucunda ülkemizde yapılacak bilimsel çalışmalarda kümeleme algoritmalarından ne zaman ve nasıl yararlanılacağı ortaya konmuş olacak ve yeni değerlendirme yöntemleri yardımıyla bilimsel çalışmaların ürettiği bilgilerin kalitesi daha da yükselmiş olacaktır.

2. GENEL BİLGİLER

2.1. Veri Madenciliği ve Kümeleme Analizi

Veri madenciliği, büyük miktardaki verilerin ayıklanması veya maden araması olarak ifade edilmektedir. Ancak bu terim yanlış kullanılabilmektedir. Örneğin, taşlardan ve kumlarda altın arandığında taş veya kum araması yerine altın araması terimi kullanılmaktadır. Dolayısıyla veri madenciliği terimi yerine "verilerden elde edilen bilgi madenciliği" teriminin kullanılması önerilmektedir. Ancak bu adlandırma oldukça uzundur. Bilgi madenciliği terimi ise kısa bir kavramdır ve madenciliğindeki büyük miktardaki verileri tam olarak yansıtmamaktadır. Madencilik terimi süreci karakterize eden bir kelime olarak kullanılabilmektedir. Dolayısıyla popüler tercih olarak veri ve madencilik kelimelerinin karşımından oluşan veri madenciliği terimi sıklıkla kullanılmaktadır. Veri madenciliğinin amacı, terabyte boyutunda çok büyük miktardaki verileri işe yarayabilir şekile dönüştürmektedir. Veri madenciliği birbirini tekrarlayan aşamalar sonucunda gerçekleşmektedir. Bu aşamalar aşağıdaki gibi özetlenmiştir ve Şekil 2.1.1'de gösterilmiştir [1,2].



Şekil 2.1.1. Veri madenciliği aşamaları

Veri madenciliği aşamalarında karşılaşılan terimler kısaca şöyle açıklanabilir.

1. *Veri Temizleme:* Gürültülü verileri ve tutarsız verileri atılmaktadır.
2. *Veri Entagrasyonu:* Birçok veri kaynakları bu aşamada kombine edilmektedir.
3. *Veri Seçme:* Analiz yapılan veriler veri tabanından seçilerek alınmaktadır.
4. *Veri Transformasyonu:* Veriler özetlenerek uygulamaya hazır hale getirmektir.
5. *Veri Madenciliği:* Veri yapılarını keşfetmek için ileri teknikler kullanılmaktadır.
6. *Örüntü Değerlendirme:* Ölçülere dayanarak en doğru örüntüleri belirlemektir.
7. *Bilgi Sunumu:* Örüntülerin gösterimi için uygun teknikleri kullanmaktır.

Veri madenciliği başlığı altında yer alan algoritmalar yardımıyla tahmin, sınıflama ve kümeleme işlemleri yapılmaktadır. Bu yöntemler 1960' lı yıllardan itibaren çeşitli alanlarda kullanılmıştır. Örneğin, kredi skorlamada, dolandırıların tespit edilmesinde, market araştırmalarında, perakendecilikte, market bölümlerinin düzen ve bölümlerinin oluşturulmasında, planlamaların bakım ve kontrollerinin kalitelerinin değerlendirilmesi gibi. Ancak bu gruba giren birçok yöntemin kullanılabilmesi için ileri bilgisayar teknolojisine ihtiyaç duyulması nedeniyle özellikle internet çağından itibaren yaygın kullanılmaya başlanmıştır. Sağlık alanında yapılan araştırmalarda da veri madenciliği yöntemlerinin kullanım sıklığı gün geçtikçe artmaktadır. Örneğin poliklinik ve hastanelerde hastalara ait kayıtlar idari işlerde saklanmaktadır. Hastalara ait bu bilgiler tıp alanında oldukça önemli hatta hastaların hayati durumlarını etkileyecek kadar önem arz etmektedir. Hastalara ait kayıtların doğru ve etkili bir şekilde kullanılması halinde örneğin hastalık teşhisinde önemli adımları atılmış olunacaktır. Aynı zamanda veri madenciliği sayesinde hastanedeki kaynak ve maliyetlerin doğru, faydalı ve etkili bir şekilde en iyi nasıl kullanılacağına ait bir planlama olanağı da sağlanmaktadır. Son yıllarda ise kanser hastalıklarının araştırılmasında veri madenciliğinden yararlanılmaktadır. DNA dizileri incelenerek genetik hastalıklara sebep olabilecek mutasyonlar ve genetik bozukluklar incelenebilmektedir[3].

Kümeleme benzer nesneleri gruplandırma işlemidir. Bir küme içerisindeki nesneler, çalışılan özellikler bakımından birbirine benzer iken, farklı kümelerde yer alan nesneler söz konusu özellikler bakımından birbirine benzemez. Bir başka ifadeyle kümeleme yardımıyla homojen gruplar elde edilir. Bu kümeleme analizinin temelini oluşturmaktadır. Bu analiz denetimsiz (unsupervised) bir öğrenme şeklidir. Çünkü karışık halde bulunan ve kimlerin aynı gruba

gireceği önceden bilinmeyen bir veri setinde, kümeleme analizi yardımıyla homojen gruplar ortaya çıkarılmaktadır. Böylece çok sayıdaki nesne anlamlı gruplara ayrılmış olacaktır. Bu yöntemler sayesinde sadece nesneler değil özellikler de kümelenebilmektedir. Bu işlemin sonucu ise faktör analizi ile oldukça benzerlik göstermektedir. Bir yönüyle çok sayıdaki özelliği kümeleyerek veri de boyut indirgemesi sağlanmış olmaktadır. Pratikte özelliklerin kümelenmesinde faktör analizi veya yapısal eşitlik modelleri gibi yöntemler kullanıldığı için bu amaçla kümeleme analizine ihtiyaç duyulmamaktadır. Bu yüzden bundan sonraki anlatımlarda nesnelerin kümelenmesi konusu üzerinde açıklamalar yapılacaktır[3,4,5].

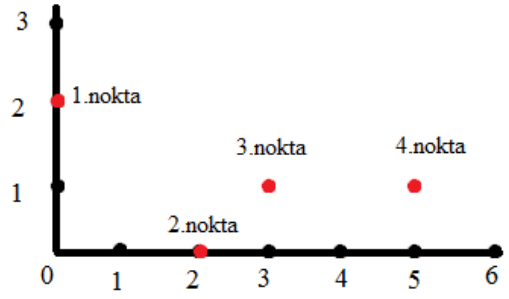
Farkında olmadan hayatımızın her alanında belli kriterleri dikkate alarak kişileri ve/veya nesneleri kümeleme işlemini sıklıkla yapmaktayız. Örneğin, çocukluk çağında kedi ve köpekleri, bitki ve hayvanları kolayca ayırt edebilmek için bazı özellikler dikkate alınır. Dolayısıyla bilinçaltımızda sürekli olarak kümeleme şemaları geliştirmekteyiz. Kümeleme analizi birçok farklı disiplinde farklı şekillerde ve amaçlarda kullanılmaktadır. Örneğin, hastalıkların coğrafik dağılımının belirlenmesi, trafik kazalarının kümelenmesi, hastanedeki personellerin yönetimi, hastane koşullarının iyileştirilmesi, ambulans hizmetlerinin ulaşım zamanlanması, hastalıkların teşhisi, katil profillerinin çıkarılması, obezite gruplarının belirlenmesi, benzer hastalıkların ortaya konması gibi birçok amaç için kullanılabilir[3,4,5].

Kümeleme analizi bazen ara analiz yöntemi olarak da kullanılmaktadır. Lojistik regresyon, diskriminant analizi, tek yönlü varyans analizi yöntemleri gibi birçok istatistik yöntem ile kombine bir şekilde kullanıldığı görülmektedir. Bu gibi durumlarda önce kümeleme analizi yardımıyla homojen gruplar oluşturulur daha sonra söz konusu diğer yöntemlerle bu grupların hangilerinin en iyi şekilde ayrıştırılabildiği incelenebilir veya oluşan homojen grupların her birisinde ayrı ayrı işlemler yapılabilir[3,4,5].

Etkili ve doğru bir kümeleme algoritmasının taşıması gereken bazı temel özellikler mevcuttur. Uygun bir kümeleme algoritması veri tabanının tek bir seferde tarayarak farklı şekillere sahip ve farklı genişliklerdeki küme yapılarını keşfetmeli ve aynı zamanda niteliksel ve niceliksel olmak üzere tüm veri türlerine uygulanabilir olmalıdır. Etkili bir kümeleme yöntemi, veri tabanı büyüklüğü ayırt etmeden büyük ve küçük veri tabanlarının her ikisi içinde elverişli olmalıdır. Bu aynı zamanda kümeleme algoritmasının ölçeklenebilirlik özelliğine sahip olup olamdığını göstermektedir. İyi bir kümeleme algoritması etkili ve sapan verilere karşı ne yapması gerektiği bilmeli ve etkilenmemelidir. Bahsedilen kriterlerin yanında iyi bir

kümeleme algoritması uygulanması kolay, yorumlanabilir, fonksiyonel ve anlaşılır olmalıdır[3,4,5].

Kümeleme algoritmalarının bazılarında orijinal veri matrisi kullanılır iken bir çoğunda benzerlik matrisi, *S*, veya uzaklık matrisi, *D*, kullanılmaktadır. Bu iki matrise genel olarak yakınlık matrisi (proximity) , *P*, adı verilmektedir. Yakınlık matrisi *nxn* boyutlu olup nesneler veya objeler arasındaki bütün ikili uzaklık veya benzerlik ölçülerini içerir. Örneğin 4 noktadan ölçülen *X* ve *Y* özelliklerine ait değerler Şekil 2.1.2' de koordinat ekseninde gösterilmiştir. Orijinal veri matrisi grafiğin hemen sol altında yer almakta olup bu noktalar arasında hesaplanan yakınlık değerleri grafiğin altındaki Tablo 1'de topluca verilmiştir. Birinci ve ikinci noktanın birbirine yakınlığı $\sqrt{(0-2)^2+(2-0)^2}$=2,828 formülü ile hesaplanır. Diğer yakınlık değerleri de benzer şekilde bulunmaktadır. Yakınlık matrisinin köşegen elemanları aynı noktanın kendine olan yakınlığı olduğu için sıfır değerini alır[6].



Şekil 2.1.2. *X* ve *Y* özelliklerinin koordinat ekseninde gösterimi

Tablo 2.1.1. Orijinal veri matrisi ve noktalar arasındaki yakınlık matrisi

Noktalar	X	Y
1.Nokta	0	2
2.Nokta	2	0
3.Nokta	3	1
4.Nokta	5	1

Orijinal veri

	1.Nokta	2.Nokta	3.Nokta	4.Nokta
1.Nokta	0	2,828	3,162	5,099
2.Nokta	2,828	0	1,414	3,162
3.Nokta	3,162	1,414	0	2
4.Nokta	5,099	3,162	2	0

Yakınlık matrisi

Benzerlik matrisi ise nesneler arasındaki yapıları ortaya koymaktadır ve aşağıdaki gibi oluşturulmaktadır[7].

$$\begin{matrix} 0 & & & \\ d(2,1) & 0 & & \\ d(3,1) & d(3,2) & 0 & \\ \dots & \dots & \dots & \dots \\ d(n,1) & d(n,2) & \dots. & 0 \end{matrix}$$

Yukarıdaki matriste yer alan *d(i,j)* değeri *i* ve *j* nesneleri arasındaki benzememezlik ölçüsüdür. Bu ölçünün alacağı değer sıfıra yaklaştıkça *i* ve *j* nesnelerinin birbirine daha benzer olduğu, sıfırdan uzaklaştıkça nesnelerin farklılaştığı söylenir. Benzerlik matrisi simetrik bir matristir. Bir başka ifadaeyle *d(i,j)=d(j,i)*'dir ve *d(i,i)* değerleri sıfıra eşittir. Benzerlik matrisinde elde edilen *d(i,j)* değerleri verinin yapısına göre farklı şekillerde hesaplanmaktadır. Bilindiği üzere bir veri setinde değişkenler niceliksel ve niteliksel olmak üzere iki ana kategoriye ayrılmaktadır. Niteliksel değişkenlerde kendi arasında sınıflandırılmış (nominal) ve sıralanmış (ordinal) olmak üzere ikiye ayrılmaktadır. Eğer sınıflandırılmış değişkende kategori sayısı iki ise bu değişken ikili (binary) değişken olarak da adlandırılmaktadır. Niceliksel değişkenler ise oransal ve eşit aralıklı olmak üzere iki alt bölümde incelenmektedir. Hesaplamalarda kullanılacak benzerlik matrisleri, değişken tipine göre aşağıdaki gibi hesaplanır[7].

Aralıklı ölçekli değişkenler için hesaplamalar

Değişkenlerin ölçü birimleri kümeleme analizi sonuçlarını etkileyebilmektedir. Örneğin, boy uzunluğunun metre veya santimetre olarak ifade edilmesi farklı sonuçların elde edilmesini sağlar. Genellikle değişken küçük birimler ile ifade edildiğinde kümeleme yapısına daha büyük etkisi olacaktır. Ölçü biriminin seçiminden kaynaklanabilecek olan bağımlılığı ortadan kaldırmak amacıyla verinin standartlaştırılması gerekmektedir. Özellikle veri hakkında herhangi bir ön bilgi olmadığında standartlaştırma oldukça kullanışlı bir yöntemdir. Ancak bazı durumlarda bazı değişkenlere daha çok ağırlık verilmek istenebilmektedir. Örneğin basketbol oynayanların kümelenme durumu incelenmek istendiğinde, ağırlığa boy uzunluğuna göre daha fazla önem verilmek istenebilir[7].

Ölçüleri standardize edebilmek için, *p* değişken olduğu varsayılsın. İlk adımda ortalama mutlak sapma değeri aşağıdaki gibi hesaplanmaktadır[7].

$$s_p = \frac{1}{n}\left(|x_{1p} - m_p| + |x_{2p} - m_p| + \cdots + |x_{np} - m_p|\right)$$

$x_{1p}, x_{2p}, \dots, x_{np}$ değerleri p değişkenin n adet birimden elde edilen ölçümleri iken, m_p, f değişkeninin bütün birimler dikkate alınarak hesaplanan ortalama değeridir ve $m_p=\frac{1}{n}(x_{1p} + x_{2p} + \cdots + x_{np})$ şeklinde hesaplanmaktadır. Ortalama mutlak sapma değeri hesaplanırken, ortalamadan sapmaların kareleri alınmamaktadır böylece uç değerlerin etkileri azaltılmış olunur[7].

İkinci adımda ölçümlerin standart ölçüleri (z-skorları) aşağıdaki gibi hesaplanır[7].

$$z_{ip} = \frac{x_{ip} - m_p}{s_p}$$

Standartlaştırma işlemi bazı özel uygulamalarda kullanışlı olamayabilir. Dolayısıyla kümeleme analizi yapılmadan önce standartlaştırma işleminin yapılıp yapılmayacağına kullanıcının karar vermesinde fayda vardır. Standarlaştırma işlemi yapmadan veya yapıldıktan sonra nesneler arasında benzelik matrisinin hesaplanması gerekmektedir. Aralık düzeyinde ölçülmüş değişkenler için benzerlik matrisi kullanılırken en çok kullanılan ölçümlerden biri Öklid uzaklığıdır ve aşağıdaki gibi hesaplanmaktadır[7].

$$d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{ip} - x_{jp})^2}$$

i=$(x_{i1}, x_{i2}, \dots, x_{ip})$ ve j=$(x_{j1}, x_{j2}, \dots, x_{jp})$ olmak üzere i ve j nesnelerine ait p boyutlu vektördür.

Diğer bilinen ölçü ise Manhattan (city block) uzaklık ölçüsüdür ve aşağıdaki gibi hesaplanmaktadır[7].

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \cdots + |x_{ip} - x_{jp}|$$

Her iki uzaklık ölçüsüde aşağıdaki koşulları yerine getirmektedir.

1. $d(i,j) \geq 0$
2. $d(i,i)$=0
3. $d(i,j) = d(j,i)$
4. $d(i,j) \leq d(i,h) + d(h,j)$

Minkowski uzaklığı ise Öklid ve Manhattan uzaklık ölçülerinin genelleştirilmiş bir halidir ve aşağıdaki gibi hesaplanmaktadır[7].

$$d(i,j) = (|x_{i1} - x_{j1}|^{\lambda} + |x_{i2} - x_{j2}|^{\lambda} + \cdots + |x_{ip} - x_{jp}|^{\lambda})^{1/\lambda}$$

Yukarıdaki eşitlikte geçen λ terimi, pozitif bir tamsayıdır.

Her bir değişken için farklı ağırlık değerleri mevcut olduğu durumlarda ağırlıklandırılmış Öklid uzaklığı kullanılmaktadır ve aşağıdaki formül yardımıyla hesaplanmaktadır. Formülde yer alan her bir *w* değerleri değişkenlerin ağırlıklarını ifade etmektedir[7].

$$d(i,j) = \sqrt{w_1(x_{i1} - x_{j1})^2 + w_2(x_{i2} - x_{j2})^2 + \cdots + w_m(x_{ip} - x_{jp})^2}$$

Bu ağırlıklandırma işlemi aynı zamanda Manhattan ve Minkowski uzaklık ölçüleri için de uygulanabilmektedir[7].

İkili değişkenler için hesaplamalar

İkili değişenler 0 ve 1 olmak üzere iki değer almaktadır. 0 değeri yokluğu, 1 değeri ise varlığı ifade etmektedir. Örneğin alkol kullanımı sorgulandığında, bu değişkenin sıfır değerini alması alkol kullanımının olmadığını, bir değerini alması ise alkol kullanımının olduğunu göstermektedir. Eğer tüm ikili değişkenlerin aynı ağırlığa sahip olduğu düşünülürse 2x2'lik kontenjans tablosu Tablo 2.1.2'deki gibi oluşturulur. Bu tabloda satıra i numaralı nesne, sütuna j numaralı nesne yerleştirilir. Daha sonra her iki nesnenin de 1 değerini aldığı değişken sayısı (*q),* her iki nesnenin de 0 değerini aldığı değişken sayısı (t), i numaralı nesnenin 0 ve j numaralı nesnenin 1 değerini aldığı değişken sayısı (s) ve son olarak i numaralı nesnenin 1 ve j numaralı nesnenin 0 değerini aldığı değişken sayısı (r) belirlenir. Daha sonra toplam değişken sayısı $p=q+r+s+t$ eşitliği ile elde edilir[7].

Tablo 2.1.2. *i* ve *j* nesneleri için oluşturulan kontejyans tablosu

		j nesnesi		Toplam
		1	0	
i nesnesi	1	q	r	$q+r$
	0	s	t	$s+t$
	Toplam	$q+s$	$r+t$	p

İkili değişkenlerde kendi aralarında simetrik ve asimetrik olmak üzere iki grupta incelenmektedir. Simetrik ikili değişkenlerde bağımlı değişkenin hangi sonucunun sıfır hangi sonucunun bir değerini aldığının bir önemi yoktur. Örneğin cinsiyet durumunda tamamen rasgele olarak her hangi bir cinsiyete 0, diğer cinsiyete 1 kodu verilebilmektedir. Bu değişkenlerin benzerlik matrisleri oluşturulurken aşağıdaki uzaklık ölçüsü değeri hesaplanmaktadır[7].

$$d(i,j) = \frac{r+s}{q+r+s+t}$$

Asimetrik ikili değişken türlerinde sonucun eşit derecede önemli olmadığı durumlarda kullanılmaktadır. Örneğin hastalık testi sonucunda negatif veya pozitif değerin alınması gibi. Asimetrik ikili değişkene ait iki sonucun da sıfır olması durumunda negatif eşleşleme (a negative match), 1 olması durumunda ise pozitif eşleşme (a positive match) söz konusudur. Bu değişken tiplerinde uzaklık ölçüleri hesaplanırken, hesaplamalarda *t*' nin değeri önemli kabul edilmeyip uzaklık değeri aşağıdaki gibi hesaplanmaktadır[7].

$$d(i,j) = \frac{r+s}{q+r+s}$$

İki tane ikili değişken arasında uzaklıklar hesaplanırken uzaklıklar yerine benzerlik kavramı da kullanılabilmektedir. Örneğin, i ve j nesneleri arasındaki benzerlik aşağıdaki gibi hesaplanabilmektedir[7].

$$sim(i,j) = \frac{q}{q+r+s} = 1 - d(i,j)$$

Yukarıdaki $sim(i,j)$ terimi Jaccard katsayısı olarak bilinmektedir ve literatürde sıklıkla kullanılmaktadır.

Sınıflandırılmış ve Sıralanmış değişkenler için hesaplamalar

Sınıflandırılmış değişken ikili değişkenin genellenmiş halidir. Kategori sayısı 2' den fazladır. Mesela saç rengi, bölgeler, meslek, medeni durum gibi değişkenler bu gruba girer. Veri

setinde sınıflandırılmış değişkenler bulunduğu zaman *i* ve *j* ile gösterilen iki nesne arasındaki uzaklık ölçüsü aşağıdaki formül yardımıyla hesaplanır[7].

$$d(i,j) = \frac{p - m}{p}$$

Bu eşitlikte *p*: kategorik yapıdaki toplam çalışılan değişken sayısını, *m* ise *i* ve *j* nesnelerinin aynı sonucu almış kategorik değişken sayısını göstermektedir. Bu formül sadece sınıflandırılmış değişkenler için kullanılır[7].

Sıralanmış değişkenler kategorik değişkenler arasında yer almaktadır. Ancak sıralanmış değişkenlerde kategoriler arasında üstünlük, büyüklük, küçüklük ilişkileri vardır. Örneğin, kanser evreleri, akademik basamaklar, ağrı şiddeti, eğitim seviyesi gibi. Bir veri setinde sıralanmış yapıda olan değişken sayısı f adet kabul edilirse bu değişkenlere ait değerlerin sıralaması 1,…., M_f olarak gösterilir. İki nesne (i ve j) arasındaki uzaklık ölçüsü aşağıdaki gibi hesaplanmaktadır[7].

i. nesnenin f değeri x_{if} ve M_f, f'in sıralanmış durumları olmak üzere, her bir x_{if} değeri karşılık gelen rankı (r_{if}) ile yer değiştirilmektedir.

Her bir sıralanmış değişkenin farklı sayıda değeri mevcuttur. Bu yüzden değişken değerleri [0,1] aralığında standartlaştırılmaktadır. Böylece her bir değişkene eşit ağırlık verilmiş olunacaktır. Bu işlem ise aşağıdaki formül yardımıyla elde edilmektedir[7].

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

Yukarıdaki işlemin ardından aralık ölçekli değişkenler için kullanılan uzaklık ölçülerinden herhangi birinin formulünde *i.* denek için *f* değişkeni yerine z_{if} değeri kullanılarak kullanılarak benzememezlik değeri hesaplanabilmektedir[7].

Oran ölçekli değişkenler için hesaplamalar

Oransal ölçeklerden sağlık alanında sıklıkla yararlanılmaktadır. Örneğin fiziksel ölçümler, kan parametreleri gibi. Bu ölçekteki değişkenler değerlerine tüm matematiksel işlemler uygulanabilmektedir[7,8].

Oran ölçekli değişkenlerde nesneler arasında benzerlik hesaplanırken üç yöntem kullanılmaktadır. Bunlardan birincisi oran ölçekli değişkenler için de aralık ölçek düzeyindeki değişkenler için kullanılan ölçümlerin kullanılmasıdır. Ancak bu yaklaşım uygulamada çok kullanışı değildir. Çünkü ölçek bozulabilmektedir. İkinci bir yaklaşım ise oran düzeyinde ölçülmüş değişkenler için logaritmik transformasyonlar uygulamaktır. Transformasyon sonucunda elde edilen y_{if} değerlerine aralık düzeyde ölçülen değişkenlerin ölçüm yöntemleri uygulanabilmektedir. Üçüncü ve son yöntem ise bu değişkenlere sürekli sıralanmış veri gibi davranmaktır. Sıralanmış değerleri aralık değerli gibi davranılmaktadır. Son iki yöntem birinci yönteme göre daha etkilidir. Hangi yöntemin seçileceğine verilen uygulamaya göre karar verilmektedir[7,8].

Karışık tipteki değişkenleri içeren veri setleri için hesaplamalar

Gerçek uygulama verilerinde genellikle tüm değişken türleri karmaşık olarak yer almaktadır. Bu durumda p tane karışık tipteki değişkenlerin olduğu varsayımı altında uzaklık ölçüsü $d(i,j)$ aşağıdaki gibi hesaplanamaktadır[7,8].

$$d(i,j) = \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{(f)}}$$

Formülde yer alan $\delta_{ij}^{(f)}$, x_{if} veya x_{jf} olduğunda, $x_{if} = x_{jf} = 0$ ve *f* değişkeni asimetrik ikili değişken olduğunda sıfıra eşittir, diğer durumlarda ise $\delta_{ij}^{(f)}$=1'dir. *i* ve *j* arasındaki uzaklığa *f* değişkeninin katkısı olan $d_{ij}^{(f)}$ değerleri, farklı değişken tipleri için farklı şekillerde hesaplanmaktadır[7,8].

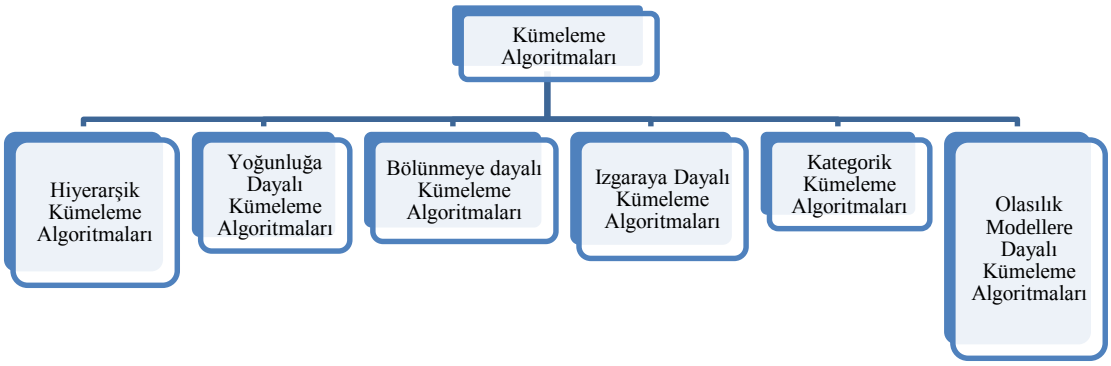
f değişkeni aralık ölçme düzeyinde ise, $d_{ij}^{(f)}=\frac{|x_{if}-x_{jf}|}{max_h x_{hf}-min_h x_{hf}}$, *h; f* değişkeni için tüm kayıp olmayan nesneler üzerinde çalışmaktadır.

f değişkeni ikili veya kategorik olduğunda, $x_{if} = x_{jf}$ durumunda $d_{ij}^{(f)} = 0$'dır. Diğer durumlarda $d_{ij}^{(f)}$=1'dir[7,8].

f değişkeni sıralanmış kategorik değişken ise, r_{if}; rank değeri hesaplanır ve $z_{if}=\frac{r_{if}-1}{M_f-1}$'dir. Standartlaştırmanın ardından aralıklı ölçek gibi hesaplanmaktadır[7,8].

f değişkeni oran ölçekte ölçülmüş bir değişken ise, ya logaritmik dönüşüm uygulanarak tranformasyon yapılan veriye aralık ölçek gibi davranılır ya da f değişkeni sürekli sıralanmış veri olarak düşünülerek r_{if} ve z_{if} değerlerinin hesaplanmasının ardından z_{if} değerlerine aralık değişkenmiş gibi davranılmaktadır[7,8].

Veri madenciliğinde kullanılan kümeleme algoritmaları, altı temel grup altında incelenir. Bu grupla Şekil 2.1.3' de topluca verilmiştir.



Şekil 2.1.3. Altı temel kümeleme grubu

İlerleyen bölümlerde bu altı temel kümeleme grubu içinde yer alan kümeleme algoritmaları tanıtılacaktır.

2.1.1. Hiyerarşik Kümeleme Algoritmaları

Hiyerarşik kümeleme algoritmalarında başlangıçta kullanıcı tarafından K (küme sayısı) parametresinin belirlenmesini istemesi gibi parametrelere ihtiyaç duyulmamaktadır. Ayrıca deterministik yöntemler değildir. Hiyerarşik yöntemler daha deterministik sonuçlar üretmekte ve veri nesnelerini kümelemek için esnek bir mekanizma sunmaktadır.

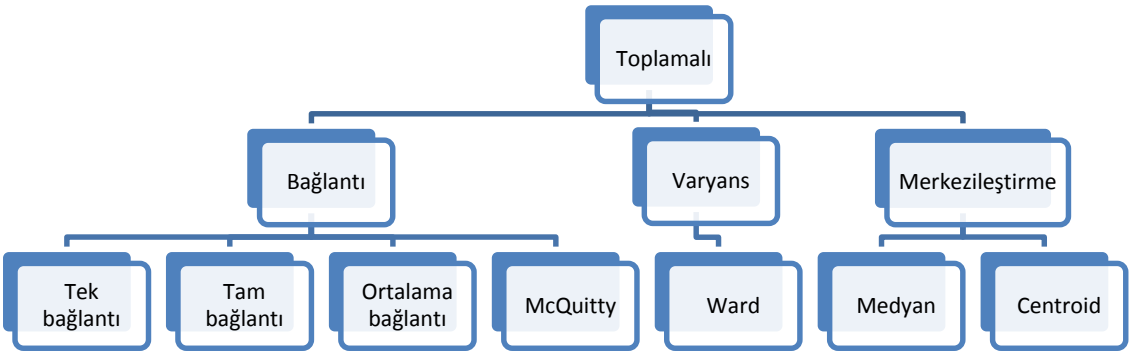
Küme hiyerarşisi standart bir ağaç yapısı kullanılarak ifade edilebilmektedir. Hiyerarşinin kökü kümelenecek olan veri nesnelerin tümünü temsil etmektedir. Ağacın her bir seviyesinde kümelere karşılık gelen düğümler oluşmaktadır. Hiyerarşinin her bir seviyesi bazı küme setlerine karşılık gelmektedir. Hiyerarşinin tabanı ağacın yapraklarından yani tekli noktalardan oluşmaktadır. Bu küme hiyrarşisine dendogram adı verilmektedir. Hiyerarşik kümeleme yöntemlerinin en temel avantajı, herhangi bir seviyede hiyerarşiye son vererek uygun kümeler elde edilebilmektedir. Bu özellik hiyerarşik kümeleme algoritmaları bölünmeli

kümeleme yöntemlerinden ayıran en önemli özelliktir. Böylece hiyerarşik kümeleme yöntemleri kullanıcı tarafından belirlenen *K* küme sayısına ihtiyaç duymamaktır.

Hiyerarşik kümeleme algoritmaları, birleştirici (Agglomerative) ve ayrıştırıcı (Divisive) olmak üzere ikiye ayrılmaktadır. Bu algortimaların yanı sıra; ROCK, BIRCH, CURE ve CHAMELEON algortimalarıda vardır. Bundan sonraki bölümlerde bu yöntemlerin açıklamaları yer alacaktır[9].

2.1.1.1. Birleştirici (Toplamalı) Kümeleme Algoritmaları (AGNES, Agglomerative nesting)

AGNES (Agglomerative Nesting) ilk olarak 1990 yılında Kaufman ve Rousseuw tarafından sunulmuştur. Algoritmada başlangıçta her bir nesne ayrı bir küme olarak kabul edilmektedir. Daha sonraki aşamalarda ise istenilen sayıda küme elde edilinceye kadar benzer özellikteki kümeler birleştirilir. Her birleştirme işleminden sonra küme sayısı bir azalmaktadır. İstenilen sayıda küme elde edilinceye kadar işlemlere devam edilir. Birleştirici kümeleme yöntemleri üç alt grup altında toplanır. Bunlardan birincisi bağlantı yöntemleridir. Bu yöntemler tek bağlantı, tam bağlantı, ağırlıklı, ağırlıksız ortalama bağlantı ve McQuitty yöntemleridir. Yöntemlerde grafik gösterimi kullanılabilmektedir. İkinci grup minimum varyans yöntemleri olarak adlandırılır ve üçüncü grup ise merkezileştirme yöntemleridir. Özet olarak toplamalı kümeleme algoritmalarının sınıflandırılması Şekil 2.1.1.1.'de gösterilmiştir.



Şekil 2.1.1.1.1. Toplamalı kümeleme algoritmaları

Benzerlik açısından sözü edilen tüm hiyerarşik yöntemleri de içine alan uygun bir formülasyon Lance-Williams tarafından geliştirilmiş ve Lance-Williams benzerlik/benzemezlik formülü olarak bilinmektedir[10,11].

Formüle göre eğer nesneler (objects) *i* ve *j*, *i U j* içerisinde birleştirilecek küme ve diğer tüm noktalar arasındaki yeni bir benzerlik / uzaklık belirtilmektedir ve aşağıdaki gibi formülüze edilmektedir[10].

$$d(i\ U\ j,k)=\alpha_i\ d(i.k)+\alpha_j d(i.k)+\beta\ d(i,j)+\gamma|d(i,k)-d(j,k)|$$

α_i,α_j , β ve γ'nin aldığı değerler aşağıdaki tabloda gösterilmiştir. Örneğin tek bağlantı yönteminde $\alpha_i=\alpha_j=\frac{1}{2}$, β=0 ve $\gamma=-\frac{1}{2}$ değerlerini kullanır ve formül aşağıdaki gibi değişmektedir[10,11].

$$d(i\ U\ j,k)=\frac{1}{2}\ d(i.k)+\frac{1}{2}\ d(i.k)-\frac{1}{2}|d(i,k)-d(j,k)|$$

Tablo 2.1.1.1.1'in ikinci sütununda yer alan formüller ise tek bağlantı yönteminin yorumuna benzer olarak uygulanmaktadır. Küme merkezlerini kullanan yöntemler söz konusu olduğunda merkez koordinatları (sütun 3'de yer alan) ve küme merkezleri arasında tanımlanan uzaklık/benzerlik (sütun 4) değerleri kullanılır. Öklid uzaklığı iki yaklaşım arasındaki eşdeğerlik için kullanılmaktadır[10,11].

a ve *b* toplanabilir iki nokta ve *c* ise başka bir nokta olsun. Öklid uzaklığı kullanılarak Lance-Williams benzemezlik formülü aşağıdaki gibi elde edilmektedir[10,11].

$$d^2(a\cup b,c)=\frac{d^2(a,c)}{2}+\frac{d^2(b,c)}{2}-\frac{d^2(a,b)}{4}=\frac{||a-c||^2}{2}+\frac{||b-c||^2}{2}-\frac{||a-b||^2}{4}$$

Yeni küme merkezi $\frac{a+b}{2}$'dir. Dolayısıyla c noktasına olan uzaklığı $||c-\frac{a+b}{2}||^2$.

Tablo 2.1.1.1.1. Hiyerarşik kümeleme yöntemleri ve formülleri[10,11]

Hiyerarşik kümeleme yöntemleri	Lance ve Williams'ın benzememezlik formülü	*i* ve *j* kümelerini birleştiren kümenin merkez koordinatları	g_i ve g_j küme merkezleri arasındaki benzememezlik

Tek bağlantı	α_i= 0,5 β = 0 γ = −0,5		
Tam bağlantı	α_i= 0,5 β = 0 γ = 0,5		
Ortalama bağlantı	$\alpha_i=\frac{\lvert i\rvert}{\lvert i\rvert+\lvert j\rvert}$ β = 0 γ = 0		
McQuitty yöntemi	α_i = 0,5 β = 0 γ =0		
Medyan	α_i= 0,5 β = -0,25 γ = 0	g=$\frac{g_i+g_j}{2}$	$\lvert\lvert g_i - g_j\rvert\rvert^2$
Merkez(centroid)	$\alpha_i=\frac{\lvert i\rvert}{\lvert i\rvert+\lvert j\rvert}$ $\beta = -\frac{\lvert i\rvert\lvert j\rvert}{(\lvert i\rvert+\lvert j\rvert)^2}$ γ = 0	g=$\frac{\lvert i\rvert g_i+\lvert j\rvert g_j}{\lvert i\rvert+\lvert j\rvert}$	$\lvert\lvert g_i - g_j\rvert\rvert^2$
Ward Yöntemi	$\alpha_i=\frac{\lvert i\rvert+\lvert k\rvert}{\lvert i\rvert+\lvert j\rvert+\lvert k\rvert}$ $\beta = -\frac{\lvert k\rvert}{\lvert i\rvert+\lvert j\rvert+\lvert k\rvert}$ γ = 0	g=$\frac{\lvert i\rvert g_i+\lvert j\rvert g_i}{\lvert i\rvert+\lvert j\rvert}$	$\frac{\lvert i\rvert}{\lvert i\rvert+\lvert j\rvert}\lvert\lvert g_i - g_j\rvert\rvert^2$

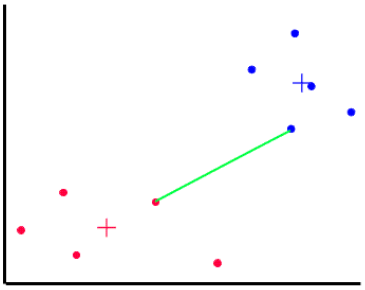
Tablo 2.1.1.1.1'de yer alan |i|, *i* kümesindeki nesnelerin sayısıdır. g_i, m uzayında bir vektör (*m* özellikler setidir). ||.|| değeri ise Öklid metriğindeki uzaklık değeridir [10,11].

Toplamalı kümeleme algoritmalarının sunumu kolay ve hızlı sonuç üretmesi (maliyeti düşük) avantajlarıdır ancak yanlış küme seçimi hatalara yol açmaktadır. Bunun için de kümeleri birleştirme işlemi dikkatlice yapılmalıdır. AGNES' de *n* nesne için *n-1* tane birleştirme yapılmaktadır. Yöntemde birleştirme işlemi yapıldıktan sonra geri dönüp tekrar değiştirme yapılamamaktadır. Çok büyük sayıda veri içeren ve uç değerlerin çok olduğu veri setlerinde iyi sonuçlar elde edilemez. Ayrıca küresel olmayan kümeler için de iyi sonuçlar üretmemektedir. Ancak bu yöntemlerin sunumunun kolay ve hızlı olması nedeniyle literatürde sıkça yararlanıldığı görülmektedir[12,13].

2.1.1.1.1. Tek Bağlantı Kümeleme Algoritması (en yakın komşu algoritması, single linkage)

Tek bağlantı yöntemi literatürde en yakın komşu yöntemi olarak da bilinmektedir. Bu yöntemde başlangıçta tüm gözlem değerleri birer küme kabul edilir ve gözlemler arasındaki

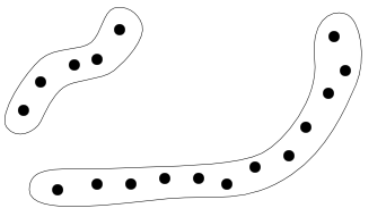
uzaklıklar hesaplanarak birbirine en yakın gözlem değerleri birleştirilir. Bu işlemler adım adım tekrarlanarak devam etmektedir. Tek bağlantı yönteminde ilk olarak en yakın olan iki nokta bulunur ve bu iki nokta bir küme olacak şekilde aynı kümeye dâhil edilir. Daha sonra yeni oluşan küme dışında kalan noktaların bu kümedeki gözlemlere olan uzaklıkları incelenir. En küçük uzaklık değerine sahip gözlem üçüncü bir gözlem olarak kümeye eklenir. Bu işlem bütün noktalar için yapılır ve kümeler oluşturulmuş olur. Gözlemler arasındaki uzaklıkların hesaplanmasında ise veri tipine uygun olan uzaklık ölçüsü kullanılır[14].



Şekil 2.1.1.1.1.1. Tek bağlantı kümeleme yönteminin şekilsel gösterimi

Şekil 2.1.1.1.1.1'den görüldüğü gibi iki küme arasındaki uzaklığı, birbirine en yakın gözlemler arasındaki uzaklık belirler.

Tek bağlantı yönteminde, küresel yapıda ve farklı boyutlarda kümeler tespit edilebilir ancak gürültülü verilere ve uç değerlere karşı duyarlı bir algoritmadır. Bunun yanı sıra yöntemin dağınık yapıda kümeler oluşturma eğilimi vardır ve aşağıdaki yapılara benzer kümeleri bulmada başarılıdır.



Şekil 2.1.1.1.1.2. Küme yapısı

En yakın komşu yöntemine göre uzaklık değerlerinin nasıl hesaplandığı aşağıdaki örnek üzerinde gösterilmiştir.

Astım hastası dört bireyin hastanede yatış sayısı (x_1) ve bir haftada kullandıkları ilaç sayısına (x_2) ait bilgiler aşağıda verilmektedir.

Tablo 2.1.1.1.1.1. Gözlem değerleri

Gözlemler	x_1	x_2
1	10	8
2	5	3
3	4	1
4	6	4

Hastaların bu iki özelliği dikkate alınarak en yakın komşu yöntemi ile kümeleme işlemi yapılırsa aşağıdaki adımların izlenmesi gerekir.

İlk olarak tüm gözlemler arasındaki mesela öklid uzaklığı bulunur. Bu uzaklık değerleri; d(i,j)=$\sqrt{\sum_{k=1}^{p}(x_{ij}-x_{jk})^2}$ formülü yardımıyla,

d(1,2)=$\sqrt{(10-5)^2+(8-3)^2}$=7,07

d(1,3)= $\sqrt{(10-4)^2+(8-1)^2}$=9,22

d(1,4)= $\sqrt{(10-6)^2+(8-4)^2}$=5,65

d(2,3)= $\sqrt{(5-4)^2+(3-1)^2}$=2,24

d(2,4)= $\sqrt{(5-6)^2+(3-4)^2}$=1,41

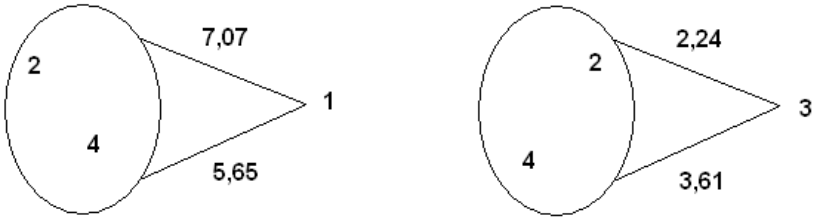
d(3,4)= $\sqrt{(4-6)^2+(1-4)^2}$=3,61

Hesaplanan değerleri aşağıda verilen uzaklık matrisine aktarılır.

Tablo 2.1.1.1.1.2. Uzaklık matrisi

Gözlemler	**1**	**2**	**3**
1			
2	7,07		
3	9,22	2,24	
4	5,65	1,41	3,61

Uzaklıklar matrisi incelendiğinde gözlemler arası en küçük uzaklığı 2 ile 4 numaralı gözlemler arasında olduğu görülmektedir (d=1,41). Dolayısıyla en yakın komşu yöntemi gözlemler arasındaki en küçük uzaklığı küme uzaklığı olarak kabul ettiği için birleştirme işlemine 2 ile 4 numaralı gözlemlerden başlanır ve bu gözlemlerin diğer gözlemler ile uzaklıkları yeniden şematize edildiğinde; uzaklıklar matrisi Tablo 2.1.1.1.1.3'deki gibi tanımlanır.

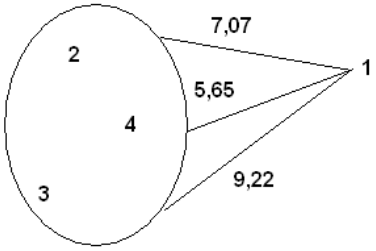


Şekil 2.1.1.1.1.3. Gözlemler ve kümeler arasındaki uzaklıklar

Tablo 2.1.1.1.1.3. Uzaklık matrisi

Gözlemler	**(2, 4)**	**1**
(2, 4)		
1	5,65	
3	2,64	9,22

Yeni oluşturulan uzaklıklar matrisine göre minimum d(i, j)=2,64'dür. Dolayısıyla yeni oluşturulacak küme 2, 4 ve 3 numaralı gözlemlerden oluşur. Bundan sonraki aşamada yeni oluşturulacak küme ile 1 numaralı gözlem arasındaki minimum uzaklık değerinin bulunabilmesi için uzaklık matrislerinin tekrar oluşturulması gerekmektedir.



Şekil 2.1.1.1.1.4. Gözlemler ve kümeler arasındaki uzaklıklar

Şekil 2.1.1.1.1.4'den de görüldüğü gibi (2, 4, 3) kümesinin 1 numaralı gözlem ile arasındaki uzaklıklar incelendiğinde en küçük uzaklığın 5,65 olduğu görülür. Bundan sonraki aşamada

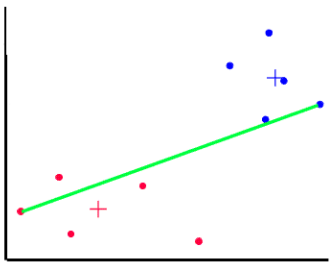
tüm noktalar birleştirilerek tek bir küme oluşturulur ve son olarak tüm aşamaların özetini yansıtacak bir tablo aşağıdaki gibi elde edilir.

Tablo 2.1.1.1.1.4. Oluşturulan kümeler ve uzaklıklar

Uzaklık	**Kümeler**
1,41	(2,4)
2,64	(1,2,4)
5,65	(1,2,3,4)

2.1.1.1.2. Tam Bağlantı Kümeleme Algoritması (en uzak komşu algoritması, complete linkage)

Tam bağlantı yöntemi literatürde en uzak komşu algoritması olarak da bilinmektedir. Bu isimle anılmasının sebebi ise küme uzaklıklarını birbirine en uzak gözlemlerin uzaklığı olarak belirlemesidir. Yöntemin uygulanışı ve yorumlanışı tek bağlantı yöntemine benzerdir. Ancak tek farklılık tek bağlantı yönteminde gözlemlerin en yakın uzaklıkları ile ilgilenilirken, tam bağlantı yönteminde gözlemlerin en uzak mesafeleri ile ilgilenilmektedir. Bu ifadeler Şekil 2.1.1.1.2.1'deki gibi şematize edilebilir[14].



Şekil 2.1.1.1.2.1. Tam bağlantı yönteminin grafiksel gösterimi

Tam bağlantı yönteminin uzaklık formülü aşağıdaki gibidir:

$$dmax(C_i, C_j)=max_{p\in C_i, p'\in C_j}|p-p'|$$

Yukarıda formülde yer alan C_i i. kümeyi $C_j$ise j. kümeyi göstermektedir. $|p-p'|$ değeri ise p ve p' noktaları arasındaki uzaklıktır.

Tam bağlantı yöntemi tek bağlantı yöntemine göre gürültülü ve uç değerlere karşı daha az duyarlıdır. Ancak konveks şekildeki kümeleri bulmak için elverişli bir yöntem değildir. Konveks şekillerin olduğu durumda tek bağlantı yönteminden yararlanılmanın daha iyi sonuçlar vereceği bilinmektedir[14].

En uzak komşu yöntemine göre uzaklık değerlerinin nasıl hesaplandığı aşağıdaki örnek üzerinde gösterilmiştir[14].

Tablo 2.1.1.1.2.1.Gözlem değerleri

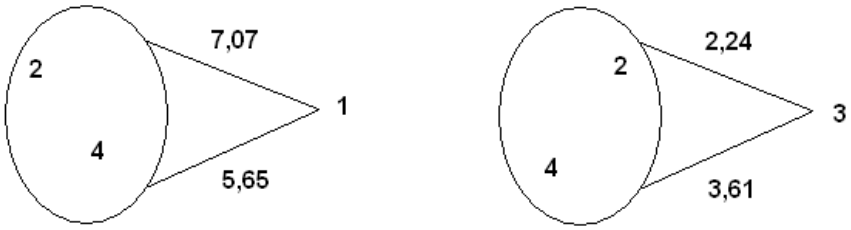
Gözlemler	x_1	x_2
1	10	8
2	5	3
3	4	1
4	6	4

En uzak komşu yöntemde ilk olarak gözlemler arasındaki uzaklıklar, uzaklık bağlantıları kullanılarak bulunmalıdır. Bunun için farklı uzaklık bağlantılarından yararlanılabilir. Ancak biz en çok kullanılan uzaklık bağlantıları arasında yer alan öklid uzaklıklarından faydalanacağız. Buna göre elde edilecek uzaklık matrisi aşağıdaki gibidir.

Tablo 2.1.1.1.2.2. Gözlemler arasın uzaklık matrisi

Gözlemler	**1**	**2**	**3**
1			
2	7,07		
3	9,22	2,24	
4	5,65	**1,41**	3,61

Yukarıdaki tablodan görüldüğü gibi min d(i,j) değerinin 1,41'tür. En yakın komşu algoritmasına benzer olarak en küçük uzaklık değeri belirlenir. Dolayısıyla bundan sonraki aşamada 2 ve 4 numaralı gözlemler birleştirilerek yeni bir küme oluşturulur. Eğer aynı minimum değere sahip iki hücrenin olduğu gözlemlenirse, herhangi bir hücre seçilerek işlemlere devam edilmesi gerekmektedir. (2, 4) kümesinin 3 ve 4 numaralı gözlemler ile uzaklıklarını incelenmesi sonucunda Şekil 2.1.1.1.2.2 elde edilmiştir.



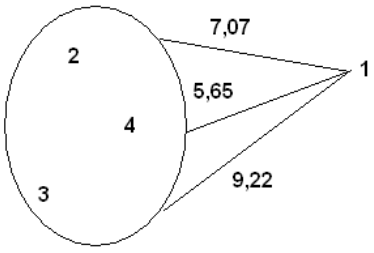
Şekil 2.1.1.1.2.2.Küme ile gözlemler arasındaki uzaklıklar

En yakın komşu yönteminde kümelerin elemanları ile diğer gözlemler arasındaki en yakın uzaklık dikkate alınırken, tam bağlantı yönteminde durum tersidir. Yani söz konusu kümenin elemanları ile diğer gözlemler eşlenerek içlerinden birbirine en uzak olan gözlemler seçilir. Bu duruma göre (2, 4) kümesinin 1 nolu gözlem ile arasındaki en büyük uzaklık 7,07 olarak tespit edilmiştir. Benzer şekilde (2, 4) kümesinin 3 nolu gözlem ile arasındaki en büyük uzaklık değeri 3,61'dir. Dolayısıyla yeni oluşturulacak uzaklık tablosu aşağıdaki gibidir.

Tablo 2.1.1.1.2.3. Uzaklık matrisi

Gözlemler	**(2, 4)**	**1**
(2, 4)		
1	7,07	
3	3,61	9,22

Yukarıdaki uzaklıklar tablosuna göre mininimum d(i,j)=3,61 olduğu görülmektedir. Dolayısıyla yeni oluşturulacak küme 2, 4 ve 3 numaralı gözlemlerden oluşmaktadır. Bundan sonraki aşamada yeni oluşturulacak küme ile 1 numaralı gözlem arasındaki maksimum uzaklık değerinin bulunabilmesi için uzaklık matrislerinin tekrar oluşturulması gerekmektedir.



Şekil 2.1.1.1.2.3. Küme ile 1 nolu gözlem arasındaki uzaklıklar

Şekil 2.1.1.1.2.3'den görüldüğü gibi (2, 4, 3) kümesinin 1 numaralı gözlem ile arasındaki uzaklıklar incelendiğinde en büyük uzaklığın 9,22 olduğu belirlenmiştir. Bundan sonraki

aşamada tüm noktalar birleştirilerek tek bir küme oluşturulur ve son olarak tüm aşamaların özetini yansıtacak bir tablo oluşturulur.

Tablo 2.1.1.1.2.4. Oluşturulan kümeler ve uzaklıklar

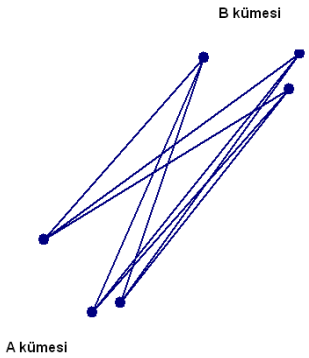
Uzaklık	Kümeler
1,41	(2, 4)
7,08	(1, 2, 4)
9,22	(1, 2, 3, 4)

2.1.1.1.3. Ortalama Bağlantı Kümeleme Algoritması (unweighted pair group method with averaging: UPGMA)

Ortalama bağlantı tekniği Sokal, Michener ve McQuitty tarafından geliştirilmiştir. Ortalama bağlantı yönteminde iki küme arasındaki uzaklık ayrı gruplarda yer alan gözlem çiftleri arasındaki ortalama uzaklık olarak belirlenmektedir. Bu teknikte de işleme tek bağlantı ve tam bağlantı tekniklerinde olduğu gibi başlanır. Ancak kümeleme kriteri olarak bir küme içindeki birim ile diğer küme içindeki birimler arasındaki ortalama uzaklıklar kullanılır. Ortalama bağlantı tekniğinde kümeler küçük varyanslar ile birbirlerine bağlıdır. Bu teknik tek bağlantı ve tam bağlantı teknikleri arasında sonuçlar vermesi nedeniyle bir alternatif yöntem olarak önerilmektedir[15].

Mekânsal olmayan toplamalı kümeleme için bu yöntem düşük varyanslı kümeleri birleştirme eğilimindedir ve aynı varyansa sahip kümeler için biraz yanlı sonuçlar vermektedir. Bu yöntemde iki küme birleştirildiğinde, yeni oluşturulan kümenin diğer kümeye olan uzaklığı daha sonradan birleştirilecek olan kümelerin ortalama uzaklığı alınarak hesaplanır. Örneğin, 1 ve 3 kümeleri birleştirilerek yeni bir küme oluşturulacak ise, bu yeni kümeye küme 1* olarak belirlensin. Küme 1*'in küme 4'e uzaklığı hesaplanmak istendiğinde, 1'in 4'e olan uzaklığı ile 3'ün 4'e olan uzaklığının ortalaması alınmaktadır. Bu uzaklıklar kümelerdeki bireysel gözlem değerleri yerine küme kombinasyonlarına bağlıdır[16,17].

Ortalama bağlantı yöntemi Şekil 2.1.1.1.3.1'deki gibi şematize edilebilir.



Şekil 2.1.1.1.3.1. Ortalama bağlantı yönteminin şekilsel gösterimi

Yöntem de tek ve tam bağlantı algoritmalarında olduğu gibi ilk olarak uzaklık ölçülerinden yararlanılarak uzaklık matrisleri oluşturulur. Ardından küme uzaklıklarının belirlenebilmesi ve hangi gözlemlerin birleştirilmesine karar verebilmek için aşağıdaki eşitlikten yararlanılmaktadır.

$$D(A,B)=T_{AB}/(N_A * N_B)$$

Yukarıdaki formülde yer alan *A* ve *B* harfleri kümeleri göstermektedir. T_{AB} değeri ise *A* ve *B* kümeleri arasındaki tüm ikili uzaklıkların toplamı iken N_A ve N_B sırasıyla *A* ve *B* kümelerindeki gözlem sayılarını göstermektedir. Hiyerarşik kümelemenin her bir aşamasında *D(A,B)* değeri minimum olan kümeler birleştirilir.

Tüm elemanlar tek bir kümeyi oluşturana kadar bu işlemlere devam edilir. Ardından küme uzaklıklarına göre özet bir tablo oluşturularak sonuçlar yorumlanır[18].

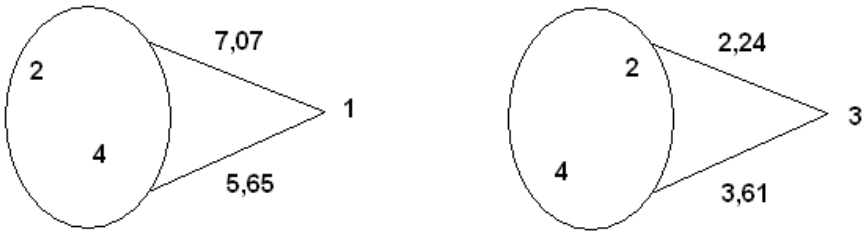
Tek bağlantı yönteminde kullanılan örnek veri seti üzerinden ortalama bağlantı yöntemi uygulanırsa aşağıdaki adımların sırasıyla gerçekleştirilmesi gerekmektedir.

Daha önceden elde edilen uzaklık matrisi Tablo 2.1.1.1.3.1'de verilmiştir.

Tablo 2.1.1.1.3.1. Gözlemler arası uzaklık matrisi

Gözlemler	**1**	**2**	**3**
1			
2	7,07		
3	9,22	2,24	
4	5,65	**1,41**	3,61

Minimum d(i,j) değerinin 1,41 olduğundan 2 ve 4 numaralı gözlemler birleştirilerek yeni bir küme oluşturularak uzaklık matrisleri bu kümelere göre yeniden hesaplanır.



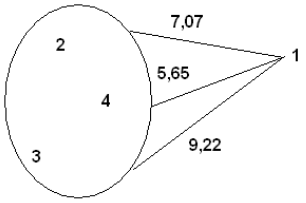
Şekil 2.1.1.1.3.2. Kümeler ile gözlemler arasındaki uzaklıklar

Ancak artık uzaklıklar matrisi oluşturulurken ortalamadan yararlanılacaktır.

Yani 2 ve 4 numaralı gözlemlerden oluşan küme ile 1 numaralı gözlem arasındaki uzaklık (7,07+5,65)/2=6,36 iken 3 numaralı gözlem ile uzaklık (2,24+3,61)/2=2,93'dir. Bu hesaplamalar sonucunda elde edilen uzaklık matrisi Tablo 2.1.1.1.3.2'deki gibidir.

Tablo 2.1.1.1.3.2. Uzaklık matrisi

Gözlemler	**(2, 4)**	**1**
(2, 4)		
1	6,36	
3	2,93	9,22



Şekil 2.1.1.1.3.3. Küme ile 1 nolu gözlem arasındaki uzaklıklar

Şekil 2.1.1.1.3.3' e göre min d(i,j)=2,93 olduğu görülmektedir. Dolayısıyla yeni oluşturulacak küme 2, 3 ve 4 numaralı gözlemlerden oluşmaktadır. 2, 3 ve 4 numaralı gözlemlerin oluşturduğu küme ile 1 numaralı gözlem arasındaki uzaklık (7,07+5,65+9,22)/3=7,31 hesaplanarak bulunur ve özet olarak aşağıdaki tablo elde edilir.

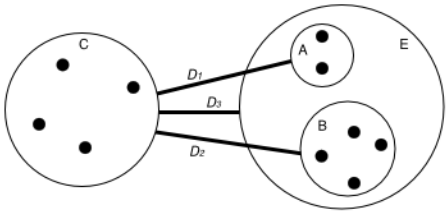
Tablo 2.1.1.1.3.3. Oluşturulan kümeler ve uzaklıklar

Uzaklık	**Kümeler**
1,41	(2, 4)

6,36	(1, 2, 4)
7,31	(1, 2, 3, 4)

2.1.1.1.4. Ağırlıklandırılmış Ortalama Bağlantı Kümeleme Algoritması (weigthed pair group method with arithmetic mean: WPGMA)

Ağırlıklandırılmış ortalama bağlantı yöntemi formüle farklı bir ağırlık eklediği için orijinal ortalama bağlantı yönteminden farklı bir yöntemdir. Bu yöntem McQuitty yöntemi olarak da bilinmektedir. Her bir birleştirme işleminden sonra yeni oluşan küme ile eski küme arasındaki uzaklıklar, birleştirilen iki kümenin uzaklıklarına bağlı olarak hesaplanır. Örneğin, aşağıdaki şekilde *A* ve *B* olmak üzere iki küme birleştirilerek yeni bir küme oluşturulsun. Bu oluşan yeni kümeye *E* kümesi adı verilsin. *E* kümesi ile *C* kümesi arasındaki uzaklık D_3 olarak tanımlansın. O halde WPGMA kümeleme yöntemine göre $D_3=(N_A * D_1 + N_B * D_2)/(N_A + N_B)$ formülü yardımıyla hesaplanır. Formülde yer alan N_A ve N_B sırasıyla *A* ve *B* kümelerinin genişliklerini, D_1; *C* kümesi ile *A* kümesi arasındaki uzaklığı, D_2 ise *C* kümesi ile *B* kümesi arasındaki uzaklığı göstermektedir [19].

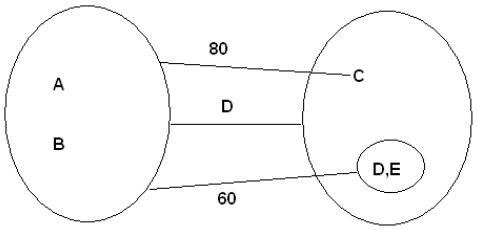


Şekil 2.1.1.1.4.1. Ağırlıklandırılmış ortalama bağlantı yönteminin şekilsel gösterimi

Tablo 2.1.1.1.4.1'deki gibi bir uzaklık matrisi verildiği varsayılsın. *A* ile *B* gözlemleri ile *D* ve *E* gözlemleri önceden birleştirilerek iki tane küme oluşturulmuş ve oluşan bu kümelerin *C* kümesi ile uzaklığı hesaplanmıştır. Bundan sonraki adımlarda WPGMA bağlantı yöntemi kullanılarak kümeleme işleminin nasıl yapıldığı Şekil 2.1.1.1.4.2'de üzerinde gösterilmiştir.

Tablo 2.1.1.1.4.1. Uzaklık matrisi

	AB	*C*	*DE*
AB	0	-	-
C	80	0	-
DE	60	40	0

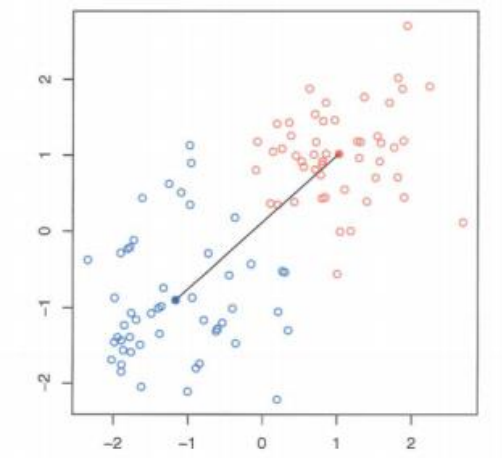


Şekil 2.1.1.1.4.2. Kümeleme işlemlerinin şekilsel gösterimi

Şekil 2.1.1.1.4.2' deki kümelere göre öncelikle *A* ve *B*'den oluşan küme ile *C* numaralı gözlem arasındaki uzaklık 80 idi. Benzer şekilde *A* ve *B* gözlemlerinden oluşan küme ile *D* ve *E*'den oluşan küme arasındaki uzaklık ise 60 idi. Yeni oluşacak kümenin uzaklığı ise $D=\frac{80+2*60}{3}=66{,}6$ olarak hesaplanmaktadır.

2.1.1.1.5. Merkez Bağlantı Kümeleme Algoritması (unweighted pair group centroid: UPGMC)

Bu yöntem 1958 yılında Sokal ve Michener tarafından geliştirilmiştir. Merkez bağlantı yöntemi en çok kullanılan bağlantı fonksiyonlarından biridir. Yöntemde iki küme arasındaki uzaklık merkezler arasındaki ortalama uzaklıktır. Bir başka ifadeyle ortalama olarak grup merkezleri kullanılmaktadır. Ancak bu kümeleme yönteminde sadece Öklid uzaklık ölçüsü kullanılabilmektedir. Kullanımı basit, anlaşılması ve yorumlanması kolay bir yöntemdir. Biyoloji alanında hiyerarşik kümeleme için standart bir yöntem haline gelmektedir. Bu yöntem aşağıdaki gibi şematize edilebilmektedir.



Şekil 2.1.1.1.5.1. Merkez yönteminin grafiksel gösterimi

Şekil 2.1.1.1.5.1'den görüldüğü gibi merkezi bağlantı skorları grup merkezleri arasındaki uzaklık kadardır. Küme merkezleri ise, küme değişkenlerine ilişkin gözlemlerin ortalama değeridir. Her oluşan yeni küme için küme merkezleri yeniden hesaplanarak küme birleştirilme işlemi yeni oluşturulan merkezler dikkate alınarak yapılmaktadır.

Birleştirilen C_i ve C_j kümeleri ile üçüncü bir küme olan C_l arasındaki uzaklık aşağıdaki formül yardımıyla hesaplanmaktadır.

$$D(C_l,(\,C_i,\,C_j)) = \frac{n_i}{n_i+n_j} D(C_l, C_i) + \frac{n_i}{n_i+n_j} D\left(C_l, C_j\right) - \frac{n_i n_j}{(n_i+n_j)^2} D\left(C_i, C_j\right)$$

Yukarıdaki formülde yer alan C_l, C_i ve C_j 'ler sırasıyla l, i ve j kümelerini ; n_i ve n_j ise sırasıyla C_i ve $C_j$kümelerindeki gözlem sayılarını göstermektedir.

Merkezi bağlantı yöntemi uç değerlerden az etkilenmektedir ancak yöntemde ters dönmeden (reversal) kaynaklanan bir karmaşıklık vardır. Bir başka ifadeyle, bir merkez çifti arasındaki uzaklık, daha önceki adımda birleşmiş olan bir başka merkez çifti arasındaki uzaklıktan daha az olabilmektedir. Bu da dendogram yorumlarını zorlaştırmaktadır [20,21].

2.1.1.1.6. Medyan Bağlantı Kümeleme Algoritması (Gower method, weighted pair group centroid method: WPGMC)

Bu yöntem Gower tarafından 1967 yılında geliştirilmiştir. Medyan bağlantı yönteminde, iki küme arasındaki uzaklık bir kümedeki gözlem ile diğer kümedeki gözlem arasındaki medyan uzaklığıdır. Yöntemde ortalama yerine medyan kullanılmaktadır. Dolayısıyla ortalama bağlantı yöntemine göre uç değerlerden fazla etkilenmemektedir. Bu algoritma merkez yöntemine benzerdir ancak medyan yönteminde birleştirilecek kümelere eşit ağırlık verilmektedir. Medyan yönteminde birleştirilen C_i ve C_j kümeleri ile üçüncü bir küme olan C_l arasındaki uzaklık aşağıdaki formül yardımıyla hesaplanmaktadır[19,22].

$$D(C_l,(\,C_i,\,C_j)=\frac{1}{2} D(C_l, C_i)+\frac{1}{2}\, D\left(C_l, C_j\right) - \frac{1}{4} D(C_i, C_j)$$

Yukarıdaki formülde yer alan C_l, C_i ve C_j 'ler sırasıyla l, i ve j kümelerini ; n_i ve n_j ise sırasıyla C_i ve $C_j$kümelerindeki gözlem sayılarını göstermektedir.

2.1.1.1.7. Ward Kümeleme Algoritması (error sum of squares linkage)

Ward yöntemi kümeleme analizine alternatif bir yaklaşım olup her bir küme içerisindeki hata kareler toplamına dayanmaktadır (küme merkezlerinden sapmaların kareler toplamı). Benzerlik ölçülerini kullanmak yerine kümeleme analizini varyans analizi problemi olarak ele alır. Bu yöntem en küçük varyans yöntemi olarak da bilinmektedir. Önceki bölümlerde anlatılan bağlantı yöntemleri kümeler arasındaki uzaklıkları kullanırken bu yöntemde küme içi hata kareler toplamı kullanılmaktadır. Ward yönteminin amacı küme içinde homojenliği maksimum yapacak şekilde küme içi kareler toplamı minimize edilerek kümeler oluşturmaktır. Bu işlem her aşamada yeniden gerçekleştirilir. Yani her aşamada yeni kümeler oluşturmak için, hata kareler toplamı küçük olan kümeler birleştirilmektedir. Tüm gözlemler tek bir küme oluşturuncaya kadar birleştirme işlemlerine devam edilir.

Bu işlemleri formül üzerinde gösterebilmek için X_{ijk} notasyonu tanımlanmaktadır. X_{ijk} simgesi i kümesinde yer alan j gözlemine ait k değişkeninin değerini ifade etmektedir. Hata kareler toplamı ise aşağıdaki gibi hesaplanmaktadır[19,22,23,24].

$$ESS(error\ sum\ of\ squares)=\sum_i \sum_j \sum_k |X_{ijk} - \bar{x}_{i.k}|^2$$

Hata kareler toplamı ile her bir değişken için bireysel gözlem değerleri ile o değişkene ait küme merkez değeri karşılaştırılmış olur. Hata kareler toplamının küçük olması, verinin küme merkezine yakın olduğunu göstermektedir[23,24].

Genel kareler toplamı aşağıdaki formül yardımıyla hesaplanmaktadır[23].

$$TSS\ (total\ sum\ of\ squares)=\sum_i \sum_j \sum_k |X_{ijk} - \bar{x}_{..k}|^2$$

Genel kareler toplamı işleminde amaç her bir değişken için bireysel gözlem değeri, ilgili değişkenin genel ortalaması ile karşılaştırmaktır. Kümenin gözlem değerini açıklama oranı ise r^2 yardımıyla hesaplanmaktadır. r^2 değeri ne kadar yüksek ise küme gözlem değerini o derecede iyi açıklamaktadır. r^2 değeri aşağıdaki eşitlikler yardımıyla hesaplanmaktadır[19,22,23].

$$r^2=\frac{TSS-ESS}{TSS}$$

Ward yönteminde amaç yukarıdaki formüllerde ifade edilen hata kareler toplamını en küçük yapan, r^2 değerini en büyük yapan kümeleri birleştirmektir. Tüm örnek birimleri tek bir küme halinde bir araya geldiğinde ise algoritma sonlandırılacaktır [23].

Ward yöntemi ile birleştirilen iki küme (C_i ve C_j) ile diğer küme (C_k) arasındaki uzaklık aşağıdaki formül yardımıyla hesaplanmaktadır[19,22,23].

$$d(C_i \text{ ve } C_j)=\frac{n_i+n_k}{n_i+n_j+n_k}d(C_i,C_k)+\frac{n_j+n_k}{n_i+n_j+n_k}d(C_j,C_k)-\frac{n_k}{n_i+n_j+n_k}d(C_i,C_j)$$

Yukarıdaki formülde yer alan n_i, n_j, n_k değerleri sırasıyla C_i, C_j ve C_k kümelerindeki nesne sayısını göstermektedir. $d(C_i \cup C_j, C_k)$ eşitliği C_i ve C_j kümelerinden oluşan yeni küme ile C_k kümesi arasındaki uzaklığı, $d(C_i, C_k)$ değeri C_i ve C_k, $d(C_j, C_k)$ değeri C_j ve C_k, $d(C_i, C_j)$ değeri C_i ve C_j kümeleri arasındaki uzaklıkları göstermektedir.

Bu bağlantı yöntemi küçük sayıdaki gözlemleri birleştirme eğilimindedir. Aynı şekle sahip ve aynı sayıda gözlem içeren kümelerin üretilmesine karşı yanlıdır. Ayrıca Ward yöntemi uç değerlere karşı da duyarlı bir bağlantı yöntemidir. Bunun yanı sıra yöntem niceliksel veriler için uygun iken ikili değişkenler için uygun değildir[19,22].

2.1.1.1.8. ROCK Kümeleme Algoritması (Robust clustering using links)

ROCK algoritması, bağlantı kavramı üzerine dayanan dayanıklı (robust) toplamalı hiyerarşik bir kümeleme algoritmasıdır[25]. Çok büyük veri setleri için de uygun bir yöntemdir [26]. Guha, Rastogi ve Shim tarafından ICDE'de sunulan ve önerilen ROCK algoritması benzerliği hesaplarken, veri noktalarını birleştirirmede bağlantıları kullanmaktadır[27,28,29]. Kümeleme algoritmalarında uzaklık yerine bağlantıları kullanmak bir avantajdır. Çünkü uzaklıklar kullanıldığında sadece iki nesne arasındaki yerel bilgiden yararlanılır oysaki bağlantı fonksiyonları kullanıldığında iki nesnenin komşuluğundaki diğer noktalarla da ilgili evrensel bir bilgi edinilmektedir. Komşulara ve bağlantılara dayanan bu kümeleme modelini anlayabilmek için bazı temel tanımların yapılması gerekmektedir.

Komşuluk

Komşuluk kavramı basitçe en benzer noktalar olarak ifade edilebilir [27,28]. Benzerlik (p_i, p_j), p_i ve p_j nokta çiftleri arasındaki yakınlığı yakalayan benzerlik fonksiyonlarından metrik veya metrik olmayan bir ölçü kullanılabilir. Benzerlik fonksiyonu 0 ve 1 arasında değerler alabilmektedir. Fonksiyon değeri 1'e yaklaştıkça noktaların benzerliğinin arttığı bilinmektedir. 0 ile 1 arasında kullanıcı tarafından bir θ eşik değeri tanımlanır ve eğer benzerlik $(p_i, p_j) \geq \theta$ ise bu iki nokta komşudur. Eşitsizlikte tanımlanan θ simgesi kullanıcı tarafından belirlenen bir parametredir ve komşuluğu dikkate almak için nokta çiftlerinin ne kadar yakın olması gerektiğini kontrol etmek için kullanılır. Benzerlik fonksiyonunun 1 olması noktaların özdeş olduğunu, fonksiyonun 0 değerini alması ise noktaların birbirine benzemediğini göstermektedir. Dolayısıyla arzu edilen yakınlığa bağlı olarak kullanıcı tarafından uygun bir θ değeri belirlenmelidir[27,28].

Benzerlik fonksiyonu Jaccard katsayısı temelli bir benzerlik ölçüsünü kullanmaktadır. Yöntem, Öklid uzaklığı yerine jaccard katsayısından yararlanmaktadır. Bu katsayı metrik olmayan bir uzaklık ölçüsüdür. Dolayısıyla merkezler değil sadece küme içerisinde kalan noktaları tanımlamaktadır. Bu fonksiyon ise aşağıdaki gibi tanımlanmaktadır[27,28].

$$Benzerlik\ (T_1, T_2) = \frac{|T_1 \cap T_2|}{|T_1 \cup T_2|}$$

Formülde yer alan T_1 ve T_2 iki ayrı küme olmak üzere $|T_1 \cap T_2|$kümelerdeki kesişen birim sayısını, $|T_1 \cup T_2|$ ise kümelerdeki farklı olan tüm birim sayısını göstermektedir.

T_1 ve T_2 işlemlerinin sahip olduğu ortak birimler ne kadar fazla ise $|T_1 \cap T_2|$ o kadar büyük ve benzerlik daha fazladır (1,2). $|T_1 \cup T_2|$'e bölmek ise θ parametresinin 0 ile 1 arasında değer almasını sağlayan ve farklı büyüklükteki kümelerin benzerliğini karşılaştırmada kullanılmasını sağlayan bir standardizasyon işlemidir.
Öneğin; küme 1, 2, 3, 4 ve 5 öğeleri ile temsil edilsin. (1, 2, 3) ve (3, 4, 5) çiftleri arasında Jaccard Katsayısı aşağıdaki gibi hesaplanmaktadır.
Kümelerin kesişim eleman sayısı=1 ve kümelerin birleşim eleman sayısı=5
Jaccard katsayısı=1/5=0,2' dir.

Bağlantılar

Bağlantı (p_i, p_j), $p_i ile\ p_j$ arasındaki ortak komşulukların sayısı olarak tanımlansın. $p_i ve\ p_j$'nin aynı kümede olabilmesi için bağlantı (p_i, p_j) değerinin büyük olması

gerekmektedir. Kümeleme problemi için bağlantı temelli yaklaşım, evrensel bir yaklaşımdır. Bu yöntem her bir nokta çifti arasındaki ilişkide komşu veri noktalarındaki evrensel bilgiyi yakalar. Bundan dolayı ROCK kümeleme algoritması noktaları tek bir kümede birleştirme kararı verirken noktalar arasındaki bağlantıyla ilgili bilgiden yararlandığı için daha robust bir yöntem olarak karşımıza çıkar [27,28].

Kriter Fonksiyonu

Yüksek derecede bağlanabilirliğe sahip olabilmesi için tek bir kümeye ait olan p_q, p_r veri nokta çiftleri için her bir kümenin bağlantı (p_q, p_r)'lerin toplamlarının maksimum olması, farklı kümelerdeki p_q, p_s noktaları için ise bağlantı (p_q, p_s)'nin toplamının minimum olması istenmektedir. Bundan dolayı bir kriter fonksiyonuna gerek duyulmaktadır. Bu kriter fonksiyonu da k küme için maksimum yapılmaya çalışılmaktadır [27].

$$E_l = \sum_{i=1}^{k} n_i * \sum_{p_q, p_r \in C_i} \frac{bağlantı(p_q, p_r)}{{n_i}^{1+2f(\theta)}}$$

Yukarıdaki formülde yer alan C_i, n_i genişliğinde i. kümeyi göstermektedir. $f(\theta)$ değeri ise $\frac{1-\theta}{1+\theta}$ ' ya eşittir.

Tüm p_q, p_r veri çiftleri için bağlantı (p_q, p_r) fonksiyonunu maksimum yapmak amaçlandığı için, $\sum_{i=1}^{k} \sum_{p_q, p_r \in C_i}$ bağlantı (p_q, p_r) gibi basit bir kriter fonksiyonu kullanılarak aynı kümedeki nokta çiftleri arasındaki linkleri toplamaktadır. Ancak bu kriter fonksiyonu aynı kümede belirlenen noktalar arasında büyük bir link değeri sağlasa da, tüm noktaların tek bir kümeye atandığı anlamına gelmemektedir. Bu sorunu gidermek için de kriter fonksiyonu olan E_l 'de C_i kümesindeki nokta çiftlerini içeren toplam link sayısı, C_i 'deki beklenen toplam link sayısına bölünür ve bu miktar n_i ve C_i'deki nokta sayısı ile ağırlıklandırılır. C_i'nin yaklaşık ${n_i}^{f(\theta)}$ tane komşusu olduğu varsayımı altında, nokta çiftleri arasındaki beklenen link sayısı ${n_i}^{1+2f(\theta)}$ olarak gösterilebilir [27].

Uyum iyiliği ölçüsü

Kriter fonksiyonu kümelerin uyumunun tahmin edilmesi için kullanılabilmektedir. Noktaları kümelemenin en iyi yolu kriter fonksiyonu için en büyük değerleri birleştirmektir. Uyum iyiliği ölçüsü ROCK algoritmasının herhangi bir aşamasında birleştirilebilecek en iyi küme çiftlerini belirlemeye yardımcı olmaktadır. Kümeleri birleştirmek için tanımlanan uyum iyiliği ölçüsü $g(C_i, C_j)$ eşitliği aşağıdaki gibi tanımlanmaktadır[27].

$$g(C_i, C_j) = \frac{link[C_i,C_j]}{(n_i+n_j)^{1+2f(\theta)}-n_i{}^{1+2f(\theta)}-n_j{}^{1+2f(\theta)}}$$

Küme çiftleri için hesaplanan yukarıdaki uyum iyiliği ölçütü, herhangi bir aşamada en iyi sonucu veren iki küme için maksimum değeri vermektedir. Ancak sadece küme çiftleri arasındaki bağlantı sayısını birleştirmede indikatör olarak kullanmak uygun olmayabilir. Çok iyi ayrılmış kümeler için bir sorun olmayabilir ancak uç değerlerin çok olduğu veya büyük kümelerin diğer kümelerle çok sayıda link sayısı olduğu durumlarda sorun olabilir. Bu sorunu gidermek için de kümeler arasındaki link sayısı beklenen link sayısına bölünmektedir. Beklenen link sayısı ise $(n_i + n_j)^{1+2f(\theta)} - n_i{}^{1+2f(\theta)} - n_j{}^{1+2f(\theta)}$ olarak bilinmektedir[27].

ROCK algoritması kullanılarak verileri kümelemek için aşağıdaki adımların izlenmesi gerekmektedir[27,30].

1. Girdi parametresi olarak *n* örnek noktalarından oluşan *S* seti ve oluşturulmak istenen küme sayısı (*k*) belirlenir.
2. Nokta çiftleri arasındaki bağlantı sayıları hesaplanır. Başlangıçta her bir nokta ayrı bir küme olarak ele alınır. Her bir *i* kümesi için, local bir yığın *q[i]* (heap) oluşturulur ve algoritma sonlanana kadar devam eder. *q[i]*, link[*i*,*j*] değeri sıfır olmayan her *j* kümesini içermektedir. *q[i]*'deki *j* kümeleri, *g(i,j)* uyum iyiliği ölçüsü dikkate alınarak azalan şekilde sıralanmaktadır. Her *i* kümesi için *q[i]* lokal yığınına ek olarak algoritma ayrıca tüm kümeleri içeren eklemeli global yığın *Q*'yu da sağlamaktadır. *Q*'da yer alan kümeler en iyi uyum iyiliği ölçüsüne göre azalan sırada sıralanmaktadır. *Q*'da yer alan çeşitli *j* kümelerin sıralanmasında *g(j,max(q[j]))* kullanılır. Her bir adımda *Q*'daki maksimum *j* kümesi ve *q[j]*'deki maksimum küme birleştirilecek en iyi küme çiftleridir.
3. Bu döngü global *Q* yığınında *k* küme oluşuncaya kadar devam etmektedir. Ayrıca, kalan diğer kümeler arasındaki her bir küme çifti için link sayısı sıfır olduğunda işlem durdurulur.

ROCK algoritması yalnızca nispi bağlılığı göz önünde bulundurduğu için yanlış kümelerin oluşturulmasına sebep olabilmektedir. CURE algoritması ise sadece iki küme noktaları arasındaki minimum uzaklığı kullanmakta, nispi bağlılığı dikkate almamaktadır. Dolayısıyla her iki yönteminde eksiklikleri mevcuttur. Her iki yöntem de nispi bağımlılık ve yakınlık ile ilgili bilgileri birlikte kullanmamaktadır. Bu eksikliklerden dolayı da bu algoritmaların yerine bu eksiklikleri giderebilmek için yeni algoritmalar tercih edilmiştir [30].

ROCK algoritmasının zaman ve yer karmaşıklığını inceleyecek olursak, algoritmanın zaman karmaşıklığı O($n^2 + nm_m m_a + n^2 logn$)'dir. Burada yer alan m_m değeri komşu sayısının maksimumunu göstermektedir. m_a ise, komşu sayılarının ortalamasını ifade ederken *n* veri girişi yapılan noktalarının sayısını göstermektedir. ROCK algoritmasının yer karmaşıklığı ise O(min{ $n^2, nm_m m_a$} olarak bilinmektedir [30].

Tablo 2.1.1.1.8.1. ROCK algoritmasının özet tanımlayıcı özellikleri

Algoritma	Özet
ROCK	Robust toplamalı hiyerarşik bir kümeleme yöntemidir. Benzerlikler hesaplanırken bağlantıları kullanır. Kümelerinin uyumunun tahmin edilmesi için kriter fonksiyonuna gerek duymaktadır. Sadece nispi bağlılığı dikkate aldığı için hatalı sonuçların oluşmasına sebep olabilmektedir. Aykırı veri setleri için kullanılabilir. Kategorik yapıdaki veriler için uygundur. Keyfi şekildeki kümeler için uygundur. Zaman karmaşıklığı O($n^2 + nm_m m_a + n^2 logn$)'dir. Yer karmaşıklığı O(min{ $n^2, nm_m m_a$}'dir.

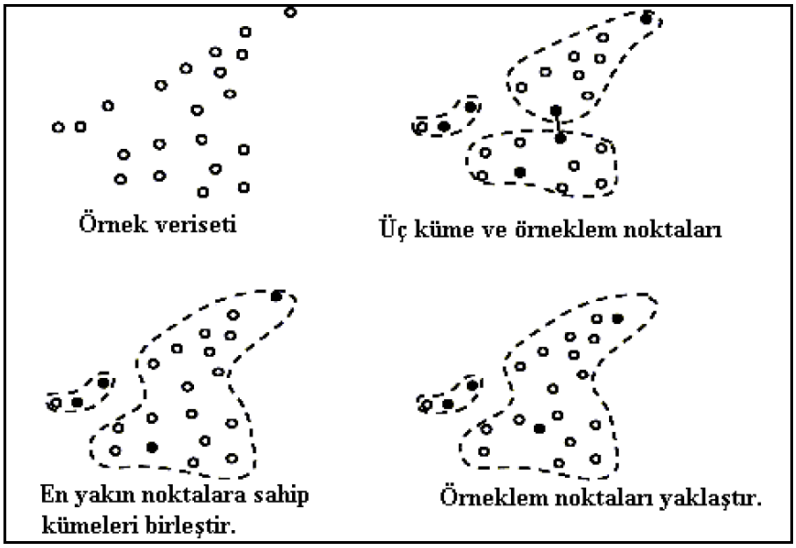
2.1.1.1.9. CURE Kümeleme Algoritması (Clustering Using REpresentatives)

Guha, Rastogi ve Shim tarafından ilk olarak SIGMOD 1998 konferansında sunulan CURE algoritması birleştirici bir kümeleme metodudur. Hiyerarşik metotların küresel olmayan ve farklı boyutlu kümeleri bulma konusundaki zayıflıklarını ve aşırı değerlere karşı hassasiyetlerini gidermek üzere ortaya konmuştur[12].

Her bir küme için birden fazla temsili nokta belirleyen CURE algoritması merkeze dayalı tekniklerden farklı olarak, küresel olmayan şekillere sahip ve büyüklük olarak çok değişiklik gösteren kümeleri tanımlamada da kullanılmaktadır[30,31].

CURE algoritmasının yüksek kaliteli kümeleri bulmak için diğer algoritmalardan farklı olarak uyguladığı iki teknik vardır. Birinci teknik, bir kümeyi temsil etmek için sadece bir adet merkez nokta kullanmak veya kümeyi içinde bulunan tüm noktalarla temsil etmek yerine her

kümeyi o küme içinden seçilmiş sabit sayıdaki dağınık (well-scattered) nesnelerle temsil edilmesidir. İkinci teknik ise, seçilen örneklem nesnelerinin, değeri 0 ile 1 arasında değişen bir daraltma katsayısına bağlı olarak ait oldukları küme merkezine doğru yakınlaştırılmasıdır. CURE algoritmasının genel çalışma şekli Şekil 2.1.1.1.9.1'de görülmektedir [12]. Öncelikle her girdiyi sanki ayrı bir kümeymiş gibi ele alır ve her adımda bu küme temsilcilerinin birbirine olan yakınlıklarına göre ya birleştirir ya da ayrı kümeler olarak tutar. Öncelikle her bir küme için *c* adet iyi dağıtılmış temsilci (örneklem) nokta seçilir Seçilen bu noktalar kümelerin fiziksel şeklini geometrik özelliğini ortaya koyar. Daha sonra bu dağıtılmış noktalar bir α katsayısıyla kümenin ortasına, merkezine doğru kaydırılır. Dağıtılmış olan noktalar bu kaydırma işleminden sonra artık o kümenin temsilcileri olarak kabul edilirler. Birleştirme işlemi istenen küme sayısı, *k*, elde edilene kadar devam eder. Bundan sonra iki küme arasındaki uzaklık, her biri bir kümeye ait olan en yakın temsilci çifti arasındaki uzaklıktır. Dolayısıyla, iki küme arasındaki uzaklık, bu kümelerin örneklem noktaları arasındaki uzaklığa eşittir. Kümeler arası uzaklıkların hesaplanmasında en yaygın kullanılan ölçüt Öklid uzaklığıdır[30].



Şekil 2.1.1.1.9.1. CURE algoritmasının genel çalışma şekli

CURE algoritmasının adımlarını aşağıdaki gibi özetlemek mümkündür:

1. Orjinal veri setinden tesadüfi bir örneklem seçilir.
2. Seçilen örneklem, bölmeler setine ayrılır.
3. Her bir bölme kısmi olarak kümelenir.

4. Aykırı değerler elimine edilir ve çok yavaş büyüyen bir küme varsa o küme çıkarılır.
5. Kısmi olarak oluşturulan kümeler yeniden kümelenir.
6. İlgili küme etiketleri ile oluşan kümeler işaretlenir[30,32].

CURE algoritmasında seçilen parametrelerin elde edilen kümeler üzerinde etkisi çok fazladır. Küme sayısı (*k*) programın sonlanma koşulunu oluşturur. Örneklem nokta sayısı (*c*), algoritmanın şekilsiz ve farklı boyutlardaki kümeleri bulmasını sağlar. *c* parametresinin küçük değerleri CURE algoritmasının merkez nokta (centroid) temelli algoritmalar gibi çalışmasını sağlarken büyük değerleri de CURE algoritmasının AGNES algoritmasına benzer şekilde sonuçlar vermesine neden olur[30].

CURE algoritmasının bir diğer parametresi olan *α,* uç verilerin etkisini azaltmada da kullanılır. Temsilcilerin bir *α* katsayısıyla kümenin merkezine kaydırılması kümedeki uç verilerin etkisini de azaltır [33,34]. *α*'nın alacağı değer 0 ile 1 arasındadır. Küçük değerli *α* dağılmış noktaların çok az yer değiştirmesine neden olurken kümelerin de şekilsel olarak uzunlaşmasına yol açar. α değerinin büyük olması ise dağılmış noktaları küme merkezine oldukça yaklaştıracağı için daha toplu halde kümeler oluşmasına neden olacaktır[33].

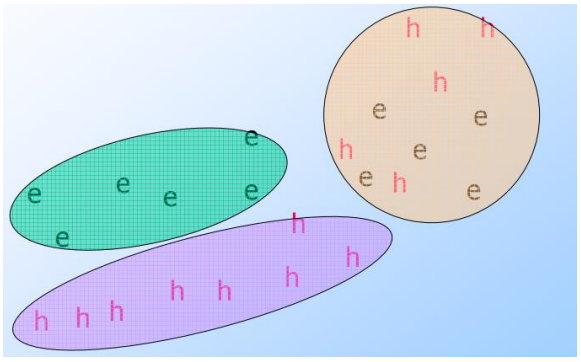
CURE algoritması ile gerçekleştirilen bilimsel çalışmalar, daraltma çarpanı *α*'nın 0.2 ile 0.7 arasındaki bir değerinin daima doğru kümeleri bulduğunu göstermektedir. *α* katsayısı 0 ile 0.1 arasında bir değer aldığında CURE algoritmasının uç noktalara karşı duyarlılığı artar. *α* katsayısının küçük değerleri uç noktaları daha az yaklaştıracağından uzun kümeleri bulur. *α* katsayısı 0.8 ile 1.0 aralığında bir değer aldığında ise CURE algoritması BIRCH veya k-means gibi geleneksel merkezi-tabanlı algoritmalara benzer sonuçlar verir. *α* katsayısının büyük değerleri dağınık noktaları merkeze doğru daha çok yaklaştıracağından bulunan kümeler daha sıkışık olur[12,34].

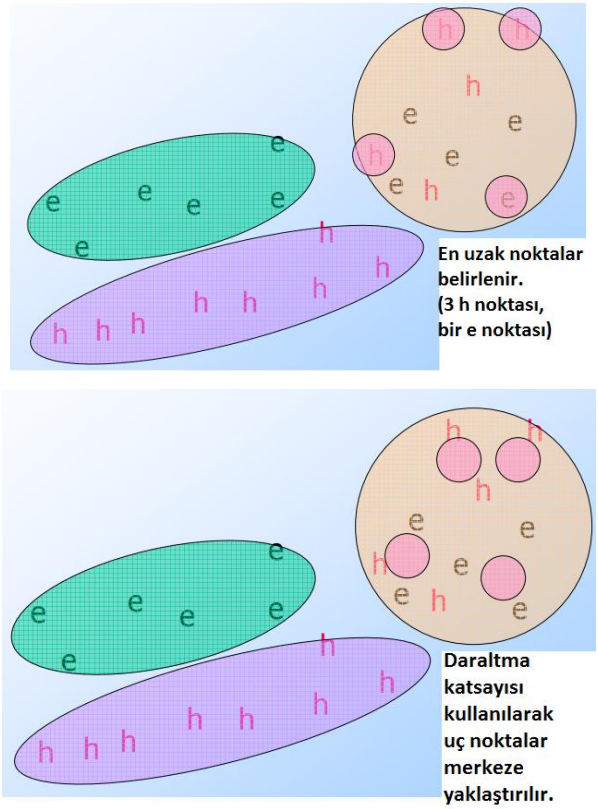
CURE algoritmasının çok büyük veri tabanlarında başarılı olmasını sağlamak için, veri tabanından rasgele bir örneklem seçilir. Kümeleme algoritması bu örneklem veri seti üzerinde uygulanır. Çok büyük veri tabanlarında, kümeler arası ayrıklık ve küme yoğunluğu azaldıkça örneklem boyutları artacaktır. Bu durumda kümeleme işleminin hızını arttırmak için oluşturulan örneklem veri seti parçalara ayrılır ve kümeleme işlemi bu parçalar üzerine uygulanır. Veri tabanındaki diğer nesnelerin oluşturulan kümelerden birine dâhil edilmesinde kümelerin örneklem noktalarına uzaklıkları dikkate alınır. CURE algoritmasının başarılı

sonuçlara ulaşabilmesi, parametrelerinin doğru seçilmesine bağlıdır. Bu nedenle en iyi sonucun bulunması için algoritmanın aynı veri seti üzerinde birkaç kez tekrarlanması gerekebilir[35]. CURE algoritması ile gerçekleştirilen deneysel çalışmalar aşağıdaki sonuçları ortaya koymaktadır.

- CURE farklı şekillerdeki kümeleri bulabilmektedir.
- CURE algoritması sıra dışılıklara karşı çok duyarlı değildir ve bu tip noktaları başarıyla atmaktadır,
- Örnekleme ve bölümleme işlemi küme kalitesinden ödün vermeden veri seti boyutunu azaltmaktadır.
- Kümeler şekilsiz olduğunda dahi, son kümeleme işlemi doğru olarak yapılabilmektedir.
- CURE algoritmasının en kötü durumlarda zaman karmaşıklığı O($n^2 logn$) olmasına karşın veri noktalarının boyutu küçük olduğu durumlarda zaman karmaşıklığı O(n^2)'dir. Bu algoritmanın yer karmaşıklığı ise O(n) olarak bilinmektedir.

CURE algoritmasının işleyişinin daha iyi anlaşılması için aşağıdaki örnek uygulama verilmiştir. Şekil 2.1.1.1.9.2' deki gibi gibi ilk olarak tesadüfi olarak örneklem setleri seçilmektedir. Seçilen örneklem setleri içerisinde yer alan uç değerler belirlenir (3 tane *h* noktası bir tane *e* noktası). Uç noktalar için daraltma katsayısı kullanılarak noktalar merkeze yaklaştırlır.





Şekil 2.1.1.1.9.2. CURE kümeleme algoritmasının örnek uygulamasının şekilsel gösterimi

CURE Algoritmasının avantajlarını aşağıdaki gibi sıralamak mümkündür[12,36,37];

- Küresel olmayan ve farklı boyutlardaki kümeleri bulur.
- Sıradışı verilere karşı duyarlı değildir.
- Rasgele örnekleme ve bölümleyici metotlar ile birlikte kullanıldığında çok büyük veri tabanlarında ölçeklenebilir zaman karmaşıklığı azalır.

CURE Algoritmasının dezavantajları ise[12,36,37];

- Kategorik veriler içeren veri tabanlarında kullanılamaz. Sadece sayısal verilerde etkilidir.
- Seçilen parametrelere çok duyarlıdır.

CURE Algoritması Tablo 2.1.1.1.9.1'de özetlenmiştir[38].

Tablo 2.1.1.1.9.1. CURE algoritmasının özet tanımlayıcı özellikleri

Algoritma	Özet
CURE	• Birleştirici bir kümeleme algoritmasıdır. • Keyfi şekildeki kümeler için uygundur. • Seçilen parametrelerin etkisi fazladır. • Daraltma katsayısı kullanarak noktaları küme merkezlerine yaklaştırır. • Sadece sayısal veriler için kullanılabilmektedir. • Benzerlikler hesaplanırken nispi yakınlıkları kullanmaktadır. • Yer karmaşıklığı O(n)'dir. • Zaman karmaşıklığı en kötü durumda O($n^2 logn$)'dir ancak noktalarının boyutu küçük olduğu durumlarda zaman karmaşıklığı O(n^2)'dir.

2.1.1.1.10. BIRCH Kümeleme Algoritması (Balanced Iterative Reducing and Clustering using Hierarchies)

Toplamalı (agglomerative) hiyerarşik kümeleme metodu olan BIRCH algoritması SIGMOD 96 konferansında Zhang, Ramakrishnan ve Livyn tarafından sunulmuştur [30,39]. Bu algoritma büyük veri tabanlarının kümelenmesi için geliştirilmiş bir algoritmadır, ayrıca gürültülü verilerin kontrol edilmesi için bu alanda öne sürülen ilk algoritmadır. Bu yöntem bilgisayar belleğinde daha az yer kaplayan bir tekniğe sahip hiyerarşik yapıda bir kümeleme algoritmasıdır [33].Yöntemin temelinde tüm gerekli bilgilere sahip bir ağaç oluşturularak kümeleme işlemleri gerçekleştirilmektedir.

Algoritma iki temel kavram üzerinde durmaktadır. Bu kavramlardan birincisi kümeleme özelliği (Clustering Feature, *CF*) ikincisi ise kümeleme özelliği ağacı (Clustering Feature Tree CF-tree)'dır. Bu iki özellik, kümelemelerin özetlenmesine ve büyük veri setlerinin ölçeklenebilirliğinin elde edilmesinde yardımcı olmaktadır [30,40].

Veri nesnelerinin alt kümesi hakkında bilgileri özetlemek için üç parametreden (N, $\vec{LS}$, SS) yararlanır. Bu parametreler;

N: Kümede bulunan nokta sayısı

$\vec{LS}$: Kümedeki noktaların değerlerinin toplamı

$\vec{LS}=\sum_{i=1}^{N} \overrightarrow{X_i}$

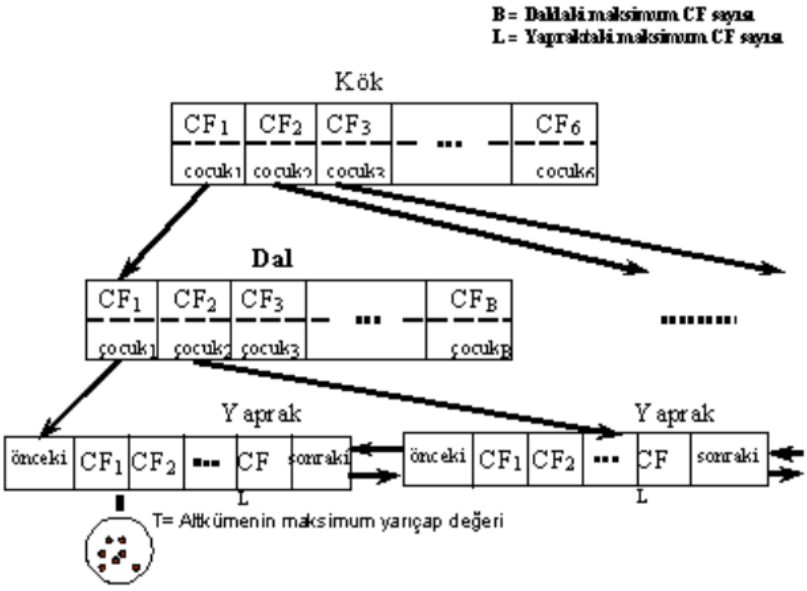
SS: Kümedeki noktaların değerlerinin karelerinin toplamıdır.

$SS=\sum_{i=1}^{N} \overrightarrow{X_i^2}$

Küme özelliği, veri setindeki tüm nesneleri saklamak yerine nesnelerin altkümeleri hakkındaki istatistiksel bilgileri taşır ve sadece küme içindeki dağılımları özetlemez, ayrıca iki alt küme arasındaki uzaklığın ve kümeleme kalitesinin ölçülmesinde kullanılır[12,30,41].

CF ağacı ise, hiyerarşik kümeleme için kümeleme özelliklerini depolayan yüksekliği dengeli bir ağaçtır [39]. Bir *CF-tree*, dallanma faktörü (branching factor) ve eşik değeri (Threshold) olmak üzere iki parametreye sahiptir. Dallanma katsayısı ağaçta bulunacak maksimum yaprak sayısını, eşik değeri ise yapraklarda oluşturulacak küme sayısının çarpımının alacağı en büyük değeri belirler[12,30,41,42].

CF ağaç yapısı Şekil 2.1.1.1.10.1'de verilmiştir.



Şekil 2.1.1.1.10.1. *CF* ağaç yapısı

BIRCH algoritmasının işlem adımları dört tane aşamayı içermektedir.

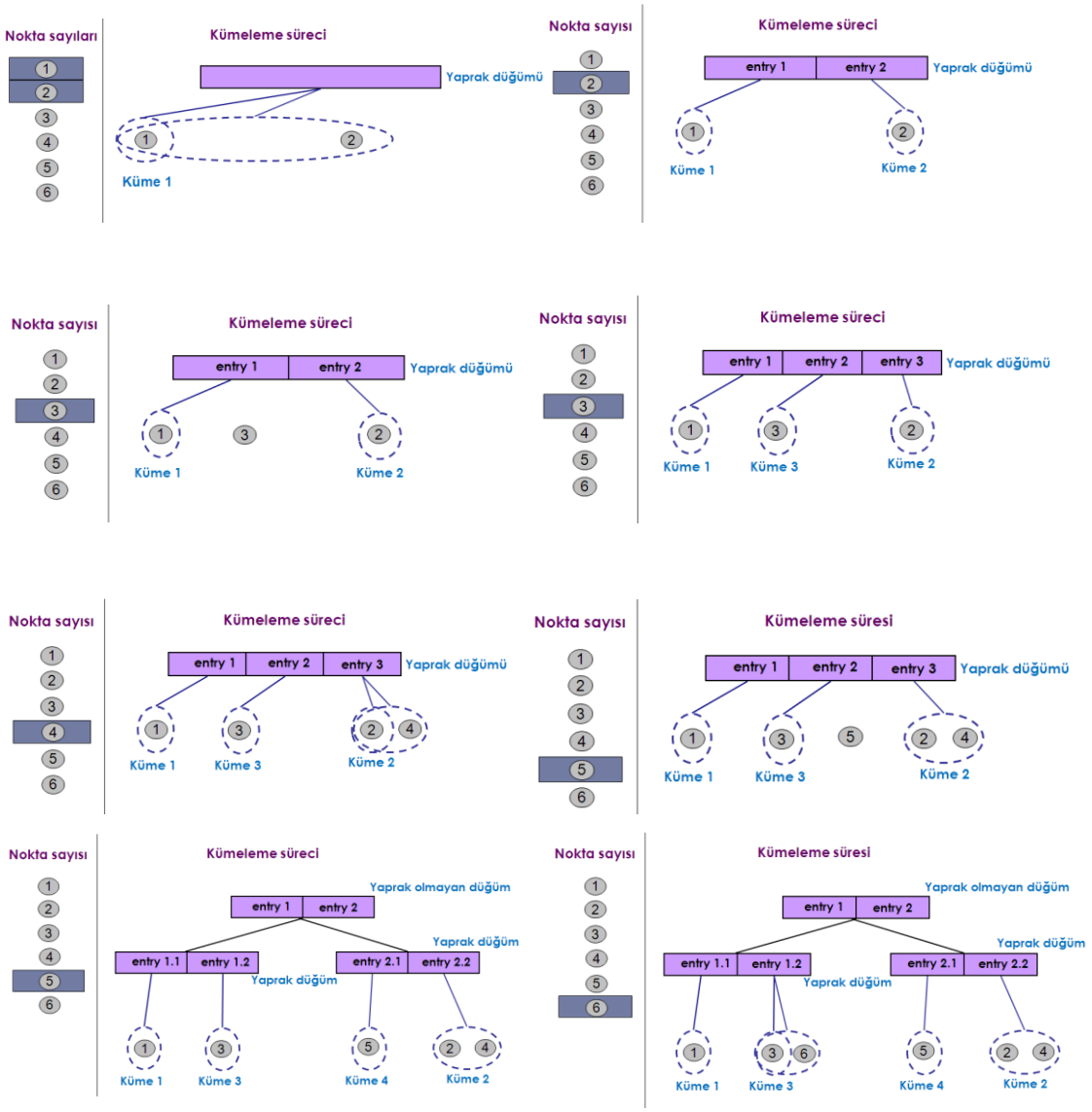
1. *CF* ağacı oluşturularak belleğe yükleme (ilk *CF* ağacı)
2. Daha küçük bir *CF* ağacı oluşturularak istenen aralıkta yoğunlaştırma (Küçültülmüş *CF* ağacı)
3. Global Kümeleme (Başarılı Kümeler)
4. Küme İyileştirme (Daha kaliteli kümeler)

Birinci aşamada veri setindeki tüm veri noktaları taranarak önceden belirlenen *N* sayısı kadar veri noktası içeren altkümeler oluşturulur. Bu alt kümelerin her biri için *CF* değeri hesaplanır. *CF* değerleri veri setine oranla çok daha az yer kapladığı için ana bellekte saklanır. Böylece

sabit diskte yer açılmaktadır. Nesneler eklendikçe *CF* değerleri kullanılarak *CF* ağacı dinamik olarak oluşur. Bu nedenle metot artandır. Her nesne kendine en yakın olan kümeye eklenir. Alt kümenin yarıçapı *CF* ağacının eşik değerini geçerse bölümlendirme işlemi gerçekleştirilir. Yeni nesnenin eklenmesinden sonra, o nesne hakkındaki bilgi ağacın köküne iletilir. *CF* ağacının boyutu eşik değerine bağlı olarak değişmektedir. Oluşturulan *CF*-ağacı bellek boyutu sınırları içinde verilen veri setinin kümeleme bilgilerini mümkün olduğunca detaylı olarak yansıtmaya çalışır. *CF* ağacının bellekte kapladığı alan ana bellek boyutundan daha büyükse daha küçük bir eşik değeri tanımlanarak *CF* ağacı yeniden inşa edilir [41]. Birinci aşamada yoğun veri noktaları altkümelerde toplanıp, ayrık veri noktaları sıradışı kabul edilip atılarak bellekte veri setine ait bir özet oluşturulur. İkinci aşama kümeleme için kullanılacak algoritmaya bağlı olarak gerçekleştirilir. *CF*-ağacının yoğunlaştırılarak küçültülmesini sağlar. Tercihe bağlı bir işlem basamağıdır. Üçüncü aşama ana belleğe yüklenen *CF*-ağacının yapraklarındaki nesnelerin hiyerarşik veya bölümleme kümeleme metotları kullanılarak kümelendiği işlem basamağıdır. Bu aşamanın sonunda istenen kümeler elde edilmiş olur. Dördüncü aşama, üçüncü aşamada elde edilen kümelerin merkez noktalarını çekirdek noktalar olarak kullanarak veri noktalarını kendilerine en yakın çekirdek noktaya yerleştirir. Böylece yeni kümeler oluşur. Bu işlem bir kümedeki veri noktalarının sadece yerlerini değiştirmez, ayrıca bir veri noktasının tüm örneklerinin aynı kümede yer almasını sağlar[12].

BIRCH algoritmasının işleyişini daha iyi anlayabilmek için algoritma adımlarını bir örnek üzerinde inceleyelim. Şekil 2.1.1.1.10.2 dikkate alınarak 6 tane noktanın BIRCH algoritması ile kümeleme işlemini yapalım. Şekil üzerinde görüleceği üzere 1 ve 2. noktalar ile işlemlere başlanarak, 1 nolu noktayı iki nolu noktanın yer aldığı küme ikiye koyduğumuz zaman kümenin çok büyüdüğünü gözlediğimizi varsaydığımızda 1 nolu noktaya küme 1, iki nolu noktaya küme 2 adını verelim. Nokta 3 ise küme 1'e (entry 1) daha yakındır. Ancak eğer nokta 3'ü küme 1'e dahil edersek küme çok büyüyeceğinden nokta 3'ü diğer iki kümeden ayrı bir küme oluşturarak şekilde üçüncü bir kümeye dahil edilir. Nokta 4 ise küme 2'ye daha yakındır. Nokta 4 'de kümede aşırı bir büyümeye sebep olmadığından dolayı küme 2'ye dahil edilebilir. Nokta 5, küme 2'ye daha yakındır. Küme 3'e dahil edildiğinde ise kümede aşırı bir büyüme gerçekleşecektir. Ancak düğüm sayısındaki entrylerin sayısındaki kısıtlılıktan dolayı (işlemlerin başında küme sayısı belirlenmiştir) küme 3 bölünememektedir. Bu yüzden yaprak olmayan düğümler oluşur ve iki tane entryde ikişer küme olmak üzere 4 ayrı küme oluşur. Son olarak Nokta 6 ise, küme 3 e yakın olduğundan ve kümeyi aşırı büyütmediğinden dolayı

küme 3'e dâhil edilir. Sonuçta nokta 1 küme 1'e; nokta 3 ve 6 küme 3'e; nokta 5 küme 4'e ve nokta 2 ve 4 de küme 2'ye dâhil olmuştur.



Şekil 2.1.1.1.10.2. Altı noktanın BIRCH algoritması ile kümelenmesi

BIRCH Algoritmasının avantajlarını ise şöyle özetleyebiliriz[12,30,41].

- Sadece ilk aşamada *Girdi (INPUT=I)/Çıktı (OUTPUT=O)* işlemleri gerçekleştirilir. Daha sonraki aşamalarda veriler ana bellekten okunur.
- Aykırı değerlerin etkisini azaltmaktadır.
- Veri setini bir kez tarayarak iyi bir kümeleme sağlar.
- Nesne sayısı ile algoritmanın doğrusal ölçeklenebilirliği ve kümeleme güvenilirliği iyidir.

BIRCH Algoritmasının dezavantajları ise[12,30,41];

- Yalnızca sayısal verilerde kullanılabilir.

- Verinin okunma sırasına karşı duyarlıdır.
- Sadece küresel ve boyutları benzer kümeleri bulabilmektedir.
- Bu nedenlerden dolayı araştırmacının doğal küme olarak kabul ettiği küme ile aynı sonuçları vermeyebilir.

Tablo 2.1.1.1.10.1. BIRCH algoritmasının özet tanımlayıcı özellikleri

Algoritma	Özet
BIRCH	• Büyük veri tabanları için iyi sonuçlar üretmektedir. • Gürültülü verilerin kontrol edilmesi için geliştirilmiş ilk algoritmadır. • *CF* ve *CF*-tree yardımıyla kümeleme işlemlerini gerçekleştirir. • Sayısal veri setleri için kullanılmaktadır. • Küresel kümeler için kullanılmaktadır. • Hesaplama karmaşıklığı *O(n)*'dir.

2.1.1.1.11. CHAMELEON Kümeleme Algoritması (hierarchical clustering algorithm using dynamic modeling)

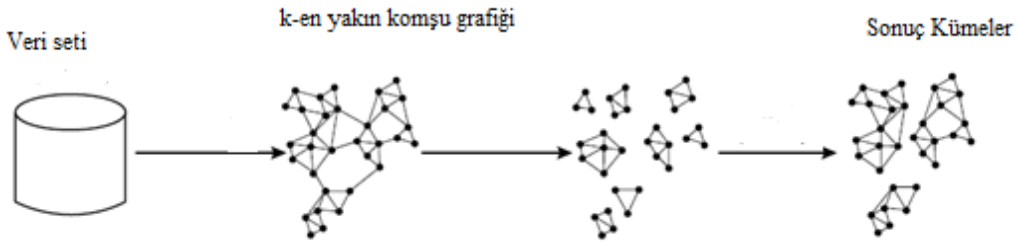
Küme çiftleri arasındaki benzerlikleri belirleyen ve hiyerarşik bir kümeleme algoritması olan CHAMELEON, ilk olarak 1999 yılında Karypis, Han ve Kumar tarafından kullanılmıştır. Hiyerarşik kümeleme yöntemleri arasında yer alan ROCK ve CURE algoritmalarının dezavantajları nedeniyle Chameleon algoritması türetilmiştir. Daha önceki bölümlerden hatırlanacağı üzere, ROCK algoritmasının düzeninde küme bağlanabilirlikleri vurgulanmakta ancak küme yakınlıkları ile bilgiler göz ardı edilmektedir. CURE algoritması ise ROCK algoritmasının tersine küme yakınlıkları ile ilgilenirken küme bağlanabilirliklerini ihmal etmektedir. CHAMELEON algoritmasında küme benzerliği küme içindeki nesnelerin ne kadar iyi bağlanmış olduğu ve kümelerin yakınlığı temeline dayanılarak değerlendirilir. Bu sebeple CHAMELEON algoritması kullanıcı kaynaklı bir modeldir ve otomatik olarak birleştirilen küme içi özelliklere uyum sağlayabilmektedir. Algoritmadaki birleştirme işlemi doğal ve homojen kümelerin keşfini kolaylaştırmakta ve bir benzerlik fonksiyonu yardımıyla tüm veri türleri için kolaylıkla uygulanabilmektedir[43,44].

CHAMELEON algoritması, grafik oluşturmak için *k* en yakın komşu grafik yaklaşımını kullanmaktadır. Bu grafikte kenar kesimleri minimum yapılmaya çalışılmaktadır. Bunun sebebi kenar kesimlerinin noktalar arasındaki benzerliği yansıtmasıdır. Kenar kesimleri ile küme benzerliği arasında doğru orantılı bir ilişki vardır. Grafiğin her bir köşesi bir veri

nesnesini temsil etmektedir ve eğer bir nesne diğer bir nesnenin, k adet en benzer nesne arasında ise iki köşe (nesne) arasında bir kenar oluşmaktadır. Kenarlar nesneler arasındaki benzerlikleri yansıtmak için de ağırlıklandırılmaktadır. Algoritma k en yakın komşu grafiğini nispeten küçük alt kümeye bölümlemek için grafik bölümleme algoritmasını kullanır. Daha sonra ise benzerliğe dayalı tekrarla alt kümeler birleştirilen bir toplamalı hiyerarşik kümeleme algoritmalarını kullanır. En yakın alt küme çiftlerini belirlemek için ise hem kümelerin bağlanabilirlik özelliğini hem de kümelerin yakınlık özelliğinden faydalanılmaktadır[43,44].

Özet olarak CHAMELEON algoritması iki aşamada gerçekleşir. İlk aşamada grafik bölümleme algoritması kullanır. İkinci aşamada ise kümeleri birleştirmek için toplamalı hiyerarşik veri madenciliği algoritmalarından yararlanılmaktadır.

K en yakın komşu grafiği dinamik modelleme yapısını kullanmaktadır. Bir başka ifade ile bu grafikte bir nesnenin komşu yarıçapı nesnenin bulunduğu alanın yoğunluğuna göre belirlenmektedir. Yoğun bir bölgede komşuluk dar tanımlanırken seyrek bir bölgede komşuluk daha geniş tanımlanmaktadır. DBSCAN gibi yoğunluğa dayalı yöntemlerle karşılaştırıldığında daha doğal kümeler oluşturma özelliğine sahiptir. Ayrıca bölgedeki yoğunluk, kenarların ağırlıkları olarak kaydedilir. Bir başka ifadeyle yoğun bölgedeki kenarların seyrek bölgedeki kenarlara göre daha ağır bir eğilimi vardır[43,44].



Şekil 2.1.1.1.11.1. CHAMELEON algoritmasının şekilsel gösterimi

Grafik bölümleme algoritması kenar kesimi (edge cut) minimize edecek şekilde k en yakın komşu grafiğini bölümlendirir. Yani bir C kümesi C_i ve C_j olmak üzere iki kümeye bölünür.

Daha önceden de belirtildiği gibi Chameleon algoritması nispi bağlanabilirliğe yakınlıklara göre C_i ve C_j küme çiftleri arasındaki benzerlikleri belirlemektedir. Nispi bağlanabilirlik $RI(C_i, C_j)$ ifadesiyle gösterilirken nispi yakınlık $RC(C_i, C_j)$ ifadesiyle gösterilmektedir[43,44].

C_i ve C_j kümeleri arasında nispi bağlanabilirlik, C_i ve C_j kümelerinin iç bağlanabilirliğine göre normalleştirilebilen C_i ve C_j kümeleri arasındaki mutlak bağlanabilirlik şeklinde tanımlanır. Nispi bağlanabilirlik hesaplamak için aşağıdaki formülden yararlanılmaktadır[43,44].

$$RI(C_i,C_j)=\frac{|EC_{\{C_i,C_j\}}|}{\frac{1}{2}(|EC_{C_i}|+|EC_{C_j}|)}$$

Yukarıdaki formülde yer alan $EC_{\{C_i,C_j\}}$, C_i ve C_j kümelerini içeren kümenin kenar kesitini gösterirken EC_{C_i} veya EC_{C_j}, C_i veya C_j'yi kabaca iki eşit parçaya bölen kenar kesitlerinin toplamlarının minimumudur[43,44].

$C_i$ve C_j arasındaki nispi yakınlık RC(C_i, C_j) ise, iki kümenin iç yakınlığına göre normalize edilmiş $C_i$ve C_j arasındaki mutlak yakınlıktır. Nispi yakınlık değeri aşağıdaki gibi formülüze edilmektedir[43,44].

$$RC(C_i, C_j)= \frac{\overline{S_{E_{\{C_i,C_j\}}}}}{\frac{|C_i|}{|C_i|+|C_j|}\bar{S}_{EC_{C_i}}+\frac{|C_j|}{|C_i|+|C_j|}\bar{S}_{EC_{C_j}}}$$

Formülde yer alan $\overline{S_{E_{\{C_i,C_j\}}}}$, C_i ile C_j köşelerini birleştiren kenarlarının ortalama ağırlıklarıdır. $\bar{S}_{EC_{C_i}}$(veya $\bar{S}_{EC_{C_j}}$) ise, C_i (veya C_j) kümesinin minimum kesim açıortayına ait kenarlarının ortalama ağırlığıdır.

Chameleon algoritması bağlanabilirliği ve yakınlığı birlikte kullanabilmek için bir fonksiyon kullanır. Chameleon bu fonksiyonu maksimum yapan küme çiftlerini seçmektedir. Çünkü amacımız hem bağlanabilirliğin hem de yakınlığı yüksek olan çiftleri birleştirmektir. Yani kümeleri $RI(C_i,C_j)$* $RC(C_i, C_j)$'yi maksimum olacak şekilde seçmektedir. Bu formül her iki parametreye de eşit önem vermektedir. Ancak bu ölçüden birinin daha yüksek olması küme oluşturmada tercih edilebilir. Bu sebeple Chameleon $RI(C_i,C_j)$* $RC(C_i,C_j)^{\alpha}$'yi maksimum yapan küme çiftlerini seçer. Formülde yer alan α kullanıcı tarafından belirlenen bir parametredir. Eğer α>1 seçilirse, Chameleon algoritması nispi yakınlığa daha fazla önem verirken α<1 olması durumunda nispi bağlanabilirliğe daha fazla önem vermektedir. Chameleon keyfi şekildeki kümelerin bulunması için çok iyi bilinen BIRCH ve yoğunluk tabanlı bir algoritma olan DBSCAN algoritmalarından daha güçlü bir algoritmadır. Ancak çok boyutlu veriler için maliyet yani hesaplama karmaşıklığı n nesne için O(n^2)'dir[43,44].

Tablo 2.1.1.1.11.1. CHAMELEON algoritmasının özet tanımlayıcı özelikleri

Algoritma	Özet
CHAMELEON	• Küme bağlanabilirliğini ve yakınlığını birlikte kullanmaktadır. • Keyfi yapıdaki kümeler için kullanılmaktadır. • ROCK ve CURE algoritmalarının dezavantajlarını gidermektedir. • Hesaplama karmaşıklığı O(n^2)'dir. • Grafik bölümleme ve toplamalı hiyerarşik algoritmalarından yararlanarak kümeleme işlemlerini iki aşamada gerçekleştirmektedir.

2.1.1.2. Ayrıştırıcı Hiyerarşik Kümeleme Algoritmaları (divisive hierarchical clustering: DIANA)

DIANA (Divisive Analysis) ilk olarak 1990 yılında Kaufman ve Rousseuw tarafından sunulmuştur. Birleştirici kümeleme algoritmalarının tersine yukarıdan aşağıya doğru bir yapısı vardır. Yani ilk başta tüm nesneler tek bir küme olarak kabul edilir. Daha sonra her bir nesne kendi içinde bir küme oluncaya kadar işlemlere devam edilir. Bu işlem her nesne kendi başına bir küme oluşturana kadar veya belli bir sonlandırma koşulu sağlanana kadar devam eder. Sonlandırma koşulu istenen sayıda küme elde edilmesi veya en yakın iki küme arasındaki uzaklığın verilen eşik değerin üzerinde olması sağlanır[12,42,45,46].
Birleştirici kümeleme algoritmalarına benzer olarak ayrıştırıcı hiyerarşik kümeleme algoritmalarının da sunumu ve yorumlamasının kolay olmasına rağmen yanlış küme seçimi hatalara sebep olmaktadır. Yöntem çok büyük sayıda veri içeren ve uç değerleri olan çalışmalar için başarılı sonuçlar vermemektedir.

Ayrıştıcı kümeleme algoritmaları monotetik ve politetik olmak üzere iki ana alt başlık altında incelenmektedir.

2.1.1.2.1. Monotetik Ayrıştırıcı Kümeleme Algoritması

Bu kümeleme algoritması 1990 yılında Kaufman ve Rousseeuw tarafından sunulmuştur. Bu yöntemde her bir bölünme tek bir değişkene dayanmaktadır. Ancak diğer birçok hiyerarşik

yöntemler tüm değişkenleri birlikte kullandığı için politetiktir. Monotetik kümeleme algoritması büyük bir kümeden başlayarak kümelerin bir hiyerarşisini kurmaktadır. Aynı kümedeki tüm gözlemler tüm değişkenler için aynı değere sahip olana kadar kümeler bölünmektedir. Her bir aşamada, tüm kümeler tek bir değişkenin değerlerine göre ayrılmaktadır. Bir küme tüm gözlem değerleri 1'i içeren gözlemlerden oluşurken, diğer küme tim gözlem değerleri 0'ı içeren gözlemlerden oluşmaktadır. İkili özellikteki veri tipleri için kullanılmaktadır. Yöntem kayıp veri olduğunda çalışmamaktadır. Dolayısıyla kayıp verinin varlığı durumunda uygun veri kayıp değerlendirme yöntemleri kullanılarak kayıp veriler doldurulur ardından yöntemin uygulaması gerçekleştirilmelidir. [46]

Monotetik kümeleme algoritması ikili değişkenler içeren veri matrisleri üzerinde çalışmaktadır. Yöntemde ilk olarak ikili veri matrisindeki tüm kayıp değerler tahmin edilen değerler ile yer değiştirir. x_{if}'nin kayıp olduğunu varsayın, o zaman başka bir değişken düşünülür ve kontenjans tablosu oluşturulur.

Tablo 2.1.1.2.1.1. Kontenjans tablosu

Özellikler	g	
f	1	0
1	a_{fg}	b_{fg}
0	c_{fg}	d_{fg}

f ile g özellikleri arasındaki ilişki aşağıdaki şekilde tanımlanmaktadır.

$$A_{fg} = |a_{fg}d_{fg} - b_{fg}c_{fg}|$$

$a_{fg}+d_{fg} + b_{fg} + c_{fg}$ toplamı f özelliğinin kayıp olmayan toplam gözlem sayısını göstermektedir. Her bir g özelliği için A_{fg}hesaplandıktan sonra, f özelliğinin maksimum olduğu özelliğinin ilişkileri için *t* özelliği belirlenir.

$$A_{ft} = maxA_{fg}$$

t özelliği yardımıyla *f* kayıp değerleri aşağıdaki şekilde tahmin edilmektedir.

Eğer $a_{fg}d_{fg} - b_{fg}c_{fg}$>0 ise x_{if}=x_{it} 'dir.

Aksi durumda x_{if}=1 − x_{it}'dir.

Oluşturulan tamamlanmış veri matrisi ile işlemlere devam edilir [47]. Monotetik yöntemlerde yeni gelen bir nesne kolayca sınıflandırılmaktadır [48].

2.1.1.2.2. Politetik Ayrıştırıcı Kümeleme Algoritması

Yöntem 1964 yılında Macnaughton-Smith ve arkadaşları tarafından geliştiril olup algoritması tüm özelliklerin bilgilerini kullanır. Ayrıca kümeleme için yakınlık matrisi kullanılmasından dolayı bu yöntemler birleştirici yöntemlere benzerdir. Hesaplama hızını artırmak için tüm olası bölünmelere göz önüne almaktan kaçınmaktadır. Grup içinde diğerinden en uzak olan nesneyi bularak işlemlere devam eder. Bir başka ifade ile veri seti ayrılan ve kalan grup olmak üzere iki gruba ayrılır. Geri kalan grubundaki her nesne için, diğer nesneler ile ortalama benzerlikleri hesaplanır ve ayrılan gruptaki nesneler ile ortalama farklılıkları çıkarılır[48].

Hiyerarşik kümeleme yöntemleri hakkında verilen bilgileri Tablo 2.1.1.2.2.1' deki gibi özetleyebiliriz.

Tablo 2.1.1.2.2.1. Hiyerarşik kümeleme algoritmalarının özet tanımlayıcı özellikleri

Yöntem	Özet
Tek Bağlantı	• Birleştirici hiyerarşik kümeleme algoritmasıdır. • En yakın komşu yöntemi olarak bilinmektedir. • İki küme arasındaki uzaklık iki kümenin birbirine en yakın gözlemleri arasındaki uzaklıktır. • Küresel ve farklı boyuttaki kümeler için kullanılabilir. • Uç değerlere karşı duyarlıdır. • Kümeler oluştururken dağınık yapıdaki kümeleri oluşturma eğilimindedir.
Tam Bağlantı	• Birleştirici hiyerarşik kümeleme algoritmasıdır. • En uzak komşu algoritması olarak bilinir • Küme uzaklıkları olarak birbirine en uzak gözlemler arasındaki uzaklık alnır. • Uç değerlere karşı daha az duyarlıdır. • Konveks şekildeki kümeleri bulmak için elverişli değildir. • Konkav yapıdaki kümeler için uygundur.
Ortalama Bağlantı	• Birleştirici hiyerarşik kümeleme algoritmasıdır. • İki küme arasındaki uzaklık ayrı gruplarda yer alan gözlem çiftleri arasındaki ortalama uzaklıktır. • Tek bağlantı ve tam bağlantı teknikleri arasında sonuçlar vermesi nedeniyle bir alternatif yöntem olarak önerilmektedir. • Düşük varyanslı kümeleri birleştirme eğilimindedir. • Aynı varyansa sahip kümeler için biraz yanlı sonuçlar

	vermektedir.
Ağırlıklandırılmış ortalama bağlantı yöntemi	• Birleştirici hiyerarşik kümeleme algoritmasıdır. • Mc Quitty yöntemi olarak bilinmektedir. • Her bir birleştirme işleminden sonra yeni oluşan küme ile eski küme arasındaki uzaklıklar birleştirilen iki kümenin uzaklıklarına bağlı olarak hesaplanır.
Merkez yöntemi	• Birleştirici hiyerarşik kümeleme algoritmasıdır. • Merkez bağlantı yöntemi en çok kullanılan link fonksiyonlarındandır. • İki küme arasındaki uzaklık merkezler arasındaki ortalama uzaklıktır. • Kullanımı basit, anlaşılması ve yorumlanması kolay bir yöntemdir. • Uç değerlerden az etkilenmektedir. • Ters dönmeden kaynaklanan (reversal) bir karmaşıklıklık vardır.
Medyan yöntemi	• Birleştirici hiyerarşik kümeleme algoritmasıdır. • İki küme arasındaki uzaklık bir kümedeki gözlem ile diğer kümedeki gözlem arasındaki medyan uzaklığıdır. • Uç değerlerden fazla etkilenmemektedir. • Birleştirilecek kümelere eşit ağırlık verilmektedir.
Ward yöntem	• Birleştirici hiyerarşik kümeleme algoritmasıdır. • En küçük varyans yöntemi olarak bilinmektedir. • Kümeleme analizini varyans analizi problemi olarak ele alır. • Küme içi hata kareler toplamı kullanılmaktadır. • Küme içinde homojenliği maksimum yapacak şekilde küme içi kareler toplamı minimize eden kümeleri oluşturmaktadır. • Aynı şekle sahip ve aynı sayıda gözlem içeren kümelerin üretilmesine karşı yanlıdır. • Uç değerlere karşı da duyarlı bir bağlantı yöntemidir. • Niceliksel veriler için uygun iken ikili değişkenler için uygun değildir.
Monotetik kümeleme yöntemi	• Bölücü hiyerarşik kümeleme algoritmasıdır. • Her bir bölünme tek bir değişkene dayanmaktadır. • Büyük bir kümeden başlayarak kümelerin bir hiyerarşisini kurulmaktadır. • Yöntem kayıp veri olduğunda çalışmamaktadır. • Aynı kümedeki tüm gözlemler tüm değişkenler için aynı değere sahip olana kadar kümeler bölünmektedir. • İkili değişkenler içeren veri matrisleri üzerinde çalışmaktadır.
Politetik kümeleme yöntemi	• Bölücü hiyerarşik kümeleme algoritmasıdır. • Tüm değişkenlerin bilgilerini kullanmaktadır. • Hesaplama hızını artırmak için tüm olası bölünmelere göz önüne almaktan kaçınmaktadır.

2.1.1.2.3. HCAD Kümeleme Algoritması (Hierarchical clutering based on attribute dependency algorithm)

HCAD kümeleme algoritması, değişken bağımlılıklarını dikkate alarak veri setini parçalamaktadır. Algoritma büyük veri setleri için var olan diğer algoritmaların çoğundan daha iyi çalışmaktadır. Hem etiketli hem de etiketsiz veri setleri için uygulanabilmektedir. Önerilen HCAD kümeleme yöntemi, ayrıştırıcı hiyerarşik kümeleme yöntemleri arasındadır. Değişken bağımlılıklarına bağlı olarak veri nesnelerini ayrıştırmaktadır. Değişken bağımlılıkları ise Spearman rank korelasyonu ile hesaplanmaktadır. Bu ölçüm iki değişken arasındaki ilişkinin kuvvetidir. *x* ve *y* değişken çiftleri arsındaki korelasyon aşağıdaki formül yardımıyla hesaplanmaktadır[49].

$$r_s = 1 - \frac{6\Sigma D^2}{N^3 - N}$$

Formülde yer alan D terimi, her bir nesnenin rankları arasındaki fark iken r_s rank korelasyonları, N ise toplam nesne sayısıdır. Değişken çiftleri arasındaki korelasyonlar hesaplanmakta ve matris formunda gösterilmektedir ve bu matrise bağımlılık matrisi (R) adı verilir. Bağımlılık matrisinden en yüksek bağımlılık değerine sahip değişkenler ayrıştırma için ele alınmaktadır. Seçilen değişkenler arasındaki eşitlik ilişkisi ile değerlendirilen ayrışmaya karar verilmektedir. Ardından seçilen değişkenler değişken listesinden çıkartılarak herhangi bir ayrıştırma işlemi olmayıncaya kadar işlemlere devam edilmektedir[49].

HCAD kümeleme algoritmasının işlem adımları aşağıda özetlenmiştir.

1. A değişken listesindeki değişken çiftleri için korelasyonlar hesaplanarak R matrisi oluşturulur.
2. R matrisindeki maksimum bağımlılık değerine sahip a_i ve a_j değişkenleri bulunur.
3. A listesinden a_i ve a_j değişkenleri silinir.
4. Denklik sınıfı bulunur.
5. Tüm k parçaları elde edilinceye kadar 1'den 4'e kadar işlemlere devam edilir.

HCAD kümeleme algoritmasının aşamaları arasında yer alan eşitlik teriminden bahsedilmiştir. A setinde S'nin eşitlik ilişkisi simetriklik, geçişlilik ve dönüşlülük özellikleri sağlandığında meydana gelmektedir. Sayısal ve karmaşık veriler için kullanılan Mahalanobis uzaklığı hesaplanarak sayısal veriler için eşitlik ilişkisi elde edilebilmektedir. Kategorik verilerde ise diğer diziler ile eşleşen nesneler bulunarak eşitlik ilişkisi hesaplanır. Kümeleme yöntemindeki bölmeler (parçalar) ise eşitlik ilişkine bağlıdır. Örneğin, kategorik bir veride a_i değişkeni {yüksek, orta, düşük}, a_j ise {evet, hayır} olduğunda en iyi durumda bölme sayısı

{(yüksek, evet), (yüksek, hayır), (orta, evet), (orta, hayır), (düşük, evet), (düşük, hayır)} şeklinde 6 tane olacaktır[49].

Yöntemde kümeleme sonuçları entropi ve saflık (purity) kavramları kullanılarak değerlendirilmektedir. Entropi değeri aşağıdaki formül kullanılarak hesaplanmaktadır.

$$E(C)=-\sum_{i=1}^{m} p_i log_2 p_i$$

Formülde yer alan p_i değeri, *i.* kümeye ait olan *m* tane küme için veri nokta olasılıklarıdır. Saflık değeri ise normalleştirilmiş ortak bilgiler kullanılarak, *H(c)* entropi değeri, c_j sınıfına ait olan nesne olasılığı hesaplanarak ve w_i kümesinde sabitlenerek aşağıdaki şekilde hesaplanmaktadır[49].

$$Saflık(\upsilon.C)=\frac{I(\upsilon,C)}{[H(\upsilon)+H(C)/2}$$

$$I(\upsilon,C)=\sum_i \sum_j P(\omega_i \cap c_j) log \frac{p(\omega_i \cap c_j)}{p(\omega_i)p(c_j)}$$

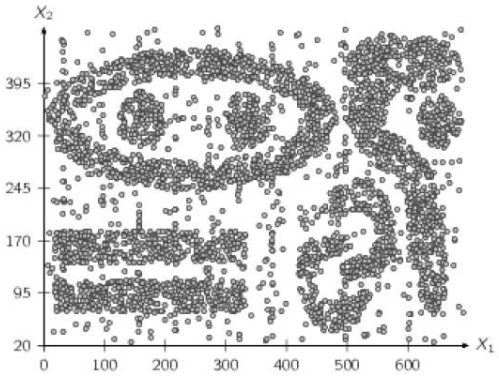
$\upsilon=\{\omega_1, \omega_2, \ldots., \omega_i\}$ kümelerin seti iken $C=\{c_1, c_2, \ldots., c_j\}$ sınıfların setidir.

HCAD kümeleme algoritması, zaman ve hesaplama karmaşıklığı bakımından yoğunluk yöntemlerinden daha avantajlıdır. Yöntemde küme sayıları ile ilgili bir kısıtlama yoktur ve düzensiz kümelerin bulunmasına da katkı sağlamaktadır. Kullanıcı tarafından daha önceden belirlenen merkez veya küme sayısı gibi parametrelerin belirlenmesine ihtiyaç yoktur. Dolayısıyla parametre seçiminde kaynaklanabilecek yanlış seçimlerin oluşması da engellenmektedir[49].

2.1.2. Yoğunluğa Dayalı Kümeleme Algoritmaları (density based clustering algorithms)

Yoğunluğa dayalı algoritmalar noktalar arasındaki uzaklıklar yardımıyla kümeleri tespit etmez, yoğunluğa bağlı olarak kümeleme işlemlerini gerçekleştirir. Düzensiz şekildeki kümeler için noktalar arasındaki uzaklıklar değerlendirilerek kümeleme işlemlerini yapmak çok zordur, hatta araştırmacıyı hatalı sonuçlara bile götürmektedir. Bu tür kümeler yoğunluğa dayalı kümeleme yöntemleri ile değerlendirilmelidir. Yoğunluğa dayalı yaklaşımların ana prensibi, yüksek ve düşük yoğunluktaki bölgeleri bulabilmek ve yüksek yoğunluklu bölgeleri düşük yoğunluklu bölgelerden ayırabilmektir. Böylece düzensiz şekildeki kümeler daha kolay

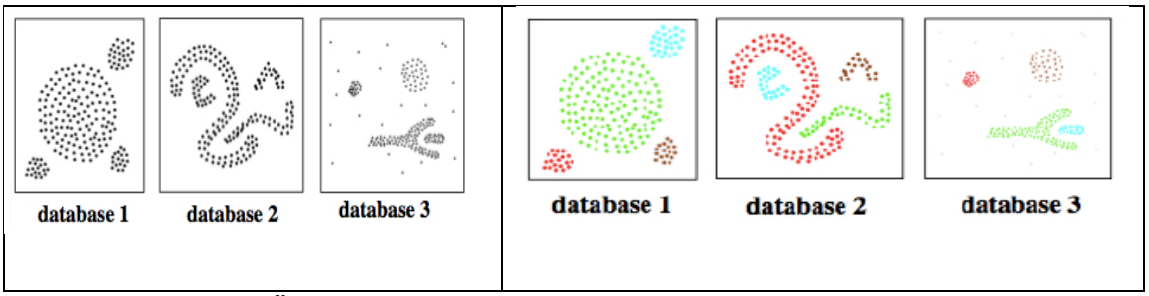
ayırt edilebilinecektir[50]. Dolayısıyla Şekil 2.1.2.1.'de verilen konveks olmayan kümeler için *K*-ortalama gibi yöntemler gerçek küme yapısını bulmada zorlanmaktadır. Çünkü bu tür kümelerde farklı kümelere ait olan iki nokta arasındaki uzaklık aynı kümede olan iki nokta arasındaki uzaklıktan daha yakın olabilme ihtimali vardır. Bu durumda yalnızca kümeler arasındaki uzaklıklardan faydalanmak doğru bir yaklaşım olmayacaktır. Yoğunluğa dayalı kümeleme algoritmalarının bir başka avantajı da gürültülü ve aykırı değerlerin olduğu verilerin bulunmasında kullanılabilir olmasıdır[51].



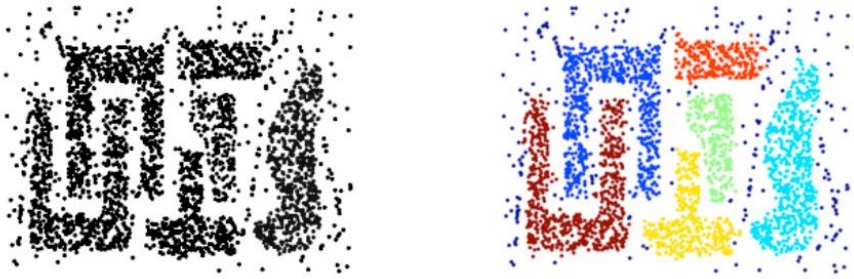
Şekil 2.1.2.1. Farklı şekillere sahip kümeler

2.1.2.1. DBSCAN Kümeleme Algoritması (A Density-Based Spatial Clustering of Application with Noise)

DBSCAN kümeleme yöntemi yoğunluğa dayalı bir algoritmadır. İlk olarak Martin Ester ve arkadaşları tarafından geliştirilmiştir [33]. DBSCAN algoritması farklı şekillere sahip düzensiz ve gürültülü verilere sahip kümelerin bulunmasında kullanılmaktadır. Diğer kümeleme yöntemleri ile düzensiz şekillere sahip kümeler incelenmesi oldukça zor ve zahmetlidir. Ayrıca hesaplamasının maliyeti de yüksektir. Aşağıdaki şekillerde düzensiz kümelerin olduğu veritabanlarında DBSCAN algoritmasının ne kadar isabetli karar verdiği görülmektedir (Şekil 2.1.2.1.1, Şekil 2.1.2.1.2)[33,52].

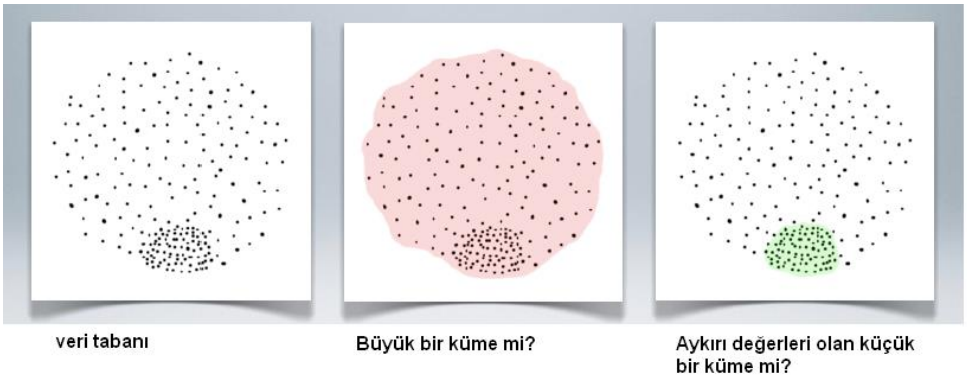


Şekil 2.1.2.1.1. Üç farklı veritabanının DBSCAN yöntemi ile kümelenme sonucu



Şekil 2.1.2.1.2. DBSCAN yöntemi kullanılarak elde edilen kümeleme sonuçları

DBSCAN algoritması tek bir girdi parametresi kullanarak veri tabanlarındaki yerel yoğunluklara bakarak büyük hacimlere sahip uzaysal veri setleri için kümeleri belirlemektedir[52]. Ayrıca yöntemde hangi bilginin aykırı veri veya gürültülü veri olarak değerlendirileceğine de karar verilmektedir. Örneğin Şekil 2.1.2.1.3'deki gibi bir veri setinin olduğu varsayılırsa noktaların gerçekten bir küme oluşturduğu sonucuna mı varılmalı yoksa aykırı değerleri fazla olan küçük çaplı bir küme mi oluşturduğu sonucuna mı varılmalıdır. İşte DBSCAN algoritması bu kararın sonucuna doğru bir şekilde varabilmektedir[53].



Şekil 2.1.2.1.3. Veri tabanı ve küme kararı

DBSCAN algoritmasının çalışma yöntemine geçmeden önce birkaç kavram tanımlamalarının yapılması gerekmektedir.

ε(Eps) komşuluğu

Bir nesnenin ε yarıçapı içindeki komşuluğuna denir. *Eps* bir nesnenin komşularını belirlemek için gerekli olan yakınlık mesafesidir. Bir başka ifadeyle bir kümedeki iki nokta arasındaki maksimum uzaklıktır. $N_{Eps}(q)=\{q \in D | dist(p,q) \leq Eps\}$ şeklinde gösterilir[52,54,55].

MinPts

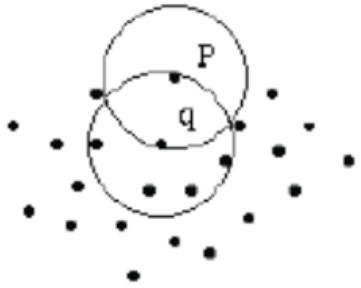
Bölgenin yoğun olarak belirlenebilmesi için ε komşuluğundaki minimum nokta sayısıdır.

Çekirdek nesne

Nesne, ε komşuluğunda en az *MinPts* adet nesne içeriyorsa çekirdek nesnedir.

Yoğunluğa erişebilirlik

$p_1,p_2,...p_n$ noktalar zinciri olduğunda $p_1=q$ ve $p_n=p$'dir. *Eps* ve *MinPts* koşulları altında *p* noktasının *q* noktasından yoğunluk erişebilirliği p_{i+1}, p_i noktası yardımıyla olmaktadır [33]. Bu durum aşağıdaki şekilde özetlenmektedir. *p* ve *q* noktaları olsun. Bu durumda *p* noktası *q*'ya doğrudan erişebilir iken *q* noktası *p*'ye doğrudan erişilebilir değildir. Çünkü *q* noktası çekirden nesnedir (MinPts=3 kabul edilndiğinde)[52].



Şekil 2.1.2.1.4. *p* noktasının *q* noktasına doğrudan erişilebilirliği

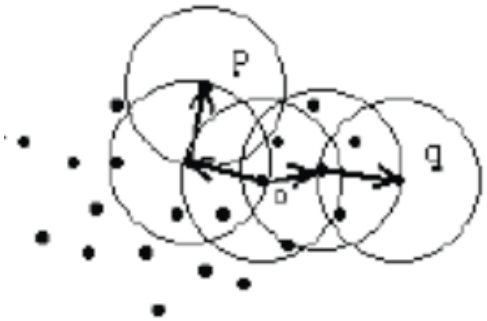
Yoğunluğa doğrudan erişebilirlik (directly density reachable)

Aşağıdaki koşullar sağlandığı durumlarda *p* noktası *q* noktasına yoğunluğa doğrudan erişilebilirdir denilir [33].

1) $p \in N_{Eps}(q), N_{Eps}(q)$
2) $|N_{Eps}(q)| \geq MinPts$

Yoğunluk bağlantısallığı (density connected)

Eps ve *MinPts* koşulları altında bir *p* noktası bir *q* noktasına bir başka nokta olan *o* noktası yardımıyla bağlanabilmektedir ve bu *o* noktası hem *p* noktasına hem de *q* noktasına yoğunluğa erişilebilirdir. Bu durum aşağıdaki gibi şematize edilmektedir[52].

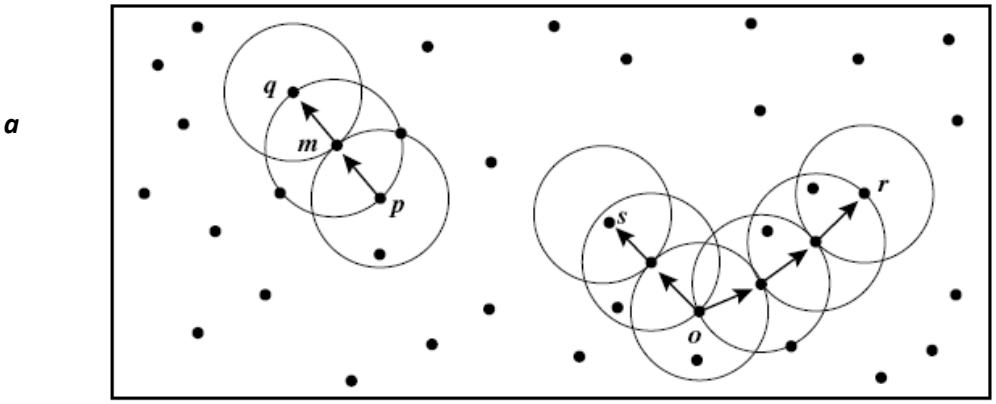


Şekil 2.1.2.1.5. Yoğunluk bağlanabilirlik

Gürültü (noise)

Veri tabanında herhangi bir kümeye ait olmayan noktalar setleridir.

Yukarıda tanımlanan kavramları özetlemek amacıyla aşağıdaki örnek verilmektedir. *MinPts* değerinin 3 olduğu varsayıldığında, Şekil 2.1.2.1.6'da yer alan *m,p,o,r* noktaları ε komşuluklarında 3 ve üçden fazla nokta bulundurdukları için çekirdek nesne olur. *q* noktası *m*' den; *m* noktası, *q*' dan doğrudan erişilebilirdir. Bu hükümlerin tersleri de doğrudur. *q* noktası, *m*'den ve *m* noktası da *p*'den doğrudan yoğunlukça erişilebilir olduğundan *q* noktası, *p*'den dolaylı olarak yoğunlukça erişilebilirdir. *q,r*,s noktaları ise birbirleriyle yoğunlukça bağlıdır. *a* noktası ise herhangi bir kümeye ait olmadığından gürültülü olarak kabul edilmektedir[54].



Şekil 2.1.2.1.6. DBSCAN algoritmasının uygulamasında kullanılan örnek veri seti

DBSCAN algoritmasında bir *q* noktasının *K* kümesine dâhil olabilmesi için iki koşul söz konusudur. İlk koşul *p* noktası *K* kümesinin elemanı olduğu durumda *p* vasıtasıyla yoğunluğa erişilebilir olmasıdır. İkinci koşul *p* noktası *q* noktası ile yoğunluk bağlantısallığa sahip olmasıdır. Bu iki durumda da *p* ve *q* noktaları *K* kümesinin elemanıdır.

DBSCAN algoritması rasgele olarak bir *p* noktası seçer ve belirlenen *MinPts* ve *Eps* şartları altındaki tüm yoğunluğa erişilebilecek noktaları belirlemektedir. Eğer *p* noktası yoğunluğun sınırını oluşturan bir nokta ya da hiçbir nokta *p* üzerinden yoğunluğa erişemiyorsa *p* yerine başka bir nokta rasgele olarak seçilerek işlemlere devam edilir. Bu işlemler tüm küme içindeki noktalar sabitleşene kadar devam etmektedir [33].

DBSCAN algoritması kullanıcı tarafından belirlenen uygun parametreler ve *MinPts* yardımıyla düzensiz şekilleri sahip kümeleri bulmada oldukça etkili bir yöntemdir. Eğer mekânsal bir endeks kullanılmış ise, algoritmanın hesaplama karmaşıklığı *O(nlogn)*'dir. Burada *n* veri setindeki nesnelerin sayısını göstermektedir. Diğer durumlarda hesaplama karmaşıklığı $O(n^2)$'dir [54].

2.1.2.2. DENCLUE Kümeleme Algoritması (Density Based Clustering)

Çeşitli kümeleme yöntemleri çok boyutlu veriler olduğunda kullanılabilmektedir. Ancak bu kümeleme yöntemlerinin etkinliği gürültülü veriler çok sayıda olduğunda ve yüksek boyutlu veriler mevcut olduğu durumlarda oldukça sınırlıdır. Bu dezavantajları giderebilmek için Hinneburg ve Keim tarafından KDD'98 konferansında yoğunluğa dayalı bir kümeleme algoritması olan DENCLUE yöntemi önerilmiştir[12]. Sunulan bu yöntem yüksek boyutlu ve gürültülü verilere karşı oldukça iyi sonuçlar üretmektedir.

DENCLUE algoritmasının temelinde yoğunluk çekici ve etki fonksiyonu olmak üzere iki kavram vardır. Kümeler matematiksel olarak yoğunluk çekiciler kavramları ile hesaplanmaktadır. Dolayısıyla DENCLUE kümeleme algoritmasının yoğun bir matematiksel alt yapısı vardır. Yöntem aslında tüm kümeleme yöntemlerinin genelleştirilmiş hali olarak düşünülebilir[12,30,33,56].

Algoritmanın işleyişine geçmeden önce birkaç kavram üzerinde durulacaktır. DENCLUE yöntemi kümeleme işlemlerini gerçekleştirirken temel olarak etki fonksiyonundan yararlanmaktadır[12,30,33,56].

Etki fonksiyonu, her bir veri noktasına ait etkinin matematiksel fonksiyonlar kullanılarak modellenmesidir. F^d, *d* boyutlu uzay olarak tanımlandığında, her hangi bir noktadaki $x \in F^d$ yoğunluk fonksiyonu olarak gösterilir. Bir veri nesnesinin $y \in F^d$ olan etki fonksiyonu $f_B^y : F^d \rightarrow R_0^+$ olarak tanımlanmaktadır[12,30,33,56].

$f_B^y(x)=f_B(x,y)$'dir.

Etki fonksiyonu, bir veri noktasının komşuları içindeki etkisini tanımlamaktadır. Etki fonksiyonu her bir veri noktası için uygulanmaktadır. Bu fonksiyona örnek olarak kare dalga fonksiyonu veya Gauss fonksiyonu verilebilmektedir [12]. Bu fonksiyonlar sırasıyla aşağıdaki gibi tanımlanmaktadır[12,30,33,56].

Kare dalga fonksiyonu

$f_{kare}(x,y)=0$ eğer $d(x,y) > \sigma$

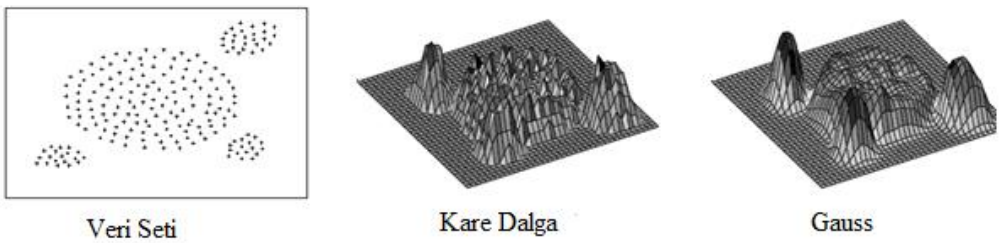
$=1$ diğer durumda

Formülde yer alan σ değeri, bir noktanın kendi komşuluğunda olan etkisini göstermektedir.

Gauss baskınlık fonksiyonu

$$F_{Gauss}(x,y)=e^{-\frac{d(x,y)^2}{2\sigma^2}}$$

Fonksiyonda yer alan *d(x,y),* Öklid uzaklığı gibi simetrik bir fonksiyondur[12,30,33,56].
Şekil 2.1.2.2.1'de iki boyutlu bir uzayda yer alan veri noktaları seti için, kare dalga ve gauss etki fonksiyonu kullanıldığında elde edilen genel yoğunluk fonksiyonu yer almaktadır[12,30,33,56].



Şekil 2.1.2.2.1. İki boyutlu bir uzayda Kare dalga ve Gauss etki fonksiyonu

Genel yoğunluk fonksiyonu ise tüm veri noktalarının toplanması ile hesaplanmaktadır. Bir başka ifadeyle yoğunluk fonksiyonu, tüm veri noktalarının etki fonksiyonlarının toplamından

oluşmaktadır. $D=\{x_1,....x_N\}$ özellik vektörü tanımlanan N veri nesnesi için genel yoğunluk fonksiyonu aşağıdaki gibi tanımlanmaktadır[12,30,33,56].

$$f_B^D(\mathrm{x})=\sum_{i=1}^{N} f_B^{x_p}(x)$$

Genel yoğunluk fonksiyonunun matematiksel formu, keyfi yapıdaki kümeleri bulmaya olanak sağlamaktadır. Ancak birçok veri noktası genel yoğunluk fonksiyonuna katkıda bulunmamaktadır. Bu yüzden de DENCLUE algoritması genel yoğunluk fonksiyonuna katkı sağlayan veri noktalarını göz önüne alan lokal bir genel yoğunluk fonksiyonu kullanmaktadır[12,30,33,56].

Genel yoğunluk fonksiyonu hesaplandıktan sonra yoğunluk çekiciler matematiksel olarak belirlenmektedir. Yoğunluk çekiciler, genel yoğunluk fonksiyonunun lokal maksimumudur. Eğer genel yoğunluk fonksiyonu sürekli ve herhangi bir noktada değişebilir özelliğe sahip ise, genel yoğunluk fonksiyonu gradiyenti kullanılarak yoğunluk çekiciler bulunmaktadır. Dolayısıyla yoğunluk çekicilerin hesaplanabilmesi için öncelikle genel yoğunluk fonksiyonunun gradiyentinin tanımlanması gerekmektedir. $f_B^D(\mathrm{x})$'nin gradiyent fonksiyonu aşağıdaki gibi tanımlanmaktadır[12,30,33,56].

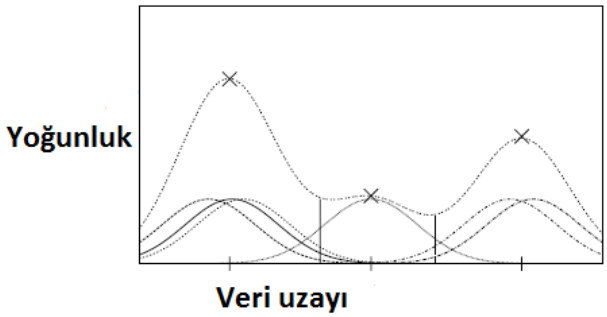
$$\nabla f_B^D(\mathrm{x})=\sum_{i=1}^{N}(x_i - x)\, f_B^{x_p}(x)$$

Gauss etki fonksiyonu durumunda gradiyent aşağıdaki gibi tanımlanmaktadır.

$$\nabla f_{Gauss}^D(\mathrm{x})=\sum_{i=1}^{N}(x_i - x)\, e^{-\frac{d(x,x_i)^2}{2\sigma^2}}$$

Etki fonksiyonu simetrik ve sürekli yapıda iken, gradiyent fonskyionu bu özelliklerden bağımsızdır[12,30,33,56].

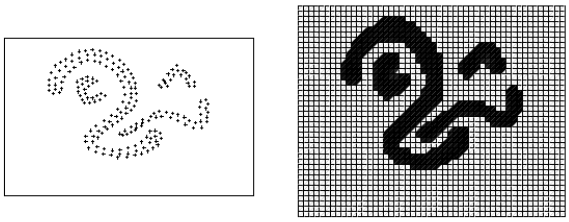
x^*, f_B^D yoğunluk fonksiyonunun lokal maksimumu ise, $x^* \in F^d$ noktası yoğunluk çekici olarak adlandırılır. Şekil 2.1.2.2.2'de tek boyutlu bir uzayda, yoğunluk çekiciliğe örnek verilmiştir. Şekilde sürekli bir etki fonksiyonu için x ekseni veri uzayını, y ekseni ise yoğunlukları göstermektedir. Bu yoğunlukların maksimumu ise çarpı işareti ile (x) belirtilmiştir. Bu x işaretleri, yoğunluk çekicileri ifade etmektedir[12,30,33,56].



Şekil 2.1.2.2.2. Yoğunluk çekici için örnek grafik

DENCLUE algoritması σ ve ξ olmak üzere iki parametre kullanmaktadır. ξ parametresi yoğunluk çekicilerin anlamlı olup olmadıklarını ifade etmektedir. Bir başka ifadeyle yoğunluk çekicilerin önemli olacağı en düşük yoğunluk seviyesidir. Bu parametre yoğunluk çekici sayısının azalmasına ve performansın artmasına neden olmaktadır. σ paremetresi ise bir veri noktasının kendi çevresine (komşularına) olan etkisini göstermektedir. ξ parametresi *MinPts*, σ paremetresi ise *Eps* olarak düşünülebilir[12,30,33,56].

DENCLUE algoritması diğer kümeleme yöntemlerinin genelleştirilmiş halidir. Örneğin, keyfi şekillere sahip kümeler bulunurken σ = *Eps* ve ξ= *MinPts* olacak şekilde kare dalga fonksiyonu kullanıldığında DBSCAN algoritması ile aynı sonuçlar ve kümeler elde edilmektedir. Bu durum Şekil 2.1.2.2.3'de gösterilmiştir. Seçilen uygun parametreler ve fonsiyonlar sonucunda bu iki algoritmanın benzer kümeleme sonuçları verdiği görülmektedir. Ancak şu da belirtilmelidir ki bu iki algoritma kare dalga fonksiyonu kullanıldığı için benzer sonuçlar üretmiştir. Dolaysıyla DENCLUE algoritmasının genelleştirilmesinde seçilen parametrelerin ve fonksiyon türlerinin de çok önemli etkileri olmaktadır[12,30,33,56].



(a) DBSCAN (b) DENCLUE

Şekil 2.1.2.2.3. DBSCAN ve DENCLUE algoritmaları sonucunda elde edilen küme yapıları

σ ve ξ parametrelerinin seçimi kümeleme sonuçlarını ve yoğunluk çekici sayısını etkilemektedir. σ_{maks} değeri, global yoğunluk fonksiyonunun yalnızca bir yoğunluk çekicisi

olduğu durumdaki minimum değerdir. σ_{min} ise, fonksiyonun N yoğunluk çekicisi olduğundaki maksimum değeridir. Eğer σ değeri σ_{min} değerine eşit ise, her bir noktanın kendisi bir küme olarak belirlenmektedir. Yoğunluk çekici σ'nın artması ile birlikte noktaları birbirleri ile birleştirmektedir. σ değeri σ_{maks} değerine eşit olduğunda tüm noktalar tek bir küme olarak kabul edilir. Dolayısıyla kullanıcı uygun kümeler elde edebilmek için σ parametresini ayarlamaktadır. Veri tabanı gürültülü verileri içermediğinde *D* veri tabanının tüm yoğunluk çekicileri önemlidir ve ξ parametresi $0 \leq \xi \leq min_{x^* \in X}\{f^D(x^*)\}$ aralığında belirlenmelidir. Ancak birçok durumda veri tabanları gürültülü veriler içermektedir $(D=D_C \cup D_N)$. Gürültü seviyesi ise $||D_N||\sqrt{2\pi\sigma^2}^d$ ile tanımlanır ve bu durumda ξ parametresi aşağıda tanımlanan aralıklar arasında seçilmelidir[12,30,33,56].

$$||D_N||\sqrt{2\pi\sigma^2}^d \leq \xi \leq min_{x^* \in X}\{f^{D_C}(x^*)\}$$

Yukarıdaki formülde yer alan D_C terimi kümeleri, D_N terimi ise gürültüleri içermektedir.
Tanımlamalar ışığında DENCLUE algoritması iki adımda gerçekleşmektedir. Birinci aşama ön kümeleme aşamasıdır. Bu aşamada, veri alanı ile ilgili bölümün bir haritası oluşturulmaktadır. Bu haritalama işlemi ise, yoğunluk fonksiyonu hesabının hızlandırılmasında kullanılmaktadır. İkinci aşama ise, asıl kümeleme aşamasıdır. Bu aşamada, algoritma yoğunluk çekicileri ve yoğunluk çekicilere karşılık gelen yoğunluk çekilen noktaları tanımlamaktadır[12,30,33,56].

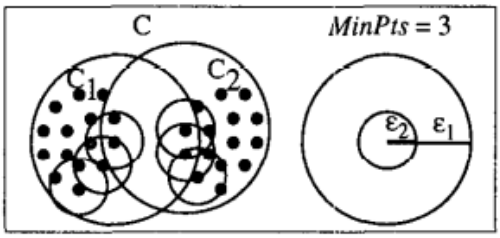
DENCLUE algoritması yüksek miktarda gürültülü verilere karşı değişmez yapıdadır. Bir başka ifadeyle gürültülü verilerden etkilenmemektedir. Ayrıca DENCLUE algoritması yüksek boyutlu veriler içinde çok başarılı sonuçlar üretmektedir. Algoritma matematiksel bir yapıya sahip olmakla birlikte diğer kümeleme algoritmasının genelleştirilmiş hali olarak bilinmektedir. Ancak DENCLUE algoritması uygulanırken başlangıç parametrelerinin seçiminin dikkatli bir şekilde yapılması gerekmektedir. Algoritmanın hesaplama karmaşıklığı ise *O(nlogn)*'dir. Burada *n,* nesne sayısını göstermektedir[12,30,33,56].

2.1.2.3. OPTICS Kümeleme Algoritması (Ordering Points to Identify Clustering Structure)

OPTICS algoritması ilk olarak Ankerst ve arkadaşları tarafından 1999 yılında SIGMOD konferansında sunulmuştur[30,57,58,59,60]. Bir önceki bölümde incelenen DBSCAN algoritması için temelde girilmesi gereken bazı parametreler vardır. Bu parametreler kullanıcılar

tarafından belirlenen $\mathcal{E}$ ve *MinPts* parametreleridir. Ancak bu parametreleri her zaman doğru olarak belirlemek zordur. Özellikle çok boyutlu verilerde parametreleri belirlemek daha güçtür. Başlangıç parametreleri yanlış olarak belirlenmesi ise kullanıcıyı hata sonuçlara götürecektir. DBSCAN algoritmasının başlangıç parametrelerin belirlenmesi önkoşulunun dezavantajlarını gidermek için OPTICS algoritması önerilmektedir.

OPTICS algoritması açıklanmadan önce sıralama kavramından bahsetmek gerekecektir. Sabit bir *MinPts* değeri ile yüksek yoğunluğa sahip kümeler, düşük yoğunluğa sahip kümeleri de içerecektir. Bu ilişkiler Şekil 2.1.2.3.1'de gösterilmiştir[30,57,58,59,60].



Şekil 2.1.2.3.1. MinPts değeri 3 olduğunda OPTICS algoritmasında oluşturulan kümeler

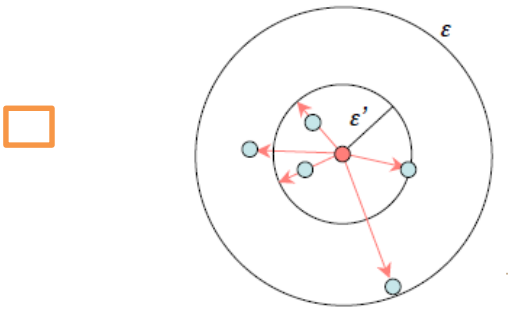
Şekil incelendiğinde $\varepsilon_2 < \varepsilon_1$ koşulu altında C_1 ve C_2 kümelerinin yoğunluğa dayalı olduğu ve C kümesinin ε_1'e dayanarak bir küme olduğu görülmektedir. Aynı zamanda C kümesi C_1 ve C_2 kümelerini de içermektedir[30,57,58,59,60].

DBSCAN algoritması farklı uzaklık parametreleri için aynı zamanda çalıştırılabilir. Ancak sabit bir sonuç elde edebilmek için kümeleri genişletirken nesneleri işleyen bir sıralama kuralına uyulması gerekmektedir[30,57,58,59,60].

OPTICS algoritması, DBSCAN algoritmasının sonsuz sayıda değer alabilen uzaklık parametresi ε_i ile çalıştırılması olarak düşünülebilir. ε_i parametresi $0 \leq \varepsilon_i \leq \mathcal{E}$ arasında değer alabilmektedir. $\mathcal{E}$ üretilen uzaklık (generating distance) olarak bilinmektedir. Sabit belirlenen *MinPts* parametreleri için $0 \leq \varepsilon_i \leq \mathcal{E}$ aralığında her nokta için $\mathcal{E}$ değerlerini bulur ve bu değerleri bir grafik üzerinde gösterir. Dolayısıyla yöntemde kümeler $\mathcal{E}$ değerlerine bağlı olarak belirlenmiş olur. OPTICS algoritmasının bu avantajına rağmen çok boyutlu veriler için bu tür grafiklerin kullanımı ve yorumu güçleşecektir. OPTICS algoritmasının DBSCAN algoritmasından tek farkı küme üyelerinin belirlenmemesidir. OPTICS algoritması küme üyelerinin belirlenmemesi yerine nesnelerin işlenme sırasını depolar ve bu bilgi küme

üyeliklerini atamak için geliştirilmiş bu yöntemde kullanılır. Bu bilgi her bir nesne için sadece iki değer içermektedir. Bu bilgiler çekirdek uzaklık ve ulaşılabilir uzaklıktır[30,57,58,59,60].

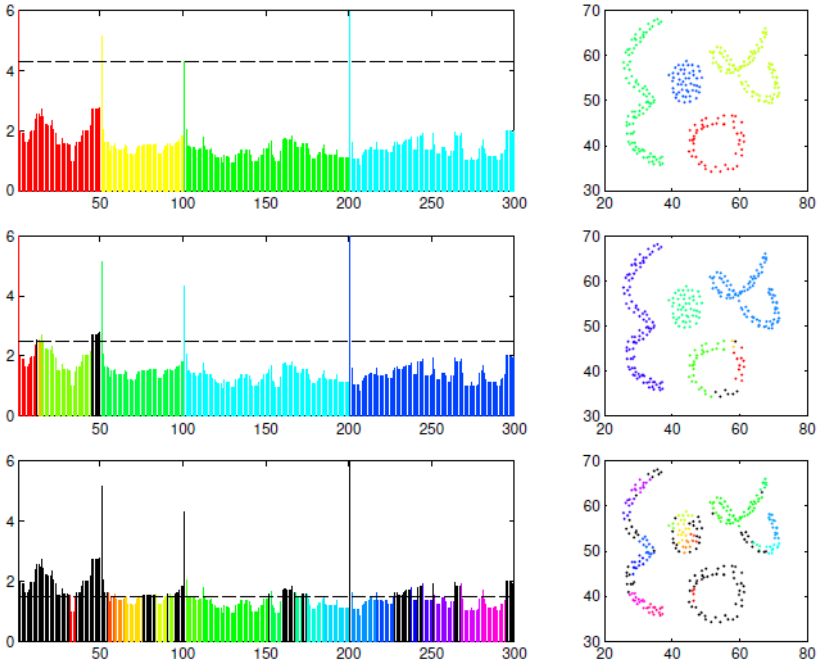
Çekirdek uzaklık (core distance), *p* çekirdek nesne olacak şekilde *p* ve $\mathcal{E}$ komşuluğundaki bir nesne arasındaki en küçük uzaklıktır ve Şekil 2.1.2.3.2'de $\mathcal{E}'$ olarak gösterilmiştir. *p* noktasının *erişilebilir uzaklığı (reachability distance)* ise *p*'nin *o* çekirdek nesnesinden yoğunluk erişilebilir olduğu en küçük uzaklıktır. Bu tanımlamalar aşağıdaki şekilde gösterilmiştir. *MinPts*'nin 3 olduğu kabul edilirse, şekilde yer alan kırmızı ok erişilebilir uzaklığı gösterirken, $\mathcal{E}$ üretilen uzaklık, $\mathcal{E}'$ ise çekirdek uzaklığı göstermektedir. Çekirdek uzaklık içerisinde yer alan noktalar için ulaşılabilir uzaklık, çekirdek uzaklık kadardır. Örneğin $\mathcal{E}=8$ olarak kabul edelim. Kırmızı noktanın çekirdek uzaklığı turuncu kutu ile belirlenen nesne ile arasındaki uzaklıktır. Kırmızı noktanın kalan diğer iki noktalara göre erişilebilirlik uzaklığı, kırmızı noktanın çekirdek uzaklığı kadardır. Ancak çekirdek uzaklık dışında kalan diğer iki noktanın erişilebilirlik uzaklığı kırmızı nokta ile Öklid uzaklıkları kadardır. Çünkü bu noktaların kırmızı noktaya kadar ki Öklid uzaklıkları çekirdek uzaklıktan büyüktür[30,57,58,59,60].



Şekil 2.1.2.3.2. OPTICS algoritmasında kullanılan parametrelerin şekilsel gösterimi

OPTICS kümeleme algoritması nesnelerin sıralamasını oluşturarak çalışmaktadır ve her bir nesnenin çekirdek uzaklığı ve erişilebilirlik uzaklığını bir önceki nesneye göre depolamaktadır. Bu bilgilere dayanarak da ulaşılabilirlik grafiği çizilmektedir. Aşağıdaki şekilde bu grafiklere ve grafiklerin oluşturduğu kümelere örnekler verilmiştir. Grafikte yer alan *X* ekseni sıralı verileri gösterirken *Y* ekseni ise *Eps* uzaklığını ifade etmektedir. Soldaki grafikler farklı eşik değerleri için erşilebilirlik grafiklerini gösterirken sağdaki şekiller OPTICS algoritması sonucunda elde edilen kümeleri ifade etmektedir. Grafikte yer alan siyah çizgiler aşırı, uç değerleri göstermektedir ve $\mathcal{E}$ değeri 6 olarak ayarlanmıştır. $\mathcal{E}$* değeri ise

sırasıyla 3, 2.5 ve 1.5 olarak alınmış ve sol tarafta yer alan kümeleme sonuçları elde edilmiştir[30,57,58,59,60].



Şekil 2.1.2.3.3. Ulaşılabilirlik grafikleri

OPTICS algoritmasının hesaplama karmaşıklığı DBSCAN algoritmasına eşittir yani *O(nlogn)* olarak ifade edilmektedir. Burada *n* nesne sayısını belirtmektedir.

2.1.2.4. Make Density Based Kümeleme Algoritması

Bu kümeleme algoritmasında kümeler oluşturulurken veri tabanındaki yoğunluk özellikleri dikkate alınmaktadır. Yöntemde kümeler ve dolayısıyla da sınıflar kolayca tanımlanabilmektedir. Çünkü sahip olunan nokta sayısına göre yoğunluk artışı meydana gelmektedir. Veri tabanındaki elementler çekirdek ve sınır nokta olmak üzere iki farklı şekilde sınıflandırılmaktadır[61].

P(x) olasılık yoğunluk fonksiyonu altında, x_1, x_2,....,x_n'ler noktaların bir örnekleridir. *P(x)* kümülatif olasılık fonksiyonu ise aşağıdaki gibi tanımlanmaktadır[61].

$$P(x)=\sum x_i \leq xp(x_i)$$

P(x), x'in tahmin fonksiyonudur. *P(x)* değeri, *x* merkezli, *h* genişliğinde bir pencere dikkate alınarak tahmin edilmektedir. *h* parametresinin genişliği yoğunluk tahmininin yayılımını ve

düzgünlüğünü göstermektedir. Eğer yayılım çok büyük ise, daha büyük ortalama bir değer elde edilir. Küçük olduğunda ise, pencerede yeteri kadar noktanın olmadığı sonucuna varılır. Pencere içerinde kalan noktalar için ($|z|\leq 1/2$),*p(x)* olasılık fonksiyonuna $1/hn$'in katkısı olacaktır. Pencere dışında kalan noktaların ($|z|>1/2$) ise olasılık fonksiyonuna katkısı sıfırdır[61].

$$K(z)=1/\sqrt{2}exp\{-z^2/2\}$$

Yukarıdaki formüldeki $z=x-x_i/2$'dir.

K(z) eşitliğine göre *x*, dağılımın ortalaması olarak *h* ise dağılımın standart sapması olarak işlem görmektedir[61].

2.1.3. Yoğunluk Dağılım Fonksiyonuna Dayalı Kümeleme Algoritması

Algoritma 2010 yılında Jianhao Tan ve arkadaşları tarafından geliştirilmiştir. Önerilen kümeleme yöntemi olasılık yoğunluk fonksiyonunu kullanmaktadır. Algoritmanın temel özelliği, matematiksel fonksiyonları kullanmasıdır. Diğer birçok algoritmanın genelleştirilmiş halidir. Çok sayıda gürültülü veri içeren veri setleri için oldukça mükemmel sonuçlar üretmektedir. Yüksek boyutlu, düzensiz şekillere sahip veri setleri için anlaşılır matematiksel tanımlamalar kullanmaktadır. Geliştirilen algoritma aşağıdaki kavramları içermektedir[62].

Noktaların Yoğunluk Dağılım Fonksiyonu

d boyutlu bir uzayda *p* ve *i* nesneleri olduğu varsayımı altında, *p* ve *i*'nin etki fonksiyonu $Density_B^i: F^d \rightarrow R_0^+$ 'dır ve aşağıdaki temel etki fonksiyonu tanımlanmaktadır[62].

$$Density_B^i(p)=Density_B(p,i)$$

Etki fonksiyonu yapay bir fonksiyon olabilir ve bazı komşularda iki nokta arasındaki uzaklıklar yardımıyla belirlenebilmektedir. Kare dalga etki fonksiyonu hesaplamak için *d(p,i)* uzaklık fonksiyonu, Öklid uzaklığı gibi simetrik ve geçişli bir uzaklık ölçüsü kullanılmaktadır ve aşağıdaki gibi hesaplanmaktadır[62].

$$Density_{square}(p,i)=\begin{cases}0, & eğer d(p,i)=0\\ 1, & diğer\ durumlarda\end{cases}$$

veya Gauss etki fonksiyonu,

$$Density_{Gauss}(p,i)=e^{-\frac{d(p,i)^2}{2\sigma^2}}$$

olarak elde edilir.

p nesnesinin ($p \in F^d$) yoğunluk fonksiyonu, tüm veri noktalarının toplam etkileri olarak tanımlanmaktadır. *K* veri nesnesi verildiğinde $D = (x_1, x_2, \ldots\ldots, x_k) \subset F^d$ olmak üzere p'nin yoğunluk fonksiyonu aşağıdaki gibi tanımlanmaktadır [62].

$$Density_B^D(p) = \sum_{i=1}^{k} Density_B^i(p)$$

ve Gauss etki fonksiyonu kullanılarak elde edilen p'nin yoğunluk fonksiyonu,

$$Density(p) = Density_B^D(p) = \sum_{i=1}^{k} e^{-\frac{d(p,i)^2}{2\sigma^2}}$$ 'dur.

d(p,i), p ile en yakın komşu noktası arasındaki Öklid uzaklığıdır. Bu değer ne kadar büyük ise noktanın yoğunluğu o kadar büyüktür[62].

Çekirdek nokta

ε komşuluğunda, en az belirlenen MinPts kadar komşu nokta içeren noktalara denilmektedir.

Merkez nokta

Merkez nokta en yoğun noktadır ve *p_core* olarak tanımlanır. *p_core* merkez nokta olarak belirlenmektedir ve en yakın komşusu aday listeye eklenir. Eğer herhangi bir q aday listesinin yoğunluk değeri, merkez noktanın yoğunluğundan ve yoğunluk eşik değerinden büyük ise aşağıdaki gibi tanımlanır[62].

$$density(p) \geq \alpha * density(p\_core)$$

Yukarıdaki durumda çekirdek nokta aday listeye eklenir. Ancak diğer durumlarda nokta sınıf etikeri olarak verilmiştir ve çekirdek listeden silinir. Her bir sınıflama uzantısı, yoğunluk çekirdek noktanın verilen oranının altında olduğunda son verilir. İşlemler kümeleme bitinceye kadar devam etmektedir[62].

2.1.4. Bölümleyici Kümeleme Algoritmaları (Partitional Clustering Algorithms)

Bölümleyici algoritmalar literatürde hiyerarşik algoritmaların dışında en çok kullanılan yöntemler arasındadır. Bölümleyici kümeleme algoritmaları, özel bir amaç fonksiyonunu optimize ederek verideki var olan gruplaşmaları keşfetmektedir. Bu işlemleri yaparken de iteratif olarak bölünmelerin kalitesini artırmaktadır. Bölümleyici algoritmalarda genellikle her

bir kümeyi temsil edecek prototip noktalar kullanıcı tarafından belirlenir. Bu yüzden aynı zamanda bu algoritmalar prototipe dayalı kümeleme algoritmaları (prototype-based clustering algorithms) olarak da bilinir. Bölümleyici yöntemlerde, başlangıç küme setlerinin verilmesi gerekmektedir. Bu kümeler ardından iteratif olarak geliştirilmektedir. Ancak hiyerarşik kümeleme yöntemlerinde bireysel veri noktalarından tek bir küme olacak şekilde başlanarak kümeler oluşturulabilmektedir[63].

Bölümleyici kümeleme algoritmalarında farklı iterasyon aşamalarında seçilen temsilci noktalar küme merkezleri gibi sanal (virtual) noktalar olabilmektedir. Bölümleyici kümeleme algoritmaları denilince ilk akla gelen *K*-ortalama yöntemidir. Dolayısıyla ilk olarak bölümleyici algoritmaların temelini oluşturan bu algoritmadan bahsedilicek ardından diğer bölümleyici algoritmalar tanıtılacaktır[63].

2.1.4.1. *K*-ortalama Kümeleme Algoritması (K-means clustering algorithm)

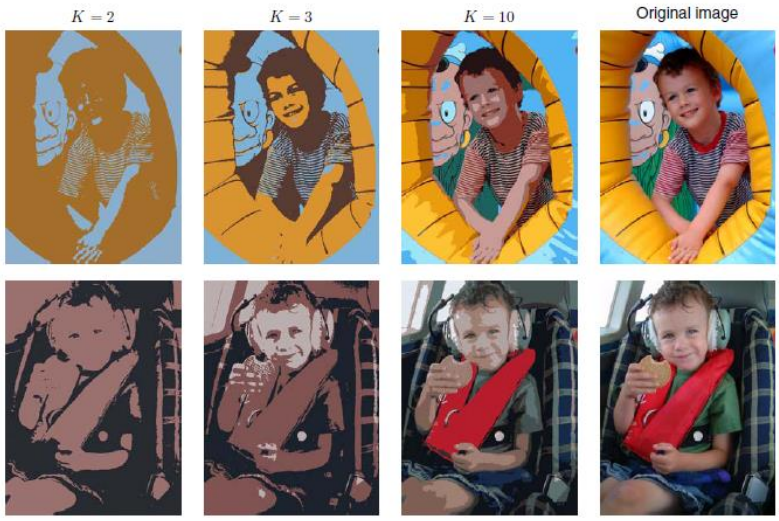
K-ortalama kümeleme yöntemi literatürde en çok kullanılan bölümleyici kümeleme yöntemlerindendir. Hiyerarşik olmayan, gözetimsiz öğrenme özelliğine sahip bir kümeleme yöntemidir. Uygulaması ve yorumlanması basit olduğundan literatürde bu yöntemden sıklıkla yararlanıldığı görülmektedir. Veri madenciliği, endüstri gibi alanlarda bu yöntem sıklıkla kullanılmaktadır. En eski kümeleme yöntemlerinden biri olan *K*-ortalama yöntemi 1957 yılında ilk kez Hugo Steinhaus'un öne sürdüğü bir fikir olmasına rağmen 1967 yılında J.B. MacQueen tarafından geliştirilmiştir[64]. Yöntemin *K*-ortalama olarak anılmasının sebebi *K*'nın başlangıçta belirlenen ve oluşturulacak küme sayısını göstermesi, ortalamalar'ın ise küme benzerliğinin ölçülmesinde kullanılan ağırlık merkezini göstermesidir. Yöntemde ağırlık merkezleri noktaların ait oldukları kümeleri temsil etmektedir.

K-ortalama algoritması *n* adet nesneyi giriş parametresi olarak belirlenen *K* adet kümeye bölmektedir. Kümeleme sonucunda ise diğer yöntemlerde de olduğu gibi küme içi benzerliğin maksimum olması amaçlanmaktadır. Başarılı bir sonuç elde edebilmek için en iyi *K* değeri belirlenmelidir. Bunun için de deneme yanılma yönteminden yararlanılmalıdır[12].

Algoritmada başlangıçta kümeyi temsil edecek *K* adet nesne belirlenmelidir. Belirlenen bu nesneler ilk küme merkezlerini oluşturmaktadır. Başlangıç küme merkezlerinin seçimi *K*-ortalama'nın sonucunu önemli oranda etkiler. Başlangıç noktalarının belirlenmesinde çeşitli teknikler vardır. Bu tekniklerden bazıları aşağıdaki gibidir[65,66].

- *K* sayısı kadar rastgele veri seçilip küme merkezleri olarak atanır
- Veriler rastgele *K* tane kümeye atanır ve küme ortalamaları alınarak başlangıç küme merkezleri belirlenir.
- En uç değerlere sahip veriler küme merkezleri olarak seçilir.
- Veri setinin merkezine en yakın noktalar başlangıç noktaları olarak seçilir.

Şekil 2.1.4.1.1'de farklı *K* değerleri için kümeleme sonucunda elde edilen fotoğraflar yer almaktadır. Görüldüğü gibi k=2 için fotoğraf net değil iken *K* artırıldığında ve *K*=10 iken fotoğraftaki netliğin arttığı belirlenmiştir.



Şekil 2.1.4.1.1. Farklı *K* değerlerine göre fotoğraf görünümleri

İkinci adımda ise her bir nesnenin *K* merkez değerlerine uzaklıkları karşılaştırılmaktadır. Ortaya çıkan her nesnenin her *K* merkezine uzaklık değerini göze alarak bu nesnelerin hangi kümelere ait olduğu belirlenir. Nesne, hangi merkeze uzaklık açısından daha yakınsa o merkezin kümesine dâhil olmaktadır. Daha sonra, her bir kümenin ortalama değeri hesaplanarak yeni küme merkezleri belirlenir ve tekrar nesne-merkez uzaklıkları incelenir[55,67]. Nesnelerin her biri bir kümeye dâhil olana kadar ve küme sınırları değişimi bitene kadar bu döngü devam eder.

Özet olarak *K* ortalama algoritmasının aşamaları adım adım olarak şöyle ifade ederiz:

1. Başlangıçta küme merkezini belirlemek için *D* veri tabanında *K* tane alt küme oluşturulacak şekilde rasgele *n* tane nesne seçilir.

2. Her nesnenin bütün özellikleri dikkate alınarak ortalaması hesaplanır. Merkez nokta kümedeki nesnelerin özelliklerinin ortalamasıdır.
3. Her nesne en yakın merkez noktanın olduğu kümeye dâhil edilir.
4. Nesnelerin kümelemesinde değişiklik olmayana kadar adım 2'ye geri dönülür [68].

N boyutlu uzayda, *N* örnekli kümelerin verildiği varsayılsın. Bu uzay $\{C_1, C_2 \ldots C_k\}$ biçiminde *K* kümeye ayrılsın. C_k kümesinin ortalama vektörü aşağıdaki formüldeki gibi hesaplanmaktadır[69].

$$M_k = \frac{1}{n_k} \sum_{i=1}^{n_k} X_{ik}$$

Formülde yer alan X_{ik}, C_k kümesine ait olan *i.* örnektir. Küme içi değişme ise C_k ile merkezi arasındaki Öklid uzaklıklarının toplanması ile elde edilmektedir. Uzaklık ölçüsü olarak city block (Manhattan), Chebychev uzaklık formülleri de kullanılabilir. City block yöntemi, sıradışı noktaların etkisinin azaltılmak istendiği durumlarda kullanılmaktadır. Chebychev yöntemi ise iki nesne arasındaki maksimum uzaklığa bağlı olduğundan bu yöntem kullanıldığında farklı komşuluklar elde edilecektir. Ayrıca küme içi değişme karesel hata olarak da bilinmektedir. C_k kümesi için karesel hata aşağıdaki formül yardımıyla hesaplanmaktadır[69].

$$e_i^2 = \sum_{i=1}^{n_k} (x_{ik} - M_k)^2$$

Bu işlemler bütün kümeler yapıldığında toplanılırsa toplam kare hata değeri elde edilmektedir ve aşağıdaki şekilde hesaplanmaktadır.

$$E_k^2 = \sum_{k=1}^{K} e_k^2$$

Yukarıdaki formülden de görüldüğü üzere bütün kümeler için karesel hata değeri E_k^2 ifadesi ile gösterilmektedir. Karesel hatayı elde etmekteki amaç, E_k^2 minimize eden *K* kümeyi bulmaktır. Dolayısıyla da bu değer bir önceki adımdaki elde edilsen karesel hata değerine göre daha küçük olması beklenmektedir[69].

K-ortalama yönteminin performansını etkileyen en önemli faktörler başlangıç merkezlerinin seçimi ve *K* küme sayısının tahmin edilmesidir. Literatürde bu faktörlerin çözümlerine ilişkin çalışmalar yer almaktadır. Başlangıç küme seçimi olarak MacQueen tarafından *K* kümenin

rasgele seçilmesi önerilmektedir. Ancak kümeleme algoritmasının performansını etkileyen başka merkez seçimi yöntemleri de vardır. Bunlar aşağıdaki gibidir[70-74].

1. *Hartigan ve Wong*: En yakın komşu yoğunluğu konsepti kullanılarak, bu yöntemde çok iyi bir şekilde ayrılmış olan noktalar ve çok boyutlu alan çevresi içerinde büyük sayıda noktalar içerenler başlangıç noktası olarak iyi temsilciler olarak kabul edilmektedir.
2. *Milligan:* Dendogram yardımıyla oluşturulan toplayıcı hiyerarşik kümeleme yöntemlerinin sonuçlarından yararlanarak, Ward yönteminden elde edilen sonuçları kullanmaktadır. Ward yöntemi, iki küme arasındaki uzaklıkları değerlendirmek için hata kareler toplamı yardımıyla başlangıç merkezlerini belirlemektedir.
3. *Bradley ve Fayyad:* Veriden rasgele olarak alt örnekler seçilir ve rasgele çekirdekler kullanılarak bu alt örneklere *K*-ortalama kümeleme yöntemi uygulanır. Bu alt örneklerin her birinden merkezler toplanır ve bu merkezleri içeren yeni bir veri seti oluşturulur. Yeni veri seti, bu merkezleri başlangıç çekirdekleri olarak kullanarak kümeleme işlemlerini gerçekleştirmektedir. Elde edilen minimum SSE en iyi çekirdek seçimin yapıldığını garanti etmektedir.
4. *K-ortalama++: Bu* algoritma *K*-ortalama kümeleme için başlangıç merkez seçimini dikkatli bir şekilde yapmaktadır. Algoritma, başlangıç merkezinin rasgele seçildiği basit bir olasılığa dayalı yaklaşımı kullanmaktadır. Bir sonraki seçilen merkez, var olan seçilmiş merkeze en uzak olandır. Bu seçim kriteri ağırlıklanıdırılmış olasılık skoruna bağlıdır. Seçim işlemleri *K* tane merkez elde edinceye kadar devam etmektedir. Ardından *K*-ortalama kümeleme yöntemi bu merkezleri kullanarak kümeleme işlemlerini gerçekleştirmektedir.

K-ortalama yönteminde ikinci önemli faktör küme sayılarının belirlenmesidir. Bu amaçla kullanılan yöntemler aşağıda verilmiştir[75-83].

Calinski-Harabasz İndeksi: Bu indeks aşağıdaki gibi ifade edilmektedir.

$$CH(K)=(B(K)/((K-1)))/(W(K)/(N-K))$$

Formülde yer alan *N*, veri noktalarının sayısını ifade etmektedir. Küme sayısına bu eşitlik maksimum yapılarak karar verilmektedir. *B(K)* ve *W(K)*, değerleri ise sırasıyla kümeler arası ve küme içi kareler toplamıdır.

Gap istatistiği: Bu yöntemde, aynı değişen değerlere sahip *B* farklı veri seti, orijinal veri olarak üretilir. Küme içi kareler toplamı her bir farklı sayıda kümeler için hesaplanır.

$W_b^*(K)$, b. tek düze veri seti için küme içi kareler toplamını göstermektedir.

$$Gap(K)=\frac{1}{B} * \sum_b log(W_b^*(K)) - \log(W(K))$$

Seçilen küme sayısı *K*'nın en küçük değeridir ve aşağıdaki eşitliği sağlamaktadır.

$Gap(K) \geq Gap(K+1) - s_{k+1}$

s_{k+1} terimi, $log(W_b^*(K))$'nin standart sapmasının tahminidir.

Akaike Bilgi Kriteri (AIC): AIC, *K* değerini tahmin ederken log-olabilirlik ve minimum tanımlayıcı uzaklık (minimum description length (MDL)) dikkate alınarak geliştirilmiş bir kriterdir. *M* veri setinin boyutunu göstermektedir. SSE, *K* kullanılarak elde edilen hata kareler ortalamasını ifade etmektedir. *K*-ortalama yöntemi modifiye edilmiş aşağıdaki AIC kriterini kullanmaktadır.

$$KOrt_{AIC}: K = argmin_K[SSE(K) + 2MK]$$

Bayesian Bilgi Kriteri (BIC): BIC, bayes posterior olasılığına bir transformasyon uygulayarak asimptotik bir yaklaşım kullanmaktadır. AIC'ye benzer olarak, hesaplamalar olabilirliğin logaritmasına bağlıdır. *N*, nokta sayılarını ifade etmektedir. BIC fonksiyonunun minimum yapan *K* değeri, *K*-ortalama kümeleme yöntemi uygulanırken başlangıç parametresi olarak seçilmektedir.

$$BIC=\frac{-2*\ln(L)}{N} + \frac{K*\ln(N)}{N} = \frac{1}{N} * \ln(\frac{N^K}{L^2})$$

Duda ve Hart Yöntemi: Hiyerarşik kümeleme sürecinde yer alan dendogramdaki en uygun kesim noktasını içeren bir tahmin yöntemidir. Dendogramı kesme yöntemleri ise şöyledir.

- Eşik değeri kullanıcı tarafından belirlenen benzerlik seviyesinde kesilir.
- En başarılı birleştirmeler arasında uzaklığın en büyük olduğu yerde dendogram kesilir.
- Birleştirme sonucunda küme yoğunluğu belirlenen eşik değerden düşük olduğu durumunda süreç durdurulur.

Silhouette Katsayısı: Bu katsayı, hem küme içi hem de kümeler arası uzaklıklar dikkate alınarak hesaplanmaktadır. Verilen bir x_i noktası için, ilk olarak aynı kümedeki yüm

noktaların ortalama uzaklıkları hesaplanır. Bu değer a_i'dir. Ardından x_i'yi içermeyen her bir küme için, x_i'den ortalama uzaklıklar bulunur. Bu değer ise b_i'dir. Bu iki değer (a_i ve b_i) kullanılarak bir noktanın Silhouette katsayısı hesaplanabilir. Veri setindeki tüm Silhouette katsayılarının ortalaması, noktaların ortalama Silhouettelerinin genişliğidir. Kümeleme kalitesinin değerlendirilmesi için, tüm noktaların ortalama Silhouette katsayıları hesaplanmaktadır.

$$S=\frac{\sum_{i=1}^{N}\frac{b_i-a_i}{\max(a_i,b_i)}}{N}$$

Newman ve Girvan Yöntemi: Bu yöntemde dendogram grafik olarak incelenir ve kenarlar arasındaki benzememezlik ölçüsü olarak kullanılan betweeness skoru önerilmektedir. Süreçi grafikteki tüm kenarların betweeness skorunun hesaplanması ile başlar. Ardından en büyük skora sahip kenar çıkarılır. Bağlantılı bileşenlerin son seti elde edilinceye kadar kalan kenalarların betweeness skoru yeniden hesaplanır. Bu süreçte elde edilen setin önem düzeyi en iyi *K* tahmini olarak verilir.

ISODATA: Bu yöntem, en yakın merkez yöntemine dayanarak verileri kümelemeyi önermektedir. İlk olarak kümeleri elde etmek için *K*-ortalama algoritması çalıştırılır. Ardından eşik değerden düşük olan uzaklıklı kümeler birleştirilir. Benzer olarak, bir küme içi standart sapma değeri belirlenen eşik değeri aşar ise, o küme bölünür.

K-ortalama yönteminin uygulama adımlarını göstermek amacıyla aşağıda basit bir örnek verilmiştir. Dört noktadan ölçülen iki değişkene ait veriler Tablo 2.1.4.1.1' de verilmiştir.

Tablo 2.1.4.1.1. Dört farklı nokta için ölçülen iki değişkene ait veriler

	Değişken 1	Değişken 2
X_1	6	4
X_2	5	3
X_3	12	2
X_4	10	8

Küme sayısını *K=2* olarak belirlensin. İlk iki gözlem değeri rasgele olarak küme 1'e, son iki gözlem değeri ise küme 2'ye atansın. Yani,

$$C_1=\{X_1, X_2\} \ , \ C_2=\{X_3, X_4\}$$

Bu kümelere göre küme merkezleri aşağıdaki şekilde hesaplanmaktadır.

$$M_1=\{\frac{6+5}{2},\frac{4+3}{2}\}=\{5.5,3.5\}$$

$$M_2=\{\frac{12+10}{2},\frac{2+8}{2}\}=\{11,5\}$$

Her iki küme için küme içi değişmeler ise aşağıdaki gibi hesaplanmaktadır.

$$e_1^2=[(6-5.5)^2+(4-3.5)^2]+[(5-5.5)^2+(3-3.5)^2]=1$$

$$e_2^2=[(12-11)^2+(2-5)^2]+[(10-11)^2+(8-5)^2]=20$$

Böylece toplam kare-hata aşağıdaki gibi olacaktır.

$$E^2=e_1^2+e_2^2=1+20=21$$

Her iki merkezin tüm noktalar ile uzaklıkları aşağıdaki şekilde hesaplanmaktadır.

İlk olarak birinci nokta (6,4)'ün M_1 ve M_2'ye uzaklıkları hesaplanacaktır.

$d(M_1,X_1)=\sqrt{(5.5-6)^2+(3.5-4)^2}=0{,}707$

$d(M_2,X_1)=\sqrt{(11-6)^2+(5-4)^2}=5{,}099$

İkinci nokta olan X_2=(5,3)'ün M_1 ve M_2'ye uzaklıkları;

$d(M_1,X_2)=\sqrt{(5.5-5)^2+(3.5-3)^2}=0{,}707$

$d(M_2,X_2)=\sqrt{(11-5)^2+(5-3)^2}=6{,}324$

Üçüncü nokta olan X_3=(12,2)'ün M_1 ve M_2'ye uzaklıkları;

$d(M_1,X_3)=\sqrt{(5.5-12)^2+(3.5-2)^2}=6{,}671$

$d(M_2,X_3)=\sqrt{(11-12)^2+(5-2)^2}=3{,}162$

Dördüncü nokta olan X_4=(10,8)'ün M_1 ve M_2'ye uzaklıkları;

$d(M_1, X_4)$=$\sqrt{(5.5-10)^2+(3.5-8)^2}$=6,364

$d(M_2, X_4)$=$\sqrt{(11-10)^2+(5-8)^2}$=3,162

Noktaların her iki merkeze olan uzaklıkları aşağıdaki tabloda topluca verilmiştir.

Tablo 2.1.4.1.2. Noktaların merkezlere olan uzaklığı

	M_1'den uzaklık	M_2'den uzaklık
X_1	0,707*	5,099
X_2	0,707*	6,324
X_3	6,671	3,162*
X_4	6,364	3,162*

X_1 ve X_2 gözlem değerleri M_1 merkezine, X_3 ve X_4 gözlem değerleri ise M_2 merkezine daha yakın olduğu için sonuç kümeler;

C_1=$\{X_1, X_2\}$

C_2=$\{X_3, X_4\}$'dür.

Eğer kümelerde önceki adımlara göre herhangi bir değişme olsaydı örneğin X_1 noktası M_2 merkezine daha yakın olsaydı C_1=$\{X_1\}$,C_2=$\{X_2, X_3, X_4\}$ olacak şekilde küme merkezleri yeniden hesaplanarak hata kareleri bulunur ve önceki adıma göre herhangi bir değişme olmadığı durumda iterasyonlara son verilerek sonuç kümesi bulunur.

K-ortalama kümeleme yöntemi sayısal veriler için kullanılmaktadır. Gürültülü ve uç değerlerden aşırı derecede etkilenmektedir. Hesaplama karmaşıklığı *O(tkn)*'dir. Burada *t* iterasyon sayısını, *K* küme sayısını, *n* ise nesne sayısını göstermektedir. Karmaşıklığı diğer yöntemlere göre azdır ve uygulaması kolaydır. *K*-ortalama kümeleme yöntemi büyük veri setleri için başarılı sonuçlar üretebilmektedir. *K*-ortalama algoritması genel olarak küresel kümeleri bulmada daha başarılı bir yöntemdir. Ancak yöntemin dezavantajı küme sayısını belirlemekteki zorluktur. Bunun yanı sıra kümeleme sonucunda bazı kümelerde eleman olmayabilir yani kümeleme sonucunda boş küme oluşabilmektedir ki bu durumda uygun optimizasyon yöntemleri kullanılmalıdır. Kümelerin oluşturmada rol oynayan özelliklerin hangisinin daha çok kümenin oluşmasında etkisi olduğu ise bilinmemektedir[12,84] .

2.1.4.2. Geliştirilmiş *K*-ortalama Kümeleme Algoritması

Klasik *K*-ortalama yöntemi kümeleme algoritması olarak uygulamalarda sıklıkla kullanılmaktadır. Ancak yöntemin avantajlarının yanında dezavantajlarıda mevcuttur. *K* değeri, *K*-ortalama kümeleme algoritması için oldukça önemli bir faktördür. Farklı başlangıç değerinin seçimi farklı kümelerin elde edilmesine neden olacaktır. Ayrıca *K*-ortalama yöntemi başlangıç küme merkezlerinin seçiminde de oldukça fazla etkilenmektedir. Bunun yanı sıra sonuç kümelerini etkileyen gürültülü verilere karşı da hassas bir algoritmadır. *K*-ortalama yöntemi ayrıca düzensiz şekillere sahip kümeleri bulmakta doğru sonuçlar vermemektedir. Yöntemin her bir iterasyonunda nesneleri ait oldukları küme merkezlerine göre ayarlamak gerekmektedir. Geliştirilen *K*-ortalama kümeleme algoritması, başlangıç küme seçimi için en uygun merkezleri belirlemek için önerilmiştir. Aynı zamanda küme merkezleri ve küme merkez mesafesindeki veri noktalarını en uygun şekilde hesaplamaktadır[85].

Geliştirilen *K*-ortalama yönteminin işlem adımlarına geçmeden önce bazı kavramların tanımlarının yapılması gerekmektedir.

Veri noktaları ve küme merkezi arasındaki uzaklık: Veri noktası x_i ve küme merkezi k_j arasındaki uzaklık aşağıdaki gibi tanımlanmaktadır[85].

$$d_{j,i} = \sqrt{(x_{i1} - k_{j1})^2 + (x_{i2} - k_{j2})^2 + \cdots + (x_{iw} - k_{jw})^2}$$

w, x_i veri noktasının değişkenlerinin sayısıdır.

Yoğunluk parametresi Γ: Yarıçap içerisindeki veri noktalarının sayısıdır. γ, yarıçap olmak üzere x_i ne kadar yoğun ise, yoğunluk parametresi değeri de o kadar büyüktür[85].

Veri noktalarının çekirdeği: Veri noktası, γ komşuluğında belirlenen minimum sayıdaki veri noktalarını içeriyorsa, o nokta çekirdek veri noktası olarak adlandırılmaktadır[85].

Küme merkezi: Klasik kümeleme yönteminden farklı olarak geliştirilmiş olan bu yöntemde küme merkezlerinden veri noktalarının uzaklıklarına göre bir ağırlıklandırma yapılmaktadır. Küme merkezine yakın olan noktalar daha büyük ağırlığa sahip iken, küme merkezinden uzak noktalar daha az ağırlığa sahiptir. Küme merkezinin formülü aşağıdaki gibidir[85].

$$k = \frac{d_{jh}}{D} x_{J1} + \frac{d_{j(h-1)}}{D} x_{J2} + \cdots + \frac{d_{j2}}{D} x_{J_{(h-2)}} + \frac{d_{j1}}{D} x_{jh}$$

Formülde yer alan *j*, küme *j*'yi ifade ederken, *h* kümedeki veri noktalarının sayısı, d_{jh} ise *c* kümesine ait olan *h.* veri noktası ile küme merkezi arasındaki uzaklıktır ve aşağıdaki kısıtlar sağlanmalıdır [85].

$$d_{j1} \leq d_{j2} \leq \cdots \leq d_{jh}, \frac{d_{j1}}{D} + \frac{d_{j2}}{D} + \cdots + \frac{d_{jh}}{D} = 1 \ (*)$$

Veri noktaları ve küme merkezi arasındaki Öklid uzaklığı: Veri noktası ve küme merkezi arasındaki uzaklık noktaların hangi kümeye ait olduğunu belirlemektedir ve aşağıdaki formül yardımıyla hesaplanmaktadır[85].

$$d_{ji} = (1 - \frac{\sigma_j}{\sigma}) d_{ji} \ (**)$$

Yukarıdaki formülde yer alan *j*; c_j kümesini, *i*;x_i veri noktasını, d_{ji} ise veri noktası ile küme merkezi arasındaki Öklid uzaklığını ifade etmektedir. σ_j, c_j kümesinin hata kareleri iken σ, *K* tane *c* kümelerinin hata kareleri toplamıdır[85].

Yukarıdaki tanımlamaların ardından geliştirilmiş *K*-ortalama kümeleme algoritmasının işlem adımları aşağıdaki gibi özetlenmektedir.
Yöntemde $2\gamma = 1/100$ ve yoğunluk eşik değeri $\tau = n/100$ olarak belirlendiğinde,

Veri seti (*x*), *n* veri nesnesi ve *K* küme sayısı olmak üzere;

1. *Adım:* Küme merkezi başlatılır.

- *X* veri setinden x_i veri noktası seçilir ve istatistik olarak tanımlananlar ayarlanır ve x_i ile *X* veri setindeki diğer bir veri noktası arasındaki uzaklık hesaplanır. Eğer eşik uzaklığı karşılarsa, veri noktası istatistik olarak tanımlanır ve x_i'nin yoğunluk değeri bir artıtılır.
- İstatistik olarak tanımlanmayan bir veri noktası seçilir ve yoğunluk değeri hesaplanır. Veri setindeki tüm noktalar istatistik olarak tanımlanana kadar üst maddede tanımlanan adım tekrar edilir.
- Eşik değerden daha büyük olan yoğunluk değerine sahip veri noktası seçilir ve D setinde karşılık gelen yüksek yoğunluğa eklenir.

- D setinde yüksek yoğunluğuna karşılık gelen veri noktası filtrelenir ve başlangıç küme merkezi setine eklenir. Takip eden *k-1* veri noktası, *k* başlangıç küme merkezleri arasında en büyük uzaklığa sahip olan noktadır.

2. *Adım: X* veri setindeki *n* veri noktası kendine en yakın kümeye atanır.
3. *Adım:* (*) formülü kullanılarak her bir *K* küme merkezi ayarlanır.
4. *Adım:* (**) formülü kullanılarak her bir küme merkezinden farklı veri noktalarının uzaklıkları hesaplanır ve *n* veri noktası karşılık gelen kümelere yeniden dağıtılır.
5. *Adım:* (*) formülü kullanılarak her bir *K* küme merkezi ayarlanır.
6. *Adım:* Kriter fonksiyonu kullanılarak yakınsamanın sağlanıp sağlanmadığına karar verilir ve eğer yakınsamış ise devam edilir, diğer durumda 4. adıma tekrar dönülür.

Geliştirilen *K*-ortalama kümeleme algoritması kümeleme sürecinde durağan sonuçlar üretmekte kalmayıp aynı zamanda gürültülü verilerin etkilerini azaltarak kümeleme sonuçlarının daha doğru ve etkili bir şekilde elde edilmesine olanak sağlamaktadır[85].

2.1.4.3. *K*-Medoids Kümeleme Algoritması

Yöntem ilk olarak 1987 yılında Kaufman ve Rousseeuw tarafından *K*-ortalama algoritmasının dezavantajlarını gidermek için geliştirilmiştir[66,86,87]. *K*-ortalama yönteminde veri setinde çok büyük değeri alan nesne, kümenin ortalamasını ve merkezini aşırı derecede etkileyebilmektedir. Ancak *K*-medoids algoritmasında ortalama yerine temsili noktalar kullanıldığı için bu dezavantaj ortadan kaldırılmıştır.

Medoid kelimesi küme içinde en merkeze yerleştirilmiş olan nesne anlamına gelmektedir. *K*-medoid yöntemi veri setindeki nesneler ile temsili nokta arasındaki benzersizliklerin toplamını minimum yapmayı amaçlamaktadır. Bu hata kriteri aşağıdaki gibi tanımlanmaktadır.

$$E=\sum_{j=1}^{k}\sum_{p\in C_j}|p-o_j|$$

Formülde yer alan *E*, veri setindeki tüm nesnelerin toplam mutlak hata değeri, p, C_j kümesindeki belli bir nesneyi temsil eden bir nokta ve o_j, C_j'nin temsili nesnesidir[88].

Kümeleme literatüründe temsilci nesnelere merkeztipler (centrotypes) denilmektedir[66,79].

Temsilci nesne seçmenin bazı temel nedenleri ve amaçları vardır. Öncelikle bir kümeyi tanımlamak için temsilci nesnelere ihtiyaç duyulmaktadır. Temsilci noktalar bu kümelerin özelliklerini en iyi şekilde taşımaktadır.

K-medoids algoritması kümeyi temsil edecek noktayı bulmak için küme elemanlarının ortalamasını almak yerine kümenin en merkez noktasındaki elemanı yeni küme merkezi olarak almaktadır[89].

Eğer bulunan bu kümenin kullanımı uygun ise daha detaylı yapılacak araştırmalar için örnek büyük veri sayısı içeren çalışmalar için de bu temsilci noktalar kullanılabilmektedir. Böylece zamandan, maliyetten ve uygulanabilirliği açısından avantaj sağlanır.

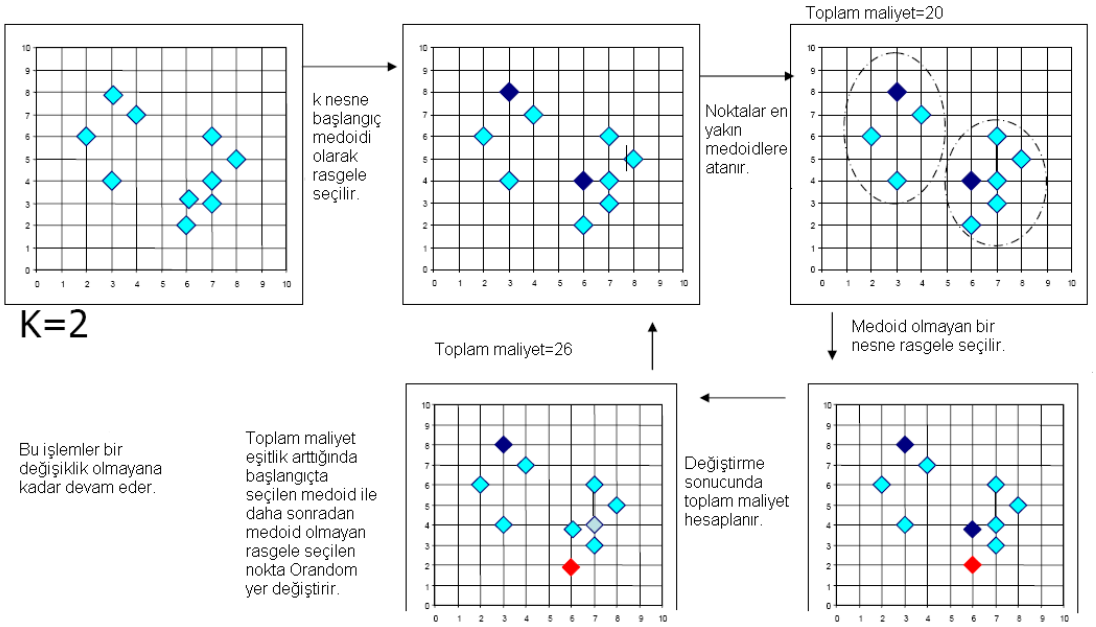
K-medoids algoritmasında birinci aşama yapılandırma aşamasıdır. Bu aşama kümelemenin başlangıç aşamasıdır, *k* adet temsilci nesne seçilene kadar devam eder. Başlangıç merkezleri rastgele atanabileceği gibi çeşitli işlemler sonucu da belirlenebilir. Algoritmanın ikinci aşaması değiştirme (Swap) aşamasıdır. Bu aşama temsilci nesneleri geliştirerek kümeleme işleminin verimini arttırmak için uygulanır. Her bir (*i*,*h*) çifti için hesaplama yapılır. *i* seçilmiş, *h* ise seçilmemiş bir nesnedir. Değişim ihtimallerinin kümelemeye nasıl bir etkisi olduğu incelenerek her bir kombinasyon için kümeleme kalitesi hesaplanır[66,79].

Algoritma girdi parametresi olarak *k* belirlenir. Burada *k* parametresi, *n* nesneleri arasında bölünen kümelerin sayısını göstermektedir. Aşağıdaki adımlarda medoid veya merkezi nesnelere dayalı bölünmeli kümeleme yöntemi *K*-medoids'in işlem basamakları gösterilmektedir[90].

- Girdi parametresi olarak *K* ve *D* belirlenmelidir. Burada *K* küme sayısını, *D* nesneleri içeren veri setini göstermektedir.
- Çıktı parametresi olarak *K* adet küme setleri oluşur. Bu küme setleri tüm nesnelerin kendi en yakın medoidlerinin benzemezliklerinin toplamını minimuma indirmelidir.
- Başlangıçta *D*'de keyfi olarak *t* adet temsili nesne seçilir.
- Daha sonra her bir nesne en yakın medoidi olan kümeye atanır. Medoid olmayan bir nesne rasgele olarak seçilir ve O_{random} olarak gösterilir. Ardından O_j yerine değiştirilen (swap) O_{random} 'a göre toplam noktalar hesaplanır.
- Eğer $S<0$ ise, *h* medoid'in yeni seti O_j yerine O_{random} olacaktır.

Yukarıdaki işlemler değişim olmayan kadar tekrar etmektedir. O_{random} değerinde herhangi bir değişimin olmadığı durumlarda algoritma durdurulur ve kümeler belirlenir[90].

Şekil 2.1.4.3.1.1'de bu işlem adımları şekiller ile gösterilmiştir.



Şekil 2.1.4.3.1.1. *K*-medoid kümeleme yöntemi

K-medoids kümeleme algoritması uç verilerin kümeleme üzerindeki etkisini azaltmaktadır. Yani veri setinde uç değerler ve gürültülü veriler olduğunda *K*-ortalama yöntemi yerine *K*-medoids algoritması kullanılmalıdır. Ancak yöntemin dezavantajı yine *K*-ortalama yönteminde olduğu gibi küme sayısının başlangıçta belirlenmesidir. Bunun için de en uygun küme sayısının belirlemek için birden fazla deneme gerçekleştirmek gerekmektedir. Yöntemin hesaplama karmaşıklığı $O(k(n-k)^2)$'dir[90].

Bunun yanı sıra *K*-medoids yöntemi *K*-ortalama yöntemine göre daha maliyetli bir algoritmadır. Ayrıca *K*-medoids algoritması da farklıbüyüklüklerde ve küresel olmayan kümeleri doğru tespit edemez[66].

2.1.4.4. Kernel *K*-ortalama Kümeleme Algoritması (kernel *K*-means clustering)

Kernel *K*-ortalama kümeleme yöntemi, klasik *K*-ortalama yönteminin lineer olmayan bir uzantısıdır. İlk olarak 2002 yılında Girolami tarafından geliştirilmiş iteratif bir yöntemdir. Bu yöntemde ilk olarak giriş uzayındaki noktaları çok boyutlu uzaya, $\boldsymbol{\phi}(.)$ lineer olmayan transformasyonu kullanarak haritalandırır ve böylelikle küme hatası minimize edilir. Uzayda veri noktası ile küme merkezi arasındaki uzaklık kernel fonksiyonu kullanılarak

hesaplanmaktadır. $\boldsymbol{\phi}(X)$. $\boldsymbol{\phi}(Y)$ noktası uzayda *X* ve *Y* noktası arasındaki resmi (image) üretmektedir. Bu nokta *K(X.Y)* fonksiyonu kullanılarak hesaplanabilir. *K(X.Y)* fonksiyonu *K:DXD→R* olmak üzere bir kernel fonksiyonudur. Aşağıda bazı standart kernel fonksiyonları yer almaktadır[55,91].

p derecesinde Polynomial kernel: K(X,Y)=(X*Y+*c*)^p, *p*: pozitif tamsayı ve *c*∈*R*

Gaussian (RBF) kernel: K(X.Y)=exp($-\frac{||X-Y||^2}{2\sigma^2}$*),* $\boldsymbol{\sigma}$∈*R* Gaussian kernel parametresidir.

*Sigmoidal kernel: K(X.Y)=tanh(α(X*Y)+b), a ve b ∈R*

*Lineer kernel: K(X.Y)=X*Y*

İteratif kümeleme sürecinde X_i ve X_j veri noktaları için $K(X_i, X_j)$'yi hesaplamaya gereksinim vardır. Böylece kernel matris olarak adlandırılan $H=[K_{ij}]_{n*n}$ matrisi elde edilir[91].

n veri seti genişliğidir. Kernel matrisi önceden hesaplanarak kaydedilir. Böylece zaman ve yer karmaşıklığı $O(n^2)$'dir. Bu kernel *K*-ortalama yönteminin dezavantajıdır. Çünkü bu karmaşıklık yöntemin çok büyük veri setleri için kullanımının uygun sonuçlar üretemeyeceğini göstermektedir. Ayrıca yöntemin başka bir dezavantajı da başlangıçta küme sayısını belirleme ihtiyaç duyulması ve çekirdek noktaların ve kernel fonksiyonlarının kümeleme sonucunu etkilemesidir[55,91].

Yöntemin işleyiş adımlarına bakabilmek için $D=\{X_1, X_2, ..., X_n\}$ olacak şekilde bir veri seti düşünülsün. *K* gerekli olan küme sayısı ve $M^{(0)}$ başlangıç çekirdek noktası ve $\Pi^{(0)}$ başlangıç bölme sonucunu göstermektedir. $M^{(0)}=\{m_1^{(0)}, m_2^{(0)}, ..., m_k^{(0)}\}$ giriş uzayındaki noktalardır. Kernel *K*-ortalama yöntemi *D*, *K* ve $\Pi^{(0)}$'ı giriş parametresi olarak alır, veri seti için son bölme sonuçlarını Π_D çıkış parametresi olarak verir[55,91].

Kriter fonksiyonunu minimize edebilmek için kullanılan fonksiyon aşağıdaki gibi verilmektedir.

$$J=\sum_{j=1}^{k}\sum_{X_i\in C_j}\|\phi(X_i-\mu_j\|^2$$

Yukarıdaki formülde yer alan μ_j değeri C_j kümesinin ortalamasını göstermektedir ve aşağıdaki formül yardımıyla hesaplanmaktadır[55,91].

$$\mu_j = \sum_{X_i \in C_j} \frac{\phi(X_i)}{|C_j|}$$

Taşınılmış uzayda $\boldsymbol{\phi}(X_i)$ ve $\boldsymbol{\phi}(X_j)$ veri noktaları arasındaki uzaklık,

$$\|\phi(X_i) - \phi(X_j)\|^2 = \phi^2(X_i) - 2\boldsymbol{\phi}(X_i)\boldsymbol{\phi}(X_j) + \phi^2(X_j)$$

$$= \mathrm{K}(X_i, X_i) - 2\mathrm{K}(X_i, X_j) + \mathrm{K}(X_j, X_j)\text{'dir.}$$

$\boldsymbol{\phi}(.)$ transformasyonu bilinmeden $\|\phi(X_i) - \mu_j\|^2$ aşağıdaki formül yardımıyla hesaplanmaktadır[55,91].

$$\|\phi(X_i) - \mu_j\|^2 = \|\phi(X_i) - \sum_{X_i \in C_j} \frac{\phi(X_i)}{|C_j|}\|^2$$

$$= \phi(X_i) * \phi(X_i) - \mathrm{F}(X_i, C_j) + \mathrm{G}(C_j),$$

Yukarıdaki formülde yer alan $F(X_i, C_j)$ ve $G(C_j)$ eşitlikleri ise aşağıdaki gibi hesaplanmaktadır.

$$F(X_i, C_j) = -\frac{2}{|C_j|} \sum_{X_l \in C_j} \phi(X_l) * \phi(X_i)$$

$$G(C_j) = \frac{1}{|C_j|^2} \sum_{X_l \in C_j} \sum_{X_s \in C_j} \phi(X_l) * \phi(X_s)$$

Bu iteratif yöntemin işleyişi aşağıdaki algoritmada özetlenmiştir[55,91].

1. Her bir C_j kümesi için $|C_j|$ ve $\mathrm{G}(C_j)$ bulunur.
2. Her bir X_i ve C_j kümesi için $\mathrm{F}(X_i, C_j)$ hesaplanır.
3. $\|\phi(X_i) - \mu_j\|^2$ eşitliği hesaplanır ve X_i noktası en yakın merkeze atanır.
4. j=1'den K' ya kadar $\mu_j = \sum_{X_i \in C_j} \frac{\phi(X_i)}{|C_j|}$ formülü kullanılarak μ_j güncellenir.
5. Kümelerin sonuçları birbirine yakınsayana kadar 1. adımdan 4. adıma kadar işlemler tekrar edilir.

Son bölme sonuçları $\Pi_D = \{C_1, C_2, \ldots, C_k\}$ olarak elde edilir.

Kernel *K*-ortalama yönteminin dezavantajlarını gidermek amacıyla Zhang ve Rudnicky 2002 yılında büyük veri setleri için uzay (space) karmaşıklıklarını ele alan bloğa dayalı (block

based) bir yaklaşım önermişlerdir. Bu yaklaşımda kernel matrisi olan *H*, önceden hesaplanır ve bu bilgi iteratif sürece başlamadan önce ikincil hafızada depo edilir. Daha sonra kernel matrisi *H*, bloklara ayrılır. Her bir bloğun genişliğine *I/O* kapasitesine ve ulaşılabilir ana hafıza genişliğine bakılarak karar verilir. Böylece her bir iterasyondaki *I/O* işlem sayısı blok sayısına eşit olur, ancak birçok sayıda blok var ise bu işlemin maliyeti yüksek olacaktır. Fakat bu geliştirilen yöntem (bloğa dayalı) uzay karmaşıklığı sorunlarını gidermektedir. Bloğa dayalı bu kümeleme yönteminde de tüm kernel matrisini hesaplamaya gereksinim vardır. Eğer zaman karmaşıklığı azaltılacak ise büyük veri setleri için bu yöntemi kullanmak uygun bir seçim olmayacaktır[91,92].

Radha Chitta ve arkadaşları 2011 yılında yaklaşık (approximate) kernel *K*-ortalama olarak adlandırılan randomize yaklaşımlı bir algoritma önermektedirler. Bu yaklaşım ile klasik kernel *K*-ortalama yönteminin zaman ve uzay karmaşıklığı azaltılmak amaçlanmıştır. Yaklaşık kernel *K*-ortalama yönteminde rasgele olacak şekilde q genişliğinde örnekler seçilir ve bu örnekler *B* harfi ile sembolize edilmektedir. *D*'deki veri noktaları ve q örnek noktaları arasındaki kernel benzerlik matrisi H_B hesaplanır. H_B matrisinin genişliği $n*q$ boyutundadır. Bu matrisin boyutun $n*n$ boyutlu tam kernel matrisinden daha azdır ($q<n$). İteratif süreç başlangıç bölünmesi ile başlar ve her bir iterasyonda H_B matrisi her bir veri noktası için en yakın küme merkezini tahmin etmek için kullanılır. Süreç, her bir kümenin en yakın küme elemanları değişmeyene kadar devam eder. Yaklaşık *K*-ortalama yönteminde zaman karmaşıklığı $O(q^2kr+qnkr+q^2n)$'dir. Buradaki r, yakınsamaya kadar ki iterasyon sayısını göstermektedir. Yöntemin uzay karmaşıklığı ise $O(nq)$'dur[91,93].

Hitendra Sarma ve arkadaşları 2011 yılında kernel *K*-ortalama yönteminin zaman ve uzay karmaşıklığı problemini ele almak için prototipe dayalı hibrit bir yaklaşım (prototype-based hybrid) önermişlerdir. Bu yöntem iki aşamada gerçekleşmektedir. İlk aşamada, veri seti grouplet sayısına bölünür. Groupletler kernel uzayında bulunur ancak her bir grouplet protip tarafından temsil edilir ki buna lider denir. Liderlerin kümesi eşik parametresine bağlıdır ve $O(n)$ zamanında türetilmiştir. İkinci aşamada, liderlerin bir kümesinin bölümünü türetmek için kernel *K*-ortalama yöntemi uygulanır. Sonuç olarak her bir lider kendi grubu ile yer değiştirir. Bu hibrit yöntemin zaman karmaşıklığı $O(n+p^2)$'dir. Burada p liderlerin boyutunu göstermektedir. Uzay karmaşıklığı ise yine $O(n+p^2)$'dir. Uygun bir eşik değeri seçildiğinde bu yöntem klasik kernel *K*-ortalama yönteminden daha hızlı sonuçlar verdiği görülmüş ancak kümeleme sonuçlarında küçük sapmalar oluşmuştur[91,94].

İki aşamalı kernel *K*-ortalama yöntemi (two-step kernel *K*-means) Radha Chitta ve arkadaşları tarafından 2011 yılında klasik kernel *K*-ortalama yönteminin hesaplama karmaşıklığını azaltmak için önerilmiştir. İlk aşamada, ver setinden *s* veri noktası rasgele seçilir ve optimum küme merkezi sadece bulunan örnek veri noktalarına bağlıdır. Daha sonra ikinci aşamada, tüm örneklenmemiş veri noktaları uzayda en yakın merkeze sahip olan kümeye atanır. Yapay küme merkezleri bağlı olunan uzaydaki optimum küme merkezlerini temsil etmek için kullanılır. Bu yaklaşım hem zaman hem de hafıza (memory) karmaşıklığını azaltır. Ancak klasik kernel *K*-ortalama yöntemi yerine bu yöntem kullanıldığında kümeleme sonuçlarında daha fazla sapma olduğu belirlenmiştir. Bunun sebebi de yapay küme merkezlerinin gerçek küme merkezlerini tam olarak temsil etmemesinden kaynaklanmaktadır. Bu sapmanın azaltılmasının bir yolu yapay küme merkezlerinin yerine gerçek, kesin küme merkezlerinin kullanılmasıdır. Bu yöntemi uygulayan algoritma ise tek geçişli kernel *K*-ortalama yöntemidir[91,93].

2.1.4.5. Tek Geçişli Kernel *K*-ortalama Kümeleme Algoritması (single pass kernel *K*-means clustering)

D veri setinde $S=\{X_1, X_2, \ldots, X_s\}$ rasgele seçilen örnekler olsun. *S* rasgele örneğinde *s*, model (Pattern) sayısını göstermektedir. $M^{(0)}$ başlangıç çekirdek noktaları, $\Pi^{(0)}$ veri setinin başlangıç bölümü ve $\Pi_s=\{C_1^S, C_2^S, \ldots, C_K^S\}$'dir[91].

Tek geçişli kernel *K*-ortalama yönteminde kriter fonksiyonu aşağıdaki gibi tanımlanmaktadır. Bu fonksiyon minimize edilmek istenmektedir.

$$J(X)=\sum_{X_i \in C_j^S} \|\phi(X_i) - \phi(X)\|^2.$$

Yukarıdaki formülde yer alan *X*, *d* boyutlu vektör $X=(x_1, x_2, \ldots, x_d)^T$'dür. Gradyan azaltma yöntemi amaç fonksiyonunu minimum yapan $X=X^*$ noktasını bulmak için kullanılır. Amaç fonksiyonu konveks bir fonksiyondur ve yerel bir minimum olmaması gibi sorun yoktur. X^*, C_j^S kümesinin kesin merkezi olarak alınır ve M_j olarak gösterilir[91].

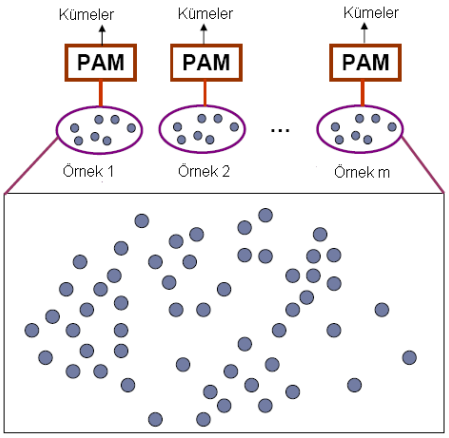
Önerilen bu yöntemde veri seti yalnızca bir kere taranır. Genel çalışma zamanı, başlangıçtaki rasgele örneklere (*S*) bağlıdır. Yöntemin zaman karmaşıklığı $O(s^2+t+nk)$'dır. Burada *s*, *S* rasgele örnekleminin genişliğini, *t* gradyan yöntemi kullanılarak *K* kesin küme merkezlerini bulmak içim gerekli olan zamanı, *n* başlangıçtaki veri seti genişliğini ve *K* ise gerekli olan

küme sayısını göstermektedir. Uzay karmaşıklığı ise $O(s^2)$ 'dir. $s<n$ olmasından dolayı önerilen yöntem klasik kernel *K*-ortalama yönteminden daha hızlıdır ve tek geçişli kernel *K*-ortalama yöntemi çok büyük veri setleri için kolayca uygulanabilir ve yorumlanabilir. İki aşamalı kernel *K*-ortalama yöntemi bu yöntemden daha hızlı çalışmasına rağmen, küme kesinliğinde daha fazla bir kayıp meydana gelmektedir[91].

2.1.4.6. CLARA Kümeleme Algoritması (Clustering Large Applications)

CLARA algoritması 1990 yılında Kaufman ve Rousseeuw tarafından geliştirilmiştir. CLARA algoritması niceliksel veriler için kullanılan bölümleyici bir yöntemdir ve büyük sayıda veri setleri içeren araştırmalar için kullanılmaktadır. Bu kümeleme algoritmasından daha önce geliştirilen PAM ve *K*-medoids algoritmaları büyük sayıda veriler için uygun sonuçlar üretmemektedir. Dolayısıyla bu soruna çözüm olarak CLARA algoritması sunulmuştur[79].

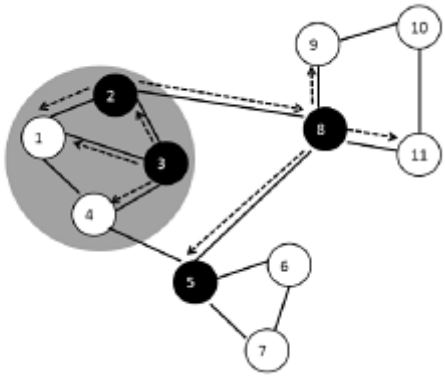
CLARA kümeleme algoritmasında tüm veri setinin taranmasına gerek yoktur. Yöntem örnek bir grup ele alarak işlem adımlarını gerçekleştirir. Bu örnek grup tüm veri setini temsil edebilecek nitelikte olmalıdır. Aksi takdirde algoritma sonucunda elde edilen sonuçlar güvenilir olmayacaktır. Bu örnek grup içerisindeki merkez noktalar, temsili noktalar bulunur ve PAM algoritması uygulanır. Örnek veri setindeki merkez noktalar artık tüm veri setinin merkezleridir ve kümeleme işlemleri bu merkez noktalarına göre yapılmaktadır. Gerçek merkez noktaları bu örneklem içine alınmış olmalıdır. Daha sonra yeniden bir örnek küme daha seçilir ancak bu durumda da yine temsilci, merkez noktalar başlangıçta belirlenen noktalardır. Bu da algoritmanın kaliteli ve hızlı sonuçlar üretmesine sebep olacaktır. Yukarıda anlatılan adımlar aşağıdaki Şekil 2.1.4.6.1'deki gibi özetlenebilir[12,33,55,79,89,95].



Şekil 2.1.4.6.1. CLARA algoritmasının şekilsel gösterimi

Bu tekrar işlemi 5 defa yinelendiğinde ve her tekrarda 40+2*K* örnek seçilmesi durumunda algoritma en iyi sonucu üretmektedir. Çünkü küme sayısı 1 ile 30, örneklerde 42 ile 100 nesne arasında olmak zorundadır[95,96].

Şekil 2.1.4.6.2'de CLARA algoritmasının çalışma şekli şematize edilmiştir. Şekilden de görüldüğü gibi algoritma ilk olarak rasgele örnek seçer ve daha sonra en iyi komşusuna taşınır. Böylelikle tüm komşular algoritmada dikkate alınmış olur. CLARA algoritması bir alt grafik (subgraph) oluşturur ve bundan sonraki işlemlerde bu alt grafik dikkate alınarak gerçekleştirilmektedir[97].



Şekil 2.1.4.6.2. CLARA Algoritması

CLARA algoritması asagıdaki gibi özetlenebilir.

Adım 1: i=1,2,3,4,5 için asağıdaki adımları tekrarlanır.
Adım 2: Orijinal veri setinden tesadüfi olarak 40+2*K*' lık bir örneklem çekilir ve çekilen bu örneğin *K* medoidini belirlemek için PAM algoritması uygulanır.
Adım 3: Tüm veri seti içindeki her bir nesne için o nesneye en çok benzer *K*-medoidler belirlenir.
Adım 4: Bir önceki adımda elde edilen kümelerin, ortalama uzaklıkları hesaplanır. Bu değer mevcut minimum değerden daha küçük ise minimum değer olarak atanır ve *Adım 2*' de elde edilen *K* medoidleri en iyi medoid kümesi olarak kabul edilir.
Adım 5: Bir sonraki iterasyona baslamak için *Adım 1*' e geri dönülür[30,95].

PAM algoritması veri setindeki en iyi *K*-medoidleri bulurken, CLARA seçilen örnekler arasındaki en iyi *K*-medoidi bulmaktadır. CLARA algoritmasının iyi bir sonuç vermesi seçilen örneğe bağlıdır. Eğer örnekleme sürecinde en iyi *K*-medoid seçilmez ise CLARA

algoritması kesinlikle en iyi kümeleme sonucunu vermeyecektir. Eğer örnekleme yanlı olursa yine iyi bir kümeleme olmayacaktır. Ancak yöntem örnekler ve *K*-medoidler iyi seçildiğinde büyük veri setleri için iyi kümeleme sonuçları üretecektir. CLARA algoritmasının karmaşıklığı ise $O(k(40+k)^2+k(n-k))$'dır. Burada *K* küme sayısını, *n* ise nesne sayısını göstermektedir[12,33,55,79,89,95].

2.1.4.7. PAM Kümeleme Algoritması (Partitioning Around Medoids)

PAM algoritması 1990 yılında Kaufman ve Rousseeuw tarafından önerilmiştir[79]. Algoritma, *n* tane nesneyi bulabilmek için her bir kümedeki temsilci nesneleri belirleme yaklaşımını benimsemektedir. Burada sözü edilen temsilci nesnelere medoid adı verilmektedir. Medoid, küme içerisinde en merkezde olan nesneyi göstermektedir. Medoidler seçildikten sonra, her bir seçilmeyen nesne kendine en çok yakın (benzeyen) olan medoide atanır. Bir başka ifadeyle O_j seçilmemiş bir nesne ve O_m seçilmiş bir medoid ise, $d(O_j, O_m)=min_{O_e} d(O_j, O_e)$ ise O_j'nin O_m olarak temsil edilen kümeye ait olduğu söylenebilir. Formülde yer alan min_{O_e} tüm O_e medoidleri içindeki en küçük değeri gösterirken, $d(O_1, O_2)$ notasyonu O_1 ve O_2 arasındaki benzememezlik veya uzaklığı ifade etmektedir[33,98].

PAM algoritmasında, kümelemenin kalitesi, nesnelerin kendi medoidleri ile arasındaki ortalama benzememezlikleri ile ölçülmektedir. *K* medoidi bulabilmek için PAM ilk adımda *n* nesneyi keyfi olarak seçmektedir. Daha sonra her bir adımda kümeleme kalitesinde bir iyileşme olana kadar seçilmiş nesne O_m ile seçilmemiş olan O_p nesnesinin yerlerinin değiştirir[98].

Yöntemin işleyişi detaylı olarak açıklanmadan önce bazı tanımlamaların yapılması gerekmektedir.

O_m, yer değiştirilebilir olan medoidleri,
O_p, O_m yerine değiştirilen yeni meodidleri,
O_j, yer değiştirilebilecek veya yer değiştirmeyecek olan diğer medoid olmayan nesneleri,
$O_{j,2}$, A ve M olmaksızın O_j'ye en yakın o anki medoidi göstermektedir.

O_m ve O_p arasındaki değişimin etkisini formülize etmek için PAM algoritması tüm medoid olmayan O_j nesneleri için C_{jmp} maliyetini hesaplamaktadır.

O_j'nin aşağıdaki durumlarına göre C_{jmp}'nin tanımlanması da farklılık gösterecektir[98].

Durum 1: O_j'nin O_m ile temsil edilen kümeye ait olduğu varsayılsın. Ayrıca O_j, O_p'den ziyade $O_{j,2}$'ye daha yakın olsun. Yani $d(O_j, O_p) \geq d(O_j, O_{j,2})$'dir. Formülde yer alan $O_{j,2}$, O_j'ye en yakın ikinci medoiddir. Bu yüzden O_m, O_p ile yer değiştirirse O_j, $O_{j,2}$ ile temsil edilen kümeye dâhil olur. Dolayısıyla C_{jmp} aşağıdaki gibi hesaplanmaktadır.

$$C_{jmp} = d(O_j, O_{j,2}) - d(O_j, O_m)$$

Yukarıdaki eşitlik her zaman negatif olmayan bir C_{jmp} değeri vermektedir[33,98].

Durum 2: O_j yine O_m ile temsil edilen kümeye ait olsun. Ancak bu kez O_j, $O_{j,2}$' ye O_p' den daha yakın olsun. Yani $d(O_j, O_p) < d(O_j, O_{j,2})$ olduğu varsayılsın. Böylece eğer O_m, O_p ile yer değiştirirse O_j nesnesi O_p ile temsil edilen kümeye dâhil olmaktadır. Bu durumda O_j için maliyet aşağıdaki formül yardımıyla hesaplanmaktadır[33,98].

$$C_{jmp} = d(O_j, O_p) - d(O_j, O_m)$$

Durum 1'in aksine O_j'nin O_m' ye mi O_p' ye mi daha yakın olup olmamasına göre C_{jmp} negatif ve pozitif değerleri alabilmektedir.

Durum 3: O_J'nin O_m ile temsil edilen küme yerine başka bir kümeye ait olduğu düşünülsün. Bu kümenin temsilci nesnesi $O_{j,2}$ olsun. Ancak O_j, O_p yerine $O_{j,2}$' ye daha benzer olsun. Bu durumda, O_m, O_p ile yer değiştirse bile $O_{j,2}$ ile temsil edilen kümede kalmaya devam edecektir. Böylece maliyet değeri sıfıra eşit ($C_{jmp}=0$) olacaktır[2,3].

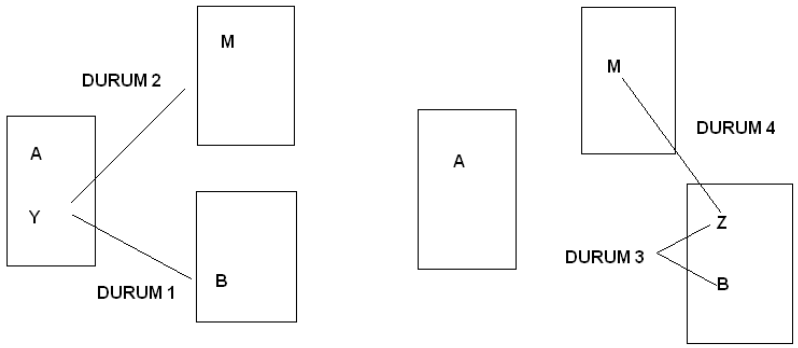
Durum 4: O_j, $O_{j,2}$ ile temsil edilen kümeye ait olsun. Ancak O_j, O_p ile karşılaştırıldığında $O_{j,2}$'ye daha az benzer olsun. O zaman O_m' yi O_p ile değiştirmek, O_j' nin O_p kümesinden $O_{j,2}$ kümesine geçmesine sebep olacaktır ki bu durumda maliyet aşağıdaki formül yardımıyla hesaplanır[33,98].

$$C_{jmp} = d(O_j, O_p) - d(O_j, O_{j,2})$$

Maliyet değeri her zaman negatif değer almaktadır.

Yukarıdaki tüm durumlar birleştirildiğinde, O_m' nin O_p ile yer değiştirmesi sonucunda elde edilen toplam maliyet $TC_{mp}=\sum_j C_{jmp}$ olacaktır.

PAM algoritmasının işleyişini daha iyi anlamak için *A* ve *B* olmak üzere 2 medoidin olduğu varsayılsın. *A* medoidi yeni medoid olan *M* ile yer değiştirsin. Tüm nesneler (*Y* nesneleri), *A* ile temsil edilen kümenin içinde olsun. Yer değiştirmeler ışığında yeni medoidlerin bulunması gerekmektedir. Bunun için iki durum söz konusudur. İlk durumda *Y, M* ile temsil edilen kümeye değil *B* ile temsil edilen kümeye dâhil olur. İkinci durumda *Y, M* ile temsil edilen kümeye dâhil olur ve *B* ile temsil edilen kümede herhangi bir değişiklik meydana gelmez. *A* kümesinde yer alan *Y* objeleri dışında, *B* kümesinde yer alan *Z* nesnelerinin de dikkate alınması gerekmektedir. Yer değiştirme ışığında *Z, B*'de de kalabilir temsili noktası *M* olan yeni kümeyede dâhil olabilir. Bu durumlar Şekil 2.1.4.7.1'de şematize edilerek gösterilmiştir[98].



Şekil 2.1.4.7.1. *A*'nın *M* olarak değiştirilmesi ile oluşabilecek dört durum

PAM algoritmasının adımları ise aşağıdaki gibi özetlenebilir[33,98].

1. *n* tane temsilci nesne keyfi olarak seçilir.
2. O_m seçilmiş, O_p seçilmemiş olmak üzere, tüm O_m ve O_p nesneleri arasındaki TC_{mp} maliyeti hesaplanır.
3. $\min_{O_m,O_p} TC_{mp}$ karşılık gelen O_m, O_p çiftleri seçilir ve eğer TC_{mp} değeri negatif ise O_m ile O_p yer değiştirir ve 2. Adıma geri dönülür.
4. Diğer durumda, her bir seçilmemiş nesne için en benzeyen temsilci nesne seçilir ve işlemler sonlandırılır[2,3].

PAM algoritma küçük veri setlerinde tatminler, iyi sonuçlar üretirken büyük veri setleri için bu durum geçerli değildir. Aslında PAM algoritmasının karmaşıklık analizi düşünüldüğünde çok da şaşırtıcı bir durum değildir. Adım 2 ve 3'de, O_m ve O_p için $K(n\text{-}K)$ tane çift vardır. Her bir çift için TC_{mp} hesaplamak için $(n\text{-}K)$ seçilmeyen nesne için değerlendirme yapmak

gerekmektedir. Bu yüzden adım 2 ve 3'de $O(K(n-K)^2)$'dir. Bu karmaşıklık sadece bir iterasyon içindir. Bu açıklamalar doğrultusunda PAM algoritmasının büyük *n* ve *K* değerleri için çok maliyetli olduğu söylenebilir. PAM yönteminin bu dezavantajlarını ortadan kaldırmak amacı ile de CLARA algoritması geliştirilmiştir. Ayrıca PAM algoritması aykırı ve gürültülü veriler varlığında *K*-ortalama yöntemine göre daha dayanıklıdır (robust) [98,99].

2.1.4.8. TP-PAM Kümeleme Algoritması (Tree Pruning PAM Algorithm)

PAM algoritması *n* nesne için *K* bölünmeye karar verebilmek için ilk olarak *K* temsilciyi rasgele olarak seçmektedir. Algoritma daha iyi küme temsilcileri elde edebilmek için de süreci tekrarlamaktadır. Böylece tüm olası nesne çiftleri analiz edilmiş olunur. Bu da PAM algoritmasının zaman alıcı ve düşük verimliliğe sahip olduğunu göstermektedir. Başlangıç küme merkezlerini rasgele seçmek, küme doğruluğunu ve algoritmanın etkinliğini önemli düzeyde etkilemektedir. Dolayısıyla bu algoritmada istikrarlı sonuçlar elde edilememektedir. Her seferinde başlangıç medoidleri değiştirildiğinde (ve başlangıç medoidler istenilen küme merkezlerine yakın değilse) sonuçlar değişecektir. PAM algoritmasının bu eksikliklerini gidermek amacıyla 2012 yılında Feng Bo ve arkadaşları tarafından TP-PAM algoritması önerilmiştir[100].

TP-PAM algoritması iki ana aşamadan oluşmaktadır. İlk aşamada, başlangıç temsili noktalarının rasgele belirlenmesi yerine TP (tree pruning) adı verilen bir yöntem kullanılarak optimize başlangıç küme merkezleri belirlenir. 2. Aşamada ise başlangıç temsilci noktaları olarak TP sonucunda elde edilen optimize küme merkezlerini kullanarak bilinen PAM algoritması uygulanır. Böylece daha iyi küme sonuçları elde edilecek ve zaman kaybı olmayacaktır. Yeni geliştirilen bu algoritmada nesneler arasındaki benzememezlikleri hesaplamak için tüm değişken tiplerini (mixed types) de dikkate alan bir formülasyonu kullanmaktadır. Böylece *K* başlangıç küme merkezleri ön değerlendirme sonucunda elde edilmiş olur[100].

Gerçek hayattan bilindiği üzere değişkenler aralık ölçekli, simetrik ikili, kategorik ve oran ölçekli düzeyde ölçülmüş olmaktadır. Dolayısıyla Öklid ve Manhattan uzaklık formülleri karışık bir değişken tipine sahip veri setleri için kullanışlı olmayacaktır. Bu sorun aşağıdaki karma türde değişkenler için verilen formül yardımıyla çözülebilir[100].
U veri setinde *N* nesnenin olduğu varsayılsın ve her bir nesnede *m* değişken olsun. Böylece *x* objesi $x=(x_1, x_2, \ldots, x_m)$ şeklinde tanımlanabilir.

Tanım 1: x ve *y* nesneleri arasındaki uzaklık,

$$\text{dist}[x,y]=\frac{\sum_{f=1}^{m}\delta_{xy}^{f}d_{xy}^{(f)}}{\sum_{f=1}^{m}\delta_{xy}^{f}}$$ 'dir.

δ_{xy}^{f} göstergesi x_f (veya y_f)kayıp olduğunda veya $x_f = y_f = 0$ ve *f* değişkeni asimetrik ikili bir yapıda olduğunda sıfır değerini alır. Diğer durumlar için δ_{xy}^{f} göstergesi bire eşit olur. *f* değişkeninin *x* ve *y* benzememezliğine katkısı $d_{xy}^{(f)}$'dir ve *f* değişkeninin türüne bağlı olarak da farklı şekillerde hesaplanmaktadır.

Eğer f aralık ölçme düzeyinde ise $d_{xy}^{(f)}=\frac{|x_f-y_f|}{max_h h_f-min_h h_f}$ formülü yardımıyla hesaplanır. Burada *h*, *f* değişkeni için tüm kayıp olmayan nesneler için çalıştırılır. Eğer *f* ikili ve kategorik türde bir değişken ise $x_f = y_f$ olduğunda $d_{xy}^{(f)}$ sıfıra eşit olur. Diğer durumlarda ise $d_{xy}^{(f)}$ değeri bire eşit olacaktır. Eğer *f* sıralı ölçme düzeyinde ise r_{xf} rankları hesaplanır ve $z_{xf}=\frac{r_{xf}-1}{m_f-1}$ değeri bulunur. Daha sonra z_{xf}'ye aralık düzeydeki gibi muamele yapılır.

Tanım 2: C_j kümesindeki mutlak hata toplamı $E(C_j)$ aşağıdaki gibi tanımlanmaktadır[100].

$$E(C_j)=\sum_{p\in C_j}|dist(p,o_j)|$$

Yukarıdaki formülde yer alan *p*, C_j kümesi içerisindeki nesne iken, o_j değeri C_j kümesinin medoidlerini göstermektedir.

TP algoritması dört adımdan oluşmaktadır. 1. Adımda, yukarıda anlatılanlar yardımıyla veri setindeki her iki nesne için benzememezlikler ölçülerek benzememezlik matrisi olan *D* matrisi elde edilir. 2. Adımda, benzememezlik matrisi kullanılarak minimum yayılan ağaç (minimal spanning tree) oluşturulur. Bu ağaçta düğümler nesneleri temsil ederken, dallar benzememezlikleri göstermektedir. 3. Adımda, ağaçta maksimum ağırlıklık *K-1* dal bulunur ve bu dallar budanır. Sonuç oalrak ağaç *K* alt ağaca bölünmüş olur. Bu da veri setinin *K* kümeye bölündüğünü gösterir. 4. Adımda, mutlak hata formülü kullanılarak her küme için mutlak hata toplamları hesaplanır ve başlangıç küme merkezleri olarak en iyi nesneler seçilir. Sonuç olarak *K* başlangıç küme merkezleri elde edilir. *K* başlangıç küme merkezleri içinde PAM algoritması çalıştırılır[100].

2.1.4.9. Bulanık *c*-ortalama Kümeleme Algoritması (Fuzzy c-means clustering: FCM)

Bulanık *c*-ortalama yöntemi *K*-ortalama yöntemi gibi katı bir kümeleme yöntemi değildir. Bir başka ifadeyle gözlemler kesin olarak bir kümeye üye değildir. Ancak 0 ile 1 arasında değişen bir derecede kümeye üyelikler tespit edilir. Bu durumda da verilerin birden fazla kümeye üyeliği olacağından bulanıklık söz konusudur. Klasik yaklaşımlarda bir nesne ya kümeye aittir ya da değildir. Ait olduğu durumda üyelik değeri 1 değerini alırken, nesne kümenin elemanı olmadığında 0 değerini almaktadır. Ancak bulanık kümeleme yönteminde üyelik değerleri [0,1] aralığında sonsuz sayıda değer alabilmektedir. Örneğin klasik kümeleme yöntemlerinde kesin olarak sıcak-soğuk ifadeleri kullanılabilirken, bulanık kümelemede biraz soğuk-biraz sıcak ifadeleri kullanılmaktadır. Klasik kümeleme de hepsi, hiçbiri, kesin mantığı var iken bulanık kümeleme yöntemlerinde kısmi, belirli derecede mantığı söz konusudur. Dolayısıyla bulanık kümeleme algoritmalarında bir aralık söz konusudur. Nesneler 0 ile 1 arasında en yüksek olasılığa sahip kümelere atanmaktadır[101].

Her birim kümelerde eşit üyeliğe sahip ise, kümeleme tamamen bulanık olmaktadır. Kümelemenin bulanık olup olmadığına karar verebilmek için Dunn'ın parçalama katsayısı kullanılmaktadır ve aşağıdaki gibi formüle edilmektedir[101].

$$F(u)=\sum_{i=1}^{n}\sum_{v=1}^{k}u_{iv}^{2}/n$$

Yukarıdaki formülde *K* küme sayısını, *n* gözlem sayısını gösterirken u_{iv} değeri *i* nesnesinin *v* kümesine olan bilinmeyen üyeliğini göstermektedir. Yukarıdaki verilen Dunn katsayısı $[\frac{1}{K}, 1]$ arasında değer alabilmektedir. Tamamen bulanık kümeleme durumunda tüm $u_{iv}=\frac{1}{K}$ olduğundan $F(u)=\frac{1}{K}$ değerini alacaktır. Tamamen katı kümeleme durumunda ise tüm $u_{iv}=0$ ve $F(u)=1$ olacaktır. Değeri 0 ile 1 arasında değişen normalleştirilmiş Dunn katsayısı $F_n(u)=\frac{kF_k-1}{k-1}$ eşitliğinden elde edilir. Bu katsayının sıfır değerini alması tamamen bulanıklığı gösterirken, 1 değerini alması ise katı kümelenmeyi göstermektedir. Bu katsayıya aynı zamanda bulanıksızlık endeksi de (nonfuzziness index) denilmektedir[101].

Bulanık *c*-ortalama algoritması iteratif kümeleme yöntemidir. Yöntem grup içindeki hata kareler ağırlıklandırılmış amaç fonksiyonu minimize ederek optimum *c* bölümü üretmektedir. Bulanık *c*-ortalaması için amaç fonksiyonu aşağıdaki gibidir[102].

$$J_{FCM}=\sum_{k=1}^{n}\sum_{i=1}^{c}(u_{ik})^{q}d^{2}(x_k,v_i)$$

Formülde yer alan *X*, *p* boyutlu uzay vektöründeki veri setini göstermektedir. $X=\{x_1,x_2,.....,x_n\}\subseteq R^p$. *n* veri sayısını, *c* küme sayısını göstermektedir ve $2\leq c<n$'dir. u_{ik}, *i.* kümedeki x_k nesnesinin üyelik derecesi, *q* bulanık üyelikte kullanılan ağırlık değeridir. v_i, *i* küme merkezindeki prototipdir. $d^2(x_k,v_i)$, x_k nesnesi ile v_i küme merkezi arasındaki uzaklık ölçüsüdür. Bu amaç fonksiyonunun çözümü aşağıdaki iteratif süreç yardımıyla yapılmaktadır[102].

1. *c*, *q* ve ∈ (kritik değer) değerleri belirlenir.
2. Bulanık bölünme matrisi olan $U=[u_{ik}]$ başlatılır.
3. Döngü sayacı ayarlanır.
4. $U^{(b)}$ ile birlikte $\{v_i^{(b)}\}$ küme merkezleri hesaplanır.

$$v_i^{(b)}=\frac{\sum_{k=1}^{n}(u_i^{(b)})^{q}x_k}{\sum_{k=1}^{n}(u_i^{(b)})^{q}}$$

5. $U^{(b+1)}$ üyelikleri belirlenir ve *K*=1'den *n*'e kadar aşağıdaki eşitlik çözülür.
   $I_k=\{i|1\leq i\leq c, d_{ik}=\|x_k-v_i\|=0\}$. *K.* sütun matrisi için yeni üyelik değerlerini iki durum için hesaplanır.
   (a) Eğer $I_k=\boldsymbol{\phi}$ ise, $u_{ik}^{(b+1)}=\frac{1}{\sum_{j=1}^{c}(\frac{d_{ik}}{d_{jk}})^{\frac{2}{(q-1)}}}$
   (b) Diğer durumlarda tüm $i\notin I$ ve $\sum_{i\in I_k}u_{ik}^{(b+1)}=1$ için $u_{ik}^{(b+1)}=0$'dır.
   Sonraki *K*'ya geçilir.
6. Eğer $\|U^{(b)}-U^{(b+1)}\|<\in$ ise işlem sonlandırılır. Diğer durumlarsa $b=b+1$ yapılır ve adım 4'e dönülür.

Bulanık *c*-ortalama kümeleme yöntemi üst üste çakışan veri setleri için en iyi sonucu vermektedir ve *K*-ortalama algoritmasına göre nispeten daha iyi sonuçlar üretmektedir. Ancak yöntem kümelerin sayısı için ön bilgi gerektirmektedir.

2.1.4.10. CLARANS Kümeleme Algoritması (Clustering Large Applications Based on Randomized Search)

CLARANS algoritması Raymond T. Ng ve Jiawei Han tarafından 1994 yılında VLDB'94 konferansında ilk kez sunulmuş ve niceliksel yapıdaki değişkenlerin kümelenmesinde kullanılılacağı açıklanmıştır. CLARANS algoritması PAM ve CLARA yöntemlerinin

eksiklerini gidermek için geliştirilmiştir. Bu algoritma PAM ve CLARA yöntemlerinin birleşimi olarak düşünülebilir ve diğer iki algoritma ile karşılaştırıldığında daha etkili sonuçlar ürettiği bilinmektedir[103].

Yöntemde birleştirilecek nesnelere $G_{n,k}$ grafiğine bakılarak karar verilir. *n* nesne sayısı iken *K* medoid sayısıdır. $G_{n,k}$ grafiği *K* medoidlerin meydana getirdiği bir grafiktir. Grafikteki bir düğüm *K* nesne seti ile temsil edilir ve $\{O_{m_1,}\ldots\ldots, O_{m_k}\}$ ile gösterilmektedir. Sırasıyla $O_{m_1,}\ldots\ldots, O_{m_k}$ ise seçilen medoidlerdir[103].

$S_1=\{O_{m_1,}\ldots\ldots, O_{m_k}\}$ ve $S_2=\{O_{w_1,}\ldots\ldots, O_{w_k}\}$ düğümlerinin komşu olabilmesi için

$|S_1 \cap S2| = k-1$ şartı sağlanmalıdır. Dolayısıyla her bir düğüm $K(n-K)$ tane komşusu olduğu görülmektedir. Her bir düğüm *K* medoidlerin toplanması ile meydana gelir ve her bir düğüm noktası kümelemenin olduğuna işaret etmektedir. Her bir düğüm bir maliyet olarak belirlenir. Sözü edilen maliyet kelimesi ise nesneler ile kümelerin medoidleri arasındaki toplam benzememezlikler olarak tanımlanabilir[103].

PAM algoritması $G_{n,k}$ grafiğindeki minimumu arama olarak görülebilir. Her bir düğümün komşuları ile maliyeti incelenir. Maliyeti en fazla düşüren komşu düğüm ile incelenmekte olan düğüm yer değiştirecektir ve bu işlemler minimumu elde edene kadar devam edecektir. *n* ve *K*'nın büyük değerleri için $K(n- K)$ komşusu olan bir düğümü incelemek çok fazla zaman gerektirmektedir. Bu da PAM algoritmasının büyük veriler için etkili olmadığını göstermektedir. Diğer taraftan CLARA algoritması daha az komşuyu incelemeye çalışmaktadır ve orijinal grafikteki alt grafik araştırmalarını sınırlamaktadır. Buradaki problem altgrafiklerin örneklerdeki nesneler ile tanımlanmasıdır. S_α, örnekteki nesneler setini gösterse, $G_{S_{\alpha,k}}$ alt grafiği, S_α setindeki tüm düğümlerden oluşmaktadır. $G_{n,k}$ grafiğinde eğer *M* minimum düğümü gösteriyorsa, ve *M*, $G_{S_{\alpha,k}}$'yı içermiyorsa *M* hiçbir zaman $G_{S_{\alpha,k}}$'yı bulamayacaktır. Bir başka ifadeyle CLARA algoritması sabit bir örneklem kullandığı için aranılan minimum nokta o örnek içinde olmayabilir. Ancak CLARANS algoritması sabit bir örneklem taramak yerine her seferinde tesadüfi olarak örnekler çekmektedir. Tabi bunu yaparken de tüm veri tabanı taranmamaktadır. Ancak CLARA algoritmasındaki gibi bir alt grafik ile çalışmayı sınırlandırmaz. CLARA algoritması çalışmanın başında örnek bir düğüm belirlerken, CLARANS algoritması çalışmanın her bir aşamasında komşuların örneklerini

belirlemektedir. CLARANS algoritması CLARA algoritmasından daha kaliteli sonuçlar üretmektedir ve CLARANS algoritması daha az sayıda çalışma gerektirmektedir[103].

CLARANS algoritmasının işlem adımları aşağıdaki gibi özetlenebilir[103].

1. Girdi parametresi olarak maksimum komşu sayısı ve yerel miktar belirlenir. Maksimum komşu sayısı incelenecek komşu sayısının üst limitini, yerel miktar ise elde edilecek yerel minimum nokta sayısının alt sınırını göstermektedir. i=1'den başlatılır ve minimum maliyeti büyük bir değer olarak belirlenir.
2. $G_{n,k}$'da keyfi bir düğüm belirlenir.
3. j=1'den başlatılır.
4. Seçilen düğümün rasgele komşusu seçilir ve bu iki düğümün maliyeti hesaplanır.
5. Eğer S daha az bir maliyete sahip ise, incelenecek yeni düğüm S olacaktır ve tekrar adım 3'e dönülecektir.
6. S daha az bir maliyete sahip değil ise, j' yi bir artır. Eğer $j \leq$maksimum komşu sayısı ise adım 4'e gidilir.
7. Eğer j > maksimum komşu sayısı ise minimum maliyet ile var olan maliyeti karşılaştırılır. Eğer eski maliyet minimum maliyetten daha az ise minimum maliyet var olan maliyet olarak belirlenir.
8. i' yi 1 artır. Eğer i > yerel miktar ise, çıktı olarak en iyi düğüm verilir ve adımlar durdurulur. Diğer durumda adım 2'ye geri dönülür.

CLARANS algoritmasının hesaplama karmaşıklığı $O(kn)^2$'dir. Bu algoritma aykırı ve gürültülü değerlere karşı duyarlı iken konveks olmayan şekilli kümeler için uygun sonuçlar vermektedir. Bunun yanı sıra CLARANS algoritması aykırı değerleri belirlemede diğer bölümleyici yöntemler arasında yer alan CLATIN, CLARA ve PAM algoritmalarından daha iyi sonuç verdiği belirlenmiştir[103].

2.1.4.11. Geliştirilmiş CLARANS Kümeleme Algoritması (ENHANCED CLARANS: ECLARANS)

ECLARANS kümeleme algoritması, CLARANS algoritmasındaki iterasyon sayılarını azaltmak için kullanılmaktadır. Algoritma, rasgele arama işlemi yerine uygun keyfi düğüm seçerek kümeleme işlemlerini gerçekleştirir. Keyfi olarak düğüm seçmesi de bu algoritmayı diğer algoritmalardan ayıran en önemli özelliğidir. Bu yöntem işleyiş açısından CLARANS

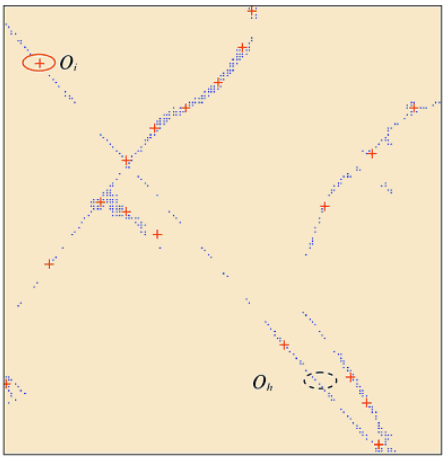
algoritmasına çok benzemektedir. Ancak ECLARANS yönteminde CLARANS algoritmasının iterasyon sayısını azaltmak için keyfi düğümler seçilmektedir.

ECLARANS algoritmasının işlem adımları aşağıdaki gibidir[104,105].

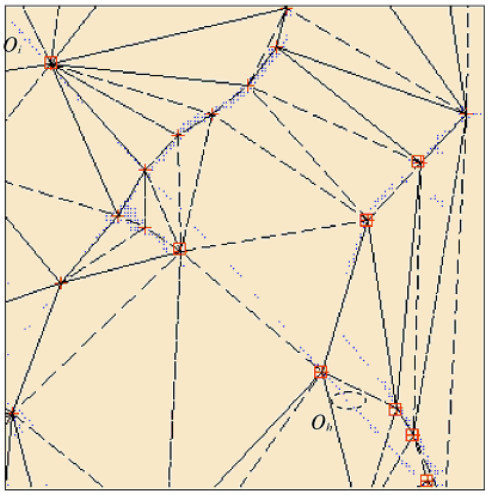
1. Girdi parametresi olarak maksimum komşu sayısı ve yerel miktar belirlenir. *i,* 1'den başlatılır ve minimum maliyeti büyük bir değer olarak belirlenir.
2. Her bir veri noktası arasındaki uzaklıklar hesaplanır.
3. Maksimum veri noktası uzaklığı *n* seçilir.
4. Keyfi bir düğüm *n:K* belirlenir.
5. *j*, 1'den başlatılır.
6. Rasgele *S* komşu seçilir ve iki düğüm maliyeti arasındaki fark hesaplanır.
7. Eğer *S* daha düşük bir maliyete sahip ise, *S* mevcut düğüm olarak belirlenir.
8. Diğer durumlarda *j* bir artırılır. Eğer *j* maksimum komşu sayısı ise adım 6'ya dönülür.
9. Diğer durumlarda eğer *j* > maksimum komşu sayısı ise var olan maliyet ile minimum maliyet karşılaştırılır. Eğer önceki maliyet minimum maliyetten daha düşük ise, yeni maliyet önceki maliyettir ve en iyi düğüm ayarlanır.
10. *i* bir artırılır. Eğer *i* > yerel miktar ise, en iyi düğüm çıktı olarak verilir. Diğer durumlarda adım 4'e dönülür.

2.1.4.12. CLATIN Kümeleme Algoritması (Clustering Large Application with Triangular Irregular Network)

Kullanıcı PAM algoritması ile çalıştığında O_i medoidi ile medoid olmayan O_h nesnelerinin yer değiştirmesi veri setinin küçük bir bölümünü etkilemektedir. Bu yüzden PAM algoritmasının ikinci adımında O_i medoidi ile medoid olmayan O_h nesnesinin yer değiştirmesi sonucunda hesaplanacak olan toplam maliyet değeri için sadece veri setinin küçük bir bölümü yeterli olacaktır (Şekil 2.1.4.12.1). Ancak bu durumda orijinal veri setinin etkinliğinin nasıl belirleneceği sorusu akıllara gelmektedir. Bu sorunu çözebilmek için üçgen ağları yöntemi kullanılmaktadır. Medoidlerin üçgen ağları, alt kümeleri bulmayı sağlayacaktır (Şekil 2.1.4.12.2). Üçgen ağları kullanılarak kümeleme işlemleri yapıldığında daha etkili yeni bir *k* medoid yöntemi elde edilmiş olunur. Bu yönteme de büyük uygulamalarda düzensiz üçgen ağı kümeleme yöntemi adı verilmektedir[103,104].



Şekil 2.1.4.12.1. PAM algoritmasında O_i medoidi ile medoid olmayan O_h nesnelerinin yer değiştirmesi (kırmızı çarpılar yeni (current) seçilmiş medoidleri, mavi noktalar ise medoid olmayan nesneleri göstermektedir).



Şekil 2.1.4.12.2. CLATIN algoritmasında O_i medoidi ile medoid olmayan O_h nesnelerinin yer değiştirmesi değiştirmesi (kırmızı çarpılar yeni (current) seçilmiş medoidleri, mavi noktalar ise medoid olmayan nesneleri, siyah tire çizgiler şimdiki medoidlerin TIN'lerini (triangular irregular network), kırmızı dikdörtgenler ise olası etkilenen medoidleri göstermektedir).

CLATIN algoritmasının işlem adımları aşağıdaki gibi sırasıyla verilmektedir[103,104].

1. Başlatma

   Başlangıç medoidi olarak *k* temsilci nesneyi keyfi olarak seç
2. *k* medoidlerin TIN'lerini oluştur

   O_i ve O_h nesne çiftleri için toplam maliyet TC_{ih}'yi hesapla (O_i: var olan medoid nesnesi, O_h medoid olmayan nesnelerden biri)

3. Medoid-TIN'lerde bağlantı analizi yardımıyla *S* nesne alt kümesini belirle
4. *S* alt kümesi nesneleri boyunca toplam maliyet TC_{ih}'yi hesapla
5. En küçük TC_{ih}'yi veren O_i ve O_h çiftlerini seç. Eğer en küçük TC_{ih} değeri minimum ise, O_i'yi O_h ile değiştir
6. Lokal olarak TIN ve kümeleme sonuçlarını güncelle
7. Adım 2'ye dön
8. Diğer durumlarda her bir seçilmeyen nesne çifti için en benzer temsilci nesneyi bul
9. İşlemleri sonlandır.

CLATIN kümeleme algoritması küresel şekillerde dâhil olmak üzere rasgele (arbitrary) tüm küme şekilleri için uygulanabilmektedir. Algoritmanın zaman karmaşıklığı ise *O(nlogn)*'dir. Karmaşıklıkta yer alan *n* nesne sayısını ifade etmektedir[103,104].

2.1.4.13. *X*-Ortalama Kümeleme Algoritması (*X*-Means Clustering Algorithm)

Klasik *K*-ortalama kümeleme algoritmasının en büyük dezavantajı *K* değerine karar verememektir. *X*-ortalama algoritması, *K*-ortalama yönteminin bu dezavantajını gidermek için öne atılmıştır. *X*-ortalama yönteminde *K* değeri kolay ve hızlıca tahmin edilebilmektedir[106]. Yöntem her bir *K*-ortalamanın çalıştırılmasından sonra harekete geçmektedir. Veriye uyması için var olan merkezlerin alt setlerinin hangisinin bölünmesi için lokal kararlar verilmektedir. Bölme kararı, BIC kriterinin hesaplanması ile gerçekleştirilmektedir. Daha önceden *K*-ortalama kümeleme algoritmasında *K* değeri kullanıcı tarafından girdi parametresi olarak belirlenirdi. Ancak bu yöntemde kullanıcı sadece *K* değerleri için bir aralık belirler, *X*-ortalama ise en iyi *K* değerini bularak kümeleme işlemlerini gerçekleştirir. *K* değerinin seçimi ise en iyi model seçim kriterinin belirlenmesi ise bulunmaktadır[106].

Algoritma verilen aralıkta en düşük değeri ile başlayarak, üst sınıra kadar merkezler ekleyerek devam etmektedir. Bu süreç içerisinde en iyi skora ait merkez eti kayıt edilir ve bu değerlere göre sonuç kümeler elde edilir[106].

K değerinin farklı değerlerinden en iyisine karar verebilmek için $Pr[M_j|D]$ olasılığı kullanılmaktadır. BIC değerinin hesaplanması için Kass ve Wasserman'in aşağıdaki formülünden yararlanılmaktadır[106].

$$BIC(M_j) = \widehat{l}_j(D) - \frac{p_j}{2} * logR$$

Yukarıdaki eşitlikte geçen $\widehat{l_j}(D)$ terimi, j. modele göre verinin log-olabilirliğidir ve maksimumu alınır. p_j ise, M_j'deki parametrelerin sayısını ifade etmektedir. Aynı zamanda bu eşitlik Schwarz kriteri olarak da bilinmektedir[106].

2.1.4.14. Cascade *K*-ortalama Kümeleme Algoritması

Klasik *K*-ortalama yönteminin bilinen en büyük dezavantajı *K* değerinin belirlenmesidir. Bu dezavantajı gidermek amacıyla Calinski Harabasz kriteri kullanılmaktadır. Cascade *K*-ortalama kümeleme algoritmasın Calinski-Harabasz kriteri dikkate alınarak en iyi *K* değerine karar verilmektedir. Bu kriter adında anlaşılacağı üzere 1974 yılında Calinski ve Harabazs tarafından önerilmiştir. Amaç küme kaliteleri dikkate alınarak en iyi *K* değerine karar verebilmektir[77,107].

Calinski-Harabasz kriteri aşağıdaki formül yardımıyla hesaplanmaktadır[77,107].

$$CH(K) = \frac{BSS(K)/(K-1)}{WSS(K)/(n-K)}$$

Formülde geçen WSS(k) kümeler içi kareler toplamıdır ve aşağıdaki gibi hesaplanmaktadır[77,107].

$$WSS(K) = \frac{1}{2}\sum_{i=1}^{K}\sum_{i,j\in C_i} d(i,j)$$

BSS(K) değeri ise kümeler arası kareler toplamını ifade etmektedir ve

$$BSS(K) = \frac{1}{2}\sum_{l=1}^{K}\sum_{\substack{i\in C_L \\ j\notin C_l}} d(i,j)$$

Formülü yardımıyla elde edilmektedir.

Farklı *K* değerleri için hesaplanan *CH* indeks değeri içerisindeki en büyük değere sahip *K* değeri, en uygun küme sayısı olarak belirlenmektedir[77,107].

Örneğin, Tablo 2.1.4.14.1'deki gibi 5 birim için Öklid uzaklıkları verilmiş olsun.

Tablo 2.1.4.14.1. Beş birim arasındaki Öklid uzaklık değerleri

	1	2	3	4	5
1	0	30	42	21	37
2	30	0	52	35	20
3	42	52	0	40	15
4	21	35	40	0	10
5	37	20	15	10	0

Tablo 2.1.4.14.1'ye göre *CH* indeksi hesaplanmak istensin. (1, 2) nolu gözlemler bir küme ve (3, 4, 5) nolu gözlemler ise diğer bir küme olsun. Uygun *K* değeri aşağıdaki gibi seçilir.

$WSS(K=2)=\frac{1}{2}[\sum_{i=1,2}\sum_{j=1,2} d(i,j) + \sum_{i=3,4,5}\sum_{j=3,4,5} d(i,j)]$

İlk küme için,

$\sum_{i=1,2}\sum_{j=1,2} d(i,j)$=(30+30)=60

İkinci küme için,

$\sum_{i=3,4,5}\sum_{j=3,4,5} d(i,j)]$=(40+45)+(40+45)+(15+10)=195

Böylece $WSS(K=2)=\frac{1}{2}[60 + 195] = 127{,}5$ olarak elde edilir.

$BSS(K=2)=\frac{1}{2}\left[\sum_{i=1,2}\sum_{j=3,4,5} d(i,j)\right]$

=[(42+21+37)+(52+35+20)+(42+52)+(21+35)+(37+40)]/2

=414/2=207

Buradan $CH(K=2)=\frac{207/(2-1)}{127{,}5/(5-1)} = 6{,}493$

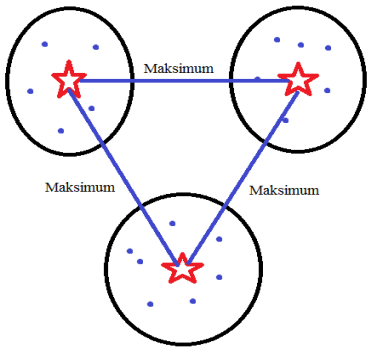
Benzer şekilde örneğin *K*=3 için de *CH* kriteri hesaplanır. *K* değeri için 2' nin mi yosa 3' ün mü tercih edileceğine karar verebilmek için *K*=3 alındığında elde edilen *CH* kriterine bakılır ve 6,493 değeri ile karşılaştırılır. Eğer *K*=3 alındığında elde edilen *CH* kriteri daha büyük ise, *K* değeri olarak üç değerinin alınması tercih edilmelidir. *K* değerinin belirlenmesinin ardından bilinen *K*-ortalama yöntemi uygulanabilmektedir[77,107].

2.1.4.15. Farthest First (En uzak İlk) Kümeleme Algoritması

En uzak ilk kümeleme algoritması Hochbaum ve Scmoys tarafından 1985 yılında geliştirilmiştir. Bu algoritma *K*-ortalama yöntemine benzerdir. Yöntem merkez noktaları seçerek, nesneleri kümelere atamaktadır. Ancak bu işlemi uzaklıkları maksimum yaparak gerçekleştirmektedir. Başlangıç çekirdek noktaları, ortalama değerlerden en büyük uzaklığa

sahip değerdir. Başlangıç çekirdekler ve *K* küme sayısı belirlenmesinin ardından kümeleme işlemleri yapılmaktadır[108,109].

En uzak ilk kümeleme yöntemi *K*-merkezin belirlenmesi problemini çözmektedir ve büyük veri setleri için de oldukça etkili bir kümeleme algoritmasıdır. Algoritma merkezleri hesaplamak için ortalamaları bulmamaktadır. Merkezleri keyfi olarak ele alır ve merkezler arasındaki uzaklıkların maksimum olması sağlanmaktadır[108,109].



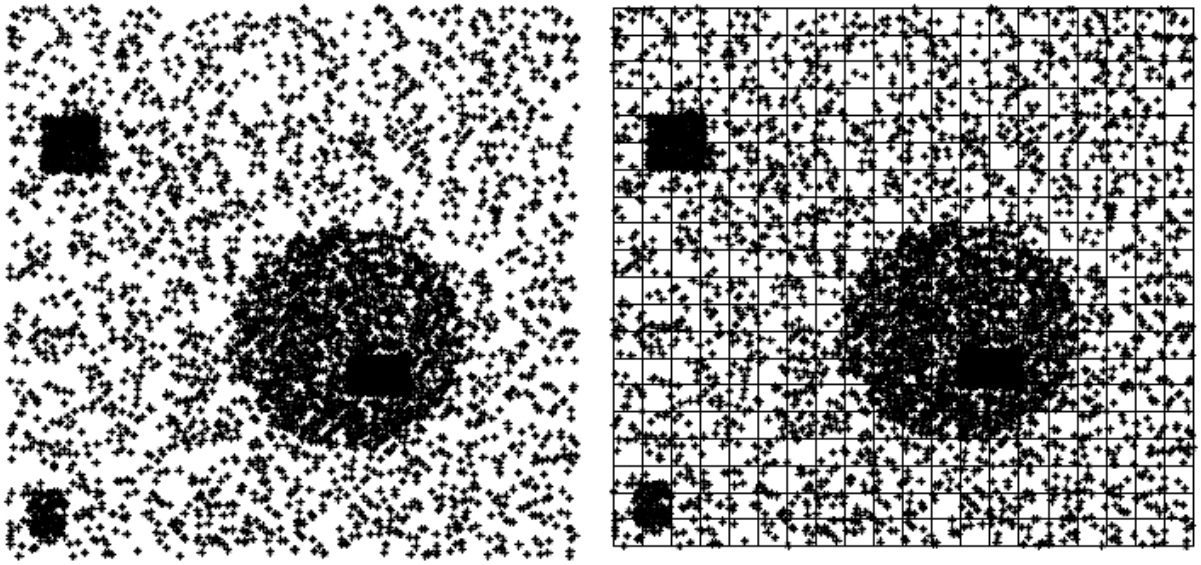
Şekil 2.1.4.15.1. *En uzak ilk* kümeleme algoritmasında merkezler arası uzaklık

En uzak ilk kümeleme algoritmasının zaman karmaşıklığı *O(nk)*'dır. Burada *n* veri setindeki gözlemlerin sayısı iken, *K* arzu edilen kümelerin sayısıdır. Geliştirilen bu kümeleme algoritması oldukça hızlıdır ve veri madenciliği uygulamalarında büyük ölçekli veriler için de uygundur[108,109].

2.1.5. Izgara Tabanlı Kümeleme Algoritmaları (Grid-based clustering algoritms)

Izgara tabanlı kümeleme algoritmaları, büyük hacimli çok boyutlu veri setlerinde oldukça etkili yöntemlerdir. Bu algoritmalar, veri uzayını sonlu sayıda hücrelere sahip ızgara yapısına bölmektedir. Bölgelere karşılık gelen kümeler, çevrelerinde yer alan noktalara göre daha yoğundur. Izgara tabanlı yöntemlerin en önemli avantajı zaman karmaşıklığı özellikle büyük veri setleri için önemli ölçüde azaltmasıdır. Direk olarak noktalarını kümelemek yerine, hücreler tarafından temsil edilen veri noktalarının komşu sınırlarları kümelendirilmektedir. Birçok uygulamada hücre sayısı veri nokta sayısından önemli ölçüde düşük olduğu için bu algoritmaların performansı oldukça iyidir[110].

Şekil 2.1.5.1'de orijinal veri değerleri ile ızgaralara bölünmüş şekilde elde edilen veri yapısı gösterilmektedir[110].



Şekil 2.1.5.1. Orijinal veri noktaları ile ızgara yapısına dönüştürülmüş veri yapısı

Izgaraya dayalı kümeleme algoritmaları 5 adımdan oluşmaktadır[110].

1. Veri uzayı sonlu sayıda hücreye bölünerek ızgara yapısı oluşturulur.
2. Her bir hücre için hücre yoğunluğu hesaplanır.
3. Hücreler yoğunluklarına göre sıralanır.
4. Küme merkezleri belirlenir.
5. Komşu hücrelere geçilir.

Hücreleri sıralamak ve küme merkezlerini seçmek için hücre yoğunluklarından yararlanıldığı için birçok ızgara tabanlı kümeleme algoritması yoğunluk tabanlı olarak ele alınabilmektedir. Ayrıca bazı ızgara tabanlı kümeleme yöntemleri, hiyerarşik kümelemeyi de kombine etmektedir. Izgara tabanlı kümeleme yöntemleri aşağıdaki özelliklere karşı duyarlıdır[110].

1. *Düzensizlik (Uniformity):* Düzensiz veri dağılımları içim küme kalitesinin istenilen seviyede olabilmesi için tek düze bir ızgara yapısı kullanmak yeterli olmayabilir.
2. *Yersellik (Locality):* Veri noktalarının şeklinde ve dağılımda lokal varyasyonlar var ise ızgara kümeleme yönteminin etkinliği, daha önceden belirlenen hücre sayısı, hücre sınırları ve önemli hücreler için yoğunluk eşik değeri ile sınırlandırılır.

3. *Boyutluluk (Dimensionality):* Performans ızgara yapısının genişliğine bağlı olduğundan boyut arttıkça ızgara yapısı da önemli derecee artacaktır ve yüksek boyutlu veriler için kümeler ölçeklenebilir özelliğine sahip olmayacaktır.

Düzensizlik olması durumunda adaptive ızgara yöntemlerinden olan MAFIA algoritması kullanılabilir. Yersellik sorunu olduğunda ise eksen döndürme yöntemlerinden olan NSGC, ADCC, ASGC ve GDILC yöntemleri önerilirken, yüksek boyutluluk sorununda CLIQUE, MAFIA, ENCLUS yöntemlerinin kullanımı önerilmektedir[110].

2.1.5.1. GRIDCLUS Kümeleme Algoritması (Grid-Based Clustering)

Schikuta ve arkadaşları ilk ızgaraya dayalı hiyerarşik kümeleme algoritması olan GRIDCLUS'u geliştirmişlerdir. Algoritma veri uzayını *d* boyutlu hiper dikdörtgenlere veya bloklara bölmektedir. Veri noktaları, *d* boyutlu uzaydaki noktalar olarak ele alınmaktadır. Veriler bloklara atandığında, kümeleme komşu arama algoritmaları tarafından yapılır. GRIDCLUS veri noktalarını ızgara yapısındaki bloklar içerisine yerleştirmekte ve ortaya çıkan bloklardaki yoğunlukları hesaplamaktadır. Ardından bloklar küme merkezleri olarak tanımlanır ve kalan kümeler ise bloklar üzerinde komşu araştırması kullanılarak oluşturulur. Küme merkezlerinden başlayarak sadece bloklar kümeler halinde birleştirilir. Komşu aranması ardışık olan küme merkezlerinden başlamaktadır. Kümeye ilave edilecek bitişik bloklar kontrol edilmektedir. GRIDCLUS algoritması, en kötü durumdaki hiyerarşik algoritmalardan bile daha fazla zaman karmaşıklığı içermektedir[111].

2.1.5.2. BANG Kümeleme Algoritması

BANG algoritması, Schikuta ve Erhort tarafından geliştirilmiştir. BANG algoritması, GRIDCLUS algoritmasının bir uzantısıdır. Algoritma, GRIDCLUS algoritmasının ızgara büyüklüğü açısından meydana gelebilecek bazı yetersizliklere çözüm getirmektedir. BANG algoritmasında GRIDCLUS'a benzer olarak veri noktalarını bloklar halinde yerleştirmektedir. BANG, blokları muhafaza etmek için BANG yapısı denilen ızgara yapısının bir çeşidini kullanmaktadır. Blokları kümeleme de yine komşu arama ve blokları yoğunluklarına göre sıralama işlemlerinden yararlanılmaktadır. Izgara yapısının saklanması için ikili ağaç kullanılmaktadır. Böylece komşu arama işlemi daha etkili bir şekilde yapılmaktadır. Bu ızgara dizinindeki ağaç ve sıralanmış blok yoğunluklar yardımıyla dendogram hesaplanmaktadır. Kümeleme aşamasında, küme merkezleri en yoğun bloklardır. GRIDCLUS ve BANG

algoritmaları iç içe geçmiş kümeleri ayırt etmede etkili olmasına rağmen, BANG algoritması büyük veri setleri için daha etkili sonuçlar üretmektedir[111].

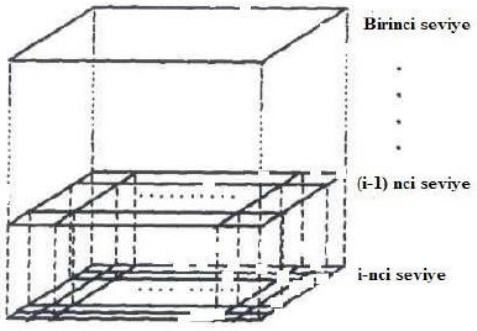
BANG algoritmasının işlem adımları aşağıdaki gibi özetlenebilir[111].

1. Dikdörtgen bloklar halinde veri uzayını bölümlenir.
2. Yoğun bloklar için ikili ağaç oluşturulur Burada bölme seviyesi ağacın düğüm derinliğine karşılık gelmektedir.
3. Yoğunluk indekslerinin hesaplandığı ve bu yoğunluklara göre azalarak sıralanan dendogram hesaplanır.
4. En yüksek yoğunluk indeksinden başlanarak, tüm komşu blokları belirlemek ve azalan sırada sınıflandırılır ve dendogramda bulunan bölgeler orjjinal blokların sağına yerleştirilir.
5. Kalan bloklar için 4. adım tekrarlanır.

2.1.5.3. STING Kümeleme Algoritması (Statistical Information Grid-Based Clustering Method)

Bölge odaklı sorguları kolaylaştırmak ve yersel veri tabanlarını kümelemek için Wang ve arkadaşları tarafından STING algoritması önerilmiştir. STING, mekânsal bölgeyi dikdörtgen hücrelere bölmekte ve hücreleri hiyerarşik ızgara yapıda ağaçlarda saklamaktadır[111].

Hiyerarşik yapının temelinde, genellikle farklı seviyelerde hücreler vardır. Üst seviyelerdeki her hücre bir sonraki alt seviyedeki hücreleri oluşturmak için bölünmektedir. Üst seviyedeki (esas) hücre veri uzayına karşılık gelmektedir. Alt seviyelerdeki hücrelerin büyüklüğü ise nesnelerin yoğunluğuna bağlıdır. STING algoritmasının hiyerarşik yapısı Şekil 2.1.5.3.1' de iki boyutlu uzayda görünmektedir[30,112].



Şekil 2.1.5.3.1. STING kümeleme algoritmasında hiyerarşik yapı

Her bir seviyedeki hücreler 4 çocuk (child) hücreye bölünmektedir. Her çocuğa karşılık gelen ana hücrenin çeyreğine denk gelmektedir. Ana hücre kendi çocuklarının birleşiminden oluşmaktadır. Seviye 1'in kök hücresi, tüm mekânsal alana karşılık gelmektedir. Her bir hücrede istatistiksel bilgiler saklanmaktadır. Bu parametreler aşağıdaki gibidir[111].

n: hücrelerdeki nesne sayısı
mean: her bir boyuttaki hücrelerdeki nesnelerin ortalaması
std: her bir boyuttaki hücrelerdeki nesnelerin standart sapması
min: her bir boyuttaki hücrelerdeki nesnelerin minimumu
max: her bir boyuttaki hücrelerdeki nesnelerin maksimumu
dist: hücrelerdeki nesnelerin dağılımı

STING, hiyerarşik ağaçta her bir hücre için özet istatistikleri saklamaktadır. Üst seviyedeki hücrelerin istatistiksel parametreleri, kolayca çocuk hücrelerin parametrelerinden hesaplanabilmektedir. Dağılım tipleri normal, tek düze, üssel olabilir. Dağılım türü bilinmediğinde ise nesne hiçbiri olarak atanmaktadır. İstatistiksel parametre ölçümleri herhangi bir yaprak düğümü için aşağıdan yukarıya olacak şekilde hesaplanır. STING algoritması kümeleme süreci için yukarıdan-aşağıya doğru bir yaklaşım benimser. Hiyerarşik ızgara yapısı ağacın kökünden başlayarak sorgulama yapmaktadır. [30,111]

STING algoritmasının işlem adımları aşağıdaki gibi özetlenebilir[111].

1. Başlamak için bir seviye belirlenir.
2. Bu seviyenin her bir hücresi için güven aralıkları hesaplanır.
3. Yukarıdan hesaplanan aralıkta hücreler ilişkili veya ilişkisiz olarak etiketlendirilir.
4. Eğer ilgilenilen seviye, yaprak seviye ise 6. adıma gidilir, değil ise 5. adıma gidilir.
5. Hiyerarşik yapıda bir alt seviyeye inilir. Yüksek seviyede ilişkili hücrelerden oluşan hücreler için 2. adıma gidilir.
6. Sorgunun özellikleri karşılandığında 8.adıma gidilir, diğer durumda 7.adıma gidilir.
7. Bu veriler ilgili hücrelere girene kadar döndürülmektedir. 9.adıma gidilir.
8. İlgili hücrelerin bölgeleri bulunur ve 9. adıma gidilir.
9. İşlem sonlandırılır.

STING algoritmasında *O(N)* zamanında ağaç oluşturulur. Burada *N* veri noktalarının toplam sayısıdır. Eğer ağaç hücrenin *K* yaprağı var ise, o zaman STING algoritmasının karmaşıklığı

O(K)'dır. Tüm ızgaraya dayalı algoritmalarda olduğu gibi STING algoritmasının da yüksek boyutlu verilerde bazı problemleri vardır. Ancak ızgara yapısının ölçeklenebilirliği bu problemler arasında değildir. STING algoritmasının birçok avantajı vardır. Bunlardan en önemlisi veri setini bir kere taramasıdır. Bunun yanı sıra yöntem istatistiki bilgileri sorgulamadan bağımsız olarak verilerin özet bilgisini vermektedir. STING algoritmasının hesaplama karmaşıklığı *O(K)*'dır ve $K<N$ olduğunda özellikle büyük veri setleri için etkili sonuçlar üretmektedir. Algoritma paralelleştirilebilir ve güncelleme işlemlerini hızlıca yapabilmektedir[111].

STING algoritmasından elde edilen küme sınırları ya yatay ya da dikey olmaktadır. Köşegen sınırlı kümeler oluşmamaktadır. Bu nedenle teknik hızlı işleme süresine sahip olmasına rağmen oluşan kümelerin kalitesini ve doğruluğunu azaltmaktadır. Bu durum ise algoritmanın dezavantajı olarak kabul edilmektedir[30,113].

2.1.5.4. WaveCluster Kümeleme Algoritması

Sheikholeslami ve arkadaşları dalga dönüşümünü kullanan (wavelet transform) ızgaraya dayalı, yoğunluk tabanlı bir kümeleme yaklaşımı önermişlerdir. Bu algoritma, veri noktalarına dalga dönüşümü uygular ve dönüşüm uygulanan veriler kullanılarak kümeler bulunur. Dalga dönüşümü, sinyal işleme tekniğidir ve sinyalleri farklı frekans alt-bantları şeklinde ayrıştırır. Dalga dönüşümü kullanmanın ana düşüncesi, veri noktalarını *d*-boyutlu sinyal olarak almaktır. *d* ise, boyut sayısını göstermektedir. Dalga dönüşümü uygulanarak orijinal özellik uzayına transformasyon yapılır ve yoğun bölgeler bu dönüşüm yapılmış uzayda aranmaktadır. Yüksek frekanslı sinyaller küme sınırları gibi daha seyrek bölgelere karşılık gelirken, düşük frekanslı sinyallerin parçaları, küme içi gibi daha yoğun bölgelere karşılık gelmektedir[114,115,116].

Veriler, çözünürlüğün farklı seviyelerindeki nesneler arasındaki göreceli mesafeleri korumak için dönüştürülmektedir. WaveCluster algoritması, girdi parametresi olarak her bir boyut için ızgara hücre sayılarını, dalgayı ve dalga transformasyonu sayısını kullanmaktadır. Algoritmanın adımları aşağıdaki gibi özetlenmektedir[114,115].

1. İlk olarak özellik uzayı bölünür, birimlere nesneler atanır ve birim özetleri hesaplanır.
2. Özellik uzayına dalga dönüşümü uygulanır.

3. Her bir seviyede dönüştürülmüş özellik uzayının altbantlarındaki bağlantılı bileşenler (kümeler) bulunur.
4. Birimler bağlı bileşenlerine etiketlenerek atanır.
5. Arama tablosu (lookup table) oluşturulur.
6. Nesneler kümelere haritalandırılır.

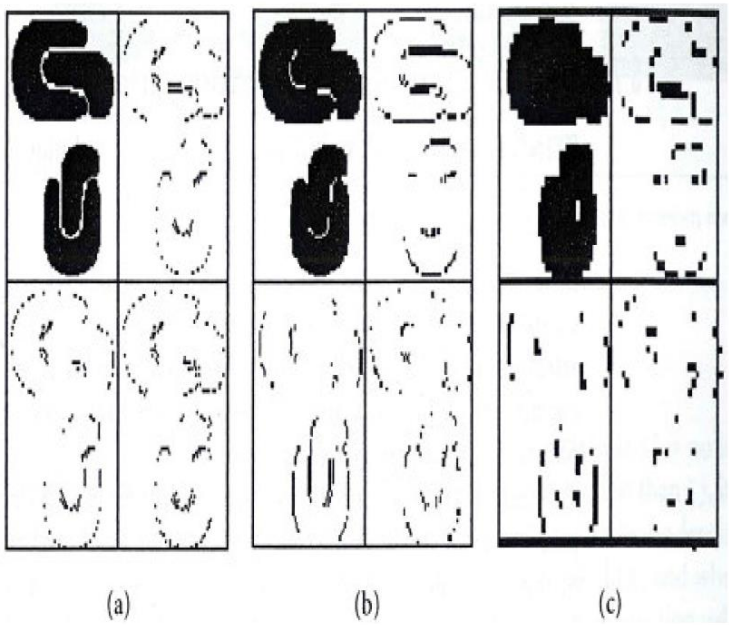
WaveCluster algoritmasının çeşitli avantajları vardır. Zaman karmaşıklığı *O(N)*'dir. *N* veri noktaları sayısıdır. Bu algoritmanın büyük veri tabanları için etkili olduğunu göstermektedir. Algoritma, veri grid sırasına ve uç değerlere karşı duyarsızdır. WaveCluster algoritması düzensiz şekildeki kümeleri de bulabilmektedir. Dalga dönüşümü, sonuçların çoklu seviyelerine olanak sağlamaktadır. Böylece kümeler daha doğru bir şekilde bulunmaktadır. Algoritma öncelikli olarak düşük boyutlu veriler için uygundur. Yüksek boyutlu veri olduğunda ise, temel bileşen analizi uygulanarak boyut sayısı azaltılmaktadır[114,115].

Wavelet dönüşümü kümeleme sürecinde birçok avantaj sağlamaktadır. Bunlardan ilki, denetimsiz kümelemeyi (unsupervisesd clustering) sağlamasıdır. Zayıf bilgiler küme sınırlarının dışına atılarak veri kümeleri dışındaki bölgeleri temizlemektedir.

Wavelet dönüşümünün çok çözümlü özelliği, doğruluğun farklı seviyelerindeki kümelerin bulunmasına olanak sağlamaktadır[114-116].

Şekil 2.1.5.4.1' de iki boyutlu nitelik uzayındaki bir örneği gösterir. Resim içindeki her bir nokta uzaysal veri kümesindeki bir nesnenin nitelik veya özellik değerlerini gösterir. Şekil 2.1.5.4.2' de ise farklı çözünürlükteki Wavelet dönüşüm sonuçları gösterilmektedir. Her bir seviye alt dört banda ayrılmıştır. Sol üst band her veri noktası üzerindeki ortalama komşuluğu, sağ üst band verinin yatay kenarlarını, sol alt band verinin dikey kenarlarını ve sağ alt band köşeleri vurgular [114-117].

Şekil 2.1.5.4.1. İki boyutlu nitelik uzayındaki bir örnek



Şekil 2.1.5.4.2. Farklı çözünürlükteki Wavelet dönüşüm sonuçları

WaveCluster algoritması hem ızgaraya dayalı hem de yoğunluğa dayalı bir algoritmadır. İyi bir kümeleme algoritmasının özelliklerini taşımaktadır. Büyük veri setleri için, düzensiz şekildeki kümeleri bulmada etkili sonuçlar üretmek ile beraber uç değerlere karşı da oldukça başarılı bir algoritmadır. Küme sayısı ve komşuluk yarıçapı gibi girdi parametrelerine ihtiyaç duymamaktadır. WaveCluster algoritması 20 boyuta kadar veri işleme özelliğine sahiptir. Ayrıca yapılan çalışmalarda WaveCluster algoritmasının BIRCH, CLARANS ve DBSCAN algoritmalarından daha yüksek performasna dsahip olduğu ve etkili sonuçlar ürettiği belirtilmiştir[116].

2.1.5.5. CLIQUE Kümeleme Algoritması (Clustering in QUEst, The Classical High Dimensional Algorithm)

Agrawal ve arkadaşları yüksek boyutlu niceliksel verilerin altuzaylarının kümelerini bulabilmek için hibrid yoğunluk ve ızgara tabanlı bir kümeleme algoritması önermişlerdir. Dolayısıyla bu yöntem yoğunluk tabanlı ve ızgara tabanlı kümeleme yöntemlerinin bir entegrasyonu olarak düşünülebilir. CLIQUE algoritması yüksek boyutlu uzayda alt uzayların kümelenmesi için önerilen ilk algoritmadır. Boyut-büyüme alt uzayı kümelenmesinde, kümeleme işlemi tek boyutlu alt uzayla başlamaktadır ve yüksek boyuta kadar artarak devam etmektedir. Kümeleme sonuçlarını daha basit bir yorumlanabilir formata getirebilmek için her bir küme için DNF (bölücü normal form, disjuntive normal form) minimal açıklaması verilmektedir[118,119].

CLIQUE algoritması, her bir boyutu ızgara yapıya böler ve hücrelerin ne kadar nokta sayısı içerdiğine bağlı olarak kümelere karar verir. CLIQUE kümeleme algoritmasının temel mantığı aşağıdaki ifadelere dayanmaktadır.

Çok boyutlu çok sayıda veri olduğunda algoritma genellikle veri alanı tek düze dağılmış veri noktalarından oluşmamaktadır. CLIQUE algoritması, uzaydaki seyrek ve yoğun bölgeleri tanımlamaktadır. Böylece veri setinin genel dağılım modeli keşfedilecektir. Toplam veri noktaları, başlangıçta belirlenen model parametresini aşıyorsa, o birimin yoğun olduğu kabul edilir. CLIQUE algoritmasında bir küme, yoğun birimlerin bağlı olduğu maksimum dizi olarak tanımlanır. CLIQUE algoritmasının işleyişi iki ana adımdan oluşmaktadır. İlk adımda veri uzayı üst üste çakışmayacak şekilde dikdörtgen biçiminde birimlere bölünmektedir ve bu birimler içinden yoğun birimler tespit edilir. Bu her boyut için yapılır. Alan budanarak parçalara bölündüğünde *k* boyutlu parça yoğun olduğu durumda *k-1* boyutlu parça incelenmektedir. Dolayısıyla *k-1* boyutlu parçada yoğunluk olmadığı durumda *k* boyutlu parçaya bakmaya gerek yoktur. Çünkü *k* boyutlu parçada yoğunluk olmamaktadır[119].

CLIQUE algoritmasının ikinci aşamasında, her bir küme için minimal bir tanımlama oluşturulmaktadır. Her bir küme için, yoğun birimlerin bağlı olduğu kümeyi kapsayan maksimal bölgeyi belirlemektedir. Ardından her bir küme için minimal kapsamı tespit etmektedir.

CLIQUE kümeleme yöntemi veri sıralamasına karşı duyarlıdır Ayrıca veri tabanı büyüklüğü arttıkça yani veri boyutu arttıkça daha iyi sonuçlar üretmektedir. Algoritmasının uygulaması basittir. Ancak elde edilen kümeleme sonuçları her zaman güvenilir değildir[30,118,119].

2.1.5.6. ENCLUS Kümeleme Algoritması (Entropy-based approach)

ENCLUS algoritması büyük ölçüde CLIQUE algoritmasına dayanmaktadır. Ancak, ENCLUS algoritması yoğunluk veya kapsama alanı gibi ölçümleri hesaplamak yerine entropiyi hesaplamaktadır. Entropi, rastgeleliğin, belirsizliğin ve beklenmeyen durumun ortaya çıkma olasılığını gösterir. ENCLUS kümeleme algoritmasında düşük bir entropi alt uzayı, yüksek yoğunluklu birimlerden oluşan bölgelere karşılık gelmektedir.

Bir alt uzayın kümelenebilmesi üç kritere dayanmaktadır. Bunlar sırasıyla kapsam, yoğunluk ve korelasyondur. Entropi yöntemi, bu üç kriteri de ölçmek için kullanılabilir. Hücre yoğunlukları arttıkça entropi değeride azalmaktadır. Belirli koşullar altında kapsama alanı (coverage) arttıkça entropi de azalmaktadır. Entropi yönteminde yüksek değerler, boyutlar arasında yüksek korelasyonu göstermektedir ve sıfır değer alması ise bağımsız boyutları ifade etmektedir[120-22].

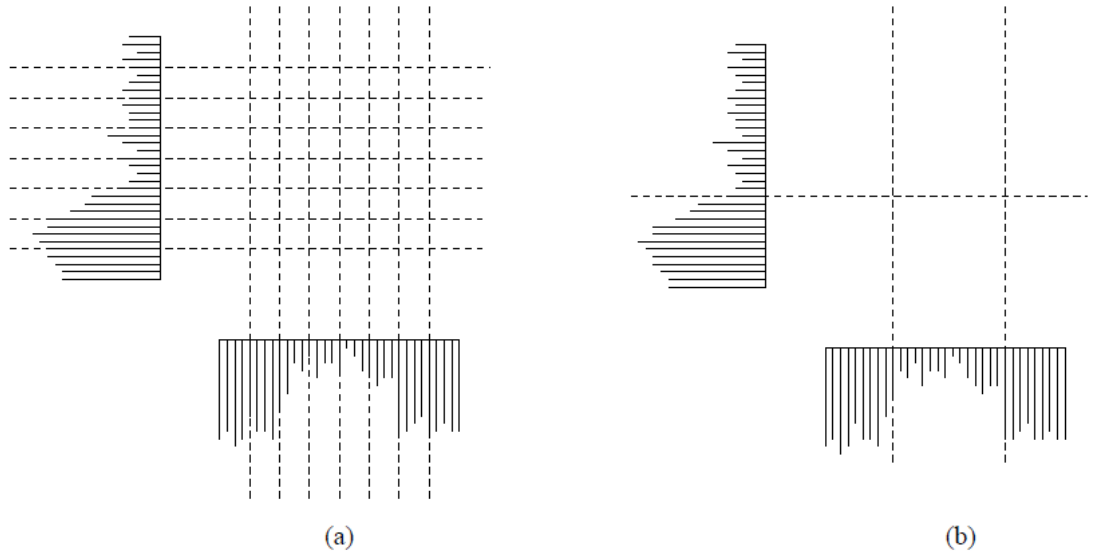
İlgilenilen alt uzaylar bulunduğunda, kümeler CLIQUE algoritmasındaki veya diğer var olan kümeleme algoritmaları gibi belirlenmektedir. ENCLUS kümeleme algoritmasının en önemli sınırlaması yüksek hesaplama maliyetine sebep olmasıdır özellikle alt uzayların entropilerini hesaplama açısından özellikle küçük yoğunluklu kümeleri belirlemek için maliyet artan bir duyarlılığa sahiptir[120-22].

2.1.5.7. MAFIA Kümeleme Algoritması (Merging of Adaptive Finite Intervals)

Bir algoritmanın hesaplama maliyeti yoğun hücrelerin belirlenmesine ve bu yoğun hücrelerin yüksek boyuta yayılmasına bağlı olarak değişmektedir. Goil ve arkadaşları 1999 yılında MAFIA kümeleme algoritması, CLIQUE algoritmasının bir uzantısı olarak önermişlerdir. Bu yöntemde boyutlar parçalanırken verinin dağılımına bağlı olan uyarlanabilir aralık genişliği (adaptive interval size) önerilmektedir. Veri tabanı başlangıçta bir kere taranarak histogram oluşturulur ve bu histogram kullanılarak bir boyut için minimum kutu sayısına karar verilir. Benzer histogram değerleri olan bitişik kutular daha büyük kutuları oluşturmak için kombine edilmektedir. Böylece boyut veri dağılımına bağlı olarak bölünmektedir. Uyarlanabilir aralık genişliği kümeleme sonuçlarının kalitesini artırmaktadır[123-4].

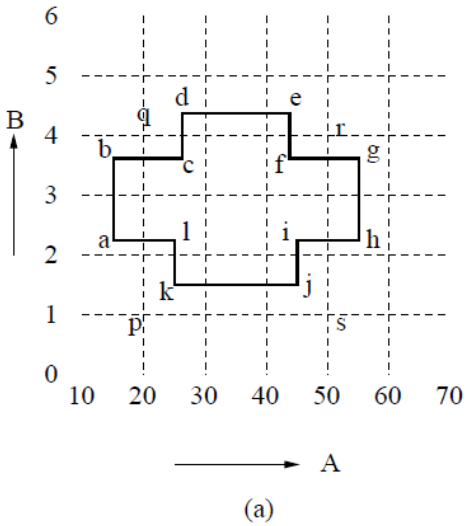
MAFIA kümeleme algoritması sabit genişlikte ızgara yapısı kullanmak yerine kümeleme sonuçlarının etkinliğini artırabilmek için uyarlanabilir ızgara yapılarını kullanmaktadır.

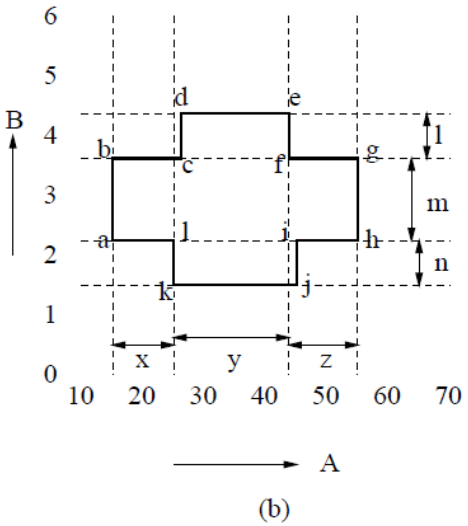
Aşağıdaki şekilde tek düze ve uyarlanabilir ızgara yapılarını göstermektedir Şekil 2.1.5.7.1 CLIQUE algoritmasında kullanılan tek düze ızgara yapısını örneklendirmektedir. Ancak bu yapı veri dağılımından bağımsızdır ve her bir adımda süreç devam ederken uyarlanabilir ızgara yapısından daha çok sayıda yoğun aday birimleri üretmektedir. Kümele tanımlarının daha uygun olabilmesi için süreç sonunda CLIQUE algoritması kümelerin minimal küme uzunluklarını tanımlar. Dolayısıyla karmaşıklık artacak ve sonuçlanan kümelerin doğruluğu azalacaktır[123-4].



Şekil 2.1.5.7.1. a)Tek düze ızgara yapısı **Şekil 2.1.5.7.1.** b)Uyarlanabilir ızgara yapısı

MAFIA algoritması uyarlanabilir ızgaraları kullandığı için küme sınırları minimal DNF tanımlarıdır ve küme sınırlarını daha doğru bir şekilde bulmaktadır. Şekil 2.1.5.7.2'de iki boyutlu bir uzayda kümeler tanımlanmaktadır. CLIQUE algoritması ile tanımlanan küme *pqrs*, küme sınırlarını kaybetmektedir. MAFIA ızgara sınırlarını kümenin sınırlarının çok yakınında oluşturmaktadır ve böylece birçok kümeyi kendi tanımından hareketle birleştirmektedir. *abcdefghijkl* kümesinin DNF ifadesi *(l,y)ʌ(m,z) ʌ(n,y) ʌ(m,x) ʌ(m,y)* olarak gösterilmektedir[123-4].





Şekil 2.1.5.7.2. a) CLIQUE kümeleri **Şekil 2.1.5.7.2.** b) MAFIA kümeleri

Yoğun aday birimin (CDU) en yüksek boyutu p, veri noktalarının sayısı N ve m bir sabit olmak üzere MAFIA algoritmasının hesaplama karmaşıklığı $O(m^p+pN)$'dir. Ancak gerçek veri setlerinde uyarlanabilir ızgara kullanan ve ilgili CDU'dan daha küçük bir set oluşturan MAFIA algoritmasının çalışma hızı CLIQUE algoritmasından 40 ila 50 kat daha fazladır[123-4].

2.1.5.8. NSGC Kümeleme Algoritması (New Shifting Grid Clustering Algorithm)

NSGC algoritması, eksen kayması (axis-shifting) ızgaraya dayalı bir algoritmadır. Izgaraya dayalı kümeleme algoritmalarının etkinliği, daha önceden belirlenen ızgara genişliğine, hücre sınırlarına ve yoğunluk eşik değerine bağlıdır. Izgaraya dayalı bu algoritmanın bağımlılıklardan kaynaklanan dezavantajlarını gidermek amacıyla eksen kayması algoritmaları geliştirilmiştir. Bu algoritmaların en önemlilerinden biri de NSGC algoritmasıdır[119,124].

Sabit ızgara yapısı, sınır etkilerden etkilenmektedir. Bunu elimine etmek için de Ma ve Chow algoritması hem yoğunluğa dayalı hem de ızgaraya dayalı bir kümeleme algoritmasıdır. Algoritma, uzayın her bir boyutunu eşit sayıda aralıklara bölmektedir. NSGC, tüm ızgara yapısını kaydırır ve yoğun hücrelere karar verebilmek için, orijinal ızgaralar ile birlikte kaydırılmış ızgara yapısını da kullanmaktadır. Bu sayede hücre genişlikleri ve sınırlarının etkisi de azaltılmış olunur. Daha sonra noktaları kümelemekten ziyade hücreleri kümelemektedir[119,124].

NSGC algoritması dört ana adımdan oluşmaktadır[119,124].

1. *Hücre oluşturma: w* iterasyon sayısı olmak üzere, uzayın her bir boyutu *2w* kadar aralıklara bölünür.
2. *Hücre atama:* Bir hücreye ait olan veri noktaları ilk olarak bulunur, daha sonra boyuta karşılık gelen yarım hücre genişliklerine kaydırılır ve kaydırılan hücrelere ait olan veri noktaları bulunur.
3. *Hücre yoğunluğu hesaplama:* Tanımlayıcı yoğunluk profilini elde edebilmek için, hem hücrelerin kendi yoğunlukları hem de en yakın komşu hücrelerin yoğunlukları kullanılır.
4. *Grup kümeleme:* Dikkate alınan hücre veya bu hücrenin komşu hücresi hiçbir gruba atanmaması ile başlar. Diğer durumlarda tüm boş olmayan hücreler atanana kadar sonraki hücre dikkate alınır.

Belirlenen kabul edilebilir hata küçük bir değer alana kadar yukarıdaki adımlar tekrarlanır. NSGC algoritmasının hesaplama karmaşıklığı $O((2w)^d)$'dir. Burada *d* çok boyutluluğu, *w* algoritmadaki iterasyon sayısını göstermektedir. Algoritmanın parametrik olmadığı iddia edilse de, algoritmanın performansı iterasyon sayısının seçimine, *w*'ya ve kabul edilen eşik hata değerine bağlıdır. *w* değeri çok düşük (veya büyük) veya eşik hata değeri çok yüksek olduğunda, kümeleme sonuçları doğru olmayacaktır. Ancak spesifik veriler için bu parametrenin en iyi değerlerinin bilinebilmesi için önsel bir yol yoktur. Hücre genişliği azaldıkça (ve iterasyon sayısı arttıkça) toplam hücre sayısı ve sonuçlanacak küme sayısı da artacaktır. Sonuçlanan kümeler çok küçük olabilir ve orijinal verideki kümelere karşılık gelemeyebilir. NSGC algoritmasının en güçlü avantajı, kaydırılan ızgara stratejisini kullandığı için keyfi şekillerin kümelerini bulmada oldukça doğru sonuçlar vermesidir[119,124].

2.1.5.9. ADCC Kümeleme Algoritması (Adaptive Deflect and Conquer Clustering)

Izgaraya dayalı kümeleme algoritmasının genellikle kalitesi, daha önceden tanımlanan ızgara genişliğine ve yoğunluk eşik değerine bağlıdır. Bu değişkenlerin etkilerini gidermek için Lin ve arkadaşları yeni bir ızgaraya dayalı kümeleme algoritması olan ADCC yöntemini geliştirmişlerdir. NSGC yöntemine benzer olarak ADCC algoritmasında daha önce tanımlanan ızgaraları ve önemli hücreleri belirlemede kullanılacak olan eşik değeri kullanmaktadır. Önemli olan yakındaki hücrelere yine bir küme oluşturmak için birleştirilebilir. Daha sonra, ızgaralar her yöne yarım hücre boyutunda döndürülür (deflect) ve

önemli hücreler yeniden tespit edilir. Sonuç olarak yeni üretilen hücreler ve başlangıç setindeki önemli hücreler birleştirilerek kümeleme sonuçları geliştirilir[119,124].

ADCC algoritmasının genel karmaşıklığı $O(m^d+N)$'dir. Burada m, her bir boyuttaki aralık sayısını, d verinin boyutunu, N ise veri nokta sayısını göstermektedir. ADCC yöntemi, NSGC algoritmasına ne kadar benzer olsa da, ADCC kümeleme algoritması kümeleri oluşturma şekline bağlı olarak biraz daha farklıdır. NSGC kümeleme algoritmasında iki ızgaranın komşuluğu bir kez incelenir. ADCC bunun yerine, iki ızgara her iki kümelemede de önemli çıkan hücreleri çakıştırarak yinelemeli olarak incelemektedir. Bu aşamada kümeleri ayırmada daha etkili bir yöntemdir. ADCC algoritması yalnızca m (boyut başına aralık sayısı) parametrenin seçimine bağlı olarak etkilenmektedir. ADCC kümeleme algoritmasının işlem adımları aşağıdaki gibi özetlenebilmektedir[119-124].

1. Başlangıç ızgara yapısı oluşturulur.
2. Önemli hücreler belirlenir.
3. İlk küme setleri oluşturulur.
4. Izgara yapısı transform edilir.
5. İkinci küme setleri oluşturulur.
6. Orijinal kümeler yeniden gözden geçirilir. İlk küme setleri ile ikinci küme setleri yinelemeli olarak kombine edilir.
7. En sondaki kümeleme sonuçları belirlenir.

2.1.5.10. ASGC Kümeleme Algoritması (Axis-Shifted Grid-Clustering)

Chang ve arkadaşları tarafından hücrelerin genişliklerinin ve sınırlarının etkilerini minimize etmek için önerilen alternatif bir kümeleme algoritmasıdır. Orijinal ızgara yapısı ve bu ızgaraya yapısına bağlı olarak başlangıç kümeleri oluşturulduktan sonra, ızgara yapısı her boyut için döndürülür ve yeniden kümeleme yapılır. Döndürülen bu ızgara yapısı keyfi bir uzaklık ile dönüştürülebilir. Bu dönüştürme işlemi orijinal hücre genişliğini değiştirmek için yapılmaktadır. Döndürülen ızgara yapısından elde edilen kümeler, orijinal kümeleri yeniden gözden geçirmede kullanılabilmektedir. ASGC kümeleme algoritması temelde yedi adımdan oluşmaktadır ve bu adımlar aşağıdaki gibi özetlenmektedir[119-124].

1. İlk ızgara yapısı oluşturulur.
2. Daha önceden belirlenen eşik değerden büyük olan hücre yoğunlukları, önemli hücreler olarak tanımlanmaktadır.
3. İlk küme seti oluşturulur. Tüm önemli komşu hücreler birleştirilir.
4. Izgara yapısı dönüştürülür. ξ gibi bir uzaklık yardımıyla orijin koordinatları kaydırılarak yeni bir ızgara yapısı oluşturulur.
5. İkinci küme setleri oluşturulur ve yeni kümeler ikinci ve üçüncü adımlar kullanılarak belirlenir.
6. Orijinal kümeler yeniden gözden geçirilir. Dönüştürülen ızgara yapısından elde edilen kümeler kullanılarak, orijinal ızgara yapısından oluşturulan kümeler yeniden incelenir.
7. En son kümeleme sonuçları oluşturulur.

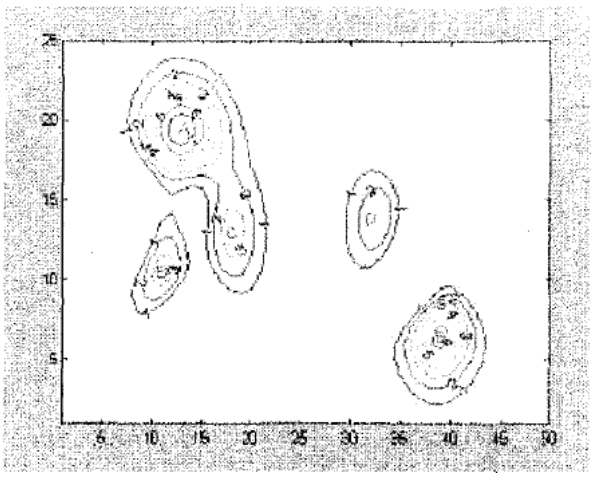
ASGC algoritmasının karmaşıklığı $O(m^d+N)$ olarak bilinmektedir. Burada N veri nokta sayısını, d verinin boyutunu, m ise her bir boyut için aralık sayısını ifade etmektedir.
ASGC kümeleme yöntemi, hücre büyüklüklerine ve hücre yoğunluk eşik değerine diğer eksen döndürme ızgara kümeleme yöntemlerine göre daha az duyarlıdır. Ancak yine bu yöntem de de hücre genişliklerinin ve hücre yoğunluk eşik değerinin seçimi çok önemlilik arz etmektedir[119,124].

2.1.5.11. GDILC Kümeleme Algoritması (A Grid-Based Density–Isoline Clustering Algorithm)

GDILC kümeleme algoritması ilk olarak Yonchang ve Junde tarafından geliştirilmiştir. Bu yöntemde yeni bir düşünce önerilmektedir. Kümeleme işlemleri yoğunluk-isoline (aynı çizgi üzerinde) kullanılarak yapılmaktadır. Algoritmanın uygulanabilmesi için tüm veri örneklerinin normal dağılması gerekmektedir ve tüm değişkenler [0,1] arasında değişen niceliksel değişkenler olmalıdır[125].

Yoğunluk-isoline kullanarak kümeleme işlemleri yapma düşüncesi çevrede var olan şekillerden, yoğunluğa ve ızgaraya dayalı kümeleme algoritmalarından gelmektedir. Bu sayede hangi yükseltinin eşik değerden yüksek olduğuna karar verilebilir. Veri örneklerinin dağılımlarının yoğunluklarından elde edilen yoğunluk-isoline şekili, kümeleri bulmak için kullanılmaktadır. Farklı isoline'lar seçilerek farklı yoğunlukta kümeler oluşturulabilmektedir. Örneğin Şekil 2.1.5.11.1'de eğer yoğunluk isoline değerini 5 olarak seçersek, küme *A* ve *E*'yi seçeriz, isoline değeri 4 olarak seçildiğinde ise *A, B, C* ve *E* kümeleri elde etmektedir. Ancak

eğer yoğunluk değeri ikiden az seçilirse *A* ve *C* kümeleri kombine edilerek tek bir küme oluşturur[125].



Şekil 2.1.5.11.1. Yoğunluk-isoline şekli

Yoğunluk-isoline kullanarak kümeleme işleminin yapılabilmesi için her iki veri veri örnek çiftleri arasındaki uzaklıkların ve bu uzaklıklarda türetilen eşik uzaklık değerinin hesaplanması gerekmektedir. Her bir veri çifti, α ve β olacak şekilde, eğer iki değerinde yoğunluğu, yoğunluk eşik değerinden büyük ve birbirleri arasındaki uzaklıklar, uzaklık eşik değerinden küçük ise bu iki veri çifti tek bir kümeyi oluşturabilir. Tüm veri çiftleri kontrol edilene kadar bu şekilde nokta çiftleri kombine edilir[125].

Genişlik eşik değerinden daha küçük genişliğe sahip kümeler gürültülü küme olarak adlandırılır ve bu örnekteki kümelerde yer alan örnekler gürültülü, sapan değer olarak ele alınır[126].

Yoğunluk-isoline kullanarak kümeleme yapmak için bazı parametrelerin tanımlanmasında fayda vardır. *X* örnek veri seti, *n* veri örnek sayısı, *Dist* uzaklık matrisi, *RT* seçilen komşu bölgenin genişliği, *DT* yoğunluk eşik değeri ve *coefRT* ve *coefDT* ayarlanabilir katsayılar olmak üzere[125].

1. Veri çiftleri arasındaki uzaklıklar hesaplanır ve *Dist(i,j)=D(X(i),X(j))* uzaklık matrisi elde edilir.
2. Komşu bölge genişliğine karar verilir.
3. Yoğunluk vektörü hesaplanır. *Dist* matrisinin her bir satırı için, *RT*'den küçük olan elementler sayılır. Hesaplanan sayı, satıra karşılık gelen veri örnek yoğunluğunun dağılımıdır. Yoğunluk vektöründen, yoğunluk-isoline şekli elde edilir.

4. Yoğunluk vektöründen, yoğunluk eşik değeri *DT* hesaplanır.
5. Kümeler kombine edilir. Eğer α ve β veri çiftlerinin yoğunlukları, *DT*'den büyük ve bu iki veri çifti arasındaki uzaklık *RT*'den küçük ise, iki veride aynı kümeye dâhil edilir. Tüm veri çiftleri ele alınana kadar bu işlemler devam eder. Tüm gözlem çiftleri kontrol edildikten sonra da sonuç kümeleri elde edilir.

Algoritma adımları gerçekleşirken veri çiftleri arasındaki uzaklıkların hesaplanması gerekmektedir. *n* örnek hacmi olmak üzere bu uzaklıklar n^2 kere hesaplanmaktadır ki *n* çok büyük olduğunda bu oldukça zaman alıcı bir işlem olacaktır. Ayrıca uzayda çok büyük uzaklık matrisleri gerekmektedir. Örneğin, bir uzaklık değerinin saklanması için 2 byte gerekiyorsa, toplam uzay için $2n^2$ byte gerekecektir. Eğer n=20000 ise uzayda 800 MB olmalıdır[126].

Daha önceden de belirtildiği gibi yoğunluk-isoline'larin kullanılabilmesi için veri çiftleri arasındaki uzaklıkların hesaplanması gerekmektedir ve zaman karmaşıklığı *O(N)*'dir. Veri örnek sayısı 50000 olduğunda uzaklık değeri $1{,}25*10^9$ kere hesaplanır.

GDILC kümeleme algoritmasında, her bir boyut *m* tane aralığa bölünür ($i=1, 2,...d$). Böylece tüm veri örnek uzayı *M* hiper-dikdörtgenlere parçalanmış olur ($M=\prod_{i=1}^{d} m_i$). Her bir boyuttaki aralıklar sıfırdan *m-1*'e kadar numaralandırılır. Her bir hücre, aralık sırası ike etiketlenir. Örneğin, C_{35} hüccresinde 1 boyutta 3 aralık, 2 boyutta 5 aralık vardır.$Ci_1, Ci_2, ... Ci_d$ *ve* $Cj_1, Cj_2, ... Cj_d$ hücreleri aşağıdaki koşullarda komşudur[126].

$$|i_p - j_p| \leq 1, \ (1 \leq p \leq d)$$

$i_1, i_2, ..., i_d$, $Ci_1, Ci_2, ... Ci_d$ hücrelerinin aralık sıra sayısı iken, $j_1, j_2, ..., j_d$ $Cj_1, Cj_2, ... Cj_d$ hücrelerinin aralık sıra sayısıdır.

α örneğinin yoğunluğu hesaplanmak istendiğinde sadece C_α ve komşu hücrelerinin örnekleri arasındaki uzaklıkları hesaplamak yeterli olacaktır. Diğer hücrelerde yer alan örnekler α gözleminden oldukça uzaktır, bu yüzden α yoğunluğuna hiçbir katkıları bulunmayacaktır. Dolayısıyla α gözlemi için C_α komşuluğunda yer almayan gözlemler dikkate alınmamaktadır[126].

GDILC kümeleme algoritmasının işlem adımları detaylı olarak aşağıda verilmektedir[125].

1. Hücreler başlatılır. Her bir boyut *m* eşit aralığa bölünür. Tüm veri uzayı m^d hücre içerisine parçalanmış olur. Her bir veri örneği koordinatlara bağlı olarak ait olduğu hücreye yerleştirilir.
2. Uzaklık eşik değeri *RT* hesaplanır. Her bir veri örneği için α ve diğer komşu hücre örnekleri arasındaki uzaklıklar hesaplanır. Bu uzaklıklar yardımıyla ortalama uzaklık hesaplanır ve *RT* uzaklık eşik değeri elde edilir.
3. Yoğunluk vektörü ve yoğunluk eşik değeri *DT* hesaplanır. Her bir veri örneği için, α'dan itibaren *RT* sınırları içerisinde olan noktalar sayılır. Yoğunluk vektöründen ortalama yoğunluk hesaplanır ve yoğunluk eşik değeri *DT* hesaplanır.
4. *DT* yoğunluk eşik değerinden büyük olan yoğunluklara sahip örnek çiftleri, küme olarak ele alınır. Ardından her bir veri örneği α, C_α komşuluğunda yoğunluk eşik değeri *DT*'den fazla olan her bir örnek kontrol edilir. Daha sonra ardışık olarak bulunan iki küme birleştirilir. Bu birleştirmeler, bütün örnek çiftleri gözden geçirilinceye kadar devam eder.
5. Gürültülü veriler çıkarılır. Anlamlı kümeler oluşturabilmek için birçok kümenin oldukça küçük olduğu ortaya çıkmaktadır. Belirli bir sayıdan küçük genişliğe sahip kümeler çıkartılır. Böylece GDILC algoritması sapan değerlerin etkisini etkili bir şekilde elimine etmektedir.

Yoğunluk- isoline kümeleri tespit etmek kolay olsa da, yoğunluk-isoline şeklini elde etmek o kadar kolay değildir. Bu şeklin çizimi *RT*'nin (uzaklık eşik değeri) seçimine bağlıdır. Eğer *RT* çok küçük seçilirse, her bir örneğin yoğunluğu oldukça küçük olacaktır. Böylece örnek yoğunlukları çok dar bir aralıkta yer alır ve verilerin dağılımının belirlenmesi zorlaşır. *RT* değeri çok büyük olduğunda ise, verilerin yoğunlukları fazla olacaktır ve yoğunlukların değerleri az değişiklik gösterecektir, elde edilen yoğunluk-isoline şekli veri örneklerinin dağılımını açıklayamayacaktır. Dolayısyla *RT* değeri *min(Dist)<RT<maks(Dist)* arasında değişmelidir. Yani *RT* değeri tüm uzaklıkların en küçüğünden büyük, en büyüğünden de küçük olmalıdır. RT değeri aşağıdaki formül yardımıyla hesaplanmaktadır[125].

$$\mathrm{RT}=\frac{mean(Dist)}{d-coefRT}$$

Formülde yer alan *Dist*, tüm ilgilenilen uzaklıkları içeren bir maris, *mean(Dist)* ise *Dist* matrisinde yer alan uzaklıkların ortalamasıdır. *D* boyut sayısını, *coefRT* ise ayarlanabilir bir katsayıdır. Kümeleme sonuçları tatmin edici olmadığında, daha iyi sonuçlar elde etmek için bu katsayı değeri değiştirilebilir. *coefRT* değerinin 20 olduğu durumlarda da kümeleme çalışmalarının iyi sonuç verdiği görülmüştür[125].

Sonuç kümelere *DT* yoğunluk değerine bakılarak karar verilmektedir. Eğer bu değer küçük ise, komşu kümeler tek bir küme olarak kombine edilmektedir. *DT* değerinin hesaplanabilmesi için aşağıdaki formülden yararlanılmaktadır[125].

$$DT=\begin{cases} 2 & n < 1000 \\ \frac{mean(Density)}{log_{10}(n)} * coefDT & n \geq 1000 \end{cases}$$

coefDT katsyaısı 0,7 ile 1 arasında değişmektedir. Ancak birçok araştırmadan görüldüğü üzere bu değer 0,95 olarak ayarlandığında kümeleme sonuçları oldukça başarılı çıkmıştır.

2.1.5.12. Izgaraya Dayalı *K*-ortalama Kümeleme Algoritması

Izgaraya Dayalı *K*-ortalama kümeleme yöntemi 2014 yılında Li Ma ve arkadaşları tarafından geliştirilmiştir. Klasik *K*-ortalama kümeleme algoritmasında başlangıçta küme sayısının belirlenmesi ve başlangıç küme merkezlerinin rasgele seçilmesi zor işlemlerdir. Bu özellikle kümeleme sonuçlarının durağan olmamasını neden olmaktadır. Aynı zamanda gürültülü verilere karşı oldukça hassasdırlar. Önerilen yöntemde uzay eşit ızgaralara bölünür ve her bir ızgarada veri noktaları sayılır. *M* sayıda grid seçilerek, maksimum veri nokta sayıları oluşturularak merkez nokta hesaplanır. *M* noktalarında, birbirinden en uzak *K* nokta ve *K*-ortalama kümeleme algoritmasının başlangıç küme merkezi olan *K* merkez noktalar bulunur. Aynı zamanda *M*'deki en büyük değer *K*'yı da içermelidir. Eğer ızgaradaki veri sayısı eşik değerden düşük ise, bu noktalar gürültü veri olarak belirlenir[127].

Geliştirilen ızgara tabanlı *K*-ortalama kümeleme algoritması, klasik *K*-ortalama yöntemine göre daha az iterasyonlarda daha kaliteli sonuçlar üretmekte ve iyi bir stabilitesi vardır[127].

Geliştirilen algoritmada ilk olarak ızgaralar boyunca veri noktalarının dağılımları kavranmaktadır. Ardından *K* değerleri DBSCAN algoritması ile hesaplanmaktadır. En son aşamada ise, ızgara yöntemi ile seçilen *K* başlangıç küme merkezleri seçilir ve gürültülü

veriler veri setinden çıkarılır. Bu özelliği ile de yöntem daha doğru ve stabil sonuçlar üretmektedir[127].

Geliştirilen ızgaraya dayalı *K*-ortalama yöntemini kavramak için bazı tanımlamaların yapılması gerekmektedir.

Izgara genişliği: Izgara kenar uzunluğu gerçek durum dikkate alınarak tanımlanmaktadır.
Gürültülü veri eşik değeri: Veri ızgarası eşik değerin altındadır ve eşik değer genellikle yapay olarak belirlenmektedir.
X ızgarasının veri merkezi: $X=(\sum_{i=1}^{n} x_i)/n$ (Burada *n* her bir ızgaradaki veri noktaları sayısıdır)
D uzaklığı: $D=\sqrt{\sum_{i=0}^{n}(x_i - x)^2}$ (Burada n verideki değişken değeri sayısıdır).
Amaç fonksiyonu J: $J=\sum_{i=1}^{k}\sum_{x_{\lambda \epsilon classi}}\sqrt{\sum_{j=1}^{q}(x_{\lambda,j} - x_{i,j})^2}$

Yukarıdaki formülde yer alan *i* sınıf sayısı, *λ* her bir sınıftaki verinin sayısı ve *j* ise değişken değerleri sayısıdır.
Doğruluk oranı W: $W=\frac{M}{N} * 100\%$ (*M* doğru sınıfa atanan veri sayıları, *N* ise toplam veri sayısıdır).

Klasik *K*-ortalama algoritması başlangıç merkezi olarak rasgele noktaları seçmektedir. Bu özellik ise kümeleminin stabil olmamasına sebep olmaktadır. Kümeleme sonuçları gürültülü noktalardan etkilendiği için bu noktaların sayısının azaltılması gerekmektedir. Bu sebeple aşağıdaki yöntemlerin uygulanması gerekmektedir[127].

1. *Gürültülü verileri çıkarmak:* Değişken değerlerine göre veri noktaları karşılık gelen ızgaralara atanır. Her bir ızgaradaki veri noktaları sayılır ve veri merkezi hesaplanır. Izgarada veri sayısı eşik değerden düşük olduğunda gürültülü veri mevcuttur. Bu gürültülü veriler kümelemede doğru sonuçlar elde edilmek için çıkarılır.
2. *K değerine karar vermek:* İlk aşamada veri ait olduğu ızgaraya atanmakta ve veri sayısı ve veri merkezi hesaplanmak idi. Izgara veri merkezi dışında kalanlar ise DBSCAN algoritmasında girdi verisi olarak kullanılacaktır. Böylece kümenin *K* değeri üretilir.
3. *Başlangıç merkezlerini belirlemek: K* değeri belirlendikten sonra, maksimum veri sayısı *M* ızgarası seçilir. *M* merkez noktasında birbirinden en uzak *K* noktalar bulunur

ve bu *K* merkez noktalar *K*-ortalama kümeleme yönteminde başlangıç küme merkezi olarak kullanınır. Aynı zamanda maksimum değer *M*, *K*'yı da içermelidir.

Izgara tabanlı *K*-ortalama yöntemi başlangıç merkez noktası seçimi, gürültülü verileri çıkarmaya karar verme ve küme sayısının belirlenmesi gibi *K*-ortalama yönteminin üç kavramı üzerinde durularak geliştirilmiştir. Yöntemde veri noktasına karşılık gelen ızgaraya atanarak başlangıç merkez seçimi problemi ortadan kaldırılır, gürültülü veriler atılarak gürültülü verilerden kaynaklanabilecek olumsuzluklar giderilir ve son olarak DBSCAN kümeleme algoritması ile *K* değerinin ne olması gerektiğine karar vermektedir. Klasik *K*-ortalama yöntemi ile karşılaştırıldığında bu yöntem başlangıç küme merkezlerinin seçiminde daha başarılıdır. Ayrıca gürültülü veriler çıkartılarak kümeleme sonuçlarının daha doğru bir şekilde elde edilecektir. Dolayısıyla önerilen bu yöntem ile klasik *K*-ortalama kümeleme algoritması geliştirilmiştir[127].

2.1.6. Kategorik Kümeleme Algoritmaları (Categorical clustering algorithms)

Birçok uygulamada veriler sayısal değişkenlerin yanında kategorik değişkenlerde içermekte veya sadece kategorik değişkenlerden oluşmaktadır. Özellikle yazılım programları ve protein etkileşim verileri kategorik veriler içerir. Kategorik verilerin varlığı çok sayıda kategorik kümeleme algoritmalarının geliştirilmesine neden olmuştur.

m tane değişken içeren kategorik veri setinde, *m* boyutlu küpler ele alınacaktır. Küpün bir hücresi kendi koordinatlarına eşit olan nesne sayıları ile haritalandırılmaktadır. Böyle bir küpte kümeler altuzaylar (subspaces) olarak adlandırılmakta ve düşük yoğunluklu nesneler altuzaylarla ayrılmaktadır. Ancak küpleri kümelemenin bazı sakıncaları vardır. Değişken değerlerinin bir sırası olmadığı için küp hücrelerinin de bir sırası yoktur. Yoğun altuzaylar araştırılıken en iyi kümenin tanımlanması için her bir boyutun sıralamaları dikkate alınabilir. Altuzayın yoğunluğu kullanıcı tarafından belirlenen yarıçap değeri ile tanımlanmaktadır. Ancak küpün farklı altuzayları için farklı yarıçaplar tercih edilmektedir. Yoğun bölgelerde daha düşük yarıçarp değeri ile çalışılırken, seyrek bölgelerde daha büyük yarıçaplar ile çalışılmalıdır[128].

Kategorik kümeleme algoritmalarının farklı uygulamalarda kullanılmasını sağlayan çeşitli özellikleri vardır. Bunlar aşağıdaki gibidir[128].

Ölçeklenebilirlik (scalability): Çalışma zamanı ve bellek gereksinimi aşırı gerektirmeden büyük veri setlerinin kümelendirilebilmesidir.

Dayanıklılık (Robustness): Diğer kalan nesnelerden ayrı olarak uç değerleri bulmada duyarlıdır. Uç değerler ise farklı popülasyondan alınan örnekleri ifade etmektedir.

Sıra Düzensizliği (Order Insensivity): Bir kümeleme algoritması girdi nesnelerin sıralarına karşı hassas olmamalıdır. Nesneleri yeniden sıralandırmak farklı kümelerin oluşmasına neden olmamalıdır. Her uygulamada sıra düzensizliği önemlidir. Çünkü bu durum sonuçların tekrarlanabilirliğini sağlamaktadır.

Minimum kullanıcı tarafından belirlenen girdiler: Başlangıçta belirlenen küme sayıları, parametreler küme sonuçlarını etkilemektedir.

Karmaşık veri tipleri: Bir veri seti hem kategorik hem de sayısal nitelikte değişken içerebilmektedir.

Kabul edilebilirlik: Nesneleri çoğaltmak ve kümeleme işlemi tekrarlamak sonuçları değiştirmemektedir.

Kategorik verilerin kümelenmesinde kullanılan çok sayıda yöntem mevcuttur. İlerleyen bölümlerde yaygın şekilde kullanılan kategorik kümeleme yöntemlerinin teorilerinden sırasıyla bahsedilecektir[128].

2.1.6.1. QROCK Kümeleme Algoritması (Quick-ROCK) kümeleme algoritması

QROCK kümeleme algoritması ROCK algoritmasının bazı dezavantajlı özelliklerini gidermek üzere ilk olarak 2005 yılında Dutta ve arkadaşları tarafından önerilmiştir. ROCK algoritması veri nesneleri arasındaki bağlantılara dayanan toplamalı bir kümeleme tekniğidir. Küme çiftleri arasında hiçbir bağlantı kalmayınca ve istenilen küme sayısı elde edildiğinde kümeleri birleştirme işine son verilmektedir. Herhangi bir bağlantı kalmayıncaya kadar kümeleri birleştirildiğinde elde edilen küme boş küme olacaktır. QROCK algoritması ise uzaklık matrisleri yerine bağlantı fonksiyonlarını kullanmaktadır. ROCK algoritması benzerlik ölçüsü olarak Jaccard katsayısından yararlanmaktadır. Ancak bu katsayı tüm kategorik değişkenlere aynı ağırlığı vermektedir. Oysaki uygulamalarda bir değişkenin kümelemeye katkısı diğer bir kategorik değişkenden çok daha fazla olabilmektedir.

Dolayısıyla bu gibi durumlarda değişkenlere aynı ağırlığı vermek kümeleme sonuçlarının hatalı olmasına sebep olacaktır. Bu dezavantajları giderebilmek için değişkenlerin ağırlıkları ile ilişki olarak geliştirilen aşağıdaki ağırlıklandırılmış benzerlik katsayısı kullanılmaktadır[129].

$$Sim(T_i,T_j)=\frac{|T_i \cap T_j|}{|T_i \cap T_j|+2\sum_{k \notin T_i \cap T_j}\frac{1}{|D_k|}}$$

$|D_k|$ değeri, üzerinde durulan veri setindeki *K.* alanın genişliğidir. QROCK algoritması, ROCK algoritmasından daha hızlı sonuç vermektedir ve aynı zamanda performansı daha iyi olan sonuçları üretmektedir. QROCK algoritması belirli bir eşik değere göre ağırlıklandırılmış bağlantı fonksiyonu yardımıyla kümeleri araştırırken, başlangıç küme sayısının belirlenmesine ihtiyaç yoktur. Ancak ROCK algoritmasında eşik değerinin yanında ayrıca başlangıçta küme sayısının kaç olacağına da karar verilmektedir. Örneğin küme sayısı (*K)* olması gereken sayıdan daha büyük belirlendiğinde, gereğinden fazla yanlış kümelerin oluşmasına yol açacaktır. Dolayısıyla küme sayısının yanlış belirlenmesinden kaynaklanabilecek hatalarda yeni geliştirilen bu algoritma yardımıyla elimine edilmektedir. QROCK bu avantajlarının yanı sıra aykırı gözlemlerin tespitinde de etkili bir yöntemdir[129].

2.1.6.2. STIRR Kümeleme Algoritması (Sieving Through Iterated Relational Reinforcement)

STIRR kümeleme algoritması, lineer olmayan dinamik sistemlere dayanan iteratif bir yöntemdir. Algoritmada veri tabanı grafik ile tanımlanmaktadır. Bu grafikte her bir değişkenin her bir farklı değeri ağırlıklandırılmış düğümler ile temsil edilir. Dolayısıyla eğer *p* değişken var ise, *i.* değişkenin domain genişliği d_i olmak üzere, grafikdeki toplam düğüm sayısı $\sum_i d_i$'dir. Başlangıç düğüm ağırlıkları tekniğe bağlı olarak rasgele olarak belirlenebilir. Her bir düğüm için iteratif olarak ağırlıklar güncellenerek yapılandırmada değişiklikler yapılır. Herhangi bir düğümün yeni ağırlığı birleştirici fonksiyona (combiner function) dayanarak hesaplanmaktadır[129-30].

Ağırlık modifikasyon işlemleri boyunca, ilişkili olan değerler birbirini etkilemektedir. Böylece kategorik değişkenler için tanımlanan herhangi bir benzerlik ölçüsüne gerek yoktur. STIRR kümeleme algoritması düğüm setlerini kümelemek için ağılıkların birden çok kopyasını saklamaktadır. Sabit bir noktaya erişildiğinde ağırlıklar her bir değişken için iki gruba ayrılır. İlk grup büyük pozitif ağırlıklar ve ikincisi küçük negatif ağırlıklardır. Büyük

pozitif ağırlıklı ve büyük negatif ağırlıklı düğümler, kümeleri belirlemek için gruplandırılır. Bu gruplar, kümeler için ön tahminlerdir. STIRR algoritması girdi veri setini niceliksel formata dönüştürmemektedir. Dolayısıyla verilerin doğal yapısını bozmamaktadır. Ayrıca seyrek dağılmış veri ile ilgili sorunları da önlemektedir. STIRR kümeleme yönteminde benzerlikler geçişlidir. Örneğin *A* ile *B* ve *B* ile *C* ilişkili ise o zaman *A*'da C ile ilişkili olur. Algoritma, doğrusal olmayan dinamik sistemlere dayalı bir sistem içinde, spektral bir grafik bölümleme tekniği kullanmaktadır. STIRR algoritmasının dezavantajı ise, bazı girdi parametreleri için birleştirme işleminde yanıltıcı sonuçlar vermesidir. Dolayısıyla bazı girdiler için döngü devam eder ve sonuç noktalara ulaşılamaz. Ayrıca her bir değişken için normalleştirme işlemini ayrı ayrı yapmaktadır. Bu değişkenlerin etki alanı içinde farklı değerleri sahip olduklarını ifade etmektedir. Değişkenlerin farklı değerler almasından daha büyük ağırlıklar elde etme eğilimine sahiptir[129-30].

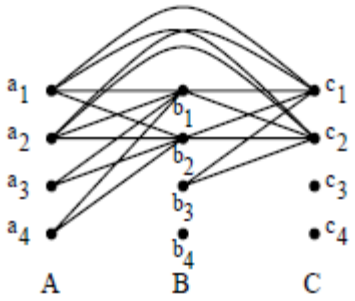
2.1.6.3. CACTUS Kümeleme Algoritması (Clustering Categorical Data Using Summaries)

CACTUS kümeleme algoritması, Ganti ve arkadaşları tarafından 1999 yılında kategorik yapıdaki değişkenler içeren verileri kümelemek amacıyla geliştirilmiştir. CACTUS algoritması özetleme, kümeleme ve doğrulama olmak üzere üç evreden oluşmaktadır. Kümeleme sürecinde, aday küme setlerinin bulunabilmesi için özet istatistikler kullanılmaktadır. Özetleme sürecinde veri setinden özet istatistikler hesaplanmaktadır. Doğrulama sürecinde ise, aday küme setlerinden gerçek küme setlerine karar verilmektedir. Algoritmada kümeler oluşturulurken özellikler arası (inter-attribute) ve özelikler içi (intra-attribute) özet istatistikler kullanılmaktadır. Kısaca bu terimleri inceleyecek olursak[129-131],

Özellikler arası özet istatistikler: Veri seti D ve $(a_i, a_j) \in D_iD_j$ olmak üzere herbir (a_i, a_j) değişken değer çiftleri için sayaç sıfırdan başlatılarak veri seti taranır. Her bir demek için $t \in D$, (tA_i, tA_j) çiftleri için sayaç değeri artırılmaktadır. D'nin taraması tamamlandığında ise eşik değerden küçük olan değerler için sayaç sıfırlanarak σ^* değeri hesaplanır. Böylece sadece güçlü bağlantılı çiftler korunmuş olacaktır. Güçlü bağlantılı çift sayısı genellikle $|D_i||D_j|$'den küçüktür. Örneğin 50 değişkenli ve her bir değişkenin 100 bireyden ölçüldüğü ve ana hafızada 100 MB yer olduğu varsayılsın. Her bir sayaç içinde 4 byte gerektirdiği varsayıldığında, sayaç 2500 $(= \frac{100*10^6}{100*100*4})$ kere tekrarlanır. Dolayısıyla 50 değişken için, 1225

değişken çifti değerlendirmek gerekmektedir. Veri seti tek bir kere taranarak tüm özellikler arası özetler hesaplanmış olacaktır[129-131].

Özellikler içi özet istatistikler: Benzer şekilde kategori sayısı az olan kategorik değişkenlerden yararlanılır ve dolayısıyla herhangi bir A_i özelliği için özellikler içi özet istatistikleri ana bellekte saklanmaktadır. $\sum_{ij}$ ana bellekte saklandığı için $\sum_{ii}^{j}$ 'yi hesaplamak oldukça hızlıdır. Ancak ilerideki anlatımlarsan da görüleceği üzere A_i ve A_j değişken çiftleri için sadece $\sum_{ii}^{j}$ değerini hesaplamak yeterli olacaktır. Şekil 2.1.6.3.1'de değişkenler arası (a) ve değişkenler içi (b) özellikleri gösterilmiştir[129-131].



A w.r.t. B	B w.r.t. C	C w.r.t. B
a_1, a_2:2	b_1, b_2:2	c_1, c_2:3
a_1, a_3:2	b_1, b_3:2	c_1, c_3:2
a_1, a_4:2	b_2, b_3:2	
a_2, a_3:2		
a_2, a_4:2		
a_3, a_4:2		

Şekil 2.1.6.3.1. a)Değişkenler arası **Şekil 2.1.6.3.1.** b)Değişkenler içi

CACTUS algoritması, sadece tek bir değişken çiftine dayanarak düğümlerin tahmin küme setlerini oluşturmaktadır. Veri setinde iki farklı değişkenin olduğunu ve bu değişkenlerin iki farklı değerleri olduğu varsayılsın. *A* değişken tipi a_i ve *B* değişken tipi ise a_j ile gösterilmek üzere a_i ve a_j' nin birlikte görüldüğü düğümler olabilmektedir. Veri setindeki bu iki değeri, birlikte görüldükleri düğümlerin oranı desteklemektedir. Eğer bu değer, daha önceden belirlenen eşik değeri aşar ise, bu iki değer güçlü bir şekilde birbirleriyle bağlantılıdır. CACTUS, ilk adımda tek bir değişkeni sabit tutarak, tüm değişken çiftleri için küme tahminlerini belirler. Ardından *n* küme için (tüm değişkenleri içeren) bu değişken ile ilgili küme tahminlerini temsil etmek amacıyla kesişim seti oluşturulmaktadır. Bireysel değişkenler ile ilgili tüm küme tahminleri üretilir ve veri setindeki kümeleri elde etmek için sentezlenir. Dolayısıyla temelde algoritmada ilk olarak A_j değişkenine bağlı olarak verilen bir A_i değişkeni için küme tahminleri bulunur. Ardından A_i için tüm küme tahminleri kesiştirilir ve ana kümeleri elde etmek için küme tahminleri (projections) sentezlenir[129,131].

2.1.6.4. LIMBO Kümeleme Algoritması (scaLable Information BOttleneck)

LIMBO kümeleme algoritması, Andritsos ve arkadaşları tarafından 2004 yılında geliştirilmiştir. LIMBO yöntemi, kategorik yapıda değişken içeren veri setlerini ağaç diyagramı oluşturarak kümelemek amacıyla kullanılır. Bu diyagramda bir düğüm, karşılık gelen kümenin üyelerine ait istatistikleri içermektedir. LIMBO, büyük veri setlerinden kaynaklanabilecek zorlukları gidermek üzere dağılımsal özetleri kullanmaktadır. Bu algoritmada tüm demetleri veya ana hafızadaki tüm kümeleri kullanmaya gerek duyulmamaktadır. Sadece yeterli istatistikler ile tanımlanmaktadır.

LIMBO, verinin küçük bir özet modelini oluşturur ve bu özetlenen veri üzerine kümeleme işlemlerini gerçekleştirmektedir. LIMBO algoritması, IB (Information Bottleneck) yapısını kullanmaktadır. Bu yöntem ile kümeleme yaparken, korunmuş olan bilgilerin miktarı belirlenmektedir[132-3].

LIMBO algoritması kümeleme işlemlerini yaparken *DCF* (Distribution Cluster Features) bilgilerini kullanmaktadır. *DCF* bilgisi, iki küme arasındaki veya bir küme ile bir nesne arasındaki uzaklığı hesaplamak için kullanılmaktadır. Bir *c* kümesi için *DCF(c)=(p(c)p(A\c))*' dir. Burada *p(c)*, *c* kümesinin olasılığı ve *p(A\c)* ise, *c* kümesindeki değişken değerlerinin koşullu olasılık dağılımıdır. Büyük veri setleri için, c_1 ve c_2 kümelerinin birleşmesinden oluşan *c** kümesi için *DCF* hesaplamasında aşağıdaki formül kullanılmaktadır[132-3].

$$DCF(c^*)=(p(c^*)p(A\backslash c^*))$$

Formülde geçen $p(c^*)=p(c_i)+p(c_j)$ iken, $p(A\backslash c^*)=\frac{p(c_i)}{p(c^*)}p(A\backslash c_i)+\frac{p(c_j)}{p(c^*)}p(A\backslash c_j)$'dir.

LIMBO kümeleme süresi üç aşamada gerçekleşmektedir. İlk aşamada, veriyi özetlemek için *DCF* ağacı oluşturulmaktadır. İkinci aşamada, seçilen küme sayısı kadar küme oluşturmak amacıyla *DCF*'nin ağaç yaprakları birleştirilmektedir. Üçüncü aşamada, her bir demet kendine en yakın olan *DCF* ile ilişkilendirilmektedir[132-3].

Aşama1: DCF ağacına yerleştirme: Demetler teker teker incelenir ve *t* demeti *DCF(t)*'ye dönüştürülür. *DCF* ağacının kökünden başlanarak bir yol izlenir. Yaprak olmayan düğümde, *DCF(t)* ve düğümlerin her *DCF* arasındaki uzaklıklar hesaplanarak *DCF(t)*'ye en yakın olan *DCF* girdisi bulunur. Ardından ağacın bir sonraki aşamaları incelenir. Herhangi bir yaprak düğümünde *DCF(c)*, *c* kümesinin içine dahil edilip edilmeyeceğine karar vermeye ihtiyaç

vardır. LIMBO algoritmasında maksimum alan sınırı olarak tanımlanan *S* giriş parametresi vardır. *E* ise *DCF* girişindeki maksimum genişliktir. Algoritmada, *E* ve *S*'ye bağlı olarak maksimum düğüm sayısı hesaplanır ve ağaç oluştururken kullanılan düğüm sayıları için bir sayaç oluşturulur. *DCF(c)*'yi içeren boş bir girdi var ise, *DCF(t)* o girişten yer değiştirilir. Eğer boş bir düğüm girişi yok ise, yaprak düğümleri iki yaprağa bölünmektedir. Birbirinden uzak iki *DCF* bulunur ve yeni yapraklar için bu tohum bilgiler kullanılır. Kalan *DCF*'ler ve *DCF(t)*'ler en yakın olan *DCF* tohumlarını içeren yapraklarda yer alırlar. Alan sınırına gelindiğinde ise, yapraklardaki herhangi iki *DCF* girişinin minimum uzaklığı ile *d(c,t)* karşılaştırılır. Eğer *d(c,t)*, minimumdan daha küçük ise, *DCF(t)* ile *DCF(c)* birleştirilir, diğer durumda ise en yakın entry'ler birleştirilir.

Aşama 2: Her bir *DCF(c)*, $c \in \tilde{C}$ kümesine karşılık gelir ve *p(A\c)* ve *p(c)* olasılığını hesaplamak için yeteri kadar istatistikleri içermektedir. IB kullanılarak yapraklardaki *DCF*'ler kümeleme için kullanılır ve *DCF*'lerin kümeleri oluşturulur.

Aşama 3: Seçilen bir *K* değeri yardımıyla ikinci aşamada *K* kümelerin temsilcisi *KDCF*'ler üretilmektedir. Son aşamada ise veri setinin bir taraması yapılır ve her bir demet kendine en yakın olan temsilcinin kümesine atanır.

LIMBO kümeleme algoritması, hiyerarşik ölçeklenebilir bir kategorik algoritmadır. Özet modeli kullanarak, sadece tek bir *K* değerine bağlı kalarak kümeleri oluşturmaz, büyük değer aralıklarında da kümeleri oluşturmaktadır. Ayrıca LIMBO kümeleme yönteminde, istatistikler oluşturularak kümeler doğrudan karşılaştırılabilmektedir. COOLCAT algoritması örnek hacmi ve temsilci seçimine, düğümlerin sırasına karşı aşırı duyarlıdır. Oysa LIMBO yöntemi verilerin giriş düzeninden etkilenmemektedir[132-3].

2.1.6.5. COOLCAT Kümeleme Algoritması (An Entropy-based Algorithm for Categorical Clustering)

Entropi kavramına dayanan ve *K*-mod algoritmasına benzeyen COOLCAT kümeleme algoritması, Barbara ve arkadaşları tarafından 2002 yılında geliştirilmiştir. Ancak kümelerin modlarını (özetlerini) kullanmak yerine küme merkezleri olarak nesneleri kullanmaktadır. Algoritma kümelerin beklenen entropilerini minimum yapmayı amaçlayan artışlı (incremental) bir algoritmadır[134].

COOLCAT yöntemi *K*-mod algoritmasının başlangıç küme modu seçiminden kaynaklanabilecek sakıncaları gidermektedir. Kümeler entropiler azaltılarak

oluşturulmaktadır. Yöntem başlangıç kümeleri oluşturmak için maksimum *K* farklı nesneleri bulmaktadır. Entropi artışını aza indirgemek için geri kalan tüm nesneler kümelerin birine yerleştirilmektedir. Dolayısıyla COOLCAT ismi, kümelerin entropilerini azaltma kavramından gelmektedir. Algoritma, kümelerin beklenen entropilerini minimum yapmaya çalışarak veri setindeki noktaları gruplandırmaktadır. Entropi, bir rassal değişkenin bilgisinin ve belirsizliğinin bir ölçüsüdür[134].

x rasgele bir değişken olduğunda, *S(X), x*'in alabileceği değerlerin kümesi ve *p(X)* ise *X*'in olasılık fonksiyonudur. Bu durumda *E(X)* yani beklenen entropi aşağıdaki formül yardımıyla hesaplanmaktadır[134].

$$E(X) = -\sum_{x \in S(X)} p(x) log\, p(x)$$

Çok değişkenli vektörlerde $\hat{x}=\{X_1,\ldots,X_n\}$ entropi aşağıdaki gibi hesaplanmaktadır.

$$E(\hat{x}) = -\sum_{x_1 \in S(X_1)} \ldots\ldots \sum_{x_n \in S(X_n)} p(\hat{x}) log p(\hat{x})$$

Yukarıdaki formülde yer alan $p(\hat{x})=p(x_1,\ldots..,x_n)$ çok değişkenli olasılık dağılımıdır.

COOLCAT yöntemi, örnek noktaları ile başlamaktadır. Başlangıç aşamasında örnek setinden en benzemez *K* nesneyi seçmektedir. Bu nesneler *K* kümelerinin ilk temsilcilerine karşılık gelmektedir. Her bir adımda kümeleme sonucundaki entropi artışını minimize etmek amacıyla kalan diğer nesneler kümelerden birine atanır. Başlatıldıktan sonra, COOLCAT veri setinde kalan nesneye uygun kümeler bulmak için hızlıca işlemeye devam etmektedir. Bu işlem ise beklenen entropi hesaplanarak yapılmaktadır. Örneğin, Tablo 2.1.6.5.1 incelendiğinde üç kümeleme işlemi karşılaştırılırsa, en küçük beklenen entropiye sahip kümeleme işlemleri kullanılmalıdır. Yani kümeleme 1 işlemi bu örnek için en uygun kümedir[134].

Tablo 2.1.6.5.1. Üç farklı kümeleme sonuçları

Küme	Kümeleme 1		Kümeleme 2		Kümeleme 3	
	Üyeler	E	Üyeler	E	Üyeler	E
Küme0	{kuyruk,4-ayak}	1	{kuyruk, 4-ayak}	2	{kuyruk, 4-ayak}	0
	{kuyruk,2-ayak}		{kuyruk yok, 0-ayak}			
Küme 1	{kuyruk yok, 0-ayak}	0	{kuyruk, 2-ayak}	0	{kuyruk,2-ayak}	2
					{kuyruk yok,0-ayak}	
Beklenen E		**0,66**		1,33		1,33

COOLCAT kümeleme algoritması başlatma (initialization) ve arttırma (incremantal) adımlarından oluşmaktadır. Başlatma adımında, **N** veri setinin genişliği olmak üzere uygun **S** örnek bulunmaktadır. Seçilen noktaların minimum ikili entropileri maksimize edilerek en benzemez *K* kayıtları bulunmaktadır. İlk olarak E(p_{s_1}, p_{s_2}) maksimize eden p_{s_1}, p_{s_2} noktaları araştırılır ve (C_1, C_2) olmak üzere iki ayrı kümeye yerleştirilmektedir. Ardından $min_{i=1,\dots j-1}$(E(p_{s_i}, p_{s_j}))'yi maksimum yapan işaretlenmemiş p_{s_j} seçilir. Kalan diğer işaretlenmemiş noktalar ($|S|-k$), arttırma adımında kümelere yerleştirmek için kullanılacaktır[135].9

COOLCAT algoritmasında, *s*'nin hesaplanmasında ise aşağıdaki formülden yararlanılmaktadır.

$$s = k\rho + k\rho \log\left(\frac{1}{\delta}\right) + k\rho\sqrt{(\log\left(\frac{1}{\delta}\right))^2 + 2\log(\frac{1}{\delta})}$$

Formülde yer alan $\rho = \frac{|D|/k}{m}$, m ise en küçük kümenin genişliğini ifade etmektedir. Eşitlikten görüldüğü gibi *s* veri setinin genişliğine bağlı değildir. Böylece COOLCAT yönteminin büyük veri setleri için uygun olduğu anlaşılmaktadır.

COOLCAT algoritmasının artırma adımı ise başlatma adımının ardından kalan diğer noktalar için beklenen entropileri hesaplayarak uygun kümeler aranması ile başlamaktadır. Bu işlem yapılırken noktalar her bir kümeye yerleştirilir ve beklenen entropiler bulunur. Hangi kümede beklenen entropi daha küçük ise o nokta en düşük entropiye sahip noktaya dahil edilir[135].

Kümelerin elde edilme kalitesi üzerine noktaların işlenme sırası oldukça etkilidir. Bu etkiyi giderebilmek için yığındaki noktaların bir bölümü için süreç yeniden tekrarlanmaktadır. Noktalar yığını kümelendikten sonra en kötü kümelere yerleştirilmiş *m* nokta seçilmektedir. Süreç bu noktaları kümelerinden çıkartılması ve yeniden kümelendirilmesi ile devam etmektedir. Noktaların hangi kümeye en iyi şekilde yerleştirileceğini anlamanın yolu, o kümedeki her bir değişken değerinin meydana gelme olasılığını izlemektir. Daha önceden bilindiği üzere, q_{ij} değeri, i yer değiştirdiğinde kümedeki V_{ij} değerinin kaç kere meydana geldiğini göstermektedir. q_{ij} değerinin olasılıklara dönüştürülmesi için küme genişliğine bölünmesi gerekmektedir. Bu olasılıklarda p_{ij} olarak adlandırılmaktadır. Her bir kayıt için, p_i=$\prod_j(p_{ij})$ değeri hesaplanır ve düşük p_i değeri kayıtın kümedeki en kötü formunu

göstermektedir. Ardından p_i değerlerine göre sıralama yapılır ve p_i değeri en küçük olan m kayıtı seçilerek yeniden süreç tekrarlanır. Her bir tekrarlanan kayıt, en düşük entropi değerine sahip kümeye yerleştirilir[135].

COOLCAT algoritmasının dezavantajı, nesne seçiminin düzenine karşı hassas olması ve karmaşıklığının karesel olmasıdır *(O(N^2))*. COOLCAT yöntemi, kategorik değişken içeren büyük veri setlerini kümelemede oldukça etkilidir. Diğer kategorik kümeleme algoritmasının aksine, COOLCAT kümeleme sonuçları farklı örneklem genişlikleri ve parametre ayarlamaları için sabittir. Algoritmada daha önce kümelenen her noktaya bakma gereksinimi duymadan yeni noktanın kümeleme işlemini gerçekleştirme özelliği vardır[134-5].

2.1.6.6. *K*-Mod Kümeleme Algoritması (K-MODES)

K-mod algoritması 1997 yılında Huang tarafından geliştirilmiştir. *K* -mod kümeleme yöntemi, büyük kategorik verilerin kümelenmesinde kullanılan ilk algoritmalardan biridir. Veri madenciliğinde, büyük veri setleri için etkili olduğu için *K*-ortalama yöntemi sıklıkla kullanılmaktadır. Ancak Öklid uzaklığı fonksiyonu ve küme merkez temsilcilerinin ortalamalarının kullanılması sebebiyle kategorik veriler için uygulanamamaktadır[136].

K-ortalama algoritmasının kategorik verilerde kullanılabilmesi için Ralambondrainy 1995 yılında kategorik değişkenleri veri setinde var-yok şeklinde belirterek 0 veya 1 ile kodlamıştır. Ancak bu yaklaşım çok boyutlu kategorik veriler için uygun bir yaklaşım değildir[136].

K-mod, Öklid uzaklık fonksiyonunun, basit uygun benzemezlik ölçüleri ile değiştirilmesiyle *K*-ortalama yönteminin kategorik değişkenler için modifiye edilmiş yaklaşımıdır.

Algoritmada küme merkezlerinin temsilcisi olarak modlar kullanılır ve bu modlar kümeleme sürecindeki her bir iterasyondaki en sık görülen kategorik değer ile güncellenmektedir. *K*-mod kümeleme algoritması *K*-ortalama yönteminin kategorik veriler için kullanılan bir uzantısıdır. *K*-mod algoritmasının *K*-ortalama yönteminden temelde üç farklılığı vardır. Bu farklılıklar sırasıyla uzaklık fonksiyonu, küme merkez temsilcileri ve iteratif kümeleme sürecidir[136].

m kategorik değişken ile tanımlanan *X* ve *Y* nesneleri arasındaki uzaklık *K*-mod yöntemi ile aşağıdaki gibi tanımlanmaktadır.

$$d(X,Y)=\sum_{j=1}^{m}\delta(x_j,y_j), \qquad \delta(x_i,y_i)=\begin{cases}0 & x_j=y_j\\ 1 & x_j\neq y_j\end{cases}$$

Yukarıdaki formüldeki $x_j \; ve \; y_j$, *X* ve *Y*'deki *j* değişkeninin değerini göstermektedir. Bu fonksiyon benzemezlik ölçüsü veya Hemming uzaklığı olarak bilinmektedir. Bu değer ne kadar büyük ise *X* ve *Y*'lerde o kadar benzemezdir. Ng, Li ve Hunag&He tarafından 2007 yılında bir başka uzaklık fonksiyonu geliştirilmiştir. Bu uzaklık ölçücü *K*-mod yöntemi için kullanılmaktadır ve *X* bir nesne, Z_l'de küme merkez olmak üzere *X* ile Z_l arasındaki benzemezlik ölçüsünü tanımlamaktadır[136].

$$\varphi(x_j,z_j)=\begin{cases}1-\frac{n_j^r}{n_l}, & x_j=z_j\\ 1\,, & x_j\neq z_j\end{cases}$$

z_j, z_l'deki *j* değişkenin kategorik değerini, n_l *l* kümesindeki nesnelerin sayısını, n_j^r ise, değişken değeri *r* olan nesnelerin sayısını göstermektedir. Geliştirilen bu formüle göre, nesnenin kategorik değeri küme merkezi ile aynı olduğunda, nesnenin uzaklığı kümedeki kategorik değerin frekansına bağlıdır[136].

K-mod kümeleme yönteminde, küme merkezleri kategorik değişkenlerin modlarının vektörleri ile temsil edilmektedir. İstatistikte mod, en çok tekrarlanan değer anlamına gelmektedir. Örneğin, *{a, b ,a, a, c, b}* veri setinin modu *a*'dır. Ancak bir veri setinde birden fazla modda olabilmektedir[136].

Bir veri setinin *m* kategorik değişkeni var ise, *z* mod vektörü $(z_1, z_2, \ldots z_m\}$ olmak üzere *m* kategorik değeri içermektedir. Her bir değer bir değişkenin modunu göstermektedir. Bir kümenin mod vektörü, kümedeki ve küme merkezindeki her bir nesne arasındaki uzaklıkların toplamını minimize etmektedir. *X* kategorik veri setini *K* adet kümeye kümelendirmek için *K*-mod algoritmasında aşağıdaki adımlar izlenmektedir[136].

Adım 1: Başlangıç küme merkezi (mod) olarak *K* nesne rasgele olarak seçilir.
Adım 2: Her bir nesne ile küme modu arasındaki uzaklık hesaplanır. Nesneler, kendilerine en yakın olan küme merkezlerine atanır. Tüm nesneler, kümelere atana kadar adımlar tekrarlanır.
Adım 3: Her bir küme için yeniden mod seçilir ve bir önceki mod ile karşılaştırılır. Eğer yeni mod ile önceki mod farklı ise, adım 2'ye dönülür, diğer durumda ise adımlar sonlandırılır.

Bahsedilen kümeleme süreci aşağıdaki *K*-mod amaç fonksiyonunu minimize etmektedir.

$$F(U,Z)=\sum_{l=1}^{k}\sum_{i=1}^{n}\sum_{j=1}^{m}u_{i,j}d(x_{i,j},z_{i,j})$$

Yukarıdaki formülde yer alan $U=[u_{i,j}]$, $n*k$ bölüm matrisi, $z=\{Z_1, Z_2,\ldots, Z_k\}$ mod vektörlerinin bir seti *d(.,.)* ise uzaklık fonksiyonudur.

K-mod kümeleme yöntemi, *K*-ortalama algoritmasına benzer olmasına rağmen, *K*-mod algoritması büyük kategorik verileri kümelemede daha etkilidir ve lokal olarak minimum kümeleme sonuçları üretmektedir[136].

K-mod algoritmasının hesaplama karmaşıklığı *O(t*K*m*n)*'dir. Burada *m* değişken sayısı, *n* nesnelerin sayısı, *K* kümelerin sayısı ve *t* ise tüm veri seti süresince iterasyon sayısıdır. Ancak birçok durumda, *t*K*m<n* olduğundan *K*-mod kümeleme yönteminin zaman karmaşıklığı *O(n)*'e yaklaşmaktadır[137].

2.1.6.7. Global *K*-MOD Kümeleme Algoritması (Global K-Modes Clustering: GKM)

Global *K*-mod algoritması 2012 yılında Tian ve arkadaşları tarafından önerilmiştir. Diğer *K*-*m*od kümeleme yöntemlerine göre küme doğruluğu ve performans açısından daha üstündür. Global *K*-mod yöntemi, veri setinin global dağılımını hesaba katmak için yeteri miktarda büyük sayıda başlangıç modunu rasgele seçmektedir. GKM yöntemi, *K*-mod algoritmasının lokal optimum özelliğinden kaynaklanabilecek kısıtlamaları azaltmak, hafifletmek için önerilmiştir. Ardından eliminasyon kriter fonksiyonu kullanarak gereksiz fazla modları ortadan kaldırmaktadır[137].

K-mod algoritması büyük kategorik veri setleri için oldukça etkili olmasına rağmen, iki temel sakıncadan etkilenmektedir. Bunlardan ilki, *K*-mod algoritmasının lokal optimum yöntem olması ve global bilgileri kapsama özelliğinin olmamasıdır. İkincisi ise, küme sonuçlarının doğruluğunun başlangıç modu seçimine olan hassasiyetidir. Kategorik verilerin değişken değerleri sürekli olmadığı için, sayısal verileri kümeleme ile karşılaştırıldığında kategorik veri kümelemede modları iteratif optimizasyon sürecinde değiştirmek daha zordur. Dolayısıyla başlangıç mod seçiminde bazı kümeler kaçırıldığında (missed), *K*-mod algoritmasında tüm optimal küme modlarını bulmak daha zor hale gelmektedir. Bundan dolayı, kümeleme işleminde başlangıç küme modu seçimi, çok önemli rol oynamaktadır. GKM algoritmasının ana düşüncesi, veri setinin global dağılımını dikkate almak için yeteri miktarda büyük sayıda

başlangıç modlarını rasgele seçmektir. İteratif optimizasyon sürecinde, gereksiz modları çıkarabilmek için aşamalı eliminasyon yöntemi uygulanmaktadır[137].

GKM yöntemi iki anahtar bileşeni içermektedir. Birincisi, yeteri miktarda büyük başlangıç küme modunun rasgele seçilmesi, ikincisi ise eliminasyon kriter fonksiyonu kullanarak aşamalı eliminasyon ile fazla kümeleri elemektir. Birçok var olan *K*-mod kümeleme yöntemlerinde, başlangıç küme modu hedeflenen kümelerin sayısı kadardır. Başlangıç küme modlarını optimize eden bu yöntemler için genellikle veri dağılımı için önsel bir bilgi gerekmektedir. Ancak çok büyük ölçekli veri setleri için bu bilgi her zaman kolaylıkla elde edilememektedir[137].

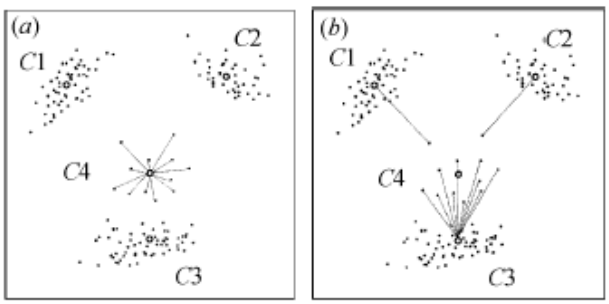
Başlangıç küme sayısı k_{ini}, sonuç hedeflenen kümeler ise k_{tar} ile gösterildiğinde k_{ini}>k_{tar} 'dır. k_{ini}'nin seçimi, veri setinin özelliğine bağlıdır ve araştırmacı tarafından belirlenmektedir. k_{ini} aşağıdaki formül yardımıyla hesaplanmaktadır[137].

$$k_{ini}=\sigma*k_{tar}$$

Yukarıdaki formülde yer alan σ, pozitif tamsayı bir katsayı olup her bir sonuç kümeye yerleştirilen başlangıç modların ortalama sayısını temsil etmektedir. σ'nın seçimi veri dağılımına bağlıdır. Ancak bazı araştırmalarda görüldüğü üzere σ değerinin 4 olarak alınması yeterli olacaktır. $k_{ini}-k_{tar}$ fazla kümenin kümeleme süreci boyunca çıkarılması gerekmektedir. Bu fazla kümeleri çıkarma işlemi iterasyonun her bir seviyesinde gerçekleşmektedir. Hangi kümenin çıkartılacağına karar vermek için çıkarılan kümelerin kalan kümelere olan etkisini ölçen eliminasyon kriter fonksiyonu (*E*) kullanılmaktadır. Çıkartılan kümedeki tek bir veri noktasının etkisi, çıkarılan kümenin moduna olan uzaklığının en yakın modlu çıkarılan kümenin uzaklığı ile farkıdır. Çıkarılan kümelerin genel etkisi ise, kümedeki tüm veri noktalarının fark uzaklıklarının toplamıdır. Ne kadar bu fark küçük ise, kümenin o kadar az etkisi vardır. Eliminasyon kriter fonksiyonu aşağıdaki formül yardımıyla elde edilmektedir[137].

$$\mathrm{E_j}=\sum_{\mathrm{x_i}\in\mathrm{C_j}}[\min \mathrm{d}(\mathrm{x_i},\mathrm{Z_l})-\mathrm{d}(\mathrm{x_i},\mathrm{Z_j})],\quad \mathrm{Z_l}\in \mathrm{Z},\mathrm{Z_l}\neq \mathrm{Z_j}$$

Formülde yer alan C_j, j. Kümeyi göstermektedir. Z=[$Z_1, Z_2,\ldots, Z_k$] aynı aşamada tüm optimal küme modlarını, $d(x_i, Z_j)$, x_i veri noktasının ($x_i \in C_j$) C_j kümesinin Z_j moduna olan uzaklığını ifade etmektedir[137] (Şekil 2.1.6.7.1).



Şekil 2.1.6.7.1. Eliminasyon kriter fonksiyonunun grafiksel gösterimi

x_i'nin Z_j dışındaki diğer küme modları ile minimal uzaklığı $d(x_i, Z_l)$'dir ve şekil b'deki gibi gösterilmektedir. En küçük E_j'ye sahip Z_j modlu C_j kümesi çıkartılabilecek olan kümelerdendir. Daha sonraki iterasyonlarda başlangıç modu olarak kalan *K-1* kümelerin modları kullanılmaktadır. Küme çıkartıldığında, kalan kümeler için *F* değeri artacak ve *F*'in birbirinden farklı *K* değeri elde edilecektir. *F* değeri küçük sonuçlanan küme, kalan kümeler üzerine en az etkiye sahiptir ve genellikle diğer kümelere ait olması muhtemeldir. Örneğin, Şekil 2.1.6.7.1 incelendiğinde C_4 kümesi C_3 kümesine yakın olduğu için çıkartılmaya en yakın kümedir. Bu durumda birçok veri noktası C_3 ile birleştirilmektedir. Ancak hangi kümenin çıkartılacağına bakmak için direk *F* değeri incelenirse, tüm küme kombinasyonlarını test etmeye gerek duyulduğu için hesaplama yoğunluğu artacaktır. Bunun yerine *F'*deki değişimi tahmin etmek için *E* kullanılır. *E* değeri ne kadar küçük ise, *F* değeri o kadar az artmaktadır. En küçük *E* değeri, *F'*deki en düşük artışı göstermektedir. GKM algoritmasının adımları aşağıdaki gibi özetlenebilir[137].

Girdiler:

- *D*: *n* nesne sayısı
- k_{ini}:başlangıç küme modlarının sayısı
- k_{tar}:arzu edilen küme sayısı

Başlangıç:

- D setinden k_{ini} başlangıç modu rasgele seçilir.

İterasyon:

- *K*, k_{ini}'den başlatılarak k_{tar}'a kadar devam eden iterasyondaki mod sayısıdır.
- Optimal çözüm için *K*-mod algoritması uygulanır.
- *K* modlu *S(K)*'lar oluşturulur.
- *S(K)*'lar için *K* modları $Z = [Z_1, \ldots . Z_k]$'dır.

- Burada *i*=1'den başlayarak *K*'ya kadar devam eder.
- Z_i küme modu için E_i eliminasyon kriter fonksiyonu hesaplanır.
  İterasyon bitirilir.

En küçük eliminasyon kriter fonksiyon değeri E_j'li Z_j modu çıkartılır ve kalan $K-1$ mod sonraki iterasyonda başlangıç modları olarak kullanılır. Değişiklik olmayana kadar işlemlere devam edilir[137].

GKM kümeleme algoritmasının hesaplama karmaşıklığı, σ'nın 4 olması durumunda yöntem iyi çalıştığı için genellikle $(\sigma-1)k+1<n$' dir. Dolayısıyla algoritmanın hesaplama karmaşıklığı *O(n)*'e yaklaşacaktır. Bu ifade ise veri seti büyüklüğü ile hesaplama karmaşıklığı arasındaki ilişkinin lineer olduğunu göstermektedir[137].

2.1.6.8. Bulanık *K*-Mod Kümeleme Algoritması (Fuzzy K-Modes Clustering Algorithm)

Bulanık *K*-mod kümeleme yöntemi ilk olarak 1999 yılında Huang ve Ng tarafından önerilmiştir. *K* -mod algoritması kategorik veriler için *K*-ortalama algoritmasının modifiye edilmiş halidir. Yöntem kategorik nesneler için basit bir eşleşmiş (matching) benzemezlik ölçüsü kullanmaktadır. *K*-ortalamadan farklı olarak, küme ortalamalarını kullanmak yerine küme modlarını kullanmaktadır ve modları bulabilmek için frekansa dayalı bir yöntemden yararlanmaktadır. Bu modifikasyonlar ile *K*-ortalama yöntemi sadece sayısal veriler için kullanılmamakta aynı zamanda modifikasyon yapılmış *K*-ortalama yöntemi bir başka ifadeyle bulanık *K*-mod yöntemi büyük kategorik veri setlerinin kümelenmesinde oldukça etkilidir[138].

$[x_1, x_2, \dots., x_m]$ ve $[y_1, y_2, \dots., y_m]$ ile tanımlanmış X ve Y iki kategorik değişken ele alındığında *X* ve *Y* arasındaki eşleşmiş benzemezlik ölçüsü aşağıdaki gibi tanımlanır[138].

$$d_c(X,Y)=\sum_{j=1}^{m}\delta(x_j,y_j),$$

Formülde yer alan $\delta(x_j,y_j)=\begin{cases}0, & x_j=y_j\\ 1, & x_j\neq y_j\end{cases}$dir.

d_c fonksiyonu, kategorik nesne setlerinde metrik uzayıdır. Geleneksel olarak basit eşleşmiş yaklaşım kategorik değişkenlere dönüştürülen ikili (binary) değişkenler için kullanılmaktadır. d_c, Hamming uzaklığının genelleştirilmiş bir şeklidir.

n tane nesneyi *K* kümeye yerleştirmenin amacı F_c'yi minimum yapan *W* ve *Z*'yi bulmaktır.

$F_c(W,Z) = \sum_{l=1}^{k}\sum_{i=1}^{n} w_{li}^{\alpha} d_c(Z_l, X_i)$

Z, *K* küme için *K*-modların setini tanımlamaktadır. $F_c(W,Z)$'yi minimimize etmek aşağıdaki algoritmadan yararlanılmaktadır[138].

1. Başlangıç noktası $Z^{(1)}$ seçilir. $F(W,Z^{(1)})$'yi minimum yapan $W^{(1)}$'e karar verilir ve *t*=1'den başlatılır.
2. $F\left(W^{(t)}, Z^{(t+1)}\right)'$i minimum yapan $Z^{(t+1)}$ belirlenir.
   Eğer $F(W^{(t)}, Z^{(t+1)}) = F(W^{(t)}, Z^{(t)})$ ise işlem durdurulur, diğer durumda adım 3'e geçilir.
3. $F(W^{(t+1)}, Z^{(t+1)})$'i minimum yapan $W^{(t+1)}$ belirlenir. Eğer $(W^{(t+1)}, Z^{(t+1)}) = F(W^{(t)}, Z^{(t+1)})$ ise işlem durdurulur, diğer durumda *t=t+1* olur ve adım 2'e gidilir.

Her bir iterasyonda *Z*'yi güncellemek için aşağıdaki frekansa dayalı yöntem kullanılmaktadır.

X; $A_1, A_2, \ldots, A_m$ kategorik değişkenleri ile tanımlanan veri seti olmak üzere, DOM(A_j)=$\{a_j^{(1)}, a_j^{(2)}, \ldots a_j^{(n_j)}\}$'dir. n_j ise, $1 \leq j \leq m$ için A_j değişkeninin kategorilerinin sayısıdır. $1 \leq l \leq k$ için Z_l küme merkezleri $[z_{l,1}, z_{l,2}, \ldots., z_{l,m}]$ olarak tanımlanmaktadır. Bu durumda $\sum_{l=1}^{k}\sum_{i=1}^{n} w_{li}^{\alpha} d_c(Z_l, X_i)$ miktarı $1 \leq j \leq m$ için $\sum_{i,x_i,j=a_j^{(r)}} w_{li}^{\alpha} \geq \sum_{i,x_i,j=a_j^{(t)}} w_{li}^{\alpha}$ ($1 \leq t \leq n_j$)'nin olduğu $Z_{l,j}=a_j^{(r)} \in$DOM(A_j) gerek ve yeter koşulunda minimize edilir.

Bulanık *K*-mod kümeleme algoritmasının en büyük dezavantajı, genellikle lokal bir minimum ile sonlandırılmasıdır. Bulanık *K*-mod algoritmasının hesaplama karmaşıklığı *O(t*K*m*n)*'dir. Burada *m* değişken sayısı, *n* nesnelerin sayısı, *K* kümelerin sayısı ve *t* ise tüm veri seti süresince iterasyon sayısıdır[138].

2.1.6.9. Yeni Bulanık *K*-prototip Kümeleme Algoritması (New Fuzzy *k*-prototype Clustering Algorithm)

Birçok uygulamada, veri setleri hem sayısal hem de kategorik özelikler içermektedir. Bu tip veriler için kullanılan en önemli algoritmalardan biri *K*-prototip yöntemidir. Ancak bu yöntem oldukça katı bir ayrım yapmaktadır. Ayrıca bölge sınırları içerisindeki veri nesnelerinin hatalı sınıflandırılmasına sebep olabilmektedir. Bu algoritmada, değişkenlerin önemliliklerini ayarlamak için benzememezlik ölçüsü sadece kullanıcı tarafından verilen parametreyi kullanmaktadır. 2012 yılında Ji tarafından geliştirilen bulanık *K*-prototip algoritmasında ortalama ve bulanık merkezleri kombine edilmektedir. Veri nesneleri ve küme prototipleri arasındaki benzememezlikleri değerlendirmek için değerlerin birlikte gerçekleşmesine dayanan yeni bir ölçü kullanmaktadır. Ayrıca bu ölçü farklı değişken önemliliklerini de dikkate almaktadır[139].

X_i karmaşık bir veri nesnesi olmak üzere, $[x_{i.1}^r, x_{i.2}^r, \ldots.., x_{i.p}^r, x_{i.p\ \ 1}^c, x_{i,p|2}^c, \ldots., x_{i.m}^c]$ vektörü şeklinde tanımlanmaktadır. *r* üst simgesine sahip ilk *p* sayısal değerleri, geri kalan özellikler ise kategorik değerleri ifade etmektedir. Dolayısıyla prototipin iki kısmı vardır. Birincisi ilk *p* merkezleri sayısal değişkenler için ortalamayı kullanırken, diğerleri kategorik değişkenler için kalan merkezler, bulanık merkezleri kullanmaktadır. *K*-prototip algoritmasında kullanılan *E* amaç fonksiyonuna benzer olarak, karılık veri nesnelerini kümelemek için yeni bir amaç fonksiyonu geliştirmiştir ve bu amaç fonksiyonu aşağıdaki gibi tanımlanmaktadır[139].

$$E(U,Q)= \sum_{j-1}^{k} \sum_{i-1}^{n} u_{ij}^{\alpha} d(x_i, Q_j)$$

$U=(u_{ij})_{n*k}$ bulanık bölüm matrisidir ve $0 \leq u_{ij} \leq 1$, $\sum_{j=1}^{k} u_{ij}=1$'dir. $Q_j=[q_{j1}, q_{j2}, \ldots, q_{jp}, \tilde{v}_{jp+1}, \ldots., \tilde{v}_{jm}]$, *j*. küme için temsilci vektör veya prototiptir. α ($1<\alpha<\infty$) belirsizlik katsayısı iken $d(x_i, Q_j)$, x_i veri nesnesi ve Q_j prototipi arasındaki benzememezlik ölçüsü olarak tanımlanmaktadır ve aşağıdaki gibi formülize edilmektedir[139].

$$d(x_i, Q_j)=\sum_{l=1}^{p}(w_l(x_{il}^r - q_{jl}^r))^2+\sum_{l=p-1}^{m} \varphi\ (x_{il}^c, \tilde{v}_{jl}{}^c)^2$$

x_{il}^r, *i.* Veri nesnesindeki sayısal değişkenin *l*. değeri

w_l, aşağıdaki gibi tanımlanmaktadır.

A_i sayısal değişkeni aynı sayıda aralık içeren T aralıklara bölünmektedir. Her bir aralık için $u[1], u[2],....,u[k]$ kategorik değerleri atanır ve

$w_l=\frac{2\sum_{k=1}^{T}\sum_{j>k}^{T}\delta(u[k],u[j])}{T(T-1)}$, dir.

u_{ij}, aşağıdaki formül yardımıyla hesaplanmaktadır.

$u_{ij}=(\sum_{z=1}^{k}(\frac{d(x_i,Q_j)}{d(x_i,Q_z)})^{\frac{1}{\alpha-1}})^{-1}$

q_{jl}^{r}, sayısal değişkenlerin merkezi olup aşağıdaki gibi tanımlanmaktadır.

$q_{jl}^{r}=\frac{\sum_{i=1}^{n}u_{ij}^{\alpha}x_{il}^{r}}{\sum_{i-1}^{n}u_{ij}^{\alpha}}$

$\tilde{v}_{jl}{}^{c}$ kategorik değişkenin bulanık merkezidir ve aşağıdaki gibi ifade edilmektedir.

$$\tilde{v}_{jl}{}^{c}=a_{jl}^{1}/w_{jl}^{1}+a_{jl}^{2}/w_{jl}^{2}+....+a_{jl}^{K}/w_{jl}^{K}+....+a_{jl}^{t}/w_{jl}^{t}$$

$w_{jl}^{K}=\sum_{i=1}^{n}Y(x_{il}^{c})$, $Y(x_{il}^{c})=\begin{cases}\frac{u_{ij}^{\alpha}}{\sum_{i-1}^{n}u_{ij}^{\alpha}}, & a_{jl}^{K}=x_{il}^{c}\\ 0, & a_{jl}^{K}\neq x_{il}^{c}\end{cases}$ ve

$$\boldsymbol{\varphi}(x_{il}^{c},\tilde{v}_{jl}{}^{c})=\sum_{k=1}^{t}\tau(x_{il}^{c},a_{jl}^{K})*\boldsymbol{\delta}(x_{il}^{c},a_{jl}^{K})$$

Yukarıda formülde yer alan $\boldsymbol{\delta}(x_{il}^{c},a_{jl}^{K})$ ise şöyle hesaplanmaktadır.

$\boldsymbol{\delta}(x_{il}^{c},a_{jl}^{K})=\frac{\sum_{j=1,i\neq j}^{m}\delta^{ij}(x_{il}^{c},a_{jl}^{K})}{m-1}$, $\delta\ \ (x_{il}^{c},a_{jl}^{K})$ özelikleri ise sırasıyla aşağıda verilmektedir[139].

- $0\leq\ \delta\ \ (x_{il}^{c},a_{jl}^{K})\leq 1$
- $\delta\ \ (x_{il}^{c},a_{jl}^{K})=\delta\ \ (a_{jl}^{K},x_{il}^{c})$
- $\delta\ \ (a_{jl}^{K},a_{jl}^{K})=0$

formülde yer alan $\tau(x_{il}^{c},a_{jl}^{K})$ değeri aşağıdaki şekilde elde edilmektedir.

$$\tau(x_{il}^{c},a_{jl}^{K})=\begin{cases}0, & x_{il}^{c}=a_{jl}^{K}\\ 1, & x_{il}^{c}\neq a_{jl}^{K}\end{cases}$$

Yukarıdaki verilen detaylı hesaplamaların ardından geliştirilen bulanık *K*-prototip algoritmasının adımları ise aşağıda sırasıyla verilmektedir[139].

Adım 1: Maksimum iterasyon sayısı, *K* küme sayısı, α değeri, *T* aralık sayısı ve ε eşik değeri verildiğinde *K* farklı nesne veri setinden rasgele seçilir ve başlangıç prototipi $Q^{(t)} = (Q_1, Q_2, \dots, Q_k)$'ya dönüştürülür. Ayrıca *t* değeri sıfırdan başlatılır.
Adım 2: $u^{(t)}$ bulanık bölüm matrisi hesaplanır.
Adım 3: Yeni prototip olan $Q^{(t+1)}$'in elde edilebilmesi için $Q^{(t)}$ güncellenir. Sayısal ve kategorik değişkenlerin prototip bölümü için tanımlanan formüllerden yararlanılır.
Adım 4: Eğer hesaplanan iki *E* değeri arasındaki fark ε eşik değerinden fazla değil ise maksimum iterasyon sayısı sıfıra eşit olur ve işlemler sonlandırılır diğer durumda *t* bir artırılır, maksimum iterasyon sayısı da bir azaltılarak adım ikiye yeniden gidilir.

Bulanık *K*-prototip algoritmasının zaman karmaşıklığı, her bir iterasyonda güncellenen prototiplere ve bölüm matrisine, iki değişken değeri arasındaki uzaklık hesaplanmasını içermektedir. Bulanık prototipleri, bölünme matrisinin ve uzaklıkların hesaplama maliyeti sırasıyla $O(Kmn)$, $O(K(p+Nm-Np)n)$, $O(m^2n+m^2S^3)$'dür. Burada *K* küme sayısı, *p* sayısal değişkenlerin sayısı, *m* ise tüm değişkenlerin sayısıdır. $p<t\leq m$ için $N=max(t)$ maksimum kategori sayısı, *n* veri nesnesi sayısı, *S* ise farklı kategorik değerlerin ortalama sayısıdır. Böylece genel zaman karmaşıklığı $O(m^2n+ m^2S^3+ K(m+p+Nm-Np)ns)$ olarak elde edilmektedir. Burada *s* değeri iterasyon sayısıdır. *K*-prototip yönteminin zaman karmaşıklığı ise $O((s+1)kn)$'dir. Dolayısıyla bulanık *K*-prototip algoritmasının zaman karmaşıklığı *K*-prototip yöntemine göre biraz daha fazladır[139].

$n>k,m,s$ olduğu durumlarda önerilen bu yöntem hiyerarşik yöntemlerden daha hızlıdır. Yer karmaşıklığı ise *X* veri seti ve prototipleri saklamak için $O(K(p+mN-pN)+mn)$ iken, bölüm matrisi *U*'yu depolamak için $O(Kn)$'dir. Böylece genel yer karmaşıklığı bulanık *K*-prototipi için $O(K(p+n+mN-pN)+mn)$ olarak elde edilmektedir[139].

2.1.6.10. Karışık Veriler İçin Geliştirilen *K*-Prototip Kümeleme Algoritması (an improved *K*-Prototypes clustering algorithm for mixed numeric and categorical data)

Bu algoritma 2013 yılında Ji ve arkadaşları tarafından geliştirilmiştir. Yöntemde kümedeki kategorik değişkenlerin prototiplerini temsil etmek için merkez dağılımı kavramı kullanılmaktadır. Ardından, karışık değişkenlerin prototiplerini temsil etmek için merkez dağılımı ile ortalama kavramı kombine edilmektedir. Böylece veri nesneleri ve küme prototipleri arasında benzememezlikleri hesaplamak için yeni bir ölçü önerilmektedir. Geliştirilen bu *K*-prototip algoritmasında değişkenlerin önemlilikleri ise Huang stratejesi ile değerlendirilmektedir. Dolayısıyla yöntemin adımlarına geçmeden önce bazı tanımlamaları yapmakta fayda vardır[140].

Dağılım merkezi: Bulanık senaryoda merkezde kategorik değişkenlerin küme merkezlerinin temsil edimesinde uygun bir ölçüdür. *Dom* $(A_j)=\{a_j^1, a_j^2, \ldots, a_j^t\}$ için *l* kümesinin dağılım merkezi C_l ile tanımlanmaktadır ve aşağıdaki gibi elde edilmektedir[140].

$$C_l'=\{c_{l1}', c_{l2}', \ldots . c_{lj}', \ldots, c_{lm}'\}$$

$$c_{lj}'=\{\{a_j^1,w_{lj}^1\},\{a_j^2,w_{lj}^2\},\ldots.\{a_j^K,w_{lj}^K\},\ldots,\{a_j^t,w_{lj}^t\}\}$$

Yukarıdaki eşitlikte yer alan $w_{lj}^K=\sum_{i=1}^{n}\eta(x_{ij})$'e eşittir.

$$\eta(x_{ij})=\begin{cases}\frac{u_{il}}{\sum_{i=1}^{n}u_{il}}, & eğer\ x_{ij}=a_j^K\\ 0, & eğer\ x_{ij}\neq a_j^K\end{cases}$$

(u_{il} ise 0 ile 1 arasında (her ikisi de dahil) bölüm matrisinin bir elementidir.)

w_{lj}^K değerinin alabileceği değerler aşağıdaki koşullarda verilmektedir[140].

$$\begin{cases}0\leq w_{lj}^K\leq 1\,, & 1\leq K\leq T\\ \sum_{K=1}^{t}w_{lj}^K=1, & 1\leq j\leq m\end{cases}$$

Yukarıdaki eşitliklerden görüldüğü üzere, dağılım merkezi, kümedeki kategorik değişkenlerin her bir değerinin meydana gelme frekansını kayıt etmektedir. Dolayısıyla bu ölçü kümedeki kategorik değişkenlerin dağılım karakteristiklerini yansıtmaktadır.

Bir değişkenin önemi, kümeleme sürecinde ilgili değişkenin etkisini göstermektedir. Huang ve arkadaşları değişkenlerin önemliliklerini hesaplamak için bir yaklaşım önermişlerdir. Önerdikleri yöntemde önemlilik, amaç fonksiyon değeri minimum yapılarak hesaplanmaktadır. Değişkenlerin önemliliklerini ayırmanın prensibi, büyük değere sahip değişkeni küme içi uzaklıkları (Within Cluster Distances (WCD)) toplamı küçük olana atamaktır (tersi de doğrudur). Bu prensip aşağıdaki gibi formülize edilmektedir[140].

Önerilen yöntemde amaç fonksiyonu aşağıdaki gibi elde edilmektedir.

$$E(U,Q,S)=\sum_{l=1}^{k}\sum_{i=1}^{n}\sum_{j=1}^{m}u_{il}s_j^{\lambda}d(x_{ij},q_{lj})$$

u_{il}, U_{n*k} bölüm matrisinin bir elementidir ve $u_{il} \in\{0,1\}$, $\sum_{l=1}^{k}u_{il}$=1'dir. s_j değeri ise j. değişkenin önemlilik değeridir ve $s_j \in\{0,1\}$, $\sum_{j=1}^{m}s_j$=1'dir. $\boldsymbol{\lambda}$>1 olmak üzere s_j önemlilikleri için üs değeridir. q_{ij}, l kümesindeki j. değişkenin merkezidir. Amaç fonksiyonunu minimum yapmak için aşağıdaki üç ana problemin çözülmesi gerekmektedir[140].

SP₁: Sırasıyla *Q*' ve *S*' olarak *Q* ve *S* sabitlenir ve *E(U,Q',S')* problemini minimize edilir.

Bu problem aşağıdaki formüller yardımıyla hesaplanmaktadır.

$$\begin{cases} u_{il}=1 \;\; eğer \sum_{j=1}^{m}s_j^{\lambda}d(x_{ij},q_{lj}) \le \sum_{j=1}^{m}s_j^{\lambda}d(x_{ij},q_{ej}) & 1\le e\le k \\ u_{il}=0 & e\neq 1 \end{cases}$$

SP₂: Sırasıyla U' ve S' olarak U ve S sabitlenir ve E(U',Q,S') problemini minimize edilir.

Bu problem aşağıdaki formüller yardımıyla hesaplanmaktadır[140].

Sayısal değişkenler için $q_{lj}=\frac{\sum_{i=1}^{n}u_{il}x_{ij}}{\sum_{i=1}^{n}u_{il}}$

Kategorik değişkenler için $q_{lj}=c'_{ij}$'dir ve c'_{ij}, 1 numaralı eşitlik ile elde edilmektedir.

$d(x_{ij},q_{lj})$ ise aşağıdaki gibi hesaplanmaktadır[140].

$$d(x_{ij},q_{lj})=\begin{cases} |x_{ij}-q_{lj}|, & eğer\ l.değişken\ sayısal\ ise \\ \varphi(x_{ij},q_{lj}), & eğer\ l.değişken\ kategorik\ ise \end{cases}$$

$$\varphi(x_{ij}, q_{lj})=\sum_{K=1}^{t} \vartheta(x_{ij},a_j^K) \text{ ve } \vartheta(x_{ij},a_j^K)=\begin{cases} 0, & x_{ij} = {a_j}^K \\ \omega_{lj}^K, & x_{ij} \neq {a_j}^K \end{cases}$$

SP₃: Sırasıyla *U*' ve *Q*' olarak *U* ve *Q* sabitlenir ve *E(U',Q',S)* problemini minimize edilir.

E(U',Q',S), $s_j'=\begin{cases} 0, & eğer\ D_j = 0 \\ \sum_{r=1}^{h}[\frac{D_j}{D_r}]^{1/(\lambda-1)}, & eğer\ D_j \neq 0 \end{cases}$ olduğunda minimize edilir.

Burada $D_j=\sum_{l=1}^{k}\sum_{i=1}^{n} u_{il}'\ d(x_{ij}, q'_{lj})$

h değeri ise $D_j \neq 0$ olan değişkenlerin sayısıdır.

Yukarıdaki tanımlamaların ardından geliştirlen *K*-prototip algoritmasının işlem adımları aşağıdaki gibidir[140].

Adım 1: Maksimum iterasyon sayısı *maxIte*, *K* küme sayısı, λ değeri, kayıp değer olmadan veri setinden rasgele farklı *K* veri nesnesi seçilir ve belirlenir. Bu parametreler başlangıç prototiplere $Q^{(t)} =(Q_1, Q_2, \dots, Q_k)$ dönüştürülür ve *t* sıfırdan başlatılır.
Adım 2: Q',S'; Q^t ve *S^t* olarak sabitlenir, *U^{t+1}* elde etmek için *E(U, Q',S')* problemi minimize edilir.
Adım 3: U',S'; U^{t+1} ve *S^t* olarak sabitlenir, *Q^{t+1}* elde etmek için *E(U', Q,S')* problemi minimize edilir.
Adım 4: U',Q'; U^{t+1} ve *Q^{t+1}* olarak sabitlenir, *S^{t+1}* elde etmek için *E(U', Q',S)* problemi minimize edilir.
Adım 5: E'de bir gelişme olmadığında veya maksimum iterasyon sayısı sıfıra eşit olduğunda işlemlere son verilir. Diğer durumda *t* bir artırılır, maksimum iterasyon sayısı bir azaltılarak adım 2'ye gidilir.

Prototipleri ve bölüm matrisleri güncellemenin hesaplama karmaşıklığı sırasıyla *O(Kmn), O(K(p+Nm−Np)n)*'dir. *K* küme sayısı, *p* sayısal değişken sayısı, *m* ise tüm değişkenlerin sayısıdır. *N=max(t)* kategorik değişkenler için maksimum değer sayısı ve *n* ise nesne sayısıdır. Dolayısıyla genel zaman karmaşıklığı *O(K(m+p+Nm−Np)nl)* olur. Burada *l* iterasyon sayısıdır. Geliştirilen bu yeni algoritmanın zaman karmaşıklığı klasik *K*-prototip algoritmasından çok az fazladır. *n>k,m,l* koşullarında hiyerarşik kümeleme algoritmasından daha hızlı sonuç vermektedir. Prototipleri ve *X* veri setini saklamak için yer karmaşıklığı

$O(K(p+mN-pN)+mn)$ iken, bölüm matrisi U'yu saklamak için ise $O(Kn)$'dir. Böylece genel yer karmaşıklığı geliştirilen yeni algoritma için $O(K(p+n+mN-pN)+mn)$ olarak elde edilir[140].

Geliştirilen bu algoritmanın literatüre üç önemli katkısı vardır. Bunlardan birincisi, algoritma karışık değişkenlerin kümelerinin özeliklerini daha etkili bir şekilde ele almaktadır ve prototip temsilcileri kümedeki hem sayısal hem de kategorik değişken değerlerinin dağılım bilgilerini içermektedir. İkincisi, yöntem yeni bir benzememezlik ölçüsü kullanarak, farklı değişkenlerin özelliklerini dikkate almaktadır. Üçüncüsü ise, veri nesnelerinin bölüm matrislerine dayanarak algoritma otomatik olarak Huang stratejisini kullanarak farklı değişken önemliliklerini hesaplamaktadır[140].

2.1.6.11. CLICKS Kümeleme Algoritması (Mining subspace clusters in categorical data via *K*-partite maximal cliques)

CLICKS algoritması, Zaki ve Peters tarafından 2007 yılında geliştirilmiştir. Bu algoritma K parçalı en büyük kliklere dayanarak kümeleme işlemlerini gerçekleştirmektedir. Bu algoritmada, büyük boyutlu veri setleri daha önce var olan algoritmalardan daha iyi ölçeklenmektedir[141].

CLICKS kümeleme algoritmasının ana katkısı, kategorik veri setleri için K parçalı klikler olmak üzere yeni bir formülasyon kullanmaktadır. Yöntemde kümeler işlem sonrasındaki K parçalı kliklere karşılık gelmektedir[141].

$A_1,.....,A_n$ kategorik değişkenler ve $D_1,....,D_n$ bu değişkenlerin değerlerini göstermektedir. A_j değişkeninin alt setinin değerleri $S_j \subseteq D_{ij}$ ve $S=S_1*....*S_K$ çapraz ürünü olarak tanımlanmaktadır. Her bir S_j değeri A_{ij} değişkeni üzerinde S'nin üretimini göstermektedir. S altuzayları elde edildiğinde $M \in S$ olmak üzere M, en büyük altuzayı ifade etmektedir. j. değişken değeri r olmak üzere $\sigma(S)$ değeri, D'deki tüm r kayıtlarının sayısıdır. $j \in \{1,....,K\}$ için $A_j \in S_{ij}$'dir. Kullanıcı tarafından belirlenen σ^{min} parametresi belirlendiğinde $\sigma(S) \geq \sigma^{min}$ koşullarında S sıklık adını almaktadır. Bağımsızlık varsayımı altında $E[\sigma(S)]=|D|*\prod_{j=1}^{k}\frac{|S_j|}{|D_{ij}|}$'dir[141].

$\sigma(S) \geq \alpha$ koşulları altında $(\alpha \in R^+)$ $\delta_\alpha(S)=1$ fonksiyonu yoğunluk indikatör fonksiyonu olarak tanımlanmaktadır. $\sigma(S) \geq \alpha$ koşulu sağlanmadığında ise $\delta_\alpha(S)$ fonksiyonu sıfıra eşit

olmaktadır. $\delta_\alpha(S)$=1 olduğu koşullarda S yoğun alt uzayı olarak adlandırılır.$\forall v_a \in S_i$ ve $\forall v_b \in S_j$ koşulları altında S_i ve S_j'ler güçlü bağlantılı olarak tanımlanır ve $\{v_a\}$*$\{v_b\}$ 2-altuzayı yoğundur. Tüm $1 \leq i < j \leq K$ koşulları altında S_i, S_j'ye güçlü bağlantılı olduğunda $S=S_1*....*S_k$ güçlü bağlantılı altuzayı adını almaktadır[141].

D kategorik bir veri seti olmak üzere ve $\alpha > R^+$, C; D'de en büyük ve güçlü bağlantılı altuzayı olduğu durumlarda $C=(C_1*....*C_k)$ K altuzayı $A_{i1}, ..., A_{ik}$ değişkenleri üzerinde bir kümedir[141].

D kategorik bir veri seti ve $V=\bigcup_{i=1}^{n} D_i$'dir. $(v_i, v_j) \in E \leftrightarrow \delta_\alpha(\{v_i\}, \{v_j\})$=1 koşullarında yönsüz (undirected) Γ_D=(V,E) grafiği, D'nin K-parçalı grafiği olarak adlandırılmaktadır.
$Cj \subseteq D_{ij}$ koşullunda $C=C_1*....*C_k$ K-alt uzayı ve D kategorik veri seti verilmiş olsun. C'nin en büyük (maximal) ve Γ_D'de yoğun K-parçalı klikleri var ise C, D'de K-kümedir[141].

CLICKS algoritması ile kümeleme işlemi üç aşamada gerçekleşmektedir ve bu aşamalarda hem tüm uzay hem de alt uzaylarla ilgilenilir. İlk aşama olan ön işleme adımında, D girdi veri tabanından K-parçalı grafikler oluşturulmaktadır. Ardından değişken değerleri bağlanabilirliklerine göre sıralanır. İkinci aşama olan klikleri belirleme adımında Γ_D grafiğinde en büyük K-parçalı klikler numaralandırılmaktadır. Üçüncü aşama olan işlem sonrası adımında ise tespit edilen klikler için yoğunluk özellikleri doğrulanmaktadır. En büyük klik yoğunluk testi sonucunda yoğun olarak kabul edilmeyebilirken, bu kliğin alt klikleri yoğun olarak kabul edilebilmektedir. Bu eksikliği tamamlamak için CLICKS kümeleme algoritması, seçici dikey genişleme yaklaşımını (selective vertical expansion) kullanmaktadır. Bu yaklaşımda yoğun olmayan maksimal kliklere neden olan alt klikler keşfedilmektedir[141].

2.1.6.12. SQUEEZER Kümeleme Algoritması

Squeezer kümeleme algoritması 2002 yılında Zengyou He ve arkadaşları tarafından önerilmiştir. Geliştirilen bu yöntem kategorik verilerin kümelenmesi için kullanılmaktadır. Ayrıca aykırı değerleri oldukça etkili bir şekilde saptamaktadır. Squeezer kümeleme yönteminin mantığı oldukça basittir. Algoritma veri tabanından kayıtları oluşturan veri gruplarını tek tek okumaktadır. Başlangıçta tüm veriler tek bir küme olarak düşülmektedir. Bunlara bağlı olarak izleyen diğer veri grupları ve kümeler arasındaki benzerlik fonksiyonları tanımlanarak ya var olan kümeye dâhil edilir ya da tüm var olan kümeler dışında başka bir kümeye dâhil edilir[142-3].

Squeezer kümeleme algoritması veri sadece bir kere taradığı için yerleşik veri setleri için oldukça etkilidir. Yöntemin temel amacı etkinliği ve ölçeklenebilirliği kombine etmektir. Bunun yanı sıra algoritma başlangıçta kullanıcı tarafından belirlenen küme sayısına ihtiyaç duymamaktadır. Bu kullanıcı açısından büyük kolaylık sağlamaktadır. Ancak algoritma için tek belirlenmesi gereken parametre veri grupları ve küme arasındaki benzerlik değeridir[142-3].

Yöntemin uygulamasına geçmeden önce bazı tanımlamaları yapmak da fayda vardır. A_1, $A_2,....,A_m$, $D_1,D_2,.....,D_m$ domainlerinin sırasıyla kategorik değişken setleri olarak tanımlanmaktadır. D, $t:t\in D_1,D_2,....,D_m$ olmak üzere veri gruplarının setidir. Her bir $tid\epsilon TID$ için, A_i değişken değeri için karşılık gelen veri grubu $val(tid,A_i)$ olarak gösterilmektedir[142-3].

Tanım 1: (Küme): Küme={tid|tid∈TID} kümesi *TID*'in bir alt setidir.

Tanım 2: C kümesi verildiğinde, *C* kümesine dayanarak A_i değişken değerlerinin seti $VAL_i(C)=\{av(tid,A_i)|tid\in C$ olarak tanımlanmaktadır.

Tanım 3: $a_i\in D_i$ olduğunda ve *C* kümesi verildiğinde, *C'*deki a_i desteği A_i'ye dayanarak $SUP(a_i)=\{tid|tidA_i=a_i\}|$ olarak tanımlanır.

Tanım 4: (ÖZET): C kümesi verildiğinde, *C*'nin özeti aşağıdaki gibi belirlenmektedir.

Özet: $\{VS_I|1\leq İ\leq m\}$ ve $VS_i=\{(a_j, Sup\ (a_j))|a_j \in VAL_i\ (C)\}$

Her bir *özet m* element içermektedir ve *m* toplam değişken sayısını göstermektedir. *Özetteki* her bir element değişken değer çiftlerinin setini ve bu setlere karşılık gelen destekleri içermektedir.

Tanım 5: (Küme yapısı (Cluster Structure, CS): *C* kümesi verildiğinde *CS* aşağıdaki gibi ifade edilmektedir.

$$CS= \{Küme, Özet\ \}.$$

Tanım 6: C kümesi ve *t* grubu verildiğinde, *C* ve *tid* arasında benzerlik aşağıdaki gibi belirlenmektedir.

$$Sim\ (C, tid)=\sum_{i=1}^{m}\left(\frac{Sup(a_i)}{|C|}\right),\ a_i=tid*A_i$$

Tanım 6'da yer alan bu tanımlama veri gruplarının kümede yer alıp almayacağını belirlemektedir. Konunun daha iyi anlaşılabilmesi için eğitim durumu, hastalık türü, yıl, hastanede yattığı bölüm ve kullandığı ilaç adı olmak üzere toplam beş kategorik yapıda değişkenden Tablo 2.1.6.12.1' deki verilerin elde edildiğini varsayalım.

Tablo 2.1.6.12.1. Beş kategorik değişkenden oluşan veri seti

Veri grubu (Tuple ID)	Eğitim durumu	Hastalık	Yıl	Yattığı bölüm	Kullanılan ilaç
1	2	1	90	A	b
2	2	2	92	B	a
3	1	3	93	C	c
4	2	3	95	C	c
5	3	4	95	D	c
6	4	1	95	B	c
7	5	1	91	B	d

Küme *C*'nin {3,4,5} nolu bireylerden oluştuğu varsayımı altında Squeezer algoritması için yapılan tanımlamalar ışığında, *Özet= {(1,1), (2,1), (3,1)}, {(3,2), (4,1)}, {(93,1), (95,2)}, {(C,2), (D,1)}, {c,3}* olarak elde edilmektedir.

Tanım 6'da tanımlanan benzerlik fonksiyonu istatistiklere dayanmaktadır. Bir başka ifadeyle veri grubu ve var olan küme arasındaki benzerlik ne kadar büyük ise, veri gurubun o kümeye ait olma olasılığı daha fazladır. Dolaysıyla benzerlik fonksiyonu *tid=6* için aşağıdaki gibi hesaplanmaktadır.

$$Sim(\{3,4,5\},6)=\sum_{i=1}^{5}\left(\frac{Sup(a_i)}{\sum_j Sup(a_j)}\right) = \left(\frac{0}{1+1+1}+\frac{0}{2+1}+\frac{2}{2+1}+\frac{0}{2+1}+\frac{3}{2}\right)=1,67$$

Her bir veri grubu için, benzerlik fonksiyonu yardımıyla var olan tüm kümeler ile benzerlikler hesaplanmaktadır. En büyük benzerlik değeri seçilmektedir. Eğer bu değer belirlenen eşik değerden büyük ise *s* olarak tanımlanır ve veri grubu en büyük benzerlik değerine sahip kümeye yerleştirilir. Ardından yeni grubu ile *CS* güncellenir. Eğer yukarıdaki koşul sağlanmaz ise, veri grubu için yeni bir küme oluşturulur. Bu işlemler veri setindeki tüm veri grupları taranana kadar devam etmektedir[142-3].

Squeezer kümeleme algoritmasının zaman ve yer karmaşıklığı veri seti genişliğine (*n*), değişken sayısına (*m*), *CS*' lerin sayısına ve her bir *CS*' nin genişliğine bağlıdır. Sonuçta elde edilecek küme sayısı *k*, her bir değişkenin *p* farklı değerinin olduğu düşünüldüğünde algoritmanın zaman karmaşıklığı *O(n*k*p*m)* iken yer karmaşılığı *O(n+k*p*m)*'dir. Ancak pratik uygulamalarda Squeezer algoritmasının zaman karmaşıklığı *O(n*k*m)* olarak

beklenmektedir. Ayrıca Squeezer yönteminin zaman karmaşıklığı veri setinin genişliği, değişken sayısı ve sonuç da elde edilen küme sayısı ile doğru orantılıdır ki bu ilişki de algoritmanın iyi ölçeklenebilir olduğunu göstermektedir[142-3].

2.1.6.13. dSQUEEZER Kümeleme Algoritması

dSqueezer kümeleme algoritması 2005 yılında Zengyou He ve arkadaşları tarafından geliştirilmiştir. Algoritma squeezer kümeleme yönteminin bir uzantısıdır. Ancak yeni geliştirilen bu algoritmanın squeezer yönteminden temel farklılığı kategorik ve sayısal nitelikteki değişkenleri de içeren karmaşık veri tabanları için kullanıyor olmasıdır. Squeezer kümeleme yönteminin kategorik verileri kümelemede oldukça etkili sonuçlar verdiği daha önceki çalışmada gösterilmiştir. Benzer olarak dSqueezer algoritmasında da basit bir strateji belirlenmektedir. Yöntemde orijinal veri setinde yer alan sayısal değişkenler kesikli hale getirilerek dönüştürülerek kategorik veri setine yerleştirilmektedir. Ardından Squeezer algoritması dönüştürülen veri seti üzerine uygulanarak kümeler belirlenmektedir[143].

Yöntem temelde iki adımdan oluşmaktadır. İlk adımda orijinal veri setinde (*D*) yer alan sayısal nitelikteki değişkenler kesikli hale getirilir ve dönüştürülen bu veri seti ise *D'* olarak tanımlanmaktadır. İkinci adımda ise bilinen Squeezer kümeleme algoritması *D'* veri seti için uygulanarak sonuç kümeler elde edilmektedir[143].

dSqueezer kümeleme yönteminin hesaplama karmaşıklığı iki parçadan oluşmaktadır. İlk parça orijinal veri setinin kesikleştirilmesi karmaşıklığı iken ikinci parça ise dönüştürülen veri seti üzerine Squeezer algoritmasının uygulanması sonucunda elde edilen karmaşıklıktır. Daha önceki geliştirilen Squeezer algoritmasında olduğu gibi dSqueezer yönteminin ikinci parçasının zaman karmaşıklığı *O(n*K*m)*'dir. Birinci parçanın karmaşıklığı ise *O(n)* olarak bilinmektedir. Dolayısıyla dSqueezer algoritmasının genel karmaşılığı *O(n*K*m)* olarak elde edilmektedir. Burada yer alan *n* değeri, veri seti genişliğini, *m* değişken sayısını, *K* ise sonuç küme sayısını ifade etmektedir. Squeezer kümeleme yöntemine benzer olarak dSqueezer algoritmasının da zaman karmaşıklığı veri seti genişliği, değişken sayısı ve sonuç küme sayısı ile doğru orantılı olması algoritmanın daha ölçeklenebilir olmasına neden olmaktadır[143].

2.1.6.14. usmSQUEEZER Kümeleme Algoritması

usmSqueezer kümeleme algoritması Zengyou He ve arakadaşiarı tarafından 2005 yılında geliştirilmiştir. Squeezer kümeleme algoritması sadece kategorik değişkenler için kullanılmakta idi ancak geliştirilen dSqueezer ve usmSquzeere yöntemleri yardımıyla karışık yapıdaki veri setleri için kümeleme işlemleri gerçekleştirilmektedir[143].

usmSqueezer kümeleme algoritması karmaşık veri seti içeren veri tabanları için birleştirilmiş bir benzerlik ölçüsü kullanmaktadır[143].

A_1^n ,A_2^n,….,A_p^n, A_{p+1}^c,…..,A_m^c seti D_1,….,D_m değişken setinde ilk *p* element sayısal değişken olarak kalan diğer elementler ise kategorik değişken olarak belirlenmektedir. Bu açıklamalar ışığında aşağıdaki tanımlamalar yapılmaktadır[143].

Tanım: Her bir A_i^n sayısal değişkeni için $(1 \leq i \leq p)$, A_i^n'nin en büyük değişken değeri $max(A_i^n)$ ve A_i^n'nin en küçük değişken değeri ise $min(A_i^n)$'dir. A_i^n'nin aralığı ise $interval(A_i^n)=max(A_i^n)-min(A_i^n)$ olarak tanımlanmaktadır[143].

Kategorik ve sayısal değişken içeren karma veri setleri için usmSqueezer kümeleme algoritması Squeezer kümeleme yapısında bazı değişiklikler uygulamaktadır. Bu değişiklikler ilerleyen tanımlamalarda incelenecektir[143].

Tanım: (Modifiye yapılmış özet, Modified Summary): *C* kümesi verildiğinde *C*'nin msummary değeri aşağıdaki gibidir[143].

$$mSummary=\{c_i|1 \leq i \leq p\} \cup \{VS_i|p+1 \leq i \leq m\}$$

Yukarıdaki formülde yer alan $c_i=(1/|C|)\sum_{j=1}^{|c|} b_{ij}$ değerine, $VS_i=\{(a_j, Sup(a_j))|a_j \in VALi(C)$'ye eşittir ayrıca b_{ij}, A_i^n'nin değişken değeri olarak tanımlanır.

Kategorik değişkenler için, değişken çiftlerini değeri ve destek değeri Squeezer kümeleme algoritması ile benzer şekildedir. Ancak sayısal değişkenler için ortalama değerler saklanmaktadır. Modifiye yapılmış özet yardımıyla, karışık değişken türlerinin içeren veri setleri için küme temsilcilerinin özetleri saklanmaktadır[143].

Tanım (Modifiye yapılmış Küme yapısı, Modified Cluster Structure, MCS): C kümesi verildiğinde *C* kümesi için modifiye yapılmış küme yapısı *CS={küme, mSummary}*'dir.

Tanım (Birleştirilmiş Benzerlik Ölçüsü, Unified Similarity Measure): C kümesi verildiğinde ve *tid∈TID* olmak üzere *t* veri grubu olduğunda *C* ve *tid* arasındaki benzerlik aşağıdaki gibi hesaplanmaktadır[143].

$$usmSim(C,tid)=\sum_{i=1}^{p}\frac{|a_i-c_i|}{Interval\ (A_i^n)}+\sum_{i=p+1}^{m}(\frac{Sup(a_i)}{|C|})$$

Formülde yer alan $a_i=tid*A_i$'ye eşittir.

Ayrıca *m* değişken sayısı olmak üzere birleştirilmiş benzerlik ölçüsü $0\leq usmSim(C,tid)\leq m$ arasında değişmektedir.

usmSqueezer kümeleme yöntemi veri setini iki kere taramaktadır. İlk taramada, her bir A_i^n ($1\leq i\leq p$) sayısal değişkeni için en büyük ve en küçük değişken değerleri tespit edilmektedir. Ardından bu iki değer arasındaki fark alınarak A_i^n aralığı hesaplanır. Veri setinin ikinci taramasında ise, Squeezer kümeleme algoritmasının karışık değişken tipleri içeren veri setleri için kullanılmasını sağlayan birleştirilmiş benzerlik ölçüsü kullanılmaktadır[143].

dSqueezer kümeleme yöntemine benzer olarak usmSqueezer algoritmasının da zaman karmaşıklığı *O(n*k*m)*'dir. Karmaşıklıkta yer alan *n* değeri veri seti genişliğini, *m* değişken sayısını ve *K* ise sonuç küme sayısını göstermektedir[143].

2.1.6.15. HIERDENC Kümeleme Algoritması (Hierarchical Density-Based Clustering of Categorical Data)

Daha önceden geliştirilen ROCK, CACTUS gibi algoritmaların çoğu etkili ve doğrudur. Ancak birçoğunda başlangıçta girdi parametrelerinin kullanıcı tarafından belirlenmesi gerekmektedir (yanlış değer seçimi ile kümeleme sonuçları kötü olacaktır). Ayrıca yöntemlerden elde edilen kümeler çok fazla olabilir ve çok fazla sapan değer içerebilmektedir. Bunun yanı sıra nesne girişi sırasına karşı duyarlıdırlar. Bu dezavantajları giderebilmek adına Andreopoulos ve arkadaşları tarafından HIERDENC kümeleme algoritması önerilmiştir[144].

HIERDENC kümeleme algoritması, kategorik değişken içeren veri setlerinin küme yapısını temsil eden bir hiyerarşi oluşturmaktadır. Başlangıçta kullanıcı tarafından belirlenen girdi

parametrelerini minimize etmektedir. Ayrıca nesne girdi sırasına karşı duyarlı değildir ve sapan değerlerle başa çıkabilmektedir[144].

m değişkenli bir kategorik veri seti, m-boyutlu kübik bir yapı olarak düşünülebilir. Bir küpün hücresi, kendi koordinatları ile aynı değere sahip nesne sayıları ile eşlenmektedir. Böyle bir küpte kümeler yoğun olan nesnelerin alt uzayı olarak kabul edilir ve düşük nesne yoğunluklu alt uzaylara ayrılmaktadır. Küpleri kümelendirmenin bazı zorlukları vardır[144].

1. Değişken değerlerinin sırası olmadığı için küp hücrelerinin de sıralaması yoktur. Ancak en iyi kümeyi belirlemek için, yoğun altuzayların araştırılmasında küpün her bir boyutunda bazı sıralamaların göz önüne alınması gerekmektedir.
2. Yoğunluk altuzayı yarıçap olarak adlandırılan bir parametre ile tanımlanmaktadır. Ancak küpin farklı altuzayları için bu yarıçapın farklı değerleri tercih edilmelidir. Yoğun alt uzaylarda düşük bir yarıçap ile işlemler gerçekleştirilirken dağınık yapıdaki altuzaylar için büyük yarıçaplar tercih edilmektedir. Kübün araştırılmasında düşük bir yarıçaptan başlanarak daha büyük yarıçaplara doğru bir araştırma da yapılabilmektedir.

HIERDENC kümeleme yöntemi, *m* kategorik değişkenlere sahip nesne setlerinin yersel yoğunluklarını temsil eden *m* boyutlu küpleri kümelemektedir. Yoğun altuzayları bulabilmek için, yarıçapın maksimum benzememezliği içerisinde kalan nesneler dikkate alınmaktadır[144].

m kategorik değişkenli $X_1, \dots, X_m$, *S* nesne seti olduğu varsayılsın. Her bir X_i değişkeni D_i domainine sahiptir. S^m uzayı domainlerin çapraz üretimleri olarak tanımlanan olasılıkların toplamlarını içermektedir. Bu $\prod_{i=1}^{m} d_i$ hücrelere sahip *m* boyutlu küp olarak görülmektedir[144].

λ fonksiyonu, $x=(x_1, \dots, x_m) \in S^m$'deki bir hücreyi *S*'de negatif olmayan nesne sayısına haritalandırmaktadır.

$$\lambda:\{(x_1, \dots, x_m) \in S^m\} \rightarrow N$$

x_0 hücresinde *r* yarıçaplı hiper küp $C(x_0,r) \subset S^m$ aşağıdaki gibi tanımlanmaktadır.

$$C(x_0,r)=\{x: x \in S^m \text{ ve } dist(x, x_0) \leq r \text{ ve } \lambda(x)>0\}$$

Dist(.) bir uzaklık fonksiyonudur. Bu yöntemde uzaklık ölçüsü olarak Hamming yöntemi kullanılmaktadır. Hamming yöntemine göre uzaklıklar aşağıdaki gibi hesaplanmaktadır[144].

$$HD(x,y)=\sum_{i=1}^{m}\delta(x_i,y_i),\ \delta(x_i,y_i)=\begin{cases}1, & eğer\ x_i \neq y_i\\ 0, & eğer\ x_i = y_i\end{cases}$$

Hamming uzaklık ölçüsü kategorik nitelikteki veriler için yaygın bir şekilde kullanılmaktadır[144].

$X \subset S^m$ altuzayının yoğunluğu, her bir X hücresinde değerlendirilen λ'ların toplamıdır. Bu yoğunluk aynı zamanda S'de rasgele nesneleri içeren hiper küplerin olasılığıdır ve aşağıdaki gibi formülize edilmektedir[143].

$$Density(X)=\sum_{c\in X}\frac{\lambda(c)}{|S|}$$

Yukarıdaki formülde yer alan |*S*| terimi, *S*'lerin genişliğidir.

HIERDENC kümeleme algoritması en yoğun $C(x_0,r) \subset S^m$ hiperküpünü araştırmaktadır. Bu hiper küp, x_0 merkezli, *S*'den rasgele nesneler içeren maksimum olasılığa sahiptir.

G_i ve G_j kümeleri arasındaki uzaklık, en yakın çift nesnelerinin uzaklığıdır ve $D(G_i, G_j)=minN\{dist(x,y): x\in G_i \ ve \ y\in G_j\}$ şeklinde tanımlanmaktadır. G_i ve G_j kümeleri $D(G_i, G_j)\leq r$ olduğunda r ile ilişkili olarak doğrudan bağlanabilirdir (directly connected).

Bu algoritma CLIQUE kümeleme yöntemine benzemektedir. Ancak HIERDENC, kategorik veriler için kullanılırken, CLIQUE sayısal veriler için uygulanmaktadır. HIERDENC kümeleme algoritması, girdi parametrelerini minimize eder, CLIQUE ise girdi parametresi olarak grid genişliği ve kümeler için global yoğunluk eşik değerini alır. Bunun yanı sıra, HIERDENC yönteminde farklı yoğunluklara sahip kümeler bulunabilmektedir. HIERDENC genellikle, kümenin merkez hiper-küpünü diğer geri kalan kümeden ayırmaktadır. Bunun sebebi ise kendi yüksek yoğunluğudur[144].

2.1.6.16. CLOPE Kümeleme Algoritması (A Fast and Effective Clustering Algorithm for Transactional Data)

CLOPE kümeleme algoritması Yang ve arkadaşları tarafından 2002 yılında geliştirilmiştir. Yöntem kategorik verilere uygulanmaktadır. Yüksek boyutlu veriler için oldukça kullanışlı bir algoritmadır. Daha önce yapılan çalışmaların çoğu kategorik nitelikteki veriler üzerinde yoğunlaşırken, kategorik veri tiplerinin bir alt çeşidi olan işlemsel veri üzerinde yoğunlaşmamıştır. Bu veri tipi için hızlı ve etkili bir kümeleme yöntemi bulmak veri yapısının yüksek boyutlu, seyrek olmasından dolayı oldukça zordur. Hiyerarşik kümeleme yöntemlerinden olan ROCK algoritması, kategorik verileri kümelemede etkilidir ancak büyük hacimdeki veri setlerinde etkili değildir. Dolayısıyla daha önceki geliştirilen kategorik veriler için kullanılan kümeleme algoritmalarının bahsedilen dezavantajlarını gidermek amacıyla CLOPE algoritması önerilmiştir[145].

CLOPE kümeleme algoritması, basit global bir kriter fonksiyonudur. Yöntem küme histogramlarının yükseklik-genişlik oranları artırarak küme içindeki işlem maddelerinin çakışmasını, örtüşmesini artırmaktadır. Histogramda daha büyük bir yükseklik/genişlik oranının küme içi benzerliklerinin daha fazla olduğunu göstermektedir. Bu basit düşünce CLOPE algoritmasının temelini oluşturmaktadır. Dolayısıyla algoritma küme histogramlarının temel özelikleri kullanılarak global bir kriter fonksiyonu tanımlanmaktadır[145].

CLOPE kümeleme algoritmasında *D* işlem veri seti tanımlanmaktadır ve bu set $\{t_1,\ldots..,t_n\}$ işlem setinden oluşmaktadır. Her bir işlem $\{i_1,\ldots..,i_m\}$ olan madde setinden oluşmaktadır. $\{C_1,\ldots,C_k\}$ kümesi $\{t_1,\ldots..,t_n\}$'nın bir bölümüdür. Dolayısıyla $C_1 \cup \ldots..\cup C_k$= $\{t_1,\ldots..,t_n\}$'dir ve $C_i \neq \varphi$ ($1 \leq i, j \leq k$). Her bir C_i bir küme olarak tanımlanır. Formüllerde yer alan *n, m* ve *K* indeksleri sırasıyla, işlem, madde ve küme sayısını göstermektedir[145].

CLOPE kümeleme algoritması basit bir yöntemdir ancak işlem veri kümelemesi için etkili bir global ölçüdür. Yöntemde grafiksel olarak yükseklik genişlik oranının yüksek olduğu durumlarda daha iyi bir kümeleme sonucunun oluştuğundan bahsedilebilmektedir[145].

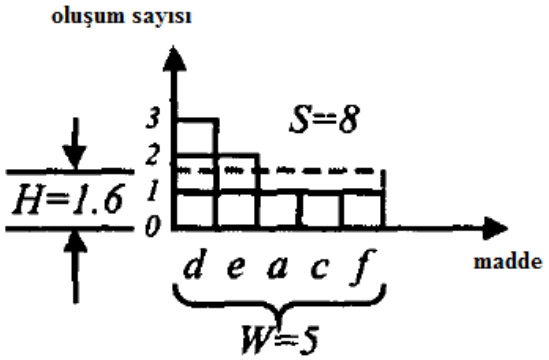
C kümesi olduğu varsayımı altında, *D(C)* ayrı (farklı) maddeler seti, *Occ(i,C), C* kümesindeki *i* maddenin meydana gelme sayısıdır. *X* ekseninde maddelere yer verilerek *C* kümesinin histogramı çizilebilmekte ve histogramdaki *Y* ekseni ise maddelerin görülme sayısına göre

sıralanmaktadır. Bir *C* kümesinin büyüklüğü *S(C)* ve genişliği *W(C)* olarak tanımlanmakta ve aşağıdaki formüller yardımıyla hesaplanmaktadır[145].

$$S(C)=\sum_{i\in D(C)} Occ(i,C) = \sum_{t_i\in C} |t_i|$$

$$W(C)=|D(C)|$$

Bir kümenin yüksekliği ise *H(C)= S(C)/W(C)* olarak tanımlanmaktadır. Histogramı daha da detaylandırmak üzere Şekil 2.1.6.16.1 incelendiğinde, *H(C), S (C)* ve *W(C)*'lerin histogramda nerelere karşılık geldiği görülmektedir[145].



Şekil 2.1.6.16.1. {*acd, de, def*} kümesinin ayrıntılı histogram gösterimi

CLOPE kümeleme algoritmasının gerçekleştirilmesi için tabi ki bu tanımlamalar yeterli olmayacaktır. Sadece yükseklik ve genişliklere bakarak kümeleme hakkında karar vermek objektif bir sonuç vermeyecektir. Dolayısıyla yapılan tanımlamalara ek olarak yeni tanımlamalarında yapılması gerekmektedir. Örneğin, {*ab, abc, acd*} kümesinin yüksekliği 2 ve {*acd, de, def*} kümesinin yüksekliği ise 1.6' dır. Ancak şu da bilindiği üzere ilk küme daha iyi bir kümedir. Çünkü benzer nesnelerin sayısı daha fazladır. CLOPE kümeleme yönteminde sadece yükseklikliere kriter fonksiyonunu tanımlamak için yeterli değildir. Çok basit bir örnek düşünülürse, {*abc, def*} veri tabanında örtüşme yoktur. Ancak {{*abc, def*}} ve {{*abc*},{*def*}} kümelerinin yükseklikleri aynıdır. Bu durumda farklı bir ölçünün kümelemede kullanılması gerekmektedir be nu ölçü gradiyenttir. Gradiyent ölçüsü aşağıdaki gibi tanımlanmaktadır[145].

$$Gradient\ G(C)= H(C)/W(C)=S(C)/W(C)^2$$

Bu durumda {{*abc*},{*def*}} kümesinin gradiyenti 1/3, diğer kümeninki ise 1/6 olarak elde edilmektedir. Gradiyenti düşük olan küme yani {{*abc*},{*def*}} kümesi daha iyi sonuçların verdiğini göstermektedir.

Bir kümenin kriter fonksiyonu tanımlanırken, işlemlerin sayısının yanında kümelerin şekillerinin de dikkate alınması gerekmektedir. $C=\{C_1,\ldots, C_k\}$ kümesi için aşağıda tanımlanan basit bir kriter fonksiyonu kullanılmaktadır[145].

$$Profit(C)=\frac{\sum_{i=1}^{k} G(C_i)*|C_i|}{\sum_{i=1}^{k}|C_i|}=\frac{\sum_{i=1}^{k}\frac{S(C_i)}{W(C_i)^2}*|C_i|}{\sum_{i=1}^{k}|C_i|}$$

Yukarıdaki kriter fonksiyonu $Profit_r(C)=\frac{\sum_{i=1}^{k}\frac{S(C_i)}{W(C_i)^r}*|C_i|}{\sum_{i=1}^{k}|C_i|}$ şeklinde genellenebilmektedir. Burada *r* terimi, pozitif gerçel bir sayıdır. Birçok durumda *r* terimi 1'den büyük alınmaktadır. Örneğin, {*abc, abcd, bcde, cde*} veri setinde {{*abc, abcd, bcde, cde*}} ve {{*abc, abcd*}, {*bcde, cde*}} kümeleri tanımlanmış olsun. İkinci kümeden daha iyi bir kar sağlamak amacıyla küme 2'nin faydası $\frac{\frac{7}{4^r}*2+\frac{7}{4^r}*2}{4}$, küme 1'in $\frac{\frac{14}{5^r}*4}{4}$ faydasından daha fazla olmalıdır. Bir başka deyişle *r* değeri ln(14/7)/ln(5/4)=3,106'dan büyük seçilmelidir[145].

Aksine seyrek veri tabanlarını gruplamak için *r*'nin küçük değerleri tercih edilmektedir. Örneğin, {*abc, cde, fgh, hij*} veri setleri için {{*abc, cde*}, {*fgh, hij*}}'den {{*abc*}, {*cde*}, {*fgh*}, {*hij*}} daha fazla fayda sağlanmak isteniyorsa ln(6/3)/ln(5/3)=1,357'den daha düşük bir *r* değeri seçilmelidir[145].

Daha önceki bölümlemeli kümeleme yaklaşımlarından bilindiği üzere, en iyi sonucu veri tabanının iteratif taraması ile gerçekleştirilmektedir. Ancak CLOPE algoritmasındaki kriter fonksiyonu global olarak tanımlanmaktadır, basit bir boyut ve genişlik ile kolayca hesaplanabilmekte ve lokal olan yöntemlerde de daha hızlı sonuçlar üretebilmektedir.

CLOPE kümeleme algoritmasında, ilk olarak başlangıç kümelerinin oluşturulabilmesi için veri seti bir kere taranmaktadır. Bu kümeler kriter fonksiyonu ile yönlendirilmektedir. Ardından kümeleme işlemini daha da hassaslaştırmak için taramalar birkaç kere gerçekleştirilmektedir. Eğer bir önceki yapılan tarama sonucu ile yeni yapılan tarama sonucu arasında bir değişiklik olmuyorsa, algoritma sonlandırılır ve sonuç kümeler çıktı olarak verilir[145].

CLOPE kümeleme yönteminin tek bir iterasyonda hesaplama karmaşıklığı *O(N*K*A)*'dır. Burada *A* işlemlerin ortalama uzunluğunu, *N* toplam işlem sayısını ve *K* ise maksimum küme sayısını ifade etmektedir. Dolayısıyla CLOPE'ın hızı, küme sayısından lineer olarak etkilenmektedir. *I/O* oranı ise veri seti genişliği ile doğrusal olarak ilişkilidir. Herhangi bir zamanda tek bir işlem hafızaya alındığı için, CLOPE algoritmasında yer gereksinimi küme özelliklerinin hafıza genişliğine bağlıdır[145].

Geliştirilen CLOPE kümeleme yöntemi, ilginç kümelerin bulunmasında oldukça etkilidir. Ayrıca algoritma veri sırasına da duyarlı değildir ve küme sayılarını kontrol ederek küçük bir domain bilgisi gerektirmektedir. Bu özellikleri sayesinde de algoritma işlem verileri için veri madenciliğinde oldukça etkili, kaliteli kümeleme sonuçları üretmektedir[145].

2.1.6.17. Bulanık CLOPE Kümeleme Algoritması (A Fuzzy CLOPE Algorithm)

CLOPE algoritması yüksek hız ve iyi performans açısından oldukça etkili bir yöntemdir. Ancak, bazı parametrelerin seçimi bu algoritmanın kümeleme sonuçlarının kalitesini etkilemektedir. Bu amaçla CLOPE yönteminin parametre seçiminden kaynaklanacak olan dezavantajını gidermek amacıyla bulanık CLOPE kümeleme algoritması 2006 yılında Jie ve arkadaşları tarafından geliştirilmiştir. Bulanık CLOPE yönteminde, en uygun parametre seçimi için modifiye edilmiş bölümlemeli bulanık derecesi tanımlanarak kümeleme geçerlik fonksiyonu kullanılmaktadır[146].

Geleneksel, klasik kümeleme yöntemleri kategorik veriyi sayısal veriye dönüştürerek işlemleri gerçekleştirmektedir. Ancak kategoriler sıralanmış değildir ve dolayısıyla klasik yöntemler kategorik verileri analiz etmede yetersiz kalmaktadır. Son yapılan araştırmalarda ise *K*-prototip ve CLOPE yöntemlerinin kategorik nitelikteki büyük veri setleri için kullanılabilir olduğunu göstermektedir[146].

Pratik uygulamalarda, birçok kümeleme algoritmasında *K* küme sayısı veya başka parametreler belirlenmektedir. Bu parametrelerin seçimi ise kümeleme sonuçlarını ve geçerliliğini önemli derecede etkilemektedir. Kümeleme geçerliliği, bulanık bölünmeye, geometrik yapıya ve veri setinin istatistiksel bilgisine dayanmaktadır. Ayrıca yöntemlerinin birçoğu kategorik verilerden ziyade sayısal veriler üzerine gerçekleştirilmiştir[146].

Bu amaçla bölünmeli bulanık derece yöntemi (Partition Fuzzy Degree, *PFD*), önerilmiştir. *PFD* yöntemi hem bulanık bölünme ile hem de veri setinin geometrik yapısı ile ilişkilidir.

Yeni modifiye edilmiş *PFD* fonksiyonu, bölünmeli entropi ve bulanık derecesini kullanan kategorik veri setleri için küme geçerlilik fonksiyonu olarak tanımlanmaktadır. Geliştirilen *PFD* fonksiyonu, parametre seçiminde önerilmekte ve kategorik veriler için gerçek denetimsiz bir kümeleme yöntemidir[146].

Yöntemin işleyişine geçmeden önce modifiye yapılmış *PFD* fonksiyonun nasıl hesaplandığına bakmakta fayda vardır. $X=\{x_1, x_2, \ldots, x_n\}$, n örnek hacimli veri seti olmak üzere, $x_i=[x_{i1}, x_{i2}, \ldots, x_{ip}]^T$, i. nesnenin p. özelliği olarak ifade edilmektedir. Bulanık kümeleme aşağıdaki matematiksel problem ile tanımlanmaktadır[146].

$$\min J_m(U,V)=\sum_{i=1}^{k}\sum_{j=1}^{n} u_{ij}^{m} d_{ij}^{2}$$

Yukarıdaki formülde yer alan K değeri $1<k<n$ arasında değişen kullanıcı tarafından önceden belirlenen küme sayısı iken m değeri $m>1$ olacak şekilde bulanık ağırlıklandırma üssüdür. $d_{ij}=||x_j-v_i||$ eşitliği x_j örneğinin, $v_i \subset R^p$'nin herhangi bir küme merkezine olan Öklid uzaklığını ifade etmektedir ($i \leq j \leq K$). u_{ij} terimi j. örneğin i. kümeye üyelik derecesini göstermektedir. Buna ek olarak $U=[u_{ij}]$, $K*n$ boyutunda bulanık bölünmeli bir matristir ve $V=[v_1, v_2, \ldots, v_K]$ $p*K$ düzeninde küme merkezi matrisidir[146].

Bulanık bölünmeli entropi değeri aşağıdaki gibi hesaplanmaktadır[146].

$$H(U;k)=-\frac{1}{n}\sum_{i=1}^{k}\sum_{j=1}^{n} u_{ij} log_a(u_{ij})$$

$a \in (1,\infty)$ olmak üzere logaritmanın tabanını ifade etmektedir ve $u_{ij}=0$ ise $u_{ij}(log)_a(u_{ij})$ değeri de sıfıra eşit olacaktır.

$1<K<n$ olmak üzere bulanık bölünmeli entropi değeri aşağıdaki özelliklere sahiptir.

$0 \leq H(U, K) \leq (log)_a K$

U kesin (crisp) bir matris ise $H(U, K)=0$'dır;

$U=[1/K]$ olduğunda $H(U, K)= (log)_a K$'dır.

Ω_c, olası optimum bölünmeli matris setini ifade ederken, aşağıdaki koşulu sağlayan (U^*, K^*) değerleri var ise, (U^*, K^*) optimum kümeleme sonucuna denk gelirken K^* optimum küme sayısını göstermektedir[146].

$$H(U^*,k^*)=min_k\{min_{\Omega_k}H(U;k)\}$$

Bölünmeli matrisin bulanıklık derecesini değerlendirmek için bölünmeli bulanık derecesinin tanımlanması gerekmektedir. *K* küme sayısı ve *U* bulanık bölünmeli matrisi verildiğinde, bölünme bulanık derecesi aşağıdaki gibi tanımlanmaktadır[146].

$$PF(U;k)=\frac{1}{n}\sum_{i=1}^{k}\sum_{j=1}^{n}|u_{ij}-(uij)_H|$$

$$(uij)_H=\begin{cases}1, u_{ij}=max_{1\le l\le k}\{u_{ij}\}\\0, \quad diğer\ durumlarda\end{cases}$$

$(uij)_H$ terimi bulanık bölünmeli matrisin sayıllaştırılmış sonuçları bulanık mantık formunda üretmektedir.

1<k<n için,

0≤PF(U;k)≤2-2/k;
U kesin bölünmeli matris ise *PF(U;k)*=0'dır.
U=[1/k] ise, PF(U;k)=2-2/k'dır.

PF(U, K) değeri ne kadar küçük değer alırsa, kümeleme sonuçları o kadar farklıdır. Dolayısıyla optimum küme sayısı *K*'yı elde edebilmek için *PF(U, K)* değerinin en küçük olması istenmektedir. Ancak *PF(U, K)* ve *H(U, K)* fonksiyonları *K* küme sayısının artması ile birlikte artma eğilimindedir. Bu amaçla, her iki fonksiyon kombine edilerek yeni bir modifiye yapılmış *PFD* fonksiyonu (*MPFD*) elde edilmektedir[146].

K küme sayısı ve *U* bulanık bölünmeli matrisi verildiğinde, bir veri seti için *MPFD*, *MPF(U, K)=(PF(U, K))/(H̃(U, K))* formülü yardımıyla hesaplanmaktadır. H̃(U, *K*) düzeltilmiş bulanık bölünmeli entropi değeridir[146].

Benzer şekilde *U* kesin bölünmeli bir matris olduğunda *MPF(U, K)=0*'a eşittir. Dolayısıyla modifiye edilmiş bu yöntem artan *K* sayısına karşı *PFD*'nin artışından kaynaklanan dezavantajları gidermektedir[146].

Bulanık CLOPE kümeleme algoritmasında $X=\{x_1, x_2, \ldots, x_n\}$, *n* örnek hacimli bir veri seti olarak tanımlanmaktadır. u_{ij}, *i.* kümeye ait olan x_j örneğinin derecesini göstermektedir.

$U=[u_{ij}]$, $k*n$ düzeninde bulanık bölünmeli bir matristir. $=\{x_1, x_2, \ldots, x_k\}$, X veri setinin K bulanık bölümüdür. H_i ise, X_i kümesinin kategorik değişkenlerinin histogramıdır ve aşağıdaki formüller yardımıyla işlemler gerçekleştirilmektedir[146].

$$S_f(X_i)=\sum_{x_j \in X_i} u_{ij}|x_j|$$

$$W_f(X_i)=|H_i|$$

Fayda ise,

$$\text{Max}\{\text{Profit}_{\text{fr}}(\text{X})=\frac{1}{n}\sum_{i=1}^{k}\frac{S_f(X_i)}{W_f(X_i)^r}|X_i|\}$$

Benzer olarak X verisinin özellik matrisi $C_x=x_1 \cup x_2 \cup \ldots \cup x_n$'dir. X_i bölümünün özellik matrisi ise C_{X_i} olarak ifade edilir. Dolayısıyla X_i'nin histogramı olan H_i aşağıdaki gibi tanımlanmaktadır.

$$H_i=\{\alpha_1, \alpha_2, \ldots, \alpha_{W_f(X_i)}\}$$

Yukarıdaki formülde yer alan α_l (l=1,2,…., $W_f(X_i)$), X_i bölümünün l özelliğini göstermektedir. Dolayısıyla H_i histogramından , x_j örneğin derecesi aşağıdaki gibi hesaplanır.

$$u_{ij}=\frac{\sum_{\alpha \in C_{x_i} \cap x_j} H_i(\alpha)-\Omega|x_j-C_{X_i}|}{|X_i|}$$

$\boldsymbol{\Omega}$ terimi bir sabittir ve penalized faktörü olarak bilinmektedir. Bu faktör $\boldsymbol{\Omega}\in(1,2)$ aralığında değişen değer almaktadır[146].

Bulanık CLOPE algoritmasının işlem adımları aşağıdaki gibi özetlenebilmektedir[146].

1. Adım: r ve $\boldsymbol{\Omega}$ parametreleri belirlenir.

2. Adım: x_j örneği seçilir ve bu örnek X_i bölümüne atanır, sırasıyla yeni bölümler X_l+1'lere karşılık delen *Profitfr(X)*'ler hesaplanır. Belirlenen en büyük *Profitfr(X)* optimum sınıflama olarak ilan edilir. Eğer $i=l+1$ ise, l=l+1 olur.

3. Adım: Her bir x_j örneğinin üyelikler değişmeyene kadar adım 2'ye devam edilir.

Uygun bir bölünme elde edebilmek için r parametresinin belirlenmesi zor bir durumdur. Bu problemi çözmek için yukarıda anlatılmış olan MPFD yöntemi kullanılmaktadır. MPFD, r'nin

bir fonksiyonudur. Optimum r^* ve optimum küme sayısı K^* ya karar verebilmek için CLOPE algoritması aşağıdaki kriter fonksiyonunu kullanmaktadır.

$$\text{MPF(U*,k*)}=min_r\{min_{\Omega_r} MPF(U;r)$$

Optimum r^* değeri bulunduktan sonra optimum küme sayına karşılık gelen K^* değeri de kolay bir şekilde bulunabilmektedir[146].

2.1.6.18. MULIC Kümeleme Algoritması (Multiple Layer Clustering)

MULIC kümeleme algoritması, kategorik veriler için kullanılmaktadır. MULIC algoritması, zaman etkisi ile birlikte doğruluğu dengelemeye çalışan ve HIERDENC kümeleme yönteminden daha hızlı sonuç veren bir yöntemdir[144].

MULIC kümeleme algoritması, küpleri ana hafızda saklamaz ve zamanı azaltmak için basitleştirilmiş yöntemler kullanmaktadır. İlk olarak yoğun bir alandan kümelemeye başlar ve φ değişkeni ile temsil edilmiş bir yarıçap ile dışa doğru genişler. MULIC küme genişliğinde olduğunda, HIERDENC kümeleme algoritması gibi tüm üye nesneleri araştırmamaktadır. Ancak yine de kümelerin içeriğini özetlemek için mod kullanır. Bir *c* kümesinin modu $\mu_c=\{\mu_{c1},\ldots..,\mu_{cm}\}$ vektörüdür. Burada μ_{ci}, verilen *c* kümesi içindeki *i.* değişkenin en sık aldığı değerdir. MULIC algoritmasında *o* nesnesi kümelendiğinde, nesnenin eklendiği *c* kümesi en düşük benzemezlik modu μ_c'ye sahiptir. *o* ve μ_c arasındaki benzemezlik ölçüsü ise Hamming uzaklığı ile hesaplanmaktadır. Ancak herhangi bir ölçüde kullanılabilir[144].

MULIC kümeleme algoritmasında var olan kümeler içine nesneleri yerleştirmek için maksimum benzemezlik kriteri φ kullanılarak, bir küme kademeli olarak düzenlenmiş tabakalardan oluşmaktadır. MULIC'de kullanıcının başlangıçta küme sayısını belirtmesine ihtiyaç yoktur ve algoritma uç değerleri belirleyebilmektedir[144].

Bazı durumda iki kümenin üst tabakalarının benzerliği, herhangi iki kümenin alt ve üst tabakalarının benzemezliklerinden daha küçük olabilmektedir. Bu sakıncadan kaçınmak için MULIC, iki kümenin üst tabaka modlarının benzememezlikleri maksimum tabaka genişliğinden daha küçük olan küme çiftlerini birleştirmektedir. Böylece MULIC tüm kümelerin üst tabaka modlarını da korumaktadır. ϕ benzememezlik ölçüsü içerisinde bulunan iki veya daha fazla nesne var ise MULIC bir küme oluşturmaktadır. MULIC'de kümelenen başlangıç nesneleri modu ve kümelemeyi etkilemektedir. Dolayısıyla yöntemde nesne

sıralama süreci kullanılmaktadır. Bu süreçte ilk olarak daha yoğun olan nesneler kümelendirilir ve modlar en yoğun olan değerleri içerecektir. Ancak rasgele nesneler sıralandığında aynı sonuçlar elde edilmeyecektir[144].

MULIC kümeleme algoritmasında kullanılan mod, kümelerin içeriklerinin bir özetini vermektedir ve sadece moddan φ uzaklık içerisinde kalan nesneleri kümelemektedir. HIERDENC kümeleme yönteminin herhangi bir r değeri için yeni kümeler üretmesi gibi MULIC'de ϕ herhangi bir değeri için yeni kümeler oluşturmaktadır. Algoritma φ değerini artırarak düzensiz şekillere sahip kümeleri bulabilmektedir. MULIC kümeleme yönteminin, diğer hiyerarşik algoritmalarda olduğu gibi kümeleri elde etmek için bir kesim noktasına ihtiyacı yoktur. Bu algoritmanın avantajıdır. Yöntem, nesnelerin kümelenmesi sırasında kümeleri birleştirmemektedir. Bunun yerine uygulamada bazı kümelerin birleştirilmesi isteniliyorsa bu işlem tüm nesneler kümelendikten sonra gerçekleştirilmektedir. MULIC algoritması kümeleme süresinde, bazı büyük kümelerin birleştirilmesinden kaynaklanabilecek olan küme yapısının kaybedilmemesini amaçlamaktadır[144].

2.1.6.19. DILCA Kümeleme Algoritması (Distance Learning of Categorical Attribute)

DILCA, kategorik değişken değerleri arasındaki uzaklıkları hesaplar ve bu tekniği hiyerarşik bir yaklaşım ile uygulamaktadır. Kategorik veriler için iki değer arasındaki uzaklıkların hesaplanması, kategorik değişken değerlerinin sıralanmış olmaması sebebiyle çok net değildir. İki değer arasında farklılıkların hesaplanması direk olarak olası değildir. Kategorik değişkenler için en basit ölçü çakışma (overlap) ölçüsüdür. Bu ölçü, eşleştirilmiş iki nesnedeki değişken sayısıyla orantısal olarak artar. Çakışma ölçüsü sadece değer çiftleri arasındaki eşitliği ölçmektedir. Ancak farklı değer alanlar arasındaki ayrımı yapamamaktadır. Herhangi farklı değer çiftleri için benzerlik değeri değişmemektedir. Dolayısıyla yeni bir ölçünün önerilmesi ihtiyacı doğmuştur[147].

$D=\{d_1,d_2,.....,d_n\}$ veri setinin $F=\{X,Y,....Z\}$ olmak üzere m kategorik değişkenden oluştuğu varsayılsın. X kategorik bir değişken olmak üzere değerleri arasındaki uzaklıklar hesaplanmak istenmektedir. DILCA yöntemi iki ana aşamadan oluşmaktadır. İlk aşama tek bir değişken bakımından diğer tüm değişkenlerin ilişkili alt setlerini seçmektir. İkinci aşama ise aynı değişkenin değerleri arasındaki uzaklık ölçüsünün hesaplanmasıdır. İlk aşamayı gerçekleştirmek için simetrik belirsizlik ölçüsü (symmetric uncertainty: SU)

kullanılabilmektedir. Bu ölçü iki değişkenin ortak bağımlılıklarını ölçmektedir. Hesaplamada payda yer alan değer, ortak bilgidir[147].

Bu belirsizlik değişkenler üzerindeki toplam belirsizlikler yardımıyla normalleştirilmektedir ve aşağıdaki formül elde edilmektedir.

$$SU(X,Y) = 2 * \frac{IG(X|Y)}{H(X) + H(Y)}$$

$$H(Y) = -\sum_i P(y_i) log_2(P(y_i)), \quad IG(X|Y)=H(X)-H(X|Y)$$

Yukarıdaki formülde yer alan *H(X)* ve *H(Y)* terimleri entropileri ifade etmektedir. *SU(X,Y)* ölçüsü 0 ile 1 arasında değişmektedir. Eğer bu eşitlik 1 değerini alırsa, iki değişkenden biri hakkında bilgi var ise diğerinin değeri tamamen tahmin edilebilmekte, 0 değerini alırsa *X* ve *Y* bağımsızdır. *SU* ölçüsünün diğer yöntemlerden avantajı entropi farklılarına dayanmasıdır. Ayrıca bu ölçü değişkenlerin değerlerinin sayısına karşıda yanlı değildir[147].

X değişkeni verildiğinde bağlam değişkenler seti *context(X)* seçilmek istenmektedir. *Context(X)* en iyi *SU* değerine sahiptir. *X*'nin bağlamı olarak ele alınan değişken sayıları problemin çözümü için oldukça önemlidir. *SU* değerleri ile körele olan bağlam değişkenlerinin sayısı hesaplanmak istenmektedir. Bu amaç ile aşağıdaki formül kullanılmaktadır. Bu eşitlik *SU*'nun beklenen değerine dayanmaktadır. *X* değişkeni bilindiğinde, her bir Y değişkeni için *SU*'ların hesaplanması istenmektedir. Bu ölçü $SU_X(Y)$ ile ifade edilmektedir[147].

$$E[SU_X] = \frac{\sum_{Y \in context(X)} SU_X(Y)}{|context(X)|}$$

Yukarıdaki formülde geçen |*context(X)*| terimi, *context(X)* setinin önemlilik düzeyidir. *X* bağlamlarına karar vermek amacıyla aşağıdaki eşitliği sağlayan değişkenler kullanılmaktadır.

$$SU_X(Y) \geq \sigma\, E[SU_X]$$

σ, kullanıcı tarafından belirlenen bir parametredir. Eşik değerin seçilmesinde ortlamaya etkilerin kontrol edilmesi için oldukça kullanışlı bir parametredir. Bu parametrenin değerleri 0 ile 1 arasında değişmektedir[147].

İkinci aşama ise aynı değişkenin değerleri arasındaki uzaklık ölçülerinin hesaplanması aşaması idi. Bu aşamada, hedef kategorik değişkenin her bir değer çiftleri arasındaki uzaklıklar hesaplanmaktadır. *Y* bağlam değişkeninin y_K değeri bilindiğinde, *X*'in x_i değerinin koşullu olasılık değeri $P(x_i|y_K)$ olarak ifade edilmektedir. DILCA yönteminde x_i ve x_j arasındaki uzaklık değeri aşağıdaki formül yardımıyla hesaplanmaktadır[147].

$$dist(x_i, x_j) = \sqrt{\sum_{Y \in context(X)} \sum_{y_k \in Y} (P(x_i|y_k) - P(x_j|y_k))^2}$$

Her bir *Y* bağlam değişkeni ve her bir $y_K \in Y$ için x_i ve x_j değerlerinin koşullu olasılıkları arasındaki farklılıklar hesaplanmaktadır. Ardından sonuç uzaklıkların hesaplanması için Öklid uzaklığı kullanılmaktadır[147].

Sonuç olarak DILCA yöntemi, veri madenciliğinde sıklıkla kullanılan nominal veri türleri için kullanılmaktadır. Context'e dayalı bir uzaklık ölçüsüdür. Böylece bu yöntem ile kategorik verileren kaynaklanabilecek yanlış hesaplamalar giderilmektedir[147].

2.1.7. Olasılık Modellerine Dayalı Kümeleme Algoritmaları (probabilistic model-based clustering techniques)

Olasılık modeline dayalı kümeleme yöntemleri görüntü segmentasyonu, el yazı tanıma, doküman kümelemeye kadar birçok uygulamada yaygın bir şekilde kullanılmaktadır. Modele dayalı kümeleme yaklaşımları olasılık yaklaşımlarını kullanarak gözlenen verilerle, bazı matematiksel modeller arasındaki uyumu en iyi duruma getirmektedir. Bazı yöntemler, genellikle verinin ilgili olasılık dağılımlarının karmasından oluşmaktadır. Uygulamada her bir küme parametrik olasılık dağılımları (Gaussian veya Poisson dağımları) ile matematiksel olarak ifade edilebilir. Dolayısıyla kümeleme problemi artık parametre tahmini problemine dönüşmektedir. Böylece veri, *K* karmaşık yapıda bileşen dağılımları ile modellenebilmektedir[148-9].

2.1.7.1. Karmaşık modeller

Kümeleme analizinde karmaşık modeller farklı şekillerde ele alınmaktadır. Kümelenecek olan gözlemler çeşitli bileşenlerin birinden alınmıştır ve problem veriye en iyi uyacak her bir bileşen için parametre tahminlerini yapmaktır. Dolayısıyla ilk olarak karmaşık model

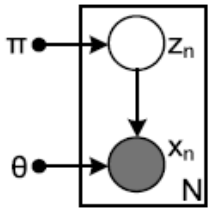
yapısından bahsedilecek ardından en çok kullanılan Gaussian ve Bernoulli karmaşık modelleri üzerinde durulacaktır[148-9].

N gözlemden oluşan X=$\{x_1, \ldots, x_N\}$ veri noktalarının olduğu varsayılsın x_n rasgele değişkeni, k bileşenin karmaşıklığına göre dağılmaktadır. Her bir bileşen matematiksel olarak bir parametre dağılımı ile temsil edilmektedir. Formül olarak x_n'in karmaşık dağılım veya olasılık yoğunluk fonksiyonu aşağıdaki gibi yazılmaktadır[148-9].

$$p(x_n) = \sum_{k=1}^{K} \pi_k p(x_n|\theta_k)$$

Yukarıdaki formülde her alan $\pi_1, \ldots, \pi_k$ değerleri olasılıkların karışımıdır (mixing probabilities). Her bir θ_k, k. bileşenin belirtilen parametre setidir ve $p((x_n|\theta_k)$ ise bileşen dağılımıdır. Geçerli olasılıklar bulabilmek için $\{\pi_k\}$'nın aşağıdaki koşulları sağlaması gerekmektedir[148-9].

$0 \leq \pi_k \leq 1$ ($k=1, \ldots K$) ve $\sum_{k=1}^{K} \pi_k = 1$



Şekil 2.1.7.1.1. Karmaşık modelin grafiksel gösterimi, yuvarlaklar rasgele değişkenleri ve gölgeli olan ve olmayan şekiller ise gözlenen ve gizli (gözlenemeyen) değişkenleri göstermektedir.

z_n , kategorik rasgele değişkeninin olasılığı $p(z_n = k) = \pi_k$ olmak üzere 1,….,K değerlerini almaktadır. x_n'in koşullu olasılığı $z_n = k$ olduğunda $p(x_n|\theta_k)$'ya eşittir. x_n'in marjinal olasılığı, p(x_n)'deki tüm olası z_n'lerin ortak dağılımlarının toplanması ile elde edilmektedir. Dolayısıyla z_n, x_n rasgele değişkeninin bir bileşeni olarak düşünülmektedir. x_n'in kategori etiketlerini göstermek için tek bir kategorik z_n değişkeni yetine, K boyutlu ikili rasgele z_n vektörü kullanılmaktadır.z_{nk}=$(z_n)_k$ olduğunda element bire eşittir diğer elementleri ise 0'a eşittir. Örneğin, K=5 kümeli bir değişken olduğu varsayılsın ve z_{n4}=1 olduğu durumda kümeye karşılık gelen x_n gözlenmiş olsun. Bu durumda z_n=$(0,0,0,1,0)^T$ olarak gösterilecektir. $0 \leq z_{nk} \leq 1$'dir ve $\sum_{k=1}^{K} z_{nk} = 1$. Dolayısıyla z_n'in marjinal olasılığı, π_k açısından aşağıdaki gibi tanımlanmaktadır[148-9].

$p(z_n)=\pi_1^{z_{n1}}\pi_2^{z_{n2}}\ldots\ldots\pi_K^{z_{nK}}=\prod_{k=1}^{K}\pi_K^{z_{nK}}$

z_n bilindiği durumda x_n'in koşulla dağılımı $p(x_n|z_n)=\prod_{k=1}^{K}p(x_n|\theta_k)^{Z_{nk}}$'dır.

Ortak dağılım $p(z_n)\,p(x_n|z_n)$ hesabı ile bulunmaktadır ve x_n'in marjinal olasılığı aşağıdaki gibi elde edilir[148-9].

$$p(x_n)=\sum_{z_n}p(z_n)p(x_n|z_n)=\sum_{k=1}^{K}\pi_k p(x_n|\theta_k)$$

Yukarıda verilen x'in marjinal dağılımı gizli değişkeni de içeren karmaşık bir modelin formülasyonunda denk gelmektedir.

Verilen bir veri noktaları seti, N kere süreç tekrarlanarak üretilebilir ve her bir veri noktası x_n için bir kez,

*Gizli bileşen etiketi olan $z_n\sim Mult_K(1,\pi)$ seçilir. Bu seçim x_n'den k. Bileşeni seçmektedir.

*Koşullu olasılık $p(x_n|\theta_k)$'ya bağlı olarak k. Bileşenden bir veri noktası x_n seçilir.

Marjinal dağılım $p(x_n)=\sum_{z_n}p(x_n,z_n)$ formunda gösterildiğinde, her bir gözlenen veri noktası x_n için, karşılık gelen gizli bir değişken z_n vardır. Bayes teoremi kullanılarak, x_n bilindiğinde, z_{nk}=1 durumunda koşullu olasılık değeri aşağıdaki eşitlikler yardımıyla hesaplanmaktadır[148-9].

$$p(z_{nk}=1|x_n)=\frac{p(z_{nk}=1)p(x_n|z_{nk}=1)}{\sum_{j=1}^{K}p(z_{nj}=1)p(x_n|z_{nj}=1)}=\frac{\pi_k p(x_n|\theta_k)}{\sum_{j=1}^{K}\pi_j p(x_n|\theta_j)}$$

Yukarıdaki formülde yer alan π_k, x_n veri noktasının k bileşenden üretilen önceki (prior) olasılığıdır, $p(z_{nk}=1|x_n)$ ise k bileşenden gelen gözlenen x_n veri noktasının sonraki (posterior) olasılığını vermektedir. Bir sonraki anlatımlarda $p(z_{nk}=1|x_n)$ eşitliği γz_{nk} olarak gösterilecektir[148-9].

Karmaşık modelin genel parametreleri $\Theta=\{\pi_1,\ldots.\pi_K,\theta_1,\ldots.,\theta_K\}$ olarak ifade edilmektedir. Veri noktalarının bağımsız olarak dağılımdan üretildiği varsayımı altında, tüm veri noktalarını üretme olasılığı aşağıdaki gibidir[148-9].

$$p(X|\Theta)=\prod_{n=1}^{N}\sum_{k=1}^{K}\pi_k p(x_n|\theta_k)$$

Eşitliğin her iki tarafının logaritması alındığında,

$\log p(X|\Theta)=\sum_{n=1}^{N} log \sum_{k=1}^{K} \pi_k p(x_n|\theta_k)$ olmaktadır.

Parametre tahminleri için maksimum olabilirlik tahmini istatistiksel yaklaşım olarak kullanılabilmektedir[148-9].

Θ_{ML}=arg$max_{\Theta}\{logp(X|\Theta)\}$.

Ancak bazı durumlarda parametreler ile ilgili olarak $p(\Theta)$ bilgisi önceden bilinebilir ve karmaşık modeller içerisine dâhil edilebilir. Bunun yerine maksimum sonraki tahmin (MAP) tahmin yöntemi kullanılır ve aşağıdaki eşitlikler yardımıyla hesaplanmaktadır[148-9].

Θ_{MAP}= arg$max_{\Theta}\{ logp(X|\Theta)+logp(\Theta)\}$

MLE ve MAP parametre tahminleri için birleştirilmiş bir yaklaşımdır. Bu yaklaşım normal dağılma durumunda ve birçok diğer problemlerde iyi tanımlanmaktadır.

Yukarıda anlatılan temel kavramların ardından kümeleme algoritmalarında en çok kullanılan karmaşık modellerden olan Gaussian ve Bernoulli karma modelleri sırasıyla bahsedilecektir[148-9].

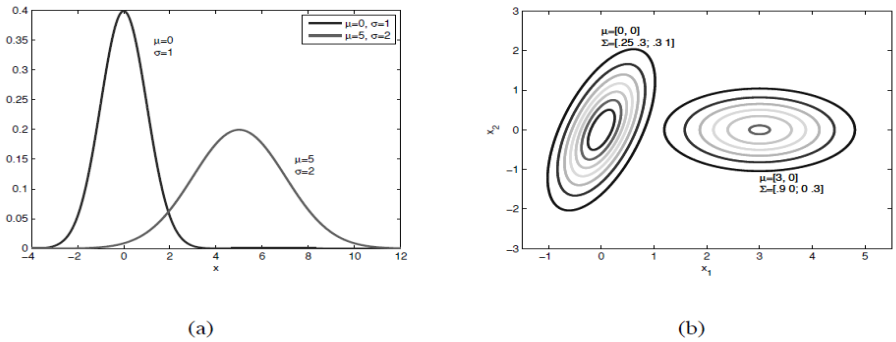
Gaussian karmaşık modeli

Gaussian karmaşık modeli (GMM), en çok bilinen karma modellerdendir. Her bir bileşen Gaussian dağılımına uymaktadır. Birçok uygulamada kümeleme algoritmalar için GMM yaygın bir şekilde kullanılmaktadır[148-9].

Gaussian aynı zamanda normal dağılım olarak da bilinmektedir. Bu model sürekli değişkenlerin dağılımları için çok kullanılan bir modeldir. Tek bir x durumunda, Gaussian dağılımı aşağıdaki formda ifade edilebilmektedir[148-9].

$N(x|\mu,\sigma^2)=\frac{1}{(2\pi\sigma^2)^{1/2}} \exp\{-\frac{1}{2\sigma^2}(x-\mu)^2\}$

Bilindiği üzere μ ortalamayı, σ^2 ise varyansı ifade etmektedir. Aşağıdaki şekilde parametrelerin farklı değerleri için Gauss dağılımı şekilsel olarak gösterilmektedir[148-9].



Şekil 2.1.7.1.1.2. Farklı parametre değerleri için Gauss dağılımı

Gaussian dağılımda μ_k=$\frac{\sum_{n=1}^{N}\gamma(z_{nk})x_n}{\sum_{n=1}^{N}\gamma(z_{nk})}$ ve $\gamma(z_{nk}) = \frac{\pi_k N(x_n|\mu_k,\Sigma_k\ )}{\sum_{j=1}^{K}\pi_j N(x_n|\mu_j,\Sigma_j\ )}$'dir.

2.1.7.1.1. Bernoulli karma modeli

Karma modellerden birçok araştırma Gaussian gibi sürekli değişkenler için kullanılan dağılımlar üzerinde yoğunlaşsada, birçok kümelemede ikili ve kesikli değişkenleri içeren karma modeller kullanılmaktadır. Bu kesikli değişkenlerin karma modellerinin incelenmesinde Bernoulli dağılımı kullanılacaktır. Bu dağılım aynı zamanda latent sınıf analizi olarak da bilinmektedir[148-9].

Bernoulli karma modeli, karma modellerin özel bir versiyonudur. Her bir k bileşeni, $\mu_k = \{\mu_{k1}, \ldots\ldots, \mu_{kD}\}^T$ parametreli olmak üzere D boyutlu Bernoulli dağılımı aşağıdaki gibidir[148-9].

$p(x|\mu_k)$=$\prod_{i=1}^{D}{\mu_{ki}}^{xi}(1-\mu_{ki})^{1-x_i}$

K bileşen sayısı olmak üzere, K Bernoulli bileşenli karmaşık model formu $p(x|\mu,\pi) = \sum_{k=1}^{K}\pi_k p(x|\mu_k)$'dır. Burada μ=$\{\mu_1,\ldots., \mu_K\}$ ve π=$\{\pi_1, \ldots\ldots, \pi_K\}$ olmaktadır.

X=$\{x_1, \ldots\ldots, x_N\}$ veri seti verildiğinde, model için log-olabilirlik aşağıdaki gibi hesaplanmaktadır[148-9].

$$logp(X|\mu,\pi) = \sum_{n=1}^{N}\log\{\sum_{k=1}^{K}\pi_k\prod_{i=1}^{D}{\mu_{ki}}^{x_{ni}}(1-\mu_{ki})^{1-x_{ni}}$$

2.1.7.1.2. EM Kümeleme Algoritması

EM algoritması expectation-maximization kelimelerinin kısaltmalarından oluşmaktadır. Bu algoritma ortalama, kovaryansları içeren parametreler hakkında tahminler yapmak ile başlamaktadır. Ardından beklenen adım (expectation) ve en büyükleme (maximization) adımları olmak üzere iki adım alternatifi vardır. Beklenen adımda (E-adımı), önceki olasılıkları hesaplamak için var olan parametreleri kullanmaktadır. En büyükleme adımında (M-adımı) güncellenen olasılıklar yardımıyla log-olabilirlik maksimum yapılarak ortalama ve kovaryanslar yeniden tahmin edilir. E ve M adımları yakınsama kriteri sağlanana kadar döngüsel olarak devam etmektedir[148-9].

EM algoritması gizli değişkenli olasılık modellerinin MLE veya MAP tahminlerini bulmak için popüler iteratif bir yöntemdir. EM kümeleme algoritmasının E aşamasında gizli z_{nk} değişkenlerinin değerleri tahmin edilmeye çalışılır. M aşamasında ise E aşamasındaki tahminlerine dayanarak model parametreleri güncellenir. Gaussian karma modeli ortalama (μ), kovaryans matrisi ($\sum$) ve karmaşık ağırlıkları (π) içermektedir. E aşamasında, z_{nk} gizli değişkeninin posterior olasılıkları hesaplanır. Bayes teoremi kullanılarak, aşağıdaki eşitlik elde edilir[148-9].

$$p(z_{nk} = 1|x_n, \mu_k, \Sigma_k) = \frac{\pi_k N(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x_n|\mu_j, \Sigma_j)} = \gamma(z_{nk})$$

M aşamasında ise, $Q(\Theta,\Theta^{(t)})$ miktarı maksimize edilmesi gerekmektedir. Bağımsızlık varsayımı altında, tamamlanmış verinin log olabilirlik bekleneni aşağıdaki gibidir[148-9].

$$Q(\Theta,\Theta^{(t)}) = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk})^{(t)} log \pi_k N(x_n|\mu_k, \Sigma_k)$$

Ortalama vektörü $\{\mu_k\}$ için başlangıç değerleri seçilir. Ardından aşağıdaki iki aşama tekrarlanmaktadır[148-9].

E aşaması: Maliyet en aza indirgenerek her bir veri noktası en yakın kümeye atanır (r_{nk} ve μ_k $sabitlenerek$).

M aşaması: Kümelerler için ortalama vektörler güncellenir.

E aşamasındaki r_{nk} aşağıdaki gibi hesaplanmaktadır[148-9].

$$r_{nk}=\begin{cases}1, eğer\ k = argmin_j \left|x_n - \mu_j\right|^2 \\ 0, \qquad\qquad diğer\ durumda\end{cases}$$

M aşamasındaki μ_k ise $\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$ eşitliği ile güncellenir.

Gaussian dağılımı ardından, Bernoulli karmaşık modeli parametreleri tahmin edilerek olabilirlik fonksiyonları maksimum yapılarak EM algoritması kullanılabilir. Gizli değişkenin posterior dağılımı dikkate alınarak tamamlanmış verinin log-olabilirliğinin bekleneni aşağıdaki gibi hesaplanmaktadır[148-9].

$Q(\Theta,\Theta^{(t)})=\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})\{log\pi_k + \sum_{i=1}^{D}[x_{ni} log\mu_{ki} + (1-x_{ni})\log(1-\mu_{ki})]\}$

$\gamma(z_{nk})$ posterior olasılıktır. EM algoritmasının E şamasında, Bayes teoremi kullanılarak tahmin edilen posterior olasılık aşağıdaki gibi elde edilmektedir[148-9].

$$\gamma(z_{nk}) = \frac{\pi_k p(x_n|\mu_k)}{\sum_{j=1}^{K}\pi_j p(x_n|\mu_j)}$$

M aşamasında ise, yukarıdaki log olabilirlik fonksiyonu maksimum yapılarak μ_{ki} ve π_k elde edilir[148-9].

$$\mu_{ki}=\frac{\sum_{n=1}^{N}\gamma(z_{nk})x_{ni}}{\sum_{n=1}^{N}\gamma(z_{nk})},\ \pi_k=\frac{\sum_{n=1}^{N}\gamma(z_{nk})}{N}$$

EM algoritmasına başlamak için, parametreler için başlangıç değerleri gerekmektedir. Bir başlangıç noktası kullanmak kaydıyla, EM algoritması log olabilirliğin değerini her zaman artırmaktadır ve algoritma lokal maksimumu garanti etmektedir. Genel olarak karışık olasılıklar $\pi_k = \frac{1}{K}$ 'dir ve μ_{ki} parametresi, (0.25, 0.75) arasında tek düze olarak değişen rasgele değerler setidir. Ardından $\sum_i \mu_{ki}$=1 kısıtının sağlanması için normalleştirilmektedir[148-9].

EM algoritmasının her bir döngüsünde olabilirlik fonksiyon değeri de artmaktadır. EM kümeleme yönteminin anlaşılması kolay ve uygulama adımları da oldukça basittir. Ayrıca diğer yöntemlere oranla bu algoritmanın maliyeti daha azdır. Kayıp veriler olduğu durumda bu verileri tahmin edebilme özelliğine sahiptir. Ancak parametre tahminlerinin kovaryans matrisinin bir tahminin direk olarak hesaplayamamaktadır ve bazı durumlarda yakınsama durumu çok yavaş gerçekleşmektedir[148-9].

3. MATERYAL VE METHOD

3.1. Veriler

Kardiyovasküler hastalıklar, son yıllarda ülkemizde en çok rastlanan rahatsızlıklar arasında yer almaktadır. Kardiyovasküler terimi, kalp ve damar rahatsızlıklarının genel adı olarak kullanılır. Kardivasküler rahatsızlıklar, ölüm nedenleri arasında yer aldığı gibi kişinin yaşam kalitesinin de önemli ölçüde bozulmasına yol açmaktadır. Dolayısıyla kardiyovasküler sorunların artışını engellemek ve bu sorunlar nedeniyle meydana gelen ölümleri azaltmak için risk faktörlerinin belirlenmesi ve bu risk faktörlerinden hareketle riskli olan kişilere gerekli müdahalelerde bulunularak kişinin kardiyovasküler hastalık nedeniyle meydana gelebilecek ölüm risklerinin azaltılması gerekmektedir. Kardiyovasküler hastalıkların risk faktörleri (özellikler) çok çeşitlidir. Başlıca risk faktörleri, ileri yaş, sigara ve alkol kullanımı, obezite, diyabet rahatsızlığının var olması, hipertansiyon, hiperkolesterolemidir. Dolayısıyla hareketsiz yaşam içinde bulunan, spor yapmayan, hazır gıda tüketen, sağlıklı beslenmeyen, alkol ve sigara kullanan bir kişinin kardivasküler rahatsızlık geçirme riski oldukça fazladır.

Ülkemizde bu hastalıkların değerlendirilmesinde oldukça basit yollar izlenmektedir. Örneğin kişinin kolesterol değeri yüksek ise hemen kolesterol ilaçlarına başlanabilmektedir. Oysaki kişi belki de o kolesterol değeri ile kaliteli bir yaşam sürebilmektedir. Kişinin biyolojik mekanizmasına göre bu değer değişebilmektedir. Dolayısıyla aslında hastaya basit bir ilaç verirken bile hastalığı etkileyen tüm risk faktörleri aynı anda incelenerek kişinin durumu hakkında karar verilmelidir. Böylece daha doğru ve kaliteli tedavi yöntemleri izlenecektir.

Bu tez çalışmasında, kardiyovasküler hastalıklara karşı kişilerin risk durumlarını değerlendirirken, dünyada yaygın kullanılan yöntemler arasında yer alan Framingham risk skoru kullanılmıştır. Framingham risk skoru, Amerika'da Massachusetts eyaletinde yer alan Framingham kasabasında yaşayan bireyler üzerinde ileriye dönük bir çalışma sonucunda hesaplanmıştır. Amerikan Kalp Birliği ise bu verileri kullanarak Framingham risk cetveli oluşturmuştur. Bu cetveli oluşturma amacı ise kardiyovasküler risk altındaki bireyleri tespit ederek gerekli önlemleri alıp ölüm oranlarını azaltmaktadır. Ancak bazı durumlarda bu risk skorunu hesaplamaya gerek yoktur. Örneğin kişi yakın bir zamanda koroner arter rahatsızlığı geçirmişse zaten 10 yıllık riski %20'nin üzerinde olacaktır. Benzer şekilde bir kişi risk faktörlerinden en az ikisine sahip değilse o kişi için de risk değerlerinin hesaplanmasına ihtiyaç yoktur. Çünkü bu kişinin kardiyovasküler rahatsızlığı geçirme riski olukça düşük çıkacaktır.

Framingham risk cetvelini eleştirenlerin sayısı da oldukça fazladır. Araştırmacılar bu risk cetvelinin sadece koroner kalp hastalığı riskini tahmin ettiği için eleştirmektedirler. Aynı zamanda risk cetveli oluşturulurken az sayıda, beyaz bireyin olduğu ve sosyo ekonomik olarak orta düzeyde bir bölgede yapılan çalışmanın tüm kişilere genellenebilir olması da bir başka eleştiri noktasıdır.

Sağlıklı bireylerin verilerinden yola çıkarak bir risk skoru oluşturup kişilerin 10 yıllık koroner arter hastalığı riskinin olup olmadığının araştırılmasında kullanılan Framingham risk skoru kadın ve erkekler için ayrı ayrı skorlanmaktadır. Skorlamada yaş, cinsiyet, sigara içme durumu, sistolik kan basıncı, total kolesterol, HDL, tansiyon ilacı kullanma durumu değişkenleri dikkate alınmaktadır. Kişinin ailesinde kalp krizi geçirme geçmişi mevcut ise Framingham skoru sonucunda elde edilen risk yüzdesi iki kat artmaktadır.

Çalışmamızda kullanılan veriler, 2012-2014 yılları arasında 3 yıllık periyot içerisinde kilo vermek ve periyodik muayene için Düzce Üniversitesi Tıp Fakültesi Aile Hekimliği Polikliniğine başvuran kişiler üzerinden toplanmıştır. Toplamda 4788 kişiden ölçümler yapılmıştır. Kümeleme analizi yapmadan önce verilerde var olan eksik gözlemler için normal dağılmayan değişkenler için medyan değerleri, normal dağılan değişkenler için ortalama, kategorik değişkenler için ise mod değerleri hesaplanarak kayıp veri sorunu ortadan kaldırılmıştır. Ayrıca çalışmaya alınan kişilerin Framingham risk skoru yazılmış bir makro yardımıyla hesaplanmış ve bireyler risk skorlarına göre düşük riskli grup, orta riskli grup ve yüksek riskli grup olarak gruplandırılmıştır.

Framingham risk cetveline göre bireyler, %10'un altında riske sahip ise "düşük riskli grup", %10 ile %20 arasında "orta düzeyde riskli grup" ve %20' nin üzerinde "yüksek riskli grup" olarak sınıflandırılmaktadır. Bu durumda 3 farklı grup ortaya çıkmaktadır. Bu bilgiden hareketle gerçekleştirilen bu tez çalışmasının uygulama bölümünde kullanılan kümeleme yöntemlerine göre 3 kümenin elde edilmesi hedeflenmiştir. Uygulamada iki amaç ortaya konmuştur. Bunlardan birincisi, Framingham risk skoru hesaplanırken kullanılan değişkenler yardımıyla kümeleme analizi yapıldığında kişilerin hangi gruba düştüğünün belirlenmesi, ikincisi ise literatürde önerilen ve klinik pratikte önemsenen ilave risk faktörlerini de dikkate alarak yeniden kümeleme analizi yapıldığında bireylerin hangi risk grubuna düştüğünün belirlenmesidir. Bu iki amacı gerçekleştirmek için farklı kümeleme algoritmaları kullanılmış ve ortaya çıkan kümelerin Framingham risk skoruna göre belirlenen gruplarla uyumu incelenmiştir. Ayrıca Framingham risk skoruna göre risk grupları oluşturulurken kadın ve

erkekler için ayrı ayrı ve aile öyküsü olan ve olmayanlar için ayrı ayrı hesaplamalar yapılmaktadır. Bu çalışmada ise kümeleme algoritmalarında cinsiyet ve aile öyküsü de bir risk faktörü olarak modelde yer almıştır.

Uygulamada öne sürülen birinci ve ikinci amacı gerçekleştirmek için kümeleme analizlerinde kullanılan risk faktörleri Tablo 3.1.1' de topluca verilmiştir.

Tablo 3.1.1. Kümeleme analizlerinde kullanılan risk faktörleri

Risk faktörü numarası	**Birinci amacı gerçekleştirmede kullanılan risk faktörleri**	**İkinci amacı gerçekleştirmede kullanılan risk faktörleri**
1	Yaş (Sürekli değişken)	Yaş (Sürekli değişken)
2	Cinsiyet (Kadın-Erkek)	Cinsiyet (Kadın-Erkek)
3	Sigara İçme Durumu (İçmiyor-İçiyor)	Sigara İçme Durumu (İçmiyor-İçiyor)
4	Aile Öyküsü (Yok-Var)	Aile Öyküsü (Yok-Var)
5	Sistolik Kan Basıncı (Sürekli değişken)	Sistolik Kan Basıncı (Sürekli değişken)
6	Tansiyon ilacı kullanma durumu (evet-hayır)	Tansiyon ilacı kullanma durumu (evet-hayır)
7	HDL (Sürekli)	HDL (Sürekli)
8	Total Kolesterol (Sürekli)	Total Kolesterol (Sürekli)
9		Diyastolik Kan Basıncı (Sürekli değişken)
10		Kalça Çevresi (Sürekli)
11		Vücut Kütle İndeksi (Sürekli)
12		İç Yağlanma Değeri (Sürekli
13		Trigliserid (Sürekli)
14		Ürik Asit (Sürekli)
15		Bel Çevresi (Sürekli)
16		Açlık Kan Şekeri (Sürekli)
17		Homa (IR) (Sürekli)
18		Viseral Şişmanlık İndeksi (Sürekli)
19		Vücut ŞİŞmanlık İndeksi (Sürekli)
20		ATPIII (Negatif, Pozitif)

3.2. Uygulamada Kullanılan Kümeleme Algoritmaları ve Paket Programlar

Değerlendirmelerin ilk aşamasında veri setinde yer alan sayısal değişkenlere ait tanımlayıcı değerler ortalama ve standart sapma olarak, kategorik yapıdaki değişkenlere ait tanımlayıcı değerler ise sayı ve yüzde olarak hesaplandı ve tablolar halinde verildi. Framingham risk skoru hesaplanırken, yaş, cinsiyet, sigara kullanma durumu, sistolik kan basıncı, total kolesterol, HDL ve tansiyon ilacı kullanma durumuna ait veriler kullanıldı. Uygulamadaki amacı gerçekleştirmek için bu değişkenler dikkate alınarak 6 farklı kümeleme algoritması

yardımıyla Framingham risk skoruna göre 3 farklı risk grubu elde edildiği için, 3 farklı küme elde edildi. Ardından bu değişkenlere aile öyküsü de eklenerek yeniden kümeler oluşturuldu. Kümelemede, *K*-ortalama kümeleme yöntemi, başlangıç küme seçiminde daha iyi sonuçlar verdiği, daha kısa sürede sonuca ulaşılabildiği ve modelin küme içi hata kareler toplamını düşürdüğü için tercih edildi. Uzaklık ölçüsü olarak Öklid uzaklığı kullanıldı ve küme sayısı başlangıçta 3 olarak belirlendi. Öklid uzaklığının kullanılmasının nedeni aşırı değerlerden kaynaklanabilecek olan olumsuzluklardan etkilenmemesidir. Çekirdek değeri ise farklı değerlerin uygulanması sonucunda küme içi hata kareler toplamını en düşük yapan değer olan 35 seçildi. Verilerdeki ölçüm farklılıklarının etkilerini ortadan kaldırmak için gözlem değerlerinden aritmetik ortalamaları çıkartılıp bu farka ait standart sapmaya bölünerek standartlaştırıldıktan sonra *K*-ortalama kümeleme yöntemi uygulandı. Ardından veri setine EM kümeleme, *En uzak ilk* (Farthest First) kümeleme, yoğunluk (density) kümeleme, *K*-medoid ve cascade *K*-ortalama kümeleme yöntemleri uygulandı. K-ortalama yöntemi sadece sayısal veriler için uygulanabilen bir yöntem olmasına rağmen çok değişkenli olarak uzaklık ölçülerinin normal dağılması sebebiyle yine bu algoritmadan yararlanılabilmektedir. Ayrıca veri setinde oldukça az sayıda kategorik veriler mevcuttur. Uygulamanın ikinci amacında, risk gruplarının daha iyi tahmin edilmesini sağlamak hedeflendiği için modele yeni değişkenler eklendi. Eklenen değişkenler Tablo 3.1.1' de topluca verilmiştir. Her iki amaç sonrasında oluşturulan kümeler arasında anlamlı düzeyde ayrım yapan değişkenleri belirlemek için, sayısal değişkenler için tek yönlü ANOVA kullanıldı. ANOVA sonrasında anlamlı düzeyde farklı kümeleri belirlemek için Tukey çoklu karşılaştırma testinden yararlanıldı. Benzer amaçla kategorik yapıdaki cinsiyet, tansiyon ilacı kullanma, ailede kalp rahatsızlığı olma ve sigara içme durumları ile oluşturulan kümeler arasındaki ilişkiler Pearson Ki-Kare testi ile incelendi. Son olarak, Framingham risk skoruna göre belirlenen risk grupları ile kümeleme algoritmalarından elde edilen kümeler arasındaki uyum Kappa istatistiği ile değerlendirildi. İstatistiksel anlamlılık düzeyi 0.05 olarak alındı ve analizlerde WEKA (ver. 3.4.11), Rapid Miner (ver. 6.4) ve SPSS (ver. 18) paket programlarından yararlanıldı.

4. BULGULAR

Çalışmaya katılan kişilerin %83.5' ü (n=3998) kadın, %31.3'ü (n=1501) sigara tüketmekte, %12.1'i (n=579 kişi) tansiyon ilacı kullanmakta ve %42.6'sının ise (n=2039 kişi) ATPIII değeri pozitifti. Yaş, cinsiyet, sigara içme durumu, sistolik kan basıncı, total kolesterol, HDL, tansiyon ilacı kullanma durumlarının değerlendirilmesi sonucunda elde edilen Framingham risk skoru sonucunda kişilerin %4.2' sinde (n=203) yüksek risk tespit edildi. Aile öyküsü de

dikkate alındığında bu oran %4.4' e (n=212) yükseldi. Birinci amacı gerçekleştirmek amacıyla kümeleme algoritmalarına dâhil edilen kategorik yapıdaki değişkenlerin kategorilerine ve Framingham risk skoruna göre oluşturulan risk gruplarına düşen birey sayı ve %' leri Tablo 4.1' de topluca verildi.

Tablo 4.1. Kategorik değişkenlerin kategorilerinin dağılımı

Kategorik risk faktörleri ve risk grupları		**Sayı**	**%**
Cinsiyet	Kadın	3998	83.5
	Erkek	790	16.5
Sigara İçme Durumu	İçmiyor	3287	68.7
	İçiyor	1501	31.3
Tansiyon İlaç Kullanım Durumu	Yok	4209	87.9
	Var	579	12.1
ATPIII	Negatif	2749	57.4
	Pozitif	2039	42.6
Framingham Risk Skoru (Aile Öyküsü Hariç)	Düşük Risk	4017	83.9
	Orta Risk	568	11.9
	Yüksek Risk	203	4.2
Framingham Risk Skoru (Aile Öyküsü Dahil)	Düşük Risk	4002	83.6
	Orta Risk	574	12
	Yüksek Risk	212	4.4

Kümeleme algoritmalarında kullanılan tüm sayısal değişkenlere ait tanımlayıcı değerler ortalama, standart sapma, minimum ve maksimum olarak Tablo 4.2'de verildi.

Tablo 4.2. Kümelemede kullanılan sayısal yapıdaki değişkenlere ait tanımlayıcı değerler

Sayısal Değişkenler	**Ortalama**	**Standart Sapma**	**Minimum**	**Maksimum**
HOMA	3.042	2.2852	0.06	30.42
Yaş(YIL)	37.343	12.1024	18	80
Sistolik Kan Basıncı (mm-Hg)	124.849	17.5585	80	250
Diyastolik Kan Basıncı (mm-Hg)	81.244	11.8682	50	140
Bel Çevresi (cm)	99.777	14.0637	55	164
Kalça Çevresi (cm)	116.239	13.7314	61	175
Vücut Kütle İndeksi (kg/m2)	33.398	7.0186	14.5	66.44
İç yağlanma (kg)	9.606	4.2799	1	39
Açlık Kan Şekeri (mg/dl)	95.221	9.5745	57	126

Trigliserit (mg/dl)	132.155	76.3988	29	1000
Total Kolesterol (mg/dl)	192.692	39.9267	52	396
HDL (mg/dl)	51.102	12.4480	21	108
Ürik Asit (mg/dl)	4.871	1.1367	2	10.4
Viseral Şişmanlık İndeksi	5.155	4.0239	0.23	63.94
Vücut Şişmanlık İndeksi	38.933	8.3707	12.43	80.14

Veri setindeki yaş, cinsiyet, sigara kullanma durumu, sistolik kan basıncı, total kolesterol, HDL ve tansiyon ilacı kullanma durumu değişkenleri dikkate alınarak *K*-ortalama, cascade *K*-ortalama, *en uzak ilk*, EM, density ve *K*-medoid kümeleme yöntemlerinin uygulanması sonucunda iterasyonlar 5 adımda sonlandırılmış yani 5 adımda uygun kümeler tespit edilmiştir.

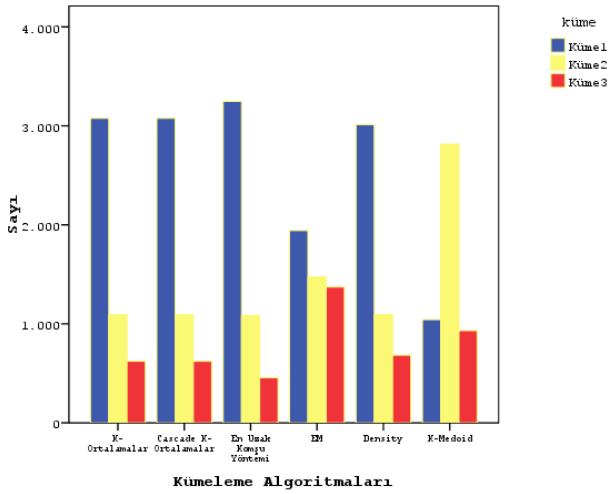
Altı farklı kümeleme algoritması sonucunda elde edilen küme özellikleri değerlendirildiğinde küme 1; kardivasküler riski düşük olan bireyleri, küme 2; orta riskli olanları ve küme 3; yüksek riskli olanları ifade ettiği görüldü. Küme sonuçları incelendiğinde ise EM kümeleme algoritmasının riskli bireyleri bulmaya karşı daha eğilimli olduğu, en uzak ilk yönteminin ise kardiyovasküler riski düşük olan bireyleri bulma eğiliminde olduğu belirlendi. Ancak genel olarak *K*-ortalama, cascade *K*-ortalama, *en uzak ilk* ve density kümeleme algoritmalarının sonucunda kümelere yerleşen bireylerin benzer oranda olduğu gözlendi. İncelenen 6 farklı algoritmaya göre elde edilen kümelere düşen bireylerin sayı ve % olarak dağılımı Tablo 4.3' de topluca verildi.

Tablo 4.3. Kümeleme algoritmalarına ait kümelere düşen bireylerin dağılımı

		Sayı	**%**
K-ortalama	Küme 1 (Düşük risk)	3075	64.2
	Küme 2 (Orta risk)	1091	22.8
	Küme 3 (Yüksek risk)	622	13
Cascade *K*-ortalama	Küme 1 (Düşük risk)	3075	64.2
	Küme 2 (Orta risk)	1091	22.8
	Küme 3 (Yüksek risk)	622	13
En uzak ilk	Küme 1 (Düşük risk)	3246	67.8
	Küme 2 (Orta risk)	1086	22.7
	Küme 3 (Yüksek risk)	456	9.5
EM	Küme 1 (Düşük risk)	1942	40.6
	Küme 2 (Orta risk)	1474	30.8

	Küme 3 (Yüksek risk)	1372	28.7
Make Density Based (Density)	Küme 1 (Düşük risk)	3013	62.9
	Küme 2 (Orta risk)	1092	22.8
	Küme 3 (Yüksek risk)	683	14.3
K-Medoid	Küme 1 (Düşük risk)	1042	21.8
	Küme 2 (Orta risk)	2815	58.8
	Küme 3 (Yüksek risk)	931	19.4

Kümelere düşen bireylerin algoritmalara göre dağılımı Şekil 4.1' de gösterildi.



Şekil 4.1. Kümelere düşen bireylerin dağılımı

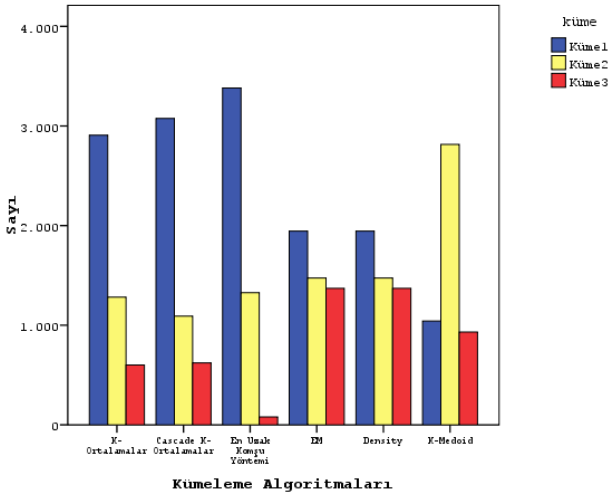
Kümeleme işlemlerine yaş, cinsiyet, sigara kullanma durumu, sistolik kan basıncı, total kolesterol, HDL, tansiyon ilacı kullanma durumuna ilaveten aile öyküsü değişkeni de ilave edilerek *K*-ortalama, cascade *K*-ortalama, *en uzak ilk*, EM, density ve *K*-medoid algoritmalarının uygulanması sonucunda elde edilen kümeleme sonuçları aşağıdaki gibi elde edildi.

Aile öyküsünün olup olmama durumu da dikkate alındığında, density haricinde kalan diğer kümeleme algoritmalarının sonuçları aile öyküsü alınmadan elde edilen kümeleme sonuçları ile benzer çıktığı ancak aile öyküsü modele alındığında density kümeleme algoritmasında kardiyovasküler riski düşük olan bireylerin birçoğunun orta riskli veya çok riskli gruplara dâhil edildiği görüldü. Bu koşulda elde edilen kümelere düşen bireylerin dağılımı Tablo 4.4' de verildi.

Tablo 4.4. Aile öyküsü de modele alındığında Kümeleme algoritmalarından elde edilen kümelere düşen bireylerin dağılımı

		Sayı	%
K-ortalama	Küme 1 (Düşük risk)	2907	60.7
	Küme 2 (Orta risk)	1281	26.8
	Küme 3 (Yüksek risk)	600	12.5
Cascade *K*-ortalama	Küme 1 (Düşük risk)	3075	64.2
	Küme 2 (Orta risk)	1091	22.8
	Küme 3 (Yüksek risk)	622	13
En uzak ilk	Küme 1 (Düşük risk)	3381	70.6
	Küme 2 (Orta risk)	1327	27.7
	Küme 3 (Yüksek risk)	80	1.7
EM	Küme 1 (Düşük risk)	1944	40.6
	Küme 2 (Orta risk)	1474	30.8
	Küme 3 (Yüksek risk)	1370	28.6
Density	Küme 1 (Düşük risk)	1944	40.6
	Küme 2 (Orta risk)	1474	30.8
	Küme 3 (Yüksek risk)	1370	28.6
K-Medoid	Küme 1 (Düşük risk)	1042	21.8
	Küme 2 (Orta risk)	2815	58.8
	Küme 3 (Yüksek risk)	931	19.4

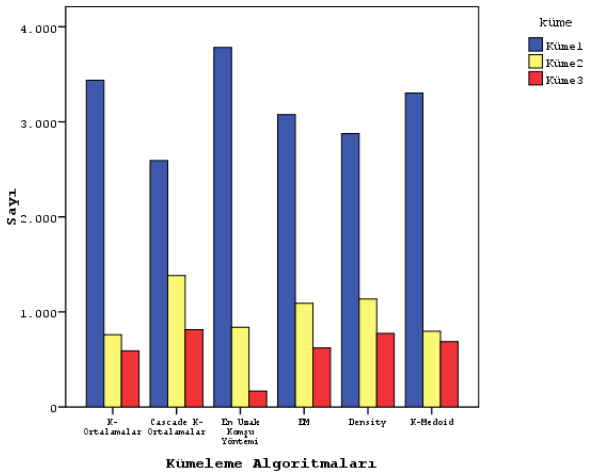
Aile öyküsünün de modele dâhil edilmesi sonucunda elde edilen kümelere düşen bireylerin algoritmalara göre dağılımı Şekil 4.2' de de gösterildi.



Şekil 4.2. Kümelere düşen bireylerin dağılımı

Materyal ve Metot bölümünde Tablo 3.1.1' de yer alan tüm risk faktörleri dikkate alınarak *K*-ortalama, cascade *K*-ortalama, *en uzak ilk*, EM, density ve *K*-medoid algoritmalarının uygulanması sonucunda elde edilen kümeleme sonuçları aşağıdaki gibi bulundu.

Cascade *K*-ortalama yöntemine göre yüksek riskli kişilerin oranı (%17), diğer kümeleme yöntemlerinin yüksek riskli olarak sınıflandırdığı bireylerden daha fazla bulundu. *En uzak ilk* kümeleme algoritmasının sonuçları değerlendirildiğinde ise kardiyovasküler riski düşük olan bireylerin oranının, diğer kümeleme algoritmalarının sonuçlarına göre daha fazla olduğu görüldü. Kümelere düşen birey sayılarının algoritmalara göre dağılımı Şekil 4.3' ve Tablo 4.5' de de topluca verilmiştir.



Şekil 4.3. Kümelere düşen bireylerin dağılımı

Tablo 4.5. Kümelere düşen bireylerin dağılımı

		Sayı	**%**
K-ortalama	Küme 1 (Düşük risk)	3437	71.8
	Küme 2 (Orta risk)	760	15.9
	Küme 3 (Yüksek risk)	591	12.3
Cascade K-ortalama	Küme 1 (Düşük risk)	2592	54.1
	Küme 2 (Orta risk)	1383	28.9
	Küme 3 (Yüksek risk)	813	17
En uzak ilk	Küme 1 (Düşük risk)	3782	79
	Küme 2 (Orta risk)	839	17.5
	Küme 3 (Yüksek risk)	167	3.5
EM	Küme 1 (Düşük risk)	3075	64.2
	Küme 2 (Orta risk)	1091	22.8
	Küme 3 (Yüksek risk)	622	13

Density	Küme 1 (Düşük risk)	2877	60.1
	Küme 2 (Orta risk)	1136	23.7
	Küme 3 (Yüksek risk)	775	16.2
K-Medoid	Küme 1 (Düşük risk)	3303	69
	Küme 2 (Orta risk)	797	16.6
	Küme 3 (Yüksek risk)	688	14.4

Framingham risk skoru hesaplanırken kullanılan sayısal yapıdaki değişkenler dikkate alınarak (aile öyküsü dahil) farklı kümeleme algoritmaları sonucunda elde edilen kardiyovasküler riski düşük, orta, yüksek grupları arasında, söz konusu sayısal değişkenlerin ortalamalarının karşılaştırılması sonucunda elde edilen tanımlayıcı istatistikler ve *p* değerleri Tablo 4.6' da verildi. Uygulamada kullanılan 6 farklı kümeleme algoritması yardımıyla elde edilen kümeler arasında yaş, total kolesterol ve HDL ortalamaları bakımından anlamlı fark bulundu (her bir karşılaştırma için $p<0,001$). Ancak *K*-ortalama kümeleme yöntemi sonucunda elde edilen kümeler arasında sistolik kan basıncı ortalaması bakımından anlamlı fark yok iken ($p=0,441$), cascade *K*-ortalama, *en uzak ilk*, EM, Density ve *K*-medoid algoritmaları sonucunda anlamlı farklılık bulundu (her biri için $p<0,001$). Anlamlı düzeyde farklı olan kümeler, Tablo 4.6' da ortalamaların yanına yerleştirilen tamamen farklı harflerle gösterildi.

Tablo 4.6. Aile öyküsü değişkeni modele alınarak elde edilen kümelerde sayısal değişkenlerin tanımlayıcı değerleri ve kümelerin karşılaştırma sonuçları

Kümeleme Algoritmaları	**Kümeler**	**Yaş**		**Sistolik Kan Basıncı**		**Total Kolesterol**		**HDL**	
		Ort∓SD*	***p***	**Ort∓SD**	***p***	**Ort∓SD**	***p***	**Ort∓SD**	***p***
K-Ortalama	Düşük Risk	$37,06^{a}$∓12,556	<0,001	124,80∓18,225	0,441	$190,48^{b}$∓39,433	<0,001	$52,94^{a}$∓12,093	<0,001
	Orta Risk	$39,27^{b}$∓11,457		124,56∓17,135		$197,35^{a}$∓40,860		$51,10^{b}$∓12,633	
	Yüksek Risk	$34,59^{c}$∓10,449		125,66∓14,965		$193,43^{ab}$∓39,462		$42,15^{c}$∓9,538	
Cascade K-Ortalama	Düşük Risk	$36,61^{a}$∓2,474	<0,001	$124,65^{a}$∓18,028	<0,001	$190,06^{a}$∓39,181	<0,001	$52,79^{a}$∓11,963	<0,001
	Orta Risk	$37,58^{b}$∓10,730		$123,34^{a}$∓16,482		$195,51^{b}$∓41,245		$51,38^{b}$∓12,903	
	Yüksek Risk	$40,54^{c}$∓11,988		$128,42^{b}$∓16,550		$200,72^{c}$∓39,879		$42,23^{c}$∓10,034	
En Uzak İlk	Düşük Risk	$35,36^{a}$∓10,849	<0,001	$23,88^{b}$∓16,676	<0,001	$190,56^{b}$∓39,551	<0,001	$50,76^{b}$∓12,266	<0,001
	Orta Risk	$42,33^{b}$∓13,696		$127,23^{a}$∓19,314		$197,77^{a}$∓40,452		$52,01^{a}$∓12,883	
	Yüksek Risk	$38,43^{c}$∓10,158		$126,26^{ab}$∓19,568		$198,12^{ab}$∓39,316		$50,08^{ab}$∓12,065	
EM	Düşük Risk	$27,49^{a}$∓5,904	<0,001	$114,61^{a}$∓11,170	<0,001	$170,45^{a}$∓29,665	<0,001	$53,65^{a}$∓12,249	<0,001
	Orta Risk	$39,00^{b}$∓8,865		$127,13^{b}$∓10,671		$193,87^{b}$∓30,924		$42,44^{b}$∓7,043	
	Yüksek	$49,54^{c}$∓9,401		$136,90^{c}$∓21,878		$222,97^{c}$∓40,918		$56,79^{c}$∓12,374	

	Risk								
Density	Düşük Risk	27,49[a]∓5,904	<0,001	114,61[a]∓11,170	<0,001	170,45[a]∓29,665	<0,001	53,65[a]∓12,249	<0,001
	Orta Risk	39,00[b]∓8,865		127,13[b]∓10,671		193,87[b]∓30,924		42,44[b]∓7,043	
	Yüksek Risk	49,54[c]∓9,401		136,90[c]∓21,878		222,97[c]∓40,918		56,79[c]∓12,374	
K-Medoid	Düşük Risk	29,57[a]∓8,925	<0,001	115,50[a]∓12,970	<0,001	143,70[a]∓17,421	<0,001	49,78[a]∓11,666	<0,001
	Orta Risk	37,73[b]∓11,585		126,97[b]∓17,668		191,79[b]∓18,442		49,24[a]∓11,195	
	Yüksek Risk	44,85[c]∓11,581		128,87[c]∓18,065		250,21[c]∓28,868		58,19[b]∓14,256	

*Ortalama∓Standart Sapma

Framingham risk skoru hesaplanırken kullanılan kategorik yapıdaki değişkenlerin kategorilerinin, oluşturulan kümelerdeki dağılımı ve kümelerin bu değişkenler bakımından karşılaştırılma sonuçları Tablo 4.7' de özetlendi. Tablo 4.7. incelendiğinde *K*-ortalama kümeleme algoritması sonucunda elde edilen kümeler arasında cinsiyet, sigara kullanma durumu ve aile öyküsü dağılımları bakımından anlamlı farklılıklar tespit edildi (p<0.05). Farklılıklar detaylı olarak incelendiğinde, *K*-ortalama yöntemin sonucunda elde edilen yüksek riskli gruplarda erkeklerin görülme oranı kadınlara göre anlamlı derecede yüksek ve sigara içenlerin anlamlı derecede yüksek riskli gruba ait olduğu görüldü. Düşük riskli olarak belirlenen kümede (küme 1) aile öyküsü olmayanların oranı anlamlı derecede yüksek bulundu (her biri için p<0,05). Cascade *K*-ortalama ve *K*-medoid kümeleme algoritmalarının uygulanması sonucunda elde edilen kümeler arasında aile öyküsü dağılımı bakımından anlamlı farklılık gözlenmedi. Ancak bu kümeler arasında cinsiyet dağılımı ve sigara içme durumu bakımından anlamlı farklılık belirlendi (her biri için p<0,001). *En uzak ilk*, EM ve density kümeleme algoritmaları sonucunda kadınların, sigara içmeyenlerin ve aile öyküsü olmayanların, kardiyovasküler riski düşük olan grupta yer alma olasılıkları anlamlı derecede yüksek çıktı (her bir karşılaştırma için p<0,05).

Tablo 4.7. Aile öyküsü modele alınarak yapılan kümeleme işlemleri sonrasında modeldeki kategorik risk faktörlerinin kategorilerinin kümelere dağılımı ve kümelerin bu değişkenler bakımından karşılaştırılma sonuçları

K-ortalama			**KÜMELER**			**p**
			Düşük	Orta	Yüksek	
Cinsiyet	Kadın	n	2907	1091	0	<0,001
		Satıra göre %	72.7	27.3	0	
		Sütuna göre %	100.0	85.2	0	
	Erkek	n	0	190	600	
		Satıra göre %	0	24.1	75.9	
		Sütuna göre %	0	14.8	100.0	
Sigara Kullanma	İçmiyor	n	2907	0	380	<0,001
		Satıra göre %	88.4	0	11.6	

Durumu		Sütuna göre %	100.0	0	63.3	
	İçiyor	n	0	1281	220	
		Satıra göre %	0	85.3	14.7	
		Sütuna göre %	0	100	36.7	
Aile Öyküsü	Yok	n	2858	1253	597	0,029
		Satıra göre %	60.7	26.6	12.7	
		Sütuna göre %	98.3	97.8	99.5	
	Var	n	49	28	3	
		Satıra göre %	61.3	35.0	3.8	
		Sütuna göre %	1.7	2.2	0.5	
Cascade K-ortalama						
Cinsiyet	Kadın	n	2907	1091	0	<0,001
		Satıra göre %	72.7	27.3	0	
		Sütuna göre %	94.5	100.0	0	
	Erkek	n	168	0	622	
		Satıra göre %	21.3	0	78.7	
		Sütuna göre %	5.5	0	100.0	
Sigara Kullanma Durumu	İçmiyor	n	3075	0	212	<0,001
		Satıra göre %	93.6	0	6.4	
		Sütuna göre %	100.0	0	34.1	
	İçiyor	n	0	1091	410	
		Satıra göre %	0	72.7	27.3	
		Sütuna göre %	0	100	65.9	
Aile Öyküsü	Yok	n	3026	1069	613	
		Satıra göre %	64.3	22.7	13	
		Sütuna göre %	98.4	98.0	98.6	
	Var	n	49	22	9	
		Satıra göre %	61.3	27.5	11.3	
		Sütuna göre %	1.6	2	1.4	
En uzak ilk						
Cinsiyet	Kadın	n	2642	1283	73	<0,001
		Satıra göre %	66.1	32.1	1.8	
		Sütuna göre %	78.1	96.7	91.3	
	Erkek	n	739	44	7	
		Satıra göre %	93.5	5.6	.9	
		Sütuna göre %	21.9	3.3	8.8	
Sigara Kullanma Durumu	İçmiyor	n	3022	216	49	
		Satıra göre %	91.9	6.6	1.5	
		Sütuna göre %	89.4	16.3	61.3	
	İçiyor	n	359	1111	31	<0,001
		Satıra göre %	23.9	74.0	2.1	
		Sütuna göre %	10.6	83.7	38.8	
Aile Öyküsü	Yok	n	3379	1327	2	
		Satıra göre %	71.8	28.2	0	
		Sütuna göre %	99.9	100.0	2.5	<0,001
	Var	n	2	0	78	
		Satıra göre %	2.5	0	97.5	
		Sütuna göre %	0.1	0	97.5	
EM						
Cinsiyet	Kadın	n	1850	882	1266	
		Satıra göre %	46.3	22.1	31.7	<0,001
		Sütuna göre %	95.2	59.8	92.4	
	Erkek	n	94	592	104	

		Satıra göre %	11.9	74.9	13.2	
		Sütuna göre %	4.8	40.2	7.6	
Sigara Kullanma Durumu	İçmiyor	n	1487	814	986	<0,001
		Satıra göre %	45.2	24.8	30.0	
		Sütuna göre %	76.5	55.2	72.0	
	İçiyor	n	457	660	384	
		Satıra göre %	30.4	44.0	25.6	
		Sütuna göre %	23.5	44.8	28.0	
Aile Öyküsü	Yok	n	1924	1446	1338	0,011
		Satıra göre %	40.9	30.7	28.4	
		Sütuna göre %	99.0	98.1	97.7	
	Var	n	20	28	32	
		Satıra göre %	25.0	35.0	40.0	
		Sütuna göre %	1.0	1.9	2.3	
Density						
Cinsiyet	Kadın	n	1850	882	1266	<0,001
		Satıra göre %	46.3	22.1	31.7	
		Sütuna göre %	95.2	59.8	92.4	
	Erkek	n	94	592	104	
		Satıra göre %	11.9	74.9	13.2	
		Sütuna göre %	4.8	40.2	7.6	
Sigara Kullanma Durumu	İçmiyor	n	1487	814	986	<0,001
		Satıra göre %	45.2	24.8	30.0	
		Sütuna göre %	76.5	55.2	72.0	
	İçiyor	n	457	660	384	
		Satıra göre %	30.4	44.0	25.6	
		Sütuna göre %	23.5	44.8	28.0	
Aile Öyküsü	Yok	n	1924	1446	1338	0,011
		Satıra göre %	40.9	30.7	28.4	
		Sütuna göre %	99.0	98.1	97.7	
	Var	n	20	28	32	
		Satıra göre %	25.0	35.0	40.0	
		Sütuna göre %	1.0	1.9	2.3	
K-Medoid						
Cinsiyet	Kadın	n	910	2303	785	<0,001
		Satıra göre %	22.8	57.6	19.6	
		Sütuna göre %	87.3	81.8	84.3	
	Erkek	n	132	512	146	
		Satıra göre %	16.7	64.8	18.5	
		Sütuna göre %	12.7	18.2	15.7	
Sigara Kullanma Durumu	İçmiyor	n	767	1896	624	<0,001
		Satıra göre %	23.3	57.7	19.0	
		Sütuna göre %	73.6	67.4	67.0	
	İçiyor	n	275	919	307	
		Satıra göre %	18.3	61.2	20.5	
		Sütuna göre %	26.4	32.6	33.0	
Aile Öyküsü	Yok	n	1026	2768	914	0,881
		Satıra göre %	21.8	58.8	19.4	
		Sütuna göre %	98.5	98.3	98.2	
	Var	n	16	47	17	
		Satıra göre %	20.0	58.8	21.3	
		Sütuna göre %	1.5	1.7	1.8	

Yukarıda tanımlanan işlem adımları kümeleme işlemlerine aile öyküsü dahil edilmeden tekrar edildi ve aşağıdaki sonuçlar elde edildi.

Framingham risk skoru hesaplanırken kullanılan sayısal değişkenlerin ortalamaları bakımından, farklı kümeleme algoritmaları sonucunda elde edilen kardiyovasküler riski düşük, orta, yüksek olan kümeler arasında anlamlı farklılık olup olmadığı karşılaştırıldı ve elde edilen sonuçlar Tablo 4.8'de özetlendi. Tablo 4.8 incelendiğinde, oluşturulan kümeleme arasında yaş, sistolik kan basıncı total kolesterol ve HDL ortalamaları bakımından anlamlı fark olduğu görüldü (her bir karşılaştırma için p<0,001). Farklı olan kümeleri göstermek amacıyla ortalamalar üzerine harfler konuldu. Harflendirmede tamamen farklı olan harfler, o değişken bakımından kümeler arasında anlamlı farklılık olduğunu ifade ederken, aynı veya ortak harfe sahip kümeler arasında anlamlı bir farklılık olmadığı söylenebilir.

Framingham risk skoru hesaplanırken kullanılan kategorik değişkenlerin kategorilerinin elde edilen kümelere dağılımı ve kümelerin bu değişkenler bakımından karşılaştırılma sonuçları Tablo 4.9' da özetlendi. Tablo 4.9 incelendiğinde tüm kümeleme algoritmaları sonucunda yüksek riskli olarak belirlenen kümede, erkeklerin kadınlara göre ve sigara kullananların kullanmayanlara göre anlamlı düzeyde daha fazla oranda olduğu gözlendi (her bir karşılaştırma için p<0,001).

Tablo 4.8. Aile öyküsü değişkeni modele alınmadan elde edilen kümelerde sayısal değişkenlerin tanımlayıcı değerleri ve kümelerin karşılaştırma sonuçları

Kümeleme Algoritmaları	Kümeler	Yaş		Sistolik Kan Basıncı		Total Kolesterol		HDL	
		Ort∓SD*	*p*	Ort∓SD	*p*	Ort∓SD	*p*	Ort∓SD	*p*
K-Ortalama	Düşük Risk	36,61[a]∓12,474	<0,001	124,65[a]∓18.028	<0,001	190,06[a]∓39,181	<0,001	52,79[a]∓11,963	<0,001
	Orta Risk	37,582[b]∓10,730		123,34[a]∓16,482		195,51[b]∓41,245		51,38[b]∓12,903	
	Yüksek Risk	40,542[c]∓11,988		128,42[b]∓16,550		200,72[c]∓39,879		42,23[c]∓10,035	
Cascade K-Ortalama	Düşük Risk	36,61[a]∓12,474	<0,001	124,65[a]∓18.028	<0,001	190,06[a]∓39,181	<0,001	52,79[a]∓11,963	<0,001
	Orta Risk	37,582[b]∓10,730		123,34[a]∓16,482		195,51[b]∓41,245		51,38[b]∓12,903	
	Yüksek Risk	40,542[c]∓11,988		128,42[b]∓16,550		200,72[c]∓39,879		42,23[c]∓10,035	
En Uzak İlk	Düşük Risk	36,66[a]∓12,284	<0,001	124,83[a]∓17,851	<0,001	190,64[b]∓39,193	<0,001	51,99[a]∓12,196	<0,001
	Orta Risk	37,41[a]∓10,478		123,23[b]∓16,393		195,55[a]∓41,319		51,41[a]∓12,912	
	Yüksek Risk	41,96[b]∓13,360		128,79[c]∓17,557		200,45[a]∓39,926		44,044[b]∓10,770	
EM	Düşük Risk	27,50[a]∓5,911	<0,001	114,59[a]∓11,188	<0,001	170,43[a]∓29,695	<0,001	53,65[a]∓12,256	<0,001
	Orta Risk	38,95[b]∓8,887		127,08[b]∓10,653		193,84[b]∓30,860		42,45[b]∓7,054	
	Yüksek	49,54[c]∓9,395		136,95[c]∓21,826		222,96[c]∓40,919		56,77[c]∓12,376	

	Risk								
Density	Düşük Risk	36,83[a]∓12,546	<0,001	124,61[a]∓18,095	<0,001	190,10[b]∓39,308	<0,001	53,01[a]∓11,986	<0,001
	Orta Risk	37,57[a]∓10,726		123,34[a]∓16,474		195,51[a]∓41,227		51,42[b]∓12,950	
	Yüksek Risk	39,21[b]∓12,003		128,27[b]∓16,370		199,58[a]∓39,450		42,17[c]∓9,418	
K-Medoid	Düşük Risk	29,57[a]∓8,925	<0,001	115,50[a]∓12,970	<0,001	143,70[a]∓17,421	<0,001	49,78[a]∓11,666	<0,001
	Orta Risk	37,73[b]∓11,585		126,97[b]∓17,668		191,79[b]∓18,442		49,24[a]∓11,195	
	Yüksek Risk	44,85[c]∓1,581		128,87[c]∓18,065		250,21[c]∓28,868		58,19[b]∓14,256	

*Ortalama∓Standart Sapma

Tablo 4.9. Aile öyküsü modele alınmadan yapılan kümeleme işlemleri sonrasında modeldeki kategorik risk faktörlerinin kategorilerinin kümelere dağılımı ve kümelerin bu değişkenler bakımından karşılaştırılma sonuçları

K-ortalama			**KÜMELER**			***p***
			Düşük	**Orta**	**Yüksek**	
Cinsiyet	Kadın	n	2907	1091	0	<0,001
		Satıra göre	72.7	27.3	0	
		Sütuna göre	94.5	100	0	
	Erkek	n	168	0	622	
		Satıra göre	21.3	0	78.7	
		Sütuna göre	5.5	0	100	
Sigara Kullanma Durumu	İçmiyor	n	3075	0	212	<0,001
		Satıra göre	93.6	.0	6.4	
		Sütuna göre	100.0	.0	34.1	
	İçiyor	n	0	1091	410	
		Satıra göre	.0	72.7	27.3	
		Sütuna göre	.0	100.0	65.9	
Cascade K-ortalama						
Cinsiyet	Kadın	n	2907	1091	0	<0,001
		Satıra göre	72.7	27.3	.0	
		Sütuna göre	94.5	100.0	.0	
	Erkek	n	168	0	622	
		Satıra göre	21.3	.0	78.7	
		Sütuna göre	5.5	.0	100.0	
Sigara Kullanma Durumu	İçmiyor	n	3075	0	212	<0,001
		Satıra göre	93.6	.0	6.4	
		Sütuna göre	100.0	.0	34.1	
	İçiyor	n	0	1091	410	
		Satıra göre	.0	72.7	27.3	
		Sütuna göre	.0	100.0	65.9	
En uzak ilk						
Cinsiyet	Kadın	n	2907	1086	5	<0,001
		Satıra göre	72.7	27.2	.1	
		Sütuna göre	89.6	100.0	1.1	
	Erkek	n	339	0	451	
		Satıra göre	42.9	.0	57.1	
		Sütuna göre	10.4	.0	98.9	
Sigara Kullanma Durumu	İçmiyor	n	3246	0	41	<0,001
		Satıra göre	98.8	.0	1.2	

		Sütuna göre	100.0	.0	9.0	
	İçiyor	n	0	1086	415	
		Satıra göre	.0	72.4	27.6	
		Sütuna göre	.0	100.0	91.0	
EM						
Cinsiyet	Kadın	n	1851	879	1268	<0,001
		Satıra göre	46.3	22.0	31.7	
		Sütuna göre	95.3	59.6	92.4	
	Erkek	n	91	595	104	
		Satıra göre	11.5	75.3	13.2	
		Sütuna göre	4.7	40.4	7.6	
Sigara Kullanma Durumu	İçmiyor	n	1485	814	988	<0,001
		Satıra göre	45.2	24.8	30.1	
		Sütuna göre	76.5	55.2	72.0	
	İçiyor	n	457	660	384	
		Satıra göre	30.4	44.0	25.6	
		Sütuna göre	23.5	44.8	28.0	
Density						
Cinsiyet	Kadın	n	2907	1091	0	<0,001
		Satıra göre	72.7	27.3	.0	
		Sütuna göre	96.5	99.9	.0	
	Erkek	n	106	1	683	
		Satıra göre	13.4	.1	86.5	
		Sütuna göre	3.5	.1	100.0	
Sigara Kullanma Durumu	İçmiyor	n	3013	0	274	<0,001
		Satıra göre	91.7	.0	8.3	
		Sütuna göre	100.0	.0	40.1	
	İçiyor	n	0	1092	409	
		Satıra göre	.0	72.8	27.2	
		Sütuna göre	.0	100.0	59.9	
K-Medoid						
Cinsiyet	Kadın	n	910	2303	785	<0,001
		Satıra göre	22.8	57.6	19.6	
		Sütuna göre	87.3	81.8	84.3	
	Erkek	n	132	512	146	
		Satıra göre	16.7	64.8	18.5	
		Sütuna göre	12.7	18.2	15.7	
Sigara Kullanma Durumu	İçmiyor	n	767	1896	624	<0,001
		Satıra göre	23.3	57.7	19.0	
		Sütuna göre	73.6	67.4	67.0	
	İçiyor	n	275	919	307	
		Satıra göre	18.3	61.2	20.5	
		Sütuna göre	26.4	32.6	33.0	

Uygulamadaki ikinci amacı değerlendirmek için literatürde kardiyovasküler hastalıkların risk faktörü olarak tanımlanan ve klinik pratikte önemsenen ilave risk faktörleri de dikkate alındı ve altı farklı kümeleme algoritması yardımıyla risk grupları yeniden oluşturuldu. Bu amaç için dikkate alınan risk faktörleri Tablo 3.1.1'in üçüncü sütununda topluca verildi. Kümeleme işlemleri sonrasında ortaya çıkan kümelerde dikkate alınan sayısal yapıdaki risk faktörlerinin

tanımlayıcı değerleri ve kümelerin bu risk faktörleri bakımından karşılaştırma sonuçları Tablo 4.10' da topluca verildi.

Tablo 4.10 incelendiğinde, HOMA değişkeni hariç diğer değişkenlerin ortalamaları bakımından kümeleme sonucunda belirlenen risk grupları arasında anlamlı farklılıklar olduğu belirlendi (her biri için $p<0.05$). Hangi kümelerin ortalamalarının anlamlı derecede diğer kümelerden yüksek veya düşük olduğu ise ortalamalar üzerindeki harflerle gösterildi. Uygulanan kümeleme algoritmaları sonucunda kardiyovasküler risk gruplarında HOMA ortalamaları benzer bulundu. Ancak literarürde HOMA değişkeninin kardiyovasküler hastalıklar için bir risk faktörü olduğu bilgisi ile sıklıkla karşılaşılmaktadır[150].

Tablo 4.10. Kümeleme algoritmalarına alınan tüm sayısal değişkenlerin oluşan kümelerdeki tanımlayıcı değerleri ve kümelerin karşılaştırma sonuçları

Sayısal Değişkenler	Kümeler	*K*-Ortalama		Cascade *K*-Ortalama		En Uzak İlk		EM		Density		*K*-Medoid	
		Ort∓SD*	*p*	Ort∓SD*	*p*	Ort∓SD*	*p*	Ort∓SD*	*p*	Ort∓SD*	*p*	Ort∓SD*	*p*
Yaş	Düşük Risk	35,32^{a}∓11,54	<0,001	35,57^{a}∓10,39	<0,001	35,56^{a}∓11,57	<0,001	36,61^{a}∓12,47	<0,001	37,00^{a}∓12,55	<0,001	32,25^{a}∓11,65	<0,001
	Orta Risk	37,59^{b}∓10,87		38,1^{b}∓10,07		44,68^{b}∓11,38		37,58^{b}∓10,73		37,22^{a}∓10,74		40,53^{b}∓11,59	
	Yüksek Risk	48,74^{c}∓10,29		51,16^{c}∓9,02		40,84^{c}∓11,46		40,54^{c}∓11,98		38,78^{b}∓12,19		43,68^{c}∓11,82	
Sistolik Kan Basıncı	Düşük Risk	121,76^{a}∓16,18	<0,001	117,39^{a}∓13,48	<0,001	122,19^{b}∓16,03	<0,001	124,65^{a}∓18,03	<0,001	124,82^{a}∓18,25	<0,001	123,27^{a}∓16,98	<0,001
	Orta Risk	127,11^{b}∓15,23		128,32^{b}∓14,29		135,30^{a}∓19,61		123,34^{a}∓16,48		122,91^{b}∓16,31		130,12^{b}∓19,17	
	Yüksek Risk	139,85^{c}∓19,72		142,70^{c}∓19,31		132,34^{a}∓18,06		128,42^{b}∓16,55		127,77^{x}∓16,29		126,27^{c}∓16,99	
Total Kolesterol	Düşük Risk	188,17^{a}∓38,89	<0,001	182,29^{a}∓36,98	<0,001	188,77^{a}∓39,02	<0,001	190,06^{a}∓39,18	<0,001	190,13^{a}∓39,28	<0,001	176,57^{a}∓28,17	<0,001
	Orta Risk	198,51^{b}∓39,97		200,59^{b}∓39,35		206,10^{b}∓39,37		195,51^{b}∓41,25		193,96^{b}∓40,83		213,68^{b}∓39,25	
	Yüksek Risk	211,45^{c}∓39,41		212,39^{c}∓39,47		214,08^{c}∓41,81		200,72^{c}∓39,88		200,32^{c}∓39,96		245,74^{c}∓30,48	
HDL	Düşük Risk	53,59^{a}∓12,31	<0,001	55,18^{a}∓12,19	<0,001	52,67^{a}∓12,46	<0,001	52,79^{a}∓11,96	<0,001	53,13^{a}∓12,02	<0,001	51,26^{a}∓10,84	<0,001
	Orta Risk	42,33^{b}∓9,69		43,93^{b}∓9,71		46,67^{b}∓10,21		51,38^{b}∓12,90		51,60^{b}∓12,8		41,14^{b}∓9,20	
	Yüksek Risk	47,89^{c}∓10,38		50,26^{c}∓11,74		37,67^{c}∓8,18		42,23^{c}∓10,03		42,82^{c}∓9,80		61,83^{c}∓13,62	
HOMA	Düşük Risk	3,07∓2,35	0,182	3,06∓2,41	0,259	3,05∓2,31	0,560	3,07∓2,33	0,436	3,06∓2,31	0,506	3,02∓2,32	0,161
	Orta Risk	2,90∓2,07		2,95∓2,20		3,03∓2,19		3,01∓2,17		3,03∓2,19		2,98∓2,08	
	Yüksek Risk	3,05∓2,15		3,10∓2,01		2,85∓2,24		2,95∓2,23		2,96∓2,34		3,19∓2,344	
Diyastolik Kan Basıncı	Düşük Risk	79,49^{a}∓11,11	<0,001	76,56^{a}∓9,60	<0,001	79,67^{b}∓11,06	<0,001	81,16^{a}∓11,92	<0,001	81,19^{a}∓12,02	<0,001	80,37^{a}∓11,54	<0,001
	Orta Risk	82,58^{b}∓11,28		83,84^{b}∓10,57		87,30^{a}∓13,09		80,33^{a}∓11,84		80,01^{b}∓11,85		84,43^{b}∓12,82	
	Yüksek Risk	89,70^{c}∓12,97		91,75^{c}∓12,40		86,24^{a}∓12,01		83,21^{b}∓11,44		83,24^{c}∓11,02		81,70^{c}∓11,62	
Bel Çevresi	Düşük Risk	97,02^{a}∓13,64	<0,001	93,01^{a}∓12,62	<0,001	97,79^{a}∓13,93	<0,001	98,81^{a}∓13,99	<0,001	98,68^{a}∓13,88	<0,001	99,09^{a}∓14,47	<0,001
	Orta Risk	104,79^{b}∓13,02		106,35^{b}∓10,86		106,73^{b}∓11,79		99,04^{a}∓13,49		98,36^{a}∓13,63		105,44^{b}∓11,88	
	Yüksek Risk	109,33^{c}∓11,59		110,15^{c}∓11,39		109,65^{c}∓12,11		105,82^{b}∓13,95		105,90^{b}∓13,78		96,49^{c}∓12,53	
Kalça Çevresi	Düşük Risk	115∓16^{a}∓166	<0,001	111,23^{a}∓12,89	<0,001	114,88^{a}∓13,65	<0,001	116,88^{a}∓14,29	<0,001	117,24^{a}∓14,27	<0,001	116,40^{a}∓14,32	<0,001
	Orta Risk	114,05^{a}∓10,80		119,84^{b}∓11,43		122,66^{b}∓12,77		116,63^{a}∓12,999		115,80^{b}∓13,483		118,15^{b}∓11,98	
	Yüksek Risk	125,27^{b}∓12,97		126,04^{c}∓12,66		114,62^{a}∓13,73		112,32^{b}∓11,278		113,14^{c}∓11,383		113,19^{c}∓12,24	
Beden-Kitle İndeksi	Düşük Risk	32,55^{a}∓6,95	<0,001	30,29^{a}∓6,17	<0,001	32,52^{a}∓6,88	<0,001	33,56^{a}∓7,234	0,012	33,70^{b}∓7,234	0,001	33,22^{a}∓7,29	<0,001
	Orta Risk	33,08^{a}∓5,73		35,87^{b}∓5,57		37,18^{b}∓6,57		33,35^{ab}∓6,746		32,90^{a}∓7,010		35,32^{b}∓6,19	
	Yüksek Risk	38,71^{b}∓6,58		39,09^{c}∓6,51		34,23^{c}∓5,62		32,64^{b}∓6,330		32,97^{a}∓6,102		31,98^{c}∓6,03	
İç Yağlanma	Düşük Risk	8,50^{a}∓3,53	<0,001	7,60^{a}∓3,41	<0,001	9,02^{a}∓4,13	<0,001	8,95^{a}∓3,653	<0,001	8,79^{a}∓3,378	<0,001	9,14^{a}∓4,24	<0,001
	Orta Risk	12,49^{b}∓5,388		11,17^{b}∓3,89		11,28^{b}∓3,73		8,84^{a}∓3,125		8,65^{a}∓3,084		11,44^{b}∓4,35	
	Yüksek Risk	12,29^{b}∓3,67		13,32^{c}∓3,78		14,26^{c}∓5,01		14,16^{b}∓5,831		14,00^{b}∓5,810		9,69^{c}∓3,81	
Açlık Kan Şekeri	Düşük Risk	93,88^{a}∓8,88	<0,001	91,94^{a}∓7,70	<0,001	94,19^{a}∓9,09	<0,001	95,16^{a}∓9,501	<0,001	95,13^{a}∓9,506	<0,001	94,54^{a}∓9,31	<0,001
	Orta Risk	96,57^{b}∓10,19		97,81^{b}∓9,89		99,32^{b}∓10,30		94,50^{a}∓9,398		94,22^{b}∓9,322		97,24^{b}∓10,58	
	Yüksek Risk	101,2^{c}∓10,08		101,24^{c}∓10,12		97,91^{b}∓10,41		96,78^{b}∓10,068		97,00^{c}∓9,952		96,14^{c}∓9,18	
Trigliserid	Düşük Risk	114,6^{a}∓57,95	<0,001	100,11^{a}∓44,31	<0,001	118,32^{a}∓61,28	<0,001	121,61^{a}∓63,226	<0,001	119,01^{a}∓56,432	<0,001	102,81^{a}∓35,68	<0,001
	Orta Risk	184,6^{b}∓111,71		176,14^{b}∓94,0		164,25^{b}∓67,82		133,84^{b}∓63,307		128,56^{b}∓63,307		258,10^{b}∓92,74	
	Yüksek Risk	166,3^{c}∓72,35		159,47^{c}∓76,05		284,20^{c}∓163,46		181,30^{c}∓113,85		186,18^{c}∓121,67		126,10^{c}∓41,97	
Ürik Asit	Düşük Risk	4,65^{a}∓1,02	<0,001	4,51^{a}∓0,99	<0,001	4,76^{a}∓1,10	<0,001	4,71^{a}∓1,042	<0,001	4,65^{a}∓1,002	<0,001	4,75^{a}∓1,09	<0,001
	Orta Risk	5,63^{b}∓1,25		5,32^{b}∓1,18		5,07^{b}∓1,08		4,69^{a}∓1,012		4,67^{a}∓0,986		5,41∓1,22	
	Yüksek Risk	5,16^{c}∓1,09		5,21^{c}∓1,11		6,13^{c}∓1,30		5,95^{b}∓1,194		5,96^{b}∓1,183		4,82∓1,06	
Viseral Adopsity İndeks	Düşük Risk	4,32^{b}∓3,05	<0,001	3,51^{a}∓2,07	<0,001	4,40^{a}∓3,07	<0,001	4,69^{a}∓3,397	<0,001	4,59^{a}∓2,898	<0,001	3,88^{a}∓1,88	<0,001
	Orta Risk	7,44^{a}∓5,93		7,56^{b}∓5,26		7,07^{b}∓3,86		5,58^{b}∓4,251		5,22^{b}∓3,585		11,43^{b}∓5,66	
	Yüksek Risk	7,03^{a}∓4,17		6,27^{c}∓3,86		12,46^{c}∓9,62		6,66^{c}∓5,683		7,14^{c}∓6,701		3,96^{a}∓1,91	
Body Adopsity İndeks	Düşük Risk	38,92^{a}∓8,15	<0,001	36,32^{a}∓7,57	<0,001	38,16^{a}∓8,25	<0,001	40,07^{a}∓8,462	<0,001	40,65^{a}∓8,290	<0,001	38,91^{ab}∓8,48	0,002
	Orta Risk	33,88^{b}∓6,21		39,74^{b}∓7,51		43,61^{b}∓7,42		39,88^{a}∓7,212		39,19^{b}∓7,784		39,52^{a}∓8,45	
	Yüksek Risk	45,45^{c}∓7,51		45,85^{c}∓7,96		32,89^{c}∓5,66		31,61^{b}∓5,719		32,17^{c}∓5,682		38,31^{b}∓7,66	

*Ortalama∓StandartSapma

Tüm değişkenlerin modele alındığı koşulda oluşan kümeler arasında kategorik değişkenlerin kategorileri bakımından anlamlı fark olup olmadığı incelendi ve elde edilen sonuçlar Tablo 4.11 ' de özetlendi. Tablolar değerlendirildiğinde; *K*-ortalama, cascade *K*-ortalama, EM, density ve *K*-medoid kümeleme algoritmaları sonucunda elde edilen kümeler arasında kalp hastalığı aile öyküsü bakımından anlamlı farklılık olmadığı belirlendi. EM ve Density kümeleme algoritmalarına göre elde edilen kümeler arasında tansiyon ilacı kullanma sıklığı bakımından anlamlı farklılık bulunmaz iken, *en uzak ilk* algoritmasında elde edilen kümeler arasında cinsiyet, sigara içme durumu, aile öyküsü, tansiyon ilacı kullanma durumu ve ATPIII sonuçlarının dağılımı bakımından anlamlı farklılık olduğu belirlendi (her biri için p<0,001).

Tablo 4.11. Tüm değişkenlerin modele alındığı koşulda kategorik değişkenlere ait kategorilerin *K*-ortalama algoritmasıyla elde edilen kümelere dağılımı

			***K*-Ortalama**			***p***
			Düşük risk grubu	**Orta risk grubu**	**Yüksek risk grubu**	
Cinsiyet	Kadın	n	3207	231	560	<0,001
		Satıra göre %	80.2	5.8	14.0	
		Sütuna göre %	93.3	30.4	94.8	
	Erkek	n	230	529	31	
		Satıra göre %	29.1	67.0	3.9	
		Sütuna göre %	6.7	69.6	5.2	
Sigara içme durumu	İçmiyor	n	2876	146	265	<0,001
		Satıra göre %	87.5	4.4	8.1	
		Sütuna göre %	83.7	19.2	44.8	
	İçiyor	n	561	614	326	
		Satıra göre %	37.4	40.9	21.7	
		Sütuna göre %	16.3	80.8	55.2	
Kalp hastalığı aile öyküsü	Yok	n	3379	745	584	0,527
		Satıra göre %	71.8	15.8	12.4	
		Sütuna göre %	98.3	98.0	98.8	
	Var	n	58	15	7	
		Satıra göre %	72.5	18.8	8.8	
		Sütuna göre %	1.7	2.0	1.2	
Hipertansiyon ilaç kullanımı	Yok	n	3311	728	170	<0,001
		Satıra göre %	78.7	17.3	4.0	
		Sütuna göre %	96.3	95.8	28.8	
	Var	n	126	32	421	
		Satıra göre %	21.8	5.5	72.7	
		Sütuna göre %	3.7	4.2	71.2	
ATPIII	Negatif	n	2503	214	32	<0,001
		Satıra göre %	91.1	7.8	1.2	
		Sütuna göre %	72.8	28.2	5.4	
	Pozitif	n	934	546	559	
		Satıra göre %	45.8	26.8	27.4	
		Sütuna göre %	27.2	71.8	94.6	

Tablo 4.12. Tüm değişkenlerin modele alındığı koşulda kategorik değişkenlere ait kategorilerin cascade *K*-ortalama algoritmasıyla elde edilen kümelere dağılımı

			Cascade *K*-Ortalama			***p***
			Düşük risk grubu	**Orta risk grubu**	**Yüksek risk grubu**	
Cinsiyet	Kadın	n	2277	998	723	<0,001
		Satıra göre %	57.0	25.0	18.1	
		Sütuna göre %	87.8	72.2	88.9	
	Erkek	n	315	385	90	
		Satıra göre %	39.9	48.7	11.4	
		Sütuna göre %	12.2	27.8	11.1	
Sigara içme durumu	İçmiyor	n	2051	528	708	<0,001
		Satıra göre %	62.4	16.1	21.5	
		Sütuna göre %	79.1	38.2	87.1	
	İçiyor	n	541	855	105	
		Satıra göre %	36.0	57.0	7.0	
		Sütuna göre %	20.9	61.8	12.9	
Kalp hastalığı aile öyküsü	Yok	n	2554	1353	801	0,230
		Satıra göre %	54.2	28.7	17.0	
		Sütuna göre %	98.5	97.8	98.5	
	Var	n	38	30	12	
		Satıra göre %	47.5	37.5	15.0	
		Sütuna göre %	1.5	2.2	1.5	
Hipertansiyon ilaç kullanımı	Yok	n	2542	1317	350	<0,001
		Satıra göre %	60.4	31.3	8.3	
		Sütuna göre %	98.1	95.2	43.1	
	Var	n	50	66	463	
		Satıra göre %	8.6	11.4	80.0	
		Sütuna göre %	1.9	4.8	56.9	
ATPIII	Negatif	n	2396	249	104	<0,001
		Satıra göre %	87.2	9.1	3.8	
		Sütuna göre %	92.4	18.0	12.8	
	Pozitif	n	196	1134	709	
		Satıra göre %	9.6	55.6	34.8	
		Sütuna göre %	7.6	82.0	87.2	

Tablo 4.13. Tüm değişkenlerin modele alındığı koşulda kategorik değişkenlere ait kategorilerin *En uzak ilk* algoritmasıyla elde edilen kümelere dağılımı

			En Uzak İlk Yöntemi			***p***
			Düşük risk grubu	**Orta risk grubu**	**Yüksek risk grubu**	
Cinsiyet	Kadın	n	3168	802	28	<0,001
		Satıra göre %	79.2	20.1	.7	
		Sütuna göre %	83.8	95.6	16.8	
	Erkek	n	614	37	139	
		Satıra göre %	77.7	4.7	17.6	
		Sütuna göre %	16.2	4.4	83.2	
Sigara içme durumu	İçmiyor	n	2998	285	4	<0,001
		Satıra göre %	91.2	8.7	.1	
		Sütuna göre %	79.3	34.0	2.4	
	İçiyor	n	784	554	163	
		Satıra göre %	52.2	36.9	10.9	
		Sütuna göre %	20.7	66.0	97.6	
Kalp hastalığı aile öyküsü	Yok	n	3750	791	167	<0,001
		Satıra göre %	79.7	16.8	3.5	
		Sütuna göre %	99.2	94.3	100.0	
	Var	n	32	48	0	
		Satıra göre %	40.0	60.0	.0	
		Sütuna göre %	.8	5.7	.0	
Hipertansiyon ilaç kullanımı	Yok	n	3637	410	162	<0,001
		Satıra göre %	86.4	9.7	3.8	
		Sütuna göre %	96.2	48.9	97.0	
	Var	n	145	429	5	
		Satıra göre %	25.0	74.1	.9	
		Sütuna göre %	3.8	51.1	3.0	
ATPIII	Negatif	n	2704	45	0	<0,001
		Satıra göre %	98.4	1.6	.0	
		Sütuna göre %	71.5	5.4	.0	
	Pozitif	n	1078	794	167	
		Satıra göre %	52.9	38.9	8.2	
		Sütuna göre %	28.5	94.6	100.0	

Tablo 4.14. Tüm değişkenlerin modele alındığı koşulda kategorik değişkenlere ait kategorilerin EM algoritmasıyla elde edilen kümelere dağılımı

			EM			***p***
			Düşük risk grubu	**Orta risk grubu**	**Yüksek risk grubu**	
Cinsiyet	Kadın	n	2907	1091	0	<0,001
		Satıra göre %	72.7	27.3	.0	
		Sütuna göre %	94.5	100.0	.0	
	Erkek	n	168	0	622	
		Satıra göre %	21.3	.0	78.7	
		Sütuna göre %	5.5	.0	100.0	
Sigara içme durumu	İçmiyor	n	3075	0	212	<0,001
		Satıra göre %	93.6	.0	6.4	
		Sütuna göre %	100.0	.0	34.1	
	İçiyor	n	0	1091	410	
		Satıra göre %	.0	72.7	27.3	
		Sütuna göre %	.0	100.0	65.9	
Kalp hastalığı aile öyküsü	Yok	n	3026	1069	613	0,578
		Satıra göre %	64.3	22.7	13.0	
		Sütuna göre %	98.4	98.0	98.6	
	Var	n	49	22	9	
		Satıra göre %	61.3	27.5	11.3	
		Sütuna göre %	1.6	2.0	1.4	
Hipertansiyon ilaç kullanımı	Yok	n	2698	962	549	0,892
		Satıra göre %	64.1	22.9	13.0	
		Sütuna göre %	87.7	88.2	88.3	
	Var	n	377	129	73	
		Satıra göre %	65.1	22.3	12.6	
		Sütuna göre %	12.3	11.8	11.7	
ATPIII	Negatif	n	1849	590	310	<0,001
		Satıra göre %	67.3	21.5	11.3	
		Sütuna göre %	60.1	54.1	49.8	
	Pozitif	n	1226	501	312	
		Satıra göre %	60.1	24.6	15.3	
		Sütuna göre %	39.9	45.9	50.2	

Tablo 4.15. Tüm değişkenlerin modele alındığı koşulda kategorik değişkenlere ait kategorilerin Density algoritmasıyla elde edilen kümelere dağılımı

			Make Density Based			***p***
			Düşük risk grubu	**Orta risk grubu**	**Yüksek risk grubu**	
Cinsiyet	Kadın	n	2875	1077	46	<0,001
		Satıra göre %	71.9	26.9	1.2	
		Sütuna göre %	99.9	94.8	5.9	
	Erkek	n	2	59	729	
		Satıra göre %	.3	7.5	92.3	
		Sütuna göre %	.1	5.2	94.1	
Sigara içme durumu	İçmiyor	n	2877	0	410	<0,001
		Satıra göre %	87.5	.0	12.5	
		Sütuna göre %	100.0	.0	52.9	
	İçiyor	n	0	1136	365	
		Satıra göre %	.0	75.7	24.3	
		Sütuna göre %	.0	100.0	47.1	
Kalp hastalığı aile öyküsü	Yok	n	2829	1114	765	0,557
		Satıra göre %	60.1	23.7	16.2	
		Sütuna göre %	98.3	98.1	98.7	
	Var	n	48	22	10	
		Satıra göre %	60.0	27.5	12.5	
		Sütuna göre %	1.7	1.9	1.3	
Hipertansiyon ilaç kullanımı	Yok	n	2511	1009	689	0,261
		Satıra göre %	59.7	24.0	16.4	
		Sütuna göre %	87.3	88.8	88.9	
	Var	n	366	127	86	
		Satıra göre %	63.2	21.9	14.9	
		Sütuna göre %	12.7	11.2	11.1	
ATPIII	Negatif	n	1711	646	392	<0,001
		Satıra göre %	62.2	23.5	14.3	
		Sütuna göre %	59.5	56.9	50.6	
	Pozitif	n	1166	490	383	
		Satıra göre %	57.2	24.0	18.8	
		Sütuna göre %	40.5	43.1	49.4	

Tablo 4.16. Tüm değişkenlerin modele alındığı koşulda kategorik değişkenlere ait kategorilerin *K*-Medoid algoritmasıyla elde edilen kümelere dağılımı

			K-Medoid			***p***
			Düşük risk grubu	**Orta risk grubu**	**Yüksek risk grubu**	
Cinsiyet	Kadın	n	2828	563	607	<0,001
		Satıra göre %	70.7	14.1	15.2	
		Sütuna göre %	85.6	70.6	88.2	
	Erkek	n	475	234	81	
		Satıra göre %	60.1	29.6	10.3	
		Sütuna göre %	14.4	29.4	11.8	
Sigara içme durumu	İçmiyor	n	2341	472	474	<0,001
		Satıra göre %	71.2	14.4	14.4	
		Sütuna göre %	70.9	59.2	68.9	
	İçiyor	n	962	325	214	
		Satıra göre %	64.1	21.7	14.3	
		Sütuna göre %	29.1	40.8	31.1	
Kalp hastalığı aile öyküsü	Yok	n	3250	782	676	0,849
		Satıra göre %	69.0	16.6	14.4	
		Sütuna göre %	98.4	98.1	98.3	
	Var	n	53	15	12	
		Satıra göre %	66.3	18.8	15.0	
		Sütuna göre %	1.6	1.9	1.7	
Hipertansiyon ilaç kullanımı	Yok	n	2990	636	583	<0,001
		Satıra göre %	71.0	15.1	13.9	
		Sütuna göre %	90.5	79.8	84.7	
	Var	n	313	161	105	
		Satıra göre %	54.1	27.8	18.1	
		Sütuna göre %	9.5	20.2	15.3	
ATPIII	Negatif	n	2203	100	446	<0,001
		Satıra göre %	80.1	3.6	16.2	
		Sütuna göre %	66.7	12.5	64.8	
	Pozitif	n	1100	697	242	
		Satıra göre %	53.9	34.2	11.9	
		Sütuna göre %	33.3	87.5	35.2	

Kardiyovasküler hastalıklar için risk faktörü olma şüphesi taşıyan tüm değişkenler modele alındığında elde edilen sonuçlara ait tablolar genel olarak değerlendirilirse, sayısal risk faktörleri içinde sadece HOMA değişkeninin kümelerin elde edilmesinde anlamlı bir etkisi olmadığı, buna karşın diğer sayısal değişkenlerin kümelerin oluşturulmasında benzer düzeyde anlamlı etkiye sahip olduğu söylenir. Ayrıca ailede kalp rahatsızlığı öyküsü olup olmaması bakımından sadece *En Uzak İlk* kümeleri arasında anlamlı farklılık bulunmuş, diğer algoritmalara göre oluşturulan kümelerin belirlenmesinde bu risk faktörünün etkisi anlamlı bulunmamıştır. EM ve Density

algoritmalarına göre elde edilen kümelerin ayrıştırılmasında hipertansiyon ilacı kullanma durumunun etki etmediği belirlenmiştir.

Bu sonuçlara göre, modele alınan tüm değişkenlerin sadece En Uzak İlk algoritmasına göre elde edilen kümelerin ortaya çıkmasında anlamlı etki sağladığı söylenebilir.

Aile öyküsü modele alınmadan Framingham skoru hesaplanırken kullanılan risk faktörleri yardımıyla elde edilen kümeleme sonuçlarının birbirleriyle uyumları Kappa istatistiği ile değerlendirilmiş ve elde edilen sonuçlar Tablo 4.17 topluca verilmiştir. Tablo 4.17 incelendiğinde, her ne kadar uygulamada kullanılan 6 algoritmanın kümeleri arasında anlamlı uyum gözlense de Kappa istatistiğinin değeri bazı koşullarda oldukça küçük bulunmuştur. Çalışmada kullanılan toplam birey sayısı ve kümelere düşen birey sayıları büyük olduğu için küçük Kappa değerleri bile istatistik olarak anlamlı bulunmuştur. Bu durumda Kappa değeri büyük olan uyumlar incelendiğinde, Cascade *K*-ortalama kümeleri ile *K*-ortalama kümeleri arasında %100 uyum bulunmuş, ayrıca Density kümeleri ile sadece EM kümeleri arasında düşük uyum bulunmuş, diğer algoritmalara ait kümeler ile oldukça yüksek uyum göstermiştir. *K*-medoid algoritmasına ait küme sonuçlarının diğer 5 algoritma kümeleri ile çok düşük düzeyde uyumlu olduğu görülmüştür.

Tablo 4.17 Aile öyküsü dikkate alınmadan Framingham skoru hesaplanırken kullanılan risk faktörleri yardımıyla elde edilen kümeleme sonuçlarının birbirleriyle uyumları

		Cascade *K*-ortalama	En Uzak İlk	EM	Density	*K*-Medoid
K-ortalama	KAPPA	1.000	0,925	0,053	0,972	0,024
	p	**<0,001**	**<0,001**	**<0,001**	**<0,001**	**0,003**
Cascade *K*-ortalama	KAPPA		0,925	0,053	0,971	0,024
	p		**<0,001**	**<0,001**	**<0,001**	**0,003**
En Uzak İlk	KAPPA			0,049	0,897	0,019
	p			**<0,001**	**<0,001**	**0,014**
EM	KAPPA				0,051	0,036
	p				**<0,001**	**<0,001**
Density	KAPPA					0,022
	p					**0,006**

Framingham skoru hesaplanırken kullanılan risk faktörleri ile birlikte aile öyküsü de modele alınarak elde edilen kümeleme sonuçlarının birbirleriyle uyumları Tablo 4.18 topluca verilmiştir. Tablo 4.18 değerlendirildiğinde, her ne kadar uygulamada kullanılan

6 algoritmanın kümeleri arasında anlamlı uyum gözlense de çalışmadaki toplam birey sayısının büyük olması, kappa değeri küçük olan bazı karşılaştırma sonuçlarının uyumlarının istatistik olarak anlamlı çıkmasına neden olmuştur. Kappa değeri büyük olan uyumlar incelendiğinde, *K*-ortalama kümeleri, Cascade *K*-ortalama kümeleri ve *En Uzak İlk* kümeleri arasında oldukça yüksek uyum olduğu, Density kümeleri ile EM kümeleri ve *K*-Medoid kümeleri arasında %100 uyum sağlandığı ve EM ile *K*-Medoid arasında ise orta düzeyde uyumun var olduğu gözlenmiştir.

Tablo 4.18. Framingham skoru hesaplanırken kullanılan risk faktörlerine ilaveten aile öyküsü de dikkate alınarak elde edilen kümeleme sonuçlarının birbirleriyle uyumları

		Cascade *K*-ortalama	En Uzak İlk	EM	Density	*K*-Medoid
K-ortalama	KAPPA	0,860	0,564	0,063	0,063	0,015
	p	**<0,001**	**<0,001**	**<0,001**	**<0,001**	**0,05**
Cascade *K*-ortalama	KAPPA		0,608	0,053	0,053	0,024
	p		**<0,001**	**<0,001**	**<0,001**	**<0,001**
En Uzak İlk	KAPPA			0,031	0,031	0,023
	p			**0,001**	**<0,001**	**0,003**
EM	KAPPA				1.000	0,334
	p				<0,001	**<0,001**
Density	KAPPA					1.000
	p					**<0,001**

Risk faktörü olarak düşünülen değişkenlerin tamamı dikkate alınarak uygulanan kümeleme algoritmaları yardımıyla elde edilen kümelerin uyumları değerlendirilmiş ve elde edilen sonuçlar Tablo 4.19' da topluca verilmiştir. Tablo 4.19 incelendiğinde tüm kümeleme algoritmaları arasındaki uyumu ölçen Kappa katsayısının istatistik olarak anlamlı olduğu sonucuna varılmıştır. Uyumlar daha detaylı olarak incelendiğinde, en yüksek uyumun EM ve Density kümeleme algoritmaları arasında, en düşük uyumun ise Density ve *K*-medoid kümeleme algoritmaları arasında olduğu görülmüştür. Ayrıca *K*-ortalama, Cascade *K*-Ortalama, *En Uzak ilk* ve Density yöntemlerinin sonuçları birbirleriyle anlamlı derecede uyumlu olup uyumun derecesi ise orta düzeydedir.

Tablo 4.19. Değişkenlerin tamamı içeren kümelerin birbirleriyle uyumları

		Cascade *K*-ortalama	En Uzak İlk	EM	Density	*K*-Medoid
K-ortalama	KAPPA	0,495	0,252	0,252	0,248	0,175
	p	**<0,001**	**<0,001**	**<0,001**	**<0,001**	**<0,001**
Cascade *K*-ortalama	KAPPA		0,285	0,195	0,169	0,225
	p		**<0,001**	**<0,001**	**<0,001**	**<0,001**
En Uzak İlk	KAPPA			0,374	0,333	0,147
	p			**<0,001**	**<0,001**	**<0,001**
EM	KAPPA				0,894	0,043
	p				**<0,001**	**<0,001**
Density	KAPPA					0,026
	p					**0,015**

Aile öyküsü hariç Framingham risk skorunun hesaplamasında kullanılan değişkenler yardımıyla elde edilen kümeler ile veri setindeki tüm değişkenler kullanılarak elde edilen kümelerin kendi içlerindeki uyumları değerlendirilmiş ve sonuçları Tablo 4.20' de topluca verilmiştir. Tablo 4.20 incelendiğinde, farklı değişkenlerin alındığı iki ayrı veri setinde Density algoritmasına ait kümelerin birbirleriyle uyumu en yüksek bulunurken diğer algoritmaların farklı değişken setlerine ait kümeleri arasında orta düzeyde ancak istatistik olarak anlamlı uyum verdiği gözlenmiştir. Bu sonuç bize Density algoritmasına ilave edilen değişkenlerin kümelemede önemli bir değişiklik meydana getirmediğini, buna karşın ilave değişkenlerin diğer algoritmaların kümelerinin şekillenmesini etkilediğini düşündürmüştür.

Tablo 4.20. Farklı değişken setleri kullanılarak oluşturulan kümelerin kendi içinde uyumları

	K-ortalama	Cascade *K*-ortalama	En Uzak İlk	EM	Density	*K*-Medoid
Kappa	0,252	0,195	0,414	0,053	0,919	0,228
p	**<0,001**	**<0,001**	**<0,001**	**<0,001**	**<0,001**	**<0,001**

Framingham risk skorunun hesaplamasında kullanılan diğer değişkenler ile birlikte aile öyküsü de dikkate alınarak oluşturulan kümeler ile veri setindeki tüm değişkenler kullanılarak elde edilen kümelerin kendi içlerindeki uyumları değerlendirilmiş ve sonuçları Tablo 4.21' de topluca verilmiştir. Tablo 4.21 değerlendirildiğinde, farklı değişkenlerin alındığı iki ayrı veri setinde Density ve EM algoritmalarının kümeleri kendi içinde istatistik olarak anlamlı bulunsa da çok düşük uyuma sahip olduğu belirlenmiştir. Bu sonuç bize, aile öyküsünün ilave edildiği durumda söz konusu iki

algoritmanın farklı değişken setlerine göre oluşturulan kümelerinin değişken setlerinden önemli düzeyde etkilendiğini göstermektedir. Diğer algoritmaların kümeleri arasındaki uyumlarda orta düzeyde bulunmuş, bu sonuç değişken setinin kümeleme üzerine etkisini göstermektedir. Bütün bu sonuçlar kümelemede kullanılacak değişkenlerin seçiminin önemini vurgulamakta olup, kümelemeye anlamlı etki sağlayan değişkenlerin alınması gerektiği vurgulanabilir. Ölçülen özellikleri bir bütünün parçası olarak düşündüğümüzde bu bütünü etkileyen ve birbirleriyle ilişki içerisinde bulunan değişkenlerin sonuç üzerindeki etkisi bir kez daha gösterilmiş olmaktadır.

Tablo 4.21. Farklı değişken setleri kullanılarak oluşturulan kümelerin kendi içinde uyumları

	K-ortalama	Cascade *K*-ortalama	En Uzak İlk	EM	Density	*K*-Medoid
Kappa	0,284	0,195	0,387	0,053	0,056	0,228
p	**<0,001**	**<0,001**	**<0,001**	**<0,001**	**<0,001**	**<0,001**

Verileri en iyi kümeleyen algoritmaları bulabilmek amacıyla ilk olarak Kappa katsayıları incelendi. Ardından Framingham skoruna göre oluşturulan risk grupları içinde düşük riskli olarak belirlenen bireylerin, kümeleme analizleri sonucunda düşük veya orta riskli bulunması önemsendi ve Framingham skoruna göre riskli olan bireylerin ise kümeleme analizlerinde riskli çıkma oranının yüksek olması tercih edildi. Ayrıca Framingham skoruna göre orta riskli bulunan bireylerin, kümeleme analizleri sonucunda orta riskli veya yüksek riskli olması durumu tercih edildi.

Yapılan değerlendirmeler sonucunda, aile öyküsü dikkate alınarak hesaplanan Framingham risk grupları ile aile öyküsü dikkate alınarak hesaplanan kümeleme grupları arasındaki en yüksek uyuma sahip kümeleme algoritmasının 0,206 ile *en uzak ilk* yönteminin olduğu belirlenmiştir. Risk şüphesi taşıyan değişkenlerin tamamı dikkate alınarak oluşturulan kümeleler ile Framingham skorununa gören elde edilen risk grupları arasındaki uyum ise en yüksek 0,325 ile yine *en uzak ilk* yönteminin olduğu belirlenmiştir. Hesaplanan uyum katsayıları istatistik olarak anlamlı ancak orta derecede bir uyumu göstermektedir. Framingham skoruna göre oluşturulan risk grupları ile Framingham değişkenleri ile birlikte aile öyküsü de dikkate alınarak yapılan kümeleme analizleri sonucunda elde edilen kümeler arasındaki uyumlar incelendiğinde Tablo 4.22'deki sonuçlara ulaşıldı. Tablo 4.22 değerlendirildiğinde, kümelerle Framingham risk

grupları arasında düşük veya orta düzeye yakın ancak istatistik olarak anlamlı uyumun elde edildiği görüldü.

Tablo 4.22. Framingham skoruna göre oluşturulan risk grupları ile Framingham değişkenleri ile birlikte aile öyküsü de dikkate alınarak yapılan kümeleme analizleri sonucunda elde edilen kümeler arasındaki uyumlar

	K-ortalama	Cascade *K*-ortalama	En Uzak İlk	EM	Density	*K*-Medoid
Kappa	0,128	0,177	0,206	0,146	0,146	0,054
p	**<0,001**	**<0,001**	**<0,001**	**<0,001**	**<0,001**	**<0,001**

Framingham skoruna göre oluşturulan risk grupları ile tüm değişkenler dikkate alınarak yapılan kümeleme analizleri sonucunda elde edilen kümeler arasındaki uyumlar incelendiğinde Tablo 4.23' deki sonuçlara ulaşıldı. Tablo 4.23 incelendiğinde, kümelerle Framingham risk grupları arasında orta düzeye yakın ancak istatistik olarak anlamlı uyumun elde edildiği görüldü.

Tablo 4.23. Framingham skoruna göre oluşturulan risk grupları ile tüm değişkenler dikkate alınarak yapılan kümeleme analizleri sonucunda elde edilen kümeler arasındaki uyumlar

	K-ortalama	Cascade *K*-ortalama	En Uzak İlk	EM	Density	*K*-Medoid
Kappa	0,249	0,198	0,325	0,177	0,152	0,165
p	**<0,001**	**<0,001**	**<0,001**	**<0,001**	**<0,001**	**<0,001**

Tablo 4.22 ve Tablo 4.23 'deki Kappa değerleri incelendiğinde Framingham risk grupları ile en yüksek uyumun *en uzak ilk* yönteminden elde edildiği görüldü. Framingham risk grupları ve Framingham risk faktörleri (aile öyküsü dâhil) kullanıldığında farklı algoritmalardan elde edilen kümeler arasında çapraz tablolar oluşturularak grupların dağılımı incelendi. Sonuç olarak Framingham risk faktörleri (aile öyküsü dâhil) kullanıldığında en isabetli kararların EM algoritmasına ait kümelerden elde edilebileceği kanaatine varıldı. Framingham skoruna göre düşük riskli olan bireylerin %20,9'u EM algoritması sonucunda yüksek riskli olarak bulundu ve Framingham skoruna göre yüksek riskli olarak adlandırılan bireylerin ise %71.7' si EM algoritması sonucunda da yüksek riskli olarak tespit edildi. Framingham skoruna göre yüksek riskli olarak adlandırılan bireylerin tamamı EM algoritması yardımıyla ya orta veya yüksek riskli olarak belirlendi. Bunlara ilaveten, Framingham skoruna göre orta

düzeyde riskli bulunan bireylerin, EM algoritması yardımıyla %33,6'sı orta riskli bulundu.

Framingham risk grupları ve tüm değişkenler dikkate alınarak oluşturulan kümeler arasında çapraz tablolar oluşturularak grupların dağılımı incelendiğinde ise, en isabetli kararların Density algoritmasına ait kümelerden elde edildiği görüldü. Framingham skoruna göre düşük riskli olan bireylerin %12,9'u Density algoritması sonucunda yüksek riskli olarak bulundu ve Framingham skoruna göre yüksek riskli olarak adlandırılan bireylerin ise %54.7' si Density algoritması sonucunda da yüksek riskli, %17' si düşük ve %28.3' ü ise orta riskli olarak tespit edildi. Ayrıca Framingham skoruna göre orta düzeyde riskli bulunan bireylerin, Density algoritması yardımıyla %33,6'sı orta riskli bulundu.

Her iki veri setinde de Framingham skoruna göre düşük riskli bulunan bireylerin en fazla oranda *en uzak ilk* kümelemede de düşük riskli olduğu görüldü.

5.TARTIŞMA VE SONUÇ

Veri madenciliği yöntemleri, çok sayıda değişken ve çok sayıda bireyleri içeren büyük veri setleri için geliştirilmiş, genellikle bireyleri veya değişkenleri sınıflandırmak ve bireyler veya değişkenler arasındaki benzerlikten yola çıkarak gruplandırmak amacıyla kullanılan çok çeşitli algoritmalardan oluşur. Kümeleme analizi veri madenciliğinde en çok yararlanılan ve homojen grupların oluşturulmasında kullanılan yöntemler grubudur. Böylece kümeleme algoritmalarında benzer özelliklere sahip olan bireyler veya ilişkili değişkenler bir araya getirilir. Birbirine benzemeyen bireyler ayrı gruplarda yani kümelerde yer alır. Aynı gruptaki bireyler birbirlerine ne kadar çok benziyor ise kümelemenin o derece iyi yapıldığı sonucuna varılır. Ayrıca kümeleme analizinde nesneler normal dağılmasa bile, uzaklık ölçülerinin normal dağılması yeterlidir. Yöntemin varsayımlarının katı olmaması tüm alanlarda uygulanmasına olanak sağlar[1,2].

Kümeleme algoritmaları çok çeşitli çalışma alanlarında kullanılmaktadır veya kullanıma uygundur. Örneğin facebook, twitter sosyal ağlarında metinlerin kümelemesi yapılabilirken, bir okulda öğrencilerin sosyal yeteneklerine göre gruplar oluşturulabilir, hastalıkların veya farklı hastalıklara sahip semptomların sınıflandırılmasında, laboratuvar ve klinik bulgular birleştirilerek hastalık gruplarının oluşturulmasında,

salgın hastalıkların belirlenmesinde, bölgesel analizlerde veya kalp seslerinin farklı morfolojilerinin belirlenmesinde, fizyolojik durumların sınıflandırılmasında, MR ve ultrason gibi çekimler yardımıyla tümörlerin tespitinde kümeleme algoritmalarından yararlanılabilmektedir. Aynı zamanda kümeleme algoritmaları, birçok istatistiksel testten önce homojen grupların ortaya çıkarılması amacıyla bir ön analiz yöntemi olarak da tercih edilmektedir[1,2].

Veri setindeki değişkenlerin tiplerine veya veride sapan/etkili gözlemlerin bulunup bulunmamasına göre çok çeşitli kümeleme algoritmaları geliştirilmiştir. Özellikle son 10 yıl içerisinde kümeleme algoritmalarının sayısında oldukça büyük artış olduğu gözlenmiştir. Bu sonuç kümeleme işlemlerine olan gereksinimin arttığının bir göstergesidir. Literatürde en çok kullanılan kümeleme algoritmaları arasında, uygulamasının kolay olması ve ilk geliştirilen kümeleme algoritmaları arasında yer alması sebebiyle *K*-ortalama yöntemi yer alır. Ancak yapılan çalışmaların önemli bir bölümünde bu algoritmanın uygulanması sırasında değişken tipi çok önemsenmediği için elde edilecek sonuçların güvenilirliği olumsuz yönde etkilenmektedir. Ayrıca veriler hakkında herhangi bir ön bilgi mevcut değil ise başlangıçta yanlış *K* değerinin belirlenmesi yanlış kümeleme sonuçlarının elde edilmesine neden olacaktır[151]. Yöntemin bir diğer dezavantajı ise gürültülü ve uç değerlere karşı fazla hassas ve kategorik değişken sayısının çok sayıda olduğu durumlardan olumsuz etkilenmektedir. Büyük veri setleri için uygun bir yöntem olmasına rağmen küresel olmayan kümeleri bulmada iyi sonuçlar üretmemektedir [12,84]. Bu olumsuzlukları ortadan kaldırmak amacıyla son yıllarda çeşitli algoritmalar geliştirilmektedir. Kümeleme amacıyla kullanılan ve özellikle son yıllarda geliştirilen algoritmalar en yoğun olarak WEKA programı içerisinde yer almaktadır. Bu programda yer alan kümeleme algoritmalarının etkinliklerinin karşılaştırılması için çeşitli çalışmalar yapılmıştır. WEKA programında yer alan JAVA kodları yardımıyla son yıllarda geliştirilmiş yeni algoritmalar ile verileri kümeleme imkânı sağlanmaktadır. Ancak JAVA kodlarının doğru bir şekilde yazılabilmesi için kullanıcının ileri derecede kod yazma yeteneğine sahip olması ve uygulayacağının kümeleme algoritması hakkında detaylı bilgilere sahip olması gerekmektedir.

Kümeleme algoritmaları özellikle ülkemizde sağlık alanı araştırmalarında henüz çok yaygın kullanılmamaktadır. Örneğin tanı ve tedavi işlemleri için birçok durumda hastalara ait çok sayıda özellik bir arada ölçülmektedir. Ölçülen özellikler arası ilişkiler

biyolojik olarak kurgulanamadığı zaman farklı istatistiksel değerlendirmelerle farklı sonuçlara ulaşılabilmektedir. Bu durum ise karar verilemeyen durumlar, tanısı konulamayan veya tanısı yanlış konulan hastalıklara neden olabilir. Böyle durumlarda değişkenler arası ilişkileri birlikte dikkate alan çok değişkenli yöntemlere olan ihtiyaç artmaktadır. Ancak hangi istatistik yöntem kullanılırsa kullanılsın biyolojik ilişkilerin doğru bir şekilde tanımlanmadığı veya konu ile ilgili değişkenler dikkate alınmadığı zaman elde edilen sonuçların doğruluğu ve genellenebilirliği tartışma konusu olacaktır. Bu yüzden benzer özellikler içeren bireylerin oluşturduğu kümeleri değerlendirirken klinik bilgilerin de göz önüne alınması gerekir[1,2,3].

Gerçekleştirilen bu tez çalışması ile bağlantılı olarak, Zheng ve arkadaşları 2005 yılında EM, *En uzak ilk* ve *K*-ortalama kümeleme algoritmalarını bir veri setinde uygulamalı olarak karşılaştırmışlardır. EM algoritmasının hangi kritere bakılırsa bakılsın, *K*-ortalama ve *En uzak ilk* algoritmalarından daha üstün olduğu sonucuna varmışlardır. EM'nin tüm veri setleri için *K*-ortalama ve *En uzak ilk* yönteminden daha küçük standart sapmaya sahip olduklarını tespit etmişler ve bu durumun EM algoritmasının diğer iki yöntemden daha durağan sonuçlar verdiğini ifade ettiğini belirtmişlerdir. *K*-ortalama ve *En uzak ilk* yöntemleri karşılaştırıldığında ise *K*-ortalama yönteminin doğruluk oranlarına ve log marjinal olabilirlik skoruna bakarak daha iyi sonuç verdiğini gözlemişlerdir[152].

Osama Abbas 2008 yılında farklı kümeleme algoritmalarını karşılaştırmış, *K*-ortalama ve EM algoritmalarının performansının hiyerarşik kümeleme yöntemlerinden daha iyi olduğu sonucuna varmışlardır. Ayrıca büyük veri tabanları için EM ve *K*-ortalama algoritmalarının oldukça iyi sonuçlar üretebileceklerini vurgulamışlardır. Ancak *K*-ortalama ve EM algoritmaları gürültülü veriler için oldukça hassas olduğu belirlenmiştir[153].

Tagaram Madhulatha 2011 yılında yaptığı çalışma da *K*-ortalama ve *K*-medoid kümeleme algoritmalarını karşılaştırmış ve küçük veri setleri için *K*-ortalama yönteminin kaliteli ve hızlı sonuçlar ürettiğini gözlemlemişlerdir. *K*-medoid yönteminin ise daha büyük veri setleri için kullanabilir olduğunu, büyük veri setleri için *K*-ortalama yöntemine göre daha hızlı ve daha iyi küme grupları oluşturduğunu ifade etmişlerdir[154]

Singh ve Chauhan 2011 yılında en çok kullanılan kümeleme algoritmaları arasında yer alan *K*-ortalama ve *K*-medoid yöntemlerini uygulamalı olarak karşılaştırmışlardır.

Gerçekleştirdikleri çalışma sonucunda, her iki yönteminde küresel şekile sahip küçük veya orta hacimdeki veri setleri için iyi sonuçlar ürettiğini gözlemlemişler. Her iki kümeleme algoritması için de başlangıçta *K* değerinin doğru bir şekilde belirlenmesi gerektiğini belirtmişlerdir. *K*-ortalama yönteminin hesaplama maliyetini daha düşük bulmuşlar ancak gürültülü ve aşırı uç değerlere karşı oldukça hassas olduğunu gözlemlemişler. *K*-medoid yöntemi ise gürültülü ve uç değerlere karşı hassas değildir ancak *K*-ortalama yöntemine göre hesaplama maliyeti oldukça fazla olduğunu görmüşlerdir[155,156]

Shrivastava ve Arya 2012 yılında *K*-ortalama, yoğunluk yöntemi ve *En uzak ilk* kümeleme algoritmalarını uygun bir veri seti üzerine uygulayarak karşılaştırmıştır. Kümeleme yöntemlerinin performansları ise yanlış sınıflandırılan örneklerin yüzdeleri ile ölçülmüştür. Bu oran ne kadar düşük çıkmışsa kümelemenin performansının da o derecede düşük olduğunu kabul etmişlerdir. *En uzak ilk* kümeleme algoritması, *K*-ortalama ve yoğunluk yöntemlerine göre daha iyi sonuçlar vermiştir. *En uzak ilk* yöntemi küme sayısından bağımsız olduğunu ancak *K*-ortalama yöntemi küme sayısına oldukça bağımlı sonuçlar ürettiğini gözlemlemişlerdir. *En uzak ilk* kümeleme algoritması hızlı sonuç üretmektedir. Ayrıca bu algoritmanın aynı küme sayılarında diğer yöntemlere göre daha düşük hatalı sınıflama olasılığına sahip olduğu görülmüş, küme sayısındaki değişikliklerin ise hatalı sınıflandırılan birey oranlarında önemli bir değişime neden olmadığı belirlenmiştir[157].

Sharma ve arkadaşları 2012 yılında WEKA'da kullanılan algoritmaları karşılaştırmışlar ve EM kümeleme algoritmasının gerçek veri setleri için oldukça kullanışlı olduğu sonucuna varmışlardır. Ancak algoritma yapısı gereği oldukça karmaşıktır. *En uzak ilk* kümeleme algoritmasının ise büyük ölçekli veri tabanları için kullanışlı olduğunu öne sürmüşlerdir. *K*-ortalama yöntemi bakımından değerlendirildiklerinde ise yöntemin fazla miktarda değişkenler içeren veri setleri için hiyerarşik kümelemeye göre daha hızlı sonuçlar verdiğini belirtmişlerdir. Ayrıca *K*-ortalama yöntemi daha sıkı yapıda kümeleri oluşturmaktadır. Yöntemin dezavantajının ise kümeleme sonucunda oluşturulan kümelerin kalitelerinin karşılaştırılmasının zor olduğunu öne sürmüşlerdir. Sabit küme sayısı ile de *K* değerinin ne olacağı hakkında tahminde bulunmanın ise zor olacağını belirtmişlerdir. *K*-ortalama yöntemi global olmayan kümeleri bulmada iyi çalışmamaktadır. Yapılan çalışmada en kolay olan yöntemin *K*-ortalama olduğu

sonucuna varılmış ve kümeleme yöntemleri değerlendirilirken kullanılabilecek, öğrenmesi zor olmayan WEKA programını önermektedirler[109].

Godara ve Yadav'ın 2013 yılında gerçekleştirdikleri çalışmada *K*-ortalama, hiyerarşik ve density kümeleme yöntemlerini incelemişlerdir. Yaptıkları çalışma sonucunda *K*-ortalama yönteminin oldukça hızlı sonuçlar verdiğini ve etkili bir performansının olduğunu belirlemişlerdir. Ayrıca diğer iki algoritmaya oranla kümeleme sonuçları daha doğrudur. Make density kümeleme algoritması *K*-ortalama yöntemine göre daha doğru sonuçlar üretmiştir. Ancak *K*-ortalama yöntemine kıyasla daha fazla zaman aldığı tespit edilmiştir. Zaman ve doğruluk kriterlerinin her ikisi de birlikte değerlendirildiğinde bahsedilen üç algoritma içerisinde en iyi yöntemin *K*-ortalama kümeleme algoritması olduğu sonucuna varmışlardır[158].

Singh ve Dubey 2013 yılında *K*-ortalama ve *En uzak ilk* algoritmalarını karşılaştırmışlardır. Yaptıkları çalışma sonucunda doğruluk oranlarının *K*-ortalama yöntemine göre daha yüksek olduğunu gözlemlemişler ve daha hızlı sonuçlar vermiştir[159].

Revathi ve Nalini 2013 yılında çalışmamızda kullandığımız *K*-ortalama, *En uzak ilk* ve make density based algoritmalarını karşılaştırmışlardır. Tüm algoritmalarda küme sayısı arttıkça kümeleri oluşturmada da harcanan zamanın arttığını gözlemlemişler. Bunun yanı sıra *En uzak ilk* yöntemindeki uygulamaların çok kısa bir zaman sürdüğü, *K*-ortalamanın ise en uzun zamanda kümeleme sonuçlarını verdiği sonucuna varmışlardır. Böylece *K*-ortalama yönteminin çok büyük veri setleri için kullanımının uygun olmayacağını önermektedirler[160].

Balabantaray ve arkadaşları 2015 yılında *K*-ortalama ve *K*-medoid yöntemlerini karşılaştırmışlardır. Karşılaştırmalar sonucunda *K*-ortalama yönteminin *K*-medoid kümeleme algoritmasından daha iyi sonuçlar verdiğini görmüşlerdir. Benzerlik ölçüsünü geliştirdiği için büyük veri setleri için *K*-ortalama yönteminin daha iyi sonuç ürettiğini belirtmişlerdir[161].

Kakkar ve Parashar 2014 yılında WEKA'da kullanılan *K*-ortalama, hiyerarşik yöntem, EM ve yoğunluğa dayalı kümeleme algoritmalarını karşılaştırmışlardır. Gerçekleştirdikleri çalışma sonucunda *K*-ortalama yönteminin hiyerarşik ve EM algoritmalarından daha hızlı sonuçlar ürettiğini gözlenmemişler. Ancak yine de *K*-

ortalama yönteminin yoğunluk algoritmasına göre daha yavaş kaldığını belirlemişler. Ayrıca log-olabilirlik oran değerleri karşılaştırıldığında en yüksek değere sahip yöntemin yoğunluğa dayalı algoritma olduğu görülmüştür. Bu değerlendirmelere ek olarak *K*-ortalama, EM ve yoğunluğa dayalı yöntemlerin aynı küme örneklerine sahip olduklarını gözlemlemişler. Ancak EM algoritmasının kümeleri oluşturmada *K*-ortalama ve yoğunluğa dayalı yöntemden daha çok zaman aldığını gözlemlemişler. Dolayısıyla *K*-ortalama ve yoğunluğa dayalı algoritmanın EM yönteminden daha iyi olduğunu belirtmişlerdir. Yoğunluğa dayalı algoritma daha az zaman almasına rağmen, *K*-ortalama yönteminin kümeleri oluştururken daha iyi olduğunu gözlemlemişlerdir. Hiyerarşik yöntem ise *K*-ortalama yönteminden daha fazla zaman almış ve küme özelliklerinin de iyi olmadığı belirlenmiştir. Sonuç olarak kriterler değerlendirildiğinde en iyi yöntemin *K*-ortalama olduğu sonucuna varmışlardır[162].

Jung ve arkadaşları tarafından 2014 yılında *K*-ortalama ve EM kümeleme algoritmalarını karşılaştırmışlardır. Yaptıkları çalışma sonucunda *K*-ortalama yönteminin EM algoritmasından doğruluğunun daha fazla olduğunu tespit etmişlerdir. Ancak *K*-ortalama yöntemi EM yöntemine göre daha fazla zaman aldığını belirlemişlerdir[163]

Goyal 2014 yılında WEKA'da kullanılan COBWEB, DBSCAN, EM, *En uzak ilk* ve *K*-ortalama kümeleme algoritmalarının veri setleri üzerinde uygulayarak en iyi yöntemlerin EM ve *K*-ortalama olduğu sonucuna varmıştır[164].

Ramya ve Rao 2014 yılında *K*-ortalama, EM, CLOPE, COBWEB, *En uzak ilk*, Filtered Cluster, Hiyerarşik kümeleme, make density based kümeleme algoritmalarının etkilnliklerini farklı veri setleri üzerinde karşılaştırmışlardır. Karşılaştırmalar sonucunda en az zaman karmaşıklığına sahip kümeleme algoritmasının *En uzak ilk* yöntemi olduğu sonucuna varmışlardır. Ayrıca tüm veri setleri için uygulanan farklı algoritmalardan en tutarlı olan algoritmanın *En uzak ilk* kümeleme yöntemi olduğunu gözlemlemişlerdir[165].

Jung ve arakadaşları 2015 yılında yaptıkları çalışmada ise EM algoritmasının doğruluğunun oldukça yüksek olduğunu göstermişlerdir ve algoritmanın süreci de oldukça hızlı bir şekilde tamamlanmıştır. *K*-ortalama yönteminin de doğruluk oranı oldukça yüksek çıkmasına rağmen EM algoritması ile karşılaştırıldığında *K*-ortalamanın daha yavaş olduğu sonucuna varmışlardır[163].

Çalışmamızda kullandığımız Framingham risk skoru birçok uluslararası ve ulusal çalışmalarda kardiyovasküler hastalık geçirme riskinin tahmininde kullanılır. Bu skor 1200 üzerindeki makalelerde kardiyovasküler hastalığının dönüm noktasının bulunmasında kullanılmıştır. Ancak bu risk skoru hesaplanırken modelde sınırlı sayıda risk faktörü yer almaktadır. Bu faktörler; yaş, cinsiyet, total kolesterol seviyesi, sistolik kan basıncı, sigara kullanma ve diyabet olma durumlarını içerir. Ancak çalışmada kullandığımız veri setinde yer alan kişilerin hiç birinde diyabet tanısı olmadığı için bu skorun hesaplanmasında diyabet etkisi bulunmamaktadır. Framingham risk skoru 10 yıllık periyot süresince bireyin kardiyovasküler rahatsızlık geçirme riskini tahmin eden matematiksel bir modeldir. Ancak yöntemin dezavantajları mevcuttur. Framingham risk skoru sadece bireyler için uygulanan ve bireylerin daha önceden kalp hastalığı teşhisi olmadan uygulanan bir tahmin modelidir. Hesaplanan Framingham risk skoru sadece kardiyovasküler hastalık riski için tahmin edebilmektedir. Diğer kalp rahatsızlıkları için bu skor kullanılamamaktadır. Skorun bir başka dezavantajı ise yapılan kohort çalışmasında genç bireylerin daha az temsil edilmesidir. Dolayısıyla Framingham risk skoru yaş üzerine yoğun bir vurgu yapmaktadır. Genç insanlarda da kardiyovasküler hastalık geçirme riski olabilirken, Framingham risk skorunda bu grubun daha az temsil edilmesi bir eksikliktir. Bu risk skoru Framingham bölgesinde yaşayan kişilerden elde edilen bilgiler yardımıyla geliştirilmiştir. Ancak bu bölge daha çok beyaz ırkları ve homojen grupları içermektedir. Dolayısıyla geliştirilen Framingham risk skoru farklı etnik ve kültürel yapılar için doğru olmayabilir veya eksik olabilir. Örneğin Japon Amerikalı erkeklerde Framingham risk skoru ile hesaplanan risk 5 yıl daha fazla tahmin edilmiştir. Bu da bu skorun her etnik yapı ve grup için uygulanamayacağının bir göstergesidir. Framingham risk skoru diyabetli olan popülasyonların kardiyovasküler rahatsızlık geçirme risklerinin hesaplanabilmesi için de kullanışlı değildir. Çünkü Framingham risk skorunun uygulandığı kohort grubu bir çoğunluğu diyabet rahatsızlığına sahip bireylerden oluşmaktadır. Bunun dışında bir başka dezavantajı ise, tahmin ederken kullandığı eşitliklerde koroner rahatsızlıkların potansiyel risk faktörleri arasında yer alan aile öyküsü ve prematüre koroner hastalık bilgilerini dikkate almamaktadır. Özellikle aile öyküsü klinik olarak hastalığın teşhisi için iyi bir ayraçtır. Framingham risk skorunun diğer bir potansiyel dezavantajı ise sadece 10 yıllık bir periyotu içeren bir tahmin modelini geliştirebiliyor olabilmesidir. Bunun dışında 10 yıllık modelde risk faktörlerinin de etkileri zamana bağlı olarak daha kuvvetli olabilmektedir. Ancak Framingham risk skoru bu etkiyi ihmal etmektedir. Dolayısıyla

ülkemiz için kardiyovasküler hastalık geçirme riskinin tahmininde altın standart test olarak kullanılmamalıdır. Ancak bu skordan bir ön değerlendirme niteliğinde yararlanılabilir[166].

Biz de bu doğrultuda en çok ölüme sebebiyet veren hastalıklar arasında yer alan kardiyovasküler hastalıkların risk faktörlerini araştırdık. Bu risk faktörlerinden hareketle de kişileri düşük riskli, orta riskli ve yüksek riskli olmak üzere üç kümeye ayırdık. Ardından Framingham skorundan elde edilen skor grupları ile farklı kümeleme algoritmaları sonucunda elde edilen kümeleri karşılaştırdık. Ancak çalışmada bazı kısıtlılıklar mevcuttur. Literatürde birçok kümeleme algoritmasının olmasına rağmen yalnızca altı yöntem gerçek veri seti üzerinde tartışılmıştır. Kullanılan yöntemler WEKA ve Rapid Miner programları içerinde yer alan yöntemlerdir.

Elde edilen bulgular değerlendirildiğinde, veri setimizin büyük olması ve kategorik değişkenlerin az miktarda olması sebebiyle *K*-medoid yönteminin, *K*-ortalama yönteminden daha kullanabilir olduğunu belirlendi. *K*-medoid yönteminin kappa uyumunun düşük olmasına rağmen, *K*-ortalama yönteminin büyük veri setlerinde kullanılamaması ve sayısal değişkenler için uygun olması sebebiyle *K*-ortalama yöntemi veri setimiz için güvenilir sonuçlar üretmemesi sebebiyle *K*-medoid yönteminin kullanımının daha uygun olacağı belirlendi. Dolayısıyla *K*-ortalama algoritmasının kısıtlılıkları ve uygulama alanları dikkate alındığında çalışmamızda kullanılan veri seti için nispeten uygun olmayan bir kümeleme yöntemi olacağı kanaatine varılmıştır. Başlangıçta *K* değeri hakkında bilgi sahibi olmamız sebebiyle cascade *K*-ortalama yönteminin avantajlarından yararlanılamamıştır. Benzer olarak bu yöntemde *K*-ortalama algoritması gibi uygulanan veri seti için nispeten diğer yöntemlere göre daha az uygundur. Framingham risk skoru ile en uyumlu olan algoritma ise *En uzak ilk* algoritmasıdır. Algoritma büyük veri setleri için oldukça hızlı ve etkili sonuçlar üretmektedir. Aynı zamanda kullanılan veri seti için de en hızlı sonucu veren algoritma *En uzak ilk* kümeleme algoritması olduğu belirlendi. Olasılıklar değerlendirildiğinde ise en iyi yöntemlerin EM ve Make Density based yöntemi olduğu görüldü. Bu yöntemlerde benzer olarak büyük veri setleri için uygundur. EM algoritması uygulama ve anlaşılması bakımından oldukça basittir. Maliyeti ve zaman kaybı oldukça azdır. Aynı zamanda eğer veride kayıp nesneler olması durumda bile bu verileri tahmin edebilme özelliğine sahiptir. Bu bilgilere ek olarak Make Density based kümeleme algoritmasının yüksek riskli bireyleri bulma eğilimine sahip olduğunu görüldü. Bir

başka ifadeyle gerçekte Framingham skoru sonucunda riskli olan bireylerin kümeleme algoritmalarının uygulanması sonucunda da gerçekten riskli olma oranı en yüksek Make density based kümeleme algoritması olduğunu belirlendi. Her iki veri setinde de düşük riskli bireyleri bulmaya en elverişli yöntem ise en uzak komşu algoritması olduğu tespit edilmiştir. Çünkü gerçekte Framingham skoru sonucunda düşük riskli olanların, en uzak komşu yöntemi sonucunda da yüksek bir oranın düşük riskli olduğu görülmüştür.

Ancak en iyi yöntem olarak belirlenen bu yöntemler klinik olarak değerlendirilmelidir. Tüm kümeleme algoritmaların sonuçları teker teker irdelenerek sonuçlar yorumlanmalıdır. Uygulanan bu çalışmada klinik olarak kardiyovasküler hastalığı geçirme riski yüksek olan bireyleri bulmak daha değerli olduğu için çapraz tablolarda bu oranın yüksek olması istenmektedir. Ancak bazı çalışmalarda örneğin düşük riskli olan bireyleri bulmak başarı olarak kabul edilebilmektedir. Bu durumda sonuçlar bu değerlendirme ölçüsünde incelenmelidir. Veri setlerinde şu algoritma kesin olarak daha iyi sonuç verir gibi kesin bir sonuca varmak araştırıcıları hataya düşürebilmektedir. Kümeleme algoritmalarındaki kriterler, varsayımlar, algoritmaların kullanım koşulları, dezavantaj ve avantajları bir bütün olarak değerlendirilerek sağlık alanında klinik bilgiler ışığında kümeleme yöntemleri yorumlanmalıdır. İstatistiksel yöntemler bilindiği üzere klinik bulguları destekler nitelikte olmalıdır ki uygulamada rahatlıkla ve doğru bir şekilde kullanılma imkânı olsun.

Sonuç olarak sağlık alanında kümeleme algoritmalarının uygulamalarının artırılmasını önermekteyiz. Doğru yöntem kullanıldığı takdirde sağlık politikalarının geliştirilmesine ve hastalık riski olan bireyler belirlenerek önem alınır. Böylece halkımızın yaşam kalitesinin artmasına neden olunacak ve ortalama yaşam süresi artacaktır. Dolayısıyla basit bir kümeleme algoritması bile belki de sağlık alanında çok fazla değişiklerin olmasına ve tıpta gelişmelere neden olacaktır.

6.KAYNAKLAR

1. Han J, Kamber M, Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc. USA; 2006: p.5-10.

2. Kob HC, Tan G. Data mining applications in healthcare. J Healthc Inf Manag. 2005;19(9): 64-72.

3. Köktürk F, Ankaralı H, Sümbüloğlu V. Veri madenciliği yöntemlerine genel bakış. Türkiye Klinikleri Journal of Biostatistics. 2009; 1(1): 20-5.

4. Fraley C, Raftery AE. How many clusters? Which clustering method? Answers via model-based cluster analysis. The Computer Journal. 2005; 41 (8): 578-588.

5. Ferligoj A, Batagelj V. Some types of clustering with relational constraint. Psychometrika.1983; 48: 541-552.

6. An Introduction to Cluster Analysis for Data Mining, http://www-users.cs.umn.edu/~han/dmclass/ cluster_survey_10_02_ 00.pdf, Erişim Tarihi: 15.12.2014.

7. Han J, Kamber M, Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc. USA; 2006: p.386-397

8. İstatistiksel Araştırmalarda Ölçme Yöntemleri ve Ölçek Türleri, http://tip.baskent.edu.tr/egitim/mezuniyetoncesi/calismagrp/ogrsmpzsnm12/10.3.pdf, Erişim Tarihi:19.07.2014.

9. Aggarwal CC, Reddy CK (Eds). Data Clustering Algorithms and Applications. CRC Press. USA;2014: p.100-1.

10. Anderberg MR. Cluster Anaysis for Applciations. Academic Press. New York;1973.

11. Murtagh F, Contreras P. Methods of hierarchical clustering. [Electoronic Journal]. http://arxiv.org/pdf/1105.0121.pdf. May 2011; 21.

12. Demiralay M. Hiyerarşik Kümeleme Metodları İle Veri Madenciliği Uygulamaları 2005, Marmara Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek lisans tezi, 124 sayfa, İstanbul, (Doç.Dr. Yılmaz Çamurcu).

13. Bilgin A. Merkez Tabanlı Kümeleme Algoritmalarının Karşılaştırılması.2008, Kocaeli Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek lisans tezi, 204 sayfa, Kocaeli, (Yrd. Doç. Dr. Nevcihan Duru).

14. Özkan Y. Veri Madenciliği Yöntemleri. Papatya Yayıncılık. İstanbul; 2008:143-8.

15. Hubert L. Approximate evaluation techniques for the single-link and complete-linkhierarcihal clustering procedures. Journal of the American Statistical Association.1974;69: 698-704.

16. Sokal RR and Michener CD. A statistical method for evaluating systematic relationships. Univ. Kansas Sci. Bull.1958; 38(2): 1409–1438.

17. McQuitty LL. Similarity analysis by reciprocal pairs for discreteand continuous data. Educ. Psychol. Meas.1966; 26(4): 825-831.

18. Sangün L. Temel Bileşenler Analizi, Ayırma Analizi, Kümeleme Analizleri ve Ekolojik Verilere Uygulanması Üzerine Bir Araştırma. 2007, Çukurova Üniversitesi, Fen Bilimleri Enstitüsü, Doktora tezi, 261 sayfa, Adana, (Prof. Dr. Mustafa Akar).

19. Carvalho AXY, Albuquerque PHM, Junior GRA, Guimaraes RD. Spatial hierarchical clustering. Rev. Bras. Biom. 2009;27(3): 411-442.

20. Hair JF, Anderson RE, Tatham R L, Black WC. Multivariate Data Analysis with Readings. Prentice-Hall. USA;1995.

21. Yalçın N. Kümeleme Analizi ve Uygulaması. 2013, Fırat Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek lisans tezi, 66 sayfa, Elazığ, (Doç. Dr. Mahmut Işık).

22. Frank IE, Todeschını R. The Data Analysis Handbook. Elsevier. 1994; p.154.

23. Cluster Analysis Ward'd Method, http://sites.stat.psu.edu/~ajw13/stat505/fa06/19_cluster/09_cluster_wards.html, Erişim Tarihi: 19.09.2014.

24. Murtagh F, Legendre P. Ward's hierarchical agglomerative clustering method: Which algorithms implement Ward's criterion? Journal of Classification.2014;31: 274-295.

25. Aggarwal CC, Reddy CK (Eds). Data Clustering Algorithms and Applications. CRC Press. USA;2014: p.287.

26. Theodoridis S, Koutroumbas K. Pattern Recognition. 4th ed. USA: Academic Press; 2009: p.653-700.

27. Guha S, Rastogi R, Shim K. ROCK: A robust clustering algorithm for categorical attributes. Proceedings of the IEEE International Conference on Data Engineering. Sydney. 345-366. 23-26 March 1999.

28. Baş F. Kategorik Veri Analizinde Kullanılan Algoritmaların Performanslarının Karşılaştırılması Üzerine Bir Çalışma. 2013, Gazi Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek lisans tezi,68 sayfa, Ankara, (Prof. Dr. Semra Oral Erbaş).

29. Rafsanjani MK, Varzaneh ZA and Chukanlo NE. A survey of hierarchical clustering algorithms, TJMCS.2012;5(3): 229-240.

30. Akın YK. Veri Madenciliğinde Kümeleme Algoritmaları ve Kümeleme Analizi.2008, Marmara Üniversitesi, Sosyal Bilimler Enstitüsü, Doktora tezi, 189 Sayfa, İstanbul, (Prof. Dr. Şahamet Bülbül).

31. Karypis G, Han EH, Kumar V. CHAMELEON: A hierarchical clustering algorithm using dynamic modeling. IEEE Computer. 1999;32(8): 68-75.

32. Han J, Kamber M, Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc. USA; 2006: p.359.

33. Silahtaroğlu G. Veri Madenciliğinde Kümeleme Analizi ve Öğretim Başarısının Değerlendirilmesine İlişkin Bir Uygulama.2004, İstanbul Üniversitesi, Sosyal Bilimler Enstitüsü, Doktora tezi, 216 Sayfa, İstanbul, (Prof. Dr. Öner Esen).

34. Sudipto G,Rastogi R, Shim K. CURE: An efficient clustering algorithm for large databases. roceedings ACM SIGMOD International Conference on Management of Data. Washington.73-84, 2-4 June 1998.

35. Demiralay M,Çamurcu Y. CURE, AGNES ve *K*-means algoritmalarındaki kümeleme yeteneklerinin karsılaştırılması. İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi.2005;4(8):1-18.

36. Rafsanjani MK, Varzaneh ZA and Chukanlo NE. TJMCS.2012;5(3): 229-240.

37. Valgeirsson AG. Erlingsson B. Einarson İS. Using clustering to index image descriptors: A performance evaluation. ReykJavik University, *B.Sc. Project* 2003.

38. Rani Y, Manju and Rohil H. Comparative analysis of BIRCH and CURE hierarchical clustering algorithm using WEKA 3.6.9. The SIJ Transactions on Computer Science Engineering & its Applications.2014;2(1): 25-9.

39. Durmuş MS. Veri Kümeleme Algoritmalarının Performansları Üzerine Karşılaştırmalı Bir Çalışma.2005, Pamukkale Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek lisans tezi, 134 Sayfa, Denizli, (Yrd. Doç. Dr. Serdar İplikçi).

40. Zhang T,Raghu R, Miron L. BIRCH: An efficient data clustering method for very large databases. In Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data. Canada, 103-114, 4-6 July 1996.

41. Han J, Kamber M, Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc. USA; 2006: p.412-3.

42. Bilgin TT. Veri Madenciliğinde Kümeleme Analizi Yöntemi Uygulaması.2003, Marmara Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek lisans tezi, 135 Sayfa, İstanbul, (Doç. Dr. Yılmaz Çamurcu).

43. Han J, Kamber M, Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc. USA; 2006: p.416-7.

44. Karypis G, Han EH and Vipin Kumar. CHAMELEON: A hierarchical clustering algorithm using dynamic modeling. IEEE Computer.1999; 32(8): 68-9.

45. Kaufman L, Rousseeuw PJ. Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley and Sons. USA;2005: p.253-263.

46. R Documentation https://www.r-project.org/other-docs.html, Erişim Tarihi: 1.01.2015.

47. Monothetic Analysis (MONA), http://www.unesco.org/webworld/idams/advguide/Chapt7_1_6.htm, Erişim Tarihi: 15.02.2015.

48. Budayan C. Determination of Strategic Groups of Major Turkish Construction Companies By Using Self Organizing Map And Cluster Analysis. 2008, Middle East Technical Unıversity, Natural and Applied Sciences, PhD thesis, 289 Pages, Ankara, (Doç. Dr. İrem Dikmen Toker, Prof. Dr. M. Talat Birgönül).

49. Anuradha J,Tripathy BK. Hierarchical clustering algorithm based on attribute dependency for attention deficit hyperactive disorder. I.J. Intelligent Systems and Applications. 2014;06: 37-45.

50. Rehman M, Mehdi SA. Comparison of density-based clustering algorithms. [ElectronicJournal], http://www.researchgate.net/publication/242219043_COMPARISON_OF_DENSITY-BASED_CLUSTERING_ALGORITHMS, 2014; 5.

51. Zaki MJ, Meira Jr W. Data Mining and Analysis: Fundamental Concepts and Algorithms. Chapter 15: Density-based Clustering. Cambridge University Press. New York;2014: p.372.

52. BäCKLUND H, Hedblom A, Neijman N. DBSCAN: A density-based spatial clustering of application with noise. Linköpings Universitet. http://staffwww.itn.liu.se/~aidvi/courses/06/dm/Seminars2011/DBSCAN(4).pdf, Erişim tarihi: 15.04.2015.

53. Ester M, Kriegel HP, Sander J, Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings 2nd International Conference on Knowledge Discovery and Data Mining. Oregeon, 226-231,1996.

54. Han J, Kamber M, Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc. USA; 2006:p.418-9.

55. AL-ZAND HAR. Bölümleyici Kümeleme Algoritmalarının Farklı Veri Yoğunluklarında Karşılaştırılması.2013, Gazi üniversitesi, Bilişim Enstitüsü, Yüksek lisans tezi,107 Sayfa, Ankara, (Yrd. Doç. Dr. Hacer Karacan).

56. Hinneburg A, Keim DA. An efficient approach to clustering in large multimedia databases with noise. Proc. 4th International Conference on Knowledge Discovery and Data Mining. New York, 58-65, 27-31 August 1998.

57. Bilgin TT, Çamurcu Y. DBSCAN, OPTICS ve *K*-means kümeleme algoritmalarının uygulamalı karşılaştırılması. Politeknik Dergisi.2005; 8(2):139-145.

58. Ankerst M, Breunig MM, Kriegel HP, Sander J. OPTICS: Ordering points to identify the clustering structure. ACM SIGMOD International Conference on Management of Data. Philadelphia, 49-60, June 1999.

59. Han J, Kamber M, Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc. USA; 2006:p.418-420.

60. Density-based algorithms for active and anytime clustering. Dissertation, LMU München: Faculty of Mathematics, Computer Science and Statistics. http://edoc.ub.uni-muenchen.de/17533/, Erişim Tarihi: 01.04.2015.

61. Priyadarishini A, Karthik S, Anuradha J and Tripathy BK. Diagnosis of psychopathology using clustering and rule extraction using rough set. Advances in Applied Science Research. 2011; 2 (3): 346-362.

62. Tan J, Zhang J and Li W. An improved clustering algorithm based on density distribution function. Computer and Information Science. 2010;3(3):23-9.

63. Aggarwal CC, Reddy CK (Eds). Data Clustering Algorithms and Application. CRC Press. USA; 2014: p.88.

64. MacQueen J. Some methods for classification and analysis of multivariate observations. In Procedings of the Fifth Symposium on Math, Statistics and Probability. California, 271-297, 1967.

65. The University of Edinburgh Scholl of Informatics, www.inf.ed.ac.uk/teaching/courses/inf2b/ learnnotes/ inf2b-sr13-notes.pdf, Erişim Tarihi: 10.02.2015.

66. Işık M. Bölünmeli Kümeleme Yöntemleri ile Veri Madenciliği Uygulamaları.2006, Marmara Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek lisans tezi, 92 Sayfa, İstanbul, (Yrd. Doç.Dr. Özgül Vayvay).

67. Jun W, Zheng OA. The Research of K-means clustering algorithm based on association rules. International Conference on Challenges in Environmental Science and Computer Engineering. Wuhan, 285-6, 6-7 March 2010.

68. Wang J, Su X. An improved *K*-Means clustering algorithm. 3rd International Conference on Communication Sofware and Networks, Xi'an, 44-46, 27-29 May 2011.

69. Özkan Y. Veri Madenciliği Yöntemleri. Papatya Yayıncılık. İstanbul; 2008.

70. Aggarwal CC, Reddy CK. Data Clustering Algorithms and Application. CRC Press. USA; 2014: p.91.

71. Arthur D, Vassilvitskii S. *K*-means++: The advantages of careful seeding. In Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms. New Orleans, 1027–1035, 7-9 January 2007.

72. Bradley PS, Fayyad UM. Refining initial points for *K*-means clustering. In Proceedings of the Fifteenth International Conference on Machine Learning. San Francisco, 91-9, 24-27 July 1998.

73. Hartigan JA and Wong MA. Algorithm as 136: A *K*-means clustering algorithm. Journal of the Royal Statistical Society.1979;28(1):100-8.

74. Milligan GW. A Monte Carlo study of thirty internal criterion measures for cluster analysis. Psychometrika.1981; 46(2):187–199.

75. Aggarwal CC, Reddy CK. Data Clustering Algorithms and Application. CRC Press. USA; 2014: p.92-93.

76. Ball GH, Hall DJ. ISODATA, A Novelmethod of Data Analysis and Pattern Classification. Stanford Research Institude; California; 1965: 15-31.

77. Cali´nski T and J. Harabasz. A dendrite method for cluster analysis. Communications Instatistics Theory and Methods.1974; 3(1):1–27.

78. Duda RO, Hart PE, Stork DG. Pattern Classification and Scene Analysis.2nd Ed.USA: John Wiley Wiley&Sons;2001:p.526-8.

79. Drineas P, Frieze A, Kannan R, Vempala S, Vinay V. Clustering large graphs via the singular value decomposition. Machine Learning. 2004;56(1):9-33.

80. Mojena R. Hierarchical grouping methods and stopping rules: An evaluation. The Computer Journal.1977;20(4):359–363.

81. Newman MEJ, Girvan M. Finding and evaluating community structure in networks. Physical Review E.2003; 69(2):026113.

82. Tibshirani R, Walther G, and Hastie T. Estimating the number of clusters in a data set via the gap statistic. Journal of the Royal Statistical Society: Series B (Statistical Methodology).2001; 63(2):411–423.

83. Yeung KY, Fraley C, Murua A,Raftery AE and Ruzzo WL. Model-based clustering and data transformations for gene expression data. Bioinformatics.2001; 17(10):977–987.

84. Syed AA. Performance Analysis of *K*-means Algorithm and Kohonen Networks.2004, Florida Atlantic University, Master of Science, Yüksek lisans tezi,127 Sayfa, Amerika, (Prof. Dr. Abhijit Pandya).

85. Zhang C and Fang Z. An improved *K*-means clustering algorithm. Journal of Information & Computational Science.2013;10(1):193-9.

86. Department of Statistics, http://www.stats.ox.ac.uk/search?queries_standard_query=documents&search_page_10996_submit_button=., Erişim Tarihi: 09.04.2015.

87. Salem SA, Nandi AK. New assessment criteria for clustering algorithms. Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing. Mystic, 285-290, 28-30 September 2005.

88. Han J, Kamber M, Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc. USA; 2006:p.404-5.

89. Fırat E. Öğrenci Harf Notlarının *K*-means Kümeleme Algoritması ile Belirlenmesi.2012, İstanbul Teknik Üniversitesi, Fen Edebiyat Fakültesi, 27 Sayfa, İstanbul, (Yrd. Doç.Dr. Ahmet Kırış).

90. Velmurugan T and Santhanam T. Computational complexity between *K*-means and *K*-medoids clustering algorithms for normal and uniform distribution of data points. Journal of Computer Science.2010;6(3):363-368.

91. Sarma TH, Viswanath P and Reddy BE. Single pass kernel *K*-means clustering method. S$\bar{a}$dhan$\bar{a}$. 2013;38(3):407-419.

92. Zhang R and Rudnicky A. A large scale clustering scheme for kernel k-means. ICPR. 2012: 289–292.

93. Chitta R, Jin R, Havens TC, Jain AK. Approximate kernel k-means: Solution to large scale kernel clustering. In: Proceedings of 17th ACM SIGKDD International Conference on Knowledge Discovery in Databases. San Diago, 895-903, 21-24 August 2011.

94. Hitendra ST, Viswanath P, Eswara RB. A fast and approximate kernel k-means clustering method for large datasets. In: Proceedings of International Conference on Recent Advances in Intelligent Computational Systems. India, 545–550, 22-24 Semptember 2011.

95. Ng RT, Han J. Efficient and effective clustering methods for spatial data mining. Proceedings of the 20th VLDB Conference. Santiago, 146-147, 12-15 September 1994.

96. Han J, Kamber M, Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc. USA; 2006:p.407.

97. Aggarwal CC, Reddy CK. Data Clustering Algorithms and Application. CRC Press. USA; 2014: p.261.

98. Raymond TN and Han J. CLARANS: A method for clustering objects for spatial data mining. IEEE Transactions on Knowledge and Data Engineering.2002;14(5):1003-1016.

99. Han J, Kamber M, Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc. USA; 2006:p.406.

100. Bo F,Wenning H,Gang C, Dawei J,Shuining Z. An improved PAM algorithm for optimizing initial cluster center. 3rd International Conference on Software Engineering and Service Science. Beijing,24-27, 22-24 June 2012.

101. Yılancı V. Bulanık kümeleme analizi ile Türkiye'deki illerin sosyoekonomik açıdan sınıflandırılması. Süleyman Demirel Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi.2010;15(3):453-470.

102. Yang Y. Image segmentation by fuzzy c-means clustering algorithm with a novel penalty term. Computing and Informatics.2007;26: 17-31.

103. Sivaram K and Saveetha D. An effective algorithm for outlier detection. Global Journal of Advanced Engineering Technologies. 2013;2: 35-40.

104. Aggarwal S and Kaur P. Survey of partition based clustering algorithm used for outlier tetection. IJETT. 2013;1(5): 687-690.

105. Soni D,Jha N and Sinwar D. Discovery of outlier from database using different clustering algorithms. Indian J. Edu. Inf. Manage. 2012;1(9): 388-391.

106. Pelleg D, Moore AW. X-means: Extending *K*-means with efficient estimation of the number of clusters. Proceedings of the Seventeenth International Conference on Machine Learning. Stanford,727-734, 29 June-2 July 2000.

107. Günay Atbaş AC. Kümeleme Analizinde Küme Sayısının Belirlenmesi Üzerine Bir Çalışma. 2008, Ankara Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek lisans tezi,68 Sayfa, Ankara, (Yrd. Doç.Dr. Cemal Atakan).

108. Vadeyar D Yogish HK. Farthest First clustering in links reorganization. International Journal of Web & Semantic Technology.2014;5(3):17-24.

109. Sharma N, Bajpai A and Litoriya R. Comparison the various clustering algorithms of weka tools. International Journal of Emerging Techonology and Advanced Engineering.2012;2(5):73-80.

110. Aggarwal CC, Reddy CK (Eds). Data Clustering Algorithms and Application. CRC Press. USA; 2014: p.128-9.

111. Cheng W,Wang W,Batista S. Grid-based Clustering, in Data Clustering: Algorithms and Applications. CRC Press. USA; 2012:p.132-3.

112. Wang, W,Yang J,Muntz R. STING: A statistical information grid approach to spatial data mining. International Conference of Very Large Data Bases. Athens,186-195,25-29 August 1997.

113. Qiang Z,Zheng Z, Zhi S,Edward D. WINP: A window-based incremental and paralel clustering algorithm for very large databases. 17th IEEE International Conference on Tools with Artificial Intelligence. Hong Kong, 8, 16 November 2005.

114. Cheng W,Wang W,Batista S. Grid-based Clustering, in Data Clustering: Algorithms and Applications. CRC Press. USA; 2012:p.134.

115. Sheikholeslami G,Chatterjee S, Zhang A. Wavecluster: A multi-resolution clustering approach for very large spatial databases. International Conference Very Large Data Bases. New York,428-439, August 1998.

116. Han J, Kamber M, Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc. USA; 2006:p.427-8.

117. Bilgin A. Merkez Tabanlı Kümeleme Algoritmalarının Karşılaştırılması.2008, Kocaeli Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek lisans tezi.204 Sayfa, Kocaeli, (Yrd. Doç. Dr. Nevcihan Duru).

118. Cheng W,Wang W,Batista S. Grid-based Clustering, in Data Clustering: Algorithms and Applications. CRC Press. USA; 2012:p.139.

119. Han J, Kamber M, Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc. USA; 2006:p.436-8.

120. Parsons L, Haque E, Liu H. Evaluating subspace clustering algorithms. SIAM International Conference on Data Mining. Florida,48-56, 22-24 April 2004.

121. Karar Destek Sistemleri, http://www.slideshare.net/oguzzhanoguzz/karar-aalar-ve-entropi-kds, Erişim Tarihi: 04.05.2015.

122. Cheng W,Wang W,Batista S. Grid-based Clustering, in Data Clustering: Algorithms and Applications. CRC Press. USA; 2012:p.140.

123. Goil S, Nagesh H,Choudhary A. MAFIA: Efficient and scalable subspace clustering for very large data sets. In Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Diago, 443-452, August 1999.

124. Cheng W,Wang W,Batista S. Grid-based Clustering, in Data Clustering: Algorithms and Applications. CRC Press. USA; 2012:p.141.

125. Yanchang Z,Junde S. GDILC: A grid-based density-isoline clustering algorithm. International Conferences on Info-Tech and Info-Net Proceedings.China,140-5, 29 October-1 November 2001.

126. Yan B, Deng G. Improved clustering algorithm based on density-isoline. Open Journal of Statistics.2015;5: 303-310.

127. Ma L,Gu L,Li B,Zhou L and Wang J. An improved grid-based *K*-means clustering algorithm. Advanced Science and Technology Letters. 2014;73: 1-6.

128. Aggarwal C, Reddy CK (Eds). Data Clustering Algorithms and Application. CRC Press. USA; 2014: p. 278-9.

129. M. Dutta, A.Kakoti Mahanta, Arun K. Pujari, QROCK: A quick version of the ROCK algorithm for clustering of categorical data. Pattern Recognition Letter. 2005;26: 2364-2373.

130. Advance Database Management. http://webcache.googleusercontent.com/search?q=cache:PM3Q17ZGD8kJ:student

s.ou.edu/M/Felix.Mathew-1/finalreport.doc+&cd=1&hl=tr&ct=clnk&gl=tr, Erişim Tarihi: 05.05.2015.

131. CACTUS-Clustering Categorical Data Using Summaries, http://www.cs.cornell.edu/johannes/papers/1999/kdd1999-cactus.pdf, Erişim Tarihi 06.06.2015.

132. Aggarwal CC, Reddy CK (Eds). Data Clustering Algorithms and Applications. CRC Press. USA; 2014: p.289.

133. Andritsos P, Tsaparas P,J.Miller R, and C.Sevcik K. LIMBO: Scalable clustering of categorical data. Advances in Database Technology.2004;2992:123-146.

134. Aggarwal CC, Reddy CK (Eds). Data Clustering Algorithms and Applications. CRC Press. USA; 2014: p.286-7.

135. Barbara D,Couto J., Li Y., COOLCAT: An entropy-based algorithm for categorical clustering. In: Proc. of CIKM'02. Japan, 582-589,2002.

136.Clustering Categorical Data with *K*-Modes, http://www.irma-international.org/viewtitle/10828/, Erişim Tarihi:15.03.2015.

137. Tian B, Kulikowski CA, Leiguang G, Bin Y, Lan H and Chunguang Z. A global *K*- modes algorithm for clustering categorical data, Chinese Journal of Electronics.2012;21(3):460-5.

138. Huang Z and Michael KN, A fuzzy *K*-modes algorithm for clustering categorical data. IEEE Transactions On Fuzzy Systems.1999;7(4):446-452.

139. Ji J, Pang W, Zhou C, Han X and Wang Z, A fuzzy k-prototype clustering algorithm for mixed numeric and categorical data. Konwledge-Based Systems.2012; 30: 129-135.

140. Ji J, Bai T, Zhou C, Ma C and Wang Z. An improved k-prototypes clustering algorithm for mixed numeric and categorical data. Neurocomputing.2013;120: 590-6.

141. Zaki MJ, Peters M. CLICKS: Mining subspace clusters in categorical data via *K*- partite maximal cliques. 21st International Conference on Data Engineering. Tokyo. 355 -6, 05-08 April 2005.

142. He Z, Xu X and Deng S. Squeezer: An efficient algorithm for clustering categorical data, J. Comput. Sci. & Technol. 2002; 17(5):611-624.

143. He Z, Xu X and Deng S, Scalable algorithms for clustering large datasets with mixed type attributes. Int J Int Syst. 2005;20(10):1077-1089.

144. Andreopoulos B,An A, Wang X. Hierarchical Density-Based Clustering of Categorical Data and a Simplification. Springer. Berlin;2007:p.11-22.

145. Yang Y, Guan X, You J. CLOPE: A Fast and Effective Clustering Algorithm for Transactional Data. KDD '02 Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Canada, 682-687, 23-25 July 2002.

146. Jie L, Xinbo G, Licheng J. A fuzzy CLOPE algorıthm and its optimal parameter choice. Chinese Journal Of Electronics.2006; 23(3):384-388.

147. Ienco, D, Pensa RG and Meo R. From context to distance: Learning dissimilarity for categorical data clustering. ACM Trans. Knowl. Discov. Data. 2012;6(1):00- 022.

148. Aggarwal CC, Reddy CK (Eds). Data Clustering Algorithms and Application. CRC Press. USA; 2014: p.61-76.

149. Servi T. Çok Değişkenli Karma Dağılım Modeline Dayalı Kümeleme Analizi.2009, Çukurova Üniversitesi, Fen Bilimleri Enstitüsü, Doktora tezi,266 Sayfa, Adana, (Prof. Dr. Hamza Erol).

150. Bonora E, Formentini G, Calcaterra F, Lombardi S,Marini F, Zenari L, Saggiani F, Poli M, Perbellini S, Raffaelli A, Cacciatori V, Santi L, Targher G, Bonadonna R, Muggeo M.HOMA estimated insülin resistance is an independent predictor of cardiovascular disease in type 2 diabetic subject. Diabets Care.2002;7: 1135-1141.

151. Hamerly G, Elkan C. Learning the *K* in *K*-Means. Advances in Neural Information Processing Systems 16. Canada, 8-13 December 2013.

152. Zheng X, Cai Z, Li Q. An Experimental Comparison of Three Kinds of Clustering Algorithms. International Conference on Neural Networks and Brain Conference. China, 13-15 October 2005.

153. Abbas OA. Comparisons between data clustering algorithms. The International Arab Journal of Information Technology.2008; 5(3):320-5.

154. Madhulatha TS. Comparison between *K*-Means and *K*-Medoids clustering algorithms. Advances in Computing and Information Technology Communications in Computer and Information Science. 2011;198: 472-481.

155. Singh SS, Chauhan NC. *K*-means v/s *K*-Medoids: A Comparative Study. National Conference on Recent Trends in Engineering&Techonology. India, 13-14 May 2011.

156. Tiwari M, Singh R. Comparative investigation of *K*-means and *K*-medoid algorithm on iris data. International Journal of Engineering Research and Development. 2012; 4(8):69-72.

157. Shrisvastava V, Arya PN. A study of various clustering algorithms on retail sales data. International Journal of Computing, Communications and Networking. 2012;1(2): 98-74.

158. Godara S, Yadav R. Performance analysis of clustering algorithms for character recognition using Weka tool. International Journal of Advanced Computer and Mathematical Sciences. 2013; 4(1):119-123.

159. Singh B, Dubey G. A comparative analysis of different data mining using WEKA. International Journal of Innovative Research&Studies. 2013; 2(5):380-391.

160. Revathi S, Nalini T. Performance comparison of various clustering algoritm, International Journal of Advanced Research in Computer Science and Software Engineering. 2013; 3(2): 66-72.

161. Balabantaray RC, Sarma C, Jha M. Document clustering using *K*-means and *K*-medoids. International Journal of Knowledge Based Computer Systems.

http://arxiv.org/ftp/arxiv/papers/1502/1502.07938.pdf . [Electronic Journal] ,February 2015;1.

162. Kakkar P, Parashar A. Comparison of different clustering algrithms using WEKA tool. International Journal of Advanced Research in Technology, Engineering and Science. 2014; 1(2): 20-2.

163. Jung YG, Kang MS, Heo J. Clustering performance comparison using *K*-means and expectation maximization algorithm. Biotechnol Biotechnol Equip. 2014; 28(1): 44-8.

164. Goyal VK. An experimental analysis of clustering algorithms in data mining using Weka tool. International Journal of Innovative Research in Science & Engineering. 2014; 2(4): 171-6.

165. Ramya D, Rao D. Performans evaluation of learning by example techniques over different datasets. International Journal of Innovative Research in Science, Engineering and Technology. 2014; 3(9): 15960-5.

166. Hemann BA, Bimson WF, Taylor AJ. The Framingham risk score: an appraisal of its benefits and limitations. Am Hearth Hosp J. 2007; 5(2): 91-6.

ÖZGEÇMİŞ

9.2.1990 yılında Kadıköy'de doğdum. İlkokul, ortaokul ve lise eğitimimi İstanbulda'da tamamladıktan sonra 2007 yılında Gazi Üniversitesi Fen-Edebiyat Fakültesi İstatistik Bölümünü kazanıp, 2011 yılında okul üçüncülüğü derecesi ile üniversiteden mezun oldum. 2012 Aralık ayında Öğretim Üyesi Yetiştirme Programı çervesinde Düzce Üniversitesi Sağlık Bilimleri Enstitüsü Biyoistatistik ve Tıbbi Bilişim Anabilim Dalına araştırma görevlisi olarak atandım ve yüksek lisans eğitimime aynı üniversitede devam ettim. 2015 yılının Ağustos ayında "Sağlık Alanında Yapılan Araştırmalarda Kümeleme Algoritmalarının Kullanımı: Bir Uygulama" isimli tezimle yüksek lisans eğitimimim sonuna gelmiş bulunmaktayım.