ÇOK AMAÇLI GENETİK ALGORİTMA İLE KARIŞIK VERİLERİN SINIFLANDIRILMASI

ONUR CAN SERT

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ

TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TEMMUZ 2012
ANKARA

Fen Bilimleri Enstitü onayı

_____

Prof. Dr. Ünver Kaynak

Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

_____

Doç. Dr. Erdoğan Doğdu

Anabilim Dalı Başkanı

Onur Can SERT tarafından hazırlanan ÇOK AMAÇLI GENETİK ALGORİTMA İLE KARIŞIK VERİLERİN SINIFLANDIRILMASI adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

_____

Yrd. Doç. Dr. Tansel ÖZYER

Tez Danışmanı

Tez Jüri Üyeleri

Başkan : Yrd. Doç. Dr. Esra KADIOĞLU _____

Üye : Doç. Dr. Bülent TAVLI _____

Üye : Yrd. Doç. Dr. Tansel ÖZYER _____

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

Onur Can SERT

Üniversitesi : **TOBB Ekonomi ve Teknoloji Üniversitesi**

Enstitüsü : **Fen Bilimleri**

Anabilim Dalı : **Bilgisayar Mühendisliği**

Tez Danışmanı : **Yrd. Doç. Dr. Tansel ÖZYER**

Tez Türü ve Tarihi : **Yüksek Lisans – Temmuz 2012**

ONUR CAN SERT

ÇOK AMAÇLI GENETİK ALGORİTMA İLE KARIŞIK VERİLERİN SINIFLANDIRILMASI

ÖZET

Son yıllarda gittikçe büyüyen veri kümeleri içerisinden kullanıcının işine yarayacak olan saklı bilgiye ulaşmak ve çıkarmak gittikçe önemini arttıran bir araştırma konunusudur. Bu bilgiler üzerinden veriler arasında bulunan ilişkiler saptanabilir ve çeşitli yöntemler kullanılarak bu verilerin öbeklenmesi ve sınıflandırılması sağlanabilir. Bu bilgilerin çıkartılması adına bir çok algoritma geliştirilmiştir ve bu işlemler şu anda bankacılık, biyoenformatik, sağlık sektörü ve benzeri bir çok alanda aktif olarak kullanılmaktadır.

Sadece numerik veya sadece kategorik öznitelikler içeren veri kümeleri için bu öbekleme işlemlerini yapan k – means, k – modes gibi algoritmalar mevcuttur fakat numerik ve kategorik özniteliklerin karışık olarak yer aldığı veri kümeleri için çözüm üreten çok sayıda yöntem bulunmamaktadır.

Bu tezde karışık özniteliklerden oluşan veri kümelerinin öbeklenmesine yönelik bir araştırma yapılmış ve bu doğrultuda bir çözüm yöntemi önerilmiştir. Önerilen çözüm yönteminde karışık öznitelikler içeren veri kümeleri özniteliklerinin türleri doğrultusunda ayrılmakta ve değerlendirilmekte daha sonra ise numerik ve kategorik olarak ayrı ayrı alınan sonuçlar birleştirilerek sonuca ulaşılmaktadır. Bu işlemlerin yapılabilmesi adına numerik ve kategorik öznitelikler için farklı uzaklık (benzerlik) metrikleri tanımlanmıştır. Son olarak ise tanımlanan bu uzaklık metrikleri bir k – means yapısına oturtularak istenilen algoritma elde edilmiştir. Bu algoritmadan elde edilen sonuçlar üzerinden çeşitli metrikler doğrultusunda ideal öbek sayıları tespit

edilmeye çalışılmış ve elde edilen sonuçların başarımları saflık metriği adı verilen bir metrik hesaplanmış ve farklı yöntemler ile elde edilen sonuçlarla karşılaştırılmıştır.

Anahtar Kelimeler: öbekleme, sınıflandırma, kategorik veriler, numerik veriler, karışık veriler, çok amaçlı öbekleme, genetik algoritma, küçük ve büyük ölçekli veri kümesi

University : **TOBB Economics and Technology University**
Institute : **Institute of Natural and Applied Sciences**
Science Programme : **Computer Engineering**
Supervisor : **Assistant Prof. Dr. Tansel ÖZYER**
Degree Awarded and Date : **Master of Science – July 2012**

ONUR CAN SERT

CLUSTERING MIXED DATASETS USING MULTI OBJECTIVE GENETIC ALGORITHM

ABSTRACT

Collecting and extracting the useful information for users from the datasets becomes very popular and important among the research areas of computer sciences. For using the extracted information people can easily create links between the different data and make clustering or classification operations with them. In order to do that information extraction process, there are remarkable number of algorithms are developed and they are used in areas like banking, bioinformatics and medicine.

There are lot of algorithms which are do clustering operations for datasets which are included only numerical attributes or only categorical attributes. However the number of the algortihms convenient for the mixed datasets, which are included both numerical and categorical attributes, are very low.

In this thesis, it has been stutied on developing a new clustering algorithm for all the three types (numerical, categorical and mixed) of datasets. The algorithm which is proposed is seperating the types of the attributes as numerical and categorical, calculating the distances between the data and returning a clustering result. For calculating the distance between two datum, there are fitness functions. Fitness functions are also seperated for numerical and categorical attributes and they are use in the same way as the fitness functions in the k – modes and k – means algorithm. Finally the clustering results, which are returned from the algorithm, are evaluated and the optimal clustering numbers are detected. The success of the results are evaluated with purity index and they are compared with the results of the other algorithms.

Keywords: clustering, classification, categorical datasets, numerical datasets, mixed datasets, multi objective clustering, genetic algorithm, small scale dataset, large scale dataset

TEŞEKKÜR

Bu tezin hazırlanmasında yardım ve katkılarıyla beni yönlendiren değerli Hocam Yrd. Doç. Dr. Tansel Özyer'e, yüksek lisans eğitimim boyunca bana değerli katkılarda bulunan TOBB Ekonomi ve Teknoloji Bilgisayar Mühendisliği bölümü Hocalarıma, tez çalışmalarım boyu bana büyük yardımları olan değerli arkadaşım Kayhan Dursun'a, yüksek lisans çalışmalarım boyunca bana maddi konularda destek verip verimli bir şekilde çalışmama katkıda bulunan TÜBİTAK'a ve en önemlisi beni bugünlere getiren değerli aileme teşekkürü borç bilirim.

İÇİNDEKİLER

TEZ BİLDİRİMİ iii
ÖZET iv
ABSTRACT vi
TEŞEKKÜR viii
İÇİNDEKİLER ix
GRAFİKLERİN LİSTESİ xi
ŞEKİLLERİN LİSTESİ xii
TABLOLARIN LİSTESİ xiii
KISALTMALAR xiv
SEMBOL LİSTESİ xv
1. GİRİŞ 1
2. İLGİLİ ÇALIŞMALAR 4
2.1. Veri Madenciliği 4
2.2. Öbekleme 5
2.2.1. Parçalı Öbekleme Algoritmaları 6
2.2.2. Hiyerarşik Öbekleme Algoritmaları 7
2.2.2.1. Birleşen Öbekleme Algoritmaları 8
2.2.2.2. Ayrılan Öbekleme Algoritmaları 8
2.3. Karışık Veri Kümeleri için Öbekleme Algoritmaları 9
2.4. Genetik Algoritma 11
3. GENETİK ALGORİTMA YARDIMIYLA ÇOK AMAÇLI ÖBEKLEME 13
3.1. Genel Bakış 13
3.2. Kromozomların Kodlanması 16
3.3. Karışık Çaprazlama (Shuffle Crossover) 17

3.4. Mutasyon 20
3.5. Kullanılan Uygunluk Fonksiyonları 20
3.5.1. K – Modes İçsel Uzaklık 21
3.5.2. K – Modes Dışsal Uzaklık 22
3.5.3. K – Means İçsel Uzaklık 23
3.5.4. K – Means Dışsal Uzaklık 24
3.5.5. Kategorik Öznitelikler İçin Önerilen Olasılıksal Bir Yöntem 25
3.5.6. Numerik Öznitelikler İçin Önerilen Ağırlıklı Bir Yöntem 30
3.5.7. Kategorik Öznitelikler İçin Önerilen Ağırlıklı Bir Yöntem 33
3.5.8. Hem Numerik Hem de Kategorik Öznitelikler İçin Önerilen Ağırlıklı Bir Yöntem 34
3.6. K – Means / K – Modes Operatör 37
4. DOĞRU SAYIDAKİ ÖBEK SAYISININ TESPİTİ VE DEĞERLENDİRME METRİKLERİ 39
4.1. H – Confidence ile Öbeklerin Birleştirilmesi 39
4.2. Saflık İndeksi 40
4.3. Delta IEE Kare Metriği ile Öbek Sayısı Tahmini 42
5. DENEYLER 45
5.1. Iris Veri Kümesi 45
5.2. Zoo Veri Kümesi 47
5.3. Congressional Voting Veri Kümesi 48
5.4. Heart Disease Veri Kümesi 50
5.5. Australian Credit Veri Kümesi 52
6. SONUÇ 54
KAYNAKLAR 56
ÖZGEÇMİŞ 59

GRAFİKLERİN LİSTESİ

Grafik 5.1. Iris veri kümesi için $\Delta^2 IEE$ sonuçları 46
Grafik 5.2. Zoo veri kümesi için $\Delta^2 IEE$ sonuçları 48
Grafik 5.3. Congressional Voting veri kümesi için $\Delta^2 IEE$ sonuçları 49
Grafik 5.4. Heart Disease veri kümesi için $\Delta^2 IEE$ sonuçları 51
Grafik 5.5. Australian Credit veri kümesi için $\Delta^2 IEE$ sonuçları 53

ŞEKİLLERİN LİSTESİ

Şekil 2.1. Veri tabanlarında bilgi keşfi süreci (VTBK) 4
Şekil 3.1. Genetik algoritma yardımıyla karışık veri kümeleri için çok amaçlı öbekleme algoritmasının akış şeması 14
Şekil 3.2. Önerilen algoritma sonucunda ortaya çıkan örnek bir matris 15
Şekil 3.3. Her elamanın gösterimi için 3 bit ayrılan bir kromozom dizisinin gösterimi 16
Şekil 3.4. İki kromozom için olağan çaprazlama örneği 18
Şekil 3.5. İki kromozom için karışık çaprazlama örneği 19
Şekil 3.6. Mutasyon işlemi 20
Şekil 4.1. Örnek bir öbekleme sonucu 41

TABLOLARIN LİSTESİ

Tablo 3.1. Örnek sinema veri kümesi 25
Tablo 3.2. Öznitelik değerlerinin bir arada bulunma olasılıkları 29
Tablo 3.3. Öznitelik değerleri arasında hesaplanan uzaklık değerleri 30
Tablo 3.4. Numerik öznitelik içeren örnek veri kümesi..... 31
Tablo 3.5. Numerik öznitelikler üzerinde ayrıklaştıma işlemi yapılmış örnek veri kümesi 32
Tablo 5.1. Kullanılan veri kümeleri ve özellikleri 45
Tablo 5.2. Iris veri kümesi için saflık indeksi sonuçları 46
Tablo 5.3. Zoo veri kümesi için saflık indeksi sonuçları 47
Tablo 5.4. Congressional Voting veri kümesi için saflık indeksi sonuçları..... 49
Tablo 5.5. Heart Disease veri kümesi için saflık indeksi sonuçları 50
Tablo 5.6. Australian Credit veri kümesi için saflık indeksi sonuçları..... 52

KISALTMALAR

Kısaltma	Açıklama
NSGA	Non – dominated Sorting Genetic Algorithm
VTBK	Veritabanı Bilgi Keşfi
EE	Expected Entropy
IEE	Increase Rate of Expected Entropy

SEMBOL LİSTESİ

Bu çalışmada kullanılmış olan simgeler açıklamaları ile birlikte aşağıda sunulmuştur.

Simge	Açıklama
$\boldsymbol{D}$	Veri kümesi
$\boldsymbol{C_k}$	k numaralı öbeğin merkezi
$\boldsymbol{C_k^r}$	k numaralı öbeğin sadece numerik öznitelikler doğrultusunda hesaplanan merkezi
$\boldsymbol{C_k^c}$	k numaralı öbeğin sadece kategorik öznitelikler doğrultusunda hesaplanan merkezi
$\boldsymbol{A}$	Tüm öznitelik içerisindeki i numaralı öznitelik
$\boldsymbol{A_i^r}$	Numerik öznitelikler içerisindeki i numaralı öznitelik
$\boldsymbol{A_i^c}$	Kategorik öznitelikler içerisindeki i numaralı öznitelik
$\boldsymbol{N}$	Veri kümesi içerisinde yer alan toplam eleman sayısı
$\boldsymbol{N_k}$	k. öbekteki toplam eleman sayısı
$\boldsymbol{k}$	Toplam öbek sayısı
$\boldsymbol{b}$	Her bir elemanın kromozom içerisinde ifade edildiği bit sayısı
$\boldsymbol{w^r}$	Numerik öznitelikler için ağırlık değeri
$\boldsymbol{w^c}$	Kategorik öznitelikler için ağırlık değeri
$\boldsymbol{x_i}$	x elamanı içerisinde yer alan j özniteliğinin değeri
$\boldsymbol{d(x,y)}$	x ve x veri elemanları arasındaki uzaklık
$\boldsymbol{\delta(x_i,y_i)}$	x ve y elemanlarının i. öznitelikleri arasındaki uzaklık değeri

BÖLÜM 1

1. GİRİŞ

Öbekleme elde bulunan bir veri kümesini alt kümelere bölme ve bu kümeler ile ifade etme şeklidir. Bu öbekleme işlemleri elde bulunan çeşitli veri kümeleri üzerinde kullanılarak bu veri kümelerinin özellikleri hakkında bilgi edinilebilir. Elde edilen bilgiler günlük hayatta bankacılık, biyoloji, sosyoloji gibi birçok alanda kullanılmaktadır. Bu alanlarda kulanımı için birer örnek vermek gerekirse örneğin bankacılık sektöründe müşterilerin özniteliklerini içeren bir veri kümesi doğrultusunda kredi verilmeye uygun olan ve olmayan müşterilerin kümelenmesi, biyolojide öznitelik olarak çeşitli deney sonuçları elde edilmiş olan hücrelerin hastalıklı olduğu veya hastalıksız olduğu şeklinde iki kümeye bölünmesi ve son olarak sosyolojide yine kişileri ve bu kişilerin çeşitli öznitelikleri için bir kümeleme işlemi yapıp hangi insanların hangi partiye oy verdiğinin kümelenmesi gibi örnekler verilebilir.

Öbekleme işlemi yapan algoritmalar kullanıcıdan aldığı veya kendisinin belirlediği doğrultuda veri kümesinin kaç ayrı öbek içerdiğini tespit etmekte ve çeşitli uygunluk fonksiyonları kullanarak bir sonraki aşamada veriler arasındaki uzaklıkları hesaplamakta ve bu uzaklıklar doğrultusunda öbekleme işlemlerini yapmaktadır. Bu noktada öbek sayısının belirlenmesi, verilerin tutulması şekli ve uygunluk fonksiyonları için birçok farklı yaklaşım önerilmiştir.

Aynı zamanda öbeklenecek küme içerisinde tutulan verilerin sahip oldukları özniteliklere göre de yapılacak işlemler çeşitlilik göstermektedir. Bunun nedeni veri kümesinin içerdiği özniteliklerin numerik değerlerden, kategorik değerlerden veya hem kategorik hem de numerik değerlerden oluşabilecek olmasından kaynaklanmaktadır. Öbekleme sonucunda elde edilen sonuçların hangi ölçüde başarılı oldukları da çeşitli metrikler ile hesaplanabilmektedir.

Bu tez kapsamında yeni bir öbekleme algoritması geliştirilmiştir. Bu algoritma ile ilgili konuya yapılan katkılar aşağıdaki gibi sıralanabilir;

- Algoritma hem kategorik öznitelikler içeren veri kümeleri, hem numerik öznitelikler içeren veri kümeleri hem de numerik ve kategorik öznitelikleri bir arada içereren karışık tipteki veri kümeleri için çalışacak şekilde geliştrilmiştir.
- Genetik algoritma tabanlı bir algoritma oluşturulmuş ve veriler algoritmada yer alan kromozomlar içerisinde ikilik tabanda ifade edilmiştir.
- Birden fazla uygunluk fonksiyonu kullanılarak çok amaçlı bir yapı oluşturulmuştur. Farklı uygunluk fonksiyonları doğrultusunda değerler hesaplanmış ve daha NSGA[1] ismi verilen algoritma ile alternatif sonuçlar elde edilmiştir.
- Öbekleme işlemi yapılırken kullanıcı tarafından bir bilgi girişinden yararlanılmadan, veri kümesinin kaç adet öbeğe bölünmesi gerektiği çeşitli değerler için otomatik olarak hesaplanmıştır ve algoritmanın sonunda en uygun olan öbek sayısının algoritma tarafından seçilmesi sağlanmıştır.
- Farklı metrikler yardımıyla uygun olarak seçilen öbek sayısının farklı durumlardaki uygunlukları gözden geçirilmiş ve alternatif sonuçlar üretilmiştir.
- Algoritma sonucunda elde edilen sonuçların başarımı, saflık indeksi ismi verilen bir metrik yardımıyla hesaplanmıştır.

Bu tez çalışması şu şekilde düzenlenmiştir. Bölüm 1'de, tez çalışmasının fikir temellerini içeren ve yapılan çalışma hakkında kısa bir bilgi veren giriş bölümü bulunmaktadır. Bölüm 2'de, veri madenciliği, öbekleme, daha önceki çalışmalarda geliştirilmiş olan öbekleme algoritmaları ve genetik algoritma yapısı gibi çalışmanın temelini oluşturan konular hakkında genel bilgiler anlatılmıştır. Bölüm 3'de, bu çalışmada önerilen genetik algoritma yardımıyla çok amaçlı öbekleme algoritmasına değinilmiş ve bu algoritmada kullanımış olan uygunluk fonksiyonlarından bahsedilmiştir. Bölüm 4'te, öbekleme algoritması sonucu elde edilmiş olan sonuçların, doğru sayıdaki öbek değerlerinin tespit edilmesinden ve sonuçların değerlendirilmesinde kullanılan metriklerden bahsedilmiştir. Bunu takiben Bölüm 5'te, önerilen yöntemlerin çeşitli veri kümeleri üzerinde yapılan test işlemlerine ve elde edilen sonuçların farklı öbekleme algoritmaları doğrultusunda elde edilen

sonuçlarla olan karşılaştırılmalarına yer verilmiştir. Bölüm 6, sonuç bölümünde ise, varılan sonuçlar açıklanarak, gelecek çalışmalara değinilip tez çalışması sonuçlandırılmıştır.

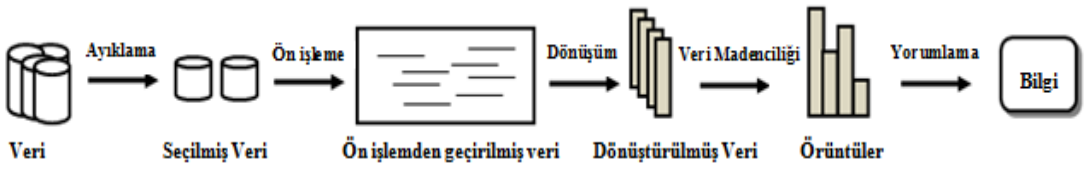
BÖLÜM 2

2. İLGİLİ ÇALIŞMALAR

Bu bölümde tez çalışmasında geliştirilen yöntemin temellerinde kullanılan kavramlar ile ilgili bilgiler verilecektir.

2.1. Veri Madenciliği

Veriler doğru analiz edildikleri zaman kişilere oldukça yararlı ve yönlendirici bilgiler sağlayabilmektedirler. Teknolojinin ilerlemesi ve imkanların artması ile birlikte farklı alanlarda elde edilen verilerin sayısında büyük oranda artışlar olmaktadır. Bunun bir sonucu olarak da farklı alanlarda çok çeşitli ve farklı büyüklüklerde veri bankaları oluşmuştur. Bu veri bankaları üzerinde çeşitli işlemler yaparak bilgi çıkarımı işlemi yapılmasına veritabanı bilgi keşfi (VTBK)[2] ismi verilmiştir. Bu bilgi keşfi sürecinde birçok farklı aşamadan geçilmektedir. Bu süreçteki aşamalar Şekil 2.1'de görülebilir.



Şekil 2.1. Veri tabanlarında bilgi keşfi süreci (VTBK)

Bu süreçte ilk olarak ayıklama ve veri seçme işlemi yapılmaktadır. Bu işlemler kullanılabilecek olan verilerin kullanılamayacak olan verilerden ayrılması işlemidir. Örneğin özniteliklerinin değerleri bilinmeyen veya bir kısım bilgileri eksik olan veriler verimli olarak kullanılamazlar ve bunun bir sonucu olarak yapılan işlemlerden istenilen sonuçlar alınamaz. Bir sonraki aşama ön işleme ve dönüştürme aşamasıdır. Bu aşamalarda seçilmiş olan verilerden oluşan veri kümeleri üzerinde çeşitli ön işleme ve dönüştürme işlemleri yapılmaktadır. Bu işlemlerin yapılmasının sebebi veri kümesini tercih edilen algoritma doğrultusunda işlenebilir hale getirmektir. Örneğin seçilen işleme yöntemi sadece kategorik öznitelikler için çalışacak bir

algoritma ise ve elde bulunan veri kümesi numerik öznitelikler içeriyorsa gerekli işleme ve dönüştürme işlemleri yapılarak veri kümesi seçilen algoritma ile çalışabilecek hale getirilebilmektedir. Son olarak yapılan işlemler ise verinin işlenerek bilgi çıkarılması ve elde edilen bu bilginin yorumlanmasıdır. Bu şekilde veri içerisinden istenilen bilgiler elde edilerek çeşitli sonuçlara varılabilir ve bu doğrultuda yorumlar yapılabilir.

Veri madenciliği bilgisayar bilimleri içerisinde yer alan birçok farklı alandan yardım almaktadır. Yardım alınan bu alanlara yapay zeka, veritabanı sistemleri, istatistik gibi alanlar örnek olarak verilebilir. Verilerden bilgi çıkarma işleminin yapılmasının yanında bu işlemin verimli ve hızlı bir biçimde yapılması da büyük ölçüde önem taşımaktadır. Bu işlemlerin hızlı yapılabilmesi için de bilgisayar bilimlerinde yer alan çeşitli yöntemler ve algoritmalardan yardım alınmaktadır.

Verilerin içerisiden bilgi çıkarılma işlemleri de veri kümesi üzerinde öbekleme, sınıflandırma, ilişki çıkarımı yapma gibi işlemler yapılarak elde edilen sonuçlar yorumlanarak yapılabilmektedir. Bu işlemler içinde çeşitli öbekleme ve kümeleme algoritmaları kullanılmaktadır.

2.2. Öbekleme

Kümeleme problemi, elde bulunan n adet eleman içeren ve m adet özniteliğe sahip olan bir veri kümesini çeşitli algoritmalar ve elde bulununan bilgiler doğrulutusuda k adet alt kümeye ayırma işlemidir.

Bir veri kümesi üzerinde kümeleme işlemi yapılırken temel olarak 2 adet alt problemle karşılaşılmaktadır. Bu problemlerden ilki kümeleme işlemi yapılırken, veriler arasındaki farklılıkları ortaya koyan bir uzaklık metriğinin tanımlanmasının gerekliliğidir. Bu problem için önerilen çözüm büyük bir önem taşımaktadır. Bunun sebebi bu noktada önerilecek olan uzaklık metriğinin başarısı doğrudan sonucun başarısını etkileyecektir. Veriler arasındaki uzaklık hesaplanırken veri kümesinin içerisinde yer alan verilerin sahip oldukları öznitelikler kullanılmaktadır. Çözülmesi gereken ve büyük önem taşıyan diğer alt problem ise ortaya konulan algoritmanın

verimli bir şekilde çalışıyor olması gerekliliğidir. Bunun sebebi ise önerilen çözüm teorik olarak ne kadar başarılı olsa da, büyük ölçekli bir veri üzerinde uygulanamadığı takdirde anlamını yitirmesinden kaynaklanmadadır.

Veri kümesi içerisinde yer alan verilerin sahip olduğu öznitelikler de farklı türlerde bulunabilmektedirler. Öznitelikler ya numerik ya da kategorik olabilirler. Bu doğrultuda ortaya numerik, kategorik ve karışık olmak üzere 3 farklı tipte veri kümesi çıkabilir. Kullanılacak olan uzaklık metriğinin seçiminde veri kümesinin türü de büyük önem taşımaktadır. Örneğin, tamamı numerik özniteliklerden oluşan bir veri kümesi üzerinde uzaklık metriği olarak Öklid Uzaklığı (Euclidean Distance) kullanıldığında görece başarılı sonuçlar elde edilebilecekken kategorik ve karışık bir veri kümesi üzerinde bu uzaklık metriği uygulanmaya çalışılırsa, bu uzaklık metriğinin yapıya uygun olmadığı ve başarısız sonuçlar alındığı gözlemlenecektir.

Öbekleme algoritmaları genel olarak iki farklı kategori altında incelenmektedir.[3] Bu kategoriler şu şekildedir;

- Parçalı Öbekleme Algoritmaları (Partitional Clustering Algorithms)
- Hiyerarşik Öbekleme Algoritmaları (Hierarchical Clustering Algorithms)

2.2.1. Parçalı Öbekleme Algoritmaları

Parçalı öbekleme algoritmaları iteratif bir yapı üzerine kurulu algoritmalardır. Bu algoritmalarda elde bulunan veri kümesinin kaç ayrı alt kümeye ayrılacağı bilgisi, çalıştırılacak olan algoritmaya verilir ve algoritma bu doğrultuda çalıştırılır. Her iterasyonda belirlenen uzaklık metriği doğrultusunda toplam uzalık minimize edilmeye çalışılır. Örneğin k adet alt kümeye ayrılacak olan n elemanlı ve j numaralı alt kümenin merkezi C_j ile ifade edilen bir veri kümesi için toplam uzalık şu şekilde ifade edilmektedir.

$$\delta = \sum_{i=1}^{n} \| d_i - C_j \| \tag{2.1}$$

Algoritma belirlenen iterasyon sayısı tamamlandığında veya toplam uzaklık metriği belirli bir noktaya geldiğinde durmaktadır ve oluşturulan öbekleri geriye sonuç olarak döndürmektedir.

Parçalı öbekleme algoritmalarının en çok kullanılanlarına arasında k – means[4, 5] algoritması yer almaktadır. Bu algoritma bulunancak olan k adet öbek için ilk olarak rastgele bir şekilde öbeklere atadığı veriler üzerinden bir merkez noktası hesaplar ve daha sonra her iterasyonda öbekler içerisindeki elemanların değişmesi doğrultusunda bu merkez noktasını günceller. Bunun yanında bir diğer algoritma ise PAM'dir.[6] PAM, k – means'in aksine öbeklerin merkez noktalarını hesaplamak ve kullanmak yerine ilk öbekleri en iyi temsil eden, bir bakıma öbek içerisinde yer alan her bir veriye en yakın olan, veriyi temsilci olarak seçmektedir. Bu seçilen elemanlara medoid ismi verilmektedir. PAM daha sonra merkezleri güncellemek yerine medoidleri güncelleyerek en iyi sonuca ulaşmaya çalışmaktadır. PAM'den yola çıkılarak geliştirilen bir diğer algoritma ise CLARAN'dır.[7] Bu algoritmanın temel yapısı PAM'inki ile aynıdır. Fakat medoid'lerin seçimi işleminde bu iki algoritma farklılık göstermektedir. PAM ilk aşamada tüm elemanlar arasında rastgele bir şekilde medoidleri seçerken CLARAN ilk başta bir örnekleme yapmakta ve medoidleri bu örneklem içerisinde yer alan elemanlar arasından seçerek ilk aşamada daha doğru noktaları medoid olarak seçmeye çalışmaktadır. Tüm bu algoritmalar arasında k – Means oldukça kullanışlı bir algoritmayken PAM ve CLARAN çalışma zamanları açısından bakıldığında verimli ve kullanışlı algoritmalar değillerdir.

Bahsedilen algoritmaların yanında bir elemanın bir öbeğe ait olmasının zorunlu olmadığı farklı oranlarla farklı öbekler içerisinde yer alabileceği fuzzy c – means algoritması[8, 9] da parçalı öbekleme algoritmalarına bir örnektir. Bu algoritma da daha öncede bahsedildiği üzere bir veri 0 ile 1 arasında bir ağırlıkta birden fazla öbek içerisinde yer alabilmektedir.

2.2.2. Hiyerarşik Öbekleme Algoritmaları

Hiyerarşik öbekleme algoritmaları[10, 11] parçalı öbekleme algoritmalarının aksine iteratif bir yapı içerisinde değildir. Bu algoritmalarda bir ağaç yapısı oluşturularak

öbekleme işlemleri yapılmaya çalışılır. Ağacın oluşturulacağı yöne ve şekle göre bu algoritmalar aşağıdaki gibi iki temel yapıya ayrılmaktadır.

- Birleşen Öbekleme Algoritmaları (Agglomerative Clustering Algorithms)
- Ayrılan Öbekleme Algoritmaları (Divisive Clustering Algorithms)

2.2.2.1.Birleşen Öbekleme Algoritmaları

Birleşen yapıdaki öbekleme algoritmaları dipten tepeye doğru birleşerek ilerleyen bir yapıyı temel almışlardır. Bu algoritmalar ile öbekleme yapılırken her eleman başlı başına bir öbek olarak kabul edilir. Daha sonra her aşamada belirlenen uzaklık metriği doğrultusunda birbirlerine en yakın olan öbekler birleştirilerek daha büyük öbekler meydana getirilir. Bu işleme elde bulunan toplam öbek sayısı, istenilen k değerine düşünceye kadar devam edilir. Bu algoritmalarda parçalı algoritmalardaki gibi sürekli bir devam etme durumu yoktur. Algoritma en fazla (n - 1) defa, başka bir deyişle tüm elemanları aynı öbekte toplayana kadar çalışmasına devam edecektir ve sonra sonlanacaktır.

2.2.2.2.Ayrılan Öbekleme Algoritmaları

Ayrılan yapıdaki öbekleme algoritmaları ise tepeden dibe doğru ayrılarak ilerleyen bir yapıyı temel almışlardır. Bu algoritmalarda birleşen öbekleme algoritmalarının aksine ilk aşamada tüm elemanlar tek bir öbeğe ait kabul edilir ve daha sonra bu büyük öbek belirlenen uzaklık metriği doğrultusunda birbirlerinden en uzak olacak şekilde iki alt parçaya bölünür. Bu işleme elde istenilen k adet öbek elde edilinceye kadar devam edilir. Bu algoritmada da birleşen öbekleme algoritmalarına benzer olarak en fazla (n - 1) tekrar yapılmaktadır. Algoritma çalışabilecek olan en fazla tekrarda çalıştırıldığında elde her bir eleman bir öbek olarak kalacak ve algoritma sonlanacaktır.

Hiyerarşik öbekleme algoritmalarının en yaygın olanlarından biri, kategoriksel fayda (category utility)[12] temeline dayanan COBWEB[13] ve bu algoritmanın türevleri olan COBWEB/3[14], ECOWEB[15] ve ITERATE[16] alogritmlarıdır. Bu

algoritmalar farklı veri ikililerinin farklı öznitelik değer durumları için bir arada bulunma olasılıklarını en yüksek tutmaya, yani kategorisel faydalarını maksimize etmeye, çalışmakta ve bu doğrultuda bir arada bulunma ihtimali en yüksek olan verileri birleştirerek öbekleri büyütmektedir. Bunların yanında yine olasılıksal bilgi doğrultusunda öbekleme yapan AUTOCLASS[17] algoritması bulunmaktadır. Bu algoritma temelde Bayes Teoremi (Bayesian Probability)[18] üzerinden olasılıkları hesaplayıp bu doğrultuda öbekleri birleştirme işlemi yapmaktadır.

Sadece kategorik veriler için geliştirilmiş olan ROCK[19] algoritması bir diğer algoritmadır. Bu algoritma seçilen öbeğin sadece komşuları üzerinden bir benzerlik metriği üreterek öbekleme yapmaktadır. Bir diğer kategorik veri kümeleri için geliştirilmiş olan hiyerarşik öbekleme algoritması CACTUS[20] algoritmasıdır. Bu algoritmada benzerlikleri veriler arası değil öznitelikler arası bularak bir öbekleme işlemi yapmaya çalışmaktadır.

Genel olarak hiyerarşik öbekleme algoritmaları, parçalı öbekleme algoritmalarına göre daha az sayıda iterasyonda tamamlansa da söz konusu büyük veri kümeleri olduğunda hesaplanan uzaklık metrikleri ve hesap yapılması gereken eleman sayısının çokluğunda ötürü verimli çalışmamaktadırlar. Bu doğrultuda büyük veri kümelerinde hiyerarşik öbekleme yapabilmek adına BIRCH[21] isimli bir algoritma geliştirilmiştir. Bu algoritma ilk kez çalıştırıldığında bir ön öbekleme işlemi yapmaktadır ve daha sonra her çalıştırıldığında yaptığı bu ön öbekleme işleminin sonuçları üzerinden çalışmaktadır. Bu şekilde algoritmanın toplam çalışma hızı iyileştirilmeye çalışılmıştır.

2.3. Karışık Veri Kümeleri için Öbekleme Algoritmaları

Karışık veri kümeleri üzerinde öbekleme işlemi yapmak sadece numerik öznitelikler içeren ya da sadece kategorik özellikler içeren veri kümeleri üzerinde öbekleme işlemi yapmaya oranla daha zordur. Bunun sebebi kategorik öznilikler için seçilen uzaklık metriklerinin numerik öznitelikler üzerinde numerik öznitelikler için seçilen uzaklık metriklerinin ise kategorik öznitelikler üzerinde verimli bir şekilde

çalışmamasından kaynaklanmaktadır. Bu problemin önüne geçmek adına bir çok farklı yöntem önerilmiştir.

Önerilen bu yöntemlerden ilki kategorik olan özniteliklerin numerik özniteliklere çevrilip daha sonra sadece numerik uzaklık metrikleri kullanılarak öbekleme işlemi yapılmasıdır. Fakat bu çevrim işlemi her zaman uygun bir biçimde yapılamamaktadır. Bunun sebebi eğitim durumu veya gelir seviyesi gibi özniteliklerin farklı değerleri için numerik değerler atanabilir olup aralarında da bir uzaklık değeri tanımlanabilirken renk gibi özniteliklerin farklı değerleri için bu işlem yapılamamaktadır. Bu nedenle tüm kategorik öznitelikler doğru bir şekilde numerik özniteliklere çevrilememekte ve bunun bir sonucu olarak da öbekleme işlemi doğru bir şekilde yapılamamaktadır.

Önerilen bir diğer yöntem, numerik özniteliklerin değerlerinin kategorik öznitelik olacak şekilde güncellenmesi, yani bir ayrıklaştırma (discretization) işlemi yapılması ve daha sonra tamamen kategorik özniteliklere sahip olan yeni veri kümesinin kategorik veri kümelerine uygun bir uzaklık metriği kullanılarak öbekleme işlemin yapılması şeklindedir. Fakat numerik öznitelikleri kategorik özniteliklere çevirme işleminde de sıkıntılar oluşabilmektedir. Bu sıkıntılar çevrim işlemini yaparken hangi aralıklarla numerik değerlerin bölünmesi gerektiğinin bilinmemesinden kaynaklanmaktadır. Örneğin aralıklar çok büyük değerler ile bölündüyse birbirlerinden farklı olması gereken değerlerin aynı öznitelik değerini almasına ve bunun sonucunda bir bilgi kaybı olmasına neden olmakta veya aralıklar çok küçük değerler ile bölündüyse bu da bazı çok geniş aralığa yayılmış numerik öznitelikler için çok fazla sayıda kategorik öznitelik değerinin oluşmasına ve algoritmanın çalışma verimliliğinin düşmesine neden olmaktadır. Her iki durum oluştuğunda da algoritmanın çalışmasına olumsuz şekilde etki etmektedir.

Son olarak önerilen ve en başarılı şekilde çalışan yöntem ise Huang'ın Masraf Fonksiyonu (Huang's Cost Function)'dur.[22] Huang'ın Masraf Fonksiyonu kategorik öznitelikleri kendi aralarında kategorik bir uzaklık metriği ile değerlendirip bir uzaklık değeri hesaplamayı, numerik öznitelikleri kendi aralarında numerik bir uzaklık metriği ile değerlendirip ayrı bir uzaklık değeri hesaplamayı ve son olarak

bulduğu bu farklı numerik ve kategorik uzaklık değerlerinden toplam bir uzaklık değeri bulmayı ve bu toplam uzaklık değeri doğrultusunda öbekleme yapmayı önermektedir. Örneğin n elamanlı, m_r adet numerik m_c adet kategorik olmak üzere toplamda m adet öznitelik içeren d^r şeklinde bir numerik uzaklık metriğine sahip d^c şeklinde bir kategorik uzaklık metriğime sahip ve C^r şeklinde bir numerik merkeze sahip C^c şeklinde bir kategorik merkeze sahip veri kümesi için toplam uzaklık şu şekilde hesaplanmaktadır;

$$\delta = \sum_{i=1}^{n} \vartheta\left(d_i - C_j\right) \tag{2.2a}$$

$$\vartheta = \sum_{t=1}^{m_r}\left(d_{it}^r - C_{jt}^r\right) + \sum_{t=1}^{m_c}\left(d_{it}^c - C_{jt}^c\right) \tag{2.2b}$$

2.4. Genetik Algoritma

Genetik algoritma[23, 24], doğadaki evrimsel süreç izlenerek ortaya atılmış olan bir fikirdir. Burada evrimsel süreç ile anlatılmak istenen doğada her zaman en uyumlu olan yapının güçlenerek yaşamını sürdürmesi bunun aksine zayıf olan yapıların ise tamamen ortadan yok olması durumudur. Genetik algoritmanın çalışma yapısında da başarımı yüksek olan sonuçların bir sonraki nesile aktırılarak daha iyi sonuçlar elde edilmesi, başarımı düşük olanların ise silinerek dikkate alınmaması şeklinde bir işleyiş mevcuttur.

Genetik algoritmalar aynı zamanda tek bir sonuç üretmek yerine birden fazla sonuç, bir bakıma bir sonuç kümesi üretmektedir. Bunun sebebi bulunan birbirinden farklı sonuçların istenilen sonuca aynı oranda yakın olmasından kaynaklanmaktadır. Bu durum algoritmayı kullanan kişiye de sonuçlar açısından bir çeşitlilik sunmaktadır. Bu çeşitliliğin oluşmasının sebebi, genetik algoritma içerisinde yapılan çaprazlama[25] ve mutasyon işlemleridir. Bu işlemler biyolojide, DNA'ların birbiri ile çaprazlanması sonucu ortaya iki DNA'nın ortak ürünü olan yeni DNA'ların çıkması ve bir DNA'nın mutasyona uğraması sonucu yeni bir özellik kazanması

mantığı temel alınarak yapılmıştır. Genetik algoritmada da iki başarılı sonuçdan yeni bir sonuç üretilirse, yeni üretilen sonucun atalarından daha iyi bir sonuç olacağı ve bir sonuç üzerinde mutasyon olursa sonucun daha iyi bir noktaya gelebileceği kabul edilmiştir.

Genetik algoritmalar bir sonucun başarımını ölçerken, kullanıcı tarafından tanımlanmış olan fonksiyonlar doğrultusunda bu işlemi yapmaktadırlar. Bu fonksiyonlara uygunluk fonksiyonu (fitness function) ismi verilmektedir. Genetik algoritma bulduğu sonucun uygunluk fonksiyonundan aldığı değer doğrultusunda bir sonraki nesilde o sonucun kalmasının ya da silinmesinin gerekliliğine karar vermektedir. Bu nedenle uygunluk fonksiyonun seçilmesi işlemi genetik algoritmanın iyi sonuçlar vermesi açısından en önemli noktadır.

Genetik algoritmalar hiçbir zaman için bir problemin en iyi çözümünü vermeyi garanti etmezler. Ancak diğer algoritmların çalışma masrafının çok yükseldiği ve sonuçların alınamadığı, örneğin arama uzayının çok büyük ve karmaşık olduğu, elde bulunan bilgiyle matematiksel çözümün bulunamadığı veya problemin belirli bir matematiksel model ile ifade edilemediği problemlerde, genetik algoritma oldukça etkili ve hızlı bir biçimde çalışmakta ve en iyi sonuç olmasa bile iyiye yakınsayan bir sonucu kullanıcıya döndürmektedir. Bu özelliği de genetik algoritmanın en çok tercih edilmesine neden olan özelliklerinden biridir.

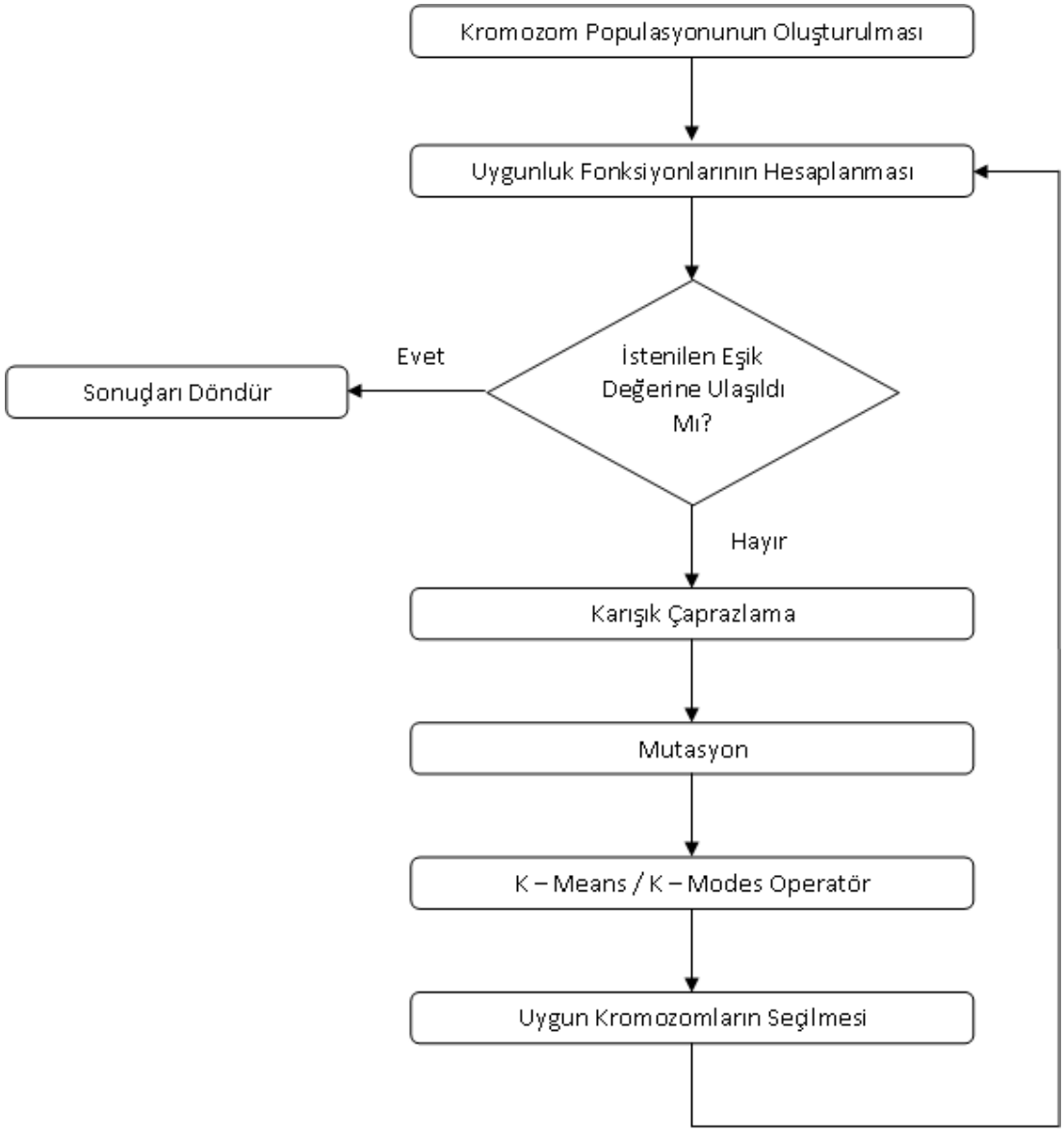
BÖLÜM 3

3. GENETİK ALGORİTMA YARDIMIYLA ÇOK AMAÇLI ÖBEKLEME

3.1. Genel Bakış

Veri kümelerinin genetik algoritma yardımıyla çok amaçlı bir biçimde öbeklenmesi geleneksel genetik alogritmanın çalışma yapısına oldukça benzemektedir. Algoritmanın genel çalışma yapısı Şekil 3.1'de görülebilir. Algoritma da ilk olarak kaç adet kromozomun bulunacağının tanımlanması gerekmektedir. Verilen bu kromozom sayısı algoritmanın sahip olduğu popülasyonu simgelemektedir. Kromozom sayısını gösteren bu değer P ile ifade edimiştir. Daha sonra oluşturulan P adet kromozom belirlenen uygunluk fonksiyonları doğrultusunda, karışık çaprazlama (shuffle crossover) adı verilen bir çaprazlama yöntemi ve mutasyon işlemlerinden geçirilerek yeni nesiller ortaya çıkarılmaktadır. Uygunluk fonksiyonları her bir neslin elemanları için çalıştırılmaktadır ve uygunluk fonksiyonunun verdiği sonuç doğrultusunda bu elemanların ait oldukları öbekler tespit edilmeye çalışılmaktadır. Her bir eleman bu uygunluk fonksiyonunun sonucuna göre kendine en yakın olan öbek ile eşleştirilmektedir. Ancak önerilen yöntem karışık veri kümeleri için olduğundan tek bir uygunluk fonksiyonu bulunmamakta; numerik öznitelikler için ayrı kategorik öznitelikler için aynı uygunluk fonksiyonları algoritma içerisinde yer almaktadır.

Her iterasyon sonucunda toplamda 2 x P adet kromozom içeren sonuç olarak geriye dönmektedir. Geriye dönen bu sonucun içerisinde her bir kromozomun elemanlarının, tanımlanan uygunluk fonksiyonlarının çalıştırılması sonucunda elde edilen toplam değerleri de tutulmaktadır. Bu noktada yapılması gereken 2 x P büyüklüğündeki sonuçlar arasında en iyi olan P adet sonucun seçilmesidir. Bu işlem de NSGA (Non – Dominated Sorting Algorithm) (Domine Edilmeyeni Sıralama Algoritması) adı verilen bir sıralama algoritması yardımıyla yapılmaktadır. Bu sıralama algoritması temel olarak baskınlık ilkesi doğrultusunda çalışmaktadır.



Şekil 3.1. Genetik algoritma yardımıyla karışık veri kümeleri için çok amaçlı öbekleme algoritmasının akış şeması

NSGA'ya göre işlem sonunda iki vektör elde edilmektedir. Bu vektörlerden ilki y vektörüdür. y vektörü içerisinde farklı uygunluk fonksiyonlarının sonuçlarını tutmaktadır. Diğer vektör ise e(x) vektörüdür. Bu vektör de tanımlanan kısıtları içerisinde tutmaktadır. Bu vektörler sayesinde problemin bulunduğu alanı ve çeşitliliği ifade edebiliriz. Daha sonraki aşamada NSGA pareto derecelendirme sistemini[26, 27] kullanarak elde edilecek olası sonuçları seçmemize olanak sağlamaktadır. Örneğin elimizde y_1 ve y_2 olmak üzere iki sonucumuz var ise bu iki

sonuç NSGA yardımı ile karşılaştırılır. Algoritma sonucunda y_1 sonucunun y_2 sonucunu domine ettiği gözlemlenirse y_1, y_2'nin üstünde bir pareto katmanında yer alır. Eğer algoritma sonuca göre y_2 sonucunun y_1 sonucunu domine ettiği gözlemlenirse y_2, y_1'in üstünde bir pareto katmanında yer alır. Eğer iki sonuç birbirlerine üstünlük sağlayamıyorsa yani domine eden bir taraf yoksa iki sonuçta aynı pareto katmanında yer almaktadır. Matematiksel olarak ifade etmemiz gerekirse;

$x = (x_1, x_2, \dots, x_n) \in X$ ve

$y = (y_1, y_2, \dots, y_k) \in Y$ iken

$y = f(x) = \big(f_1(x), f_2(x), \dots, f_k(x)\big)$ için

$Max \,||\, Min\big(e(x) = \big(e_1(x), e_2(x), \dots, e_m(x)\big) \geq 0\big)$

Bu işlemler kullanıcı tarafından belirlenen iterasyon sayısı kadar tekrar edilmektedir. Bir diğer seçenek ise uygunluk fonksiyonlarının sonuçlarında istenen düzeyde bir gelişme olmadığı zamana kadar algoritmanın çalışmaya devam etmesi şeklinde olabilmektedir. Her iki yöntem de kullanılabilir olsa da ikinci yöntem algoritmanın bir karar mekanizmasına sahip olması ve rastgele bir iterasyon sayısında değil çalışması gerektiği kadar çalışmasını sağladığı için daha verimli bir yöntemdir.

Algoritma, çalışması tamamlandıktan sonra geriye n x n'lik bir M matrisi döndürmektedir. Burada n ile ifade edilen veri kümesi içerisinde yer alan elemanların sayısıdır. Sonuç olarak geriye dönen matris Şekil 3.2'deki şekildedir.

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} \\ x_{21} & x_{22} & x_{23} & x_{24} & x_{25} \\ x_{31} & x_{32} & x_{33} & x_{34} & x_{35} \\ x_{41} & x_{42} & x_{43} & x_{44} & x_{45} \\ x_{51} & x_{52} & x_{53} & x_{54} & x_{55} \end{bmatrix}$$

Şekil 3.2 Önerilen algoritma sonucunda ortaya çıkan örnek bir matris

Bu matris içerisinde her bir x_{ij} değeri veri kümesi içerisinde yer alan i numaralı eleman ile j numaralı elemanın, genetik algoritmanın sonunda üretilen en kuvvetli P adet jenerasyonun içerisinde kaç defa bir arada bulunduğunu gösterir. Bir başka deyişle i ve j numaralı elemanların kaç sonuç içerisinde aynı öbek içerisinde yer aldığını göstermektedir. Bu matris sayesinde farklı sonuçlar içerisinde farklı elemanların birbirleri aralarında ilişki gözetlenebilmektedir. Elde edilen sonuçlar çok amaçlı bir öbekleme işleminden[36, 37] geçtiği için farklı uygunluk fonksiyonlarından elde edilen sonuçlar doğrultusunda oluşturulan bu matris ile iki eleman arasında bir bağ olup olmadığı, eğer bir bağ var ise bu bağın ne kadar kuvvetli olduğu saptanabilmektedir.

3.2. Kromozomların Kodlanması

Algoritma ilk çalışmaya başladığında P adet kromozomun oluşturulması gerekmektedir. Bu kromozomlar oluşturulurken ilk aşamada tamamen rastgele değerler atanmaktadır. Her kromozom içerisinde veri kümesinde bulunan elemanların hangi öbeğe ait olduğu bilgisini tutmaktadır. Örneğin i numaralı eleman 3 numaralı öbeğe aitse kromozom içerisinde tutulan dizide i'nin ait olduğu alanda 3 değeri yer almaktadır. Ancak yapılan çalışmada bu değerler onluk tabandaki sayı sisteminde değil ikilik tabandaki sayı sisteminde (bitwise) tutulmaktadır. Bir kromozom dizisinin görünümü Şekil 3.3'de görülebilir. Kromozom dizisi içerisinde n adet eleman içeren bir D veri kümesinin elemanlarının ait olduğu öbekler, her bir eleman dizi içerisinde b adet bit ayrılarak gösterilmiştir.

1	**0**	**0**	1	1	1	…	…	…	**1**	**0**	**1**

Şekil 3.3. Her elamanın gösterimi için 3 bit ayrılan bir kromozom dizisinin gösterimi

Her bir eleman için ayrılmış olan bu b adet bit sayısı aynı zamanda veri kümesinin bölündüğü toplam alt küme sayısını da göstermektedir.Bu b değeri algoritma tarafından otomatik olarak hesaplanmaktadır. Bu sayı kullanıcı tarafından da sisteme verilebilmektedir ancak bu durumda program kısıtlanmış olacaktır. Bu nedenle b değerinin algoritma tarafından hesaplanması tercih edilmektedir. Bu hesap veri

kümemiz içerisinde yer alan toplam eleman sayısı olan n değerinin karekök değerinin ikilik tabanda gösterildiği toplam bit sayısı ile ifade edilebilecek en büyük değerin seçilmesi şeklinde yapılmaktadır. Örneğin, elimizde D = x_1, x_2, ..., x_n, şeklinde bir veri kümesi var ise;

$$b = \log(\lceil\sqrt{n}\rceil) \tag{3.1}$$

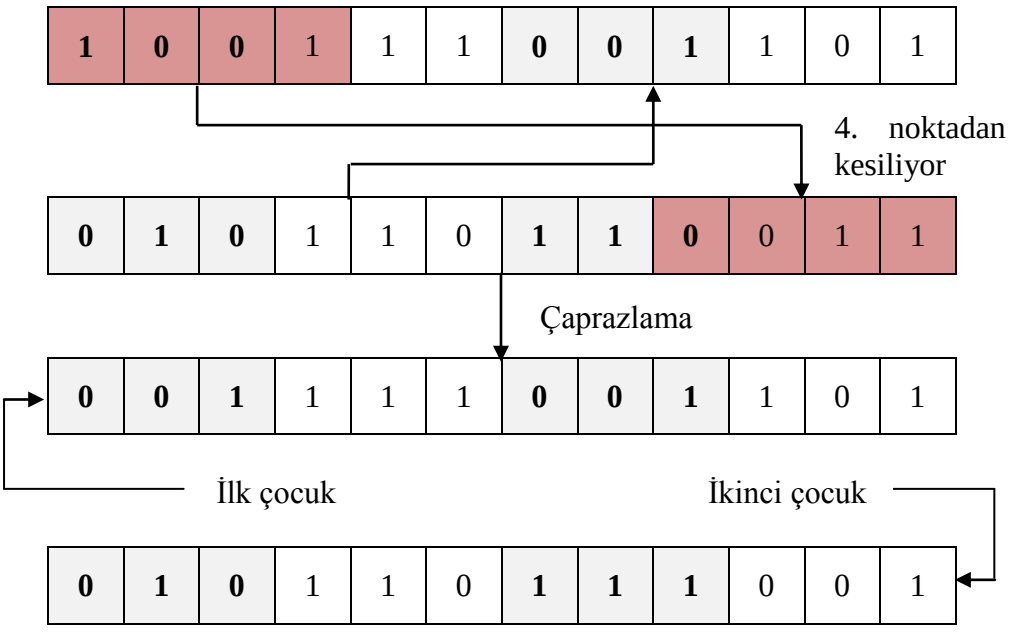
Bu noktada en büyük değerin seçilmesinin sebebi, işlemler bit bazında yapıldığı için daha sonra yapılacak olan çaprazlama ve mutasyon işlemlerinde sınır dışına çıkma ihtimalinin önüne geçilmesine adınadır. Örneğin 100 elemanı bulununa bir veri kümesi için yapılan işlemler sonucunda 4 bit ile gösterim yapılabileceği sonucuna varılır. Eğer burada kullanılacak olan en büyük değer 1011 olursa daha sonra herhangi bir işlem sonucunda 1111 gibi bir değer oluşması sonucunda sıkıntı oluşacaktır.

3.3. Karışık Çaprazlama (Shuffle Crossover)

Algoritma çalışırken sahip olunan kromozomlar kullanılarak yeni kromozomlar üretilmektedir. Algoritmanın başında sahip olunan kromozom sayısının P'den her iterasyon sonunda 2P'ye çıkmasının sebebi de budur. Yeni kromozomların oluşmasına çaprazlama işlemi neden olmaktadır. Çaprazlama işlemi, DNA'ların birbirleri ile çaprazlanması sonucu yeni DNA'ların ortaya çıkması örnek alınarak modellenmiştir. Bu işlem yapılırken başarımı yüksek olan kromozomların birbirleri ile çaprazlanması sonucu ortaya çıkacak olan yeni kromozomların da başarımlarının yüksek olacağı ve daha kuvvetli olacakları kabul edilmektedir. İki kromozom için yapılan olağan bir çaprazlama örneği Şekil 3.4'de görülebilir.

Bizim önerdiğimiz çaprazlama yöntemi genetik algoritmanın içerisinde yer alan olağan çaprazlama yöntemlerine göre farklılıklar göstermektedir. Normalde genetik algoritma içerisinde yapılan çaprazlama işleminde çaprazlama işlemi sırasında kromozomların sahip oldukları dizi üzerinde çaprazlama işleminin yapılacağı nokta dizinin tam ortası olarak belirlenmektedir veya buna bir alternatif olarak dizinin üzerindeki bir nokta kullanıcı tarafından algoritma çalıştırılmadan önce

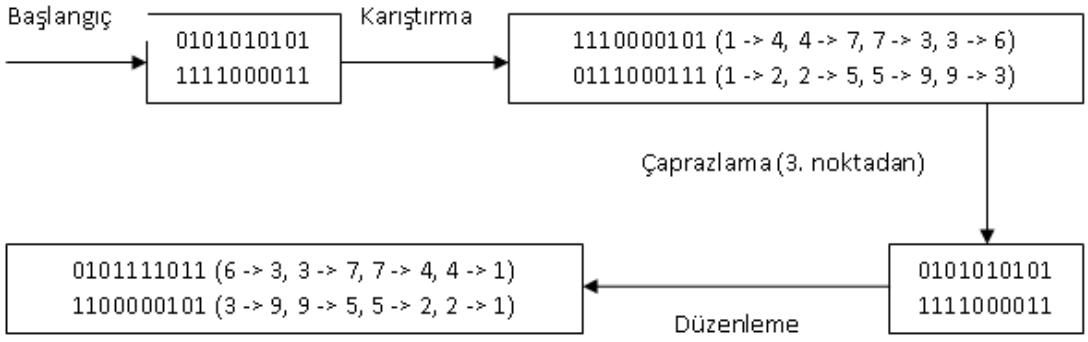
seçilmektedir. Daha sonra bu çaprazlama işlemi daima bu nokta üzerinden yapılmaktadır. Bunun bir sonucu olarak da birbirleri ile benzer kromozoların eşleşmesi sonucu yeni nesilde ortaya çıkan kromozomlar birbirleri ile neredeyse aynı olması gibi bir problem çıkmaktadır. Kromozomların birbirleri ile aynı olması ya da yüksek oranda benzeşiyor olması çeşitliliği düşürmekte ve yeni sonuçların ortaya çıkmasına engel olmaktadır. Bu durumla karşılaşıldığında yeni sonuçlar elde edilemediği için algoritmanın en iyi noktaya mı ulaştığı yoksa daha iyi sonuçlar etme ihtimalin olması durumu bilinememektedir. Bu nedenle önerilen algoritma içerisinde çaprazlama işlemi yapılırken geleneksel metotlardan farklı bir metot önerilmiştir. Önerilen metot doğrultusunda çaprazlama işlemi yapılırken kromozomların sahip oldukları diziler üzerindeki kesim noktaları rastgele bir biçimde belirlenmektedir. Bu yapılan değişiklik sonucunda yeni nesillerde ortaya çıkan kromozomlar arasındaki farklarda büyük bir değişim ve çeşitliliğin arttığı gözlenmektedir.



Şekil 3.4. İki kromozom için olağan çaprazlama örneği

Çeşitliliğin arttıralabilmesi adına çaprazlama işlemine yapılan bir diğer ek ise kromozomların dizileri üzerinde yapılan karıştırma işlemidir. Karıştırma işlemi kromozomlar birbirleri ile çaprazlanmadan önce yapılmakta ve temel olarak ikilik sayı sisteminde tutulan dizilerin içerisinde yer alan bitlerinin kendi içersinde yer

değiştirmesi şeklinde ifade edilebilmektedir. Bu işlem başarımı yüksek olan bir kromozomun içerdiği dizi üzerinde küçük bir kaç yer değiştirme işlemi yapılarak daha kuvvetli bir kromozom üretebilme olasılığı varsayılarak yapılmaktadır. Aynı zamanda bu işlem daha önce oluşan kromozomlardan farklı kromozomlar oluşmasına da olanak sağlamaktadır. Karıştırma işlemi yapılırken kromozomun dizisi içerisinde yer değiştiren bitlerin eski ve yeni yerleri ayrı bir dizi içerisinde tutulmaktadır. Karıştırma işleminin ardından kromozomlar arasında çaprazlama işlemi yapılmakta ve ortaya çıkan yeni kromozomlar üzerinde de düzenleme işlemi yapılmaktadır. Düzenleme işlemi, karıştırma işleminin tam olarak zıttı olan bir işlemdir. Bu işlem sonucunda amaçlanan kromozomların dizileri içerisinde yer alan değerlerin karıştırma işlemi öncesindeki bozulmamış durumlarına geri getirilmesidir. Düzenleme işlemi yapılırken karıştırma işlemi sırasında oluşturulan ve yer değiştiren bitlerin eski ve yeni yerlerini içeren diziden yararlanılmaktadır. Bu dizi aracılığıyla yer değiştiren elemanlar tespit edilmekte ve bulunmaları gereken yerlere tekrardan geri alınmaktadırlar. İki kromozom için karışık çaprazlama sürecinin örnek bir işleyişi Şekil 3.5'de görülebilir.



Şekil 3.5. İki kromozom için karışık çaprazlama örneği

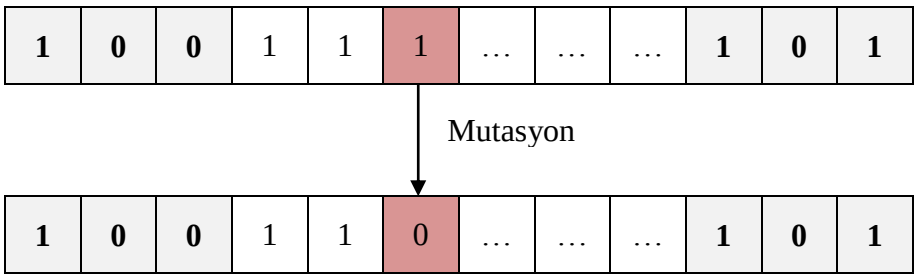
Karışık çaprazlama işlemi her iterasyonda elde bulunan her bir kromozom ikilisi için yapılmaktadır. Çaprazlanacak olan kromozom ikilileri tamamen rastgele bir şekilde belirlenmektedir ve her bir iterasyon sırasında bir kromozom sadece tek bir defa çaprazlama işlemi sokulmaktadır. İki kromozomun çaprazlanması işlemi sonucunda ortaya iki adet yeni nesil kromozom çıkmaktadır.

Önerilen bu yeni çaprazlama yöntemi sayesinde elde edilen yeni nesil kromozomlar geleneksel çaprazlama yöntemlerinin sonucunda üretilen kromozomlara oranla çok daha çeşitlilik göstermektedir. Bu da algoritmanın daha verimli ve iyi sonuçlar verecek biçimde çalışmasına olanak sağlamaktadır.

3.4. Mutasyon

Mutasyon işlemi de çeşitliliği arttırmak adına yapılan bir işlemdir. Mutasyon, kromozomların sahip oldukları dizilerin elemanları üzerinde oluşabilmektedir. Her kromozom üzerinde mutasyon işlemi meydana gelmek zorunda değildir. Kromozomun mutasyona uğrayıp uğramayacağı kullanıcının algoritmada belirleyeceği eşiğe bağlıdır. Bu eşik değeri ne kadar düşük tutulursa kromozom üzerinde de mutasyon olma olasılığı o kadar artmaktadır.

Mutasyon işlemi temel olarak kromozom içerisinde yer alan dizideki bir bit değerinin değişmesidir. Bu değişim eğer bitin değeri 0 ise 1'e dönüşmesi, 1 ise 0'a dönüşmesi şeklinde olmaktadır. Dizi içerisinde yer alan her bir bitin mutasyona uğrama ihtimali vardır ve birden fazla bit bir iterasyon içersinde mutasyona uğrayabilir. Örnek bir mutasyon işlemi Şekil 3.6'de görülebilir.



Şekil 3.6. Mutasyon işlemi

3.5. Kullanılan Uygunluk Fonksiyonları

Genetik algoritmanın işleyişi sırasında her iterasyonda her bir kromozomun elemanları için uygunluk fonksiyonları çalıştırılmalı ve kromozomların başarımı hakkında bir sonuca varılmalıdır. Uygunluk fonksiyonları iki eleman arasındaki

benzerliği veya uzaklığı verecek olan metrikler olarak da düşünülebilir. Yapılan çalışmada bu mesafeleri ölçebilmek adına kategorik özellikler üzerinden 4 adet, numerik özellikler üzerinden 3 adet ve hem kategorik hem de numerik öznitelikler üzerinden 1 adet olmak üzere toplamda 8 adet uygunluk fonksiyonu tanımlanmıştır. Bu fonksiyonlar sırası ile k – modes içsel uzaklık, k – modes dışsal uzaklık, k – means içsel uzaklık, k – means dışsal uzaklık, kategorik öznitelikler için önerilen olasılıksal bir yöntem, numerik öznitelikler için önerilen ağırlıklı bir yöntem, kategorik öznitelikler için önerilen ağırlıklı bir yöntem ve hem numerik hem de kategorik öznitelikler için önerilen ağırlıklı bir yöntemdir.

Bu uzaklıkların hesaplandığı D veri kümesi D = $\{x_1, x_2 \ldots x_n\}$ şeklinde ifade edilmektedir ve x_i değeri veri kümesi içerisindeki i numaralı elemanı göstermektedir. Veri kümesi içerisinde m_r adet numerik öznitelik, m_c adet kategorik öznitelik olmak üzere toplamda m adet öznitelik yer almaktadır. Öznitelik kümesi içerisinde numerik öznitelikler $A^r = \{A_1^r, A_2^r \ldots A_{m_r}^r\}$ ve kategorik öznitelikler $A^c = \{A_1^c, A_2^c \ldots A_{m_c}^c\}$ şeklinde ifade edilmektedir. Tüm öznitelik kümesi ise $A = A^r \cup A^c = \{A_1, A_2 \ldots A_m\}$ şeklinde ifade edilmektedir. A_j^r değeri numerik öznitelikler kümesi içerisindeki j numaralı değeri, A_j^c değeri ise kategorik öznitelikler kümesi içerisindeki j numaralı değeri göstermektedir. Her bir A_j değeri için bu değerin içerisinden seçildiği bir değer kümesi bulunmaktadır. Bu küme tüm öznitelikler için $V = V_1 \cup V_2 \cup \ldots V_m$ şeklinde ve j numaralı öznitelik için $V_j = \{V_{j1}, V_{j2}, \ldots, V_{js}\}$ şeklinde gösterilmektedir. Burada V_{jk}, j numaralı öznitelik için verilen değer kümesi içerisinde yer alan k numaralı elemanı temsil etmektedir.

3.5.1. K – Modes İçsel Uzaklık

K – Modes içsel uzaklık[28, 29] kategorik öznitelikler için kullanılan bir uygunluk fonksiyonudur. Bu uygunluk fonksiyonu her bir kromozom için D veri kümesi içerisindeki tüm $x_i - y_i$ ikilileri üzerinden hesaplanmaktadır. Hesaplama işlemi iki veri kümesi elemanının tüm kategorik öznitelik değerlerinin karşılaştırılması doğrultusunda yapılmaktadır. Örneğin sahip olunan bir A_j^r kategorik özniteliği için bu iki eleman aynı değere sahip ise bu öznitelik doğrultusunda aralarında bir uzaklık

yoktur aksi durumda özniteliğin değerleri farklı ise aralarında bir uzaklık vardır. Bu durumu formüle edicek olursak;

$$d(X,Y) = \sum_{i=1}^{m} \delta(x_i, y_i) \qquad (3.2a)$$

$$\delta(x_i, y_i) = \begin{cases} 0 & if\ x_i = y_i \\ 1 & if\ x_i \neq y_i \end{cases} \qquad (3.2b)$$

Bu hesaplama işlemi $x_i - y_i$ ikililerinin sahip olduğu tüm kategorik öznitelikler için yapılmaktadır ve sonuç olarak bu iki elaman arasındaki toplam içsel uzaklık değeri bulunmaktadır. Önerilen algoritmada k – modes içsel uzaklık uygunluk fonksiyonu kullanılırken amaçlanan, veri kümesi üzerinden oluşturulan öbeklerin sahip oldukları elemanların kendi aralarında oluşturdukları tüm ikililerden elde edilen toplam uzaklık değerini her iterasyonda azaltmaya çalışmaktır. Bir başka deyişle elde edilen toplam uzaklık değerinin minimize edilmesi amaçlanmaktadır. Bunun sebebi aynı öbek içerisinde yer alan elemanlar arasındaki toplam mesafenin en aza indirilmek istenmesidir. Bir öbek içerisindeki elemanlar birbirlerine ne kadar yakın olursa bu o elemanların birbirlerine o kadar benzedikleri anlamına gelmektedir ve çalışmada istenen aynı öbek içerisinde yer alan elemanların birbirleri ile olan benzerliklerinin yüksek olmasıdır.

3.5.2. K – Modes Dışsal Uzaklık

K – Modes dışsal uzaklık uygunluk fonksiyonu da kategorik öznitelikler için kullanılan bir uygunluk fonksiyonudur. Bu fonksiyon çalışma şekli olarak k – modes içsel uzalık uygunluk fonksiyonuna oldukça benzemektedir. Bu uygunluk fonksiyonunda da aynı k – modes içsel uzaklık uygunluk fonksiyonunda olduğu gibi her bir kromozom için D veri kümesi içerisindeki tüm $x_i - y_i$ ikilileri üzerinden uzaklık değerleri hesaplanmaktadır. Hesaplama işlemi aynı k – modes içsel uzaklık uygunluk fonksiyonunda olduğu gibi iki veri kümesi elemanının tüm kategorik öznitelik değerlerinin karşılaştırılması doğrultusunda yapılmaktadır.

Bu uygunluk fonksiyonunun hesaplanma işlemi de $x_i - y_i$ ikililerinin sahip olduğu tüm kategorik öznitelikler için yapılmaktadır ve iki eleman arasındaki toplam uzaklık değeri bulunmaktadır. Ancak bu uygunluk fonksiyonu ile hesaplanmak istenen bir önceki yöntemdekinin aksine içsel değil dışsal toplam uzaklıktır. Burada dışsal uzaklık ile anlatılmak istenen, içsel uzaklıktaki gibi her öbeğin kendi içerisinde yer alan elemanlarının kendi aralarındaki toplam uzaklığın bulunması değil, veri kümesi içerisinde yer alan elemanların kendi bulundukları öbek dışındaki elemanlar ile olan toplam uzaklıklarının hesaplanmasıdır. Bu durumda k – modes dışsal uzaklık uygunluk fonsiyonunun değeri, k – modes içsel uzaklık uygunluk fonksiyonunun değerinin aksine minimize değil maksimize edilmeye çalışılmaktadır. Bunun sebebi öbekleri mümkün olduğunca birbirine yakın ve diğer öbekler içerisindeki elemanlara uzak olacak şekilde oluşturmak, birbilerine en yakın olan elemanları aynı öbekler içerisinde toplama isteğidir.

3.5.3. K – Means İçsel Uzaklık

K – Means içsel uzaklık uygunluk fonksiyonu numerik öznitelikler için kullanılan bir uygunluk fonksiyonudur. Bu uygunluk fonksiyonu veri kümesi içerisinde yer alan elemanların sahip olduğu özniteliklerin sayısal değerleri kullanılarak hesaplanmaktadır. Bu uygunluk fonksiyonu da aynı k – modes içsel uzaklık uygunluk fonksiyonunda olduğu gibi bir D veri kümesi içerisindeki tüm $x_i - y_i$ ikilileri üzerinden hesaplanmaktadır. Bu iki eleman arasındaki uzaklık bulunurken Öklid Uzaklığı (Euclidean Distance) kullanılmaktadır. Hesaplanan uzaklık aşağıdaki gibi formülize edilmiştir;

$$d(X,Y) = \sqrt{\sum_{i=1}^{m} \delta(x_i, y_i)} \qquad (3.3a)$$

$$\delta(x_i, y_i) = (x_i - y_i)^2 \qquad (3.3b)$$

Bu uygunluk fonksiyonu temelde $x_i - y_i$ ikililerinin sahip olduğu tüm numerik özniteliklerin değerlerinin bir nokta olarak kabul edilip daha sonra bu noktalar

arasındaki toplam öklid uzaklığının bulunması işlemi şeklinde düşünülebilir. Önerilen bu yöntem de amaçlanan aynı k – modes içsel uzaklık uygunluk fonksiyonunda olduğu gibi veri kümesi üzerinden oluşturulan öbeklerin sahip oldukları elemanların kendi aralarında oluşturdukları tüm ikililerden elde edilen toplam uzaklık değerini her iterasyonda azaltmaya çalışmaktır. Yani bu uygunluk fonksiyonunda da amaçlanan toplamda elde edilen değerin minimize edilmesidir. Böylelikle aynı öbek içerisinde yer alan elemanların birbirlerine mümkün oldukça yakın olması sağlanmaya çalışılmaktadır.

3.5.4. K – Means Dışsal Uzaklık

K – Means dışsal uzaklık uygunluk fonksiyonu da numerik öznitelikler için kullanılan bir uygunluk fonksiyonudur. Bu uygunluk fonksiyonu da aynı k – means içsel uzaklık fonksiyonu gibi veri kümesi içerisinde yer alan elemanların sahip oldukları numerik özniteliklerin değerleri kullanarak, her bir kromozom için D veri kümesi içerisindeki tüm x_i – y_i ikilileri üzerinden uzaklık değerleri hesaplanmaktadır. Hesaplama işlemi aynı k – means içsel uzaklık uygunluk fonksiyonunda olduğu gibi Öklid Uzaklığı (Euclidean Distance) doğrultusunda yapılmaktadır.

Bu uygunluk fonksiyonunun hesaplanma işlemi de x_i – y_i ikililerinin sahip olduğu tüm numerik öznitelikler için yapılmaktadır ve iki eleman arasındaki toplam uzaklık değeri bulunmaktadır. Ancak bu uygunluk fonksiyonu ile hesaplanmak istenen bir önceki yöntemdekinin aksine içsel değil dışsal toplam uzaklıktır. Burada dışsal uzaklık ile anlatılmak istenen, k – modes dışsal uzaklık uygunluk fonksiyonunda olduğu gibi, içsel uzaklıktaki gibi her öbeğin kendi içerisinde yer alan elemanlarının kendi aralarındaki toplam uzaklığın bulunması değil, veri kümesi içerisinde yer alan elemanların kendi bulundukları öbek dışındaki elemanlar ile olan toplam uzaklıklarının hesaplanmasıdır. Bu durumda k – means dışsal uzaklık uygunluk fonsiyonunun değeri, k – means içsel uzaklık uygunluk fonksiyonunun değerinin aksine minimize değil maksimize edilmeye çalışılmaktadır. Bunun sebebi öbekleri mümkün olduğunca birbirine yakın ve diğer öbekler içerisindeki elemanlara uzak

olacak şekilde oluşturmak, birbilerine en yakın olan elemanları aynı öbekler içerisinde toplama isteğidir.

3.5.5. Kategorik Öznitelikler İçin Önerilen Olasılıksal Bir Yöntem

Daha önce de bahsedildiği üzere kategorik öznitelikler arasında tam olarak istenen şekilde çalışan bir uzaklık metriği tanımlanamamaktadır. Ancak en çok kullanılan uzaklık hesaplama şekli k – modes içsel ve k – modes dışsal uygunluk fonksiyonlarının hesaplanmasında kullanılan yöntemdir.[30] Bu yöntemden bir örnek üzerinden tekrardan bahsetmemiz gerekirse, elimizde filmlerden oluşan bir veri kümemiz olduğunu düşünelim. Bu veri kümese filmlerinin ad bilgilerinin yanında yönetmen bilgisini, aktör bilgisini ve tür bilgisini de birer kategorik öznitelik olarak tutmaktadır. Bu veri setinden bir parça örnek olarak Tablo 3.1'de görülebilir.

	Yönetmen	Aktör	Tür	Öbek
$\boldsymbol{t_1}$(Godfather II)	Scorsese	De Niro	Polisiye	$\boldsymbol{c_1}$
$\boldsymbol{t_2}$(Goodfellas)	Coppola	De Niro	Polisiye	$\boldsymbol{c_1}$
$\boldsymbol{t_3}$(Vertigo)	Hitchcock	Stewart	Gerillim	$\boldsymbol{c_2}$
$\boldsymbol{t_4}$(N by NW)	Hitchcock	Grant	Gerilim	$\boldsymbol{c_2}$
$\boldsymbol{t_5}$(Bishop's Wife)	Koster	Grant	Komedi	$\boldsymbol{c_2}$
$\boldsymbol{t_6}$(Harvey)	Koster	Stewart	Komedi	$\boldsymbol{c_2}$

Tablo 3.1. Örnek sinema veri kümesi

Bu tabloda yer alan c değeri ise öbekleme sonucu veri kümesi içerisinde yer alan elemanların hangi öbeğe ait olduklarını göstermektedir. Elimizdeki örnekte c_1 ve c_2 olmak üzere iki farklı öbek bulunmaktadır. Bu öbekleme işlemi yapılırken bahsedilen uzaklık metriği kullanılırsa basit olarak elemanların öznitelikleri arasındaki farklar bir uzaklık olarak geriye dönecektir. Örneğin t_4 ve t_5 arasındaki uzaklığı hesaplayacak olursak yönetmen bilgilerine bakıldığında birinin yönetmeninin Hitchcok diğerinin ise Koster olduğunu görülmektedir. Bu durumda bu öznitelik doğrultusunda aralarında bir uzaklık vardır. Bir sonraki öznitelik olan aktör bilgisinde ikisin de değerinin Grant olduğunu görülmektedir. Buna göre aktör bilgisi özniteliği doğrultusunda aralarında bir uzaklık yoktur. Son olarak ise tür özniteliğinin

değerine bakıldığında ilk filmin türünün Gerilim ikinci filmin ise türünün Komedi olduğunu görülmektedir. Buna göre tür özniteliği için de aralarında bir uzaklık vardır. Bütün bu öznitelikler göz önüne alındığında ve değerleri toplandığında t_4 ve t_5 arasındaki toplam uzaklığı 2 olarak bulunmaktadır.

Bahsedilen uzaklık hesaplama yöntemi oldukça sık bir şekilde kullanılan bir yöntem olsa da bazı durumlar için tam olarak mantıklı bir tabana oturmamaktadır. Örneğin değer kümesi 'genç, orta yaş, yaşlı' değerlerini içeren bir öznitelik için genç ve orta yaş değerleri aralarındaki uzaklık x olarak tanımlanabilirken yine aynı şekilde genç ve yaşlı arasındaki uzaklığın da x olarak tanımlanması mantıklı değildir ve farklı böyle durumlarda farklı uzaklık değerlerinin kullanımı gerekmektedir. Bu problemin önüne geçilebilmesi adına tüm elemanların ve özniteliklerinin üzerinden olasılıksal olarak bir uzaklık değeri hesaplayan yeni bir yöntem önerilmiştir. Önerilen bu yeni uzaklık metriği temelde veri kümesi içerisindeki elemanların sahip oldukları özniteliklere ait değer kümelerinin durumuna ve homojenliğine göre bir hesap yapmaktadır. Örneğin Tablo 3.1'de verilen örnek veri kümesi doğrultusunda tür özniteliği yönetmen bilgisi ve aktör bilgisi özniteliklerine göre daha tutarlı ve verimli bir öbekleme bilgisi sunmaktadır.

Kategorik öznitelikler için önerilen bu uzaklık metriğini hesaplarken A_i^c değerini bir kategorik öznitelik olarak ve x – y değerlerini de bu özniteliğin sahip olduğu değer kümesinin farklı elemanı olarak kabul edelim. Bu iki eleman arasındaki uzaklık hesaplanırken temel olarak sahip oldukları değerlerin diğer öznitelik değerleri ile birliktelikleri gözlenecektir. Bu durumda farklı bir öznitelik kümesi olarak da A_j^c tanımlanmaktadır. Hesaplama işlemine geçilmeden önce yapılması gereken son işlem ise A_j^c özniteliğinin sahip olduğu değerler içerisinde w şeklinde bir alt küme ve bu alt kümenin tersini tutan ~w şeklinde bir alt küme oluşturmaktır. Bu alt kümeler bulunduktan sonra w kümesi içerisindeki değerler ile x değerinin bir arada bulunma olasılığı olan $P_i(w/x)$ ve w alt kümesinin tersi olan ~w alt kümesi içerisindeki değerler ile y değerinin bir arada bulunma olasılığı olan $P_i(\sim w/y)$ değeri hesaplanacaktır. Daha sonra bu iki olasılık değerinin toplamı bize A_i^c özniteliğinin değer kümesinin sahip olduğu x ve y değerlerinin A_j^c özniteliğinin w ve ~w alt

kümeleri doğrultusunda aralarındaki uzaklığı verecektir. Bunu matematiksel bir şekilde ifade etmek gerekirse;

$$\delta_w^i(x,y) = P_i(w/x) + P_i(\sim w/y) \tag{3.4}$$

İçerdiği eleman sayısı n olan bir küme 2^n adet alt kümeye sahip olmaktadır. Bu durumda A_j^c özniteliği için de $2^{|A_j^c|}$ adet farklı w ve buna göre ~w alt kümesi üretilebilmektedir. Bu durumda da her bir w değeri için x – y ikilisi arasındaki uzaklığın hesaplanması gerekmektedir. Bizim istediğimiz ve kullanılacak olan uzaklık w alt kümeleri doğrultusunda x – y ikilisi için hesaplanan en büyük değere sahip olan uzaklıktır. En büyük değerin elde edildiği alt küme ω ile ve bu alt kümenin tersi ~ω ile ifade edilmektedir. Buna göre x – y ikilisi arasındaki uzaklığı matematiksel bir biçimde ifade etmek gerekirse;

$$\delta^{ij}(x,y) = P_i(\omega/x) + P_i(\sim w/x) \tag{3.5}$$

A_i^c özniteliğinin değer kümesi içerisinde yer alan x – y değer ikilisi için uzaklık eşitsizlik 3.5'de gösterildiği gibi hesaplanmaktadır. Bu hesap yapıldıktan sonra istediğimiz ise hesaplanan bu değeri 0 ile 1 arasında bir değere atamaktır yani bir normalizasyon işlemi yapmaktır. Bunun yapılabilmesi için de elde edilen sonuçtan 1 çıkarılarak normalize edilmiş sonuca ulaşılmıştır. Bunun nedeni iki uzaklık değeri hesaplanırken iki olasılık fonksiyonu toplandığı için olabilecek en büyük değerin bu doğrultuda 2 olabilmesinden kaynaklanmaktadır. Bu normalizasyon işlemi de yapıldıktan sonra önerilen uzaklık metriğinin son hali şu şekildedir;

$$\delta^{ij}(x,y) = P_i(\omega/x) + P_i(\sim w/x) - 1 \tag{3.6}$$

Ancak sadece A_i^c özniteliğinin değer kümesi içerisinde yer alan x – y değer ikilisi arasında net bir uzaklık değeri hesaplamak için A_j^c özniteliği doğrultusunda yapılan hesaplama yeterli değildir. Bu uzaklığın hesaplanabilmesi için geriye kalan tüm öznitelikler doğrultusunda x – y ikililerinin bu özniteliklerin değer kümelerindeki maksimum uzaklıklarının bulunması gerekmektedir. Bu işlem yapıldıktan sonra x – y

ikilisi arasındaki net uzaklık elde edilebilir. Bu işlem de şu şekilde ifade edilmektedir;

$$\delta(x,y) = \left(\frac{1}{m-1}\right) \sum_{j=1\ldots m, i\neq j} \delta^{ij}(x,y) \tag{3.7}$$

Bu işlem bize A_i^c özniteliğinin kendisi dışındaki tüm öznitelikler doğrultusunda x – y ikilisi için yapılan uzaklık hesaplamalarının toplamını verecektir. Bu hesaplamada yapılan bölme işleminin sebebi yine elde edilen toplam değeri 0 ile 1 arasında bir değere atamak, normalize etmek isteğinden kaynaklanmaktadır. Her bir öznitelik için x – y arasındaki uzaklık en fazla 1 olabileceği için tüm öznitelikler doğrultusunda uzaklıklar hesaplanıp toplam bir uzaklık bulunduktan sonra uzaklık hesabı yapılan toplam öznitelik sayısına bölünerek 0 ile 1 arasında bir değer elde edilmektedir.

Öznitelik değerlerinin arasındaki uzaklık tanımı bu şekilde yapıldıktan sonra A_i^c özniteliğinin değer kümesinin sahip olduğu x – y ikilisinin uzaklık değeri için şu bilgileri ifade edebiliriz;

$$0 \leq \delta(x,y) \leq 1 \tag{3.8a}$$

$$\delta(x,y) = \delta(y,x) \tag{3.8b}$$

$$\delta(x,x) = 0 \tag{3.8c}$$

Tanımlanan bu uzaklık değerini, Tablo 3.1'de tanımladığımız örnek veri kümesi üzerinden bir örnek ile anlatalım. İlk olarak yapılması gereken değerleri arasında uzaklık hesabı yapılacak olan değer kümesi ikilisinin diğer öznitelikler ile bir arada bulunma olasılıklarının hesaplanmasıdır. Örneğin, aktör bilgisi öznitelik doğrultusunda De Niro – Stewart ikilisi için bir uzaklık değeri hesaplanacak olursa ilk olarak De Niro ve Stewart değerlerinin yönetmen bilgisi özniteliğinin içerdiği değerler ve tür özniteliğinin içerdiği değerler ile hangi olasılıklarla bir arada bulunduğu hesaplanmalıdır. Yapılan bu hesaplamalar sonucunda oluşturulan olasılık tablosu Tablo 3.2'de görülebilir.

Yönetmen / Oyuncu	Tür / Oyuncu
$P = (Scorsese\ /\ De\ Niro) = 1\ /\ 2$	$P = (Polisiye\ /\ De\ Niro) = 1$
$P = (Coppola\ /\ De\ Niro) = 1\ /\ 2$	$P = (Gerilim\ /\ De\ Niro) = 0$
$P = (Hitchcock\ /\ De\ Niro) = 0$	$P = (Komedi\ /\ De\ Niro) = 0$
$P = (Koster\ /\ De\ Niro) = 0$	$P = (Polisiye\ /\ Stewart) = 0$
$P = (Scorsese\ /\ Stewart) = 0$	$P = (Gerilim\ /\ Stewart) = 1\ /\ 2$
$P = (Coppola\ /\ Stewart) = 0$	$P = (Komedi\ /\ Stewart) = 1\ /\ 2$
$P = (Hitchcock\ /\ Stewart) = 1\ /\ 2$	
$P = (Koster\ /\ Stewart) = 1\ /\ 2$	

Tablo 3.2. Öznitelik değerlerinin bir arada bulunma olasılıkları

Bir sonraki aşamada bu değerler doğrultusunda aktör bilgisi özniteliğinin De Niro ve Stewart değerleri arasındaki uzaklık yönetmen bilgisi özniteliği doğrultusunda hesaplanacaktır. Yapılacak olan bu hesaplama, maksimum uzaklık değerini verecek olan ω değer alt kümesi ve bu alt kümenin tersi ∼ω üzerinden yapılmaktadır. Bu durumda De Niro – Stewart değer ikilisi için en büyük uzaklık değerini verecek olan değerler yönetmen bilgisi özniteliği için $\omega = \{\text{Scorsese}, \text{Coppola}\}$ ve $\sim\omega = \{\text{Hitchcock}, \text{Koster}\}$ şeklinde olmalıdır. Bu doğrultuda yapılan hesaplama şu şekildedir;

$$\delta^{Aktör,Yönetmen}(De\ Niro, Stewart)$$
$$= P(Scorsese/De\ Niro) + P(Coppola/De\ Niro)$$
$$+ P(Hitchcock/Stewart) + P(Koster/Stewart) - 1$$
$$= 1/2 + 1/2 + 1/2 + 1/2 - 1 = 1$$

Yönetmen bilgisi özniteliğine göre aktör bilgisi özniteliğinin De Niro ve Stewart değerleri arasındaki uzaklık yönetmen bulunmuştur. Bu noktadan sonra yapılması gereken aktör bilgisi özniteliğinin De Niro ve Stewart değerleri arasındaki uzaklığın tür özniteliği doğrultusunda hesaplanmasıdır. Yapılacak olan bu hesaplama, maksimum uzaklık değerini verecek olan ω değer alt kümesi ve bu alt kümenin tersi ∼ω üzerinden yapılmaktadır. Bu durumda De Niro – Stewart değer ikilisi için en büyük uzaklık değerini verecek olan değerler tür özniteliği için $\omega = \{\text{Polisiye}\}$ ve $\sim\omega = \{\text{Gerilim}, \text{Komedi}\}$ şeklinde olmalıdır. Bu doğrultuda yapılan hesaplama şu şekildedir;

$$\delta^{Aktör,Tür}(De\ Niro, Stewart)$$
$$= P(Crime/De\ Niro) + P(Thriller/Stewart)$$
$$+ P(Comedy/Stewart) - 1 = 1 + 1/2 + 1/2 - 1 = 1$$

Tür özniteliğine göre de aktör bilgisi özniteliğinin De Niro ve Stewart değerleri arasındaki uzaklık yönetmen bulunmuştur. Bu noktadan sonra yapılması gereken tek işlem tüm öznitelikler doğrultusunda bulunan bu uzaklıkların toplanmasıdır ve hesap yapılan öznitelik sayısına bölünmesidir. Bu doğrultuda De Niro ve Stewart arasındaki toplam uzaklık şu şekilde hesaplanacaktır;

$$\delta(De\ Niro, Stewart)$$
$$= \left(\delta^{Aktör,Yönetmen}(De\ Niro, Stewart)\right.$$
$$\left. + \delta^{Aktör,Tür}(De\ Niro, Stewart)\right) / (3-1)$$
$$= (1+1) / 2 = 1$$

Bu işlemlere göre aktör bilgisi özniteliği içerisinde yer alan değerlerden De Niro ve Stewart arasındaki uzaklığın değeri 1 olarak bulunmuştur. Bu işlem dışında Tablo 3.1'de yer alan örnek veri kümesi içerisindeki tüm öznitelikleri arasındaki uzaklık değerleri Tablo 3.3'de görülebilir.

Yönetmen	Aktör	Tür
$\delta(Scorsese, Coppola) = 0$	$\delta(De\ Niro, Stewart) = 1$	$\delta(Polisiye, Komedi) = 1$
$\delta(Scorsese, Hitchcock) = 1$	$\delta(De\ Niro, Grant) = 1$	$\delta(Polisiye, Gerilim) = 1$
$\delta(Scorsese, Koster) = 1$	$\delta(Stewart, Grant) = 0$	$\delta(Gerilim, Komedi) = 1/2$
$\delta(Hitchock, Coppola) = 1$		
$\delta(Koster, Coppola) = 1$		
$\delta(Koster, Hitchcock) = 1/2$		

Tablo 3.3. Öznitelik değerleri arasında hesaplanan uzaklık değerleri

3.5.6. Numerik Öznitelikler İçin Önerilen Ağırlıklı Bir Yöntem

Kategorik öznitelikler için önerilen olasılıksal yöntem doğrultusunda, numerik öznitelikler için de kullanılması önerilen ve numerik özniteliklerin ağırlıklarını

hesaplayarak bu yönde numerik özniteliklere belirli ağırlıklar veren bir yöntem önerilmiştir.[30] Bu yöntem kullanılarak veri kümesi üzerinde numerik özniteliklerin önemleri tespit edilerek bu önem değerleri doğrultusunda ağırlıkları hesaplanmaktadır. Bu durumda veri kümesi içerisinde yer alan her bir numerik öznitelik farklı oranda sonuca etki etmektedir. Bu işlemin yapılmasının sebebi veri kümesi içerisinde öbekler arasında fazla farklılık göstermeyen veya veri kümesinde aynı öbek içerisinde yer alan elemanların sahip olduğu öznitelik değerlerinin çok heterojen bir biçimde dağılması sonucu öbekleme işlemi yapılırken etkin biçimde kullanılamayan özniteliklerin ağırlıklarının azaltılarak etkisiz bir hale getirilmesi ve daha başarılı bir öbekleme yapılmak istenmesidir.

Bu yöntemde bahsedilen önem değerleri, kategorik öznitelikler için önerilen olasılıksal yöntem doğrultusunda değerlerin bir arada bulunma olasılıkları ile hesaplanan değerlerdir. Bu ağırlık değerleri 0 ile 1 arasında değişmektedir. Bu ağırlık hesaplama işlemi kategorik öznitelikler için önerilen olasılıksal yöntemi kullandığı ve kategorik öznitelikler üzerinde uygulandığı için ilk olarak ağırlıkları hesaplanacak olan numerik öznitelikler üzerinde bir ayrıklaştırma (discretization) işlemi yapılmalı ve bu numerik öznitelikler kategorik bir yapıya çevrilmelidir. Örneğin elimizde Tablo 3.4'deki gibi bir veri kümemiz var ise bu veri kümesinde γ özniteliği üzerinde bahsedilen ayrıklaştırma işlemi yapılmalıdır.

Öznitelik ($\boldsymbol{\alpha}$)	Öznitelik ($\boldsymbol{\beta}$)	Öznitelik ($\boldsymbol{\gamma}$)
A	C	1.1
A	C	2.9
A	D	3.1
B	D	4.9
B	C	4.0
A	D	3.2
A	D	4.8

Tablo 3.4. Numerik öznitelik içeren örnek veri kümesi

Ayrıklaştırma işlemi elde bulunan numerik değerleri belirli aralıklarla bölmek ve bu aralıklarda kalan değerlere denk düşen kategorik değerleri atama işlemidir. Elimizde

bulunan veri kümesinde γ özniteliği üzerinde ayrıklaştırma işlemi yapıldıktan sonra olası elde edilecek değerler Tablo 3.5'deki gibi görünmektedir.

Öznitelik ($\boldsymbol{\alpha}$)	Öznitelik ($\boldsymbol{\beta}$)	Öznitelik ($\boldsymbol{\gamma}$)
A	C	a
A	C	a
A	D	b
B	D	b
B	C	b
A	D	b
A	D	b

Tablo 3.5. Numerik öznitelikler üzerinde ayrıklaştıma işlemi yapılmış örnek veri kümesi

Yapılan ayrıklaştırma işlemi sonrası A_i^n şeklinde olan bir numerik öznitelik $u_i^n[1], u_i^n[2], \dots, u_i^n[S]$ olmak üzere S adet aralığa bölünmektedir. Bu noktadan sonra ayrıklaştırılmış olan bu yeni kategorik özniteliğin değer kümesinin içerdiği her bir $u_i^n[r] - u_i^n[s]$ ikilisi için kategorik öznitelikler için önerilen olasılıksal yöntemdeki gibi $\delta(u_i^n[r], u_i^n[s])$ uzaklık değerleri hesaplanmakta ve ağırlık değeri olan w_i bulunmaktadır. Bunu matematiksel bir dille ifade edicek olursak;

$$w_i = \sum_{k=1}^{S} \sum_{j>k}^{S} \delta(u_i^n[r], u_i^n[s]) / \left(S\,(S-1)\,/\,2\right) \tag{3.9}$$

Bu doğrultuda verdiğimiz örnek veri kümesi üzerinde γ özniteliğinin ağırlık değerini hesaplayacak olursak, bu özniteliğin değer kümesi içerisinde yer alan tek ikili olan a – b ikilisi için hesap yapılacaktır. Başka bir deyişle $\delta(x, y)$ değerinin hesaplanması γ özniteliğinin ağırlığının hesaplanması için yeterlidir. Bu işlem şu şekildir;

$$w_i = \delta(x, y) / \left(2\,(2-1)\,/\,2\right) = 0.70\,/(2\,/\,2) = 0.70$$

Yapılan işlem sonucunda elde edilen değere göre γ numerik özniteliğinin ağırlığı 0.70'dir. Bundan sonraki aşamada bulunan bu ağırlık değeri daha önce bahsedilen

numerik öznitelik uygunluk fonksiyonlarından biri ile birlikte kullanılabilir. Bu da aşağıdaki gibi gösterilmektedir;

$$d(X,Y) = w_i \sum_{i=1}^{m} \delta(x_i, y_i) \tag{3.10}$$

3.5.7. Kategorik Öznitelikler İçin Önerilen Ağırlıklı Bir Yöntem

Önerilen bu yöntem de veri kümesinin sahip olduğu kategorik öznitelikler için özelleştirilmiş bir ağırlık hesaplama yöntemidir.[31] Burada da hesaplanan ağırlık ile veri kümesi içerisinde önemi düşük olan özniteliklerin ve önemi yüksek olan öznitelikler tespit edilmektedir. Bu doğrultuda düşük öneme sahip olan özniteliklerin öbekleme sonucuna olan etkileri düşürülmekte ve yüksek öneme sahpi olan özniteliklerin öbekleme sonucuna olan etkileri arttırılmaktadır.

Bu yöntem doğrultusunda ağırlıklar hesaplanırken, numerik öznitelikler için önerilen ağırlıklı yöntemde önerilenin aksine, öznitelik değerlerinin bir arada bulunma olasılıklarını hesaplayan ve bu doğrultuda ağırlıklar bulan bir yapı kullanılmamıştır. Önerilen bu uygunluk fonksiyonunda ağırlıklar hesaplanırken, veri kümesi içerisinde yer alan kategorik özniteliklerin sahip oldukları değer kümelerinin elemanlarının hangi sıklıkla hangi öbekler içerisinde geçtiği bilgisi kullanılmaktadır. Bu hesap yapıldıktan sonra da yine numerik öznitelikler için önerilen ağırlıklı yöntemde önerilenden farklı olarak her bir öznitelik için ağırlık hesaplanmamaktadır. Bunun yerine veri kümesi içerisinde yer alan ve aralarındaki uzaklık hesaplanan her x – y ikilisi için bu ikilinin sahip oldukları öznitelik değerleri doğrultusunda bu ikiliye özel bir ağırlık hesaplanması söz konusudur. Ağırlıklar, x ve y elemanlarının kategorik özniteliklerinin sahip oldukları değerlerin frekans değerlerinin bulundukları öbek içerisindeki değerlere oranı doğrultusunda hesaplanmaktadır. Örneğin n adet elemana, m^c adet kategorik özniteliğe sahip ve k adet öbeğe ayrılacak olan, her bir öbeğin C_k ile ifade edildiği, öbeklerin merkezlerini simgeleyen kategorik özniteliklerin değerlerinin $A^c_{k,i}$ şeklinde gösterildiği bir D veri kümesi için hesaplama şu şekilde yapılmaktadır;

$$d(X,Y) = \sum_{i=1}^{m} \delta(x_i, y_i) \qquad (3.11a)$$

$$\delta(x_i, y_i) = \begin{cases} 1 - w_{k,i} & if\ x_i = y_i \\ 1 & if\ x_i \neq y_i \end{cases} \qquad (3.11b)$$

$$w_{k,i} = \left(freq\left(A_{k,i}^{c} / C_k\right) / |C_k|\right) * \left(freq\left(A_{k,i}^{c} / D\right) / |D|\right) \qquad (3.11c)$$

Yapılan bu ağırlık hesaplama işleminin önerilen diğer ağırlık hesaplama yöntemlerinden farkı her iterasyon sonrasında ağırlıkların tekrar hesaplanıyor ve güncelleniyor olmasıdır. Önerilen diğer yöntemlerde ağırlıklar tek bir defa hesaplanmakta ve programın çalıştığı süre içerisinde hesaplanan bu ağırlıklar kullanılmaktadır. Bir diğer fark ise diğer önerilen yöntemlerde hesaplanan uzaklıklar farklı k – modes uygunluk fonksiyonları veya k – means uygunluk fonksiyonları ile kullanılabilirken, hesaplanan bu uzaklık değerinde bu şekilde bir seçenek bulunmamakta ve başka bir uygunluk fonksiyonu ile birleştirilmeden kendi başına kategorik öznitelikler için bir uygunluk fonksiyonu görevi görmektedir.

3.5.8. Hem Numerik Hem de Kategorik Öznitelikler İçin Önerilen Ağırlıklı Bir Yöntem

Önerilen bu uygunluk fonksiyonu da aynı numerik öznitelikler için önerilen ağırlıklı yöntemde ve kategorik öznitelikler için önerilen ağırlıklı yöntemde olduğu gibi özniteliklerin ağırlıklarını bulmakta ve bu doğrultuda kullanılacak olan uzaklık metriğinin hesaplanan ağırlık değeri kadar etkili olmasını sağlamaktadır. Bu ağırlık hesaplama yöntemi[32], veri kümesinin sahip olduğu hem numerik öznitelikler hem de kategorik öznitelikler için kullanılabilir. Ancak kategorik özniteliklerin ve numerik özniteliklerin ağırlıkları hesaplanırken bu farklı iki tipteki öznitelikler birbirlerinden bağımsız olarak değerlendirilmektedirler.

Önerilen bu yöntemde yapılan hesaplama işlemi, numerik öznitelikler için önerilen ağırlıklı yöntemdekinin ve kategorik öznitelikler için önerilen ağırlıklı yöntemdekinin aksine olasılıksal bir yaklaşım kullanmamıştır. Hesaplanan ağırlık

değerleri, veri kümesi içerisinde yer alan özniteliklerin elemanlar arası değişimi doğrultusunda değerlendirilmiştir. Başka bir deyişle bir özniteliğin farklı öbekler içerisinde yer alan elamanlar için değişiminin düşük olması sonucu o özniteliğin belirleyiciliğinin düşük olduğu aksine farklı öbekler içerisinde yer alan elemanlar için özniteliğin değerleri arasındaki değişimin yüksek olması sonucu seçilen özniteliğin belirleyiciliğinin yüksek olduğu varsayılmıştır.

Bu uygunluk fonksiyonunda önerilen ağırlık hesaplama yönteminin, numerik öznitelikler için önerilen ağırlıklı yöntemde ve kategorik öznitelikler için önerilen ağırlıklı yöntemde önerilen ağırlık hesaplama yönteminden bir diğer farkı ise elde edilen ağırlık değerleridir. Bu yöntemde önerilen ağırlık değerleri kendi aralarında bir denge sağlamak durumundadırlar. Bu durumu daha açık bir şekilde ifade etmek gerekirse, numerik öznitelikler için önerilen ağırlıklı yöntemde ve kategorik öznitelikler için önerilen ağırlıklı yöntemde ağırlıktan öte bir önem değeri hesaplanmaktadır ve buna göre tüm öznitelikler önemli veya tüm öznitelikler önemsiz olarak kabul edilebilir. Ancak önerilen bu ağırlık hesaplama yönteminde böyle bir durum oluşmamaktadır. Belirli öznitelikler aynı ağırlık değerine sahip olmakla birlikte tüm özniteliklerin yüksek ağılıkta olması veya hepsinin düşük ağırlıkta olması gibi bir durum söz konusu değildir. Buna göre aşağıdaki eşitsizlik sağlanmalıdığır.

$$\sum_{j=1}^{m} w_j = 1, \qquad 0 \leq w_j \leq 1 \tag{3.12}$$

Ağırlıklar hesaplanırken veri kümesi içerisindeki her elemanın sahip olduğu numerik ve kategorik özniteliklerin değerlerinin, veri kümesi içerisinde yer alan diğer elemanların sahip oldukları numerik ve kategorik öznitelik değerlerine olan toplam uzaklıkarı bulunmaktadır. Bu işlem n adet elemana, $|A^c|$ adet kategorik ve $|A^r|$ adet numerik olmak üzere toplam m adet özniteliği sahip ve k adet öbeğe bölünecek olan bir D veri kümesi için şu şekildedir;

$$d_j^c = \sum_{l=1}^{n} \sum_{i=1}^{n} d^c\left(x_{i,j}, y_{l,j}\right), \qquad i \neq j \tag{3.13a}$$

$$d_j^r = \sum_{l=1}^{n} \sum_{i=1}^{n} d^r\left(x_{i,j}, y_{l,j}\right), \qquad i \neq j \tag{3.13b}$$

Veri kümesinin sahip olduğu kategorik ve numerik öznitelikler için yukarıdaki toplam uzaklık değerleri hesaplandıktan sonra hesaplanan bu değerler normalize edilirler. Yapılan bu normalizasyon işleminin nedeni numerik ve kategorik özniteliklerin uzaklık değerlerinin birbirinden çok farklı olabilmesinden kaynaklanmaktadır. Örneğin numerik öznitelikler için bu uzaklık değerleri çok büyük sayılar olabilcekken, kategorik öznitelikler için bu uzaklık değerleri en fazla veri kümesinin içerdiği eleman sayısı kadar olabilmektedir. Bu durum dışında farklı numerik öznitelikler doğrultusunda hesaplanan uzaklık değerleri arasında da büyük farklar ortaya çıkabilir. Bu sebeplerden ötürü elde edilen toplam uzaklık değerleri normalize edilerek, değerleri 0 ile 1 arasına sıkıştırılmaktadır. Bu işlem numerik ve kategorik öznitelikler için elde edilen toplam uzaklık değerinin elde edilebilecek olan en büyük toplam uzaklık değerine bölünmesi ile elde edilmektedir. Bu normalizasyon işlemlerini matematiksel bir şekilde ifade edicek olursak;

$$norm\left(d_j^c\right) = d_j^c / \left(n * (n-1)\right) \tag{3.14a}$$

$$norm\left(d_j^r\right) = d_j^r / \left(n * (n-1) * \left(\max\left(A_j^r\right) - \min\left(A_j^r\right)\right)\right) \tag{3.14b}$$

Normalizasyon işlemleri yapıldıktan sonra elde edilen değerler doğrultusunda ağırlık değerleri hesaplanmaktadır. Bu hesap işlemi de elde edilen normalize edilmiş olan özniteliklerin toplam uzaklık değerlerin diğer özniteliklik değerleri ile belirli aralıklarda orantılanması şeklinde yapılmaktadır. Bu işlemler yapılırken h^c ile ifade edilen hesaplanan $\mathrm{norm}\left(d_j^c\right)$ değerleri sonucunda uzaklık değeri 0 olmayan kategorik özniteliklerin sayısı, h^r ile ifade edilen hesaplanan $\mathrm{norm}\left(d_j^r\right)$ değerleri sonucunda uzaklık değeri 0 olmayan numerik özniteliklerin sayısı, β ise sabit bir değişkendir. Bu doğrultuda ağırlık hesabı aşağıdaki gibi yapılmaktadır;

$$w_j^c = \begin{cases} 0 & if\ norm\left(d_j^c\right) = 0 \\ \dfrac{1}{\sum_{t=1}^{h}\left[\dfrac{norm\left(d_j^c\right)}{norm\left(d_t^c\right)}\right]^{\frac{1}{\beta-1}}} & if\ norm\left(d_j^c\right) \neq 0 \end{cases} \qquad (3.15a)$$

$$w_j^r = \begin{cases} 0 & if\ norm\left(d_j^r\right) = 0 \\ \dfrac{1}{\sum_{t=1}^{h}\left[\dfrac{norm\left(d_j^r\right)}{norm\left(d_t^r\right)}\right]^{\frac{1}{\beta-1}}} & if\ norm\left(d_j^r\right) \neq 0 \end{cases} \qquad (3.15b)$$

β sayısı 1 değeri dışında herhangi bir değer alabilmektedir. Bu sayının değeri doğrultusunda hesaplanan ağırlık değerleri ve hassaslıkları değişmektedir. Son olarak elde edilen numerik öznitelikler için olan ağırlık değerleri herhangi bir k – means uzaklık metriği, kategorik öznitelikler için olan ağırlık değerleri ise herhangi bir k – modes uzaklık metriği ile birlikte kullanılabilir.

3.6. K – Means / K – Modes Operatör

Kromozomlara ilk aşamada rastgele bir biçimde atanmış olan veya daha sonradan kromozomların kendi aralarında çaprazlanması sonucunda ortaya çıkan yeni kromozomların içerisinde yer alan verilerin yer aldıkları öbekler üzerinden hesaplanan uygunluk fonksiyonları, kromozomların başarımları ve devamlılıklarını sürdürüp sürdüremeyecekleri hakkında fikir sahibi olmamızı sağlamaktadır. Daha sonra da başarımı yüksek olan bu kromozomlar seçilerek en iyi sonuçlar elde edilmeye çalışılmaktadır. Ancak kromozomların çaprazlanması sonucu elde edilen yeni kromozomların her zaman çok daha iyi ve başarımı yüksek olan kromozomlar olarak ortaya çıkmamakta ve bu sebepten algoritma çok uzuk sürebilmektedir. Algoritmanın çalışma süresinin kısaltılması ve en iyi sonuçlara hızlıbir şekilde ulaşılabilmesi adına k – means / k – modes operator ismi verilen bir ara operatör tanımlanmıştır.

Bu operatörün görevi uygunluk fonksiyonları hesaplandıktan ve başarımı yüksek olan kromozomlar seçildikten sonra, bu kromozomlar içerisinde yer alan verilerin

daha uygun bir öbek içerisine aktarılıp aktarılamayacağının kontrolünü yapmaktır. Bu işlem yapılırken de veri kümesi içerisinde yer alan her bir elemanın öbeklerin sahip oldukları merkezlere olan uzaklıkları hesaplanmakta ve her bir eleman merkezine en yakın olan öbek içerisine yerleştirilecek şekilde güncellenmektedir. Örneğin bir veri kümesi içerisinde yer alan x elamanı, öbekleme işlemi sonucunda elde edilmiş olan $\mathrm{C} = \mathrm{C}^{\mathrm{r}}\ \mathrm{U}\ \mathrm{C}^{\mathrm{c}}$ şeklindeki öbeklerin merkez noktaları ve $\mathrm{A} = \mathrm{A}^{\mathrm{r}}\ \mathrm{U}\ \mathrm{A}^{\mathrm{c}} = \{\mathrm{A}_1, \mathrm{A}_2\ \ldots\ \mathrm{A}_{\mathrm{m}}\}$ şeklindeki öznitelikler için öbeğin merkezi ve öznitelik arasındaki uzaklık aşağıdaki gibi hesaplanmaktadır;

$$d(x, C_k) = \sum_{i=1}^{|A^r|} \delta(x, C_{k,i}^r) + \sum_{j=1}^{|A^c|} \delta(x, C_{k,j}^c) \tag{3.16}$$

BÖLÜM 4

4. DOĞRU SAYIDAKİ ÖBEK SAYISININ TESPİTİ VE DEĞERLENDİRME METRİKLERİ

4.1. H – Confidence ile Öbeklerin Birleştirilmesi

Önerilen algoritmada diğer öbekleme algoritmalarının aksine veri kümesinin kaç adet öbek içerisinde toplanacağı bilgisi algoritmaya girilmemektedir ve daha önce de bahsedildiği üzere öbek sayısı veri kümesi içerisinde yer alan toplam eleman sayısının karekök değerinin ikilik tabanda gösterildiği toplam bit sayısı doğrultusunda belirlenmektedir. Bu durumda algoritma tam olarak kaç adet öbek bulunduğunu bilmediği için farklı öbek adetleri ile de yapılmış sonuçlar görüntülenmek istenmektedir. Bunun için bir yöntem öbek sayısının otomatik hesaplanması yerine belirli aralıktaki sayıların her biri için algoritmanın baştan çalıştırılmasıdır. Örneğin bir veri kümesi öbek sayısı 1 ile 10 arasında değerler alacak şekilde teker teker çalıştırılır. Ancak bu durumun maliyeti oldukça yüksektir ve algoritmayı kullanan kişiye çok fazla zaman kaybettirmektedir. Bu şekilde masraflı bir işlemin yapılmasındansa bu durumun iyileştirilebilmesi adına farklı bir yöntem kullanılmıştır. Bu yöntem h – confidence'dır.

Algoritma tamamlandığında kullanıcıya ilk aşamada belirttiği sayıda kromozom dönmektedir. Geriye dönen bu kromozomlar algoritma sonunda başarımı en yüksek bulunan ve en kuvvetli nesil olarak seçilen kromozomlardır. Bu kromozların içerisinde hesaplanan sayıda öbeğe bölünmüş veri kümesi sonuçları her bir kromozomun dizisi olarak bulunmaktadır. Bu kromozomlar içerisinde bulunan bilgiler doğrultusunda veri kümesi içerisinde yer alan her bir x – y ikilisinin toplamda kaç kromozomda aynı öbek içerisinde yer aldığı bilgisi hesaplanır ve hesaplanan bu değerlerden n x n büyüklüğünde bir birliktelik matrisi oluşturulur. Örneğin geriye sonuç olarak 100 kromozomun döndüğü bir algoritma için x – y ikilisi bu kromozomların 34 tanesinde aynı öbek içerisinde yer almışlar ise birliktelik

matrisindeki değerleri 34 olmaktadır. Bu işlem veri kümesinde içerisinde yer alan her x – y ikilisi için tekrar edilmektedir.

Bu aşamadan sonra dipten tepeye doğru ilerleyen birleşen yapıdaki bir algoritma kullanılarak öbekler kendi aralarında teker teker birleştirilmektedir. Bu birleştirme işlemi, genetik öbekleme algoritması sonucunda elde edilen kromozomlar yardımıyla hesaplanan birliktelik matrisi içerisindeki değerler doğrultusunda yapılmaktadır. Birliktelik matrisi içerisinde yer alan bu değerlerden bir uzaklık değeri hesaplanmakta ve birbiri ile en yakın olan öbekler bir üst seviyeye birleşerek çıkmaktadırlar. Hesaplanan bu uzaklık değeri öbekler içerisindeki elemanların hepsinin aynı anda bir arada bulunduğu kromozom sayısı ve toplam kromozom sayısının birbirleri ile oranlanması sonucu elde edilmektedir. Bu uzaklık matematiksel olarak aşağıdaki gibi ifade edilmektedir. Bu ifadede num() fonksiyonu içerisine aldığı elemanların tüm kromozomlar içerisinde kaç defa bir arada bulunduğunu belirtmektedir.

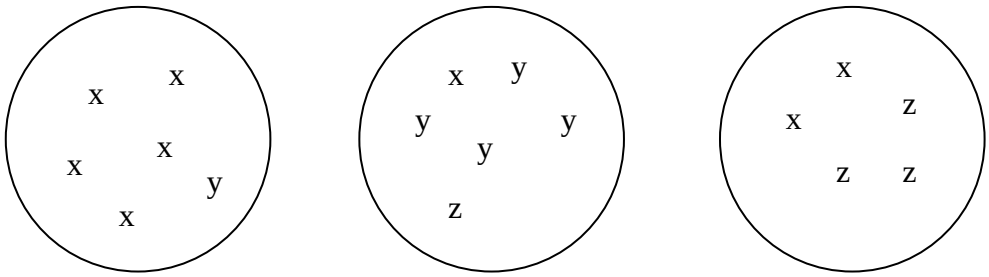
$$h(x_1, x_2, \dots, x_n) = \frac{num(x_1, x_2, \dots, x_n)}{\max\big(num(x_1), num(x_2), \dots, num(x_n)\big)} \tag{4.1}$$

Bu işlem sonucunda algoritma çalıştırıldığı anda hesaplanan, oluşturulacak öbek sayısından başlayarak tek öbek kalıncaya kadar birleştirme işlemi yapılmaktadır. Yapılan her birleştirme işlemi sonucunda da elde edilen yeni öbek dağılımları kullanıcaya geri döndürülmektedir.

4.2. Saflık İndeksi

Öbekleme işlemi yapıldıktan sonra elde edilen sonuçların ne kadar doğru bir şekilde öbeklendiğinin kontrol edilmesi gerekmektedir. Bu kontrol işleminin yapılabilmesi için saflık indeksi (purity index)[33] ismi verilen bir metrik kullanılmıştır. Saflık indeksi temel olarak öbekleme işlemi sonucunda elde edilmiş olan öbekler içerisinde yer alan verilerin ne kadar homojen bir biçimde dağılmış olduğundan yola çıkar bir hesap yapmaktadır. Örneğin algoritmanın çalışması tamamlandıktan sonra Şekil 4.1'deki gibi bir öbekleme sonucu elde edilmiş olsun. Buna göre sonuçlara

bakıldığında ilk öbek içerisinde 5 adet x türünden ve 1 adet y türünden, ikinci öbek içerisinde 1 adet x türünden, 4 adet y türünden ve 1 adet z türünden, üçüncü öbekte ise 2 adet x türünden ve 3 adet z türünden olmak üzere toplamda 17 adet eleman bulunmaktadır.



Öbek 1 Öbek 2 Öbek 3

Şekil 4.1. Örnek bir öbekleme sonucu

Yapılan bu öbekleme işleminden sonra yapılan yukarıda da bahsedildiği gibi öbeklerin saflık derecesinin tespitidir. Bu saflık, bir öbek içerisinde yer alan en dominant, yani en fazla sayıda bulunan elemanlar tespit edilip daha sonra bu elemanların toplam sayılarının, tüm veri kümesi içerisinde yer alan elemanların toplam sayısı ile oranlanması şeklinde hesaplanmaktadır. İçerisinde n adet eleman, k adet öbek içeren ve her bir elemanın türünün t_i ile gösterildiği bir D veri kümesi için yapılacak olan saflık indeksi hesabının matematiksel gösterimi şu şekilde olmalıdır;

$$purity = \frac{1}{n}\sum_{k=1}^{|C|}\max\big((t_1 \cap C_k),(t_2 \cap C_k),\dots,(t_i \cap C_k)\big) \qquad (4.2)$$

Bu saflık indeksi verilen örnek veri kümesi için hesaplanacak olursa hesap aşağıdaki gibi olacaktır;

$$purity = \frac{1}{17}\,(\max(5,1,0) + \max(1,4,1) + \max(2,0,3))$$

$$= \frac{1}{17}\,(5+4+3) = \frac{12}{17} = 0.71$$

Hesaplanan saflık değeri yaklaşık olarak 0.71 bulunmaktadır. Saflık değeri 0 ile 1 arasında bir değer almaktadır. Elde edilen sonuçlar veri kümesinde bulunan verilerin olması gereken ve algoritma sonucunda bulundukları öbeklerin doğruluğunun yüzde değeri olarak da düşünülebilir. Elde edilen sonuçlar doğrultusunda hesaplanan saflık değerinin 1'e yaklaşması istenilen durumdur ve elde edilen sonuçların en iyi sonuca yaklaştığını göstermektedir.

4.3. Delta IEE Kare Metriği ile Öbek Sayısı Tahmini

Önerilen genetik öbekleme algoritmasında veri kümesinin kaç öbeğe bölüneceği bilgisi diğer öbekleme algoritmalarında olduğu gibi kullanıcı tarafından girilmemektedir ve bu sayı daha önce de bahsedildiği üzere algoritma tarafından veri kümesinin içerdiği eleman sayısı doğrultusunda otomatik olarak hesaplanmaktadır. Bu işlemin otomotik olarak yapılması tamamen kullanıcıdan bağımsız bir algoritma kullanma imkanı sunarken başka bir taraftanda istenmeyen durumlar oluşmasına sebep olabilmektedir. Algoritma veri kümesinin içerdiği eleman sayısı doğrultusunda veri kümesinin kaç öbeğe ayrılacağını hesapladığı için fazla eleman içeren veri kümeleri için bu öbek sayısı da fazla çıkacaktır. Örneğin algoritma 950 adet veri içeren bir veri kümesi üzerinde çalıştırılacak ise alogritma bu veri kümesini yaptığı öbek sayısı hesabı doğrultusunda 32 ayrı öbekte toplamaya çalışacaktır. Ayrıca veri kümesi içerisindeki eleman sayısı arttıkça bu öbek sayısı da üssel olarak artmakdatır. Ancak, genelde veri kümesinin bölünmesi gereken öbek sayıları bu kadar yüksek değildir. Bu sebeple algoritma içerisinde veri kümesi için en uygun olan öbek sayılarının tespit edilmesi gerekmektedir. Uygun sayıların tespit edilmesi de daha önce belirtildiği üzere delta IEE kare metriği[34] ile yapılmaktadır.

Delta IEE kare metriği farklı sayıda öbeklere bölünerek elde edilmiş olan, öbekleme algoritmasının sonuçları üzerinden hesaplanan bir metriktir. Önerilen algoritmada farklı öbek sayıları için en baştan bir hesaplama işlemi yapılmamaktadır. Ancak önerilen h – confidence metodu ile 1 ile ilk aşamada hesaplanan öbek sayısı arasındaki tüm değerler için sonuçlar elde edilmektedir. Delta IEE kare metriğinin dayandığı temel nokta h – confidence sonucu elde edilen farklı öbekleme

sonuçlarının içerisinde yer alan değişimdir. İlk aşamada entropi (entropy) yardımıyla elde edilen farklı sonuçlar için düzensizlik hesaplanır. Bu düzensizlik hesaplaması işlemi, algoritma sonucunda elde edilen öbekler içerisinde aynı öbekte yer alan elemanların sahip oldukları öznitelik değerlerinin değişimi göz önüne alınarak yapılmaktadır. Entropi hesabı olasılıksal bir hesap olduğu ve numerik öznitelikler üzerinde çok sağlıklı sonuçlar alınamadığı için, hesaplama işleminden önce numerik öznitelikler üzerinde bir ayrıklaştırılma işlemi yapılmakta ve veri kümesi içerisinde yer alan tüm öznitelikler kategorik öznitelik olacak şekilde bir düzenleme işlemi yapılmaktadır. Bu hesaplama işlemi m adet özniteliğe sahip olan bir veri kümesi için matematiksel olarak aşağıdaki gibi ifade edilebilir;

$$E(\tau) = \frac{1}{m} \sum_{i=1}^{m} E(A_i) \tag{4.3a}$$

$$E(A_i) = -\frac{1}{log_2|A_i|} \sum_{a_{ij} \in A_i} P(a_{ij}) log_2 P(a_{ij}) \tag{4.3b}$$

Daha sonra bu entropi işlemi kullanılarak her öbek için beklenen entropi (expected entropy) (EE) değeri hesaplanmaktadır. Bu değer de öbeklerin kendi içlerindeki düzensiliklerinden yola çıkarak kümeleme işlemi bittikten sonra tüm veri kümesinin düzensizliğini hesaplamak için yapılmaktadır. Bu hesaplama işlemi matematiksel olarak aşağıdakigibi ifade edilmektedir;

$$EE(C^k) = \sum_{i=1}^{k} \frac{n_i}{n} E(C_i) = \frac{1}{n} \sum_{i=1}^{k} n_i E(C_i) \tag{4.4}$$

Bu aşamadan sonra, elde ettiğimiz bu entropi ve beklenen entropi değerleri kullanılarak farklı öbek sayıları için elde edilmiş olan sonuçlar için beklenen entropideki artış oranı (increase rate of expected entropy) (IEE) hesaplanabilmektedir. Bu hesaplama işlemi k öbeğe bölünmüş olan ve k + 1 öbeğe bölünmüş olan iki sonucun beklenen entropi değerlerinin farkı alınarak hesaplanmaktadır. Bu hesap aşağıdaki gibi yapılmaktadır;

$$IEE(k) = EE(C^k) - EE(C^{k+1}) \quad (4.5)$$

Son olarak, elde edilen beklenen entropideki artış oranı değerlerinden yola çıkarak IEE sonuçlarının 1. dereceki diferansiyel farkı (the differantial order of IEE curve) (ΔIEE) ve IEE sonuçlarının 2. dereceki diferansiyel farkı (second differantial order of IEE curve) (Δ^2IEE) değerleri hesaplanmaktadır. Bu hesaplar aşağıdaki şekilde ifade edilmektedir;

$$\Delta IEE(k) = \Delta IEE(k) - \Delta IEE(k+1) \quad (4.6)$$

$$\Delta^2 IEE(k) = \Delta^2 IEE(k-1) - \Delta^2 IEE(k) \quad (4.7)$$

Bu değerler hesaplandıktan sonra hesaplanan değerlerdeki artış ve bu değerler bir grafiğe yerleştirildiğinde elde edilen tavan ve taban noktaları doğrultusunda veri kümesi için ideal olan öbek sayıları seçilebilmektedir. Bu işlem yapılırken en yüksek değere sahip noktaların ideal öbek sayıları olarak seçilmesinin sebebi, o noktalarda yapılan öbekleme işleminin kendisinden önce ve sonra gelen sonuçlar ile karşılaştırıldığında en büyük fark yaratan nokta olmalarından kaynaklanmaktadır.

BÖLÜM 5

5. DENEYLER

Bu bölümde önerilen yöntemler doğrultusunda yapılan deneylerin ayrıntılarına ve sonuçlarına yer verilmiştir. Deneyler yapılırken önerilen yöntemin her alanda çalışabildiğini göstermek adına 1 adet sadece numerik özniteliklerden oluşan veri kümesine, 2 adet sadece kategorik özniteliklerden oluşan veri kümesine ve 2 adet de hem kategorik hem numerik öznitelikler içeren karışık türdeki veri kümesi üzerinde deneyler yapılmıştır. Veri kümeleri ve özellikleri Tablo 5.1'de görülebilir.

Veri Kümesi	Öbek Sayısı	Eleman Sayısı	Kategorik Öznitelik Sayısı	Numerik Öznitelik Sayısı	Toplam Öznitelik Sayısı
Iris	3	150	-	4	4
Zoo	7	101	17	-	17
Congressional Voting	2	435	16	-	16
Heart Disease	2	270	8	5	13
Australian Credit	2	690	8	6	14

Tablo 5.1. Kullanılan veri kümeleri ve özellikleri

Veri kümeleri sırası ile önerilen yöntem doğrultusunda belirli uygunluk fonksiyonları ile çalıştırılmış ve sonuçlar alınmıştır. Daha sonra elde edilen bu sonuçlar üzerinden saflık indeksleri hesaplanmış ve sonuçların başarımı ölçülmüş bir sonraki aşamada ise elde edilen sonuçlar için ideal öbek sayısı tahmin edilmeye çalışılmıştır. Bu bölümde sırası ile veri kümeleri için elde edilen sonuçlar görülebilir.

5.1. Iris Veri Kümesi

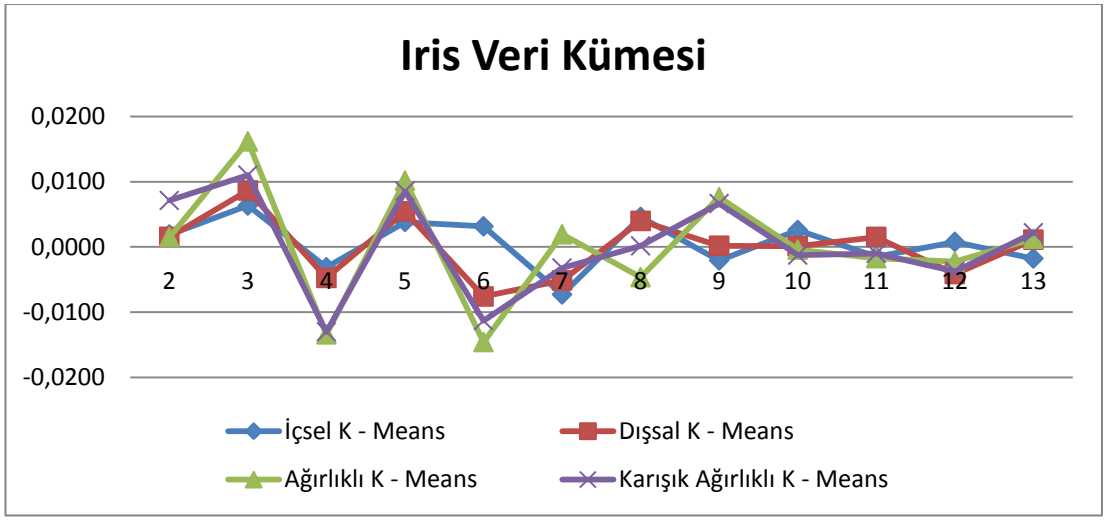
Iris veri kümesi sadece numerik özniteliklerden oluşan bir veri kümesidir. İçerisinde 3 farklı öbek içerisinde yer alması gereken toplam 150 adet veri bulunmaktadır. Bu veriler 4 farklı öznitelik içermektedir ve içerdiği bu 4 öznitelik de numerik değerlerden oluşmaktadır. Bu veri kümesi üzerinde önerilen algoritma k – means

içsel uzaklık, k – means dışsal uzaklık, ağırlıklı k – means, hem kategorik hem numerik öznitelikler için olan ağırlıklı karışık ağırlıklı k – means uygunluk fonksiyonları ile çalıştırılmış ve elde edilen sonuçlar orjinal k – means algoritmasının çalıştırılması sonucu elde edilen sonuçlar ile karşılaştırılmıştır. Elde edilen bu saflık indeksi sonuçları Tablo 5.2'de görülebilir.

Iris Veri Kümesi					
Öbek Sayısı	Karışık Ağırlıklı K - Means	Ağırlıklı K- Means	İçsel K - Means	Dışsal K - Means	K - Means
3	0,93	0,95	0,90	0,85	0,88
4	0,93	0,94	0,91	0,86	0,88
5	0,94	0,96	0,91	0,88	0,89
6	0,95	0,96	0,91	0,88	0,89
7	0,95	0,97	0,91	0,88	0,90
8	0,95	0,97	0,91	0,88	0,90

Tablo 5.2. Iris veri kümesi için saflık indeksi sonuçları

Bir sonraki aşamada farklı öbek sayılarındaki sonuçlar üzerinden $\Delta^2 IEE$ metriği hesaplanmakta ve ideal öbek sayısının bulunduğu nokta tespit edilmektedir. İdeal öbek sayısı seçilirken $\Delta^2 IEE$ değerinin en yüksek değere ulaştığı nokta belirlenmekte ve bu nokta ideal öbek sayısı olarak seçilmektedir. Hesaplanan delta $\Delta^2 IEE$ değerlerinin yer aldığı sonuçlar Grafik 5.1'de görülebilir.



Grafik 5.1. Iris veri kümesi için $\Delta^2 IEE$ sonuçları

Grafik üzerinde görülebileceği üzere farklı uygunluk fonksiyonlarının sonuçları için öbek sayının 3 olduğu noktada $\Delta^2 \boldsymbol{IEE}$ değerleri yüksektir. Iris veri kümesinin ideal öbek sayısı bu doğrultuda önerilen yöntemde 3 olarak bulunmaktadır ve olması gereken değer ile karşılaştırıldığında elde edilen değerin doğru olduğu görülmektedir. Bunun yanında önerilen yöntemde elde edilen sonuçlar orjinal k – means algoritmasına oranla saflık indeksi açısından daha yüksek bulunmuştur.

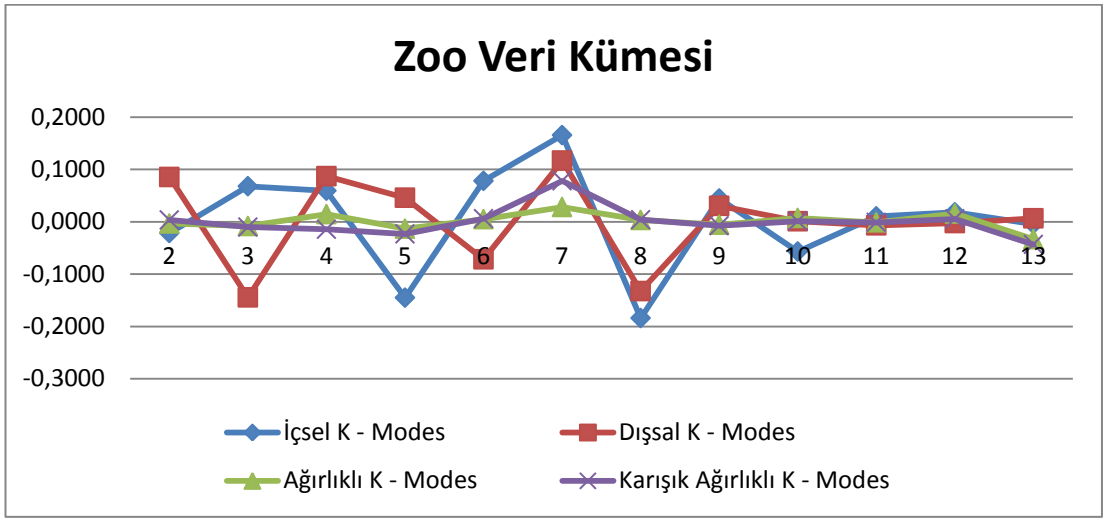
5.2. Zoo Veri Kümesi

Zoo veri kümesi sadece kategorik özniteliklerden oluşan bir veri kümesidir. İçerisinde 7 farklı öbek içerisinde yer alması gereken toplam 101 adet veri bulunmaktadır. Bu veriler 17 farklı öznitelik içermektedir ve içerdiği bu 17 öznitelik de kategorik değerlerden oluşmaktadır. Bu veri kümesi üzerinde önerilen algoritma k – modes içsel uzaklık, k – modes dışsal uzaklık, ağırlıklı k – modes, hem kategorik hem numerik öznitelikler için olan ağırlıklı karışık ağırlıklı k – modes, olasılıksal k – modes uygunluk fonksiyonları ile çalıştırılmış ve elde edilen sonuçlar orjinal k – modes algoritmasının çalıştırılması sonucu elde edilen sonuçlar ile karşılaştırılmıştır. Elde edilen bu saflık indeksi sonuçları Tablo 5.3'de görülebilir.

Zoo Veri Kümesi						
Öbek Sayısı	Karışık Ağırlıklı K - Modes	Ağırlıklı K- Modes	Olasılıksal K - Modes	İçsel K - Modes	Dışsal K - Modes	K - Modes
7	0,78	0,74	0,90	0,78	0,70	0,71
8	0,79	0,74	0,92	0,82	0,82	0,72
9	0,80	0,75	0,93	0,84	0,82	0,76
10	0,85	0,77	0,96	0,88	0,83	0,79
11	0,88	0,81	0,97	0,88	0,83	0,80
12	0,92	0,83	0,97	0,92	0,85	0,83
13	0,95	0,85	0,98	0,94	0,85	0,84
14	0,96	0,86	0,98	0,95	0,86	0,85
15	0,97	0,86	0,98	0,95	0,88	0,86
16	0,97	0,87	0,98	0,97	0,88	0,87

Tablo 5.3. Zoo veri kümesi için saflık indeksi sonuçları

Bir sonraki aşamada farklı öbek sayılarındaki sonuçlar üzerinden $\Delta^2 IEE$ metriği hesaplanmakta ve ideal öbek sayısının bulunduğu nokta tespit edilmektedir. İdeal öbek sayısı seçilirken $\Delta^2 IEE$ değerinin en yüksek değere ulaştığı nokta belirlenmekte ve bu nokta ideal öbek sayısı olarak seçilmektedir. Hesaplanan delta $\Delta^2 IEE$ değerlerinin yer aldığı sonuçlar Grafik 5.2'de görülebilir.



Grafik 5.2. Zoo veri kümesi için $\Delta^2 IEE$ sonuçları

Grafik üzerinde görülebileceği üzere farklı uygunluk fonksiyonlarının sonuçları için öbek sayının 7 olduğu noktada $\Delta^2 IEE$ değerleri yüksektir. Zoo veri kümesinin ideal öbek sayısı bu doğrultuda önerilen yöntemde 7 olarak bulunmaktadır ve olması gereken değer ile karşılaştırıldığında elde edilen değerin doğru olduğu görülmektedir. Bunun yanında önerilen yöntemde elde edilen sonuçlar orjinal k – modes algoritmasına oranla saflık indeksi açısından daha yüksek bulunmuştur.

5.3. Congressional Voting Veri Kümesi

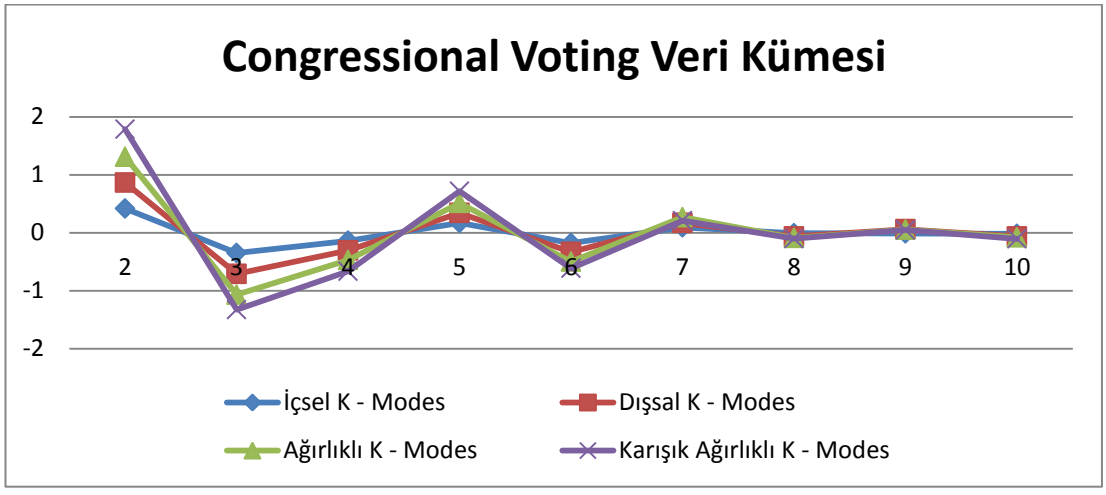
Congressional Voting veri kümesi sadece kategorik özniteliklerden oluşan bir veri kümesidir. İçerisinde 2 farklı öbek içerisinde yer alması gereken toplam 435 adet veri bulunmaktadır. Bu veriler 16 farklı öznitelik içermektedir ve içerdiği bu 16 öznitelik de kategorik değerlerden oluşmaktadır. Bu veri kümesi üzerinde önerilen algoritma k – modes içsel uzaklık, k – modes dışsal uzaklık, ağırlıklı k – modes, hem kategorik

hem numerik öznitelikler için olan ağırlıklı karışık ağırlıklı k – modes, olasılıksal k – modes uygunluk fonksiyonları ile çalıştırılmış ve elde edilen sonuçlar orjinal k – modes algoritmasının çalıştırılması sonucu elde edilen sonuçlar ile karşılaştırılmıştır. Elde edilen bu saflık indeksi sonuçları Tablo 5.4'de görülebilir.

Congressional Voting Veri Kümesi						
Öbek Sayısı	Karışık Ağırlıklı K - Modes	Ağırlıklı K- Modes	Olasılıksal K - Modes	İçsel K - Modes	Dışsal K - Modes	K - Modes
2	0,77	0,79	0,83	0,74	0,66	0,61
3	0,82	0,84	0,85	0,77	0,68	0,69
4	0,82	0,87	0,85	0,80	0,68	0,78
5	0,83	0,87	0,85	0,85	0,69	0,78
6	0,83	0,87	0,87	0,85	0,70	0,81
7	0,84	0,88	0,89	0,85	0,71	0,82
8	0,84	0,88	0,89	0,85	0,77	0,81

Tablo 5.4. Congressional Voting veri kümesi için saflık indeksi sonuçları

Bir sonraki aşamada farklı öbek sayılarındaki sonuçlar üzerinden $\Delta^2 IEE$ metriği hesaplanmakta ve ideal öbek sayısının bulunduğu nokta tespit edilmektedir. İdeal öbek sayısı seçilirken $\Delta^2 IEE$ değerinin en yüksek değere ulaştığı nokta belirlenmekte ve bu nokta ideal öbek sayısı olarak seçilmektedir. Hesaplanan delta $\Delta^2 IEE$ değerlerinin yer aldığı sonuçlar Grafik 5.3'de görülebilir.



Grafik 5.3. Congressional Voting veri kümesi için $\Delta^2 IEE$ sonuçları

Grafik üzerinde görülebileceği üzere farklı uygunluk fonksiyonlarının sonuçları için öbek sayının 2 olduğu noktada $\Delta^2 IEE$ değerleri yüksektir. Congressional Voting veri kümesinin ideal öbek sayısı bu doğrultuda önerilen yöntemde 2 olarak bulunmaktadır ve olması gereken değer ile karşılaştırıldığında elde edilen değerin doğru olduğu görülmektedir. Bunun yanında önerilen yöntemde elde edilen sonuçlar orjinal k – modes algoritmasına oranla saflık indeksi açısından daha yüksek bulunmuştur.

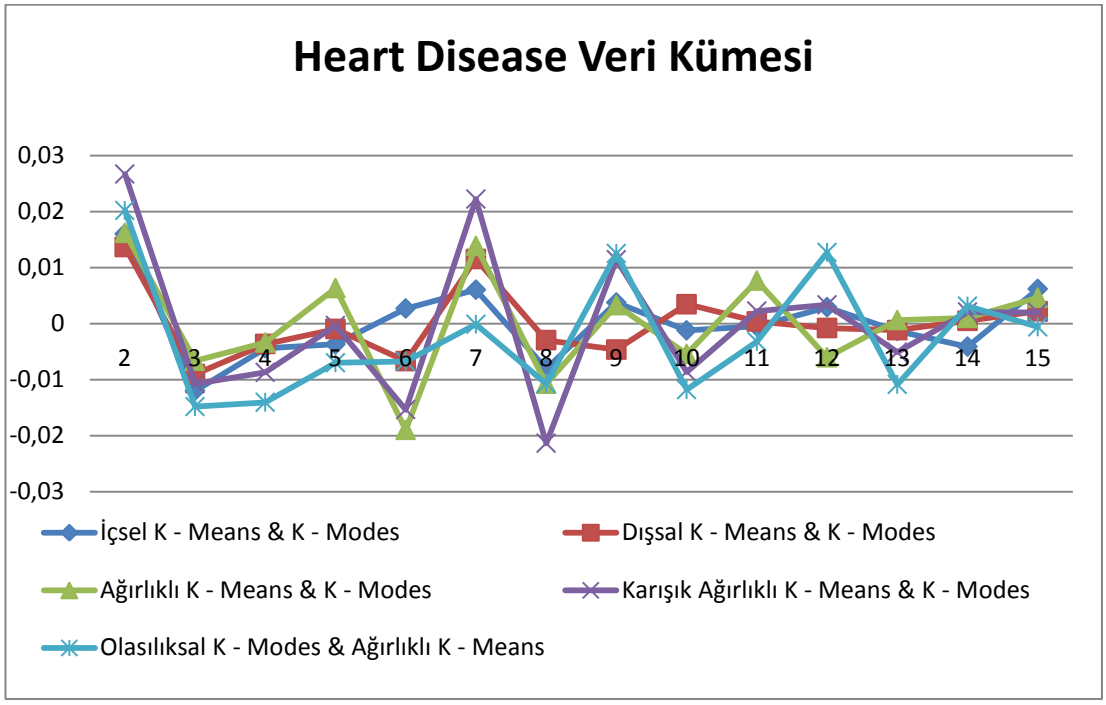
5.4. Heart Disease Veri Kümesi

Heart Disease veri kümesi hem kategorik hem de numerik özniteliklerden oluşan karışık bir veri kümesidir. İçerisinde 2 farklı öbek içerisinde yer alması gereken toplam 270 adet veri bulunmaktadır. Bu veriler 13 farklı öznitelik içermektedir ve içerdiği bu 13 özniteliğin 8 tanesi kategorik diğer 5 tanesi ise numerik değerlerden oluşmaktadır. Bu veri kümesi üzerinde önerilen algoritma k – modes & k – means içsel uzaklık, k – modes & k – means dışsal uzaklık, ağırlıklı k – modes & k – means, hem kategorik hem numerik öznitelikler için olan ağırlıklı karışık ağırlıklı k – modes & k – means, olasılıksal k – modes & ağırlıklı k – means, uygunluk fonksiyonları ile çalıştırılmış ve elde edilen sonuçlar ECOWEB algoritmasının çalıştırılması sonucu elde edilen ve Huang'ın algoritmasının çalıştırılması sonucu elde edilen sonuçlar ile karşılaştırılmıştır. Elde edilen bu saflık indeksi sonuçları Tablo 5.5'de görülebilir.

Heart Disease Veri Kümesi							
Öbek Sayısı	Karışık Ağırlıklı K - Modes & K - Means	Ağırlıklı K- Modes & K - Means	Olasılıksal K - Modes & Ağırlıklı K - Means	İçsel K - Modes & İçsel K - Means	Dışsal K - Modes & Dışsal K - Means	ECOWEB	Huang
2	0,78	0,81	0,84	0,76	0,70	0,74	0,66
3	0,78	0,81	0,85	0,76	0,72	0,74	0,67
4	0,79	0,81	0,85	0,77	0,72	0,75	0,67
5	0,79	0,82	0,87	0,77	0,74	0,75	0,69
6	0,82	0,83	0,87	0,78	0,74	0,76	0,69
7	0,82	0,83	0,87	0,78	0,74	0,79	0,69
8	0,82	0,83	0,87	0,79	0,75	0,79	0,70

Tablo 5.5. Heart Disease veri kümesi için saflık indeksi sonuçları

Bir sonraki aşamada farklı öbek sayılarındaki sonuçlar üzerinden $\Delta^2 IEE$ metriği hesaplanmakta ve ideal öbek sayısının bulunduğu nokta tespit edilmektedir. İdeal öbek sayısı seçilirken $\Delta^2 IEE$ değerinin en yüksek değere ulaştığı nokta belirlenmekte ve bu nokta ideal öbek sayısı olarak seçilmektedir. Hesaplanan delta $\Delta^2 IEE$ değerlerinin yer aldığı sonuçlar Grafik 5.4'de görülebilir.



Grafik 5.4. Heart Disease veri kümesi için $\Delta^2 IEE$ sonuçları

Grafik üzerinde görülebileceği üzere farklı uygunluk fonksiyonlarının sonuçları için öbek sayının 2 olduğu noktada $\Delta^2 IEE$ değerleri yüksektir. Heart Disease veri kümesinin ideal öbek sayısı bu doğrultuda önerilen yöntemde 2 olarak bulunmaktadır ve olması gereken değer ile karşılaştırıldığında elde edilen değerin doğru olduğu görülmektedir. Bunun yanında önerilen yöntemde elde edilen sonuçlar ECOWEB algoritmasına ve Huang'ın algoritmasına oranla saflık indeksi açısından daha yüksek bulunmuştur.

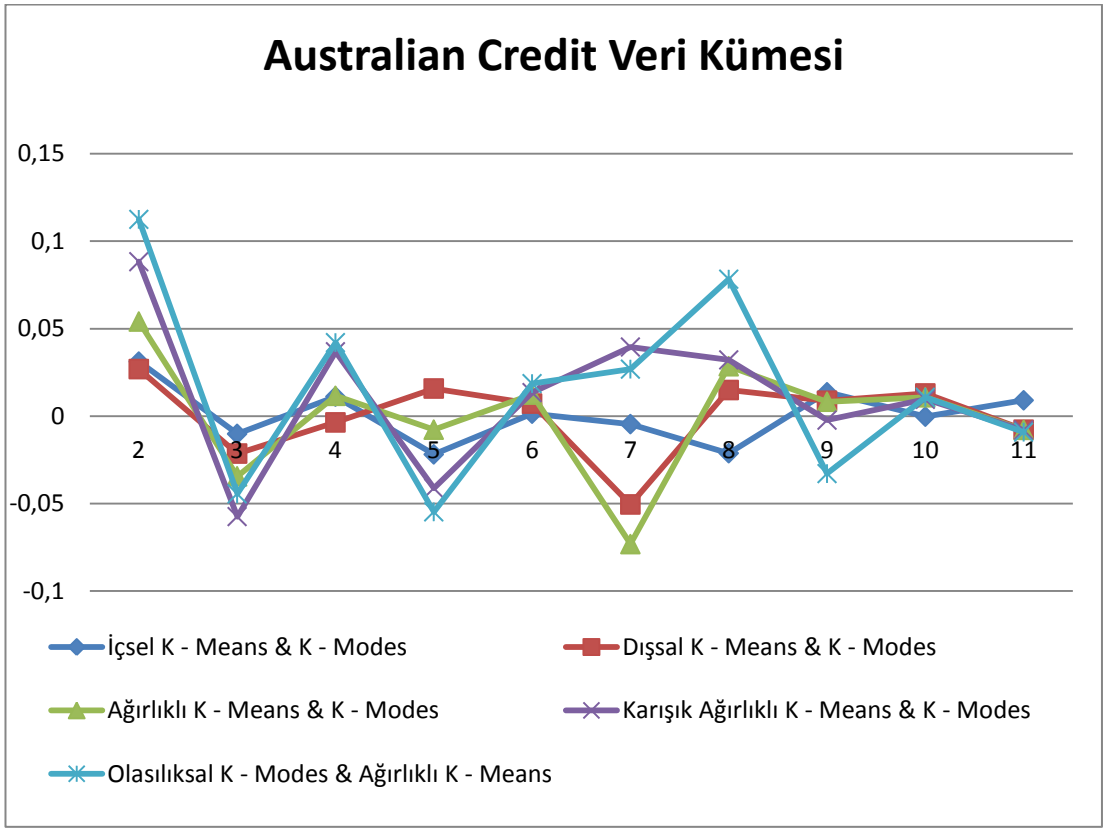
5.5. Australian Credit Veri Kümesi

Australian Credit veri kümesi hem kategorik hem de numerik özniteliklerden oluşan karışık bir veri kümesidir. İçerisinde 2 farklı öbek içerisinde yer alması gereken toplam 690 adet veri bulunmaktadır. Bu veriler 14 farklı öznitelik içermektedir ve içerdiği bu 14 özniteliğin 8 tanesi kategorik diğer 6 tanesi ise numerik değerlerden oluşmaktadır. Bu veri kümesi üzerinde önerilen algoritma k – modes & k – means içsel uzaklık, k – modes & k – means dışsal uzaklık, ağırlıklı k – modes & k – means, hem kategorik hem numerik öznitelikler için olan ağırlıklı karışık ağırlıklı k – modes & k – means, olasılıksal k – modes & ağırlıklı k – means, uygunluk fonksiyonları ile çalıştırılmış ve elde edilen sonuçlar Modha ve Spangler'ın algoritmasının[35] çalıştırılması sonucu elde edilen ve Huang'ın algoritmasının çalıştırılması sonucu elde edilen sonuçlar ile karşılaştırılmıştır. Elde edilen bu saflık indeksi sonuçları Tablo 5.6'de görülebilir.

Australian Credit Veri Kümesi							
Öbek Sayısı	Karışık Ağırlıklı K - Modes & K – Means	Ağırlıklı K- Modes & K - Means	Olasılıksal K - Modes & Ağırlıklı K - Means	İçsel K - Modes & İçsel K - Means	Dışsal K - Modes & Dışsal K - Means	Modha & Spangler	Huang
2	0,81	0,83	0,87	0,81	0,79	0,83	0,78
3	0,81	0,83	0,88	0,81	0,79	0,84	0,78
4	0,82	0,84	0,88	0,81	0,80	0,84	0,80
5	0,82	0,84	0,88	0,81	0,82	0,84	0,81
6	0,83	0,84	0,89	0,82	0,82	0,85	0,81
7	0,84	0,85	0,89	0,82	0,82	0,85	0,81
8	0,85	0,85	0,89	0,83	0,82	0,85	0,82

Tablo 5.6. Australian Credit veri kümesi için saflık indeksi sonuçları

Bir sonraki aşamada farklı öbek sayılarındaki sonuçlar üzerinden $\Delta^2 IEE$ metriği hesaplanmakta ve ideal öbek sayısının bulunduğu nokta tespit edilmektedir. İdeal öbek sayısı seçilirken $\Delta^2 IEE$ değerinin en yüksek değere ulaştığı nokta belirlenmekte ve bu nokta ideal öbek sayısı olarak seçilmektedir. Hesaplanan delta $\Delta^2 IEE$ değerlerinin yer aldığı sonuçlar Grafik 5.5'de görülebilir.



Grafik 5.5. Australian Credit veri kümesi için $\Delta^2 IEE$ sonuçları

Grafik üzerinde görülebileceği üzere farklı uygunluk fonksiyonlarının sonuçları için öbek sayının 2 olduğu noktada $\Delta^2 IEE$ değerleri yüksektir. Australian Credit veri kümesinin ideal öbek sayısı bu doğrultuda önerilen yöntemde 2 olarak bulunmaktadır ve olması gereken değer ile karşılaştırıldığında elde edilen değerin doğru olduğu görülmektedir. Bunun yanında önerilen yöntemde elde edilen sonuçlar Modha ve Spangler'ın algoritmasına ve Huang'ın algoritmasına oranla saflık indeksi açısından daha yüksek bulunmuştur.

BÖLÜM 6

6. SONUÇ

Veri madenciliği günümüzde bilgisayar bilimlerinin en önemli konularından biri haline gelmiştir ve bununla birlikte verilerin öbeklenmesi ve bu öbekler üzerinden sonuçlara varılması işlemi de önemini arttırmıştır. Bu doğrultuda öbekleme algoritmarı üzerine yapılan çalışmalarda oldukça fazladır. Öbekleme işleminin bu oranda önemini arttırmış olması ve öbekleme işleminin güvenilir ve hızlı bir şekilde yapılmak istenmesi bu tezde yapılan çalışmanın amacının temelini oluşturmaktadır.

Bu tezde yapılan çalışma, içerisinde hem numerik hem de kategorik özniteliklere sahip olan verileri barındıran veri kümelerinin öbeklenmesi işlemidir. Sadece numerik özniteliklere sahip verilerden oluşan veri kümelerinin veya sadece kategorik özniteliklere sahip verilerden oluşan veri kümelerinin öbeklenmesi için bir çok algoritma önerilmiştir. Ancak hem kategorik hem de numerik özniteliklere sahip verileri içeren veri kümelerinin öbeklenmesi için çok fazla yöntem ortaya konmamıştır. Bu tezde önerilen algoritmanın bu alanda başarılı çalışan bir algoritma olması amaçlanmıştır.

Önerilen algoritmanın temelinde genetik algoritma kullanılmış ve bu doğrultuda hem hızlı hem de iyi sonuçları üretebilecek bir algoritma ortaya konulmuştur. Algoritma içerisinde öbekleme işlemi yapılırken veriler arasındaki mesafeyi hesaplamaya yönelik bir çok uygunluk fonksiyonu tanımlanmıştır ve bu uygunluk fonksiyonlarının sonuçları çok amaçlı yapı üzerinde kullanılarak seçim işlemleri yapılmıştır. Algoritmanın öbekleme işlemi yapmasının yanında kullanıcıdan bağımsız olarak veri kümesinin ideal olarak kaç öbeğe bölünmesi gerektiği kararını vermesi de sağlanmıştır. Bunun içinde öbekler içi değişimden yola çıkarak bir mesafe hesaplayan bir yöntem önerilmiştir.

Algoritmanın başarımını ölçmek adına popüler ve bir çok çalışmada kullanılmış olan veri kümeleri arasından 5 adet veri kümesi seçilmiş ve seçilen bu veri kümeleri üzerinde önerilen algoritma ile çeşitli uygunluk fonksiyonları doğrultusunda

öbekleme işlemleri yapılmış ve elde edilen sonuçlar üzerinden de uygun öbek sayısı tespit edilmeye çalışılmıştır. Üzerinde öbekleme işlemi yapılan veri kümeleri aynı zamanda karşılaştırma yapılacak olan algoritmalarla da çalıştırılmış ve bu algoritmalar için de sonuçlar alınmıştır. Bir sonraki aşama da elde edilen sonuçların başarımı saflık indeksi adı verilen bir metrik ile ölçülmüş ve önerilen yöntem ile diğer yöntemlerin başarımları bu indeks doğrultusunda karşılaştırılmıştır.

Elde edilen sonuçlara bakıldığında önerilen algoritmanın hem sadece numerik öznitelikler içeren verilere sahip olan veri kümeleri üzerinde hem sadece kategorik öznitelikler içeren verilere sahip olan veri kümeleri üzerinde hem de numerik ve kategorik öznitelikleri bir arada içeren verilere sahip olan veri kümeleri üzerinde başarılı bir şekilde çalıştığı, hem iyi bir öbekleme sonucu verdiğini hem de uygun sayıdaki öbek sayısını otomatik bir biçimde tespit edebildiğini, gözlemlenmiştir.

Kısaca bu tez çalışmasının amacına ulaştığı ve karışık verilerin öbeklenmesi ile ilgili konuda literatüre önemli katkılar yapıldığı söylenebilir. Zaten bu tez ile alakalı saygın bir dergide[1] ve uluslararası bir konferansta[2] yayınlanmış olan çalışmalar bu durumun en güzel örneğidir.

[1] Journal of Universal Computer Science
[2] ASONAM 2011

KAYNAKLAR

[1] Srinivas, N., Deb, K.: Multi-Objective Function Optimization Using NonDominated Sorting Genetic Algorithms, Evolutionary Computation, 2(3), 1995, 221–248.

[2] Usaman Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth : The KDD process for extracting useful knowledge from volumes of data : Magazine Communications of the ACM Volume 39 Issue 11, Ekim 1996 Sayfa 27 – 34

[3] Jain, A.K., Murty, M.N., Flynn., P.J., Data clustering: A review. CM Computing Surveys (CSUR), 31(3), 264-323, Temmuz 1999.

[4] Duda, R., Hart P., Pattern Classification and Scene Analysis, John Wiley and Sons, New York, 1973.

[5] Jain, A.K. , Dubes, R.C., Algorithms for Clustering Data, Prentice Hall, Englewood Cliff, New Jersey, 1988.

[6] Kaufman, L., Rousseeuw, P.J., Clustering by means of Medoids, in Statistical Data Analysis Based on the L_1–Norm and Related Methods, edited by Y. Dodge, North-Holland, 1987, sayfa 405 – 416.

[7] Ng, R., Han, J., Efficient and effective clustering method for spatial data mining, in: Proceedings of the 20th International Conference on Very Large Data Bases, Santiago, Chile, 1994, sayfa 144 – 155.

[8] J.C. Dunn, Some recent investigations of a new fuzzy partitional algorithm and its application to pattern classification problems, Journal of Cybernetics 4, 1974, sayfa 1 – 15.

[9] Bezdek, J.C., Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, 1981, New York.

[10] Fisher, D.H., Knowledge acquisition via incremental conceptual clustering, Machine Learning 2 (2), 1987, sayfa 139–172.

[11] Lebowitz, M., Experiments with incremental concept formation, Machine Learning 2 (2), 1987, sayfa 103–138.

[12] Gluck, M., Corter, J., Information, uncertainty, and the utility of categories, in: Proceedings of Seventh Annual Conference in Cognitive Society, 1985, sayfa 283–287.

[13] Huang, Z., Ng, M.K., A fuzzy k-modes algorithm for clustering categorical data, IEEE Transactions on Fuzzy Systems 7 (4) ,1999, sayfa 446–452.

[14] McKusick, K., Thomson, K., COBWEB/3: A portable implementation, Technical Report FIA-90-6-18-2, NASA Ames Research Center, 1990.

[15] Reich, Y., Fenves, S.J., The formation and use of abstract concepts in design, in: D.H. Fisher, M.J. Pazzani, P. Langley (Eds.), Concept Formation:

Knowledge and Experience in Unsupervised Learning, Morgan Kaufman, Los Altos, Calif, 1991, sayfa 323 – 352.

[16] Biswas, G., Weingberg, J., Fisher, D.H., ITERATE: A conceptual clustering algorithm for data mining, IEEE Transactions on Systems, Man, and Cybernetics 28C, 1998, sayfa 219–230.

[17] Cheesman, P., Stutz, J., Bayesian classification (AUTO-CLASS): Theory and results, Advances in Knowledge Discovery and Data Mining, 1995.

[18] Bayes, Thomas, and Price, Richard, "An Essay towards solving a Problem in the Doctrine of Chance. By the late Rev. Mr. Bayes, communicated by Mr. Price, in a letter to John Canton, M. A. and F. R. S.". Philosophical Transactions of the Royal Society of London 53 (0), 1763, sayfa 370–418.

[19] Guha, S., Rastogi, R., Kyuseok, S., ROCK: A robust clustering algorithm for categorical attributes, in: Proceedings of 15th International Conference on Data Engineering, Sydney, Australia, 23 – 26 Mart 1999, sayfa 512–521.

[20] Ganti, V., Gekhre, J.E., Ramakrishnan, R., CACTUS-clustering categorical data using summaries, in: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1999, sayfa 73–83.

[21] Zhang, T., Ramakrishnan, R., Livny, M., BIRCH: An efficient data clustering method for very large databases, in: SIGMOD Conference, 1996, sayfa 103–114.

[22] Huang, Z., Clustering large data sets with mixed numeric and categorical values, in: Proceedings of the First Pacific-Asia Conference on Knowledge Discovery and Data Mining, World Scientific, Singapore, 1997.

[23] Eiben, A. E. et al, "Genetic algorithms with multi-parent recombination". PPSN III: Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: 78–87. ISBN 3-540-58484-6, 1994.

[24] Koza, John, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press. ISBN 0-262-11170-5, 1992.

[25] Kenneth, A., De Jong, William, M., Spears: An Analysis of the Interacting Roles of Population Size and Crossover in Genetic Algorithms. PPSN 1990: sayfa 38-47.

[26] Zitzler, E., Evolutionary Algorithms for Multi-Objective Optimization: Methods and Applications, PhD thesis, Zurich: Swiss Federal Institute of Technology (ETH), Aachen, Germany, 1999.

[27] Fudenberg, D., Tirole, J., Game Theory. MIT Press. Chapter 1, Section 2.4. ISBN 0-262-06141-4, 1983.

[28] Zhexue Huang, A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining, In Research Issues on Data Mining and Knowledge Discovery, 1-8, 1997

[29] Zhexue Huang. 1998. Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values. Data Min. Knowl. Discov. 2, 3 Eylül 1998, sayfa 283-30

[30] Ahmad, L. and Dey, A., "K-Mean Clustering Algorithm for Mixed Numeric and Categorical Data," Data & Knowledge Engineering, 63(2), 2007, sayfa 503 – 527

[31] S.Aranganayagi and K.Thangavel, Improved K-Modes for Categorical Clustering Using Weighted Dissimilarity Measure, International Journal of Engineering and Mathematical Sciences 5:2, 2009.

[32] Z. Huang, M. Ng, H. Rong and Z. Li, Automated Variable Weighting in K-Means Type Clustering, IEEE Transactions on Pattern Analysis and Machine Intelligence, V27, 2005, sayfa 657-668

[33] Yiming Yang, An evaluation of statistical approaches to text categorization, Journal of Information Retrieval 1 (1–2), 1999, sayfa 67–88.

[34] Chen, K. and Liu L., "Towards Finding Optimal Partitions of Categorical Datasets", Technical Report, 2003

[35] D.S. Modha, W.S. Spangler, Feature weighting in k-mean clustering, Machine Learning 52 (3), 2003, sayfa 217–237.

[36] Ozyer, T., Alhajj, R.: Deciding on Number of Clusters by Multi-Objective Optimization and Validity Analysis, Journal of Multi-Valued Logic and Soft Computing 14(3), 2008, 457-474.

[37] Ozyer, T., Alhajj, R.: Parallel Clustering of High Dimensional Data by Integrating Multi-Objective Genetic Algorithm with Divide and Conquer, Appl. Intell. 31(3), 2009, 318-331.

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, Adı	:	SERT, Onur Can
Uyruğu	:	T.C.
Doğum tarihi ve yeri	:	29.05.1988 Ankara
Medeni hali	:	Bekâr
Telefon	:	0 (536) 443 29 34
Email	:	ocsert@etu.edu.tr onurcansert@gmail.com

Eğitim

Derece	**Eğitim Yeri**	**Mezuniyet Tarihi**
Y. Lisans	TOBB ETÜ Bilgisayar Mühendisliği	2012 (beklenen)
Lisans	TOBB ETÜ Bilgisayar Mühendisliği	2010

İş Deneyimi		**Görev**
2010 – 2012	TOBB ETÜ	Ders ve Araştırma Asistanlığı
2010 – 2010	Kasırga Bilişim Elektronik Ltd.	Yazılım Programlama
2009 – 2009	Yüce Bilgi Sistemleri	Yazılım Programlama
2008 – 2008	ETC – IS	Yazılım Programlama

Yabancı Dil

İngilizce (ileri seviye)

Yayınlar

1. Onur Can Sert, Kayhan Dursun, Tansel Özyer, Jamal Jida, Reda Alhajj: The Unification and Assessment of Multi-Objective Clustering Results of Categorical Datasets with H-Confidence Metric. J. UCS 18(4): 507-531 (2012)

2. Onur C. Sert, Kayhan Dursun, Tansel Özyer: Ensemble of Multi-Objective Clustering Unified With H-Confidence Metric as Validity Metric, IEEE ASONAM 2011, Advances in Social Network Analysis and Mining