T.C.

İSTANBUL AYDIN ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Birkaç Veri Kümesi ile WEKA ve MATLAB Üzerinde Kümeleme Algoritmalarının Karşılaştırılarak İncelenmesi

YÜKSEK LİSANS TEZİ

Mustafa TAKAOĞLU

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

AĞUSTOS - 2016

T.C.

İSTANBUL AYDIN ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Birkaç Veri Kümesi ile WEKA ve MATLAB Üzerinde Kümeleme Algoritmalarının Karşılaştırılarak İncelenmesi

YÜKSEK LİSANS TEZİ

Mustafa TAKAOĞLU

(Y1413.010029)

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Zafer ASLAN

Ağustos, 2016

T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ MÜDÜRLÜĞÜ

Yüksek Lisans Tez Onay Belgesi

Enstitümüz Bilgisayar Mühendisliği Ana Bilim Dalı Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı **Y1413.010029** numaralı öğrencisi **Mustafa TAKAOĞLU**'nun **"BİRKAÇ VERİ KÜMESİ İLE WEKA VE MATLAB ÜZERİNDE KÜMELEME ALGORİTMALARININ KARŞILAŞTIRILARAK İNCELENMESİ"** adlı tez çalışması Enstitümüz Yönetim Kurulunun 19.07.2016 tarih ve 2016/19 sayılı kararıyla oluşturulan jüri tarafından .Oybirliği. ile Tezli Yüksek Lisans tezi olarak .Kabul. edilmiştir.

Öğretim Üyesi Adı Soyadı	**İmzası**
Tez Savunma Tarihi :11.08.2016	
1)Tez Danışmanı: Prof. Dr. Zafer ASLAN	.....
2) Jüri Üyesi : Yrd. Doç. Dr. Metin ZONTUL	.....
3) Jüri Üyesi : Yrd. Doç. Dr. Ferdi SÖNMEZ	.....

Not: Öğrencinin Tez savunmasında **Başarılı** olması halinde bu form **imzalanacaktır**. Aksi halde geçersizdir.

YEMİN METNİ

Yüksek Lisans Tezi olarak sunduğum "Birkaç Veri Kümesi ile Weka ve Matlab Üzerinde Kümeleme Algoritmalarının Karşılaştırılarak İncelenmesi" adlı çalışmanın, tezin proje safhasından sonuçlanmasına kadarki bütün süreçlerde bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurmaksızın yazıldığını ve yararlandığım eserlerin Bibliyografya'da gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve onurumla beyan ederim. (10/08/2016)

Mustafa TAKAOĞLU

Aileme ve Sevdiklerime,

ÖNSÖZ

Günümüz teknolojisinin doğurmuş olduğu kaçınılmaz sonuçlardan biri de büyük veri yığınlarıdır. Facebook, Twitter ve Instagram gibi son zamanların trend uygulamalarında bırakın günlük veri akışını, anlık hesaplanmış veri akışının büyüklüğü dahi her geçen gün tarifi zor bir aşamaya ulaşmaktadır. Ülkemizde ise bu tarz küresel ölçekli olmasada, insanların düzenli olarak kullandıkları Sahibinden, Gittigidiyor, YemekSepeti, Ekşisözlük ve nice web sitesinde hatırı sayılır büyüklükte veri akışı ve kayıdı tutulmaktadır. Peki bu verilerin arka planda mühendislere, firma yöneticilerine ve ilgili kişilere verilerin saklanılması için harcanılan büyük rakamlar dışında nasıl bir faydası olabilir? İşte bu aşamada, alışılageldik istatistik işlemleri dışında biraz da yeni sayılabilecek "Veri Analistliği ve Veri Madenciliği" kavramları karşımıza çıkıyor. Veri Analistleri bahsetmiş olduğum çeşitli veri yığınları üzerinde ORANGE, WEKA, R, RattleGUI, MATLAB gibi programlarla kümeleme, sınıflandırma, genetik algoritmalar gibi bazı teknikler kullanılarak veri madenciliği ile faydalı olarak nitelendirebileceğimiz sonuçlar elde ederler. Örneğin bir e-ticaret sitesinde çalışan veri analisti, havanın yağmurlu olduğu günlerde satılan ürünleri dikte edip, haftalık hava durumuna göre, vitrinde gösterilecek ürünlerin kolayca satılması düşünülen ürünlerden belirlemesi ile cironun artırılmasını sağlayabilir. İşte bu gelişmeler ışığında yüksek lisans eğitimimde üzerinde durmak istediğim alan olarak "Veri Madenciliği" ile ilgili bir konu belirledim. Bu sebeple çok tercih edilen veri madenciliği uygulamaları olan WEKA ve MATLAB üzerinde kümeleme algoritmalarını denemek istedim. Bu doğrultuda bana yardımlarını esirgemeyen başta Sayın Prof. Dr. Zafer ASLAN olmak üzere, Boğaziçi Üniversitesi Kandilli Rasathanesi ve Deprem Araştırma Enstitüsü'ne, her türlü desteğini benden esirgemeyen kıymetli aileme ve kardeşim Faruk TAKAOĞLU'na teşekkürlerimi borç bilirim.

Ağustos, 2016 Mustafa TAKAOĞLU
Bilgisayar Mühendisi

İÇİNDEKİLER

Sayfa

KISALTMALAR

AGNES	**:** Aglomeratif Yerleştirme
BIRADS	**:** Göğüs Görüntüleme, Raporlama ve Veri Sistemleri
C	**:** C Programlama Dili
CAD	**:** Bilgisayar Destekli Teşhis
DBSCAN	**:** Yoğunluk Tabanlı Gürültülü Mekansal Kümeleme Algoritmaları
DDSM	**:** Mamografi Taraması için Dijital Veritabanı
DIANA	**:** Cihaz Analizi
DWT	**:** Ayrık Dalgacık Dönüşümü
EM	**:** Beklenti Maksimasyonu
FOS	**:** Birinci Dereceden İstatistik
IRM	**:** Entegre Bölge Eşleştirme
KDD	**:** Veritabanları Bilgi Keşfi
KHD	**:** Kalp Hızı Değişkenliği
KKY	**:** Konjestif Kalp Yetmezliği
RFID	**:** Radyo Frekansı Tanımlama
STHDA	**:** Yüksek Verimli Veri Analizi için İstatistiksel Araçlar
WEKA	**:** Bilgi Analizi için Waikato Ortamı
MATLAB	**:** Matris Laboratuarı

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

BİRKAÇ VERİ KÜMESİ İLE WEKA VE MATLAB ÜZERİNDE KÜMELEME ALGORİTMALARININ KARŞILAŞTIRILARAK İNCELENMESİ

ÖZET

Günümüzde teknoloji takip edilmesi zor bir hızla ilerlemektedir. Bu ilerlemenin bir sonucu olarak teknolojiden direkt veya dolaylı olarak etkilenen sektörlerde birçok yeni iş alanları ve alt sektörleri oluşmuştur. Bilişim teknolojilerinin alt dallarından biri olan veri tabanı sistemleri de bu teknolojik gelişimden etkilenmiş ve kendi içerisinde yeni teknolojik dallara sahip olmuştur. Veri tabanı analizi veya veri madenciliği buna örnektir. Önceki zamanlarda verilerin önemli olanları saklanmakta ve geri kalanı oluşturdukları ek depolama maliyetleri yüzünden kullanılmamakta iken, gelişen yapay zeka ve bilimsel esnek hesaplama yöntemleri ile bu önemsiz gibi gözüken veriler ciddi önem kazanmıştır. Veri madenciliği üzerinde bilgi sahibi olan bireyler çalıştıkları sektörler ile paralel olarak; geleceğe yönelik çeşitli tahminler, firmalarının içinde bulundukları durumun anlık incelenmesi, sosyal medya verilerine göre müşteri memnuniyeti ve bunu arttırmak için yapılması gerekenler gibi birçok alanda başarılı sonuçlar elde edebilirler. Daha çok tahmin ve analiz işlemlerinin önem kazandığı bu günlerde, tez çalışmamızda bu işlemlerin kullanıldığı kümleme algoritmaları ele alınmıştır. Bu tez çalışmasındaki amacımız K-Means kümeleme algoritması, Expectation Maximization kümeleme algoritması ve Hiyerarşik kümeleme algoritmları üzerinde derinlemesine bilgi sahibi olmak ve edindiğimiz bilgileri uygun yazılım platformları üzerinde denemektir. Bu amaçla, Boğaziçi Üniversitesi Kandilli Rasathanesi ve Deprem Araştırma Enstitüsünden alınan iklim verileri anlatılan kümeleme algoritmaları üzerinde denenmiştir. Kümeleme algoritmalarının herbiri MATLAB ve WEKA programları üzerinde uygulanmıştır. Elde edilen sonuçlar üzerinden kullanmış olduğumuz programlar ve algoritmaların karşılaştırılmaları yapılmıştır. Son olarak MATLAB ve WEKA kullanımlarının avantajları ve dezavantajlarından bahsedilmiştir.

Anahtar Kelimeler : *WEKA, MATLAB, Kümeleme Algoritmaları.*

USING COUPLE OF DATASETS, ANALYSIS BY COMPARING CLUSTERING ALGORITHMS BETWEEN WEKA AND MATLAB PLATFORMS

ABSTRACT

Nowadays, it is impossible to follow technological developments because of it's rapidly growing trend. As a result of this trend, new branches of business sectors appear. One of the sub branch of technology area which is called as database systems faced with same situation. It is effected by this technological growing trend. Data mining sector had been derivated like this way. Former database systems only cares personal and meta datas and see the other datas as a weight. But today, with the help of artifical intelligence and scientific flexible calculation ways data stacks has become more important than ever. The people who become experienced about data mining and it's usage, can use their abilities in parallel business sectors.For instance, finance sector personnels can perform their datamining skills on predicting the future of their special area. On the other hand, company can measure gladness of their customers according to social media responses. Business men can predict or estimate their current and future position in the global and local market. As a summary, expectation of the employees which includes tech workers and business specialists are using data mining solutions for prediction. And prediction job has always links with clustering algorithms. That's why I choose comparing clustering algorithms over some Boğaziçi University Kandilli Observatory and Earthquake Research Institute values of Turkey as a master thesis of mine. For accomplish this task, I used Matlab and Weka platforms as computer programmes. I choosed mostly used clustering algorithms which are K-Means, Expectation Maximization and Hierarchical Clustering algorithms to compare with each other. With the help of these comparisons, I would like to be experienced about clustering algorithms and their mostly used platforms. Firstly, I defined each algorithm on both platforms and than I opportunity to compare them with each other according to their advantages and disadvantages.

Keywords : *WEKA, MATLAB, Clustering Algorithms.*

1. GİRİŞ

1.1 Çalışma Konusu

Yüksek lisans tezinde çalışma konusu, üç farklı kavram kombine edilerek elde edilmiştir. Kümeleme algoritmaları, WEKA ve MATLAB bu doğrultuda ele alınmıştır. Ayrıca tezimizden üretilecek makalede, MATLAB'ın bir uzantısı olan Wavelet Toolbox ile elde ettiğimiz bazı sonuçlar kullanılarak, oluşan kümelerde verilerin incelenmesi ve yorumlanması amaçlanmış, bu sebeple çalışmanın konusunda yer almıştır. Ayrıca yüksek lisans tezinde yapılan ayrıntılı araştırmalar sonucu algoritmanın tarihi, gelişimi, çeşitleri ve özellikle üzerinde durduğumuz kümeleme algoritmaları ayrıntılı bir biçimde araştırılmış ve açıklanmıştır. Veri Madenciliği, WEKA, MATLAB ve MATLAB Wavelet Toolbox hakkında, kümeleme algoritmalarının kullanımı dışında, birçok öğretici bilgi verilmiştir. Ayrıca çalışmada kullanılan veriler büyük önem arz etmektedir. Çünkü kümeleme algoritmalarının uygulanması sonrası elde edilen sonuçlar, yukarıda belirtildiği üzere hazırlanmakta olan makalede bu sonuçların yorumlanmaları durumunda, iklimsel çıkarımlar yapılmasında önemli sonuçlar elde edilmesinde büyük fayda sağlamıştır.

1.2 Tezin Amacı

Veri madenciliğine duyulan yoğun ilgi sebebiyle yüksek lisans tezinde, veri madenciliği konusuyla alakalı bir alanda çalışma amaçlanmıştır. Bu doğrultuda öncelikle veri madenciliğinin ne olduğu ve hangi platformlar kullanılarak geliştirildiği araştırılmıştır. Yapılan araştırmalar sonucunda hazırlanacak yüksek lisans tezinin teknik bir konu üzerinde yapılmasına kadar verilmiştir.

Literatür araştırmaları sonucu WEKA ve MATLAB platformlarında belirlenmiş kümeleme algoritmalarıyla, kümeleme işleminin yapılıp, elde edilen sonuçların karşılaştırılması ve Wavelet Toolbox ile yorumlanması amaçlanmıştır.

Bu sebeple kullanılan veri kümelerinin iklimsel değerlerden oluşmasına önem verilmiş ve tezimizden türetilecek makalede Wavelet'de alınan sonuçların yorumlanması ışığında elde edilen iklimsel çıkarımların açıklanması amaçlanmıştır.

1.3 Literatür Araştırması

Aşağıda yaptığımız literatür araştırması sonucu elde ettiğimiz makaleler başlıklar halinde verilmiştir. Literatür araştırmasında belirlenen makalelerin incelenmesi sonucunda; yüksek lisans tezimizde faydalandığımız yada konsept olarak alakalı bulduğumuz tezlerin önemli noktaları açıklanmıştır.

Classification of the different thickness of oil firm based on wavelet transform spectrum information (Farklı kalınlıktaki petrol firmasının dalgacık dönüşümü spektrum bilgisine bağlı olarak sınıflandırılması)

Çinli bilim adamları bu makalelerinde petrol sıvısının akışkanlık kalınlığının üzerinde Wavelet dalgacık dönüşümü ile araştırma yapmışlardır. Alışıla geldik yöntemlerin dezavantajlarını göz önünde bulundurarak, daha avantajlı bir sınıflandırma yöntemi sunabilmek amacıyla yapılan bu çalışmada daha tutarlı sonuçlara ulaşmışlardır. Bu doğrultuda sınıflandırma metodlarını Wavelet dalgacık dönüşümünü spektral verilerine dayandırarak yapmışlardır. Bu yöntemde tamamen entegre edilmiş hiperspektral verilerin güçlü spektral süreklilikleri ve zaman frekansları ile wavelet dönüşümü kullanılarak olumlu sonuçlar elde edilmiştir. Bu makalede kendi tezimizle alakalı ve faydalı gördüğümüz nokta; wavelet sınıflandırmasının güzel bir örneğinin bulunmasıdır. Ayrıca makale içerisinde waveletin kullanım şekli ve özellikleri hakkında da çok güzel bilgilere ulaşılmıştır (Bin, Chen, Dongmei, Jianyong, Shouchang, Yajie ve Yi, 2015).

Performance analysis of discrete wavelet transform based first-order statistical texture features for hardwood species classification (Ayrık dalgacık dönüşümü ile birinci dereceden istatiksel ahşap türlerinin doku özelliklerine göre sınıflandırmanın performans analizi)

Bu araştırma direkt olarak Wavelet dönüşümü aşamaları hakkında temel bilgiler vermekte olup. Çok başarılı bir çalışmadır. Hindu bilim adamları makalelerinin özet bölümünde araştırmalarını şu şekilde ifade etmişlerdir. Basit ve verimli bir yöntem olan

DWT, Discrete Wavelet Transform, yani ayrık dalgacık dönüşümünü birinci dereceden istatistiksel dokuyu FOS, First-Order Statistical esas alarak ahşap türlerinin mikroskobik görüntülerini sınıflandırmaya çalışmışlardır. 25 çeşit 500 adet mikroskobik ahşap fotografı kullanılarak bu araştırma yapılmıştır. Her görüntü db1- db10 aralığındaki filtrelerden kullanılarak analiz edilmiş ve sınıflandırma yapılmıştır. Elde edilen sonuçlara göre kullanılan veriler doğrultusunda db3 kaynaklı analiz en verimli sonuçları vermiş, 96.80% oranında, ve elde edilen istatistiksel sonuçlar (0-1) aralığında olmuştur (Anand, Dewal, Gupta ve Yadav, 2015).

Wavelet packet texture descriptors based four-class birads breast tissue density classification (Birads göğüs dokusu sınıflandırmasına göre dalgacık paket doku tanımlayıcıları)

Bu araştırmalarında Hindu bilim adamları meme kanseri ve türevi hastalıkların teşhisinde kullanılan CAD sistemleri yani bilgisayar destekli teşhis sistemlerine alternatif yada daha başarılı bir yöntem sunmak amacıyla Wavelet sınıflandırma algoritmaları kullanarak DDSM, Digital Database for Screening Mammography, verileri ile kıyaslama yapmaktadırlar. Burada karşılaştırdıkları CAD sistemi dördüncü derece BIRADS, Breast Imaging Reporting and Data Systems, standartları meme dokusu yoğunluğu sınıflandırması esas alınacaktır. Araştırma toplamda 480 mamogram ile yapılmıştır. Her veri için mean, standard deviation ve energy descriptorları hesaplanmıştır. Ve mesleki bilgileri doğrultusunda gerekli karşılaştırmaları yapmışlardır. Radyologların meme dokusu yoğunluğundaki olumsuz değişiklikleri erken teşhis edebilmesi çok önemlidir. Bu doğrultuda tedavinin şekli ve süresi daha sağlıklı belirlenir. Bu araştırma sonucunda ise Wavelet sınıflandırması ile yapılan işlemler çok başarılı sonuçlar ortaya koymuştur. Toplamda 73.7% oranında bir sınıflandırmada başarı görülmüştür. Ayrıca kullanılan sistemler 75%, 68.3%, 55%, ve 96.6%'lık BIRADS-I için kazanımlar elde etmiştir. BIRADS-II, BIRADS-III ve BIRADS-IV sınıflarında da benzer kazanımlar elde edilmiştir (Bhadauria, Kumar ve Virmani, 2015).

Weka yazılımında k-ortalama algoritması kullanılarak konjestif kalp yetmezliği hastalığı teşhisi

Bu araştırmalarında Türk bilimadamları WEKA veri madenciliği uygulaması ile bir kümeleme algoritması olan K-Means algoritmasını kullanarak konjestif kalp yetmezliği olan kişilerin teşhisi üzerine çalışmışlardır. Direkt olarak makalemizde üzerinde çalıştığımız bir konu olması sebebiyle çalışmalarımız için çok faydalı bir makaledir. Araştırmacılar bu doğrultuda istedikleri teşhisi koyabilmeleri için KHD verilerini WEKA'da K-Means algoritmasına sokmuş ve sonuçları incelemişlerdir. KHD verilerini 29 adet KKY hastası ve 54 adet kontrol grubunu oluşturan kişiden almışlardır. Sonuç olarak 4 küme kullanarak yaptıkları araştırmada 98.79% oranında başarı elde etmişlerdir. Ayrıca makalelerine WEKA hakkında da bazı faydalı bilgiler paylaşılmıştır (İşler ve Narin, 2012).

Scalable integrated region-based image retrieval using IRM and statistical clustering (IRM ve istatistiksel kümeleme kullanılarak ölçeklenebilir entegre bölge tabanlı görüntü alma)

Bilim adamları Wavelet kullandıkları makalelerinin başında, istatistiksel kümeleme algoritması ölçeklenebilir görüntü alma sistemlerinin tasarımında önemlidir demişlerdir. Bu sebeple de çalışmalarında bölge segmentasyonuna dayalı indeksleme ve görüntü almak için ölçeklendirilebilir bir algoritma sunmuşlardır. Methodları istatistiksel kümeleme kullanır ve bölge özellikleri ve IRM, Entegre Bölge Eşleştirme, arasındaki genel benzerlikleri değerlendirmek için geliştirmişlerdir. Makalelerinde ayrıntılı bir şekilde yapılan çalışmalar açıklanmıştır. Araştırmalarının sonucunda 200.000 resim kullanılarak yüksek verimlilik ve algoritmada sağlamlık elde edilmiştir. Kümeleme verimliliğinin daha Spiyi bir istatistik kümeleme algoritması kullanılarak artırılabileceğini söylemektedirler. Daha iyi bir modelleme ve eşleştirme şeması ile elde edilecek doğruluk oranı artırılacaktır denilmektedir. Ayrıca bu araştırmalarını ilerleyen çalışmalarında biomedical ve multimedia verileri üzerinde de test etmeyi planlamaktadırlar (Du ve Wang, 2001).

A New wavelet-median-moment based method for multioriented video text detection (Çok odaklı bir video metin tespiti için yeni bir dalgacık medyan moment tabanlı yöntem)

Singapur ve Hindistan ortak çalışması olan bu araştırmada bilim adamları yeni bir method geliştirmeye çalışmışlardır. Benim bu makaleyi faydalı bulmamızın sebebi içerisinde Wavelet ve K-Means kümeleme algoritmalarından faydalanılmasıdır. Araştırmacılar kesin olarak başarılı diye adlandırılacak bir kazanım elde edememiş olsadalarda, edinilen seviyede fark edilen eksiklikler giderilebilir durmaktadır. Bu sebeple gelecek vaadeden kıymetli bir çalışmadır. Burada yeni bir Wavelet median moments özelliği geliştirip videolardan text detection dediğimiz işlemi yapmaya çalışılmıştır (Dutta, Pal, Shivakumara ve Tan, 2010).

WaveCluster with differential privacy (WaveCluster diferansiyel gizlilik)

Bu makalede bilim adamları WaveCluster'ın grid tabanlı kümeleme algoritmalarının önemli bir parçası olduğunu ve serbest çizilmiş bir şekil kümesi bulmada yeterli olduğunu belirtmektedirler. Bu sebeple çalışmalarında yeni ve genel geçerliliği olan bir Wavelet dönüşüm tekniği geliştirmek ve bunu yaparkende WaveCluster algoritmasının diferensiyel bütünlüğünüde sağlamaya özen göstermektedirler. Bu sebeple de 4 küme üzerinde 2 tekniğe odaklanmışlardır ve bu doğrultuda araştırmalarını yapmışladır. Teknik anlamda çok değerli olan bu çalışma, kendi tezimizin oluşturulmasında çok faydalı olmuştur. Wavelet ve kümeleme algoritmalarının incelenmesi noktasında sağlamış olduğu farkındalık literatür taramasında bu makaleyi seçmemizin başlıca sebebidir (Chen, Chirkova ve Yu, 2015).

Applicability of data mining algorithms for recommendation system in e-learning (E-öğrenme öneri sistemi için veri madenciliği algoritmaları uygulanabilirliği)

Makalelerinde öğrencilere online eğitim sistemlerinde ders tavsitesinde bulunacak bir veri madenciliği algoritması üzerine çalışmışlardır. K-Means kümelemesi ve Apriori algoritması birleştirilerek Java'da bir algoritma geliştirerek bunu yapmaya çalışmışlardır. WEKA'da platformunda da test etmişlerdir.

Araştırmalarında ayrıntılı bir şekilde konuya nasıl yaklaştıklarını ve nasıl yaptıklarını anlatmışlardır. Sonuç olarak geliştirdikleri algoritmanın, hali hazırda kullanılan Apriori algoritmasından daha iyi olduğu ortaya çıkmıştır (Aher ve Lobo, 2012).

Spectral clustering algorithm for navie users (Naif kullanıcılar için spektral kümeleme algoritması)

Makalelerinde Spektral Kümeleme Algoritması üzerinde çalışmışlardır, K-Means algoritması birçok yerde kıyaslama açısından bahsedilmiştir. Makalelerinin isminin de içinde geçen, profesyonel olmayan sıradan kullanıcıların, Navie, Spektral kümelemeyi kolaylıkla kullanabilmeleri ve profesyonel bir bilgiye sahip olmadan optimize edilmiş kümelerin sayısına erişmelerini amaçlamışlardır. Iris verileri üzerinde çalışmışlardır ve WEKA programı kullanmışlardır. Spektral Kümeleme Algoritması ile bu veri kümesi kolayca üç kümeye ayırılmıştır. Bunu standart K-Means algoritması ile yapmak çok daha zor ve çetrefilli olacağı gibi yeterli mesleki alt yapısı olmayan bir kullanıcının içinden çıkabileceği bir durum değildir. Birçok açıdan öğretici nitelikte olan bu makalede bizim de kendi çalışmamızda üzerinde durduğumuz WEKA ve K-Means algoritması hakkında bilgiler vermesi sebebiyle çok faydalı bir kaynak olmuştur (Rao, Suryanarayana ve Veeraswamy, 2015).

Using wavelets to classify documents (Dalgacık dönüşümünü kullanarak belgeleri sınıflandırmak)

Wavelet sınıflandırması üzerinde durulan bu makalede kelimelerden oluşan metinlerin sinyal gibi davranıp, sinyal işleme aletlerince sinyalmiş gibi analiz edilmesi üzerinde çalışmışlardır. Bu yeni yaklaşım sayesinde sınıflandırma algoritmalarında çok daha verimli ve hızlı bir sonuç alınması amaçlanmıştır. Malumunuz günümüzde Fourier ve Cosine ayrık transformasyon çeşitleri metin sınıflandırılmasında çoğunlukla kullanılmaktadır. Elde edilen sonuçlar ışığında bu yaklaşımın daha da genişletilip tüm wavelet tiplerinde de kullanılması gerektiğini göstermiştir. Alışılageldik yöntemlere nazaran çok daha iyi sonuçlar alınmıştır. Wavelet hakkında bilgi edinmek açısından çok faydalı bir makaledir. Sınıflandırma ve kümeleme algoritmaları aynı şey olmasAda birbirinden ayrıldıkları farkları görmek ve mantığını anlamak açısından çok yararlı bir araştırmadır (Castro, Pinheiro, Souza ve Xexéo, 2008).

Açık kaynak kodlu veri madenciliği programları: WEKA'da örnek uygulama

Gazi Üniversitesi Elektronik-Bilgisayar Bölümü Bilim İnsanları bu araştırmalarında veri madenciliği programları WEKA, R ve RapidMiner(YALE) hakkında faydalı bilgiler vermişlerdir. Ayrıca WEKA'da geliştirilmiş örnek bir uygulama da yapmışlardır. WEKA günümüzde popüler bir veri madenciliği programı olması sebebiyle makalenin çoğunda yer almaktadır. WEKA'da veri temizleme, veri dönüştürme ve modelleme çok güzel açıklanmıştır. Ayrıca R programının da açıklanması çok faydalı olmuştur. Çünkü R, grafikler üzerinde veri analizi açısından çok önemli bir programdır. Son olarak makalelerinde bu üç veri madenciliği programı karşılaştırılmış ve çok faydalı bir çalışma elde edilmiştir (Dener, Dörterler ve Orman, 2009).

Formation of attribute spaces using wavelets in automatic classification of explosives (Dalgacık dönüşümünü kullanarak patlayıcıların otomatik olarak sınıflandırılması)

Bulgar Bilim İnsanları nitelik uzayının oluşumunu patlayıcıların sınıflandırılmasını kullanarak açıklamayı amaçlamışlardır ve bunun için de Wavelet dönüşümünü kullanmışlardır. Wavelet dönüşümünü kavrayabilmek ve program hakkında yapılan deneyler ile bilgi sahibi olmak amacıyla bu makaleyi literatür taramasına eklemiş bulunmaktadır. Bu makalede elde edilen sonuçları sırayla açıklayacak olursak, patlatıcıların temassız ultrasonik sınıflandırılması başarıyla gerçekleştirilmiştir. Deneysel olarak ayrık Wavelet dönüşümünü hayali görüntü tanıma methoduna adapte ederek patlayıcıların sınıflandırılması kanıtlanmıştır. Patlayıcıların sınıflandırılmasında ayrık Wavelet dönüşümünün kullanılması, uygun Wavelet ve ayrışma seviyesi sağlandığında olumlu sonuçlar vermiştir (Ilarionov, Shopov, Simeonov ve Kilifarev, 2008).

Wavelet-based anytime algorithm for k-means clustering of time series (Zaman serilerinde k-kümeleme kümelemesiyle dalgacık tabanlı her zaman algoritması)

Direkt olarak hazırlamakta olduğumuz makale ile alakalı olan bu çalışmada, K-Means ve Haar Wavelet dönüşümü kullanılmıştır. Zaman serileri üzerinde çalışılan bu makale sonucunda çok etkileyici sonuçlar elde edilmiştir. Kümeleme algoritması olarak kullandıkları K-Means algoritmasına farklı bir yaklaşımda bulunarak, her çözümün sonundaki final merkezlerini tekrardan bir sonraki çözümün başlangıç merkezi olarak

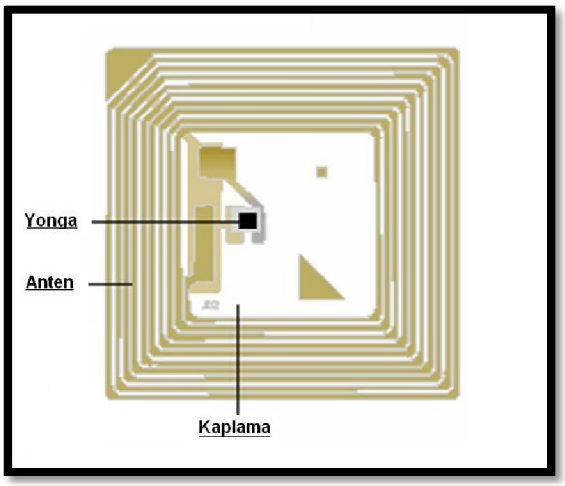
kullanarak algoritmanın çalışma zamanını büyük bir oranda hızlandırmış ve daha kaliteli bir kümeleme elde etmişlerdir. Anytime algoritması dedikleri bu yaklaşım sayesinde geleneksel K-Means algoritmasına kıyasla çok daha hızlı sonuç elde edilmiştir ve elde edilen kümeleme daha kaliteli sonuç vermiştir. Burada dikkat edilmesi gereken bir diğer husus da şudur: Anytime algoritması kullanıcıyı program çalışırken, herhangi bir aşamada müdahale etme olanağı tanımaktadır (Gunopulos, Keogh, Lin ve Vlachos, 2003).

Clustering signals using wavelets (Dalgacık dönüşümünü kullanarak sinyal kümelemesi)

Makalelerinde sinyaller üzerinde çalışmışlardır. Sinyal kümelemesi yapmak için Wavelet dönüşümünde faydalanmışlardır. İlk olarak Wavelet hakkında bilgi verilen makalede daha sonra kullandıkları Wavelet tabanlı sinyal kümeleme prosedürlerini anlatmışlardır. Bu bilgiler ışığında da Wavelet de sinyal verileri üzerinde bir kümeleme çalışması yapıp sonuca bağlamışlardır. Çok öğretici bir makaledir. Wavelet hakkında bilgisi zayıf olan herkesin okurken çok şey öğreneceği bu makale, kendi tezimizin Wavelet ile kümeleme algoritmaları kullanılan kısımlarında çok faydalı olacaktır (Misiti, Misiti, Oppenheim ve Poggi, 2007).

Rfid-enabled supply chain detection using clustering algorithms (Kümeleme algoritmalarını kullanarak RFID özellikli tedarik zinciri algılama)

Öncelikle burada hatırlamamız gereken şey RFID kavramının günümüzde neye tekabül ettiğidir. RFID türkçe açıklamasıyla: Radyo frekansı ile tanımlama günümüzde çokça kullanılan radyo frekansları yardımıyla herhangi bir nesneyi gerekli aletler kullanıldığı zaman tanımamızı sağlayan bir çeşit barkod sistemidir. Şekil 1.1' de gördüğümüz üzere, RFID bileşenleri; yonga, anten ve kaplamadan oluşmaktadır.



Şekil 1.1 : RFID Etiket Örneği

Yukarıdaki bilginin ardından bu makalelede, Weka platformunda 7 adet farklı kümeleme algoritması kullanarak, Rfid kullanılan tedarik zincirinin tespitini yapmaya çalışılmıştır. Bu doğrultuda birlikte çalıştıkları firmanın yetkililerinden yaşanılan zorluklar hakkında bilgiler almış ve sahtekarlık, dolandırıcılık probleminin üzerine odaklanmışlardır.Bu sebeple denedikleri 7 kümeleme algoritmalarının sonuçlarını inceleyerek ihtiyacı karşılamaya yönelik bir yaklaşım geliştirmişlerdir. Bizde özel sipariş yazılım sayılabilecek bu çalışmalarını makale haline getiren araştırmacılar çok bilgilendirici bir makale yayınlamışlardır (Azahar, Hassan ve Singh, 2015).

2. MATERYAL VE METOD

2.1. Materyal

Tezimde kümeleme algoritmalarının Weka ve Matlab platformlarında uygulanıp elde edilen sonuçların incelenmesi amacıyla üç veri kümesi üzerinde çalışılmıştır. Bunlar:

- Kandilli sıcaklık verileri (Son bir asrın verilerini içermektedir)
- Kandilli rüzgar şiddeti verileri (Son bir asrın verilerini içermektedir)
- Türkiye illeri ortalama sıcaklık verileri (Son bir asrın verilerini içermektedir)

Verilerimizin kullanımı aşamasında 64 satırlık gruplar oluşturacak şekilde bir seçim yapılmıştır. Bunun sebebi Matlab'da kümeleme algoritmalarının uygulanması sonucu elde edilecek verilerin, Wavelet Toolbox'da kullanımı aşamasında kolaylık sağlanmasıdır. Veri kümelerimizin uzantılarını da belirtecek olursak; Kandilli sıcaklık verileri ve Türkiye illeri ortalama sıcaklık verileri santigrat, Kandilli rüzgar şiddeti verilerinin uzantıları m/sn'dir. Bu verilerin ve kümeleme algoritmalarının işlenip, karşılaştırmaların yapılabilmesi amacıyla aşağıda belirtilen donanım ve yazılımlar kullanılmıştır. Bunlar:

- Bir adet kişisel bilgisayar ve taşınabilir hard disk
- Weka (Veri madenciliği programı)
- Matlab (Çok paradigmalı sayısal hesaplama yazılımı)
- Matlab Wavelet Toolbox programı

2.2. Metod

Çalışmamızda metod olarak aşağıdaki bölümlerde bahsedilen algoritmalardan üç tanesi seçilmiştir. Bunlar:

- K-Means Kümeleme Algoritması
- Hiyerarşik Kümeleme Algoritması

- Expectation Maximization Kümeleme Algoritması

Ayrıca Wavelet Toolbox'da analizlerimizden derlediğimiz veriler kullanılarak çıkarımlarda bulunulmuştur. Elde edilen çıkarımlar tezimizden türetilen bir makalede açıklanmak amacıyla hazırlanmaktadır.

2.2.1. Algoritmanın tanımı

Algoritma için ulaşabildiğimiz bilgi kaynaklarında birbirine benzer ve aynı yola çıkan bir çok açıklama bulunmaktadır. Peki daha basite indirgeyerek algoritmayı tanımlamak gerekirse nasıl bir tanım yapılmalıdır? En basit şekilde algoritmaya kısaca çözüm yolu denilebilir. Sorunun, problemin, yapmak istediğimiz şeyin ne olduğu çok önemli değildir. Yapılması gereken şey onu çözmektir. Bu aşamada kabul görmüş iki yaklaşım vardır. Bir tanesi içgüdüsel olarak sorunlara yaklaşmak ve çözüme kavuşturmak. Ki bu durumda eksiklikler ve hatalar iş çözülürken görülür ve tamamlanır. Dolayısıyla çözüm tahmin edilenden daha maaliyetli olur. Burada maaliyetten kasıt zaman, harcanan iş gücü ve para olarak düşünülebilir. İkinci yöntem ise olaya bilimsel bir açıdan yaklaşmak ve onu nitel ve nicel olarak etraflıca inceleyip bir plan yardımıyla çözüme kavuşturmaktır. Bunu başarabilmek için gerekli olan şeyleri kabaca şu şekilde açıklayabiliriz. Öncelikle problemin tespiti çok önemlidir. Problem tespit edildikten sonra bunu nasıl çözeceğimize odaklanıp kabaca bir çözüm sistemi geliştirilir. Eğer bu sistem matematiksel bir soruna odaklanıyorsa kullanacağımız veriler o doğrultuda seçilir. Aksi durumda kullanılacak değerler uygun bir veri türünde seçilir. Verilerimiz ve çözüm yolumuz belli olduğunda, işleri daha anlaşılır yapmak amacıyla akış diyagramı dediğimiz tekniğe başvururuz. Bunlar genelde hep bir bütün gibi ele alınır ve anlatılır. Ancak burada algoritma kavramının karşılığı, geliştirmiş olduğumuz çözüm yöntemidir. Olayı biraz daha bilgisayar ile alakalı hale getirip basit bir örnek algoritma yaparsak. C platformunda sadece toplama işlemi yapan bir program yapmak istediğimizi düşünelim. Burada ilk olarak öğrenmemiz gereken şey toplamanın nasıl yapıldığı ve hangi veriler ile yapıldığını öğrenmek olacaktır. Yapacağımız araştırmalar sonucunda C platformunda basit bir toplama işlemi yapmak istiyorsak kullanıcıdan talep edeceğimiz verinin bir türü olması gerektiğini görürüz. Burada program çalıştığında alınacak sonucun basit olması amacıyla, verileri integer yani tam sayı belirliyoruz. Ayrıca seçilebilecek birçok veri türü

olduğunu da hatırlatmak gerekir; char, int, float, double gibi. Algoritmamızı oluştururken ilk önce toplanacak iki adet integer veri bulmamız gerektiğini biliyoruz. Bunları programı kullanan kişiden istememiz gerekmekte. Bu sebeple program çalıştırıldığında kullanıcıdan iki adet integer veri istenilecek. Sonra bu iki veriyi toplamamız gerekir. Biraz C programında bilgi sahibi olduğumuzda görüyoruz ki, toplanacak iki verinin bir başka değişkene atanması doğru bir yaklaşımdır. Bu sebeple integer olan bir c değeri programımıza tanıtıp, kullanıcıdan aldığımız iki değeri toplayıp burada saklıyoruz. Sonra c değerini ekrana baştırıp en son adım olan algoritmayı sonlandırıyoruz. Bu tasarladığımız algoritmayı sıralayacak olursak:

1. Başla

2. Birinci veriyi iste (birinci veri a olsun)

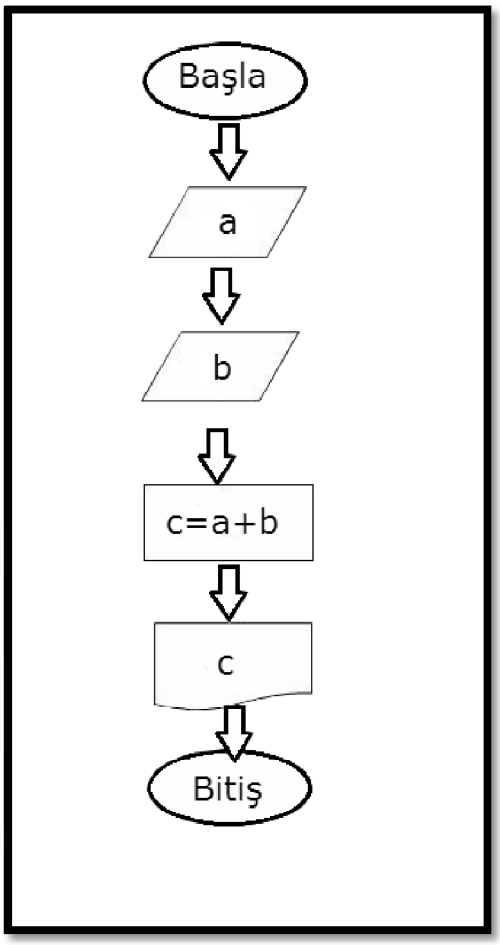
3. İkinci veriyi iste (ikinci veri b olsun)

4. İki veriyi toplayıp üçüncü değişkene kaydet (üçüncü veri c olsun)

5. Üçüncü değişkeni ekrana gönder

6. Bitiş

Bu altı adımdan oluşam çözüm yönteminin tamamı algoritma olarak adlandırılır. Yapmış olduğumuz bu örnek, konunun anlaşılması amacıyla verilebilecek en yalın ve basit örnektir. Sonraki adımda algoritmanın akış diyagramında gösterilmesi gerekmektedir. Evrensel akış diyagramı şekilleri kullanılarak hazırlanan bu diyagramlar algoritmanın kodlanmasından önce yazılımcıya yol gösterme açısından büyük önem arz etmektedir. Algoritmamızın akış diyagramını gösterecek olursak:



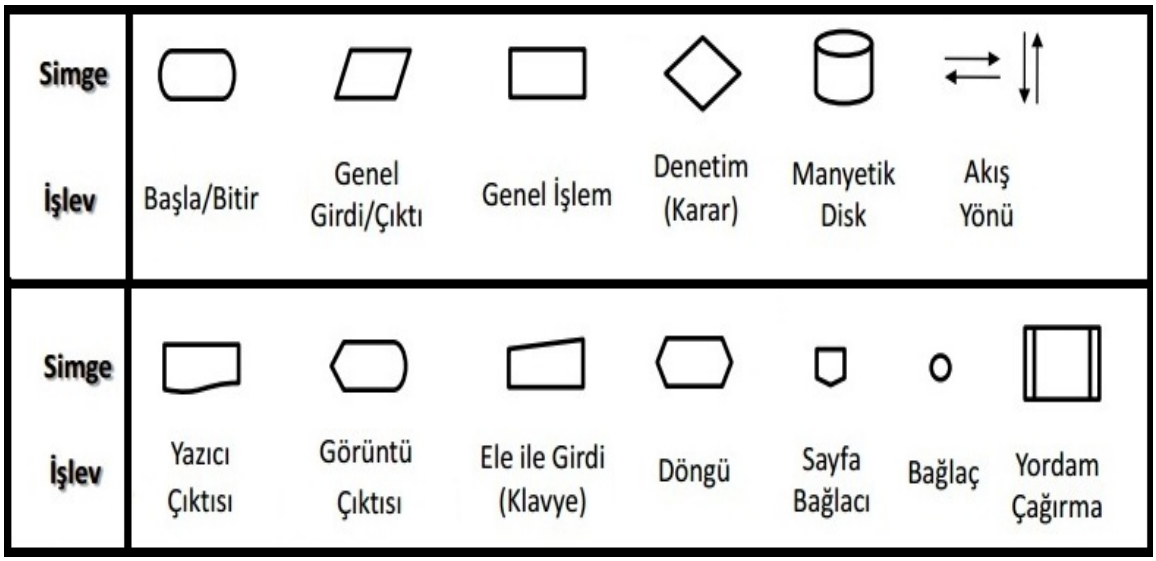
Şekil 2.1 : Akış Diyagramı Örneği

Şekil 2.1'de gördüğümüz üzere, yazılı olarak belirlediğimiz algoritmamız gayet basit ve anlaşılır bir biçimde akış diyagramına çekilmiştir. Bundan sonraki adım bu algoritmayı programa dökmektir. Aşağıda yazdığımız kod, bu algoritma ve akış diyagramının karşılığıdır ve istenileni eksiksiz vermektedir.

```
#include <stdio.h>

#include <stdlib.h>

int main()

{

int a;

int b;
```

```
printf("ilk integer sayiyi giriniz: \n");

scanf(“%d”,&a);

printf("ikinci integer sayiyi giriniz: \n");

scanf(“%d”,&b);

int c;

c=a+b;

printf("\nToplamlari: %d",c);

return 0;

}
```

Şekil 2.2'de evrensel olarak kullanılan akış diyagramı sembolleri ve anlamları hakkında bilgiler verilmiştir. İnternette akış diyagramı çiziminde kullanılan birçok site bulunmaktadır. Görüntü olarak çok daha profesyonel akış diyagramları çizmenize yardımcı olur. Ancak basit bir Microsoft Paint programı ile ihtiyacınız olan diyagramı çizmeniz mümkündür.



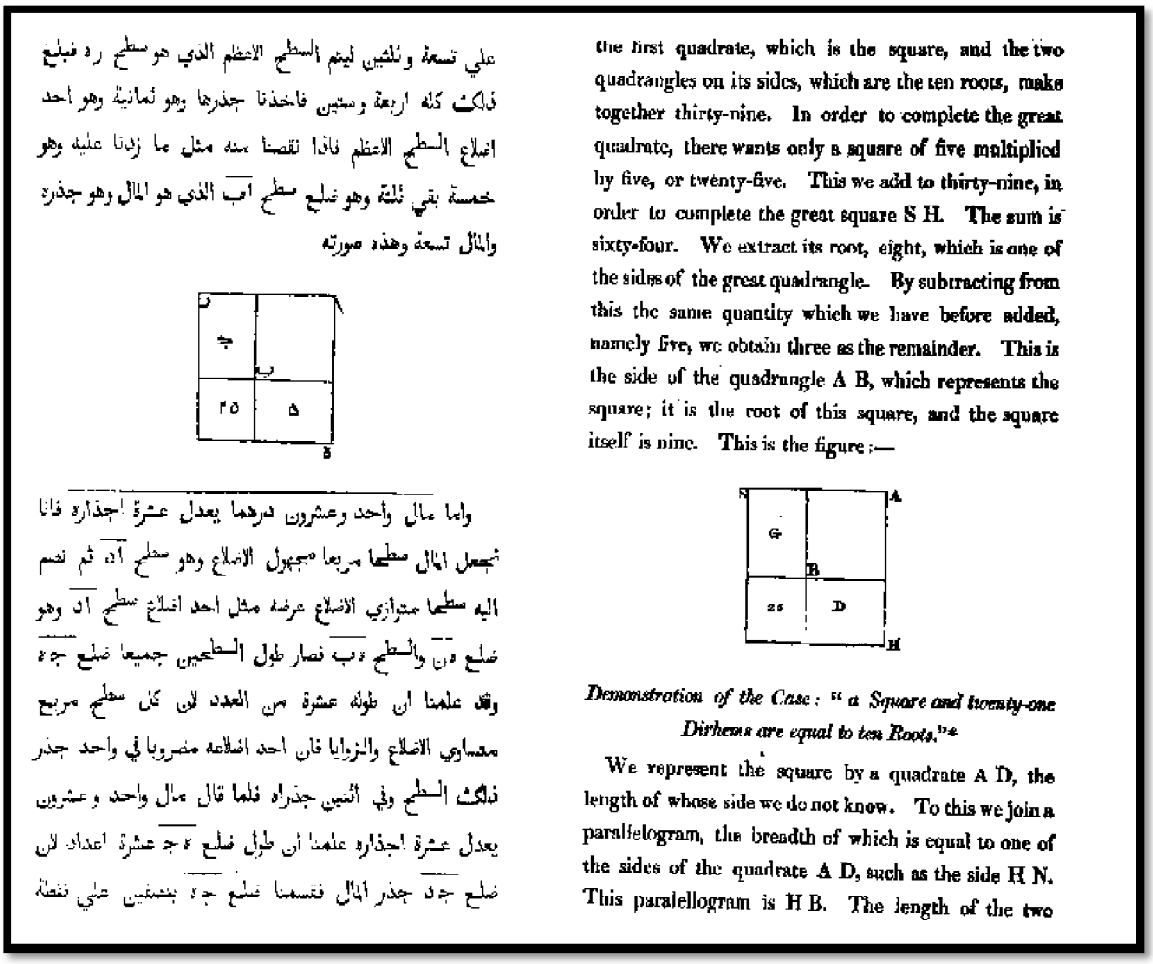
Şekil 2.2 : Akış Diyagramı Sembolleri

2.2.2. Algoritmanın tarihçesi

Günümüzde algoritma denildiğinde insanların aklında matematik ile özleşmiş bir anlam uyanır. Çok doğru bir tespittir bu. Çünkü algoritmaya ismini veren ve ilk matematik kitabı olarak kabul edilen "Hisab el-cebir ve el-mukabala", matematik denildiğinde akla ilk gelen isimlerden biri olan Harizmi tarafından yazılmıştır. Matematiğin babası olarak tanınır. Asıl adı Muḥammad ibn Mūsā al-Khwārizmī olan Harizmi 780 ve 850 yılları arasında yaşamıştır.

Şekil 2.3 : Al-Khwārizmī

Yapmış olduğu çalışmalar ve yazdığı kitaplar latinceye çevrilmiş ve Avrupa medeniyetinin gelişmesinde büyük katkı sağlamıştır. Algoritma kelimesi, Harizmi'nin isminin latince yazılması sonucu ortaya çıkmıştır. Al-Khwārizmī fonetik olarak Avrupalılarca telafuz edildiğinde Algoritmi, Algorithm olarak söylenmesi ve bunun yazıya dökülmesi sonucu günümüzün "Algoritma" kelimesi ortaya çıkmıştır. Yukarıda Şekil 2.3 'de Harizmi'nin resmini görebilirsiniz (Url-1). Ayrıca Şekil 2.4'de Harizmi'nin Cebir Kitabı'ndan bir sayfa ve bunun Fredrick Rosen'in The Algebra of Al-Khwarizmi adlı kitabında ki bir çevirisini görebilirsiniz (Karpinski, 1912).



على تسعة وثلثين ليتم السطح الاعظم الذي هو سطح رد فبلغ ذلك كله اربعة وستين فاخذنا جذرها وهو ثمانية وهو احد اضلاع السطح الاعظم فاذا نقصنا منه مثل ما زدنا عليه وهو خمسة بقي ثلثة وهو ضلع سطح اب الذي هو المال وهو جذره والمال تسعة وهذه صورته

واما مال واحد وعشرون درهما يعدل عشرة اجذاره فانا نجعل المال سطحا مربعا مجهول الاضلاع وهو سطح اد ثم نضم اليه سطحا متوازي الاضلاع عرضه مثل احد اضلاع سطح اد وهو ضلع هن والسطح ه ب فصار طول السطحين جميعا ضلع جه وقد علمنا ان طوله عشرة من العدد لان كل سطح مربع متساوي الاضلاع والزوايا فان احد اضلاعه مضروبا في واحد جذر ذلك السطح وفي اثنين جذراه فلما قال مال واحد وعشرون يعدل عشرة اجذاره علمنا ان طول ضلع ه ج عشرة اعداد لان ضلع جد جذر المال فقسمنا ضلع جه بنصفين على نقطة

the first quadrate, which is the square, and the two quadrangles on its sides, which are the ten roots, make together thirty-nine. In order to complete the great quadrate, there wants only a square of five multiplied by five, or twenty-five. This we add to thirty-nine, in order to complete the great square S H. The sum is sixty-four. We extract its root, eight, which is one of the sides of the great quadrangle. By subtracting from this the same quantity which we have before added, namely five, we obtain three as the remainder. This is the side of the quadrangle A B, which represents the square; it is the root of this square, and the square itself is nine. This is the figure:—

Demonstration of the Case: "a Square and twenty-one Dirhems are equal to ten Roots."

We represent the square by a quadrate A D, the length of whose side we do not know. To this we join a parallelogram, the breadth of which is equal to one of the sides of the quadrate A D, such as the side H N. This parallelogram is H B. The length of the two

Şekil 2.4 : Cebir Kitabı Örneği ve Çevirisi

Tabii ki ilk algoritma örnekleri Harizmi tarafından verilmemiştir. MÖ 1600 Babiller, MÖ 300 Öklid, MÖ 200 Eratosthenes Elemesi olarak bilinen Eratosthenes'in algoritma üzerindeki çalışmaları örnek verilebilir. Milattan önce 300 yıllarında öklid algoritması ve Öklid'in aksinomları oldukça önemlidir (Cooke, 2005). Babil kil tabletlerin de saat, yer ve astronomik olayların hesaplanması gibi durumlar için algoritmik prosedürler kullanıldığı görülmektedir (Aaboe, 2001). Bunlar dışında milattan sonraki zamandan günümüze kadar geliştirilen bazı algoritmalar ve bilimadamlarını hatırlamak gerekirse: 263 yılında Liu Hui, Gaussal elemeyi açıklamıştır.

John Napier 17. yy.'da geliştirdiği method ile algoritma kullanan hesaplamalar yapmıştır. Yine 17. yy.'da Isaac Newton, Newton-Raphson metodunu geliştirmiştir. Aynı yüzyıl'da Joseph Raphson da bu methodu bağımsız olarak bulmuştur. 19. yy.'da Carl Friedrich Gauss, Cooley-Tukey algoritmasını geliştirmiştir.

20. yy. başlarında Boruvka algoritması ardından Boris Delaunay tarafından, Delaunay üçgen bölümlemesi geliştirilmiştir. Yirminci yüzyıl ortalarında ise geliştirilen algoritmalardan bahsedecek olursak; John von Neumann'ın geliştirdiği Birleştirmeli Sıralama örnek verilebilir. İki yıl sonra George Dantzig'in geliştirdiği Simplex Algoritması bir örnektir.

1950'ler David A. Huffman, Harold H. Seward, Joseph Kruskal, Robert Prim, R.Bellman ve L. R. Ford, Edsger Dijkstra, D. L. Shell, Paul de Casteljau'nun geliştirdikleri algoritmalar ile geçmiştir. Bu algoritmalar yine sırasıyla; Huddman kodlaması, Radis sıralaması, Kruskal algoritması, Prim algoritması, Bellman-Ford algoritması, Dijkstra algoritması, Shell sıralaması ve De Casreljau algoritmalarıdır.

1960'lı yıllarda C. A. R. Hoare'nin Hızlı sıralaması, L. R. Ford ve D. R. Fulkerson'nun geliştirdiği Ford-Fulkerson algoritması, Jack E. Bresenham'ın geliştirdiği Bresenham doğru algoritması, J. W. J. Williams tarafından geliştirilen Öbek sıralama, James Cooley ve John Tukey, Cooley-Tukey algoritmasını yeniden bulmuşlardır. Vladimir Levenshtein Levenshtein aralığını geliştirmiitir. T. Kasami bağımsız olarak Cocke-Younger-Kasami(CYK) algoritmasını geliştirmiştir. Andrew Viterbi, Viterbi algoritmasını açıklamıştır. D. H. Younger, Cocke-Younger-Kasami(CYK) bağımsız olarak geliştirmiştir. Grafik A*arama algoritması Peter Hart, Nils Nilsson ve Bertram Raphael tarafından geliştirilmiştir.

Donald Knuth ve P. B. Bendix tarafından geliştirilen Knuth-Bendix completion algoritması 1970'li yılların başlarında açıklanmıştır. Ronald Graham, Graham taramasını gerçekteştirmiştir. RSA şifreleme algoritması Clifford Cocks tarafından geliştirilmiştir. R. A. Jarvis, Jarvis march algoritmasını duyurmuştur. John Pollard p-1 algoritmasını açıklamıştır. Genetik algoritma John Holland tarafından popülerleştirilmiştir. John Pollard ro algoritmasını açıklamıştır. Aho-Corasick algoritması Alfred V. Aho ve Margaret J. Corasick tarafından geliştirilmiştir. Eugene Salamin ve Richard Brent, Salamin Brent algoritmasını açıklamışlardır. Knuth-Morris-Pratt algoritması, Donald Knuth ve Vaughan Pratt ve bağımsız olarak da J. H. Morris tarafından geliştirilmiştir. Boyer-Moore string arama algoritması açıklamıştır. RSA şifreleme algoritması Ron Rivest, Adi Shamir ve Len Adleman'ca yeniden bulunmuştur. LZ77 ve LZ78

algoritmaları Abraham Lempel ve Jacob Ziv tarafından geliştirilmiştir. G. Bruun, Bruun algoritmasını duyurulmuştur. Leonid Khachiyan Khachiyan ellipsoit metodunu duyurmuşlardır.

1980'ler de, İkinci dereceden eleme metodu Carl Pomerance tarafından açıklanmıştır. Simulated annealling, S. Kirkpatrick, C. D. Gelatt ve M. P. Vecchi tarafından geliştirilmiştir. LZW algoritması Terry Welch tarafından geliştirilmiştir. İç nokta algoritması Narendea Karmarkar tarafından açıklanmıştır. Simulated annealing bağımsız olarak V. Cerny tarafından geliştirmiştir. Blum Blum Shub, L. Blum, M. Blum ve M. Shub tarafınca önerilmiştir. Özel sayı alanı elemesi John Pollard tarafından geliştirilmiştir.

90'lar da, Carl Pomerance, Joe Buhler, Hendrik Lenstra ve Leonard Adleman tarafından geliştirilen Genel sayı alanı elesi, Özel sayı elesi yöntemi duyurulmuştur. Beklemesiz Senkronizasyon Maurice Herlihy tarafından geliştirilmiştir. D. Deutsch ve R. Jozsa tarafından Deutsch-Jazso algoritması duyurmuşlardır. Shor algoritması, Peter Shor tarafından açıklanmıştır. Burrows-Wheeler dönüşümü Michael Burrows ve David Wheeler tarafından geliştirilmiştir. Lov K. Grover, Grover algoritmasını açıklamıştır. RIPEMD-160 Hans Dobbertin, Antoon Bosselaers ve Bart Preneel tarafından geliştirilmiştir. Andrew Tridgell, rsync algoritmasını duyurmuştur. Bruce Schneier, John Kelsey ve Niels Ferguson, 99 yılında Yarrow algoritmasını tasarlamışlardır.

2000'li yıllara geldiğimizde ise LZMA sıkıştırma algoritması geliştirilmiştir. 2002 yılında ise AKS primality test, Manindra Agrawal, Neeraj Kayal ve Nitin Saxena tarafından geliştirilmiştir(URL2).

Buradan da anlaşılacağı gibi, algoritma, üzerinde çokça çalışılan ve geçmişi çok eskilere dayanan bir kavramdır. Son olarak şundan da bahsetmek gerekir; günümüzde ilk bilgisayar hakkında ortak kanı, Alan Turing tarafından 1936 yılında icat edilen Turing Machine'dir. Halen günümüzde kullanılan tüm programlama dilleri, temelde Turing'in geliştirdiği algoritma ve prensiplerden türetildiği için, programlama dillerinin özünde mantıksal bir benzerlik görülür.

2.2.3. Algoritma çeşitleri ve kümeleme algoritmaları

2.2.3.1. Algoritma çeşitleri

Algoritmalar günümüzde birçok sektörde direkt yada dolaylı olarak kullanılmaktadır. Her sorunun çözümüne getirilen yaklaşımlar farklı olabilmekle birlikte, hepsi doğru sonuca ulaşabilir. Burada algoritmanın kullanılacağı spesifik durum önemlidir. Örneğin sıralanacak bir veri yığını olduğunu düşünelim. Bu veri yığınının büyüklüğünün, kullanılacak sıralama algoritması seçilirken çok önemli olduğunu söyleyebiliriz. Çünkü bazen çok hızlı çalışan komplike algoritmalar küçük verilerde, çok daha basit bir mantıkla çalışan basit algoritmalara kıyasla daha verimsiz çalışır. Bunun sebebi, algoritmanın kullanılacağı doğru durumu tespit etmenin önemidir. Ayrıca bilgi olarak şunu da paylaşmakta fayda görüyorum. Algoritmanın kalitesi, çalışma hızı ve kullandığı bellek ile orantılıdır. Yani bir algoritma ne kadar hızlı çalışıyorsa ve ne kadar az bellek kullanıyorsa, o kadar kaliteli, o kadar verimli bir algoritma demektir.

Algoritmalar yaptıkları işler, odaklandıkları alanlarla doğru orantılı olarak gruplanırlar. Yapılan araştırmalar ve bilgiler ışığında algoritmaları sınıflandırmak gerekirse kabaca altı gruptan bahsedebilmekteyiz. Bunları başlık başlık belirmek gerekirse:

1- Birleşimsel Algoritmalar: Sıra algoritmaları, Genel birleşim algoritmaları, Grafik algoritmalarından oluşmaktadır.

2- Hesaplamalı Matematiksel Algoritmalar: Sayısal algoritmalar, Teorik sayı algoritmaları, Bilgisayar cebiri algoritmaları, Soyut cebir algoritmaları, Geometri algoritmaları, Optimasyon algoritmalarından oluşmaktadır.

3- Hesaplamalı Bilim Algoritmaları: Astronomi algoritmaları, Biyoinformatik algoritmaları, Geoscience algoritmaları, Dilbilim algoritmaları, Tıp algoritmaları, Fizik algoritmaları, İstatistik algoritmalarından oluşmaktadır.

4- Bilgisayar Bilimi Algoritmaları: Bilgisayar mimarisi algoritmaları, Bilgisayar grafikleri algoritmaları, Kriptografi algoritmaları, Sayısal mantık algoritmaları, Makine öğrenmesi ve istatistiki sınıflandırması algoritmaları, Programlama dili teorisi algoritmaları, Kuantum algoritmaları, hesaplama ve otomata teorisi algoritmalarından oluşmaktadır.

5- Bilgi Teorisi ve Sinyal İşleme: Kodlama Teorisi Algoritmaları (Kodlama Teorisi, Hata bulma ve düzeltme, Kayıpsız sıkıştırma algoritmaları, Kayıplı sıkıştırma algoritmaları vb.), Dijital sinyal işleme algoritmaları (Görüntü işleme gibi.).

6- Yazılım Mühendisliği Algoritmaları: Veritabanı algoritmaları, Dağıtık sistemler algoritmaları, Bellek ayırma ve serbest bırakma algoritmaları, İşletim sistemleri algoritmaları(Ağ, Süreç senkronizasyon, zamanlama ve disk planlaması algoritmaları.) gibi algoritmalardan oluşmaktadır.

2.2.3.2. Kümeleme algoritmaları

Kümeleme sınırlı bir obje grubu üzerinde uygulanan bir sınıflandırma yöntemidir (Dubes ve Jain, 1988). Kümeleme sınıflandırmanın özel bir türüdür (Kendall, 1966). Ayrıca kümelemenin unsupervised yani eğiticisiz öğrenme prensibine sahip olduğunu da söylemek gerekir. Günümüzde gerek kitaplarda ve gerek internet ortamında birçok kümeleme algoritmasından bahsedilir. Yaptığımız araştırmalar sonucu ulaşılan güncel ve popüler kümeleme algoritmalarından seçtiklerimizi bu bölümde açıklayacağız. Bunları ayrıntılı açıklamadan önce sıralamak gerekirse:

1- K-Means Kümeleme Algoritması

2- Hiyerarşik Kümeleme Algoritması

3- Spektral Kümeleme Algoritması

4- Mean-shift Kümeleme Algoritması

5- Affinity Propagation Kümeleme Algoritması

6- DBSCAN Kümeleme Algoritması

7- Expectation Maximization Algoritması

Ayrıca kümelemede bazı ölçütler bulunmaktadır. Bunlar yardımıyla objeler kümelenirken birbirleriyle uyumlu olanlar belirlenir. Eğer elimizdeki veri kümesi sayısal verilerden oluşuyorsa bu uyumu anlayabilmek için kullanılabilecek yöntemleriz vardır. Bunlardan en çok kullanılanlarını aşağıda sıralayıp açıklanmıştır. Elimizdeki verilerin sayılsal olmadığı durumlarda ise farklı çözüm yöntemlerine başvurulur. Biz daha çok sayısal veriler üzerinde çalıştığımız için o konular hakkında bilgi verilmemiştir.

Öklid mesafesi

Öklid mesafesinin bulunmasında aşağıda (2.1)'de gösterilen formül kullanılmaktadır.Bu formül, i ve j noktalarının arasındaki uzaklığı bulmamızı sağlar. Burada p değişkeni, p boyutlu bir uzay demektir ve k değişken indeksidir.

$$d(i,j) = \sqrt{(|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \ldots + |x_{ip} - x_{jp}|^2)} \quad \textbf{(2.1)}$$

Ayrıca formulün sağlanması için aşağıdaki 2.1a,b,c,d, denklemlerindeki şartlar aranmaktadır.

$$d(i,j) \geq 0 \quad \textbf{(2.1a)}$$

$$d(i,i) = 0 \quad \textbf{(2.1b)}$$

$$d(i,j) = d(j,i) \quad \textbf{(2.1c)}$$

$$d(i,j) \leq d(i,k) + d(k,j) \quad \textbf{(2.1d)}$$

Çalışmamızda ele aldığımız K-Means, Expectation Maximization ve Hiyerarşik kümelemelerde genellikle veriler arası mesafenin hesaplanmasında öklid mesafe fonksiyonları kullanılmıştır. WEKA'da yapılan analizlerin büyük çoğunluğu yine Öklid fonksiyonu ile yapılmıştır.

Manhattan mesafesi

Manhattan mesafesinin bulunmasında aşağıda (2.2)'de gösterilen formül kullanılmaktadır. Bu formül, i ve j noktalarının arasındaki uzaklığı bulmamızı sağlar. Burada p değişkeni, p boyutlu bir uzay demektir ve k değişken indeksidir.

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \ldots + |x_{ip} - x_{jp}| \quad \textbf{(2.2)}$$

Ayrıca formulün sağlanması için aşağıdaki 2.2a,b,c,d, denklemlerindeki şartlar aranmaktadır.

$$d(i,j) \geq 0 \quad \textbf{(2.2a)}$$

$$d(i,i) = 0 \quad \textbf{(2.2b)}$$

$$d(i,j) = d(j,i) \quad \textbf{(2.2c)}$$

$$d(i,j) \leq d(i,k) + d(k,j) \quad \textbf{(2.2d)}$$

Minkowski mesafesi

Minkowski mesafesinin bulunmasında aşağıda (2.3)'da gösterilen formül kullanılmaktadır. Bu formül, i ve j noktalarının arasındaki uzaklığı bulmamızı sağlar. Burada p ve qdeğişkeni, p ve q boyutlu bir uzay demektir ve k değişken indeksidir.

$$d(i,j) = \sqrt[q]{(|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + \ldots + |x_{ip} - x_{jp}|^q)} \quad \textbf{(2.3)}$$

Ayrıca formulün sağlanması için aşağıdaki 2.3a,b,c,d, denklemlerindeki şartlar aranmaktadır.

$$d(i,j) \geq 0 \quad \textbf{(2.3a)}$$

$$d(i,i) = 0 \quad \textbf{(2.3b)}$$

$$d(i,j) = d(j,i) \quad \textbf{(2.3c)}$$

$$d(i,j) \leq d(i,k) + d(k,j) \quad \textbf{(2.3d)}$$

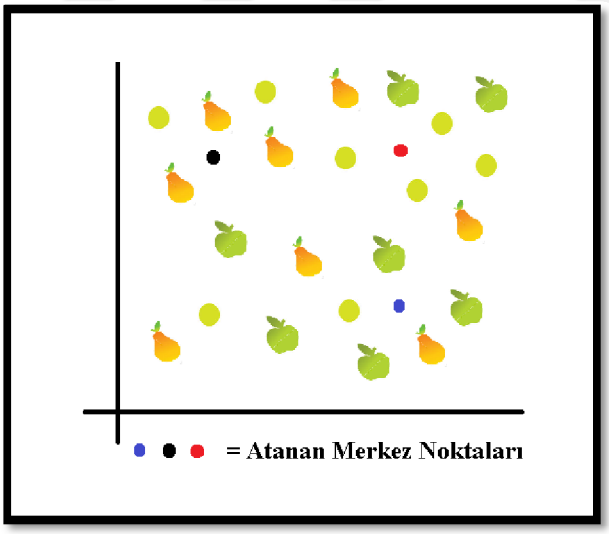
K-Means kümeleme algoritması

K-Means algoritması, kümeleme algoritmalarının içinde belki de en eski, en çok kullanılan ve bir o kadar da basit bir algoritmadır. Unsupervised yani eğiticisiz öğrenme prensibine sahiptir. Avantajları ve dezavantajları vardır, ancak büyük verilerdeki hızlı çalışması sebebiyle tartışılmaz en popüler algoritmalardan biridir. Eski bir algoritma denilmesinin sebebi ilk kez K-Means isminin 1967 yılında J. B. MacQueen tarafından kullanılmış olmasıdır. Gerçi K-Means algoritmasının mantığı 1957 yılı Hugo Steinhaus'un yaptığı çalışmalara dayanmaktadır (Steinhaus, 1957).

K-Means algoritmasında, kümelenecek verilerden her biri sadece bir kümenin elemanı olabilir. Bu kümelerin temsil edildiği noktalara ise merkez noktası denir. Dezavantaj olarak söyleyebileceğimiz belkide en önemli husus, algoritmanın kulanılacak verinin bölüneceği küme sayısını, kullanıcının girmesine bağlı olarak belirlemesi durumudur. Bu sebeple doğru küme sayısı belirlenene kadar deneme yanılma yöntemine başvurulması gerekebilir. K-Means işleminin başarıyla tamamlanması için bazen birkaç kez fonksiyonun çağırılması gerekebilir. Çünkü ilk seferde oluşan kümelerin içindeki benzerlik uyumu tutmayabilir. Fonksiyonun birkaç tekrardan sonra, kümelerdeki değişimin durması, elde edilen kümelerde istenilen sonucun alındığı anlamına gelir. Bir

diğer dezavantajda gürültülü verinin kullanımıdır. Kümeleme esnasında benzer veriler seçilirken, verideki gürültü gibi etkenler dikkate alınmaz.

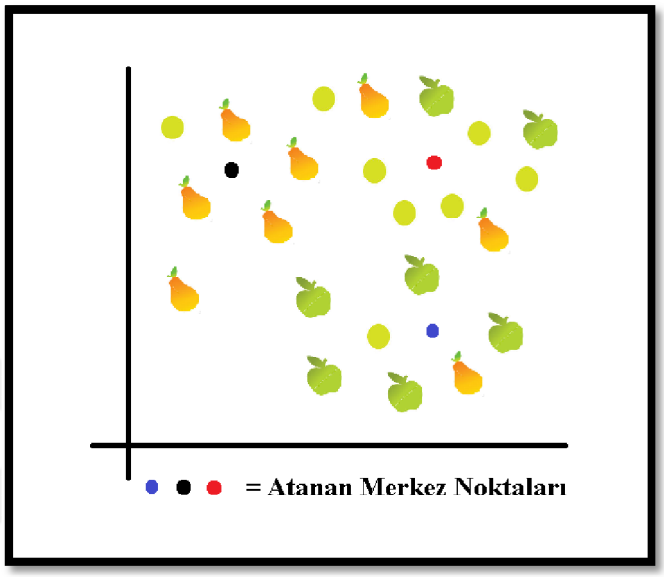
Örnek olarak büyük bir masa üzerinde karışık olarak bulunan bir miktar elma, armut ve erik meyvelerini düşünelim. Bu basit örnekteki verilerimizin üç çeşit meyveden oluştuğunu bildiğimiz için k değerini üç olarak belirleyelim. K-Means algoritmasını, bu veri grubuna uyguladığımızda, üç adet noktanın elimizdeki veri kümesinde merkez noktaları olarak atandığını görürüz. Sonra ki adımda bu merkez noktaları, kendilerine yakın olan veri topluluğuna yaklaşıp onların merkez noktası olurlar. Sonra çevrelerinde bulunan aynı ve miktarca fazla olan veri gurubunu çekip, farklı olan grupları ise kümelerine dahil etmezler. Diğer merkez noktaları da, başka kümelerde kullanılmamış ancak kendi kümesiyle benzerlik gösteren verileri çekerek kendi kümesine bağlar. Her adımda bu işlemleri yapan algoritmamız, işlemini bitirdiğinde, küme içinde benzerliğin yüksek olduğu üç farklı küme oluşturur. Oluşan bu kümeler, benzerlik açısından birbirine yakınlık göstermez.



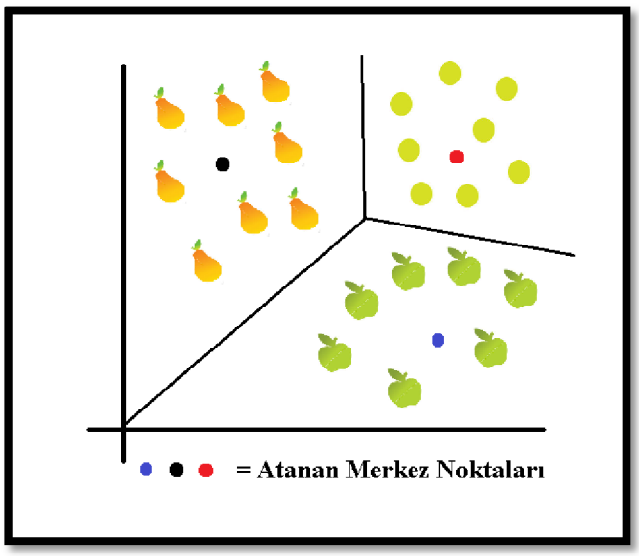
Şekil 2.5 : K-Means Örneği - 1

Açıkladığımız örneğimizin görselini Şekil 2.5'de gösterildiği gibi ifade edebiliriz. K-Means algoritmasında k değeri olarak girdiğimiz değerin üç olması sebebiyle algoritmamız üç merkez noktasını atamış bulunuyor. Daha sonra bu merkez noktaları birkaç yer değişimi sonucunda son merkez noktalarına yerleşip çevrelerindeki benzer verileri kümeliyorlar.

Aşağıda Şekil 2.6'da gördüğünüz üzere, çervresinde ki benzer verileri kümeleyen algoritmamızın, işlemini tamamlaması için birkaç tekrara daha ihtiyacı olduğu görülmektedir.



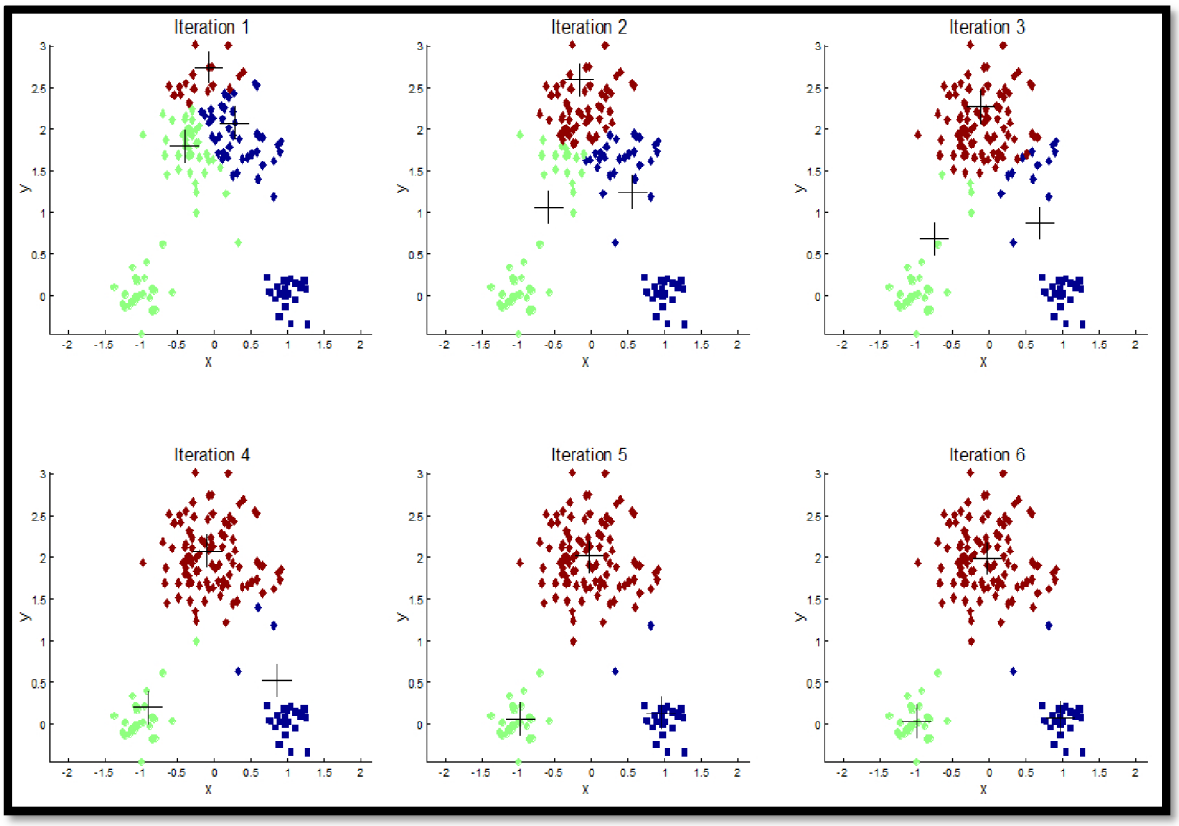
Şekil 2.6 : K-Means Örneği - 2



Şekil 2.7 : K-Means Örneği - 3

Yukarıdaki Şekil 2.7'de görüldüğü gibi K-Means algoritmamız elindeki veriyi üç farklı kümeye bölmüştür. Kümeler birbirine benzememektedir ve kendi içlerinde benzerlik göstermektedirler. Ancak uygulamada elde edilen sonuçlar, yukarıdaki örnekte olduğu gibi keskin bir ayrılma göstermeyebilir.

Aşağıdaki Şekil 2.8'de ise Dr. Andrei Pandre'nin blog sayfasında yapmış olduğu daha komplike bir K-Means örneğini inceleyebilirsiniz (URL3).



Şekil 2.8 : K-Means Örneği - 4

K means algoritmasının formülsel olarak ifadesi için (2.4) ve (2.5)'de ki denklemleri incelemeliyiz.

$$J(V) = \sum_{i=1}^{c} \sum_{j=1}^{ci} (||x_i - v_j||)^2 \qquad \textbf{(2.4)}$$

Burada '$||x_i - v_j||$', x ve y arasında ki öklid mesafesi, 'c_i', i^{th} kümesindeki veri noktalarının sayısı, c ise küme merkezlerinin sayısıdır.

K-Means Kümelemenin Algoritmik Adımları:

X = $\{x_1,x_2,x_3,……..,x_n\}$ kümesi veri noktalarının, V = $\{v_1,v_2,…….,v_c\}$ ise merkez noktalarının kümesi olsun.

1) Rastgele 'c' küme merkezlerini seç.

2) Her veri ile küme merkezlerinin arasındaki mesafeyi hesapla.

3) Küme merkeziyle arasındaki mesafe, diğer küme merkezleriyle olan mesafeden daha az olan veriyi, yakın olan o küme merkezine ata.

4) Yeni küme merkezini (2.5)'de ki denklemle yeniden hesapla:

$$v_i = (1 / c_i) \sum_{j=1}^{ci} x_i \quad \textbf{(2.5)}$$

'c_i', i^{th} kümesindeki veri noktalarının sayısını ifade etmektedir.

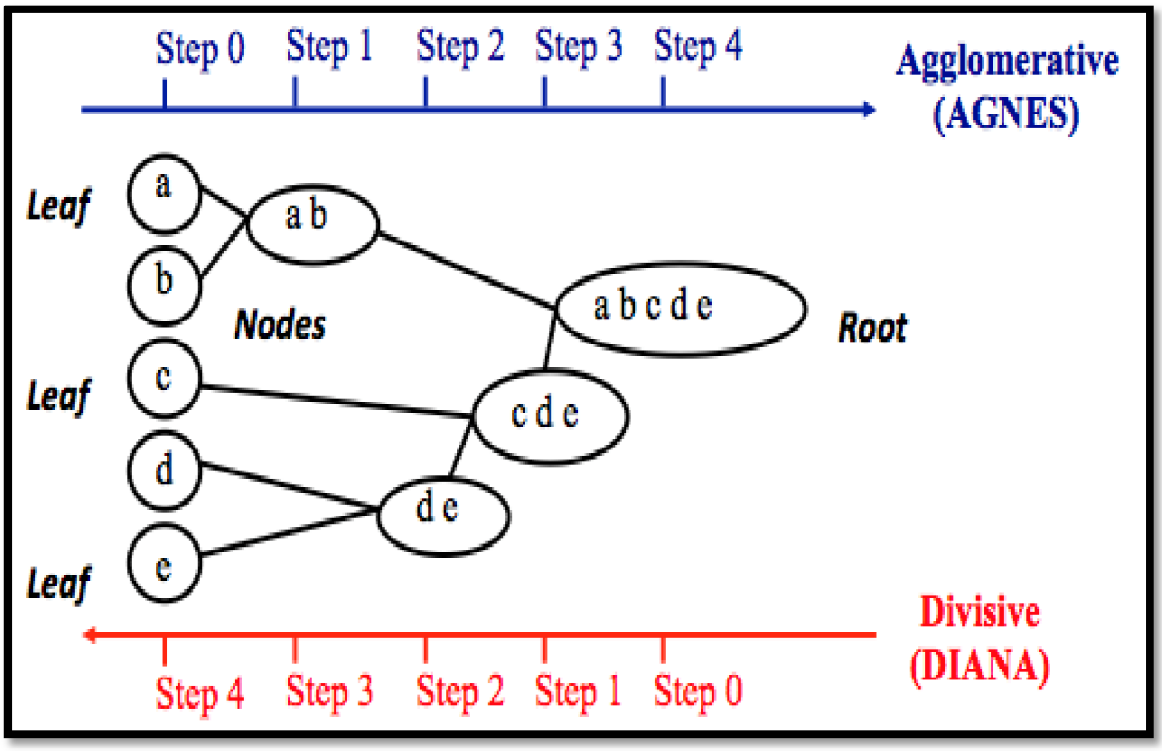
5) Her verir noktasıyla, yeni küme merkezleri arasındaki mesafeyi yeniden hesapla.

6) Eğer hiçbir veri noktası atanmadıysa dur, diğer durumda üçüncü adımdan itibaren tekrar et (URL4).

Tezimizin bu bölümünde anlatılan K-Means algoritması, karşılaştırmalarda kullanılmak üzere seçilmiştir. K-Means algoritmasının çalışması sonucu elde edilen sonuçlar üçüncü bölüm olan "Analiz" kısmında verilmiştir. Weka ve Matlab'da üç farklı veri kümesi üzerinde denenen K-Means algoritmasından elde edilen sonuçların kıyaslanması dördüncü bölüm olan "Sonuç ve Öneriler" kısmında açıklanmıştır.

Hiyerarşik kümeleme algoritması

Hiyerarşik algoritmalar iki başlık altında incelenirler. Bunlar AGNES (Agglomerative Nesting) yani Aglomerativ Kümeleme ve DIANA (Divise Analysis) yani Bölücü Hiyerarşik Kümeleme'dir. AGNES'de aşağıdan yukarıya doğru bir kümeleme mantığı vardır. Verilerin her biri başlangıç aşamasında birer küme olarak kabul edilir ve bunlar arasından en benzer olan ikililer kümelenir. Bu işlem kümelenecek başka bir veri kalmayıncaya kadar devam eder. Sonuç ağacı ise dendrogramda gösterilir. Şekil 2.9'da inceleyebilirsiniz (URL5).



Şekil 2.9 : Dendrogram Örneği

DIANA'da ise AGNES'e göre tam ters bir mantık görülür. Yukarıdan aşağıya doğru kümelere böler. Bütün kümelerde tek bir veri kalıncaya kadar bu işlem devam eder. Şekil 2.9'da inceleyebilirsiniz.

Hiyerarşik kümelemede, kümeler arasındaki benzerlikler ve yakınlık, farklı yöntemlerle belirlenebilir. Aşağıda öncelikle bu yöntemlerin isimleri, daha sonraki başlıklarda ise açıklamaları ve örnekleri verilmiştir.

Bunlar:

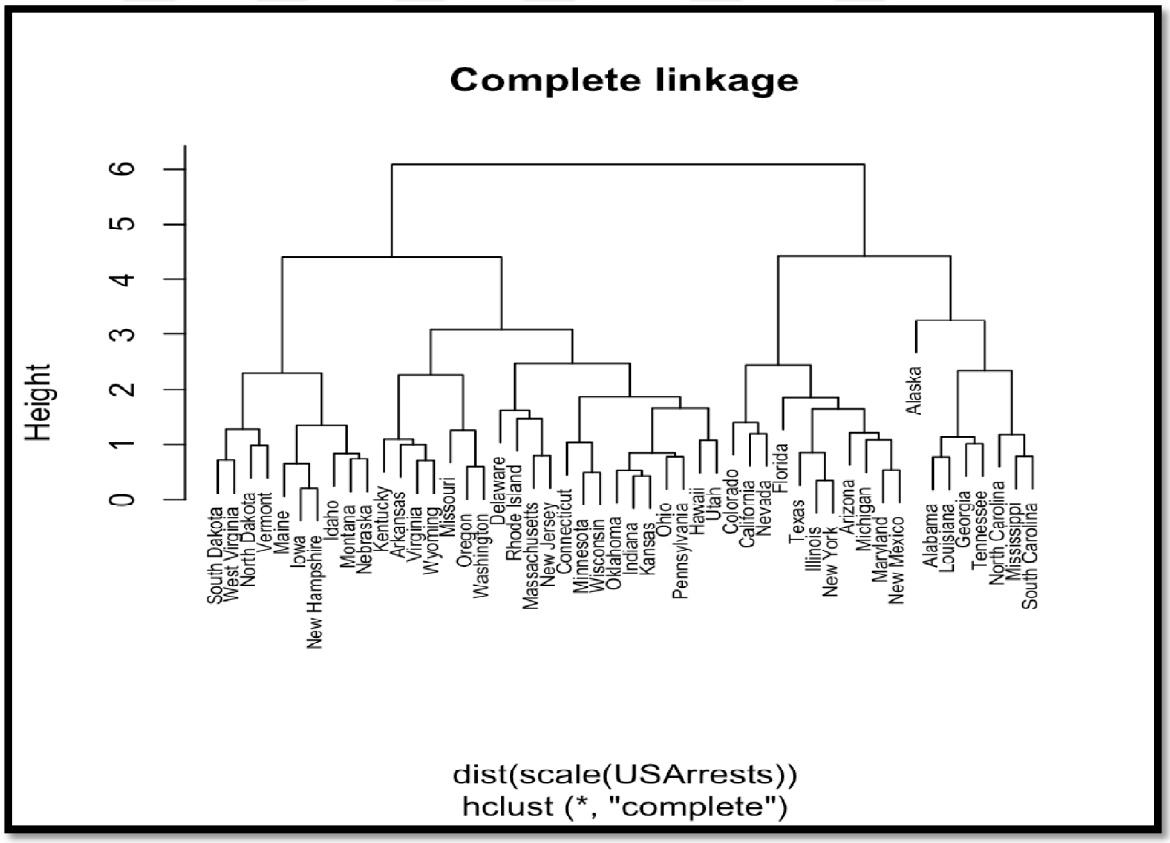
- Tam Bağlantılı Kümeleme
- Tek Bağlantılı Kümeleme
- Ortalama Bağlantılı Kümeleme
- Ward'ın Minimum Varyans Yöntemi

Tezimizde karşılaştırma aşamasında kullanılmak üzere seçilen hiyerarşik kümeleme algoritması gerek WEKA, gerek MATLAB platformlarında çalıştırılmış olup üçüncü bölüm olan "Analiz" kısmında kullanılmıştır. Elde edilen sonuçların karşılaştırılması ise dördüncü bölüm olan "Sonuç ve Öneriler" bölümünde verilmiştir.

Özellikle WEKA'da Hiyerarşik algoritmanın kullanılmasında WEKA platformundaki hazır fonksiyonlar sayesinde kümeleme işlemi aşağıda belirttiğimiz yöntemler denenerek çalıştırılabilmiştir. Üç veri kümesi üzerinde çalıştırılan Hiyerarşik kümeleme algoritması, sadece WEKA platformunda toplamda 12 kez çalıştırılarak bu yöntemlerin verilerimiz üzerindeki sonuçlarını net bir şekilde göstermiştir.

Tam bağlantılı kümeleme

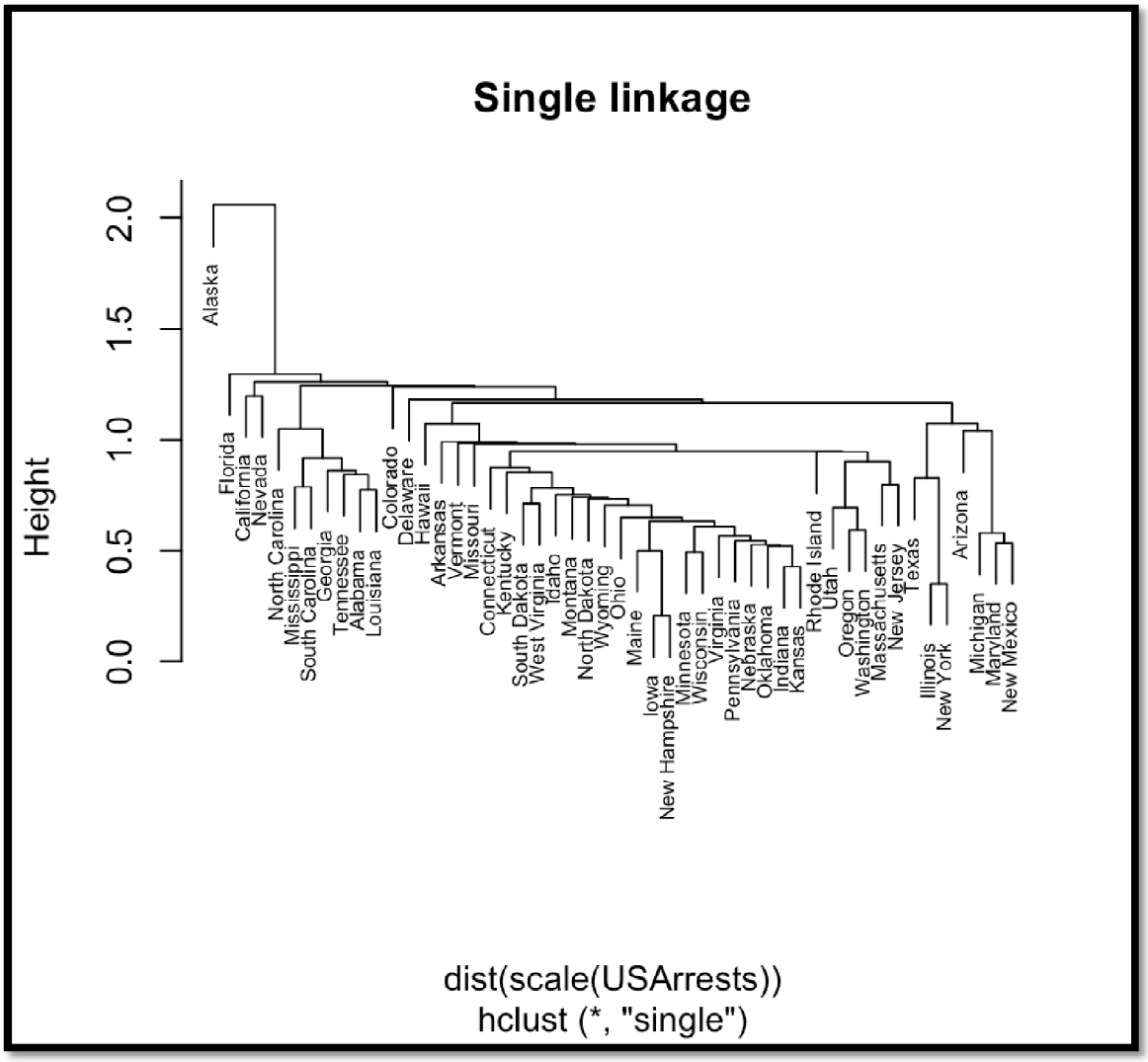
Tam bağlantılı kümelemeyi incelerken Şekil 2.10'da ki dendrogramdan faydalanacağız. Birinci ve ikinci aşamalarda algoritmamız tüm ikililerin farklılıklarını hesaplıyor ve bu farklılıkları iki küme arasındaki mesafe olarak kullanıyor (URL5).



Şekil 2.10 : Tam Bağlantılı Kümeleme

Tek bağlantılı kümeleme

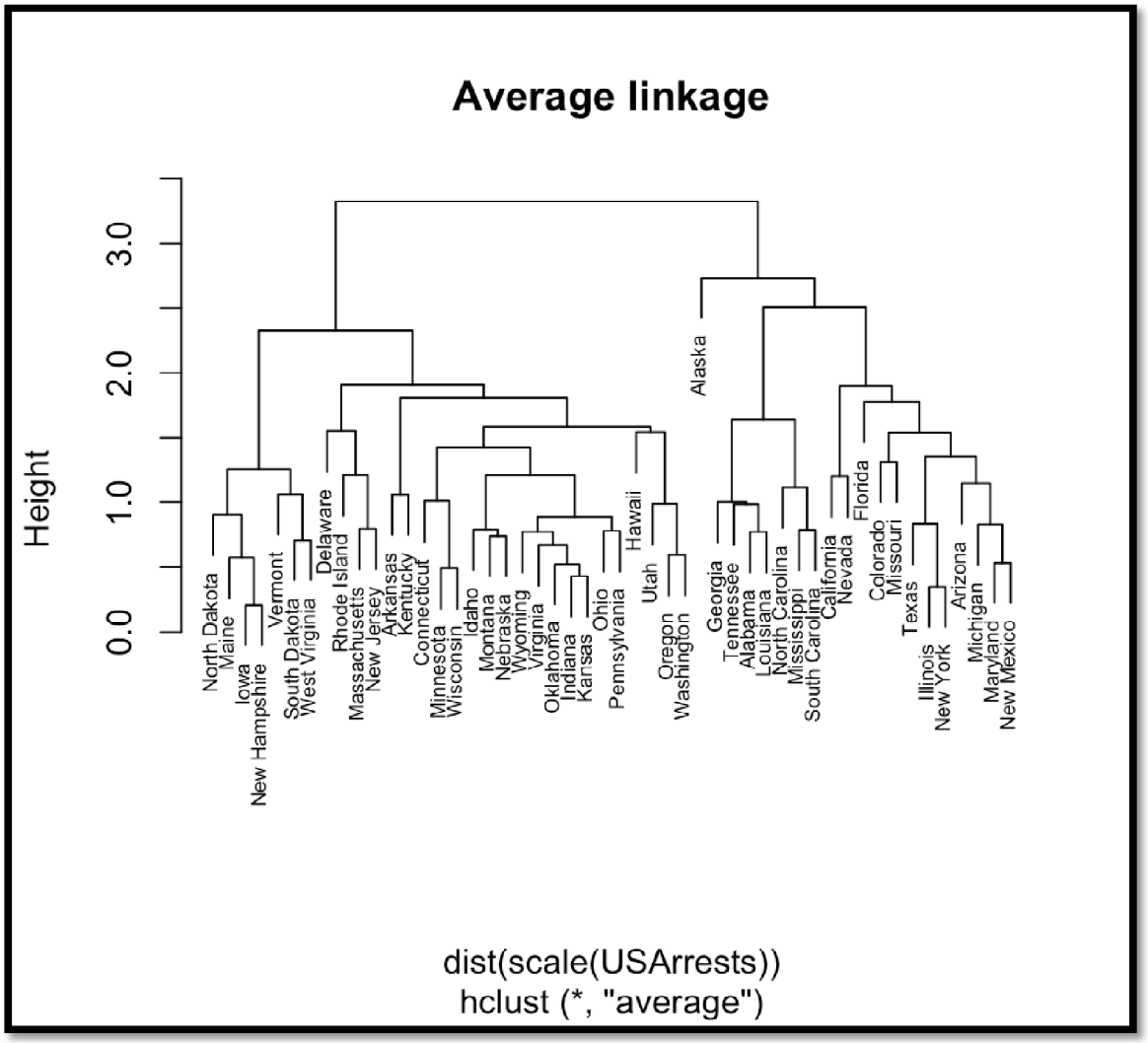
Tek bağlantılı kümeleme algoritması, birinci ve ikinci aşamadaki kümelerin ikililerinin farklılıklarını hesaplayıp, bunların en ufak farklılığını bağlantı kriteri olarak belirleyerek çalışır. Şekil 2.11'de ki dendrogramda, tek bağlantılı kümeleme örneğini inceleyebilirsiniz (URL5).



Şekil 2.11 : Tek Bağlantılı Kümeleme

Ortalama bağlantılı kümeleme

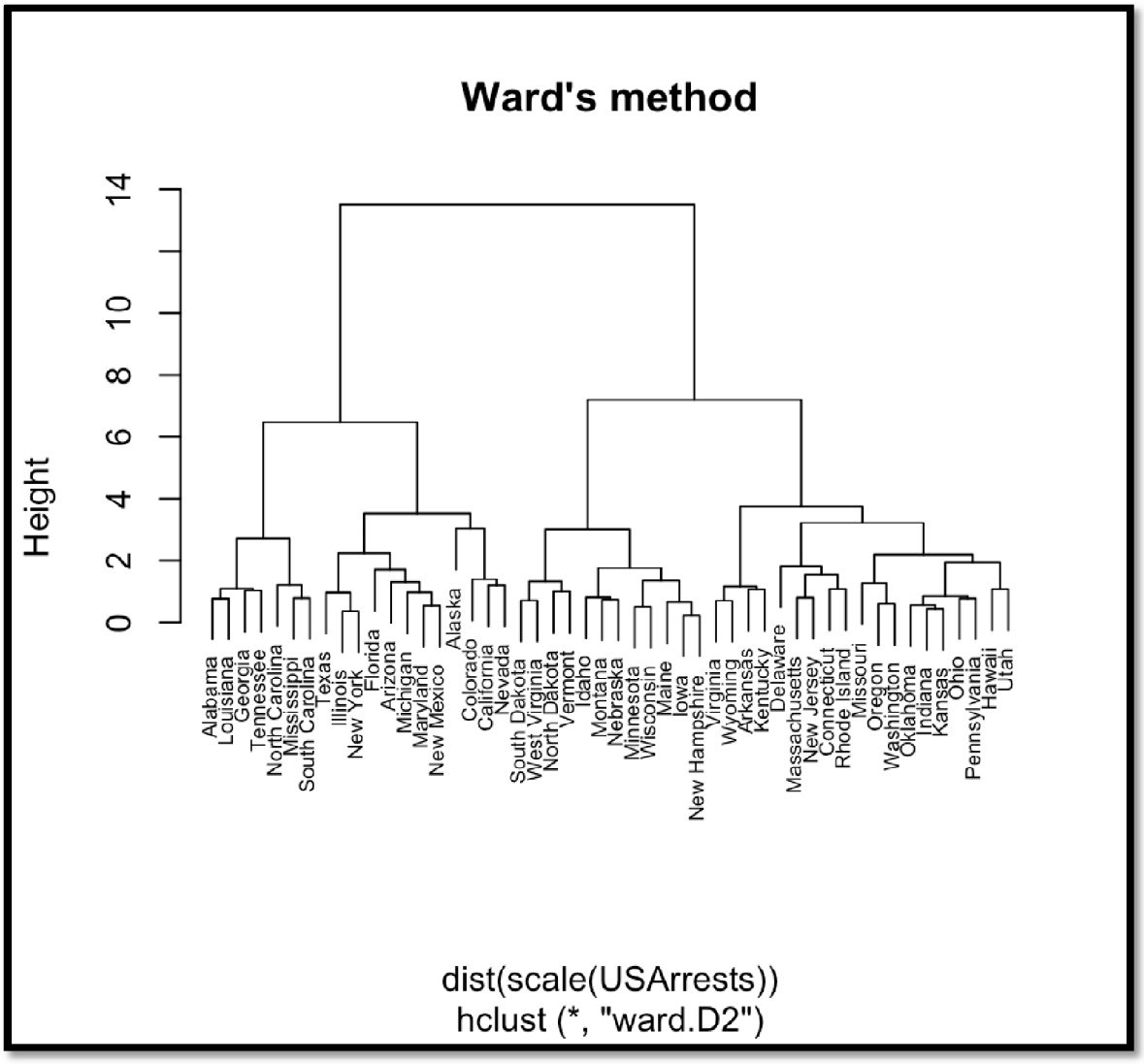
Ortalama bağlantılı kümeleme algoritması, birinci ve ikinci aşamadaki kümelerin ikililerinin farklılıklarını hesaplayıp, bunların ortalama farklılığını iki küme arasındaki mesafe olarak kabul ederek çalışır. Şekil 2.12'de bu algoritmanın dendrogramını inceleyebilirsiniz (URL5).



Şekil 2.12 : Ortalama Bağlantılı Kümeleme

Ward'ın minimum varyans yöntemi

Bu yöntemde küme içi varyans minimize edilir. Her adımda mesafesi en az olan iki küme birleştirilir. Şekil 2.13'de bu algoritmanın dendrogramını inceleyebilirsiniz (URL5).



Şekil 2.13 : Ward Yöntemi

Naive implementation diye bilinen yöntemler ile yukarıda açıklanan verilerin arasında ki mesafeler hesaplanır. Hiyerarşik kümelemenin karmaşıklığını hesaplamak için $O(N^3)$ yada $O(N^2 \log N)$ denklemleri kullanılır. Ayrıca hiyerarşik kümelemenin çok büyük verilerde başarısız sonuçlar verdiğini belirtmekte fayda vardır.

Spektral kümeleme algoritması

Spektral kümeleme algoritmasında, kullanılacak matris işleme sokulmadan önce kesilir. Yani matris boyutlarında ufalma olur. Elde edilen matris ile daha kolay kümeleme işlemi yapılır. Bu kesme işlemi genelde üç şekilde yapılır: Minimum, Average (Ratio) ve Normalized olarak. Spektral algoritmayı incelerken "Normalize (Normalized) Kesim" üzerinde duracağız.

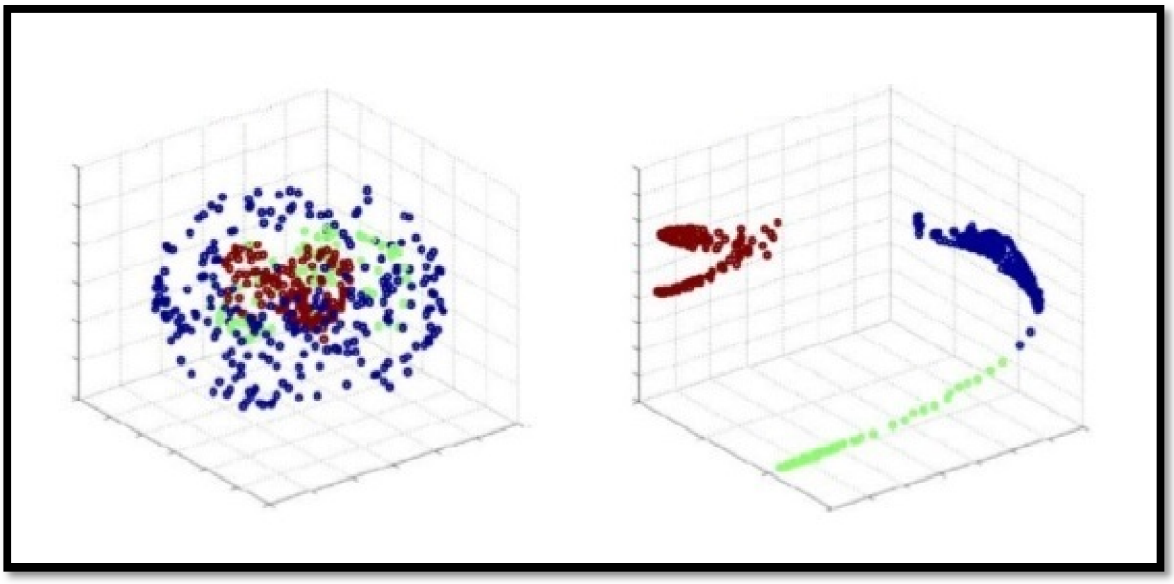
Benzer verilerden oluşan simetrik bir A matrisi düşünelim. $A_{ij} \geq 0$ olmak üzere i ve j arasındaki benzerlik değeri olsun. Normalize kesim algoritması matris noktalarını *v* özvektörünü esas alarak B_1 ve B_2 kümelerine böler. Bu bölme işlemi simetrik normalize laplace denklemi (2.6) ile ifade edilir.

$$L = I - C^{-1/2} AC^{-1/2} \quad \textbf{(2.6)}$$

Burada C, köşegen matristir ve (2.7)'de ki gibi hesaplanır.

$$Cii = \sum_j A_{ij} \quad \textbf{(2.7)}$$

Spektral kümeleme daha çok görüntü verileri üzerinde kullanılmaktadır. Dışbükey verilerin kümelenmesinde çok etkilidir. Şekil 2.14'de, S. Y. Kim'in, spektral kümeleme algoritması uygulanmadan önceki ve sonraki hallerinin gösterildiği örneği inceleyebilirsiniz (URL6).



Şekil 2.14 : Spektral Kümeleme Örneği

Tezimizde günümüzün popüler kümeleme algoritmalarından Spektral kümeleme algoritması karşılaştırmalar için seçilememiştir. Bunun sebebi algoritmamızın Weka platformunda yaşadığı problemlerdir. Spektral algoritma Weka'da programla gelen yada sonradan indirilebilen bir kümeleme algoritması değildir. Bu sebeple algoritmanın Java'da yazılıp platforma entegre edilmesi gerekir. Java'da mükemmel sonuçlar verdiği halde algoritmamız Weka'ya entegre edildiğinde sağlıksız sonuçlar ve programda hatalara sebep olmuştur. Bu sebeple Spektral kümeleme algoritması karşılaştırılacak algoritmalar arasına seçilmemiştir.

Mean-shift kümeleme algoritması

Mean-shift kümeleme algoritması günümüz popüler kümeleme algoritmalarından biridir. K-means algoritmasında olduğu gibi, oluşturulacak küme sayısını önceden belirtmemize gerek yoktur. Veri kümelerinde önceden belirtilen herhangi bir şekil oluşmaz. Oluşan şekiller kullanılan verinin işlenmezi sonucunda çıkan özgün şekilerdir. Bant genişliği (Bandwidth) parametresinin seçimine göre çalışır. Algoritmanın çalışacağı pencere boyutu çok önemlidir, dikkatle seçilmelidir.

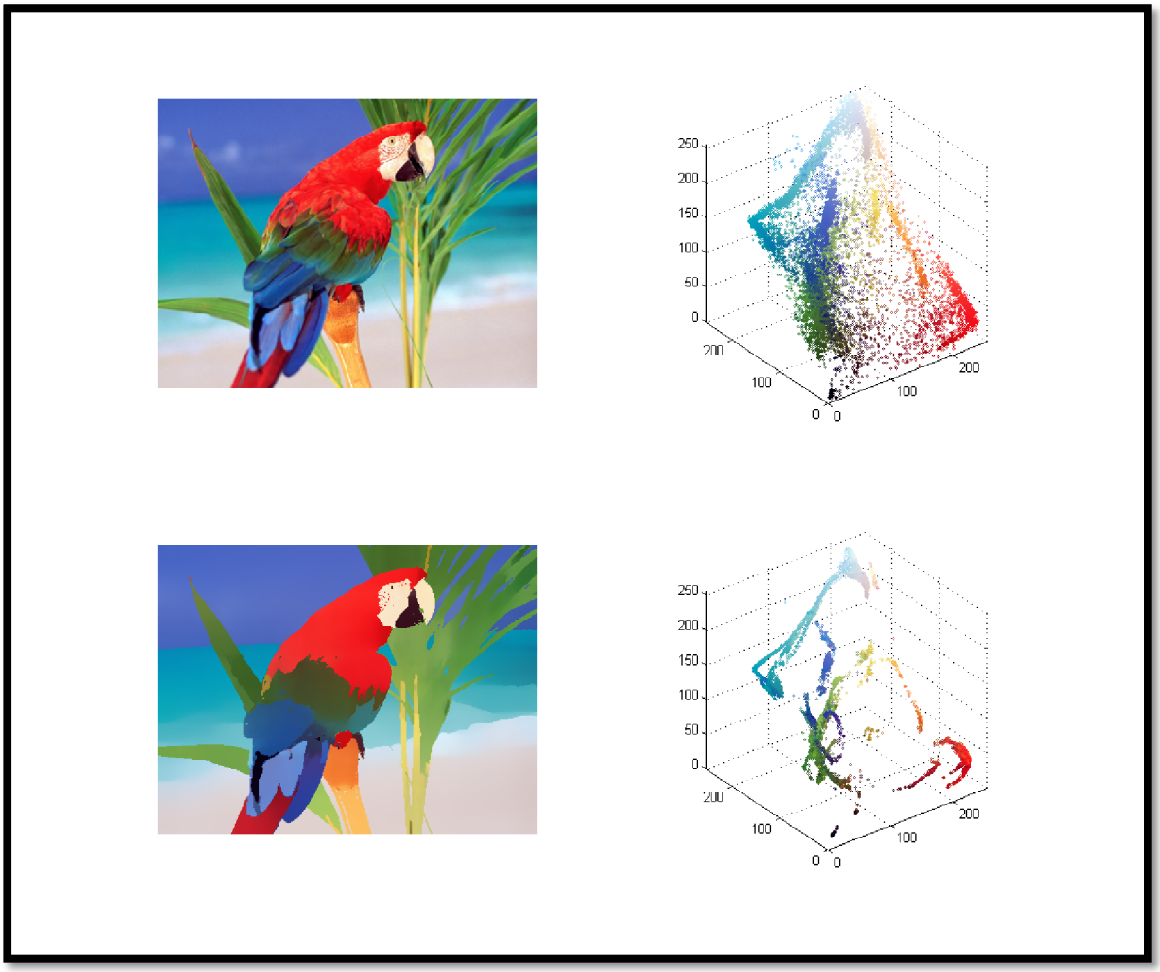
Mean-shift algoritması birleşme işlemi bitene kadar tekrar eden bir yöntemdir. Tüm veriler kendilerine benzer olan kümelere dağıtıldığında sonlanır. Ayrıca görüntü verilerinin kullanılacak uygun formata dönüşümü sonrası mean-shift ile kümelenmesi çok yaygın bir durumdur. Flat (Flat Kernal) ve Gauss çekirdek (Gaussian Kernal) metodları mean-shift kümelemesinde sıkça tercih edilen iki yöntemdir. Gauss çekirdek metodu, verilerin merkeze olan yakınlığını daha net gösterdiği için tercih edilirliği Flat çekirdek yöntemine nazaran daha fazladır. Tezimde Gauss çekirdek metodu incelenmiştir.

Tahmini bir x değeri ile başlayalım. Çekirdek fonksiyonu $K(x_i - x)$ verilmiş olsun. Bu fonksiyon yakınsal noktaların ağırlığına göre ortalama değeri belirler. Tipik Gauss çekirdek methodu kullanılarak aradaki mesafe tahmin edilir. Yoğunluk ağırlıklı ortalama K tarafından belirlenir. (2.8)'de görebilirsiniz.

$$K(xi - x) = e^{-c||x_i - x||^2} \quad \textbf{(2.8)}$$

$$m(x) = \frac{\sum_{xi \in N(x)} K(x_i - x) x_i}{\sum_{xi \in N(x)} K(x_i - x)} \quad \textbf{(2.9)}$$

(2.9)'da hesaplanan m(x) - x farkına mean shift denir (Fukunaga ve Hostetler, 1975). Ayrıca Şekil 2.15'de Yue Wu'nun MathWorks® platformunda paylaştığı "Mean Shift Pixel Cluster" projesinde, mean shift algoritması uygulanmadan önce ve sonraki hallerinin paylaşıldığı örneği inceleyebilirsiniz (URL7).

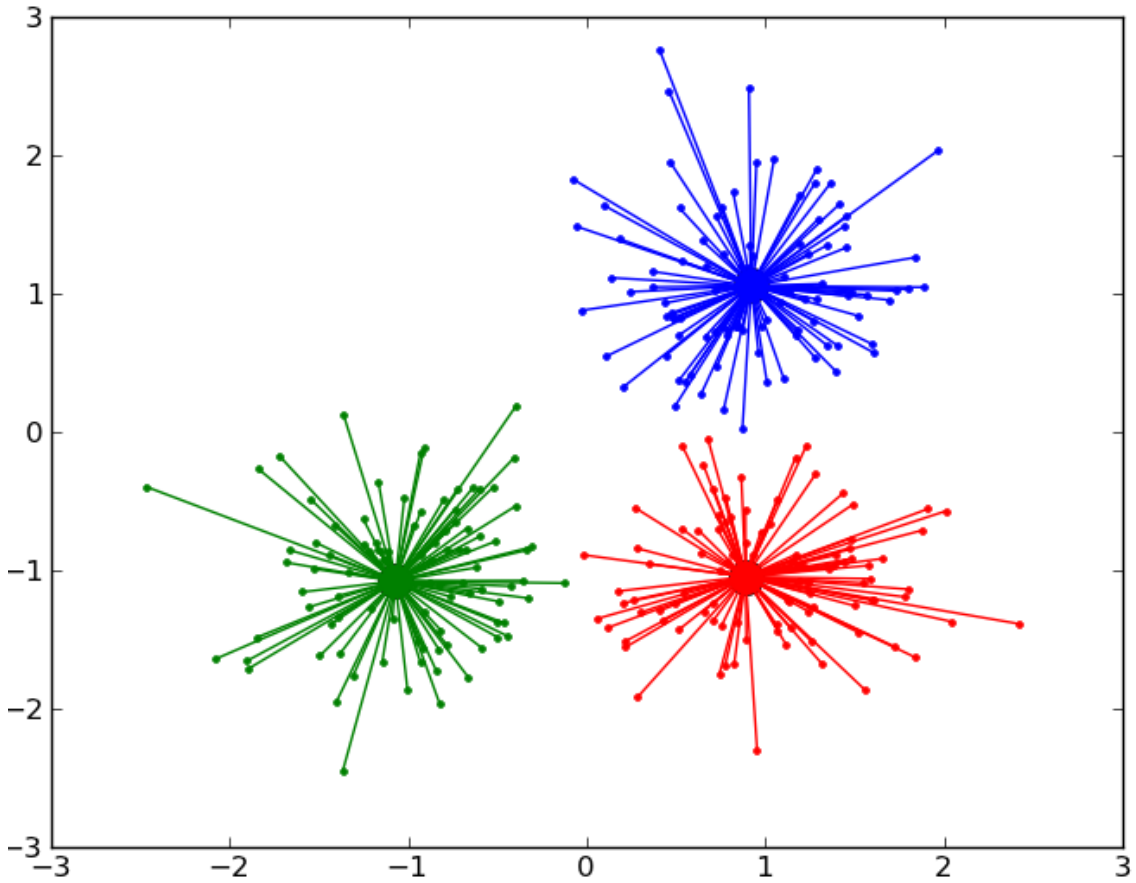


Şekil 2.15 : Mean Shift Örneği

Mean-Shift algoritması tezimizde karşılaştırılacak algoritmalara seçilememiştir. Bunun sebebi kullandığımız verilerin algoritmayla uyumsuz olmasıdır.

Affinity propagation kümeleme algoritması

Affinity propagation kümeleme algoritmasında veri noktaları arasındaki veri geçişi esas alınır. Bu kümeleme algoritmasında da önceden oluşacak küme sayısını belirtmeye gerek yoktur. Büyük veriler kullanıldığında ölçeklendirmede problemler yaşanılabilir. Ancak veri büyüklüğü binlere ulaşmayan örneklerde hızlıca kümeleme işlemini gerçekleştirir.



Şekil 2.16 : Affinity Propagation Örneği

Şekil 2.16'da affinity propagation kümeleme algoritmasının, sentetik iki boyutlu üç veri kümesi ile yapılmış örneğini inceleyebilirsiniz (Dueck ve Frey, 2007). Algoritma hakkında ise şu bilgileri verebiliriz:

x_1'den x_n'e kadar haklarında herhangi bir bilgimizin olmadığı veri noktalarımızın dizisi olsun. s ise iki nokta arasındaki benzerliği belirleyen fonksiyon olsun. Örneğin $s(x_i,x_j) > s(x_i,x_k)$ ve x_i'nin x_j'ye, x_k'dan daha benzer olduğu durumlar.

Algoritma çalışırken iki durum ön plana çıkar."Responsibility" matrisi R, r(i,k) ve "availability" matrisi A, a(i,k). İki matriste başlangıçta sıfır atanarak doldurulmuştur. Daha sonra algoritmanın çalışmasıyla aşağıdaki (2.10), (2.11), (2.11a) adımları tekrar ederek bu matrisdeki değerler değiştirilirler.

$$r(i,k) \leftarrow s(i,k) - \max_{k' \neq k}\{a(i,k') + s(i,k')\} \quad \textbf{(2.10)}$$

$$a(i,k) \leftarrow \min\left(0, r(k,k) + \sum_{i' \notin \{i,k\}} \max(0, r(i',k))\right) \quad \textbf{(2.11)}$$

$$i \neq k, \; a(k,k) \leftarrow \sum_{i' \neq k} \max(0, r(i',k)) \quad \textbf{(2.11a)}$$

(2.10)'da responsibility matrisinde ki değişim yapılır. Daha sonra (2.11) ve (2.11a)'da availability matrisinde ki değişim yapılır(Dueck ve Frey, 2007).

Affinity Propagation algoritması tezimizde karşılaştırılacak algoritmalar arasına seçilememiştir. Weka platformunda yaşanılan aksaklıklar bu sonucu doğurmuştur. Yayınlanan unofficial paketler ve kendi kodlarımız denenmiş olmasına rağmen, Spektral kümelemede yaşanılan benzer sorunlarla karşılaşılmıştır. Bu sebeplerden dolayı Affinity Propagation algoritması karşılaştırılacak algoritmalara seçilmemiştir.

DBSCAN kümeleme algoritması

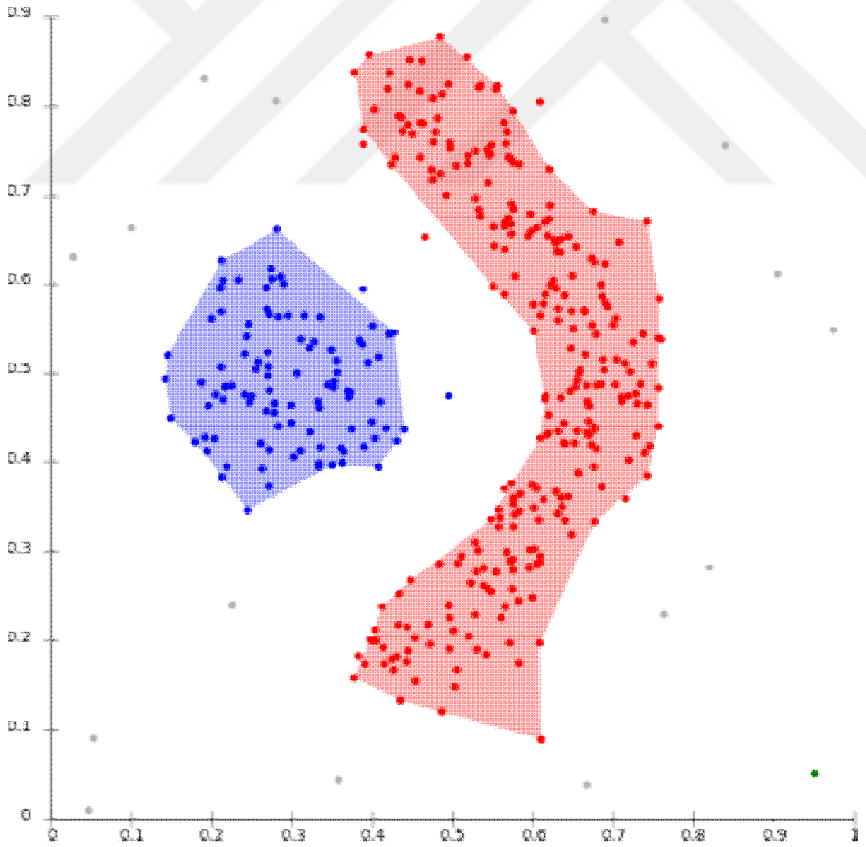
DBSCAN (Density-Based Spatial Clustering of Applications With Noise) algoritması Martin Ester, Hans-Peter, Jörg Sander ve Xiaowei Xu tarafından 1996 yılında önerilmiştir (Ester, Kriegel, Sander ve Xu,1996). DBSCAN, density-based spatial clustering of applications with noise, algoritması günümüzde çokça kullanılan popüler bir algoritma olup, yoğunluk temel alınarak tasarlanmış ve veritabanlarıyla çalışabilen bir algoritmadır.

DBSCAN algoritması, veri noktalarının yoğunluğunun fazla olduğu yerlerde ki verileri gruplar. Bu gruba uzak ve ses olan noktalar, yoğunluğu düşük olan bölgeler olarak işaretlenir. DBSCAN algoritmasında, algoritma çalıştırılmadan önce oluşturulacak küme sayısının belirtilmesi istenmez. Algoritma sese duyarlıdır, ayrıca çalıştıktan sonra oluşan şekiller alışılageldik küme şekillerinde olmaz. Bunun sebebi algoritmada ki "MinPts" parametresidir.

Oluşan bu farklı şekiller, algoritmanın sağlıksız çalıştığı anlamına gelmez. Dikkatle incelendiğinde, algoritmanın çalışması sonucu elde edilen şekillerde, kümelerin birbirine karışmadığını görebiliriz.

DBSCAN algoritmasında iki parametre kullanılır. Bunlardan biri biraz önce bahsedilen 'MinPts' ve diğeri ise 'ε' parametresidir. Parametrelerin atanmasında kullanılan verilerin iyi kavranması büyük önem arz etmektedir. Çünkü hatalı parametre atamaları algoritmanın sağlıksız çalışmasına sebep olur. Bunun sebebi genellikle kullanılan veriler hakkında yeterli bilgi sahibi olunmamasından kaynaklanır.

Şekil 2.17'de DBSCAN algoritmasına örnek bir görüntü bulunmaktadır. Bu görüntüden anlaşıldığı üzere, oluşan şekiller K-Means veya diğer kümeleme algoritmalarında görmeye alıştığımız küme şekillerinden farklıdır.



Şekil 2.17 : DBSCAN Kümeleme Örneği

DBSCAN (Density-Based Spatial Clustering of Applications With Noise) algoritması karşılaştırdığımız kümeleme algoritmalarına seçilmemiştir. Bunun sebebi Weka platformunda yaşanılan problemlerdir. Weka bünyesinde DBSCAN algoritmasına benzer bir algoritma bulundurmaktadır. Ancak yapılan işlem ve elde edilen sonuçlar DBSCAN algoritmasıyla benzerlik göstermemektedir. Yaptığımız araştırmalarda Weka'da çalıştırılacak yayınlanmış unofficial paketlere rastlanmamış ve daha önceki denemelerimizde yaşadığımız sorunlardan edindiğimiz tecrübeler ışığında, Java'da bu algoritma için herhangi bir kod geliştirme ihtiyacı duyulmamıştır. Bu sebeplerden dolayı karşılaştırılacak algoritmalar arasına seçilmemiştir.

Expectation maximization algoritması

Popüler bir algoritma olan EM, Expectation Maximization, tekrarlamalı bir iyileştirme algoritması olarak bilinir. İki adımdan oluşur. Bunlar E ve M adımlarıdır. E adımında algoritma genellikle Kalman filtresini kullanarak verinin son ve güncel durumunu (state) tahmin eder. M adımında ise bu E adımından gelen verileri kullanılarak, maksimum benzerliği hesapladıktan sonra parametrelerin merkezleri için iyileştirme yapar. Hızlı çalışan bir algoritmadır ama elde ettiği sonuçlar yüksek tutarlılık oranına sahip olmayabilir. Ayrıca expectation maximization algoritması birçok açıdan K-means algoritmasına benzetilir. Hatta K-means uzantısı olarak da düşünülebilir. Expectation ve Maximization adımları için aşağıdaki (2.12) ve (2.13)'da ki formüller kullanılır. θ, bilinmeyen parametredir. q, dikkat edilmemiş veriler üzerinde keyfi bir olasılık dağılımıdır.

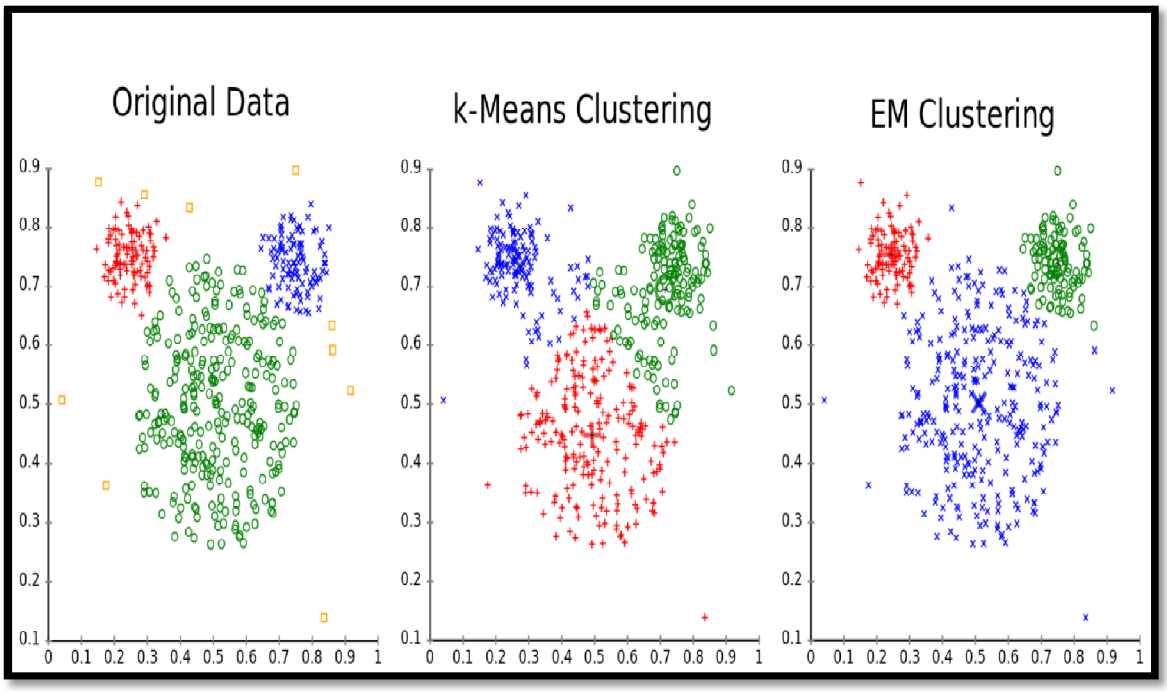
E adımında F'yi maksimize etmek için q seçilir. Fonksiyon aşağıdaki gibidir.

$$q^{(t)} = \arg_q \max F(q, \theta^{(t)}) \quad \textbf{(2.12)}$$

M adımında F'yi maksimize etmek için θ seçilir. Fonksiyon aşağıdaki gibidir.

$$\theta^{(t+1)} = \arg_\theta \max F(q^{(t)}, \theta) \quad \textbf{(2.13)}$$

Aşağıda ki Şekil 2.18'de K-means ve EM algoritmaları ile yapılmış bir çalışmanın sonuçları gözükmektedir. Bu örnekleri inceleyip algoritmalar hakkında fikir sahibi olabilirsiniz (URL8).



Şekil 2.18 : K-means ve EM Kümeleme Algoritmaları Örneği

Tezimizde Expectation Maximization algoritması karşılaştırılacak algoritmalardan biridir. Weka ve Matlab platformunda başarı ile çalıştırılabilen bir algoritmadır. Üç veri kümesi ile üçüncü bölüm olan "Analiz" kısmında kullanılmıştır. Elde edilen sonuçların karşılaştırılması ise dördüncü bölüm olan "Sonuç ve Öneriler" kısmında verilmiştir.

Sonuç olarak bu bölümde açıklanan güncel ve popüler kümeleme algoritmaları olan; Hiyerarşik Kümeleme, K-Means, DBSCAN, Spektral Kümeleme, Mean-Shift ve Expectation Maximization algoritmalarından daha önce belirtildiği gibi K-Means, Hiyerarşik Kümeleme ve Expectation Maximization Kümeleme algoritmaları seçilmiş, Weka ve Matlab platformlarında uygulanılmıştır.

2.2.4. Veri madenciliği, weka, matlab ve wavelet

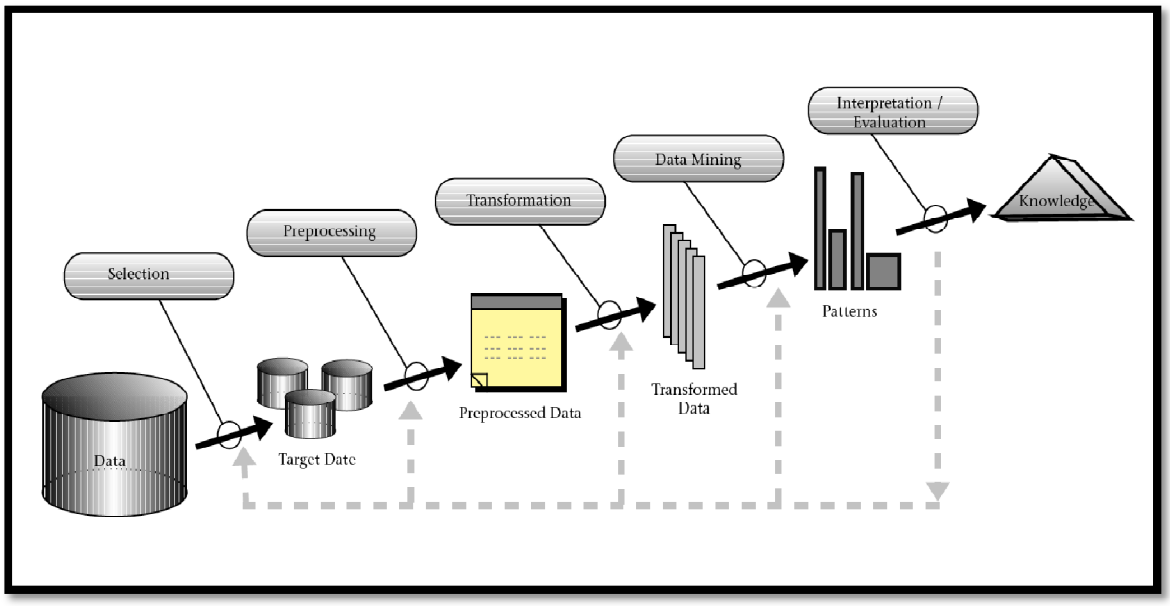
2.2.4.1. Veri madenciliği

Veri madenciliği, adından da anlaşılacağı üzere bilimsel bir madencilik işlemidir. Nasılki günümüz madenciliğinde çalışılacak alan öncelikle araştırılır, ne aranılacağı belirlenir, saha keşifi yapılır.

Arazinin temizlenmesi ve madenciliğe uygun bir hale getirilirmesi, kullanılacak ekipmanın belirlenmesi gibi madenciliğe başlamadan yapılması gereken ön hazırlıklar varsa, veri madenciliğinde de kullanılacak veri merkezleri ve türevlerinde benzer hazırlıklar yapılır ve bunun gibi birçok birbiriyle bağlantılı adımlar bütününe veri madenciliği denir.

Anlaşılacağı gibi veri madenciliği ismi, bütün sürecin genel adı olsada, aslında bu sürecin adımlarından biridir. Peki veri madenciliği için bir tanım yapmamız gerekirse nasıl bir tanım yapabiliriz? Veri madenciliği, elimizde bulunan veri yığınından, spesifik bir konu ile alakalı, tahmin yada çıkarımda bulunmamızı sağlayan bir bilgi çıkarımı işlemidir. Veri madenciliğine getirilmiş bazı yaklaşımlar bulunmaktadır. Bunlardan ön plana çıkmış olanlarından biri de KDD(Knowledge Discovery in Databasess)'dir.

Veri madenciliği, veritabanlarında bilgi madenciliği işleminin analiz adımıdır. Ve KDD, Knowledge Discovery in Databasess, işlemi şu adımlardan oluşur: Seçim (selection), önişleme (pre-processing), dönüştürme (transformation), veri madenciliği (data mining) ve yorumlama/değerlendirme (interpretation/evaluation). Şekil 2.19'da KDD sürecini inceleyebilirsiniz (Fayyad, Piatetsky-Shapiro, Smyth, 1996).



Şekil 2.19 : Veri Madenciliği Aşamaları

Veri madenciliğinde yapılan işlemleri kısaca açıklamak gerekirse, verinin temizlenmesi sürecin ilk adımıdır. Verinin kullanıma sokulmadan önce gürültünün ve alakasız verilerin ayıklanması gerekmektedir. Bu düzenlemelere verinin temizlenmesi denilir. Temizlenmiş verilerde bütünleştirme yapılır. Çünkü verilerin kaynaklarında farklılıklar olabilir. Verinin bütünleştirilmesi sayesinde, kullanılan tüm verilerin kaynağı bir olur. Temizlenmiş ve kaynakları bir olan bu veriler üzerinde seçme işlemi yapılır. Seçme işlemi, analizde kullanılacak olan alakalı verileri belirlemek amacıyla yapılır. Daha sonra bu veriler kullanılacak programın işleyebileceği bir forma dönüştürülür. Bütün bu yapılan adımlar, veri madenciliği adımına verilerimizi hazırlamak için yapılan ön işlemlerdir. Veri madenciliğinde ise kümeleme, sınıflandırma, genetik algoritmalar ve benzeri birçok işlem yapılarak çıkarımlar yapılır. Veri madenciliğinde elde edilen çıkarımlar değerlendirilir ve bunlar kullanıcıya sunulur. Bu adımların tümüne veri madenciliği denir. Günümüzde birçok alanda yararlanılan veri madenciliği sayesinde birçok ekonomik kazanım sağlanır ve olası olumsuzluklara karşı önlem alınır.

Veri madenciliğinde kullanılan birçok program bulunmaktadır. Bunlardan açık kaynak kodlu olanlarını söylemek gerekirse; R, Orange, OpenNN, Carrot2, Gate, Elki, Torch ve Weka gibi örnekler verilebilir. Burada bahsedilenler dışında, yapılacak veri madenciliğiyle alakalı daha uygun programlarda bulunabilir. Ayrıca ücretli birçok programlar da vardır. Bunlardan bazıları; RapidMiner, IBM SPSS Modeler, Microsoft Analysis Services, Oracle Data Mining, Clarabridge gibi programlardır.

2.2.4.2. Weka

Günümüzde üzerine çokça araştırma yapılan Weka, Yeni Zelanda'nın Waikato Üniversitesi tarafınca ücretsiz GNU lisansı ile üretilmiş modüler bir veri madenciliği uygulamasıdır. Bünyesinde birçok method, algoritma, hazır fonksiyon ve kütüphane bulunmaktadır. Modüler özelliği sayesinde yeni geliştirilen yada standart programla birlikte gelmeyen birçok özellik; fonksiyon, algoritma, method vb. gibi, kullanıcının isteği doğrultusunda weka platformundan ücretsiz bir şekilde indirilip programa entegre edilerek kullanılabilir.

Weka programı geliştirilme aşamasında Java programlama dili kullanılarak üretildiği için kütüphaneleride .jar uzantılıdır. Bu, kullanıcılara büyük kolaylıklar sağlamaktadır. Uzantısının .jar olması, Java ile üretilmiş birçok program yada projeye entegre işleminde büyük kolaylıklar sağlar.

Veri madenciliği aşamasında Weka platformunda; sınıflandırma, kümeleme, ilişkilendirme, veri ön işleme ve görselleme işlemleri kolayca yapılabilmektedir. Bu işlemlerin yapılabilmesi için, kullanılacak verilerin uzantısının arff olması gerekmektedir. Ayrıca farklı uzantılardaki verilerin dönüşümü kolayca yapılabilmektedir.

Programın ihtiyaç duyduğu arff uzantılı dosyanın kullanıcı tarafından hazırlanacağını düşünürsek, kullanılacak dosyayı şu şekilde hazırlamamız gerekir. Öncelikle not defteri programında bir sayfa açıp @relation yazılıp yanına dosyamızın kullanılacağı iş ile alakalı bir isim verilir. Daha sonra @attribute yazılarak yanına özelliğin ifade ettiği kavramın ismi yazılır. Yanına uzantısı belirtilir. Bu uzantı; numeric, nominal, real, string ve date formatında olabilir. Bu özellikleri tanıttıktan sonra tanıtılan özelliklere değer girilir. Bunu yapabilmek için öncelikle @data yazılır ve altına aralarında virgül olacak şekilde belirtilen uzantıya uygun veriler girilir. Her satırda girilecek veriler bittiğinde, bir alt satıra geçip, sırayla veri girme işlemi elimizdeki tüm verilerin girilmesi bitene kadar tekrar eder.

Ufak çaplı bir uygulama yapılacağı zaman not defterinde kolayca arff uzantılı dosyamızı hazırlayabiliriz. Bu dosyayı kaydederken formatını arff yaparak kolayca işlemimizi tamamlamış oluruz. Aşağıdaki Çizelge 3.1'de örnek bir arff dosyası görebilirsiniz.

Çizelge 2.1 : Örnek Arff Dosyası

```
@RELATION growth

@ATTRIBUTE year NUMERIC
@ATTRIBUTE exportdollar NUMERIC
@ATTRIBUTE importdollar NUMERIC
@ATTRIBUTE population NUMERIC
@ATTRIBUTE gsyh NUMERIC
@ATTRIBUTE debt NUMERIC
@ATTRIBUTE unemployment NUMERIC
@ATTRIBUTE uneducatedpeople NUMERIC
@ATTRIBUTE growthratio NUMERIC

@DATA

2007,107271749904,170062714501,70586256,8430000000,250300000000,10.3,5347461,4.7
2008,132027195626,201963574109,71517100,9510000000,281000000000,11,4930012,0.7
2009,102142612603,140928421211,72561312,9530000000,269200000000,14,4672257,-4.8
2010,113883219184,185544331852,73722988,10990000000,291900000000,11.9,3825644,9.2
2011,134906868830,240841676274,74724269,12980000000,304200000000,9.8,3171270,8.5
2012,152461736556,236545140909,75627384,14170000000,336900000000,9.2,2788643,3.2
2013,151796483759,251650641956,76667864,16450000000,366300000000,8.8,2654643,4
2014,40274721485,57483983376,78607394,17350000000,388200000000,10,2234851,5
```

Yukarıdaki çizelge de gördüğünüz veriler, lisans eğitimimde Türkiye ile alakalı bazı veriler üzerinden, büyüme oranları hakkında forecasting formulünü kullanarak tahmin yaptığım çalışmamın verileridir. Arff dosyasını yukarıdaki kurallara uyarak kolayca hazırlayabilirsiniz.Tezimizin ana konularından biri olan Weka programı üçüncü bölüm olan "Analiz" aşamasında kullanılmış olup elde edilen sonuçlar yine aynı bölümde açıklanmıştır. Dördüncü bölüm olan "Sonuç ve Öneriler" kısmında, kümeleme algoritmaları karşılaştırıldığı gibi, Weka ve Matlab arasında da kazanılan tecrübeler ışığında bir karşılaştırma yapılmıştır.

2.2.4.3. Matlab

Matlab, MathWorks tarafından geliştirilen, dördüncü nesil, ücretli bir programlama dilidir. İstatistiksel, matematiksel, simulasyon ve birçok işlemlerin yapılabildiği Matlab, C, C++, Java, Fortran, Python gibi programlama dillerine uyumlu olarak çalışabilmektedir. Bu sebeple geniş bir kullanım alanı vardır. Özellikle endüstriyel ve akademik alanlarda çalışan kişilerin çokça faydalandığı bir uygulamadır.

Tezimizde belirlediğimiz üç kümeleme algoritmasını analiz etmek amacıyla Matlab programını kullanmayı tercih ettik. Kullanımı kolay, anlaşılır ve hızlı çalışan bir uygulama olan Matlab sayesinde algoritmalarımızın kodlarını çalıştırdığımızda, elde ettiğimiz sonuçları hem "Analiz" bölümünde açıklayıp, hem de bu sonuçlardan derlediğimiz verileri 2.2.4.4.'de açıkladığımız Wavelet Toolbox'da uyguladık ve ortaya çıkan sonuçları yorumladık.

2.2.4.4. Wavelet

Wavelet diye tezimizde kısaca belirtilen Wavelet Toolbox, Matlab'ın bir ürünü olup, kullanımı kolay, yüksek verimli ve güvenilir sonuçlar veren ücretli bir uygulamadır. Wavelet genellikle sinyal ve görüntü analizlerinde kullanılmaktadır. Kullanılan verinin kesinliğine göre elde edilen sonuçlar, doğru orantılı olarak, daha tutarlı olur. Wavelet, Matlab'ın bir uzantısı olduğu için Matlab'ın birçok özelliğine sahiptir. Bunlar fonksiyon komut satırı, analiz, sentez ve sıkıştırma, veriyi temizleme vb. gibi. Wavelet'in kullanıldığı birçok analiz ve işlemler daha önce Fourier tekniğiyle çözüm getirilen durumlardır. Malum Fourier sinyal spektumlarının ve frekans sistemlerinin analizinde ortaya atılan ilk yöntemdir. Wavelet ise Fourier dönüşümünün, zaman penceresi (time window) ile birlikte yorumunu yapabilmektedir. Ayrıca Çizelge 3.2'de Fourier dönüşümünün ve Wavelet dönüşümünün formülsel ifadelerini görebilirsiniz.

Çizelge 2.2 : Fourier Dönüşümü ve Wavelet Dönüşümü Formülsel İfadeleri

Dönüşüm Türü	Formülsel İfadesi	Değişken Tanımı
Fourier Dönüşümü	$f(\acute{\varepsilon}) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x \acute{\varepsilon}}\, dx$	$\acute{\varepsilon}$: frekans
Wavelet Dönüşümü	$X(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} \overline{\Psi\left(\frac{t-b}{a}\right)} x(t) dt$	a : artış b : zaman

Wavelet dönüşümünün birçok çeşiti bulunmaktadır. Bunlardan bazılarını aşağıda maddeler halinde görebilirsiniz.

* Sürekli Wavelet Dönüşümü

* Ayrık Wavelet Dönüşümü

* Karmaşık Wavelet Dönüşümü

* Çift Wavelet

* Çoklu Çözünürlük Analizi

* Gabor Wavelet

* Morlet Wavelet

* Jpeg 2000

*MrSID

*ECW

*DjVu

Yukarıda belirtilen çeşitlerinden başka, spesifik durumlar için kullanılabilecek birçok farklı çeşiti daha bulunmaktadır. Buradan da anlaşılacağı üzere Wavelet Toolbox çok geniş ve farklı alanlarda kullanılabilen bir uygulamadır. Örneğin konuşma ve ses işleme, görüntü ve video işleme gibi 1-D ve 2-D uygulamalarında da yüksek verimle kullanılabilmektedir.

Tezimizden türetilecek olan makalede Wavelet Toolbox uygulaması, Matlab'ta çalıştırdığımız algoritmaların çıkardığı sonuçların derlenmesi ile elde edilen verilerin incelenmesi ve yorumlanması amacıyla kullanılmıştır.

3. ANALİZ

3.1. Weka Analizleri

Bu bölümde, ele aldığımız üç kümeleme algoritmasının, Weka'da kullanılması sonucu elde edilen sonuçları açıklanmıştır. Kümeleme algoritmalarının üzerinde denendiği üç farklı veri kümemiz bulunmaktadır. Bu verilerin nerelerden edinildiği ve ne ile alakalı olduğu önceki bölümlerde belirtilmiştir.

Aşağıdaki üç başlıkta açıklanan sonuçlara ulaşılmasında ortak bazı adımlar bulunmaktadır. Bunlar; Weka platformunun çalıştırılması ve Explorer uygulamasının seçilmesi. Kullanılacak verinin açılması ve kümeleme algoritmaları sekmesinin seçilmesi. Bu kümeleme algoritmalarından kullanacağımız algoritmanın seçilip, üzerinde gerekli değişikliklerin yapılması gibi adımlardır.

3.1.1. K-Means kümeleme algoritması analizleri

Küme sayısı, k değeri, her üç veri kümesi işlemlerinde de 2'den 10'a kadar alınıp, her k değeri için 1'den 50'ye kadar seed değeri denenerek yapılmıştır. Aşağıdaki veri kümeleri için alınan sonuçlar; bu 1350 denemeden alınan en verimli sonuçlarıdır.

Kandilli ortalama rüzgar şiddeti verileri analizi

Bu veri kümesinin K-means kümeleme algoritmasına sokulması sonucu elde edilen en verimli sonuç aşağıda açıklanmıştır. Veriler arası mesafenin ölçümü için Öklid fonksiyonu kullanılmıştır. K değeri 10 ve seed değeri 44 olan kümelemede en verimli sonuca ulaşılmıştır. Algoritma bu sonuca ulaşmak için 6 kez tekrar etmiştir. Aşağıda WEKA'nın çalışması sonucu elde edilen sonuçların özeti verilmiştir. Tam çıktıyı ekler bölümünde bulunan Ek G'de inceleyebilirsiniz. Aşağıdaki Çizelge 3.1'de elde edilen özetlenmiş K-Means analizi sonuçlarını inceleyebilirsiniz.

Çizelge 3.1: K Means Kandilli Ortalama Rüzgar Şiddeti Analiz Çizelgesi

=== Run information ===					
Scheme: weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 10 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 44					
Relation: ortruzsid					
Instances: 64					
Attributes: 13					
yil, ocak, subat, mart, nisan, mayis, haziran, temmuz, agustos, eylul, ekim, kasim, aralik					
Test mode: evaluate on training data					
=== Clustering model (full training set) ===					
kMeans					
======					
Number of iterations: 6					
Within cluster sum of squared errors: 8.160678975344503					
Initial starting points (random):					
Cluster 0: 1998,3.1,3,3.9,3,2.8,2.6,3.1,3.1,2.8,2.8,3.3,4.1					
Cluster 1: 1965,4,5.2,3.1,2.9,2.2,3.4,3.8,4.1,3.5,3.5,5.3,4.6					
Cluster 2: 1964,4.3,4,2.8,3.4,3.2,3.1,4,3.1,2.7,1.6,2.8,5					
Cluster 3: 1995,4.3,3.5,3.9,3.1,3.3,2.8,3.2,2.9,2.6,3.1,3.8,4.1					
Cluster 4: 2000,3.5,3.1,2.8,2.2,2.6,3,2.4,3,2.7,2.6,2.1,2.7					
Cluster 5: 1990,4.2,4.9,4.7,3.9,4.9,4.2,5.8,5.3,4,4.4,4.6,4.7					
Cluster 6: 1978,4.3,4.5,4.6,3.4,3.6,3.2,4.4,4.3,3.8,3.8,3.7,4.5					
Cluster 7: 1988,4.2,5,4.8,4.6,3.6,2.6,4.3,4.7,4.1,5.5,5.8,5.2					
Cluster 8: 2010,4,3.3,2.9,3,2.4,2.5,2.8,2.4,2.6,2.7,2.8,3.8					
Cluster 9: 1957,3.7,3.2,5.1,3.6,2.3,4.1,3.9,4.3,3.3,3.5,3.3,5.7					
Time taken to build model (full training data) : 0 seconds					
=== Model and evaluation on training set ===					
Clustered Instances					
0	7 (11%)	4	1 (2%)	8	14 (22%)
1	4 (6%)	5	4 (6%)	9	8 (13%)
2	6 (9%)	6	11 (17%)		
3	3 (5%)	7	6 (9%)		

Kandilli maksimum, minimum ve ortalama sıcaklık verileri analizi

Bu veri kümesinin K-means kümeleme algoritmasına sokulması sonucu elde edilen en verimli sonuç aşağıdaki Çizelge 3.2'de açıklanmıştır. K değeri 10 olup, seed değeri 31 olan kümelemede algoritma 7 tekrarla en verimli sonuca ulaşılmıştır.

Çizelge 3.2: Kandilli Maksimum, Minimum ve Ortalama Sıcaklık Verileri Çizelgesi

=== Run information ===										
Scheme: weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 10 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 31										
Relation: scklk										
Instances: 64										
Attributes: 4										
yil, maxscklk, minscklk, ortscklk										
Test mode: evaluate on training data										
kMeans										
Number of iterations: 7										
Within cluster sum of squared errors: 1.9529026194779933										
Initial starting points (random):										
Cluster 0: 1998,18.7,11,14.2										
Cluster 1: 1999,19.9,11.6,15										
Cluster 2: 1960,19.9,10.6,14.3										
Cluster 3: 1983,18.3,10.3,13.5										
Cluster 4: 1955,19.2,11.1,14.4										
Cluster 5: 1958,19.8,10.2,14.3										
Cluster 6: 1959,18.3,9.4,13.1										
Cluster 7: 2010,19,12.1,15.1										
Cluster 8: 1980,17.9,9.8,13.2										
Cluster 9: 1967,19.1,9.8,13.7										
	Cluster#									
Attribute	Full Data	0	1	2	3	4	5	6	7	8	9
	(64.0)	(10.0)	(7.0)	(1.0)	(10.0)	(2.0)	(5.0)	(4.0)	(5.0)	(9.0)	(11.0)
Yil	1983.5	1996.9	2002.8571	1966	1989.2	1953.5	1962.2	1958	2012.8	1982.7778	1967
Maxscklk	18.7625	18.91	19.8857	20.7	18.15	19.1	19.68	18	18.98	17.9	18.7
Minscklk	10.5422	10.79	11.6571	11.3	10.21	11.05	10.3	9.575	12.06	9.8222	10.1091
ortscklk	13.9328	14.06	14.9571	15.3	13.47	14.45	14.26	13.125	14.86	13.1444	13.7364
Time taken to build model (full training data) : 0 seconds											
Clustered Instances											
0	10 (16%)				5	5 (8%)					
1	7 (11%)				6	4 (6%)					
2	1 (2%)				7	5 (8%)					
3	10 (16%)				8	9 (14%)					
4	2 (3%)				9	11 (17%)					

Türkiye illeri ortalama sıcaklık verileri analizi

Bu veri kümesinin K-means kümeleme algoritmasına sokulması sonucu elde edilen en verimli sonuç kısaltılarak aşağıdaki Çizelge 3.3'de açıklanmıştır. Sonucun değiştirilmemiş hali EK A'da bulunan, Türkiye illeri ortalama sıcaklık verileri K-Means analizi bölümünde verilmiştir. Veriler arası mesafenin ölçümü için Öklid fonksiyonu kullanılmıştır. K değeri 10 olup, seed değeri 35 olan kümelemede en verimli sonuca ulaşılmıştır. Algoritma bu sonuca ulaşmak için 5 kez tekrar etmiştir.

Çizelge 3.3: Türkiye İlleri Ortalama Sıcaklık Verileri K-Means Çizelgesi

=== Run information ===					
Scheme: weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 10 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 35					
Relation: trortscklk					
Instances: 64					
Attributes: 49					
yil, adana, afyon, ankara, antalya, artvin, balikesir, bandirma, bilecik, bolu, burdur, bursa, canakkalediyarbakir, edirne, elazig, erzincan, erzurum, eskisehir, gaziantep, giresun, gumushane, hatay, isparta, istanbul, izmir, kars, kastamonu, kayseri, kocaeli, konya, kutahya, malatya, manisa, mardin, mersin, mugla, ordu, rize, samsun, siirt, sinop, sivas, tekirdag, trabzon, urfa, usak, van, zonguldak					
Test mode: evaluate on training data					
=== Clustering model (full training set) ===					
kMeans					
======					
Number of iterations: 5					
Within cluster sum of squared errors: 20.305759581092552					
Time taken to build model (full training data) : 0.02 seconds					
=== Model and evaluation on training set ===					
Clustered Instances					
0	12 (19%)	4	5 (8%)	8	3 (5%)
1	10 (16%)	5	5 (8%)	9	9 (14%)
2	3 (5%)	6	4 (6%)		
3	6 (9%)	7	7 (11%)		

3.1.2. Hiyerarşik kümeleme algoritması analizleri

Bu bölümde veri kümelerimiz, hiyerarşik kümeleme algoritmaları olan tek, tam, ward ve ortalama hiyerarşik kümeleme algoritmalarına sokulup elde edilen sonuçlar aşağıda gösterilmiştir. Veriler arasındaki mesafe Öklid fonksiyonuyla hesaplanmıştır. Küme sayısı tüm verilerde 2 olarak seçilmiştir.

Kandilli ortalama rüzgar şiddeti verileri analizi

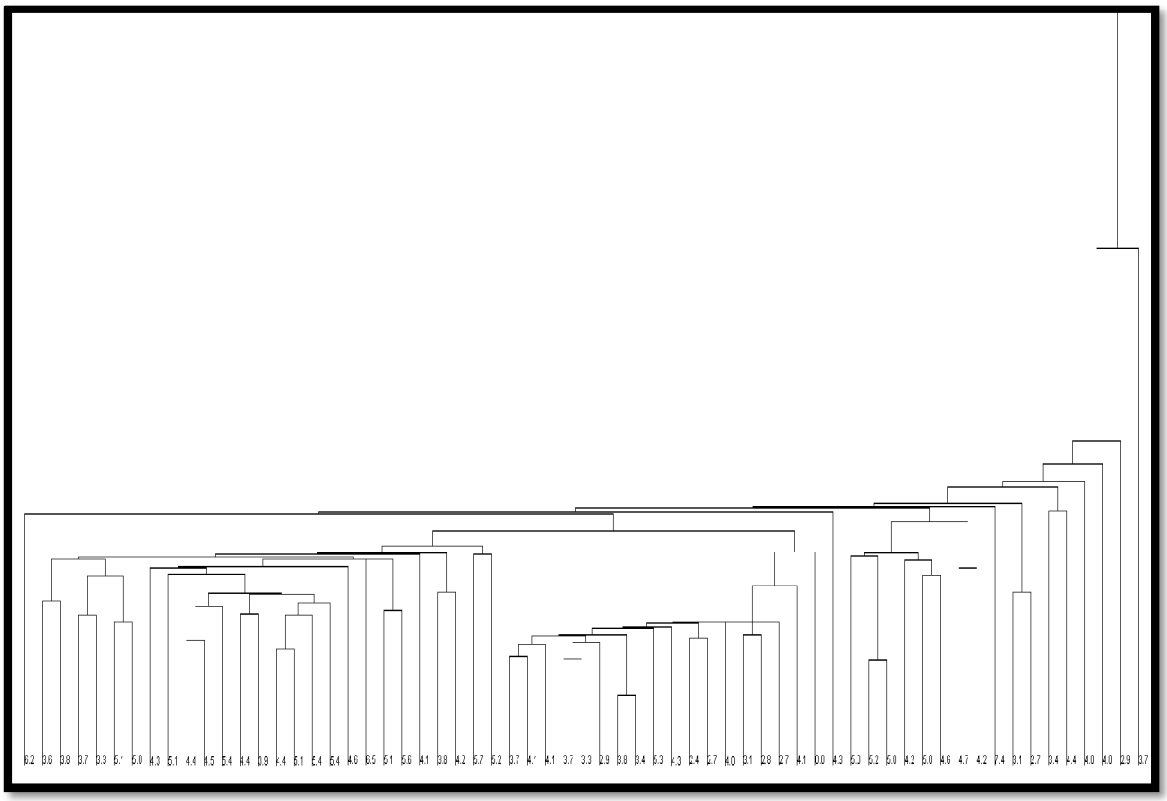
Tek bağlantılı kümeleme analizi

Tek bağlantılı kümeleme algoritmasının WEKA'da çalıştırılması sonucu elde edilen kümeleme çıktısının sonucu Çizelge 3.4'de ve dendrogram sonucu aşağıda verilmiştir.

Çizelge 3.4: Ortalama Rüzgar Şiddeti Verisi Tek Bağlantılı Kümeleme Çizelgesi

=== Run information ===
Scheme: weka.clusterers.HierarchicalClusterer -N 2 -L SINGLE -P -A "weka.core.EuclideanDistance -R first-last"
Relation: ortruzsid, Instances: 64, Attributes: 13
yil, ocak, subat, mart, nisan, mayis, haziran, temmuz, agustos, eylul, ekim, kasim, aralik
Cluster 0
((((((((((6.2:0.51056,(((((((3.6:0.33246,3.8:0.33246):0.08556,((3.7:0.30385,3.3:0.30385):0.07972,(5.1:0.29102,5.0:0.29102):0.09254):0.03446):0.00414,((((4.3:0.39992,(5.1:0.38878,(((4.4:0.25384,4.5:0.25384):0.06917,5.4:0.32301):0.02587,((4.4:0.30606,3.9:0.30606):0.04133,(((4.4:0.23585,5.1:0.23585):0.06781,5.4:0.30366):0.0252,5.4:0.32886):0.01853):0.00149):0.0399):0.01114):0.00181,4.6:0.40172):0.0163,6.5:0.41802):0.0001,(5.1:0.3124,5.6:0.3124):0.10572):0.00405):0.0074,4.1:0.42956):0.00147,(3.8:0.35192,4.2:0.35192):0.07912):0.0144,(5.7:0.42756,5.2:0.42756):0.01787):0.03041,((((((((((3.7:0.21843,4.1:0.21843):0.02677,4.1:0.2452):0.01683,((3.7:0.21435,3.3:0.21435):0.03605,2.9:0.2504):0.01163):0.00094,(3.8:0.14062,3.4:0.14062):0.12235):0.01533,5.3:0.2783):0.00185,4.3:0.28016):0.00713,(2.4:0.25673,2.7:0.25673):0.03056):0.00196,4.0:0.28925):0.00054,(3.1:0.26319,2.8:0.26319):0.02659):0.00087,2.7:0.29065):0.07288,4.1:0.36353):0.07,0.0:0.43352):0.04232):0.03472):0.00357,4.3:0.51414):0.00747,(((5.0:0.42409,(5.2:0.21345,5.0:0.21345):0.21064):0.00724,(4.2:0.41564,(5.0:0.38606,4.6:0.38606):0.02958):0.01569):0.0631,(4.7:0.39928,4.2:0.39928):0.09515):0.02718):0.00269,7.4:0.5243):0.00816,(3.1:0.35052,2.7:0.35052):0.18194):0.03213,(3.4:0.51543,4.4:0.51543):0.04916):0.01252,4.0:0.57711):0.03487,4.0:0.61198):0.04692,2.9:0.6589):0.39135,3.7:1.05025)
Time taken to build model (full training data) : 0 seconds
Clustered Instances
0 63 (98%) | 1 1 (2%)

Kümelemenin dendrogramını aşağıdaki Şekil 3.1'de inceleyebilirsiniz.



Şekil 3.1 : Kandilli Ortalama Rüzgar Şiddeti Tek Bağlantılı Kümeleme Dendrogramı

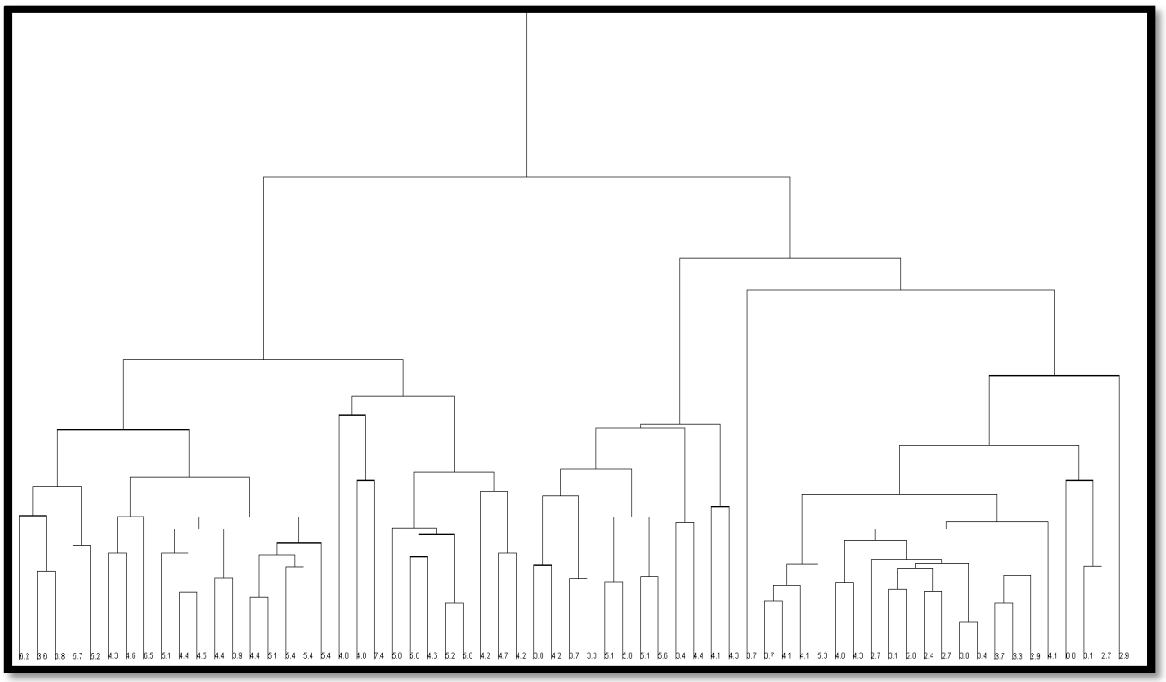
Tam bağlantılı kümeleme analizi

Tam bağlantılı kümeleme algoritmasının çalıştırılması sonucu elde edilen kümeleme çıktısı ve dendrogramı saklanmıştır. Elde edilen bu sonuçlarda program çıktısı aşağıdaki Çizelge 3.5'de verilmiştir. Programın oluşturduğu dendrogram ise Şekil 3.2'de verilmiştir.

Çizelge 3.5: Ortalama Rüzgar Şiddeti Verisi Tam Bağlantılı Kümeleme Çizelgesi

=== Run information ===
Scheme: weka.clusterers.HierarchicalClusterer -N 2 -L COMPLETE -P -A "weka.core.EuclideanDistance -R first-last"
Relation: ortruzsid
Instances: 64
Attributes: 13
yil, ocak, subat, mart, nisan, mayis, haziran, temmuz, agustos, eylul, ekim, kasim, aralik
Test mode: evaluate on training data
=== Clustering model (full training set) ===
Cluster 0
(((((6.2:0.53892,(3.6:0.33246,3.8:0.33246):0.20646):0.10988,(5.7:0.42756,5.2:0.42756):0.22124):0.21346,(((4.3:0.40172,4.6:0.40172):0.13492,6.5:0.53665):0.14962,(((5.1:0.39922,(4.4:0.25384,4.5:0.25384):0.14539):0.09157,(4.4:0.30606,3.9:0.30606):0.18473):0.04396,(((4.4:0.23585,5.1:0.23585):0.15561,(5.4:0.34888,5.4:0.34888):0.04258):0.04563,5.4:0.43709):0.09766):0.15152):0.17598):0.26219,((4.0:0.91366,(4.0:0.67334,7.4:0.67334):0.24032):0.07427,((5.0:0.49555,((5.0:0.38606,4.6:0.38606):0.08467,(5.2:0.21345,5.0:0.21345):0.25727):0.02482):0.20652,(4.2:0.63323,(4.7:0.39928,4.2:0.39928):0.23395):0.06884):0.28586):0.1365):0.68652,((((((3.8:0.35192,4.2:0.35192):0.2615,(3.7:0.30385,3.3:0.30385):0.30957):0.10254,((5.1:0.29102,5.0:0.29102):0.24413,(5.1:0.3124,5.6:0.3124):0.22276):0.18079):0.14948,(3.4:0.51543,4.4:0.51543):0.35001):0.01804,(4.1:0.57227,4.3:0.57227):0.31121):0.62356,(3.7:1.3856,((((((3.7:0.21843,4.1:0.21843):0.06255,4.1:0.28098):0.07534,5.3:0.35632):0.26187,((((4.0:0.28925,4.3:0.28925):0.15488,(2.7:0.3758,(((3.1:0.26319,2.8:0.26319):0.07741,(2.4:0.25673,2.7:0.25673):0.08388):0.01754,(3.8:0.14062,3.4:0.14062):0.21753):0.01765):0.06833):0.04668,((3.7:0.21435,3.3:0.21435):0.10232,2.9:0.31667):0.17415):0.02563,4.1:0.51644):0.10175):0.18768,(0.0:0.67495,(3.1:0.35052,2.7:0.35052):0.32443):0.13092):0.25792,2.9:1.06379):0.32181):0.12144):0.30392)
Time taken to build model (full training data) : 0 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 63 (98%)
1 1 (2%)

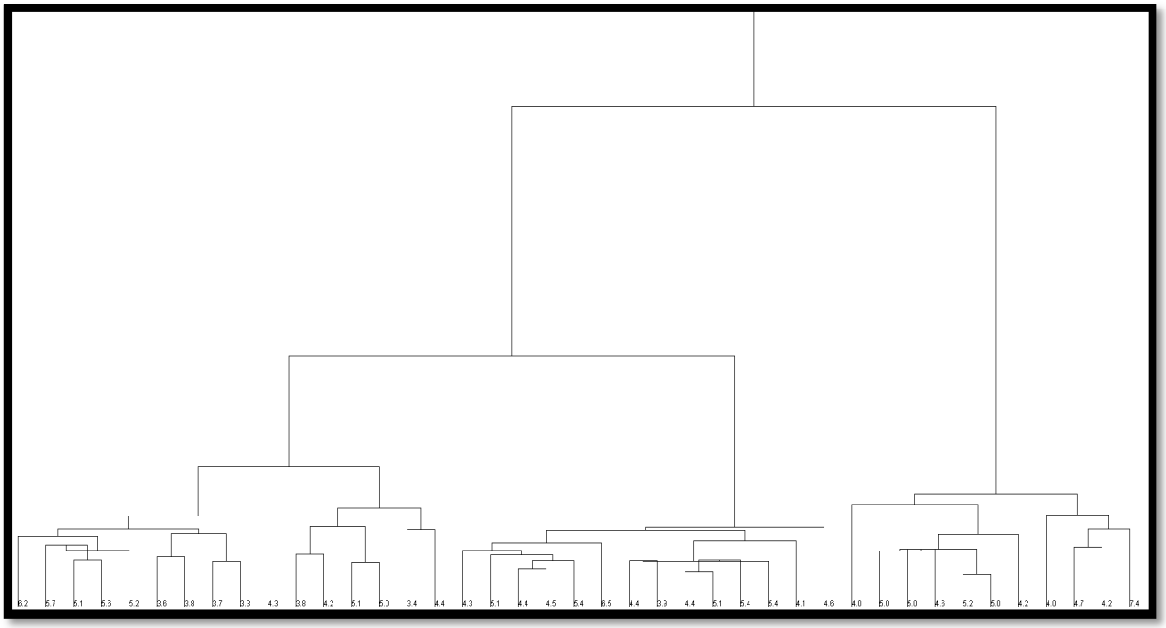
Kümelemenin dendrogramını aşağıdaki Şekil 3.2'de inceleyebilirsiniz.



Şekil 3.2 : Kandilli Ortalama Rüzgar Şiddeti Tam Bağlantılı Kümeleme Dendrogramı

Ward metodu analizi

Algoritmanın çalıştırılması sonucu elde edilen kümeleme çıktısı EK H'de verilmiştir. Kümelemenin dendrogramı ise aşağıdaki Şekil 3.3'de verilmiştir.



Şekil 3.3 : Kandilli Ortalama Rüzgar Şiddeti Ward Metodu Dendrogramı

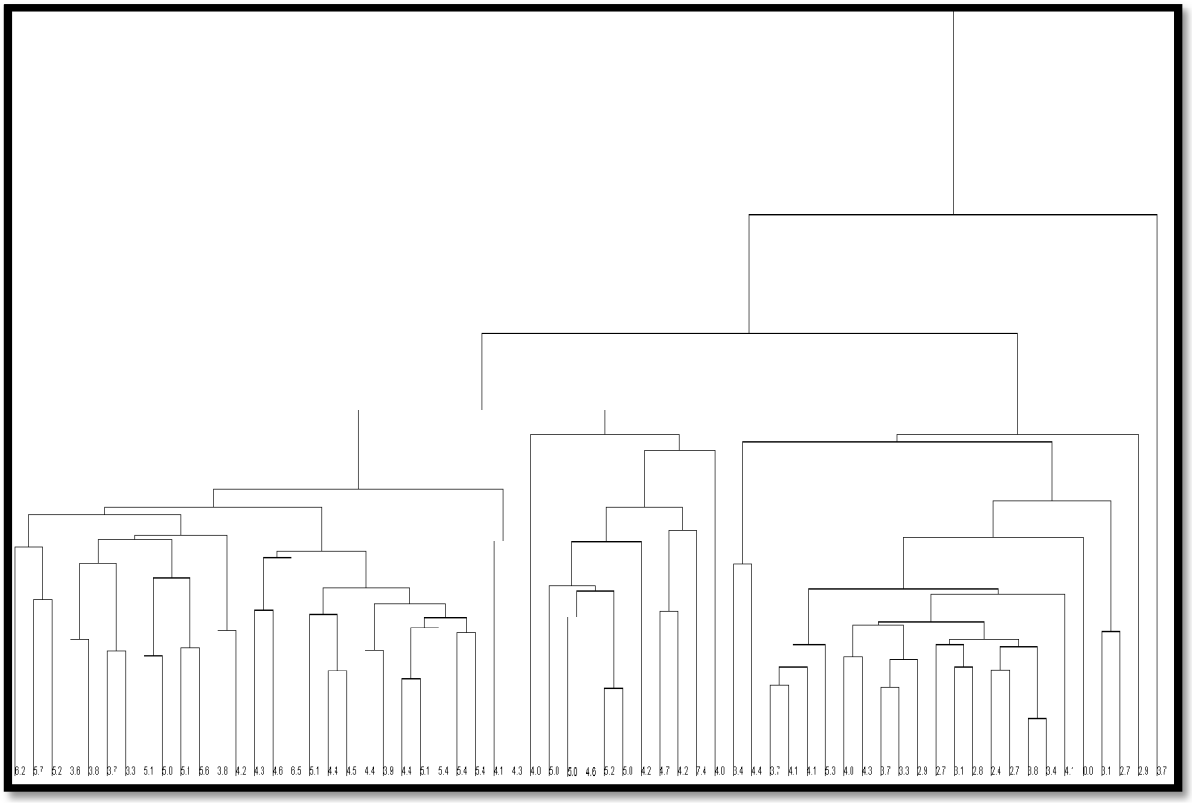
Ortalama bağlantılı kümeleme analizi

Algoritmanın çalıştırılması sonucu elde edilen kümeleme çıktısı aşağıdaki Çizelge 3.6'daki gibidir.

Çizelge 3.6: Ortalama Rüzgar Şiddeti Verisi Ort. Bağlantılı Kümeleme Çizelgesi

=== Run information ===
Scheme: weka.clusterers.HierarchicalClusterer -N 2 -L AVERAGE -P -A "weka.core.EuclideanDistance -R first-last"
Relation: ortruzsid
Instances: 64
Attributes: 13
yil, ocak, subat, mart, nisan, mayis, haziran, temmuz, agustos, eylul, ekim, kasim, aralik
Test mode: evaluate on training data
=== Clustering model (full training set) ===
Cluster 0
(((((((6.2:0.5567,(5.7:0.42756,5.2:0.42756):0.12915):0.07826,((((3.6:0.33246,3.8:0.33246):0.18466,(3.7:0.30385,3.3:0.30385):0.21327):0.05835,((5.1:0.29102,5.0:0.29102):0.18795,(5.1:0.3124,5.6:0.3124):0.16658):0.09649):0.00898,(3.8:0.35192,4.2:0.35192):0.23253):0.05052):0.01893,(((4.3:0.40172,4.6:0.40172):0.13028,6.5:0.532):0.01682,((5.1:0.394,(4.4:0.25384,4.5:0.25384):0.14016):0.06166,((4.4:0.30606,3.9:0.30606):0.11241,(((4.4:0.23585,5.1:0.23585):0.12316,5.4:0.35902):0.02535,(5.4:0.34888,5.4:0.34888):0.03549):0.03411):0.03719):0.09316):0.10507):0.04186,(4.1:0.57227,4.3:0.57227):0.12349):0.19213,(4.0:0.82965,((((5.0:0.46203,((5.0:0.38606,4.6:0.38606):0.06425,(5.2:0.21345,5.0:0.21345):0.23685):0.01173):0.10653,4.2:0.56856):0.08407,((4.7:0.39928,4.2:0.39928):0.197,7.4:0.59628):0.05635):0.13814,4.0:0.79076):0.03889):0.05825):0.18794,(((3.4:0.51543,4.4:0.51543):0.29832,((((((3.7:0.21843,4.1:0.21843):0.04466,4.1:0.26309):0.05749,5.3:0.32058):0.1333,((((4.0:0.28925,4.3:0.28925):0.07729,((3.7:0.21435,3.3:0.21435):0.06918,2.9:0.28353):0.08301):0.00746,((2.7:0.32057,(3.1:0.26319,2.8:0.26319):0.05737):0.01208,((2.4:0.25673,2.7:0.25673):0.05836,(3.8:0.14062,3.4:0.14062):0.17446):0.01757):0.04135):0.06923,4.1:0.44324):0.01065):0.12603,0.0:0.57992):0.08855,(3.1:0.35052,2.7:0.35052):0.31795):0.14527):0.01637,2.9:0.83011):0.24572):0.28632,3.7:1.36215)
Time taken to build model (full training data) : 0 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 63 (98%)
1 1 (2%)

Kümelemenin dendrogramını aşağıdaki Şekil 3.4'de inceleyebilirsiniz.



Şekil 3.4 : Kandilli Ortalama Rüzgar Şiddeti Ortalama Bağlantılı Kümeleme Dendrogramı

Kandilli maksimum, minimum ve ortalama sıcaklık verileri analizi

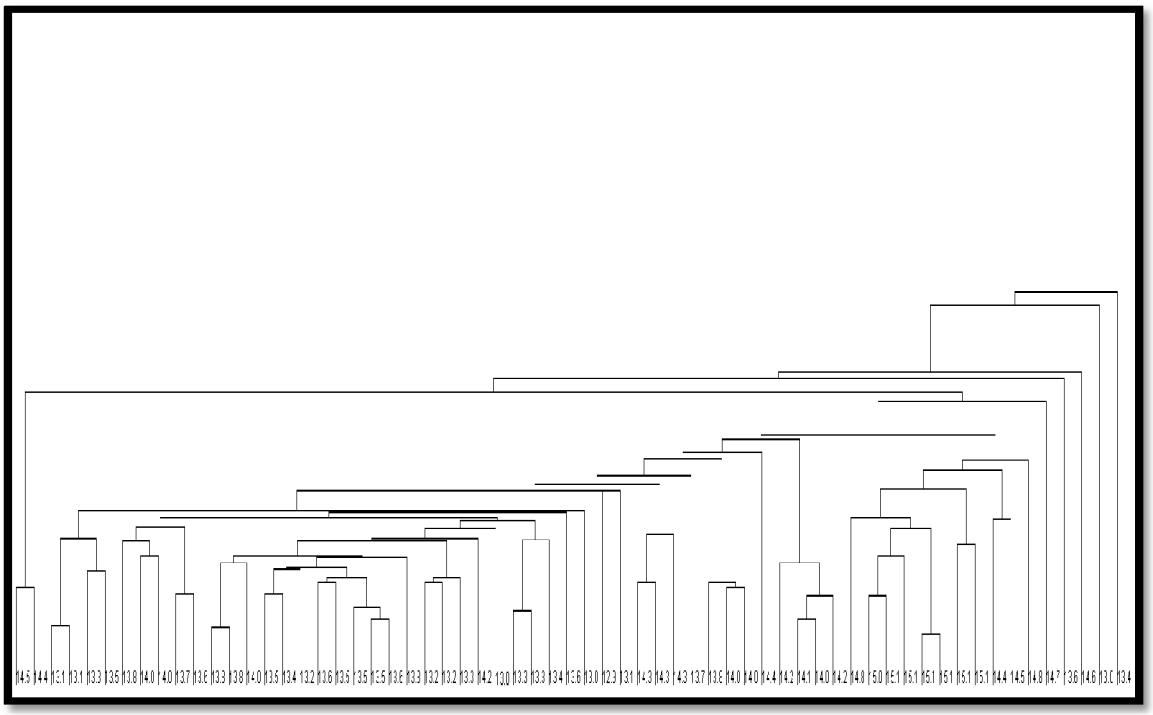
Tek bağlantılı kümeleme analizi

Algoritmanın çalıştırılması sonucu elde edilen kümeleme çıktısı ve dendrogram kaydedilmiştir. Aşağıdaki Çizelge 3.7'de algoritmanın verdiği sonuçlar bulunmaktadır. Ayrıca elde edilen dendrogram Şekil 3.5'de verilmiştir.

Çizelge 3.7: Max. Min. ve Ort. Sıcaklık Verisi Tek Bağlantılı Kümeleme Çizelgesi

=== Run information ===
Scheme: weka.clusterers.HierarchicalClusterer -N 2 -L SINGLE -P -A "weka.core.EuclideanDistance -R first-last"
Relation: scklk
Instances: 64
Attributes: 4
yil, maxscklk, minscklk, ortscklk
Test mode: evaluate on training data
=== Clustering model (full training set) ===
Cluster 0
((((((14.5:0.09056,14.4:0.09056):0.18077,((((((((((13.1:0.05513,13.1:0.05513):0.08074,(13.3:0.10674,13.5:0.10674):0.02912):0.02644,(((((13.8:0.13378,(14.0:0.12027,14.0:0.12027):0.01351):0.01298,(13.7:0.08488,13.6:0.08488):0.06188):0.00861,(((((((13.8:0.05376,13.8:0.05376):0.05963,14.0:0.11339):0.00613,((((13.5:0.08478,13.4:0.08478):0.02311,13.2:0.10789):0.00127,((13.6:0.09602,13.5:0.09602):0.00357,(13.5:0.07317,(13.5:0.06209,13.6:0.06209):0.01108):0.02642):0.00958):0.00975,13.8:0.11892):0.00061):0.01374,((13.2:0.0952,13.2:0.0952):0.00439,13.3:0.09959):0.03368):0.00277,14.2:0.13604):0.00914,13.0:0.14518):0.00671,((13.3:0.06853,13.3:0.06853):0.06605,13.4:0.13458):0.01073):0.00349):0.00501,13.6:0.16038):0.00181,13.0:0.16219):0.00011):0.01865,12.9:0.18095):0.0006,13.1:0.18155):0.00469,((14.3:0.09616,14.3:0.09616):0.04368,14.3:0.13985):0.04639):0.00863,13.7:0.19486):0.01579,(13.8:0.09528,(14.0:0.09045,14.0:0.09045):0.00483):0.11537):0.00532,14.4:0.21597):0.01285,(14.2:0.11265,((14.1:0.06222,14.0:0.06222):0.0211,14.2:0.08332):0.02933):0.11618):0.00318,((((14.8:0.15497,(((15.0:0.08332,15.1:0.08332):0.03621,15.1:0.11953):0.02679,(15.1:0.0462,15.1:0.0462):0.10012):0.00865):0.02732,(15.1:0.13083,15.1:0.13083):0.05146):0.01722,(14.4:0.15365,14.5:0.15365):0.04586):0.00942,14.8:0.20894):0.02306):0.03174,14.7:0.26374):0.0076):0.01452,13.6:0.28585):0.00446,14.6:0.29031):0.06345,13.0:0.35376):0.01141,13.4:0.36517)
Time taken to build model (full training data) : 0 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 63 (98%)
1 1 (2%)

Kümelemenin dendrogramını aşağıdaki Şekil 3.5'de inceleyebilirsiniz.



Şekil 3.5 : Kandilli Maksimum Minimum ve Ortalama Sıcaklık Tek Bağlantı Dendrogramı

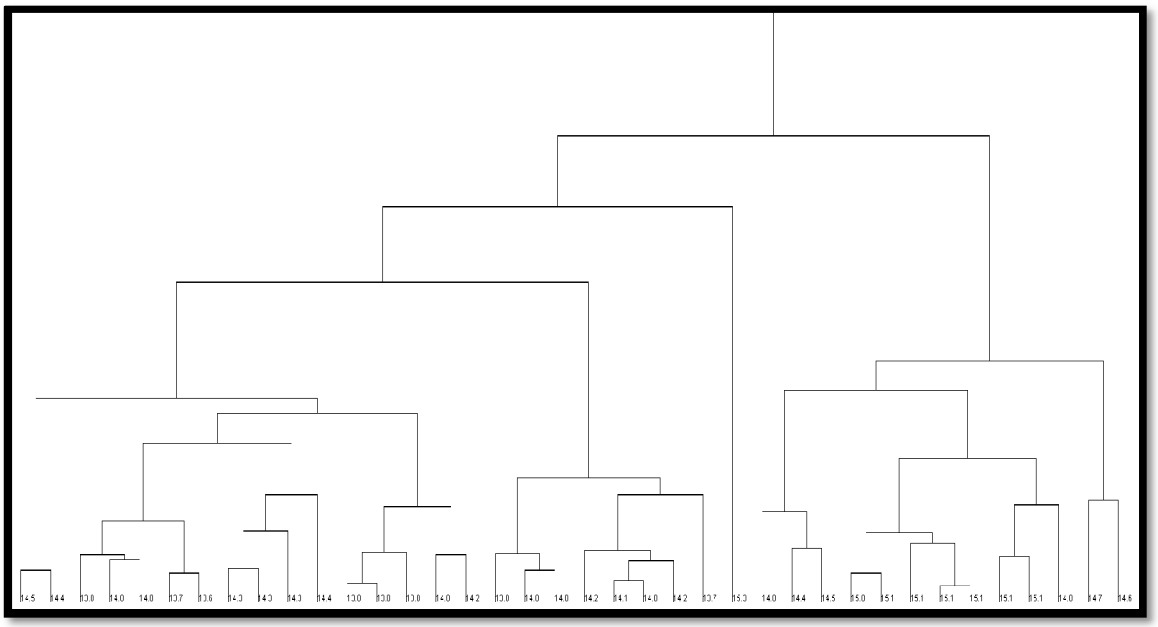
Tam bağlantılı kümeleme analizi

Algoritmanın çalıştırılması sonucu elde edilen kümeleme çıktısı aşağıdaki Çizelge 3.8'de verilmiştir. Ayrıca oluşan dendrogram ise Şekil 3.6'da verilmiştir. Elde edilen bu iki sonuç incelendiğinde, daha önceki farklı veri kümesinin işlenmesi sonucu elde edilen çıktılar karşılaştırıldığında; Dendrogramların ve program çıktılarının birbirinden farklı ancak verilerin işlenme şekli, elde edilen dendrogramların mantıksal benzerliği ve algoritmanın çalışma performansı açısından büyük benzerlikler göstermektedir.

Çizelge 3.8: Max. Min. ve Ort. Sıcaklık Verisi Tam Bağlantılı Kümeleme Çizelgesi

=== Run information ===
Scheme: weka.clusterers.HierarchicalClusterer -N 2 -L COMPLETE -P -A "weka.core.EuclideanDistance -R first-last"
Relation: scklk
Instances: 64
Attributes: 4
yil, maxscklk, minscklk, ortscklk
Test mode: evaluate on training data
=== Clustering model (full training set) ===
Cluster 0
(((((14.5:0.09056,14.4:0.09056):0.48645,((((13.8:0.13387,(14.0:0.12027,14.0:0.12027):0.0136):0.0952,(13.7:0.08488,13.6:0.08488):0.14418):0.22226,(((14.3:0.09616,14.3:0.09616):0.10634,14.3:0.2025):0.10014,14.4:0.30265):0.14868):0.08274,(((13.8:0.05376,13.8:0.05376):0.08879,13.8:0.14256):0.12917,(14.0:0.13604,14.2:0.13604):0.13569):0.26234):0.04294):0.33169,((13.8:0.13821,(14.0:0.09045,14.0:0.09045):0.04776):0.21579,((14.2:0.14959,((14.1:0.06222,14.0:0.06222):0.0567,14.2:0.11892):0.03067):0.15502,13.7:0.30461):0.04939):0.5547):0.2106,15.3:1.1193):0.20463,(((14.8:0.25717,(14.4:0.15365,14.5:0.15365):0.10352):0.34214,(((15.0:0.08332,15.1:0.08332):0.11439,(15.1:0.16742,(15.1:0.0462,15.1:0.0462):0.12122):0.03029):0.21137,((15.1:0.13083,15.1:0.13083):0.14715,14.8:0.27798):0.13109):0.19023):0.08384,(14.7:0.29031,14.6:0.29031):0.39284):0.64077)
Cluster 1
((13.0:0.50768,((13.6:0.3748,(13.1:0.05513,13.1:0.05513):0.31967):0.02362,((13.3:0.10674,13.5:0.10674):0.12549,((13.5:0.08478,13.4:0.08478):0.09867,13.2:0.18345):0.04878):0.16618):0.10927):0.487,((((12.9:0.25833,((13.2:0.0952,13.2:0.0952):0.03266,13.3:0.12786):0.13047):0.09326,((13.0:0.16219,13.0:0.16219):0.03903,13.1:0.20122):0.15037):0.13169,(((((13.6:0.09602,13.5:0.09602):0.0089,13.5:0.10492):0.07436,(13.5:0.06209,13.6:0.06209):0.11719):0.04841,13.6:0.22768):0.08666,((13.3:0.06853,13.3:0.06853):0.07567,13.4:0.1442):0.17014):0.16894):0.2113,13.4:0.69458):0.3001)
Time taken to build model (full training data) : 0 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 38 (59%)
1 26 (41%)

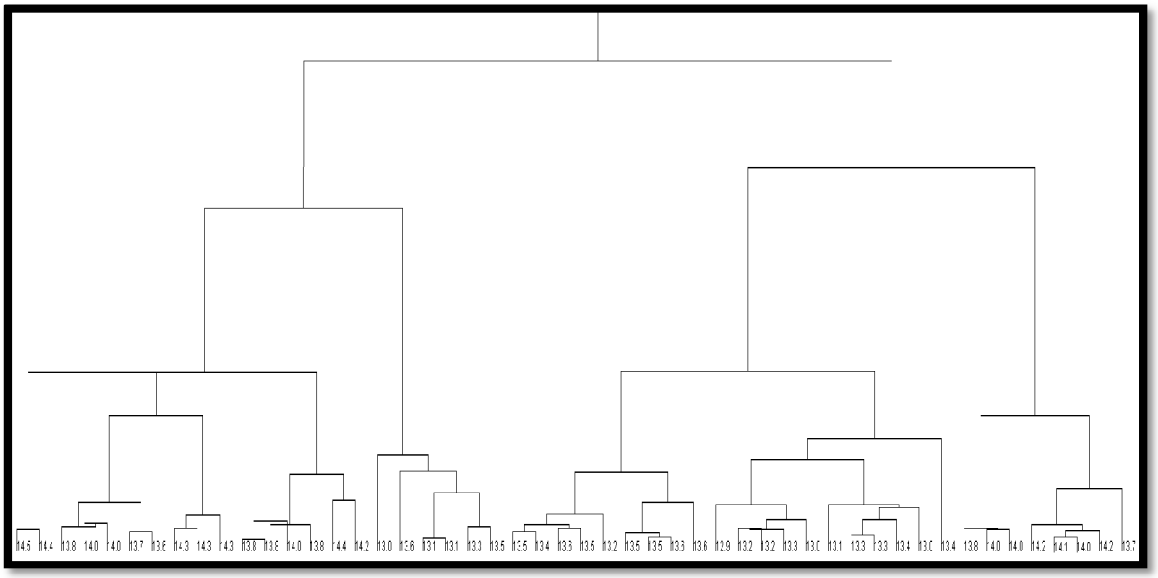
Kümelemenin dendrogramını aşağıdaki Şekil 3.6'da inceleyebilirsiniz.



Şekil 3.6 : Kandilli Maksimum Minimum ve Ortalama Sıcaklık Tam Bağlantı Dendrogramı

Ward metodu analizi

Algoritmanın çalıştırılması sonucu elde edilen kümeleme çıktısı Ek I'da verilmiştir. Kümelemenin dendrogramını aşağıdaki Şekil 3.7'de inceleyebilirsiniz.



Şekil 3.7 : Kandilli Maksimum Minimum ve Ortalama Sıcaklık Ward Metodu Dendrogramı

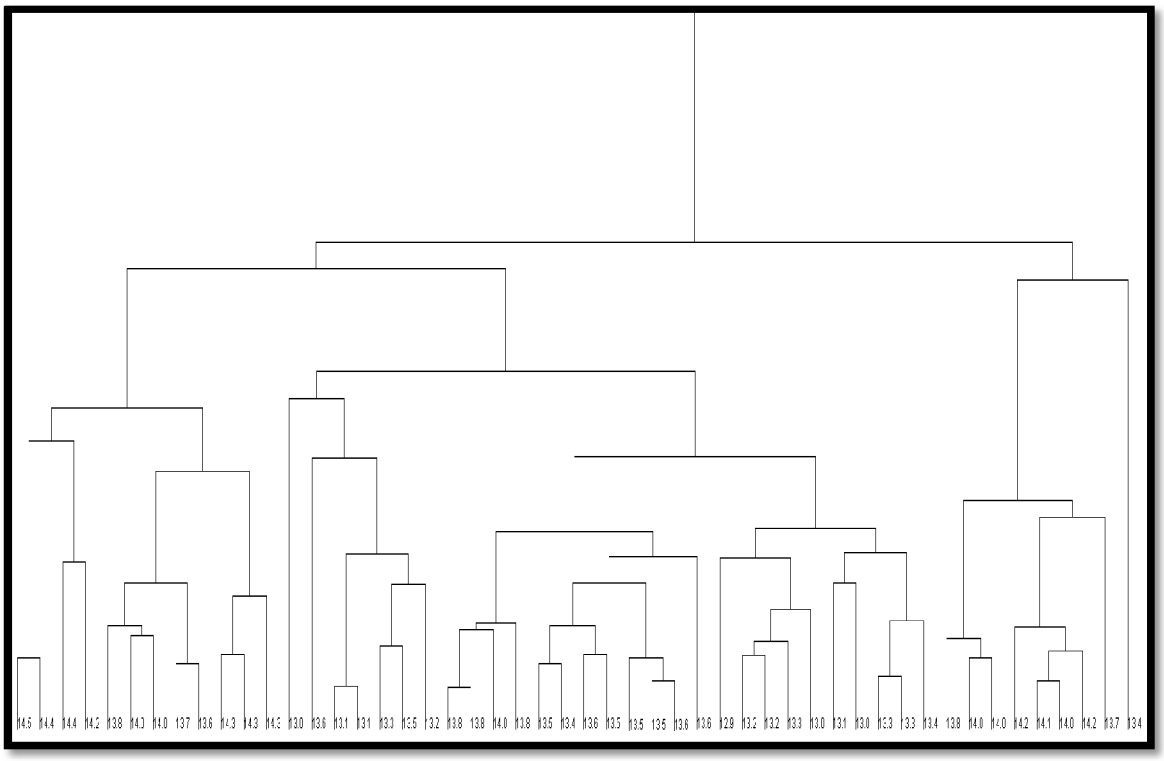
Ortalama bağlantılı kümeleme analizi

Algoritmanın çalıştırılması sonucu elde edilen kümeleme çıktısı aşağıdaki Çizelge 3.9'daki gibidir.

Çizelge 3.9 : Max. Min. ve Ort. Sıcaklık Verisi Ort. Bağlantılı Kümeleme Çizelgesi

=== Run information ===
Scheme: weka.clusterers.HierarchicalClusterer -N 2 -L AVERAGE -P -A "weka.core.EuclideanDistance -R first-last"
Relation: scklk
Instances: 64
Attributes: 4
yil, maxscklk, minscklk, ortscklk
Test mode: evaluate on training data
=== Clustering model (full training set) ===
Cluster 0
(((((14.5:0.09056,14.4:0.09056):0.28004,(14.4:0.21597,14.2:0.21597):0.15464):0.04109,(((13.8:0.13382,(14.0:0.12027,14.0:0.12027):0.01355):0.05383,(13.7:0.08488,13.6:0.08488):0.10278):0.14395,((14.3:0.09616,14.3:0.09616):0.07501,14.3:0.17118):0.16043):0.08009):0.17946,((13.0:0.4251,(13.6:0.34885,((13.1:0.05513,13.1:0.05513):0.16951,((13.3:0.10674,13.5:0.10674):0.0794,13.2:0.18615):0.0385):0.12421):0.07625):0.03343,(((((13.8:0.05376,13.8:0.05376):0.07445,14.0:0.12822):0.00749,13.8:0.13571):0.11778,((((13.5:0.08478,13.4:0.08478):0.04763,(13.6:0.09602,13.5:0.09602):0.03639):0.05608,(13.5:0.09189,(13.5:0.06209,13.6:0.06209):0.0298):0.09661):0.03367,13.6:0.22216):0.03133):0.09708,((12.9:0.22032,(((13.2:0.0952,13.2:0.0952):0.01852,13.3:0.11373):0.04094,13.0:0.15466):0.06566):0.0389,((13.1:0.18849,13.0:0.18849):0.03807,((13.3:0.06853,13.3:0.06853):0.07086,13.4:0.13939):0.08717):0.03266):0.09135):0.10796):0.13263):0.03408,(((13.8:0.11675,(14.0:0.09045,14.0:0.09045):0.02629):0.17818,((14.2:0.13074,((14.1:0.06222,14.0:0.06222):0.0389,14.2:0.10112):0.02962):0.14045,13.7:0.27119):0.02373):0.28235,13.4:0.57727):0.04797)
Cluster 1
(15.3:0.78404,(((14.8:0.22834,(14.4:0.15365,14.5:0.15365):0.07469):0.16237,((((15.0:0.08332,15.1:0.08332):0.06973,15.1:0.15304):0.02346,(15.1:0.0462,15.1:0.0462):0.1303):0.12424,((15.1:0.13083,15.1:0.13083):0.11263,14.8:0.24346):0.05728):0.08997):0.06149,(14.7:0.29031,14.6:0.29031):0.16189):0.33184)
Time taken to build model (full training data) : 0 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 50 (78%)
1 14 (22%)

Kümelemenin dendrogramını aşağıdaki Şekil 3.8'de inceleyebilirsiniz.



Şekil 3.8 : Kandilli Maksimum Minimum ve Ortalama Sıcaklık Ortalama Bağlantı Dendrogramı

Türkiye illeri ortalama sıcaklık verileri analizi

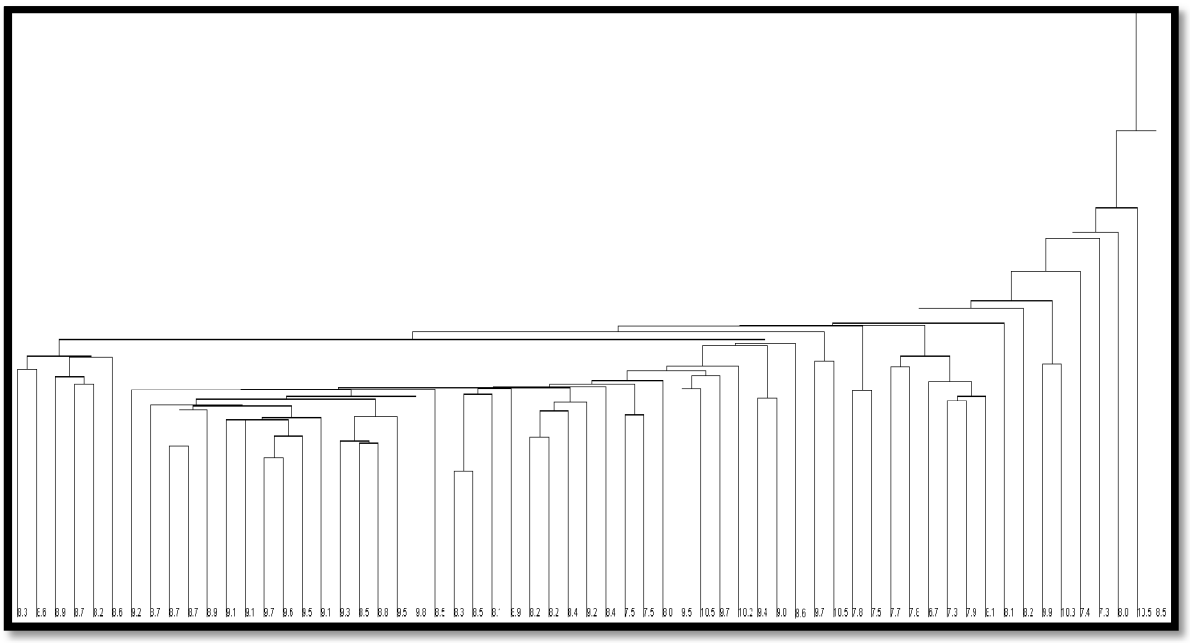
Tek bağlantılı kümeleme analizi

Algoritmanın çalıştırılması sonucu başarılı sonuçlar elde edilmiştir. Bu sonuçlardan biri programın çalışması sonucu elde edilen bilgi çıktısıdır. Bunu Çizelge 3.10'da inceleyebilirsiniz. Ayrıca elde edilen bir diğer sonuç olan dendrogram çıktısı ise Şekil 3.9'da verilmiştir. Bu iki sonucun incelenmesi sonucu daha önce ele aldığımız iki veri kümesi ile elde edilen sonuçlar arasında mantıksal ve performans olarak büyük benzerlikler görülmektedir.

Çizelge 3.10 : Türkiye İlleri Ort. Sıcaklık Verisi Tek Bağlantılı Kümeleme Çizelgesi

=== Run information ===
Scheme: weka.clusterers.HierarchicalClusterer -N 2 -L SINGLE -P -A "weka.core.EuclideanDistance -R first-last"
Relation: trortscklk
Instances: 64
Attributes: 49
yil, adana, afyon, ankara, antalya, artvin, balikesir, bandirma, bilecik, bolu, burdur, bursa, canakkale, diyarbakir, edirne, elazig, erzincan, erzurum, eskisehir, gaziantep, giresun, gumushane, hatay, isparta, istanbul, izmir, kars, kastamonu, kayseri, kocaeli, konya, kutahya, malatya, manisa, mardin, mersin, mugla, ordu, rize, samsun, siirt, sinop, sivas, tekirdag, trabzon, urfa, usak, van, zonguldak
Test mode: evaluate on training data
=== Clustering model (full training set) ===
Cluster 0
((((((((((((8.3:0.60743,8.6:0.60743):0.03458,((8.9:0.59099,(8.7:0.57064,8.2:0.57064):0.02035):0.04809,8.6:0.63908):0.00293):0.03739,((((((((((9.2:0.55678,(((8.7:0.51858,(((8.7:0.41902,8.7:0.41902):0.08945,8.9:0.50847):0.00756,(((9.1:0.48382,9.1:0.48382):0.00053,((9.7:0.39014,9.6:0.39014):0.05184,9.5:0.44199):0.04236):0.00395,9.1:0.4883):0.02773):0.00255):0.01494,((9.3:0.4332,(8.5:0.42669,8.8:0.42669):0.0065):0.05804,9.5:0.49123):0.04229):0.00744,9.8:0.54096):0.01582):0.0018,8.5:0.55858):0.00359,(((8.3:0.35735,8.5:0.35735):0.18859,8.1:0.54595):0.01546,8.9:0.56141):0.00077):0.00158,(((8.2:0.44148,8.2:0.44148):0.06512,8.4:0.5066):0.01897,9.2:0.52557):0.03819):0.00131,8.4:0.56507):0.00633,(7.5:0.49443,7.5:0.49443):0.07697):0.00802,8.0:0.57942):0.02413,((9.5:0.56123,10.5:0.56123):0.0319,9.7:0.59312):0.01043):0.01465,10.2:0.6182):0.04849,(9.4:0.53587,9.0:0.53587):0.13082):0.00525,8.6:0.67195):0.00746):0.02028,(9.7:0.62785,10.5:0.62785):0.07184):0.01264,(7.8:0.55403,7.5:0.55403):0.15829):0.00262,((7.7:0.61438,7.8:0.61438):0.02604,(6.7:0.57899,((7.3:0.52906,7.9:0.52906):0.01283,8.1:0.5419):0.03709):0.06142):0.07453):0.00679,8.1:0.72174):0.03385,8.2:0.75559):0.02014,(9.9:0.62132,10.3:0.62132):0.15441):0.07261,7.4:0.84834):0.07968,7.3:0.92802):0.01517,8.0:0.94318):0.059,10.5:1.00219):0.19004,8.5:1.19222)
Cluster 1
((10.7:0.58255,10.8:0.58255):1.01094,10.3:1.59349)
Time taken to build model (full training data) : 0 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 61 (95%)
1 3 (5%)

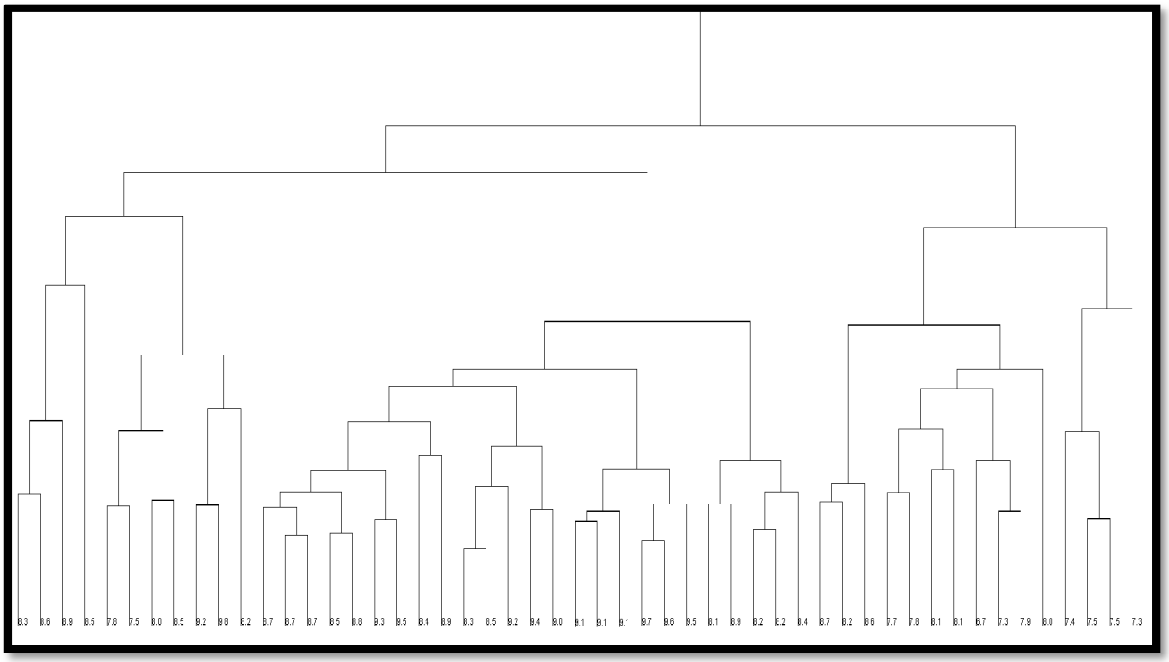
Kümelemenin dendrogramını aşağıdaki Şekil 3.9'da inceleyebilirsiniz.



Şekil 3.9 : Türkiye İlleri Ortalama Sıcaklık Tek Bağlantılı Kümeleme Dendrogramı

Tam bağlantılı kümeleme analizi

Algoritmanın çalıştırılması sonucu elde edilen kümeleme çıktısı EK J'de verilmiştir. Kümelemenin dendrogramını aşağıdaki Şekil 3.10'da inceleyebilirsiniz.



Şekil 3.10 : Türkiye İlleri Ortalama Sıcaklık Tam Bağlantılı Kümeleme Dendrogramı

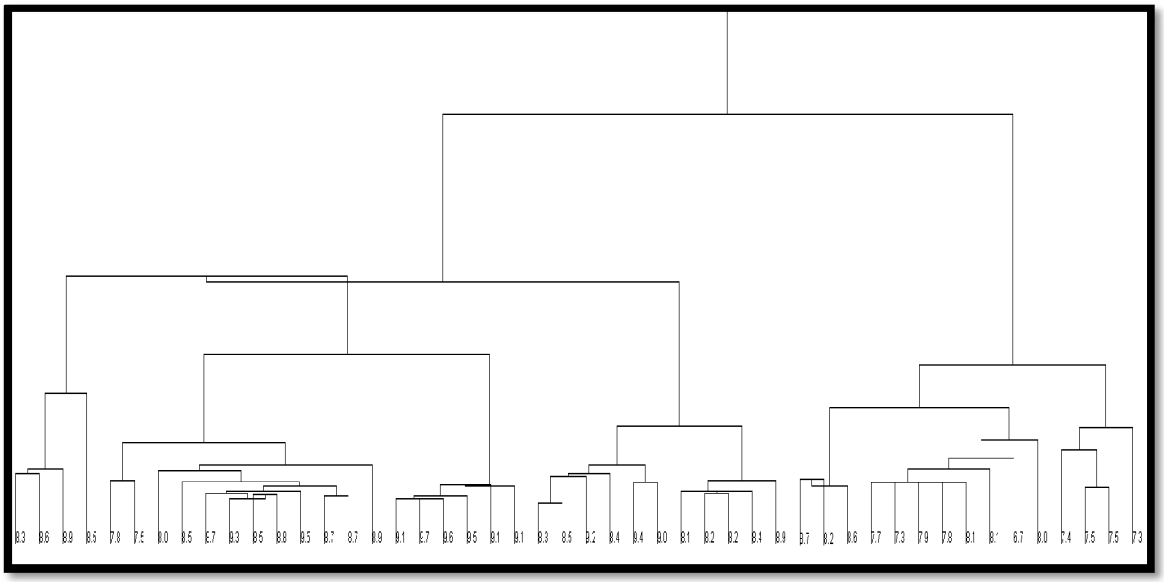
Ward metodu analizi

Algoritmanın çalıştırılması sonucu elde edilen kümeleme çıktısı Çizelge 3.11 gibidir.

Çizelge 3.11 : Türkiye İlleri Ort. Sıcaklık Verisi Ward Metodu Çizelgesi

=== Run information ===
Scheme: weka.clusterers.HierarchicalClusterer -N 2 -L WARD -P -A "weka.core.EuclideanDistance -R first-last"
Relation: trortscklk, Instances: 64, Attributes: 49
yil, adana, afyon, ankara, antalya, artvin, balikesir, bandirma, bilecik, bolu, burdur, bursa, canakkale, diyarbakir, edirne, elazig, erzincan, erzurum, eskisehir, gaziantep, giresun, gumushane, hatay, isparta, istanbul, izmir, kars, kastamonu, kayseri, kocaeli, konya, kutahya, malatya, manisa, mardin, mersin, mugla, ordu, rize, samsun, siirt, sinop, sivas, tekirdag, trabzon, urfa, usak, van, zonguldak
Test mode: evaluate on training data
Cluster 0
((((((8.3:0.60743,8.6:0.60743):0.04366,8.9:0.65109):0.65471,8.5:1.3058):1.00487,(((7.8:0.55403,7.5:0.55403):0.32593,((8.0:0.62875,(8.5:0.54481,(((8.7:0.43747,(9.3:0.39075,(8.5:0.42669,8.8:0.42669):-0.03594):0.04672):0.0248,9.5:0.46228):0.04232,(8.7:0.41902,8.7:0.41902):0.08558):0.04021):0.08394):0.05682,8.9:0.68557):0.19439):0.77171,((((9.1:0.39581,(9.7:0.39014,9.6:0.39014):0.00567):0.01489,9.5:0.4107):0.10329,9.1:0.51399):-0.00829,9.1:0.50569):1.14598):0.659):-0.03959,(((((8.3:0.35735,8.5:0.35735):0.2335,9.2:0.59085):0.023,8.4:0.61385):0.07129,(9.4:0.53587,9.0:0.53587):0.14928):0.33595,((8.1:0.45635,((8.2:0.44148,8.2:0.44148):0.02071,8.4:0.46219):-0.00584):0.10308,8.9:0.55942):0.46167):1.24999):7.5999,((((8.7:0.57064,8.2:0.57064):-0.06505,8.6:0.50559):0.67826,((((7.7:0.53846,(((7.3:0.52906,7.9:0.52906):0.00681,7.8:0.53588):-0.00214,8.1:0.53374):0.00472):0.11657,8.1:0.65503):0.0921,6.7:0.74713):0.15706,8.0:0.90419):0.27965):0.36086,((7.4:0.81024,(7.5:0.49443,7.5:0.49443):0.31581):0.20335,7.3:1.01359):0.53111):8.32628)
Cluster 1
((((((9.2:0.55678,9.8:0.55678):-0.0539,9.5:0.50287):0.19219,(9.7:0.64713,(9.7:0.62785,10.5:0.62785):0.01928):0.04793):0.19451,8.2:0.88957):1.14821,(((8.6:0.64175,(10.2:0.6182,10.5:0.6182):0.02355):0.20728,(9.9:0.62132,10.3:0.62132):0.22771):0.18948,10.5:1.03852):0.99926):4.22536,((10.7:0.58255,10.8:0.58255):1.28175,10.3:1.8643):4.39884)
Time taken to build model (full training data) : 0.03 seconds, Clustered Instances
0 48 (75%) 1 16 (25%)

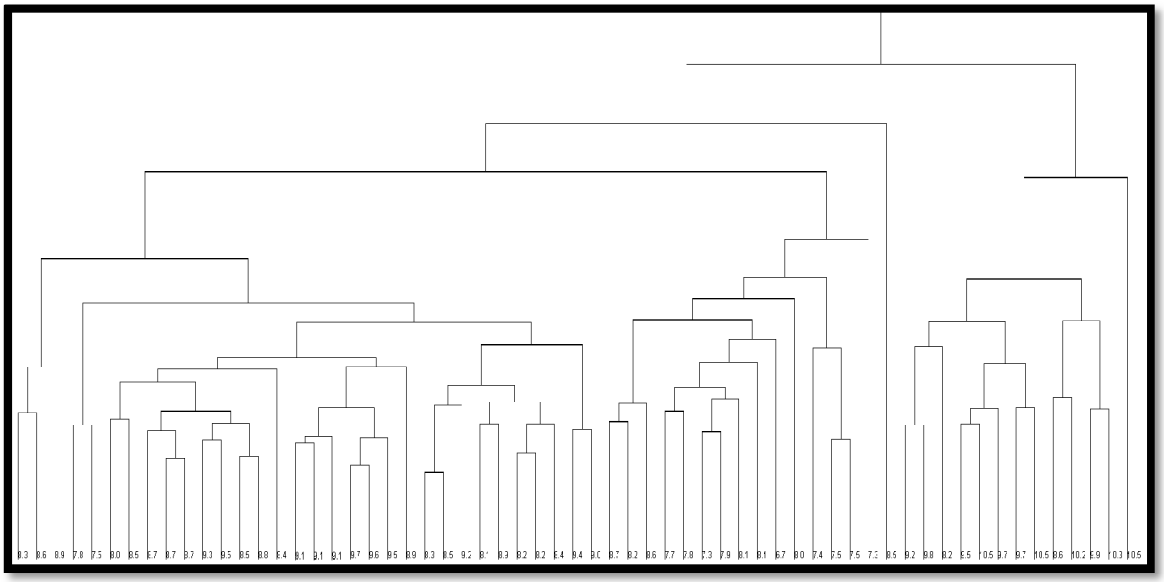
Kümelemenin dendrogramını aşağıdaki Şekil 3.11'de inceleyebilirsiniz.



Şekil 3.11 : Türkiye İlleri Ortalama Sıcaklık Ward Metodu Dendrogramı

Ortalama bağlantılı kümeleme analizi

Algoritmanın çalıştırılması sonucu elde edilen kümeleme çıktısı EK K'da verilmiştir. Kümelemenin dendrogramını aşağıdaki Şekil 3.12'de inceleyebilirsiniz.



Şekil 3.12 : Türkiye İlleri Ortalama Sıcaklık Ortalama Bağlantılı Kümeleme Dendrogramı

3.1.3. Expectation maximization kümeleme algoritması analizleri

Verilerimizi Expectation Maximization algoritmasına sokmadan önce değiştirmemiz gereken bir değer bulunmaktadır. Bu değer, oluşturulacak küme sayısının belirlenmesidir. Bunu seçerken özellikle k-means kümelemesinde başarılı sonuçlar elde ettiğimiz k değeri dikkate alınmıştır.

Kandilli ortalama rüzgar şiddeti verileri analizi

Kandilli verilerinin EM kümeleme algoritmasına sokulması sonucu elde edilen kümeleme çıktısının önemli kısımları aşağıdaki Çizelge 3.12'deki gibidir. Kümeleme çıktısının tam hali EK B'de verilmiştir.

Çizelge 3.12 : Kandilli Ort. Rüzgar Şiddeti Verisi E.M. Çizelgesi

=== Run information ===			
Scheme: weka.clusterers.EM -I 100 -N 10 -X 10 -max -1 -ll-cv 1.0E-6 -ll-iter 1.0E-6 -M 1.0E-6 -K 10 -num-slots 1 -S 100			
Relation: ortruzsid			
Instances: 64			
Attributes: 13			
yil, ocak, subat, mart, nisan, mayis, haziran, temmuz, agustos, eylul, ekim, kasim, aralik			
Test mode: evaluate on training data			
=== Clustering model (full training set) ===			
EM			
==			
Number of clusters: 10			
Number of iterations performed: 0			
Time taken to build model (full training data) : 0.03 seconds			
=== Model and evaluation on training set ===			
Clustered Instances			
0	2 (3%)	5	14 (22%)
1	17 (27%)	6	8 (13%)
2	6 (9%)	7	1 (2%)
3	10 (16%)	8	1 (2%)
4	4 (6%)	9	1 (2%)
Log likelihood: -4.47007			

Kandilli maksimum, minimum ve ortalama sıcaklık verileri analizi

Kandilli verilerinin EM kümeleme algoritmasına sokulması sonucu elde edilen kümeleme çıktısı aşağıdaki Çizelge 3.13'deki gibidir.

Çizelge 3.13 : Kandilli Max. Min. ve Ort. Sıcaklık Verisi E.M. Çizelgesi

=== Run information ===										
Scheme: weka.clusterers.EM -I 100 -N 10 -X 10 -max -1 -ll-cv 1.0E-6 -ll-iter 1.0E-6 -M 1.0E-6 -K 10 -num-slots 1 -S 100										
Relation: scklk										
Instances: 64										
Attributes: 4										
yil, maxscklk, minscklk, ortscklk										
Test mode: evaluate on training data										
Number of clusters: 10										
Number of iterations performed: 0										
Cluster										
Attribute	0	1	2	3	4	5	6	7	8	9
	(0.08)	(0.09)	(0.17)	(0.16)	(0.06)	(0.13)	(0.11)	(0.06)	(0.02)	(0.13)
yil										
Mean	1999.2	1995.1667	1978.9091	1966.6	1958	1995.7	1959.71	1993.25	1966	2011
std. dev.	3.1145	8.4004	4.5925	7.2449	4.6904	9.4378	5.7363	7.8049	0	2.9277
maxscklk										
mean	19.84	17.75	18.0727	18.75	18	18.8875	19.5143	18.5	20.7	19.3
std. dev.	0.2408	0.3507	0.224	0.2953	0.6055	0.29	0.3579	0.7745	0	0.6824
minscklk										
mean	11.4	10.0333	9.9818	10.11	9.575	10.8125	10.5143	10.25	11.3	11.9875
std. dev.	0.2739	0.2944	0.2822	0.268	0.236	0.1808	0.4562	0.1291	0	0.3044
ortscklk										
mean	14.76	13.1833	13.3545	13.76	13.125	14.0625	14.3143	13.55	15.3	14.95
std. dev.	0.305	0.1722	0.2162	0.1647	0.1258	0.1408	0.1574	0.1291	0	0.2138
Time taken to build model (full training data) : 0.02 seconds										
Clustered Instances										
0	5 (8%)			5	8 (13%)					
1	6 (9%)			6	7 (11%)					
2	11 (17%)			7	4 (6%)					
3	10 (16%)			8	1 (2%)					
4	4 (6%)			9	8 (13%)					
Log likelihood: -4.26326										

Türkiye illeri ortalama sıcaklık verileri analizi

Türkiye illeri ortalama sıcaklık verilerinin EM kümeleme algoritmasına sokulması sonucu elde edilen kümeleme çıktısının kısaltılmış hali aşağıdaki Çizelge 3.14'deki gibidir. Kümeleme çıktısının değiştirilmemiş hali ekler bölümünde EK C'de verilmiştir.

Çizelge 3.14 : Türkiye İlleri Ort. Sıcaklık Verisi E.M. Çizelgesi

=== Run information ===
Scheme: weka.clusterers.EM -I 100 -N 10 -X 10 -max -1 -ll-cv 1.0E-6 -ll-iter 1.0E-6 -M 1.0E-6 -K 10 -num-slots 1 -S 100
Relation: trortscklk
Instances: 64
Attributes: 49
yil, adana, afyon, ankara, antalya, artvin, balikesir, bandirma, bilecik, bolu, burdur, bursa, canakkalediyarbakir, edirne, elazig, erzincan, erzurum, eskisehir, gaziantep, giresun, gumushane, hatay, isparta, istanbul, izmir, kars, kastamonu, kayseri, kocaeli, konya, kutahya, malatya, manisa, mardin, mersin, mugla, ordu, rize, samsun, siirt, sinop, sivas, tekirdag, trabzon, urfa, usak, van, zonguldak
Test mode: evaluate on training data
=== Clustering model (full training set) ===
EM
Number of clusters: 10
Number of iterations performed: 3
Time taken to build model (full training data) : 0.07 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 11 (17%)
1 8 (13%)
2 6 (9%)
3 3 (5%)
4 8 (13%)
5 7 (11%)
6 8 (13%)
7 3 (5%)
8 5 (8%)
9 5 (8%)
Log likelihood: -10.08073

3.2. Matlab Analizleri

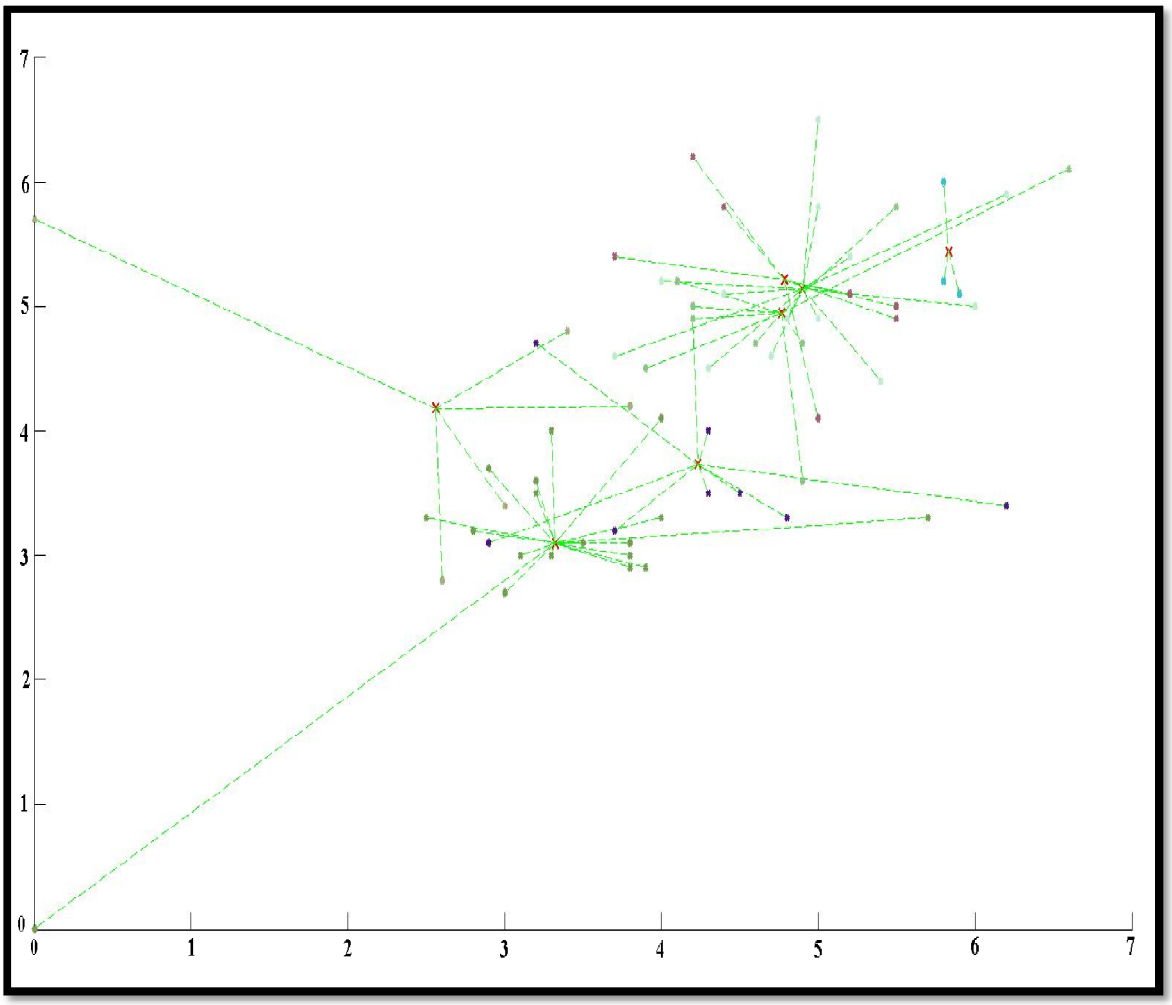
Bu bölümde, ele aldığımız üç kümeleme algoritması olan K-Means, Expectation Maximization ve Hiyerarşik Kümeleme Algoritmalarının MATLAB platformunda kullanılması sonucu elde edilen sonuçlar açıklanmıştır. Veri olarak 2.1.Materyal bölümünde belirtilen Kandilli'den alınmış üç farklı veri kümesi üzerinde çalışılmıştır. MATLAB platformunun en önemli özelliklerinden biri olan görselliği sayesinde elde edilen sonuçlar görsellik açısından çalışmamıza zenginlik katmıştır.

3.2.1. K-Means kümeleme algoritması analizleri

MATLAB platformunda kullandığımız K-Means algoritmasının, EK D'de vermiş olduğumuz K-Means MATLAB kodu ile yapılan analizlerinin sonuçları aşağıdaki bölümlerde verilmiştir. Elde edilen sonuçlardan en verimli olanları seçilip aşağıda verilmiştir.

Kandilli ortalama rüzgar şiddeti verileri analizi

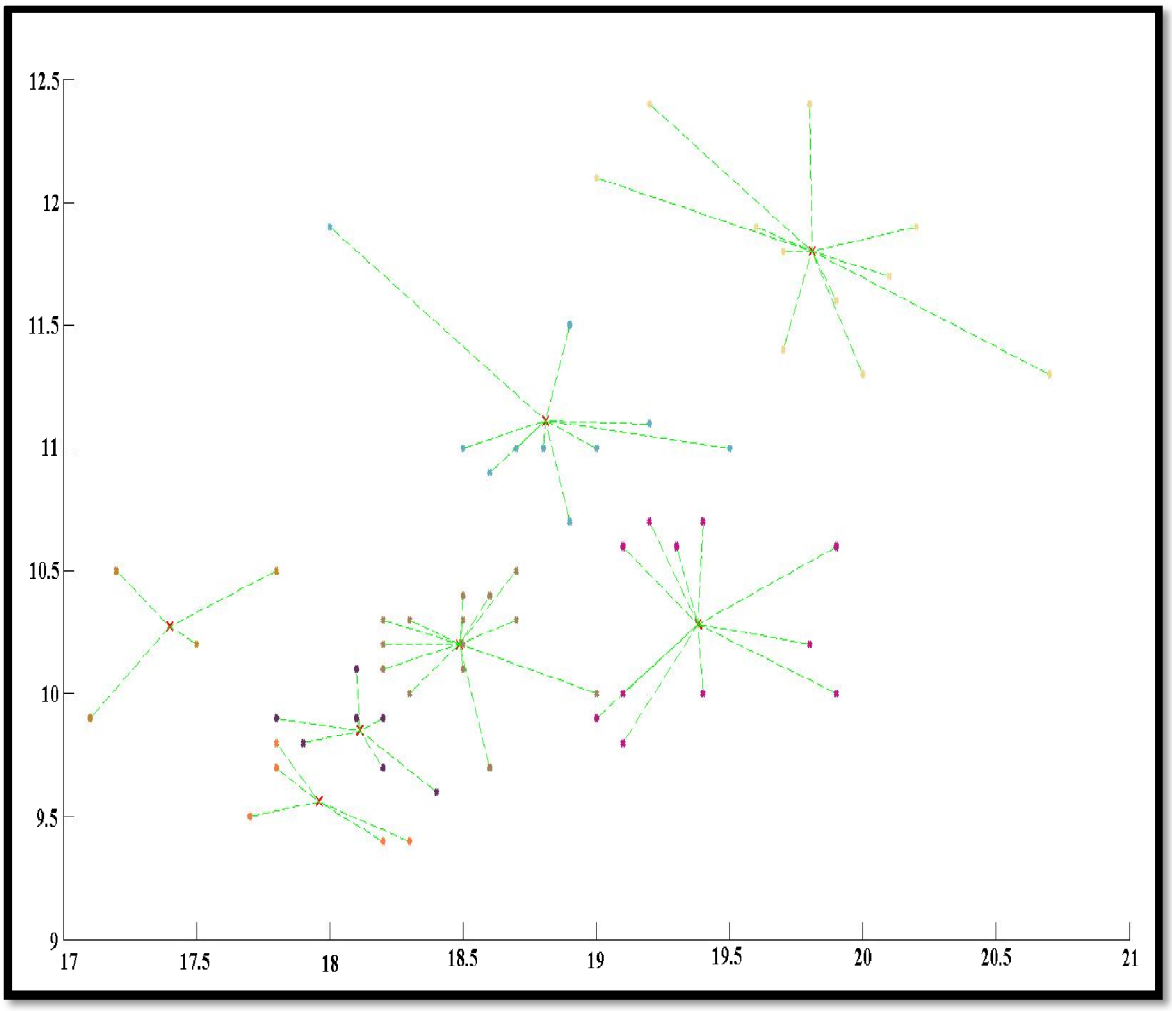
Kandilli ortalama rüzgar şiddeti veri kümesinin MATLAB platformunda K-means kümeleme algoritmasına sokulması sonucu elde edilen en verimli sonuç Şekil 3.13'de verilmiştir. Veriler arası mesafenin ölçümü için Öklid fonksiyonu kullanılmıştır. Kümeleme işlemi sonucu 7 küme oluşmuştur.



Şekil 3.13 : Kandilli Ortalama Rüzgar Şiddeti Verileri K-Means Kümeleme Algoritması MATLAB Sonucu

Kandilli maksimum, minimum ve ortalama sıcaklık verileri analizi

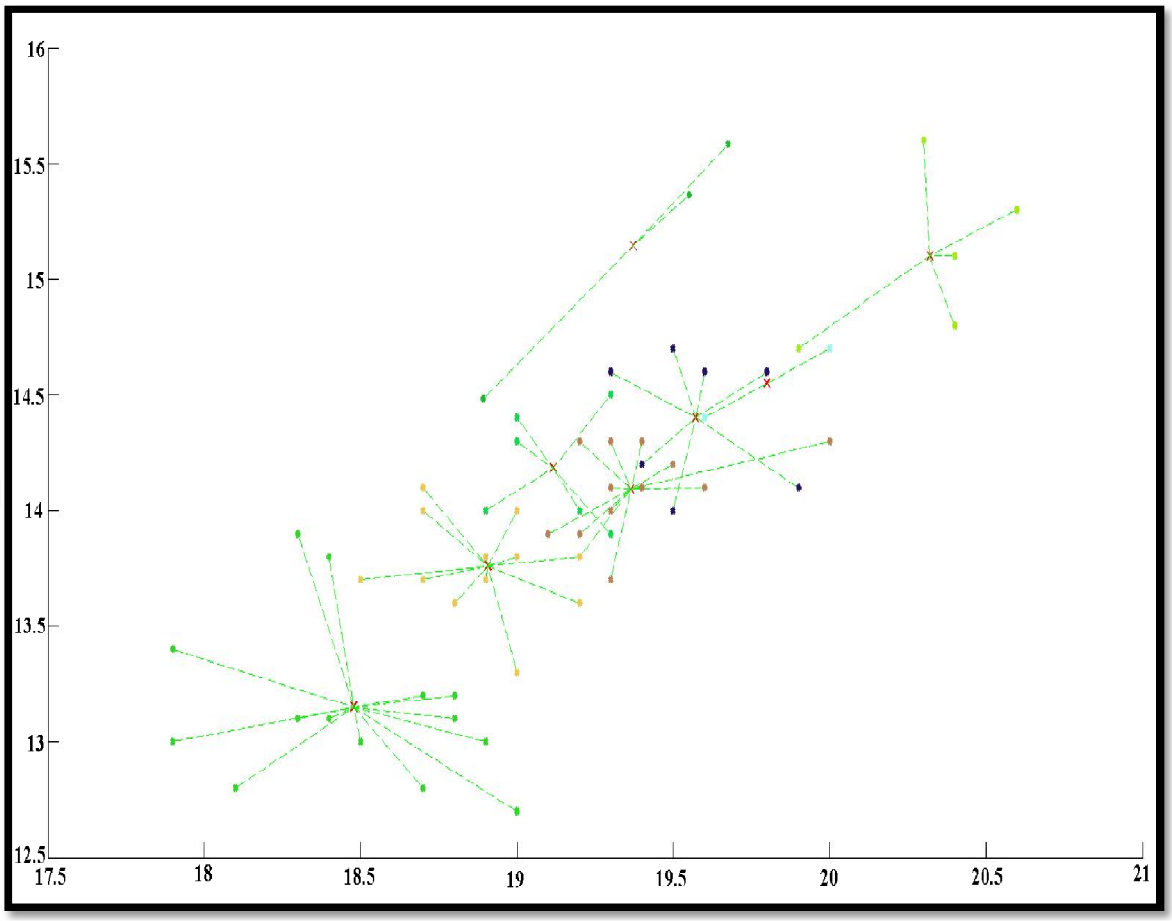
MATLAB platformunda Kandilli Maksimum, minimum ve ortalama sıcaklık verilerinin K-means kümeleme algoritmasına sokulması sonucu elde edilen en verimli sonuç Şekil 3.14'de verilmiştir. Veriler arası mesafenin ölçümü için Öklid fonksiyonu kullanılmıştır. Kümeleme işlemi sonucu 7 küme oluşmuştur. Aşağıdaki şekilde oluşan kümelerin görseli verilmiştir.



Şekil 3.14 : Kandilli Maksimum, Minimum ve Ortalama Sıcaklık Verileri K-Means Kümeleme Algoritması MATLAB Sonucu

Türkiye illeri ortalama sıcaklık verileri analizi

Türkiye illerinin ortalama sıcaklık verileri kümesinin MATLAB platformunda K-means kümeleme algoritmasına sokulması sonucu elde edilen en verimli sonuç Şekil 3.15'de verilmiştir. Veriler arası mesafenin ölçümü için Öklid fonksiyonu kullanılmıştır. Kümeleme işlemi sonucu 8 küme oluşmuştur. Oluşan kümelerin görselini aşağıda inceleyebilirsiniz.



Şekil 3.15 : Türkiye İlleri Ortalama Sıcaklık Verileri K-Means Kümeleme Algoritması MATLAB Sonucu

3.2.2. Hiyerarşik kümeleme algoritması analizleri

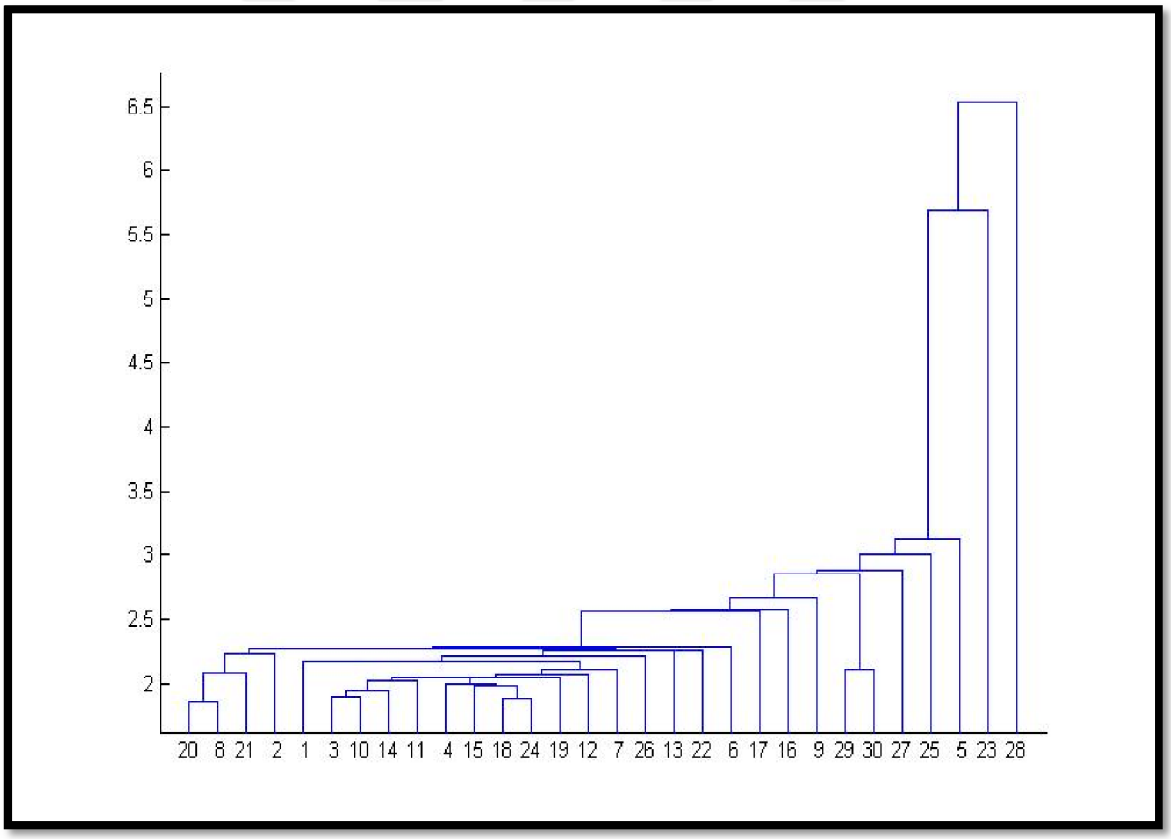
EK E'de verilen, "Hiyerarşik Kümeleme Algoritması MATLAB Kodu" ile 2.1. Materyal bölümünde bahsedilen veri kümelerimizin MATLAB platformunda çalıştırılması sonucu aşağıdaki sonuçlar elde edilmiştir. Her veri kümesi için Tek, Tam, Ward ve Ortalama bağıntılı kümeleme analizi yapılmıştır. Toplamda elde edilen 12 dendrogram aşağıdaki ilgili veri kümelerinin alt başlıklarında verilmiştir. MATLAB platformunda kullandığımız Hiyerarşik Kümeleme kodları ile elde edilen sonuçlar, WEKA'da elde edilen sonuçlara kıyasla daha görsel olduğu için, önceki bölümlerde WEKA programının vermiş olduğu görsellik dışındaki metinsel veriler MATLAB programının vermiş olduğu sonuçlarda bulunmamaktadır.

MATLAB'da kullandığımız Hiyerarşik Kümeleme kodunun çalışması sonucu elde edilen verilerimizin doğruluğu, WEKA'da elde ettiğimiz sonuçlarla karşılaştırıldığında büyük tutarlılık göstermektedir.

Kandilli ortalama rüzgar şiddeti verileri analizi

Tek bağlantılı kümeleme analizi

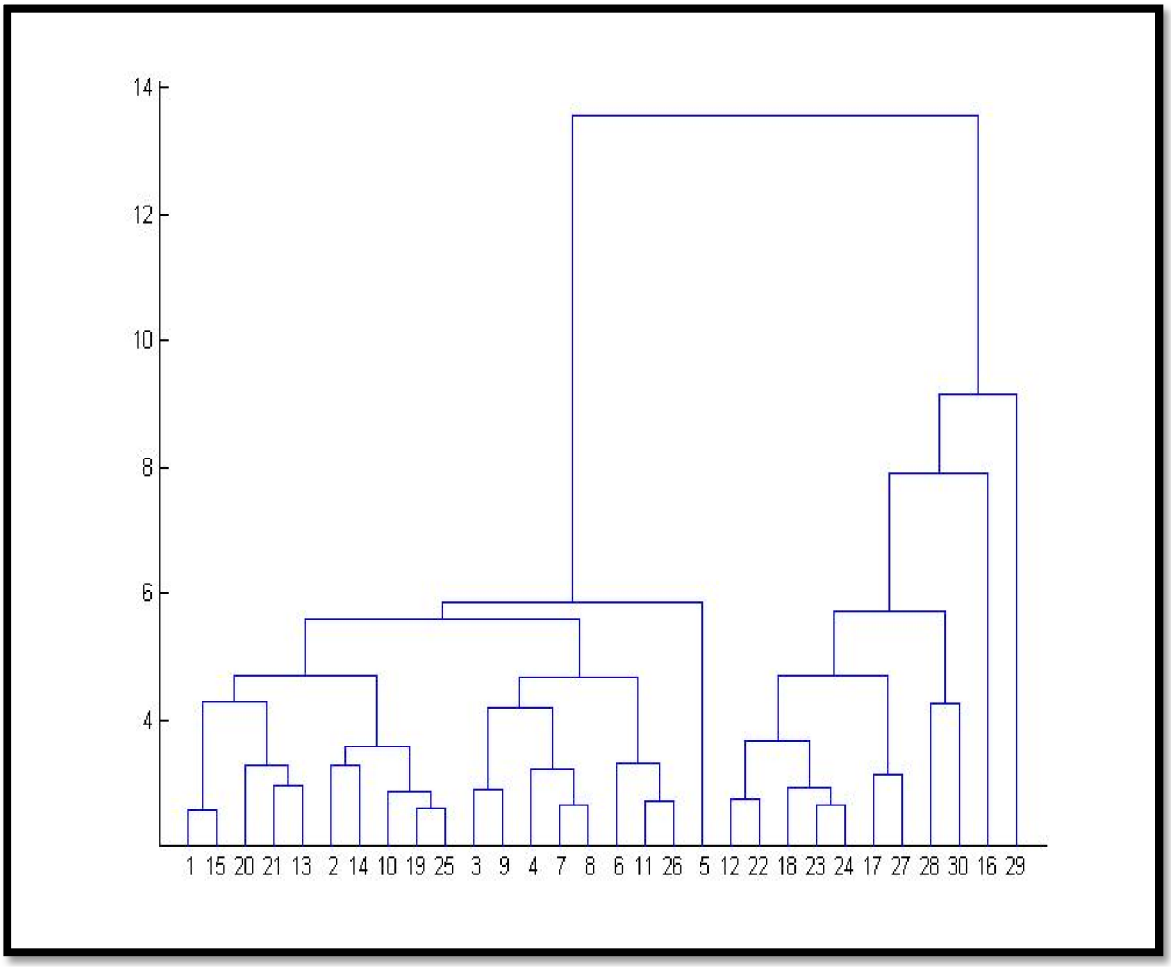
Hiyerarşik Kümeleme için kullandığımız MATLAB kodunun Kandilli ortalama rüzgar şiddeti verilerinde kullanılması sonucu elde edilen dendrogram aşağıdaki gibidir. Oluşan Tek Bağlantılı Kümeleme dendrogramını aşağıda verilen Şekil 3.16'da inceleyebilirsiniz. Oluşan görüntü WEKA analizlerinde elde edilen Şekil 3.1'deki dendrograma büyük benzerlik göstermektedir.



Şekil 3.16 : Kandilli Ortalama Rüzgar Şiddeti Tek Bağlantılı Kümeleme Dendrogramı

Tam bağlantılı kümeleme analizi

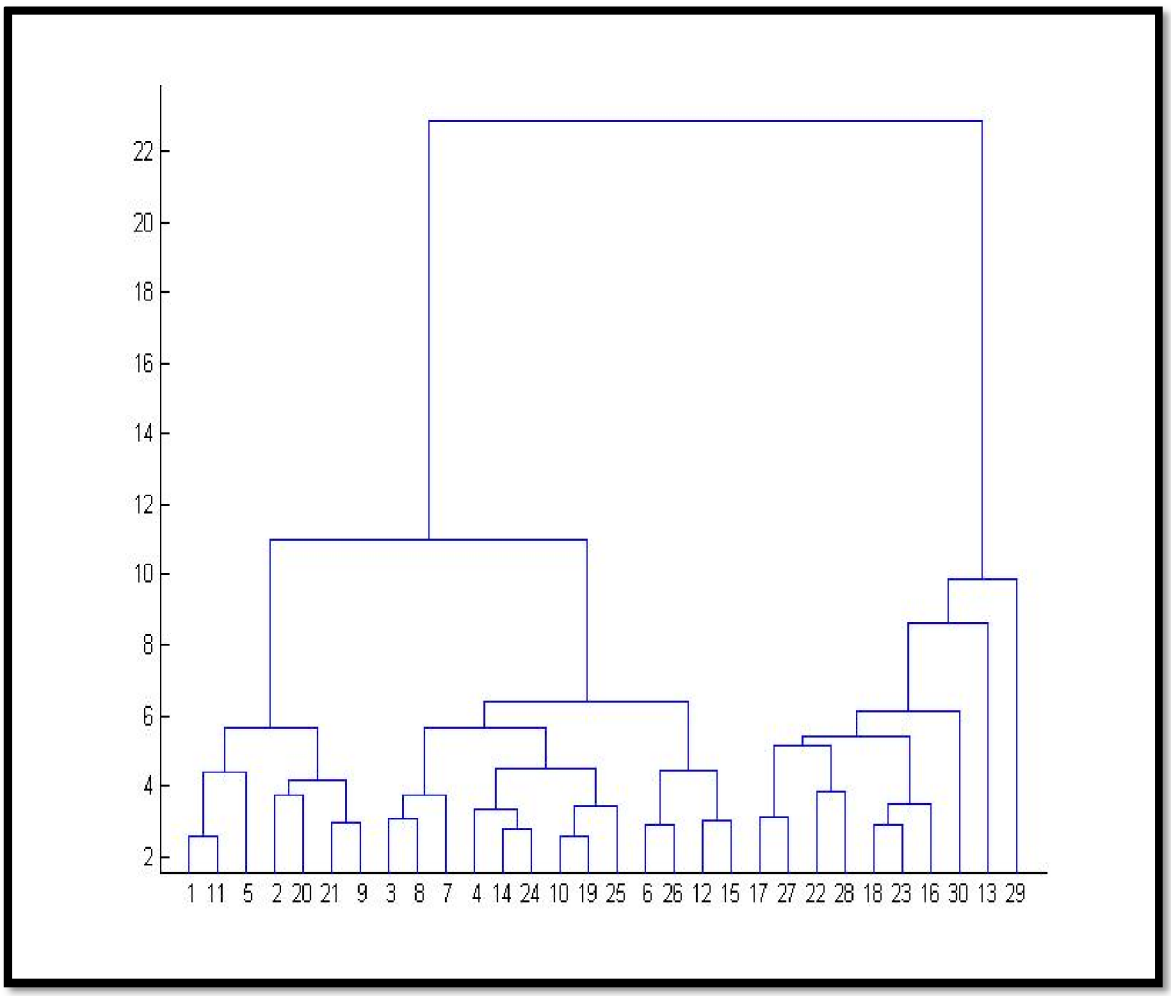
MATLAB platformunda EK E'de verilen Hiyerarşik Kümeleme Algoritması kodu ve Kandilli ortalama rüzgar şiddeti verileri uygulanmıştır. Bu uygulama sonucu elde edilen Tam Bağıntılı Kümeleme dendrogramı elde edilmiştir. Elde edilen görüntü 3.1.2.'de yapılan WEKA platformundaki Hiyerarşik Kümeleme sonuçlarından Şekil 3.2'deki dendrograma büyük benzerlik göstermektedir. Oluşan dendrogramı incelemek için aşağıda verilen Şekil 3.17'deki MATLAB program çıktısını kullanabilirsiniz.



Şekil 3.17 : Kandilli Ortalama Rüzgar Şiddeti Tam Bağlantılı Kümeleme Dendrogramı

Ward metodu analizi

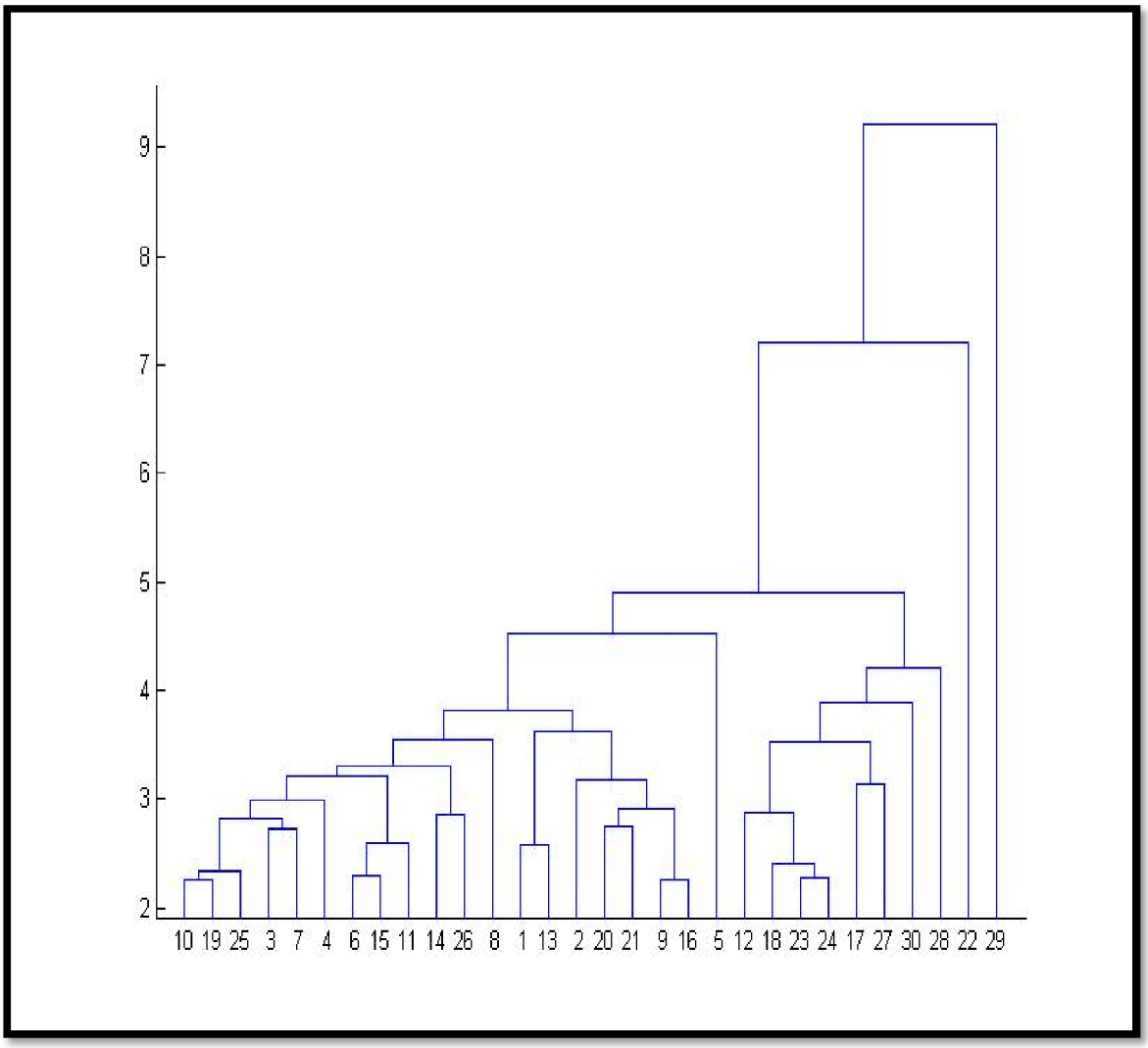
Ward Metodu'nun MATLAB platformunda, Kandilli ortalama rüzgar şiddeti verileri ile uygulanması sonucu elde edilen dendrogram, aşağıda verilen Şekil 3.18'deki gibidir. Elde edilen sonuç, çalışmamızın 3.1.2. bölümündeki WEKA platformuyla yapılan Şekil 3.3'deki Ward Metodu analiziyle karşılaştırıldığında, aralarında büyük benzerlikler görülmektedir. Bu bilgiler ışığında elde edilen dendrogramı aşağıdaki Şekil 3.18'de inceleyebilirsiniz.



Şekil 3.18 : Kandilli Ortalama Rüzgar Şiddeti Ward Metodu Kümeleme Dendrogramı

Ortalama bağlantılı kümeleme analizi

Hiyerarşik Kümeleme için kullandığımız Ortalama Bağıntılı MATLAB kümeleme kodunun, Kandilli ortalama rüzgar şiddeti verilerinde kullanılması sonucu elde edilen dendrogram aşağıdaki gibidir. Oluşan Ortalama Bağlantılı Kümeleme dendrogramını aşağıda verilen Şekil 3.19'da inceleyebilirsiniz. Ayrıca elde ettiğimiz dendrogram, tezimizin 3.1.2. bölümündeki WEKA analizlerinde elde edilen Şekil 3.4'deki dendrograma büyük benzerlik göstermektedir.

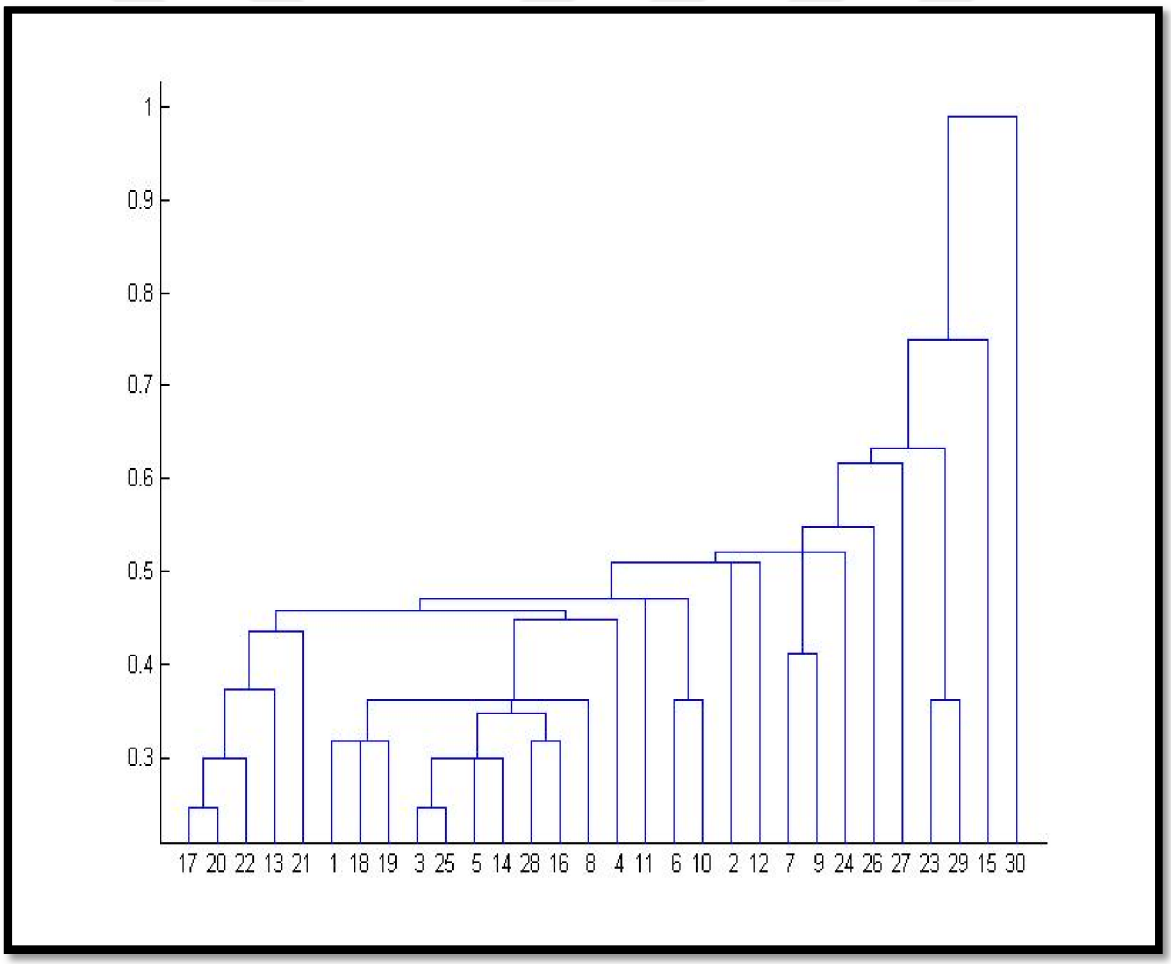


Şekil 3.19 : Kandilli Ortalama Rüzgar Şiddeti Ort. Bağlantılı Kümeleme Dendrogramı

Kandilli maksimum, minimum ve ortalama sıcaklık verileri analizi

Tek bağlantılı kümeleme analizi

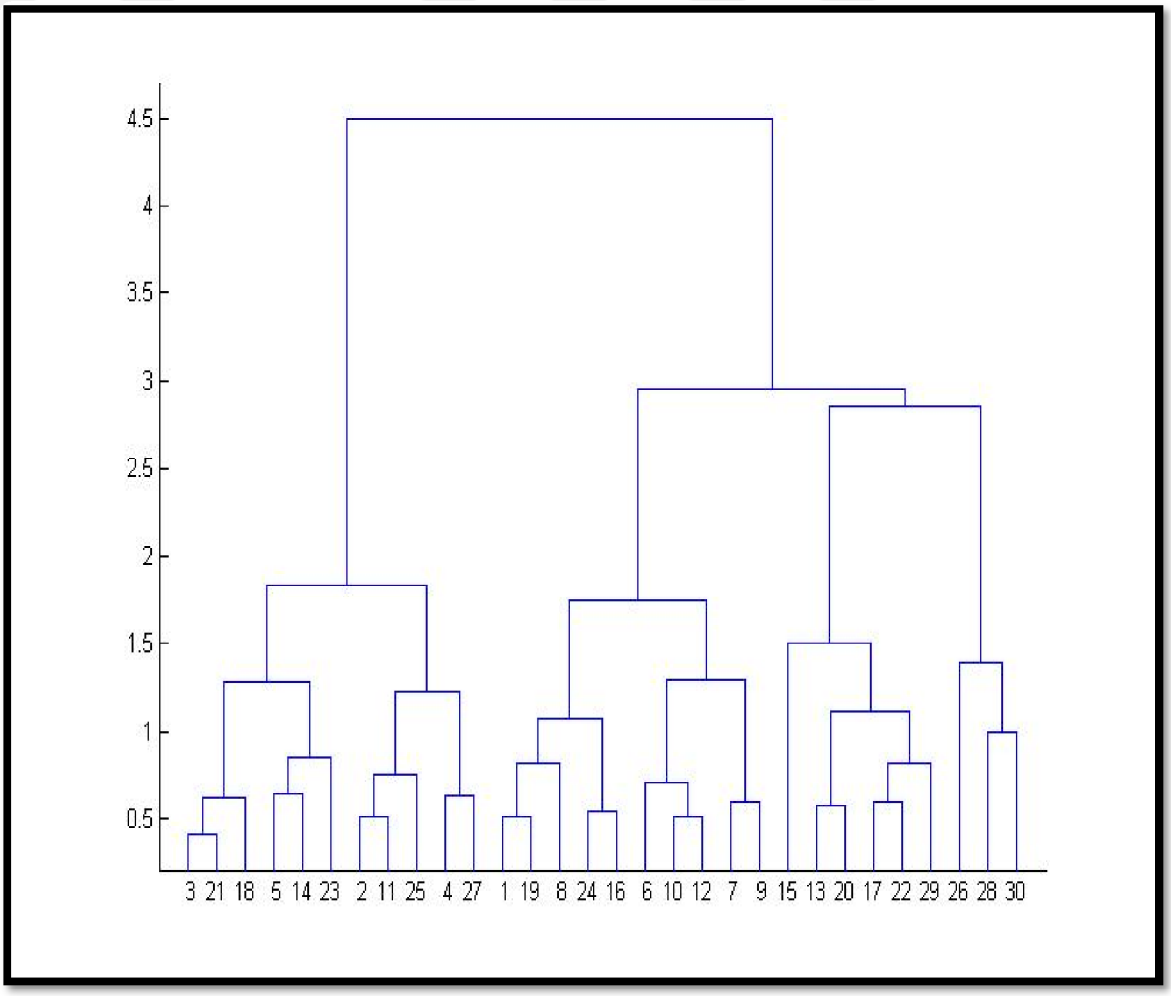
Hiyerarşik Kümeleme için kullandığımız Tek Bağlantılı MATLAB kümeleme kodunun Kandilli maksimum, minimum ve ortalama sıcaklık verilerinde kullanılması sonucu elde edilen dendrogram aşağıdaki gibidir. Oluşan Tek Bağlantılı Kümeleme dendrogramını aşağıda verilen Şekil 3.20'de inceleyebilirsiniz. Ayrıca oluşan görüntü tezimizin 3.1.2. bölümünde WEKA analizlerinde elde edilen Şekil 3.5'deki dendrograma paralellik göstermektedir.



Şekil 3.20 : Kandilli Maksimum, Minimum ve Ortalama Sıcaklık Verileri Tek Bağlantılı Kümeleme Dendrogramı

Tam bağlantılı kümeleme analizi

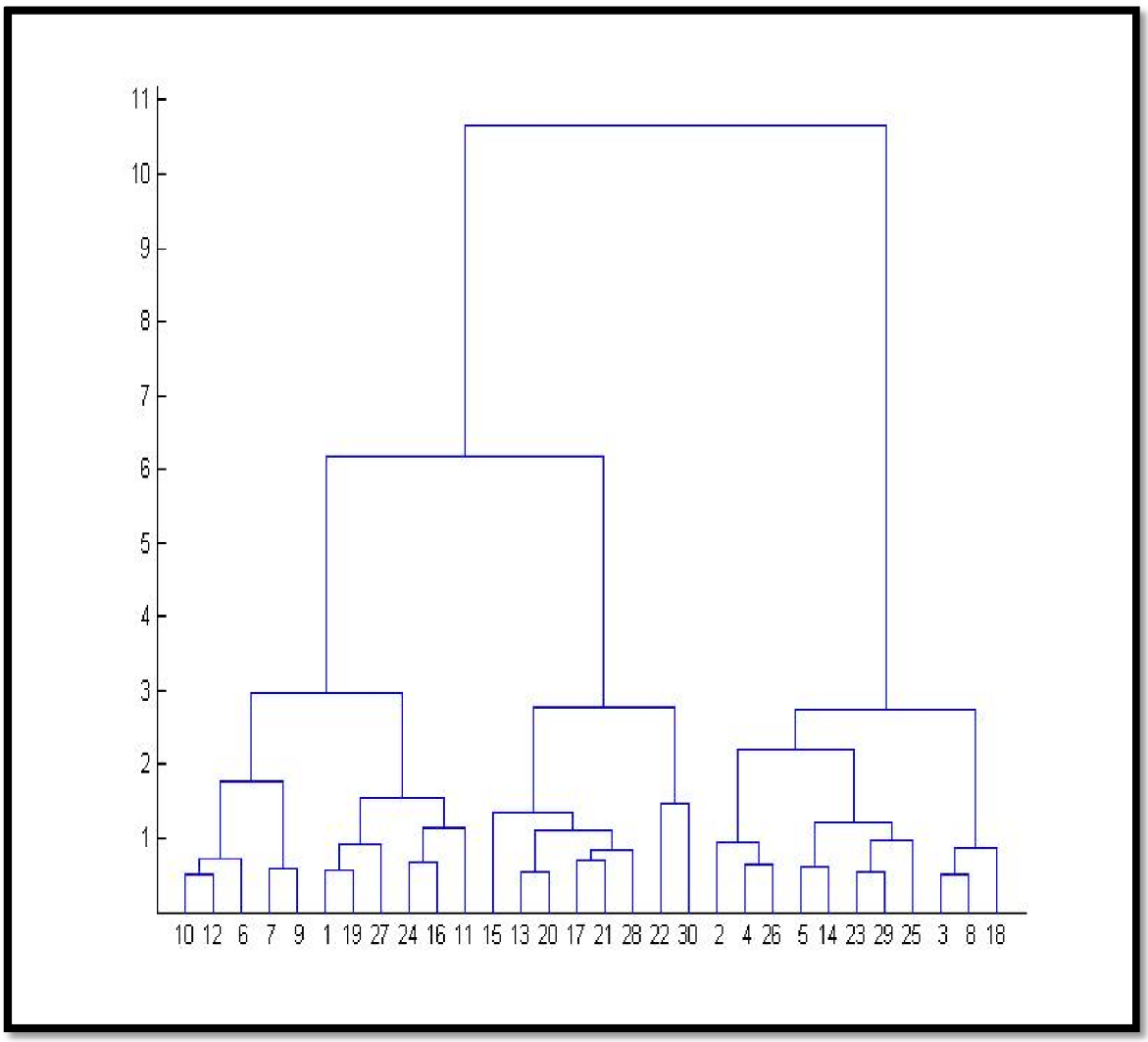
MATLAB platformunda, EK E'de verilen Hiyerarşik Kümeleme Algoritması kodu ve Kandilli maksimum, minimum ve ortalama sıcaklık verileri uygulanmıştır. Bu uygulama sonucu elde edilen Tam Bağıntılı Kümeleme dendrogramı elde edilmiştir. Elde edilen dendrogram, çalışmamızın 3.1.2. bölümünde yapılan WEKA platformundaki Hiyerarşik Kümeleme sonuçlarından Şekil 3.6'daki dendrograma büyük benzerlik göstermektedir. Oluşan dendrogramı incelemek için aşağıda verilen Şekil 3.21'deki MATLAB program çıktısını kullanabilirsiniz.



Şekil 3.21 : Kandilli Maksimum, Minimum ve Ortalama Sıcaklık Verileri Tam Bağlantılı Kümeleme Dendrogramı

Ward metodu analizi

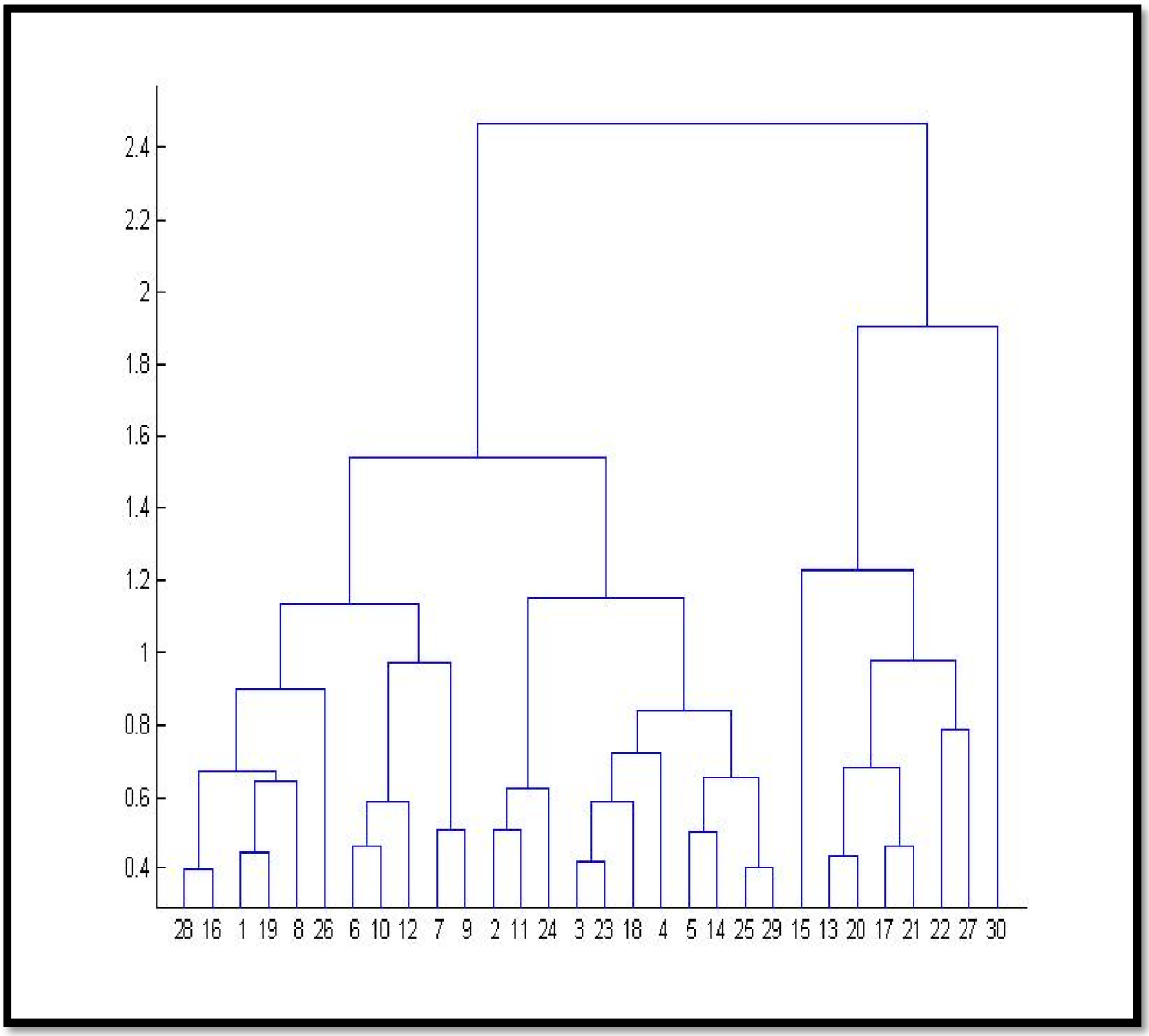
Ward Metodu'nun MATLAB platformunda Kandilli maksimum, minimum ve ortalama sıcaklık verileri ile uygulanması sonucu elde edilen dendrogram, aşağıda verilen Şekil 3.22'deki gibidir. Elde edilen sonuç, tezimizin 3.1.2. bölümündeki WEKA platformuyla yapılan Şekil 3.7'deki Ward Metodu analiziyle karşılaştırıldığında, aralarında büyük benzerlikler görülmektedir. Bu bilgiler ışığında elde edilen dendrogramı aşağıdaki Şekil 3.22'den inceleyebilirsiniz.



Şekil 3.22 : Kandilli Maksimum, Minimum ve Ortalama Sıcaklık Verileri Ward Metodu Kümeleme Dendrogramı

Ortalama bağlantılı kümeleme analizi

Hiyerarşik Kümeleme için kullandığımız Ortalama Bağıntılı MATLAB kümeleme kodunun, Kandilli maksimum, minimum ve ortalama sıcaklık verilerinde kullanılması sonucu elde edilen dendrogram aşağıdaki gibidir. Oluşan Ortalama Bağlantılı Kümeleme dendrogramını aşağıda verilen Şekil 3.23'da inceleyebilirsiniz. Ayrıca elde ettiğimiz dendrogram, tezimizin 3.1.2. bölümündeki WEKA analizlerinde elde edilen Şekil 3.8'deki dendrograma büyük benzerlik göstermektedir.

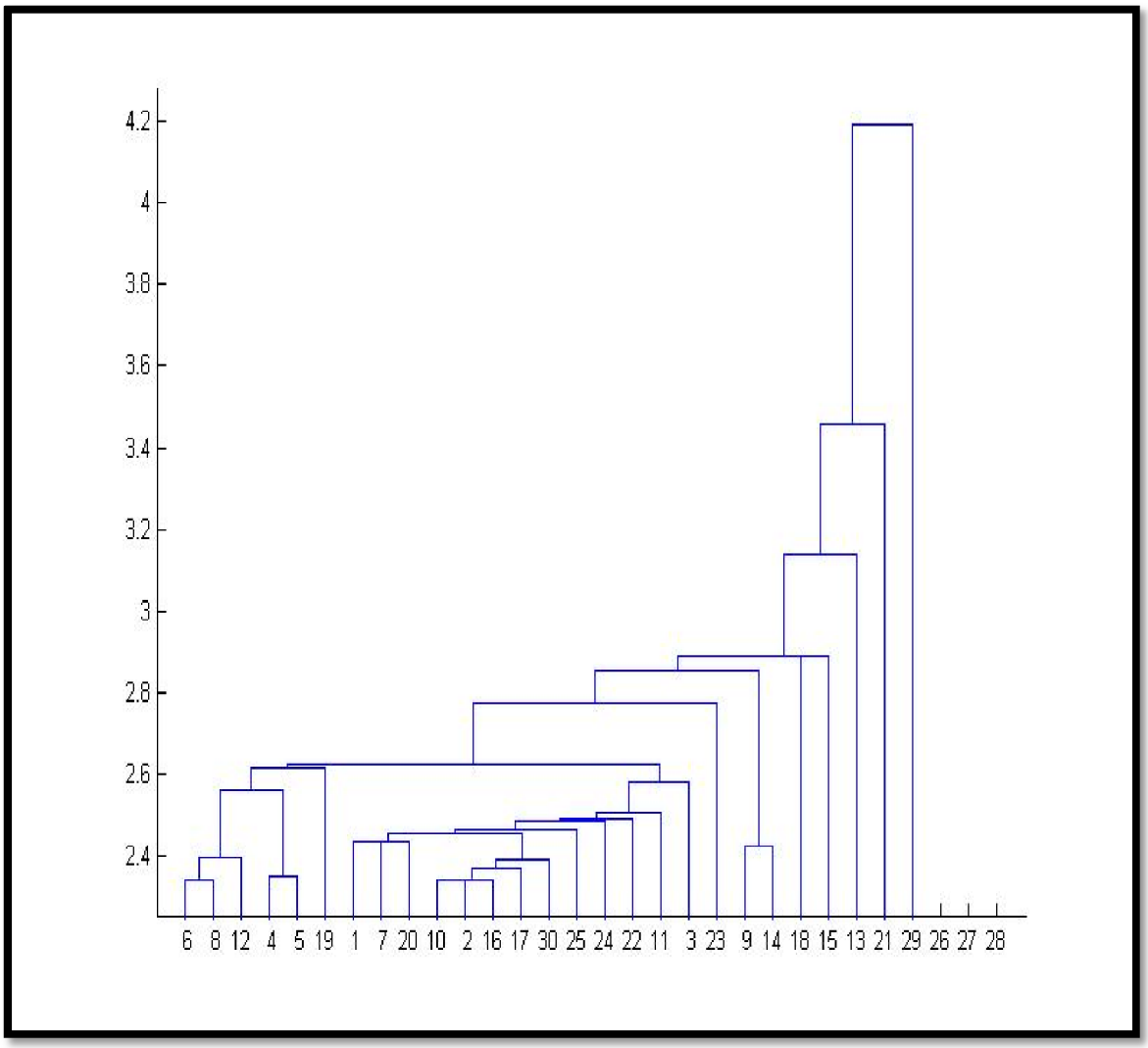


Şekil 3.23 : Kandilli Maksimum, Minimum ve Ortalama Sıcaklık Verileri Ortalama Bağlantılı Kümeleme Dendrogramı

Türkiye illeri ortalama sıcaklık verileri analizi

Tek bağlantılı kümeleme analizi

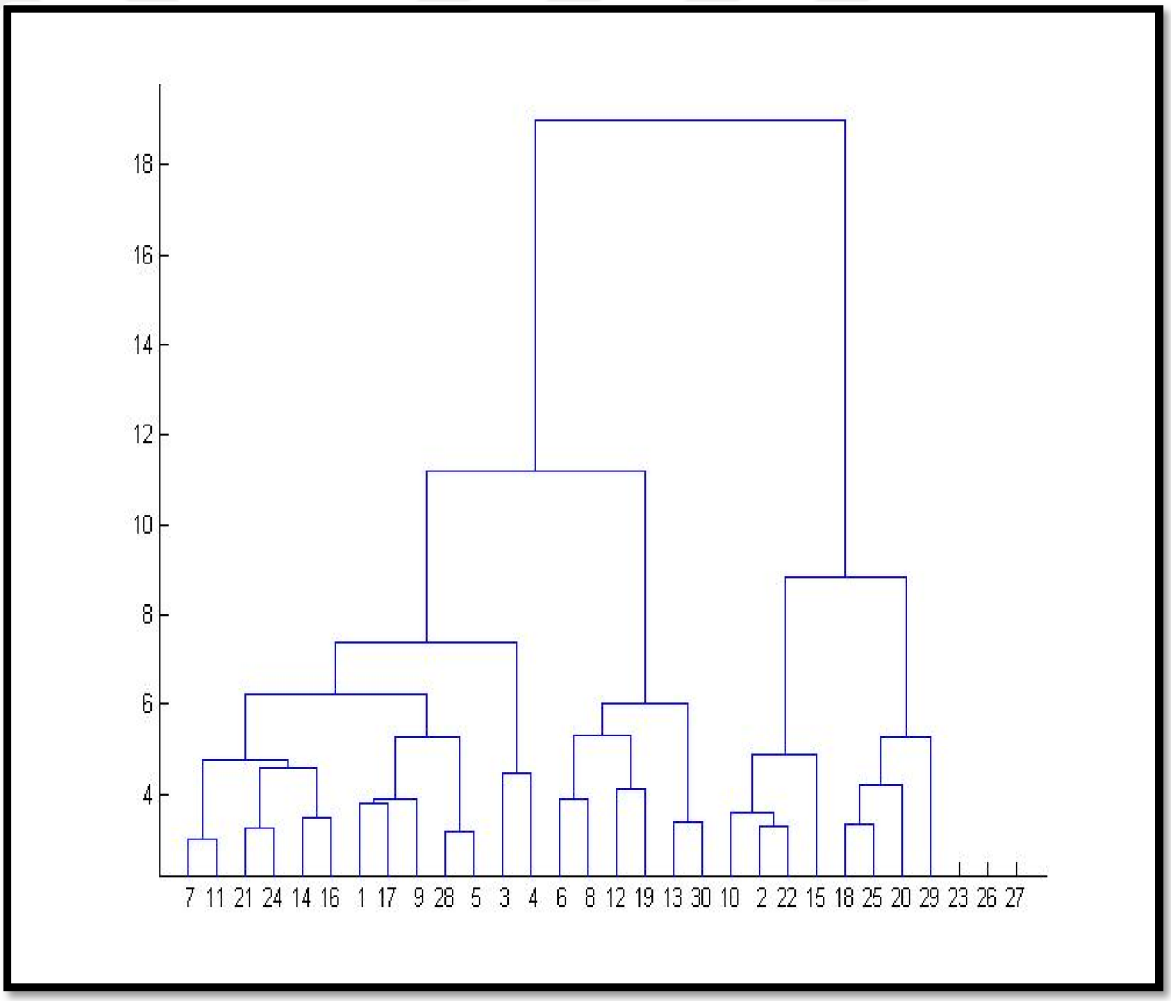
Hiyerarşik Kümeleme için kullandığımız Tek Bağlantılı MATLAB kümeleme kodunun Türkiye illeri ortalama sıcaklık verilerinde kullanılması sonucu elde edilen dendrogram aşağıdaki gibidir. Oluşan Tek Bağlantılı Kümeleme dendrogramını aşağıda verilen Şekil 3.24'de inceleyebilirsiniz. Ayrıca oluşan görüntü tezimizin 3.1.2. bölümünde WEKA analizlerinde elde edilen Şekil 3.9'deki dendrograma paralellik göstermektedir.



Şekil 3.24 : Türkiye İlleri Ortalama Sıcaklık Verileri Tek Bağlantılı Kümeleme Dendrogramı

Tam bağlantılı kümeleme analizi

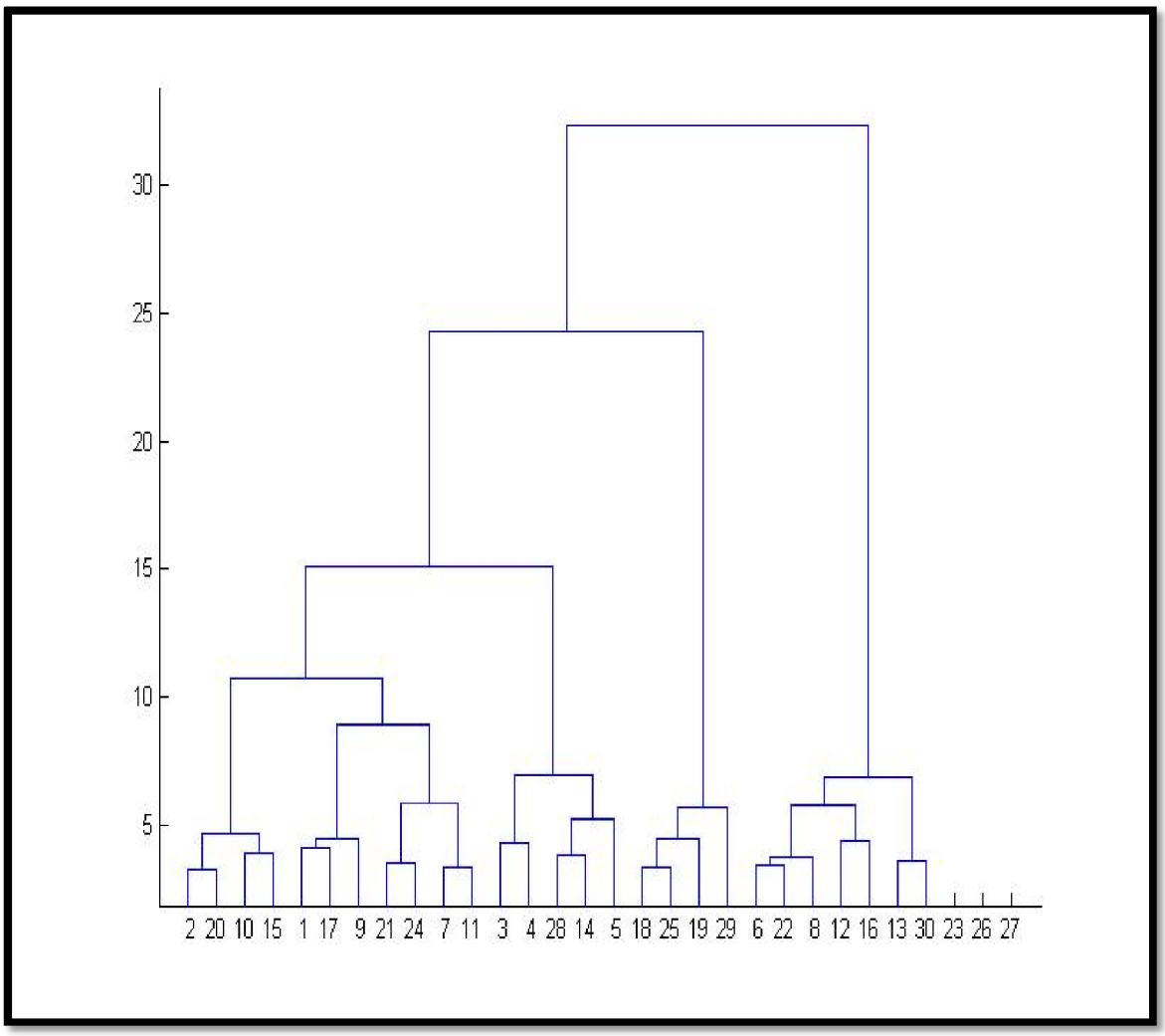
MATLAB platformunda, EK E'de verilen Hiyerarşik Kümeleme Algoritması kodu ve Türkiye illeri ortalama sıcaklık verileri uygulanmıştır. Bu uygulama sonucu elde edilen Tam Bağıntılı Kümeleme dendrogramı elde edilmiştir. Elde edilen dendrogram, çalışmamızın 3.1.2. bölümünde yapılan WEKA platformundaki Hiyerarşik Kümeleme sonuçlarından Şekil 3.10'daki dendrograma büyük benzerlik göstermektedir. Oluşan dendrogramı incelemek için aşağıda verilen Şekil 3.25'deki MATLAB program çıktısını kullanabilirsiniz.



Şekil 3.25 : Türkiye İlleri Ortalama Sıcaklık Verileri Tam Bağlantılı Kümeleme Dendrogramı

Ward metodu analizi

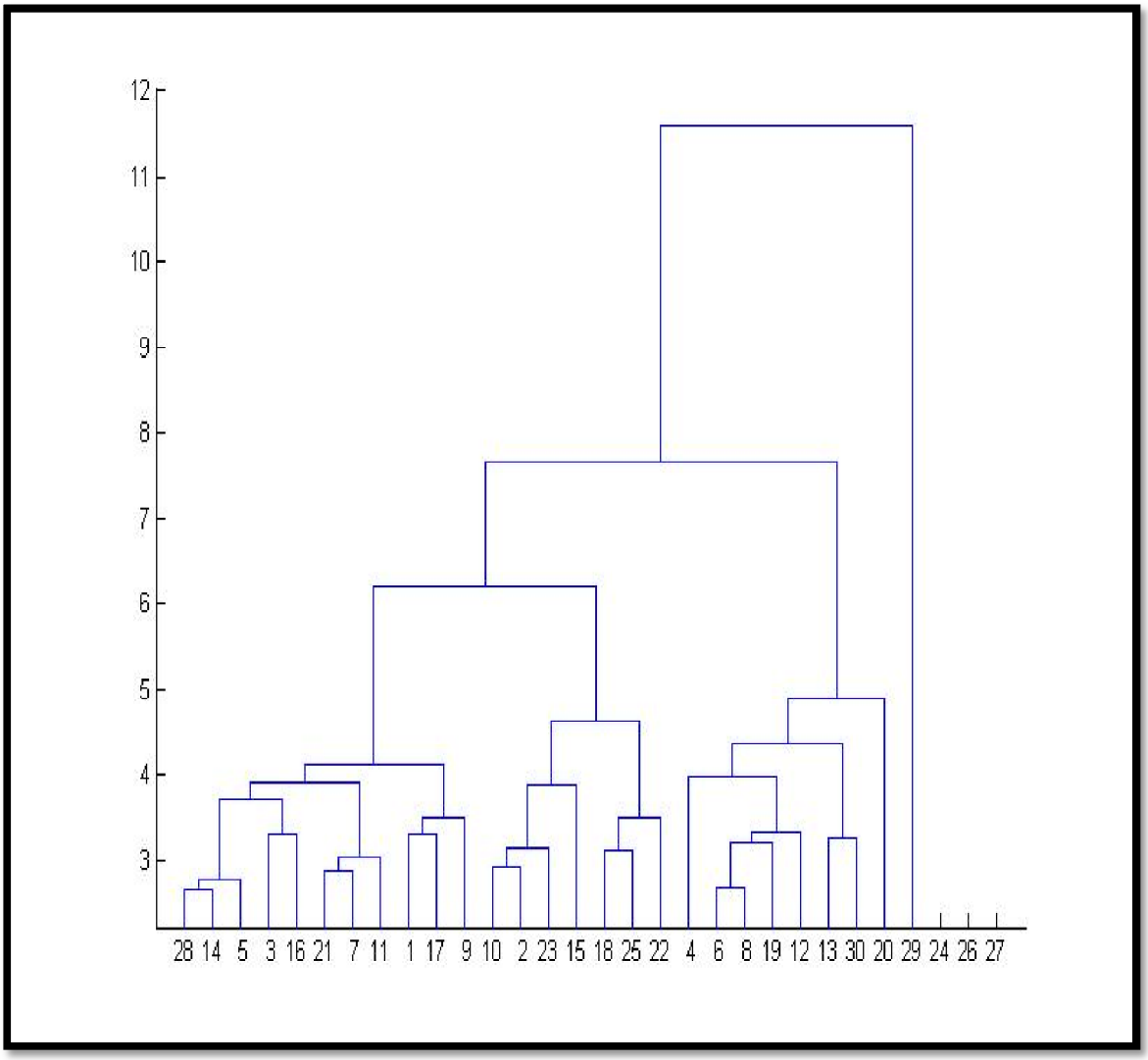
Ward Metodu'nun MATLAB platformunda Türkiye illeri ortalama sıcaklık verileri ile uygulanması sonucu elde edilen dendrogram, aşağıda verilen Şekil 3.26'daki gibidir. Elde edilen sonuç, tezimizin 3.1.2. bölümündeki WEKA platformuyla yapılan Şekil 3.11'deki Ward Metodu analiziyle karşılaştırıldığında, aralarında büyük benzerlikler görülmektedir. Bu bilgiler ışığında elde edilen dendrogramı aşağıdaki Şekil 3.26'den inceleyebilirsiniz.



Şekil 3.26 : Türkiye İlleri Ortalama Sıcaklık Verileri Ward Metodu Kümeleme Dendrogramı

Ortalama bağlantılı kümeleme analizi

Hiyerarşik Kümeleme için kullandığımız Ortalama Bağıntılı MATLAB kümeleme kodunun, Türkiye illeri ortalama sıcaklık verilerinde kullanılması sonucu elde edilen dendrogram aşağıdaki gibidir. Oluşan Ortalama Bağlantılı Kümeleme dendrogramını aşağıda verilen Şekil 3.27'da inceleyebilirsiniz. Ayrıca elde ettiğimiz dendrogram, tezimizin 3.1.2. bölümündeki WEKA analizlerinde elde edilen Şekil 3.12'deki dendrograma büyük benzerlik göstermektedir.



Şekil 3.27 : Türkiye İlleri Ortalama Sıcaklık Verileri Ortalama Bağlantılı Kümeleme Dendrogramı

3.2.3. Expectation maximization kümeleme algoritması analizleri

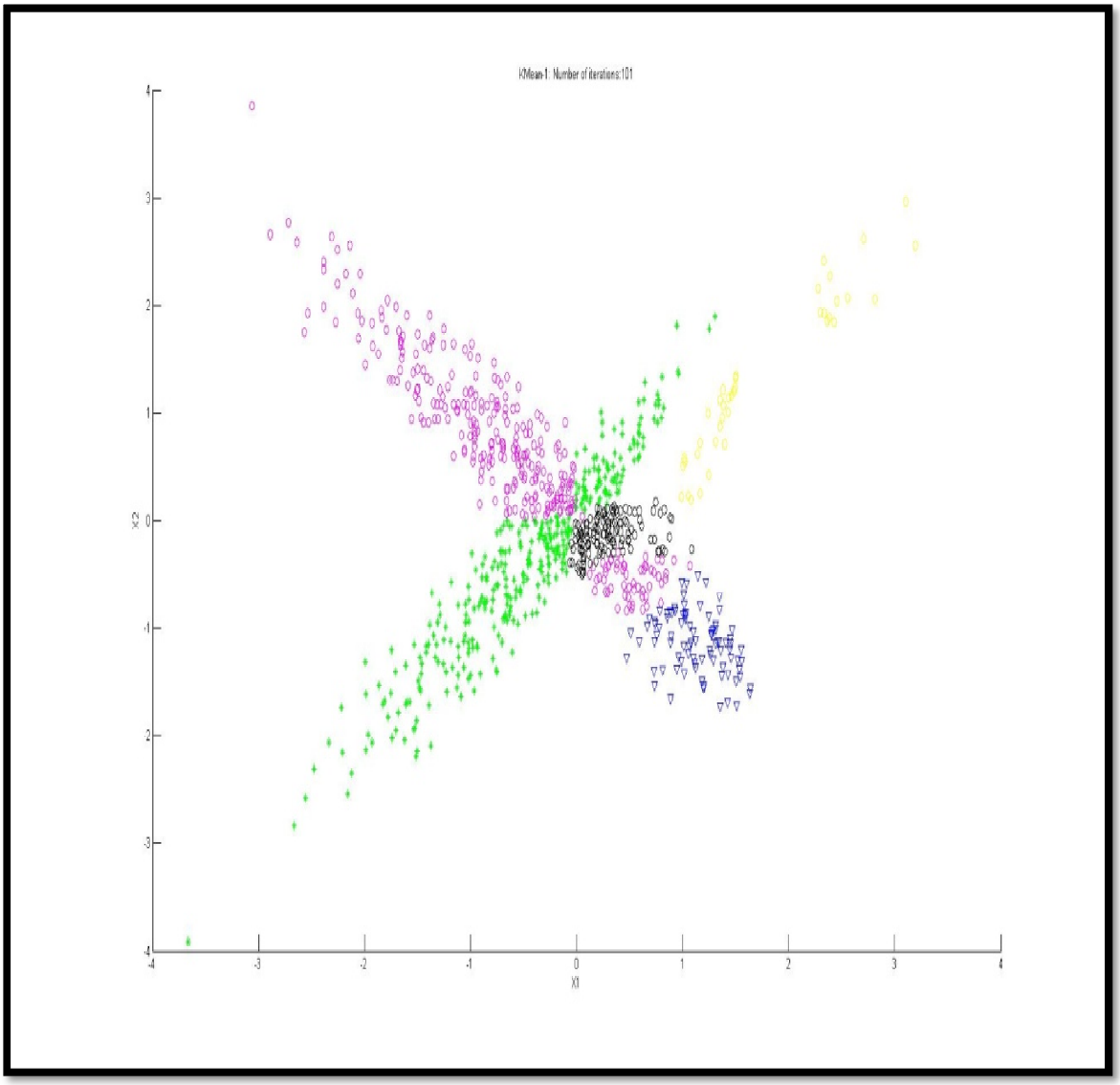
MATLAB platformunda Expectation Maximization kümeleme algoritması uygulanırken WEKA'da olduğu gibi programda oluşacak küme sayısını belirlemek gibi bir pratik durum bulunmamaktadır. Bu sebeple kullanmış olduğumuz kodun ilgili yerinde bu değeri belirtmemiz gerekmektedir.

Expectation Maximization algoritması her ne kadar K-Means algoritmasının geliştirilmiş bir hali gibi düşünülse de, çok daha komplike bir algoritmadır. Birçok tekrar ve küme merkezlerinin hareket ettirilmesi sonucu ideal lokasyon belirlenir ve merkez noktasının hareket etmediği duruma kadar algoritma kendini tekrar etmeye devam eder. Bu sebeple kullandığımız kod olan EK F'de bulunan Expectation Maximization kümeleme algoritması kodu verilerimiz olan; Kandilli ortalama rüzgar şiddeti verileri, Kandilli maksimum, minimum ve ortalama sıcaklık verileri ve Türkiye illeri ortalama sıcaklık verilerine teker teker uygulanıp elde edilen en başarılı sonuçlar aşağıdaki başlıklarda açıklanmıştır. Seçilen k değerleri belirlenirken 3.2.1. bölümündeki K-Means sonuçları dikkate alınarak hareket edilmiştir.

WEKA platformunda yapmış olduğumuz Expectation Maximization kümeleme sonuçlarına nazaran MATLAB'da elde edilen sonuçlar tamamen görüntü verileridir. WEKA'da elde edilen metinsel sonuçların görsel halleri her ne kadar incelenebilir olsa dahi, anlaşılırlık ve görsellik açısından MATLAB'da elde ettiğimiz sonuçlara kıyasla çok daha zayıftır. MATLAB'da elde ettiğimiz sonuçlar yukarıda açıklanan sebeplerden ötürü metinsel açıdan WEKA'da uyguladığımız Expectation Maximization uygulamalarına nazaran daha zayıftır. Kod üzerinde yapılacak bazı değişiklikler ile MATLAB platformunda istenilen bilgiler elde edilebileceği gibi, bu tarz kodda yapılan değişiklikler iyi bir yazılımcı tarafından yapılmaması durumunda, programda hatalar ortaya çıkmasına ve çalışmamasına sebep olabilir. Bu sebeple çalışmamızda elde edilen sonuçlar metinsel veriler değildir. Yukarıda açıklanan sebeplerden ötürü aşağıda verilen sonuçlar tamamen görsel verilerden oluşmaktadır.

Kandilli ortalama rüzgar şiddeti verileri analizi

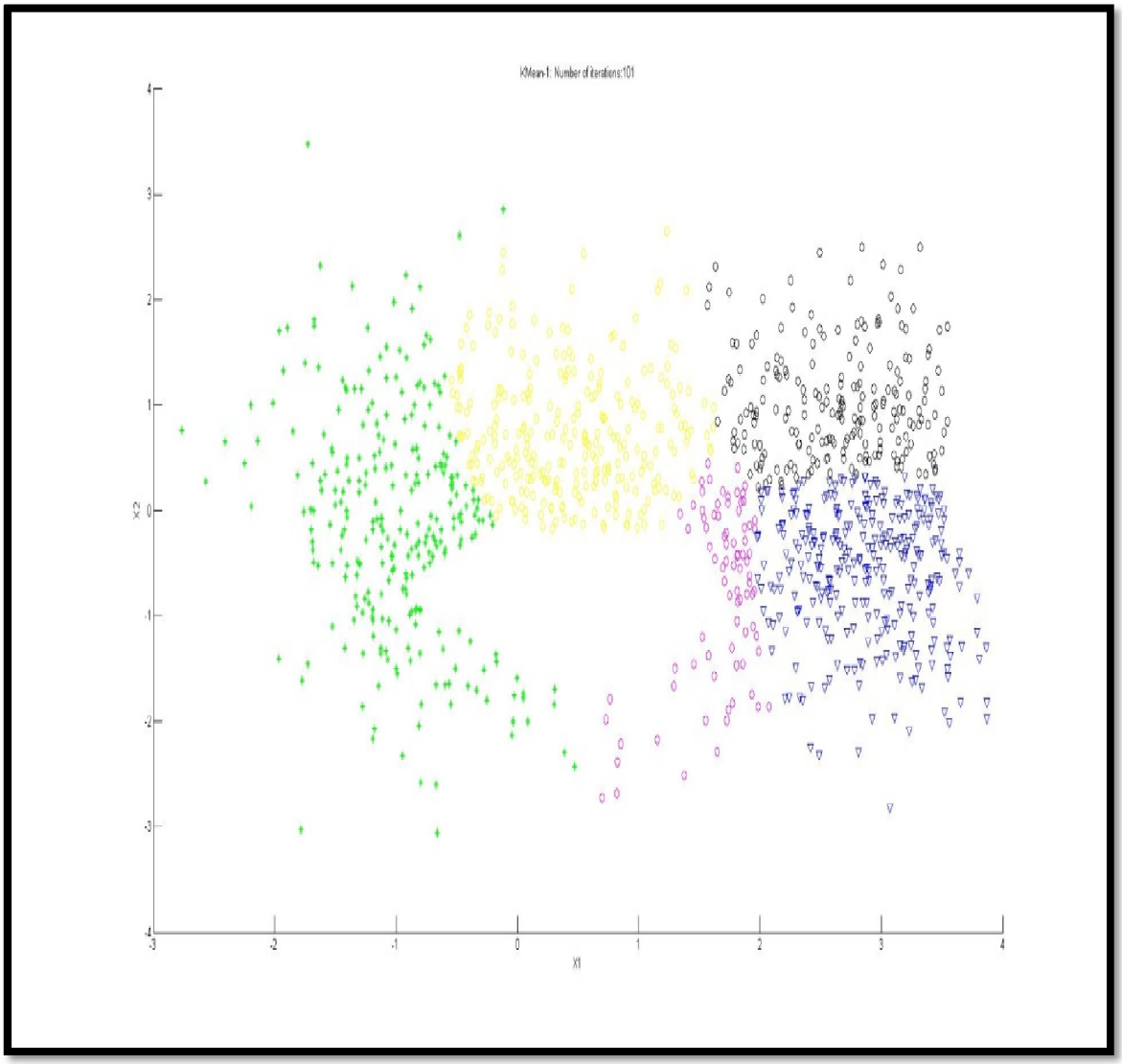
Kandilli ortalama rüzgar şiddeti verilerinin EK F'de kodları verilen Expectation Maximization kümeleme algoritmasına sokulması sonucu elde edilen kümeleme görüntüsü Şekil 3.28'de verilmiştir. Expectation Maximization algoritmasının uygulanması sonucu elde edilen aşağıdaki en optimal sonuç incelendiğinde, 7 küme oluştuğu görülmüştür.



Şekil 3.28 : Kandilli Ortalama Rüzgar Şiddeti Verileri E. M. Kümeleme Analizi

Kandilli maksimum, minimum ve ortalama sıcaklık verileri analizi

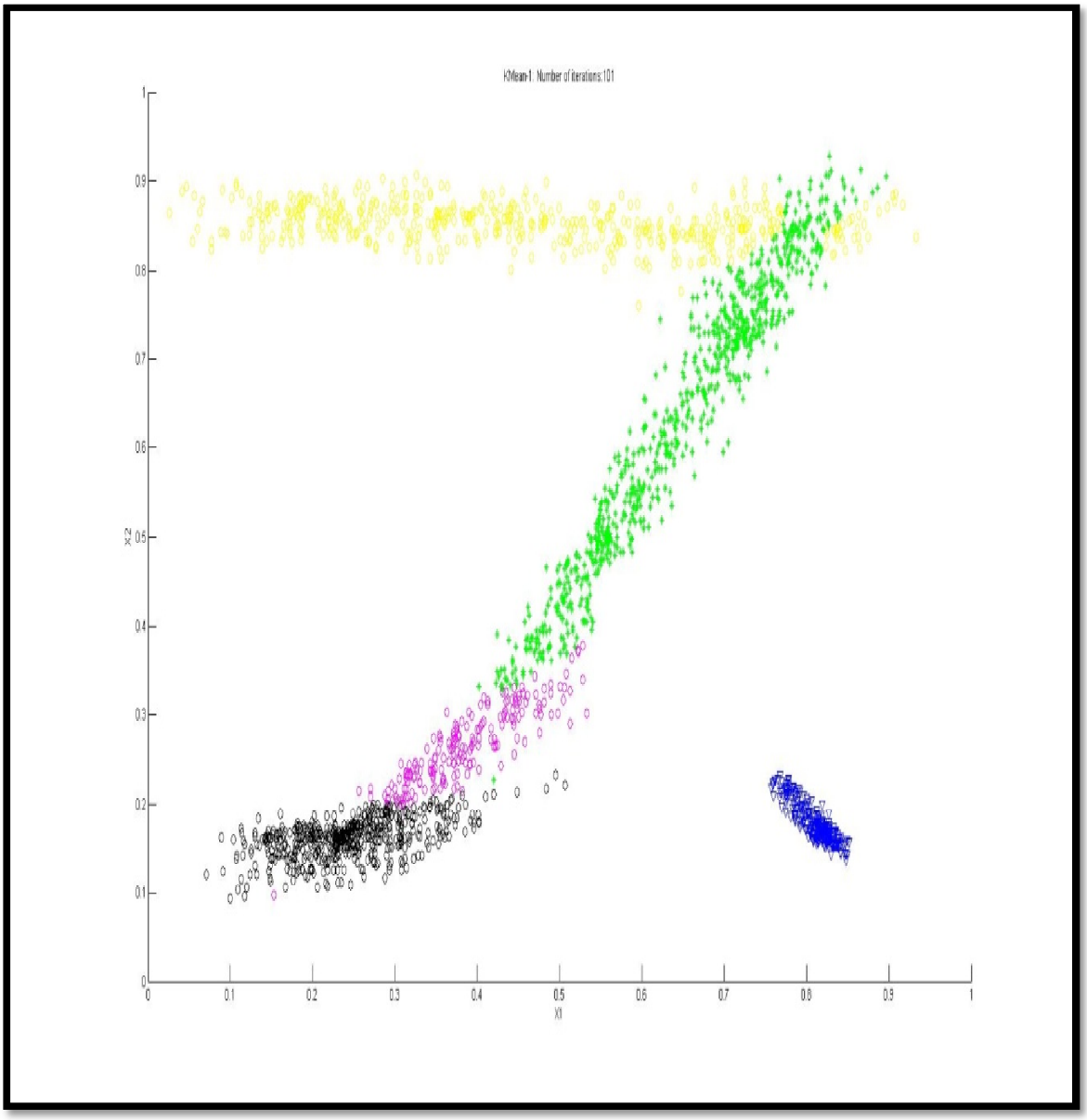
Çalışmamızın EK F bölümünde verilen Expectation Maximization kümeleme algoritması kodlarının, Kandilli maksimum, minimum ve ortalama sıcaklık verileri ile MATLAB'da uygulanması sonucu elde edilen kümeleme görüntüsü Şekil 3.29'da verilmiştir. Expectation Maximization algoritmasının uygulanması sonucu elde edilen aşağıdaki en optimal sonuç incelendiğinde, 7 küme oluştuğu görülmektedir.



Şekil 3.29 : Kandilli Maksimum, Minimum ve Ortalama Sıcaklık Verileri E. M. Kümeleme Analizi

Türkiye illeri ortalama sıcaklık verileri analizi

Türkiye illeri ortalama sıcaklık verilerinin Expectation Maximization kümeleme algoritmasına sokulması sonucu elde edilen kümeleme çıktısını aşağıdaki Şekil 3.30'daki gibidir. Oluşan sonuçlar incelendiğinde 8 küme merkezi ile optimal sonuca ulaşıldığı görülmektedir.



Şekil 3.30 : Türkiye İlleri Ortalama Sıcaklık Verileri E. M. Kümeleme Analizi

4. SONUÇ VE ÖNERİLER

4.1. Sonuç

Yapılan araştırmalar sonucunda elde edilen veriler incelendiğinde, öncelikle kullanılan platformların birbirlerinden ayrıldıkları farklarını açıklamak gerekmektedir. Öncelikle WEKA platformu açık kaynak kodlu bir programdır. Ücretsiz olarak programa erişebilir, kullanabilir ve üzerinde değişiklikler yapabilirsiniz. Ve paket program ile bilgisayarınıza kurduğunuz WEKA programı yapmak istediğiniz birçok veri madenciliği ve istatistiksel işlemi çok büyük bir sürat ve başarı ile gerçekleştirir. Yine içinde hazır gelen fonksiyonlar üzerinde istediğiniz değişiklikleri yapabilir ve bu değişikliklerin gerekli optimasyonları ile uğraşmak zorunda kalmazsınız. Program sorunsuz bir şekilde verilerinizi işler ve sonuçları kullanımınıza sunar. WEKA programında uzmanlaşmış yazılımcıların veri madenciliği üzerinde yapamayacakları hiçbir istatistiksel işlem yoktur. Programın belkide tek olumsuz yanı, profesyonel olmayan yazılımcıların yada meslek dışı eğitimsiz kullanıcıların, program üzerinde kayıtlı olmayan fonksiyonları tanıtmak amacıyla geliştirdikleri yada web sitelerinden edindikleri kodları programa entegre etme aşamasında yaşadıkları problemlerdir. WEKA programı kendi içinde çok tutarlı ve verimli bir programdır ancak dışarıdan, kullanıcılar tarafından yazılan kodlar iyi dizayn edilmediğinde program hatalar verir ve içinden çıkılamayacak durumlara gelinebilir. Bu sebeple WEKA'da kullanmak istediğimiz ancak programda kayıtlı olmayan bir fonksiyon gördüğünüzde ilk yapacağımız iş, programın fonksiyon kütüphanesinden paket program ile gelmeyen ama yüklenmeye hazır fonksiyonları kontrol etmek ve eğer aradığımız fonksiyon bu listede bulunmuyorsa ve iyi bir yazılımcı değilsek başka bir programı kullanmayı düşünmemiz gerekmektedir. MATLAB ise bu açıdan çok daha esnek olanaklar sunan, çok başarılı bir platformdur. MATLAB platformu ve uzantıları ücretli olmasına rağmen öğrenci yahut bilim insanı olmanız durumunda çok ekonomik bedeller ile kullanabileceğiniz çok kullanışlı ve birçok platformda ihtiyacınızı görebileceğiniz kaynak kodun bulunduğu bir platformdur.

İyi bir yazılım bilgisi ile MATLAB'da yapılamayacak hiçbir veri madenciliği ve istatistiksel işlem yoktur. İstediğiniz fonksiyonu MATLAB'da çalıştırabilir, üzerinde denemek istediğiniz yenilikleri yapabilir ve çıkan yazılımsal kodları kolaylıkla halledebilirsiniz. Ayrıca MATLAB platformunda çalıştığımız zaman verilerinizin işlenmesi sonucunda çıkan sonuçlar WEKA ve diğer birçok veri madenciliği platformuna nazaran daha fazla görsellik içermektedir. Verilerimizin üzerinde görmek istediğimiz işlem adımlarını çok daha kolay animasyon olarak görebilir ve metinsel olarak almak istediğimiz çıktılarıda kodda gerekli değişiklikleri yaptıktan sonra alabiliriz. Bu açılardan bakıldığında platform olarak MATLAB platformu ücretli olmasına rağmen WEKA'ya kıyasla daha kullanışlı bir platformdur. Seçmiş olduğumuz kümeleme algoritmaları olan K-Means, Hiyerarşik Kümeleme ve Expectation Maximization algoritmaları her iki platformda da başarı ile uygulanmıştır.

K-Means algoritması günümüzde en çok tercih edilen kümeleme algoritması olması sebebiyle ilk uygulanan algoritma olmuştur. WEKA'da yapılan analizler sonucunda her veri kümesinde de 10 küme oluşumu ile en başarılı sonuçlar elde edilmiştir. 1350 denemeden fazla yapılan çalışmalar sonucunda her veri kümesi için elde edilen optimal değerler tezimizin 3.1.1. bölümünde verilmiştir. Elde edilen veriler metinsel olarak çok başarılı ve performans olarak çok hızlı olup, görsel açıdan MATLAB platformunda tezimizin 3.2.1. bölümünde yapmış olduğumuz analizlere kıyasla daha zayıftır. Ancak bu durum WEKA'da hazır kurulu olan K-Means algoritmasının kullanım kolaylığı ve performans yüksekliği açısından MATLAB platformunun gerisinde kalmasına sebep değildir. WEKA, K-Means algoritmasının uygulanmasında MATLAB'a kıyasla çok daha verimli ve başarılı sonuçlar vermiştir. WEKA'da elde edilen sonuçların başarısı hata kare toplamı ile hesaplanmıştır. Bu sonuçlar aşağıdaki Çizelge 4.1'de verilmiştir. Bu sonuçlara göre en başarılı K-Means işlemi 7 tekrar ve 1.9529026194779933 değeriyle Kandilli maksimum, minimum ve ortalama sıcaklık verisi ile yapılan çalışmada alınmıştır.

Çizelge 4.1: WEKA Hata Kare Toplamı Sonuçları.

	Hata Kare Toplamı	Tekrar
Kandilli Ortalama Rüzgar Şiddeti Verileri	8.160678975344503	6
Kandilli Mak., Min. ve Ort. Sıcaklık Verileri	1.9529026194779933	7
Türkiye İlleri Ortalama Sıcaklık Verileri	20.305759581092552	5

MATLAB platformunda elde ettiğimiz K-Means sonuçları görsellik açısından mükemmel ancak WEKA'ya kıyasla performans ve kullanım kolaylığı açısından daha zayıftır. Çünkü MATLAB platformunda her denemek istediğiniz değişiklik için kodda değişiklik yapmanız gerekirken WEKA'da bu çok daha kolay yapılabilmektedir. Kısacası WEKA hem genel kullanıcılar hemde yazılım uzmanları için K-Means algoritmasının kullanılmasında çok daha verimli bir algoritmadır. Hiyerarşik kümeleme algoritması WEKA platformunda ele alındığında çok kolay ve süratli bir şekilde uygulanmıştır. İstediğimiz birçok değişiklik kullanım rahatlığı açısından kolayca yapılmış ve sağlıklı sonuçlar alınmıştır. MATLAB platformunda da Hiyerarşik kümeleme algoritması kolaylıkla yazılabilmiş ve istenilen bağlantı şekline göre sonuçlar kolayca alınmıştır. Bu kümeleme algoritmasında her iki platformda da elde edilen sonuçlar neredeyse aynı ve tutarlı sonuçlardır. Ancak performans ve kullanım kolaylığı açısından bakıldığında WEKA çok daha başarılı olmuştur.

Expectation Maximization algoritması her iki platformda da başarıyla uygulanmış ve sonuçları analizler bölümünde verilmiştir. Elde edilen sonuçları WEKA'da incelediğimizde metinsel açıdan neredeyse istatistiksel olarak merak edilen tüm sorular cevaplanmış ve performanslı bir şekilde işlem sonuçlanmıştır. MATLAB platformunda ise bu algoritmanın çalıştırılabilmesi için uzun süren bir araştırma sonucu ihtiyaç duyulan kod hazırlanmış ve başarı ile çalıştırılmıştır. Her üç veri kümesi için de başarı ile çalışan algoritmamızın sonuçları tezimizin 3.2.3. bölümünde verilmiştir. MATLAB'da da WEKA'daki gibi istatistiksel veriler metinsel olarak elde edilebilir ancak bunun için gerekli kod düzeltmelerinin yapılması gerekmektedir. Ancak elde edilen sonuçlar görsellik ve anlaşılabilirlik açısından WEKA'ya nazaran çok daha başarılıdır. Her bir veri kümesinin işlenmesi aşamasında animasyon olarak adımlarının tek tek görülmesi ve son halinin ortaya çıkması sayesinde Expectation Maximization

algoritmasının nasıl çalıştığı çok iyi bir şekilde anlaşılmıştır. WEKA ise bu görsellik açısından çok zayıf kalmıştır. Kısacası WEKA sisteminde kayıtlı olan fonksiyonlarda yüksek performans ve kullanım kolaylığına sahip olmasına rağmen, gelişmiş ve komplike algoritmaların sisteminde kayıtlı olmaması ve var olan algoritmalarında görselliğinin MATLAB'a kıyasla daha zayıf olması sebebiyle komplike algoritmalarda MATLAB'a kıyasla daha az başarılı olmuştur.

Yapmış olduğumuz tez çalışması sonucunda, tecrübesiz kullanıcıların WEKA platformunu basit ve sistemde kayıtlı kümeleme algoritmalarının kullanımında tercih etmeleri durumunda daha başarılı, anlaşılır, kullanımı kolay ve performanslı sonuçlar alacakları görülmüştür. Tecrübeli kullanıcıların ise MATLAB gibi kullanım serbestliği ve komplike algoritmalarda aldığı performanslı sonuçlar ile kümeleme algoritmalarının uygulanmasında tercih etmeleri durumunda daha başarılı sonuçlar alacakları görülmüştür.

4.2. Öneriler

Bu bölümde yapmış olduğumuz araştırmalar ve analizlerin sonuçlarından faydalanarak kümeleme algoritmaları üzerinde daha sonra yapılabilecek araştırma konuları aşağıdaki gibi sıralanabilir:

Günümüzde "Data Mining" yani "Veri Madenciliği" denildiğinde yaptığımız araştırmalar sonucunda elde edindiğimiz bilgiler ışığında özellikle Türkiye'de teknoloji sektörünün bulunduğu durum göz önüne alındığında yapılması gereken birçok araştırma olduğu görülmektedir. Bankacılık, taşımacılık, güvenlik, tarım, otomotiv, hayvancılık kısacası her alanda gerek sınıflandırma, gerek kümeleme, gerekse yapay sinir ağları ile eğitilmiş sistemler ile yapılacak veri madenciliği sonucunda ekonomik açıdan birçok fayda sağlanabileceği gibi günümüz bilim dünyasında da bu konularda yapılan araştırmaların azlığı sebebiyle ortaya çok faydalı çalışmalar çıkabilir. Doktora eğitimi aşamasında sahip olunan bu bilgiler ile yukarıda belirtilen alanlardan birinde yapılacak bir doktora tezi sayesinde önemli bilgiler elde edilebilir.

Yapmış olduğumuz bu çalışma esnasında farkettiğimiz ve kümeleme algoritmalarını test edebileceğimiz ORANGE ve özellikle R programları ile WEKA yada MATLAB

platformları arasında, farklı konularda karşılaştırılmalar yapılabilir. Bu sayede hem tecrübeli hem de tecrübesiz kullanıcıların veri madenciliği ve istatistik konularında seçmeleri gereken en optimal uygulamayı belirlemeleri mümkündür.

Ayrıca tezimizde bahsettiğimiz ancak kullanmadığımız birçok kümeleme algoritması ve algoritmanın tarihçesi bölümünde bahsettiğimiz birçok ilkel veya komplike algoritmaları MATLAB gibi çalışma alanı geniş bir platformda uygulayıp, iklim değişimi vb. konularda seçilebilecek veriler üzerinde çalışarak çeşitli çıkarımlarda bulunulabilir.

KAYNAKLAR

Aaboe, A. (2001). Episodes from the Early History of Astronomy. New York: Springer, pp. 40- 62, ISBN 0-387-95136-9

Aher, B. S. ve Lobo, L. (2012). Applicability of Data Mining Algorithms for Recommendation System in E-Learning. ICACCI '12, August 03 - 05 2012, CHENNAI, India, ACM 978-1-4503-1196-0/12/08.

Anand, R. S., Dewal, M. L., Gupta, S., Yadav, A. R. (2015). Performance analysis of discrete wavelet transform based first-order statistcal texture features for hardwood species classification. 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015), Procedia Computer Science 57 (2015) 214 - 221.

Azahar, F. T., Hassan, R., Singh, M. M. (2015). RFID-Enabled Supply Chain Detection Using Clustering Algorithms. IMCOM '15, January 08 - 10 2015, BALI, Indonesia, ACM 978-1-4503-3377-1/15/01.

Bhadauria, S. H., Kumar, I., Virmani, J. (2015). Wavelet Packet Texture Descriptors based four-class BIRADS Breast Tissue Density Classification. 4thInternational Conference on Eco-friendly Computing and Communication Systems, Procedia Computer Science 70 (2015) 76 - 84.

Bin, L., Chen, S., Dongmei, S., Jianyong, C., Shouchang, C., Yajie, Z., Yi, M. (2015).Classification of the Different Thickness of the Oil Film based on Wavelet Transform Spectrum Information. International Oil Spill Response Technical Seminar, Aquatic Procedia 3 (2015) 133 – 143.

Castro, F. P., Pinheiro, A. W., Souza, d. J., Xexéo, G. (2008). Using Wavelets to Classify Documents. 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, 978-0-7695-3496-1/08, IEEE DOI 10.1109/WIIAT.2008.221.

Chen, L., Chirkova, R., Yu, T. (2015). WaveCluster with Differential Privacy. CIKM'15, October 19–23, 2015, Melbourne, Australia, ACM. ISBN 978-1-4503-3794-6/15/10, DOI: http://dx.doi.org/10.1145/2806416.2806546.

Cooke, R. L. (2005). The History of Mathematics: A Brief Course. John Wiley & Sons. ISBN 9781118460290.

Dener, M., Dörterler, M., Orman, A. (2009). Açık Kaynak Kodlu Veri Madenciliği Programları: WEKA'da Örnek Uygulama, ab.org.tr/ab09/bildiri/42.pdf, 18.01.2013.

Du, Y. ve Wang Z. J. (2001). Scalable Integrated Region-based Image Retrieval using IRM and Statistical Clustering. JCDL'01, June 24-28, 2001, Roanoke, Virginia, USA, ACM 1-58113-345-6/01/0006.

Dubes, C. R. ve Jain, K. A. (1988). Algorithms for Clustering Data. Prentice Hall Englewood Cliffs, New Jersey 07632, pp. 55 - 141, ISBN 0-13-022278-X

Dueck, D. ve Frey, J. B. (2007). Clustering by passing messages between data points. Science 315 *(*5814*):*972976. doi*:*10.1126/science.1136800.PMID 17218491.

Dutta, A., Pal, U., Shivakumara, P., Tan, L. C. (2010). A New Wavelet-Median-Moment based Method for Multi-Oriented Video Text Detection. DAS '10, June 9-11, 2010, Boston, MA, USA, ACM 978-1-60558-773-8/10/06.

Ester, M., Kriegel, H. P., Sander, J., Xu, X. (1996). Simoudis, Evangelos; Han, Jiawei; Fayyad, Usama M., eds. A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press. pp. 226–231. ISBN 1-57735-004-9. CiteSeerX: 10.1.1.121.9220.

Fayyad, U., Piatetsky-Shapiro, G., Smyth, P. (1996). "From Data Mining to Knowledge Discovery in Databases". American Association for Artificial Intelligence. 0738-4602-1996

Fukunaga K. ve Hostetler D. L. (1975). The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition. IEEE Transactions on Information Theory (IEEE) 21 (1): 32– 40. doi: 10.1109/ TIT.1975. 055330.

Gunopulos, D., Keogh, E., Lin, J., Vlachos, M. (2003). A Wavelet-Based Anytime Algorithm for K-Means Clustering of Time Series. In: Proceedings of Workshop on Clustering High Dimensionality Data and Its Applications, pp. 23-30 (2003).

Ilarionov, R., Shopov, N., Simeonov, I., Kilifarev, H. (2008). Formation of Attribute Spaces Using Wavelets in Automatic Classification of Explosives. International Conference on Computer Systems and Technologies - CompSysTech'08, Gabrovo, Bulgaria, June 12-13, 2008, ACM ISBN: 978-954-9641-52-3/08/06.

İşler Y. ve Narin A. (2012). Weka Yazılımında k-Ortalama Algoritması Kullanılarak Konjestif Kalp Yetmezliği Hastalarının Teşhisi. SDU Teknik Bilimler Dergisi, 2012, 2 (4) 21-29.

Karpinski, L. C. (1912). History of Mathematics in the Recent Edition of the Encyclopædia Britannica. American Association for the Advancement of Science.

Kendall, M. G. (1966). Discrimination and Classification. In Multivariate Analysis (P.R. Krishnaiah, ed.), Academic Press, Inc., New York, pp. 165-185

Misiti, M., Misiti, Y., Oppenheim, G., Poggi, M. J. (2007). Clustering Signals Using Wavelets. Springer-Verlag Berlin Heidelberg 2007. F. Sandoval et al. (Eds.): IWANN 2007, LNCS 4507, pp. 514-521,2007.

Rao, V. G., Suryanarayana, V. S., Veeraswamy, G. (2015). Spectral Clustering Algorithm for Navie Users. ICARCSET '15, March 06 - 07, 2015, Unnao, India, ACM 978-1-4503-3441-9/15/03, http://dx.doi.org/10.1145/ 2743065.2743076.

Steinhaus, H. (1957). Sur la division des corps meteriels en parties. Bull. Acad. Polon Sci. (in French) 4(12): 801-804. MR 0090073. Zbl 0079. 16403.

Internet Kaynakları :

Url-1 <*http://tarihmeydani1071.blogspot.com.tr/2013/08/harezmi.html*>, alındığı tarih: 27.03.2016.

Url-2 <*https://tr.wikipedia.org/wiki/Algoritmaların_tarihsel_sıralaması*>,alındığı tarih: 28.03.2016.

Url-3 <*https://apandre.wordpress.com/visible-data/cluster-analysis/*>, alındığı tarih: 29.03.2016.

Url-4 <*https://sites.google.com/site/dataclusteringalgorithms/k-means-clustering-algorithm*>, alındığı tarih: 29.03.2016.

Url-5 <*http://www.sthda.com/english/wiki/hierarchical-clustering-essentials-unsupervised-machine-learning*>, alındığı tarih: 29.03.2016.

Url-6 <*http://www.slideshare.net/soyeon1771/spectral-clustering*>, alındığı tarih: 30.03.2016.

Url-7 <*http://www.mathworks.com/matlabcentral/fileexchange/40990-mean-shift-pixel-cluster/content/meanShiftPixCluster.m*>, alındığı tarih: 30.03.2016.

Url-8 <*https://en.wikipedia.org/wiki/Expectation–maximization_algorithm*>, alındığı tarih: 2.04.2016.

Url-9 < *http://ncdd.com.br/* >, alındığı tarih: 15.04.2016

Url-10 <*http://www.mathworks.com/matlabcentral/profile/authors/4171145-sultan-alzahrani?requestedDomain=www.mathworks.com*>, alındığı tarih: 17.04.2016

EKLER

EK A: Türkiye İlleri Ortalama Sıcaklık Verileri K-Means Analizi Sonucu

EK B: Kandilli Ortalama Rüzgar Şiddeti Verileri EM Analizi Sonucu

EK C: Türkiye İlleri Ortalama Sıcaklık Verileri EM Analizi

EK D: K-Means MATLAB Kodu

EK E: Hiyerarşik Kümeleme Algoritması MATLAB Kodu

EK F.1: Expectation Maximization Kümeleme Algoritması MATLAB Ana Program Kodu

EK F.2: Expectation Maximization Kümeleme Algoritması MATLAB Ana Program Fonksiyonlarının Kodları

EK G : K-Means Kümeleme Algoritması Kandilli Ortalama Rüzgar Şiddeti Verileri Analizi

EK H : Kandilli Ortalama Rüzgar Şiddeti Ward Metodu Çıktısı

EK I : Kandilli Maksimum Minimum ve Ortalama Sıcaklık Ward Metodu Çıktısı

EK J : Türkiye İlleri Ortalama Sıcaklık Tam Bağlantılı Kümeleme Çıktısı

EK K : Türkiye İlleri Ortalama Sıcaklık Ortalama Bağlantılı Kümeleme Çıktısı

EK A

A.1 : Türkiye İlleri Ortalama Sıcaklık Verileri K-Means Analizi Sonucu

=== Run information ===

Scheme: weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 10 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 35

Relation: trortscklk

Instances: 64

Attributes: 49

yil, adana, afyon, ankara, antalya, artvin, balikesir, bandirma, bilecik, bolu, burdur, bursa, canakkalediyarbakir, edirne, elazig, erzincan, erzurum, eskisehir, gaziantep, giresun, gumushane, hatay, isparta, istanbul, izmir, kars, kastamonu, kayseri, kocaeli, konya, kutahya, malatya, manisa, mardin, mersin, mugla, ordu, rize, samsun, siirt, sinop, sivas, tekirdag, trabzon, urfa, usak, van, zonguldak

Test mode: evaluate on training data

=== Clustering model (full training set) ===

=== kMeans ===

Number of iterations: 5

Within cluster sum of squared errors: 20.305759581092552

Initial starting points (random):

Cluster 0:
1984,19.2,14,10.4,8.7,14.5,4,12.8,14,11.5,8.6,11.3,11.7,13.4,15.7,13.2,13.5,7.6,4.8,10.2,15.9,8.7,8.8,17.2,11.1,13.8,15.6,4.5,9.4,8.9,13.9,10.9,10.2,11.6,15,16.1,12.7,13.8,13.5,13.6,13.4,9.8,13,7.9,13.1,15.1,17.4,11.2,9.6

Cluster 1:
1982,18.9,13.7,10.3,8.3,14.2,3.7,12.6,13.8,11.4,8.3,10.9,11.6,13.2,14.3,13.1,12.6,6.5,3.9,10.1,15.4,8.3,8.3,16.7,10.8,13.6,15.3,3.2,8.9,8.4,13.7,10.5,10.1,10.8,14.8,14.8,12.3,13.5,13.5,13.4,13.5,8.6,12.9,6.9,12.9,14.9,16.7,11.1,8.2

Cluster 2:
1998,20.4,15.1,12,9.9,16.1,4.4,13.7,14.7,12.6,9.8,12.7,12.6,13.9,15.8,13.4,14.8,8.7,4.8,11.4,17,10,10,18.5,12.6,14.6,16.4,4.3,10.7,9.8,15,12.1,11.4,12.8,16,16.6,13.8,14.8,14.8,15,14.5,10.6,14.1,8.4,13.6,16.4,18.3,12.5,10.3

Cluster 3:
1947,19.9,14.1,11.1,9.1,15.3,4.4,13.3,14.3,11.8,8.8,12,12,13.7,15.5,13.1,13.6,7.3,4.9,10.6,16.3,9.5,9.2,17.8,11.7,13.9,16.3,4.2,10,9.6,14.1,11.5,10.6,11.9,15.8,16,13.2,14.5,14.8,14.2,14.8,9.5,14,8.6,13.2,15.8,17.7,

11.7,9.2

Cluster 4:
1970,19.6,14.6,11.3,9.2,15.1,4.7,13.2,14.4,12,9,12,12,14,16.3,13.5,13.9,7.7,5.4,10.8,16.6,9.5,9.4,17.6,11.9,14.3,16,4.6,10,9.4,14.5,11.5,10.7,12.1,15.5,16.9,13,14.2,14.5,14.5,14.5,10.2,13.8,8.4,13.6,15.9,18.2,11.7,9.5

Cluster 6:
1967,18.3,13.9,9.6,7.6,13.9,2.9,12.3,13.5,10.9,8,10.5,11.1,13.3,13.9,12.8,11.7,5.2,3.3,9.6,14.6,8.1,7.8,16.4,10.3,13.6,15.4,2.4,9,7.2,13.8,9.7,9.4,9.8,14.8,14.5,11.6,13.5,13.7,13.3,13.7,8.2,13.2,6.1,12.9,14.9,16.1,10.5,7.5

Cluster 7:
1952,19.3,14.6,11.4,9.1,15.2,4.1,13.5,14.6,12.2,9.1,12.3,12.2,14.5,14.8,13.8,13.1,6.8,4.4,10.9,15.5,9.5,9.2,17.3,12.1,14.4,16.5,3.9,9.9,9.1,14.5,11.5,11,11.2,15.9,15.5,12.9,14.8,14.8,14.3,14.9,9.1,14.1,8.1,13.8,16,17,12,8.2

Cluster 8:
1942,19,12.7,9.9,7.8,14.4,3.3,11.9,12.8,10.4,7.5,11,10.5,12.3,14.6,11.3,12.7,6.4,3.9,9.2,15.5,8.4,8.2,17,10.7,12.3,15.1,3.3,8.6,8.6,12.6,10.4,9.2,11,14.6,15.2,12.3,13.6,13.4,13.2,13.4,9,12.6,7.6,11.5,14.7,16.9,10.5,8.6

Cluster 9:
1943,18.7,12.8,9.8,7.2,14.3,3.3,12.1,13.2,10.5,7.2,10.9,10.8,12.7,14,12.1,12.1,6.1,3.7,9.1,15,8.4,8.2,16.7,10.5,12.7,15.3,3.5,8.6,8,12.7,10,9.3,10.3,14.7,14.5,12,13.6,13.8,13.4,13.9,8.2,12.9,6.9,12,15,16.3,10.7,8.2

Missing values globally replaced with mean/mode

Final cluster centroids:

	Cluster#										
Attribute	Full Data	0	1	2	3	4	5	6	7	8	9
	(64.0)	(12.0)	(10.0)	(3.0)	(6.0)	(5.0)	(5.0)	(4.0)	(7.0)	(3.0)	(9.0)
yil	1969.5	1980.6667	1981.7	2000	1956.8333	1984.2	1975	1970.25	1947	1941	1954.7778
adana	19.1672	19.2583	18.98	19.4	19.5667	19.7	20.32	18.275	19.1286	18.9667	18.4667
afyon	14.0047	14.0417	13.88	15.1667	14.3167	14.6	15.1	13.7	14	13	13.0333
ankara	10.6172	10.7417	10.32	12.1	10.8667	11.28	11.84	9.625	10.6857	10.0333	9.6556
antalya	8.7141	8.6917	8.3	12.6667	8.9	9.38	9.84	7.625	8.7	8.0333	7.4889
artvin	14.8766	14.7167	14.44	19.2	15.0167	15.1	15.68	14.05	14.7286	14.4	14.1111
balikesir	4.1813	3.925	3.46	12.4667	4.1333	4.72	4.94	2.875	3.8714	3.8	

2.8222

bandirma	12.8797	12.8917	12.66	15.1	13.1667	13.26	13.8	12.525	12.8	12.1	11.9333
bilecik	13.2266	14.0417	13.77	0	14.2333	14.36	14.88	13.7	13.9	13.0667	13.0444
bolu	11.5594	11.625	11.37	13.3	11.8	12.1	12.64	10.975	11.5714	10.7	10.5778
burdur	8.6141	8.6583	8.39	11.2667	8.8333	9.2	9.6	7.925	8.5714	7.7	7.5444
bursa	11.5	11.525	11.11	13.6	11.7667	11.76	12.58	10.7	11.5571	11	10.7556
canakkale	11.7687	11.7833	11.53	15.4	11.8833	12.18	12.64	11.275	11.6	10.8	10.6889
diyarbakir	13.3516	13.4417	13.2	16	13.9333	13.72	14.3	13.45	11.9714	12.4	12.7444
edirne	15.0359	15.3917	14.86	16.2333	15.45	15.96	16.26	13.55	14.4429	14.4667	14.2
elazig	13.0141	13.1	12.85	14.5333	13.35	13.4	13.78	13.125	12.8143	11.6	12.2889
erzincan	12.9609	13.325	12.78	13.3667	13.3333	14.02	14.58	11.425	12.4571	12.6667	11.9778
erzurum	6.975	7.05	6.53	12.1333	7.0667	7.94	8.42	5.175	6.4286	6.5	5.6333
eskisehir	4.3266	4.6083	3.93	5.8333	4.6833	5.34	5.08	3.025	4.1857	4.5333	3.2889
gaziantep	10.2641	10.3667	10.12	11.2333	10.55	10.9	11.36	9.65	10.3143	9.5	9.3
giresun	15.7141	16.1667	15.6	16.1333	15.85	16.52	16.9	14.675	15.2286	15.3667	14.8556
gumushane	9.1453	8.7917	8.52	15.2667	9.2167	9.54	10.12	8.125	9.1143	8.7667	8.0667
hatay	8.7423	8.6833	8.4	9.9	9.0833	9.48	10	7.875	8.8586	8.5667	7.8333
isparta	17.0766	17.3667	17	18.8333	17.4833	17.72	18.28	16.3	15.5571	16.9333	16.4667
istanbul	11.2875	11.3833	10.95	13.0333	11.5333	11.66	12.5	10.375	11.3429	10.8	10.4333
izmir	13.7578	13.7917	13.57	15.3333	14.0333	14.18	14.74	13.625	13.7143	12.6667	12.8889
kars	15.8703	15.7333	15.48	18.8333	16.2833	16.02	16.64	15.65	15.8	15.1333	

15.1111
kastamonu 3.6438 3.8 3.05 5.8 3.9333 4.78 4.74 2.325 3.4857 3.7333
2.6222
kayseri 9.4094 9.375 9.15 10.1667 9.7333 10.14 10.66 8.9 9.6286 8.9 8.4
kocaeli 8.8453 8.75 8.34 11.4333 9.35 9.48 10.22 7.525 8.8429 8.7 7.8556
konya 13.9422 13.9417 13.77 15.5333 14.2167 14.46 15.06 13.7 13.9143 12.9
12.9889
kutahya 10.8484 10.9333 10.54 12.1333 11.15 11.54 12.14 9.725 10.9857 10.5333
9.8444
malatya 10.2297 10.3417 10.02 11.3 10.5167 10.86 11.36 9.525 10.3 9.5
9.2889
manisa 11.2578 11.5333 10.9 14.4 11.45 12.14 12.68 9.6 10.6571 10.9333
10.1444
mardin 15.3109 15.225 14.98 17.7667 15.7 15.52 16.16 15.025 15.2571 14.6667
14.5111
mersin 15.5672 15.9667 15.43 17 15.9167 16.3 16.72 14.225 14.9143 15
14.7222
mugla 12.7469 12.65 12.35 20.2333 12.9333 13.12 13.68 11.625 12.6 12.3333
10.7222
ordu 14.0172 13.9583 13.6 15.8333 14.4833 14.12 14.84 13.575 14.0286 13.5333
13.4778
rize 14.025 13.8083 13.58 15.0333 14.3667 14.52 15.08 13.675 14.3857 13.8667
13.3111
samsun 13.8625 13.75 13.59 15 14.1167 14.56 14.94 13.275 14.0714 13.6
12.9667
siirt 13.9609 13.6833 13.48 15.1 14.2833 14.44 14.86 13.75 14.3857 13.8333
13.3111
sinop 9.5297 9.65 9.08 17.0667 9.45 10.2 10.4 7.925 8.5286 9.0333 8.2111
sivas 13.1547 13.1 12.91 14.8667 13.6833 13.8 14.32 13.05 13.6 12.9333
11.3444
tekirdag 7.65 7.6917 6.99 9.9667 7.9833 8.34 8.92 6.275 7.8286 7.8 6.6667
trabzon 13.0266 13.0667 12.8 14.7333 13.4167 13.36 13.86 13.05 12.9 11.8
12.2444
urfa 15.3453 15.2167 15.09 15.4 15.7167 15.96 16.5 14.9 15.6429 15.1333
14.5889
usak 17.225 17.5667 17.05 19.1 17.4667 18 18.34 16.05 16.5714 16.7
16.3333

van	11.3266	11.4083	11.04	13.1333	11.5833	11.8	12.4	10.6	11.3143	10.6667
10.4556										
zonguldak	8.7984	9.0917	8.52	10.6	9.1167	9.98	9.9	7.425	8.1714	8.7333
7.7556										

Time taken to build model (full training data) : 0.02 seconds

=== Model and evaluation on training set ===

Clustered Instances

0	12 (19%)
1	10 (16%)
2	3 (5%)
3	6 (9%)
4	5 (8%)
5	5 (8%)
6	4 (6%)
7	7 (11%)
8	3 (5%)
9	9 (14%)

EK B

B.1 : Kandilli Ortalama Rüzgar Şiddeti Verileri EM Analizi Sonucu

=== Run information ===

Scheme: weka.clusterers.EM -I 100 -N 10 -X 10 -max -1 -ll-cv 1.0E-6 -ll-iter 1.0E-6 -M 1.0E-6 -K 10 -num-slots 1 -S 100

Relation: ortruzsid

Instances: 64

Attributes: 13

yil, ocak, subat, mart, nisan, mayis, haziran, temmuz, agustos, eylul, ekim, kasim, aralik

Test mode: evaluate on training data

=== Clustering model (full training set) ===

== EM ==

Number of clusters: 10

Number of iterations performed: 0

	Cluster									
Attribute	0	1	2	3	4	5	6	7	8	9
	(0.03)	(0.25)	(0.09)	(0.17)	(0.06)	(0.23)	(0.11)	(0.02)	(0.02)	(0.02)
yil										
mean	2014.5	1963.25	1966.8333	2000.1818	1977.5	1981.9333	2007.4286	1993	2007	2009
std. dev.	0.7071	9.2556	8.9088	4.9157	10.4083	7.3043	5.0285	0	0	0
ocak										
mean	3.2	4.65	4.4833	3.3909	5.4	5.0067	3.2571	0	3.8	0
std. dev.	0.8485	0.7492	0.8565	0.4867	1.0863	0.7469	0.5192	0	0	0
subat										
mean	3.5	4.6625	4.85	3.6364	3.7	5.26	3.0143	5.7	3.1	0
std. dev.	0.9899	0.8617	0.2881	0.5104	0.9487	0.7661	0.2035	0	0	0
mart										
mean	2.9	4.2062	4.0667	3.3091	3.8	4.4733	3.2	4.8	3.3	0
std. dev.	0.1414	0.6361	0.6346	0.378	1.2302	0.4559	0.2646	0	0	0
nisan										
mean	2.8	3.7125	3.5833	3.0727	3.7	4.0267	2.7857	4.5	2.6	0
std. dev.	0.4243	0.683	0.371	0.3797	0.5354	0.6628	0.3132	0	0	0
mayis										
mean	2.6	2.9688	3.2167	2.8273	3.05	3.9	2.6857	0	2.4	0

std. dev.	0.1414	0.4254	0.5811	0.2832	0.4655	0.6908	0.3237	0	0	0
haziran										
mean	2.85	3.3312	3.7833	2.6818	2.775	3.3733	2.7857	0	2.4	2.5
std. dev.	0.2121	0.4078	0.4875	0.1991	0.4646	0.6341	0.1574	0	0	0
temmuz										
mean	0	3.8188	3.9	3.0545	3.125	4.4333	2.8714	3	3.3	2.9
std. dev.	0.9696	0.4665	0.4	0.383	0.556	0.6275	0.4152	0	0	0
agustos										
mean	3.1	3.7063	4.2833	2.8636	2.55	4.48	2.8714	3.4	0.8	3
std. dev.	0.1414	0.5348	0.2858	0.2656	0.9037	0.5185	0.3302	0	0	0
eylul										
mean	3.1	3.2563	3.85	2.6818	2.325	3.9467	2.5	2.8	0.2	2.7
std. dev.	0.1414	0.3444	0.3674	0.1722	0.5123	0.3889	0.3	0	0	0
ekim										
mean	3.15	3.075	3.9833	2.7818	3.025	4.1467	2.6714	2.4	2.6	2.7
std. dev.	0.3536	0.5825	0.796	0.343	0.4573	0.8175	0.138	0	0	0
kasim										
mean	2.8	3.2375	5.15	3.1273	2.625	4.5	2.5857	3.5	3.3	2.3
std. dev.	0.1414	0.5018	0.695	0.5605	0.45	0.674	0.3338	0	0	0
aralik										
mean	2.9	4.625	4.7667	3.9364	4.05	4.96	2.4429	3.7	2.9	3.6
std. dev.	0.2828	0.8903	0.918	0.6071	0.4509	0.8007	1.1238	0	0	0

Time taken to build model (full training data) : 0.03 seconds

=== Model and evaluation on training set ===

Clustered Instances

0	2 (3%)
1	17 (27%)
2	6 (9%)
3	10 (16%)
4	4 (6%)
5	14 (22%)
6	8 (13%)
7	1 (2%)
8	1 (2%)
9	1 (2%)

Log likelihood: -4.47007

EK C

C.1 : Türkiye İlleri Ortalama Sıcaklık Verileri EM Analizi

=== Run information ===

Scheme: weka.clusterers.EM -I 100 -N 10 -X 10 -max -1 -ll-cv 1.0E-6 -ll-iter 1.0E-6 -M 1.0E-6 -K 10 -num-slots 1 -S 100

Relation: trortscklk

Instances: 64

Attributes: 49

yil, adana, afyon, ankara, antalya, artvin, balikesir, bandirma, bilecik, bolu, burdur, bursa, canakkalediyarbakir, edirne, elazig, erzincan, erzurum, eskisehir, gaziantep, giresun, gumushane, hatay, isparta, istanbul, izmir, kars, kastamonu, kayseri, kocaeli, konya, kutahya, malatya, manisa, mardin, mersin, mugla, ordu, rize, samsun, siirt, sinop, sivas, tekirdag, trabzon, urfa, usak, van, zonguldak

Test mode: evaluate on training data

=== Clustering model (full training set) ===

== EM ==

Number of clusters: 10

Number of iterations performed: 3

	Cluster									
Attribute	0	1	2	3	4	5	6	7	8	9
	(0.17)	(0.12)	(0.09)	(0.05)	(0.12)	(0.11)	(0.12)	(0.05)	(0.08)	(0.08)
yil										
mean	1977.0906	1949.6242	1975.6667	1969.6667	1970.5	1957.4286	1977.75	2000	1942.8	1987.6
std. dev.	10.5525	9.3927	16.09	17.2498	17.3997	9.5447	8.6277	0.8165	2.7857	5.9867
adana										
mean	19.3364	19.125	20.2667	18.2	19.6625	18.4571	18.9875	19.4	18.78	18.86
std. dev.	0.0979	0.2107	0.2427	0.216	0.2118	0.3245	0.2204	0.3559	0.2638	0.1855
afyon										
mean	14.1636	13.9875	15.0333	13.7	14.425	13.0571	13.875	15.1667	12.98	13.82
std. dev.	0.2057	0.1691	0.335	0.216	0.2222	0.1294	0.2385	0.4784	0.2135	0.172
ankara										
mean	10.7455	10.7	11.7667	9.4667	11.125	9.6143	10.3	12.1	9.94	10.4
std. dev.	0.2189	0.3674	0.2055	0.1247	0.2385	0.4257	0.1936	0.7257	0.1497	0.1897
antalya										
mean	8.7273	8.7	9.7833	7.5	9.2	7.4857	8.2875	12.6667	7.82	8.3
std. dev.	0.2339	0.2916	0.2339	0.0816	0.1936	0.2587	0.1763	0.793	0.449	0.1673
artvin										
mean	14.7727	14.725	15.6	14	15.1125	14.1143	14.475	19.2	14.28	14.34
std. dev.	0.1543	0.2107	0.2708	0.4546	0.1452	0.2231	0.2332	0.216	0.1939	0.2059

balikesir										
mean	3.8546	3.8875	5	2.7333	4.425	2.7143	3.325	12.4667	3.56	3.94
std. dev.	0.1373	0.314	0.7767	0.17	0.2727	0.2799	0.4294	0.3682	0.463	0.3878
bandirma										
mean	12.9636	12.8125	13.75	12.5	13.2375	11.9	12.6625	15.1	12.08	12.6
std. dev.	0.1666	0.2848	0.1893	0.216	0.1654	0.2726	0.1576	0.3559	0.1327	0.0632
bilecik										
mean	14.1091	13.9125	14.85	13.6667	14.3	13.0143	13.7875	0	13.1	13.72
std. dev.	0.1621	0.318	0.25	0.17	0.1871	0.2642	0.1763	3.0073	0.1673	0.16
bolu										
mean	11.7091	11.5375	12.5833	10.9	11.95	10.5714	11.3625	13.3	10.66	11.36
std. dev.	0.1379	0.1728	0.2115	0.1633	0.2	0.2312	0.1576	0.4967	0.1855	0.1625
burdur										
mean	8.7364	8.575	9.5333	7.8667	9.0125	7.5714	8.375	11.2667	7.6	8.38
std. dev.	0.1494	0.1785	0.2211	0.1247	0.2421	0.205	0.2046	0.5558	0.2966	0.1166
bursa										
mean	11.5455	11.5625	12.4833	10.6333	11.8125	10.7571	11.125	13.6	10.9	11.12
std. dev.	0.1827	0.287	0.2544	0.4989	0.4106	0.3017	0.2107	0.5888	0.1549	0.2482
canakkale										
mean	11.8454	11.5875	12.6333	11.2	12.025	10.6429	11.5375	15.4	10.82	11.5
std. dev.	0.1777	0.2421	0.2134	0.1414	0.1984	0.3332	0.2058	0.5715	0.172	0.2098
diyarbakir										
mean	13.5818	12.1749	14.2333	13.4667	13.8375	12.8	13.275	16	12.46	13.02
std. dev.	0.2124	3.3474	0.2981	0.3859	0.3389	0.1512	0.1299	0.3559	0.2059	0.098
edirne										
mean	15.4727	14.5	16.3	13.3667	15.6125	14.3	14.725	16.2333	14.22	15.12
std. dev.	0.4245	0.4899	0.4082	0.6182	0.4885	0.3423	0.4323	0.1886	0.3487	0.2315
elazig										
mean	13.2091	12.8875	13.75	13.1333	13.35	12.3571	12.95	14.5333	11.78	12.64
std. dev.	0.2937	0.454	0.4031	0.4028	0.2958	0.1917	0.2	0.0471	0.3544	0.196
erzincan										
mean	13.3182	12.5	14.55	11.2333	13.675	11.9714	12.575	13.3667	12.4	13.22
std. dev.	0.3713	0.2062	0.2217	0.33	0.3832	0.205	0.2634	0.4714	0.3347	0.2857
erzurum										
mean	7.0182	6.4625	8.3833	4.9333	7.5	5.5429	6.275	12.1333	6.28	7.12
std. dev.	0.304	0.3603	0.3484	0.2494	0.5074	0.3698	0.1984	0.6236	0.2926	0.3311
eskisehir										
mean	4.5364	4.225	5.25	2.8333	4.9875	3.2286	3.7375	5.8333	4.12	4.54
std. dev.	0.2932	0.4054	0.6344	0.5249	0.3516	0.4742	0.3871	0.4643	0.6882	0.4271
gaziantep										
mean	10.4454	10.3	11.3167	9.5667	10.7125	9.3	10.1125	11.2333	9.42	10.1
std. dev.	0.1671	0.15	0.1675	0.1247	0.2027	0.2563	0.1536	0.7409	0.2482	0.1414
giresun										
mean	16.1091	15.275	16.9	14.6333	16.2	14.8714	15.525	16.1333	15.14	15.8
std. dev.	0.3777	0.2165	0.3559	0.1247	0.3317	0.3769	0.3192	0.2867	0.32	0.3899
gumushane										
mean	8.7546	9.1125	10.05	8.0333	9.4125	8	8.4625	15.2667	8.58	8.8

std. dev.	0.1971	0.1833	0.3594	0.17	0.1615	0.1309	0.2643	0.4028	0.325	0.4099
hatay										
mean	8.6818	8.8637	9.9167	7.7333	9.3125	7.7571	8.3125	9.9	8.38	8.7
std. dev.	0.2167	0.2407	0.4017	0.17	0.1965	0.1678	0.2421	0.6683	0.325	0.429
isparta										
mean	17.3545	15.7624	18.2167	16.2667	17.6875	16.4571	16.975	18.8333	16.76	17
std. dev.	0.2311	3.2419	0.3891	0.2625	0.2088	0.2718	0.2727	0.3091	0.2498	0.2
istanbul										
mean	11.3818	11.3625	12.4	10.2667	11.6375	10.4143	10.95	13.0333	10.68	11
std. dev.	0.1946	0.308	0.2449	0.2867	0.3871	0.344	0.2121	0.6018	0.16	0.2366
izmir										
mean	13.9	13.7125	14.6833	13.6333	14.0875	12.9143	13.6	15.3333	12.72	13.48
std. dev.	0.2089	0.2522	0.3532	0.2055	0.1965	0.1884	0.2693	0.3091	0.2227	0.2315
kars										
mean	15.8364	15.8	16.5667	15.7	16.1875	15.1286	15.5125	18.8333	15.1	15.42
std. dev.	0.2227	0.324	0.2055	0.4967	0.2315	0.1906	0.1269	0.3091	0.1897	0.1327
kastamonu										
mean	3.8091	3.475	4.8333	2.3333	4.3625	2.4143	2.7125	5.8	3.58	3.74
std. dev.	0.5299	0.2905	0.7087	0.2494	0.3806	0.327	0.3723	0.4967	0.3868	0.3929
kayseri										
mean	9.4182	9.6	10.5667	8.8333	9.975	8.3429	9.2	10.1667	8.78	9.14
std. dev.	0.2037	0.3202	0.3197	0.1247	0.2332	0.1761	0.3	0.66	0.2713	0.2871
kocaeli										
mean	8.8909	8.8125	10.1667	7.4333	9.3875	7.7857	8.2625	11.4333	8.46	8.44
std. dev.	0.2778	0.1833	0.3399	0.2625	0.2666	0.3523	0.287	0.9809	0.3441	0.1356
konya										
mean	14.0545	13.9	14.9833	13.7	14.3125	13.0143	13.7875	15.5333	12.9	13.7
std. dev.	0.2016	0.1871	0.367	0.216	0.1965	0.1245	0.2571	0.3859	0.2098	0.2
kutahya										
mean	10.9455	10.9875	12.1167	9.6667	11.3625	9.7857	10.45	12.1333	10.34	10.64
std. dev.	0.2463	0.1833	0.1067	0.3682	0.2736	0.398	0.2398	0.9463	0.28	0.2417
malatya										
mean	10.4182	10.2625	11.3167	9.4333	10.6875	9.2714	10.0125	11.3	9.44	10.04
std. dev.	0.185	0.2233	0.1572	0.1247	0.2421	0.2914	0.2027	0.6377	0.1855	0.1356
manisa										
mean	11.5091	10.7	12.65	9.4333	11.8125	10.1143	10.725	14.4	10.66	11.3
std. dev.	0.3423	0.2449	0.1708	0.2867	0.337	0.2799	0.2861	0.6683	0.3382	0.3347
mardin										
mean	15.3091	15.2625	16.0833	15.0667	15.65	14.5143	15.0125	17.7667	14.6	14.9
std. dev.	0.1975	0.308	0.1951	0.4497	0.2	0.2531	0.1452	0.4028	0.1673	0.1095
mersin										
mean	16.0363	14.975	16.7667	14.0667	16.025	14.8286	15.3	17	14.74	15.64
std. dev.	0.2603	0.5426	0.359	0.6128	0.5068	0.3149	0.4743	0.3742	0.372	0.196
mugla										
mean	12.7273	12.5875	13.6333	11.5667	13.0625	10.4	12.3	20.2333	12.14	12.3
std. dev.	0.1355	0.226	0.1795	0.2867	0.2118	3.4847	0.2236	0.4497	0.2577	0.2608
ordu										

mean	14.0636	14.0375	14.7333	13.5667	14.3375	13.5143	13.6875	15.8333	13.46	13.48
std. dev.	0.1823	0.2913	0.2494	0.4922	0.269	0.1959	0.1452	0.3771	0.196	0.1939
rize										
mean	13.7909	14.3625	15.0167	13.6667	14.4875	13.2286	13.6375	15.0333	13.76	13.68
std. dev.	0.2712	0.2446	0.4017	0.5312	0.2522	0.1278	0.3533	0.3859	0.32	0.3763
samsun										
mean	13.7273	14.0625	14.8833	13.1	14.3875	12.8857	13.55	15	13.46	13.82
std. dev.	0.2863	0.2288	0.5113	0.216	0.2204	0.2295	0.3571	0.3742	0.3007	0.3868
siirt										
mean	13.7182	14.325	14.8333	13.8	14.4	13.2	13.5875	15.1	13.78	13.42
std. dev.	0.2757	0.3345	0.3636	0.6976	0.3808	0.1927	0.3689	0.432	0.2786	0.3311
sinop										
mean	9.5909	8.6125	10.4667	7.8	9.8	8.2429	8.9125	17.0667	8.66	9.48
std. dev.	0.2392	0.3444	0.4269	0.3742	0.3742	0.5315	0.348	0.3091	0.463	0.2786
sivas										
mean	13.1818	13.55	14.25	13.0333	13.775	12.3714	13.025	14.8667	10.86	12.8
std. dev.	0.3298	0.364	0.35	0.3859	0.2487	0.1385	0.3307	0.3399	4.1345	0.3098
tekirdag										
mean	7.6727	7.8	8.9167	6.2	8.2	6.5571	6.85	9.9667	7.5	7.4
std. dev.	0.2562	0.2	0.3131	0.0816	0.2784	0.2382	0.1936	0.9286	0.4382	0.2683
trabzon										
mean	13.2	12.9375	13.8	13.0667	13.375	12.3	12.875	14.7333	11.9	12.64
std. dev.	0.2216	0.3533	0.3697	0.3091	0.2332	0.1852	0.1984	0.2494	0.2608	0.2059
urfa										
mean	15.2273	15.6375	16.4167	14.8	15.8625	14.5286	15.1	15.4	15	15.2
std. dev.	0.3048	0.2546	0.4017	0.216	0.1495	0.1385	0.3	0.3742	0.3286	0.4561
usak										
mean	17.5818	16.6375	18.3333	15.9667	17.7125	16.4	16.9875	19.1	16.46	17.2
std. dev.	0.2657	0.4091	0.2867	0.2625	0.3756	0.2268	0.3789	0.2449	0.3611	0.3847
van										
mean	11.4545	11.3125	12.3167	10.5	11.7375	10.4143	11.05	13.1333	10.64	11.04
std. dev.	0.1924	0.2666	0.2034	0.1633	0.1409	0.3314	0.1936	0.665	0.12	0.1744
zonguldak										
mean	9.0454	8.2125	10	7.4	9.5125	7.6571	8.2	10.6	8.48	9.24
std. dev.	0.3056	0.3756	0.6583	0.0816	0.6071	0.4655	0.2915	0.216	0.3311	0.2871

Time taken to build model (full training data) : 0.07 seconds

=== Model and evaluation on training set ===

Clustered Instances

0	11 (17%)	5	7 (11%)
1	8 (13%)	6	8 (13%)
2	6 (9%)	7	3 (5%)
3	3 (5%)	8	5 (8%)
4	8 (13%)	9	5 (8%)

Log likelihood: -10.08073

EK D

D.1. K-Means MATLAB Kodu

(URL9)

```
% k-means implementation code
clear; clc; close all;
% verilerin excel dosyasından okunması
filename = 'C:\Users\Faruk\Desktop\TROSV.xlsx'; %MMOS-ORS.xlsx
Range = 'B2:M65';
Sheet1 = 'Sheet1';
x= xlsread(filename,Sheet1,Range);
%centroid sayısı
k = 8;
%iteration - tekrar sayısı
max_it = 20;
% initialization step
low = 1;
high = size(x,1);
centros = zeros(k,size(x,2));
indice = zeros(1,k);
% İlk centroid kurulması
for var = 1:k
indice(var) = round(low + (high - low)*rand);
centros(var,:) = x(indice(var),:);
end
% döngü kurulumu
classe = zeros(high,1);
var_cond = 1;
W = 1;
while W
% Veriler ve centroidler arası mesafenin öklid le hesaplanması
dist = zeros(high,k);
```

```
for var = 1:high
for c = 1:k
soma = 0;
for s = 1:size(x,2)
soma = soma + (x(var,s) - centros(c,s)).^2;
end
dist(var,c) = sqrt(soma);
end
end
% matris düzenlenmesi
ant_classe = classe;
% sınıflara değer atama
for var = 1:high
 [~,indice] = min(dist(var,:));
classe(var) = indice;
end
% ilk sonuçları görselleştirme
colors = rand(k,3);
clf;
plota(x,classe,colors);
pause(1);
plotaCentroide(centros);
plota_linhas(x,classe,centros);
pause(1);
% matrisin düzenlenmesi
nov_classe = classe;
% yeni centrodilerin hesaplanması
for c = 1:k
y = x(classe == c,:);
centros(c,:) = sum(y)/size(y,1);
end
% Döngü parametresi değitirilmesi
if ant_classe == nov_classe
```

```
W = 0;
end
% Kontrol
if var_cond > max_it
W = 0;
end
% Döngü şartı
var_cond = var_cond + 1;
end
% Final-Sonuçlandırma
classe = nov_classe;
num_it = var_cond;
% görsel sonuçlar için random renk alma ve çizim
colors = rand(k,3);
figure(1);
hold on;
for i = 1: length(x)
p = plot(x(i,1), x(i,2),'mx','LineWidth',3');
set(p, 'color', colors(classe(i),:))
end
% Kmeans Centroid Çizim kodları:
function plotaCentroide(c)
plot(c(:,1),c(:,2),'mx','LineWidth',2,...
'MarkerEdgeColor','r',...
'MarkerFaceColor','r',...
'MarkerSize',10);
end
% Centroidler ile verileri arasında çizgi çizimi
% matris verilerine göre noktalar arası çizgi çizme
function plota_linhas(x,classe,centros)
n = length(classe);
figure(1);
hold on;
```

```
for i = 1 : n
pos = classe(i);
plot([x(i,1),centros(pos,1)], [x(i,2), centros(pos,2)],'g--');
pause(0.05);
end
end
% Matrislerdeki verilerin ekrana çizimi
function [  ] = plota( x,classes,colors )
%matrislerin renkli çizimi fonksiyonu
n = length(classes);
%figure(1);
hold on;
for i = 1 : n
p = plot(x(i,1), x(i,2),'mx','LineWidth',3);
set(p, 'color', colors(classes(i),:))
end
end
% Centroid değerlerinin hesaplanması
% Verilerin matris değerlerinde centroidlerin hesaplanması
function centros = centroide(x,classes)
k = ver_classes(classes);
centros = zeros(k,size(x,2));
for c = 1:k
y = x(classes == c,:);
centros(c,:) = mean(y);
end
end
```

EK E

E.1. Hiyerarşik Kümeleme Algoritması MATLAB Kodu

```
% Hiyerarşik Kümeleme Başlangıç Programı:
% Hiyerarşik Kümeleme Algoritması
close all;
clear all;
% Excel dosyalarından verilerin okunması
filename = 'C:\Users\Mustafa\Desktop\MMOS.xlsx';
Range = 'B2:AX65';
Sheet1 = 'Sheet1';
X= xlsread(filename,Sheet1,Range);
% Sınıflandırma yapılmadan önce ilk durumun çizimi
cmap = hsv(size(X, 2)); % Renk haritası oluşturma
for i = 1:size(X, 2)
plot(X(1,i),X(2,i),'s','MarkerEdgeColor',cmap(i,:),'MarkerFaceColor',cmap(i,:))
hold on
end
% Do verilen X matrisine HC algoritması uygulanması
k = 7; % Küme sayısı veriyoruz. Bu sayıya ulaşıldığı taktirde sistem duracak
Z = linkage(X,'average');
dendrogram(Z)
hold on;
Z1 = linkage(X,'ward');
dendrogram(Z1)
hold on;
Z3 = linkage(X,'complete');
dendrogram(Z3)
hold on;
Z5 = linkage(X,'single');
dendrogram(Z5)
hold on;
X = hcluster(X, k);
```

```
% Kümeleme yapıldıktan sonra verilerin son halinin gösterimi
figure;
for i = 1:size(X, 2)
g = X(end, i);
plot(X(1,i),X(2,i),'s','MarkerEdgeColor',cmap(g,:),'MarkerFaceColor',cmap(g,:))
hold on
end
% Grupları güncelleyen fonksiyon:
function [X] = updateGroups(X, newG, oldG)
% Vektör halindeki verilerin eskilerinin yenileri ile güncellenmesi
for i = 1:size(X, 2)
if X(end, i) == oldG
X(end, i) = newG;
end
end
end
%Mean değerlerini hesaplayan fonksiyon
function [mean] = meanVec(X, g)
% X matrisindeki grupların Mean değerlerinin hesaplanması
ind = find(X(end,:) == g);
vectorSum = sum(X(1:end-1, ind), 2);
counts = size(ind, 2);
mean = vectorSum ./ counts;
end
% Hiyerarşik kümeleme kodu
function [X] = hcluster(X, k)
% verilen X veri kümesinde hiyerarşik kümelemenin yapılması
% Vikipedi algoritmanın adımlarının oluşturulmasında faydalanılan
% platformdur.
% Add class information to data matrix X:
n = size(X, 2);
X = [X; 1:n];
means = X(1:end-1, :);
```

```
numGroups = n;
% gruplar arası mesafe distM'e tanımlanır
distM = Inf(n, n);
while numGroups > k
for i = 1:n
for j = i+1:n
distM(i, j) = d(means(:, i), means(:, j));
end
end
% en düşük mesafeli öğe belirlenir.
 [~, ind] = min(distM(:));
 [newG, oldG] = ind2sub(size(distM), ind);
% Gryp bilgileri güncellenir:
X = updateGroups(X, newG, oldG);
% Yeni mean değerlerinin hesaplanması:
for i = 1:n
if i ~= oldG
means(:, i) = meanVec(X, i);
else
means(:, i) = Inf(size(X, 1)-1, 1);
end
end
numGroups = numGroups - 1;
end
end
% iki nokta arası öklid hesaplama fonksiyonu
function [dist] = d(x, y)
% x ve y iki noktalar arası öklid mesafesinin hesaplanması
dist = sum((x - y).^2);
end
```

EK F

F.1.

(URL10)

```
% Ana program
clear;
%TROSV MMOS ORS verilerinin seçilimi
prompt = 'Please enter 1 for dataset1 for MMOS, 2 for dataset2 for ORS, 3 for dataset3 for TROSV ->>> ';
fileNumber = sscanf(input(prompt, 's'), '%d');
%Random veya Kmeans ile EM training seçilimi
prompt = 'Select 1 for random intialization or 2 for KMEAN intialization for EM ->>> ';
intailization_strategy =  sscanf(input(prompt, 's'), '%d');
% r tekrar değeri seçilimi
prompt = 'Select r value ->>> ';
r =  sscanf(input(prompt, 's'), '%d');
%k centroid sayısı seçilimi
prompt = 'Select k value ->>> ';
k =  sscanf(input(prompt, 's'), '%d');
file_Name1 = 'dataset1.txt';
file_Name2 = 'dataset2.txt';
file_Name3 = 'dataset3.txt';
file_Name_test = 'dataset_test.txt';
if fileNumber == 1
file_used = file_Name1;
elseif  fileNumber == 2
file_used = file_Name2;
else
file_used = file_Name3;
end
% Eğitim verisinin txt uzantılı dosyadan okunması fonksiyonu
```

```
x = readTrainingData(file_used);
sizeX = size(x,1);
membership=0;
sses = []
vect = 0;
for i = 1:r
% başlangıç kurulumu
 [means_init,converiances_init] =
intialization_step(intailization_strategy,x,k,i,file_used);
% Gaussian Mixture Learning kullanılarak Expectation ve mazimization adımlarının
geliştirilmesi
 [means,converiances,P,log_p_self,liklihood] =
GaussianMixtureLearning(x,means_init,converiances_init,k,i);
means_init_tracker{i} = means_init;
means_tracker{i} = means;
converiances_tracker{i} = converiances;
log_p_self_tracker{i}=log_p_self;
vect(i) = log_p_self(end)
P_tracker{i} = P;
% Verilerin birbirlerine olan tahmini benzerlik veya yakınlıklarının hesaplanması
end
 [maxLog,idx] = max(vect);
best_P =  P_tracker{idx};
disp('intialization means are as follows: (each row is a correspondance mean for a
cluster)')
best_means_init = means_init_tracker{idx}
disp('Intial converiances are as follows: ')
best_converiances = converiances_tracker{idx};
converiances_init =best_converiances{1};
for i=1:k
converiances_init{i}
end
best_log_p_self =  log_p_self_tracker{idx};
```

```
disp('Last updated means are as follows: (each row is a correspondance mean for a
cluster)')
best_means = means_tracker{idx}
converiances =best_converiances{end};
disp('Last updated converiances are as follows: ')
for i=1:k
converiances{i}
end
membership(1:sizeX) = 0;
for j=1:sizeX
maxVal = max(best_P(j,:));
membership(j) = find(best_P(j,:) == maxVal);
end
membership = membership';
% sonuçların çizilmesi
doPlot_EM(x,membership,best_means_init,best_means,best_converiances{1},best_conv
eriances{end},r,size(best_log_p_self,2),k);
figure(3);
plot(1:size(best_log_p_self,2),best_log_p_self);
% ReadTraningData fonksiyonu
function x = readTrainingData(fileName_Training)
disp('Reading training data')
fileID_Training = fopen(fileName_Training);
x = textscan(fileID_Training,'%f %f');
fclose(fileID_Training);
% bu komut ile matris değerlerini array'a çeviriyoruz.
x= cell2mat(x);
end
%Inıtıalization fonksiyonu
function [means,converiances] = intialization_step(intailization_strategy,x,k,i,file_used)
display('Instailization');
file_Name1 = 'dataset1.txt';
file_Name2 = 'dataset2.txt';
```

```
file_Name_test = 'dataset_test.txt';
sizeX = size(x,1);
if(intailization_strategy == 1)
%centoridlerin verilen verilerden rastgele seçilimi
centeroid_idx = datasample(1:sizeX,k,'Replace',false);
centeroid_idx = sort(centeroid_idx);
means = x(centeroid_idx,:);
overall_convariance = cov(x);
for j = 1:k
converiances{j} = overall_convariance/k;
end
elseif(intailization_strategy == 2)
centeroid_idx = datasample(1:sizeX,k,'Replace',false);
centeroid_idx_keeper{i} = centeroid_idx;
disp('final membership: ')
centeroid_idx = sort(centeroid_idx);
% Kmeans kullanılarak centroid'lerin belirlenmesi
 [finalmembership,finalcentroids,numOfIterations] =
KMEANS_Part1(x,centeroid_idx,i);
means = finalcentroids;
for j = 1:k
converiances{j} = cov(x(finalmembership==j,:));
end
end
end
```

F.2

(URL10)

```
% Kmeans_Part1 fonksiyonu
% fonksiyon girdileri:
% x = veri bloğu
% r = tekrar sayısı
% Çıktı : benzerlik gösteren membership değeri
function [membership,centroids,count,sse_self] =
KMEANS_Part1(x,intialCentroid_idx,r)
sizeX = size(x,1);
numOfClunters = size(intialCentroid_idx,2);
count = 0;
centroids = x(intialCentroid_idx,:);
sse_self = [];
membership0 = returnMemberShip(x,centroids);
keepLooping = true;
while keepLooping
count=count+1;
r
i=count
disp('----');
sse = computeSSE(x,centroids);
sse_self = [sse_self,sse];
centroids = update_centeroids(x,membership0);
membership = returnMemberShip(x,centroids);
if(isequal(membership0,membership))
keepLooping = false;
end
membership0 =  membership; end end
%computeSSE fonksiyonu:
% Noktalar arası mesafenin ölçülmesi
```

```
function sse = computeSSE(x,centroids)
numOfClusters = size(centroids,1);
sizeX = size(x,1);
theMin = realmax;
sse=0;
for j = 1:sizeX
theMin = realmax;
for i=1:numOfClusters
theMin = min(theMin,euclidean_distance(centroids(i,:),x(j,:))^2);
end
sse= sse + theMin;
end
end
%update_centroids fonksiyonu
function centroids = update_centeroids(x,membership)
% cluster numalarının bulunası ‘membership’ değerine %bakılarak
unq = unique(membership);
numOfClusters = max(size(unq));
centroids = [];
% her küme id’sine göre
for i=unq'
% her kümenin yeni mean değerinin hesaplanması
x_i=x(membership==i,:);
%yeni centroid değerlerinin bulunması ile değerlerin değişiminin güncellenmesi
c = mean(x_i);
centroids = [centroids; c];
end
end
% return_membership fonksiyonu
function membership = returnMemberShip(x,centeroid)
sizeX = size(x,1);
numOfClusters = size(centeroid,1);
membership = zeros(sizeX,1);
```

```
for  i = 1 : sizeX
distances =[];
% X noktasından diğer tüm centroidlere olan uzaklığın hesaplanması
for j = 1:numOfClusters
newVal = euclidean_distance(centeroid(j,:),x(i,:));
distances = [distances newVal];
end
% x noktasına en yakın centroid in belirlenmesi
membership(i)=nearestCenteroid(distances);
end
end
%nearestCentroid fonksiyonu
function c = nearestCenteroid(centroids)
c = find(centroids == min(centroids));
end
 % GaussianMixtureLearning fonksiyonu
function   [new_means,new_converiances,new_P,log_p_self,liklihood] =
GaussianMixtureLearning(x,means,converiances,k,r)
keepLooping = true;
MaxIteration = 100;
logliklihoodThreshold = 10e-9;
logliklihoodThresNew = 0;
logliklihoodThresOld = 0;
count = 0;
liklihood(1:k)=1/k;
log_p_self=[];
%%
log_p= ComputeLogLiklihood(x,means,converiances,liklihood);
logliklihoodThresNew = log_p;
log_p_self=[log_p_self log_p];
%%
converiancess = converiances;
meanss = means;
```

```
while keepLooping
display('Keep looping');
r
count = count+1;
i=count
logliklihoodThresOld = logliklihoodThresNew;
P = E_Step(x,liklihood,meanss,converiancess);
meanss_1=meanss;
[meanss,converiancess,liklihood,log_p]=M_Step(P,x,meanss_1);
logliklihoodThresNew = log_p;
log_p_self=[log_p_self log_p];
val = abs((logliklihoodThresNew - logliklihoodThresOld)/logliklihoodThresOld);
if(count >= MaxIteration || val<logliklihoodThreshold)
keepLooping = false;
end
end
new_means = meanss;
new_converiances = {converiances,converiancess};
new_P = P;
end
```

EK G

=== Run information ===
Scheme: weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 10 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 44
Relation: ortruzsid
Instances: 64
Attributes: 13
yil, ocak, subat, mart, nisan, mayis, haziran, temmuz, agustos, eylul, ekim, kasim, aralik
Test mode: evaluate on training data
=== Clustering model (full training set) ===
kMeans
======
Number of iterations: 6
Within cluster sum of squared errors: 8.160678975344503
Initial starting points (random):
Cluster 0: 1998,3.1,3,3.9,3,2.8,2.6,3.1,3.1,2.8,2.8,3.3,4.1
Cluster 1: 1965,4,5.2,3.1,2.9,2.2,3.4,3.8,4.1,3.5,3.5,5.3,4.6
Cluster 2: 1964,4.3,4,2.8,3.4,3.2,3.1,4,3.1,2.7,1.6,2.8,5
Cluster 3: 1995,4.3,3.5,3.9,3.1,3.3,2.8,3.2,2.9,2.6,3.1,3.8,4.1
Cluster 4: 2000,3.5,3.1,2.8,2.2,2.6,3,2.4,3,2.7,2.6,2.1,2.7
Cluster 5: 1990,4.2,4.9,4.7,3.9,4.9,4.2,5.8,5.3,4,4.4,4.6,4.7
Cluster 6: 1978,4.3,4.5,4.6,3.4,3.6,3.2,4.4,4.3,3.8,3.8,3.7,4.5
Cluster 7: 1988,4.2,5,4.8,4.6,3.6,2.6,4.3,4.7,4.1,5.5,5.8,5.2
Cluster 8: 2010,4,3.3,2.9,3,2.4,2.5,2.8,2.4,2.6,2.7,2.8,3.8
Cluster 9: 1957,3.7,3.2,5.1,3.6,2.3,4.1,3.9,4.3,3.3,3.5,3.3,5.7
Missing values globally replaced with mean/mode
Final cluster centroids:

		Cluster#								
Attribute	Full Data	0	1	2	3	4	5	6	7	8	9
	(64.0)	(7.0)	(4.0)	(6.0)	(3.0)	(1.0)	(4.0)	(11.0)	(6.0)	(14.0)	(8.0)
yil	1983.5	1996.8571	1958.75	1962.8333	1989.3333	2009	1989.25	1977.7273	1979.6667	2007.8571	1959.625
ocak	4.1922	2.6143	4.525	4.6167	5.4667	0	5.225	4.9727	4.5333	3.4071	4.6625
subat	4.2859	3.7143	5.175	4.3833	4	0	5	5.3364	4.9833	3.3857	4.1625

mart 3.8547 3.5571 3.825 3.5167 4.3333 0 4.825 4.2455 4.4333 3.1357 4.4875
nisan 3.4703 3.2714 3.4 3.5167 3.5333 0 4.525 3.5091 4.2167 2.8571 3.9875
mayis 3.0469 2.4143 2.9 2.7667 3.3 0 4.8 3.4 3.55 2.6643 3.1
haziran 3.0828 2.3857 3.45 2.9667 2.8333 2.5 4.15 3.2818 3.4167 2.6857 3.4
temmuz 3.5375 3.1714 3.8 3.3833 3.3333 2.9 5.05 3.8182 4.4167 2.4714 4.0625
agustos 3.5531 3.0571 4 2.6667 3.1667 3 4.85 4.0909 4.6 2.6929 3.975
eylul 3.1656 2.6857 3.55 2.5833 2.6333 2.7 4.075 3.6455 4.1 2.4857 3.4625
ekim 3.2922 2.7429 3.35 2.4333 3.2667 2.7 4.1 3.5182 5.1 2.7357 3.375
kasim 3.5609 3.1286 4.95 2.7167 3.2667 2.3 4.55 3.9 5.3667 2.8 3.1625
aralik 4.2125 4.1714 4.275 4.2333 4.1667 3.6 5.075 4.9545 4.8333 2.8214 4.8125

Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

0	7 (11%)	4	1 (2%)	8	14 (22%)
1	4 (6%)	5	4 (6%)	9	8 (13%)
2	6 (9%)	6	11 (17%)		
3	3 (5%)	7	6 (9%)		

EK H

=== Run information ===
Scheme: weka.clusterers.HierarchicalClusterer -N 2 -L WARD -P -A "weka.core.EuclideanDistance -R first-last"
Relation: ortruzsid
Instances: 64
Attributes: 13
yil, ocak, subat, mart, nisan, mayis, haziran, temmuz, agustos, eylul, ekim, kasim, aralik
Test mode: evaluate on training data
=== Clustering model (full training set) ===
Cluster 0
((((((6.2:0.47298,((5.7:0.41151,(5.1:0.3124,5.6:0.3124):0.09911):-0.03216,5.2:0.37935):0.09363):0.04746,((3.6:0.33246,3.8:0.33246):0.1551,(3.7:0.30385,3.3:0.30385):0.18371):0.03288):0.08622,4.3:0.60665):0.33041,(((3.8:0.35192,4.2:0.35192):0.18915,(5.1:0.29102,5.0:0.29102):0.25004):0.11877,(3.4:0.51543,4.4:0.51543):0.14441):0.27722):0.73956,((((4.3:0.37688,(5.1:0.34752,((4.4:0.25384,4.5:0.25384):0.05436,5.4:0.3082):0.03932):0.02936):0.04765,6.5:0.42454):0.08665,(((4.4:0.30606,3.9:0.30606):-0.00188,(((4.4:0.23585,5.1:0.23585):0.07539,5.4:0.31125):-0.00901,5.4:0.30223):0.00195):0.13439,4.1:0.43858):0.07261):0.02726,4.6:0.53844):1.13818):1.66459,((4.0:0.6894,(((5.0:0.37316,(5.0:0.38606,4.6:0.38606):-0.0129):0.01266,(5.2:0.21345,5.0:0.21345):0.17236):0.10019,4.2:0.486):0.2034):0.06228,(4.0:0.61227,((4.7:0.39928,4.2:0.39928):0.11831,7.4:0.51759):0.09468):0.1394):2.58955)
Cluster 1
((3.7:1.14423,((((((3.7:0.21843,4.1:0.21843):-0.00708,4.1:0.21135):0.05437,5.3:0.26572):0.4602,((((4.0:0.28269,(2.7:0.26087,(3.1:0.22448,((2.4:0.25673,2.7:0.25673):-0.00426,2.8:0.25246):-0.02798):0.03639):0.02183):0.03774,(3.8:0.14062,3.4:0.14062):0.17981):0.05743,(4.1:0.37595,(4.3:0.26565,((3.7:0.21435,3.3:0.21435):0.02148,2.9:0.23583):0.02982):0.1103):0.00192):0.10835,0.0:0.48621):0.23971):0.0153,2.9:0.74121):0.10808,(3.1:0.35052,2.7:0.35052):0.49877):0.29493):0.11803,3.6:1.26226)
Time taken to build model (full training data) : 0.01 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 41 (64%)
1 23 (36%)

EK I

=== Run information ===
Scheme: weka.clusterers.HierarchicalClusterer -N 2 -L WARD -P -A "weka.core.EuclideanDistance -R first-last"
Relation: scklk
Instances: 64
Attributes: 4
yil, maxscklk, minscklk, ortscklk
Test mode: evaluate on training data
=== Clustering model (full training set) ===
Cluster 0
(((((14.5:0.09056,14.4:0.09056):0.6694,(((13.8:0.10371,(14.0:0.12027,14.0:0.12027):-0.01656):0.1036,(13.7:0.08488,13.6:0.08488):0.12243):0.36904,((14.3:0.09616,14.3:0.09616):0.05696,14.3:0.15312):0.42323):0.18361):0.00283,((((13.8:0.05376,13.8:0.05376):0.07297,14.0:0.12674):-0.01426,13.8:0.11248):0.21123,(14.4:0.21597,14.2:0.21597):0.10774):0.43908):0.6951,(13.0:0.41016,(13.6:0.33723,((13.1:0.05513,13.1:0.05513):0.19383,(13.3:0.10674,13.5:0.10674):0.14221):0.08827):0.07293):1.04772):2.06798,((((((13.5:0.08478,13.4:0.08478):0.02989,(13.6:0.09602,13.5:0.09602):0.01866):0.0465,13.2:0.16118):0.17421,((13.5:0.07771,(13.5:0.06209,13.6:0.06209):0.01562):0.12915,13.6:0.20686):0.12854):0.43154,(((12.9:0.20102,(((13.2:0.0952,13.2:0.0952):-0.00448,13.3:0.09072):0.04338,13.0:0.1341):0.06691):0.18608,(13.1:0.19952,(((13.3:0.06853,13.3:0.06853):0.06602,13.4:0.13455):0.05033,13.0:0.18488):0.01464):0.18758):0.09161,13.4:0.4787):0.28824):1.66935,((13.8:0.0952,(14.0:0.09045,14.0:0.09045):0.00475):0.48062,((14.2:0.11343,((14.1:0.06222,14.0:0.06222):0.02648,14.2:0.08869):0.02474):0.15477,13.7:0.2682):0.30762):1.86047):1.08957)
Cluster 1
((15.3:0.63808,((14.8:0.19769,(14.4:0.15365,14.5:0.15365):0.04404):0.32719,(((15.0:0.08332,15.1:0.08332):0.0509,15.1:0.13422):0.09591,(15.1:0.0462,15.1:0.0462):0.18393):0.29475):0.1132):0.13689,(((15.1:0.13083,15.1:0.13083):0.09348,14.8:0.22431):0.16011,(14.7:0.29031,14.6:0.29031):0.09411):0.39054)
Time taken to build model (full training data) : 0 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 50 (78%)
1 14 (22%)

EK J

=== Run information ===
Scheme: weka.clusterers.HierarchicalClusterer -N 2 -L COMPLETE -P -A "weka.core.EuclideanDistance -R first-last"
Relation: trortscklk
Instances: 64
Attributes: 49

yil, adana, afyon, ankara, antalya, artvin, balikesir, bandirma, bilecik, bolu, burdur, bursa, canakkale, diyarbakir, edirne, elazig, erzincan, erzurum, eskisehir, gaziantep, giresun, gumushane, hatay, isparta, istanbul, izmir, kars, kastamonu, kayseri, kocaeli, konya, kutahya, malatya, manisa, mardin, mersin, mugla, ordu, rize, samsun, siirt, sinop, sivas, tekirdag, trabzon, urfa, usak, van, zonguldak

Test mode: evaluate on training data
=== Clustering model (full training set) ===
Cluster 0
((((((8.3:0.60743,8.6:0.60743):0.34264,8.9:0.95007):0.63067,8.5:1.58074):0.31038,(((7.8:0.55403,7.5:0.55403):0.3482,(8.0:0.57942,8.5:0.57942):0.32281):0.35129,((9.2:0.55678,9.8:0.55678):0.44633,8.2:1.00311):0.25041):0.63761):0.20464,(((((((8.7:0.54611,(8.7:0.41902,8.7:0.41902):0.1271):0.07336,(8.5:0.42669,8.8:0.42669):0.19278):0.0987,(9.3:0.49123,9.5:0.49123):0.22694):0.2293,(8.4:0.78723,8.9:0.78723):0.16025):0.16083,(((8.3:0.35735,8.5:0.35735):0.28753,9.2:0.64488):0.18889,(9.4:0.53587,9.0:0.53587):0.29791):0.27454):0.07646,(((9.1:0.48382,9.1:0.48382):0.0446,9.1:0.52842):0.19911,((9.7:0.39014,9.6:0.39014):0.1722,9.5:0.56234):0.16518):0.45725):0.22246,((8.1:0.56141,8.9:0.56141):0.19837,((8.2:0.44148,8.2:0.44148):0.17754,8.4:0.61902):0.14076):0.64744):0.68854):0.86538,((((8.7:0.57064,8.2:0.57064):0.0835,8.6:0.65414):0.73744,((((7.7:0.61438,7.8:0.61438):0.29747,(8.1:0.72174,8.1:0.72174):0.19011):0.18346,(6.7:0.76159,(7.3:0.52906,7.9:0.52906):0.23252):0.33372):0.09219,8.0:1.1875):0.20408):0.45126,((7.4:0.90006,(7.5:0.49443,7.5:0.49443):0.40563):0.57093,7.3:1.47099):0.37185):1.1183)
Cluster 1
(((((8.6:0.67195,10.2:0.67195):0.53852,(9.9:0.62132,10.3:0.62132):0.58915):0.10927,10.5:1.31974):0.56131,(((9.5:0.56123,10.5:0.56123):0.09364,9.7:0.65487):0.27869,(9.7:0.62785,10.5:0.62785):0.30571):0.94749):1.479,((10.7:0.58255,10.8:0.58255):1.39198,10.3:1.97453):1.38552)
Time taken to build model (full training data) : 0 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 51 (80%)
1 13 (20%)

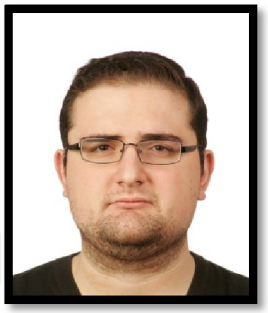
EK K

=== Run information ===
Scheme: weka.clusterers.HierarchicalClusterer -N 2 -L AVERAGE -P -A "weka.core.EuclideanDistance -R first-last"
Relation: trortscklk
Instances: 64
Attributes: 49

yil, adana, afyon, ankara, antalya, artvin, balikesir, bandirma, bilecik, bolu, burdur, bursa, canakkale, diyarbakir, edirne, elazig, erzincan, erzurum, eskisehir, gaziantep, giresun, gumushane, hatay, isparta, istanbul, izmir, kars, kastamonu, kayseri, kocaeli, konya, kutahya, malatya, manisa, mardin, mersin, mugla, ordu, rize, samsun, siirt, sinop, sivas, tekirdag, trabzon, urfa, usak, van, zonguldak

Test mode: evaluate on training data
=== Clustering model (full training set) ===
Cluster 0
((((((8.3:0.60743,8.6:0.60743):0.18861,8.9:0.79605):0.44795,((7.8:0.55403,7.5:0.55403):0.50286,(((((8.0:0.57942,8.5:0.57942):0.1541,((8.7:0.53234,(8.7:0.41902,8.7:0.41902):0.11333):0.0812,((9.3:0.49123,9.5:0.49123):0.07465,(8.5:0.42669,8.8:0.42669):0.13919):0.04765):0.11998):0.05294,8.4:0.78646):0.04682,((((9.1:0.48382,9.1:0.48382):0.02454,9.1:0.50836):0.12103,((9.7:0.39014,9.6:0.39014):0.11202,9.5:0.50216):0.12722):0.16821,8.9:0.79759):0.03569):0.14479,((((8.3:0.35735,8.5:0.35735):0.27851,9.2:0.63586):0.08262,((8.1:0.56141,8.9:0.56141):0.09201,((8.2:0.44148,8.2:0.44148):0.12133,8.4:0.56281):0.09061):0.06506):0.16676,(9.4:0.53587,9.0:0.53587):0.34937):0.09283):0.07883):0.1871):0.3547,((((((8.7:0.57064,8.2:0.57064):0.07597,8.6:0.64661):0.3444,((((7.7:0.61438,7.8:0.61438):0.09785,((7.3:0.52906,7.9:0.52906):0.13531,8.1:0.66438):0.04785):0.10267,8.1:0.8149):0.0946,6.7:0.9095):0.08151):0.08657,8.0:1.07758):0.08491,(7.4:0.8742,(7.5:0.49443,7.5:0.49443):0.37977):0.28829):0.16103,7.3:1.32352):0.27517):0.20449,8.5:1.80319):0.24532,(((((9.2:0.55678,9.8:0.55678):0.32257,8.2:0.87935):0.10422,(((9.5:0.56123,10.5:0.56123):0.06277,9.7:0.62399):0.18448,(9.7:0.62785,10.5:0.62785):0.18062):0.17509):0.17456,((8.6:0.67195,10.2:0.67195):0.31562,(9.9:0.62132,10.3:0.62132):0.36625):0.17056):0.41969,10.5:1.57781):0.47069)
Cluster 1
((10.7:0.58255,10.8:0.58255):1.20146,10.3:1.78401)
Time taken to build model (full training data) : 0.02 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 61 (95%)
1 3 (5%)

ÖZGEÇMİŞ



Ad - Soyad : Mustafa TAKAOĞLU

Doğum Tarihi ve Yeri : 26.06.1990

E-posta : mustafatakaoglu@outlook.com

Tel : 0544 227 22 61

ÖGRENİM DURUMU : Yüksek Lisans (Devam Etmekte)

- **LİSANS** : İstanbul Aydın Üniversitesi - Mühendislik Mimarlık Fakültesi - Bilgisayar Mühendisliği (2009-2014)
- **YÜKSEK LİSANS** : İstanbul Aydın Üniversitesi - Bilgisayar Mühendisliği Ana Bilim Dalı - Bilgisayar Mühendisliği Programı

MESLEKİ DENEYİMLER

Tarih : 2012-2013

Meslek veya Pozisyonu : Stajyer Öğrenci

İşyeri Adı : TAKAOĞLU GIDA, EMLAK, PAZARLAMA, ULUSLARARASI TİCARET (İstanbul / Türkiye)

Tarih **:** 2013-2014

Meslek veya Pozisyonu **:** Stajyer Öğrenci

İşyeri Adı **:** TAKAOĞLU GIDA, EMLAK, PAZARLAMA, ULUSLARARASI TİCARET (İstanbul / Türkiye)

ÖDÜLLER

Özel İhlas Koleji Okul Birinciliği (2009)
İstanbul Aydın Üniversitesi İngilizce Hazırlık Okulu 2009-2010 Eğitim Yılı Bilgisayar Mühendisliği Bölüm Birinciliği
İstanbul Aydın Üniversitesi 2010-2011 Eğitim Yılı Bilgisayar Mühendisliği Bölüm İkinciliği ve Onur Sertifikası
İstanbul Aydın Üniversitesi 2011-2012 Eğitim Yılı Bilgisayar Mühendisliği Bölüm İkinciliği
İstanbul Aydın Üniversitesi 2012-2013 Eğitim Yılı Bilgisayar Mühendisliği Bölüm İkinciliği ve Yüksek Onur Sertifikası
İstanbul Aydın Üniversitesi 2013-2014 Eğitim Yılı Bilgisayar Mühendisliği Bölüm İkinciliği