T.C.
BEYKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

# VERİ MADENCİLİĞİNDE KATEGORİSEL DEĞİŞKENLER İÇİN VERİ KÜMELEMEDE AĞAÇ YÖNTEMİ İLE YENİ BİR ALGORİTMA UYGULAMASI

YÜKSEK LİSANS TEZİ

BURAK ÇAKIR

**Enstitü Ana Bilim Dalı: Matematik - Bilgisayar**

**Tez Danışmanı: Yrd. Doç. Dr. Gökhan SİLAHTAROĞLU**

EYLÜL, 2008
İSTANBUL

**T.C.**
**BEYKENT ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRLÜĞÜ**
**TEZLİ YÜKSEK LİSANS TEZ SINAV TUTANAĞI**

21./.11./2008

Enstitümüz Matematik Bilgisayar Anabilim dalı Bilgi Teknolojileri Programı yüksek lisans öğrencilerinden 060862003 numaralı ***Burak ÇAKIR'a*** *"Beykent Üniversitesi Lisansüstü Eğitim - Öğretim Yönetmeliği'*nin ilgili maddesine göre hazırlayarak, Enstitümüze teslim ettiği ***"VERİ MADENCİLİĞİNDE KATEGORİSEL DEĞİŞKENLER İÇİN VERİ KÜMELEMEDE AĞAÇ YÖNTEMİ İLE YENİ BİR ALGORİTMA UYGULAMASI"*** *tezini*, Yönetim Kurulumuzun 13.10.2008 tarih ve 2008/17 sayılı toplantısında seçilen ve Fakülte binasında toplanan jüri üyeleri huzurunda, ilgili yönetmeliğin *(c) bendi gereğince* aday tarafından savunulmuş ve sonuçta adayın tezi hakkında ***oybirliği*** ile ***Kabul*** kararı verilmiştir.

İşbu tutanak, 4 nüsha olarak hazırlanmış ve Enstitü Müdürlüğü'ne sunulmak üzere tarafımızdan düzenlenmiştir.

DANIŞMAN
YRD. DOÇ. DR. GÖKHAN SİLAHTAROĞLU

ÜYE
YRD. DOÇ. DR. TURHAN KARAGÜLER

ÜYE
PROF. DR. ESAT HAMZAOĞLU

# ÖNSÖZ

Veri madenciliği, ülkemizde henüz yeterli kaynak bulunmayan, eksikliği hissedilen bir alandır. Kurumların elinde çok ciddi veritabanları oluşmaya başlamıştır. Hemen her disiplin tarafından çeşitli amaçlar için kullanılan veri madenciliği alanındaki gelişmeler, akademisyen ve araştırmacı bilim çevreleri tarafından, uluslararası bilimsel dergiler ve kongreler aracılığıyla takip edilmektedir.

Kümeleme ve ağaç yöntemleri, veri madenciliği alanında sık kullanılan yöntemlerdendir. Bu alanda sürekli yeni gelişmeler olmaktadır. Bu çalışmada, veri madenciliği alanında kullanılmak üzere kategorisel değişkenlere sahip veri tabanlarında ağaç yöntemiyle veri kümeleme için yeni bir algoritmanın kullanılabilirliğinin gösterilmesi için hazırlanmıştır.

Bu tezin, veri madenciliği konusu ile ilgilenenlere yardımcı olmasını ve farklı bir bakış açısı göstermede yararlı olmasını dilerim.

Tezin hazırlanmasında ve algoritmanın oluşturulması aşamasında benden yardımlarını esirgemeyen; veri madenciliği konusunda ülkemizde büyük bir boşluğu dolduran değerli danışman hocam Yrd. Doç. Dr. Gökhan SİLAHTAROĞLU'na teşekkür ederim. Tezin yazım aşamasında bana sağladığı katkılardan ve gösterdiği hoşgörüden dolayı eşim Aynur ÇAKIR'a teşekkür ederim.

# ÖZET

Veri madenciliğinde veri farklı şekiller ve gürültü içerdiği zaman büyük popülasyonları ayıklama önemli bir problemdir. İyi bir algoritma mekanızması veya metodu kümeleri bulmak açısından etkili olmalıdır. Ayrıca boyut büyüdükçe uzayın karmaşıklığı ve zaman karmaşıklığı önemli hale gelmeye başlar.

Bu tez çalışmasında, veri madenciliği alanında kullanılmak üzere kategorisel değişkenlere sahip veri tabanlarında ağaç yöntemiyle veri kümeleme için yeni bir algoritmanın kullanılabilirliğinin gösterilmesi gerçekleştirilmiştir. Ağaç yöntemi kullanarak simgeler sahip oldukları değerlere ayrılmıştır. Bu değerler ve bölümler sayesinde bir veri tabanını mümkün olan en az parçadan en çok parçaya doğru sıralayarak bir ağaç oluşturulması gerçekleştirilmiştir.

Çalışmanın birinci bölümünde veri madenciliği, kullanıldığı alanlar ve veri madenciliğinin gelişimi konularına değinilmiştir. İkinci bölümünde veri madenciliğinde kullanılan yöntemler, algoritmalar ve bu tezin konusu olan algoritmaya yakın olanları konu edilmiştir.

Tez çalışmasının üçüncü bölümünde ise söz konusu olan algoritma ve uygulanması gösterilmiştir. Kategorisel verilerden oluşan bir veri tabanı ile gerçekleştirilen sonuçlar ele alınmış, aynı kategoride olan diğer algoritmalarla karşılaştırılması gerçekleştirilmiştir. Sentetik bir veri tabanı ile elde edilen sonuçlar gösterilmiştir.

Dördüncü bölümde ise, elde edilen bilgiler doğrultusuna, sonuçlar incelenmiş ve algoritmanın uygulanabilirliği hakkında yorumlar yapılmıştır.

Anahtar Kelimeler: Veri madenciliği, kümeleme, ağaç, ayrım, budama, entropi, gini

# ABSTRACT

The major reason that data mining became one of the hottest current technologies of the information age is the wide availability of huge amounts of data and the need for turning such data into useful information and knowledge. As computer systems getting cheaper and computer power increases, the amount of data available to be collected and processed increases. Therefore using techniques that operates very well with large amounts of data becomes an obvious choice. The information and knowledge gained can be used for applications ranging from business management, production control, and market analysis, to engineering design and science exploration.

In this study, a new data mining algorithm used and tested for categorical variable. This algorithm improved by Yrd. Doç. Dr. Gökhan SİLAHTAROĞLU. This algorithm to call "A Tree Approach to Clustering Data with Categorical Variables". In the literature there are different approaches to form tree. To determine the best attribute, used an equal-split parameter. After forming the clusters, used another clustering algorithm such as PAM, CLARA or K-Means to reduce the number of leaves to the number required by the user.

Keywords: Data mining, clustering, tree, split, pruning, entropy, gini.

# İÇİNDEKİLER

# ŞEKİLLER LİSTESİ

# TABLOLAR LİSTESİ

# 1. BÖLÜM

## 1.1 Veri Madenciliği

Bilgisayarların yaşamımıza daha çok girmesiyle birlikte, artık her yaptığımız işlem sayısal ortamda kayıt altına alınmaya başlandı. Marketlerde yaptığımız alışverişlerde aldığımız her bir ürün, hatta alıp bir süre sonra iade ettiğimiz bir ürün ve o ürünle birlikte aldığımız diğer ürünler bilgisayarlarda, veritabanlarında tutulmaya başlandı. Hastanelerde, belediyelerde veya ticarette yaptığımız her işlem artık anında veritabanlarında yerini alıyor. Hatta bir mağazaya, alışveriş merkezine girerken ya da çıkarken, bazen de yolda yürürken kameraya çekilen görüntülerimiz bile bir veritabanı oluşturuyor. Bütün bunlar bir yığın halinde depolanırken içlerinde kim bilir ne gibi bilgiler gizlidir. Tüm bu veriler, veritabanlarında çıkarılmayı bekleyen değerli bir maden gibi durmaktadır. Bir bakıma etrafımız bir sürü veri varken bu veriler bilgiye dönüşmeyi beklemektedirler.

Veri madenciliği 1990'lardan beri, veri depolama araçları, barkod ve RFID teknolojilerine paralel olarak gelişmekte ve kullanım alanı yayılmakta olan bir konudur. Bu nedenle, kullanılan yer ve zamana göre çeşitli tanımları yapılmıştır. Çünkü her geçen gün daha da geliştiği için bugün yapılan bir tanım yarın yetersiz kalabilmektedir. En yaygın tanımlardan bir tanesi şöyle der: Veri madenciliği daha önceden bilinmeyen, geçerli ve uygulanabilir bilgilerin geniş veritabanlarından elde edilmesi ve bu bilgilerin işletme kararları verirken kullanılmasıdır.

Burada altının çizilmesi gereken noktalardan birincisi elde edilecek bilginin "önceden bilinmeyen" olmasıdır. Veri madenciliği sonunda ulaşılacak bilginin önceden bilinmiyor olmasından kasıt, elde edilecek sonucun tahmin edilmemesi anlamını taşımaktadır. Zaten tahmin edilebilen, beklenen sonuçlar için veri madenciliği kullanmak pek de ekonomik olmayacaktır. Ayrıca veri madenciliği tahmin edilen, öngörülen ya da başka yöntemlerle çıkarılmış sonuçların ispatını yapmak üzere kullanılacak bir araç da değildir. Ayrıca, veri madenciliği daha önce hiç akla gelmemiş, düşünülmemiş sonuçları önümüze koymasıyla diğer

yöntemlerden farklılık gösterir. "Daha önce bilinmeyen" ya da tahmin edilemeyenle ilgili en ünlü örnek ise, artık klasikleşmiş, kulaktan kulağa anlatılan ve veri madenciliğinin "bilinmeyenini" çarpıcı bir şekilde önümüze koyan bira - çocuk bezi örneğidir:

Bir perakende mağazalar zincirinin yaptığı veri madenciliği araştırmasının sonuçlarına göre bira ile çocuk bezi satışları arasında, özellikle Cuma günleri, güçlü bir ilişki vardır. Çocuk bezi satın alan kişilerin büyük çoğunluğu aynı zamanda bira da satın almaktadırlar. Daha doğrusu, Cuma günleri çocukları için alışverişe çıkan babalar arada kendileri için de alışveriş yapmaktadırlar [Cabena, 1998].

Gartner Group tarafından yapılan bir diğer tanımda ise veri madenciliği, istatistik ve matematik tekniklerle birlikte örüntü tanıma (pattern recognition) teknolojilerini kullanarak, depolama ortamlarında saklanmış bulunan veri yığınlarının elenmesi ile anlamlı yeni korelasyon, örüntü ve eğilimlerin keşfedilmesi sürecidir [Akpınar, 2000]. Veri madenciliğini salt bir tanım olarak ele alırsak gerçekte var olan önemini tam olarak yansıtmamış oluruz. Salt bir tanımda veri madenciliği için bir tahmin aracı gibi yaklaşılabilir ya da veri madenciliği basit bir bilgisayar programı gibi görülebilir. Elbette ki sonuç olarak veri madenciliği kullanılacaksa bir bilgisayar yazılımı üzerinden kullanılacaktır; ancak veri madenciliğini tek bir programı anlamak ve onun arayüzünü kullanmak olarak el almak bu bilim alanına büyük bir haksızlık olacaktır. Oysa kullanım alanlarına baktıkça durumun hiç de öyle olmadığı görülecektir. Bu nedenle öncelikle veri madenciliğinin kullanım alanlarına ve amaçlarını bilip daha sonra bu amaçlar için geliştirilmiş teknik algoritma ve yazılımları kullanmakta yarar vardır.

## 1.2 Veri Madenciliğinin Uygulama Alanları

Veri madenciliği bankacılık, pazarlama, sigortacılık, sağlık gibi değişik alanlarda uygulanmaktadır. Veri madenciliğinin kullanılmasında sektör farkı gözetilmemekle beraber, geniş veri ambarlarının oluşturulmasına olanak veren, perakende satış, sigortacılık, sağlık gibi alanlarda kullanılması daha yaygın ve daha doğrudur.

## 1.3 Diğer Uygulamalar

Pazarlama ve risk yönetimi dışında veri madenciliği şu alanlarda da kullanılır:

### 1.3.1 İşaret İşleme

Telefon hatlarındaki parazitlenmeden dolayı oluşacak kayıpları ve buna bağlı olarak konuşmada ortaya çıkan gürültüyü yok etme.

### 1.3.2 Biyoloji

DNA sıra (veri) analizi. İnsanda yaklaşık 100.000 gen vardır. Hastalıklara yol açan gen sıralama örneklerini binlerce gen arasından bulmak, tanımlamak oldukça zor bir iştir. Veri madenciliğiyle geliştirilen sıralama örnek analizi ve benzerlik arama yöntemleri DNA verisi üzerinde analiz yapmayı kolaylaştırır [Kut, Yılmaz] .

### 1.3.3 Tıp

Bazı hastalıkların %100 kesin teşhisi mümkün olmamaktadır. Örneğin gebelik esnasında çocukta oluşabilecek herhangi bir down sendromu riskinin kesin tanısı dış bulgularla sağlanamamaktadır. Buradaki dış bulgulardan kasıt, anneden alınacak kan örneği, ultrason ile bebeğin görüntülenmesi, anne adayının yaşı, hamilelik ayı aldığı kilo vs gibi bulgulardır. Ancak bu bulguların hemen hiç biri hekime %100 tanı koyma olanağı vermez; %100 veya %100'e çok yakın bir tanı için anne karnından alınacak sıvının incelenmesi de gerekmektedir. Oysa bu işlemde de 1/300 oranında bir düşük riski vardır. Dolayısıyla bu işleme girmeden önce hekimin anne karnındaki bebekte down sendromu olduğundan kuşkulanması gerekmektedir. Bu aşamada yukarıda söz edilen dış bulgular ve veri madenciliği teknikleri devreye girmektedir.

Daha önce bu işlem uygulanmış, dış bulguları ve operasyon sunucu kaydedilmiş hasta adaylarına ait veritabanı, veri madenciliği algoritmaları tarafından incelenerek, bir makine öğrenmesi, sınıflandırma, karar ağacı vs. gerçekleştirilir. Daha sonra

gerçekleştirilen bu sisteme -örneğin karar ağacı- mevcut anne adayının bilgileri girilerek bebekteki risk oranı belirlenir. Bu oranın büyüklüğüne bağlı olarak hekimin bir yarar risk analizi yapıp operasyona karar vermesi kolaylaşır.

Tıp alanında bunun gibi ameliyat riski taşıyan ancak, ameliyat öncesinde gerçekten ameliyat olması gerektiği tam olarak anlaşılamayan hasta ve hastalıklar için de veri madenciliği yöntemi kullanılır.

Ayrıca parmak izi tespiti, yüz şeklinden kimlik tespiti, insan sesinin bilgisayar ve diğer elektronik aygıtlarda komut olarak kullanılması konularında da kullanılan yapaya zeka teknikleri veri madenciliği için de geçerlidir.

Amerika Bankası kendi ürünlerini kullanan banka müşterilerinin tespitinde veri. madenciliği kullanır ve müşteri ihtiyaçlarını karşılamak için ürün ve servislerden oluşan paketler sunar; Farmer Group şirketinin "Risk Analiz Yönetim" paketi, sigorta oranı belirlenmesi, yatırım portföyü yönetimi, iyi ve kötü kredi riskleri taşıyan şirketlerin ve müşterilerinin belirlenmesinde veri madenciliği kullanır; Twentieth Century Fox adlı film şirketi ise fatura bilgilerini analiz ederek hangi aktörün, hangi filmin, hangi bölgede daha çok izlendiğini tespit ederek yeni film projelerini başlatmış ve bölge bazında gösterimler sunmuştur [Kut ve Yılmaz].

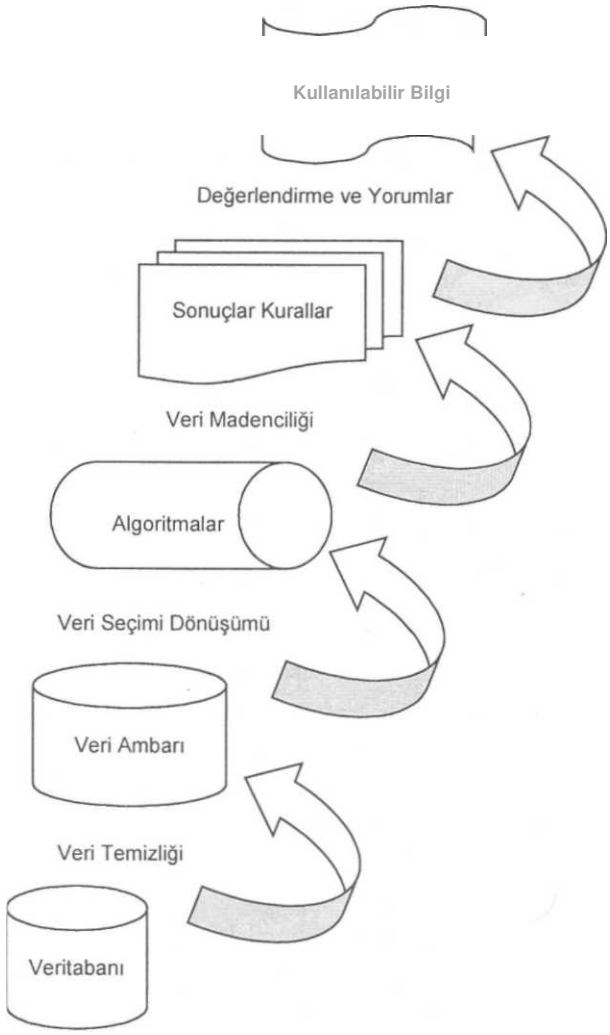
Görüldüğü gibi veri madenciliği birçok ve birbirinden farklı konuyla ilgilenmekte, başka bir deyişle birbirinden farklı birçok konu veri madenciliği yöntem ve teknikleri aracılığı ile geliştirilmektedir. Bunun böyle olmasının başlıca sebebi veri madenciliği ve yapay zeka konularının -makine öğrenmesi vs.- bir birlerine algoritma ve teknik olarak yaklaşmalarıdır. Aslında veri madenciliğinin yapay zeka, makine öğrenmesi adı altında geliştirilen algoritmaları kullanmakta ve ayrıca veri madenciliği üzerine çalışanların geliştirdiği algoritmalar da yapay zeka alanına büyük yarar sağlamaktadır.

## 1.4 Veri Madenciliği İçin Verilerin Hazırlanması

Veri madenciliğin ortaya çıkmasının ve günümüzde yaygın olarak kullanılıp bu konu üzerine araştırmalar yapılmasının en büyük nedenlerinden bir tanesi, günümüzde büyük veritabanlarının erişilebilir olmasıdır. Bugün süper marketlerde yapılan alışverişlerden tutunda, diğer kısım ve bölümlerde çalışan tüm personelle ilgili her türlü bilgi bilgisayarların belleklerinde tutulmaktadır. Ancak, bu veritabanlarındaki bilgilerin tamamının gerçek ve doğru bilgiler olduğunu kimse %100 garanti edemez; ayrıca bu bilgilerin, mevcut haliyle yapacağımız çalışmaya hizmet edeceği de kesin değildir [Kimball]. Bu yüzden, elimizdeki bilgilerin belirli işlemlerden geçirilmesi gerekebilir.

Elimizdeki veritabanı bazı kayıtlar yönünden eksik olabilir. Örneğin, veritabanında kayıtlı birçok kişinin medeni hali belirliyken, bu bilgi bazı kayıtlarda eksik olabilir; yani hiç kayıt girilmemiş olabilir, bu eksikliği kayıp veriler (missing data) olarak isimlendirebiliriz. Bunun dışında, kayıtların bir kısmındaki bilgiler, aşırı uç değerler ya da yanlış girilmiş değerler olabilir; bunun en çarpıcı örneği bir kişinin doğum tarihinin 1046 olarak girilmesi olabilir. Bu gibi bilgilere gürültü ya da gürültülü veri denilir [Han-2001]. Bunun dışında, verilerin bir kısmı, gerçekten yanlış, anlamsız bilgiler içerebilir; örneğin ürün kodları yanlış girilmiş olabilir. Ayrıca, bir bilgi bir kaç farklı yere gereksiz bir şekilde girilmiş, aynı anlama gelebilecek birden fazla bilgi olabilir. Örneğin, kayıtlı kişilerin hem yaşları hem de doğum tarihleri tutulmuşsa bunlardan bir tanesi kesinlikle fazladır. Bazen mevcut değişkenlerin birleşmesi ve tek bir değişken gibi işleme girmesi mümkün olabilir; bu hem veri madenciliği çalışması esnasında bilgisayar çalışma zamanı karmaşıklığını azaltacak hem de elde edilecek sonuçların güvenilirliğini ve kalitesini arttıracaktır. Zaman karmaşıklığını önlemek için tıpkı istatistik çalışmalarda yapıldığı gibi ana kütleden bir örnekleme alınarak eldeki veri boyutu düşürülür. Ancak bu örneklemenin yapılabilmesi için diğer istatistik çalışmalarından farklı olarak elimizde ana kütle verilerinin tamamının olması gerekmektedir.

Veri madenciliği çalışmasının en başında yapılması gereken şey verilerin hazırlanmasıdır. Bu konuyu verilerin temizlenmesi ve verilerin yeniden yapılandırılması olarak iki başlık altında inceleyebiliriz.




**Şekil - 1.1 Veri madenciliği ve bilgi keşfi süreci**

## 1.5 Veri Madenciliği Modelleri

Veri madenciliği konusu kullanıldıkları alanlara göre değişik modellere ayrılmaktadır. Bu modeller değer tahmini model, veritabanı kümeleme modeli, bağlantı analizi ve fark sapmaları olarak dört ana başlık altında toplanabilir; bu modeller literatürde operasyon veya yöntem isimleriyle de anılmaktadır [Cabena, 1998]. Bu operasyonlar ya da yöntemler uygulamada değişik amaçlar için

kullanılırken birçok teknik ve algoritmalardan yararlanılmaktadır; kullanılan teknik ve algoritmalar genel olarak tahminleyici, tanımlayıcı veya her iki yaklaşımı da içerebilirler.

Herhangi bir amaç için, örneğin hedef müşteri kitlesinin seçimi gibi tek bir modelin kullanımı söz konusu olmayabilir; diğer modeller de farklı teknik ve algoritmalarla aynı amaç için kullanılabilirler.

Veri madenciliğinde kullanılan algoritma, teknik ve modeller sonuçta birer bilgisayar yazılımıdır. Bu yazılımlar her ne kadar matematiksel ve algoritmik olarak birbirlerinden ayrılsalar da bazı ortak özellikleri vardır; bu özelliklerden bir tanesi yazılımların öğrenme işlemidir. Yazılımlar kendilerine giriş olarak verilen verilere bakıp inceleyerek değişik algoritmalarla bu verilerden bazı sonuçlar ve kurallar çıkarırlar. Bu inceleme işlemine ise "öğrenme" adı verilir. Daha sonra, bu çıkarımlar verilerin diğer kısmına uygulanarak sınanırlar. Yazılım bir bakıma kendi kendini sınav yaparak ne kadar öğrendiğini sınar. Bulduğu sonuca göre de gerekirse çıkarımlarını (kurallar vs) yeniler. Yenilenen bu yeni bulguların ayrı bir işlemle doğrulanması gerekmektedir. Bu işleme doğrulama denilir. Daha sonra aşırı öğrenme olup olmadığını da kontrol edilmesi gerekir. Aşın öğrenme algoritmanın çıkardığı kuralların sadece üzerinde çalıştığı veriler için geçerli olması, dışarıdan başka veriler geldiğinde, üretilen kuralların geçersiz olması durumudur. Bu durumda, üzerinde çalışılan veriler üzerinde, denendiğinde tam sonuç veren, çıkarım ve kurallar, aynı performansı dış veriler üzerinde gösteremezler.

Veri madenciliğinde sınıflandırma, kümeleme, bağlantı analizi ve dolandırıcılık tespiti gibi konularda kullanılmak üzere birçok algoritma geliştirilmiştir. Bu algoritmaların bazıları sadece sınıflandırma ya da kümeleme gibi konuları ilgilendirirken bazıları ise değişik versiyonlarla birden fazla konuda kullanılabilmektedir. Örneğin, genetik algoritmalar, yapay sinir ağları gerek sınıflandırma ve gerekse de kümeleme modellerinde kullanılabilirler; oysa Aproiri algoritması sadece birliktelik kurallarının belirlenmesinde kullanılan bir algoritmadır.

### 1.5.1 Değer Tahmini Modeli

Değer tahmini ya da tahminsel modeldeki öğrenme daha çok bir insanın öğrenmesine benzemektedir. İnsan tüm yaşamı boyunca çevresini sürekli gözleyerek bir şeyler öğrenir. Aslında çevremizdeki her şey gerçek bir veritabanından başka bir şey değildir. Tahminsel model de kendisine verilen veritabanını inceleyerek, bu veritabanındaki temel unsurları birbirine benzeterek tanımlamaya, onları isimlendirmeye ve sınıflamaya çalışmaktadır. Tıpkı bir çocuğun kadın ve erkek cinsiyetlerini sınıflandırması gibi. Çocuk için ilk önce cinsiyet kavramı yani bir sınıflandırma yoktur; daha sonra anne-baba, teyze, hala, amca, kendinden büyük ve küçük erkek ve kız çocuklarını görür. Aslında tüm bunlar çocuk için bir veritabanıdır. Bu veritabanını inceleyen çocuk kadınla erkek arasındaki temel farkları belirler daha sonra kendisine hiç tanımadığı kız çocuğu gösterildiğinde bir önceki deneyimine/ öğrenmesine dayanarak bunun kız olduğuna karar verir. Aslında yaptığı tamamen bir sınıflandırma, genelleme yapma işlemidir; daha doğrusu bir tahmindir. Bu şekildeki bir öğrenmeye de denetimli öğrenme' adı verilir [Cabena, 1998].

Denetimli öğrenmede, öğreniciye (öğrenici bir algoritmadır) nesneler ve nesnelerin özellikleri ve yine bu nesnelerin tanımlanmış, ilerideki aşamada tahmini istenecek olan değişkenleri verilir. Veri madenciliğindeki nesneler günlük yaşamdaki nesneler gibidir; ancak, buradaki nesneler veritabanındaki her bir kayıttır. Şekil-2.3 deki ürünler tablosu nesnelerden oluşmaktadır. Örneğin, veritabanındaki su nesnesinin özellikleri veritabanına girilmiştir. Normal yaşamda yukarıda belirtilen, bir çocuğun öğrenme işleminde, gördüğü nesnelerin özelliklerini değerlendirilir. Veri madenciliğinde de yine, tasarlanan algoritma ya da program veritabanına girilmiş olan nesnelerin özelliklerini değerlendirir. Günlük yaşamda, çocuk su ile bardak nesnesinin birbirinden farklı nesneler olduğunu sürekli görerek, birden fazla defa bu nesnelere maruz kalarak öğrenir. Hatta başlangıçta bu iki nesneyi işaretlerle büyüklere tanıtmaya çalışır; bu nesneleri tanır, fakat isimlerini bilmez. Çünkü onun için nesnelerin özelliklerinin önemi vardır; isimlerinin değil! Daha sonra, bu nesnelerin sabun, deterjan, su vs gibi isimlerini de öğrenerek bu isimleri nesnelerin özellikleriyle özdeşleştirir. Bundan sonra herhangi bir nesnenin bir veya birkaç özelliği verilince bu nesnelerin isimlerini tahmin eder. Veri madenciliği algoritmaları

da veritabanındaki ya da daha doğru bir deyişle oluşturulmuş veri ambarındaki nesnelerin özelliklerini, nesnelerin isimleriyle ilişkilendirerek bu nesnelerin birbirinden farklı ya da benzer, aynı sınıftan nesneler olduklarını bulur ve öğrenir. Daha soma, kendisine verilen değişik özellikleri değerlendirerek bu özelliğe sahip olan nesnenin ismini tahmin eder.

| Adı | Renk | Miktar | Kategori | Tip |
|---|---|---|---|---|
| Su | Berrak | 0,5lt | Gıda | sıvı |
| Deterjan | Yeşil | 330 gr. | Temizlik | jel |
| Sabun | Kırmızı | 150 gr. | Temizlik | katı |
| Bardak | Muhtelif | 10 Pak. | Piknik | Plastik |

**Tablo - 1.1 Ürünler tablosu**

Denetimli öğrenmenin tersi duruma ise denetimsiz öğrenme denilir. Denetimsiz öğrenmede nesnelerin özellikleri verilirken tahmin için kullanılacak herhangi bir parametre verilmez; yani nesnelerin isimleri verilmez. Örneğin, yukarıda verilen ilk örnekte, iki cinsiyet arasındaki farkı öğrenen çocuğa, erkek-kadın parametreleri, diğer örnek için su, deterjan, bardak vs değerleri denetimsiz öğrenme için verilmez.

Tahmini yaklaşım, uygulamada, hedef müşteri kitlesinin seçimi, pazar sepeti analizi, çapraz satış, müşteri ilişkileri yönetimi gibi alanlarda kullanılır. Bu yaklaşımdan yola çıkarak geliştirilen en temel iki teknik, karar ağaçları ve sınıflandırmadır. Ayrıca genetik algoritmalar, yapay sinir ağları, Bayes yöntemi gibi yöntemler de bu amaç için kullanılırlar. Bu model çerçevesinde geliştirilen teknik ve algoritmalar ileride ayrıntılı bir şekilde ele alınmaktadır.

### 1.5.2 Bağlantı Analizi

Tahmini modelde kullanılan yazılım kendisine verilen veritabanının bir bütün olarak düşünür ve öğrenmesini de bu bütünü temel alarak gerçekleştir. Oysa bağlantı analizinde veritabanındaki her bir kayıt veya kayıtlar grubu arasında bir bağlantı, ilişki yaratılmaya çalışılır. Bağlantı analizi bir veritabanındaki kayıtlar ya da bir graf üzerindeki düğümler arasında çok rastlanan kuralları ortaya çıkarır.

Bağlantı analizinin en çok kullanıldığı alanlardan bazıları şunlardır: Çapraz satış, stok fiyat hareketleri ve hedef müşteri kitlesinin belirlenmesi [Cabena, 1998]. Bağlantı analizi dört ana başlık altında incelenebilir. Bunlar birliktelik kuralları, örüntü tanıma, ardışık zaman örüntüleri ve benzer zaman keşfidir.

### 1.5.3 Birliktelik Kuralları

Birliktelik kuralı' belirli türlerdeki veri ilişkilerini tanımlayan bir modeldir. Bu yönden de tanımlayıcı bir modeldir. Herhangi bir ürün alındığında bu ürünün yanında bir başka ürünün de satın alınması bir birliktelik kuralı verir. Ürünler ve bu ürünlerin birlikte alınmaları söz konusu olunca, hemen anlaşılacağı gibi birliktelik kuralları daha çok perakendecilik sektöründe faaliyet gösteren işletmelerde uygulanmaktadır. Örneğin bir süpermarkette yapılan alışverişlerin incelenip hangi ürünün hangi ürünle birlikte satın alındığının belirlenmesi birliktelik kurallarını ilgilendirir. Bunun dışında örneğin iletişim ağlarında meydana gelen hataların belirlenmesinde de kullanılabilir [Dunham, 2003].

Bilindiği gibi veriler bilgisayar üzerinden sayısal olarak iletilirler; her bir verinin taşıdığı bir sayısal değer vardır. Bu sayısal değerler arsında ilişki kurularak hangi sayısal değerden sonra hangi sayısal değerin gelebileceğinin otomatik olarak belirlenip söz konusu sayısal değerin kaybolması, okunamaması gibi durumlarda bu sayının yerine gelebilecek sayısal değer (tahmini) otomatik olarak konularak iletişimin kopmaması sağlanır. Örneğin, verilen mesaj "..gün onu göremedim" olsun, "gün" sözcüğünün önünde 2 karakter kaybolmuştur. Bu durumda yazılım, daha önceden incelediği veritabanından çıkarttığı birliktelik kurallarıyla bu eksikliği

karşılaştıracak ve olası olarak "bugün, o gün" gibi olasılıkları bulacak ve otomatik olarak en muhtemel seçeneği "..gün" yerine koyacaktır. Unutulmamalıdır ki! Hiçbir aşamada insan müdahalesi yoktur. Yani "bugün, o gün" olasılıkları tamamen veritabanına girilmiş belki de milyonlarca kayıt üzerinde, yine yazılımın yaptığı bir keşifle bulunmaktadır. Bu yüzden de veri madenciliği bilgi keşfi olarak da anılır.

Görüldüğü gibi birliktelik kuralları aslında olaylar arasındaki probabilastik korelasyonu tanımlar. Olaylar arasındaki korelasyon ise sık sık beraber gözlenen olaylardır. Herhangi bir veritabanında birliktelik kurallarının tanımlanması veritabanı bilgi keşfi sürecinin ilk adımıdır.' Veritabanındaki herhangi bir X'in aynı zamanda Y'yi içermesi bir birlikteliktir. Bu durum çok bilinen bir örnekle şöyle açıklanabilir: "Bira içeren %30 alışverişin, %2'si aynı zamanda çocuk bezi de içermektedir." Burada %30 güven seviyesini, %2 ise bu güven seviyesine olan desteği belirtmektedir [Joshi].

Bu ilişkileri ortaya çıkartmaya çalışan bir analistin bir takım kurallar elde edebilmesi için minimum kabul edilebilir destek ve güven değerlerini belirlemesi gerekir. Birçok birliktelik kuralları algoritması "üret ve test et stratejisi" kullanır. Örneğin Apriori algoritması k. döngüsünde, aday k-dizilerini belirler ve bunların güven seviyeleri de veritabanının taranmasıyla yapılır. Bulunan birçok güven seviyesi ise çoğu zaman önceden belirlenmiş olan güven seviyesinden düşük çıkacağı için bu güven seviyesi düşük çıkan aday diziler ret edilirken, muhtemelen daha az sayıda olan diğer adaylar kural üretmek için kullanılacaktır: Bunun dışında kullanıcıdan, yani veri madenciliği analizini yapacak kişiden ayrıca bir destek değeri alınır [Orhunbilge, 1999]. Destek değeri elde edilen bilgilerin mevcut benzer bilgiler içindeki oranını temsil eder. Minimum güven, minimum destek seviyeleri Apriori ve bu konuyla ilgili diğer algoritmalarla birlikte ileride ayrıntılı olarak ele alınmaktadır.

### 1.5.4 Örüntü Tanıma

Örüntü tanıma, daha önce belirlenmiş bir model diyebileceğimiz çok boyutlu bir örüntünün veritabanındaki benzerlerini ya da 'en benzerini' aramaktır [Dunham, 2003]. Herhangi bir yazılı metni tanımak ya da o metnin çok benzerini bulmak örüntü tanımanın konusuna girer. Bunun dışında parmak izi, ses, yüz tanıma, kan hücrelerinin karşılaştırılması, el yazılarının tespiti gibi alanlarda da uygulanır. Dolayısıyla örüntüden kasıt el, yüz resim, çizim ve ses gibi varlıkların sayısal ortamda sergiledikleri şekildir.

Aslında, örüntü tanımada yapılan işlem bir çeşit sınıflandırmadır. Elimizde, ulaşılması gereken bir örnek vardır ve biz bu örneğin benzerini veya mümkünse aynısı aramaktayız. Bir algoritmayla bu örneğe benzeyenleri bir araya topluyor, daha sonra bunları en benzerden, en aza doğru sıralıyoruz.

Futbol maçlarında fanatizmi ve şiddeti sona erdirmek ya da en azından azaltmak için alınan önlemlerden bir tanesi de daha önceden olay çıkarttığı belirlenmiş kimselerin statlara alınmamasıdır. Ancak kimlik tespitine rağmen, başkasının kimliğini kullanarak ya da kalabalıktan istifade ederek, bu kişiler statlara girebilmektedirler. Bu kişilerin eldeki fotoğrafları statta çekim yapan bir kameradan alınan seyirci fotoğrafları ile sürekli olarak karşılaştırılarak, bu kişilerin yakalanması sağlanmaktadır. Burada kullanılan çeşitli örüntü tanıma algoritmaları vardır. Hemen hepsi 0 ve l'lerden oluşan çeşitli matrisleri karşılaştırarak bir benzerlik aramaktadır.

Günümüzde, klavye ve fare gibi giriş cihazları yerlerini, doğal insan sesine bırakmaya başlıyorlar; bilgisayara herhangi bir dosyayı aç denildiğinde bilgisayar doğrudan gidip o dosyayı açıyor. Aslında bu işi yaparken, bilgisayar söylenen cümleyi anlamsal olarak algılamıyor. Sizin ağzınızdan çıkan sesle daha önceden kaydedilmiş sözcüklerin sayısal ortamdaki örüntülerini/seslerini karşılaştırarak örüntü tanıma işlemi yapılıyor ve bulunan örüntünün karşısındaki komut yerine getiriliyor[Joch, 1994].

Başka bir örnek basketbol oyunundan verilebilir. Topun, hangi oyuncu tarafından hangi yolları izleyerek hareket ettiği belirlenerek, daha sonra belirli bir oyuncunun, söz gelişi 3 ya da 5 saniye sonra ne yapacağının önceden belirlenmesi bir örüntü tanıma uygulamasıdır.

Örüntü tanıma için geliştirilmiş birçok algoritma vardır. "K-en yakın komşu algoritması", "doğrusal sınıflayıcı", "üstsel sınıflayıcı" bunlardan sadece birkaç tanesidir. Bu algoritmalardan doğrusal olanlar yalnızca doğrusal işlevleri yerine getirirken, doğrusal olmayanlar her türlü işlevi yerine getirebilir.

### 1.5.5 Ardışık Zaman Örüntüleri

Yukarıda örüntü (pattern) sözcüğünün, herhangi bir çizim, ses, resim, parmak izi vs gibi bir şekil olduğundan söz edilmişti. Bu örneklere ek olarak, bir kimsenin yaptığı işler de örüntü olarak tanımlanabilir. Örneğin bir müşterinin süt, peynir ve ekmek satın alması bir örüntüdür. Bu noktadan hareket edilerek bir müşterinin birinci gün A ürünü, onu izleyen gün veya günlerden birinde B ürünü ve daha sonraki bir günde de C ürünü alması ise yine bir örüntü oluşturacaktır. Ancak bu sefer birbirini izleyen, yani zaman içinde ardışık olan bir örüntü oluşturacaktır.

Veri madenciliği, perakendecilik sektöründen gelen yoğun ilgiyle gelişimini sürdürmektedir. Barkod teknolojisindeki gelişme, perakendecilik sektöründe faaliyet gösteren işletmelerin büyük miktarda ve sık denilebilecek aralıklarda oluşan bilgileri, süratle depolamasına olanak sağlamıştır. Satış miktarlarını gösteren bu verilere sepet verisi diyebiliriz[Agrawal, 1995]. Bu veriler genel olarak işlem tarihi, satılan ürünler, işlem sıra numarası gibi bilgileri içerir. İşlem yapılan yerde, burası tipik olarak bir süpermarkettir; üye kartı ve klüp kartı diyebileceğimiz kartlar kullanılıyorsa bu bilgiler arasında müşteri numarası da bulunabilir. Bu kayıtlar incelenerek, ardışık zaman örüntüleri belirlenebilir. Eğer müşterilerin bir kısmı, önce bir CD oynatıcısı, daha sonraki bir zamanda ABC ismindeki bir filmi, ardından XYZ ismindeki bir filmi satın alıyorlarsa bu bir ardışık örüntü oluşturacaktır. Başka müşteriler ABC filmi ile XYZ filmi arasındaki zaman diliminde, farklı filmler ya da farklı ürünler alsalar bile onlarda bu örüntüye dahildirler.

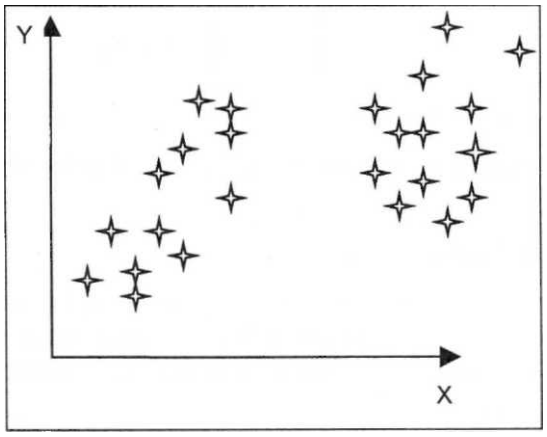
### 1.5.6 Kümeleme Analizi

Kümeleme analizi (clustering) veri madenciliğinin en önemli alanlarından birisidir; amacı, nesneleri birbirlerine olan benzerliklerine göre gruplara ayırmaktır. Elde bulunan veriler incelenerek birbirlerine benzeyenler bir kümeye, benzemeyenler ise bir başka kümeye toplanırlar.

Verilerin kümeleme analizine göre modellenmesinde matematik, istatistik, makine öğrenimi ve yapay zeka gibi birçok alandan yararlanılır. Makine öğrenimi açısından, her bir küme gizli bir örüntüyü temsil eder ve uygulanan öğrenme ise bir denetimsiz öğrenmedir. Bu açıdan bakıldığında ise kümelemeyi, "gizli örüntülerin ortaya çıkartılması için uygulanan bir denetimsiz öğrenme yaklaşımı" olarak tanımlanabilir [Berkin].

Kümelemedeki öğrenmenin denetimsiz öğrenme olmasının nedeni önceden belirlenmiş sınıfların olmayışıdır. Önceden sınıflar belirli olsaydı, zaten bu kümeleme değil, bir sınıflandırma modeli adını alacaktı. Önceden sınıflar belirli iken, yani kadın ve erkek diye iki ayrı sınıf varken yapılan (algoritmik) öğrenmeye denetimli öğrenme; herhangi bir sınıf ismi verilmeden yapılan öğrenmeyeyse denetimsiz öğrenme denilir. Örneğin, veritabanındaki kayıtlarda her kaydın yanında kadın veya erkek bilgisi yazılıyor olsun, bu durumda veritabanı üzerinde yapılan herhangi bir (kadın veya erkek olduğuna dair) kural çıkarma işlemi denetimli öğrenmedir. Ancak aynı veritabanında, kayıtların yanında kadın mı erkek mi olduğu bilgisi yok iken yapılan kural çıkarma işlemi denetimsiz öğrenmedir. Bu işlem aynı zamanda veritabanını (iki) kümeye ayırma, yani kümeleme işlemidir. Burada kadın/erkek gibi bir etiket ya da sınıf olmayacağı için kümeleme kayıtlar arasındaki benzerlik veya mesafe ölçütüne göre yapılır.

İki verinin benzerliğinden kasıt ise aralarındaki mesafenin ölçülmesi ve değerlendirilmesidir [Nanopoulos, 2001]. Bu değerlendirme, veritabanındaki diğer verilere kıyasla iki verinin ne kadar yakın ya da benzer oldukları açısından yapılabileceği gibi önceden belirlenmiş kısıtlar eşik değerleri çerçevesinde de yapılabilir.




**Şekil - 1.2 Euclid uzayında veriler**

Örneğin X ve Y özelliklerine sahip olan veriler topluluğu düşünelim. Bu verileri X ve Y değerlerine göre iki boyutlu bir Euclid uzayında Şekil-2.7 deki gibi gösterilebilir.

Şekil - 1.2 de görülen noktaların birbirlerine olan mesafesi esas alındığında toplamda iki kümenin oluştuğu açıktır. Ancak, değişken sayısı arttıkça bu kümeleri görmek iki boyutlu uzayda olduğu gibi kolay olmayacaktır. Örneğin, A={1, 1, 2, 2, 5} dizisine, B={1, 2, 3, 4, 2} dizisi mi, yoksa C={1, 3, 5, 1, 3} dizisi mi yakındır? Bunun yanıtını Şekil 2.7 deki gibi bir grafikte görerek ya da göstererek vermek kolay olmayacaktır. Bu nedenle, diziler arasındaki mesafe matematiksel olarak ölçülmelidir. Yani mesafe (A, B) ile mesafe (A, C) bilinmelidir.

Öyleyse, bir grup noktanın küme olup olmadığının belirlenmesi için bir mesafe ölçümüne ihtiyaç vardır. İki nokta arasındaki uzaklığı mes (x, y) olarak gösterilirse, mes (x, y) bize x ile y arasındaki uzaklığı verecektir. Genel olarak aşağıdaki gibi kabul edilir [Ulman].

mes (x, y) = 0 ise iki nokta arasındaki uzaklık sıfırdır.

mes (x, y) = mes (y, x) ise mesafe simetriktir.

mes (x, y) = mes (x, z) + mes (z, y) ise bir üçgen eşitsizliği vardır.

## 1.6 Sınıflandırma Teknikleri ve Algoritmaları

Sınıflandırma en çok bilinen veri madenciliği tekniklerinden birisidir; resim, örüntü tanıma, hastalık tanıları, dolandırıcılık tespiti, kalite kontrol çalışmaları ve pazarlama konuları sınıflandırma tekniklerinin bolca kullanıldığı alanlardır. Sınıflandırma tahminleyici bir modeldir; havanın bir sonraki gün nasıl olacağı ya da bir kutuda ne kadar mavi top olduğunun tahmin edilmesi aslında bir sınıflandırma işlemidir [Dunham, 2003].

Sınıflandırma işleminde elimizdeki sınıf veya istatistiksel değimiyle bağımlı değişken hem sınıfsal hem de sürekli değer taşıyabilir; bu anlamda regresyon ve çok terimli regresyona yaklaşmaktadır [Akpınar, 2000]. Ancak, veri madenciliği çerçevesinde bu istatistiksel yöntemlerin dışında sınıflandırma işleminde "Bayesyen sınıflandırma algoritması", "karar ağaçlarına dayalı algoritmalar", "yapay sinir ağları" [Lipmann, i987] temelli algoritmalar ve"k-en yakın komşu algoritması" gibi birçok teknik ve algoritma geliştirilmiştir.

### 1.6.1 Karar Ağaçları

Karar ağaçları sınıflandırma problemlerinde en çok kullanılan algoritmalardan birisidir. Diğer yöntemlerle kıyaslandığında karar ağaçlarının yapılandırılması ve anlaşılması daha kolaydır denilebilir [Agrawal, 1993]. Bu teknikte sınıflandırma için bir ağaç oluşturulur; daha sonra, veritabanındaki her bir kayıt bu ağaca uygulanır ve çıkan sonuca göre de bu kayıt sınıflandırılır. Temel olarak iki adımdan oluştuğu

söylenebilir: Birincisi ağacın kurulması, ikincisi de verilerin teker teker ağaca uygulanarak sınıflandırmanın gerçekleştirilmesi şeklindedir.

Karar ağacına dayalı olarak geliştirilen algoritmalar genel olarak aşağıda verilen kaba kod çerçevesinde çalışır:

D: Öğrenme veritabanı

T: Kurulacak ağaç

T = 0  // başlangıçta ağaç boş küme

Dallara ayırma kriterlerini belirle

T = kök düğümü belirle

T = dallara ayrılma kurallarına göre kök düğümü dallara ayır;

Her bir dal için

do

Bu düğüme gelecek değişkeni belirle

if (durma koşuluna ulaşıldı)

    Yaprak ekle ve dur

else

loop

Verilen kaba kodda sözü edilen durma/sonlandırma koşulunu açıklamakta yarar vardır. Karar ağacı kurulurken eldeki veritabanının bir kısmı öğrenme işlemi için kullanılarak ağaç oluşturulacaktır; bu arada veritabanının bir kısmı da oluşturulan ağacı test etmek için kullanılır. Ağaç oluşturulurken kurulan sistemin çalışıp çalışmadığı belirlenir. Eğer ağaç belirlenen düzeyde çalışıyorsa dallanma durdurulur ve sınıflandırma tamamlanır. Programdaki durdurma kriteri ağacın hassasiyetini de ortaya koyar. Geç durdurulan bir ağaç daha fazla dallanacak ve ağaç daha geniş olacak, çalışma süresi uzayacaktır. Bunun karşılığında ise daha duyarlı sonuç verecektir. Erken durdurulan ağaç ise her ne kadar daha hızlı çalışsa da tam öğrenmenin gerçekleşmeme olasılığını her zaman taşıyacaktır [Dunham, 2003].

Ağaç oluşturmada yapılan işlemlerden bir tanesi de budama işlemidir. Budama ağaçta oluşmuş sonucu etkilemeyen ve sınıflamaya herhangi bir katkısı olmayan dalların ağaçtan alınmasıdır. Bir bakıma gereksiz ayrıntıların sonuçtan çıkartılması işlemidir. Ağaçta birçok düğüm ve dal oluşursa, ağacın alt dalları ve yapraklarına ulaşan veri sayısı da azalacaktır; bu da ağacın hassasiyetini azaltacaktır [Cabena, 1998]. Budamanın gerçekleştirilmesi için kullanılan algoritmanın işleyiş biçimi budamanın hızı açısından önemlidir; ancak daha önemli bir unsur ise budamanın hangi ölçüte göre yapılacağının belirlenmesidir. Yani biraz önce sözünü ettiğimiz "gereksiz ayrıntıların" ne olduğunun belirlenmesidir.

Kullanılan algoritmaların çoğunda varsayılan (default) değer olarak %5 - %30 arası değerlerden düşük anlamlılık gösteren değerler budanırken, bu anlamlılığın belirlenmesi kullanıcıya bırakılmaktadır. Budama, gerek ağacın kurulumu esnasında gerekse de kurulduktan sonra yapılabilir.

Karar ağaçlarına dayalı olarak geliştirilen birçok algoritma vardır. Bu algoritmalar birbirlerinden kök, düğüm ve dallanma kriteri seçimlerinde izledikleri yol açısından ayrılırlar.

### 1.6.2 İstatistiğe Dayalı Algoritmalar

Veri madenciliğinde verilerin önceden verilen sınıflara göre ayrılması, aslında, gelecekte elde edilecek sonuçların tahminidir; yani sınıfların tahminidir. Regresyon, lojistik regresyon, zaman serileri analizi ve Bayesyen yaklaşım gibi istatistiksel yöntemler kullanılarak bu sınıflandırma işlemleri gerçekleştirilebilir.

#### 1.6.2.1 Bayesyen Sınıflandırma

Bayesyen sınıflandırma tekniği, elde var olan, hali hazırda sınıflanmış verileri kullanarak yeni bir verinin mevcut sınıflardan herhangi birine girme olasılığını hesaplayan bir yöntemdir. Bayesyen kuralına dayalı geliştirilmiş algoritma ve sınıflandırma teknikleri bu adla anılır.

#### 1.6.2.2 Regresyon

Regresyon analizi herhangi bir değişkenin bir veya daha fazla başka değişkenler arasındaki ilişkinin matematiksel bir denklem şeklinde yazılmasıdır, yazılan bu denkleme regresyon denklemi adı verilir.

Regresyon, sınıflandırma için aşağıdaki iki yaklaşım çerçevesinde kullanılır [Dunham, 2003].

Bölme: Veriler sınıfa bağlı olarak çeşitli bölgelere ayrılır.

Tahmin: Çıktı değerinin hesaplanması için formüller üretilir.

Bir bağımlı değişkenin tek bir bağımsız değişkenle açıklanabildiği durumlarda kullanılan regresyona "basit regresyon analizi" denilirken bağımlı değişkenin birden fazla bağımsız değişkenle açıklandığı durumlarda kullanılan regresyona ise "çoklu regresyon analizi" denilir. Bunun dışında kullanılan fonksiyonun, yani oluşturulan denklemin türüne göre de bir ayırım yapılacak olunursa "doğrusal" ve "doğrusal

olmayan" regresyon analizi olarak yine ikiye ayrılabilir. Bu durumda, yukarıda söz edilen her iki ayırımın kombinasyonları da oluşacaktır ve basit doğrusal, basit doğrusal olmayan vs gibi ayırımlar da yapılabilecektir.

En küçük kareler yöntemiyle elde edilen doğrusal bir regresyon denklemi aşağıdaki gibi verilebilir:

$$y = a + b\,x + e$$

Burada a doğrusal fonksiyonun sabitidir; b ise doğrusal fonksiyonun eğimidir. Yani regresyon katsayısı olarak da adlandırılan x'teki bir birimlik değişimin, y üzerinde, yine y cinsinden yaratacağı değişikliği gösteren bir katsayıdır [Orhunbilge, 1999]. y bağımlı değişkeni yani tahmin edilecek değişkeni temsil etmektedir. Veri madenciliği açısından ise y sınıfları temsil etmektedir, y'nin alacağı değere göre verilen x değerinin hangi sınıfa dahil olacağı tahmin edilecektir, x ise bağımsız değişkeni temsil etmektedir; daha doğrusu sınıflanacak verinin giriş değeridir. Bu giriş değerleri belirli bir regresyon denkleminde yerine konularak bir y değeri hesaplanacak ve bu hesaplanan y değerine göre elimizdeki x değerlerinin hangi sınıfı temsil ettiği tahmin edilmiş olacaktır.

Birden fazla x değeri için ise çoklu doğrusal regresyon denklemi ise aşağıdaki gibi olacaktır.

$$y = a + b_1x_1 + b_2x_2 + \ldots + b_ix_i + e$$

Regresyon analizinde daha üst dereceden fonksiyonlar kullanılmaz. Bunun nedeni eldeki verilere çok bağımlı sonuçlar elde edilecek olmasıdır. Veri madenciliğinde eldeki verilere aşırı bağlı sonuçlar elde edilmesine aşırı öğrenme denilir. Aşırı öğrenme durumunda elde edilen sonuçlar öğrenme için kullanılan verilere uygulandığında çok iyi sonuçlar verirken, dışarıdan başka veriler üzerinde kullanıldığında daha az hassas sonuçlar elde edilmesine yol açar [Cabena, 1998].

### 1.6.3 Mesafeye Dayalı Sınıflandırma Algoritmaları

Sınıflandırma yapılırken eldeki verilerin birbirlerine olan uzaklığı veya benzerliği kullanılarak sınıflamanın gerçekleştirilmesi bir başka sınıflandırma tekniğidir. Veriler arasındaki mesafe ölçülürken en çok kullanılan mesafe Euclid mesafesidir. Mesafeye dayalı algoritmalardan en bilineni "K-en yakın komşu algoritması" dır. Bu algoritmanın literatürde geliştirilmiş birçok türevi vardır. Genel olarak tümü aynı davranışı kullanarak işlem yaparlar.

### 1.6.3.1 K- En Yakın Komşu

En yaygın algoritmalardan birisidir. İngilizce kaynaklarda K-Nearest Neighbor ya da KNN şeklinde ifade edilir. Sınıflandırma yapılırken veritabanındaki her bir kaydın diğer kayıtlarla olan uzaklığı hesaplanır. Ancak, bir kayıt için diğer kayıtlardan sadece k adedi göz önüne alınır. Algoritmanın isminden de anlaşılacağı gibi bu k adet kayıt, başka bir deyişle veritabanındaki nokta, mesafesi hesaplanan noktaya diğer kayıtlara nazaran en yakın olan kayıtlardır. Bu yöntem coğrafi bilgi sistemlerinde çok kullanılır; belirlenen bir noktaya en yakın şehir, istasyon vs belirlenmesi aslında k-en yakın komşu algoritmasının temelini oluşturur [Beyer, 1999].

Algoritmada k değeri önceden seçilir; değerinin yüksek olması birbirlerine benzemeyen noktaların bir araya toplanmasına, çok küçük seçilmesiyse birbirine benzediği, yani aynı sınıfın noktaları oldukları halde, bazı noktaların ayrı sınıflara konmasına ya da o tür noktalar için ayrı sınıfların açılmasına neden olur. Tipik k değerleri 3.5ve7dir [Khan, 2002].

Algoritmanın çalışma ilkesi aşağıda verilen adımlarla özetlenebilir:

Uygun bir mesafe ölçüm uzayı belirle. (Euclid mesafe ölçümü en çok kullanılandır.) Birbirine en yakın k adet noktayı belirle. Belirlenen grubun en çok rastlandığı sınıfı belirle. Bu gruba belirlenen sınıfın ismini ata.

Bugüne kadar, k-en yakın komşu algoritmasının daha etkin ve daha hızlı çalışan birçok uyarlaması tasarlanmıştır. Sınıflandırmanın dışında kümeleme işlemlerinde de kullanılabilir.

## 1.7 Yapay Sinir Ağları

Yapay sinir ağları biyolojik sinir ağlarından esinlenerek geliştirilmiş bir bilgi işleme sistemidir. Yapay sinir ağları, yapay sinir hücrelerinin birbirleriyle çeşitli şekilde bağlanmasından oluşur ve genel olarak katmanlar şeklinde düzenlenir. En belirgin özellikleri birbirlerine bağlı nöronlar, bağlantılar arasındaki ağırlıkların belirlenmesi ve ateşleme fonksiyonudur. Geçmişi 1942 yılına kadar gitmektedir.

## 1.8 Birliktelik Kuralları ve İlişki Analizi

Veritabanlarındaki bilgi miktarı arttıkça birçok kurum ve kuruluş sahip oldukları bilgiler arasındaki ilişkileri ortaya çıkarma çabası içerisine girmiştir. Böylesi yığınlar halindeki bilgiler arasındaki ilişkiler kurumlar için altın değerinde sonuçlar doğurabilecek kararlarının alınmasında önemli rol oynamaktadır. İlişki analizi veritabanındaki bir dizi bilgi ya da kaydın diğer kayıtlarla olan bağlantısını açıklayan işlemler dizisidir. Yani bir kayıt varken, herhangi bir başka kaydın var olma olasılığı nedir? Ya da bu iki kayıt varken, diğer bir üçüncü hatta dördüncü kaydın veritabanına girme olasılığı nedir? İlişki analizi bu tür soruların yanıtım verir ve verilerin birlikte olan kurallarını ortaya çıkarır.

İlişki analizi satış-pazarlamadan, ürün katalog tasarımlarına kadar birçok alanda kullanılmaktadır. Örneğin, herhangi bir ürün satın alırken, bu ürünün yanında başka bir ürün ya da ürünlerin satın alınması, bu ürünler arasındaki bağlantıyı ortaya koyar. Bu bağlantıların ortaya çıkartılması ve bunun bir kural olarak ortaya konması ise birliktelik kuralları yani ilişki analizi konusuna girer. Literatürde bu tür çalışmalara "pazar sepeti analizi" denilir. Pazar sepeti analizi müşterilerin alışveriş alışkanlıklarının veritabanındaki bilgiler aracılığıyla ortaya çıkartılması işlemidir. Bu alışveriş alışkanlıklarının ortaya çıkartılması alış-veriş merkezindeki ürünlerin

yerleştirilmesi, marketin alanının tasarımı ve markette sergilenecek ve satılacak olan ürünlerin belirlenmesinde yardımcı olur.

### 1.8.1 AIS Algoritması

AIS algoritması, geniş nesne kümeleri üretmek için geliştirilmiş bir algoritmadır. 1993 yılında geliştirilmiştir. Veritabanındaki isimlerin, yani nesne isimlerinin A 'dan Z'ye sıralanması kısıtını taşır.

AIS algoritması veritabanını birçok kez tarar ve her tarama esnasında tüm işlemleri okur. İlk tarama esansında veritabanındaki nesneleri, teker teker sayarak hangilerinin geniş nesneler olduğunu belirler. Bunlardan geniş olanlar aday nesne kümeleri olarak işaretlenirler. Bir işlem tarandıktan sonra, bir önceki taramada geniş oldukları belirlenen nesne kümeleriyle, o işlemin nesneleri arasındaki ortak nesne kümeleri belirlenir. Belirlenen bu ortak nesne kümeleri işlemde mevcut olan diğer nesnelerle birleştirilerek yeni aday kümeler oluşturulur. Herhangi bir 1 nesne kümesi, bir işlemdeki nesnelerle birleşip aday kümelerden birini oluşturabilmesi için, birleşeceği nesnenin hem geniş olması hem de harf sırası açısından nesne kümesi içindeki tüm nesnelerden sonra geliyor olması gerekir.

AIS algoritması bu adımı gerçekleştirebilmek için, bir budama tekniği kullanır. Budama tekniğinin özünde, aday kümeler içindeki gereksiz kümelerin silinmesi vardır. Bu adımdan sonra, her aday kümesinin desteği hesaplanır. Destek seviyeleri minimum destek seviyesine eşit veya bu seviyeden büyük çıkanlar geniş nesne kümesi olarak işaretlenirler. Bir sonraki taramada bu geniş işareti taşıyan kümeler, yukarıda anlatıldığı gibi bir sonraki aday kümelerin belirlenmesi için kullanılır [Agrawal, 1993].

## 1.8.2 SETM Algoritması

Bu algoritmada, $L_k$ geniş nesne kümesinin her bir elemanı iki parametreden oluşur; bunların bir tanesi nesnenin ismiyken diğeri bu nesneyi ayırt etmeye yarayan bir özellik numarasıdır. Algoritma içinde bu numara TID (Tansaction Identification) olarak kullanılır. Benzer şekilde her bir aday nesne kümeleri,

$C_k$, <TID, Nesne Kümesi İsmi>

formatında tutulur. [Houtsma ve Swami, 1993]

AIS algoritmasında olduğu gibi SETM algoritması da veritabanını birçok kez tarar. İlk tarama esnasında veritabanındaki nesneleri, teker teker sayarak hangilerinin geniş nesneler olduğunu belirler. Sonraki taramalarda, bir önceki geçişte geniş olarak işaretlenmiş nesne kümelerini kullanarak aday kümeleri belirler. Farklı olarak,

SETM algoritması aday kümelerle birlikte üzerinde çalışılan işlemin TID bilgisini de tutar. Bundan sonra, aday nesne kümeleri nesne ismine göre sıraya dizilir ve küçük nesne kümelen silinir. Eğer veritabanı TID numarasına göre sıralanmışsa, bir sonraki tarama esnasında herhangi bir işlemdeki geniş nesne kümeleri $L_k$ 'nin TID

numarasına göre sıralanmasıyla elde edilir. Bu şekilde veritabanı bir kaç kez taranır. Artık başka herhangi bir geniş nesne kümesi bulunamadığında algoritma sonlandırılır.

SETM algoritmasında TID bilgisinin de tutulması, algoritmanın yer karmaşıklığını arttıracaktır, bu dezavantajın dışında başka bir eksi nokta ise, aday nesne kümesinin desteği hesaplanırken $C_k$ sıralanmış halde değildir, bunun için nesne kümelerinin bir

kez daha sıraya dizilmesi gerekecektir. Bu da zaman karmaşıklığını arttıran bir unsurdur.

### 1.8.3 Apriori Algoritması

Apriori algoritması bağlantı analizlerinin yapılıp bağlantı kurallarının ortaya çıkartılması konusunda en çok bilinen ve kullanılan algoritmadır. Geniş nesne kümelerinin ortaya çıkartılması işlemleri için kullanılır.

Geniş nesne kümelerini ortaya çıkartan algoritmalar eldeki tüm verileri birçok kez tararlar. İlk taramada, her bir nesnenin destek seviyesi, hesaplanarak kullanıcı tarafından başlangıçta girilen minimum destek seviyesi ile karşılaştırılır ve her bir nesnenin geniş olup olmadığına bakılır. Bundan sonraki her tarama bir önceki taramada geniş olarak tespit edilmiş nesnelerden başlar ve geniş nesne kümeleri oluşturulur. Bu geniş nesne kümelerine aday nesne kümeleri denir. Taramanın sonunda ise hangi aday nesne kümesinin gerçekten geniş olduğu kontrol edilir. Daha önce de belirtildiği gibi bir nesne kümesinin geniş olarak adlandırılabilmesi için o nesne kümesinin kullanıcı tarafından verilen minimum destek seviyesinin üzerine bir destek seviyesine sahip olması gerekir. Bir sonraki taramada, yine bir önceki taramada geniş olarak seçilen nesne kümelerinden başlanır ve veritabanın sonuna kadar bu nesne kümelerinin destekleri hesaplanır. Bu işlem, başka yeni geniş nesne kümeleri bulunamayana kadar sürer. [Agrawal ve Spirant, 1994]

Apriori algoritması daha önceden ortaya atılmış olan AlS ve SETM algoritmalarından her bir geçişte aday nesne kümelerinin sayılma ve bu aday kümelerinin üretilme şekliyle ayrılır. Hem AIS algoritmasında hem de SETM algoritmasında, tarama esnasında, veriler okunurken aday nesne kümeleri üretilir. Bir işlem (T) (tansaction) okunduktan sonra, geniş nesne kümelerinin bu işlemlerde olup olmadığına da bakılır. Yeni aday nesne kümelerinin üretilmesi ise işlemlerdeki diğer nesnelerle elde edilen geniş nesne kümelerinin birleştirilmesiyle üretilir [Agrawal, i993]. Tabii bu da, gereksiz yere, aslında küçük nesne kümesi olan birçok aday nesne kümesinin sanki geniş nesne kümesiymiş gibi üretilmesi ve sayılması sonucunu doğurur. Bu da algoritmanın zaman karmaşıklığını arttırır.

Apriori algoritması ise aday nesneleri üretirken veritabanındaki işlemleri hiç işin içine sokmadan, yalnızca bir önceki taramada geniş olduğu tespit edilmiş nesne

kümelerini kullanarak oluşturur. Apriori algoritması geniş bir nesne kümesinin herhangi bir alt kümesinin de geniş olacağı kabulüne dayanır. Böylece k adet nesneden oluşmuş bir nesne kümesi, k-\ adet nesneye sahip geniş nesne kümelerinin birleştirilmesi ve alt kümeleri geniş olmayanların silinmesiyle elde edilebilir. Bu birleşme ve silme işlemi sonunda daha az sayıda aday nesne kümeleri oluşacaktır.

Agrawal ve Srikant tarafından geliştirilen Apriori algoritması 1994 yılında 20. VLDB (Very Large Database Endowment) konferansında sunulmuştur. Bu bildiride, Agrawal ve Srikant algoritmanın çalışma ayrıntılarını ve algoritmanın kaba kodunu şu şekilde sunar [Agrawal ve Srikant, 1994].

- Verilerin ilk taranması esnasında, geniş nesne kümelerinin tespiti için, tüm nesneler sayılır.

- Bir sonraki tarama, k'ıncı tarama olsun, iki aşamadan oluşur.

- Apriori-gen fonksiyonu kullanılarak, (k-l)'inci taramada elde edilen, $L_{k-1}$, nesne kümeleriyle, $C_k$ aday nesne kümeleri oluşturulur.

- Sonra veritabanı taranarak, $C_k$ 'daki adayların desteği sayılır.

- Hızlı bir sayım için, verilen bir *l* işlemindeki, $C_k$ 'yı oluşturan adayların çok iyi belirlenmesi gerekir.

### 1.8.4 AprioriTid Algoritması

Önceki ayrıtta belirtildiği gibi, algoritmalar desteği hesaplamak için tüm veritabanını tarar; ancak her aşamada veritabanın tamamının taranmasına gerek olmayabilir. Bu yaklaşımla Agrawal, Apriori algoritmasıyla birlikte AprioriTid algoritmasını da sunmuştur.

AprioriTid algoritması da taramadan önce aday nesne kümelerini belirlemek için Şekil-4.2 de görülen apriori-gen fonksiyonunu kullanır. Apriori-gen en büyük farkı ilk geçişten sonra veritabanının destek seviyesini bulmak için taranmamasıdır. Bu iş için $C_k$ kullanılır. SETM algoritmasında olduğu gibi $C_k$ 'nin her elemanı <TID,

$\{X_k\}$> formundadır. Burada $X_k$, TID numaralı işlemde bulunan potansiyel geniş k nesne kümesidir, k = 1 iken C, veritabanına karşılık gelir. Bununla beraber her nesne, nesne kümesiyle yer değiştir, k > 1 olduğu durumlarda $C_k$ algoritmanın onuncu adımında olduğu gibi üretilir, t işlemindeki $C_k$ bir elemanı <TID, c> şeklindedir. Burada c, t işlemindeki $C_k$ ya ait bir aday elemanıdır, {c e $C_k$ \c}. Eğer bir işlemin,

herhangi bir k nesne kümesi adayı yoksa bu durumda $C_k$ nm bu işlem için herhangi bir girdisi, elemanı olmayacaktır. Daha doğrusu bu işlemin TID numarasını taşımıyor olacaktır. Böylece $C_k$ 'daki girdi sayısı, özellikle bu k değerleri için, veritabanındaki işlem sayısından daha küçük olabilir. Bunun dışında yine büyük k değerleri için her girdi kendisine karşılık gelen işlemden daha küçük olabilir. Çünkü o işlemde çok az sayıda aday barınıyor olabilir. Ancak, küçük k değerleri için bunun tersi olacaktır; yani girdiler kendilerine karşılık gelen işlemlerden daha büyük olabileceklerdir [Agrawal ve Srikant, 1994].

# 2. BÖLÜM

## 2.1 Kümeleme Analizi

Kümeleme analizi, sınıflandırmada olduğu gibi sahip olunan verileri gruplara ayırma işlemidir. Kümeleme yabancı kaynaklarda clustering ya da segmentation olarak adlandırılmaktadır. Sınıflandırma işleminde, sınıflar önceden belirli iken kümelemede sınıflar önceden belirli değildir. Verilerin hangi gruplara/kümelere, hatta kaç değişik gruba ayrılacağı eldeki verilerin birbirlerine olan benzerliğine göre belirlenir. Belirlenen her bir gruba da küme ismi verilir. Kümeleme analizi biyoloji, tıp, antropoloji, pazarlama, ekonomi ve telekominasyon gibi birçok ve birbirinden çok farklı alanlarda kullanılmaktadır [Dunham, 2003].

Örneğin elimizdeki bir perakende mağazasına ait veritabanında, müşterilerimizin sadece isim ve yaşlarının tutulduğunu varsayalım. Bu durumda, müşterilerimizi kümelere ayırmak istersek, onları yaşlarına göre ayırmamız doğru olacaktır. Dolayısıyla yaşları nispeten birbirlerine yakın olanlar aynı kümede toplanacaktır. Yaşları 20, 22, 26, 27, 40, 45, 46, 47, 49 ve 49 olan 10 müşterimiz varsa, 20-27 yaşları arasında olanlar birbirlerine, veritabanındaki diğer kişilere göre daha yakın olduğundan bir kümede toplanırken, yaşlan 40-49 arası olanlar ise bir başka kümede toplanacaktır. Oysa veritabanındaki yaşlar 19, 20, 21, 21, 21, 26, 26, 26, 27, 27 ve 28 şeklinde olsaydı, 19 -21 arası bir kümede, 26 -28 arasın da bir küme olması daha anlamlı olacaktı. Bu durumda, bir önceki örnekte 20 yaşındaki bir müşteriyle, 27 yaşındaki bir müşteri aynı kümede tutulurken, ikinci örnekte bunlar ayrı kümelerde bulunacaklardır. Buradan da görüldüğü gibi veriler sadece taşıdıkları değerlere göre değil, diğer verilerle olan benzerliğe ve veritabanındaki diğer verilerin durumuna göre de kümelere ayrılıyorlar. Bu durumda kümeleme sonuçları dinamiktir diyebiliriz. Bu da kümelemeyi sınıflandırmadan ayıran bir başka özelliktir.

Kümelemenin matematiksel tanımı ise şu şekilde yapılabilir:

Elimizde $D = \{X_1, X_2, X_3, \ldots, X_n\}$, n= 1,2, ..., m veritabanı olsun, her bir $X_n$ bir kaydı temsil etsin. $X = \{x_1, x_2, x_3, \ldots, x_i\}$ , i = 1,2,...,m her bir $x_i$, ad, soyad, yaş ve

gelir gibi özellikler olsun. Kümelemedeki amaç D veritabanını, j adet K kümesine bölmek ve $K_j \subseteq D$ koşulunun sağlanmasıdır.

Daha önce belirtildiği gibi kümeleme analizinde, sınıflandırmadan farklı olarak belirlenecek kümelerin özellikleri önceden bilinmemektedir ve üstelik ortaya çıkacak küme sayısı da belirli değildir. Ancak, algoritmaların zaman karmaşıklığını, alınacak sonuçların kullanılabilirliğini artırmak için, literatürdeki algoritmaların bir kısmı ya küme sayısını ya da her bir kümede bulunacak eleman veya bu elemanlar arasındaki minimum - maksimum benzerlik uzaklık ölçütünü kullanıcıdan ister.

### 2.1.1 Benzerlik ve Uzaklık

Veritabanındaki veriler kümelere ayrılırken, benzerlik ve uzaklık kavramlarından yararlanılır. Bu, veritabanındaki her bir kaydın diğer bir kayıtla olan benzerliği ya da her bir kaydın veritabanındaki diğer kayıtlardan olan uzaklığı olduğu gibi oluşturulan gerçek ve aday kümeler arasındaki mesafe ve benzerliği de içerir. Sözgelimi, veriler birbirlerine olan uzaklığa göre başlangıçta 8 ayrı kümeye ayrıldılarsa, bu 8 ayrı kümenin gerçekten farklı özelliklere sahip birer küme olup olmadığının da belirlenmesi gerekecektir. Bu durumda, oluşturulmuş bu kümeler arasındaki mesafe / benzerlik de ölçülmelidir. Birbirlerinden pek de farklı olmayan kümeler birleştirilerek tek bir küme haline dönüştürülebilir. Bu işlem, tüm veriler taranıp kümeler ortaya çıktıktan sonra yapılabileceği gibi veritabanın taranması ve veriler arasındaki benzerlik ve mesafenin ölçümü esnasında da yapılabilir. Bu nedenle kümeler arasındaki mesafenin ölçülmesiyle iki veya daha fazla kümenin birleştirilmesi söz konusu olduğu gibi aynı zamanda, bir kümeden birden fazla küme üretilmesi de söz konusu olacaktır. Bunun için de sürekli olarak kümelerin büyüklüğü ve çapı ölçülmelidir.

D olarak göstereceğimiz bir veritabanındaki

$D = \{X_1, X_2, X_3, \ldots, X_n\}$, n = 1, 2 ... m,

$X_m$ ile $X_j$ arasındaki mesafe, $mes(X_n, X_j)$

Euclid uzayında, şu şekilde hesaplanır:

$$\text{ben}(X_m, X_j)_{\text{DICE}} = \frac{2\sum_{i=1}^{n} x_{mi}x_{ji}}{\sum_{i=1}^{n} x_{mi}^2 + \sum_{i=1}^{n} x_{ji}^2}$$

bu mesafeye Euclid mesafesi denir [Eğecioğ1u, 2001].

Benzerlik kavramı ise mesafenin tersi bir anlam içerir ve iki veri arasındaki yakınlığı

$$ben(X_m, X_j) = \frac{1}{1 + mes(X_m, X_j)}$$

şeklinde ifade edilebilir.

Bununla beraber, benzerlik ölçümü için çeşitli yöntemler vardır [Rasınussen, 1992] [Bacher]. Bunlardan birincisi Dice benzerlik ölçümüdür. ben($X_n$, $X_j$), $X_n$ ile $X_j$ arasındaki benzerlik Dice benzerlik ölçüm yöntemiyle şu şekilde hesaplanır.

$$\text{ben}(X_m, X_j)_{\text{DICE}} = \frac{2\sum_{i=1}^{n} x_{mi}x_{ji}}{\sum_{i=1}^{n} x_{mi}^2 + \sum_{i=1}^{n} x_{ji}^2}$$

Aynı benzerlik Jaccard ile şu şekilde hesaplanır;

$$\text{ben}(X_m, X_j)_{\text{JACCARD}} = \frac{\sum_{i=1}^{n} x_{mi}x_{ji}}{\sum_{i=1}^{n} x_{mi}^2 + \sum_{i=1}^{n} x_{ji}^2 - \sum_{i=1}^{n} x_{mi}x_{ji}}$$

Başka bir benzerlik ölçümü ise Cosine'dir. Cosine yöntemi ile aşağıdaki gibi hesaplanır;

$$\text{ben}(X_m, X_j)_{\text{COSINE}} = \frac{\sum_{i=1}^{n} x_{mi} x_{ji}}{\sqrt{\sum_{i=1}^{n} x_{mi}^2 \sum_{i=1}^{n} x_{ji}^2}}$$

Bu yöntemlerin dışında, benzerlik ölçümünde kullanılan bir başka yöntem ise Overlap'tir.

$$\text{ben}(X_m, X_j)_{\text{OVERLAP}} = \frac{\sum_{i=1}^{n} x_{mi} x_{ji}}{\min\left(\sum_{i=1}^{n} x_{mi}^2, \sum_{i=1}^{n} x_{ji}^2\right)}$$

### 2.1.2 Kümeleme Analizinin Sınıflandırılması

Literatürde birçok kümeleme algoritmasının adı geçmektedir. Algoritmalar birbirlerinden, kümelemenin oluşturuluş şekline göre ayrıldıkları gibi kullanılan yeri türüne, yapılacak olan çalışmanın amacına göre de farklılıklar gösterirler [Han ve Kamber, 2001]. Kümeleme algoritmaları, genel olarak hiyerarşik ye bölümlemeli olarak ikiye ayrılırken, bu konuda yapılmış bir literatür taraması bu algoritmaların daha alt bölümlere ayrılabileceğini göstermektedir [Berkhin].

• Hiyerarşik Yöntemler

• Toplaşım (Agglomerative) Kümeleme Algoritmaları

• Bölünür (Dizsiye) Kümeleme Algoritmaları

• Bölümlemeli (Partitioning) Yöntemler

• Yer değiştiren Algoritmalar

• Olasılıksal Algoritmalar

• K-Medoid Yöntemler

• K-Means Yöntemler

• Yoğunluğa Dayalı Algoritmalar

- Yoğunluğa Dayalı Bağlantılı Kümeleme

- Yoğunluk fonksiyonlu kümeleme

• Grid Temelli Yöntemler

• Kategorik Verinin Yinelenmesine Dayanan Yöntemler

• Kısıtlara Dayanan Yöntemler

• Makine Öğrenmesi Alanında Kullanılan Yöntemler

• Gradient İnme ye Yapay Sinir Ağları

• Ölçeklenebilir Kümeleme Yöntemleri

### 2.1.3 Hiyerarşik Yöntemler

Hiyerarşik yöntemler bir küme ağacı yaratır. Bu küme ağacındaki her bir düğüm oğullara sahiptir; ağaç yapraklarla son bulur. Aşağıdan yukarıya, toplaşım kümeleme

algoritmaları ve yukarıdan aşağıya bölünür kümeleme algoritmaları olarak iki grupta toplanabilir.

Toplaşım kümeleme algoritmaları, başlangıçta veritabanındaki her bir noktayı bir küme olarak görür. Bu kümeleri birleştire birleştire birbirinden ayrı kümeler oluşturur.

Bölünür kümeleme algoritmaları ise başlangıçta veritabanındaki tüm noktaları tek bir kümeymiş gibi görür. Veritabanını taradıkça, birbirine benzemeyen noktaları kümeden dışarı atarak önceden verilmiş, k kadar kümeye dağıtır. Hiyerarşik kümeleme algoritmaları benzerlik ve mesafe ölçütlerini kullandıkları için kullanılması kolay, hemen hemen her türlü veri türüne uygun ve esnek algoritmalardır; ancak özellikle bölünür algoritmalar için k küme sayısının verilmesi bir dezavantajdır.

Başka bir dezavantaj ise bu kategorideki algoritmalar bir kümeyi oluşturduktan sonra, yapıları gereği oluşturulan bu kümeyi bir daha kontrol etmezler. Bu algoritmalar yukarıda sözü edilen bağlantı ölçümlerini, kümeler arası mesafe ölçümü kullanırlar. Birçok algoritma N x N bir mesafe ya da benzerlik matrisi çıkartarak, kümelemeyi bu matrise dayanarak yapar. Bu matris, her bir elemanın diğer elemanlarla olan benzerliğini veya aralarındaki mesafeyi gösteren matristir.

Kümeleme analizinde, algoritmaların zaman ve yer karmaşıklığını en çok artıran unsur da bu mesafe/benzerlik matrisidir. Kümeleme analizi için geliştirilen algoritmalar şu ya da bu şekilde söz konusu mesafe/benzerlik matrisini, özellikle yer karmaşıklığını azaltmak için bellekte tutmadan ya da kısaltarak tutarak kümeleme işlemlerini yerine getirirler. Örneğin, belirli bir eşik değerinin altındaki benzerliklerin atlanması, kullanılan bilgisayarın belleğini alan karmaşıklığı açısından rahatlatacaktır [Olson, 1993].

### 2.1.3.1 SLINK Algoritması ve Tek Bağlantı Tekniği

Slink algoritması tek bağlantı ya da en yakın komşu tekniğini kullanır [Sibson, 1973]. Bu teknikte, kümeler arasındaki mesafe ölçülürken, iki küme içinde birbirine en yakın iki elemanın uzaklığı ya da başka bir deyişle iki kümeyi en yakın kılan elemanların mesafesi kümeler arası mesafe olarak kabul edilir. Algoritmanın zaman karmaşıklığı $0(n^2)$'dir.

Öncelikle eldeki verilerin, mesafe/benzerlik matrisi çıkartılır; bu matrisi bir ağaç haline dönüştürür. Şebeke modellerinden en küçük maliyetli ağaç çıkartılarak, verilen eşik değerine göre kümeler oluşturulur.

### 2.1.3.2 CURE Algoritması

Kümele işlemi yapılırken, oluşturulan kümelerin kalitesini en çok etkileyen faktör, ana veri topluluğu içinde diğer verilerden uzakta bulunan ve sayıları az olup aslında hiç bir kümeye ait olmaması gereken uç verilerdir. CURE (Clustering Using Reprisentatives- Temsilciler Kullanarak Kümeleme) algoritması bu uç verilerin oluşturulan kümelerin kalitesini etkilememesi düşüncesiyle 1998 yılında geliştirilmiş bir algoritmadır. Küresel bir geometrik şekil taşımayan veri gruplarının kümelenmesi için oldukça elverişli bir algoritmadır.

CURE algoritması öncelikle her girdiyi sanki ayrı bir kümeymiş gibi ele alır ve her adımda bu küme temsilcilerin birbirlerine olan yakınlıklarına göre ya birleştirir ya da ayrı kümeler olarak tutar. Öncelikle her bir küme için c adet iyi dağıtılmış temsilci nokta seçilir. Seçilen bu noktalar kümelerin fiziksel şeklini geometrik özelliğini ortaya koyar. Daha sonra bu dağıtılmış noktalar bir a katsayısıyla kümenin ortasına, merkezine doğru kaydırılır. Dağıtılmış olan noktalar bu kaydırma işleminden sonra artık o kümenin temsilcileri olarak kabul edilirler. Bundan sonra iki küme arasındaki uzaklık, her biri bir kümeye ait olan en yakın temsilci çifti arasındaki uzaklıktır.

Temsilcilerin bir a katsayısıyla kümenin merkezine kaydırılması kümedeki yüzey anomalilerini tolere ettiği gibi uç verilerin etkisini de azaltır. çünkü uç veriler tipik bir şekilde merkezden uzakta yer alırlar ve sonuç olarak da bu veriler merkeze doğru daha fazla hareket etmiş olacaklardır. Bu uç verilerin uzun mesafeli hareketleri farklı iki kümenin birleştirilmesini önleyecektir [Guha, 1998]. Kullanılan α katsayısı aynı zamanda, oluşan kümelerin şeklini belirlemede de kullanılabilir. α'nın alacağı değer 0-1 arasındadır. Küçük değerli α dağılmış noktaların çok az yer değiştirmesine neden olurken kümelerin de şekilsel olarak uzunlaşmasına yol açar. α değerinin büyük olması ise dağılmış noktaları küme merkezine oldukça yaklaştıracağı için daha toplu halde kümeler oluşacaktır. α = 1 durumunda ise CURE algoritması merkezi temelli algoritmalara yaklaşacaktır.

CURE algoritmasının en kötü durumdaki zaman karmaşıklığı $0(n^2 1og\ n)$'dir. Bununla beraber, Guha 1997'de hazırlamış olduğu bir teknik raporda, veri noktalarının boyutu küçük olduğunda bu zaman karmaşıklığının $0(n^2)$ olduğunu söyler [Guha, 1997]. Algoritmanın alan/yer karmaşıklığı ise O(n)'dir.

Bunun dışında, algoritmanın bellekte daha az yer işgal etmesini sağlamak için tüm veriler üzerine algoritma koşturulmadan önce, ana kümeden belirli bir miktarda örnek alınarak CURE algoritması bu örnek küme üzerinde uygulanır. Rastgele yapılan bu örnekleme oluşturulacak kümelerin kalitesini arttırmaktadır.

### 2.1.3.3 CHAMELEON Algoritması

Chameleon algoritması ilk olarak 1999 yılında Karypis ve arkadaşları tarafından geliştirilmiş bir algoritmadır [Karypis, 1999]. Chameleon algoritması, iki küme arasındaki benzerliği dinamik bir model kullanarak belirler. Diğer algoritmalardan farklı olarak iki alt kümenin birbirine olan benzerliği ve yakınlığı bu iki kümeden her birinin kendi iç benzerlikleri ve yakınlıkları ile kıyaslanarak belirlenir ve bu karşılaştırma sonucunda bu iki alt küme birbirine yakınsa birleştirilir. Bu sayede daha kaliteli ve homojen kümeler yaratılmış olunur. Benzerlik / mesafe matrisinin oluşturulabildiği tüm veri türleri ve veri kümeleri için uygulanabilecek bir algoritmadır.

Daha önce incelenen SLINK, CURE ve ileride incelenecek olan DBSCAN ve K-means vs. gibi algoritmalar bazı veri kümelerinin tespit edilmesinde yanılabilmektedirler.

### 2.1.3.4 BIRCH

BlRCH, çok büyük veritabanlarının kümelenmesi için geliştirilmiş bir algoritmadır. Ayrıca gürültülü verilerin kontrol edilmesi için bu alanda öne sürülen ilk algoritmadır [Zhang, 1996].

Çok büyük veritabanlarının kümelere ayrılması için tutulması gereken NxN matrisini bilgisayar belleği açısından maliyeti, başka bir deyişle alan karmaşıklığı çok yüksek olacaktır. BIRCH algoritması bilgisayar belleğinde daha az yer kaplayan bir tekniğe sahip hiyerarşik yapıda bir kümeleme algoritmasıdır. Temel olarak, kümelemenin yapılabilmesi için bir ağaç oluşturulur ve gerekli tüm bilgilere haiz bu ağaç taranarak kümeleme işlemleri gerçekleştirilir. Kümeleme, her bir düğümde mesafe ölçümleri için gerekli bilgilerin tutulduğu bu ağaç üzerinde gerçekleştirilir.

Kümeleme özellikleri diye adlandırılan ve kümeler hakkındaki bazı bilgileri içeren "bir ağaçtan" yararlanılması BIRCH algoritmasının en tipik özelliğidir. Ancak, sadece sayısal veriler üzerinde kullanılabilir.

Yukarıda sözü edilen bu ağaç CF ağacı olarak adlandırılır ve bu ağaç üç bilgiyi tutar ve algoritmanın koşturulması esnasında bu bilgileri sürekli olarak günceller. Bu bilgiler:

N: Kümede bulunan nokta sayısı

LS: Kümedeki noktaların değerlerinin toplamı

SS: Kümedeki noktaların değerlerinin karelerinin toplamıdır.

CF ağacı her bir düğümün alabileceği düğüm sayısı belirli olan dengeli bir ağaçtır. Her bir düğüm kendisine bağlı olan alt düğümlerle ilgili CF değerlerini (N, LS, SS) tutar. Ağacın her bir yaprak düğümü ise bir kümeyi temsil eder. Dendrogamda kullanılan tekniğin tam tersi olarak CF ağacı yukarıdan aşağı doğru çalışır yani toplaşım algoritması değil, hiyerarşik fakat bölünür bir kümeleme algoritmasıdır. Ağaca yeni noktalar eklendikçe CF ağacı yaratılmış olur; her bir nokta kendisine en yakın olan yaprağa bağlanır ve yaprakların büyüklüğü (T eşik değeri) daha önceden algoritmaya verilmelidir. Noktalar eklene eklene büyüyen yaprak T eşik değerini aşarsa, ağaçta dengeleme veya bölme işlemi yapılır. Buradaki T eşik değeri aslında yaprağın çapıdır. Büyük T değerleri küçük ağaçların oluşmasına neden olurken, T değeri azaldıkça ağacın büyüklüğü de artacaktır. Büyük ağaçlar bellekte daha fazla yer tutacağı için T değeriyle oynanarak bilgisayarın belleğindeki yer karmaşıklığı ayarlanabilir. Bu işlemler yapılırken, veritabanının birden fazla defa okunup taranmasına gerek yoktur; dolayısıyla algoritmanın zaman karmaşıklığı O(n)'dir.

CF ağacının oluşturulması;

Veritabanımız D= $\{t_1, t_2, \dots, t_i\}$, i=1, 2, … ,n olsun

$t_i$ elemanının ekleneceği yaprağı bul

Eşik değeri aşılmamışsa, bu yaprağa (kümeye) $t_i$ ekle

CF özelliklerini yeniden hesapla

aksi takdirde;

yeterli yer varsa $t_i$ 'yi ayrı bir yaprak olarak ata

yoksa

yaprağı ikiye böl ve $t_i$ 'yi uygun olana ekle.

CF ağacının oluşturulmasından sonra tam kümeleme işlemine geçilir. CF ağacında bazı kısıtlar oluşacağından elde edilen kümeler doğal kümeler olmayabilir. Bu nedenle eldeki kümeler (tercihen merkezci yaklaşıma uygun) başka bir algoritmayla birleştirilir. Gerekirse bu işlemden sonra ikinci bir kümeleme daha yapılabilir.

### 2.1.4 Bölümlemeli Yöntemler

Bölümlemeli yöntemlerde n adet nokta önceden verilen k küme sayısına ($k < n$) göre kümelere ayrılır. Hiyerarşik yöntemlerin tersine kullanıcı tarafından verilen bazı kriterlere uygun kümeler yaratılırken, yaratılacak küme sayısı önceden belirlidir. Kullanıcı algoritmaya kümeler arasındaki minimum/maksimum mesafeyi ve kümelerin iç benzerlik kriterlerini de vermek zorundadır [Giudici, 2004].

Bölümlemeli algoritmalar genel olarak hiyerarşik algoritmalardan daha hızlı çalışırlar; çünkü hiyerarşik algoritmalardaki gibi bir benzerlik/mesafe matrisi kullanmak zorunda değillerdir. Bundan dolayı da büyük veritabanlarının kümelenmesinde hiyerarşik yöntemlere göre daha uygundur. Bununla beraber önceden verilen kritere uygun birden fazla sonuç çıkarmak mümkün olabilir. Bu durumda algoritmanın gerçekten en uygun çözümü bulup bulamadığı ise hiç bir zaman bilinmeyecektir [Dunham, 2003]. Bunun öğrenilebilmesi için verilerin dağıtılarak, sıra ve yerleri değiştirilerek, algoritmanın tekrar koşturulması gerekecek ve çıkan sonuçların birbirleriyle kıyaslanması gerekecektir. Bu da zaman maliyetini oldukça artıracaktır.

#### 2.1.4.1 K- Ortalama (K-Means) Algoritması

K-Ortalama algoritması sürekli olarak kümelerin yenilendiği ve en uygun çözüme ulaşana kadar devam eden döngüsel bir algoritmadır. Bölümlemeli algoritmaların tipik özelliklerini taşır. Bu alandaki benzer algoritmaların çoğu ya K-Ortalama algoritmasından esinlenerek ya da bu algoritmanın geliştirilmesiyle ortaya çıkmıştır. Dolayısıyla bu algoritmanın anlaşılması bundaki sonraki algoritmaların mantığının kavranmasında önemli bir rol oynayacaktır [Han Jiawei ve Kamber Micheline, 2001].

İlk olarak 1967 yılında ortaya atılan [MacQueen, 1967] K-Ortalama algoritması eldeki verileri k adet kümede ve kümelerin ortalamalarına göre kümelere ayırır. k küme sayısı kullanıcı tarafından verilir. Burada kastedilen ortalama daha önce belirtilen küme merkezidir.

Girdiler:

D= $\{t_1, t_2, ... ,t_n\}$ // eldeki veritabanı

K // verilen küme sayısı

Algoritma:

Keyfi olarak $m_1, m_2, \dots ,m_k$ ortalama belirle.

her bir $t_i$'yi en yakın olduğu $m_i$'nin kümesine ata.

Kümelere ait $m_1, m_2, \dots ,m_k$ değerlerini yeniden hesapla.

Küme elemanlarında herhangi bir değişiklik yoksa dur.

ilk adımdan itibaren tekrar et.

Çıktı:

K adet küme

### 2.1.4.2 PAM Algoritması

PAM algoritması k adet kümeyi bulmak için seçilen temsilcilerin etrafına ana kümedeki tüm elemanları toplayarak ve her defasında bu temsilcileri değiştirerek kümeleme işlemini tamamlar.

PAM algoritmasının temsilci olarak seçtiği noktaya medoid denilir; dolayısıyla bu algoritma k-medoid algoritması olarak da anılır. Bu temsilci (medoid) seçiminden kasıt ise kümenin merkezine yakın mesafede bulunan noktanın belirlenmesidir. K adet küme için seçilen k adet temsilci belirlendikten sonra, veritabanındaki temsilci olmayan diğer noktalar (veriler) kendilerine en çok benzeyen temsilcinin etrafında toplanır. Daha matematiksel bir ifadeyle, eğer $t_i$ bir temsilci ve $t_j$ ise temsilci olmayan bir başka nokta olsun, eğer $d(t_i, t_j)= \min_{te} d(t_j, t_e)$ ise $t_j$, $t_i$ tarafından temsil edilen kümeye aittir. Burada $\min_{te}$ tüm temsilciler içindeki en küçüğü ifade ederken, $d(t_a, t_b)$ ise $t_a$ ve $t_b$ noktaları arasındaki mesafe veya benzeşmezliği ifade etmektedir. Bu durumda bir kümenin kalitesi o kümedeki temsilciyle diğer noktalar arasındaki ortalama mesafe ya da ortalama benzememe değeriyle ölçülebilir. PAM algoritmasında tüm benzerlik / mesafe ölçütleri kullanılabilir [Raymond ve Han, 1994].

### 2.1.4.3 CLARA Algoritması

CLARA algoritması bütün veritabanın tarayarak temsilci noktalar seçmek yerine, veritabanından rastgele bir örnek kümeyi alarak, PAM algoritmasını bu örnek küme üzerine uygular. Bu uygulama sonucunda oluşacak olan kümelerin her birinin temsilcisi belirlenir. Daha sonra ana kümeyi oluşturan veritabanından bir örnek küme daha seçilir. Bu esnada ilk temsilcilerin rastgele seçilmesi yerine bir önceki aşamada belirlenmiş temsilciler kullanılır. Bu da algoritma içinde temsilci değişimini azaltacak ve algoritma hem daha hızlı bir şekilde işleyecek hem de daha kaliteli sonuçlar verecektir. Bu tekrar örnekleme işleminin 5 defa yinelenmesi ve her defasında 40 + 2k adet örnek seçilmesinin en iyi sonucu verdiği Kaufman ve Rousseeuew (1990) tarafından rapor edilmiştir.

PAM algoritmasıyla kıyaslandığında CLARA algoritmasının daha geniş veri tabanlarında güvenli bir şekilde çalışabildiği belirlenmiştir; 10 kümede 1000 eleman gibi.

### 2.1.4.4 CLARANS Algoritması

CLARANS algoritması verilen n adet nesnenin temsilciler aracılığıyla ve bir şebeke diyagramından yararlanılarak k adet kümeye ayrılması şeklinde özetlenebilir.

$G_{n,k}$ ile temsil edilen bu şebeke diyagramında her bir düğüm $\{O_1, O_2, \dots, O_k\}$'den oluşan k adet nesneyi temsil eder. $O_1, O_2, \dots, O_k$ bir bakıma temsilcilerdir. İki düğüm birbirinden sadece bir nesnenin değişimiyle ayrılıyorsa bu iki düğüm komşu olarak kabul edilir.

CLARANS algoritmasının iki parametresi vardır: maks-komşu (maxneighbor) ve yerel-miktarı (numlocal). Maks-komşu parametresi incelenecek komşu sayısının üst limitini ifade ederken, yerel-miktarı ise elde edilecek yerel minimum nokta sayısının alt sınırını ifade etmektedir.

# 3. BÖLÜM

## 3.1 Çalışmanın Konusu

Bu tez çalışmasında, eşitlik ayırıcı (equal-split) olarak adlanlandırılan parametre kullanılarak kategorisel verilere sahip bir veri tabanı üzerinde kümeleme algoritması gerçekleştirilmiştir.

Temel işlem, veri tabanındaki herhangi bir alan içindeki veri sayısı ve değişken adedinden yararlanmaktadır. Bu parametre sayesinde hangi alan ya da alanların veri tabanını mümkün olan en az parçaya ayırabileceği tespit edilmektedir.

EP: Eşitlik-Ayrıcı Parametre (Equal-Split Parameter).

Herbir değişkenin merkeze uzaklığının mutlak toplamı,

N: Toplam veri sayısı,

NV: Her alan içindeki değişken sayısı,

$NV_i$: Her alan içindeki değişkenden kaç adet olduğu,

CA: Değişken merkezi,

$$\mathrm{CA} = \frac{N}{NV}$$

$$\mathrm{EP} = \sum_{i=1}^{n} \left| CA - NV_i \right|$$

## 3.2 Çalışmanın Amacı

Bu çalışmanın ilk amacı, öne sürülen algoritmanın uygulanabiliriliğini denetlemektir. İkinci amaç ise mevcut yöntemlerle arasındaki benzerlik ve farklılıkarı ortaya koymaktır. Karmaşıklığı O(n) olan bu algoritma için aynı tür verilerle işlem yapabilen diğer algoritmalarla karşılaştırma yapılmıştır.

## 3.3 Çalışmaya Konu Olan Algoritma

| | |
|---|---|
| Begin<br>Calculate Area Count<br>mlEP:=-1<br>Loop for All Areas<br>Do Begin<br>N:= Total Data Count (for current area)<br>NV:= Total Variable count (for current area)<br>CA:= $\frac{N}{NV}$<br>EP = $\sum_{i=1}^{n} \lvert CA - NV_i \rvert$<br>If mlEP>EP then mlEP:=EP<br>End;<br>End. | A1) Alan sayısını hesapla. Calculate Area Count<br>A2) Sıradaki alana geç.<br>A3) Toplam veri sayısını hesapla (N) .<br>A4) Toplam değişken sayısını bul (NV) .<br>A5) Merkezi hesapla (CA = $\frac{N}{NV}$ ) .<br>A6) Her bir değişkenin merkeze olan uzaklığının mutlak toplamını hesapla<br>EP = $\sum_{i=1}^{n} \lvert CA - NV_i \rvert$ .<br>A7) En küçük değerli değişkeni kök olarak seç.<br>A8) Değişken değerlerinden birbirine eşit olan varsa herhangi birini seç.<br>A9) İşlemi tekrarlamak için A2. adıma dön.<br>A10) Tüm veriler aynı olduğunda yaprağa ulaşılmış demektir.<br>A11) Dur. |

Kümeleme işleminde ; Veriler oluşturulan ağaca yayılacaktır. Her düğümdeki veri miktarı, verilerin merkezi, standard sapma, çap, yarıçap özelliklerinin kriter olarak seçilenleri ya da istenilenleri hesaplanır. Örneğin kazanç (Gain) ya da Gini indeksi (Gini index) işlemlerinde olduğu gibi. Eşik değerleri baştan verildiğinde budama işlemi yapılabilir. Örneğin bir sonraki düğüme geçişte veri niceliğinde anlamlı bir değişme yoksa dallanmaya gerek yok demektir. Bu durumda o düğüm (node), yaprak (leaf) haline dönüştürülebilir. Yapraklarda oluşan verilerin merkezi hesaplanıp merkezleri birbirine yakın olan yapraklar birleştirilebilir.

## 3.4 Uygulama

Kullanılan sentetik veri tabanlarından birisinin temsili gösterimi ve algoritmanın uygulanması sonucu elde edilen sonuçlar aşağıda gösterilecektir. Kullanılan veri tabanı 10000 veriden oluşmaktadır. Verilerde özellikle bizim bulunmasını istediğimiz alanlar ile ilgili değerler verilmiştir. Aynı zamanda veri tabanında mevcut olan alan ve verilerden bazıları yanıltıcı olması açısından özellikle yerleştirilmiştir.

### 3.4.1 Kullanılan Veri Tabanı

Kullanılan bu veri tabanında toplam veri sayısı 10000 'dir. Mevcut verilerin bazıları sayısal olmalarana rağmen kategorisel olarak sınıflandırılmıştır.

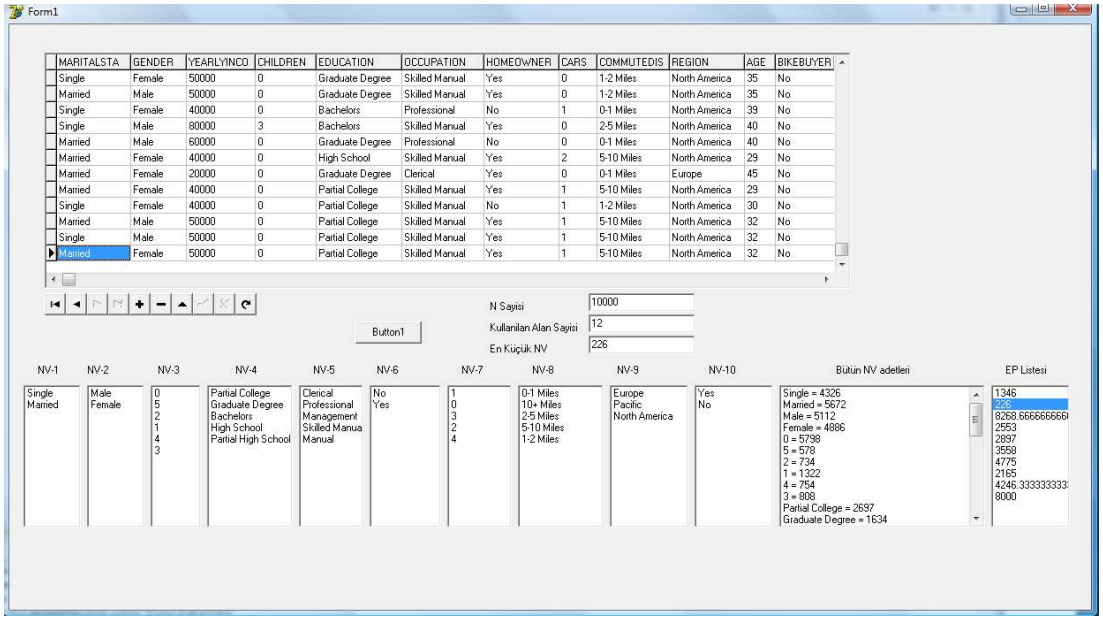
| Alan Adı | Veri Sayısı (NV) | $NV_1$ | $NV_2$ | $NV_3$ | $NV_4$ | $NV_5$ | $NV_6$ |
|---|---|---|---|---|---|---|---|
| **Marital Status** | 2 | 5673 | 4327 | | | | |
| **Gender** | 2 | 4887 | 5113 | | | | |
| **Children** | 6 | 5799 | 1323 | 735 | 809 | 755 | 579 |
| **Education** | 5 | 2578 | 1635 | 1979 | 2698 | 1110 | |
| **Occupation** | 5 | 1423 | 1730 | 1399 | 2953 | 2495 | |
| **Home Owner** | 2 | 3221 | 6779 | | | | |
| **Cars** | 5 | 1802 | 2394 | 3993 | 989 | 822 | |
| **Commute Distance** | 5 | 3081 | 1630 | 1810 | 1520 | 1959 | |
| **Region** | 3 | 2928 | 5456 | 1616 | | | |
| **BikeBuyer** | 2 | 9000 | 1000 | | | | |

**Tablo - 3.1 Algoritmadan Alınan İlk Değerler**

| Marital Status | Gender | Children | Education | Occupation | Home Owner | Cars | Commute Distance | Region | Bike Buyer |
|---|---|---|---|---|---|---|---|---|---|
| Single | Male | 0 | Partial College | Clerical | No | 1 | 0-1 Miles | Europe | Yes |
| Married | Female | 0 | Graduate Degree | Clerical | Yes | 0 | 0-1 Miles | Europe | Yes |
| Single | Female | 2 | Bachelors | Skilled Manual | No | 1 | 0-1 Miles | North America | Yes |
| Single | Female | 0 | High School | Professional | Yes | 2 | 5-10 Miles | Pacific | Yes |
| Married | Female | 1 | Bachelors | Clerical | Yes | 0 | 2-5 Miles | Europe | Yes |
| Single | Female | 1 | Partial College | Manual | No | 0 | 0-1 Miles | Europe | Yes |
| Single | Male | 0 | Partial College | Skilled Manual | No | 2 | 1-2 Miles | North America | Yes |
| Married | Female | 0 | Partial College | Clerical | Yes | 1 | 1-2 Miles | North America | Yes |
| Single | Female | 2 | Bachelors | Skilled Manual | Yes | 1 | 2-5 Miles | North America | Yes |
| Married | Female | 0 | Partial College | Clerical | Yes | 1 | 1-2 Miles | North America | Yes |
| Single | Male | 0 | Partial College | Clerical | Yes | 1 | 1-2 Miles | North America | Yes |
| Single | Female | 2 | Bachelors | Professional | Yes | 2 | 5-10 Miles | Pacific | Yes |
| Single | Female | 0 | Graduate Degree | Professional | Yes | 0 | 2-5 Miles | North America | Yes |
| Single | Female | 0 | Partial College | Professional | No | 2 | 2-5 Miles | Europe | Yes |
| Single | Male | 0 | Graduate Degree | Clerical | Yes | 0 | 0-1 Miles | Europe | Yes |
| Single | Female | 0 | Bachelors | Professional | Yes | 1 | 5-10 Miles | Pacific | Yes |
| Married | Male | 0 | Bachelors | Professional | No | 1 | 0-1 Miles | Pacific | Yes |
| Single | Female | 0 | Partial College | Skilled Manual | No | 1 | 1-2 Miles | North America | Yes |
| Single | Male | 0 | Bachelors | Clerical | Yes | 0 | 0-1 Miles | Europe | Yes |
| Single | Female | 0 | Bachelors | Professional | No | 1 | 0-1 Miles | Pacific | Yes |
| Married | Female | 1 | Partial College | Professional | Yes | 2 | 1-2 Miles | North America | Yes |
| Married | Female | 0 | Bachelors | Management | Yes | 2 | 5-10 Miles | North America | Yes |
| Married | Male | 0 | Graduate Degree | Management | Yes | 1 | 5-10 Miles | Pacific | Yes |
| Married | Male | 0 | Graduate Degree | Professional | Yes | 0 | 0-1 Miles | North America | Yes |
| Married | Female | 0 | Bachelors | Clerical | No | 0 | 0-1 Miles | Pacific | Yes |
| Single | Female | 1 | Partial High School | Clerical | Yes | 2 | 5-10 Miles | Pacific | Yes |
| Single | Male | 0 | Graduate Degree | Professional | Yes | 0 | 2-5 Miles | North America | Yes |
| Married | Male | 5 | Partial High School | Professional | Yes | 4 | 10+ Miles | Pacific | Yes |
| Married | Female | 0 | Bachelors | Management | Yes | 2 | 0-1 Miles | Pacific | Yes |
| Single | Female | 0 | High School | Skilled Manual | Yes | 1 | 5-10 Miles | North America | Yes |

**Tablo - 3.2 Veri Tabanı Örneği**

### 3.4.2 Delphi Programlama Dili İle Örnek Uygulama – 1




**Şekil - 3.1 Delphi Programı Uygulaması - 1**

Algoritma çalışmaya başladığında veritabanındaki bütün alanları kullanarak N, NV, $NV_i$ değerlerini hesaplar. Tablo 3.2 de verilen örnek veritabanı ele alındığında hesaplanan değerler Tablo 3.1 de verilmiştir. Bu durum şöyle açıklanabilir; Tabloda 10000 adet veri bulunmaktadır. Böylece N sayısı ilk başta 10000 olarak belirlenmiş olur. Tablodaki 1. alan "Marital Status" alanıdır. Bu alan "Married" ve "Single" olmak üzere iki adet veri türü içermektedir. Bu durumda "Marital Status" alanı için NV değeri 2 olarak hesaplanır. Aynı şekilde 2. alan olan "Gender" alanı "Male" ve "Female" olarak iki adet veri türü içermektedir. "Gender" alanı için veri türü sayısı NV değeri de 2 olarak hesaplanır. Bütün alanlardaki NV değerleri bu şekilde hesaplandıktan sonra $NV_i$ değerlerinin hesaplanmasına geçilir. Algoritma her bir alandaki NV değerlerinden o alanda kaç adet veri olduğunu hesaplayacaktır. Örneğin Tablo 3.1 de de görüleceği gibi "Education" alanında NV değeri 5'tir. Bu 5 değeri veri türü sayısı olarak "Bachelors", "Graduate Degree", "High School", "Partial College", "Partial High School", bilgilerinden elde edilmiştir. "Education" alanında, "Bachelors" verisinden 2578, "Graduate Degree" verisinden 1635, "High School" verisinden 1979, "Partial College" verisinden 2698, "Partial High School" verisinden 1110 adet bulunmaktadır. Buradaki alanlar sırasıyla $NV_1$, $NV_2$, $NV_3$, $NV_4$, $NV_5$ olarak kabul edilmektedir. Böylece Tablo 3.1'de de görüleceği üzere "Education"

alanı için NV = 5, "Bachelors" olan $NV_1$ = 2578, "Graduate Degree" olan $NV_2$ = 1635, "High School" olan $NV_3$ = 1979, "Partial College" olan $NV_4$ = 2698, "Partial High School" olan $NV_5$ = 1110 olarak hesaplanmıştır. Benzer şekilde tüm alanlar için ayrı ayrı bu veriler rekürsif (özyinelemeli) olarak hesaplanırken aynı zamanda EP değerleri de hesaplanmaktadır. Her bir alan için EP değerinin hesaplanması algoritmada yer aldığı gibi EP = $\sum_{i=1}^{n} |CA - NV_i|$ formülünden yararlanılarak gerçekleşecektir.

Yukarıda verilen değerleri kullanarak "Education" alanı için EP değerinin hesaplanmasını gösterecek olursak;

$$CA_{\text{"Education"}} = \frac{N}{NV} = \frac{10000}{5} = 2000$$

$$EP_{\text{"Education"}} = \sum_{i=1}^{n} |CA - NV_i|$$

$$EP_{\text{"Education"}} = |2000 - 2578| + |2000 - 1635| + |2000 - 1979| + |2000 - 2698| + |2000 - 1110|$$

$EP_{\text{"Education"}}$ = 2552 olarak hesaplanmış olur. Bu değer Tablo 3.1'de de gösterilmiştir.

Birinci aşama sonucunda en küçük EP değerine sahip olan değişken kök olarak seçilir. Tablo 3.4'te de algoritmanın çalışması sırasında, her bir alanın her bir veri türü için aynı şekilde EP değerlerini nasıl hesapladığı örneklenmiştir. Örneğin bir sonraki dallanmada algoritma, "Marital Status" un "Married" ve "Single" alanları için ayrı ayrı aynı işlemleri gerçekleştirecektir. Bu aşama şu şekilde açıklanabilir;

Tablo 3.1 e bakıldığında "Marital Status" alanının "Married" durumu için toplam veri sayısı 5673 , "Single" için 4327 olarak hesaplanmıştır. Söz konusu olan "Married" değeri olduğunda göz önüne alınacak durumlar sadece "Marital Status" alanının "Married" değeri aldığında diğer alanlarda mevcut olan toplam veri sayısı ve verilerdir. N, NV, $NV_i$ değerleri buna göre değişiklik gösterecektir. Bu durumda N=5673 olarak değişecektir. Mesela bu aşamada "Children" alanı için yapılacak işlemleri gösterecek olursak;

"Marital Status" = "Married" iken

N = 5673

NV = 6

CA = $\frac{5673}{6} = 945.5 \cong 945$

"Children" = "0" iken $NV_1 = 2948$

"Children" = "1" iken $NV_2 = 916$

"Children" = "2" iken $NV_3 = 339$

"Children" = "3" iken $NV_4 = 512$

"Children" = "4" iken $NV_5 = 542$

"Children" = "5" iken $NV_6 = 416$

$$EP = |945-2948| + |945-916| + |945-339| + |945-512| + |945-542| + |945-416|$$

EP = 4005 olur.

"Marital Status" = "Single" iken

N = 4327

NV = 6

CA = $\frac{4327}{6} = 721.16 \cong 721$

"Children" = "0" iken $NV_1 = 2851$

"Children" = "1" iken $NV_2 = 407$

"Children" = "2" iken $NV_3 = 396$

"Children" = "3" iken $NV_4 = 297$

"Children" = "4" iken $NV_5 = 213$

"Children" = "5" iken $NV_6 = 163$

$$EP = |721-2851| + |721-407| + |721-396| + |721-297| + |721-213| + |721-163|$$

EP = 4260 olur.

Her aşamada bir sonraki aşamaya geçmeden en küçük değerli EP seçilmiş olur. Böylece seçilen alan bir sonraki düğüm olacaktır. Bütün EP değerleri bu şekilde dallanma işlemi sayesinde hesaplanmış olur. Burada amaç veri tabanındaki alanları kullanarak veri tabanını en az sayıda fakat eşit bölen alanları ortaya çıkarmaktır. Eşitlikten kasıt veri sayısı gözönüne alındığında belirli bir alana yoğunlaşmanın kökte değil yapraklara doğru olmasıdır. Aksi halde veriler rastgele bölünmüş olacaktır. Az sayıda olmasından kasıt ise mümkün oldukça iki parçaya ayırabilmektir. Elbette bu alanlardaki veri türü sayısı ile de orantılıdır. Böylece kök,

düğümler ve yapraklar belirlenmiş olmaktadır. Dikkat edilmesi gereken nokta, eğer veri tabanında aynı tür veriden sadece bir adet bulunuyorsa ya da bir alanda kayıt sayısı kadar veri türü mevcut ise $\sum_{i=1}^{n}|CA - NV_i|$ formülünden elde edilecek sonucun sıfır ( 0 ) olmasıdır. Formül mutlak değerler toplamına, bir bakıma eşit uzaklık mesafesi hesaplamaya, dayandığı için asla negatif değer üretemeyeceğinden elde edilecek en küçük değer sıfır olduğunda bu değer kök olarak seçilecektir. Bu durumda veritabanı eşit parçalara ayrılamayacaktır. Eğer bu tür verilere sahip alanlar var ise kök değeri olarak onlar seçileceğinden algoritma doğru çalışmayacaktır. Bu duruma engel olmak için iki seçenek mevcuttur. Birincisi bu algoritmaya uygun verilere sahip veritabanlarının kullanılmasıdır. İkinci yol ise algoritma programlanırken, EP değerlerinin seçim aşamasında iki probleme ait kriterleri kontrol etmektir. Bu işlem şu şekilde gerçekleştirilebilir. EP değerlerinin en küçüğü seçilmeden en küçük olarak öngörülen EP değerine ait alandaki veri türü sayısı kontrol edilebilir. EP'nin ait olduğu veri türü sayısı 1 (NV=1) ise ya da veri türü sayısı veri sayısı ile aynı (NV=N) ise o EP değeri pas geçilip bir sonraki küçük değer en küçük olarak seçilebilir.

### 3.4.3 Excel 'de Elde Edilen Sonuçların İncelenmesi

| | NV | | EP | NV-1 | NV-2 | NV-3 | NV-4 | NV-5 | NV-6 |
|---|---|---|---|---|---|---|---|---|---|
| **Marital Status** | 2 | 2 | 1346 | 673 | 673 | * | * | * | * |
| **Gender** | 2 | 1 | 226 | 113 | 113 | * | * | * | * |
| **Children** | 6 | 10 | 8265 | 4132 | 343.7 | 931.7 | 857.7 | 911.7 | 1088 |
| **Education** | 5 | 4 | 2552 | 578 | 365 | 21 | 698 | 890 | * |
| **Occupation** | 5 | 5 | 2896 | 577 | 270 | 601 | 953 | 495 | * |
| **Home Owner** | 2 | 6 | 3558 | 1779 | 1779 | * | * | * | * |
| **Cars** | 5 | 8 | 4774 | 198 | 394 | 1993 | 1011 | 1178 | * |
| **Commute Distance** | 5 | 3 | 2162 | 1081 | 370 | 190 | 480 | 41 | * |
| **Region** | 3 | 7 | 4245 | 405.3 | 2123 | 1717 | * | * | * |
| **BikeBuyer** | 2 | 9 | 8000 | 4000 | 4000 | * | * | * | * |

**Tablo - 3.3 Algoritmanın Birinci Aşamasından Örnek Veriler**

| Marital Status | 5673 | EP | NV-1 | NV-2 | NV-3 | NV-4 | NV-5 | NV-6 |
|---|---|---|---|---|---|---|---|---|
| Married | Gender | 367 | 183.5 | 183.5 | * | * | * | * |
| | Children | 4005 | 2003 | 29.5 | 606.5 | 433.5 | 403.5 | 529.5 |
| | Education | 1587.6 | 444.4 | 65.6 | 41.6 | 349.4 | 686.6 | * |
| | Occupation | 2176.4 | 474.6 | 19.4 | 613.6 | 722.4 | 346.4 | * |
| | Home Owner | 3487 | 1744 | 1744 | * | * | * | * |
| | Cars | 2130.4 | 26.4 | 17.4 | 1021 | 467.6 | 597.6 | * |
| | Commute Distance | 1230.8 | 615.4 | 105.6 | 228.6 | 266.6 | 14.6 | * |
| | Region | 2796 | 503 | 1398 | 895 | * | * | * |
| | BikeBuyer | 4673 | 2337 | 2337 | * | * | * | * |
| | | | | | | | | |
| Marital Status | 4327 | EP | NV-1 | NV-2 | NV-3 | NV-4 | NV-5 | NV-6 |
| Single | Gender | 141 | 70.5 | 70.5 | * | * | * | * |
| | Children | 4259.67 | 2130 | 314.2 | 325.2 | 424.2 | 508.2 | 558.2 |
| | Education | 1005.6 | 133.6 | 299.4 | 20.6 | 348.6 | 203.4 | * |
| | Occupation | 783.6 | 102.4 | 289.4 | 12.6 | 230.6 | 148.6 | * |
| | Home Owner | 71 | 35.5 | 35.5 | * | * | * | * |
| | Cars | 2696.4 | 224.4 | 376.6 | 971.6 | 543.4 | 580.4 | * |
| | Commute Distance | 1008.4 | 465.6 | 264.4 | 38.6 | 213.4 | 26.4 | * |
| | Region | 1644.67 | 97.67 | 724.7 | 822.3 | * | * | * |
| | BikeBuyer | 3327 | 1664 | 1664 | * | * | * | * |

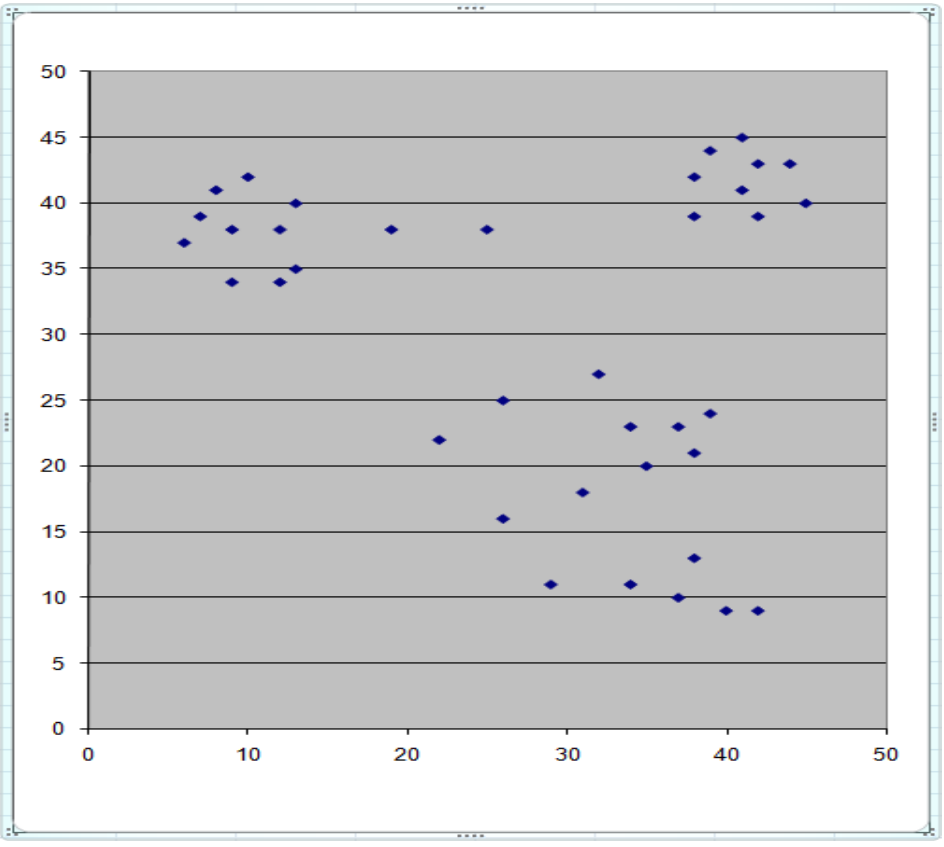
**Tablo - 3.4 Algoritmanın İkinci Aşamasından Örnek Veriler**

### 3.4.4 Algoritmanın ve Uygulamanın Sonuçları

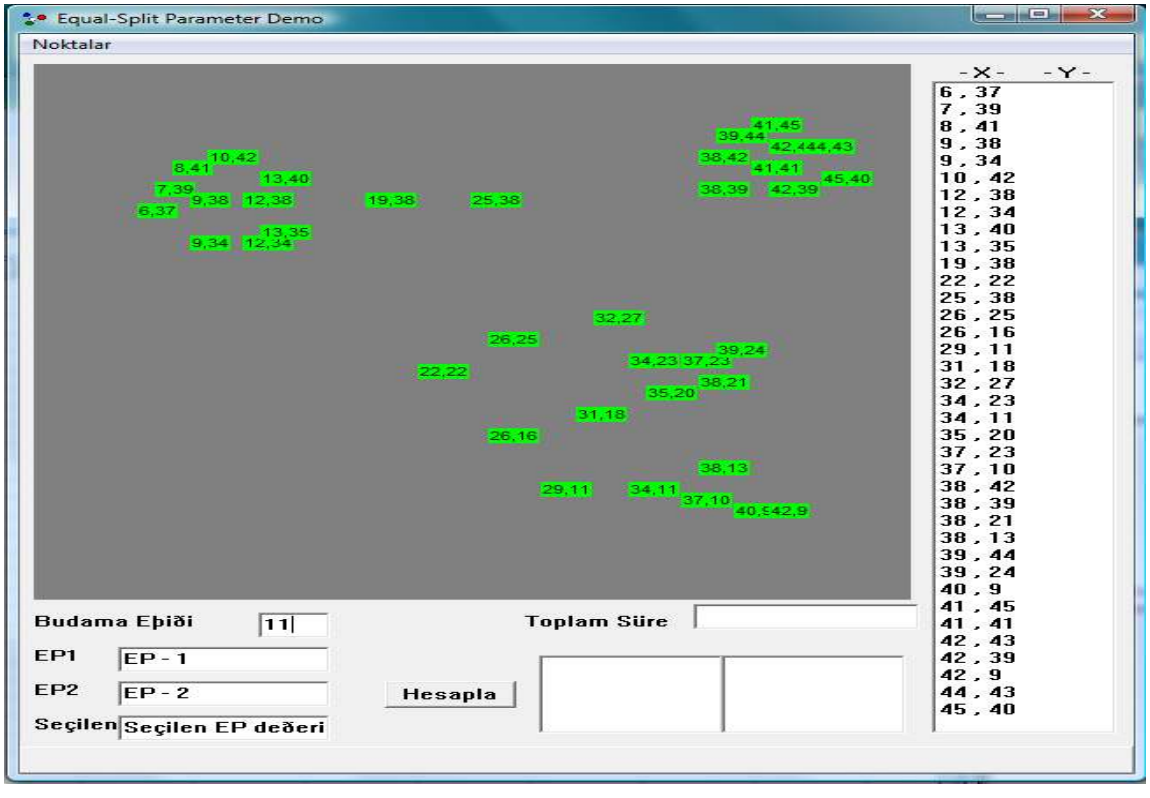
Bu bölümde çalışması yapılan algoritmanın uygulama sonuçlarını değerlendirmek için başka bir örnek üzerinde elde edilen bilgiler değerlendirilecektir.

Delphi programlama dilinde gerçekleştirilen bu çalışmada, aşağıdaki tabloda yer alan verilerin kümelemesi yapılmaya çalışılmıştır.

| Değer 1 | Değer 2 |
|---|---|
| 6 | 37 |
| 7 | 39 |
| 8 | 41 |
| 9 | 38 |
| 9 | 34 |
| 10 | 42 |
| 12 | 38 |
| 12 | 34 |
| 13 | 40 |
| 13 | 35 |
| 19 | 38 |
| 22 | 22 |
| 25 | 38 |
| 26 | 25 |
| 26 | 16 |
| 29 | 11 |
| 31 | 18 |
| 32 | 27 |
| 34 | 23 |
| 34 | 11 |
| 35 | 20 |
| 37 | 23 |
| 37 | 10 |
| 38 | 42 |
| 38 | 39 |
| 38 | 21 |
| 38 | 13 |
| 39 | 44 |
| 39 | 24 |
| 40 | 9 |
| 41 | 45 |
| 41 | 41 |
| 42 | 43 |
| 42 | 39 |
| 42 | 9 |
| 44 | 43 |
| 45 | 40 |

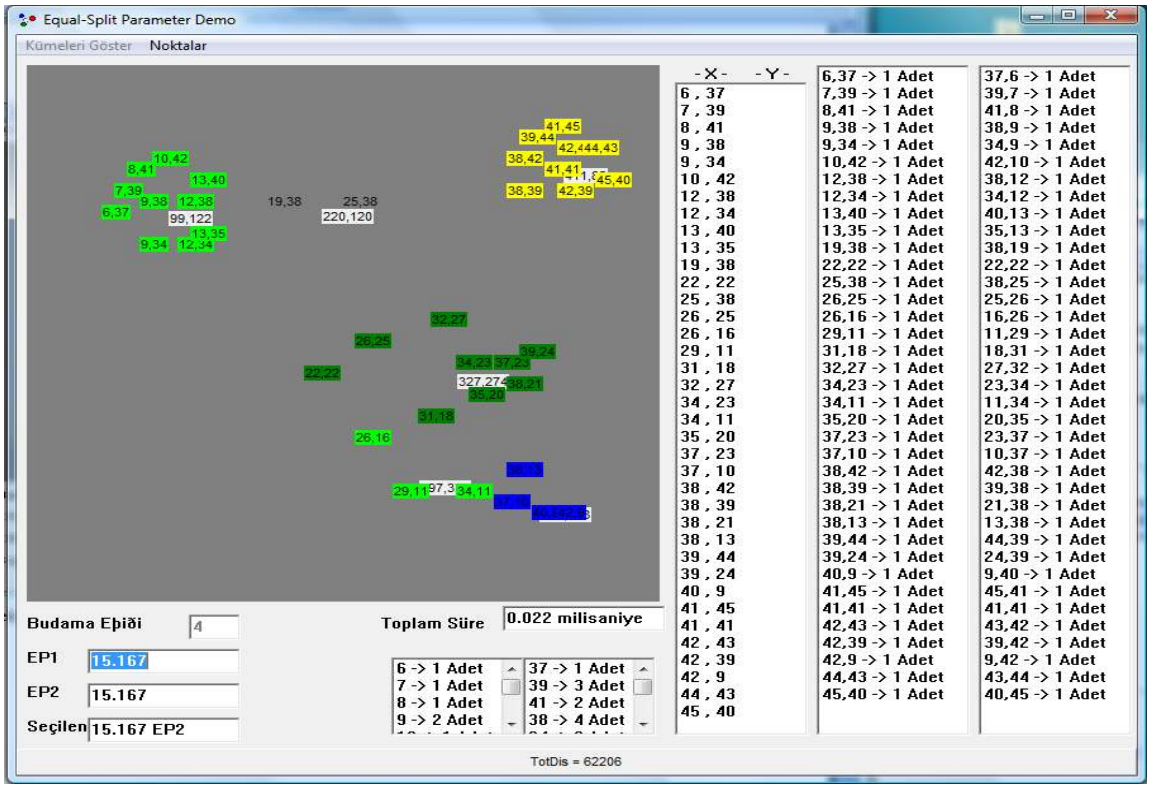



**Tablo 3.5 Değerlendirmede Kullanılan Veriler**

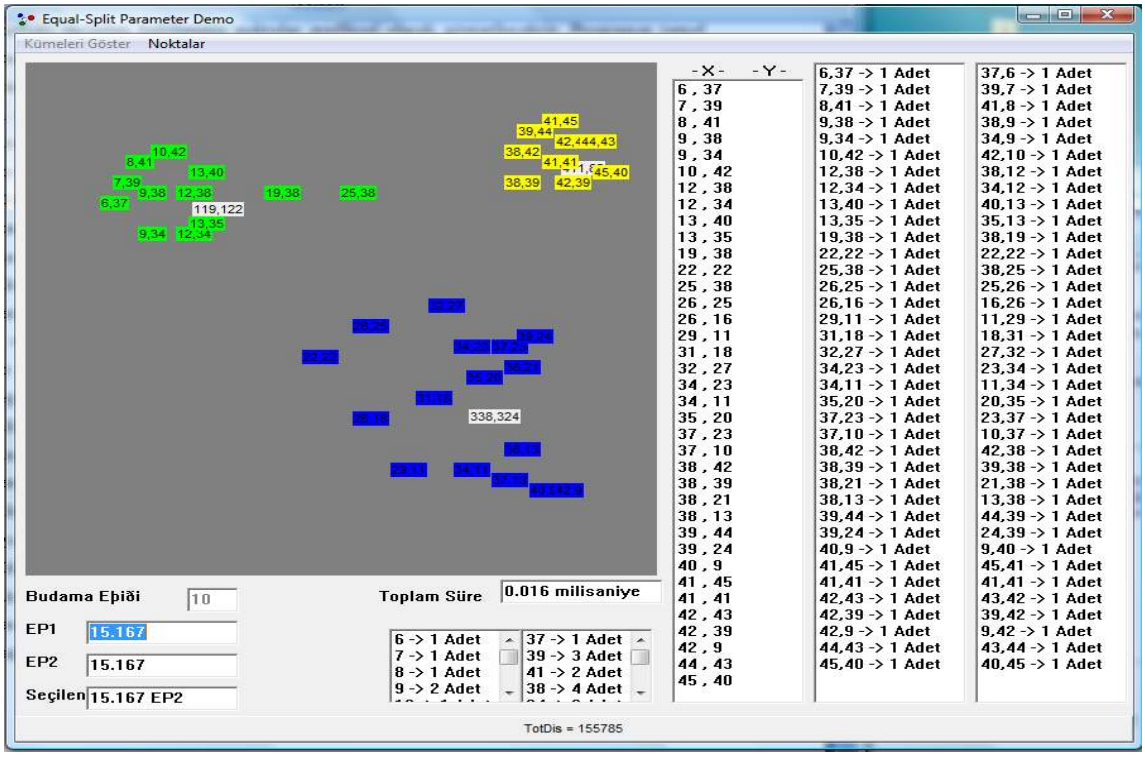



**Şekil 3.2 Delphi Programı Uygulaması - 2**

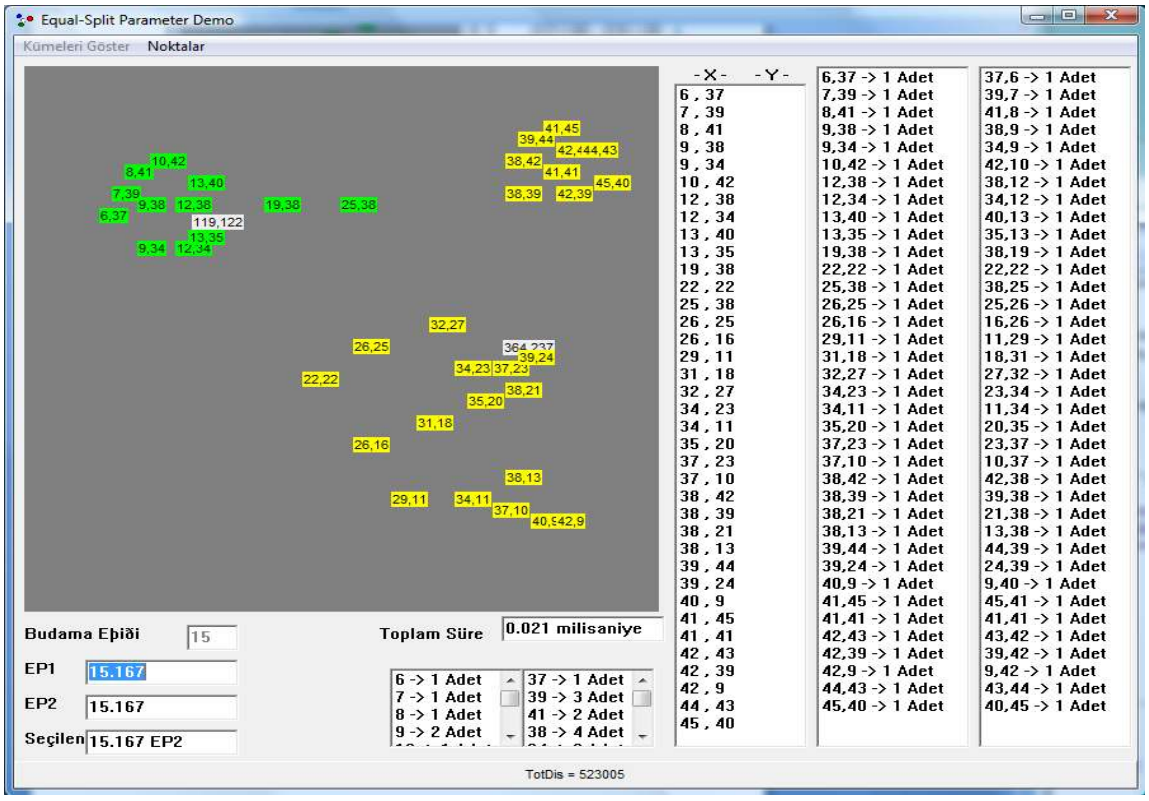
Şekil 3.2 de Tablo 3.5 de verilerin delphi programında bir düzlemde gösterilmesinin gerçekleştirildiği görülmektedir. Program ilk çalıştığında veriler tablo olarak görülmemektedir. Programın menüsündeki "Noktalar" seçeneği tıklandığında sağ taraftaki önceden belirlenmiş noktalar grafiksel olarak gösterilecektir. Programın temel amacı doğru bir biçimde ve mümkün olan en kısa sürede verilen noktaları kümelere ayırmaktır. "Noktalar" seçeneği tıklanarak veriler grafiksel olarak gösterildikten sonra bu değerleri kullanarak algoritma işletilecektir. Bunun için aşağıda bulunan "Hesapla" butonu kullanılacaktır. Bir önceki uygulamada anlatıldığı gibi verilerin kümelenmesi için ağaç hesaplamaları yapılacak ve ardından kümeleme işlemine geçilecektir. Kümeleme işleminin gerçekleştirilmesi için "Budama Eşiği" kullanılması gerekmektedir. Bunun sebebi kullanılan algoritmanın çok detaylı bir ağaç üretmesidir. Doğal olarak bu ağacın budama işlemine tabi tutulması gerekecektir. Bunun için küme merkezi, veri miktarı, standard sapma gibi özellikler kullanılabilir. Bu uygulamada budama için küme merkezi ve çap değerleri kullanılmıştır. Bu işlemlerin ne kadar süre aldığı hesaplanmış ve ekrana yazdırılmıştır. "Hesapla" butonuna tıklandığında Tablo 3.5 'teki Değer1 ve Değer2 alanında yer alan veriler için EP değerleri hesaplanır. Hesaplama sonucunda her veriden kaç adet değer olduğu da ekranda gösterilecektir. Diğer EP hesaplamaları için aynı işlemler tekrar edilip ağaç oluşturulacaktır. "Budama Eşiği" alanındaki veriye göre birbirine yakın noktalar aynı kümelere alınarak ağaç budanacak ve küme sayısı belirlenecektir. Bu işlemden sonra oluşan kümelerin gösterilmesi için menüden "Kümeleri Göster" butonu tıklandığında kümler gösterilecektir. Burada dikkat edilmesi gereken durum "Budama Eşiği" değerleridir. düşük budama değerleri küme sayısını arttırırken yüksek budama değerleri küme sayısını azaltacaktır. Uygulamada yaklaşık 3 adet budama aralığı tespit edilmiştir. "0 - 7" arasındaki değerler için 6 ya da daha fazla küme, "8 - 12" arasındaki değerler için 4 ya da daha az küme, "13 - 25" arasındaki değerler için 2 ya da daha az küme, oluştuğu görülmüştür. Bu durumlar aşağıdaki şekillerde gösterilmiştir. Bu örnek verilerle programın çalışması, verilen budama değerine göre, ortlama 22 milisaniyedir. Bu süreye ekrana çizim için geçen süre dahil edilmemiştir.




**Şekil 3.3 Budama Eşiğinin 8'den Küçük Olduğu Durum**




**Şekil 3.4 Budama Eşiğinin 8-12 Arasında Olduğu Durum**




**Şekil 3.5 Budama Eşiğinin 12'den Büyük Olduğu Durum**

## 3.5 Çalışmanın Önemi

Bu çalışma gelişmekte olan veri madenciliği uygulamalarında yeni bir algoritma kullanarak farklı bir bakış açısı elde etmeyi hedeflemesiyle önem taşımaktadır. Oluşturulan algoritma daha da geliştirilip diğer algoritmalarla da birleştirilerek veri madenciliği uygulamalarında etkin sonuçlara ulaşmaya katkıda bulunacaktır.

## 3.6 Çalışmanın Kısıtları

Bu tez çalışmasının en önemli kısıtlarından birisi, kullanılan algoritmanın yeni olması ve bu sebepten belirli kriterlere uyan veri tabanlarında çalışabilmesidir. Seçilen veri tabanlarında bazı özel durumlara dikkat edilmesi gerekmektedir. Bu durumlardan birisi herhangi bir alanda var olan veri çeşidinin her bir kayıt için

birbirinden farklı olmaması gerekliliğidir. Bir diğer olumsuz durum ise herhangi bir alanda sadece bir adet veri çeşidi bulunmasıdır ki her iki durumda da algoritma hatalı sonuçlar üretmektedir.

## 3.7 Araştırmanın Modeli ve Hipotez

Bu tezde öne sürülen algoritma için geniş bir literatür taraması yapılmıştır. Elde edilen araştırma sonuçları bu çalışmada temel alınarak sonuç elde edilmeye çalışılmıştır.

**Hipotez:** Kullan algoritma veri madenciliği uygulamalarında kullanılabilecek ve mevcut algoritmalara yakın derecede etkili olacaktır.

## 3.8 Algoritmaya Ait Özellikler

Kullanılan algoritmanın veri setini yeniden taramasına gerek olmamasından dolayı karmaşıklığı O(n) 'dir. Bu özelliği sayesinde daha fazla karmaşıklık değerine sahip olan algoritmalardan daha hızlı sonuç vermektedir. Karşılaştırmak gerekirse, BIRCH algoritmasında bir CF (Clustering-Feature) ağacı yaratılır. CF ağacı dallanma faktörü ve budama parametersi ile yükseklik dengeli bir ağaç olur. BIRCH algoritmasında en kısa yoldan sınıflara ulaşmak için en uygun simge kök olarak seçilmiştir. Böylece algoritma veri tabanındaki en uygun alan ile bir sonraki düğümü (node) belirlemektedir. Burada "en uygun" ifadesi ile, veri tabanını kabaca iki ya da daha fazla eş parçaya bölmek kastedilmektedir. Buna en yakın alan en uygun alan olarak seçilir. Literatürde ağacı oluşturmak için farklı yöntemler vardır. Bunlardan biri entropi (entropy) kavramıdır. ID3 ve C4.5 algoritmaları düğüm temsilcilerini bulmak için entropi yöntemini kullanırlar.

Kullanılan algoritma her dal ve ayrı olarak onun düğümleri için EP'yi yeniden hesaplayan tekrarlamalı bir algoritmadır. Algoritma prosedürümüz doğal kümeler oluşturduğu için ağaç budanmaya ihtiyaç duymaktadır. BIRCH algoritmasındaki gibi eşik değerlerini kullanarak budama yapılmakta. Yapraklardan başlayarak, kullanıcı tarafından verilen eşik değerlerinin altında kalan her yaprak en yakınındakine dahil oluyor. Böylece algoritma tarafından üretilen doğal kümelerin sayısı azalıyor. Kümeler biçimlendirildikten sonra PAM, CLARA ya da K-Means gibi kümeleştirme algoritmaları kullanılarak yaprakların sayısı istenilen sayıya düşürülebiliyor.

## 3.9 Hipotezin Testi

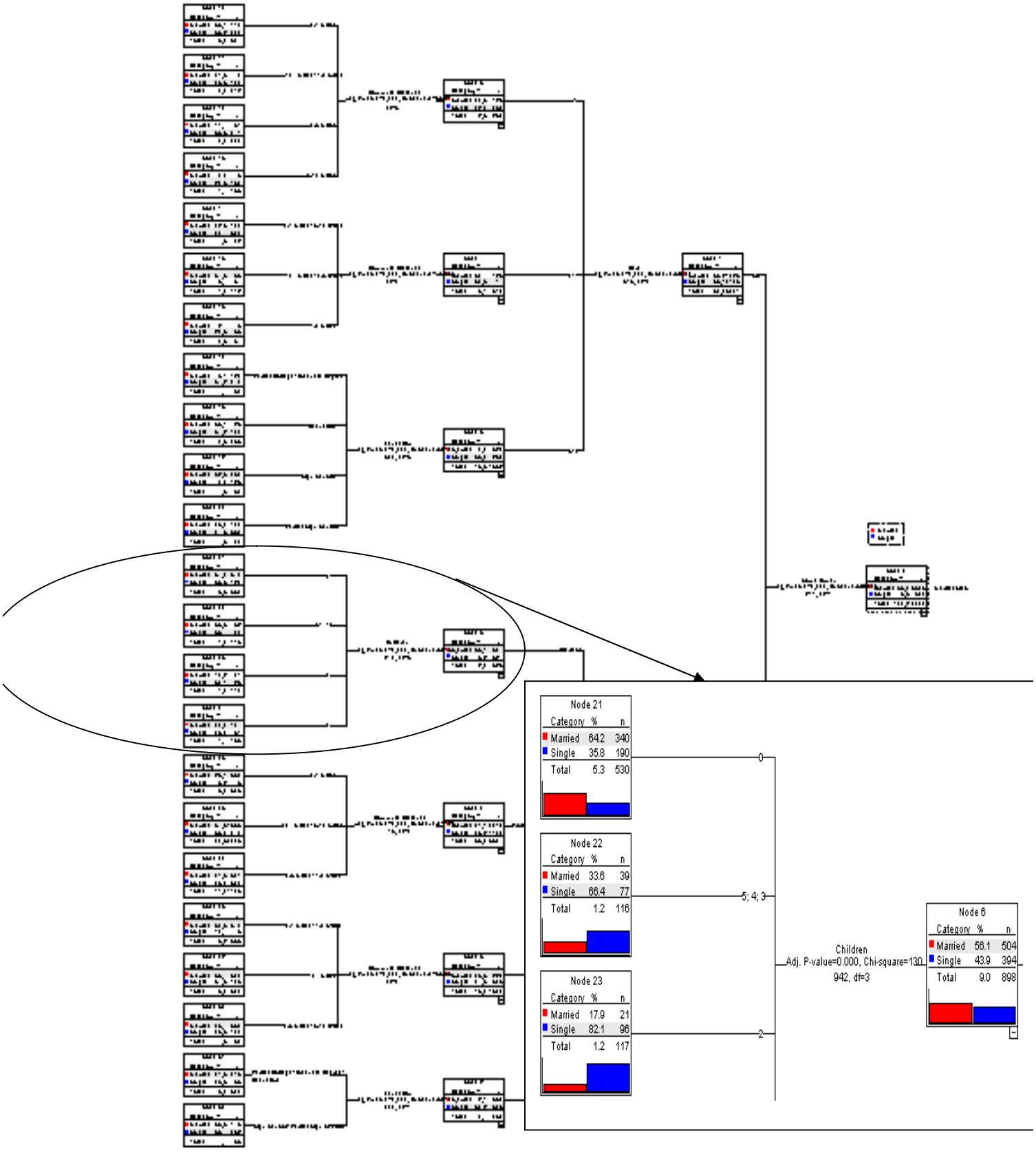
Bu araştırmada öne sürülen hipotezi test etmek amacıyla, BIRCH ve CLARA algoritmalarının performansları kullanıldı. Algoritma veri setini yeniden taramaya ihtiyaç duymamaktadır. Kullanılan algoritmanın karmaşıklığının da O(n) olması önemli bir avantaj sağlamıştır.

## 3.10 Spss Programı İle Elde Edilen Sonuçlar

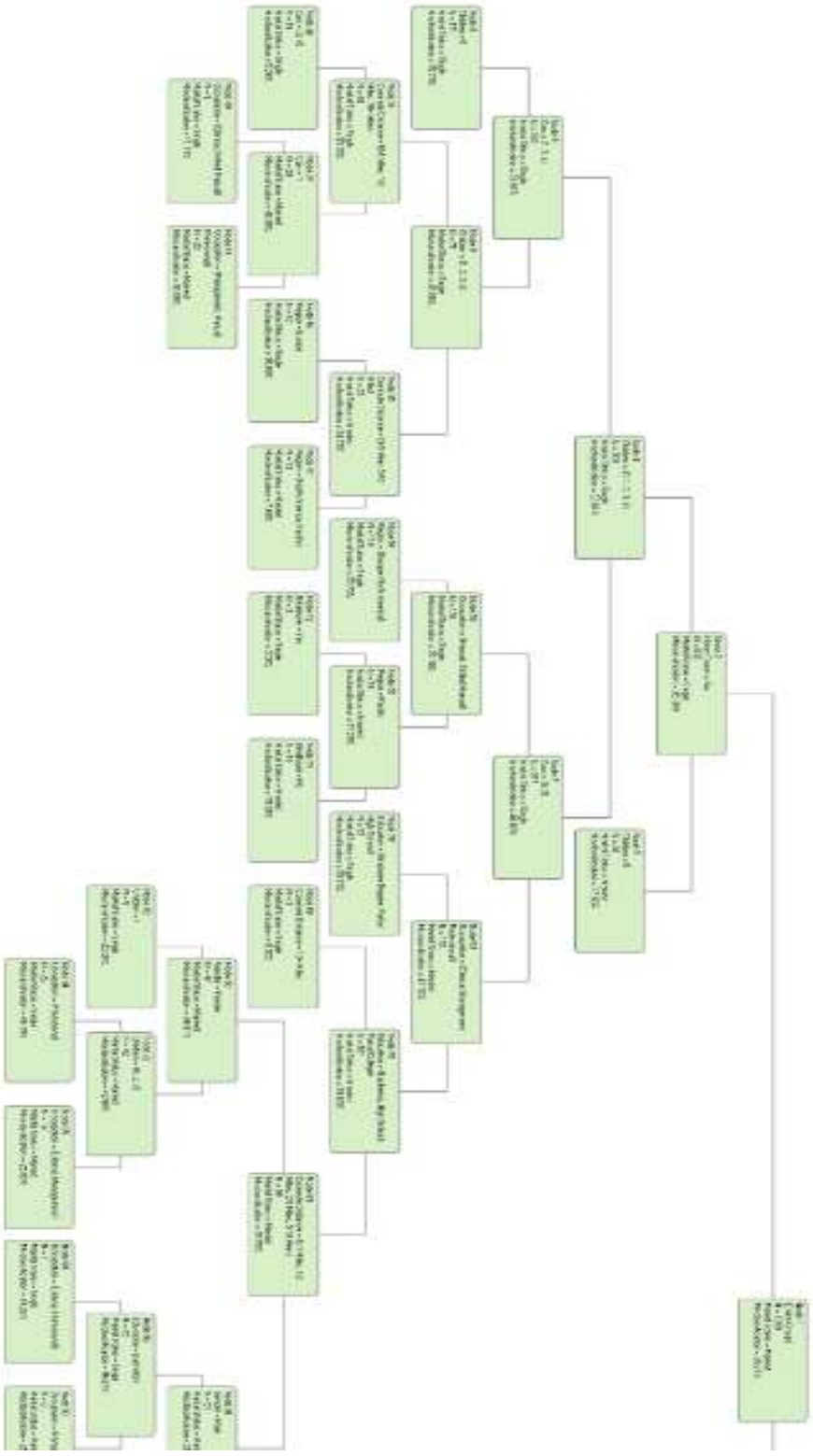
**Model Summary**

| | | |
|---|---|---|
| Specifications | Growing Method | CLARANS |
| | Dependent Variable | Marital Status |
| | Independent Variables | Gender, Children, Education, Occupation, Home Owner, Cars, Commute Distance, Region, BikeBuyer |
| | Validation | None |
| | Maximum Tree Depth | 3 |
| | Minimum Cases in Parent Node | 100 |
| | Minimum Cases in Child Node | 50 |
| Results | Independent Variables Included | Home Owner, Cars, Commute Distance, Education, Occupation, Children |
| | Number of Nodes | 33 |
| | Number of Terminal Nodes | 23 |
| | Depth | 3 |

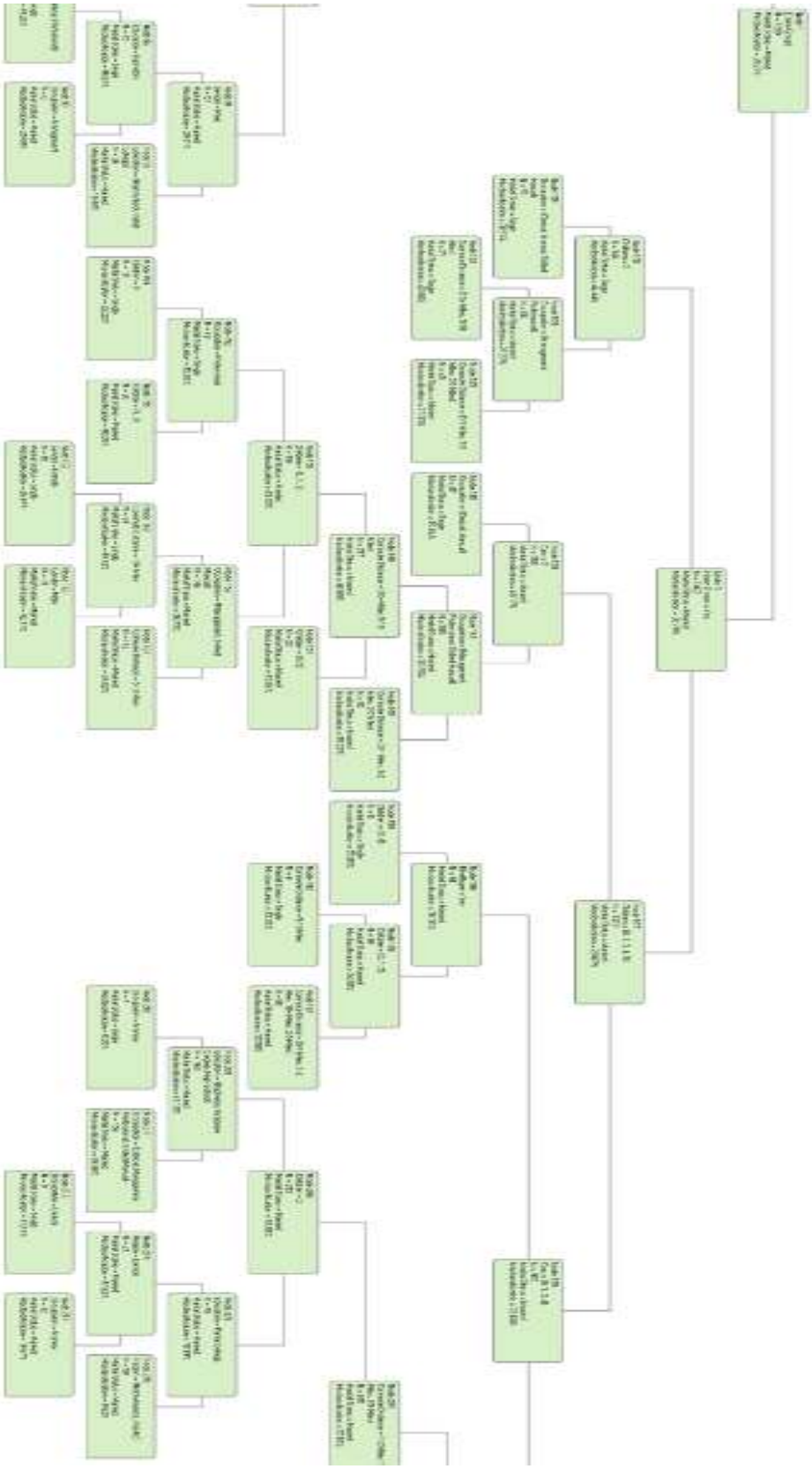
**Tablo 3.6 SPSS Sonuçları**


Node 21
Category % n
Married 64.2 340
Single 35.8 190
Total 5.3 530
0
Node 22
Category % n
Married 33.6 39
Single 66.4 77
Total 1.2 116
5; 4; 3
Node 23
Category % n
Married 17.9 21
Single 82.1 96
Total 1.2 117
2
Children
Adj. P-value=0.000, Chi-square=130.942, df=3
Node 6
Category % n
Married 56.1 504
Single 43.9 394
Total 9.0 898

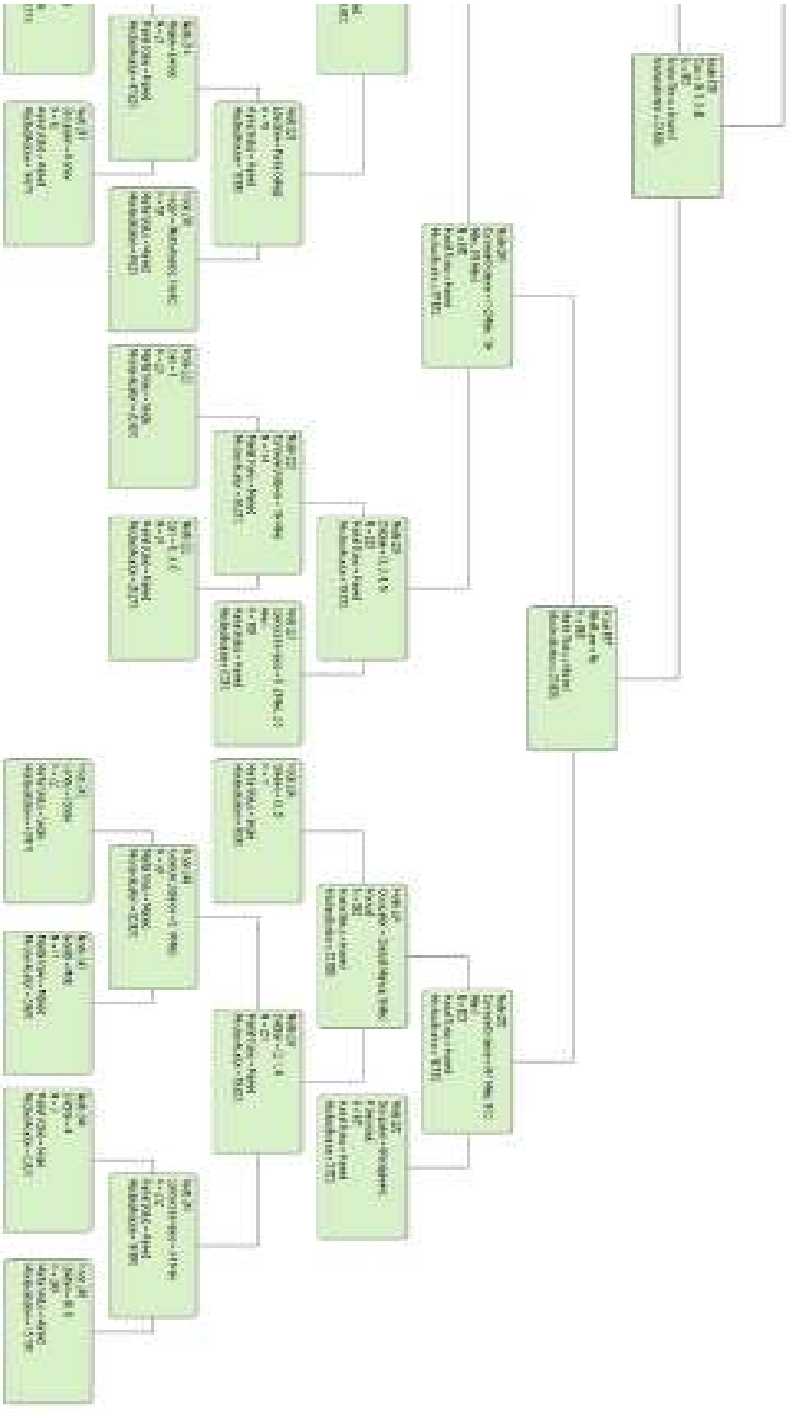
**Şekil - 3.6 SPSS Ağacı**



**Şekil - 3.7 Algoritmadan Elde Edilen Ağaç - 1**



**Şekil - 3.8 Algoritmadan Elde Edilen Ağaç - 2**



**Şekil - 3.9 Algoritmadan Elde Edilen Ağaç - 3**

# 4. BÖLÜM

## 4.1. SONUÇ VE ÖNERİLER

### 4.1.1 Genel Sonuçlar

Bu bölümde, tez konusu kapsamında gerçekleştirilen uygulamanın sonuçları özetlenmektedir. Ayrıca algoritmanın kısıtları ve daha iyi hale getirilmesine yönelik çözümler değerlendirilmektedir.

Bu çalışma, tasarlanan algoritmanın veri madenciliğinde kullanılanlar gibi kendine bir yer edineceğini gösterebilmek amacıyla gerçekleştirilmiştir. Bu algoritmanın, karmaşıklığı daha fazla olan diğer algoritmalara oranla tercih edilip daha da geliştirilebilecek bir algoritma olduğu görülmektedir.

Veriyi kümelere ayırmak için ağaç metodunu Zhang, BIRCH algoritmasında kullanmıştır. Kümeleri biçimlendirmek için CF (Clustering - Feature) ağacı kullandı ve kümeleri tek bir tarama bazı tarifsel istatiksel parametreleri kullanarak ve zıtlaştırma kümeleri oluşturdu. CF ağacında , tüm veri tabanını temsil eder; veri ayıklamanın bir dalı olan sınıflandırma için başka bir ağaç yöntemi kullanılabilir. veri tabanında yer alması olası, doğal kümeleri bulmak için bir karar ağacı kullanabiliyoruz. Bizim yöntemimizde robot, BIRCH yönteminde olduğu gibi tüm veri tabanını temsil etmiyor, buna rağmen tüm veri tabanı bölümünün en belirgin kısmıdır. Kısımların yaklaşık önemlerine-anlamlarına göre veri tabanını analiz ederek, doğal minimal kümelere ulaşıyoruz. Bu da oldukça iyi bir performans sağlamaktadır.

Daha önce bölüm 3.8 'de de değinildiği gibi algoritmanın bazı kısıtlılıkları bulunmaktadır. Geliştirme süreci devam eden bir algoritma olduğu için bu kısıtlar zamanla ortadan kalkabilecektir.

Çalışmada, farklı bir algoritma kullanılarak yeni bir yol elde edilmeye çalışılmış ve başarılı olabilecek bir sonuç elde edilmeye çalışılmıştır.

### 4.1.2 Gelecek Araştırmalar İçin Öneriler

Bizim yöntemimiz entropi ya da gini indeks yöntemine daha yakındır. Bu yöntemler de kök verisi kümelere ayrılmıştır.

Ek olarak, verileri ve dalların arasında yer alan nodları ayırmak için kümeleri ve alt-kümeleri tanımladık. Kök de dahil olmak üzere, her nod veritabanını eşite en yakın parçalara ayırmak için hesaplanmıştır. Bu algoritmanın temel aldığı ölçüt budur. Böylece bir sonraki aşamada kümeleme algoritmaları için daha iyi sonuç verecek bir algoritma elde edilmeye çalışılmıştır. Bu algoritmanın, dikkate alındığında ve diğer sistemlere entegre edildiğinde iyi sonuçlar vereceği düşünülmektedir.

KAYNAKLAR

Agrawal Rakesh ve Shafer John c., Parallel Mining of Association Rules, IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 6, 1996, s. 962-969.

Agrawal Rakesh ve Shafer John c., Parallel Mining of Association Rules, 1995.

Agrawal Rakesh ve Srikant Ramakrishnan, Fast Algorithms for Mining Associalion Rules, 20. VLDB Konferansı, Şili, 1994, s.487-499.

Agrawal Rakesh ve Srikant Ramakrishnan, Mining Sequential Patterns, 11. Uluslararası Data Engineering Konferansı, 1995.

Agrawal Rakesh ve ark., Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications, ACM-SIGMOD Management of Data Konferansı, 1998, s. 94 - 105.

Agrawal Rakesh ve ark., Database Mining: A Performance Perspective, IEEE Transactions on Knowledge and Data Engineering, Aralık 1993, s.914-925.

Agrawal Rakesh, vd., Mining Association Rules between Sets of Items in Large Databases, 1993 ACM SIGMOD Konferansı Bildirisi, USA, 1993, s. 2.

Akpınar Haldun, Veritabaniarında Bilgi Keşfi ve Veri Madenciliği, İ.Ü. İşletme Fakültesi Dergisi, C:29, sayı: lINisan 2000, s. 1-22.

Ankerst Michael ve ark., OPTICS : Ordering Points to Identify the Clustering Structure, ACM SIGMOD Management of Data Konferansı, Philadelphia, 1999, s. 49- 60.

Bacher Johann, Cluster Analysis, (çevrimiçi) www.soziologie.wiso.unierlangen.de/koeln/script/chap3.pdf , (29/06/2008).

Berkhin Pavel, Survey of Clustering Data Mining Techniques, (çevrimiçi) http://citeseer.nj.nec.com/berkhin02survey.html , (7/4/2008).

Beyer Kevin ve ark., When ls 'Nearest Neighbor' Meaningful?, 7. Uluslararası Database Theory Konferansı (lCDT'99), İsrail, 1999, s. 217-235.

Cabena Peter ve ark., Discovering Data Mining: From Concept to Implementation, USA, International Business Machines Corporation, 1998, s.12.

Cheung David Wai-Lok, ve ark., A Fast Distributed Algorithm for Mining Associalion Rules, PDlS Bildirisi, 1996.

Colin Andrew ve Journal Dobbs, Building Decision Trees with the ID3 Algorithm, Haziran 1996.

Dempster Arthur. P. vd., Maximum Likelihood from Incomplete Data via the EM AIgorithm, journal of the Royal Statistical Society Agglomerative, B serisi, Vol. 39, 1977, s. 1-38.

Dunham Margaret H., Data Mining Introductory and Advanced Topics, Prentice Hall, Pearson Education Inc., New Jersey, 2003, s. 8.

Efe Önder ve Kaynak Okyay, Yapay Sinir Ağları ve Uygulamaları, Boğaziçi Üniversitesi, 2004, s.1O-12.

Eğecioğlu Ömer, Parametric Approximation Algorithms for High-Dimensional Euclidean Sim ilarity, 5. Principles and Practice of Knowledge Discovery in Databases (PKDD'OI) Konferansı, Freiburg, Almanya, Eylül 2001, s. 79-90.

Ester Martin ve ark., A Density-Based Algorithm for Discovering Clusters in Large Databases with Noise, 2. Uluslararası Knowledge Discovery and Data Mining Konferansı, 1996. Fausett Laurene, Fundamentals of Neural Networks, Prentice-Hall, 1994, s. 3.

Frenkel Brian, Using Artificial Intelligence to Detect Fraud in Credil Cards, (çevrimiçi) http://csel.cs.colorado.edu/~cs3202/papers2/BrianFrenkel.html , (20/07/2008).

Giudici Paolo, Applied Data Mining: Statistical Methods for Business and Industry, Wiley, 2004, s. 83.

Guha Sudipto ve ark., CURE: A Clustering Algorithm for Large Databases, Teknik Rapor, Beıı Laboratuvarları, Murra'y' Hill, 1997.

Guha Sudipto ve ark., CURE: An Efficient Clustering Algorithm for Large Databases, ACM SIGMOD Konferansı, i998, s. 73-84.

Han Jiawei ve Kamber Micheline, Data Mining Concepts and Techniques, Morgan Kaufman Publishers, Academic Press, 200 i, s. 106.

Joshi Karuna Pande, Analysis of Data Mining Algorithms, (çevrimiçi) http://www.ernestedu.com/proj_rpt.htm, (04/04/2008)

Hinneburg Alexander ve Keim Daniel A., An Efficient Approach to Clustering in Large Multimedia Databases with Noise, Uluslararası Knowledge Discovery and Data Mining Konferansı (KDD'98), ABD, 1998, s. 58-65.

Kauffman L., Rousseeuw PJ, Finding Groups in Data: An Introduction to Cluster Analysis, John Wiley and Sons, 1990.

Khan Maleq ve ark., K-Nearest Neighbor Classification on Spatial Data Streams Using P-Trees, 6. Pasifik Asya Knowledge discovery and Data Mining Konferansı PAKKDD'02, Taiwan, 2002, s. 517-518.

Kimball Raph, DBMS Dealing With Dity Data (çevrimiçi) http://www.dbms-mag.com/9609d14.html (04/05/2008)

Kut Alp ve Yılmaz Sedat, Veri Madencilik Uygulamaları, İnternet ders notu, (çevrimiçi) courses.cs.deu.edu.tr/cse572/VeriMadencilikUygulamalar.doc (24/03/2008).

Lipmann Richard P., An Introduction to Computing with Neural Nets, IEEE ASSP Dergisi" Nisan 1987, s. 155- 162.

MacQueen, J., Some Methods for Classification and Analysis of Multivariate Observations, 5. Berkeley Matematiksel İstatistik ve Olasılık Sempozyumu, University of California Press, 1967, s. 281-297.

Manish Mehta vd., SLIQ: A Fast Scalable Classifier for Data Mining, S. Uluslararası Extending Database Technology Konferansı, Avignon, Fransa, Mart 1996.

Olson Clark F, Parallel Algorithms For Hierarchical Clustering, Teknik Rapor, 1993,

(Çevrimiçi) http://citeseer.ist.psu.edul17291.html (30/06/2008). .

Orhunbilge Neyran, Uygulamalı Regresyon ve Korelasyon Analizi, Avcıol Basım_Yayın, İstanbul, 1999, s. 9.

Quinlan 1. Ross, Induction of Decision Trees, Journal of Machine Learning, 1986, s. 81-106.

Quinlan, J. Ross, Simplifying Decision Trees", International Journal of Man-Machine Studies, sayı: 27,1987, s. 221-234.

Sibson R, An Optimally Efficient Algorithm for the Single Link Cluster Method, The Computer Journal, Vol. 16, Issue I, 1973.

Xiaowei Xu vd., A Distribution-Based Clustering Algorithm for Mining in Large Spatial Databases, ICDE, sayı: 14, 1998, s. 324-331.

Yamaç Aytekin, Temel Bileşenler Analizi ile Türkiye 'nin AB Ülkeleri İçindeki Yeri, Kara Harp Okulu Bilim Dergisi, 2002-2, (çevrimiçi), www.kho.edu.tr/yayinlar/bilimdergisi/bilimder/doc/2002-2/4_bilder.doc,(21/09/2008).

Yohannes Yisehac ve Webb Patrick, Classifications and Regression Trees, CART: A user Manuel For Identifying Indicators of Vulnerability to Famine and Chronic Food Insecurity, y.y, International Food Policy Research Institute, 1999, s. 15.

Zhang Tian, ve ark., BIRCH: An Efficient Data Clustering Method for Very Large Databases,