
**BİLECİK ÜNİVERSİTESİ**

**Fen Bilimleri Enstitüsü**

**Bilgisayar Mühendisliği Anabilim Dalı**

# GERÇEK TIBBİ VERİLER ÜZERİNDE VERİ MADENCİLİĞİ YÖNTEMLERİNİ KULLANARAK HASTALIK TEŞHİSİ

**Hatice ÇATALOLUK**
**Yüksek Lisans Tezi**

**Tez Danışmanı**
**Yrd. Doç. Dr. Metin KESLER**

**BİLECİK, 2012**

| | |
|---|---|
|  <br>**BİLECİK ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI** | **YÜKSEK LİSANS**<br><br>**JÜRİ ONAY FORMU** |

Bilecik Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ……..……tarih ve …… sayılı kararıyla oluşturulan jüri tarafından …… tarihinde tez savunma sınavı yapılan Hatice Çataloluk'un "Gerçek Tıbbi Veriler Üzerinde Veri Madenciliği Yöntemlerini Kullanarak Hastalık Teşhisi" başlıklı tez çalışması Bilgisayar Mühendisliği Anabilim Dalında YÜKSEK LİSANS tezi olarak oy birliği/oy çokluğu ile kabul edilmiştir.

## JÜRİ

**ÜYE**

**(TEZ DANIŞMANI) :** Yrd. Doç. Dr. Metin KESLER

**ÜYE :** Yrd. Doç. Dr. Cihan KARAKUZU

**ÜYE :** Yrd. Doç. Dr. Uğur YÜZGEÇ

## ONAY

Bilecik Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ……/……/……tarih ve ……/…… sayılı kararı.

İMZA/MÜHÜR

# ÖZET

Günümüzde veri madenciliği çoğu kritik problemin çözümünde önemli bir rol oynamaktadır. Özellikle tıp alanında medikal verilerin veritabanlarında saklanmasıyla birlikte oluşan büyük veri yığınları, veri madenciliği yöntemleri için çok tercih edilen bir uygulama alanı olmaktadır. Veri madenciliği tekniklerini kullanan biyomedikal sistemler sayesinde hızlı ve etkili bir şekilde bilgilerin elde edilmesi, hekimlere ve hastalara büyük fayda sağlamaktadır.

Bu tez çalışmasında k-en yakın komşu (KNN) ve k-means algoritmaları detaylı bir şekilde incelenmiştir. Ayrıca bu algoritmalar kullanılarak tıp alanında hekimlerin kullanabileceği, dermatolojik hastalıkların teşhisi için tahmin yapabilen ve hasta kayıtlarının nitelikleri arasındaki ilişkileri analiz etme imkanı sunabilen yardımcı bir karar verme sistemi tasarlanmış ve gerçekleştirilmiştir.

**Anahtar Kelimeler**

Veri madenciliği, K en yakın komşu, K-means, Biyomedikal, Tıp Bilişimi, Dermatoloji

# ABSTRACT

Nowadays data mining plays a significant role in solving most of the critical problems. Especially in medical field, storage of medical data in databases creates a large mass of data which is being the most preferred application area for data mining methods. Obtaining information quickly and efficiently through the biomedical systems which use data mining techniques, provide a great benefit to physicians and patients.

In this thesis k-nearest neighbor (KNN) and k-means algorithms have been investigated in detail. Also using these algorithms an assistant decision-making system which is available to physicians in the medical field, can predict for the diagnosis of dermatological diseases and provide an opportunity to analyze the relationships between characteristics of patient records has been designed and carried out.

**Keywords**

Data mining, K nearest neighbour, K-means, Biomedicine, Medical Informatics, Dermatology.

# TEŞEKKÜR

Tez çalışmamda bana yardım ve desteğini hiçbir zaman esirgemeyen tez danışmanım, değerli hocam Sayın Yrd. Doç. Dr. Metin KESLER'e, konu seçimimdeki yardımları için bana değerli zamanını ayıran Sayın Doç. Dr. Nevcihan DURU'ya, Bilecik Üniversitesi Bilgisayar Mühendisliği Bölümü hocalarıma, çalışma arkadaşlarıma ve son olarak da hayatım boyunca her zaman yanımda olan ve beni teşvik eden çok sevdiğim aileme, gösterdikleri manevi destek ve anlayışları için içtenlikle teşekkür ederim.

Hatice ÇATALOLUK

# İÇİNDEKİLER

# ÇİZELGELER DİZİNİ

**Sayfa No**

# ŞEKİLLER DİZİNİ

**Sayfa No**

# 1. GİRİŞ

Son yıllarda hızlı şekilde gelişen teknolojiler sayesinde, birçok alanda veriler kolaylıkla depolanabilmektedir. Teknolojinin getirdiği bu elverişli şartlar hızlı bir veri artışını da meydana getirmektedir. Bu hızlı veri artışı sonucu oluşan büyük veri yığınlarını inceleyebilmek ve içlerinden geleceğe yönelik anlamlı bilgiler elde etmek için geleneksel sorgulama ve raporlama araçları yetersiz kaldığından veri madenciliği yöntemlerine olan ihtiyacı oluşturmuştur. Dolayısıyla büyük veri yığınlarının oluştuğu birçok farklı sektörde veya alanda veri madenciliği kullanılabilmektedir. Ancak eldeki bu verilerin kaliteli ve güvenilir olması da son derece önemlidir. Çünkü kullanılan veriler elde edilecek bilgilerin de kalitesini etkilemektedir.

Veri madenciliğin kullanıldığı alanlardan biri olan tıp (sağlık-bakım) alanı da veri madenciliğinin kullanıldığı en önemli alanlardan birisidir. Çünkü tıp alanındaki veriler hayati önem taşıyan verilerdir. Bu durum, tıbbi veriler arasından elde edilecek bilgi keşiflerini de önemli kılmaktadır.

Tıbbi alanda kullanılan veri madenciliğinin, ileride birçok klinik araştırma gerektiren, hem ekonomik hem de insan sağlığı açısından sakıncaları olan tıbbi araştırmaların yerini bir nebze de olsa doldurarak tıbbi araştırmalar için yeni bir ufuk sağlayacağı düşünülmektedir (Kaya vd., 2003).

Tarihçe olarak hastane bilgi sistemlerindeki veri madenciliği tekniklerinin ilk kullanımı 1970'lere dayanmaktadır. Daha sonraki yıllarda bu gelişme uzman sistemlerle devam etmiştir. Ancak uzman sistemler tıp alanında güçlü araçlar sunmasına rağmen, tıp alanındaki verilerin hızlı bir biçimde değişmesi ve uzmanlar arasında oluşan görüş farklılıkları nedeniyle çok yaygınlaşamamışlardır. Takip eden yıllar içerisinde özellikle 1990'lı yıllarda hastaların ilerideki sağlık durumları ve maliyet tahminleri gibi konuları araştırmak için sinir ağları kullanılmaya başlanmıştır (Yıldırım vd., 2008).

Bu tez çalışmasının amacı k-en yakın komşu (KNN) ve k-means algoritmalarını detaylı bir şekilde incelemek ve bu algoritmaları kullanarak tıp alanında hekimlerin kullanabileceği dermatolojik hastalıkların teşhisi için tahmin yapabilen ve hasta

kayıtlarının nitelikleri arasındaki ilişkileri analiz etme imkanı sunabilen yardımcı bir karar verme sistemi tasarlayıp gerçekleştirmektir.

Tez çalışmasında kullanılan dermatoloji veri setini literatürde kullanmış bazı veri madenciliği çalışmalarının olduğu görülmüştür. Bunlardan biri; KNN, Bayesian, karar ağacı ve Dempster-Shafer teorisi yöntemlerini göğüs kanseri ve cilt yaralarını sınıflandırmak için kullanan ve bu veri madenciliği yöntemlerinin karşılaştırılmasının yapıldığı çalışmadır. Yapılan çalışma sonucunda veri madenciliği yöntemlerinin farklı veri seti için farklı performanslar gösterdiği söylenmiştir. Karar ağacı için %94.9, Bayesian için %95 ve KNN için ise %42.5 oranında bir doğruluk değeri elde edilmiştir (Aslandogan vd., 2004). Bu çalışmaya göre, aynı veri seti olan dermatoloji veri seti için, tez çalışmamda Bölüm 5.4'de belirtilen KNN algoritmasının performansı çok daha yüksek çıkmıştır. Dermatoloji veri seti üzerinde uygulanan normalizasyon ve veri temizleme gibi önişleme işlemleri sayesinde bu sonuç elde edilmiştir.

Ubeyli ve Guler tarafından yapılan bir çalışmada ise dermatoloji veri seti üzerinde uyarlamalı sinirsel-bulanık çıkarım sistemi (ANFIS) kullanılarak, %95.5 oranında bir sınıflandırma doğruluk değeri elde edilmiştir (Ubeyli ve Guler, 2005).

Dermatoloji veri seti üzerinde yapılan başka bir çalışmada ise, Ubeyli veri seti üzerinde yapay sinir ağı tabanlı combined neural network models (CNN) yöntemini uygulayarak, %97.77 oranında bir sınıflandırma doğruluk değeri elde etmiştir. (Ubeyli, 2009).

Xie ve Wang, yine dermatoloji veri seti üzerinde hastalık teşhisi için, destek vektör makineleri ile yeni bir hibrit özellik seçme yöntemini bir arada kullanan bir sınıflandırma modeli oluşturmuştur. Bu sınıflandırma modeli için ise %98.61 oranında bir doğruluk değeri elde edilmiştir (Xie ve Wang, 2011).

Dermatoloji veri seti dışında daha başka birçok tıbbi veritabanları ya da veri setleri üzerinde yapılmış veri madenciliği çalışmaları vardır. Bunlara örnek olarak yapılan bir çalışmada, karaciğer hastalığının teşhis doğruluğunu artırmak için kapsamlı bir analitik yapı sağlamayı amaçlayan zeki bir teşhis modeli kurmak için sınıflandırma ve regresyon ağacı (CART) ile olay-tabanlı nedensellik (CBR) teknikleri uygun bir şekilde kullanılmıştır (Lin, 2009).

Bir doktora çalışmasında ise elektroensefolagram (EEG) verileri üzerinde 11 farklı veri madenciliği yöntemi uygulanarak epileptik aktivitelerin olup olmadığının belirlenmesi işlemi ve kullanılan bu veri madenciliği yöntemleri arasında bir performans karşılaştırması yapılmıştır (Albayrak, 2008).

Başka bir çalışmada da, veri madenciliği teknikleri kullanılarak gırtlak kanseri operasyonlarından oluşan tıbbi verileri analiz etmek için, bir yazılım aracı geliştirilmiştir. Bu medikal yazılımda da k-means kümeleme algoritması, veri kümesindeki yoğunlukları belirtmek ve iki boyutlu grafiklerde görüntülemek için kullanılmıştır. Bu uygulamada gırtlak kanseri operasyonlarının nedensel bağlantılar içeren karakteristikleri veri madenciliği teknikleri ile ortaya çıkarılmıştır (Dincer ve Duru, 2009).

Literatürde medikal veritabanları üzerinde yapılmış çok sayıda veri madenciliği çalışmasının olmasına rağmen, medikal veritabanları için geliştirilen yazılım uygulamalarının sayısının az olduğu görülmüştür.

Bu tez kapsamında yapılan çalışmalar altı bölümde anlatılmıştır. İlk bölümde tez çalışması için ele alınan problemin tanımından ve öneminden, benzer literatür çalışmalarından, tezin amacından ve kapsamından bahsedilmiştir.

İkinci bölümde, veri madenciliğine bir giriş olması amacıyla veri madenciliği ile ilgili temel kavram ve tanımlar belirtilmiştir. Veri madenciliğinin süreci, yöntemleri ile uygulamaları ve kullanım alanları açıklanmıştır.

Üçüncü bölümde uygulama içerisinde kullanılan veri madenciliği yöntemlerinden olan KNN sınıflandırma algoritması detaylı bir şekilde açıklanmış ve incelenmiştir.

Dördüncü bölümde de uygulama içerisinde kullanılan diğer bir veri madenciliği yöntemi olan k-means kümeleme algoritması detaylı bir şekilde ele alınmış ve açıklanmıştır.

Beşinci bölümde, tez çalışması için gerçekleştirilmiş olan dermatolojik hastalıklara yönelik yardımcı karar verme sistemi hakkında bilgi verilmiştir. Geliştirilen yazılımın kullanıcı arayüzleri tanıtılarak kullanımları açıklanmıştır. Ayrıca bu bölümde

uygulama içerisinde kullanılan KNN algoritmasına ait elde edilen test sonuçları detaylı bir şekilde belirtilmiştir.

Son olarak altıncı bölümde ise, yapılan tez çalışması sonucu elde edilen kazanımlar belirtilerek çalışma ile ilgili bir değerlendirme yapılmıştır.

# 2. VERİ MADENCİLİĞİ

## 2.1 Veri Madenciliği Nedir?

Günümüzde veriler çeşitli depolama cihazlarında çok hızlı bir şekilde birikmektedir. Verilerin bu şekilde birikmesi bir yandan da değerli bilgileri oluşturmaktadır. Ancak veriler arasında saklanmış bu değerli bilgilere veri analizi tekniklerinin yardımı olmadan ulaşmak oldukça zordur. Bu durum makine öğrenmesinden ayrı yeni bir alanın geliştirilmesine neden olmuştur. İşte bu yeni alan, veri madenciliğidir (Akpınar, 2000).

Literatürde 1980'lerde yer almaya başlayan veri madenciliği istatistik, makine öğrenmesi, veritabanı yönetimi, yapay zeka ve örüntü tanıma gibi birçok disiplinin kesişiminde yer alan yeni bir bilim dalıdır (Hand vd., 2001).

Veri madenciliği bilgisayar endüstrisinde hızlı büyüyen alanlardan birisidir. Veri madenciliği yöntemleri bilim ve sanayideki çoğu probleme hitap etmektedir. Biyoinformatik, eczacılık, bankacılık, perakende, spor, eğlence vb. gibi birçok alanlar için başarılı sonuçlar sağlamıştır (Wang ve Fu, 2005).

Veri madenciliği, çok büyük veriler arasından yararlı veya değerli bilginin bulunmasıdır. Gartner Group şirketine göre ise, "Veri madenciliği depolarda (veri ambarında) saklanan çok büyük boyuttaki verileri inceleyerek anlamlı yeni korelasyonların, örüntülerin ve eğilimlerin keşfedilmesi sürecidir." (Larose, 2005). Başka bir tanıma göre, veri sahibi için verilerin hem anlaşılabilir hem de yararlı olması amacıyla değişik yöntemlerle özetlenmesi ve veriler arasından beklenmeyen ilişkileri bulmak için büyük gözlemsel veri setlerinin analizidir (Hand vd., 2001).

## 2.2 Veri Ambarı

Veri ambarı, karar destek amacına yönelik çeşitli işlemsel veritabanlarındaki bilgileri depolamak için kullanılan bir veritabanı sistemidir. Veri ambarının, çeşitli veritabanlarındaki verileri sadece tek bir diske atarak oluşturulması mümkün değildir. Toplanan bu verilere, özellik isimleri ile kullanımları arasındaki olası tutarsızlıkları ortadan kaldırmak, özelliklerin ve değerlerin anlamsallıklarını (semantic) öğrenmek gibi birçok bütünleştirme görevlerinin uygulanması gerekmektedir. İşlemsel veritabanlarını

daha detaylı anlamayı ve daha fazla elle (manüel olarak) aracılık gerektirdiği için veri ambarlarını oluşturmak genelde masraflı, zor ve bazen yıllarca süren bir işlemdir (Hand vd., 2001). Bu yüzden veri madeniliği için her zaman bir veri ambarının olması zorunlu değildir. Bir veya daha fazla işlemsel veritabanından basit bir şekilde sadece okunabilir bir veritabanı çıkartarak da veri madenciliği uygulanabilmektedir (Two Crows Corporation, 1999) (Demirdümen, 2008).

Veri ambarlarının temel özellikleri aşağıda açıklanmıştır:

- Konu odaklıdır:

  Aynı olay veya varlık ile ilgili olan veriler birbirlerine bağlı bir yapıdadır. Örnek olarak, bir veri ambarı müşteri, ürün gibi varlıklar ya da satış, sipariş alma veya teslimat gibi olaylar için düzenlenmiş olabilmektedir (Silahtaroğlu, 2008).

- Bütünleşiktir:

  Veri ambarlarında birden fazla veritabanı birleştirilmiştir. Bunun yanı sıra veritabanlarına dosyalar, internet sayfaları vb. gibi farklı veri kaynaklarındaki veriler de aktarılmış olabilmektedir. Tüm bu veriler veritabanıyla bütünleştirilmiştir. Ayrıca veri içerinde tekrarlanan alanlar için gerekli önişleme uygulamaları da yapılmıştır (Silahtaroğlu, 2008).

- Zaman boyutu vardır:

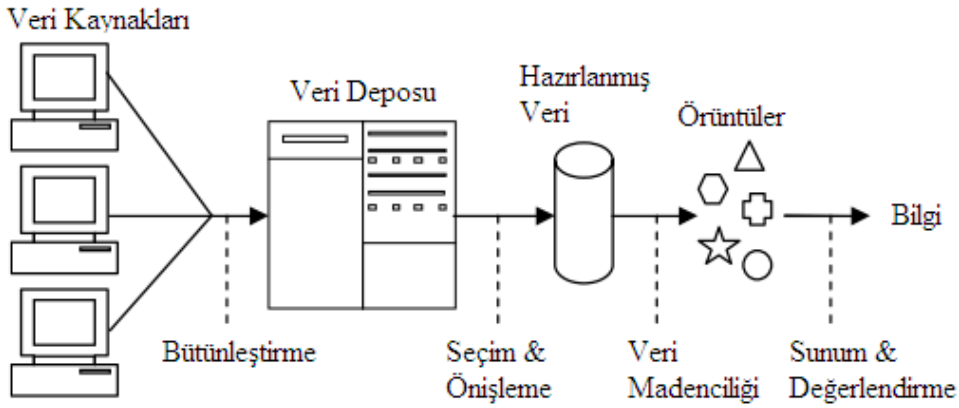
  Veri ambarında zaman kavramı vardır ve veri ambarındaki tüm veriler zamanın belirli bir anına aittir. Veri ambarındaki veriler geçmişteki değerlerle de ilgilidir. Zaman içindeki bir nokta ile veri birleştirilerek değerlendirilir. Bir bilgiye ait en az beş yıllık değerlerin veri ambarı içerisinde yer alması gerekmektedir (Özkan, 2008).

- Sadece okunabilirdir:

  Veri ambarındaki veriler sadece okunabilirdir. Bu yüzden veri ambarındaki veriler değiştirilemez, silinemez ve veri ambarına yeni veri eklenemez. Yani veri ambarlarında yeni veri giriş çıkışı için uygun bir mimari yoktur (Silahtaroğlu, 2008).

## 2.3 Veri Madenciliği Süreci

Çoğu kişi veri madenciliğinin bilgi keşfi (BK) kavramı ile aynı şeyi ifade ettiğini zannetmektedir. Bazılarına göre ise veri madenciliği BK'nin en önemli adımı olarak görülmektedir. Şekil 2.1.'de BK'nin süreci ve bu süreç içerisindeki veri madenciliğinin konumu gösterilmiştir.




Şekil. 2.1. Bilgi keşfinin (BK) süreci (Bramer, 2007).

### 2.3.1 Bütünleştirme

Bütünleştirme, farklı veri kaynaklarından elde edilen verilerin yeni bir veri kümesi altında birleştirilmesi işlemidir. Veri setlerinin birleştirilmesi kolay bir işlem değildir. Farklı kaynaklarda yer alan aynı nitelikler için farklı gösterim biçimleri (farklı derecelendirmeler veya değerler) kullanılmış veya birbirlerinden farklı niteliklermiş gibi gösterilmiş olabilmektedir. Bu durumda bu nitelikler aynı veri seti içerisine taşındıklarında, veri seti içerisinde uyumsuz ve gereksiz veriler söz konusu olacaktır. Bütünleştirme işlemi bu sorunların oluşmasını engellemeyi amaçlamaktadır.

### 2.3.2 Seçim ve önişleme

Seçim işleminde seçilen verilerin tüm veri uzayını temsil edebilmesi oldukça önemlidir. Çünkü veri tabanlarındaki işlem hızları artmasına rağmen büyük veritabanları üzerinde bir veya birden fazla modelin denenmesi oldukça zaman ve maliyet gerektirmektedir. Bunun yerine verinin bütününü temsil edecek şekilde bir parça üzerinde işlemler yapmak bu sorunları ortadan kaldıracaktır (Kaya vd., 2003).

Bu adımda veritabanı dışından da ihtiyaç duyulan veriler amaçlarına ve özelliklerine göre seçilebilmektedir.

Önişleme ise farklı ortamlardan elde edilen bu ham verilerin işlenerek veri madenciliği modelinin kullanabileceği biçime getirilmesi işlemidir (Şişaneci, 2009).

#### 2.3.2.1 Veri indirgeme

Veriler içerisindeki bazı değişkenler diğer değişkenlerle çok benzer bilgiler taşıyabilmektedir (Nisbet vd., 2009). Bu durumda aynı bilgi için birden çok değişkenin var olması gereksiz olmaktadır. Dolayısıyla aynı bilgiyi taşıyan değişkenlerden yalnızca biri veriler içerisinde saklanmalıdır. Buna veri indirgeme denmektedir. Bu işlem ile çalışma zamanının iyileştirilmesi hedeflenmektedir.

Gereksiz değişkenlerin elimine edilmesine ilave olarak, hedef değişkenle yüksek oranda ilişkili olan değişkenler de elimine edilebilmektedir. Bu işlem en optimum modelin üretilme olasılığını artırmaktadır (Nisbet vd., 2009). Buna da veri boyutu indirgemesi denmektedir.

PCA (Principal Components Analysis/Temel Bileşenler Analizi) ve SVD (Singular Value Decomposition/Tekil Değer Ayrıştırması) adı verilen çeşitli lineer cebir tabanlı yaklaşımlar boyut indirgeme konusunda yararlanılan başlıca tekniklerdir (Tan vd., 2006)(Bozkır, 2009). "Karhunen Loeve" yöntemi olarak da bilinen PCA yöntemi bir değişkenler kümesinin varyans-kovaryans yapısını, bu değişkenlerin doğrusal birleşimleri yoluyla açıklamaktadır. Böylelikle değişkenler kümesi üzerinde yorumlama ve boyut indirgemesi işlemini sağlayabilen çok değişkenli bir istatistiksel metottur (Han ve Kamber, 2001)(Silahtaroğlu, 2008)(Bozkır, 2009).

#### 2.3.2.2 Veri temizleme

Gerçek dünya verileri genellikle insan girişlerinden kaynaklanan hatalar yüzünden eksik, tutarsız ve gürültülü olmaya yatkındırlar. Veri temizleme yordamları kaybolan verileri tamamlamayı, veri içerisindeki ayrık gözlemleri tanımlayarak gürültülü verileri gidermeyi ve tutarsızlıkları düzeltmeye çalışmaktadır (Han ve Kamber, 2001).

Çoğu zaman veri setinin değişkenleri, veri seti için uygun olmayan değerlere (boş alan) sahip olabilmektedir. Bu kayıtların veri seti içerisinden silinmesi gerekmektedir. Çünkü bu kayıtların modelleme veri seti içerisinde yer alması sadece modelin kafasını karıştıracak ve modelin tahmin etme gücünü azaltacaktır (Nisbet vd., 2009).

#### 2.3.2.3 Veri dönüştürme

Bazı veri madenciliği algoritmaları eğer üzerinde çalıştığı sayısal değerler standartlaştırılmışsa en iyi şekilde sonuç vermektedir. Standartlaştırma (normalizasyon) tüm sayısal değerlerin ortak bir aralığa dönüştürülmesi anlamına gelmektedir (Nisbet vd., 2009).

Min-max normalizasyonu ve z-score normalizasyonu nümerik nitelikler üzerinde uygulanabilecek standartlaştırma yöntemlerindendir.

Min-max normalizasyonu ile nümerik veriler 0 ile 1 arasındaki sayısal değerlere dönüştürülmektedir. Eşitlik E.2.1'te min-max normalizasyonunun bağıntısı verilmiştir. Bu bağıntıdaki $X'$normalleştirilmiş değeri, X gözlem değerini, $\mathrm{X_{min}}$ gözlem değerlerinin en küçüğünü ve $\mathrm{X_{max}}$ ise gözlem değerlerinin en büyüğünü temsil etmektedir.

$$X' = \frac{\mathrm{X} - \mathrm{X_{min}}}{\mathrm{X_{max}} - \mathrm{X_{min}}} \qquad \text{(E.2.1)}$$

Z-score standartlaştırma yönteminde ise verilerin aritmetik ortalaması ve standart sapması dikkate alınarak yeni değerlere dönüşüm işlemi gerçekleştirilmektedir. Z-score standartlaştırması için eşitlik E.2.2.'deki bağıntı kullanılmaktadır.

$$X' = \frac{\mathrm{X} - \overline{\mathrm{X}}}{\sigma_{\mathrm{x}}} \qquad \text{(E.2.2)}$$

Bu bağıntıdaki $X'$ dönüştürülmüş gözlem değerini, X gözlem değerini, $\overline{\mathrm{X}}$ gözlem değerlerinin aritmetik ortalamasını ve $\sigma_{\mathrm{x}}$ ise gözlem değerlerinin standart sapmasını temsil etmektedir.

### 2.3.3 Veri madenciliği yönteminin uygulanması

Yukarıda bahsedilen adımlardan sonra, veriler veri madenciliği algoritmalarının uygulanabilmesi için hazır hale gelmiş olmaktadır. Bu adımda probleme yönelik olarak veri madenciliği temel modellerinden sınıflandırma, kümeleme veya birliktelik kuralları uygulanmaktadır.

### 2.3.4 Sunum ve değerlendirme

Veri madenciliği yöntemleri veriler üzerinde uygulandıktan sonra diğer bir aşama ise, sunum ve değerlendirmedir. Veri madenciliği yöntemlerinden elde edilen sonuçlar

üzerinde çeşitli düzenlemeler yapıldıktan sonra (görselleştirme katarak) kullanıcılar tarafından anlaşılabilir bir hale getirilerek sunum gerçekleştirilmektedir. Veri madenciliği yöntemlerinin sonuçlarını göstermek için genellikle grafikler kullanılmaktadır (Özkan, 2008).

Değerlendirme aşamasında ise keşfedilen bilgi geçerlilik, yenilik, yararlılık ve basitlik ölçütlerine göre değerlendirilmektedir (Sever ve Oğuz, 2002) (Haberal, 2007).

## 2.4 Veri Madenciliği Yöntemleri

Veri madenciliğinde temel olarak üç ana yöntem vardır. Bunlar sınıflandırma, kümeleme ve birliktelik kurallarıdır. Veriler içerisindeki gizli örüntülerin ortaya çıkarılması amacıyla sınıflandırma modelleri kullanılmaktadır. Verilerin kendi aralarındaki benzerliklerden yola çıkarak gruplandırılması kümeleme yöntemleri ile gerçekleştirilmektedir. Birliktelik kuralları yöntemleri ile de gözlemlerin birbirleriyle olan ilişkisi ele alınarak hangi olayların birlikte gerçekleştiği ortaya koyulmaktadır (Özkan, 2008).

### 2.4.1 Sınıflandırma

Veri madenciliğinde sınıflandırma çok çeşitli alanlarda ve sıkça kullanılan bir yöntemdir. Hangi sınıfa ait olduğu bilinmeyen kayıtların sınıflarının belirlenmesi için sınıflandırma algoritmalarına ihtiyaç duyulmaktadır. Dolayısıyla sınıflandırma modeli tahminleyici bir modeldir. Sınıflandırmada amaç, verilerin içerdiği ortak özellikleri kullanarak söz konusu bu verileri ayrıştırmaktır. Sınıflandırma algoritmaları bu amacı gerçekleştirebilmek için bir sınıflandırma modeli kurmaya çalışmaktadır (Özkan, 2008).

Sınıflandırma matematiksel olarak aşağıdaki gibi tanımlanabilir:

$D = \{t_1, t_2, \dots t_n\}$ bir veritabanı ve her bir $t_i$ bir kayıt olmak üzere,

$C = \{C_1, C_2, \dots C_m\}$ ise m adet sınıftan oluşan sınıflar kümesi olmak üzere,

$f: D \rightarrow C$ ve her bir $t_i$ bir sınıfa dahil olmalıdır.

Her bir $C_j$ ayrı bir sınıftır ve her bir sınıf kendisine ait kayıtları içerir. Yani,

$$C_j = \{t_i \mid f(t_i) = C_j, 1 \leq i \leq n, ve\ t_i \in D\}.$$

Bir sınıflandırma işleminde mevcut sınıflar (bağımlı değişkenler) hem kategorik hem de sürekli değer taşıyabilmektedir; bu durumda sınıflandırma işlemi regresyon ve çok terimli regresyona yaklaşmaktadır (Silahtaroğlu, 2008)(Akpınar, 2000).

Sınıflandırma algoritmalarında bir öğrenme işlemi söz konusudur. Dolayısıyla eldeki verilerin, bir kısmı eğitim diğer kısmı ise test işleminde kullanılmak üzere, ikiye ayrılması gerekmektedir.

Sınıflandırma algoritmalarına örnek olarak Bayesyen sınıflandırma algoritması, k-en yakın komşu algoritması, karar ağaçları, sınıflandırma ve regresyon ağaçları (CART), destek vektör makinesi ve yapay sinir ağları temelli algoritmalar verilebilir.

### 2.4.2 Kümeleme

Kümeleme, eldeki verileri sınıflara ya da kümelere ayırarak yapılan bir veri madenciliği modelidir (Haberal, 2007). Kümeleme analizi eldeki verileri kümelere ayırmaya çalışırken uzaklık veya benzerlik ölçütüne ihtiyaç duymaktadır.

Her biri birtakım özelliklerden oluşan veri noktalarına ve aralarındaki benzerlik ölçütüne göre tespit edilen kümeler için, aynı kümede yer alan veri noktaları birbirlerine yakındır. Ancak birbirlerinden farklı kümede yer alan veri noktaları birbirlerine daha az yakındır (Alluri, 2005)(Kumar ve Joshi, 2005).

Kümeleme yöntemleri için amaç; kümeler arası benzerliğin düşükken, küme içi benzerliğin yüksek olmasını sağlamaktır. Bu hedefi sağlayan iyi bir kümeleme algoritmasının veriler üzerinde uygulanması sonucu elde edilecek kümeler de nitelikli olacaktır.

Kümeleme işlemi sınıflandırma da olduğu gibi eldeki verileri gruplandırmaya çalışmaktadır. Ancak sınıflandırma işleminde verilerin sınıfları önceden bilinirken kümeleme işleminde verilerin sınıfları önceden bilinmemektedir.

Kümeleme yöntemleri market araştırması, veri analizi, örüntü tanıma ve imge işleme gibi pek çok uygulamada geniş bir şekilde kullanılmaktadır. Örneğin pazarlamacılar için satın alma örüntülerine göre müşterilerin gruplarını karakterize etmede ve farklı müşteri gruplarını keşfetmede yardımcı olabilmektedir. Biyoloji alanında ise, bitki ve hayvan taksonomilerini elde etmek, benzer işlevselliğe sahip

genleri kategorize etmek ve popülasyonların doğasında olan yapılar hakkında fikir edinmek için kullanılabilmektedir (Han ve Kamber, 2001).

Kümeleme yöntemleri de kendi aralarında temel olarak aşağıdaki gibi sınıflandırılmaktadır (Han ve Kamber, 2001).

1. Bölümlemeli (Partitioning) Yöntemler
2. Hiyerarşik (Hierarchical) Yöntemler
3. Yoğunluğa Dayalı (Density-based) Yöntemler
4. Izgara Temelli (Grid-based) Yöntemler
5. Model Tabanlı (Model-based) Yöntemler

### 2.4.3 Birliktelik kuralları

Birliktelik kurallarının amacı, büyük veri setlerindeki kategorik değişkenlerin belirli değerleri arasındaki ilişkileri ya da birliktelikleri tespit etmektir. Bu teknik, analist ve araştırmacılara büyük veri setlerindeki gizli örüntüleri ortaya çıkarmaya izin vermektedir (Nisbet vd., 2009).

Birliktelik kurallarında iki önemli kavram söz konusudur. Bunlar destek (support) ve güven (confidence) sınırları kavramlarıdır. Küçük bir veri setinden bile çok fazla birliktelik kuralı elde edilebildiğinden dolayı, bu sınırlar kullanılarak kural sayısını aza indirmek ve geçerliliği olan ilişkiler üzerinde yoğunlaşmak hedeflenmektedir.

Birliktelik kuralları yöntemi temel olarak iki aşamadan oluşmaktadır (Şen, 2008):

1. Sık geçen nesne kümelerinin belirlenmesi: Buna göre her nesne kümesinin sık geçenler kümesinde yer alabilmesi için minimum destek şartını sağlaması gerekmektedir. Yani her nesnenin destek değerinin önceden belirlenmiş olan minimum destek değerinden büyük olmalıdır.
2. Sık geçen nesne kümelerinden güçlü ilişki kurallarının oluşturulması: Bunun için ise, elde edilen ilişki kurallarının minimum destek ve minimum güven şartını sağlamaları gerekmektedir.

Birliktelik kuralları algoritmaları basit kategorik değişkenleri, ikili değişkenleri ve/veya çoklu hedef değişkenleri incelemek için kullanılabilmektedir. Algoritma veri içerisindeki mevcut farklı kategorilerin sayısını istemeden veya önemli birlikteliklerin karmaşıklığını veya maksimum faktöriyel derecesi ile ilgili herhangi bir ön bilgiye

ihtiyaç duymadan birliktelik kurallarını belirleyebilmektedir (apriori ve türevleri hariç)(Nisbet vd., 2009).

## 2.5 Veri Madenciliği Uygulamaları Ve Kullanım Alanları

Veri madenciliği uygulama alanı olarak oldukça geniş bir alana sahiptir. Veri madenciliğinin uygulanabilmesi için sadece veritabanlarında depolanmış büyük miktarlardaki verilerin olması yeterlidir. Dolayısıyla pek çok sektör veya iş için kullanılabilir bir disiplindir.

Günümüzde pek çok alana hitap eden veri madenciliğinin başlıca kullanım alanları olarak aşağıdakiler sayılabilir (Eker, 2004)(Telcioğlu, 2007):

1. Pazarlama
   - ✓ Müşteri segmentasyonu
   - ✓ Müşterilerin demografik özellikleri arasındaki bağlantıların kurulması
   - ✓ Çeşitli pazarlama kampanyaları
   - ✓ Mevcut müşterilerin elde tutulması için geliştirilecek pazarlama stratejilerinin oluşturulması
   - ✓ Pazar sepeti analizi
   - ✓ Çapraz satış analizleri
   - ✓ Müşteri değerlemesi
   - ✓ Müşteri ilişkileri yönetimi
   - ✓ Çeşitli müşteri analizleri
   - ✓ Satış tahminleri
2. Bankacılık
   - ✓ Farklı finansal göstergeler arasındaki gizli korelasyonların bulunması
   - ✓ Kredi kartı dolandırıcılıklarının tespiti

- ✓ Müşteri segmentasyonu
- ✓ Kredi taleplerinin değerlendirilmesi
- ✓ Usulsüzlük tespiti
- ✓ Risk analizleri
- ✓ Risk yönetimi

3. Sigortacılık
   - ✓ Yeni poliçe talep edecek müşterilerin tahmin edilmesi
   - ✓ Sigorta dolandırıcılıklarının tespiti
   - ✓ Riskli müşteri tipinin belirlenmesi
4. Perakendecilik
   - ✓ Satış noktası veri analizleri
   - ✓ Alışveriş sepeti analizleri
   - ✓ Tedarik ve mağaza yerleşim optimizasyonu
5. Borsa
   - ✓ Hisse senedi fiyat tahmini
   - ✓ Genel piyasa analizleri
   - ✓ Alım-satım stratejilerinin optimizasyonu
6. Telekomünikasyon
   - ✓ Kalite ve iyileştirme analizleri
   - ✓ Hisse tespitleri
   - ✓ Hatların yoğunluk tahminleri
7. Sağlık ve İlaç

- ✓ Test sonuçlarının tahmini
- ✓ Ürün geliştirme
- ✓ Tıbbi teşhis
- ✓ Tedavi sürecinin belirlenmesi

8. Endüstri

- ✓ Kalite kontrol analizleri
- ✓ Lojistik
- ✓ Üretim süreçlerinin optimizasyonu

9. Bilim ve Mühendislik

- ✓ Ampirik veriler üzerinde modeller kurarak bilimsel ve teknik problemlerin çözümlenmesi.

Bu kullanım alanlarına yönelik olarak bazı veri madenciliği uygulamaları da aşağıda belirtilmiştir:

- Parmak izi veya yüz tanıma teknolojileri ile kimlik tespit edilmesi.
- Bir kanser hastasının kemoterapiye cevap verme olasılığının tahmin edilmesi ve böylelikle bakım kalitesini etkilemeden sağlık-bakım maliyetlerinin azaltılması (Bramer, 2007).
- Bir kredi kartı şirketinin müşteri işlemlerinin verilerinden oluşan veri ambarını kullanarak dolandırıcılıkların tespit edilmesi (Bramer, 2007).
- DNA verisi üzerinde veri madenciliği yöntemleri ile analiz yapılarak hastalıklara yol açan gen sıralama örneklerinin tespit edilmesinin kolaylaştırılması (Silahtaroğlu, 2008).
- Bir süpermarket zincirinin müşteri işlemleri verileri kullanılarak yüksek değerli müşteri hedeflemesinin optimize edilmesi (Bramer, 2007).
- Tümleşik devre yongalarının (chiplerin) üretim kusurlarının azaltılması (Bramer, 2007).

- Telefon hatlarındaki parazitlenmelerden kaynaklanacak veri kayıplarının ve bununla ilgili olarak da konuşmalarda ortaya çıkan gürültünün yok edilmesi (Silahtaroğlu, 2008).
- Televizyon yöneticilerinin, pazar payını en üst düzeye çıkarmak ve reklam gelirlerini artırmak için televizyon programlarını düzenlemelerine izin veren, televizyon programları için seyirci payının tahmin edilmesi (Bramer, 2007).

# 3. K-EN YAKIN KOMŞU (KNN) ALGORİTMASI

## 3.1 Giriş

K-en yakın komşu (KNN) yönteminin tarihçesi onlarca yıl öncesine dayanmaktadır (Chakrabarti vd., 2009). KNN algoritması 1950'li yılların başlarında ilk kez tanımlanmış olmasına rağmen 1960'lı yıllara kadar popüler olamamıştır. Çünkü o zamanlardaki hesaplama gücü, k-en yakın komşu algoritmasının ihtiyaç duyduğu büyük çapta bir eğitim veri seti için henüz yeterli değildi. Ancak artan hesaplama gücü ile birlikte, algoritma özellikle örüntü tanıma alanında yaygın bir şekilde kullanılmaktadır (Han ve Kamber, 2001).

K-en yakın komşu algoritması genellikle sınıflandırma amaçlı kullanılmasına rağmen kestirim ve tahminleme için de kullanılabilen bir veri madenciliği yöntemidir (Larose, 2005). Coğrafi bilgi sistemlerinde de oldukça çok kullanılan bir yöntemdir (Beyer vd., 1999).

K-en yakın komşu algoritması çoğunlukla tüm özellikler sürekli değer (lineer) olduğunda tercih edilmektedir. Ancak algoritmanın kategorik özelliklerle de başa çıkması için üzerinde değişiklik yapılabilmektedir (Bramer, 2007).

Algoritmanın adının içerisinde de geçen k değeri, komşu olan kayıtların sayısını temsil etmektedir. KNN algoritması, verilen n adet prototip örüntüye ve bu örüntülerin doğru sınıflandırılmasına göre, sınıfı bilinmeyen bir örüntüyü en yakın komşu gruba atamaktadır (Tosun, 2006).

K en yakın komşu algoritması örnek tabanlı öğrenmenin bir örneğidir. Örnek tabanlı öğrenmede eğitim örnekleri birebir saklanmaktadır ve bilinmeyen bir test örneğinin eğitim setinin hangi üyesine en yakın olduğuna karar vermek için uzaklık fonksiyonu kullanılmaktadır. En yakın eğitim örneği bulunduktan sonra, test örneği için sınıf tahmin edilmektedir (Chakrabarti vd., 2009). Bunun için de test örneği eldeki mevcut eğitim kümesindeki en benzer olduğu kayıtlarla karşılaştırılmaktadır (Larose, 2005).

Örnek tabanlı veya başka deyişle komşuya dayalı öğrenme yöntemindeki algoritmalar, verilen bir test kümesini sınıflandırmak için son ana kadar herhangi bir

model oluşturmadan beklemektedir. Yani eğitim seti verildiğinde algoritma eğer gerekli ise önişleme yaparak depolar ve test kümesi verilene kadar bekler. Algoritma test veri setini gördüğü zaman, test veri setini sınıflandırmak için depo edilen eğitim setine benzerliğine göre genelleştirme yapmaktadır. Dolayısıyla örnek tabanlı öğrenme algoritmaları eğitim veri setini göstermek için az iş, sınıflandırma ve tahminleme yapmak için ise çok iş yapmaktadırlar. Çünkü örnek tabanlı öğrenme algoritmaları örnekleri saklayarak depolamaktadır (Han ve Kamber, 2001).

Algoritmanın eğitim örneklerinin sayısı arttıkça, KNN algoritması birden fazla yakın komşu kullanmak için sezgisel mantık kullanmaktadır. Ancak sadece birkaç örnek olduğunda bu tehlikeli bir durum olmaktadır. Bu durum; k ve örnek sayısının her ikisi de sonsuz olduğunda $k/n \rightarrow 0$ şeklinde de gösterilebilmektedir. Bu nedenle veri kümesi için hata olasılığı teorik olarak minimuma yaklaşmaktadır (Chakrabarti vd., 2009).

K en yakın komşu algoritması gibi örnek tabanlı öğrenme yöntemleri için, mümkün olduğunca nitelik değerlerinin farklı kombinasyonları ile dolu, zengin bir veritabanına erişmek hayati önem taşımaktadır. Dolayısıyla özellikle eğitim sınıfında ender rastlanan sınıflandırmalar veritabanında yeterli düzeyde temsil edilmelidir. Böylelikle algoritma sadece yaygın sınıflandırmaları tahmin etmeyecektir. Bu yüzden veri setinin dengeli olması gerekmektedir. Dengelemeyi gerçekleştirmek için kullanılan başka bir yöntem ise; daha yaygın sınıflandırmalı kayıtların oranının azaltılmasıdır (Larose, 2005).

K en yakın komşu algoritmasının en büyük avantajlarından birisi uygulama açısından kolay olmasıdır. Üstelik kayıtların özellikleri dikkatli seçilmiş ve uzaklık hesaplamalarında dikkatli ağırlıklandırılmış ise, oldukça iyi sonuç vermektedir (Nisbet vd., 2009).

Örnek tabanlı öğrenme algoritmaları bir sınıflandırma veya tahminleme yaparken, işlemsel olarak pahalıdırlar. Verimli depolama tekniklerine ihtiyaç duymaktadırlar ve paralel donanım üzerinde gerçekleştirmek için çok uygundurlar. Bu yöntem algoritmaları doğal olarak artımlı öğrenmeyi desteklemektedir. Diğer öğrenme algoritmaları tarafından kolayca tanımlanamayan hiperpoligonal şekillere sahip kompleks karar uzaylarını modelleyebilmektedirler (Han ve Kamber, 2001).

## 3.2 Algoritmanın Çalışma Şekli

K en yakın komşu algoritması karşılaştırma yoluyla öğrenmeye dayalıdır. Yani algoritma verilen bir test veri setini benzeri olan eğitim örnekleri ile karşılaştırmaktadır. Eğer eğitim örnekleri n adet özellikten oluşmaktaysa, her örnek n-boyutlu bir uzayda bir noktayı temsil etmektedir. Böylelikle tüm eğitim veri seti n-boyutlu örüntü uzayında saklanmaktadır. Bilinmeyen bir örnek verildiğinde, k en yakın komşu algoritması örüntü uzayında bu örneğe en yakın olan eğitim örneklerini aramaktadır. Bu eğitim örnekleri, bilinmeyen yeni örneğin, k en yakın komşusu olmaktadır. Algoritmadaki yakınlık kavramı uzaklık ölçüsü açısından ele alınmaktadır (Han ve Kamber, 2001). Genellikle uzaklık ölçütü olarak Öklid ve Manhattan uzaklık ölçütleri kullanılmaktadır (Tan vd., 2006).

K en yakın komşu algoritmasında, bilinmeyen örneğe k en yakın komşularının arasından en yaygın olan (sık rastlanan) sınıf atanmaktadır (Han ve Kamber, 2001).

Algoritmanın çalışma ilkesi aşağıda özetlenmektedir (Wu vd., 2008):

***Girdi :*** *D; eğitim örneklerinin veri seti, z; özellik değerlerinin bir vektörü olan test veri örneği ve L; veri noktalarını etiketlemek için kullanılacak sınıflar kümesi.*

***Çıktı :*** $c_z \in L$, *z'nin sınıfı*

***foreach*** *veri noktası* $y \in D$ ***do***
| *z ve y arasındaki uzaklığı, d(z, y), hesapla;*
***End***

*z için k en yakın eğitim veri noktalarının kümesini,* $N \subseteq D$, *seç;*

$$c_z = \underset{v \in L}{argmax} \sum_{y \in N} I(v = class(c_y))$$

Burada I(·) eğer argümanı doğru ise 1 değerini aksi halde 0 değerini geri döndüren bir gösterge fonksiyonudur.

Algoritmanın depolama karmaşıklığı O(n)'dir. Burada n, eğitim veri setindeki kayıt sayısını göstermektedir. Algoritmanın zaman karmaşıklığı da O(n)'dir. Çünkü hedef ile her eğitim veri nesnesinin arasındaki uzaklığı hesaplamaya ihtiyacı vardır. Ancak k en yakın komşu algoritmasının diğer sınıflandırma teknikleri gibi sınıflandırma modelini kurmak için zamana ihtiyacı yoktur (Wu vd., 2008).

## 3.3 Kombinasyon Fonksiyonu

K-en yakın komşu algoritması hangi kayıtların yeni ve sınıflandırılmamış kayda en yakın olduklarına karar vermek için bir metoda ihtiyaç vardır. Ayrıca bulunan bu benzer kayıtların, yeni kayıtın sınıflandırma kararını elde etmek için ne şekilde birleştirileceğinin belirlenmesi gerekmektedir (Larose, 2005). Dolayısıyla algoritmanın bir kombinasyon (birleştirme) fonksiyonuna ihtiyacı vardır. Kombinasyon fonksiyonu için kullanılan iki yöntem vardır. Bunlar basit oylama ve ağırlıklı oylama metotlarıdır.

### 3.3.1 Basit Oylama

Sınıf etiketlerini kombine etmek için en basit yöntemdir. Bu yönteme göre sınıflandırma kararı çoğunluk oya göre belirlenmektedir. Ancak bu yöntem, eğer k yakın komşuların test örneğine olan uzaklıkları birbirlerine göre çok geniş değerlerde değişirse, bu k yakın komşuların içlerinden en yakınları sınıf değerini belirlemede daha etkin olmasına karşın, bu durum dikkate alınmamaktadır.

### 3.3.2 Ağırlıklı Oylama

Ağırlıklı oylama (Weighted Voting), yeni kayıta daha yakın veya başka deyişle benzer olan komşuların daha uzak komşulara göre daha ağırlıklı etki göstermesi ilkesine dayanan bir yöntemdir. Böylelikle sınıflandırma kararında yakın komşuların daha fazla söz hakkı olmaktadır. Ayrıca ağırlıklı oylamada bir kayıt için birden fazla sınıflandırma kararının eşit oy çıkması daha az olası bir durumdur ve böylelikle ağırlıklı oylama, algoritmanın kesin tek bir sonuç vermesini kolaylaştırmaktadır (Larose, 2005).

Ağırlıklı oylama yönteminde bir veri kaydının sınıflandırmaya etkisi, sınıflandırılacak yeni veri kaydına olan uzaklığı ile ters orantılıdır (Larose, 2005). Söz konusu olan ağırlıklı oylama yöntemi eğitim verileri için aşağıdaki bağıntıya göre ağırlıklı uzaklıkları hesaplamaktadır (Özkan, 2008):

$$d(i,j)' = \frac{1}{d(i,j)^2} \qquad \text{(E.3.1)}$$

Bu bağıntıdaki *d(i,j)* ifadesi i ve j veri kayıtları arasındaki Öklid uzaklığını temsil etmektedir. Sınıf değerleri arasında en büyük ağırlıklı oylama değerine sahip olan, yeni kaydın ait olduğu sınıf olarak kabul edilmektedir (Özkan, 2008).

Uzaklık sıfır olduğunda, hesaplamada uzaklığın tersi tanımsız olacağından, yeni kayda uzaklığı sıfır olan tüm kayıtların çoğunluk olan sınıflandırması seçilmektedir (Larose, 2005).

### 3.4 Algoritmanın Sorunları

K-en yakın komşu algoritmasının performansını etkileyen birkaç problem vardır. Bunlardan ilki k değerinin seçimidir. Eğer k değeri çok küçük olursa, elde edilecek sonuç gürültülü verilere karşı çok hassas olacaktır. Bu durumun tam tersi k değeri çok büyük olursa, bu sefer de diğer sınıflara ait olan birçok veri noktası yakın komşulara dahil olacaktır. Bu yüzden en iyi k değeri çapraz doğrulama (cross-validation) yöntemi kullanılarak elde edilebilmektedir. Yine de yeteri düzeyde eğitim örneği verildiği takdirde, k değerinin büyük değerler alması algoritmanın gürültülü verilere karşı daha dirençli olmaktadır (Wu vd., 2008). Algoritma için kullanılan tipik k değerleri 3.5 ve 7'dir (Khan vd., 2002). Dolayısıyla eğitim setinin büyüklüğüne göre bu k değerlerinden birisi de tercih edilebilmektedir.

KNN algoritması, basit ve etkili olmasına rağmen yavaş bir yöntemdir. Çünkü; bilinmeyen bir test örneğinin, eğitim setinin hangi üyesine en yakın olduğunu bulmak için en belirgin yol, test örneğinin eğitim setindeki her üyeye olan uzaklığını hesaplamak ve en küçüğünü seçmektir. Bu yüzden tek bir tahmin yapmak için gereken zaman eğitim örneklerinin sayısı ile doğru orantılı olacaktır. Başka bir deyişle bütün bir test kümesini işlemek için gereken zaman, eğitim ve test kümelerindeki örneklerin sayısı ile orantılıdır (Chakrabarti vd., 2009).

K en yakın komşu algoritmasının zengin bir veritabanına sahip olması doğru sınıflandırmalar yapabilmesi açısından çok önemlidir. Ancak eğer ana bellek alanı üzerinde kısıtlayıcılar var ise, kolay erişim için bu zengin veritabanına bakım yapmak problemli bir hale gelebilmektedir. Çünkü ana bellek dolabilir ve yardımcı belleğe erişim de yavaş olur. Bu yüzden eğer veritabanı sadece k en yakın komşu algoritması için kullanılacaksa, sadece bir sınıflandırma sınırına yakın olan veri noktalarının saklanması faydalı olacaktır (Larose, 2005).

# 4. K-MEANS ALGORİTMASI

## 4.1 Giriş

Hiyerarşik olmayan bir kümeleme yöntemi olan k-means (k-ortalamalar) algoritması, bu grup kümeleme yöntemleri arasında önemli bir yere sahiptir. Genellikle diğer birçok hiyerarşik olmayan kümeleme yöntemleri, k-means algoritmasının üzerinde yapılan değişiklikler sonucu ya da k-means algoritmasından ilham alınarak ortaya çıkmıştır (Larose, 2005).

K-means algoritmasının fikir tarihçesi 1950'li yılların sonuna doğru oluşmaya başlamıştır. "K-means" terimi ilk kez 1967 yılında James MacQueen tarafından kullanılmıştır (MacQueen,1967). Ama aslında standart k-means algoritması, ilk kez 1957 yılında darbe kodu modülasyonu için bir teknik olarak Stuart Lloyd tarafından önerilmiştir. Ancak 1982'ye kadar yayınlanmamıştır (Lloyd,1982).

K-means algoritması teknik olarak diğer yöntemlere nispeten daha hızlı, kolay adapte edilebilir, kolay anlaşılır ve bir o kadar da etkili bir yöntemdir. Büyük veri setlerinde hızlı bir şekilde çalışabilmektedir. Bu nedenle k-means algoritması, hiyerarşik olmayan kümeleme yöntemleri arasında en çok bilinen ve en geniş biçimde kullanılan algoritmalardan biridir. Denetimsiz bir öğrenme yöntemi olan k-means algoritması, her verinin sadece bir kümeye ait olmasına izin veren bir atama mekanizmasına sahiptir (Davidson, 2002). Bu nedenle algoritma sonucunda elde edilen kümeler kesin ve birbirinden ayrık bir yapıya sahiptir.

K-means algoritmasının örüntü tanıma, yapay sinir ağlarının denetimsiz öğrenmesi, sınıflandırma analizi, yapay zeka, imge işleme, makineli görme gibi birçok alanda uygulaması mevcuttur (Teknomo, 2006).

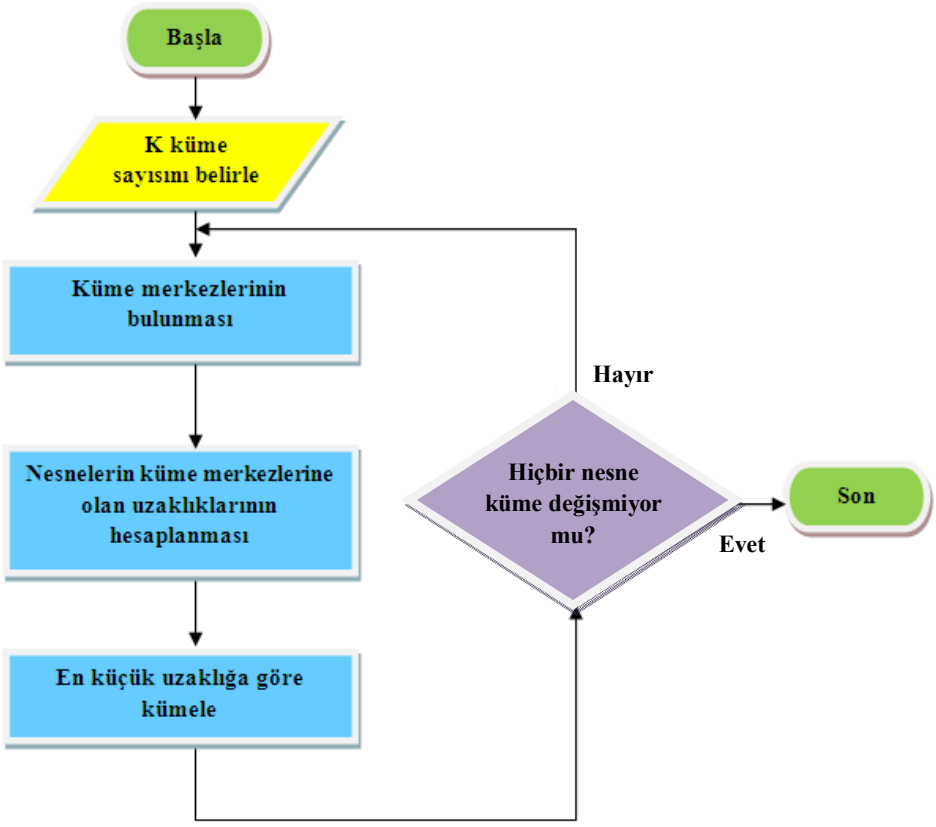
K-means, tekrarlamalı (yinelemeli) ve mesafeye dayalı olan bir algoritmadır. Algoritma, kullanıcıdan alınan k giriş parametresi değerine göre, nesneleri (kayıtları) k adet kümeye ayırmaya çalışır. Sonuçta oluşan kümelerin küme içi benzerlik değeri yüksek olurken, kümeler arası benzerlik değeri düşük olur. Buradaki küme benzerlik değeri, küme içerisindeki nesnelerin ortalama (mean) değeri göz önünde tutularak

hesaplanmaktadır. Kümenin ortalama değeri, kümenin ağırlık merkezi ya da kütle merkezi olarak da gösterilebilmektedir (Han ve Kamber, 2001).

## 4.2 Algoritmanın Çalışma Şekli

K-means algoritması, d-boyutlu bir vektör uzayındaki noktalar tarafından temsil edilen nesnelere (kayıtlara) uygulanmaktadır. $D$, d-boyutlu bir vektör uzayını ve $n_i \epsilon R^d$ ise i'nci nesne ya da veri noktasını temsil etmek üzere ve $D = \{n_i \mid i = 1, \dots, N\}$ olmak üzere; algoritma d-boyutlu vektör gruplarını kümelemektedir. Yani k-means algoritması $D$'yi k adet nesneler (noktalar) kümesine bölmektedir. Öyle ki her $n_i$ nesnesi sadece tek bir k bölümüne düşmektedir. Böylelikle hangi nesnenin hangi kümeye ait olduğunu takip edebilmek için her nesneye bir küme ID'si atanabilmektedir. Dolayısıyla, aynı küme ID'sine sahip olan nesneler aynı kümede olurken, farklı küme ID'lerine sahip olan nesneler farklı kümelerde olmaktadır (Wu vd., 2008).

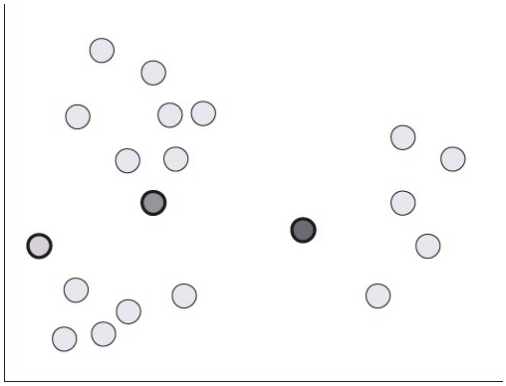
K-means algoritmasının çalışmasını özetleyen akış diyagramı Şekil 4.1.'de gösterilmiştir.




Şekil 4.1. K-means kümeleme algoritmasının akış diyagramı.

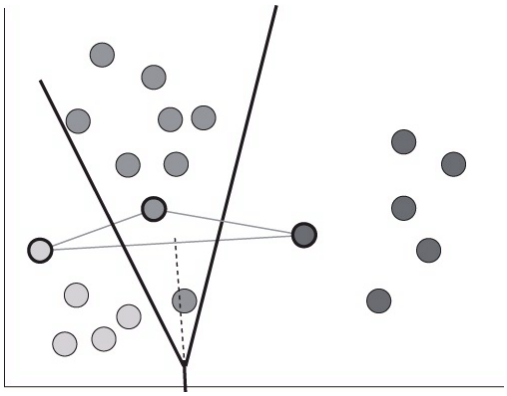
Algoritmanın genel olarak işleyişi şu şekilde gerçekleşmektedir: Öncelikle veri setinden k adet rastgele nesne seçilir. Başlangıç olarak, seçilen bu ilk nesnelerin her biri sırasıyla, söz konusu olan k adet kümenin küme merkezini (ortalamasını) ifade etmektedir. Geri kalan diğer tüm nesneler ise, nesne ile küme ortalaması arasındaki uzaklık değerlerine bağlı olarak en benzer (yakın) olduğu kümeye atanır. Algoritma daha sonra her küme için yeni ortalama değerlerini hesaplayarak küme merkezlerini yeniler (Han ve Kamber, 2001).

Anlaşılacağı üzere aslında k-means algoritması iki önemli adım arasında çalışmaktadır. Bu iki adım "atama" ve "güncelleme" adımlarıdır. Başlangıç küme merkezlerinin değerleri için k adet nesne seçildikten sonra algoritma atama ve güncelleme adımlarını sırası ile gerçekleştirir. Şekil 4.2.'de veri seti içerisinden rastgele bir biçimde üç adet küme merkezi seçilmiştir (Berry ve Linoff, 2004).



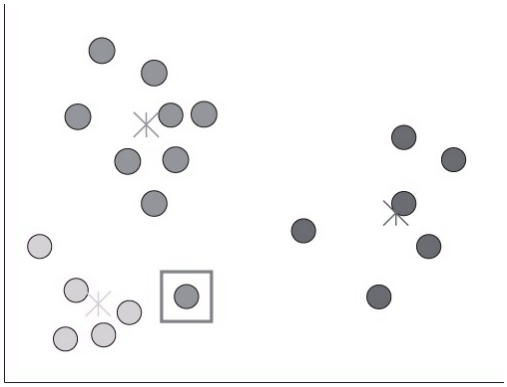
Şekil 4.2. Üç nesnenin küme merkezi olarak seçilmesi (Berry ve Linoff, 2004).

Başlangıç küme merkezleri rastgele seçildikten sonraki diğer adım ise atama işlemidir. Şekil 4.3.'de görüldüğü üzere, her nesne en yakın olduğu küme merkezine göre atanarak başlangıç kümeleri biçimlenmiştir. Yani diğer bir deyişle, her nesne en benzer olduğu küme merkezine atanmıştır. Oluşan kümelerin sınırları da "Y" şeklinde gösterilmiştir.



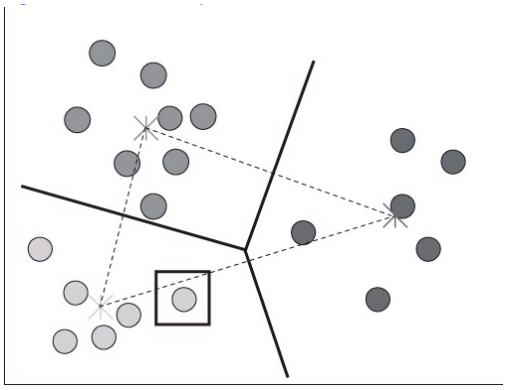
Şekil 4.3. Başlangıç kümelerin biçimlenmiş hali (Berry ve Linoff, 2004).

Sonraki adım ise güncelleme adımıdır. Şekil 4.4., bu güncelleme adımını göstermektedir. Güncelleme adımında, atama adımından sonra değişen kümelerin yeni küme merkezleri belirlenir. Küme merkezleri küme üyelerinin ortalama değerini ifade etmektedir. Şekil 4.4.'de yeni küme merkezleri "*" şeklinde gösterilmiştir. Kare içerisine alınan nesne ise yeni küme merkezlerinin değerlerine göre bu nesnenin kümesinin değişeceğini göstermektedir.



Şekil 4.4. Küme merkezlerinin güncellenmesi (Berry ve Linoff, 2004).

Şekil 4.5. ise, kare içerisine alınan nesne yeni kümesine atandıktan ve yeni küme merkezleri hesaplandıktan sonraki kümelerin son halini göstermektedir. Yeni küme merkezlerinin değerlerine göre nesnelerin ait olduğu kümelerde herhangi bir değişiklik olmamıştır. Bu nedenle algoritma sonlandırılmıştır.



Şekil 4.5. Küme merkezlerinin son yer değişikliği nesne atamalarını değiştirmediğinden algoritma sonlandırılır (Berry ve Linoff, 2004).

Atama adımı sırasında bir nesnenin küme üyeliği durumunda bir değişiklik meydana geldiğinde, küme merkez değerlerinde de bir değişiklik söz konusu olmaktadır. Bu durum, güncelleme adımının da çalıştırılmasına neden olmaktadır.

Bu süreç yeni atama işlemleri olmayıncaya kadar (sonlanma kriteri) ya da kriter fonksiyonu yakınsayana kadar devam etmektedir. Bu kriter fonksiyonu için genellikle eşitlik E.4.1'te gösterilen karesel hata ölçütü kullanılmaktadır (Han ve Kamber, 2001).

$$E = \sum_{i=1}^{k} \sum_{n \epsilon C_i} \|n - m_i\|^2 \qquad \text{(E.4.1)}$$

Bu eşitlikteki $E$ değeri veri setindeki tüm nesnelerin karesel hata değeri toplamını; ($n$ ve $m_i$ çok boyutlu değişkenler olmak üzere) $n$ verilen bir nesnenin uzayda temsil ettiği noktayı ve $m_i$ ise $C_i$ kümesinin ortalamasını ifade etmektedir. Yani bu eşitliğe göre, her kümenin her nesnesinin ait olduğu kümenin merkezi ile arasındaki Öklid uzaklıklarının karesi alınarak, tüm bu uzaklık değerleri toplanmaktadır. Bu kriter, sonuçta oluşacak kümeleri sık yoğunlukta ve birbirinden ayrık yapmaya çalışmaktadır (Han ve Kamber, 2001).

Karesel-hata değeri ile kümelemenin amacı, verilen k değeri için, küme içi değişimlerin toplamı olarak da ifade edilen $E$ değerini minimize eden k kümelerini bulmaktır. O nedenle k-means algoritmasında $E$ değerinin bir önceki iterasyona göre azalması beklenmektedir ve en düşük $E$ değerine sahip olan kümeleme sonucu en iyi çözümü vermektedir (Özkan, 2008).

Algoritmanın durması için diğer bir alternatif ise, küme merkezlerinin artık değişmemesidir (Larose, 2005). Yani diğer bir deyişle, tüm $C_1$, $C_2$, … , $C_k$ kümeleri için, küme merkezine ait olan tüm nesnelerin o kümede kalmaya devam etmesi durumunda algoritma sonlandırılır (Larose, 2005). Bu durum, iki defa üst üste aynı kümelerin bulunmasını ifade etmektedir.

K-means algoritmasına ait kaba kod (pseudo code) aşağıda verilmiştir (Dunham, 2003)(Silahtaroğlu, 2008).

***Girdiler:***

- *D={$t_1$, $t_2$, … , $t_n$}  // eldeki veritabanı*
- *K  // verilen küme sayısı*

***Algoritma:***

*(1. Adım) Keyfi olarak $m_1$, $m_2$, … , $m_k$ küme merkezlerini belirle.*
*(2. Adım) Her bir $t_i$' yi en yakın olduğu $m_i$' nin kümesine ata.*
*(3. Adım) Kümelere ait $m_1$, $m_2$, … , $m_k$ değerlerini yeniden hesapla.*
*(4. Adım) Küme elemanlarında herhangi bir değişiklik yoksa dur.*
*(5. Adım) İkinci adımdan itibaren tekrar et.*

***Çıktı:***

- *K adet küme*

Algoritmanın ikinci adımında belirtilen yakınlık kriterini ölçmek için, başka birçok mesafe ölçütü ( Manhattan, Minkowski, Mahalanobis, vb… ) olmasına rağmen, genellikle Öklid uzaklık ölçütü kullanılmaktadır. Üçüncü adımda belirtilen kümelere ait $m_1$, $m_2$, … , $m_k$ küme merkezleri ise şu şekilde hesaplanmaktadır. Bir kümeye ait ($a_1$, $b_1$, $c_1$), ($a_2$, $b_2$, $c_2$), …, ($a_n$, $b_n$, $c_n$) şeklinde n tane veri noktasının (nesne) olduğunu varsayalım. Bu kümenin ağırlık merkezinin koordinatları $\left(\sum \frac{a_i}{n}, \sum \frac{b_i}{n}, \sum \frac{c_i}{n}\right)$ şeklinde bulunmaktadır ve bu değer kümenin ortalama vektörünü ifade etmektedir (Larose, 2005).

## 4.3 Örnek

K-means algoritmasını kullanarak Çizelge 4.1.'de gösterildiği üzere 16 nesneden oluşan ufak bir veri seti üzerinde kümeleme işlemi gerçekleştirilmiştir. Bu nesneler "x" ve "y" isminde iki adet özellikten oluşmaktadır (Bramer, 2007).

Çizelge 4.1. Kümeleme için örnek veri seti (Bramer, 2007).

| x | y |
|---|---|
| 6.8 | 12.6 |
| 0.8 | 9.8 |
| 1.2 | 11.6 |
| 2.8 | 9.6 |
| 3.8 | 9.9 |
| 4.4 | 6.5 |
| 4.8 | 1.1 |
| 6.0 | 19.9 |
| 6.2 | 18.5 |
| 7.6 | 17.4 |
| 7.8 | 12.2 |
| 6.6 | 7.7 |
| 8.2 | 4.5 |
| 8.4 | 6.9 |
| 9.0 | 3.4 |
| 9.6 | 11.1 |

K küme sayısını 3 olarak belirlediğimizi varsayalım, bu durumda 3 adet başlangıç küme merkez değerlerini seçmemiz gerekecektir. Tamamen rastgele seçilen bu başlangıç küme merkezleri Çizelge 4.2'de gösterilmiştir.

Çizelge 4.2. Başlangıç küme merkezleri (Bramer, 2007).

| | **Başlangıç** | |
|---|---|---|
| | **x** | **y** |
| **1. Küme Merkezi** | 3.8 | 9.9 |
| **2. Küme Merkezi** | 7.8 | 12.2 |
| **3. Küme Merkezi** | 6.2 | 18.5 |

Çizelge 4.3.'de "U1", "U2" ve "U3" olarak isimlendirilen sütunlarda 16 nesnenin her birinin belirlenmiş olan bu üç küme merkezine olan uzaklıkları sırasıyla gösterilmektedir. Bu uzaklıklar Öklid mesafe ölçütüne göre hesaplanmıştır. İlk nesnenin 1. küme merkezine olan uzaklığı aşağıda gösterildiği gibi bulunmaktadır:

$$\sqrt{(6.8-3.8)^2+(12.6-9.9)^2}=4.0$$

Çizelge 4.3.'de "Küme" olarak isimlendirilen sütunda ise, nesnelerin en yakın olduğu küme merkezine göre kümeleri atanmaktadır.

Çizelge 4.3. İlk iterasyon sonucu (Bramer, 2007).

| x | y | U1 | U2 | U3 | Küme |
|---|---|---|---|---|---|
| 6.8 | 12.6 | 4.0 | 1.1 | 5.9 | 2 |
| 0.8 | 9.8 | 3.0 | 7.4 | 10.2 | 1 |
| 1.2 | 11.6 | 3.1 | 6.6 | 8.5 | 1 |
| 2.8 | 9.6 | 1.0 | 5.6 | 9.5 | 1 |
| 3.8 | 9.9 | 0.0 | 4.6 | 8.9 | 1 |
| 4.4 | 6.5 | 3.5 | 6.6 | 12.1 | 1 |
| 4.8 | 1.1 | 8.9 | 11.5 | 17.5 | 1 |
| 6.0 | 19.9 | 10.2 | 7.9 | 1.4 | 3 |
| 6.2 | 18.5 | 8.9 | 6.5 | 0.0 | 3 |
| 7.6 | 17.4 | 8.4 | 5.2 | 1.8 | 3 |
| 7.8 | 12.2 | 4.6 | 0.0 | 6.5 | 2 |
| 6.6 | 7.7 | 3.6 | 4.7 | 10.8 | 1 |
| 8.2 | 4.5 | 7.0 | 7.7 | 14.1 | 1 |
| 8.4 | 6.9 | 5.5 | 5.3 | 11.8 | 2 |
| 9.0 | 3.4 | 8.3 | 8.9 | 15.4 | 1 |
| 9.6 | 11.1 | 5.9 | 2.1 | 8.1 | 2 |

Her iterasyon sonucunda oluşan yeni kümelerin yeni küme merkezleri hesaplanır. Çizelge 4.4.'de ilk iki iterasyon sonucunda oluşan küme merkez değerleri gösterilmektedir. 3. küme merkezi değeri dışında diğer iki küme merkezinde de değişiklik görülmüştür. K-means algoritmasının sonlandırma kriteri olan hiçbir küme merkezi değişmeyene dek yeni küme merkezi değerleri hesaplanmaya devam edilecektir.

Çizelge 4.4. İlk iki iterasyon sonrası küme merkezleri (Bramer, 2007).

| | **Başlangıç** | | **1. İterasyon Sonu** | | **2. İterasyon Sonu** | |
|---|---|---|---|---|---|---|
| | **x** | **y** | **x** | **y** | **x** | **y** |
| **1. Küme Merkezi** | 3.8 | 9.9 | 4.6 | 7.1 | 5.0 | 7.1 |
| **2. Küme Merkezi** | 7.8 | 12.2 | 8.2 | 10.7 | 8.1 | 12.0 |
| **3. Küme Merkezi** | 6.2 | 18.5 | 6.6 | 18.6 | 6.6 | 18.6 |

## 4.4 K-means Algoritmasının Zayıf Yönleri

Diğer birçok algoritmada olduğu gibi, k-means algoritmasının da performansını etkileyen bazı sınırlayıcıları ve zayıf durumları vardır. Bu durumlar aşağıda özetlenmiştir (Teknomo, 2006):

- K küme sayısının önceden belirlenmesi gerekmektedir.
- Veri setindeki verilerin sayısı az olduğu durumlarda, algoritma gerçek kümeleri bulamaz. Böyle bir durumda, veri setindeki verilerin sıraları değiştirilip algoritma çalıştırıldığında farklı kümeleme sonuçları ortaya çıkmaktadır.
- Algoritma başlangıç durumuna karşı duyarlıdır. Bu yüzden algoritma yerel optimuma takılabilmektedir.
- Hangi özelliğin kümelemeye daha çok katkıda bulunduğu asla bilinemediğinden dolayı, her özelliğin aynı ağırlığa sahip olduğu varsayılmaktadır.
- Aritmetik ortalamanın zafiyeti yüzünden uçtaki verilere (outlier) karşı dayanıklı değildir.
- Uzaklığa dayalı bir algoritma olduğu için sonuçta elde edilen kümeler dairesel şekildedir.

Yukarıda da bahsedildiği gibi k-means algoritması için en uygun k değerini seçmek zor olabilmektedir. Eğer veri setinin doğal olarak kaç bölümden oluştuğu hakkında bir bilgi mevcutsa, bu değere göre k sayısı belirlenebilir. Ancak veri seti hakkında herhangi bir fikir yoksa, bu durumda farklı k değerleri ile algoritma denenerek

algoritmanın karesel hata değerini (E.4.1) minimize eden k değeri seçilebilir (Wu vd., 2008).

K-means algoritmasını sınırlayan durumların başında, algoritmanın dışbükey olmayan maliyetler üzerindeki açgözlü yaklaşım (greedy approach) kökenli doğası gelmektedir. Bu yaklaşım algoritmanın sadece yerel optimuma yakınsamasına neden olmaktadır. Aslında bu durum, algoritmanın ilk küme merkezlerinin yerlerine karşı daha duyarlı olduğu anlamına gelmektedir. İlk küme merkezlerine farklı şekilde değerlerin verilmesi, aynı veri seti üzerinde bile çok farklı kümelere yol açabilmektedir. Dolayısıyla kötü bir başlangıç kötü bir kümeleme sonucu ortaya çıkartabilmektedir. Algoritmanın farklı başlangıç küme merkezleri değerleri ile birçok kez çalıştırılıp, en iyi sonucu veren başlangıç değerlerinin seçilmesi ya da yakınsamış çözüm hakkında sınırlı bir yerel aramanın yapılması belli bir ölçüye kadar yerel minimum problemine karşı koyabilmektedir (Wu vd., 2008).

K-means algoritması gürültülü ve aykırı (outlier) verilerden de çok etkilenmektedir (Silahtaroğlu, 2008). Bu nedenle k-means algoritması çalıştırılmadan önce veri seti üzerinde bir önişleme işlemi gerçekleştirilerek uçtaki veriler temizlenebilir veya sonuçlar üzerinde bir sonişleme işlemi yapılarak küçük kümeler elenebilir ya da yakın olduğu büyük küme ile birleştirilebilir (Wu vd., 2008).

K-means algoritması özellikle büyük k değerleri ile ya da çok yüksek boyutlu uzayda yer alan veriler üzerinde çalıştırıldığında, algoritmanın icrasının herhangi bir yerinde, $D$'nin tüm $n_i$ noktalarının $m_j$ olmayan başka bir küme merkezine daha yakın olmasına rağmen, yine de $m_j$ küme merkezi olarak var olabilmektedir. Yani j kümesi boş bir küme olabilmektedir. Böyle bir problem söz konusu olduğunda büyük kümelerden bazı noktaları çalarak ya da boş kümenin küme merkezi yeniden başlatılarak problemle başa çıkılabilmektedir (Wu vd., 2008).

# 5. UYGULAMA

## 5.1 Giriş

Bu bölümde, tez çalışması olarak gerçekleştirilmiş olan yazılım tanıtılacaktır. Hazırlanan bu yazılımın adı, "Dermatolojik Hastalıklara Yönelik Yardımcı Karar Verme Sistemi"dir. Adından da anlaşılacağı üzere bu uygulama dermatolojideki erythemato-squamous hastalıklarının teşhisine yardımcı olmak ve bu hastalıklarla ilgili birtakım klinik veya patolojik özellikler arasında analiz yapmak amacıyla geliştirilmiş bir sistemdir.

Gerçekleştirilen uygulama bir biyomedikal sistem olup dermatoloji hekimlerinin kullanımı için tasarlanmıştır. Uygulama Visual Studio 2008 kullanılarak geliştirilmiştir. Programlama dili olarak C# ve dermatoloji veri seti için veritabanı olarak MSSQL Server tercih edilmiştir.

## 5.2 Veri Seti

Bu çalışmada kullanılan dermatoloji veri seti California Irvine Üniversitesi'nin makine öğrenmesi havuzundan elde edilmiştir (UCI Repository of Machine Learning Databases, 1998). Bu veri seti Prof. Dr. Nilsel İlter tarafından gerçek hastalardan alınan bilgilerle oluşturulmuştur.

Veri seti 366 kayıttan oluşmaktadır. Veri seti 33 tane lineer (tam sayı) ve 1 tane de nominal (kategorik) özellik olmak üzere toplam 34 özellikten oluşmaktadır. Veri setindeki her kayıt bir hastaya ait klinik ve patolojik özelliklerden oluşmaktadır. Bunlardan 22 tanesi deri örneklerinin değerlendirilmesiyle (biyopsi ile) elde edilmiş patolojik özelliklerdir. Geriye kalan 12 özellik ise eritem ve pullanmanın derecesi, lezyon sınırlarının belirgin olup olmaması, kaşıntının ve koebner olayının varlığı, papüllerin şekli, hastalığın aile geçmişinde olup olması, ağız mukozasının, dirseklerin, diz ve kafa derisinin etkilenmesi ve hastanın yaşı gibi klinik bulgulardan elde edilen klinik özelliklerdir. Klinik özellikler, hekim tarafından kolaylıkla gözlemlenebilen özelliklerdir. Aile geçmişi ve hastanın yaşı dışındaki tüm klinik özellikler var olup olmamalarına göre ve derecelerine göre 0-3 arasında bir tamsayı değeri almaktadır. Aile geçmişi özelliği için ise eğer hastanın ailesinde bu hastalıklardan biri varsa 1 ve yoksa 0

değeri kullanılmıştır. Veri setinde kayıp değerleri olan kayıtlar çıkartıldığından, uygulama 358 kayıt üzerinde çalıştırılmıştır.

Veri setinde yer alan 12 adet klinik özellik ve veritabanında kullanılan alan isimleri Çizelge 5.1.'de gösterilmiştir.

Çizelge 5.1. Veri setindeki klinik özellikler.

| Klinik Özellikler | |
|---|---|
| Özellik_1 | Kızarıklık |
| Özellik_2 | Ölçeklendirme |
| Özellik_3 | Belirgin sınırlar |
| Özellik_4 | Kaşıntı |
| Özellik_5 | Koebner olgusu |
| Özellik_6 | Poligonal kabartılar |
| Özellik_7 | Foliküler kabartılar |
| Özellik_8 | Oral mukoza tutulumu |
| Özellik_9 | Diz ve dirsek tutulumu |
| Özellik_10 | Saçlı deri tutulumu |
| Özellik_11 | Aile geçmişi |
| Özellik_34 | Yaş |

Dermatoloji veri setinde yer alan 22 adet patolojik özellik ve veritabanında kullanılan alan isimleri Çizelge 5.2.'de gösterilmiştir. Bu patolojik özelliklerin her biri derecesine göre 0-3 arasında bir sayısal değere sahip olmaktadır. Dermatolojide erythemato-squamous hastalıklarının ayırıcı teşhisi için sadece klinik özellikler yeterli olmadığından patolojik özelliklere ihtiyaç duyulmaktadır. Çünkü bu hastalık grubundaki tüm deri hastalıkları için klinik özellikler çok benzer olmaktadır.

Çizelge 5.2. Veri setindeki patolojik özellikler

| Patolojik Özellikler | |
|---|---|
| Özellik_12 | Melanin incontinence |
| Özellik_13 | Eosinophils in the infiltrate |
| Özellik_14 | PNL infiltrate |
| Özellik_15 | Fibrosis of the papillary dermis |
| Özellik_16 | Exocytosis |

| | |
|---|---|
| Özellik_17 | Acanthosis |
| Özellik_18 | Hyperkeratosis |
| Özellik_19 | Parakeratosis |
| Özellik_20 | Clubbing of the rete ridges |
| Özellik_21 | Elongation of the rete ridges |
| Özellik_22 | Thinning of the suprapapillary epidermis |
| Özellik_23 | Spongiform pustule |
| Özellik_24 | Munro microabcess |
| Özellik_25 | Focal hypergranulosis |
| Özellik_26 | Disappearance of the granular layer |
| Özellik_27 | Vacuolisation and damage of basal layer |
| Özellik_28 | Spongiosis |
| Özellik_29 | Saw-tooth appearance of retes |
| Özellik_30 | Follicular horn plug |
| Özellik_31 | Perifollicular parakeratosis |
| Özellik_32 | Inflammatory monoluclear inflitrate |
| Özellik_33 | Band-like infiltrate |

Veri setindeki her kayıt 6 hastalık sınıfından biri ile sınıflandırılmıştır. Bu sınıfların isimleri, veri setinde karşılıkları olarak kullanılan sayı kodları ve her hastalık için veri setinde yer alan hasta sayısı Çizelge 5.3.'de gösterilmiştir.

Çizelge 5.3. Dermatoloji veri setindeki sınıflar ve kodları.

| Kodu | Sınıf Adı | Hasta Sayısı |
|---|---|---|
| 1 | Psoriasis | 111 |
| 2 | Seboreic dermatitis | 60 |
| 3 | Lichen planus | 71 |
| 4 | Pityriasis rosea | 48 |
| 5 | Chronic dermatitis | 48 |
| 6 | Pityriasis rubra pilaris | 20 |

Tez çalışmasında dermatoloji veri setini kullanmadan önce veri setindeki verilere min-max normalizasyonu yöntemi uygulanarak tüm sayısal veriler 0-1 aralığına indirilmiştir. Çünkü KNN sınıflandırma ve k-means kümeleme algoritmalarının her ikisi de uzaklık ölçütü kullanmaktadır. Bu yüzden farklı sayısal değer aralığına sahip özelliklerin her birinin algoritmalardaki uzaklık hesaplarında eşit etki göstermesi için aynı sayısal değer aralığına çekilmesi gerekmektedir. Aksi takdirde dermatoloji veri seti

için yaş özelliği diğer tüm özelliklere göre daha fazla etki gösterecektir. Çünkü aile geçmişi özelliği haricinde diğer tüm özelliklerin aldığı sayısal değer aralığı 0-3 arasındadır. Ancak yaş özelliliği aldığı sayısal değer aralığı diğer özelliklere göre çok geniş bir aralıktır.

Bu veri seti, KNN sınıflandırma algoritmasının eğitim ve test işlemlerinde kullanılmak üzere eğitim ve test veri seti olmak üzere ikiye ayrılmaktadır. Eğitim veri seti her kayıtın hem özellik değerlerini hem de sınıf değerlerlerini içermektedir. Sınıflandırma yönteminin test işlemi k-kere çapraz doğrulama yöntemi kullanılarak yapılmıştır. Dolayısıyla veri setindeki her kayıt test veri setinde yer alıncaya kadar test işlemine devam edilmiştir. K-kere çapraz doğrulama yöntemi için k değeri 10 olarak belirlenmiştir. Bu yöntemin sonunda her test işlemi için bir karmaşıklık matrisi elde edilmiştir. Bu çalışmada 10-kere çapraz doğrulama yöntemi kullanıldığı için, 10 adet karmaşıklık matrisi oluşturulmuştur.

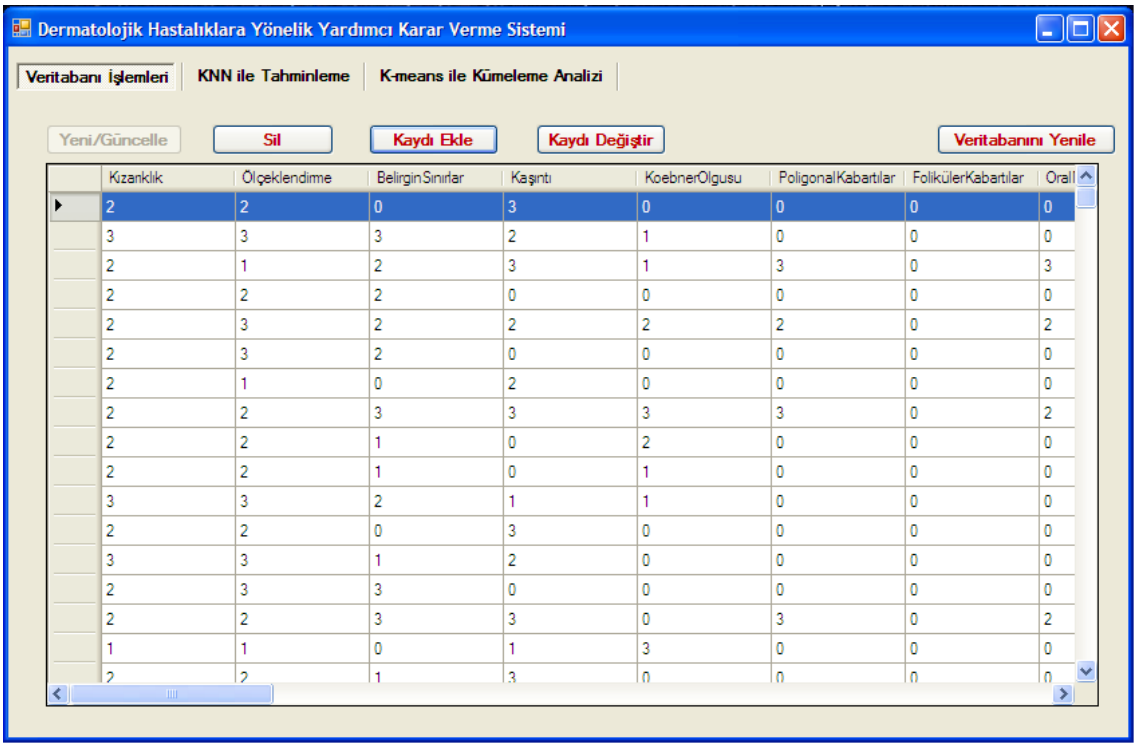
## 5.3 Kullanıcı Arayüzleri

Geliştirilen uygulama temel olarak "Veritabanı işlemleri", "KNN ile Tahminleme" ve "K-means ile Kümeleme" olmak üzere üç kullanıcı arayüzünden oluşmaktadır. Uygulamanın en üstünde ise bu arayüzler arasındaki geçişleri sağlayarak ana menü görevi üstlenen bir TabControl yer almaktadır. Uygulamanın bu arayüzleri aşağıdaki başlıklarda ayrıntılı bir şekilde açıklanmıştır.

### 5.3.1 Veritabanı işlemleri arayüzü

Uygulama açıldığında ilk ekrana gelen kullanıcı arayüzü, veritabanı işlemleri ekranıdır. Bu arayüz yazılımın üzerinde çalışacağı dermatoloji veritabanındaki tüm hasta kayıtlarının verilerini göstermektedir. Böylece veritabanında bulunan tüm hasta kayıtları hekimler tarafından görüntülenerek incelenebilmektedir. Ayrıca bu arayüz, sistemin kullanacağı veritabanına yeni hasta kaydı ekleme ve mevcut hasta kayıtlarını düzeltme veya silme gibi veritabanı işlemlerinin yapılabilmesine de izin vermektedir.

Şekil 5.1.'de veritabanı işlemleri arayüzü gösterilmiştir.




Şekil 5.1. Veritabanı işlemleri arayüzü.

Veritabanı işlemleri arayüzü ile, sistem kullanıcılarının bir yandan tıbbi analizleri elde ederken diğer yandan da veritabanının zenginleşmesine katkı sağlamaları hedeflenmiştir. Böylelikle veritabanının gelişmesi ile elde edilecek analiz sonuçlarının da kalitesini artırmış olacaktır.

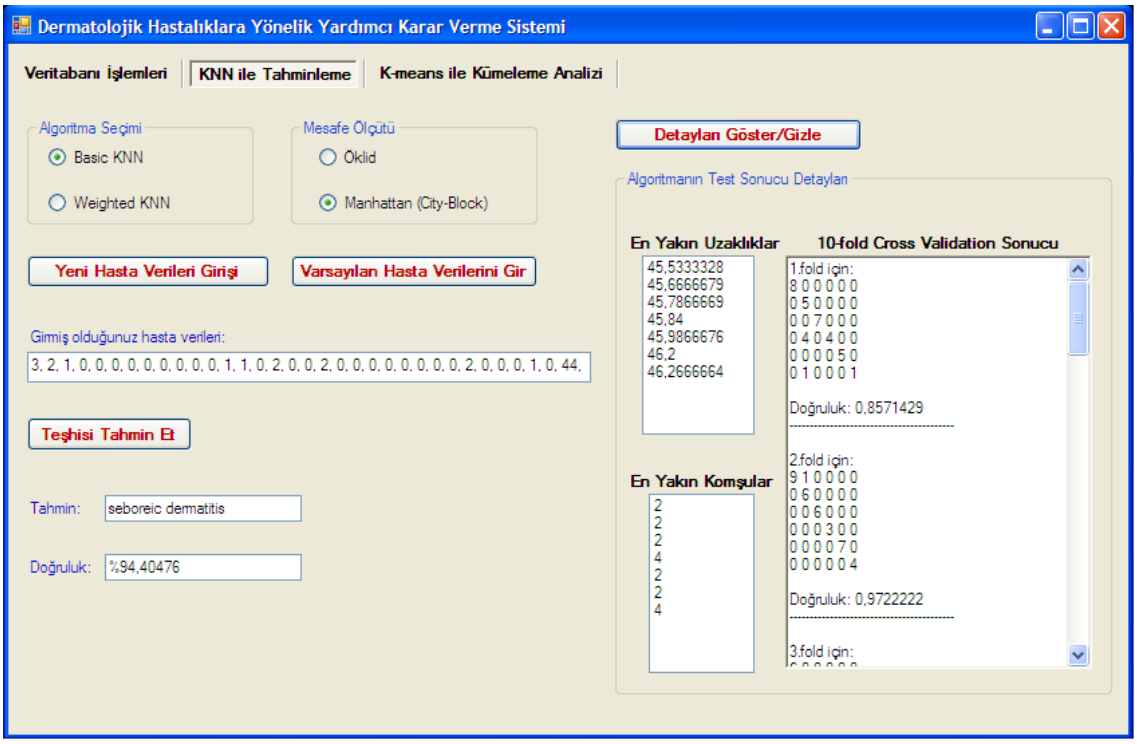
### 5.3.2 KNN ile tahminleme arayüzü

Bu arayüz ile k-en yakın komşu sınıflandırma algoritması kullanılarak, girilen yeni hasta verileri için hastalık teşhisi tahmini elde edilmektedir. Bu sayede bu arayüz ile, dermatolojide erythemato-squamous hastalıkların ayırıcı teşhisi için hekimlere yardımcı olmak amaçlanmıştır.

Uygulamada kullanılan KNN algoritmasının k değeri 7 olarak belirlenmiştir. 7 değerinin seçilmesinin nedeni KNN algoritması için tipik değerlerden biri olmasıdır (Silahtaroğlu, 2008)(Khan vd., 2002).

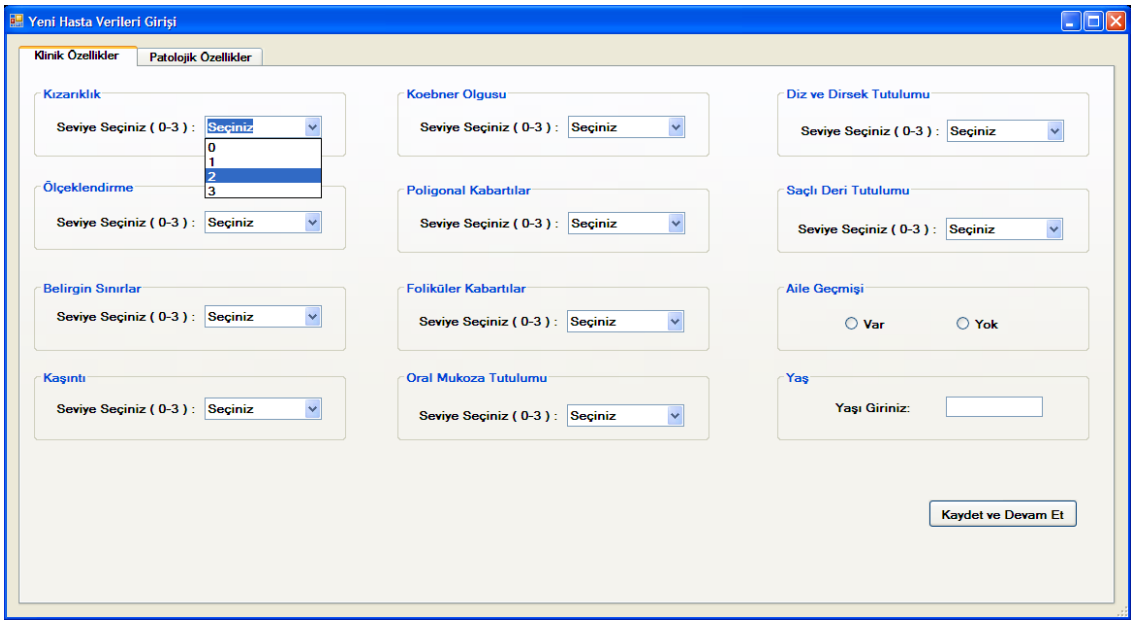
Şekil 5.2.'de de görüldüğü üzere, bu arayüzde KNN algoritması için "Basic KNN" ve "Weighted KNN" olmak üzere iki ayrı yöntem seçeneği mevcuttur. Yine aynı şekilde algoritmanın kullanacağı mesafe ölçütü için de "Öklid" ve "Manhattan" olmak

üzere iki ayrı seçenek vardır. Bu ön ayarlama seçenekleri sayesinde KNN algoritmasını farklı şekillerde çalıştırabilme imkanı sunulmaktadır.




Şekil 5.2. KNN ile tahminleme arayüzü.

KNN algoritması ile ilgili gerekli ayarlar yapıldıktan sonra test edilecek olan yeni hastanın verilerini sisteme girmek için "Yeni Hasta Verileri Girişi" butonu kullanılmaktadır. Bu butonun tıklanılmasıyla birlikte Şekil 5.3.'de gösterilen yeni bir pencere ekrana gelmektedir. Ekrana gelen bu yeni pencerede yeni hastaya ait tüm klinik ve patolojik özellikler girildikten sonra, tekrar KNN ile tahminleme arayüzüne dönülmektedir. Bu arayüzde yer alan diğer bir buton olan "Teşhisi Tahmin Et" butonu ile de girilen bu hasta verileri için KNN algoritması çalıştırılarak hastalık teşhis sonucu ve doğruluk yüzdesi elde edilmektedir.




Şekil. 5.3. Yeni hasta verileri girişi arayüzü.

KNN ile tahminleme arayüzünün diğer tarafında ise KNN algoritmasının performansını test etmek için kullanılan 10-kere çapraz doğrulama yönteminin detaylı sonucu ile en yakın 7 komşu kaydının yeni kayda olan uzaklık ve sınıf bilgilerine de ulaşılabilmektedir. Algoritmanın detaylı test sonuçları olarak ise 10-kere çapraz doğrulama yönteminin her 10 iterasyonun sonucunda elde edilen karmaşıklık matrisi ve doğruluk değerleri gösterilmektedir.

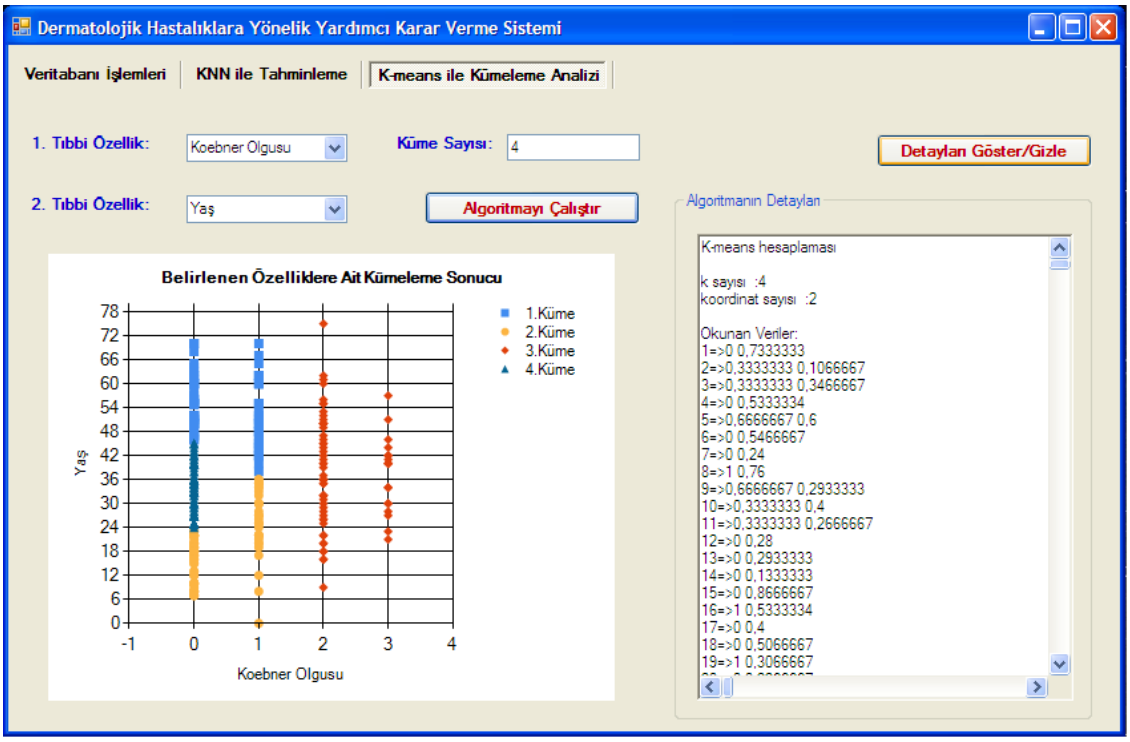
### 5.3.3 K-means ile kümeleme arayüzü

K-means ile kümeleme arayüzünde ise diğer bir veri madenciliği tekniği olan kümeleme işlemi yapılmaktadır. K-means kümeleme algoritması kullanılarak veri setinde yer alan herhangi iki tıbbi özellik arasında bir kümeleme analizi gerçekleştirilebilmektedir.

Bu arayüzün amacı, hekimlerin geçmiş hasta verilerini kullanarak herhangi iki tıbbi özellik arasında etkileşimi inceleyebilmeleri ve geleceğe yönelik doğru kararlar verebilmelerini sağlamaktır.

Kümeleme analizini gerçekleştirebilmek için küme sayısı ve aralarındaki etkileşim incelenmek istenen herhangi iki tıbbi özellik ismi, uygulamanın kullanıcı arayüzündeki ilgili yerlerde belirtildikten sonra, "Algoritmayı Çalıştır" butonuna tıklanmaktadır ve böylelikle k-means algoritması çalıştırılmaktadır. Algoritmanın

çalışması sonrasında elde edilen sonuç ise bir grafik olarak sunulmaktadır. Ancak her zaman elde edilen grafiklerin anlamlı olacağı söylenememektedir. Çünkü birbirleriyle ilişkili olmayan herhangi iki tıbbi özellik için elde edilecek kümeleme sonucu da ilişkisiz yani anlamsız olacaktır. Yazılımdaki bu k-means kümeleme modülü daha çok veriler hakkında bilgi birikimi olan hekimler tarafından uygun tıbbi özellikler seçilerek gerçekleştirilecektir. Bu işlemlerin gerçekleştirilebildiği k-means ile kümeleme arayüzü, Şekil 5.4.'de gösterilmektedir.




Şekil 5.4. K-means ile kümeleme arayüzü.

Yine bu arayüzde de k-means algoritmasının çalışma esnasında gerçekleştirdiği işlemlerin detayları kullanıcılara gösterilebilmektedir.

Şekil 5.4.'de gösterilen kümeleme analizi örneğinde, klinik özelliklerden koebner olgusu ile yaş verileri arasında bir kümeleme işlemi gerçekleştirilmiştir. Küme sayısı olarak da 4 tercih edilmiştir.

## 5.4 Uygulama Sonuçları

### 5.4.1 Performans değerlendirme yöntemleri

Uygulamada gerçekleştirilen erythemato-squamous hastalıklarının teşhis tahmininin performansını ölçmek için üç ayrı yöntem kullanılmıştır. Bu yöntemler sınıflandırma doğruluğu, k-kere çapraz doğrulama ve karmaşıklık matrisidir.

#### 5.4.1.1 Sınıflandırma doğruluğu

Bu performans ölçme yönteminde sınıflandırmanın doğruluğuna ilave olarak duyarlılık ve kesinlik ölçümleri de yapılabilmektedir. Tüm bu ölçümlerin hesaplanabilmesi için aşağıdaki bilgiler kullanılmaktadır (Sınıflandırıcı modeli için pozitif ve negatif olmak üzere iki sınıfın söz konusu olduğu varsayılmaktadır):

- ✓ TP: Doğru pozitif sayısı
- ✓ TN: Doğru negatif sayısı
- ✓ FP: Yanlış pozitif sayısı
- ✓ FN: Yanlış negatif sayısı

Buna göre bir sınıflandırıcı modelinin yapmış olduğu tahminin doğruluk değeri eşitlik E.5.1'te gösterildiği gibi hesaplanmaktadır.

$$\text{Doğruluk (\%)} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{E.5.1}$$

Bir sınıflandırıcı modelinin yapmış olduğu tahminin hata değeri ise eşitlik E.5.2'teki gibi bulunmaktadır. Buna göre hata değeri 1−Doğruluk olarak da ifade edilebilir.

$$\text{Hata (\%)} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{E.5.2}$$

Bir sınıflandırıcı modelinin yapmış olduğu tahminin duyarlılık (hassaslık) değeri eşitlik E.5.3'te gösterilen bağıntı ile hesaplanmaktadır.

$$\text{Duyarlılık (\%)} = \frac{\text{TP}}{\text{TP} + \text{TN}} \tag{E.5.3}$$

Bir sınıflandırıcı modelinin yapmış olduğu tahminin kesinlik değeri ise eşitlik E.5.4'te gösterilen bağıntı ile hesaplanmaktadır.

$$\text{Kesinlik (\%)} = \frac{\text{TP}}{\text{TP} + \text{TF}} \qquad \text{(E.5.4)}$$

#### 5.4.1.2 Karmaşıklık matrisi

Karmaşıklık matrisi, test verisi üzerinde bir sınıflandırma modeli tarafından yapılan tahmini sınıflar ile gerçek sınıflar hakkında bilgi veren bir matristir. Çizelge 5.4.'te iki sınıfın olduğu bir sınıflandırıcı modelinin karmaşıklık matrisi gösterilmektedir (Polat ve Güneş, 2007).

Çizelge 5.4. Karmaşıklık matrisi.

| **Gerçek** | **Tahmin** | |
|---|---|---|
| | *Negatif* | *Pozitif* |
| *Negatif* | a | b |
| *Pozitif* | c | d |

Bu matristeki a değeri gerçekte *Negatif* sınıfının örneği olan doğru tahminlerin sayısını, b değeri gerçekte *Negatif* sınıfının örneği olan yanlış tahminlerin sayısını, c değeri gerçekte *Pozitif* sınıfının örneği olan yanlış tahminlerin sayısını ve d değeri ise gerçekte *Pozitif* sınıfının örneği olan doğru tahminlerin sayısını temsil etmektedir.

#### 5.4.1.3 K-kere çapraz doğrulama

Çapraz doğrulama, eldeki mevcut veriyi iki parçaya ayırarak öğrenme algoritmalarını karşılaştırmak ve değerlendirmek için kullanılan bir yöntemdir. İkiye ayrılan verinin bir parçası modeli eğitmek için, diğeri ise modelin geçerliliğini denetlemek için kullanılır.

K-kere çapraz doğrulama ise çapraz doğrulama yönteminin bir çeşididir. K-kere çapraz doğrulama yönteminde öncelikle veri k adet eşit parçaya (fold) bölünür. Eğitim ve test işlemleri k iterasyon sayısı kadar gerçekleştirilir. Her iterasyonda farklı bir parça (alt küme), test verisi olarak kullanılır. Geriye kalan k-1 alt küme ise eğitim için kullanılır ve bu işlem her alt küme bir kez test için kullanılıncaya kadar devam eder. Sonuç olarak her bir alt küme için doğruluk gibi bir performans ölçütü elde edilir. Elde edilen tüm bu doğruluk değerlerinin ortalaması alınarak değerlendirme tamamlanır.

### 5.4.2 Basic KNN algoritmasının test sonucu

Dermatoloji veri seti üzerinde Basic KNN (Basit oylama) algoritmasının test sonucu için kullanılan 10-kere çapraz doğrulama yöntemi ile elde edilen karmaşıklık matrisleri aşağıdaki çizelgelerde detaylı bir şekilde gösterilmiştir. Bu performans değerleri Öklid ve Manhattan mesafe ölçütleri için elde edilen en iyi sonuçlardır.

Çizelge 5.5. Basic KNN için elde edilen en iyi sonuçların 1.alt küme testine ait karmaşıklık matrisleri.

| **Öklid mesafe ölçütü ile** | | | | | | | **Manhattan mesafe ölçütü ile** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 6 | 0 | 0 | 0 | 0 | 0 | **1** | 8 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 8 | 0 | 0 | 0 | 0 | **2** | 0 | 5 | 0 | 0 | 0 | 0 |
| **3** | 0 | 0 | 8 | 0 | 0 | 0 | **3** | 0 | 0 | 7 | 0 | 0 | 0 |
| **4** | 0 | 2 | 0 | 2 | 0 | 0 | **4** | 0 | 4 | 0 | 4 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 6 | 0 | **5** | 0 | 0 | 0 | 0 | 5 | 0 |
| **6** | 0 | 0 | 0 | 0 | 0 | 3 | **6** | 0 | 1 | 0 | 0 | 0 | 1 |

Çizelge 5.6. Basic KNN için elde edilen en iyi sonuçların 2.alt küme testine ait karmaşıklık matrisleri.

| **Öklid mesafe ölçütü ile** | | | | | | | **Manhattan mesafe ölçütü ile** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 12 | 0 | 0 | 0 | 0 | 0 | **1** | 9 | 1 | 0 | 0 | 0 | 0 |
| **2** | 0 | 5 | 0 | 1 | 0 | 0 | **2** | 0 | 6 | 0 | 0 | 0 | 0 |
| **3** | 0 | 0 | 8 | 0 | 0 | 0 | **3** | 0 | 0 | 6 | 0 | 0 | 0 |
| **4** | 0 | 0 | 0 | 4 | 0 | 0 | **4** | 0 | 0 | 0 | 3 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 5 | 0 | **5** | 0 | 0 | 0 | 0 | 7 | 0 |
| **6** | 0 | 0 | 0 | 0 | 0 | 1 | **6** | 0 | 0 | 0 | 0 | 0 | 4 |

Çizelge 5.7. Basic KNN için elde edilen en iyi sonuçların 3.alt küme testine ait karmaşıklık matrisleri.

| **Öklid mesafe ölçütü ile** | | | | | | | **Manhattan mesafe ölçütü ile** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 13 | 0 | 0 | 0 | 0 | 0 | **1** | 6 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 4 | 0 | 0 | 0 | 0 | **2** | 0 | 10 | 0 | 1 | 0 | 0 |
| **3** | 0 | 0 | 8 | 0 | 0 | 0 | **3** | 0 | 0 | 6 | 0 | 0 | 0 |
| **4** | 0 | 0 | 0 | 4 | 0 | 0 | **4** | 0 | 0 | 0 | 6 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 6 | 0 | **5** | 0 | 0 | 0 | 0 | 3 | 0 |
| **6** | 0 | 0 | 0 | 0 | 0 | 1 | **6** | 0 | 0 | 0 | 0 | 0 | 4 |

Çizelge 5.8. Basic KNN için elde edilen en iyi sonuçların 4.alt küme testine ait karmaşıklık matrisleri.

| **Öklid mesafe ölçütü ile** | | | | | | | **Manhattan mesafe ölçütü ile** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 13 | 0 | 0 | 0 | 0 | 0 | **1** | 14 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 6 | 0 | 2 | 0 | 0 | **2** | 0 | 1 | 0 | 0 | 0 | 0 |
| **3** | 0 | 0 | 5 | 0 | 0 | 0 | **3** | 0 | 0 | 7 | 0 | 0 | 0 |
| **4** | 0 | 1 | 0 | 3 | 0 | 0 | **4** | 0 | 1 | 0 | 6 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 4 | 0 | **5** | 0 | 0 | 0 | 0 | 4 | 0 |
| **6** | 0 | 0 | 0 | 0 | 0 | 2 | **6** | 0 | 1 | 0 | 0 | 0 | 2 |

Çizelge 5.9. Basic KNN için elde edilen en iyi sonuçların 5.alt küme testine ait karmaşıklık matrisleri.

| **Öklid mesafe ölçütü ile** | | | | | | | **Manhattan mesafe ölçütü ile** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 5 | 0 | 0 | 0 | 0 | 0 | **1** | 11 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 5 | 0 | 0 | 0 | 0 | **2** | 0 | 8 | 0 | 0 | 0 | 0 |
| **3** | 0 | 0 | 8 | 0 | 0 | 0 | **3** | 0 | 0 | 7 | 0 | 0 | 0 |
| **4** | 0 | 3 | 0 | 6 | 0 | 0 | **4** | 0 | 0 | 0 | 4 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 7 | 0 | **5** | 0 | 0 | 0 | 0 | 4 | 0 |
| **6** | 0 | 0 | 0 | 0 | 0 | 2 | **6** | 0 | 1 | 0 | 0 | 0 | 1 |

Çizelge 5.10. Basic KNN için elde edilen en iyi sonuçların 6.alt küme testine ait karmaşıklık matrisleri.

| **Öklid mesafe ölçütü ile** | | | | | | | **Manhattan mesafe ölçütü ile** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 13 | 1 | 0 | 0 | 0 | 0 | **1** | 18 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 8 | 0 | 0 | 0 | 0 | **2** | 0 | 4 | 0 | 0 | 0 | 0 |
| **3** | 0 | 0 | 9 | 0 | 0 | 0 | **3** | 0 | 0 | 7 | 0 | 0 | 0 |
| **4** | 0 | 0 | 0 | 2 | 0 | 0 | **4** | 0 | 0 | 0 | 2 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 2 | 0 | **5** | 0 | 0 | 0 | 0 | 3 | 0 |
| **6** | 0 | 0 | 0 | 0 | 0 | 0 | **6** | 0 | 0 | 0 | 0 | 0 | 1 |

Çizelge 5.11. Basic KNN için elde edilen en iyi sonuçların 7.alt küme testine ait karmaşıklık matrisleri.

| **Öklid mesafe ölçütü ile** | | | | | | | **Manhattan mesafe ölçütü ile** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 14 | 0 | 0 | 0 | 0 | 0 | **1** | 8 | 1 | 0 | 0 | 0 | 0 |
| **2** | 0 | 6 | 0 | 0 | 0 | 0 | **2** | 0 | 9 | 0 | 1 | 0 | 0 |
| **3** | 0 | 0 | 5 | 0 | 0 | 0 | **3** | 0 | 0 | 8 | 0 | 0 | 0 |
| **4** | 0 | 1 | 0 | 1 | 0 | 0 | **4** | 0 | 2 | 0 | 3 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 6 | 0 | **5** | 0 | 0 | 0 | 0 | 4 | 0 |
| **6** | 0 | 1 | 0 | 0 | 0 | 2 | **6** | 0 | 0 | 0 | 0 | 0 | 0 |

Çizelge 5.12. Basic KNN için elde edilen en iyi sonuçların 8.alt küme testine ait karmaşıklık matrisleri.

| **Öklid mesafe ölçütü ile** | | | | | | | **Manhattan mesafe ölçütü ile** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 8 | 0 | 0 | 0 | 0 | 0 | **1** | 9 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 5 | 0 | 2 | 0 | 0 | **2** | 0 | 7 | 0 | 1 | 0 | 0 |
| **3** | 0 | 0 | 8 | 0 | 0 | 0 | **3** | 0 | 0 | 9 | 0 | 0 | 0 |
| **4** | 0 | 1 | 0 | 6 | 0 | 0 | **4** | 0 | 1 | 0 | 2 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 5 | 0 | **5** | 0 | 0 | 0 | 0 | 5 | 0 |
| **6** | 0 | 0 | 0 | 0 | 0 | 1 | **6** | 0 | 0 | 0 | 0 | 0 | 2 |

Çizelge 5.13. Basic KNN için elde edilen en iyi sonuçların 9. alt küme testine ait karmaşıklık matrisleri.

| **Öklid mesafe ölçütü ile** | | | | | | | **Manhattan mesafe ölçütü ile** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 13 | 0 | 0 | 0 | 0 | 0 | **1** | 12 | 0 | 0 | 0 | 1 | 0 |
| **2** | 0 | 5 | 0 | 0 | 0 | 0 | **2** | 0 | 5 | 0 | 0 | 0 | 0 |
| **3** | 0 | 0 | 4 | 0 | 0 | 0 | **3** | 0 | 0 | 7 | 0 | 0 | 0 |
| **4** | 0 | 1 | 0 | 6 | 0 | 0 | **4** | 0 | 2 | 0 | 4 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 2 | 0 | **5** | 0 | 0 | 0 | 0 | 5 | 0 |
| **6** | 0 | 1 | 0 | 0 | 0 | 4 | **6** | 0 | 0 | 0 | 0 | 0 | 0 |

Çizelge 5.14. Basic KNN için elde edilen en iyi sonuçların 10. alt küme testine ait karmaşıklık matrisleri.

| **Öklid mesafe ölçütü ile** | | | | | | | **Manhattan mesafe ölçütü ile** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 13 | 0 | 0 | 0 | 0 | 0 | **1** | 13 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 3 | 0 | 0 | 0 | 0 | **2** | 0 | 2 | 0 | 0 | 0 | 0 |
| **3** | 0 | 0 | 8 | 0 | 0 | 0 | **3** | 0 | 0 | 7 | 0 | 0 | 0 |
| **4** | 0 | 3 | 0 | 2 | 0 | 0 | **4** | 0 | 1 | 0 | 3 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 5 | 0 | **5** | 0 | 0 | 0 | 0 | 8 | 0 |
| **6** | 0 | 1 | 0 | 0 | 0 | 1 | **6** | 0 | 0 | 0 | 0 | 0 | 2 |

Öklid mesafesini kullanan Basic KNN algoritmasına ait doğruluk ve hata ölçümleri Çizelge 5.15'de belirtilmiştir.

Çizelge 5.15. Öklid mesafesini kullanan Basic KNN için elde edilen en iyi sonucun performans ölçümleri.

| **Doğruluk** | **Hata** |
|---|---|
| %94,14 | %5,86 |

Manhattan mesafesini kullanan Basic KNN algoritmasına ait doğruluk ve hata ölçümleri ise Çizelge 5.16'de belirtilmiştir.

Çizelge 5.16. Manhattan mesafesini kullanan Basic KNN için elde edilen en iyi sonucun performans ölçümleri.

| **Doğruluk** | **Hata** |
|---|---|
| %94,4 | %5,6 |

### 5.4.3 Weighted KNN algoritmasının test sonucu

Dermatoloji veri seti üzerinde Weighted KNN (Ağırlıklı oylama) algoritmasının test sonucu için kullanılan 10-kere çapraz doğrulama yöntemi ile elde edilen karmaşıklık matrisleri aşağıdaki çizelgelerde detaylı bir şekilde gösterilmiştir. Bu performans değerleri Öklid ve Manhattan mesafe ölçütleri için elde edilen en iyi sonuçlardır.

Çizelge 5.17. Weighted KNN için elde edilen en iyi sonuçların 1.alt küme testine ait karmaşıklık matrisleri.

| **Öklid mesafe ölçütü ile** | | | | | | | **Manhattan mesafe ölçütü ile** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 8 | 0 | 0 | 0 | 0 | 0 | **1** | 15 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 7 | 0 | 1 | 0 | 0 | **2** | 0 | 3 | 0 | 0 | 0 | 0 |
| **3** | 0 | 0 | 7 | 0 | 0 | 0 | **3** | 0 | 0 | 7 | 0 | 0 | 0 |
| **4** | 0 | 2 | 0 | 3 | 0 | 0 | **4** | 0 | 1 | 0 | 2 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 5 | 0 | **5** | 0 | 0 | 0 | 0 | 5 | 0 |
| **6** | 0 | 0 | 0 | 0 | 0 | 2 | **6** | 0 | 0 | 0 | 0 | 0 | 2 |

Çizelge 5.18. Weighted KNN için elde edilen en iyi sonuçların 2.alt küme testine ait karmaşıklık matrisleri.

| **Öklid mesafe ölçütü ile** | | | | | | | **Manhattan mesafe ölçütü ile** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 10 | 0 | 0 | 0 | 0 | 0 | **1** | 8 | 0 | 0 | 0 | 1 | 0 |
| **2** | 0 | 9 | 0 | 0 | 0 | 0 | **2** | 0 | 7 | 0 | 2 | 0 | 0 |
| **3** | 0 | 0 | 6 | 0 | 0 | 0 | **3** | 0 | 0 | 6 | 0 | 0 | 0 |
| **4** | 0 | 0 | 0 | 6 | 0 | 0 | **4** | 0 | 2 | 0 | 3 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 5 | 0 | **5** | 0 | 0 | 0 | 0 | 4 | 0 |
| **6** | 0 | 0 | 0 | 0 | 0 | 0 | **6** | 0 | 1 | 0 | 0 | 0 | 2 |

Çizelge 5.19. Weighted KNN için elde edilen en iyi sonuçların 3.alt küme testine ait karmaşıklık matrisleri.

| **Öklid mesafe ölçütü ile** | | | | | | | **Manhattan mesafe ölçütü ile** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 14 | 0 | 0 | 0 | 0 | 0 | **1** | 11 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 7 | 0 | 0 | 0 | 0 | **2** | 0 | 9 | 0 | 0 | 0 | 0 |
| **3** | 0 | 0 | 5 | 0 | 0 | 0 | **3** | 0 | 0 | 5 | 0 | 0 | 0 |
| **4** | 0 | 0 | 0 | 0 | 0 | 0 | **4** | 0 | 0 | 0 | 1 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 7 | 0 | **5** | 0 | 0 | 0 | 0 | 10 | 0 |
| **6** | 0 | 1 | 0 | 0 | 0 | 2 | **6** | 0 | 0 | 0 | 0 | 0 | 0 |

Çizelge 5.20. Weighted KNN için elde edilen en iyi sonuçların 4.alt küme testine ait karmaşıklık matrisleri.

| **Öklid mesafe ölçütü ile** | | | | | | | **Manhattan mesafe ölçütü ile** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 12 | 0 | 0 | 0 | 0 | 0 | **1** | 14 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 4 | 0 | 0 | 0 | 0 | **2** | 0 | 5 | 0 | 1 | 0 | 0 |
| **3** | 0 | 0 | 8 | 0 | 0 | 0 | **3** | 0 | 0 | 6 | 0 | 0 | 0 |
| **4** | 0 | 2 | 0 | 8 | 0 | 0 | **4** | 0 | 0 | 0 | 5 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 2 | 0 | **5** | 0 | 0 | 0 | 0 | 5 | 0 |
| **6** | 0 | 0 | 0 | 0 | 0 | 0 | **6** | 0 | 0 | 0 | 0 | 0 | 0 |

Çizelge 5.21. Weighted KNN için elde edilen en iyi sonuçların 5.alt küme testine ait karmaşıklık matrisleri.

| **Öklid mesafe ölçütü ile** | | | | | | | **Manhattan mesafe ölçütü ile** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 14 | 0 | 0 | 0 | 0 | 0 | **1** | 13 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 5 | 0 | 1 | 0 | 0 | **2** | 0 | 10 | 0 | 0 | 0 | 0 |
| **3** | 0 | 0 | 3 | 0 | 0 | 0 | **3** | 0 | 0 | 5 | 0 | 0 | 0 |
| **4** | 0 | 0 | 0 | 5 | 0 | 0 | **4** | 0 | 0 | 0 | 2 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 5 | 0 | **5** | 0 | 0 | 0 | 0 | 3 | 0 |
| **6** | 0 | 0 | 0 | 0 | 0 | 3 | **6** | 0 | 0 | 0 | 0 | 0 | 3 |

Çizelge 5.22. Weighted KNN için elde edilen en iyi sonuçların 6.alt küme testine ait karmaşıklık matrisleri.

| **Öklid mesafe ölçütü ile** | | | | | | | **Manhattan mesafe ölçütü ile** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 11 | 0 | 0 | 0 | 0 | 0 | **1** | 8 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 8 | 0 | 0 | 0 | 0 | **2** | 0 | 6 | 0 | 0 | 0 | 0 |
| **3** | 0 | 0 | 10 | 0 | 0 | 0 | **3** | 0 | 0 | 8 | 0 | 0 | 0 |
| **4** | 0 | 0 | 0 | 2 | 0 | 0 | **4** | 0 | 2 | 0 | 3 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 4 | 0 | **5** | 0 | 0 | 0 | 0 | 4 | 0 |
| **6** | 0 | 0 | 0 | 0 | 0 | 0 | **6** | 0 | 0 | 0 | 0 | 0 | 4 |

Çizelge 5.23. Weighted KNN için elde edilen en iyi sonuçların 7.alt küme testine ait karmaşıklık matrisleri.

| **Öklid mesafe ölçütü ile** | | | | | | | **Manhattan mesafe ölçütü ile** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 7 | 0 | 0 | 0 | 0 | 0 | **1** | 8 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 6 | 0 | 0 | 0 | 0 | **2** | 0 | 8 | 0 | 0 | 0 | 0 |
| **3** | 0 | 0 | 10 | 0 | 0 | 0 | **3** | 0 | 0 | 7 | 0 | 0 | 0 |
| **4** | 0 | 1 | 0 | 4 | 0 | 0 | **4** | 0 | 0 | 0 | 6 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 6 | 0 | **5** | 0 | 0 | 0 | 0 | 6 | 0 |
| **6** | 0 | 0 | 0 | 0 | 0 | 2 | **6** | 0 | 0 | 0 | 0 | 0 | 1 |

Çizelge 5.24. Weighted KNN için elde edilen en iyi sonuçların 8.alt küme testine ait karmaşıklık matrisleri.

| **Öklid mesafe ölçütü ile** | | | | | | | **Manhattan mesafe ölçütü ile** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 11 | 0 | 0 | 0 | 0 | 0 | **1** | 14 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 6 | 0 | 0 | 0 | 0 | **2** | 0 | 2 | 0 | 0 | 0 | 0 |
| **3** | 0 | 0 | 7 | 0 | 0 | 0 | **3** | 0 | 0 | 7 | 0 | 0 | 0 |
| **4** | 0 | 1 | 0 | 4 | 0 | 0 | **4** | 0 | 2 | 0 | 5 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 4 | 0 | **5** | 0 | 0 | 0 | 0 | 4 | 0 |
| **6** | 0 | 0 | 0 | 0 | 0 | 3 | **6** | 0 | 0 | 0 | 0 | 0 | 2 |

Çizelge 5.25. Weighted KNN için elde edilen en iyi sonuçların 9.alt küme testine ait karmaşıklık matrisleri.

| **Öklid mesafe ölçütü ile** | | | | | | | **Manhattan mesafe ölçütü ile** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 10 | 1 | 0 | 0 | 0 | 0 | **1** | 9 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 3 | 0 | 0 | 0 | 0 | **2** | 0 | 2 | 0 | 0 | 0 | 0 |
| **3** | 0 | 0 | 8 | 0 | 0 | 0 | **3** | 0 | 0 | 8 | 0 | 0 | 0 |
| **4** | 0 | 2 | 0 | 3 | 0 | 0 | **4** | 0 | 1 | 0 | 7 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 5 | 0 | **5** | 0 | 0 | 0 | 0 | 5 | 0 |
| **6** | 0 | 1 | 0 | 0 | 0 | 3 | **6** | 0 | 0 | 0 | 0 | 0 | 4 |

Çizelge 5.26. Weighted KNN için elde edilen en iyi sonuçların 10.alt küme testine ait karmaşıklık matrisleri.

| **Öklid mesafe ölçütü ile** | | | | | | | **Manhattan mesafe ölçütü ile** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 13 | 0 | 0 | 0 | 0 | 0 | **1** | 10 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 3 | 0 | 0 | 0 | 0 | **2** | 0 | 5 | 0 | 0 | 0 | 0 |
| **3** | 0 | 0 | 7 | 0 | 0 | 0 | **3** | 0 | 0 | 12 | 0 | 0 | 0 |
| **4** | 0 | 3 | 0 | 2 | 0 | 0 | **4** | 0 | 0 | 0 | 6 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 5 | 0 | **5** | 0 | 0 | 0 | 0 | 2 | 0 |
| **6** | 0 | 0 | 0 | 0 | 0 | 3 | **6** | 0 | 0 | 0 | 0 | 0 | 1 |

Öklid mesafesini kullanan Weighted KNN algoritmasına ait doğruluk ve hata ölçümleri Çizelge 5.27.'de belirtilmiştir.

Çizelge 5.27. Öklid mesafesini kullanan Weighted KNN için elde edilen en iyi sonucun performans ölçümleri.

| **Doğruluk** | **Hata** |
|---|---|
| %95,53 | %4,47 |

Manhattan mesafesini kullanan Weighted KNN algoritmasına ait doğruluk, kesinlik, duyarlılık ve hata ölçümleri Çizelge 5.28.'de belirtilmiştir.

Çizelge 5.28. Manhattan mesafesini kullanan Weighted KNN için elde edilen en iyi sonucun performans ölçümleri.

| **Doğruluk** | **Hata** |
|---|---|
| %96,36 | %3,64 |

# 6. SONUÇ VE DEĞERLENDİRME

Bu tez çalışması ile veri madenciliği tekniklerinin gerçek hayat problemlerine yönelik bir uygulaması oluşturulmuştur ve gerçekleştirilen uygulamanın, tıp alanında olduğu gibi, matematiksel modellerin olmadığı ancak çok sayıda sınıflandırma örneklerinin bulunduğu problemlerin çözümünde de başarı ile uygulanabileceği görülmüştür. Ayrıca hazırlanan uygulama, veri madenciliği tekniklerinin en önemli özelliklerinden biri olan, elde edilen bilginin uzman kişilerin anlayabileceği şekilde sunulması özelliğini de yerine getirmektedir.

Geliştirilen biyomedikal yazılım ile aşağıdaki maddeler gerçekleştirilmiştir:

- Hekimlerin sistem veritabanına yeni hasta kayıtlarını ekleyebilmesi ile veritabanının zenginleştirilmesi.
- Hekimlerin sistem veritabanındaki mevcut hasta kayıtlarını inceleyebilmesi.
- KNN algoritması ile söz konusu olan yeni hastanın klinik ve patolojik özelliklerine göre hastalık teşhisi tahmininin yapılması.
- Hastalık teşhis tahminin doğruluğunun ve etkinliğinin tespit edilmesi.
- Sistemdeki hasta verilerinin k-means algoritması ile kümelenmesi ile, hekimlerin hasta özellikleri ve ilişkileri hakkında ileriye yönelik yorum yapabilmesine izin veren analizin elde edilmesi.

Bu çalışmada; veri madenciliği algoritmalarından KNN ve k-means algoritmalarının medikal alanda nasıl kullanılabileceği konusunda örnek bir çalışma yapılmış ve sonuçları gözlenmiştir. Dermatoloji veri seti üzerinde KNN algoritması kullanılarak elde edilen hastalık teşhisleri ile ilgili sayısal sonuçlar ve yorumlar Bölüm 5.4'de verilmiştir.

Sonuç olarak, özellikle tıp alanında bu tip uygulamaların, hastalara doğru teşhis koymak için maliyeti yüksek, laboratuar şartları kısıtlı olan, hasta açısından risk taşıyan yöntemlerin söz konusu olduğu durumlarda ya da tecrübe bakımından yeterli düzeyde olmayan hekimlerin doğru teşhis koymalarına fayda sağlamada önemli bir yardımcı karar destek aracı olarak kullanılabileceği görülmektedir.

# KAYNAKLAR


Akpınar, H., "Veritabanlarında Bilgi Keşfi ve Veri Madenciliği", ***İ.Ü. İşletme Fakültesi Dergisi***, 29: 1-22 (2000).

Albayrak, M., "EEG Sinyallerindeki Epileptiform Aktivitenin Veri Madenciliği Süreci İle Tespiti", Doktora Tezi, ***Sakarya Üniversitesi Fen Bilimleri Enstitüsü***, Sakarya, 2008.

Alluri, N. R., "Evaluation Of Data Mining Methods To Support Data Warehouse Administration and Monitoring In SAP Business Warehouse", Yüksek Lisans Tezi, ***University of Applied Sciences-Furtwangen***, Germany, 2005.

Aslandogan, A., Mahajani, G., Taylor, S., "Evidence Combination in Medical Data Mining", ***International Conference on Information Technology: Coding and Computing (ITCC'04)***, Volume 2: 465-470 (2004).

Berry, M. J. A., Linoff, G., "Data Mining Techniques : For Marketing, Sales, and Customer Relationship Management", ***Wiley Publishing***, 2004.

Beyer, K. vd., "When Is 'Nearest Neighbor' Meaningful?", **7. Uluslararası Database Theory Konferansı (ICDT'99)**, İsrail, 217-235 (1999).

Bozkır, A. S., "Olap Ve Veri Madenciliği Teknolojilerinden Yararlanılarak Web Tabanlı Bir Karar Destek Sisteminin Gerçekleştirilmesi", Yüksek Lisans Tezi, ***Hacettepe Üniversitesi Fen Bilimleri Enstitüsü***, Ankara, 2009.

Bramer, M., "Principles of Data Mining", ***Springer***, London, 2007.

Chakrabarti, S., Cox, E., Frank, E., Güting, R. H., Han, J., Jiang, X., Kamber, M., Lightstone, S. S., Nadeau, T. P., Neapolitan, R. E., Pyle, D., Refaat, M., Schneider, M., Teorey, T. J., Witten, I. H., "Data Mining Know It All", ***Elsevier***, United States, 2009.

Davidson, I, Y.: "Understanding K-means Non-Hierarchical Clustering", ***Computer Science Department of State University of New York (SUNY) Technical Report, Albany***, 2002.

Demirdümen, B., "Data and Text Mining Techniques On Real Medical Data", Yüksek Lisans Tezi, ***Çankaya Üniversitesi Fen Bilimleri Enstitüsü***, Ankara, 2008.

Dincer, E., Duru, N., "Prototype of a tool for analysing laryngeal cancer operations", ***Informatics For Health & Social Care***, 34(3): 159-170 (2009).

Dunham, M. H., "Data Mining Introductory and Advanced Topics", ***Prentice Hall, Pearson Education Inc.***, New Jersey, 2003.

Eker, H., "Veri Madenciliği veya Bilgi Keşfi", http://www.bilgiyönetimi.org/cm/pages/mkl_gos.php?nt=538, 2004.

# KAYNAKLAR (Devam Ediyor)


Haberal, İ., "Veri Madenciliği Algoritmaları Kullanılarak Web Günlük Erişimlerinin Analizi", Yüksek Lisans Tezi, ***Başkent Üniversitesi Fen Bilimleri Enstitüsü***, Ankara, 2007.

Han, J. and Kamber, M., "Data Mining Concepts and Techniques", ***Academic Press***, New York, 2001.

Kaya, E., Bulun, M., Arslan, A., "Tıpta Veri Ambarları Oluşturma ve Veri Madenciliği Uygulamaları", ***Akademik Bilişim 2003***, Adana, (2003).

Khan, M. vd., "K-Nearest Neighbor Classification on Spatial Data Streams Using P-Trees", ***6. Pasifik Asya Knowledge Discovery and Data Mining Konferansı PAKKDD'02***, Taiwan, 517-518 (2002).

Kumar, V., Joshi, M., "Tutorial on High Performance Data Mining", http://www-users.cs.umn.edu/, 2005.

Larose, D. T., "Discovering knowledge in data : an introduction to data mining", ***John Wiley & Sons, Inc.***, Hoboken, New Jersey, 2005.

Lin, R., "An Intelligent Model For Liver Disease Diagnosis", ***Artificial Intelligence in Medicine***, 53-62 (2009).

Lloyd., S. P. "Least squares quantization in PCM", ***IEEE Transactions on Information Theory***, 28 (2): 129-137 (1982).

MacQueen, J. B., "Some Methods for classification and Analysis of Multivariate Observations", ***1. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability***, 281-297 (1967).

Nisbet, R., Elder, J., Miner, G., "Handbook of Statistical Analysis and Data Mining Applications", ***Elsevier***, 2009.

Özkan, Y., "Veri Madenciliği Yöntemleri", ***Papatya Yayıncılık Eğitim***, İstanbul, 2008.

Polat, K., Güneş, S., "A hybrid approach to medical decision support systems:Combining feature selection, fuzzy weighted pre-processing and AIRS", ***Computer Methods And Programs In Biomedicine,*** 88:164-174 (2007).

Sever, H., Oğuz, B. "Veri Tabanlarında Bilgi Keşfine Formel Bir Yaklaşım:Kısım I: Eşleştirme Sorguları ve Algoritmalar", ***ÜNAK, Bilgi Dünyası***, 3(2):173-204 (2002).

Silahtaroğlu, G., "Kavram Ve Algoritmalarıyla Temel Veri Madenciliği", ***Papatya Yayıncılık Eğitim***, İstanbul, 2008.

Şen, F., "Veri Madenciliği İle Birliktelik Kurallarının Bulunması", Yüksek Lisans Tezi, ***Sakarya Üniversitesi Fen Bilimleri Enstitüsü***, Sakarya, 2008.

# KAYNAKLAR (Devam Ediyor)


Şişaneci, İ., "Sağlık Sisteminde Veri Madenciliği İle Suistimal Tespiti", Yüksek Lisans Tezi, ***Gebze Yüksek Teknoloji Enstisüsü Mühendislik ve Fen Bilimleri Enstitüsü***, Gebze, 2009.

Tan, P.N., Steinbach, M., Kumar, V., "Introduction to Data Minining", ***Addison-Wesley***, 2006.

Teknomo K., "K-Means Clustering Tutorial", http://people.revoledu.com/kardi/tutorial/kMean, 2006.

Telcioğlu, M. B., "Veri Madenciliğinde Genetik Programlama Temelli Yeni Bir Sınıflandırma Yaklaşımı Ve Uygulaması", Yüksek Lisans Tezi, ***Erciyes Üniversitesi Fen Bilimleri Enstitüsü*,** Kayseri, 2007.

Tosun, T., "Veri Madenciliği Teknikleriyle Kredi Kartlarında Müşteri Kaybetme Analizi", Yüksek Lisans Tezi, ***İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü***, İstanbul, 2006.

Two Crows Corporation, "Two Crows:Introduction to Data Mining and Knowledge Discovery", Third Edition, 1999.

Ubeyli E. D., I. Guler, "Automatic detection of erythemato squmous diseases using adaptive neuro-fuzzy inference systems", ***Computer in Bialogy and Medicine***, 35:421-433 (2005).

Ubeyli, E. D., "Combined neural networks for diagnosis of erythemato-squamous diseases", ***Expert Systems with Applications***, 36(3):5107-5112 (2009).

UCI Repository of Machine Learning Databases, http://archive.ics.uci.edu/ml/datasets/Dermatology, 1998.

Wang, L., Fu, X., "Data Mining With Computational Intelligence", ***Springer***, Germany, 2005.

Wu, X., Kumar V., Quinlan J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P. S., Zhou, Z. H., Steinbach, M., Hand, D. J., Steinberg, D., "Top Ten Algorithms In Data Mining", ***Springer***, 2008.

Xie, J., Wang, C., "Using support vector machines with a novel hybrid feature selection method for diagnosis of erythemato-squamous diseases", ***Expert Systems with Applications***, 38(5):5809-5815 (2011).

Yıldırım, P., Uludağ, M., Görür, A., "Hastane Bilgi Sistemlerinde Veri Madenciliği", ***Akademik Bilişim 2008***, 429-434, (2008).

# EKLER

## Ek-1. Uygulamaya Ait Program Kodlarının Bir Kısmı

```
…
List<int> EnYakinSiniflar = new List<int>();
List<float> EnYakinUzakliklar = new List<float>();
List<float> EnYakinAgirliklar = new List<float>();
List<float[]> VeriKumesi = new List<float[]>();
List<float[]> EgitimKumesi = new List<float[]>();
List<float[]> TestKumesi = new List<float[]>();
static List<float> _yeniKayit = new List<float>();
veritabani vt;
string sql;
SqlDataAdapter myAdp;
DataSet myDS;
int[,] KarmasiklikMatrisi;
float[] DogrulukDizisi = new float[10];
bool bayrak1=false;
bool bayrak2 = false;

private void Form1_Load(object sender, EventArgs e)
{ … }

private void KNN()
{ … }

private void DataGridViewDoldur()
{ … }

private float[,] Sirala(float[,] dizi)
{ … }

private void KYakinKomsuGetir(int k, float[,] dizi)
{ … }

private void KYakinKomsuGetir_(int k, float[,] dizi)
{
    if (bayrak2==true || bayrak1==true)
    {
        for (int i = EnYakinSiniflar.Count() - 1; i >= 0; i--)
        {
            EnYakinSiniflar.RemoveAt(i);
            EnYakinUzakliklar.RemoveAt(i);
            if(radioButton2.Checked==true&&
EnYakinAgirliklar.Count()!=0)
                EnYakinAgirliklar.RemoveAt(i);
        }
        for (int i = 0; i < k; i++)
        {
            EnYakinSiniflar.Add(Convert.ToInt32(dizi[i, 1]));
            EnYakinUzakliklar.Add(dizi[i, 0]);
        }
```

```
    }
    else
    {
        for (int i = 0; i < k; i++)
        {
            EnYakinSiniflar.Add(Convert.ToInt32(dizi[i, 1]));
            EnYakinUzakliklar.Add(dizi[i, 0]);
        }
        bayrak2 = true;
    }
}

private void diziyiKaristir(int[] dizi, int boyut)
{ … }

private int[] kfold(int boyut, int k)
{
   int[] indisler = new int[boyut];
   float inc = (float)k / boyut;
   for (int i = 0; i < boyut; i++)
      indisler[i] = (int)(Math.Ceiling((i + 0.9) * inc) - 1);
   diziyiKaristir(indisler, boyut);
   return indisler;
}

private void KNN_(List<float[]> egitim, List<float[]> test)
{
    string yol = @"C:\Dosya2.txt";
    FileStream fs = new FileStream(yol, FileMode.OpenOrCreate,
FileAccess.Write);
    StreamWriter sw = new StreamWriter(fs);

    float[] TestKaydi= new float[dataGridView1.Columns.Count-2];
    int degisken = Convert.ToInt32(test.Count.ToString());
    float[] uzakliklar = new float[358];
    float[] dizim;
    float[] geciciDizi = new float[34];
    float[,] sinifliUzakliklar = new float[358, 2];
    float[,] siraliSinifliUzakliklar = new float[358, 2];
    float dogruluk = 0.0f;
    float pay = 0.0f;
    float payda = 0.0f;
    int y = 0;
    int w = 0;
    float test_indis;

    for (int k = 0; k < 10; k++)
    {
        y = 0;
        w = 0;
        pay = 0.0f;
        payda = 0.0f;
```

```
    for (int f = EgitimKumesi.Count-1; f >=0; f--)
    {
        EgitimKumesi.RemoveAt(f);
    }
    for (int f = TestKumesi.Count-1; f >=0 ; f--)
    {
        TestKumesi.RemoveAt(f);
    }
    test_indis = k;
    for (int i = 0; i < 358; i++)
    {
        if (!VeriKumesi[i][35].Equals(test_indis))
        {
             EgitimKumesi.Add(new float[36]);
             for (int j = 0; j < 36; j++)
             {
                 EgitimKumesi[w][j] = VeriKumesi[i][j];
             }
             w++;
        }
        else
       {
             TestKumesi.Add(new float[36]);
             for (int m = 0; m < 36; m++)
             {
                  TestKumesi[y][m] = VeriKumesi[i][m];
             }
             y++;
        }
     }
     egitim = EgitimKumesi;
     test = TestKumesi;
     for (int i = 0; i < 6; i++)
     {
         for (int j = 0; j < 6; j++)
         {
              KarmasiklikMatrisi[i, j] = 0;
         }
     }
     for (int q = 0; q < test.Count(); q++)
     {
         TestKaydi = (float[])test[q].ToArray();
         for (int i = 0; i < egitim.Count; i++)
         {
             dizim = (float[])egitim[i].ToArray();
             for (int j = 0; j < 34; j++)
             {
                 geciciDizi[j] = dizim[j];
             }
             if (radioButton3.Checked == true)  //Öklid
             {
                  uzakliklar[i]=OklidUzakligi(TestKaydi,
geciciDizi);
```

```
                }
                else if (radioButton4.Checked==true)//Manhattan
                {
                    uzakliklar[i]=ManhattanUzakligi(TestKaydi,
geciciDizi);
                }
            }
            for (int i = 0; i < egitim.Count(); i++)
            {
                sinifliUzakliklar[i, 0] = uzakliklar[i];
                sinifliUzakliklar[i,1]=egitim[i].ElementAt(34);
            }
            siraliSinifliUzakliklar=Sirala(sinifliUzakliklar);
            KYakinKomsuGetir_(7, siraliSinifliUzakliklar);
            TestEt(TestKaydi);
        }
        sw.WriteLine((k+1).ToString() + ".fold için:");
        for (int i = 0; i < 6; i++)
        {
            for (int j = 0; j < 6; j++)
            {
               payda += KarmasiklikMatrisi[i, j];

              sw.Write(KarmasiklikMatrisi[i,j].ToString()+" ");
        }
        sw.WriteLine("");
    }
    for (int i = 0; i < 6; i++)
    {
        pay += KarmasiklikMatrisi[i, i];
    }
    dogruluk = pay / payda;
    DogrulukDizisi[k] = dogruluk;
    sw.WriteLine("");
    sw.WriteLine("Doğruluk: " + DogrulukDizisi[k].ToString());
    sw.WriteLine("----------------------------------------");
    sw.WriteLine("");
    }
    float ortDogruluk;
    float dogrulukToplami = 0.0f;
    for (int i = 0; i < 10; i++)
    {
        dogrulukToplami += DogrulukDizisi[i];
    }
    ortDogruluk = dogrulukToplami / 10.0f;
    sw.WriteLine("DOĞRULUK: " + ortDogruluk.ToString());
    txtDogruluk.Text = "%"+(ortDogruluk*100).ToString();
    sw.Flush();
    sw.Close();
}

private void TestEt(float[] testEdilecekKayit)
{
```

```
    int gercek_sinif=0, tahmin_sinifi=0;
    if(testEdilecekKayit[34]==1)
    {
        gercek_sinif = 1;
    }
    else if (testEdilecekKayit[34] == 2)
    {
        gercek_sinif = 2;
    }
    else if (testEdilecekKayit[34] == 3)
    {
        gercek_sinif = 3;
    }
    else if (testEdilecekKayit[34] == 4)
    {
        gercek_sinif = 4;
    }
    else if (testEdilecekKayit[34] == 5)
    {
        gercek_sinif = 5;
    }
    else if (testEdilecekKayit[34] == 6)
    {
        gercek_sinif = 6;
    }

    if (radioButton1.Checked == true)  //Basic KNN
    {
        tahmin_sinifi = SonucuGoster();
    }
    else if (radioButton2.Checked == true) //Weighted KNN
    {
        AgirliklariHesapla();
        tahmin_sinifi = EnBuyukAgirlikliSinifiBul();
    }
    int x,y;
    x=gercek_sinif-1;
    y=tahmin_sinifi-1;
    KarmasiklikMatrisi[x, y] += 1;
}

private void AgirliklariHesapla()
{
    string yol = @"C:\Dosya3.txt";
    FileStream fs = new FileStream(yol, FileMode.Append,
FileAccess.Write);
    StreamWriter sw = new StreamWriter(fs);
    float agirlik;
    sw.WriteLine("");
    sw.WriteLine("Uzaklıklar \t sınıflar \t Ağırlıklar");
    if (EnYakinAgirliklar.Count != 0)
    {
        for (int i = EnYakinAgirliklar.Count() - 1; i >= 0; i--)
```

```
        {
            EnYakinAgirliklar.RemoveAt(i);
        }
    }
    foreach (float f in EnYakinUzakliklar)
    {
        agirlik = (float)(1.0 / Math.Pow(f, 2.0));
        EnYakinAgirliklar.Add(agirlik);
         sw.WriteLine(f.ToString()+"\t" +
      EnYakinSiniflar[EnYakinAgirliklar.Count()-1].ToString()
      +"\t" + agirlik.ToString());
    }
       sw.Flush();
       sw.Close();
}

private int EnBuyukAgirlikliSinifiBul()
{
      float toplam1 = 0.0f;
      float toplam2 = 0.0f;
      float toplam3 = 0.0f;
      float toplam4 = 0.0f;
      float toplam5 = 0.0f;
      float toplam6 = 0.0f;
      for (int i = 0; i < EnYakinSiniflar.Count(); i++)
      {
          if (EnYakinSiniflar[i] == 1)
          {
             toplam1 += EnYakinAgirliklar[i];
          }
          else if(EnYakinSiniflar[i] == 2)
          {
             toplam2 += EnYakinAgirliklar[i];
          }
          else if (EnYakinSiniflar[i] == 3)
          {
             toplam3 += EnYakinAgirliklar[i];
          }
          else if (EnYakinSiniflar[i] == 4)
          {
             toplam4 += EnYakinAgirliklar[i];
          }
          else if (EnYakinSiniflar[i] == 5)
          {
             toplam5 += EnYakinAgirliklar[i];
          }
          else if (EnYakinSiniflar[i] == 6)
          {
             toplam6 += EnYakinAgirliklar[i];
          }
      }
      int sonuc=0;
      float enBuyuk;
```

```
        enBuyuk = Math.Max(toplam1, toplam2);
        enBuyuk = Math.Max(enBuyuk, toplam3);
        enBuyuk = Math.Max(enBuyuk, toplam4);
        enBuyuk = Math.Max(enBuyuk, toplam5);
        enBuyuk = Math.Max(enBuyuk, toplam6);
        if (toplam1 == enBuyuk)
        {
            sonuc = 1;
        }
        if (toplam2 == enBuyuk)
        {
            sonuc = 2;
        }
        if (toplam3 == enBuyuk)
        {
            sonuc = 3;
        }
        if (toplam4 == enBuyuk)
        {
            sonuc = 4;
        }
        if (toplam5 == enBuyuk)
        {
            sonuc = 5;
        }
        if (toplam6 == enBuyuk)
        {
            sonuc = 6;
        }
        return sonuc;
}

private int SonucuGoster()
{ … }

private string TeshisiYaz(int sinif_no)
{ … }

private float OklidUzakligi(float[] X, float[] Y)
{ … }

private float ManhattanUzakligi(float[] X, float[] Y)
{ … }

private void DereceliOzelliklerIcinOnisleme()
{ … }

private void DereceliOzelliklerIcinOnisleme(float[] yeniKyt)
{ … }

private void VarsayilanHastaKaydi()
{ … }
```

```
public void kmeans()
{
    int i, j, n, l, m;
    int k;
    int boyut;
    int sayac;
    int iterasyon;
    float t;
    float ek;
    int Adr;
    bool son;
    List<int[]> Kume1 = new List<int[]>();
    List<int[]> Kume2 = new List<int[]>();
    int[] dizi = new int[2];
    int[] xDegerleri = new int[VeriKumesi.Count];
    int[] yDegerleri = new int[VeriKumesi.Count];
    int[] renkler = new int[VeriKumesi.Count];
    int[,] veriler1 = new int[2, VeriKumesi.Count];
    double[,] veriler2 = new double[2, VeriKumesi.Count];
    DataGridViewRow satir;
    int parametre1;
    int parametre2;
    int x;
    boyut = 2;
    son = false;
    iterasyon = 0;
    k = Convert.ToInt32(txtKume.Text);
    int[,] grupla1 = new int[k, VeriKumesi.Count];
    int[,] grupla2 = new int[k, VeriKumesi.Count];
    float[,] merkezler = new float[2, k];
    float[,] uzk = new float[k, VeriKumesi.Count];
    string yol = @"C:\Dosya.txt";
    FileStream fs=new
FileStream(yol,FileMode.OpenOrCreate,FileAccess.Write);
    StreamWriter sw = new StreamWriter(fs);
 parametre1=Convert.ToInt32(comboBox1.SelectedValue.ToString());
 parametre2=Convert.ToInt32(comboBox2.SelectedValue.ToString());
    sw.WriteLine("K-means hesaplaması");
    if (VeriKumesi.Count <= k)
    {
     MessageBox.Show("k degeri, kayıt sayısından büyük olamaz");
    }
    else
    {
      for (i = 0; i < dataGridView1.Rows.Count; i++)
      {
           x = i + 1;
           satir = dataGridView1.Rows[i];
           veriler1[0, i] =
     Convert.ToInt32(satir.Cells[parametre1].Value.ToString());
            veriler1[1, i] =
     Convert.ToInt32(satir.Cells[parametre2].Value.ToString());
```

```
            veriler2[0, i] =
        Convert.ToDouble(VeriKumesi[i][parametre1-1].ToString());
            veriler2[1, i] =
        Convert.ToDouble(VeriKumesi[i][parametre2-1].ToString());
         }
         for (int q = 0; q < VeriKumesi.Count; q++)
         {
            xDegerleri[q] = veriler1[0, q];
            yDegerleri[q] = veriler1[1, q];
         }
         sw.WriteLine("");
         sw.WriteLine("k sayısı  :" + k.ToString());
         sw.WriteLine("koordinat sayısı  :" + boyut.ToString());
         sw.WriteLine("");
         sw.WriteLine("Okunan Veriler:");
        for (j = 0; j < dataGridView1.Rows.Count; j++)
        {
            sw.Write((j + 1).ToString() + "=>");
            for (m = 0; m < boyut; m++)
            {
                sw.Write(veriler2[m, j].ToString() + " ");
            }
            sw.WriteLine("");
        }
        for (j = 0; j < k; j++)
        {
            for (m = 0; m < dataGridView1.Rows.Count; m++)
            {
                grupla1[j, m] = 0;
                grupla2[j, m] = 0;
            }
        }
        sw.WriteLine("İlk merkezler :");
        for (i = 0; i < boyut; i++)
        {
            for (j = 0; j < k; j++)
            {
                merkezler[i, j] = (float)veriler2[i, j];
                swWrite(merkezler[i, j].ToString() + " ");
            }
            sw.WriteLine("");
        }
        while (!son)
        {
           for (l = 0; l < k; l++)
           {
             for (n = 0; n < dataGridView1.Rows.Count; n++)
             {
                t = 0;
                for (m = 0; m < boyut; m++)
                {
                  t = t + (float)((merkezler[m, l] - veriler2[m,
n]) * (merkezler[m, l] - veriler2[m, n]));
```

```
      }
      uzk[l, n] = (float)Math.Sqrt(t);
   }
}
sw.WriteLine("Uzaklık matrisi   :");
for (i = 0; i < k; i++)
{
   for (j = 0; j < dataGridView1.Rows.Count; j++)
   {
      sw.Write(uzk[i, j].ToString() + " / ");
   }
sw.WriteLine("");
sw.WriteLine("");
}
for (i = 0; i < dataGridView1.Rows.Count; i++)
{
    ek = uzk[0, i];
    Adr = 0;
    for (j = 1; j < k; j++)
    {
       if (ek > uzk[j, i])
       {
          ek = uzk[j, i];
          Adr = j;
       }
    }
    grupla1[Adr, i] = 1;
}
iterasyon = iterasyon + 1;
sw.WriteLine("Kümeleme Matrisi:"+iterasyon.ToString());
for (i = 0; i < k; i++)
{
   for (j = 0; j < dataGridView1.Rows.Count; j++)
   {
       sw.Write(grupla1[i, j].ToString() + " ");
   }
   sw.WriteLine("");
}
sw.WriteLine("---------------------------------------");
n = 0;
for (j = 0; j < k; j++)
{
    for (m = 0; m < dataGridView1.Rows.Count; m++)
    {
        if (grupla1[j, m] != grupla2[j, m])
        {
            n = 1;
            break;
        }
    }
}
if (n == 1)
{
```

```
        for (j = 0; j < k; j++)
        {
           for (m = 0; m < dataGridView1.Rows.Count; m++)
           {
                grupla2[j, m] = grupla1[j, m];
           }
        }
        for (j = 0; j < k; j++)
        {
           for (m = 0; m < boyut; m++)
          {
              merkezler[m, j] = 0;
          }
       }
       for (j = 0; j < k; j++)
       {
           sayac = 0;
           for (i = 0; i < dataGridView1.Rows.Count; i++)
           {
               if (grupla1[j, i] == 1)
               {
                   sayac++;
                   for (n = 0; n < boyut; n++)
                   {
                       merkezler[n, j] = merkezler[n, j] +
                  (float)veriler2[n, i];
                   }
               }
           }
           for (n = 0; n < boyut; n++)
           {
               merkezler[n, j] = merkezler[n, j] / sayac;
           }
       }
       sw.WriteLine("Yeni merkezler    :");
       for (i = 0; i < k; i++)
       {
           for (j = 0; j < boyut; j++)
           {
               sw.Write(merkezler[j, i].ToString() + " ");
           }
           sw.WriteLine("");
       }
       sw.WriteLine("");
       for (j = 0; j < k; j++)
       {
           for (m = 0; m < dataGridView1.Rows.Count; m++)
           {
              grupla1[j, m] = 0;
           }
       }
}
else
```

```
        {
           son = true;
        }
 }
 sw.Flush();
 sw.Close();
 xDegerleri = xDegerleri.Distinct().ToArray();
 yDegerleri = yDegerleri.Distinct().ToArray();
 int xAralik, yAralik;
if (xDegerleri.Count() > 10)
{
    xAralik = xDegerleri.Length / 10;
}
else
{
    xAralik = 1;
}
if (yDegerleri.Count() > 10)
{
    yAralik = yDegerleri.Length / 10;
}
else
{
    yAralik = 1;
}
chart1.Series.Clear();
for (int s = 0; s < k; s++)
{
    chart1.Series.Add("" + (s + 1).ToString() + ".Küme");
    chart1.Series[s].ChartType = SeriesChartType.Point;
    chart1.Series[s].MarkerSize = 7;
    int a;
    for (a = 0; a < dataGridView1.Rows.Count; a++)
    {
        if (grupla1[s, a] == 1)
        {
chart1.Series[s].Points.AddXY(veriler1[0, a], veriler1[1, a]);
        }
    }
}
vt.baglan();
sql = "SELECT * FROM ozellikler WHERE indis=" +
Convert.ToInt32(comboBox1.SelectedValue.ToString());
SqlCommand kmt = new SqlCommand(sql, vt.con);
vt.con.Open();
SqlDataReader oku = kmt.ExecuteReader();
if (oku.Read())
{
  chart1.ChartAreas[0].AxisX.Title=""+oku["tam_adi"].ToString();
  oku.Close();              }
  vt.con.Close();
  sql = "SELECT * FROM ozellikler WHERE indis=" +
Convert.ToInt32(comboBox2.SelectedValue.ToString());
```

```
  kmt.CommandText = sql;
  vt.con.Open();
  oku = kmt.ExecuteReader();
  if (oku.Read())
  {
  chart1.ChartAreas[0].AxisY.Title=""+oku["tam_adi"].ToString();
    oku.Close();
  }
  vt.con.Close();
   chart1.ChartAreas[0].AxisY.Interval = yAralik;
   chart1.ChartAreas[0].CursorX.Interval = xAralik;
}
KmeansDetaylariniYaz();
}
private void KmeansDetaylariniYaz()
{ … }

private void btnYeni_Click(object sender, EventArgs e)
{ … }

private void btnYenile_Click(object sender, EventArgs e)
{ … }

private void btnSil_Click(object sender, EventArgs e)
{ … }

private void btnKaydet_Click(object sender, EventArgs e)
{ … }

private void btnYeniKayit_Click_1(object sender, EventArgs e)
{ … }

private void btnGuncelle_Click(object sender, EventArgs e)
{ … }

private void btnVarsayilan_Click(object sender, EventArgs e)
{ … }

private void btnTahminEt_Click(object sender, EventArgs e)
{ … }

private void btnKmeansCalistir_Click(object sender, EventArgs e)
{ … }

private void btnDetaylar_Click(object sender, EventArgs e)
{ … }

private void btnDetaylar2_Click(object sender, EventArgs e)
{ … }

public static void YeniKaydiAl(){ … }
}
}
```

# ÖZGEÇMİŞ

## Kişisel Bilgiler

| | |
|---|---|
| Adı Soyadı | : Hatice ÇATALOLUK |
| Doğum Yeri ve Tarihi | : Konya / 05.08.1987 |

## Eğitim Durumu

| | |
|---|---|
| Lisans Öğrenimi | : Selçuk Üniversitesi<br>Mühendislik-Mimarlık Fakültesi<br>Bilgisayar Mühendisliği |
| Bildiği Yabancı Diller | : İngilizce |

## İş Deneyimi

| | |
|---|---|
| Stajlar | : Dokuz Eylül Üniversitesi Rektörlük Bilgi İşlem Dairesi<br>Elfa Bilgi ve İletişim Teknolojileri |
| Çalıştığı Kurumlar | : Bilecik Üniversitesi Mühendislik Fakültesi |

## İletişim


| | |
|---|---|
| Adres | : Bilecik Üniversitesi Gülümbe Kampüsü Mühendislik Fakültesi |
| Tel | : 0228 216 01 01 - 1367 |
| E-Posta Adresi | : hatice.cataloluk@bilecik.edu.tr |


**Tarih:** 23.12.2011

**İmza:**