T.C.

İSTANBUL KEMERBURGAZ ÜNİVERSİTESİ

Fen Bilimleri Enstitüsü / Elektrik ve Bilgisayar Mühendisliği

YAPAY ÖĞRENME İLE HASTALIK RİSKİ TAHMİNİ

Savaş Karanfil

Yüksek Lisans Tezi

İstanbul, 2017

YAPAY ÖĞRENME İLE HASTALIK RİSKİ TAHMİNİ

Savaş Karanfil

İstanbul Kemerburgaz Üniversitesi

Yüksek Lisans, Elektrik ve Bilgisayar Mühendisliği ’ne

sunulmuştur.

Bu çalışma tarafımızca incelenmiş olup, kapsam ve kalite açısından Yüksek Lisans tezi olmaya yeterli bulunmuştur.

Danışman

Yrd. Doç. Dr. Ayşe Yılmazer

İnceleme Komitesi Üyeleri (İlk isim jüri başkanına, ikinci isim tez danışmanına aittir.)

Yrd. Doç. Dr. Tuğçe Ballı Altuğlu (Jüri)

Yrd. Doç. Dr. Ayşe Yılmazer (Jüri)

Yrd. Doç. Dr. Yasa Ekşioğlu Özok (Jüri)

Bu çalışma bir Yüksek Lisans tezinin tüm gerekli şartlarını taşımaktadır.

Bölüm Başkanı

Yrd. Doç. Dr. Emrullah Fatih Yetkin

Enstitü Müdürü

Üniversite onayı: 22 09 2017

Doç. Dr. Oğuz Bayat

Bu dökümandaki tüm bilgilerin akademik kural ve etiğe bağlı kalınarak yazıldığını ve tez yazım kuralları kapsamında bu çalışmada bulunan ve original olmayan bütün bilgi ve materyallerin referanslandırıldığını temin ederim.

Savaş Karanfil

İTHAF

Sevgili babam Mustafa Karanfil'in anısına

ÖZET

YAPAY ÖĞRENME İLE HASTALIK RİSKİ TAHMİNİ

Savaş Karanfil,

Yüksek lisans, Elektrik ve Bilgisayar Mühendisliği, İstanbul Kemerburgaz Üniversitesi,

Danışman: Yrd. Doç. Dr. Ayşe Yılmazer

Bilgiye sahip olmanın ve onu kullanmanın önemli olduğu günümüzde güçler dengesi bilgi üzerine yoğunlaşmaktadır. Çeşitli kaynaklardan ve yöntemlerle toplanan bilgilerin belirli bir disiplin ve sistem dâhilinde analiz edilmesi sonucunda ortaya çıkan sonuçlar, ekonomik, siyasi, sağlık ve teknolojik alanlarda kullanılmaktadır. Bilgiyi zamanında ve doğru olarak kullananlar istedikleri sonuca kestirmeden ve süratli bir biçimde ulaşmaktadırlar. Yapay Öğrenme ile eldeki verilerden üstü kapalı, çok net olmayan, önceden bilinmeyen ancak potansiyel olarak kullanışlı bilgi çıkarılabilir. Yapay Öğrenme kendi başına bir çözüm değil çözüme ulaşmak için verilecek karar sürecini destekleyen, problemi çözmek için gerekli bilgileri sağlamaya yarayan bir araçtır.

Bilgi kaynağının yanı sıra, bilginin doğruluğu da önemli bir sorundur. Bir bilginin veya daha somut ifadeyle mesela bir rakamın doğru olup olmadığı nasıl anlaşılmaktadır? Bilginin doğruluğu konusunda iki kriter vardır. Aynı sonucu işaret eden verilerin yoğun olması bilginin doğru olduğu yönündeki ilk kriterdir. Bir değer ne kadar yoğunsa o kadar inandırıcı olmaktadır. Ne kadar güçlü bir ilişki olduğu tespit edilirse, o kadar doğruluğuna hükmedilebilir. Hangi miktarda verinin

toplanması gerektiği ayrı bir sorundur. Veri miktarı, kullanılan metoda bakılmaksızın çalışmanın amacına göre belirlenmektedir.

Gün geçtikçe çoğalan veri yığınlarından anlamlı ve faydalı bilgiye ulaşabilmek için "Yapay Öğrenme" başlığı altında yöntemler geliştirilmeye başlanmıştır.

Tanımlanan problem için en uygun modelin bulunabilmesi, olabildiğince çok sayıda modelin kurularak denenmesi ile mümkündür. Bu nedenle veri hazırlama ve model kurma aşamaları, en iyi olduğu düşünülen modele varılıncaya kadar yinelenen bir süreçtir.

Çalışmanın amacı Tıp alanında uygulanması düşünülen Yapay Öğrenme çalışmalarına örnek teşkil etmesi açısından bir plan çıkarmaktır.

Bu araştırmada sağlıklı kişilerin ileride yakalanması ihtimal kalp hastalığını belirlemek, düşük risk, orta risk ve çok riskli teşhisi koymaktır. Bu sebepten, daha önce kalp hastalığına yakalanmış kişilerin kişisel verilerine ihtiyaç duyulacaktır. Bu veri setini sınıflandırmak için şu algoritmalar kullanılacaktır: yapay sinir ağları, karar ağaçları ve rasgele orman. Verilerin birçoğunun kategorik veri olmasından dolayı bu algoritmalar seçilmiştir. Sınıflara ayrılan bu verilerden sağlıklı kişilerden alınacak bazı test sonuçları ve kişilik özellikleri baz alınarak yapay öğrenme yöntemleri ile önceden eğitilen veriler test edilip, bu kişilerin hasta, hasta değil veya riskli oldukları anlaşılacaktır.

Bu hususta bakıldığında çalışmanın amacı geliştirilecek yöntembilim ile saklı olan ve bilinmeyen bilgilere ulaşmaktır. Bunun için farklı tipteki veriler sınıflandırılacak, eğitilecek yeni veriler test edilecek ve yordama yapılacaktır. Böylece kaynaktan hedefe giden süreçte hedef karar vermede etkilenecektir. Bu şekilde çıkarılmak istenen bilgiye ulaşılmış olacaktır.

Anahtar kelimeler: Yapay Öğrenme, Sınıflandırma, Yapay Sinir Ağları, Karar Ağaçları, Rastgele Orman, Öznitelik Çıkartmak, Tıp Alanında Yapay Öğrenme, Risk Değerlendirmesi.

ABSTRACT

Nowadays, it is important to have knowledge and use it. As a result of analyzing information gathered from various sources and methods within a certain discipline and system, the results are used in economic, political, health and technological fields. Those who use the information in a timely and accurate way reach the result without any expectation and speed With Machine Learning, information that is obscure, unclear, unpredictable but potentially useful can be extracted from the data at hand.

Machine Learning is not a solution on its own but a tool to support the decision process to achieve solution, and to provide the necessary information to solve the problem.

Besides the source of information, the truth of the information is also an important question. How does one know, for example, whether a piece of information or a more concrete expression, for example, a number, is correct? There are two criteria for the accuracy of information. The same result is the first criterion that information is intuitive because of the intensity of the data pointing to it. The more intense a value is, the more convincing it is. The stronger the relationship is, the more accurately it can be judged. There is a separate question about which amount should be collected. The amount of data is determined by the purpose of the study regardless of the metric used.

In order to reach meaningful and useful information from the growing number of data, methods under the title of "Machine Learning" started to be developed.

It is possible to find the most suitable model for the defined problem by experimenting with as many models as possible. For this reason, the stages of preparing data and establishing a model are a recurring process until the model is considered to be the best.

The aim of the study is to make a plan in terms of being an example for the Machine Learning studies that are considered to be applied in the field of medicine.

In this study, determining the probable cardiac disease in the future of healthy persons is to establish low risk, medium risk and very risky diagnosis. The following algorithms will be used to classify this dataset: artificial neural networks, decision trees, and random forests. These algorithms have been chosen because many of the data are catagoric data. Based on some test results and personality traits from healthy subjects, these data, which are classified into classes, will be tested with artificial learning methods and pre-educated data, and it will be understood that these persons are not sick, ill or risky.

Looking at this issue, the aim of the study is to reach the hidden and unknown information through the methodology to be developed. For this, different types of data will be classified, the new data to be trained will be tested and the procedure will be done. In this way, the target will be influenced by decision making from the source to the target. In this way, the information desired to be extracted will be achieved.

Key words: Machine Learning, Classification, Artificial Neural Networks, Decision Trees, Random Forest, Attribute Extraction, Machine Learning in Medicine, Risk Assessment.

İÇİNDEKİLER

TABLO LİSTESİ

ŞEKİL LİSTESİ

KISALTMALAR LİSTESİ

OLAP: Online Analytical Processing

VTBK: Veri Tabanından Bigi Keşfi

YSA: Yapay Sinir Ağları

ÇKA: Çok Katmanlı Ağ

TKA: Tek Katmanlı Ağ

AID: Automatic Interaction Detector

C&RT: Classification And Regression Trees

MARS: Multivariate Adaptive Regression

CART: Classification and regression tree

OOB: Out of bag

RF: Random Forest

AKB: Avrupa Kalp Birliği

EKG: Elektrokardiyografi

EKO: Ekokardiyografi

PET: Positron yayınlayıcı tomografi

BKİ: Boy kilo endeksi

TÜİK: Türkiye İstatistik Kurumu

WHO: Word Health Organization

GİRİŞ

Araştırmanın amacı sağlıklı insanların gelecekte yakalanması ihtimal kalp hastalığını bulmak ve bu doğrultuda, düşük risk, orta risk ve çok riskli gibi derecelerde sınıflara ayırmaktır. Bu sebepten ilk önce geçmişinde kalp hastalığı geçirmiş insanların verilerine ihtiyaç duyulacaktır. Edinilen bu verilerde yaşı, cinsiyeti, diabet, böbrek yetmezliği, geçirdiği ameliyatlar gibi özellikler vardır. Karışık olan bu verilerin ilk olarak sınıflara ayırabilmek için yapay öğrenme metotları denenecektir. Yapay öğrenme metotlardan biri olan yapay sinir ağları, karar ağaçları ve rasgele orman sınıflandırma algoritmaları kullanılacaktır. Verilerin birçoğunun katagorik veri olmasından dolayı bu algoritmalar seçilmiştir. Veriler sınıflandırıldıktan sonra sağlıklı insanlardan alınan bazı test sonuçları ve kişilik özellikleri temelinde yapay öğrenme yöntemleri ile önceden eğitilen veriler test edilip, bu insanların, hasta olup olmadığı ve hasta ise riskin ne kadar olduğunu anlamaya çalışılacaktır.

Araştırmanın daha rahat ilerlemesi adına kalp dotorları ile görüşmeler yapılmıştır. Verilerin elde edilmesi çalışma için en önemlisidir. Hasta verileri elde ettikten sonra, onlardan sınıflandırma yapmak için bizim işimize yarıyacak olan özellikleri seçmekte bir o kadar önem arz eder. Bir hastanenin kardiyoloji kliniğideki doktorlar ile randevular alınarak görüşmeler yapılmıştır. İstenilen metotların kullanılabilmesi için kendilerinin teknik bilgilerine başvurulmuştur. Bu sayade başhekimin de izni ile gereken ham veriye(veri seti) ulaşılmıştır.

1. YAPAY ÖĞRENME

1.1. Yapay öğrenme nedir?

Yapay öğrenme bilgisayar bilimlerinin farklı dallarda yapılan çalışmalar sonucunda ortaya çıkan bir alt daldır. 1959 senesinde yapay zeka ve örüntü tanımada yapılan araştımalar sonucu ortaya çıkmıştır. Yapay öğrenme işlevselliğini öğrenebilen ve bu öğrenmiş verilerden tahmin yapmasından alır. Bunun için birçok algoritma kullanılır. Bu algoritmalar tahminleri doğru bir şekilde yapabilmek için, daha önceden birer model oluştururlar. Daha sonra gelen kayıtları bu model doğrultusunda sınıflandırır ya da benzetim yaparlar.

1.2. Yapay öğrenmede kullanılan metotlar

Yapay öğrenme ;

- **Gözetimli Öğrenme(denetimli)**
- **Gözetimsiz Öğrenme(denetimsiz)**
- **Yarı Gözetimli Öğrenme**
- **Pekiştirmeli Öğrenme**

olmak üzere 4 kısımda inceleyebiliriz.

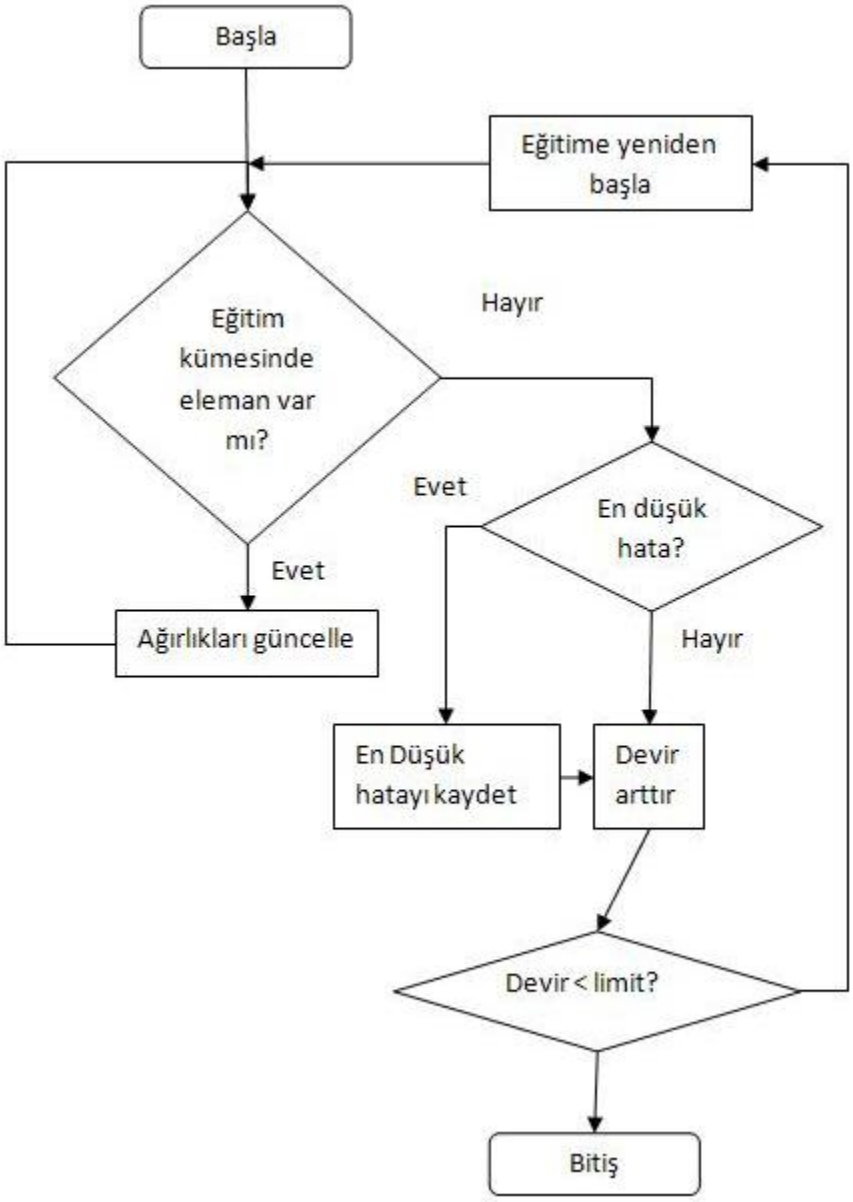
1.2.1. Gözetimli öğrenme(denetimli)

Denetimli öğrenme yönteminde giriş verisi olarak verilen veriden istenen sonuç kümesinin elde edilmesi beklenir. Gözetimli öğrenme yönteminde makineyi eğitmek için bir eğitim kümesi ve eğitim kümesinde verilen değerler için istenen çıktı değerleri verilir. Bu eğitim kümesini kullanarak makine gelecekte kullanmak üzere öğrenir. Gelecekte siz makineden bir şey istediğinizde önceden verdiğiniz eğitim kümesine bakar ve istenen şey ile eğitim kümesinde benzeyen verileri karşılaştırır ve ona göre bir çıkarım yapar [2].

Gözetimli öğrenme yönteminde Şekil 1'de görüldüğü üzere makine eğitim kümesini ve yaptığı önceki yanlışları en düşük hatayı bulmak için kullanıyor. Amaç en düşük hata payı ile en isabetli tahmin yapmak.

Örneğin: Bir ilçe olsun ve o ilçedeki ev fiyatlarını düşünün. O ilçedeki ev fiyatları verileri sizin elinizde var ve bunları makineye tanıttınız. Örnek olarak 50 metrekare evin fiyatı 70.000 tl diyelim. 100 metrekare olan bir başka evin fiyatı ise 160.000 tl olacak şekilde makineye eğitim kümemizde bu verileri tanıttık ve bizim 75 metrekare bir evimiz var diyelim. Makineden bu ev için bir fiyat biçmesini istiyoruz. Şimdi makinenin yapması gereken 100 metrekare evin fiyatını almak sonra 50 metrekare evin fiyatını almak ve öğrenmek ve daha sonra bu verilere göre bizim evimiz olan 75 metrekarelik eve bir fiyat biçmek.

Makinenin biçeceği fiyat sizce ne olabilir? Elbette ki 120.000 tl olacaktır. Burada iki verimiz(ev) vardı. Çoğu zaman bu kadar az veriyle çalışılmaz [2].



Şekil 1: Danışmanlı öğrenme algoritması [2].

Gözetimli öğrenme konusu da aslında kendi içinde yöntemlere ayrılıyor. Yukarıdaki örnekte kullanılan yöntem Lineer Regresyon yöntemiydi. Gözetimli öğrenme konusunda Lineer Regresyon gibi birçok algoritmalar var bu algoritmalar “Sınıflandırma” ve “Regresyon” olarak ikiye ayrılabilir [2].

Sınıflandırma kısmı,

- **En yakın komşular algoritması(KNN)**
- **Ağaçlar**
- **Mantıksal Regresyon**
- **Naive Bayes Algoritması**
- **Destekçi Vektör Makineleri**

olmak üzere 5 kısımda incelenir.

Regresyon kısmı,

- **Lineer Regresyon**
- **Polinomal Regresyon**
- **Karar Ağaçları**
- **Random Forests**

şeklinde gösterilebilir.

1.2.2. **Gözetimli öğrenme(denetimli)**

Gözetimsiz öğrenme, gözetimli öğrenmeden farklı olarak bizlere bir çıkış değeri vermemesidir. Gözetimsiz öğrenmenin amacı veri içerisindeki ilişkilerin ve yapının öğrenilmesidir.

Bu yöntemde algoritmanın kendi kendine keşifler yapması, gizli veya kayıp olan örüntüleri tamamlaması beklenir. Ayrıca gözetimsiz eğitimde bir eğitim kümesi bulunmaz.

Bunun dışında gözetimsiz öğrenme algoritması da kendi içinde,

- **Kümeleme**

- **İlişki analizi**

olacak şekilde ikiye ayrılır.

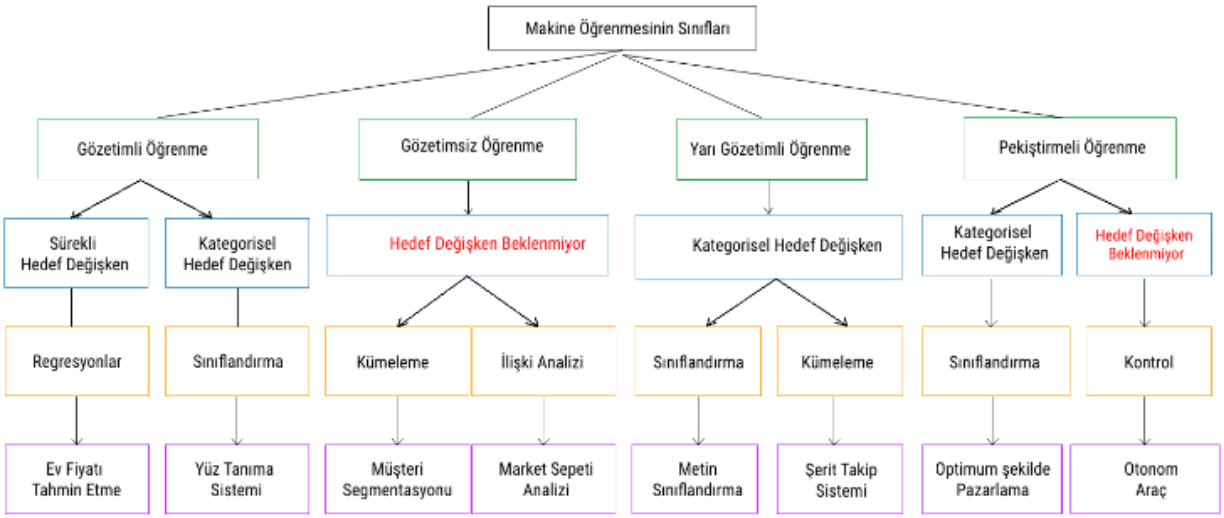
1.2.3. Yarı gözetimli öğrenme(Denetimli)

Küçük gözetimli verisetleri veya büyük gözetimsiz verisetleri ile öğrenmedir. En az hata payı ile en isabetli sonucu bulmak için etiketli veya etiketsiz verilerin ikisini de kullanır.

1.2.4. Pekiştirmeli öğrenme

Pekiştirmeli öğrenme yazılım ajanlarının ödül sistemi ile çalıştıgı bir algoritmadır. Ödül sistemini biraz daha açmak gerekirse. Örneğin: Bir ortamda bulunan bir ajanımız var bu ajanımızın en kısa sürede ortamdaki sizin belirlediğiniz hedefi bulması gerekiyor. Ajanımızın ilk aşamada zekası yok. Ajanın geçeceği yollarda ona yol göstermek için ödüller ve bunun yanında yanlış yollara girdiğinde negatif puan veren şeyler bulunuyor. Ajanımız bu topladığı artı veya eksi puanları dikkate alarak sizin belirlemiş olduğunuz bitiş çizgisine en kısa sürede varmaya çalışacaktır.

Şekil 2'de yukarıda anlatılan yapay öğrenme ve yöntemleri gösterilmiştir. Bunun yanısıra kullanıldığı birer örnek verilmiştir.



Şekil 2: Makine öğrenmesinde kullanılan yöntemler ve birer örnekleri [2].

1.2.5. Yapay öğrenmenin sektör bazlı kullanım durumları

Üretim – İmalat: Öngörülü bakım ve durum izleme, malzeme ve stok tahminleri, satın alma eğilimleri, talep tahminleri, süreç optimizasyonu ve telematik.

Perakende: Tahminli envanter planlaması, tavsiye motorları, satış ve çapraz kanal pazarlama, pazar segmentasyonu ve hedeflemesi, yatırım geri dönüşleri ve değerleme.

Sağlık ve Yaşam Bilimi: Hasta verilerini değerlendirerek gerçek zamanlı olarak uyarı ve teşhisler, hastalık tanımlama ve risk katmanlaşma, hasta triyaj optimizasyonu, proaktif sağlık yönetimi, sağlık hizmeti analizleri.

Seyahat: Dinamik ücretlendirme, sosyal medya – tüketici geri bildirim ve etkileşim analizi, müşteri şikayet çözümleri, trafik kalıpları ve tıkanıklık yönetimi.

Finans ve Mali Hizmetler: Risk analizi ve regülasyonu, müşteri segmentasyonu, çapraz satış, satış ve pazarlama kampanyası yönetimi, krediye uygunluk değerlendirmesi.

Enerji, geri bildirim ve Kamu Hizmetleri: Güç kullanım analizleri, deprem verilerinin işlenmesi, karbon emisyon ve ticaret, müşteriye özel fiyatlandırma, akıllı şebeke yönetimi, enerji arz ve talebi optimizasyonu.

Unutulmamalıdır ki; iş ve müşteri deneyimlerinizi elinizde bulunan verilerle analiz etmek ve deneyimlemek size farklı kapılar açacaktır. Şirketler için inovasyon, verimlilik, bütünlük ve kurumsal öğrenim günümüzde kaçınılmaz hale gelmiştir [2]**.**

1.3. Geniş veri

Şirketler büyük ölçekte verileri sahiptirler. Verilerin büyüklüğü şirketler için bazı problemler çıkmasına sebep olur. Bu kadar büyük veri ile uğraşmak sağlam bir donanım ve yazılım ile olur. Yapay öğrenme ile bu büyük veriyi tanıyıp onunla uyum sağlayabiliriz. Artık veriler sadece büyük değildir. Günüzde veriler geniştirde. Mesela bir web sitesi olsun. E ticaret ile ilgili bir site. Ve de bu site ait bir veritabanı. Ne kadar müşteri var ise veritabanında o kadar kayıt olacaktır. Yani müşteri sayısı çok ise çok kayıt olacaktır. Veri setindeki sütünlar ise o müşteriye ait özellikleri

temsil edecektir. Günümüzde müşteriler ya da kişiler hakkında çok fazla bilgi vardır. Nereden alışveriş yaptığı, gezindiği sayfalar, yapmış olduğu yorumlar gibi birçok bilgi var artık. Bu sebepten dolayı veritanları artık genişliyor. Yani sütün sayısı artıyor. Bu demektir ki kişilerin özellik sayısı büyüyor [1].

1.4. Sebep sonuç ilişkisi değil, öngörüler.

Yapay öğrenme en temel olarak tahminlerde bulunmak için kullanılır. Öngörüler yaparak gelecek hakında bilgi sahibi olmamıza yardım eder. İş dünyasının karşılaştığı sorunlar:

- Öneride bulunmak. Müşterileri tanıyıp onlar için özel öneriler bulmak.
- Tahminlerde bulunmak.
- Öngörülerde bulunmak. Özellikle çalışanlar için.
- Kredi riski hesaplamak gibi.

Yukarıdaki maddelerin birçoğu ortak özlliklere sahiptir. Mesela doğru bir karar vermek için birden fazla değişkene ihtiyaç vardır. Bunun anlamı geniş bir verinin olacağıdır. Tahimlerin doğruluğunu test etmek önemlidir. Bu özelliğe sahip olması gerekir elimizde verinin. Hangi ürüne tıklamış, hangi ürünü beğenmiş, sitenin sundukların olanaklardan ne kadar yararlanmış gibi. Bu işleyişin ardından ise bir öngörü ya da tahmin çıkartmak. Bu tahminin ise isabetli olması gerekliliği vardır.

Yapay öğrenme diğer metotlara göre farklılık gösterir. Mesele istatiksel yöntemlere göre biraz daha sebep sonuç ilişkişleri ile daha fazla ilgelenir. Ortam değişikliği yapıldığı zaman pek bir şeyin fark etmeyeceğidir. Bunu yapmak yerine tahmin ve öngörü yapmaya zaman ayırırsınız. Doğru kararı verebilmemiz için doğru ortamı modellememiz gerekir. Örneğin: evden çıkarken yanımıza şemsiye alalım mı? Almayalım mı? Gibi iki durumuda kaldığımı düşünelim. Karar vermek için ilk önce hava durumu tahmin etmeliyiz. Ama bunu tahmin etmek için bilgilerimiz sınırlıdır. Gökyüzünü gözlemlemeli, nasıl havanın yağış topladığını izleriz. Sınırlıdır fakat çok yardımcı olur.

Aynı durum yapay öğrenme için ise farklılık göstermez. Kişiye özel bilgiler ve tahminler kar yüzdesini çoğaltabilir ama bize kişilerin sevdiği ürünleri neden sevdiği hakkında bir bilgi vermez. Ya da onların beğenilerini nasıl etkileyip, değişekliğe doğru yön vereceğinizi anlatmaz. Bu durumların sonuçlarında yapay öğrenmenin değeri daha iyi anlaşılır [1].

1.5. Sinyali gürültüden ayırmak

Yapay öğrenmenin faydalarından yukarıda fazlasıyla değindik. Bundan sonra yapısı hakkında biraz öğrenme yapalım. Yapay öğrenmenin yapısını neler oluşturur ve nasıl çalıştığını konuya ait terimler ile ifade edelim. Bu terimlerin açıklamalarını yapmadan önce bazılarını aşağıdaki listede gösterelim [1]:

a) Öznitelik çıkartmak

b) Düzenlileştirmek

c) Çapraz doğrulama

1.5.1. Öznitelik çıkarma (feature extraction)

Sınıflandırma veya buna benzer modeller oluştururken bize lazım olan değişkenlere özellik (öznitelik) denir. Biz de model oluştururken hangi özellikler ile çalışacak ise onları çıkartmamız gerekir. Tüm veri ile model oluştıralabileceği gibi, bazı nitelikler eklenebilir. Yeni eklenen özellik diğer özelliklerden etkilenebilir. Mesela doğum gününün kişinin diğer özelliklerinden elde edilmesi buna bir örnek olabilir. Bu şekilde öznitelik sayısında artışlar ya da azalış olması modellemenin başarısını farklı yönde değiştirir.

Bir başka örnek vermek gerekir ise yüz tanıma gösterilebilir. Buradaki özellikler fotağraftaki kişi ile ilgilidir. Bu kişinin yüzündeki karekteristik özellikler bizim özelliklerizi oluşturur. Göz tipi, burun tipi, tenin rengi gibi özellikler bunların arasındadır. Bu özellikler arasından çıkartma ya da ekleme işlemleri yapılabilinir. Modelin en iyi sonucu verebilmesi için bu süreç çok önemlidir. Çünkü bundan sonrası algoritmaların uygulanması süreci başlıyacaktır. Bu özellikler ile kişi tanınmaya çalışınılır [1].

Özellik çıkartılaması için birçok yöntem bulunur. Bunların içerisinde bu süreçte kullanılan programlardan yardım alınır. Bu programların birçoğunda özellik çıkartma algoritmaları bulunur. Bu şekilde özelliklerin, sınıf özelliğini ne kadar etkileyip etkilemediği bulunabilinir. Ya da kod ile çalışan programlarda kod yazılarak özelliklerin model için ne kadar önemli olduğu bulunabilir. Daha önce de bahsedildiği gibi eğer sınıflandırma algoritmaları kullanılacak ise özellikler iyi incelenmelidir. Hangisinin sınıf özelliğine karşı etkili olup olmadığı bulunmalıdır. Böylece etkisi az olan özelliklerin çıkartılma şansı olabilir. Çıkartılıp model tekrardan kurulur vede sonuçlar gözden geçirilir. Eğer doğruluk oranları artı yönde bir artış yani daha fazla isabetli tahmin yapmış ise faydalı olur.

Sınıflandırmaya göre kümeleme biraz daha karmaşık bir konudur. Sınıflandırma ve kümeleme gibi yöntemler verileri düzenlemekte ve organize etmek için de kullanılır. Bu şekildeki yapay öğrenmeye denetimsiz öğrenme denir. Tahminlerin hedefi olarak bir ölçülen bir olgu yoktur [1].

1.5.2. Düzenlileştirme (regularization)

Yapay öğrenmedeki genelleme, modelin öğrendiği kavramların, eğitim sırasında görülmeyen örneklere ne kadar iyi uygulandığına işaret eder. Çoğu makine öğrenme modelinin amacı, gelecekte görünmeyen veriler için iyi tahminler yapmak için eğitim verilerinden genel bir şekilde genelleme yapmaktır. Modellerin ayrıntıları ve gürültüyü eğitim verisinden çok iyi öğrenmesi durumunda aşırı uyuma olur, ancak genelleme iyi olmaz, bu nedenle veri testi için performans düşüktür. Veri kümesi, öğrenilmesi gereken model parametrelerinin sayısıyla karşılaştırıldığında çok küçük olduğunda çok yaygın bir sorundur. Bu problem, milyonlarca parametrenin olması nadir görülmeyen derin sinir ağlarında özellikle şiddetlidir [2].

Düzenlileştirme aşırı uygunluğun önlenmesinde önemli bir bileşen oluşturur. Ayrıca, bazı parametrelerin sıfıra sürülmesi gibi doğruluğu muhafaza ederken modelleme kapasitesini azaltmak için bazı normalleştirme teknikleri kullanılabilir. Bu, işlemci gücünün sınırlandığı mobil ortamda modelin boyutunu düşürmek veya değerlendirme maliyetini düşürmek için istenebilir.

Günümüzde endüstride kullanılan en yaygın normalleştirme tekniklerini gözden geçiriyor:

- Veri seti büyütme
- Erken durma
- Ağırlık cezası

Veri seti büyütme: Öğrenme algoritması daha fazla eğitim verisi işlerse aşırı uyumsuz bir model (sinir ağı veya herhangi bir başka model tipi) daha iyi performans gösterebilir. Mevcut bir veri kümesi sınırlı olabilirken, bazı makine öğrenme sorunları için sentetik veri oluşturmak için nispeten kolay yollar vardır. Resimler için bazı yaygın teknikler resmin birkaç piksel, döndürme ve ölçekleme tercüme edilmesini içerir. Sınıflandırma problemleri için genellikle rasgele negatif enjekte etmek uygundur.

Sentetik verilerin nasıl üretileceği konusunda genel bir tarifi yoktur ve problemle problem arasında çok değişir. Genel ilke, gerçek dünya daki değişimleri mümkün olduğunca yansıtan işlemleri uygulayarak veri kümesini genişletmektir. Pratikte daha iyi veri kümesine sahip olmak, mimariden bağımsız olarak modellerin kalitesine önemli ölçüde yardımcı olur.

Erken durma: Modelin performans doğrulama kümesindeki performansı kötüleştiğinde, eğitim prosedürünü kesintiye uğratan erken müdahale mücadeleleridir. Geçerlilik kümesi, eğim inişinde asla kullanmadığımız, ancak test kümesinin parçası olmayan bir dizi örnektir. Doğrulama örnekleri gelecekteki test örneklerini temsil edecek şekilde düşünülmüştür. Erken durma, hiper parametrenin çağ / adım sayısını efektif olarak ayarlamaktadır.

Sezgisel olarak, model daha fazla veri gördüğünden ve kalıpları ve bağıntıları öğrenirken, hem eğitim hem de test hatası düşer. Eğitim verileri üzerinde yeterli geçtikten sonra, model, verilen eğitim setinde fazla konuşma ve öğrenme gürültüsü yaşayabilir. Bu durumda, eğitim hatası aşağıya doğru devam ederken test hatası (genelleme ne kadar iyi) kötüleşecektir. Erken durma, asgari test hatası ile bu doğru anı bulmakla ilgilidir.

Uygulamada, aslında durdurmak yerine, insanlar normalde kontrol noktalarını düzenli aralıklarla düzenli aralıklarla kaydetmek için kurulurlar ve gerçeklerden sonra en iyi adayı seçerler [2].

Ağırlık cezası L1 ve L2: Ağırlık cezası, normalleştirme için standart bir yol olup, yaygın olarak diğer model türlerini eğitmek için kullanılır. Küçük ağırlıklı bir modelin bir şekilde büyük ağırlıklı bir ağa nazaran daha basit olduğu varsayımına kuvvetle bağlıdır.

Cezalar, karşı koymak için büyük degradeler olmadığı sürece ağırlıkları küçük veya var olmayan (sıfır) tutmaya çalışır; bu da modelleri daha fazla yorumlanabilir yapar. Literatürde kilo cezaları için alternatif bir isim, ağırlıkları sıfıra düşürmeye zorladığı için "ağırlık düşüşü" denir.

L2 normu:

- Ağırlığın kare değerini cezalandırır.
- Tüm ağırlıkları daha küçük değerlere itme eğilimi gösterir.

L1 normu:

- Ağırlığın mutlak değerini (v-şekli işlevi) cezalandırır.
- Bazı ağırlıkları tam olarak sıfıra götürme eğilimi gösterir (modelde seyreklik getirir), bazı ağırlıkların büyük olmasına izin verirken.

Düzenlileştirme, yapay öğrenmenin merkezi bir temasıdır ve eğitim sonuçlarını önemli ölçüde artırabilir [2].

1.5.3. Çapraz doğrulama (cross validaition)

K-kez çapraz doğrulama yöntemi sınıflandırıcı modellerin bir veri kümesi üzerinde yapılan sınıflandırma işleminin sonuçlarının tutarlı olması için kullanılmaktadır.

Metodun uygulanmasından önce k parametresinin belirlenmesi gerekmektedir. K parametresi veri kümesinin kaç parçaya bölüneceğini belirtmektedir. Şekil 3'te görüldüğü üzere k adet sınıflandırma işlemi yapılmaktadır ve her adımda bölünen parçalardan bir tanesi test işlemi için ayrılmakta geriye kalan k-1 tanesi sınıflandırıcının eğitimi için kullanılmaktadır. K adım sonra elde edilen sınıflandırma sonuçlarının ortalaması alınarak genel sınıflandırma sonucu elde edilmektedir. K parametresi 10 olarak belirlenen bir çapraz doğrulama işlemi aşağıda

görülmektedir. 10 parçaya bölünen veri kümesindeki 9 parça eğitim kümesi ve geriye kalan diğer parça test kümesi olarak kullanılmakta ve bu işlem 10 adımda ve her adımda farklı bir parça test kümesi alınarak gerçekleştirilmektedir [2].

	1.Parça	2.Parça	3.Parça	4.Parça	5.Parça	6.Parça	7.Parça	8.Parça	9.Parça	10.Parça
1. Adım	Testi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi
2. Adım	Eğitimi	Testi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi
3. Adım	Eğitimi	Eğitimi	Testi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi
4. Adım	Eğitimi	Eğitimi	Eğitimi	Testi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi
5. Adım	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Testi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi
6. Adım	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Testi	Eğitimi	Eğitimi	Eğitimi	Eğitimi
7. Adım	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Testi	Eğitimi	Eğitimi	Eğitimi
8. Adım	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Testi	Eğitimi	Eğitimi
9. Adım	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Testi	Eğitimi
10. Adım	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Eğitimi	Testi

Şekil 3: K-kez çapraz doğrulama yöntemi [2].

1000 adet müşteri hakkında veri topladığımızı ve bu toplanan verilerle bir öngörü modeli oluşturduğumuzu farz edelim. Modeli oluşturduktan sonra ise modelimizin ne kadar isabetli öngörüde bulunduğunu ölçelim. Buradan bir doğruluk oranı çıkacaktır. Bu oran ileride karşılaştırma yapabilmek için çok işimize yarıyacaktır. Bu oranı değerlendirmek için elimizdeki veri setini eğitim seti (900) ve test seti (100) olarak ayırıyoruz.

Eğiitm setinde modeli oluşturduktan sonra tahmin oranı için bu modeli test setinde denememiz gerekir. Önemli olan kısım modelin test setini hiç tanımıyor olmasıdır. Vede eğitilmiş modelin test setindeki öngörülerinin isabeti önemli rol oynar. Böylece testler gerçekten ayrı tutulmuş olur.

Tam tersi düşünüldüğünde yani modelin test setini tanıyor olması durumunda bir ezberleme söz konusu olacaktır.

1.6.Yapay öğrenmede kaçınılması gereken hatalar

Yapay öğrenme kullanılan alanlara baktığımızda, fark etmeden ya da bilerek işlenebilecek problemler mevcuttur.

Bunlarda bazılarından bahsedelim. Herhangi bir modelin, sebep sonuç ilişikisi ile karıştırılması olabilir. Tahmin sorunlarında sebep sonuç ilişkisi belirlemek öncelikli olamaz. Bunun yerine bulunan kararların optimizasyonu yapılmaya çalışılmalıdır. Optimizasyon çalışmarı sayesinde yapılan tahminler daha başarılı olacaktır. İki kavram birbirir ile karışmadığı sürece oran yükselecektir.

"Örneklem dışı" ile "bağlam dışı" arasındaki farklı ortaya koymak önemlidir. Geliştirilen bir modelin örneklem dışında başarılı olması demek, eğer birebir aynı ortamdan yeni veri noktaları toplarsak modelin bunların çıktılarını doğru bir şekilde tahmin edebiliyor olması demektir. Ancak modelin başka bir ortama taşındığında da başarılı olacağının herhangi bir garantisi yoktur. Örneğin bir e-ticaret sitesi online alışverişlerin tutulduğu bir veritabanını kullanarak yeni müşteriler kazanmaya yardımcı olan bir model geliştirebilir. Ancak aynı model ürün gamı tamamen aynı olsa bile fiziksel dükkanlarda işe yaramayabilir.

Eldeki verinin yüksek miktarda olmasının bu sorunun üstesinden geleceğini düşünmek cezbedici olabilir, ancak bu doğru değildir. Unutulmamalıdır ki bu algoritmalar güçlerini, yeni durumları daha önce karşılaşmış oldukları benzer durumları içeren büyük veritabanlarıyla mukayese etmelerinden almaktadırlar.

Bir modeli farklı bir bağlamda uygulamaya çalıştığınızda, veritabanındaki bilgiler karşılaşılmakta olan durumlara benzer olmayacaktır. Bu sorunun basit bir çözümü yoktur. Bağlam dışı bir model, hiç bir modele sahip olmamaktan daha iyi olabilir, ancak modelin kısıtları dikkate alınmalıdır.

Yapay öğrenme modellerini eğitme sürecinin bazı kısımları otomatikmiş gibi görünse de modelin ne zaman başarılı olacağını anlamak önemli ölçüde insan muhakemesi gerektirmektedir. Dahası, sürece dahil olan düzenlileştirme ve çapraz doğrulama gibi güvenlik önlemlerinin doğru şekilde kullanıldığını garantilemek için önemli ölçüde eleştirel düşünce gerekmektedir.

Tüm bunlara rağmen, yapay öğrenmenin alternatifi olarak yalnızca insanların karar almasının da kendine özgü önyargıları ve hataları bulunmaktadır. Teknik becerinin ve insani

yargının doğru karışımıyla yapay öğrenme, büyük miktarlardaki veriyi katma değere dönüştürmek isteyen karar vericiler için yeni ve işe yarar bir araç olabilir [1].

2. VERİ TABANINDAKİ BİLGİNİN KEŞFİ

Veri tabanındaki bilgiyi kullanmak için önce onu bulup ortaya çıkartmamız gerekir. En önemli methot Veri Tabanlarında Bilginin Keşfi (VTBK) ingilizcesi ise Knowledge Discovery in Databases (KDD) olarak bilinir.

Geçtiğimiz süreçte veri tabanındaki faydalı deseni bulmak için birçok yöntem kullanılmıştır. Bu işe farklı farklı isimler verilmiştir. Örneğin: yapay öğrenme, enformasyonun keşifi, enformasynun hasadı, bilginin hasadı, örüntü işleme ve verinin madenciliği gibi isimler verilebilir. İstatislikle uğraşanlar, veriyi analiz edenler daha çok yapay öğrenme metotlarını kullanmıştır. Bunun yanı sıra veri tabanı ile uğraşanlar arasında da popüler olmuştur. Veri tabanındaki bilginin keşfi ilk olarak 1989 senesinde bir araştıma grubunun toplantısında ortaya konmuştur [3].

Aslına bakılırsa, Veri Tabanlarındaki Bilginin Keşfi eldeki verinin içindeki bilgiyi yani bizim işimizi kolaylaştıracak olan verinin çıkartılma sürecini işaret eder. Bu süreçte çeşitli yöntemler kullanılır. Ama bu birçok yöntem aslında bu işin bir basamağını teşkil eder. Yapay öğrenme de bu sürecin bir adımı olarak adlandırmak mümkün. Aynı şekilde diğer mettotlarda birer basamak görevi görürler. Veri Tabanlarındaki Bilginin Keşfi desen tanma, verinin madenciliği, yapay zeka, uzman sistemler gibi birçok alanın birleştiği ve geliştiği bir alandır. Amaç dağınık olan büyük verilerden anlamlı örüntü çıkarıp, öngörülerin yapılmasına izin vermektir. Ve de tahminler yapmak için bu bilgi gereklidir.

Bu süreçte Veri tabanlarındaki bilginin keşfinin bir basamağı olarak işlev gören yapay öğrenme veri içerisindeki deseni bulmak için istatiksel ve makine öğrenemesi gibi methotlar kullanmaktadır.

Yukarıdaki anlatına ek olarak akla VTBK'nın makine öğrenmesi, veri madenciliği ve diğer ilgili alanlar içerisindeki herhangi birinden farkı nedir gibi bir soru takılabilir. Cevap; bu sayılan

alanların VTBK'nin yapay öğrenme adımında bazı metotları ona temin etmesidir. Oysaki VTBK, verileri nereye depolanmasından ve ona erişiminden, büyük hacimdeki veri setlerine algoritmaların nasıl ölçekleneceğini, elde edilen sonuçları ne şekilde görselleşebileceği ve yorumlanacağı, insanın makine ile etkileşimini en kullanışlı olarak modellemek, veriden çıkan bilginin bütün basamaklarına odaklanır.

VTBK, mesela yapay zeka gibi bir alanın ilgi alanına girmesinden çok daha geniş bir disiplin içerisi görev yapar. Yapay zeka, verinin madenciliği, desen tanıma, istatistik, makine öğrenmesi ve diğer tekniklerin VTBK'ye katkılar vererek onu alanında daha geniş bir konuma getirir.

2.1. Veri tabanındaki bilginin keşif süreci

VTKB'nin aşamaları veritabanındaki verilerin seçilmesi ile başlar. Daha sonra seçılmiş olan bu veri ön işleme tabi tutulur. Bu işelemlerden sonra deseni çıkarabilmak için yapay öğrenme teknikleri (algoritmaları) kullanılır. Ve de çıkan örüntü ya da desenin yorumlanması şeklinde olur.

VTKB'nin aşamalarıda yapay öğrenme adımı, verinin hangi deseninin alınıp hangisinin alınmayacağı hakkında algoritmaik olarak bize fikir verir. Bütün bir veri setindeki gömülmüş, gizli kalmış olan veriden desenin nasıl yeni bir bilgi olarak alanacağını yine VTBK'nin aşamalarında öğreniriz [4].

VTBK'nin aşamaları kendini sürekli yenileyen özelliklere sahip olması gerekmektedir. Brachman ve Anand, aşamaların interaktif yapısına dikkat çekmişlerdir [5]. VTKB'nin aşamaları önemli olanları aşağıda gösterilmiştir:

1. Adım: Uygulama alanının tasarımı ve geliştirilmesini sağlamak. Ve bu aşamaların amacını belirlemek.

2. Adım: Amaca yönelik bir veri seti üretmek: değerli bilginin çıkarılacağı verisetini bulmak veya alt küme şeklinde olan veri setlerine odaklanmak. Bu adım öngörüler için önemlidir. Veri seti seçmek çok önem arz eder.

3. Adım: Verinin önişlemden geçmesi: modelleme için gerekli bilg nin toplanması, gürültü var ise kaldırılmalı, eksik veriler var ise onlar için çözümler bulunmalı vede buna benzer ön işelmlerin yapıldığı aşamadır.

4. Adım: Verinin indirgenmesi: Amaç doğrultusunda veri setinin en çok niteliyen özelliler bulunmalıdır. Ve eğer istenir ise en az temsil eden özellikler çıkartılabilir. Bu adımda veri setinin boyutunda bir değişiklik yapılırsa, öngörü başarısınıda etkilemiş olur.

5. Adım: VTBK aşamalarının amaçları ile yapay öğrenme tekniklerinin eşitlenmesi: bu aşamada sınıflandırmak, kümelemek, birliktelik yakalamak gibi teknikler uygulanır.

6. Adım: Yapay öğrenme algoritmalarının seçilmesi: Modellerin, analizlerin vede hipotezlerin seçileceği adımdır. Seçilen algoritmalar ve teknikler deseni çıkartmak için kullanılır. Bundan önce hangi algoritmalın veri setine uygulanacağı seçilmelidir. Bu aşamadaki en önemli konu budur. Uygun olmayan algoritmaların modelin başarısını düşürürken, uygun algoritmalar doğru tahmin oranını yükselteceklerdir.

7. Adım: Yapay öğrenme: Uygun olarak çıkartılmış veri setinden ilgilenilen desen çıkartılır. Bunun için sınıflandırmak, kümelemek, ağaçlar ve buna benzer metotler kullanılır.

8. Adım: Çıkartılan desenin yorumlaması: bir sonraki turlarda, adım 1 ile 7'den birine dönülmesi ihtimali ile yapay öğrenme ile desen yorumlanır.

9. Adım: Veri setinden bulunan bilginin birleştirmesi: bu bilgiler ileride yapılması muhtemel başka bir çalışmaiçin toplanabilir veya bir doküman olarak saklanabilir. Aynı şekilde rapor olarakta kullanılabilir. Bu durum bize bilgilerin doğruluğunu kotrol etme şansı da verir. Kontrolleri yaparken farkları nasıl oluştuğunu anlamamızı sağlar.

2.2. Veri tabanlarındaki bilginin keşfi sürecinin yapay öğrenme adımı

Bilginin keşfinin amaçları, sistemin kulanım hedefine göre değişir. Hedefleri ikiye ayırabiliriz:

• Doğrulaması

• Keşif edilmesi

Doğrulama kullanıcıların öngörülerini doğrulatmak ile sınırlıdır. Keşif etmek ise sistemden bağımsız olarak yeni desenler bulabilir.

İlerdeki konularda keşif etmenin amacı olarak bazı canlıların davranışlarını öngörmek için kullanılacaktır.

Yapay öğrenme üzerinde çalışılan veriye bir model giydirmek ve veri setindeki desenleri tanımayı gerektirir.

Bilgiyi çıkarırma görevini model giydirme üstlenir. Modelin nasıl olduğunu, özelliklerinin neler olduğunu, bütünüyle interaktif olup olmadığını anlamak için bu aşamada insanın yargısına ihtiyaç duyulur. Model giydirmede iki temel matematiksel kavram kullanılır. Bunlardan ilki, istatiksellik diğeri ise mantıktır. Modelde istatistiksellik olan yaklaşım belirlenemeyen etkiye, mantıksal yaklaşım ise belirli olan etkiye izin verir.

Birçok Yapay öğrenme yöntemi, veri madenciliği, desen tanıma ve istatistlikten, daha önce denenen ve test edilmiş olan teknikleri baz almıştır: Sınıflandırmak, kümelemek vb. [7].

2.3. Veri ambarları

Veri ambarı, yapay öğrenme ile eş olarak bahsedilen ve yapay öğrenme sürecinin gerçekleştiği veriyi temin eden özel bir veri tabanıdır. Terim anlamı olarak ise, çok çeşitli kaynaklardan ve çoğunlukla değişik yapıdaki verinin toplandığı ve hepsini aynı çatı altında kullanmayı hedef eden yapılardır. Bu şekildeki bir yapının birden fazla veri tabanından gelen verileri birleştirdiğini söyliyebiliriz [7].

Veri tabanında geliştirilen alakalı bir alan da, yapılan işlemlerin verisini toplamak ve onların analizini sağlamak için kullanılan veri ambarıdır. VTBK için verileri kümelemek yine veri ambarlarının görevlerindendir. Bu iş için iki temel faydası vardır. Ve bu faydaların sayesinde modellemenin başarısında büyük rol oynar.

• Veri temizleme

• Veri erişimi

2.3.1. Veri temizleme

Şirketler, sahibi oldukları büyük veri ve veri tabanlarındaki kayıtlarının her zaman bellli bir disiplin altında tutmaları gerekir. Herhangi bir metotla bir düzen altında olmayan verilerden yararlanmak son derece zordur. Bu sebepen dolayı bu işi yapmak için birçok çözüm bulunsa da, tasarımın öncesinde alınacak önlemler en etkilisi olacaktır.

2.3.2. Veri erişimi

Genel olarak veriye erişim için vede erişilmesi zor olan verilere ulaşabilmek için erişim kolaylığı sağlayacak temiz ve düzgün çalışan yöntemler kullanılmalıdır. İlk önce şirketler vede kişilerin böyle bir probleminin olmaması gerekmektedir. Bundan sonraki adım 'bu veri ile neler yapılacak?' sorusunun cevabıdır. Cevap ise VTBK'nin aşamaları olarak çıkar karşımıza.

Veri ambarlarını analiz etmenin en genel yöntemlerinden birisi Online Analytical Processing (OLAP)'tır. OLAP'ın bize sunduğu imkanlar SQL'e göre biraz daha üstündür. Şöyle sıralayabiriz: birçok boyutta hesaplama yapabilmasi, interaktif veri analizine izin vermesi, çok boyutlu verileri analiz etmesi sayılabilir. Fakat VTBK'nin araçları amacı , süreci mümkün olabildiğince otomatikleştirir.

3. TIP ALANINDA YAPAY ÖĞRENME

3.1. Tıp alanında yapay öğrenme uygulamaları

Sağlık kuruluşlarında oluşan verilerin, elektronik ortamda tutulmaya başlanması ve veriye erişimin kolaylaşması ile bu verilerin birçok sağlık probleminin bilinmeyenleri için cevap arama çalışmalarında kullanılması gündeme gelmiştir.

Hastalıkların sebepleri, risk faktörleri, tedavi yöntemleri, kullanılan ilaçlar ve etkileri, hasta laboratuvar ve demografik verileri bir araya geldiğinde, içlerinden faydalı bilgi elde etmek zorlaşmakta ve bu büyük veri içerisinde birçok değerli bilgi kaybolmaktadır. Biriken bu büyük

veriden hayati önem taşıyan bilgiler ancak veri madenciliği yöntemlerinin yetenekleri kullanılarak elde edilebilir.

Tıp alanındaki ilk yapay öğrenme uygulaması 1854 yılında John Snow tarafından kağıt kalem ile yapılmıştır. Londra'da başlayan kolera salgınında, bir harita üzerinde ölenlerin konumlarını işaretleyerek ilk kümeleme çalışmasını yapmış, bu sayede ölümlerin belli bölgelerde toplandığını fark etmiş ve su pompalarından kaynaklanan salgının giderilmesini sağlamıştır [8].

Delen ve diğerleri, göğüs kanseri hastalarının verilerine yapay öğrenme teknikleri uygulayarak, hastaların ölüm ve hayatta kalma ihtimalini tespit eden bir model geliştirmişlerdir [9].

Özekes 2006, doktora tezi çalışmasında ileri örüntü tanıma ve görüntü işleme yöntemlerini kullanarak mamografi görüntülerindeki kitlelerin ve akciğer BT görüntülerindeki nodüllerin tespit edilmesine yardımcı olan bilgisayar destekli tespit yazılımları ve teknikler araştırılmıştır [10].

Demirel 2008 yüksek lisans tezinde, meme kanseri hastalarının verisi üzerinde veri madenciliği teknikleri uygulayarak bir model elde etmiş ve onkoloğa meme kanseri hastalarına uygulanması gereken tedavi yöntemleri konusunda yardımcı olacak bir yazılım geliştirmiştir [11].

Patil ve diğerleri 2011 tarafından yazılmış olan makalede 180 yanık hastasının 2002 – 2006 yılları arasındaki verileri üzerinde sınıflandırma teknikleri uygulanarak, hastaların hayatta kalma durumuna ilişkin yüksek oranlarda tahminleme yapabilen model oluşturulmuştur [12].

Doğan 2007 yüksek lisans tezi çalışmasında, biyokimya verilerinden seçilen parametreleri temel alarak, kalp krizi (miyokard enfarktüsü), hiperlipidemi, demir eksikliği anemisi ve hipertiroidi-hipotiroidi hastalıklarının teşhisinde kullanılacak bir karar destek sistemi tasarlanmasını sağlamıştır [13].

Tahminciler 2014 yüksek lisans tezinde, bir internet sitesinde yapılan yorumları çeşitli araçlar ile okuyarak bir veri tabanı oluşturmuş, kural tabanlı metin ayrıştırma, eşleştirme ve

çeşitli yapay öğrenme uygulayarak Erythromycin ilacı yan etkileri üzerine bir araştırma yapmıştır [14].

Kumar ve diğerleri 2011 yaptıkları araştırmada toplanan verilerden desen (pattern) çıkararak diyabet, hepatit ve kalp hastalıklarında hekime yardımcı olacak akıllı tıbbi karar destek sistemleri geliştirilmiş ve bazı algoritmaların etkinlikleri karşılaştırılmıştır [15].

Sağlık hizmeti veren kurumların yönetimsel faaliyetleri açısından yapay öğrenme yöntemleri uygulanarak çeşitli kazanımlar sağlanmıştır.

İngiltere'de St. George Hastanesi'nde yapılan bir veri madenciliği çalışması ile yoğun bakım ünitesinden çıktıktan sonra yaşamını yitiren hastaların, yüzde otuz dokuzunun 48 saat daha yoğun bakımda tutularak ölüm riskinin ortadan kaldırılabileceği tespit edildi [16].

San Francisco Kalp Enstitüsü'nde mali yapının güçlenmesi, hasta bakım kalitesinin arttırılması, hastaların hastanede kalış sürelerinin kısaltılması, çalışan performanslarının arttırılması amacıyla yapay öğrenme çalışmaları başlatılmış ve sonucunda oluşan modeller amaçların gerçekleştirilmesini sağlamıştır [16].

Boehringer Ingelheim İtalya dünyadaki önemli ilaç şirketlerinden biridir. Eczanelere yönelik farklılaştırılmış satış politikaları üretmek ve en iyi müşterilerini belirleyebilmek için bir yapay öğrenme çalışması başlatmıştır. Oluşan sınıflandırma modeli sayesinde karlı müşterilerin izlenmesi, en karlı eczaneler hedef listesinin oluşturulması, müşteri ilişkilerinin doğru yönetilmesi ve pazarlama faaliyetlerinin etkinliğinin değerlendirilmesi gibi kazanımlar elde edilmiştir [16].

4. KALP VE DAMAR HASTALIKLARI

4.1. Kalp ve damar hastalıkları tanımı ve önemi

Kalp ve vücutta bulunan kan damarları sistemini olumsuz etkileyebilen hastalıkları kapsayan genel bir terimdir. Ölüm vakalarının birçoğunun sebebini kalp hastalıkları oluşturur. kalbin görevleriniden biri doku ve organlar için oksijen iletmektir. Aynı şekilde normal bir damarın iç

yüzeyi pürüzsüz bir yapıdadır. Bunun dışında kötü beslenen, sigara tüketen, tansiyonu olan, alkol kullanan kişilerde zaman geçtikçe damarların içinde bozulmalar olur.

Bu şekilde bozulmaya uğrayan yerlerde zamana bağlı olarak plak şeklinde tabakalar oluşur. Bunların en büyük zararlarında bazıları, kan akışını engellemesi ve damarların elastikiyetini değiştirmesidir. Aynı faktörleri yaşlı insanlarda da görmek mümkündür. Bu hassasiyete engel olabilmek çok büyük önem taşır. Damar sağlığını korumanın çeşitli yolları vardır. Bunlar, kollestrol seviyesini belli bir düzeyde tutmak, yüksek kan şekerinden kaçınmak, kanın basıncını da belli bir düzeyde tutmak, ürik asit seviyesini kontrol altında tutmak gibi kişilerin kendilerince alabileceği önlemlerdir. Bu önlemleri almak için, insanların düzenli aralıklar ile doktora gidip kan vermeleri ve yukarıda bahsettiğimiz değerleri ölçtürmesi gerekir. Sonuçları yorumlayan doktorlara göre çeşitli tedeviler olabilir.

TÜİK verilerine göre 2014 yılında, ölümlerin yüzde 39,8'inin sebebi kalp damar hastalıklarıdır. Diğer sebeplerin çok üzerindedir. Şekil 3'te görüldüğü gibi kadınlarda bu oran 39.6'ya çıkmaktadır [17].

	2013		2014	
	SAYI	(%)	SAYI	(%)
Toplam	**360.873**	**100,0**	**375.291**	**100,0**
Dolaşım sistemi hastalıkları	143.084	39,6	151.696	14,46
İyi ve kötü huylu tümörler	56.534	21,4	77.587	20,7
Solunum sistemi hastalıkları	36.364	9,8	40.258	10,7
Endokrin ve metebolizma hastalıkları	20.096	5,6	19.288	5,1
Dışsal yaralanma ve zehirlenmeler	20.409	5,7	16.018	4,3
Sinir sistemi hastalıkları	14.708	4,1	16.517	4,4
Diğer	50.679	14,4	53.927	14,0
Toplam	998	908	90	

Şekil 4: Ölümlerin hastalıklara göre dağılımı [17].

4.2. Kalp ve damar hastalık çeşitleri

Kalp ve damar hastalıkları WHO tarafından aşağıdaki gibi açıklanmıştır;

Koroner arter hastalığı (coronary artery disease); kalp kasını besleyen kan damarları hastalığıdır ve bu hastalığın sonucunda koroner kalp hastalığı (kalp krizi - coronary heart disease) ortaya çıkar.

Serebrovasküler hastalık (cerebrovascular disease); beyni besleyen kan damarları hastalığıdır, felç olarak bilinir.

Hipertansiyon (Hypertension); kan basıncının referans değerlere göre yüksek olmasıdır (high blood pressure).

Periferik arter hastalığı (peripheral arterial disease); kolları ve bacakları besleyen kan damarları hastalığıdır.

Romatizmal kalp hastalığı (rheumatic heart disease); kalp kası ve kalp kapakçıklarının streptokok bakterinin sebep olduğu romatizmal ateş sebebiyle zarar görmesidir.

Konjenital kalp hastalığı (congenital heart disease); doğuştan kalp yapısında bozukluk olmasıdır.

Derin ven trombozu (DVT) ve akciğer emboli (deep vein thrombosis and pulmonary embolism); DVT vücudumuzdaki derin bir vende (toplardamar) kan pıhtısı oluşmasıdır. Genelde alt bacak ya da uylukta ortaya çıkar. Bacak damarlarındaki kan pıhtıları yerinden çıkıp akciğerlere ya da kalbe gitmesi ile oluşur.

Kalp krizi ve felç (heart attack and stroke), kan pıhtısı sebebiyle kalp ya da beyne kan akışının engellenmesi ile oluşan, akut (hızlı başlayan ve / veya kısa süreli) hastalıklardır.

Bunun en yaygın nedeni kalp ve beyni besleyen kan damarlarının iç duvarlarında yağ birikintilerinin oluşmasıdır. Felç ise beyinde bulunan bir damarda kanama ya da bir pıhtı olması sebebiyle oluşabilir.

4.3. Kalp ve damar hastalık önemli risk faktörleri

Kalp damar hastalıkları ölümleri incelendiğinde, daha önce teşhis konulmamış ya da bir belirti (symptom) olmayan hastaların ölümlerin yarısından fazlasını oluşturduğu görülmektedir.

Normal bir damar insan vücudunun organlarını vede dokularına oksijen taşıyarak onları besler. Bebeklik döneminden başlayarak, yaşlı kişilerde daha fazla oranda damarın yapasında değişmeler olur. Bu değişikliklerin birçoğu damar duvarında plak denilen tabakalar oluşturur. Bu tabakalar zaman içerisinde insanda kalp ve damar hastalığı oluşmasına sebebiyet verebilir. Beslenmemizde yaptığımız değişiklikler yani daha az sağlıksız gıda tüketmek çözümlerden biri olabilir. Sağlıklı bir sekilde yaşamayı seçersek, daha iyi bir yaşlılık geçiririz.

Bu nedenle kalp damar hastalıkları risk faktörlerinin belirlenmesi büyük önem kazanmaktadır. Kalp damar hastalıkları risk faktörlerinin azaltılması hem kalp damar hastalıkları hem de bu sebeple ölümlerin azalmasını sağlamaktadır [18].

4.3.1. Değiştirilemeyen Risk Faktörleri

a. Yaş: Yaşın ilerlemesiyle kalp damar hastalıkları artış göstermektedir. Erkeklerde 45 yaş ve üzeri, kadınlarda ise 55 yaş ve üzeri riskli olarak kabul edilmektedir [19].

b. Aile öyküsü: Birinci derece akrabalarda kadınlarda 65 erkeklerde 55 yaş öncesi kalp damar hastalıkları tanısı konulmuş olması, gelecekteki kalp damar hastalıkları için öngörücü olarak kabul edilmektedir. Fakat günümüzde kalp damar hastalıkları yatkınlığı için klinik pratikte kullanımı kabul edilmiş bir tarama testi yoktur [21].

c. Cinsiyet: Genç erkeklerde kalp damar hastalıkları ve bu sebeple ölüm oranı kadınlardan 4-5 kat fazladır. 65 yaş altında erkeklerin felç olma riski kadınlara göre 2 kat daha fazladır [20].

4.3.2. Değiştirilemeyen Risk Faktörleri

a. Sigara: Kalp damar hastalıkları için ana risk faktörlerinden biri sigara kullanımıdır ve kalp damar hastalıkları saptanmış kişilerin sigara içmeye devam etmesi ölüm oranlarını arttırmaktadır [22].

b. Hipertansiyon (yüksek tansiyon): Dünya genelinde oldukça yaygın bir sağlık problemidir. Kan basıncının 140/90 mmHg eşit veya yüksek olması hipertansiyon göstergesidir. Hipertansiyonun neden olduğu hastalıklar içerisinden en sık rastlananlar ise kalp damar hastalıkları ve felçtir. Hipertansiyon tedavisi ile kallp ve damar hastalığı riski azalmaktadır [22].

c. Diyabet (şeker hastalığı): Kalp damar hastalıkları başlıca risk faktörlerinden biri de diyabettir. Diyabet hastalıklarında en önemli ölüm sebebi kalp damar hastalıkları [22].

d. Hiperlipidemi (kan yağlarının yüksekliği) : Kan yağlarından en az birinin artmasıdır. Trigliserid ve kolesterol olmak üzere iki çeşit kan yağı vardır. Yapılan tüm araştırmalarda hiperlipidemi ile kalp damar hastalıkları arasında bir ilişki çıkması sonucu önemli ve düzeltilebilir kalp damar hastalıkları faktörlerinden biri olarak kabul edilen hiperlipidemi diyet, egzersiz ve ilaç tedavisi ile kontrol altına alınabilir [22].

e. Obezite: Yağ kitlesinin yağsız vücut kitlesine göre yüksek olmasına obezite denir. Beden Kitle İndeksi (BKİ – Body Mass Index - BMI) obezite tespiti için kullanılan en bilinen yöntemdir.

BKİ değerinin 30 üzerinde olması kalp damar hastalığı riskini artırır. BKİ hesaplaması aşağıdaki gibi yapılmaktadır [22].

$$BKİ = Kilo\ (kg) / Boy^2\ (m)$$

Kalp ve damar hastalarında yukarıda belirtilen başlıca risk faktörlerinden birkaçının aynı anda bulunması dikkat çekmektedir. Bahsedilen risk faktörlerinden üç ya da daha fazlasının bir

hastada şans eseri bir arada bulunma ihtimali, üç ya da daha fazla risk faktörüne sahip kalp damar hastalıkları tanısı konmuş hastalar ile karşılaştırıldığında, ihtimale göre dört kat daha fazla gerçekleştiği görülmektedir [22].

4.4. Tanı yöntemleri

Kalp ve damar hastalıkları açısından risk taşıyan hastalara kalp ve damar hastalıkları tanısı konulabilmesi için aşağıdaki yöntemler uygulanır [23]:

a. Elektrokardiyografi (EKG)

b. Ekokardiyografi (EKO)

c. Egzersiz stres testleri – Treadmill

d. Myokard perfüzyon sintigrafisi

e. Positron yayınlayıcı tomografi (PET)

f. Koroner anjiyo CT (Çok Kesitli BT Anjiyografi) (Multislice Kardiyak BT)

g. Kardiyak MR

4.5. Risk hesaplama yöntemleri

Kalp ve damar hastalıkları riskinin tahmin edilebilmesi, risk faktörlerini ortadan kaldırabilmek, önlem alabilmek ve tedaviyi sağlayabilmek için çok önemlidir. Tahmin için ise kalp ve damar hastalıkları tanısı konmuş hastalardaki risk faktörlerinin bir arada bulunduğu gerçeği yol göstermektedir.

Günümüzde kalp ve damar hastalıkları belirlemek için bazı risk hesaplama yöntemleri geliştirilmiştir. Bu yöntemler aşağıda belirtilmiştir [24].

Framingham risk hesaplama sistemi: Amerika'nın Massachusetts eyaletine bağlı olan Framingham kasabasında gerçekleştirilen bir izlem çalışmasına dayanmaktadır.

Kasabada yaşayan 5209 erişkin ile 1948 yılında başlanmıştır, şu anda üçüncü kuşak izlenmektedir. Amerikan Kalp Birliği (AKB) bu veriler üzerinden bir risk değerlendirme sistemi geliştirmiştir. Geliştirilen sistemde belirtilen risk faktörleri ile 10 yıl içindeki kalp ve damar hastalıkları riski hesaplanır.

SCORE çalışması: Framingham çalışmasının bölgesel bir uygulama olması sebebiyle Avrupa Kardiyoloji Derneği 205178 katılımcıdan elde ettikleri verileri kullanarak SCORE çalışmasını yapmıştır. Bu çalışmada kalp ve damar hastalıkların on yıllık gelişim riski hesaplanır. Bulunan risk faktörleri ile yapılan risk hesaplamasında düşük, orta, yüksek ve çok yüksek olmak üzere dört risk sonucundan birine ulaşılır.

TEKHARF çalışması: 1990 yılında Türk halkının sağlık niteliği hakkında bilgi elde edebilmek amacıyla Türk Erişkinlerinde Kalp Hastalığı ve Risk Faktörleri (TEKHARF) çalışması başlatıldı. Çalışmada, Türkiye'nin yedi coğrafi bölgesinden ve 59 farklı yerleşim biriminden 20 yaş üzeri 3687 kişi düzenli olarak tarandı.

Sonuçta birçok istatistiki bilgi ile Türk toplumunun metabolik sendroma çok yatkın olduğu ortaya konulmuştur [24].

5. VERİ SETİ

Veri, bilginin yapılandırılıp kayıt altına alınıp, kolay analiz edilebilmesi için bir araya getirilmesine denir. Bir veya birden fazla bilgiden oluşan kümedir. Veri genellikle araştırma, gözlem, deney, sayım, ölçüm yoluyla elde edilir. Yaş, isim, telefon no, herhangi bir işlemin sonucunu ya da sınıftaki öğrencilerin yaşlarının ortalamaları birer veridir.

Ham veri etkin şekilde bilgi üretme ve analiz için önemli bir hammade olarak görülebilir. Örneğin, anketler aracılığı ile oluşturulan veriler (seçim verileri), bir oylama yapıldığında (seçim sonuçları verileri), bir kayıt yapıldığında (doğum kayıtları verisi), bir şey satın alındığında (çevrim içi satış kayıtları vb.). gibi. Veri ayrıca cep telefonları, İnternet, uydu (GPS verisi gibi) ve birçok farklı teknolojiler tarafından da oluşturulabiliyor [25].

Gündelik hayatımızda veriyi sıklıkla tablolarda düzenlenmiş buluruz.Tek bir tablonun içeriği veri seti olarak ifade edilir. Veri setini analiz ederek ondan yeni bilgi -görsel çalışmalar üretmek; karar alma, politika üretme süreci için önemlidir.

Bir veri seti, tek tek, birlikte veya bütün bir varlık olarak yönetilebilen ilişkili, ayrı ayrı, ilgili verilerin bir toplamıdır. Veri seti, bazı veri yapısı türünde düzenlenir. Örneğin bir veri tabanında bir veri seti, işletme verilerinin bir koleksiyonunu (adlar, maaşlar, iletişim bilgileri, satış rakamları vb.) içerebilir. Veritabanının kendisi, belirli bir kurumsal departmanın satış verileri gibi belirli bir bilgi türüne ilişkin verileri, bir veri seti olarak kabul edilebilir [25].

Önemsiz objelerde bile onlarla ilişkili çok sayıda veri bulunur. Aşağıdakiler bunların başlıcalarıdır:

Nitel veri (Qualitative data): birimler ya da ölçüler ile elde edilmiş değerlerin olmadığı, yapısal ve kurumsal özelliklerin taşındığı verilerdir. Kişilerin cinsiyetleri, insanların saçların renkleri nitel özelliklerdendir. Yapılan bir deneyin niteliklerini belirten verileri de gösterir. İki farklı grubu vardır.

Sınıflanabilen nitel veri: isimler, kodlar ve numaralar göstermek için sınıf özelinde ayırım yapan verilerdir. Birbirlerinden bağımsızdırlar. Havadaki ve denizdeki taşıtlar verilebilir.

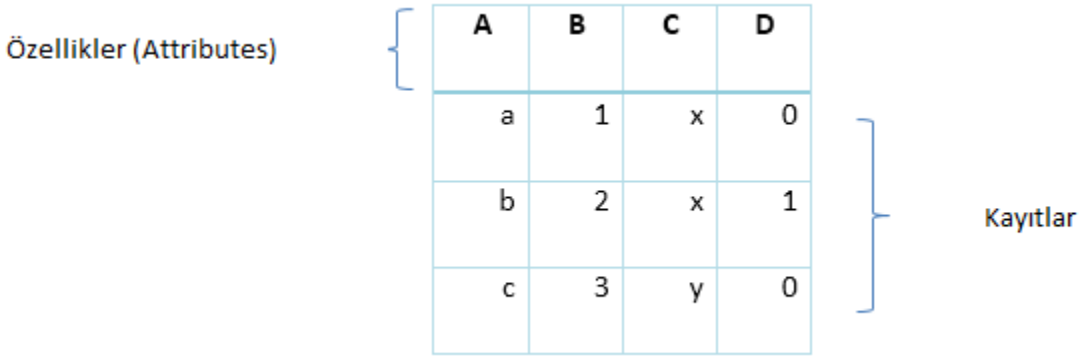
- **Sıralanabilen nitel veri:** miktarı olmayan değerlerin gösterilmesi, bir derece ile basamakları olan değerlerin, sıranın önemli olduğu değişkenlerin verileridirler. Ordudaki rütbeler, öğrencinin başarı durumları örnek gösterilebilir.

- **Nicel veri (Quantitative data):** yapılan deneylerin sayılabilen ve ölçülebilen özelliklerini gösteren, aralıkları olan veya orantıları olan veridirler. Sürekli nicel veri ve kesikli nicel veri olmak üzere iki türü vardır. Örneğin golf toplarının sayısı, ölçüsü, fiyatı, bir testteki skor [25].

Sürekli nicel veri: virgüllü sayıların değerlerini alan nicel veridir. Boyun uzunluğu.

- **Kesikli nicel veri:** Tamsayılardan oluşan değerleri alan nicel veridir. Aile nüfusu, şehir nüfusu gibi.

Kategorik veri (Categorical data): tanımladığınız veriyi bir kategoriye koyar [25].



Şekil 5: Veri seti yapısı.

Örneğin bir veri tabanında bir veri seti, işletme verilerinin bir koleksiyonunu (adlar, maaşlar, iletişim bilgileri, satış rakamları vb.) içerebilir. Veritabanının kendisi, belirli bir kurumsal departmanın satış verileri gibi belirli bir bilgi türüne ilişkin verileri, bir veri seti olarak kabul edilebilir. Aşağıdaki Şekil 5'te veri setinin yapısının ayrıntıları görünmektedir.

5.1. Hastalar adındaki excel dosyası

Bu çalışmada incelenek olan veri seti Hastalar.xlsx excel dosyasıdır. Dosya uzantıları kullanılacak programlara göre çok çeşitli formatlarda olabilir. İsminden anlaşılacağı üzere hastaların kayıtlarından oluşan bir veri setidir.

Hastaların hepsi kalp hastasıdır. Hastaların ilk olarak kendisi ile ilgili faktörler bulunur. Bunlar hastanın yaşı, cinsiyeti gibi özelliklerdir. Daha sonra hastanın kalbi ile ilgili faktörler; daha önceden yaptırmış olduğu bir takım testlerden oluşur. Ve en son olarak eğer hasta kalp ameliyatı geçirmiş ise, onunla ilgili kayıtlar.

Bunlardan 10 özellik hasta ile ilgili, 2 tanesi hastanın kalbi ile ilgili faktörler ve 2 tanesi de eğer hasta kalp ameliyatı olmuş ise, onun ile ilgili kayıtlardır.

Bu şekilde 3500 tane kayıt vardır.Şekil 6'da bu excel veri dosyasından bir görüntü verilmiştir.

Hasta_Yaş_Aralığı	Hasta_Cinsiyet	Kronik_Akçiğer_Hastalığı	Ekstrakardiyak_Arteriopati
66 - 70 Yaş	Erkek	%80 Altı %70 Üstü	Yok
66 - 70 Yaş	Erkek	%80 Altı %70 Üstü	Yok
Bilinmiyor	Erkek	%80 Altı %70 Üstü	Var
0 - 59 Yaş	Erkek	%80 Altı %70 Üstü	Var
Bilinmiyor	Kadın	%80 Altı %70 Üstü	Yok
60 - 65 Yaş	Kadın	%70 Altında	Var
Bilinmiyor	Erkek	%80 Altı %70 Üstü	Yok
Bilinmiyor	Erkek	%80 Altı %70 Üstü	Yok
Bilinmiyor	Erkek	%80 Altı %70 Üstü	Var
71 - 99 Yaş	Erkek	%80 Altı %70 Üstü	Yok
60 - 65 Yaş	Erkek	%80 Altı %70 Üstü	Yok
0 - 59 Yaş	Erkek	%80 Altı %70 Üstü	Yok
0 - 59 Yaş	Erkek	%80 Altı %70 Üstü	Yok
0 - 59 Yaş	Kadın	%80 Altı %70 Üstü	Yok

Şekil 6: Hastalar adındaki excel dosyası.

Hasta ile ilgili faktörler :

1. hasta_yas
2. hasta_cinsiyet
3. kronik_akciger_hastaligi
4. extrakardiyak_artriopati
5. gecirilmis_kardiyak_operasyon
6. bobrek_fonksiyon_bozuklugu
7. bobrek_yetmezligi
8. aktif_endokardit
9. kritik_preoperatif_durum
10. diabetes_mellitus

Hastanın kalbi ile ilgili faktörler :

1. lv_disfonksiyonu
2. pulmoner_hipertansiyon

Hastanın var ise ameliyatı ile ilgili faktörler :

1. torasik_aorta_cerrahisi
2. post_mi_vsd

5.2. Hastalar veri setinin hazırlanması

Hedef veri setlerindeki bilginin yapay öğrenme teknikleri kullanılarak analiz edilmesidir. Bunu iyi bir şekilde gerçekleştirebilmek için veri setinde bazı değişiklikler yapmamız gerekebilir.

Veri hazırlama süreci bitirldikten sonra doğruluk oranlarında pazitif yönde bir değişim olması beklenir. Ya da en azından aynı oranda olması gerekmektedir. Veriyi hazırlayan kişi bu sayede,

yapay öğrenme süreci içinde daha hızlı ve daha verimli modeller yaparak, doğru öngörü şansını artıtrır.

Modelleme yapılırken kullanılacak olan programlara göre, veriyi belli bir formata sokmak gerekir. Uygulanacak olan metotlara görede yeteri kadar verimiz olması gerekir.

Kağıt üzerinde herşey çok iyi gibi olsada iş pratiğe döküldüğü zaman verinin durumu farklı oluyor. Örnek olarak; eksik veri, tutarsız veri, gürültülü veri verilebilinir.

Hastalar verisetinede gerekli önişlemler yapıldıktan sonra, testte kullanılacak programlar (Veka, Matlab.) için hazır hale getirilir. Bunun için daha çok istatistiksel problemlerinin çözümünde yer alan SPSS programından yardım alınmıştır. Bu program ile veri settindeki özelliklere değer verip, kategorik olarak değerlerinin gözükmesini sağlanır.

Örneğin; hastanın yaşı özelliğini

- erkekse = 1
- kadınsa = 0 ,

aynı şekilde diabeti var ise 1 yok ise 0 gibi. Bu işlemlerden sonra verisetinin görünümü aşağıdaki Şekil 8'deki gibidir.

HASTA YAS	HASTA CİNSİYET	KRONİK AKCİHER HASTALIGI	EKSTRAKARDIYAK ARTERIOPATI	GECİRİLMIS KARDIYAK OPERASYON
3	0	1	0	0
0	1	0	1	0
2	0	1	0	1
1	0	0	1	0
2	1	1	0	0
1	1	1	0	0
2	1	0	1	0
3	1	0	0	0
1	1	0	0	0
0	1	0	0	0
0	1	0	0	1
0	0	1	0	0
2	1	1	0	0
3	1	1	0	0
3	1	1	0	0
2	0	0	0	0
3	0	0	1	0

Şekil 8: Ön işlemden geçmiş veri seti.

5.3. Hastalar veri setinin kullanılacak programlara göre formatlanması

Veri setini sınıflandırma algoritmalarını uygulamadan önce hangi programlar kullanılacak ise o programın dosya formatına çevirmek gerekir. Bu çalışmada Weka ve Matlab programları kullanılacağı için, veri setinde gereken değişiklikler yapılır.

5.3.1. Weka programı için değişiklik

Bunun için Weka'nın dosya formatı olan .arff uzantılı dosyaya çevirmemiz gerekiyor. Hastalar veri seti excel dosya formatındaki bi dosyaydı. Bu excel dosyasını Weka dosya formatına çevirmek için şu değişiklikler yapılır:

- İlk olarak excel dosyası .csv olarak farklı kaydedilir.
- Sonrasında veriyi ayıran noktalı virgüller, virgül ile yer değiştirilir.
- Dosyanın başına Şekil 8'deki script yazılarak .arff uzantısı ile kaydedillir.

```
@RELATION Hastalar

@ATTRIBUTE hasta_yas {0,1,2,3}
@ATTRIBUTE hasta_cinsiyet       {0,1}
@ATTRIBUTE kronik_akciger_hastaligi        {0,1}
@ATTRIBUTE extrakardiyak_artriopati  {0,1}
@ATTRIBUTE gecirilmis_kardiyak_operasyon   {0,1}
@ATTRIBUTE bobrek_fonksiyon_bozuklugu   {0,1}
@ATTRIBUTE bobrek_yetmezligi    {0,1}
@ATTRIBUTE aktif_endokardit     {0,1}
@ATTRIBUTE kritik_preoperatif_durum  {0,1}
@ATTRIBUTE diabetes_mellitus    {0,1}
@ATTRIBUTE lv_disfonksiyonu     {0,1,2}
@ATTRIBUTE pulmoner_hipertansiyon    {0,1}
@ATTRIBUTE torasik_aorta_cerrahisi   {0,1}
@ATTRIBUTE post_mi_vsd    {0,1}
@ATTRIBUTE class     {'Dusuk Risk','Orta Risk','Yuksek Risk'}

@DATA
3,0,1,0,0,1,1,1,1,1,2,1,1,1,'Yuksek Risk'
0,1,0,1,0,0,0,0,0,0,2,0,0,0,'Dusuk Risk'
2,0,1,0,1,0,0,0,0,0,2,0,0,0,'Yuksek Risk'
```

Şekil 8: Hastalar adındaki Weka dosyası.

5.3.1. Matlab programı için değişiklik

Matlab proramı için herhangi bir script yazmamıza gerek yoktur. Bunun için Matlab'ı açıp, bir veri seti oluşturulur. Ve Hastalar.mat olarak kaydedilir. Sonrasında excel dosyasındaki veri kopyala yapıştır ile içine taşınır.

6. SINIFLANDIRMA

Sınıflandırma algoritmaları denetimli / eğiticili (supervised) algoritmalardır ve sonuç olarak bir tahmin yapılır. Sınıflandırma işlemi öğrenme ve sınıflandırma olmak üzere iki aşamadan oluşur.

a. Öğrenme: Verinin tamamına ait örnekler bulunacak şekilde basitleştirilmiş bir kısmi veri kümesi oluşturulur (eğitim kümesi – training set) ve bu veri sınıflandırma algoritması ile analiz edilir.

Algoritma uygulanan öğrenme verisinde bulunan her kayıt sınıf bilgisi ile birlikte birçok başka nitelikten oluşur. Algoritma sınıf bilgisi ve diğer nitelikler arasındaki ilişkileri saptar ve tüm veri üzerinde uygulamak üzere bir model oluşturur.

b. Sınıflandırma: Birinci adımda oluşturulan model sınıflandırma için kullanılır. Modelin veri kümesine uygulanması sonucu elde edilen sınıf bilgileri veri kümesinde bulunan sınıf bilgileri ile karşılaştırılarak tahminleme doğruluğu hesaplanır. Eğer doğruluk oranı kabul edilebilir oranda ise model gelecekteki verilerde sınıf bilgisini elde etmek için kullanılabilir [25].

Kullanılan sınıflandırma algoritması tarafından oluşturulan modelin başarısını ölçmek için doğruluk (accuracy), hata oranı (error rate), hassaslık (sensitivity) ve özel etken oranı (specificity) kullanılır.

Doğruluk; doğru sınıflandırılmış kayıt sayısının toplam kayıt sayısına bölünmesi ile elde edilir.

Hata oranı; 1-doğruluk ile hesaplanır. Hassaslık ve özel etken oranları Tablo 1'de görülen karışıklık matrisi üzerinden bulunur.

C1 (Pozitif): Pozitif sınıf etiketi

C2 (Negatif): Negatif sınıf etiketi

Doğru Pozitif (True Positive TP) sınıflandırıcı tarafından doğru etiketlenmiş olan kayıt pozitif etiket sayısını ifade eder.

Doğru Negatif (True Negative TN) ise sınıflandırıcı tarafından doğru etiketleme yapılmış negatif etiket adedini gösterir.

Aynı şekilde Yanlış Pozitif (False Positive FP), yanlış etiketlenen pozitif etiket sayısını, Yanlış Negatif (False Negative FN) ise yanlış etiketlenen negatif etiket adedini ifade eder [25].

		ÖNGÖRÜLEN SINIF		
		C1 (Pozitif)	C2 (Negatif)	
GERÇEK SINIF	C1 (Pozitif)	Doğru Pozitif TP	Yanlış Pozitif FP	Pozitif P
	C2(Negatif)	Yanlış Pozitif FP	Doğru Negatif FT	Negatif N

Tablo 1: Doğruluk Matrisi (Confision Matrix) [25].

Doğru Pozitif (True Positive TP) sınıflandırıcı tarafından doğru etiketlenmiş olan kayıt pozitif etiket sayısını ifade ederken, Doğru Negatif (True Negative TN) ise sınıflandırıcı tarafından doğru etiketleme yapılmış negatif etiket adedini gösterir. Aynı şekilde Yanlış Pozitif (False Positive FP), yanlış etiketlenen pozitif etiket sayısını, Yanlış Negatif (False Negative FN) ise yanlış etiketlenen negatif etiket adedini ifade eder [25].

Hassaslık (sensitivity) = TP / (TP + FN)

Özel etken oranı (specificity) = TN / (FP + TN)

Sınıflandırma yöntemlerini kullanılan algoritmalara göre beşe ayırabiliriz [25];

a. Karar ağaçları (Decision Trees): ID3, C4.5, CART karar ağacı algoritmalarına örnek olarak verilebilir.

b. Bellek tabanlı yöntemler (Instance based): k-en yakın komşu algoritmaları örneklendirilir.

c. Bayes sınıflandırıcı (Bayes Classifier): Naive Bayes en temel algoritmasıdır.

d. Yapay sinir ağları (Artificial Neural Networks)

e. Genetik algoritmalar (Genetic Algorithm)

Sınıflandırmak için, veri setimizde böyle sınıf özelliğinin olması gerekir. Eğer veri setimizde böyle bir özellik yok ise, problemin yapısına göre biz ekleyebiliriz. Hastalar veri setinde de böyle bir yol izlenmiştir.

Hastaların yaşam riski hesaplanacağı için, yani hangi hastaların yüksek, hangi hastaların düşük risk taşıdığını sınıflandırmak için bir sınıf özelliği ekliyoruz. 14 özellik vardı veri setinde ve 15. olarak bu özelliği ekliyoruz. Bu sınıf özelliği kategoriktir. Ve üç farklı kategoridedir. Bunlar düşük risk, orta risk ve yüksek risk.

Her bir özellik karşılığı bir puan verilir, sınıf özelliği hariç diğer 14 özellik için:

- hastanın diyabetinin olması: 2 puan,
- akciyer hastalığının olması: 1 puan,
- böbrek yetmezliği olması: 5 puan,

Şeklinde en büyük sayısı 5 olmak üzere puanlar verilir. Bu puanlar Göztepe Medikalpark Hastanesi uzman doktorları tarafından verilmiş değerlerdir. Son olarak bu puanların toplamı 15. sütun olan sınıf özelliğine yazılır.

- düşük risk: 0-3 puan ,
- orta risk: 4-6 arası ve

- yüksek risk: 7 ve üzeri puan.

Örneğin; bir hastanın kayıtlarına bakılıp puanları toplamı 14 çıktı ise. Bu durumda yüksek riskli olarak sınıflandırılır. Şekil 9'da sınıf özelliği eklendikten sonraki excel dosyası gözükmektedir. Artık her kayıtın bir sınıfı vardır. Yüksek,düşük ve orta olarak sınıflanır.

PULMONER HIPERTANSIYON	TORASIK AORTA CERRAHISI	POST MI VSD	RISK PUANI
1	1	1	Yuksek Risk
0	0	0	Dusuk Risk
0	0	0	Yuksek Risk
0	0	0	Orta Risk
0	0	0	Dusuk Risk
0	0	0	Dusuk Risk

Şekil 9: Sınıf özelliği eklenmiş veri seti.

6.1. Sınıflandırmada kullanılacak algoritmalar

Önerilmiş birçok makine öğrenmesi yöntemi mevcuttur. Bunlar probleme yaklaşımlara göre farklılık gösterebilir ve bu yüzden farklı problemlerde farklı başarılara sahip olabilirler**.**

Her veri seti birbirinden farklı olduğu için, algoritma seçmek önem arz eder. Hastalar veri setinde katagorik özelliklerin fazla olması hatta hepsinin kategorik olması sebebiyle aşağıdaki algoritmalar seçilmiştir:

Yapay sinir ağları (Geri yayılım oalgoritması), karar ağaçları ve rastgele orman.

6.1.1. Yapay sinir ağları (Backpropagation)

Bilgisayar bilimleri ve yazılım alanında yapılan çalışmalar artık makinelerin de insana benzer bir öğrenme yapıp, başka bir durumla karşılaştırıp fikir yürütebilmesine izin verir. Bilgisayarlar en çok matematik ağırlıklı problemlerin çözümü ve formül olarak yazılamayan birçok zor problemin çözülmesinde kullanılır. 1950 senesinde yapılan yapay zeka araştırmaları sayesinde bugün bilgisayarların sezgiselleşen birçok yeteneği olmuştur. Günümüze kadar araştırılarak gelen bu sisyemlere zeki sistemler denir. Zeki sistemlerin içerisinde matematik ve istatisliksel

birçok bilim dalı kullanılmıştır. Buna parelel olarak zeki sistemlerdeki herbir değişim ve gelişme yapay zeka üzerinde etkili olmuştur [26].

Yapay zeka methotları kullanılan alanlara göre birçok farklar içerirler. Bu araştırmada kullanılan yapay sinir ağlarıda bunlardan biridir.

6.1.1.1. Yapay sinir ağları tanımı

İnsan beynindeki düşünme yeteneğini makineler aracılığı ile gerçekleştirmek zor bir süreçtir. YSA'lar bu süreçte insanların gözlem yapmak, öğrenmek ve düşünmek gibi faliyetlerini iyi bir şekilde yerine getirmeye çalışırlar. Bu sebepten dolayı aynı insanlar gibi mekanizmalar kurarak, olayları anlayıp onların üzerinden yorumlar yapabilirler.

Bir insanın sahip olduğu düşünmek, gözlemlemek gibi yeteneklerin çözdüğü problemler ile başa çıkabilmek zor bir iştir. Bunun için insanda da var olan, yaşayıp ya da tecrübe edip öğrenmek gerekir [26].

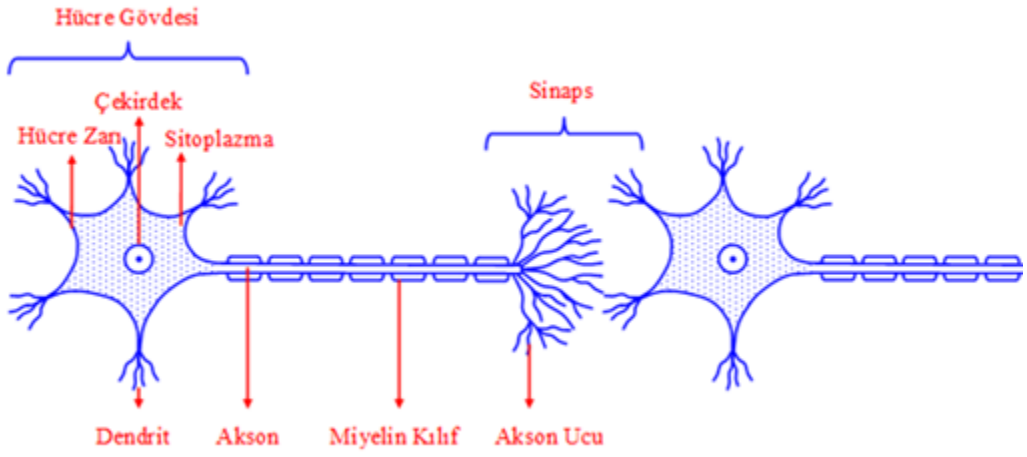
Öğrenebilmek için dışarıdan bir takım bilgiler almak gerekir. Bu bilgilerden ileride yeni ve farklı bilgilerin oluşması gerekmektedir. Problemlerin çözümünde ise bu süreç içerisinde oluşmuş bilgiler kullanılır. Bu işlemi daha iyi yapabilmek için YSA'nın keşifler yapması gerekir. Bu da sadece öğrenmek ile mümkündür. Yapay sinir ağlarının öğrenebilmek için kullanınlan platformlar bilgisayarlara ait sistemlerdir.

YSA'nın kullanmış olduğu algoritmaların gücü, matematik ve istatiksel alanlardan gelir. Haykin [28] YSA'yı şöyle tanımlamıştır: "Bir sinir ağı, basit işlem birimlerinden oluşan, deneyimsel bilgileri biriktirmeye yönelik doğal bir eğilimi olan ve bunların kullanılmasını sağlayan yoğun bir şekilde paralel dağıtılmış bir işlemcidir. Benzerlik gösterdiği noktalar şunlardır:

1. Bilgiyi ağın sayesinde, öğrenmek için bir süreç gerçekleştirir. Ve bunu çevresinden alır.
2. Elde edilen bilgileri biriktirmek için sinaptik ağırlıklar olarak da bilinen nöronlararası bağlantı güçleri kullanılır".

Vural [27] çalışmasında, yapay sinir ağlarını insan beyninin bir işlemi yapabilmek için geçirdiği sürecin bir modelini oluşturmak için kurulan bir sistem olarak anlatmıştır. YSA, yapısı katmanlı bir yapıya aittir. Bu katmanları, sinir hücrelerin bir araya gelerek bağlanmasından oluşmuştur. Aynı insan beyninde olduğu gibi yapay sinir ağları bir öğrenme evresi ile bilgileri toplar, hücrelerin arasındaki bağlantı ağırlıkları ile ise bilgiyi saklarlar. YSA'nın yapısı pareleldir. Bunun yanı sıra öğrenirken amaç edilen sonuçlara ulaşılabilmesi için, yapay sinir ağları ağırlıklarını yenileyen öğrenme algoritmalarına sahiptir.

Yapay sinir ağları tasarlanırken biyolojik sinir ağlarını model olarak kullanılmıştır. İnsanın beynindeki milyarlarca biyolojik sinir hücreleri, birbirleri ile haberleşmektedirler. Bu haberleşmeyi gerçekleştirmek için bağlantılara ihtiyaç vardır. Yapılan bağlantılar ise trilyolarla ifade edilr. Bir biyolojik sinir hücresinin yapısı Şekil 10'da gösterilmiştir:



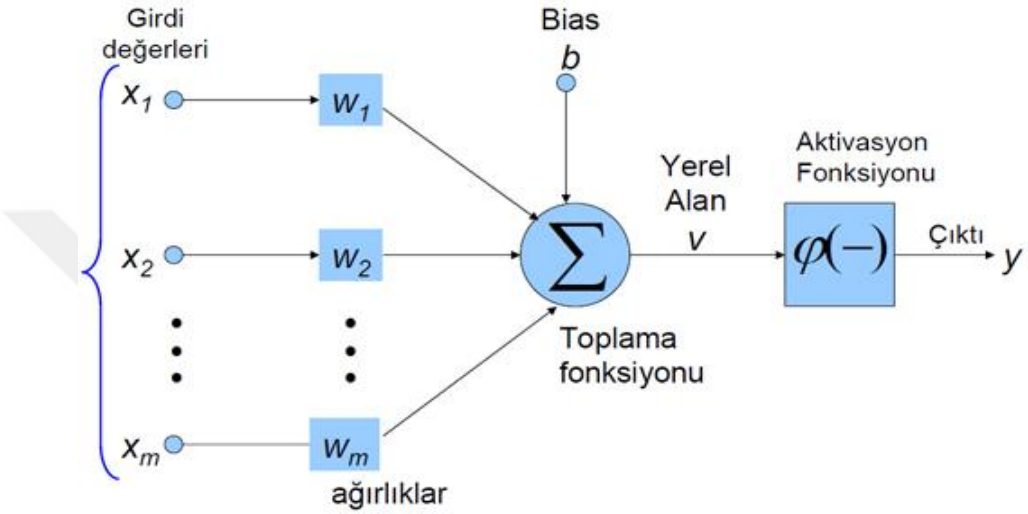
Şekil 10: Biyolojik sinir hücresi yapısı [29].

6.1.1.2. Yapay sinir hücresi

YSA' nın yapay sinir hücreleri (nöronları) bulunmakta ve bu hücreler işlem (process) elemanı olarak adlandırılmaktadır. Genel özellikleri ile bir yapay sinir hücresinin yapısı Şekil 11' de verilmektedir.

Yapay sinir hücresine dışarıdan verilen bilgilere girdi denir. Şekil 11'de girdiler 1 x , 2 x ,… şeklinde gösterilmiştir. Şekil 11'de 1 w , 2 w ,… gösterilen ağırlık değerleri ise; hücreye girdi

olarak verilen bilgilerin önemini ve hücre üzerindeki etkisini gösterir. Toplama fonksiyonu ise hücreye gelen net girdiyi hesaplayan fonksiyondur ve genellikle gelen girdilerin kendi ağırlıklarıyla çarpımlarının toplamıdır. Toplama fonksiyonu Denklem 4.1'de gösterilmektedir [28].



Şekil 11: Yapay sinir ağı hücresi yapısı [29].

Yapay sinir hücresine dışarıdan verilen bilgilere girdi denir. Şekil 10'da girdiler 1 x , 2 x ,… şeklinde gösterilmiştir. Şekil 10'da 1 w , 2 w ,… gösterilen ağırlık değerleri ise; hücreye girdi olarak verilen bilgilerin önemini ve hücre üzerindeki etkisini gösterir.

Toplama fonksiyonu ise hücreye gelen net girdiyi hesaplayan fonksiyondur ve genellikle gelen girdilerin kendi ağırlıklarıyla çarpımlarının toplamıdır. Toplama fonksiyonu Denklem 4.1'de gösterilmektedir [28].

$$NET = \sum_{i}^{n} w_i x_i \qquad (4.1)$$

Burada i x girdileri, i w ise ağırlıkları, n ise bir hücreye gelen toplam girdi sayısını göstermektedir. Literatürde farklı toplama fonksiyonları kullanılmıştır, bir fonksiyonlardan

bazıları Tablo 2'de verilmektedir. Bir problem için en uygun toplama fonksiyonunu belirlemek için bulunmuş bir formül yoktur. Genellikle deneme yanılma yöntemi ile belirlenmektedir [28].

Toplam $\text{Net} = \sum_{i=1}^{N} X_i * W_i$	Ağırlık değerleri girdiler ile çarpılır ve bulunan değerler birbirleriyle toplanarak Net girdi hesaplanır.
Çarpım $\text{Net} = \prod_{i=1}^{N} X_i * W_i$	Ağırlık değerleri girdiler ile çarpılır ve daha sonra bulunan değerler birbirleriyle çarpılarak Net Girdi Hesaplanır.
Maksimum $\text{Net} = \text{Max}(X_i * W_i)$	n adet girdi içinden ağırlıklar girdilerle çarpıldıktan sonra içlerinden en büyüğü Net girdi olarak kabul edilir.
Minimum $\text{Net} = \text{Min}(X_i * W_i)$	n adet girdi içinden ağırlıklar girdilerle çarpıldıktan sonra içlerinden en küçüğü Net girdi olarak kabul edilir.
Çoğunluk $\text{Net} = \sum_{i=1}^{N} Sgn(X_i * W_i)$	n adet girdi içinden girdilerle ağırlıklar çarpıldıktan sonra pozitif ile negatif olanların sayısı bulunur. Büyük olan sayı hücrenin net girdisi olarak kabul edilir.
Kumilatif Toplam $\text{Net} = \text{Net(eski)} + \sum_{i=1}^{N} X_i * W_i$	Hücreye gelen bilgiler ağırlıklı olarak toplanır. Daha önce hücreye gelen bilgilere yeni hesaplanan girdi değerleri eklenerek hücrenin net girdisi hesaplanır.

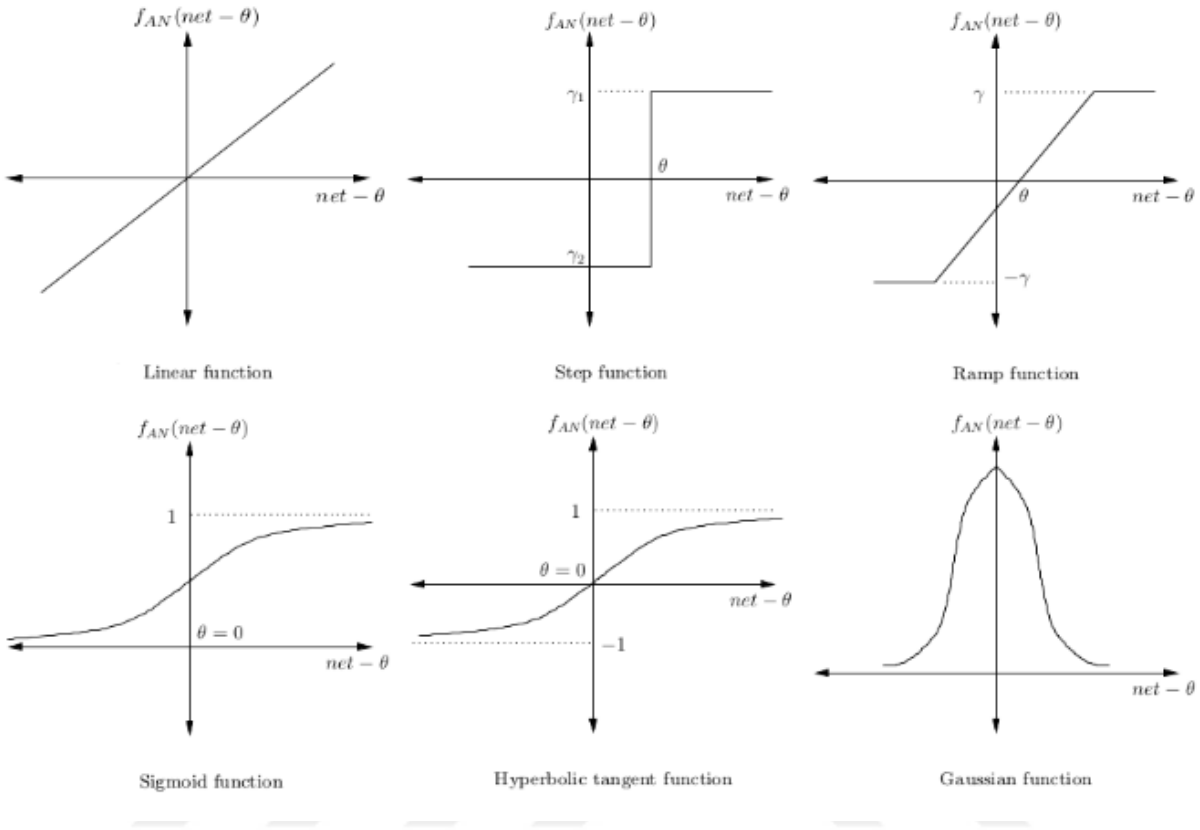
Tablo 2: Girdi değeri ile ağırlıkları çarpıldıktan sonra toplayan fonksiyonlar [29].

Aktivasyon fonksiyonu (transfer fonksiyonu) ise, toplama fonksiyonundan gelen net girdiyi işlemden geçirerek hücrenin çıktısını üreten ve genellikle doğrusal olmayan bir fonksiyondur. Kullanılan hücre modeli çeşidine göre değişik aktivasyon fonksiyonları kullanılmaktadır.

Çalışmamızda kullandığımız çok katmanlı algılayıcılı YSA'larda hiberbolik tanjant fonksiyonu (tansig) aktivasyon fonksiyonu olarak kullanılmıştır.

Tansig fonksiyonu türevi alınabilir, sürekli ve doğrusal olmayan bir fonksiyon olması nedeni ile doğrusal olmayan problemlerin çözümünde yaygın olarak kullanılmaktadır. Fonksiyonun matematiksel tanımı Denklem 4.2'de verilmektedir.

$$f(NET) = \frac{(e^{NET} + e^{-NET})}{(e^{NET} - e^{-NET})} \qquad (4.2)$$



Tablo 3: Aktivasyon fonksiyonları [29].

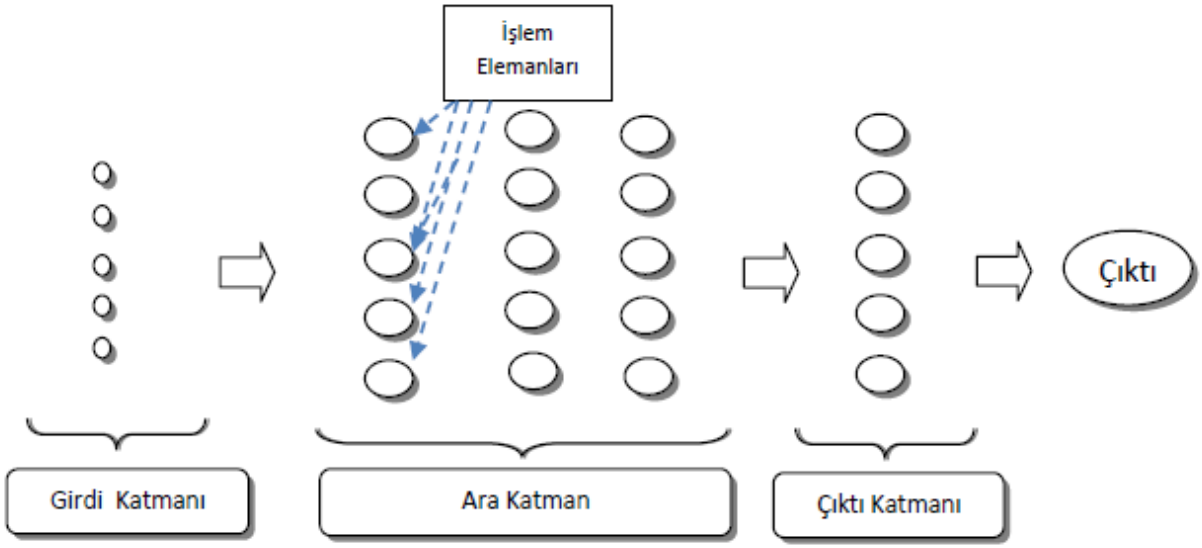
Tablo 3'de aktivasyon fonksiyonlarından adım, eşik, logsig (logaritmik sigmoid) ve tansig (hiperbolik tanjant sigmoid) fonksiyonlarının grafikleri gösterilmektedir. İşlem elemanının çıktısı aktivasyon fonksiyonu kullanılarak hesaplanır. Üretilen bu çıktı dış dünyaya veya diğer bir hücreye gönderilir. Bunun yanında hücre kendi çıktısını kendisine girdi olarak gönderebilir.

6.1.1.3. Yapay sinir ağının yapısı

YSA'nın yapısının parelel olarak çalıştığını ve de birbirleri ile bağlanarak bir araya gelmiş sinir hücreleri oluşturur. Hücrelerin bu şekilde birleşmesinden katmanlar oluşur.

Şekil 12'de bir sinir ağını oluşturan katmanlar gösterilmiştir. Girdi, çıktı ve ara olmak üzere, üç katmandan oluşur. Girdi katmanın görevi, dışarıdan elde edilen bilgileri alıp ve bunları orta katmana doğru iletir. Ara katmanında, girdi katmanından gelen bilgiler alınır ve bazı işlemlerden geçirilir. Bu sürecin sonunda bilgiler çıktı katmanına gönderilir. Çıktı katmasınıda ise ara

katmandan gelen bilgiler bir işlem sürecinden geçerek ağa girmiş olan girdilerden bir çıktı meydana getirir.



Şekil 12: Yapay sinir ağı katmanları [29].

6.1.1.4 Yapay sinir ağ modelleri

Bu araştımada sınıflandırma modelleri kurmak için çok katmanlı algılayıcılar (ÇKA) kullanılmıştır. Bu kısımda ÇKA'nın yapısını ve nasıl modellendiği incelenmiştir. Geçmişteki çalışmalar ışığında yapılan çalışmalarda ilk modeller tek katmanlı algılıyıcılar (TKA) ile, en basit algılayıcıların tasarımlarıydı. Çalışmada kullanılan algoritmanın temelini oluşturan modellerdi [26].

6.1.1.4.1. Tek katmanlı algılayıcılar

Tek katmanlı yapay sinir ağları sadece girdi (x) ve çıktı (Ç) katmanlarından oluşur. Çıktı bütün girdi ünitelerine (x) bağlanmaktadır ve her bağlantının bir ağırlığı (w) vardır. İki girdi ve bir çıktıdan oluşan tek katmanlı bir yapay sinir ağı Şekil 11'de verilmiştir.

Bu ağlarda işlem elemanlarının değerlerinin ve dolayısıyla ağın çıktısının sıfır olmasını önleyen bir eşik değeri (Φ) vardır ve değeri daima 1'dir. Ağın çıktısı ağırlık değerleri ile işleme konulmuş girdi değerlerinin eşik değeri ile toplanması ile bulunmaktadır. Bu girdi bir aktivasyon

fonksiyonundan geçirilerek ağın çıktısı olarak hesaplanmaktadır. Tek katmanlı algılayıcılarda çıktı fonksiyonu doğrusal bir fonksiyondur ve 1 veya -1 değerlerini almaktadır. Sınıflandırma problemlerinin çözümünde çıktı değeri 1 olan birinci grubu, -1 olan ise ikinci grubu göstermektedir [29].Aşağıdaki Şekil 13'te tek katmanlı algılayıcı modeli en yalın bir şekildeki şekli vardır.

Şekil 13: Tek katmanlı algılayıcı modeli [29].

Tek katmanlı algılayıcılarda önemli iki model bulunmaktadır. 1958 yılında Rosenblatt (1958) tarafından geliştirilen basit algılayıcılar (perceptron), diğeri ise 1959 yılında Widrow ve Hoff tarafından geliştirilen ADALINE modelidir.

6.1.1.4.2. Çok katmanlı algılayıcılar

Yapay sinir ağı alanındaki çalışmarı tarihte bir müddet durmuştur. Bunun sebebi Minsk adında bir araştırmacının 1969'da yayınladığı Algılayıcılar adlı kitabıydı. Minsky kitabınna kısaca değinecek olursak: girdiler ile çıktılar arasında bir doğrusallığın olmadığı zaman, çözümü sunucak olan öğrenmenin mümkün olmadığını söylemiştir. Bundan sonra doğrusal olmayan problemleri çözmek için tekrardan YSA kullanılmaya başlanmıştır.

Rumelhart ve arkadaşları tarafından önerilen çok katmanlı algılayıcılar öğrenme algoritması olarak genelde türeve dayalı geri yayılım (back propogiton) veya hata yayma algoritmaları kullanmaktadırlar. Bu nedenle çok katmanlı ağa geri yayılım ağı da denilmektedir.

ÇKA, ADALINE modelinde de kullanılan Delta öğrenme kuralının gelişmiş halini kullanmaktadır. ÇKA ağı öğrenme stratejisi öğretmenli öğrenme yöntemidir. Kurulan ÇKA ağına

eğitim seti içerisinde örnek bilgiler verilirken aynı zamanda bu bilgilerin (girdilerin) karşılığında çıkması beklenen çıktı bilgileri de verilir.

6.1.1.4.2.1. Çok katmanlı algılayıcıların yapısı

Bu çalışmada kullanılan yapıdır. ÇKA'nın yapısını Şekil 14'te görülmektedir. Eğer şekil biraz incelenirse yapının ileriye doğru bağlantılar yaptığı gözlemlenir. Aynı şekilde katmanların oluşturduğu bir yapıyı andırır. Bu katmanlar: girdi, orta ve çıktı katmanları olarak adlandırılır. Tek katmanlı ağlardan ayıran özelliklerden birisi ara katmanın olmasıdır.

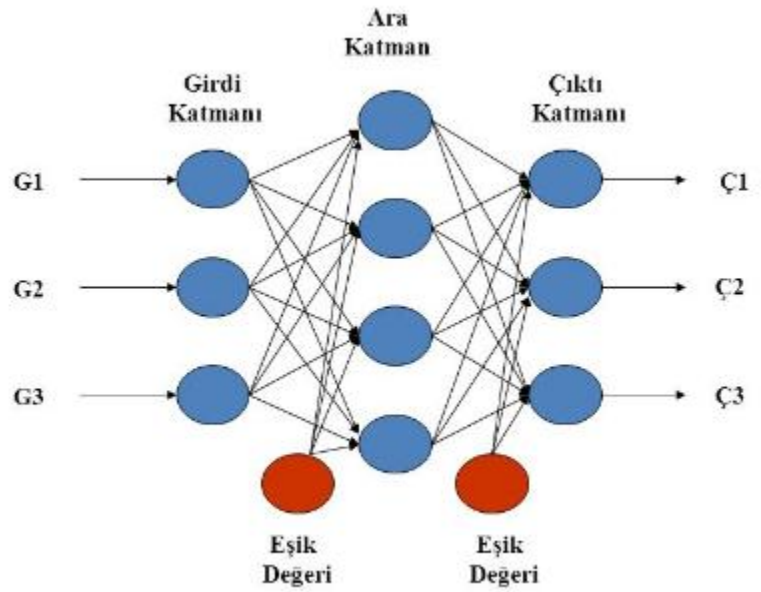
Öğrenmeyi daha iyi yapabilmek için ara katman sayısın arttırılır. Bu sayade girdilerden elde edilmiş istatiki bilgiler daha fazla olur. Ağların yapılarına göre ÇKA'lar ileri beslemeli veya da geri yayılımlı olarak tasaralanır [27].

Girdi katmanında sonucu çıkacak bir türden işlem olmaz. Sadece dışarıdan alınan veriler sisteme giriş yaparlar. Ve bilgiler bir sonraki katman olan ara katmana akıtılır. Dışarıdan alınan girdi sayılarında bir sınır yoktur. Girdi katmanındaki değişkenler ara katmanında bulunan her elemana gönderilir. Buradan anlaşılacağı üzere girdi katmanı, alınınan bilgilerin diğer katmanlara dağıtır.

Ara katmanın diğer bir adı da gizli katmandır. Bilginin işleme alındı yerdir. Girdi katmanının ilettiği bilgiler burada işlem görür. Bu işlemlerin sonuçları yine ileri doğru olarak çıktı katmanına aktarılır. ÇKA'ların yapısı itibareyle birden fazla gizli katmana sahip olabirlirler. Ara katman için çalışan işlem yapan elemenlar diğer katman olan çıktı katmanının tüm işlem elemanları ile bağlantılırdır. Ara katmanda işlemler sonrasında oluşan bilgiyi çıktı katmanındaki tüm işlem elemanlarına gönderir [29].

Ara katmandan gelen bilgiler çıktı katmanında işlem görerek, girdi katmanından girdi olarak girmiş verilerin bir çıktısını oluşturmuş olur. Ara katmanında olan birden fazla işlem elamanı, bu katman içinde geçerlidir. Bir önceki katman olan gizli katmanın bütün işlem elemanları ile bağlıdırlar. Buna rağmen çıktı elemanlarının sahip olduğu bir çıkış vardır. Bu çıkış ise nihayi sonucu verir [29].

ÇKA'nın öğrenme yöntemi, öğretmenli öğrenme yapısındadır. Bir ÇKA ile ağ kurulduğunda eğitim setinden işleme girmesi için veriler verilir. Bununla birlikte çıkmasını istenilen çıktı değerleride verilir. ÇKA öğrenme yaparken almış olduğu bilgilerden bir genelleme oluşturur. Bu sayede çözülecek problemler için bir model oluşturur. Hazırlanmış olan modeli kullanarak ileride sorulacak olan problemlere karşı çözüm üretir.



Şekil 14: Çok katmanlı algılayıcı modeli [29].

ÇKA ağının yapısı verilen her girdiye bir çıktı oluşturmak üzere tasarlanmıştır. ÇKA'nın kullandığı yöntem Genelleştirilmiş Delta Kuralıdır. Bu yöntem en küçük kareler yöntemi olarakta bilinir. Ağı oluşturmak için eğitim seti adlı veri grubunu ağa sürülmesi gerekmektedir. Böylece ağın girdileri verilmiş olur.

Eğitim setinin girdi olarak bilinen verilerin yanında, ağın eğitimden sonra çıkartması gereken çıktılarıda içermesi gerekmektedir. Genelleştirilmiş Delta Kuralı iki gruba ayrılır. Bunlar ileri beslemeli ağ ve geriye yayılım ağı olarak bilinirler.

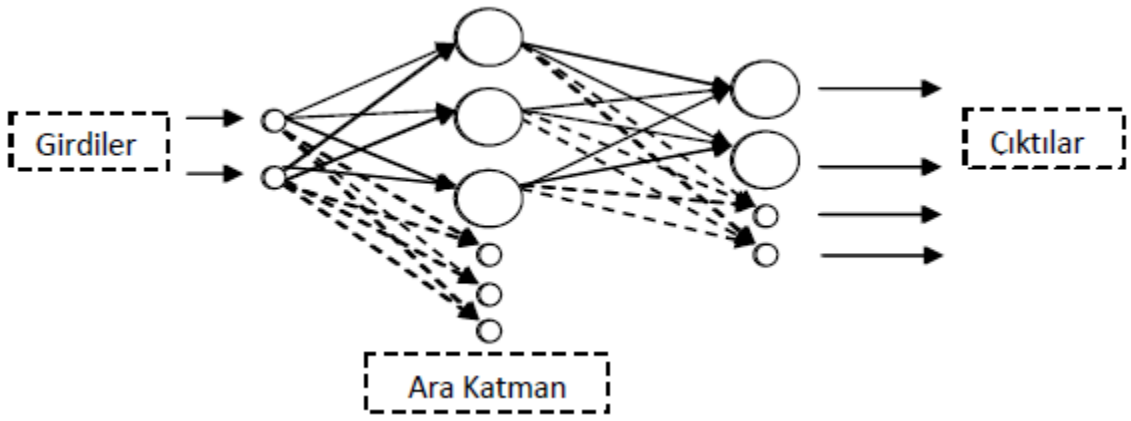
6.1.1.4.2.2. Çok katmanlı ileri beslemeli ağ

Çok katmanlı ileri beslemeli ağlar bir girdi katmanına, bir veya daha fazla ara katmana ve bir de çıktıkatmanına sahiptirler. Bu tür ağların öğrenmek için kullandığı yöntem, öğretmenli

öğrenmedir. Çıktı katmanın sahip olduğu her işlem elemanı için düzenlenebilen ağırlıkları vardır. Ve de bir önceki katman olan gizli katmandan bilgileri alır. Doğrusal olan problemlerin sınıflandırmalarında tek katmanlı ileri beslemeli ağlar kullanılır. Buna paralel olarak doğrusal olmayan ve zor problemleri sınıflandırabilmek için ise fazla olan gizli katman sayılı çok katmanlı ileri beslemeli ağlar kullanılır[28].

Bütün işlemler gizli katman da yapılır. Gizli katmanın diğer bir görevi ise girdi ve çıktı katmanlarını birbirlerine bağlamaktır. Ağın sahip olduğu birden fazla gizli katman, tek katmanla çözülemeyen problemlerin çözümünde büyük rol oynar.

Eğitimde kullanılacak olan veri seti direkt olarak girdi katmanına uygulanırlar. Giriş katmanındaki çıkış değerleri aynı şekilde direkt olarak gizli katmanın girişi olurlar. Bir sonraki katmanının girdilerini, ondan önde olan katmanın çıktıları oluşturur. Böylece ileriye doğru ilerliyen bir yapı oluşur. Çıktı katmanı ise girişte verilmiş olan desenin, ağda oluşturulmuş olan yanıtını gösterir [28].



Şekil 15: Çok katmanlı ileri beslemeli ağ modeli [29].

Şekil 15'de çok katmanlı ileri beslemeli bir ağ yapısı görülmektedir. Ağda girdi katmanı, ara katman ve çıktı katmanı olmak üzere üç katman bulunur. Burada dikkat edilmesi gereken nokta, bir ileri beslemeli ağda, işaretlerin girişten çıkışa bir veya daha fazla ara katman üzerinden yayılmasıdır.

6.1.1.5. Geri yayılım ağı ve algoritması

Çok katmanlı ağlarda sıkça kullanılan bir algoritmadır. Diğer ÇKA'lar gibi giriş katmanı, gizli katman ve de çıkış katmanlarından oluşur. Geri yayılım algoritması katmanlar arasında oluşan ağırlıkları, çıkışta oluşabilecek hataları önlemeye yönelik düzenlemeler yapar. Eğitim seti ile beraber çıkmasını istediğimiz değerler kurulmuş olan ağı eğitirken kullanılır. Bilgiler içerisinde sakladığı desen ile girdi katmanından eğitim seti ile verilir. Buradan ara katmana geçerler. Bir diğer kataman olan çıkış katmanına ağırlıkları ile ulaşırlar. Oluşturulan ağ içerisinde işlem elemenları kendi ağırlık değerlerinin aritmatik toplamını, bir sonraki katmanın tüm işlem elemanlarına gönderir. Aktivasyon fonksiyonu, bu değerlerin üretilmesinde kullanılır. Fonksiyonun çeşidine göre farklı değerler hesaplanır.

Haykin [28] çalışmasında katmanlar arasındaki ağırlıkların yenilenmesini sigmoid aktivasyon fonksiyonu kullanarak aşağıda gösterildiği gibi elde etmiştir. Çıkış katmanındaki her bir işlem elemanı için çıktı bilgisi

$$O_k = \frac{1}{1+e^{-net_k}} \tag{4.3}$$

şeklinde tanımlanmış olsun. Burada Ok çıktı katmanının aktivasyon değerini göstermektedir.

$$net_k = \sum_j W_{jk} O_j \tag{4.4}$$

Benzer şekilde ara katman için aktivasyon değerlerinin ifadesi aşağıdaki gibi bulunur.

$$O_j = \frac{1}{1+e^{-net_j}} \tag{4.5}$$

$$net_j = \sum_i W_{ij} O_i \tag{4.6}$$

Ağırlıkların yenilenmesi

$$W_{jk} = W_{jk} + \ W_{jk} \tag{4.7}$$

eşitliği ile gerçekleştirilir. Burada Wjk ağırlık yenileme değeridir. Geri yayılım algoritmasında ortalama kare hatası olarak bilinen hata kriteri kullanılabilir.

$$E = \frac{1}{2}\sum_{p}\ \sum_{k}(t_{pk} - o_{pk})^2 \tag{4.8}$$

Hataların karesi alınarak beklenen değerden uzak olan çıkış değerlerinin toplam hatayı oluşturması sağlanmaktadır. Hatayı minimum yapmak amacı ile hatanın ağırlıklara olan bağımlılığı hesaplanır ve gradyana bağlı olarak ağırlıklar hatayı düşürecek şekilde yenilenmektedir.

$$W_{jk} = -\eta(\partial \mathrm{E}/\partial W_{jk}) \tag{4.9}$$

Zincir kuralı kullanarak diferansiyel denklem çözümü aşağıdaki şekilde elde edilir.

$$\partial \mathrm{E}/\partial W_{jk} = \delta_k O_j \tag{4.10}$$

Bu eşitlik bir önceki Denklem 4.9'da yerine konursa ağırlık yenileme değeri aşağıdaki gibi elde edilir.

$$W_{jk} = -\eta\delta_k O_j \tag{4.11}$$

$$W_{ij} = -\eta\delta_j O_i \tag{4.12}$$

$$\delta_k = (t_k - O_k) f^{'}(net_k) \tag{4.13}$$

Burada δk ve δj sırası ile çıkış ve saklı katman için hata terimi, η ise öğrenme oranıdır. Çıktı katmanı için hata terimi. Ve ara katman için hata terimi ise aşağıdaki gibi hesaplanmaktadır.

$$\delta_j = f'(net_k)\sum_k \delta_{ki} W_{kj} \quad (4.14)$$

Yukarıdaki ifadelerde f ' katmanlar arası sigmoid aktivasyon fonksiyonunun türevidir. Her bir ağırlıklı bağlantı için algoritmanın üreteceği ağırlık yenileme tek tek incelenir ise ağırlık değerlerinin her bir katman için işlem elemanı aktivasyon seviyeleri dikkate alınarak verilmesi daha uygun olmaktadır.

$$W_{jk} = -\eta O_k' \; O_k O_j \quad (4.15)$$

$$W_{ij} = -\eta O_j' \sum_k^M O_k' W_{jk} \; O_k O_j \quad (4.16)$$

$$\delta_k = O_k(1-O_k)(t_k - O_k) \quad (4.17)$$

$$\delta_j = O_j(1-O_j)\sum_k \delta_k W_{jk} \quad (4.18)$$

$$O_k = f(\sum_j O_j W_{jk}) \quad (4.19)$$

$$O_j = f(\sum_i O_i W_{ij}) \quad (4.20)$$

$$O_k = (t_k - O_k) \quad (4.21)$$

Denklemlerde kullanılan terimler aşağıda tanıtılmıştır.

f: sigmoid aktivasyon fonksiyonu

δ : delta hata ifadesi

η : öğrenme oranı

tk: beklenen değer

Ok: çıktı aktivasyon seviyesi

Ok': çıkış aktivasyon seviyesinin türevi

Wjk: ara katman-çıktı katmanı arasında ağırlıklı bağlantı

Wjk': ara katman çıktı katmanı arasında ağırlıklı bağlantılar için ağırlık yenileme değeri

Wij: girdi ve ara katman arasında ağırlıklı bağlantı

Wij': girdi ve ara katman arasında ağırlıklı bağlantılar için ağırlık yenileme değeri

Elde edilen son ifadelerden yapay sinir ağında girdi değerlerinin ağırlık değerlerinin belirlenmesinde ve dolayısı ile işlem elemanlarının eğitimde önemli bir rol oynadığı görülmektedir [27].

Yapay sinir ağları, girdileri bir kez öğrendikten sonra, ağırlık değerlerini öğrenme işlemine göre ayarlar. Bu çalışmada, geri yayılım (back-propagation) algoritmasının bir türü olan Levenberg – Marquardt algoritması kullanılmıştır.

6.1.1.6.Yapay sinir ağının öğrenmesi

Yapay sinir ağlarının tasarımını bir tarafa bırakırsak, ağırlıkların kullanıcağı değerleri seçmek en önemli işlerden birirsidir. Ağdan alınacak verim ile doğrudan ilişkisi vardır.

Yukarıda bahsettiğimiz işlem, kurulan ağın eğitilmesi olarakta adlandırılır. Bunu takip eden süreçte kurulmuş olan ağın çözülecek problem ile doğru sonuçları veren ağırlık değerlerine ulaşmasına ise öğrenme denilmektedir. Öğrenme iki grubta incelenir: öğretmenli ve öğretmensiz öğrenme [29].

6.1.1.6.1. Öğretmenli öğrenme

Bu iki gruptan en yaygın olarak öğretmenli öğenme, ağların eğitilmelerinde kulanılır. Girişte verilmiş olan beklenen değerler ile eğitimin sonunda çıkmış olan sonuçlar karşılaştırırlır. Eğitim için en başta rastgele olarak verilen değerler, ağ tarafından düzenlenirler. Bu methot ile bir sonraki değer için yapılacak olan hesaplamada, beklenen değere biraz daha yaklaşılmış olur. Bu süreç işlem sırasın da işlem elemanlarında oluşabilecek hataları minimum standarda ulaştırır. Bu işlem girdi ağırlıklarını sürekli bir şekilde değiştirerek, kabul edilebilcek bir ağ performansına kadar devam eder.

Tasarlanmış olan YSA öğrenme başlatılmadan önce eğitilmelidir. Girdi katmanına verilen girdiler ile beklenen değerlerin verilme sürecine eğitim denir. Sisteme verilen bu değerlere eğitim seti denir. İçerisinde sunulan herbir değer için bir beklenen değer barındırırlar.

Bu süreç kullanılan donanıma göre farklı sürelerde sonuçlanabilir. Ağı tasarlayan kişilerin sınırladığı performansa göre sonlanır. Burada önemli olan konu, verilen girdi değerlerine en makul sonucu üretmesi beklenir. Daha yüksek seviyeler beklenmiyor ise elde edilmiş olan ağırlık değerleri kullanlır. Artık test işlemlerinde kullanılır. Bu durumda ki ağa öğrenmesini bitirmiş bir ağ denir.

Eğitim setininde ihtiyaç duyulan bütün bilgileri sağlaması durumunda, ağdan önemli özellik ve ilişkileri öğrenmesini isteyebiliriz. Kurulacak olan ağda sadece bir örnek olay eğitiliyor ise, girdileri oluşturacak olan veri setinin çok iyi seçilmesi gerekir. Aynı şekilde başlangıçta atanacak olan ağırlıklar da çok dikkatli seçilmelidir.

Önceden öğrenilen örnek olay yeni bir şey öğrenildiğinde ağ tarafından unutulabilir. Sonuç olarak sistem her şeyi birlikte öğrenmeli ve bütün örnekler için en iyi ağırlık değerlerini belirlemelidir. Bir ağı başarılı bir şekilde eğitmek için, girdi ve çıktı verilerinin ağa nasıl sunulacağı çok önemlidir. Yapay sinir ağları sadece sayısal girdi verileri ile çalışabilmekte, bu nedenle dış dünyadan alınan sembolik ve sayısal olmayan verilerin sayısal değerlere dönüştürülmesi gerekmektedir. Ayrıca, bu verilerin ölçeklendirilmesi veya ağın algılayabileceği şekle getirilmesi gerekmektedir.

Ağın tasarlama evresinden sonra eğitimi yapılır. Eğitim görmüş ağ artık test edilmek için hazırdır. Bunun için ağa sunulması beklenen test verileri olur. İçerisinde test verilerini saklayan yapıya test seti denir. Daha önce hiç görmadiği bu kayıtlardan makul çıktılar üreten ağa öğrenmiş olan ağ denir. Bu aşamada ağın ezberleme yapmadığını göstermesi gerekir. Uygulamadaki genel örnekleri öğrendiğini göstermesi önemlidir.

Widrow ve Hoff tarafından geliştirilen delta kuralı ve Rumelhart ve McClelland tarafından geliştirilen genelleştirilmiş delta kuralı algoritması öğretmenli öğrenmeye örnek olarak verilebilir.

6.1.1.6.2. Öğretmensiz öğrenme

Gelecek için kendisinden fazlaca söz ettirecek bir methot olarak çıkar karşımıza öğretmensiz öğrenme. Herhangi bir olayı kendi kendilerine öğrenebilirler. Ve bu öğrenim sürecinde aynı zamanda organizasyonuda sağlarlar. Bir diğer adları da kendi kendini örgütleyen ağlardır. Öğremensiz öğrenme metotları ile kurulmuş ağlar, ileride her çeşit makinanın yazılımına entegre edilebilir. Bu insanlar için işten, zamandan ve paradan çok büyük tasarruflara sebep olacaktır.

Öğretmensiz öğrenme yöntemleri ile oluşturulan YSA'lar girdi ağırlıklarını belirlemek için dışarıdan bir etkiye ihtiyaç duymazlar. Bunun yerine performanslarını içeriden yaptıkları gözlemler ile belirlerler.

Bu ağlar girdi verilerinde bir düzen ararlar ve ağınfonksiyonuna göre kendilerini ayarlamaktadırlar. Ağa tanıtılan bir verinin doğru veya yanlış olup olmadığı belirtilmeden, ağ onu nasıl organize edebileceği hakkında bazı bilgilere sahip olmaktadır. Bu bilgi ağ topolojisinin ve öğrenme kurallarının içine yerleştirilmiştir.

Öğretmensiz öğrenme algoritması işlem elemanlarından oluşan gruplar arasındaki işbirliğine önem vermektedir. Eğer grup içine dışarıdan bazı girdiler aktivite edilirse, grubun etkinliği artabilir. Benzer şekilde, eğer gruba dışarıdan verilen girdiler azaltılırsa, bütün grup üzerinde engelleyici etki yaratabilmektedir [29].

6.1.1.7. Yapay sinir ağlarının temel özellikleri

Yapay sinir ağlarının işlem yapma şekli pareleldir. YSA'nın bu yapısı günümüz bilgisayarlarıyla arasında oluşan bir farktır. İnsan beyninde olan öğrenebilme ve genelleme yapma yetenekleri yapay sinir ağlarının da özelliklerindendir. Yapay sinir ağlarının genelleme yeteneği öğrenme sürecinde YSA'nın doğru tepki vermesi ya da yaratması olarak bilinir. Bu yönüyle zor problemlerin üstesinden gelebilir.

Yapay sinir ağlarının parelel çalışması vede genelleme yapabilmesinin dışında başka becerileride vardır. Aşağıda bu yeteneklerden bazılarına değinilmiştir.

Yapay sinir ağları doğrusal olmayan bir yapıya sahiptir. O sebepten dolayı bu tarz problemlerin çözümünü daha kolay başarır. Yapay sinir ağlarının aldığı değerleri değiştirerek, yeni bir problem çözülebiilr. Yani problemler farklı olsa bile aynı eğitim seti ile buna cevap verebilir.

Yapay sinir ağlarının yapısı pareleldir. Bu sayede işemi yapan elemanlarının biri ya da daha fazlasında hata olursa, alınmış olan doğru kararı etkilez. Bu yapısından dolayı hataları tölere edebilirler.

6.1.1.8. Yapay sinir ağlarının avantajları

YSA'nın önemli görülmüş avantajlarının bir listesi aşağıda verilmiştir: [29].

- Önceden hiç görülmemiş örneklerin hakkında bilgi üretebilir. Buna genelleme yeteneği denir.
- Yapay sinir ağları eğitildikten sonra çok isabetli bir şekilde sınıflandırma yapabilir. Bu yeteneği ile yeni gelen örnekleri kolay bir şekilde sınıflar. Yani sınıflandırmayı iyi bir şekilde yapar.
- Eksik bir örüntünün tamamlanması gibi görevleri yerine getirebilir. Bunun benzer bir şekilde eksik kalmış bilgi ile çözülememiş bir olay hakkında öngörülerde bulunabilir.
- Yapay sinir ağları kendi kendine öğrenme yeteneğine sahiptirler.

6.1.2. Karar ağaçları

Karar ağaçları, sınıflandırma algoritmalarının arasında en popüler olanıdır. Anlaşılması kolay ve temel bir algoriması bulunur.

Karar ağacının eğitimi tümevarım eğitimidir. Sadece kararlar değil, bu kararlara ait açıklamalarıda gösterir. Eğitim kümesinden karar ağaçlarının oluşturulmasına, ağaç tümevarımı denir. Çıkarılması gereken bilgi için en genel olarak kullanılan yöntemlerden biridir. Kulanılacak

olan sınıflandırma algoritmaları için kullanılabilelecek ağacı ortaya çıkakartmaya izin verir. Bu şekilde doğru öngörü yapmak için zemin hazırlar [30].

Yapılan testlerin her biri karar ağacının bir dalı olarak ifade edilir. Oluşan dallar ise tekrardan bu testler için bir zemin oluşturur. Bu süreç bir işlemin sonu olarak, dalın yaprağında (leaf node) son bulur. En baştaki kökten yaprağa kadar inen yola kural denir. Kurallar karar ağaçları algoritmasının en önemli mekanizması olarak bilinir. Bu kurallar sınıflandırmada büyük rol oynarlar. Eğer-sonra (if-then) yapısındadırlar. Aşağıda karar ağaçlarına ait bazı analizler verilmiştir:

- Segmentasyon yaparken kullanılır,
- Günlük hayattaki bazı risk gruplarını değerlendirirken. Onları derecelemede kullanılır,
- İleride gerçekleşebilmesi muhtemel olaylar için tahmin oluşturmak,
- Parametrik modeller için kullanılır. Birden fazla veri kümesinden değişkenlerin seçilmesi,
- Alt gruplarla ilişkilerin tanımlanması,
- Sürekli olan değişkenleri, kesikli değişken olarak değiştirilmasinde, aynı zamanda kategroirileri birleştirmek için de kullanılar,

yaygın olarak kullanılırlar.

6.1.2.1. Karar ağaçları tümevarımının adımları

Bu süreçte ilk olarak boş bir ağaç oluşturulur. Arkasından eğitim setimiz gelir. Tümevarım methotdu tekrarlı bir yapıdadır. Her tekrar dört adet adım içerir. Bu adımların açıklamaları aşağıda verilmiştir [31]:

Adım 1: Eğitim setindeki bütün nesneler aynı sonuçta iseler bu sonucu kullanarak bir yaprak oluştur ve dördüncü adıma geç.

Adım 2: Karar ağacı araçlarının yardımları ile kökten başlayıp yaprağa kadar giden yolda bütün niteliklerden en iyisini seçer. Bu seçilen niteliklerin üzerinden bölme işlemi uygulanarak içsel bir düğüm oluşturulur. Bu düğüm sınıflandırmak için çok önem taşır. Bu işlemlerin sonrasında eğitim seti alt kümelerine ayrılmış olur.

Adım 3: Eğiitim setinden oluşturulan altkümeler için birinci adıma gidilir.

Adım 4: Bir düzey yukarı çıkılarak adımlar tekrar edilir.

Bu işlemlerden de anlaşılacağı üzere aslında karar ağacı algoritmaları iki temel işlem gerçekleştirir.

Bu işlemler; bölme (splitting) ve budama (pruning) işlemleridir. Algoritmaları sonlandırmak için ise çeşitli durdurma kriteleri uygulanır. Aşağıda bu sürecin nasıl işlediği verilmiştir [31]:

- Bölme: Bu işlemde veriler daha küçük alt kümelerine ayrılırlar. Tekrarlı bir süreç yapısı vardır. İlk tekrar bütün veriyi temsil eden kökte gerçekleşir. Bundan sonra yapılacak olan tekrarlar kök düğümden meydana gelmiş küçük olan alt düğümlede gerçekleşir. En iyi bölümün yapılabilmesi için değişkenler analiz edirlir.

- Budama: İstenilmeyen alt ağaçların veya oluşturulmuş olan düğümlerin ayıklanmasına denir. Daha değişik ifade ile karar ağacını daha genel bir yapıya somaktır.

- Durdurma kriteri: Ağaç oluşturmak için kullanılan algoritmar, içerisinde birçok durdurma kriteri barındırırlar.

Bu kurallara bakılarak, ağaçların sahip olduğu derinlik, bir düğümde bölme için kullanılan en az eleman sayısı ve yeni bir düğümde olması gereken en az eleman sayısı gibi faktörlere bağlıdır. Eğitim süreci bittikten sonra, yeni bir veri örneği için elimizdeki ağacın kökünden başlayarak yaprak düğümüne kadar gidilir. Bu işlem sonrasında kullanılabilecek olan bir öngörü ortaya çıkmış olur.

Eğitim süreci bittikten sonra, yeni veri örneğini test için daha önceden hazırlanmış olan ağacın kökünden başlayarak yaprakla karşılaşana kadar devam edilir. Ve bu yol sayesinde öngörüde bulunulur.

İzlenecek olan yolun, yeni gelen örnekler için bağımsız olan değişkenlerin değerleri üzerinden sağlanan karar ağacının bölme kuralı uygulanır.

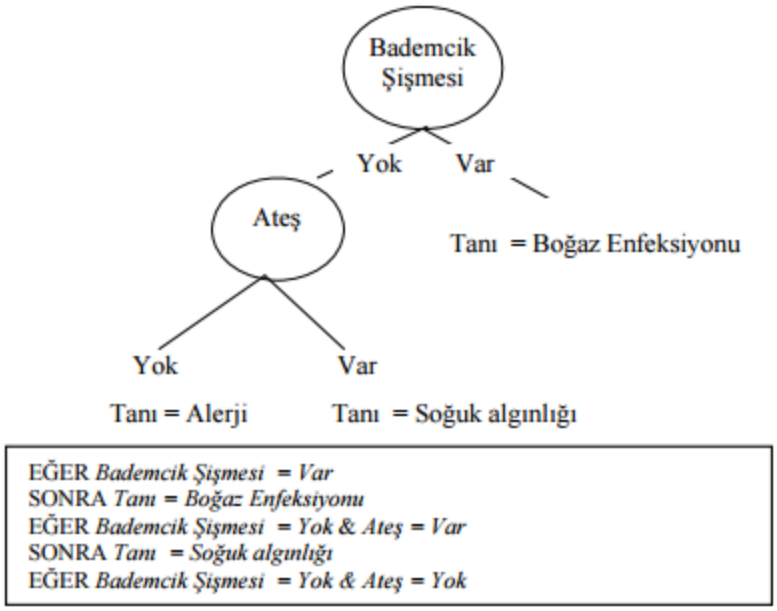
Tablo 4'de kategorik yapıda verileri olan bir hasta veri tabanı, Şekil 16'da buna ait bir karar ağacı ve bu ağaçtan elde edilen kurallar verilmiştir.

HASTA SIRA NO.SU	BOĞAZ AĞRISI	ATEŞ	BADEMCİK ŞİŞMESİ	KAN TOPLA-MASI	BAŞ AĞRISI	TANI
1	Var	Var	Var	Var	Var	Boğaz Enfeksiyonu
2	Yok	Yok	Yok	Var	Var	Alerji
3	Var	Var	Yok	Var	Yok	Soğuk algınlığı
4	Var	Yok	Var	Yok	Yok	Boğaz Enfeksiyonu
5	Yok	Var	Yok	Var	Yok	Soğuk algınlığı
6	Yok	Yok	Yok	Var	Yok	Alerji
7	Yok	Yok	Var	Yok	Yok	Boğaz Enfeksiyonu
8	Var	Yok	Yok	Var	Var	Alerji
9	Yok	Var	Yok	Var	Var	Soğuk algınlığı
10	Var	Var	Yok	Var	Var	Soğuk algınlığı

Tablo 4: Örnek hasta veritabanı [31].

6.1.2.2. Karar ağaçları algoritmaları

Karar ağaçlarının en temelindeki yöntem AID (Automatic Interaction Detector) adındaki algoritmadır. Bu yöntemin arkasından birçok yeni algoritma çıkmıştır. 1970 yılında Morgan ve Sonquist isimli iki araştırmacı tarafından bulunmuştur. Özellikleri arasındaki ilk göze çarpan, karar ağacı tabanlı bir yazılım olmasıdır.



Şekil 16:Örnek hasta veritabanı için bir karar ağacı ve kurallar [30].

Veri bilimlerinde karar ağaçları ile yapay öğrenme işlemi çok sık uygulanmasına rağmen, bilgiyi elde etmek için uzun seneler boyunca pek tercih edilmemiştir. 1984 yılında Berkeley Üniversitesi'nden Leo Breiman ve Charles J. Stone ile Stanford Üniversitesi'nden Jerry Friedman ve R. Olshen tarafından basılan "Classification And Regression Trees" adlı kitapta yeni bir karar ağacı yordamı olan C&RT algoritmalarının kullanılmasından bahsedilmektedir. Bu araştıma ile karar ağaçlarının istatistiksel alanlarda kullanılması sağlanmıştır.

1986 yılında J.R. Quinlan adlı araştırmacı karar ağaçlarına yeni bir algoritma eklemiştir. Bu karar ağacı algoritması literatüre ID3 algoritması olarak geçmiştir. 1993 yılında ise Quinlon adlı bir başka araştırmacı "Programs For Machine Learning" adlı kitabında C4.5 karar ağacı algoritmasını ortaya koymuştur .

Geliştirilen diğer algoritmalar arasında CHAID (G.V. Kass; 1980), Exhaustive CHAID (Biggs, de Ville ve Suen; 1991), MARS (Multivariate Adaptive Regression Splines; Friedman), QUEST (Quick, Unbiased, Efficient Statistical Tree; Loh ve Shih, 1997), C5.0 (Quinlan), SLIQ (Mehta, Agarwal ve Rissanen), SPRINT (Shafer, Agrawal ve Mehta) yer almaktadır (Kirchner, 2004, 116). Tablo 5'de bazı karar ağacı algoritmalarının özellikleri verilmektedir [32].

KARAR AĞACI ALGORİTMASI	ÖZELLİKLER
C&RT	Gini'ye dayalı ikili bölme işlemi mevcuttur. Son veya uç olmayan her bir düğümde iki adet dal bulunmaktadır. Budama işlemi ağacın karmaşıklık ölçüsüne dayanır. Sınıflandırma ve regresyonu destekleyici bir yapıdadır. Sürekli hedef değişkenleri ile çalışır. Verinin hazırlanmasına gereksinim duyar.
C4.5 ve **C5.0** (ID3 karar ağacı algoritmasının ileri versiyonları)	Her düğümden çıkan çoklu dallar ile ağaç oluşturur. Dalların sayısı tahmin edicinin kategori sayısına eşittir. Tek bir sınıflayıcı da birden çok karar ağacını birleştirir. Ayırma işlemi için bilgi kazancı kullanır. Budama işlemi her yapraktaki hata oranına dayanır.
CHAID (Chi-Squared Automatic Interaction Detector)	Ki-kare testleri kullanarak bölme işlemini gerçekleştirir. Dalların sayısı iki ile tahmin edicinin kategori sayısı arasında değişir.
SLIQ (Supervised Learning in Quest)	Hızlı ölçeklenebilir bir sınıflayıcıdır. Hızlı ağaç budama algoritması mevcuttur.
SPRINT (Scalable Parallelizable Induction of Decision Tree)	Büyük veri kümeleri için idealdir. Bölme işlemi tek bir niteliğin değerine dayanır. Tüm bellek sınırlamaları üzerinde nitelik listesi veri yapısı kullanarak işlem yapar.

Tablo 5: Bazı karar ağacı algoritmaları ve özellikleri [31].

6.1.3. Rastgele Orman (Random Forest)

RF (Random Forest) algoritması Leo Brieman adındaki bir araştırmacı tarafından 2001 senesinde geliştirilmiştir. Breiman, kendisinin 1996 yılında geliştirdiği Bagging yöntemi ile Ho tarafından 1998'de önerilen ve rasgele alt gruplar seçmek için kullanılan The Random Subspace tekniğini birleştirerek yeni bir yöntem oluşturmuştur [34]. Bu çalışmaları sürdürürken okuduğu kaynaklardan etkilendiğini söylemiştir. Bu kaynakların en başında gelen çalışmalar ise Amit ve German'nın çalışmalarıdır.

RF'de farklı farklı oluşturulan sınıflamadırma ağaçları (CART) ile birer orman oluşturulur. Bu yüzden bir toplu öğrenme yöntemi de denir. Orman oluşurken elde edilen sonuçlar bir araya getirilerek düzenlenir. Ve bunun sonrasında bir öngörü yapılır.

RF yönteminde ağaçlar, seçilen bootstrap örneklemleri ve her düğüm ayrımında rasgele seçilen m adet tahminci ile oluşturulur. m adet tahmincinin toplam tahminci sayısından oldukça küçük olmasına dikkat edilir. Bu süreçte oluşan karar ağaçları en geniş şekliile kalır ve budanmaz.

Regresyon için ise; yaprak düğümde az sayıda birim kalana kadar ağaçlar bölünmeye devam ederler [34].

Rastgele orman algoritması diğer yapay öğrenme algoritmalarına kıyasla çok daha iyi öngörü geçerliliği ve model yorumlaması yapar. Bu septen ötürü genellemelerde çok başarılıdır. Rastgele örnekleme yapmasıda rastgele orman algoritmasının isabetli tahminler yapmasında büyük rol oynar.

RF yönteminin tahminlerinin kesinliğinin nedenleri yanlılığı düşük sonuçlar vermesi ve ağaçlar arasındaki düşük korelasyondur. Düşük yanlılık miktarı, oldukça büyük ağaçların oluşturulması sonucu elde edilir. Mümkün olduğunca birbirinden farklı ağaçlar oluşturarak da düşük korelasyon yapısında bir topluluk elde edilir.

RF'nin bazı sınıflandırma özellikleri [34] ;

1) Çok iyi bir geçerlilik verir. Adaboost ve Destek Vektör Makinalarının(Support Vector Machines) sonuçlarından daha kesin sonuçlar verirler.

2) Sonuç verme süresi minimumdur. 100 değişkenli 100 ağaçlık bir karar ormanı, arka arkaya kurulan 3 tekil CART ile aynı sürede oluşturulur.

3) Binlerce değişkene ve fazla sayıda sınıf etiketine sahip kategorik değişken içeren, kayıp verili veya dengesiz bir dağılım sergileyen veri setlerini kullanarak sonuçlar verir.

4) Topluluğa ağaçlar eklendikçe, test setine ait hata tahmini için yanlılığı düşük sonuçlar vermeye başlar.

5) Çok fazla uyumluluk yapmaz .

RF modeli 2 parametre üzerine kuruludur. Bu parametreler; oluşturulacak olan ağaç sayısı (B) ve her düğüm ayrımında rasgele seçilecek olan tahminci sayısıdır (m). Her karar ağacı oluşturulurken, orijinal veri setindeki gözlem sayısı (n) ile aynı ölçüde olacak şekilde bootstrap yöntemi ile örneklem oluşturulur.

Bu örneklemin 2/3'ü ağacı oluşturmak için kullanılan eğitim veri seti (inBag) ve geriye kalan 1/3'ü ise kurulan modelin iç hata oranını test etmek için test veri seti (out of bag veya OOB) olacak şekilde ikiye ayrılır.

RF algoritması aşağıdaki şekilde kurulur [34]:

1) Bootstrap yöntemi ile n hacimli veri seti seçilir. Bu veri seti, eğitim veri seti (inBag) ve test veri seti (OOB) olarak ikiye ayrılır.

2) Eğitim veri seti (inBag) ile en büyük genişlikte bir karar ağacı (CART) oluşturulur ve elde edilen bu karar ağacı budanmaz. Bu ağaç oluşturulurken her düğümün bölünmesinde toplam p tane tahminci değişkenden m tanesi rasgele seçilir. Burada m<p koşulu sağlanmalıdır.

   Çünkü ağacın aşırı büyümesi ve aşırı uyum gözlenmesi istenmemektedir. Seçilen bu m tane tahminciden bilgi kazancı en yüksek olan ile dallara ayrılmak gerçekleşir. Belirlenen bu değişkenin hangi değerine göre ayrımın olacağına Gini indeksi ile karar verilir. Bu işlem her düğüm için yeni oluşturulacak dal kalmayıncaya kadar tekrar edilir.

3) Her yaprak düğüme bir sınıf atanır. Daha sonra test veri seti (OOB) ağacın en tepesinden bırakılır ve bu veri setinde yer alan her gözleme atanan sınıf kaydedilir.

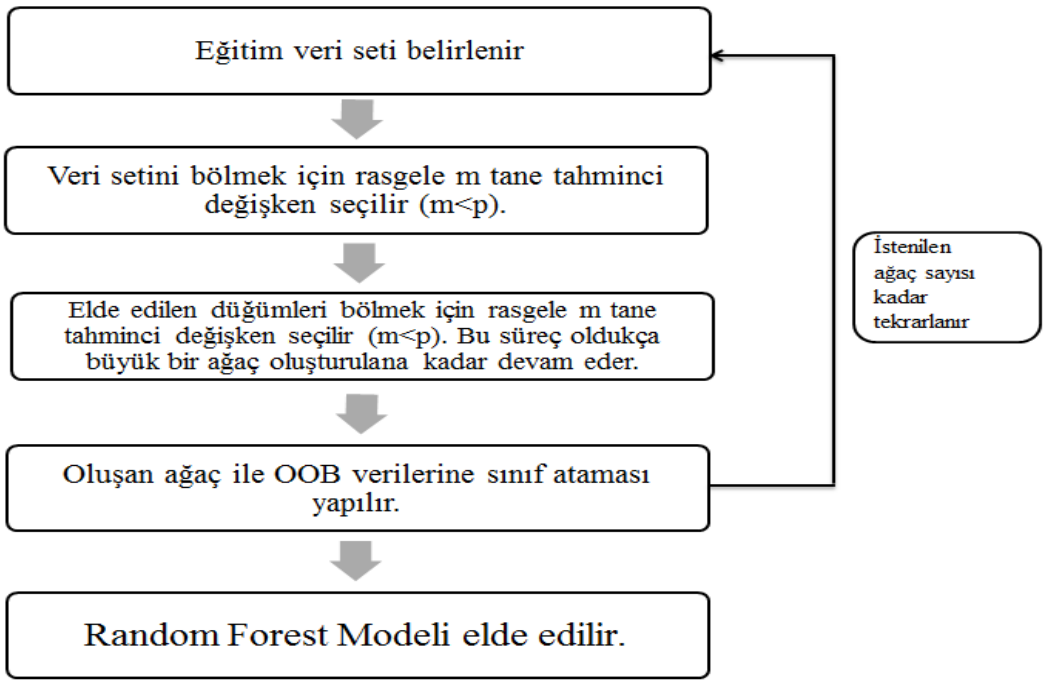
4) 1.'den 3. adıma kadar tüm aşamalar B defa tekrar edilir.

5) Ağaç oluşturulurken kullanılmayan gözlemler (OOB) ile bir değerlendirme yapılır. İncelenen bir gözlemin hangi kategorilerde kaç defa sınıflandırıldığı sayılır.

6) Her gözleme, ağaç setleri üzerinden belirlenen bir oy çoğunluğu ile sınıf ataması yapılır. Örneğin 2 kategoriye sahip bir sınıflandırma modelinde;

   bir gözlem tüm ağaçlar üzerinden en az %51 oy çoğunluğunu aldığı sınıfın etiketini taşır ve bu sınıf onun tahmin edilmiş sınıf değeri olur.

Aşağıdaki Şekil 17'de algoritmanın oluşturulma basamakları gösterilmiştir.



Şekil 17:Rasgele orman oluşturma algoritması [34].

6.1.3.1. Rastgele Orman algoritmasının özellikleri

RF algoritmasının birçok özelliği vardır. Bunlardan bazıları aşağıda açıklanmıştır.

6.1.3.1.1. Genelleme hatası (Generalization error)

Veri kümesinden bir bootstrap örneği alındığında, ağacı oluştuma sürecinde bazı gözlemlerin yer almadığı görülür. Bu gözlemlerin adına OBB denir. OBB ile bu hatalara yönelik iç öngörüler yapılır. Bütün bir gözlem için hata oranının öngörüsünü yapar. Ve bu değerleri kaydeder. Yapılan bütün gözlemlerin ortalamasını alıp genel hata oranı hesaplanabilir. Herhangi bir noktada, bu oranların ortalamasını alarak genellemenin oluşturduğu hataları hesaplayabilir [34].

6.1.3.1.2. Parametreleri ayarlama (Tunning parameters)

Rastgele orman algoritmasında karar ormanları oluşturmadan önce gerekli iki parametre mevcuttur. Her düğüm için rastgele olarak atanacak değişkenlerin sayıyıları (m) ve ağaçların sayısı (B). Rastgele orman algoritması parametreleri seçerken çok hassas davranmayabilir.

Breiman, bu parametrelerin seçimi için bazı önerilerde bulunmuştur. Breiman'a göre 500 adet ağaçtan oluşacak bir karar ormanı yeterli sayılabilir. Pek çok sınıflandırma problemi için her düğümde rasgele seçilecek olan değişken sayısı m=√p eşitliği ile hesaplanmaktadır. Burada p; veri setindeki tahminci değişkenleri sayısını göstermektedir. Regresyon ağaçlarında ise m parametresi m=p/3 olarak elde edilir.

Rastgele orman algoritmasında oluşturulmuş ormanlara fazla sayıda ağaç eklemek aşırı şekilde uyum göstermenenin sebepleri arasındadır. Ağaçlar sayısının yeterli sayıda olması gerekir. Bu sayıyı kontrol ederken OOB den yardım alınır. Şekil 18'de görüldüğü üzere OOB hata oranı belirli bir ağaç sayısından sonra sabit bir değere yakınsar.

Bazı kısıtların önceden tanımlandığı özelleştirilmiş problemlerde farklı parametreler için ayarlamalar yapılabilir. Örneğin regresyon problemlerinde ağaçların derinliğinin ya da yaprak düğümlerde kalacak olan minimum gözlem sayısının kontrol edilmesi gereklidir [34].

Şekil 18: Oluşturulan hata sayısına göre hata oranı değişimi [34].

6.1.3.1.3. Değişken önemliliği (Variable Importance)

Değişkenlerin önemi, her bir değişkenin öngörüdeki gücünü ölçmekte kullanılır. Öngörüde kulllunılacak olan değişkenlerin seçimi doğru tahminler için büyük değer taşır. Yüksek boyutlardaki veri setini daha düşük boyutlara düşürmek gerekebilir. Bu işlem istatiki analizlerin öncesinde yapılır.

Rastgele orman algoritması sınıflandırma kurallarını oluşturduğu anda değişkenlerin seçimini yapar. Modelin performansını yükseltmek, aşırı uyumu engellemektir.

Değişkenlerin önemi iki farklı yöntem ile hesaplanırlar. Bunlardan ilki Gini önemliliği adı verilen bir yöntemdir. İkincisi ise permütasyonlara dayanan değişkenlerin önemliliğidir. Bu iki yöntem birbirlerinden farklı yöntemlerdir. Fakat birbirleri için kullanılacak sonuçlar üretirler.

6.1.3.1.3.1. Gini önemliliği

Rastgele orman ağaçlarını oluştururken kullanılan Gini indexinden elde edilir. Bu indeks bir düğüme atanmış olan örneklerin karışıklığını veya eşitsizliğinin seviyelerini belli eder. Örneğin, iki sınıflı bir sınıflandırma probleminde p; k düğümünde yer alan pozitif gözlemlerin oranını ve 1-p de negatif gözlemlerin oranını göstersin. Bu durumda k düğümünde yer alan Gini indeksi aşağıdaki şekilde gösterilir:

$$Gk=2p(1-p)$$

Bir düğüm ne kadar saflaştırılırsa, Gini değeri de o kadar küçülür. Bir düğümde v değişken üzerinden bölünme gerçekleştiğinde elde edilen yeni iki düğümün Gini değeri, bölünen düğümün Gini değerinden daha küçük olur. Her bir tekil ağaç için v değişkeninin Gini önemlilik değeri bu iki değer arasındaki fark hesaplanarak elde edilir. Ormandaki tüm ağaçlar oluşturulduktan sonra, v değişkenin yer aldığı ağaçlardaki Gini önemlilikleri toplanarak v değişkenine ait önem derecesi belirlenmiş olur [33, 34].

6.1.3.1.3.2. Permütasyona dayalı değişken önemliliği

Rastgele orman algoritması ile v değişkenlerinin önemi şu şekilde sıralanır. İlk olarak OBB tarafından bitirilmiş olan gözlemler, kökten aşağıya doğru indirilir ve öngörüsü yapılan değerler bulunur. Bu süreçten sonra OOB'nin sahip olduğu diğer öngörücü değişkenlerin, v değişkeni için yapılmış gözlemler ile kıyaslanır. Ortaya çıkan yeni oluşmuş veri seti kökten aşağıya doğru indirilir ve bunun sonucunda yeni değerler belirlenirler.

Yukarıdaki işlemlerin sonucunda herbir gözlem iki tane öngörü değerini elde etmiş olur. Değeiştirilen OOB'nin yaptığı doğru öngörüler bulunur. Aynı şekilede OOB'nin orjinali olarak kullanılan doğru öngörüler de bulunur. Sonrasında orijinal olandan, değiştirilen öngörüler çıkartılarak ortaya bir fark değeri konur. Bu süreç bütün ormana uygulanıp, ormanda kaç ağaç

var ise o kadar fark değeri elde edilir. Ve farkların ortalaması bulunur. Bazı durumlarda bütün ağaçlar bağımsız olarak dağılırlar. Böyle durumlardalar için bir skor değeri bulunur. Bu değerin hesaplanması kısaca şöyledir: farkların ortalamalarınının, farkların standart sapmalarına bölünmesi ile elde edilir. Sonuç olarak ele geçen skor değerlerinin önemi, değişkenleri bir sıraya sokmak için kullanılır [33, 34].

6.1.3.1.4. Farklı sınıf büyüklükleri (Unequal class sizes)

Sınıfandırma yaparken bazı dengesini kaybetmiş veri setlerinin problem oluşturduğu gözlemlenmiştir. Bu veri setleri incelendiğinde sınıfların sahip olduğu gözlem satırlarının farkı gözükür. Gözlemlerinin sayısı büyük olan sınıflandırıcılar, aynı şekilde çok hatalı oranlar verebilirler. Rastgele orman algoritması sahip olduğu yöntemler ile bu tarz veri setlerinin sınıflarını ağırlıklandırırlar. Bu süreci gerçekleştirmekteki amaç, yöntemin doğru öngörücü değişkenlerde farklılık gösterebilmesidir. Ne varki dengeli olarak adlandırılan veri setleri bile, yüksek seviyelerdeki hata oranını maliyeti, hata oranını düşürmek için ağırlıklarda oynamalar yapılabilir [33,34].

6.1.3.1.5. Örnekler arası uzaklık (Proximity)

Veri setlerinin dengeli olup olmadığı anlamak bazı durumlarda pek mümkün değildir. Veri setinin özeliklerinin fazla olması, yani çok boyutlu olması bu çeşit bir problemi ortaya koyacaktır. Sınıfların alt grubu var mı? Değerlerde sapma gözüküyor mu? Rastgele orman algoritması bu sorular için veri setine farklı bir bakışaçısıyla yaklaşır. Bu yaklaşıma proximity measure adı verilir. Gözlemi oluşturan çiftlerin arasındaki uzaklığın hesaplanmasıdır. İki gözlemin arasında olan uzaklığa, aynı yapraktaki sonlanma oranı eşittir. Ormanda kaç ağaç var ise bu oranda onların sayıları üzerinden neticelenir. Rastgele orman algoritması bu uzaklık ölçüleriyle bir matris oluşturur.

Yukarıda bahsedilen uzaklık matrisi kare matristir. Byutları farklı olabilir ama yapısı kare ve simetrik bir matristir. Bu matriste satırlar ve sütünlar veri setinde yapılmış olan gözlemlerin kayıtlarını temsil ederler. Veri seti bütüt olarak ağaçtan aşağıya doğru itlenir. Eğer i. ve j. gözlemleri aynı yaprak düğümde sonlanırsa aralarındaki uzaklık 1 arttırılır. Veri seti ormandaki

bütün ağaçlara yerleştirilip uzaklıklar elde edildikten sonra ortaya çıkan matrisin her bir gözesi, ormandaki ağaç sayısına bölünür. Eğer iki gözlem değeri her zaman aynı yaprak düğümde sonlanırsa uzaklıkları 1'e, hiçbir zaman aynı yaprak düğümde olmazlar ise de 0'a eşit olur. Oranları oldukça yüksek olan gözlemler birbirlerine daha benzer bir yapı gösterirlerken, diğer gözlemlerle arasındaki uzaklık oranı oldukça düşük olanlar sapan değer (outlier) şüphesi taşırlar.

6.1.3.1.6. Kayıp değer atama (Missing value imputation)

Sınıflandırma modelleri oluşturmadan önce veri setini bir ön işlemden geçirmemiz gerekebilir. Bunun sebeplerinden biri kayıp değerlerin olmasıdır. Veri setinde kayıp değerlerin olması bir problem yaratır. RF'nın sahip olduğu bir algoritma ile bu kayıp değerleri veri setinde kalmasına olanak sağlar.

Kayıp depğerlerin atanma algoritması şu şekilde çalışmaktadır. İlk olarak veri setinin sahip olduğu kayıp değerler bulunur. Bulunan kayıp değişkenler süreklilik yansıtıyorsa, kayıtların medyanı alınır ve bunu değişkenin değeri olarak atanır. Bulunan kayıp değişken eğer kategorik bir yapıda ise, bu zaman da frekans değeri en yüksek olan kategori o değere atanır. Bu şekilde kayıp değerlere atama işlemi devam eder.

Bu süreç ile hazırlanmış olan veri seti ile rastgele orman algoritması kullanılarak bir model kurulur. Kurulmuş olan bu modelden bir uzaklık matrisi hesaplanır. Ağırlıklandırmak için uzaklık matrisindaki değerler kullanılır. Yapısında süreklilik olan değişkenlerin ağırlıklı ortalaması bulunur.

Yukarıda bulunan değerler kayıp olan verilere değer olarak atanırlar. Eğer eksik verinin değişkeni kategorik yapıda ise, ölçülmüş olan uzaklık oranı en yüksek değer atanır. Bu atama işlemlerinin bitmesinin ardından, yeni oluşturulmuş veri setini kullanarak tekrar RF ile bir model kurulur. Ve daha sonra bir uzaklık matrisi hesaplanır. Bu süreç kuralları hiç değiştirmeden tekrar eder.

Kayıp değerleri atamak için yapılan bu işlem iyi bir sonuç bulana kadar beş kez devam eder. Bu yöntemin bir özelliği: kayıp olan verilerin rastgele olması gerekmektedir [33,34].

Rastgele orman algoritmasının diğerlerine göre üstünlüğünü sağlayan özelliklerden bazıları aşağıda gösterilmiştir:

- Ağaç yapısı birbirlerinden bağımsız bir şekilde tasarlanmıştır. Bu sayede daha isabetli öngörü şansı verir.
- Regresyon analizlerinde öngörücü sayısı veri setinde olan gözlem sayısından küçük olamalı. Rastgele orman algoritmasında böyle bir durum sözkonusu değildir.
- Fazla sayıda ağacı kullanmak, rastgele orman algoritmasını CART'ı uygulamakta daha zor bir durum oluşturur.

  Fakat modelin durumunu değerlendirmek için OOB veri setin yardım alır ve içerideki hatayı belirler.

  Bu süreç CART için bir probleme dönüşen aşırı uyumu engeller.
- Kullanılan birçok sınıflayıcı arasında doğruluk oranı yüksektir.
- Ormanları oluştururken genelleştirme hatalarının öngörüsünü yapar.
- Kayıp olmuş verilerin öngörülerini iyi yapar.
- Dengesiz olarak sınıflanmış veri setlerindeki hataları tespit edebilir.
- Ormanları başka bir veri seti için kullanaılabilir ve saklanabilir.
- Değişken önemliliğini çok iyi becerir.

Rastgele orman algoritmasının iyi özelliklerinden gösterdik. Şimdi ise az sayıda olan kısıtlarınını listeleyelim:

- Tek bir karar ağacında olduğu gibi ortaya çıkan sonuç, ağaç yapısında görsel olarak görülmez.
- Bulunan sonuçlar için bir güven aralığı gösteremez.

7. WEKA İLE VERİ SETİNİN SINIFLANDIRILMASI

- Önce hastalar veri setini bir bütün olarak alıp (3500 kayıdı), onun üzerinde eğitim ve test yapmak. Çapraz doğrulama ile.
- Bir diğeri ise Hastalar.xlsx dosyasını ikiye bölerek, eğitim seti ve test seti oluşturmak.

✓ Hastalar.arff = 3500 kayıt.

✓ HastalarTrain.arff = 2000 kayıt ve HastalarTest.arff = 1500 kayıt şeklinde bölüyoruz.

7.1. Bütün özellikler kategorik yapıda

Elimizdeki veri setinden elde ettiğimiz veri setleri ile şu anda üç tane veri setimiz var. Hastalar.arff (3500) , HastalarTrain.arff (2000) ve HastalarTest.arff (1500).

Ve bütün özellikleri kategorik durumda. Hastalar.arff'de 3500 kayıdı çapraz doğrulama ile sınıflandıracağız.

✓ HastalarTrain.arff ' de eğitim yapıp ,

✓ Hastalar.Test.arff ' de test edilecektir.

Bulguları incelerken Test Seti sonuçları bu dosyalardan elde edilen sonuçlardır.

7.1.1.Karar ağaçları bulgular

	Çapraz Doğrulama (k=10)	Test Seti
Doğruluk Matrisi	a b c 1856 7 0 | a = Dusuk Risk 46 1024 32 | b = Orta Risk 2 143 433 | c = Yuksek Risk	a b c 821 15 0 | a = Dusuk Risk 24 391 21 | b = Orta Risk 6 105 160 | c = Yuksek Risk
Doğruluk Oranı	93.9197 %	88.9177 %

Tablo 6: Karar ağaçları bulguları.

7.1.2. Yapay sinir ağları (Backpropagation) bulgular

	Çapraz Doğrulama (k=10)	Test Seti
Doğruluk Matrisi	a b c 854 7 0 | a = Dusuk Risk 46 102 32 | b = Orta Risk 2 143 1433 | c = Yuksek Risk	a b c 721 15 0 | a = Dusuk Risk 154 88 21 | b = Orta Risk 6 3 100 | c = Yuksek Risk
Doğruluk Oranı	73.3247 %	72.007 %

Tablo 7: Yapay sinir ağları bulguları.

7.1.3. Rastgele orman bulgular

	Çapraz Doğrulama (k=10)	Test Seti
Doğruluk Matrisi	a b c 1863 0 0 | a = Dusuk Risk 28 1067 7 | b = Orta Risk 4 130 444 | c = Yuksek Risk	a b c 836 0 0 | a = Dusuk Risk 28 402 6 | b = Orta Risk 9 101 161 | c = Yuksek Risk
Doğruluk Oranı	95.23 %	90.6677 %

Tablo 8: Rastgele orman bulguları.

Tablo 6, 7 ve 8'deki sonuçları değerlendirdiğimizde: bütün veri setini, yani 3500 kayıtı üç algoritma ayrı ayrı ile eğitilmiştir. Çapraz doğrulama ile aynı kümede test edilmiştir. Karar ağaçları yüzde 93, geri yayılım algoritması yüzde 73 ve rastgele orman algoritması yüzde 95 gibi oranlar elde edilmiştir.

Veri setini train ve test olarak ayırdığımızda; train kümesinde eğitim vede test kümesinde test edilmiştir. Karar ağaçları yüzde 88, geri yayılım algoritması yüzde 72 ve rastgele orman algoritması yüzde 90'lık oranlar elde edilmiştir. Çapraz doğrulamaya göre daha düşük bir oran elde edilmiştir. Çapraz doğrulamada aynı küme üzerinde test edilmesi oranı yükselten sebeplerden sayılabilir.Rastgele orman algoritması en iyi oranları elde etmiştir.

7.2. Yaş özelliğinin sürekli alınması

Elimizde veri setinde bazı değişikler yaparak doğruluk oranınında değişiklik sağlanabilir. Onun için Hastalar.xlsx dosyasındaki hastanın yaşı özelliğini, hastanın doğum tarihinden hesaplayarak çıkarıyoruz.

Örnek olarak; artık yaş özelliği 25, 81, 74, 69 gibi sürekli veriden oluşacaktır. Bunun doğruluk oranını artıracağını bilmiyor. Amaç doğruluk oranında yakarı ya da aşağı yönde bir hareket sağlamaktır.

Aşağıda testin bulguları verilmiştir.

7.2.1. Karar ağaçları bulgular

	Çapraz Doğrulama (k=10)	Test Seti
Doğruluk Matrisi	a b c 1714 149 0 | a = Dusuk Risk 403 583 116 | b = Orta Risk 3 195 379 | c = Yuksek Risk	a b c 738 98 0 | a = Dusuk Risk 146 239 50 | b = Orta Risk 4 92 175 | c = Yuksek Risk
Doğruluk Oranı	75.55 %	74.70 %

Tablo 9: Karar ağaçları bulguları.

7.2.2. Yapay sinir ağları (Backpropagation) bulgular

	Çapraz Doğrulama (k=10)	Test Seti
Doğruluk Matrisi	a b c 1422 874 0 | a = DusukRisk 980 888 32 | b = Orta Risk 9 231 433 | c = YuksekRisk	a b c 562 15 89 | a = Dusuk Risk 98 856 21 | b = Orta Risk 6 254 160 | c = Yuksek Risk
Doğruluk Oranı	68.7845 %	67.8722 %

Tablo 10: Yapay sinir ağları bulguları.

7.2.3. Rastgele orman bulgular

	Çapraz Doğrulama (k=10)	Test Seti
Doğruluk Matrisi	a b c 1592 271 0 | a = Dusuk Risk 360 599 143 | b = Orta Risk 2 186 389 | c = Yuksek Risk	a b c 721 115 0 | a = Dusuk Risk 130 252 53 | b = Orta Risk 5 105 161 | c = Yuksek Risk
Doğruluk Oranı	73.84 %	72.84 %

Tablo 11: Rastgele orman bulguları.

Tablo 9, 10 ve 11'deki sonuçları değerlendirdiğimizde: yaş özelliğini sürekli olarak alıp veri setini, yani 3500 kayıt geri üç algoritma ile eğitilmiştir. Çapraz doğrulama ile aynı kümede test edilmiştir. Karar ağaçları yüzde 75, geri yayılım algoritması yüzde 68 ve rastgele orman algoritması yüzde 73 gibi oranlar elde edilmiştir.

Veri setini train ve test olarak ayırdığımızda; train kümesinde eğitim vede test kümesinde test edilmiştir. Karar ağaçları yüzde 74, geri yayılım algoritması yüzde 67 ve rastgele orman algoritması yüzde 72'lik oranlar elde edilmiştir. Çapraz doğrulamaya göre daha düşük bir oran elde edilmiştir. Çapraz doğrulamada aynı küme üzerinde test edilmesi oranı yükselten sebeplerden sayılabilir.

Yaş özelliğini sürekli aldıtan sonra, aynı şekilde sınıf özelliğini de sürekli bir aralıkta alıp model oluşturulur.

7.3. Sınıf özelliğinin sürekli alınması

Bu veri setinde sınıf özelliği üç farklı değer alıyordu. Toplam puanın karşılığında düşük, orta, yüksek olarak sınıflandırılıyorlardı. Şimdi ise doğruluk oranında değişiklik yapabilmek için, toplam puanı direkt olarak rakam olarak alcağız.

Hesaplanabilecek en yüksek puan 40 tır. O zaman 0 ile 40 arasında sayılardan oluşan bir sınıf özelliğimiz olacaktır.

7.3.1. Karar ağaçları bulgular

	Çapraz Doğrulama (k=10)	Test Seti
Doğruluk Oranı	85.70 %	84.72 %

Tablo 12 : Karar ağaçları bulguları.

7.3.2. Yapay sinir ağları (Backpropagation) bulgular

	Çapraz Doğrulama (k=10)	Test Seti
Doğruluk Oranı	69.9197 %	68.887 %

Tablo 13: Yapay sinir ağları bulguları.

7.3.3. Rastgele orman bulgular

	Çapraz Doğrulama (k=10)	Test Seti
Doğruluk Oranı	88.9021 %	86.2536 %

Tablo 14: Rastgele orman bulguları.

Tablo 12,13 ve 14'deki sonuçları değerlendirdiğimizde: sınıf özelliğini sürekli olarak alıp veri setini, yani 3500 kayıtı üç algoritma ile eğitilmiştir. Çapraz doğrulama ile aynı kümede test edilmiştir. Karar ağaçları yüzde 85, geri yayılım algoritması yüzde 69 ve rastgele orman algoritması yüzde 88 gibi oranlar elde edilmiştir.

Veri setini train ve test olarak ayırdığımızda; train kümesinde eğitim vede test kümesinde test edilmiştir. Karar ağaçları yüzde 84, geri yayılım algoritması yüzde 68 ve rastgele orman algoritması yüzde 86'lık oranlar elde edilmiştir. Çapraz doğrulamaya göre daha düşük bir oran elde edilmiştir. Çapraz doğrulamada aynı küme üzerinde test edilmesi oranı yükselten sebeplerden sayılabilir.

7.4. Özellik seçimi

Sınıflandırıcıya verilecek özellik sayısı, tüm bileşimleri denemeye elverişli değilse, sınıflandırıcının işini kolaylaştıracak bir ön işleme gerek duyulur. Böylelikle sınıf özelliğini etkilemeyen özellikler ayıklanabilir. Özelliklerin boyut sayısı arttığı zaman, sınıflandırma için boyut laneti devreye girebilir. Bu durum kötü etki yapar. Bu sebeplerden dolayı özellik seçme yöntemleri kullanılır. Weka programı, özellik seçimi için kullanılan sınıfları bir araya getirmiştir. Özellik seçimi yöntemleri içerisinden InfoGainAttributeEval sınıfı esas alınarak parametre seçim işlemi yapılmıştır. InfoGainAttributeEval sınıfı, bir özelliğin bilgi kazancının ölçülmesi ile sınıflandırmaya katkısını değerlendirir. Bu sınıf ile birlikte birkaç sınıfta incelenecektir.

InfoGainAttributeEval: Sınıfa göre bilgi kazancını ölçerek nitelikleri ayrı ayrı değerlendirir.

OneRAttributeEval: OneR sınıflandırıcıyı kullanarak bir niteliğin değerini değerlendirir.

SymmetricalUncertAttributeEval: Sınıfla ilgili simetrik belirsizliği ölçerek bir özellik değerini değerlendirir.

GainRatioAttributeEval: Bir sınıfın kazanç oranını ölçerek bir özellik değerini değerlendirir.

ReliefFAttributeEval: Bir örneği tekrar tekrar örnekleyerek ve aynı ve farklı sınıfın en yakın örneği için verilen özniteliğin değerini göz önüne alarak bir öznitelik değerini değerlendirir. Ayrık ve sürekli sınıf verileri üzerinde çalışabilir.

#	özellik (attribute)	derece	#	özellik (attribute)	derece
1	YAŞ	0.161	8	AKTİF ENDO	0.014
2	CİNSİYET	0.044	9	KRİTİK PREOPE	0.036
3	AKCİ.HASTALIĞI	0.010	10	DİABET	0.040
4	KARDİ.ARTİRİO	0.040	11	LV DİSFONKSİYON	0.057
5	KARDİ.OPERAS.	0.083	12	HİPERTANSİYON	0.035
6	BÖB.BOZUKLUĞU	0.026	13	AORT CERRAHİSİ	0.053
7	BÖB.YETMEZLİĞİ	0.080	14	POST Mİ VSD	0.034

Tablo 15: InfoGainAttributeEval ile özelliklerin dereceleri.

#	Özellik (attribute)	InfoGain Attribute Eval	OneRAttribute Eval	SymmetricalUncertAttribute	Gain Ratio Attribute Eval	ReliefF Attribute Eval	Average
1	YAŞ	1	1	1	7	1	2.2
2	CİNSİYET	6	9	10	13	12	8.0
3	AKCİ.HASTALIĞI	13	14	13	14	11	13
4	KARDİ.ARTİRİO	7	7	8	11	8	10.2
5	KARDİ.OPERAS.	2	2	3	4	3	2.8
6	BÖB.BOZUKLUĞU	12	12	12	9	13	11.6
7	BÖB.YETMEZLİĞİ	3	3	2	1	2	2.2
8	AKTİF ENDO	14	13	14	5	14	12.0
9	KRİTİK PREOPE	9	10	9	6	7	8.0
10	DİABET	8	5	7	8	9	7.4
11	LV DİSFONKSİYON	4	4	5	12	5	6.0
12	HİPERTANSİYON	10	8	11	10	10	9.8
13	AORT CERRAHİSİ	5	6	4	3	4	4.4
14	POST Mİ VSD	11	11	6	2	6	7.2

Tablo 16: Özelliklerin farklı seçim yöntemlerine göre sıralanması ve ortalaması.

7.4.1. Kronik akciğer hastalığı özelliği çıkarıldığında bulgular

	Çapraz Doğrulama (k= 10)	Test Seti
Karar Ağaçları	87.31 %	80.65 %
Yapay Sinir Ağları	70.70 %	69.72 %
Rastgele Orman	89.61 %	88.72 %

Tablo 17: Akciğer hastalığı çıkarıldığındaki doğruluk oranları.

Tablo 16'da görüldüğü üzere kronik akciğer hastalığı özelliği, özellik seçmek için kullanılan algoritmalara göre puanı en yüksek yani sınıf özelliğini en az etkileyen özellik seçilmiştir. Bu sebepten dolayı bu özelliği veri setinden çıkarıp sınıflandırma işlemlerini baştan yapılmıştır.

Amaç doğruluk oranında aşağı ya da yukarı bir hareket sağlamaktır. Karar ağaçları 7-8 puan, geri yayılım algoritması ortalama 5 puan ve rastgele orman algoritması 5-6 puan düşük oranlar elde edilmiştir.

7.4.2. Aktif endo özelliği çıkarıldığında bulgular

	Çapraz Doğrulama (k= 10)	Test Seti
Karar Ağaçları	93.50 %	87.45 %
Yapay Sinir Ağları	59.70 %	71.88 %
Rastgele Orman	95.17 %	89.01 %

Tablo 18: Aktif endo özelliği çıkarıldığındaki doğruluk oranları.

Tablo 16'da görüldüğü üzere aktif endo özelliği, özellik seçmek için kullanılan algoritmalara göre en sondan ikinci özelliktir. Bu sebepten dolayı bu özelliği veri setinden çıkarıp sınıflandırma işlemlerini baştan yapılmıştır. Amaç doğruluk oranında aşağı ya da yukarı bir hareket sağlamaktır. Karar ağaçları 3-5 puan, geri yayılım algoritması ortalama 5 puan ve rastgele orman algoritması 2-3 puan düşük oranlar elde edilmiştir.

7.4.3. Böbrek bozukluğu özelliği çıkarıldığında bulgular

	Çapraz Doğrulama (k= 10)	Test Seti
Karar Ağaçları	92.21 %	86.45 %
Yapay Sinir Ağları	58.10 %	70.78 %
Rastgele Orman	94.12 %	89.81 %

Tablo 19: Böbrek bozuluğu özelliği çıkarıldığındaki doğruluk oranları.

Böbrek bozukluğu özelliği, özellik seçmek için kullanılan algoritmalara göre en sondan üçüncü özelliktir. Bu sebepten dolayı bu özelliği veri setinden çıkarıp sınıflandırma işlemlerini baştan yapılmıştır. Karar ağaçları 3-5 puan, geri yayılım algoritması ortalama 5 puan ve rastgele orman algoritması 2-3 puan düşük oranlar elde edilmiştir.

7.4.4. Akciğer hastalığı ve aktif endo özelliğini birlikte çıkarıldığında bulgular

	Çapraz Doğrulama (k= 10)	Test Seti
Karar Ağaçları	86.84 %	76.45 %
Yapay Sinir Ağları	60.10 %	70.33 %
Rastgele Orman	89.47%	85.91 %

Tablo 20: Akciğer ve aktif endo özellikleri çıkarıldığındaki doğruluk oranları.

Tablo 16'ya bakıldığında sondan ikinci sırada bulanan akciğer hastalığı ve aktif endo özelliği çıkarılmıştır. Sınıflandırma işlemleri baştan yapılmıştır. Rastgele orman algoritması en iyi oranı vermiştir. Karar ağaçları 5-6 puan, geri yayılım algoritması ortalama 5 puan ve rastgele orman algoritması 3-5 puan düşük oranlar elde edilmiştir.

7.4.4. Akciğer hastalığı, aktif endo, böbrek bozukluğu özelliğini birlikte çıkarıldığında bulgular

	Çapraz Doğrulama (k= 10)	Test Seti
Karar Ağaçları	83.82 %	75.15 %
Yapay Sinir Ağları	58.66 %	69.37 %
Rastgele Orman	85.35%	84.92 %

Tablo 21: Akciğer, aktif endo ve böbrek bozukluğu özellikleri çıkarıldığındaki doğruluk oranları.

Sondan üç sırada bulanan akciğer hastalığı, böbrek yetmezliği ve aktif endo özelliği çıkarılmıştır. Amaç doğruluk oranında aşağı ya da yukarı bir hareket sağlamaktır. Sınıflandırma işlemleri baştan yapılmıştır. Rastgele orman algoritması en iyi oranı vermiştir. Karar ağaçları 10-15 puan, geri yayılım algoritması ortalama 5 puan ve rastgele orman algoritması 10-12 puan düşük oranlar elde edilmiştir.

8. MATLAB İLE VERİ SETİNİN SINIFLANDIRILMASI

8.1. Yapay sinir ağları (Backpropagation)

Matlab dosyası, hastalar.mat verileri içerir. Dosya daki değişken veriler, 14x3543' lük bir matristir.

- Verileri rasgele iki gruba ayrılır: eğitim seti ve test seti. Ağ eğitiminde 2000 örnek, test için kalan 1543 örnek kullanılmıştır.
- Giriş katındaki nöronların sayısı 14 (artı bir önyargı nöronu) olmalıdır.
- Aktivasyon fonksiyonu olarak hiperbolik tanjant fonksiyonun kullanılmıştır.
- Çıktı katmanında 1 nöron kullanılmıştır.
- Eğitim setini kullanarak ağı test edilir. Ortalama hata hesaplanır. Her örnek için, maksimum değeri veren çıkış nöronunu bulunur ve örneğe, o nörona karşılık gelen rakamı atanır. Kaç örneğin doğru tahmin edildiği bulunur.
- Test setini kullanarak ağı test edilir. Ortalama hata hesaplanır. Her örnek için, maksimum değeri veren çıkış nöronunu bulunur ve örneğe, o nörona karşılık gelen rakamı atanır. Test stinde kaç örneğin doğru tahmin edildiği bulunur.
- Gizli katmandaki nöronların sayısı ve öğrenme oranı için farklı seçenekler kullanılır. Örneğin, H = 10; 20; 30 nöron gizli katmanda ve mu = 0; 0,2; 2 öğrenim katsayıları kullanılır. Her kombinasyon için sonuçlar ayrı ayrı hesaplanır.

8.1.1. Algoritma

Input: *input size, output size, hidden neuron size, learning rate, traindata, traindesired, maximum epoch, MSE target value*

initiate randomly input–to–hidden weight matrix=>

Wx=***random***(*hidden neuron size,input size+bias*)–0.5

initiate randomly hidden–to–output weight matrix=>

Wy=***random***(*output size,hidden neuron size+bias*)–0.5

for each *iteration*

for each X^i_{input} ***in*** *traindata*

$$net^i_{hidden} = W_x * X^i_{input}$$

$$out^i_{hidden} = \tanh(net^i_{hidden})$$

$$out^i_{hidden+bias} = out^i_{hidden}\ append\ bias$$

$$net^i_{output} = W_y * out^i_{hidden+bias}$$

$$out^i_{output} = \tanh(net^i_{output})$$

$$Error^i_{output} = \sum(traindesired^i_{output} - out^i_{output})^2$$

$$\frac{\partial_{out_y}}{\partial_{net_y}} = (1 - (out^i_{output})^2)$$

$$\frac{\partial_{E_{total}}}{\partial_{out_y}} = (traindesired^i_{output} - out^i_{output})$$

$$\delta_{hidden_y} = \frac{\partial_{out_y}}{\partial_{net_y}} * \frac{\partial_{E_{total}}}{\partial_{out_y}}$$

$$\Delta W_y = -1 * learning\ rate * \delta_{hidden} * out^i_{hidden+bias}$$

$$Update\ hidden - to - output\ weights\ W_y = W_y - \Delta W_y$$

$$\frac{\partial_{out_x}}{\partial_{net_x}} = (1 - (out^i_{hidden+bias})^2)$$

$$remove\ bias\ from\ \delta_{hidden_y}$$

$$\delta_{input_x} = \frac{\partial_{out_x}}{\partial_{net_x}} * W_y * \delta_{hidden_y}$$

$$\Delta W_x = -1 * learning\ rate * \delta_{input_x} * X^i_{input}$$

$$Update\ input - to - hidden\ weights\ W_x = W_x - \Delta W_x$$

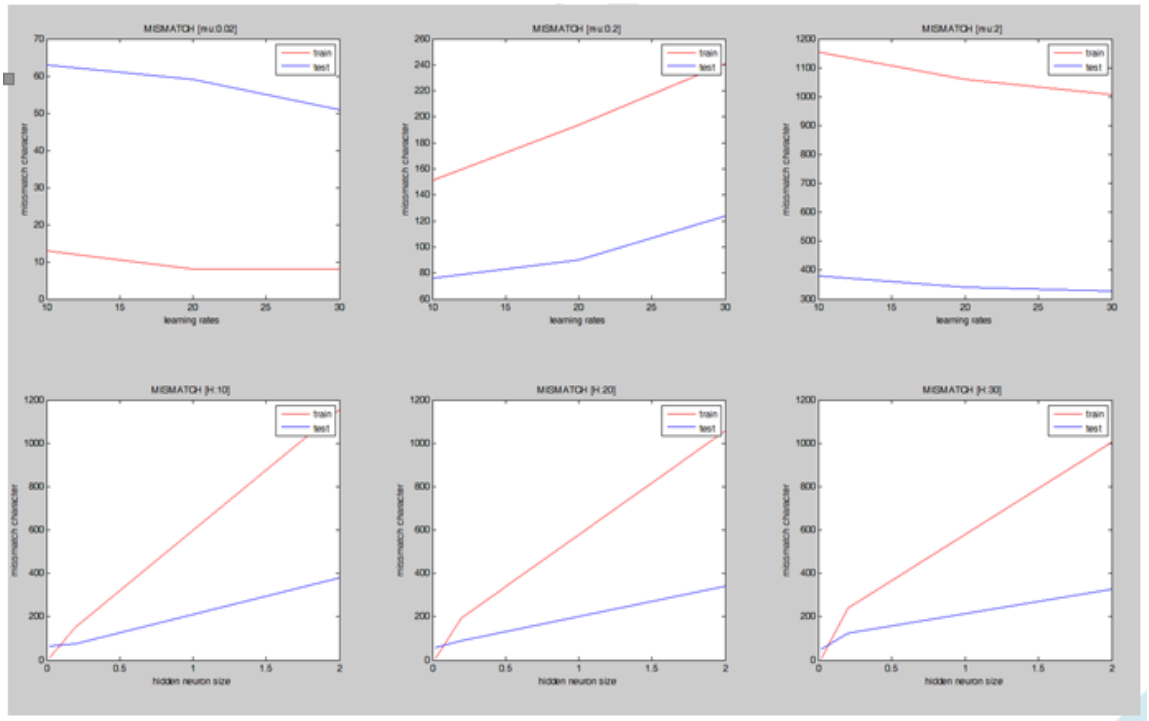
end for

$$if\ \boldsymbol{MSE} = \boldsymbol{mean}(Error^{all}_{output}) < MSE\ target\ value$$

break

end for *return* $W_x\ W_y$ *with MSE*

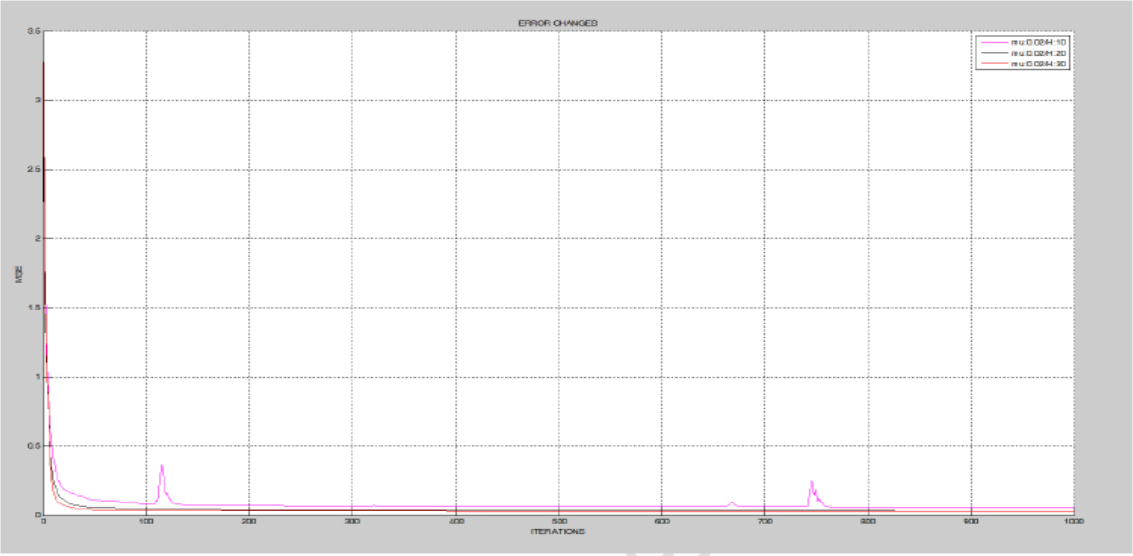
8.1.2. Bulgular (Backpropagation)

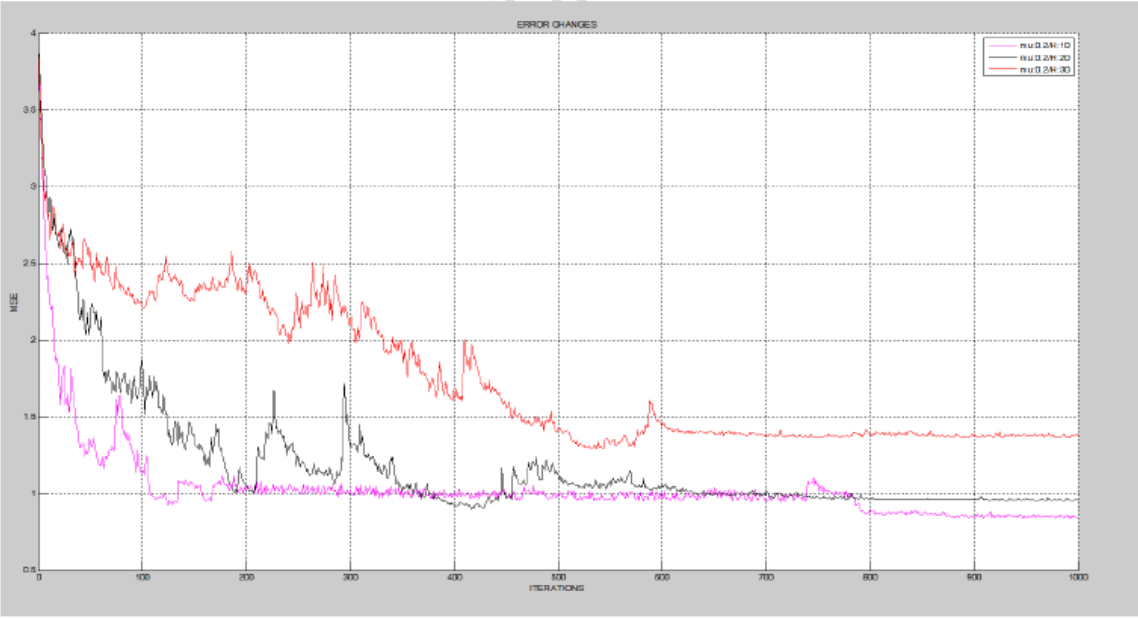


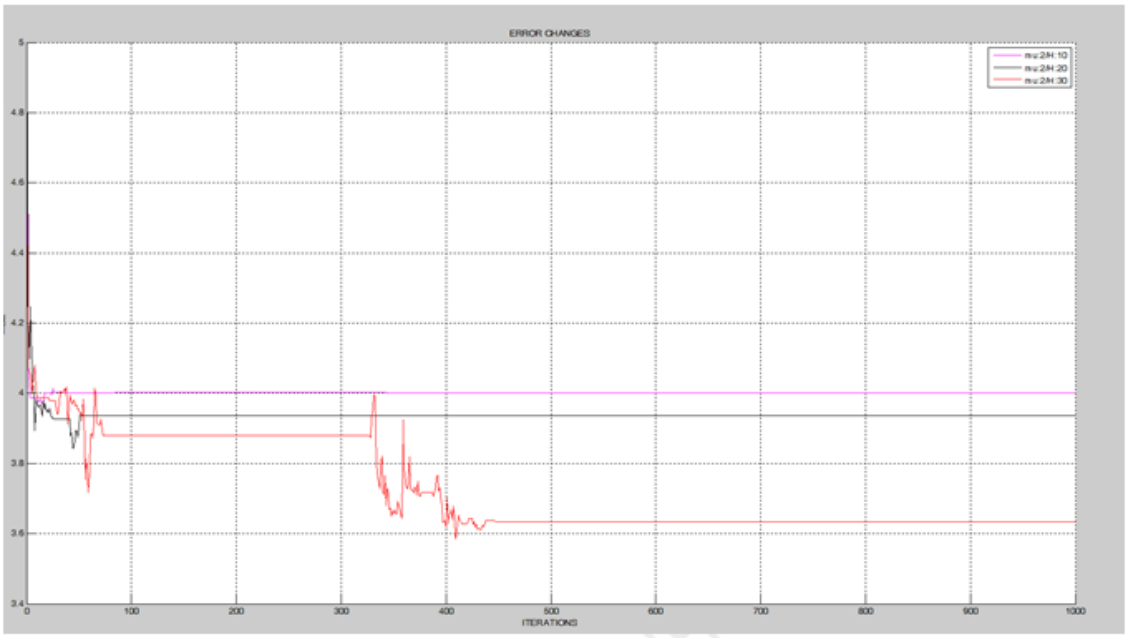
Şekil 19: Herbir iterasyonda öğrenme katsayısı ve gizli katman uyuşmayan değerleri.

Yukarıdaki grafik matlab kodunun çalıştırıldıktan sonraki bazı çıktılarıdır. Şekil 20'de öğrenme katsayısının (0.02, 0.2, 2), gizli katmandaki (10, 20, 30) herbir iterasyondaki değişimi göstermektedir.

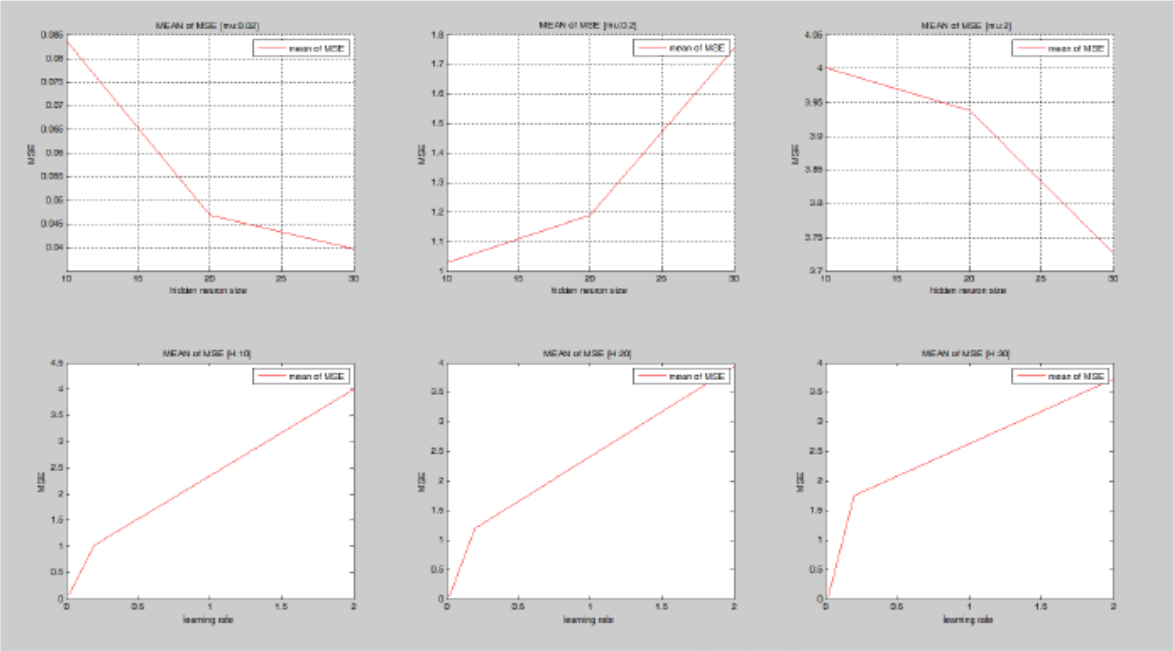
Şekil 19'da herbir iterasyonda öğrenme katsayısı ve gizli katmandaki uyuşmayan değerleri gösterir. Şekil 21'de öğrenme katsayısı ve gizli katmana göre hata değişimi gösterilmiştir.







Şekil 20: Öğrenme katsayısının(0.02,0.2,2) ,gizli katman (10,20,30) herbir iterasyondaki değişimi.



Şekil 21: Öğrenme katsayısı ve gizli katmana göre hata değişimi.

	Çapraz Doğrulama	Test Seti
Doğruluk Oranı	63.800 %	71.72 %

Tablo 22: Geri yayılım algoritması bulguları.

Geri yayılım algoritması ile doğruluk oranları çapraz doğrulama ile yüzde 63, test kümesi ile test edildiğinde yüzde 71 gibi değerler almışlardır. Veka ile yapılan modellemeyle aralarında bir iki puan oynadığı gözükmektedir.

8.2. Rastgele Orman

Matlab programı ile Ek 2' deki program çalıştırıldığında, aşağıdaki grafik ve tablolar bu programın çıktılarından bazılarıdır.

8.2.1. Bulgular

	Çapraz Doğrulama (k=10)	Test Seti
Doğruluk Oranı	92.80 %	87.72 %

Tablo 23: Rastgele orman bulguları.

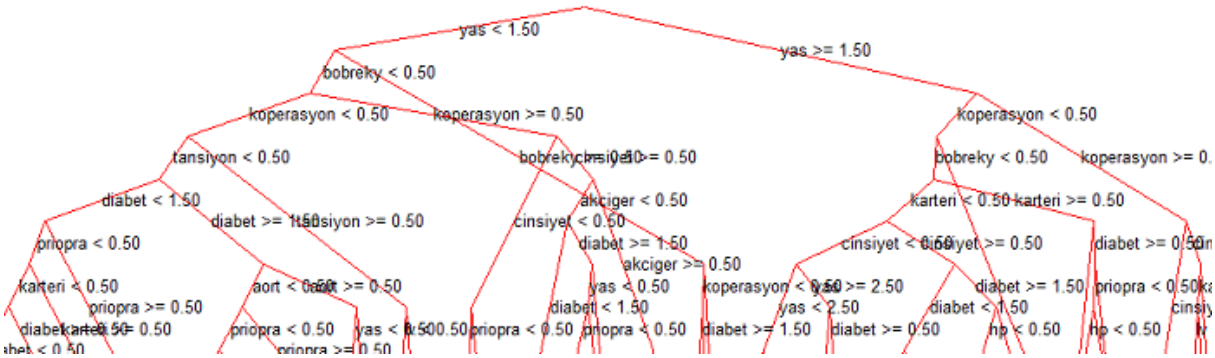
Şekil 21'de bütün özeliklerin önem sırasına göre bir histogram grafiği gösterilmiştir. Rastgele orman algoritması ile doğruluk oranları çapraz doğrulama ile yüzde 92, test kümesi ile test edildiğinde yüzde 87 gibi değerler almışlardır.

Veka ile yapılan modellemeyle aralarında bir iki puan oynadığı gözükmektedir. Matlab ile yapılan modellemede de en iyi sonuçları rastgele orman algoritması vermektedir.

8.3. Karar ağaçları

Matlab programı ile Ek 3' deki program çalıştırıldığında, aşağıdaki grafik ve tablolar bu programın çıktılarından bazılarıdır.

8.3.1. Bulgular



Şekil 22: Karar ağacı.

	Çapraz Doğrulama (k=10)	Test Seti
Doğruluk Oranı	94.50 %	89.72 %

Tablo 24: Karar ağaçları bulguları.

Şekil 22'de bütün veri setinin eğitildikten sonraki ağaç görünümü gösterilmiştir. Karar ağaçları algoritması ile doğruluk oranları çapraz doğrulama ile yüzde 94, test kümesi ile test edildiğinde yüzde 89 gibi değerler almışlardır.

Veka ile yapılan modellemeyle aralarında bir iki puan oynadığı gözükmektedir. Matlab ile yapılan modellemede de en iyi sonuçları rastgele orman algoritması vermektedir.

8.4. Özellik seçimi

Matlab programı ile Ek 4' deki program çalıştırıldığında, aşağıdaki tablo bu programın çıktılarından bazılarıdır.

8.4.1. Bulgular

Özelliklerin Sıralanması:

- 0.2787 7 bobrek_yetmezligi

- 0.2348 14 post_mi_vsd
- 0.2142 13 torasik_aorta_cerrahisi
- 0.1537   5 gecirilmis_kardiyak_operasyon
- 0.1473   8 aktif_endokardit
- 0.1103   9 kritik_preoperatif_durum
- 0.0909   1 hasta_yas
- 0.0897 10 diabetes_mellitus
- 0.089   6 bobrek_fonksiyon_bozuklugu
- 0.0883 12 pulmoner_hipertansiyon
- 0.0757   4 extrakardiyak_artriopati
- 0.0656 11 lv_disfonksiyonu
- 0.0535   2 hasta_cinsiyet
- 0.0116   3 kronik_akciger_hastaligi

Seçilen Özellikler: 7, 14, 13, 5, 8, 9, 1, 10, 6, 12, 4, 11, 2, 3 :14

Yukarıdaki bulgular matlab kodunun çıktılarıdır. Bu kod, özelliklerin sınıf özelliğini en fazla etkileyenden en az etkileyene doğru dizer. Bu durumda 3, 2, 11 sıralı özellikler, sınıf özelliğini en az etkileyen özelliklerdir. Bundan sonra sıra ile bu özellikler veri setinden çıkarılarak, modelleme baştan yapılacaktır. Doğruluk oranında bir değişiklik sağlamak için.

8.4.1.1. Akciğer hastalığı özelliği çıkarıldığında bulgular

	Çapraz Doğrulama (k= 10)	Test Seti
Karar Ağaçları	86.39 %	79.45 %
Yapay Sinir Ağları	59.30 %	69.72 %
Rastgele Orman	88.11%	87.22 %

Tablo 25: Matlab ile akciğer hastalığı özelliği çıkarıldığındaki doğruluk oranları.

Akciğer hastalığı özelliği, özellik seçmek için kullanılan algoritmalara göre en son sıradadır. Yani sınıf özelliğini en az etkileyen özellik seçilmiştir.

Bu sebepten dolayı bu özelliği veri setinden çıkarıp sınıflandırma işlemlerini baştan yapılmıştır.

8.4.1.2. Cinsiyet özelliği çıkarıldığında bulgular

	Çapraz Doğrulama (k= 10)	Test Seti
Karar Ağaçları	87.75 %	85.66 %
Yapay Sinir Ağları	59.00 %	68.77 %
Rastgele Orman	85.61%	81.92%

Tablo 26: Matlab ile cinsiyet özelliği çıkarıldığındaki doğruluk oranları.

Cinsiyet özelliği, özellik seçmek için kullanılan algoritmalara göre en sondan ikinci özelliktir. Bu sebepten dolayı bu özelliği veri setinden çıkarıp sınıflandırma işlemlerini baştan yapılmıştır. Amaç doğruluk oranında aşağı ya da yukarı bir hareket sağlamaktır. Karar ağaçları 5-6 puan, geri yayılım algoritması ortalama 5 puan ve rastgele orman algoritması 6-7puan düşük oranlar elde edilmiştir.

8.4.1.3. Lv disfonksiyon özelliği çıkarıldığında bulgular

	Çapraz Doğrulama (k= 10)	Test Seti
Karar Ağaçları	87.97 %	81.65 %
Yapay Sinir Ağları	61.70 %	69.72 %
Rastgele Orman	88.61%	87.79%

Tablo 27: Matlab ile lv disfonksiyon özelliği çıkarıldığındaki doğruluk oranları.

Lv disfonksiyon özelliği, özellik seçmek için kullanılan algoritmalara göre en sondan üçüncü özelliktir. Karar ağaçları 3-5 puan, geri yayılım algoritması ortalama 10-15 puan ve rastgele orman algoritması 2-3 puan düşük oranlar elde edilmiştir.

8.4.1.4. Akciğer hastalığı ve cinsiyet özllikleri çıkarıldıklarında bulgular

	Çapraz Doğrulama (k= 10)	Test Seti
Karar Ağaçları	83.82 %	79.55 %
Yapay Sinir Ağları	59.30 %	69.82 %
Rastgele Orman	88.11%	87.02%

Tablo 28: Matlab ile akiciğer hastalığı ve cinsiyet özellikleri çıkarıldığındaki doğruluk oranları.

Sondan ikinci sırada bulanan akciğer hastalığı ve cinsiyet özelliği çıkarılmıştır. Amaç doğruluk oranında aşağı ya da yukarı bir hareket sağlamaktır. Sınıflandırma işlemleri baştan yapılmıştır. Rastgele orman algoritması en iyi oranı vermiştir. Karar ağaçları 8-9 puan, geri yayılım algoritması ortalama 10-11 puan ve rastgele orman algoritması 8-9 puan düşük oranlar elde edilmiştir.

8.4.1.5. Akciğer hastalığı, cinsiyet ve lv disfonksiyon özellikleri çıkarıldığında bulgular

	Çapraz Doğrulama (k= 10)	Test Seti
Karar Ağaçları	82.13 %	75.75 %
Yapay Sinir Ağları	58.90 %	68.72 %
Rastgele Orman	85.11%	85.92%

Tablo 29: Matlab ile akiciğer hastalığı, lv ve cinsiyet özellikleri çıkarıldığındaki doğruluk oranları.

Sondan üç sırada bulanan akciğer hastalığı, cinsiyet ve lv disfonksiyonu özelliği çıkarılmıştır. Amaç doğruluk oranında aşağı ya da yukarı bir hareket sağlamaktır. Sınıflandırma işlemleri baştan yapılmıştır. Rastgele orman algoritması en iyi oranı vermiştir. Karar ağaçları 10-15 puan, geri yayılım algoritması ortalama 5 puan ve rastgele orman algoritması 10-12 puan düşük oranlar elde edilmiştir.

9. SONUÇLAR

Bu çalışmada, kalp ameliyatı olmuş kalp ve damar hastalarının yaşam riskini sağlayabilecek bir model oluşturmak hedeflenmiştir. Kalp damar hastalıklarının önemine Bölüm 4'te değinmiştik. Bu sebepten dolayı kalp ameliyatı olan ya da olmak için yatan hastaların yaşam riskini hesaplayabilmek çok önemlidir. Böyle bir durumda karar verme aşamasında hastaya çok yadımcı olacaktır. Aynı şekilde doktorlarda kendi aralarında yorumlar yapacaktırlar.

Kalp ameliyatlarında riskin tespit edilebilmesi için çok sayıda risk hesaplama sistemleri vardır. Bunların arından EuroSCORE metodu Avrupa'da kullanılan bir metottur. Ve başka ülkelerde de tercih edilir. Ülkemizde ise EuroSCORE metodu kullanılmasına rağmen biraz azınlıkta kalmaktadır. Türkiye'de Sosyal Güvenlik Kurumu'na ait bir risk hesaplama metodu bulunmaktadır. Bunun adı Kardiyak Risk Puanlamasıdır. Ve genelde Türk hastanelerinde bu sistem kullanılır. Bu risk hesabının sonuçlarına göre hastaların, masraflarının ne kadarını SGK'nın

karşılacağıda belirlenmiş olur. Bu araştırmada da hastaların yaşam riskini belirlemek için SGK'nın risk hesaplama sistemi kullanılmıştır.

Tez kapsamında hayati riski tahmin edebilmek için Yapay Sinir Ağları (Backpropagaiton), Karar Ağaçları Algoritması ve Rastgele Orman Algoritmaları'ndan faydalanılmıştır. Veri setindeki verinin kategorik yapısından dolayı bu algoritmalar uygun görülmüştür.

En iyi algoritma modelleme ve testten sonra Rastgele Orman'dır. Çapraz doğrulama ile 95.23% oranı ile birinci sıradadır. Aynı şekilde test set ve eğitim seti olarak modellendiğinde de Rastgele Orman algoritması 90.22% en iyi algoritmadır.

Doğruluk oranını değiştirebilmek için yaptığımız değişiklerde ise şu şekil de bir sonuç alınmıştır:

- Yaş özeliği 0, 1, 2 gibi kategorik yapıdan 0'dan başlayıp sürekli bir yapıya sokulmuştur. Bu şekilde bir modelleme yapılıp, testi yapaıldığında sonuçları değişikliğe uğratmıştır. Bu değişiklik en iyi algoritmayı 72.33% gibi değerlere düşürmüştür. Diğer algoritmalarda aynı oranda düşüş yaşanmıştır.

- Sınıf özelliğinin düşük risk, orta risk ve yüksek risk gibi kategorik yapıdan, 0'dan başlayıp en yüksek puan olan 40'a kadar bir sürekli yapıya geçirilmiştir. Yapılan değişiklerle sınıflandırıp, test ettiğimizde en iyi algoritmanın doğruluk oranı 86.24% olmuştur. Dolayısı ile diğer algoritmalar da 10 puanlık bir düşüş yaşanmıştır.

Model oluşturma sırasında kullanılacak özelliklerin belirlenmesi aşamasında InfoGainAttributeEval yöntemi seçilmiş olsa da beş farklı yöntem (ClassifierSubsetEval, CfsSubsetEval, FilteredSubsetEval, GainRatioAttributeEval, ReliefFAttributeEval) daha kullanılarak parametreler nitelik seçim işlemine tabi tutulmuş ve çıkan sonuçlardaki parametre sıralamalarının ortalamaları da göz önünde bulundurularak InfoGainAttributeEval yöntemi sonuçları ile karşılaştırılarak en uygun parametre grubu belirlenmiştir:

- Yaş özeliği hangi nitelik seçim yöntemi kullanılırsa kullanılsın, ilk sırada yer almıştır. Bu da riskin hesaplanabilmesi için gereken bilgilerin cinsiyete göre kesinlikle değişiklik gösterdiğini ifade etmektedir.

- En son sıradaki kronik akciğer hastalığının varlığı özelliği, çıkartılıp modelleme yapıldığında, doğruluk oranında değişiklikler görülmüştür. Ortalama 5 puanlık bir düşüş yaşanmıştır.

- Sondan ikinci konumda olan aktif endo özelliği çıkartılıp sınıflandırma yapıldığında ise 1 ila 2 puanlık bir düşüş olmuştur doğruluk oranında.

- Sondan üçüncü özellik olan böbrek bozukluğu özelliği de çıkartılıp yeni bir sınıflandırma yapılmıştır. Bunun sonucun da ise 2 ile 3 puanlık bir düşüş olmuştur.

- Hem kronik akciğer hastalığı, hem de aktif endo özelliği veri settinden çıkarılmıştır. Bu şekilde bir sınıflandırma yapılmıştır. Sonucunda doğruluk oranı 5 ile 6 puan düşmüştür.

- Sıralamada en sondan üç özellik veri setinden çıkarılarak yapılan modelleme de değişikler yaşanmıştır. Bu değişikliler 10 puanlık bir düşüş sağlamıştır.

- Aynı şekilde Matlab ile yapılan öznitelik sıralamasında da bir dizi elde edilmiştir. Seçilen özellikler: 7, 14, 13, 5, 8, 9, 1, 10, 6, 12, 4, 11, 2, 3 şeklindedir. Bu sıralamanın sonunda tekrar en sondan başlayıp üç özellik çıkartılmıştır. Ve tek tek modellenip test edilmiştir.

Doğruluk oranın daki değişim hep aşağıya doğru olduğu için veri setinde hiçbir özellik çıkartılamamıştır.

Gelecekte, hasta biyokimya sonuçları elde edilir edilmez, modelin otomatik olarak uygulanması ile tanı koyma sırasında doktora yardımcı olacak bir yazılım yapılması bu çalışmanın devamı olarak başlatılabilir. Kalp ve damar hastalığında teşhis sırasında kullanılan zorlu yöntemler olmadan bu gibi bir tanı yöntemi sayesinde, hem hastaya kolay tanı ile daha hızlı tedavi uygulanmaya başlanacak, hem de uygulanan teşhis yöntemleri maliyetleri

düşürülerek, hasta, devlet ve sigorta kurumları tarafından teşhis için ödenen giderlerde düşüş yaratılacaktır.

Çalışma, bu alanda yapılabilecek çalışmalara ışık tutması açısından doktor, hasta ve giderlerin düşürülmesi ile birçok farklı kuruma ve hastaya fayda sağlayacak bir potansiyele sahiptir.

KAYNAKÇA

[1] http://www.emrealadag.com/makine-ogrenmesi-nedir.html

[2] https://sebastianraschka.com/faq/docs/evaluate-a-model.html

[3] O. R. Zaine, "Principles of KDD". Ph. D. Thesis (Unpublished). University Of Alberta, Department of Computing Sciences, 1999.

[4] A.S. Koyuncugil, "Bulanık veri madenciliği ve sermaye piyasalarına uygulanması", Doktora tezi (basılmamış), Ankara Üniversitesi, Fen Bilimleri Enstitüsü, 2006.

[5] R. Brachman, T. Anand, "The Process of Knowledge Discovery in Databases: A Human-Centered Approach" Advances in Knowledge Discovery and Data Mining, ed. U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy, AAAI/MIT Press 1996.

[6] P. Cabena, P. Hadjinian, R. Stadler, J. Verhees, A. Zanasi, "Discovering Data Mining: From Concept To Implemantation", Prentice Hall PTR, Upper Saddle River, New Jersey, 195, USA. 1997.

[7] U. Fayyad, G. Piatetsky-Shapiro, P. Symth, P. "From Data Mining to Knowledge Discovery in Databases", AI Magazine, 17(3), 37-54, 1996.

[8] Jacquez, G.M., Grimson, R. & Waller, L.A.,1996. The analysis of disease clusters, part ıı: ıntroduction to 198 techniques. *Infect Control Hosp Epid*,17, ss.385-97

[9] Delen, D., Walker & G., Kadam, A., 2005. Predicting breast cancer survivability:a comparison of three data mining methods. *Artificial Intelligence in Medicine*.34 (2), ss.113-127

[10] Özekes, S., (2006). Tıbbi görüntülemede bilgisayar destekli tespit. *DoktoraTezi*. İstanbul: Marmara Üniversitesi Fen Bilimleri Enstitüsü.

[11] Demirel, B., (2008). Meme kanseri tedavi yöntemlerinin veri madenciliği ile belirlenmesi.*Yüksek Lisans Tezi*. Isparta: Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü.

[12] Patil, B.M., Joshi, R.C., Toshniwal, D. & Biradar, S., 2011. A new approach: role of data mining in prediction of survival of burn patients. *Journal of Medical Systems*. 35 (6), ss. 1531-1542

[13] Doğan, Ş., (2007). Veri madenciliği kullanarak biyokimya verilerinden hastalık teşhisi.*Yüksek Lisans Tezi*. Elazığ: Fırat Üniversitesi Fen Bilimleri Enstitüsü.

[14] Tahminciler, E. (2014). Erythromcin ilacının yan etkilerinin araştırılması üzerine veri madenciliği çalışması.*Yüksek Lisans Tezi*. İstanbul: Okan Üniversitesi Fen Bilimleri Enstitüsü.

[15] Kumar, D. S., Sathyadevi, G. & Sivanesh, S., 2011. Decision support system for medical diagnosis using data mining .*International Journal of Computer Science Issues.* 8 (3), ss.147-153.

[16] http://sbu.saglik.gov.tr/Ekutuphane/kitaplar/biyoistatistik%20(6).pdf.

[17] TUİK Türkiye İstatistikleri. http://www.tuik.gov.tr.

[18] Rothwell, P.M., Coull, A.J., Giles, M.F., Howard, S.C., Silver, L.E., Bull, L.M., Gutnikov, S.A., Edwards, P., Mant, D., Sackley, C.M., Farmer, A., Sandercock, P.A., Dennis, M.S., Warlow, C.P., Bamford, J.M. & Anslow, P., 2004. Change in stroke incidence, mortality, case-fatality, severity, and risk factors in Oxfordshire, UK from 1981 to 2004. *Oxford Vascular Study*. *Lancet*. 363(9425), ss.1925-1933

[19] Thom, T., Haase, N., Rosamond, W., Howard, V.J., Rumsfeld, J., Manolio, T., Zheng, Z.J., Flegal, K., O'Donnell, C., Kittner, S., Lloyd-Jones, D., Goff, D.C.Jr., Hong, Y., Adams, R., Friday, G., Furie, K., Gorelick, P., Kissela, B., Marler, J., Meigs, J., Roger, V., Sidney, S., Sorlie, P., Steinberger, J., Wasserthiel-Smoller, S., Wilson, M. & Wolf, P., 2006. American heart association statistics committee and stroke statistics subcommittee. *Heart Disease and Stroke Statistics*.

[20] Rosengren, A., Perk, J. & Dallongeville, J., 2009. Prevention of cardiovascular disease. *ESC textbook of cardiovascular medicine*. New York: Oxford University Press, ss.403-435.

[21] Gülel, O., 2012. Kardiyovasküler risk faktörleri. *Deneysel ve Klinik Tıp Dergisi - Journal of Experimental and Clinical Medicine*. 29(3), ss.107-116

[22] Turkiyedoktorlari.com.http://www.turkiyedoktorlari.com/hastalik-rehberi/branslar/kalp-damar/kalp-/383-kardiyovaskueler-risk-faktoerleri-.html.

[23] Florence.com.tr, 2014. Kardiyovasküler Hastalıklarda Non İnvaziv Tanı Yöntemleri. http://www.florence.com.tr/non-invaziv-tani-yontemleri.html.

[24] Kültürsay, H., 2011. Kardiyovasküler hastalık riski hesaplama yöntemleri. *Türk Kardiyol Dern Arş - Arch Turk Soc Cardiol*. 39 (4), ss.6-13.

[25] Han, J. & Kamber, M., 2006. *Data mining concepts and techniques second edition*.2. San Francisco: Morgan Kaufmann Publishers

[26] http://www.akademiya payzeka.com

[27] Ergezer, H.; Dikmen M.&Özdemir E. (2003). Yapay Sinir Ağları Ve Tanıma Sistemleri.

[28] Haykin, S. (1999).Neural Networks: A Comprehensive Foundation. New Jersey: Prentice Hall Inc.

[29] http://kod5.org/yapay-sinir-aglari-ysa-nedir/

[30] http://bilgisayarkavramlari.sadievrenseker.com/2012/04/11/karar-agaci

[31] https://prezi.com/zge2tuu6hosr/karar-agaci-analizi/

[32] http://www.isletme.istanbul.edu.tr/surekli_yayinlar/dergiler/nisan2000/1.htm

[33] https://prezi.com/te0anpko7mvz/rastgele-orman-goruntu-segmentasyonu/

[34] http://dataaspirant.com/2017/05/22/random-forest-algorithm-machine-learing/

[35] https://emrealic.wordpress.com/2011/11/24/matlab-ile-karar-agaci-uygulamasi/

[36] https://www.mathworks.com/matlabcentral/fileexchange/31036-random forest

[37] https://www.mathworks.com/matlabcentral/fileexchange/56937-feature-selection-library

EKLER

Ek 1: Geri yayılım algoritması

```
clear; clc; close all;
load('hastalar.mat');
ind=randperm(size(data,2));
train_size =1200; %train set size
input_size = 14; % bit represent a digit
output_size = 1; %1 digit for 0,1,2,3,4,5,6,7,8,9
epoch = 1000; %maximum epoch
mu  = [0.02 0.2 2]; %learning rates
H = [10 20 30]; %hidden neuron sizes
colors = ['m' 'k' 'r' 'g' 'b' 'c'];
target_MSE = 0.02;
%% Result Reports
n = size(mu,2);%learning rate vector size
m = size(H,2);%hidden neuron size vector size
mseMeans = zeros(n,m);%mseMean vector
procTime = zeros(n,m);%processtime vector
trainMissMatch = zeros(n,m);%missmatch count matrix for train data
testMissMatch = zeros(n,m);%missmatch count matrix for test data
%% seperate train and test data
train=data(:,ind(1:train_size)); % randomly select train set with train
size
test=data(:,ind(train_size+1:end)); % get remainder part as test set
%% convert value matrix to -1 and 1 output matrix
true_digits_values = (-1)*ones(output_size,size(data,2)); %create a -1
filled matrix
 for n = 1:size(true_digits,2) %for each true_digits value
     true_digits_values(true_digits(1,n)+1,n) = 1; %set indexed value 1
using true_digits_value as a index
 end
%% seperate train desired and test desired data
train_desired=true_digits_values(:,ind(1:train_size)); %get first part as
train data
test_desired=true_digits_values(:,ind(train_size+1:end)); %get remained
part as test data
%% train network for each combination of learning rate and hidden neuron
size
for i = 1:size(mu,2)
    figure
    grid on
    for j = 1:size(H,2)
            %run backpropagation algorithm and generate weight matrixes
            procTime(i,j)=cputime;%get start time
            [Wx,Wy,MSE]=backprog(input_size,...
                                 output_size,...
                                 H(j),...
                                 mu(i),...
                                 train,...
                                 train_desired,...
                                 epoch,...
                                 target_MSE);
```

```
            procTime(i,j)=cputime-procTime(i,j);%calculate time
difference and store
%% plot error change for each combination on
single figure
            hold on; semilogy(MSE,colors(j),'DisplayName',['mu:'
num2str(mu(i)) '/H:' num2str(H(j))]);
%% store each MSE
            mseMeans(i,j) = mean(MSE);
%% test network with train data and collect missmatch count
            trainMissMatch(i,j) = testMax(train,train_desired,Wx,Wy);
%% test network with test data and collect missmatch count
            testMissMatch(i,j) = testMax(test,test_desired,Wx,Wy);
    end
    legend(gca,'show'); %display error change legend
    title('ERROR CHANGES')
    xlabel('ITERATIONS') % x-axis label
    ylabel('MSE') % y-axis label
end
%% DISPLAY RESULT REPORT
disp('RESULT REPORT');
disp('------------------------------------------------');
disp('Learning Rate : mu ');
disp('Hidden Neuron Size : H ');
% Display Learning Rate / Hidden Neuron Size Pair Statistics like Error
% Mean, Train Missmatch Count , Test Missmatch
for i=1:size(mu,2);
    for j=1:size(H,2)
        disp(['Error Mean [mu:' num2str(mu(i)) '/H:' num2str(H(j)) '] = '
num2str(mseMeans(i,j))]);
        disp(['Time Spent [mu:' num2str(mu(i)) '/H:' num2str(H(j)) '] = '
num2str(procTime(i,j)) ' second.']);
        disp(['Train Mismatch [mu:' num2str(mu(i)) '/H:' num2str(H(j)) ']
Count= [' num2str(trainMissMatch(i,j)) '] Success Rate =[%' num2str(100-
(trainMissMatch(i,j)/size(train,2))*100) ']' ]);
        disp(['Test Mismatch [mu:' num2str(mu(i)) '/H:' num2str(H(j)) ']
Count= [' num2str(testMissMatch(i,j)) '] Success Rate =[%' num2str(100-
(testMissMatch(i,j)/size(test,2))*100) ']' ]);
    end
end
disp('------------------------------------------------');
%% PLOT MISMATCH
figure
contour3(mu,H,mseMeans,30);
figure;
grid on;
figureNum=1;
%plot train/test missmatch change with static neuron size
for i=1:size(mu,2);
    subplot(2,size(H,2),figureNum);
    grid on;
    x = H; %neuron size static
    trainM = trainMissMatch(i,:);
    testM = testMissMatch(i,:);
    plot(x,trainM,'r',x,testM,'b');
    legend('train','test');
```

```
    title(['MISMATCH [mu:' num2str(mu(i)) '] ']);
    xlabel('learning rates') % x-axis label
    ylabel('missmatch character') % y-axis label
    figureNum=figureNum+1;

for j=1:size(H,2)
    subplot(2,size(H,2),figureNum)
    grid on;
    x = mu; %learning rate static
    trainM = trainMissMatch(:,j);
    testM = testMissMatch(:,j);
    plot(x,trainM,'r',x,testM,'b');
    legend('train','test');
    title(['MISMATCH [H:' num2str(H(j)) '] ']);
    xlabel('hidden neuron size') % x-axis label
    ylabel('missmatch character') % y-axis label
    figureNum=figureNum+1;
end
%% PLOT MEAN of MSE
figure
grid on
figureNum=1;
%plot MEAN of MSE with static hidden neuron size for each learning rate
%value

for i=1:size(mu,2);
    subplot(2,size(H,2),figureNum);
    grid on;
    x = H;
    mseMean = mseMeans(i,:);
    hold on
    plot(x,mseMean,'r');
    legend('mean of MSE');
    title(['MEAN of MSE [mu:' num2str(mu(i)) '] ']);
    xlabel('hidden neuron size') % x-axis label
    ylabel('MSE') % y-axis label
    figureNum=figureNum+1;
end
%plot MEAN of MSE with static learning rate for each hidden neuron size
for j=1:size(H,2)
    subplot(2,size(H,2),figureNum);
    grid on;
    x = mu;
    mseMean = mseMeans(:,j);
    plot(x,mseMean,'r');
    legend('mean of MSE');
    title(['MEAN of MSE [H:' num2str(H(j)) '] ']);
    xlabel('learning rate') % x-axis label
    ylabel('MSE') % y-axis label
    figureNum=figureNum+1;
end
```

backprog.m

```
function [ Wx,Wy,MSE ] = backprog(p,m,H,mu,X,D,epochMax,MSETarget)
%Backpropagation Train Algorithm for One Hidden Layer (Ugur.Coruh)
% INPUT PARAMETERS
%   p: Network Input Number
%   m: Network Output Number
%   H: Hidden Neuron Number
%   mu: Learning Rate
%   X: Input Matrix (p x N) p is input and N is train set size
%   D: Desired for Input Matrix.(m x N) m is input and N is train set
size
%   epochMax: maximum epoch
%   MSETarget: target mean square error
%
% OUTPUT PARAMETERS
%   Wx: Input-To-Hidden Weights. Wx  (H x p+1)
%   Wy: Hidden-To-Output Weights. Wy (m x H+1)
%   MSE: Mean square error vector
%% Append -1 input teta bias to end of input matrix
[p1 N] = size(X);
bias = -1;
 X = [X;bias*ones(1,N)];
%% Initiate Random Weights
Wx = rand(H,p+1)-0.5;
Wy = rand(m,H+1)-0.5;
%% Initiate Weight Deltas with Zeros
DWy = zeros(m,H+1);
DWx = zeros(H,p+1);
%% Initiate MSE vectors with zeros for each epoch
MSETemp = zeros(1,epochMax);
%% TRAIN NETWORK
for i=1:epochMax
    Err = zeros(1,N); %initiate error vector for each input/output pair
    for n=1:N
    %% FORWARD-FUNCTIONS BEGIN
    V = Wx*X(:,n);          %Calculate Input-To-Hidden Outputs
    Z = activator(V);       %Activation Function Process
    S = [Z;bias*ones(1,1)]; %Append bias -1 to end of outputs from input
layer
    G = Wy*S;               %Calculate Hiddent-To-Output Outputs
    Y = activator(G);       %Activation Function Process
    E = D(:,n) - Y;         %Calculate output error for this input
(dEtotal/dout)
    Err(n) = sum(E.^2,1);   %Store Sum of This Inputs' Output Error
    %% BACKWARD-FUNCTIONS BEGIN
    %% output-to-hidden
    df = (1-Y.^2);          %tanh derivation (dout/dnet) convert from
out-to-net
    dGy = df.*E;            %calculation of node delta
(dout/dnet)*(dEtotal/dout)
    DWy = -1.*mu*dGy*S';    %weight delta
    Wy = Wy - DWy;          %update Hidden-To-Ouput Weights
    %% hidden-to-input
    df=(1-S.^2);            %tanh derivation
    dGx = df.*(Wy'*dGy);    %calculation of node delta
```

```
    dGx = dGx(1:end-1,:);     %remove bias
    DWx = -1.*mu*dGx*X(:,n)';
      %calculation of weight delta
    Wx = Wx - DWx;%update Input-
    To-Hidden Weights end
    %% CHECK ERROR
     mse = mean(Err);
     MSETemp(i) = mse;
     disp(['epoch = ' num2str(i) ' mse = ' num2str(mse)]);
     if (mse < MSETarget)
         MSE = MSETemp(1:i);
         return
```

testMax.m

```
function [ missMatch ] = testMax(test,test_desired,Wx,Wy)
% Testing Function for Neural Network
% INPUT PARAMETERS
%   test: test input matrix
%   test_desired : test input matrix desired pairs
%   Wx : input-to-hidden weights
%   Wy : hidden-to-output weights
%
% OUTPUT PARAMETERS
%   missMatch: missmatch count

    missMatch = 0;

    %for each test input iterate and test desired outputs
    for k=1:size(test,2)
        X =test(:,k);%get sample input
        [p1 N] = size (X);
        bias = -1;
        X = [X;bias*ones(1,N)];%append bias end of inputs
        V = Wx*X;% calculate input-to-hidden net
        Z = activator(V); % calculate input-to-hidden outputs
        S = [Z;bias*ones(1,N)]; %append bias end of hidden inputs
        G = Wy*S;% calculate hidden-to-output net
        Y = activator(G);% calculate hidden-to-output outputs

        [b,I1]=max(Y); %take maximum value index from output
        [b,I2]=max(test_desired(:,k));%take maximum value index from
desired

        % compare output index with desired output index
         if(I1~=I2)
             missMatch =missMatch+1; %increase missmatch counter and log
to screen
            S =sprintf('[%d] Sample %d, true: %d, estimated
%d',missMatch,k,I2-1,I1-1);
            disp(S);
         end
    end

end
```

activator.m

```
function [ Y ] = activator( O )
% Activation Function for Neural Network
% INPUT PARAMETERS
%   O: Net output
%
% OUTPUT PARAMETERS
%   Y: Activation Function Output
Y = tanh(O);%hyperbolic tangent function
End
```

Ek 2: Rastgele orman algoritması

RF.m

```
clear;clc;close all          %% RASTGELE ORMAN MATLAB KODU [36]

%--------------------------------------------------------------------------
% Load an example dataset provided with matlab

load hastalar.mat;

features=hastalar(:,1:14);
class=hastalar(:,15);

In = features;
Out = class;

%--------------------------------------------------------------------------
% Find capabilities of computer so we can best utilize them.

% Find if gpu is present
ngpus=gpuDeviceCount;
disp([num2str(ngpus) ' GPUs found'])
if ngpus>0
    lgpu=1;
    disp('GPU found')
    useGPU='yes';
else
    lgpu=0;
    disp('No GPU found')
    useGPU='no';
end
% Find number of cores
ncores=feature('numCores');
disp([num2str(ncores) ' cores found'])

% Find number of cpus
import java.lang.*;
r=Runtime.getRuntime;
ncpus=r.availableProcessors;
```

```
ncpus=r.availableProcessors;
disp([num2str(ncpus) ' cpus found'])

if ncpus>1
    useParallel='yes';
else
    useParallel='no';
end

[archstr,maxsize,endian]=computer;
disp([...
    'This is a ' archstr ...
    ' computer that can have up to ' num2str(maxsize) ...
    ' elements in a matlab array and uses ' endian ...

' byte ordering.'...
    ])

% Set up the size of the parallel pool if necessary
npool=ncores;

% Opening parallel pool
if ncpus>1
    tic
    disp('Opening parallel pool')

    % first check if there is a current pool
    poolobj=gcp('nocreate');

    % If there is no pool create one
    if isempty(poolobj)
        command=['parpool(' num2str(npool) ');'];
        disp(command);
        eval(command);
    else
        poolsize=poolobj.NumWorkers;
        disp(['A pool of ' poolsize ' workers already exists.'])
    end

    % Set parallel options
    paroptions = statset('UseParallel',true);
    toc

end

%-------------------------------------------------------------------------
tic
leaf=5;
ntrees=200;
fboot=1;
surrogate='on';
disp('Training the tree bagger')
b = TreeBagger(...
        ntrees,...
        In,Out,...
```

```
        In,Out,...
        'Method','regression',...
        'oobvarimp','on',...
        'surrogate',surrogate,...
        'minleaf',leaf,...
        'FBoot',fboot,...
        'Options',paroptions...
    );
toc

%-------------------------------------------------------------------------
% Estimate Output using tree bagger
disp('Estimate Output using tree bagger')
x=Out;
y=predict(b, In);
name='Bagged Decision Trees Model';

toc

%-------------------------------------------------------------------------
% calculate the training data correlation coefficient
cct=corrcoef(x,y);
cct=cct(2,1);

%-------------------------------------------------------------------------
% Create a scatter Diagram
disp('Create a scatter Diagram')

% plot the 1:1 line
plot(x,x,'LineWidth',3);

hold on
scatter(x,y,'filled');
hold off
grid on

set(gca,'FontSize',18)
xlabel('Actual','FontSize',25)
ylabel('Estimated','FontSize',25)
title(['Training Dataset, R^2=' num2str(cct^2,2)],'FontSize',30)

drawnow

fn='ScatterDiagram';
fnpng=[fn,'.png'];
print('-dpng',fnpng);

%-------------------------------------------------------------------------
% Calculate the relative importance of the input variables
tic
disp('Sorting importance into descending order')
weights=b.OOBPermutedVarDeltaError;
[B,iranked] = sort(weights,'descend');
toc
```

```
%--------------------------------------------------------------------------
disp(['Plotting a horizontal bar graph of sorted labeled weights.'])

%--------------------------------------------------------------------------
figure
barh(weights(iranked),'g');
xlabel('Variable Importance','FontSize',30,'Interpreter','latex');
ylabel('Variable Rank','FontSize',30,'Interpreter','latex');
title(...
    ['Relative Importance of Inputs in estimating Redshift'],...
    'FontSize',17,'Interpreter','latex'...
    );
hold on
barh(weights(iranked(1:10)),'y');
barh(weights(iranked(1:5)),'r');

%--------------------------------------------------------------------------
grid on
xt = get(gca,'XTick');
xt_spacing=unique(diff(xt));
xt_spacing=xt_spacing(1);
yt = get(gca,'YTick');
ylim([0.25 length(weights)+0.75]);
xl=xlim;
xlim([0 2.5*max(weights)]);

%--------------------------------------------------------------------------
% Add text labels to each bar
for ii=1:length(weights)
    text(...
        max([0 weights(iranked(ii))+0.02*max(weights)]),ii,...
        ['Column '
num2str(iranked(ii))],'Interpreter','latex','FontSize',11);
end

%--------------------------------------------------------------------------
set(gca,'FontSize',16)
set(gca,'XTick',0:2*xt_spacing:1.1*max(xl));
set(gca,'YTick',yt);
set(gca,'TickDir','out');
set(gca, 'ydir', 'reverse' )
set(gca,'LineWidth',2);
drawnow

%--------------------------------------------------------------------------
fn='RelativeImportanceInputs';
fnpng=[fn,'.png'];
print('-dpng',fnpng);

%--------------------------------------------------------------------------
% Ploting how weights change with variable rank
disp('Ploting out of bag error versus the number of grown trees')

figure
plot(b.oobError,'LineWidth',2);
xlabel('Number of Trees','FontSize',30)
```

```
drawnow

%-------------------------------------------------------------------------
fn='RelativeImportanceInputs';
fnpng=[fn,'.png'];
print('-dpng',fnpng);

%-------------------------------------------------------------------------
% Ploting how weights change with variable rank
disp('Ploting out of bag error versus the number of grown trees')

figure
plot(b.oobError,'LineWidth',2);
xlabel('Number of Trees','FontSize',30)
ylabel('Out of Bag Error','FontSize',30)
title('Out of Bag Error','FontSize',30)
set(gca,'FontSize',16)
set(gca,'LineWidth',2);
grid on
drawnow
fn='EroorAsFunctionOfForestSize';
fnpng=[fn,'.png'];
print('-dpng',fnpng);

end
```

Ek 3: Karar ağaçları algoritması

```
clc;clear all;        %% KARAR AĞAÇLARI MATLAB KODU [35]
%% Load the auto dat
load hastalar;
M = hastalar;
% We want to predict the first column...
Y = M(:,15);
% ...based on the others
X = M(:,1:14);
cols = {'yas', 'cinsiyet', 'akciger', 'karteri', 'koperasyon','bobrekb',
'bobreky','endokardit', 'priopra', 'hp', 'diabet', 'lv','tansiyon',
'aort','postmi'};

%% Build the decision tree
t = build_tree(X,Y,cols);

%% Display the tree
treeplot(t.p');
title('Decision tree ("**" is an inconsistent node)');
[xs,ys,h,s] = treelayout(t.p');

for i = 2:numel(t.p)
    % Get my coordinate
    my_x = xs(i);
```

```
    my_y = ys(i);

    % Get parent coordinate
    parent_x = xs(t.p(i));
    parent_y = ys(t.p(i));

    % Calculate weight coordinate (midpoint)
    mid_x = (my_x + parent_x)/2;
    mid_y = (my_y + parent_y)/2;

    % Edge label
    text(mid_x,mid_y,t.labels{i-1});

    % Leaf label
    if ~isempty(t.inds{i})
        val = Y(t.inds{i});
        if numel(unique(val))==1
            text(my_x, my_y, sprintf('y=%2.2f\nn=%d', val(1), numel(val)));
        else
            %inconsistent data
            text(my_x, my_y, sprintf('**y=%2.2f\nn=%d', mode(val),
numel(val)));
        end
    end
end

%--------------------------------------------------------------------------
% Find capabilities of computer so we can best utilize them.

% Find if gpu is present
ngpus=gpuDeviceCount;
disp([num2str(ngpus) ' GPUs found'])
if ngpus>0
    lgpu=1;
    disp('GPU found')
    useGPU='yes';
else
    lgpu=0;
    disp('No GPU found')
    useGPU='no';
end
```

Ek 4. Özellik seçimi

ML.m

```
function [features,weights] = MI(features,labels,Q)
%% ÖZNİTELİK SEÇİMİ MATLAB KODU [37]
if nargin <3
    Q = 12;
```

```
end

edges = zeros(size(features,2),Q+1);

% Compute feature-specific quantization bins so that each bin has
approximately equal number of
% samples in the training set
for k = 1:size(features,2)

    minval = min(features(:,k));
    maxval = max(features(:,k));
    if minval==maxval
        continue;
    end

    quantlevels = minval:(maxval-minval)/500:maxval;

    N = histc(features(:,k),quantlevels);

    totsamples = size(features,1);

    N_cum = cumsum(N);

    edges(k,1) = -Inf;

    stepsize = totsamples/Q;

    for j = 1:Q-1
        a = find(N_cum > j.*stepsize,1);
        edges(k,j+1) = quantlevels(a);
    end

    edges(k,j+2) = Inf;
end

% Quantize data according to the obtained bins
S = zeros(size(features));
for k = 1:size(S,2)
    S(:,k) = quantize(features(:,k),edges(k,:))+1;
end
I = zeros(size(features,2),1);
for k = 1:size(features,2)
    I(k) = computeMI(S(:,k),labels,0);
end

% Sort features into descending order

[weights,features] = sort(I,'descend');

%% EOF
```

```
function [I,M,SP] = computeMI(seq1,seq2,lag)

if nargin <3
    lag = 0;
end

if(length(seq1) ~= length(seq2))
    error('Input sequences are of different length');
end


% Count the frequency and probability of each symbol in seq1
lambda1 = max(seq1);
symbol_count1 = zeros(lambda1,1);

for k = 1:lambda1
    symbol_count1(k) = sum(seq1 == k);
end

symbol_prob1 = symbol_count1./sum(symbol_count1)+0.000001;


% Count the frequency and probability of each symbol in seq2
lambda2 = max(seq2);
symbol_count2 = zeros(lambda2,1);

for k = 1:lambda2
    symbol_count2(k) = sum(seq2 == k);
end

symbol_prob2 = symbol_count2./sum(symbol_count2)+0.000001;

% Compute the joint occurrence frequencies of symbol pairs at the given lag

M = zeros(lambda1,lambda2);
if(lag > 0)
    for k = 1:length(seq1)-lag
        loc1 = seq1(k);

        loc2 = seq2(k+lag);

        M(loc1,loc2) = M(loc1,loc2)+1;
    end
else
    for k = abs(lag)+1:length(seq1)
        loc1 = seq1(k);

        loc2 = seq2(k+lag);

        M(loc1,loc2) = M(loc1,loc2)+1;
    end
end

% Product of individual state probabilities as a matrix
SP = symbol_prob1*symbol_prob2';
```

```
% Product of individual state probabilities as a matrix
SP = symbol_prob1*symbol_prob2';

% Pair joint probability
M = M./sum(M(:))+0.000001;

% Compute MI
I = sum(sum(M.*log2(M./SP)));

function y = quantize(x, q)
x = x(:);
nx = length(x);
nq = length(q);
y = sum(repmat(x,1,nq)>repmat(q,nx,1),2);
```

ÖZGEÇMİŞ

Adı Soyadı: Savaş Karanfil

Doğum Yeri ve Yılı: Bursa, 26.07.1974

Medeni Hali: Bekar

Yabancı Dili: İngilizce

E-posta: savaskaranfil@hotmail.com

Eğitim Durumu

Lise Yıldırım Beyazıt Lisesi

Ön Lisans: Kadir Has Üniversitesi, Bilgisayar Programcılığı

Lisans: Anadolu Üniversitesi, İşletme

Yüksek Lisans: İstanbul Kemerburgaz Üniversitesi,

Elektrik ve Bilgisayar Mühendisliği Anabilim Dalı

Mesleki Deneyim

Performans Yazılım Şirketi-Yazılım Uzmanı

Ipsos: Operasyon Raporlama Uzman Yardımcısı (2013-2014)

Yapı Kredi Emeklilik- Veri Ambarı ve İş Zekası Uzmanı (2015-2017)