T.C.
HARRAN ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

ELEKTRİK – ELEKTRONİK MÜHENDİSLİĞİ PROJESİ

REGRESYON

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ

HAZIRLAYAN ÖĞRENCİNİN ADI-SOYADI VE NUMARASI

SABRİ BEREKET - 190507003

DANIŞMAN
Doç. Dr. BUKET SONBAŞ

2022

ŞANLIURFA

İçindekiler

1.GİRİŞ

Günümüzde sıkça ve hayatımızın her noktada kullanılabilmesi gereken makine öğrenimi algoritmaları içinde regresyon modellemeleri hakkında bir araştırma yapıp, nerede yapıldığı, nasıl kullanıldığı, hangi işlerde kullanılmasının bir fark yarattığını inceleyip bunun hakkında yorumlamalar yapmak planlanmaktadır. Regresyon, genellikle yatırım ve finansta kullanılan bağımlı ve bağımsız değişkenler arasındaki bağlantıyı bulmaya ve bağlantının gücünü belirlemeye çalışan bir istatistiksel terimdir. Bu konudaki amacımız regresyon analizinin, bağımlı değişkenin bağımsız değişkenlerdeki değişikliklere göre nasıl değiştiğinin farkındalığını görmektir. Bu şekilde değişkenler arasındaki ilişkiden çıkarım yaparak gelecekteki ilişkinin modellenmesini incelenecektir. İnceleme ve çalışmaların içerisinde yapacak olduğumuz analiz işlemi ile hata payını azaltmak ve verimliliğin artmasını sağlamak gibi bir temel amaca hizmet etmek için gözlemler yapılacaktır.

2. ARAŞTIRMA

2.1. VERİ İŞLEMLERİ

Veriler, şirketlerin bugünkü çalışma şeklini değiştiriyor. Alınacak kararlarından günlük yapılacaklara kadar her şey verilere bağlıdır. Ve bunların hiçbiri, özellikle büyük miktarda farklı kaynaklardan alınmış ham verileri faydalı bilgilere dönüştürmeden mümkün değildir. İşte burada veri manipülasyonu devreye giriyor. Verileri istenen formata ve biçime getirebilir, asıl değinmek istenen noktanın daha kolay fark edilmesini sağlar.

2.1.1. Veri Manipülasyonu Nedir?

Veri manipülasyonu, verileri daha okunabilir ve düzenli hale getirmek için değiştirme işlemine denir. Örneğin, istenen verinin daha hızlı bulunması için verileri alfabetik sıraya dizmek. Veri manipülasyonunun başka örnekleri web siteleri ve borsa olarak verilebilir. Web siteleri trafiği ölçmek, fazla tıklanan sayfaları bulmak için veri manipülasyonundan faydalanırlar. Borsacılar da borsa eğilimlerini tahmin etmek için veri manipülasyonunu kullanır.

2.1.2. Neden Veri Manipülasyonu Kullanılmalı?

İşletmelerin veri manipülasyonu kullanılmasının temel amacı, verileri manipüle ederek eğilimleri tahmin etmektir. Bu sayede müşteri davranışını anlayarak üretkenliği arttırabilir, doğru kullanım elde edilir ise maliyetleri düşürebilirler. Veri manipülasyonu sayesinde elde ettiğimiz veriler daha tutarlı bir şekilde bir arada bulunurlar. Daha önce kullanılmış eski verilere erişilmek istendiği zaman oluşacak vakit kaybı en aza indirilerek zamandan tasarruf sağlanır.

2.1.3. Veri İşleme Dili

Bahsetmiş olduğumuz veri manipülasyonunu gerçekleştirmek için kullandığımız birçok programalama dili vardır. SQL, Excel, Python, Rust gibi programlama dilleri bu tür veri işleme durumlarına etkin olarak kullandığımız programlama dilleridir. Bir veritabanına veri eklemek, çıkarmak ve güncellemek için kullanılan dillerdir. Daha temiz bir analiz için verilerin temizlenmesi ve nizami sıraya getirilmesinde yardımcı rol oynar.

2.1.4. Veri Manipülasyon Araçları Nelerdir?

Veri işleme araçları, verilerin okunurluğunu kolaylaştırmak için bizlere olanak sunar. Örneğin, bir veri işleme aracı ile elimizde bulunan verileri tarih sırasına göre sıralayabiliriz. Veri manipülasyonunun temel amacı, verinin kendisiyle değil, bir diğer veri ile olan ilişkisini değiştirmektir. Veri manipülasyon için kullanılan yaygın işlemler arasında satır ve sütun filtreleme, toplama ve matematiksel formüller bulunur.

2.1.5. Veri Manipülasyon Araçlarının Faydaları

Veri Tutarlılığı: Tutarlı bir veri formatı, verileri düzenlemeyi, okumayı ve analiz etmeyi kolaylaştırır. Verilerin farklı kaynaklar aracılığı ile elde edildiğini düşünürsek, gelen veri için ortak bir format oluşturmak ve bu format üzerinde verileri depolamak tutarlılığa katkıda bulunacaktır. Bu sayede istediğimiz formata getirdiğimiz veriyi okumada ve kullanmada kolaylık sağlayacaktır.

Fazla Veriden Arındırma: Çoğu zaman, kaynak sistemlerden gelen veriler gereksiz, hatalı veya istenmeyen bilgiler içerir. Verileri kullanışlı hale getirmek, kaliteli ve yapılan iş için yararlı verilerin alınması, bir filtreleme işlemi yapılması büyük önem arz etmektedir.

Veri Yorumlama: Birden fazla format ve iş koşulu içeren karmaşık verilerle uğraşırken, manipülasyon olmadan anlamlandırmak neredeyse imkansızdır. Verileri görselleştirmek ve anlaşılmalarını kolay bir hale getirmek konusunda çalışmamız gerekmektedir. Bu sayede kullanıcılar veriyi daha rahat anlayarak daha kolay bir kullanım kazanmış olurlar.[1]

2.1.6. Veri Analizi

Veri analizi, faydalı bilgiler bulma, sonuçları bilgilendirme ve karar vermeyi destekleme amacı ile verileri inceleme, temizleme, dönüştürme ve modelleme işlemidir. Veri analizi, farklı isimler altında çeşitli teknikleri bünyesinde bulunduran, işletme, bilim ve sosyal bilimler gibi farklı alanlarda kullanılan çok çeşitli görünüş ve yaklaşımlara sahiptir. Günümüzün iş dünyasında, veri analizi karar verme işlemlerinin daha bilimsel hale getirilmesine ve işletmelerin daha etkin çalışmalarına yardımcı olmaktadır. Ham verinin toplanması, ayıklanması ve işlenmesi sonucunda yararlı bilgiler bulma, sonuçlara ulaşma ve karar alma süreçlerini destekleyen modelleme süreci olarak tanımlanıyor. İşletmeler için veri analizi; yeni projelerde kararlar, yapılacak yatırımlar, büyüme ya da küçülme gibi kritik karar alma süreçlerinde etkin bir araç olarak kullanılabilecek bilimsel bir yöntem olarak karşımıza çıkıyor. [2]

2.2. VERİ BİLİMİ İÇİN İSTATİSTİK

İstatistik, bilgiyi kanıta dayandırma ihtiyacından ortaya çıkan istatistik yüzyıllardır veri analizinde ve veriyi anlamlandırmada kullanılmaktadır. Değişkenler arasındaki ilişkileri inceleyen, veri setini çeşitli hesaplamalarla analiz edip bulguları özetleyen, örneklemden çıkarılan sonuçların veri kütlesi için genelleştirmeye yarayan bir bilim dalı olarak istatistik çok eski yıllarda ortaya çıkmış ve veri analiz etmede yüzyıllar boyunca kullanılmıştır. İstatistiğin tarihine bakıldığında 16-17. yüzyıllara kadar dayanan çalışmalar olduğu görülmektedir. Ortalama, standart sapma gibi hesaplamalar, değişkenler arasındaki ilişkinin ortaya konması, sonuçların özetlenmesi, geleceğe yönelik tahminde bulunulması gibi çeşitli analizler için istatistikten biliminden yararlanılır. Buna bağlı olarak verinin olduğu her alanda veriyi analiz etmek için istatistikten yararlanılabileceğini söylemek yanlış olmayacaktır. Veri madenciliğinin kullanıldığı alanların hemen hepsinde istatistiksel yöntemler de kullanılabilmektedir. Kullanılacak alandan ziyade veri boyutu ve yapılmak istenen analizin niteliği hangi yöntemlerin kullanılacağı konusunda belirleyicidir. Önemli olan bu noktada veri ile ilgili nasıl bir analiz yapılacağına ve istatistik yöntemlerinden hangilerinin kullanılacağına karar verilmesidir. Veri madenciliği yöntemlerinin birçoğu temelinde istatistiksel yöntemlere dayanır. Başka bir deyişle veri madenciliğinin temelini istatistik oluşturur ve istatistik bilimi olmadan veri madenciliğinden söz etmek mümkün olamaz. Kullanılan analizler incelendiğinde; kümeleme, diskriminant, regresyon, korelasyon analizlerinin veri manipülasyonunda kullanılan istatiksel yöntemler olduğu görülmektedir. [2]

2.2.1 Veri Ön İşleme

Veri madenciliğinde kullanılacak verinin kalitesi sonuçları da etkileyeceğinden kullanılacak verilerin ön işlemden geçirilmesi oldukça önemli olmaktadır. Veri madenciliğinde güvenilirliğin artırılması için, veri ön işleme yapılmalıdır. Aksi halde hatalı girdi verileri bizi hatalı çıktıya götürecektir. Veri ön işleme, çoğu durumlarda yarı otomatik olan ve yukarıda da belirtildiği gibi zaman isteyen bir veri madenciliği aşamasıdır. Verilerin sayısındaki artış ve buna bağlı olarak çok büyük sayıda verilerin ön işlemeden geçirilmesinin gerekliliği, otomatik veri ön işleme için etkin teknikleri önemli hale getirmiştir. Veri ön işleme teknikleri yukarıdaki paragraftan da anlaşılabileceği gibi şu şekilde sıralanabilir:

1. Veri Temizleme 2. Veri Birleştirme 3. Veri Dönüştürme 4. Veri İndirgeme

2.2.2. Veri Temizleme

Veri temizleme, yinelenen veya alakasız gözlemlerin ve yapısal hataların düzeltilmesi, istenmeyen aykırı değerlerin filtrelenmesi ve eksik verilere müdahale gibi işlemleri gerektirmektedir. Yinelenen gözlemler veya alakasız gözlemler dahil olmak üzere istenmeyen gözlemleri veri kümenizden kaldırın. Yinelenen gözlemler en sık veri toplama sırasında gerçekleşir. Birden çok yerden veri kümelerini birleştirdiğinizde, yinelenen veriler oluşturma fırsatları vardır. Yapısal hatalar, verileri ölçtüğünüzde veya aktardığınızda ve garip adlandırma kuralları, yazım hataları veya yanlış büyük harf kullanımı fark ettiğinizde ortaya çıkar. Bu tutarsızlıklar yanlış etiketlenmiş kategorilere veya sınıflara neden olabilir. Çoğu zaman, bir bakışta analiz ettiğiniz verilere uymayan tek seferlik gözlemler olacaktır. Uygunsuz veri girişi gibi bir aykırı değeri kaldırmak için meşru bir nedeniniz varsa, bunu yapmak, üzerinde çalıştığınız verilerin performansına yardımcı olacaktır. Bazen veri içerisinde eksik değerler olabilir ve bu eksik kısımlar yerine 0, NaN, boşluk, NULL, undefined bulunabilir. Eksik verilerin ne şekilde temsil edildiğini bilmek bu çözümleri uygulamada önemlidir. Bu eksik değerli verilerin silinmesi yahut eksik kısımların tahmini değerlerle doldurulması gerekir. Bununla ilgili, problemin cinsine göre birçok farklı yöntem vardır.

2.2.3. Veri Birleştirme

Veri madenciliğinde genellikle farklı veri tabanlarındaki verilerin birleştirilmesi gerekmektedir. Farklı veri tabanlarındaki verilerin tek bir veri tabanında birleştirilmesiyle şema birleştirme hataları oluşur. Şema birleştirme hatalarından kaçınmak için meta veriler kullanılır. Meta veri, veriye ilişkin veridir. Veri birleştirmede önemli bir konu da indirgemedir. Bir değişken, başka bir tablodan türetilmişse fazlalık olabilir. Değişkendeki tutarsızlıklar da sonuçta elde edilen veri kümesinde fazlalıklara neden olabilir. Bu fazlalıklar korelasyon analizi ile araştırılabilir. Eğer bulunan korelasyon katsayısı yüksek bulunuyorsa, değişkenlerden biri veri tabanından çıkarılarak indirgeme yapılır.

2.2.4. Veri Dönüştürme

Veri dönüştürme ile veriler, veri madenciliği için uygun formlara dönüştürülürler. Veri dönüştürme; düzeltme, birleştirme, genelleştirme ve normalleştirme gibi değişik işlemlerden biri veya birkaçını içerebilir.

2.2.5. Veri İndirgeme

Veri indirgeme teknikleri, daha küçük hacimli olarak ve veri kümesinin indirgenmiş bir örneğinin elde edilmesi amacıyla uygulanır. Bu sayede elde edilen indirgenmiş veri kümesine veri madenciliği teknikleri uygulanarak daha etkin sonuçlar elde edilebilir. [3][4]

2.3 VERİ OKURYAZARLIĞI NEDİR?

Veri okuryazarlığı, verileri okuma, çalışma, analiz etme ve tartışma yeteneğini içerir. Veri okuryazarlığı hem günlük hayatımızda hem de iş hayatımızda pozisyonumuzdan bağımsız bir şekilde önemli hale gelmiştir. Herkes makine öğrenimi bilmek zorunda değildir fakat veri okuryazarlığı günümüzde önemini arttıran ve üzerine çalışılıp gelişmemiz gereken bir alandır.

2.3.1. Açık Veri

Özellikle sosyal platformların artması ile şu an dünyada hızla artan bir büyük veri bulunmakta. Tüm dünyada üretilen bu verinin bir ücret karşılığı beklenmeden herkese açık almasına açık veri denir. Açık veriler, herkesin erişebileceği paylaşabileceği verilerdir. Kurumlar, işletmelere ve bireyler belirli faydalar ve farkındalıklar oluşturmak için açık verileri kullanabilirler.

2.3.2. Veri ve Analitik

Devletler, kurum ve kuruluşların bilgilerini şeffaflaştırmaları sayesinde, açık veri miktarları gitgide artmaktadır. Bu artışın sonucunda veri okuryazarlığı önemini gün geçtikçe arttırmaktadır. Veri okuryazarlığı günümüzün olmazsa olmaz özellikleri arasında değerlendirilmektedir. Gartner'ın raporuna göre; Veri ve analitik, günümüz internet dünyasının temel bir parçası olduğundan ve verilerin büyük boyutlara ulaşması, çalışanları en azından veri hakkında konuşma ve anlama becerilerine sahip olmaya zorluyor. Veri anlama ve konuşma artık günlük yaşantının bir parçası oluyor. Veriler ile uğraşan kurum ve kuruluşlar için yetersiz veri okuryazarlığı gelişimin önünde büyük bir engel oluyor. Şu an özellikle online olarak büyük kurum ve kuruluşların veri okuryazarlığı hakkında eğitim verdiğini görebiliriz. Çocuk yaşlarda bile verilen bu eğitimler kâr amacı gütmeyen büyük organizasyonlar tarafından da veriliyor. School of Data'nın 'Veri Okuryazarlığı' nedir kapsamında sorulan sorusu ile yapılan araştırma da aşağıdaki şu bilgilere ulaşılmış;

- Veriye farklı yollardan nasıl ulaşılacağını bilmek.
- Veriye soru sorabilmek ve yanıt alabilmek.
- Veride çıktılar bulabilmek. (Görselleştirme vs.)
- Verinin manipüle edilerek kullanıcının amacına uygun hale getirilmesi.
- İstatistiksel analizleri veri kullanarak yapabilmek. [5]

2.3.3.Veri Görselleştirme

Veri görselleştirme, insan beyninin verileri anlamasını ve idrak etmesini kolaylaştırmak için bilgileri tablo veya grafik gibi görsel bir bağlama çevirme uygulamasıdır. Veri görselleştirmenin temel amacı, büyük veri kümelerinde kalıpları, eğilimleri ve aykırı değerleri tanımlamayı kolaylaştırmaktır. Terim genellikle bilgi grafikleri, bilgi görselleştirme ve istatistiksel grafikler dahil olmak üzere diğerleriyle birbirinin yerine kullanılır.

Veri görselleştirme, veriler toplandıktan, işlendikten ve modellendikten sonra sonuçların çıkarılması için görselleştirilmesi gerektiğini belirten veri bilimi sürecinin adımlarından biridir. Veri görselleştirme aynı zamanda verileri mümkün olan en verimli şekilde tanımlamayı, bulmayı, işlemeyi, biçimlendirmeyi ve iletmeyi amaçlayan bir araçtır. [6]

2.4. PYTHON

Python, son yıllarda dünyanın en çok kullanılan programlama dillerinden biri haline geldi. Makine öğreniminden web siteleri oluşturmaya ve yazılım testine kadar her alanda kullanılabilecek kadar büyük bir alana sahiptir. Dünyanın en popüler programlama dillerinden biri olan Python, Netflix'in tavsiye algoritmasından, sürücüsüz araba otomasyonuna kadar her alanda aktif olarak kullanılmaktadır. Python genel olarak günümüzde veri bilimi, web geliştirme, otomasyon ve test gibi aktif kod kullanılan birçok alanda kullanılmaktadır.

2.4.1 Python Nedir?

Python, popüler olarak web siteleri ve yazılımlar kodlamak, otomasyon ve veri analizi yapmak için kullanılan bir programlama dilidir. Python genel amaçlı bir dildir, yani farklı alanlarda programlar oluşturmak için kullanılabilir. Bu çok yönlülük, başlangıç seviyesindeki dostu olmasıyla birlikte, onu bugün en çok kullanılan programlama dillerinden biri haline getirmiştir. Sytnax olarak diğer programlama dillerinden kolay olması, kullanıcı kitlesinin büyük, aranan

sorun ya da yardımın kolay bulunabilir olması bakımından Python şu an ayrıca başlangıç için çok tavsiye edilen bir programlama dilidir.

2.4.2. Python Neden Bu Kadar Popüler?

Python'ın popüler olmasının birkaç sebebi var. Python'ı geliştiriciler için bu kadar ünlü yapan birkaç özelliği şöyle sıralayabiliriz;

Konuşma diline yakın basit bir sözdizimine sahiptir, bu nedenle okunması ve anlaşılması daha kolaydır. Projeler oluşturmayı ve bunları geliştirmeyi daha hızlı hale getirir. Birçok alanda kullanılması ayrıca ünlü olmasında büyük rol almaktadır. Python, web geliştirmeden makine öğrenimine kadar birçok farklı görev için kullanılabilir. Yeni başlayanlar için uygundur ve giriş seviyesi kodlayıcılar için popüler olmasını sağlar. Günümüzde kodlamaya yeni başlayan insanlara en çok tavsiye edilen dildir. Zor ve karmaşık Syntaxler yerine Python ile başlanarak kodlamanın mantığının daha kolay anlaşıldığı düşünülmektedir.

Açık kaynaktır. Ticari amaçlarla bile kullanımı ve dağıtımı ücretsizdir. Python'ın kütüphaneleri arkasında bulunan açık kaynak geliştiricileri ve birçok farklı alanda kullanıldığı için çok hızlı bir şekilde büyüyor.

2.4.3. Python ile Neler Yapılabilir?

1. Veri analizi ve makine öğrenimi
2. Web Geliştirme
3. Otomasyon veya komut dosyası
4. Yazılım testi ve prototipleme
5. Günlük görevler

Python'un kullanıldığı bu yaygın yollardan bizim kullandığımız veri analizi ve makine öğrenimine yakından bakalım.

2.4.4. Veri Analizi ve Makine Öğrenmesi

Python, veri biliminde bir temel haline geldi. Veri analistlerinin ve diğer programcıların istatistiksel hesaplamalar yapmak, veri görselleştirmeleri oluşturmak, makine öğrenimi algoritmaları oluşturmak, verileri işlemek ve analiz etmek ve verilerle ilgili diğer görevleri tamamlamak için kullanımında büyük kolaylık sağladı. Kullanıcı kitlesinin büyük olması, açık kaynak tarafında birçok işe yarar kütüphaneyi geliştiricilere sunuyor. Python, çizgi ve çubuk grafikler, pasta grafikler, histogramlar ve 3B grafikler gibi çok çeşitli farklı veri

görselleştirmeleri oluşturulmasında büyük kolaylık sağlıyor. Python ayrıca TensorFlow ve Keras gibi kodlayıcıların veri analizi ve makine öğrenimi için daha hızlı ve verimli programlar yazmasını sağlayan bir dizi kütüphaneye sahiptir. [7]

2.5. PYTHON KÜTÜPHANELERİ

2.5.1. Numpy

NumPy, genel amaçlı bir dizi işleme paketidir. Yüksek performanslı çok boyutlu bir dizi nesnesi ve bu dizilerle çalışmak için araçlar sağlar. Python ile bilimsel hesaplama için temel pakettir. Açık kaynaklı bir yazılımdır.

NumPy'nin atası Numeric, ilk olarak Jim Hugunin tarafından diğer birkaç geliştiricinin katkılarıyla oluşturuldu. 2005 yılında Travis Oliphant, rakip Numarray'in özelliklerini kapsamlı değişikliklerle Numeric'e dahil ederek NumPy'yi yarattı. NumPy açık kaynaklı bir yazılımdır ve birçok katkıda bulunanlara sahiptir. NumPy, finansal olarak desteklenen bir NumFOCUS projesidir.

NumPy, açık bilimsel kullanımlarının yanı sıra, genel verilerin verimli, çok boyutlu bir container olarak da kullanılabilir. Rastgele veri türleri, NumPy'nin çok çeşitli veritabanlarıyla sorunsuz ve hızlı bir şekilde bütünleşmesini sağlayan Numpy kullanılarak tanımlanabilir.

NumPy dizileri, listelerin aksine bellekte tek bir sürekli yerde depolanır, böylece işlemler bunlara çok verimli bir şekilde erişebilir ve işleyebilir. Bu davranışa bilgisayar biliminde referans yeri denir. NumPy'nin listelerden daha hızlı olmasının ana nedeni budur. Ayrıca en yeni CPU mimarileriyle çalışmak üzere optimize edilmiştir.

Bu önemli olanlar da dahil olmak üzere çeşitli özellikler içerir:

1. Güçlü bir N boyutlu dizi nesnesi
2. Gelişmiş (yayın) işlevleri
3. C/C++ ve Fortran kodunu entegre etmek için araçlar
4. Faydalı lineer cebir, Fourier dönüşümü ve rasgele sayı yetenekleri [8][9]

2.5.2. Pandas

Pandas, veri analizi için bir Python kütüphanesidir. 2008 yılında Wes McKinney tarafından güçlü ve esnek bir nicel analiz aracına duyulan ihtiyaçtan yola çıkılarak başlatılan pandas, en

popüler Python kütüphanelerinde biri haline geldi. Son derece aktif bir katkıda bulunanlar topluluğuna sahiptir.

Pandas, veri görselleştirme için matplotlib ve matematiksel işlemler için NumPy olmak üzere iki temel Python kitaplığının üzerine inşa edilmiştir. Pandas, bu kitaplıklar üzerinde bir sarmalayıcı görevi görerek, matplotlib'in ve NumPy'nin yöntemlerinin çoğuna daha az kodla erişmenizi sağlar. Örneğin, pandas.plot(), birden çok matplotlib yöntemini tek bir yöntemde birleştirerek birkaç satırda bir grafik çizmenizi sağlar.

Pandas'tan önce çoğu analist, veri toplama ve hazırlama için Python'u kullandı ve ardından iş akışlarının geri kalanı için R gibi daha alana özgü bir dile geçti. Pandas, analitik görevleri kolaylaştıran ve araç değiştirme ihtiyacını ortadan kaldıran verileri depolamak için iki yeni nesne türü tanıttı: liste benzeri bir yapıya sahip olan Seriler ve tablo şeklinde bir yapıya sahip olan DataFrame'ler.

Pandas'ın en önemli özellikleri şunlardır.

- İndeksli DataFrame (veri iskeleti) objeleri ile veri işlemesi yapabilmek.
- Hafızadaki veya farklı türlerde bulunan veriyi okuyabilmek ve yazabilmek için araçlar sağlamak.
- Veri sıralama ve bütünleşik kayıp veri senaryolarına karşı esnek imkanlar sunması.
- Veri setlerinin tekrar boyutlandırılması veya döndürülmesi.
- Etiket bazlı dilimleme, özel indeksleme ve büyük veri setlerini ayrıştırma özelliği.
- Veri iskeletine sütun ekleme veya var olan sütunu çıkarma.
- Veri gruplama özelliği ile ayırma-uygulama-birleştirme uygulamalarının yapılabilmesi.
- Veri setlerinin birleştirilmesi ve birbirine eklenmesi.
- Hiyerarşik eksenleri indeksleme özelliğiyle birlikte çok boyutlu veriden, daha az boyutlu veri elde edilebilmesi.
- Zaman serisi özelliği: Zaman aralığı oluşturma ve sıklık çevrimleri yapma, hareketli aralık istatistik fonksiyonları, tarih öteleme ve geciktirme.
- Veri filtrelemesi yapabilmek. [10][11]

2.5.3. Seaborn

Veri Görselleştirme, elimizde bulunan verinin görselleştirerek daha kolay anlaşılmasını hedeflemektedir. Birkaç örnek vermek gerekirse, verilerle çalışan profesyoneller, yani finansal

analistler, iş analistleri, veri analistleri, veri bilimcileri için yararlı bir araçtır. Seaborn, matplotlib üzerine kurulmuş açık kaynaklı bir Python kütüphanesidir. Veri görselleştirme ve keşifsel veri analizi için kullanılır. Seaborn, veri frameleri ve Pandas kütüphanesi ile uyumlu çalışır. Oluşturulan grafikler de kolayca özelleştirilebilir. Seaborn ile veri görselleştirmenin birkaç avantajına burada değinirsek;

Grafikler, herhangi bir makine öğrenimi veya tahmin projesinde yararlı olan veri eğilimlerini bulmamıza yardımcı olabilir.

Grafikler, teknik bilgisi olmayan kişilere verilerinizi açıklamayı kolaylaştırır.

Görsel olarak anlaşılır ve kolay grafikler, sunumlar ve raporlar okuyucu için çok daha akılda kalır hale getirebilir. [12]

2.5.4. Matplotlib

Matplotlib, Python ve onun uzantısı NumPy için bir cross-platform, veri görselleştirme ve grafik çizim kütüphanesidir. Bu özelliğinden dolayı MATLAB'a rakip ve open source bir alternatifdir. Geliştiriciler, grafikleri GUI uygulamalarına yerleştirmek için matplotlib'in API'lerini de kullanılabilir. Bir Python matplotlib scripti birkaç satır kod yardımı ile görsel veri grafiği oluşturmak için tasarlanmıştır.

Matplotlib komut dosyası katmanı iki API'yi kaplar:

1. Pyplot API, matplotlib.pyplot tarafından tepesinde bulunan Python kod nesnelerinin bir hiyerarşisidir.
2. Pyplot'tan daha fazla esneklikle birleştirilebilen nesnelerin bir OO API koleksiyonu. Bu API, Matplotlib'in arka uç katmanlarına doğrudan erişim sağlar.

2.5.5. Python'da Matplotlib ve Pyplot

Pyplot API, uygun bir MATLAB tarzı durum bilgisi içeren arayüze sahiptir. Aslında matplotlib, başlangıçta MATLAB için bir açık kaynak alternatifi olarak yazılmıştır. OO API ve arayüzü, pyplot'tan daha özelleştirilebilir ve güçlüdür, ancak kullanımı daha zor denebilir.

Birkaç örnek vermek istersek;

1. matplotlib.pyplot.figure: Figure, en üst katmandır. Bir veya fazla eksen dahil olmak üzere bir çizimde görselleştirilen her şeyi içerir.

2. matplotlib.pyplot.axes: Eksenler, bir çizimdeki öğelerin çoğunu içerir: Axis, Tick, Line2D, Text, vb. ve koordinatları ayarlar. Verilerin çizildiği alandır. Eksenler, X Ekseni, Y Ekseni ve muhtemelen bir Z Ekseni de içerir. [13]

2.5.6. Statsmodels

Statsmodels, kullanıcıların verileri keşfetmesine, istatistiksel modelleri tahmin etmesine ve istatistiksel testler gerçekleştirmesine olanak tanıyan bir Python paketidir. Farklı veri türleri ve her tahminci için kapsamlı bir tanımlayıcı istatistik, istatistiksel testler, çizim işlevleri ve sonuç istatistikleri listesi mevcuttur. SciPy'nin istatistik modülünü tamamlar.

Statsmodels, veri analizi, veri bilimi ve istatistiklere yönelik Python bilimsel yığınının bir parçasıdır. Statsmodels, NumPy ve SciPy sayısal kitaplıklarının üzerine inşa edilmiştir, veri işleme için Pandas ile bütünleşir ve R benzeri bir formül arayüzü için Patsy kullanır. Grafiksel fonksiyonlar Matplotlib kütüphanesinden base almaktadır. Statsmodels, diğer Python kitaplıkları için istatistiksel backend sağlar. Statmodels, Modified BSD (3-clause) lisansı altında yayınlanan ücretsiz bir yazılımdır. [14]

2.5.7. Sklearn

Scikit-learn, orta ölçekli supervised ve unsupervised problemler için çok çeşitli son teknoloji makine öğrenme algoritmalarını entegre eden bir Python modülüdür. Bu paket, genel amaçlı üst düzey bir dil kullanarak makine öğrenimini uzman olmayan kişilere sunmaya odaklanır. Kullanım kolaylığı, performans, belgelendirme ve API tutarlılığına vurgu yapılır. Minimum bağımlılıkları vardır ve basitleştirilmiş BSD lisansı altında dağıtılır, bu da hem akademik hem de ticari ortamlarda kullanımını teşvik eder.

Scikit-learn birçok özellikle yüklü olarak gelir. Bunlardan birkaç tanesi:

- Denetimli öğrenme algoritmaları: Genelleştirilmiş doğrusal modellerden (örneğin Doğrusal Regresyon), Destek Vektör Makinelerinden (SVM), Karar Ağaçlarından Bayes yöntemlerine kadar- hepsi scikit-learn araç kutusunun bir parçasıdır. Makine öğrenmesi algoritmalarının yaygınlaşması, scikit-learn kullanımının yüksek olmasının en büyük nedenlerinden biridir.
- Çapraz doğrulama: Sklearn kullanarak görünmeyen veriler üzerinde denetimli modellerin doğruluğunu kontrol etmek için çeşitli yöntemler vardır.

- Denetimsiz öğrenme algoritmaları: Kümeleme, faktör analizi, temel bileşen analizinden denetimsiz sinir ağlarına kadar geniş bir yelpazede makine öğrenmesi algoritmaları vardır.
- Çeşitli oyuncak veri kümeleri: Örneğin, IRIS veri kümesi, Boston House fiyatları veri kümesi.
- Feature extraction: Scikit-learn, resimlerden ve metinden özellik çıkarmak için feature extraction kullanır (ör. Kelime çantası). [15][16]

2.6. MAKİNE ÖĞRENMESİ

Makine öğrenimi, bilgisayarlara açıkça programlanmadan nasıl öğreneceklerini ve hareket edeceklerini öğretmeyi amaçlayan bir bilgisayar bilimi alanıdır. Daha spesifik olarak, makine öğrenimi, programların deneyim yoluyla "öğrenmesine" izin veren modeller oluşturmayı ve uyarlamayı içeren bir veri analizi yaklaşımıdır. Makine öğrenimi, tahmin yapma yeteneklerini geliştirmek için modellerini uyarlayan algoritmaların oluşturulmasını içerir.

2.6.1. Makine Öğrenmesi Nasıl Çalışır?

Bugün makine öğreniminin kullanılmasının birçok yolu var. Makine öğrenimi, bilgisayarlara açıkça programlanmadan öğrenme yeteneği kazandırmayı amaçlayan bir bilgisayar bilimi alanıdır. Bir programın "öğrenmek" için kullandığı yaklaşım veya algoritma, programın tamamlamak üzere tasarlandığı sorunun veya görevin türüne bağlı olacaktır.

Bu nedenle, makine öğreniminin nasıl çalıştığını anlamanın iyi bir yolu, makine öğreniminin ne tür sorunları çözmeye çalıştığını anlamak ve ardından bu sorunları nasıl çözmeye çalıştığına bakmaktır. İlk olarak, makine öğreniminin çözmeyi amaçladığı sorun türlerinin bir listesi:

2.6.2. Makine Öğrenmesi Türleri

Makine öğrenimi algoritmalarının tümü, daha fazla veri kümesi işledikçe doğruluklarını öğrenmeyi ve geliştirmeyi amaçlar. Makine öğrenimi algoritmalarının çözdüğü görevleri sınıflandırmanın bir yolu, sisteme ne kadar geri bildirim sundukları ile ilgilidir. Bazı senaryolarda, bilgisayara denetimli öğrenme adı verilen önemli miktarda etiketlenmiş eğitim verisi sağlanır. Diğer durumlarda, etiketli veri sağlanmaz ve bu denetimsiz öğrenme olarak bilinir. Son olarak, yarı denetimli öğrenmede bazı etiketli eğitim verileri sağlanır, ancak eğitim verilerinin çoğu etiketsizdir.[17]

Denetimli Öğrenme: Denetimli öğrenme, makine öğreniminin en pratik ve yaygın olarak benimsenen biçimidir. Girdi değişkenlerini tercih edilen çıktı değişkenleriyle ilişkilendiren bir matematiksel fonksiyon oluşturmayı içerir. Bilgisayarın işleyeceği verilerin örneklerini sağlayan çok sayıda etiketli eğitim veri kümesi sağlanmıştır.

Yarı Denetimli Öğrenme: Yarı denetimli öğrenme, eğitim sırasında az miktarda etiketlenmiş veriyi büyük miktarda etiketlenmemiş veriyle birleştiren bir makine öğrenimi yaklaşımıdır. Yarı denetimli öğrenme, denetimsiz öğrenme (etiketlenmiş eğitim verileri olmadan) ve denetimli öğrenme (yalnızca etiketlenmiş eğitim verileriyle) arasında yer alır. Zayıf denetimin özel bir örneğidir.

Denetimsiz Öğrenme: Denetimsiz öğrenme problemlerinde tüm girdiler etiketsizdir ve algoritmanın girdilerden kendi başına bir yapı oluşturması gerekir. Kümeleme sorunları (veya küme analizi sorunları), girdi veri kümeleri içindeki gruplandırmaları keşfetmeye çalışan denetimsiz öğrenme görevleridir.

2.6.3. Makine Öğrenimi Algoritmaları ve Problem Çözme Yaklaşımları

Algoritma, bir sorunu çözme yaklaşımıdır ve makine öğrenimi, çok çeşitli sorunları çözmek için birçok farklı yaklaşım sunar. Günümüzde makine öğrenimi uygulamalarında kullanılan en yaygın ve kullanışlı algoritmaların ve yaklaşımların bazıları şunlardır:

Yapay Sinir Ağları: Yapay sinir ağı, insan beyni gibi biyolojik sinir ağlarına dayalı bir hesaplama modelidir. Bir giriş sinyalini veya dosyasını işlemek ve onu birkaç aşamada beklenen çıktıya çevirmek için bir dizi işlev kullanır. Bu yöntem günümüzde genellikle görüntü tanıma, dil çevirisi ve diğer yaygın uygulamalarda kullanılmaktadır.

Derin Öğrenme: Derin öğrenme, yapay sinir ağlarını yoğun bir şekilde kullanan bir makine öğrenme algoritmaları ailesini ifade eder. 2016 Google Tech Talk'ta Jeff Dean, derin öğrenme algoritmalarını çok derin sinir ağları kullanmak olarak tanımlar; burada "derin", katman sayısını veya girdi ile çıktı arasındaki yinelemeleri ifade eder. Bilgi işlem gücü daha ucuz hale geldikçe, günümüz uygulamalarında öğrenme algoritmaları "derin" hale geliyor.

Küme analizi: Bir küme analizi, nesneleri diğer kümelerdeki öğelere göre birbirine daha çok benzeyen öğelerden oluşan "kümeler" halinde gruplandırmaya çalışır. Öğelerin benzerliği, bilgisayar programına sağlanan veri girişlerine bağlıdır. Kümeleme analizleri çoğunlukla denetimsiz öğrenme problemlerinde kullanıldığı için eğitim verilmez. Program, her bir girdi nesnesini tanımlamak için sağlanan veri noktalarını ve değerleri kullanacak. Halihazırda analiz

etmiş olduğu nesneler hakkındaki verilerle karşılaştıracaktır. Veri noktalarındaki ve nesnelerdeki gruplandırmaları tespit etmek için yeterli sayıda nesne analiz edildikten sonra, program nesneleri gruplandırmaya ve kümeleri tanımlamaya başlayabilir.

Bayes Ağları: Bir Bayes ağı, değişkenlerin ve bunların birbirlerine bağımlılıklarının grafiksel bir modelidir. Makine öğrenimi algoritmaları, inanç sistemini oluşturmak ve tanımlamak için bir bayes ağı kullanabilir.

Pekiştirmeli Öğrenme: Pekiştirmeli öğrenme, sisteme sağlanan geri bildirimin açıkça "doğru" veya "yanlış" olarak söylenmek yerine ödül ve ceza şeklinde geldiği bir makine öğrenimi alanını ifade eder. Bu, doğru cevabı bulmak önemli olduğunda devreye girer, ancak zamanında bulmak da önemlidir. Bu nedenle, pekiştirmeli öğrenmenin büyük bir unsuru, "keşif" ve "sömürü" arasında bir denge bulmaktır. Program, halihazırda mevcut olan bilgilerden yararlanmak yerine yeni bilgileri ne sıklıkla "keşfetmelidir"? Program, öğrenme aracısını arzu edilen bir şekilde davranması için "ödüllendirerek", keşif ve kullanım arasında en iyi dengeyi elde etmek için yaklaşımını optimize edebilir.

Karar Ağacı Öğrenmesi: Karar ağacı öğrenimi, bir çıktıya veya cevaba yol açan bir dizi sınıflandırma kullanarak girdileri işleyen bir makine öğrenimi yaklaşımıdır. Tipik olarak bu tür karar ağaçları veya sınıflandırma ağaçları ayrı bir cevap verir; ancak, regresyon ağaçları kullanılarak çıktı sürekli değerler alabilir (genellikle gerçek bir sayı).

En önemli algoritmalardan ve yaklaşımlardan bahsettiğimiz gibi birçok farklı yol daha içermektedir.

2.6.4. Makine Öğrenmesi Tarihçesi

"Makine öğrenimi" terimi ilk olarak 1959'da yapay zekâ ve bilgisayar oyunlarının öncüsü Arthur Samuel tarafından icat edildi. Ancak, Samuel aslında ilk bilgisayar öğrenme programını 1952'de IBM'deyken yazdı. Program, bilgisayarın her birini geliştirdiği bir dama oyunuydu. Oynadığı zaman, hangi hareketlerin kazanan bir strateji oluşturduğunu analiz eder. 1957'de Frank Rosenblatt, insan beyninin düşünce süreçlerini simüle etmek için tasarlanmış, algılayıcı olarak da bilinen ilk yapay bilgisayar sinir ağını yarattı.1967'de, bilgisayarları kullanarak temel örüntü tanımanın başlangıcını işaretleyen "en yakın komşu" algoritması tasarlandı. Bu erken keşifler önemliydi, ancak dönemin kullanışlı uygulamalarının eksikliği ve sınırlı bilgi işlem gücü, 1980'lere kadar makine öğrenimi ve yapay zekada uzun bir durgunluk dönemine yol açtı.

2.6.5. Günümüzde Makine Öğrenmesi

Bugün, makine öğrenimi önemli sayıda uygulamaya yerleştirilmiştir ve her gün milyonlarca insanı etkiler. Makine öğrenimine yönelik muazzam miktarda araştırma, geliştirilen birçok yeni yaklaşımın yanı sıra makine öğrenimi için çeşitli yeni kullanım örneklerinin geliştirilmesiyle sonuçlandı. Gerçekte, makine öğrenimi teknikleri, iş dünyasında yaygın bir ihtiyaç olan büyük miktarda verinin analiz edilmesi gereken her yerde kullanılabilir. Üç ana neden, iş ve araştırma uygulamalarında makine öğreniminin toplu olarak benimsenmesine yol açmıştır:

1. Bilgi işlem gücü son birkaç on yılda önemli ölçüde arttı ve çok daha ucuz hale geldi,
2. Makine öğreniminin güçleri ve kullanım durumları hakkında bilgi internetin yaygınlaşmasıyla yayıldı ve
3. Açık kaynaklı makine öğrenimi araçları daha yaygın olarak kullanılabilir hale geldi.

Biyomedikal, internet/teknoloji, lojistik ve neredeyse diğer tüm sektörlerdeki şirketler, makine öğreniminin müthiş gücünden yararlanıyor. Bugün makine öğreniminin kullanıldığı birkaç örnek:

Google, kullanıcılarının arama sorgularını daha iyi anlamak için makine öğrenimini kullanır.

Google, döndürdüğü sonuçlarla etkileşimi ölçerek sonuçlarını iyileştirmek için makine öğrenimini de kullanır.

Tıbbi araştırma kuruluşları, hastalıklar ve koşullardaki kalıpları belirleme ve sağlık hizmetlerini iyileştirme girişimlerinde muazzam miktarda insan sağlığı kaydı verisini analiz etmek için makine öğrenimini kullanıyor.

Lyft gibi yolculuk paylaşımı uygulamaları, rotaları ve fiyatlandırmayı günün saatine ve konuma göre optimize etmek için makine öğreniminden yararlanır.

E-posta programları, Spam klasörüne neyin ait olduğunu bulmak için makine öğrenimi tekniklerini kullanır.

Bankalar, şüpheli veya hileli olabilecek işlemleri ve davranışları tespit etmek için makine öğrenimini kullanıyor.

Bunlar, günümüzde makine öğrenimi tekniklerinin kullanıldığı binlerce örnekten sadece birkaçıdır. Makine öğrenimi, heyecan verici ve hızla genişleyen bir çalışma alanıdır ve uygulamaları görünüşte sonsuzdur.[18]

2.7. DOĞRUSAL REGRESYON

Doğrusal regresyon bir değişkenin değerini başka bir değişkenin değerine göre tahmin etmek için kullanılır. Tahmin edilen değişken bağımlı değişken olarak sıfatlandırılır. Değişken değerini tahmin etmek için kullandığımız diğer değişkenlere ise bağımsız değişken denir. Bu yapıt bağımlı değişkenin değerini en iyi hesaplayabilecek biri ya da daha fazla bağımsız değişkeni kullanarak doğrusal denklemin katsayılarını tahmin eder. Doğrusal regresyon tahmin edilen ve gerçek parametreler arasındaki farkları en aza indiren düz bir grafik oluşturur.

Doğrusal regresyon modelleri diğer modellere göre basittir ve tahminler üretebilen yorumlanabilir kolay matematiksel formüller oluşturur doğrusal regresyon iş rekabetinde veya akademik ortamlarda ve çok daha fazla alanda kullanılabilir.

Biyolojik, davranışsal, çevresel ve sosyal bilimlerden işletmeye kadar her alanda doğrusal regresyonun kullanıldığını görecebiliriz doğrusal regresyon modelleri bilimsel olarak kanıtlanmış bir yöntem haline gelmiştir. Doğrusal regresyon geleceği güvenilir bir şekilde tahmin etmeye çalışır. Doğrusal regresyon geniş bir istatistiksel prosedür olduğu için, doğrusal regresyon modellerinin özellikleri iyi anlaşılabilir ve çok çabuk eğitebilir.

2.7.1. Regresyon Modellerinin Avantajları ve Dezavantajları

- İyi anlaşılırsa diğer tüm ML ve DL konuları çok rahat kavranır.
- Doğrusallik nedensellik yorumları yapılabilmesini sağlar, bu durum aksiyoner ve stratejik modelleme imkânı verir.
- Değişkenlerin etki düzeyleri ve anlamlılıkları değerlendirilebilir.
- Bağımlı değişkenliğin açıklanma başarısı ölçülebilir.
- Model anlamlılığı değerlendirilebilir.
- Varsayımları vardır.
- Aykırı gözlemlere duyarlıdır.

2.7.1. Regresyonda Model Ayarı

Modeli aşırı takmadan model performanslarını en üst düzeye çıkaracak ve modeldeki varyans hatasını azaltacak şekilde ayarlamak için kullanılır. Modele uygun hiperparametre tekniğini uygulamamız gerekiyor.

2.7.2.1. Regresyonda Hata Türleri

Varyans hatası: Farklı eğitim verilerinin kullanılması durumunda tahmin edilen değerin değişeceği miktarı ifade eder.

Önyargı hatası: Modeli basitleştirmek için yapılan model varsayımlarından kaynaklanan hatadır.

2.7.2.2. Regresyon Düzenleme

Modelin aşırı uydurma doğasını azaltır. Model iyi çalışsa bile, bu, sorunun gelecekte oluşmasını önlemek için yapılır. Bu, daha fazla hata ekleyerek ve modelin daha fazla öğrenmesini sağlayarak yapılır. Bu, modelin daha fazla öğrenmesine yardımcı olacaktır. Sonuç olarak, sonraki aşamada daha fazla veri eklense bile, model bunları sorunsuz bir şekilde işleyebilecektir. Artık model performansı artacak ve düzensiz modelden daha iyi olacaktır. Düzenlileştirme yaptığımızda katsayı küçülür. Alfa'da da çok fazla ayar yaparak modelimizin yetersiz kalmadığından emin olmamız gerekiyor. Alfa bir ceza faktörüdür. Hata, noktaların çoğuna değmeyen bir çizgi çizilerek sisteme girilir. Bu düzenlileştirme modelleri, yeni veriler için fazla değişmeyecek olan eğimi azaltan modeller oluştururken katsayıyı küçültecektir. Katsayıdaki büzülme tamamen değişkenlere bağlıdır. Özellik önemliyse küçülme daha az olacaktır, ancak özellik önemli değilse küçülme daha fazla olacaktır. Eğer özellik çok önemsiz ise katsayı sıfır olur. Bu modellerin regülerize olmasının avantajı, varsayımlar kontrol edilmese bile modelin tüm işi yapmasıdır.

2.7.2.3. Regresyonda Fazla Uydurma

Model eğitim verilerinden tüm karmaşık ve gürültüyü öğrendiğinde ve eğitim verilerinde iyi performans gösterdiğinde ancak doğrulama verilerine geldiğinde iyi çalışmadığında veriler fazla uyum sağlamaktadır.

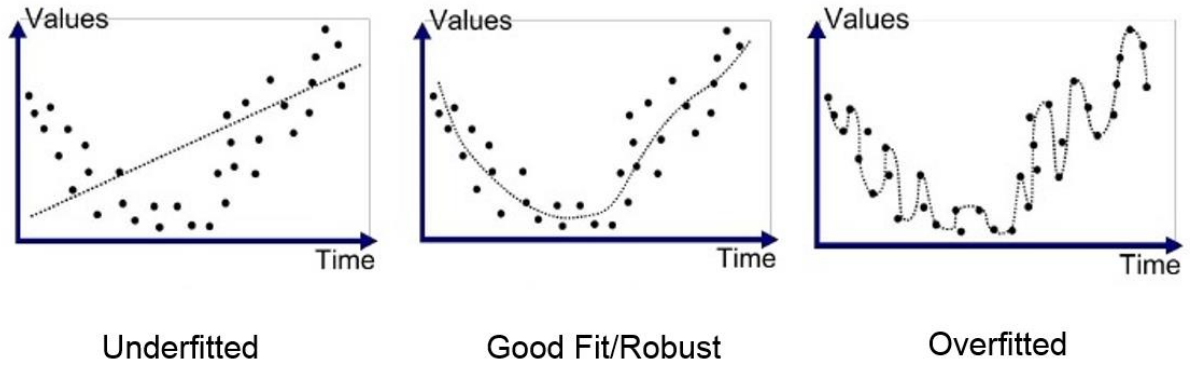
Fazla uyumu sınırlamak için makine öğrenimi algoritmalarını değerlendirirken kullanabileceğiniz iki önemli teknik vardır:

- Model doğruluğunu tahmin etmek için bir yeniden örnekleme tekniği kullanın.
- Bir doğrulama veri kümesini geri tutun.

2.7.2.4. Regresyonda Yetersizlik

Veriler yetersiz kaldığında, model temel trend verilerini öğrenir. Modeli oluşturmak için daha az veriye sahip oldunduğunda veya doğrusal olmayan verilerle doğrusal modeli oluşturmaya çalışıldığında ortaya çıkar.

- Eksik uydurma ne eğitim verilerini modelleyebilen ne de yeni verilere genelleyemeyen bir modeli ifade eder.
- Uygun olmayan bir makine öğrenimi modeli uygun bir model değildir ve eğitim verilerinde düşük performansa sahip olacağından bariz olacaktır.
- İyi bir performans ölçütü verildiğinde tespit edilmesi kolay olduğu için yetersiz donanım genellikle tartışılmaz. Çözüm, devam etmek ve alternatif makine öğrenimi algoritmalarını denemektir. Bununla birlikte, aşırı uyum sorununa iyi bir karşıtlık sağlar.



Şekil 2.1 Regresyon modellenmesinin grafiksel gösterimi [50]

2.7.2.5. Regresyonda Çapraz Doğrulama

Çapraz Doğrulama esasen bir modelin yeni bir bağımsız veri kümesinde ne kadar iyi performans gösterdiğini değerlendirmek için kullanılan bir tekniktir.

Çapraz doğrulamanın en basit örneği, verilerinizi üç gruba ayırmaktır: Train set, test set ve onaylama setidir. Train setini gördüğünüz noktada model kurup hipermetrelere göre uyarlayıp test setiyle değerlendirme işlemidir. [19][20]

2.7.2. Basit Doğrusal Regresyon

Basit Doğrusal Regresyon, iki sürekli (nicel) değişken arasındaki ilişkileri özetlememize ve incelememize izin veren istatistiksel bir yöntemdir. Örnek olarak;

İki değişken arasındaki ilişkinin ne kadar güçlü olduğunu tahmin etmek için (örneğin, yağış ve toprak erozyonu arasındaki ilişki). Bağımsız değişkenin belirli bir değerinde bağımlı değişkenin değerini tahmin etmek için (örneğin, belirli bir yağış seviyesindeki toprak erozyonu miktarı).

2.7.2.1. Basit Doğrusal Regresyon Varsayımları

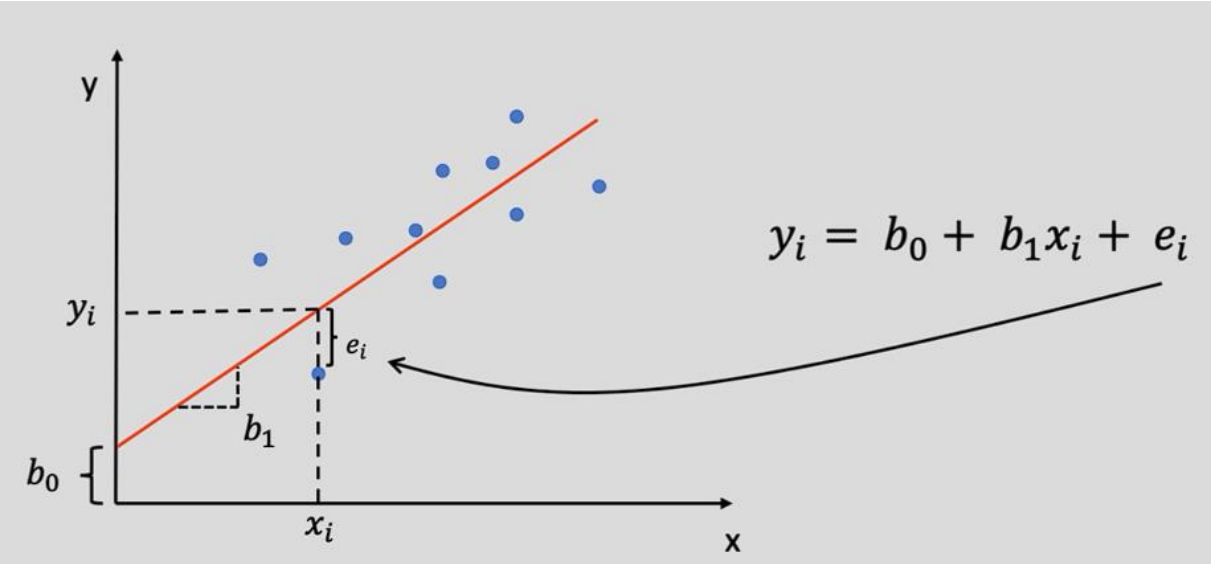
Basit doğrusal regresyon, veriler hakkında belirli tarzlarda varsayımlarda bulunur. Bu varsayımlar şunlardır:

1. **Varyansın homojenliği:** Tahminimizdeki hatanın boyutudur.
2. **Gözlemlerin bağımsızlığı:** Veri setindeki gözlemler istatistiksel olarak geçerli yöntemler kullanılarak toplanmıştır ve değişkenler arasında gizli ilişkiler yoktur. Çoklu doğrusal regresyonda, bazı bağımsız değişkenlerin aslında birbiriyle ilişkili olması mümkündür, bu nedenle regresyon modelini geliştirmeden önce bunları kontrol etmek önemlidir. Eğer iki bağımsız değişken çok yüksek oranda ilişkiliyse, o zaman regresyon modelinde bunlardan sadece biri kullanılmalıdır.
3. **Normallik:** Veriler normal bir dağılım izler.
4. **Doğrusallık:** veri noktalarından geçen en uygun çizgi, bir eğri veya bir tür gruplama faktörü yerine düz bir çizgidir.

2.7.2.2. Basit Doğrusal Regresyon Formülleri

Basit doğrusal regresyon formülü: $y = \beta_0 + \beta_1 X + \epsilon$

- y Bağımsız değişkenin (x) herhangi bir verilen değeri için bağımlı değişkenin (y) tahmin edilen değeridir.
- β_0, kesişme noktasıdır, x 0 olduğunda y'nin tahmin edilen değeridir.
- β_1, regresyon katsayısıdır- x arttıkça y'nin ne kadar değişmesini beklediğimizdir.
- X, Bağımsız değişkendir (beklediğimiz değişken y'yi etkiler).
- ϵ, tahminin hatası veya regresyon katsayısı tahminimizde ne kadar varyasyon olduğudur. [21]



Şekil 2.2. Basit Doğrusal Regresyon Geometrik Gösterim [51]

2.7.3. Çoklu Doğrusal Regresyon

Basit doğrusal regresyon, bir analistin veya istatistikçinin başka bir değişken hakkında bilinen bilgilere dayanarak bir değişken hakkında tahminlerde bulunmasına izin veren bir fonksiyondur. Çoklu doğrusal regresyon ise iki veya daha fazla bağımsız değişken ile bir bağımlı değişken arasındaki ilişkiyi tahmin etmek için çoklu doğrusal regresyon kullanılır.

Çoklu doğrusal regresyon kullanmanız gereken zamanlar:

1. İki veya daha fazla bağımsız değişken ile bir bağımlı değişken arasındaki ilişkinin ne kadar güçlü olduğunu görmek amacıyla kullanılır.
2. Bağımsız değişkenlerin bağımlı değişken ile oluşan formülasyonu için kullanılır.

2.7.3.1. Çoklu Doğrusal Regresyon Varsayımları

- Bağımlı ve bağımsız değişkenler arasındaki ilişkilerin doğrusallığı
- Gözlemlerin bağımsızlığı
- Artıkların normalliği
- Kalıntıların homoskedastisitesi
- Etkili nokta yok (aykırı değerler)

2.7.3.2. Çoklu Doğrusal Regresyon Formülleri

Çoklu doğrusal regresyon formülü: $y = \beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n + \epsilon$

- y = Bağımlı değişkenin tahmin edilen değeri
- β_0 = y kesme noktası (diğer tüm parametreler 0'a ayarlandığında y'nin değeri)
- $\beta_1 X_1$ = regresyon katsayısı (B_1) birinci bağımsız değişkenin (X_1) (diğer bir deyişle bağımsız değişkenin değerini artırmanın tahmin edilen y değeri üzerindeki etkisi)
- ... = test ettiğiniz birçok bağımsız değişken için aynısını yapın
- $\beta_n X_n$ = son bağımsız değişkenin regresyon katsayısı
- ϵ = model hatası (diğer bir deyişle, tahminimizde ne kadar varyasyon var?)

Her bağımsız değişken için en uygun doğruyu bulmak için çoklu doğrusal regresyon üç şeyi hesaplar:

- En küçük genel model hatasına yol açan regresyon katsayıları.
- Genel modelin t istatistiği.
- İlişkili p- değeri (bağımsız ve bağımlı değişkenler arasında hiçbir ilişkinin olmadığına dair boş hipotez doğru olsaydı, t-istatistiğinin tesadüfen meydana gelme olasılığı ne kadardır). [22]

2.7.4. Temel Bileşen Regresyon (PCR)

İstatistikte, temel bileşen regresyonu (PCR), temel bileşen analizine (PCA) dayanan bir regresyon analizi tekniğidir. Daha spesifik olarak, PCR, standart bir lineer regresyon modelinde bilinmeyen regresyon katsayılarını tahmin etmek için kullanılır. Temel Bileşen Regresyonu (PCR), standart doğrusal regresyonla aynı amaca hizmet eden bir regresyon tekniğidir.

Aradaki fark, PCR'nin orijinal özellikler yerine regresyon analizi için belirleyici değişkenler olarak temel bileşenleri kullanmasıdır. PCR 3 adımda çalışır:

1. PCA uygulanır.

2. Varyansın çoğunu açıklayan ilk k ana bileşeni (burada k <p), burada k çapraz doğrulama ile belirlenir.

3. Bu k ana bileşene doğrusal bir regresyon modeli (sıradan en küçük kareler kullanarak) yerleştirilir.

Buradaki fikir, daha az sayıda temel bileşenin verilerdeki değişkenliğin çoğunu ve (varsayımsal olarak) hedef değişkenle olan ilişkiyi temsil etmesidir. Bu nedenle, regresyon için tüm orijinal özellikleri kullanmak yerine, yalnızca temel bileşenlerin bir alt kümesini kullanırız.[23]

2.7.5. Kısmi En Küçük Kareler Regresyonu (PLS)

Değişkenlerin daha az sayıda ve aralarında çoklu doğrusal bağlantı problemi olmayan bileşenlere indirgenir regresyon modeli kurulması fikrine dayanır. PCR ile yaklaşım olarak aynı mantığı ifade eder. Amaç burada yüksek korelasyonlu veya çok boyut problemi gibi problemlere çözüm sunmaktır. PLS de PCR gibi bağımsız değişkenlerin doğrusal kombinasyonlarını bulur. Bu doğrusal kombinasyonlar bileşen ya da latent değişken olarak adlandırılır. PLS NIPALS'in özel bir halidir, iteratif olarak bağımlı değişken ile yüksek korelasyona sahip değişenler arasındaki gizil (latent) ilişkiyi bulmaya çalışır.

2.7.5.1. PCR ve PLS Farkları

- PCR'da doğrusal kombinasyonlar yani bileşenler bağımsız değişken uzağındaki değişkenliği maksimum şekilde özetleyecek şekilde oluşturulur. Bu durum bağımlı değişkeni açıklama yeteneği olmamasına sebep olmakta.
- PLS'te ise bileşenler bağımlı değişken ile olan kovaryansı maksimum şekilde özetleyecek şekilde oluşturulur.
- Değişkenler atılmak istenmiyorsa ve açıklanabilirlik aranıyorsa: PLS
- PLS, gözetimli boyut indirgeme prosedürü, PCR gözetimsiz boyutindirgeme prosedürü olarak görülebilir.
- İki yönteminde bir tunning parametresi vardır o da bileşen sayısıdır.
- Optimum bileşen sayısını belirlemek için CV yöntemi kullanılır.

2.7.6. Ridge Regresyon

Ridge regresyon, bağımsız değişkenlerin yüksek oranda korelasyon gösterdiği senaryolarda çoklu regresyon modellerinin katsayılarını tahmin etme yöntemidir. Amaç hata kareler toplamını minimize eden katsayıları bu katsayılara bir ceza uygulayarak bulmaktır. Ridge regresyon, lineer regresyon modellerinin bazı çok bağlantılı (yüksek derecede korelasyonlu) bağımsız değişkenlere sahip olduğu durumlarda en küçük kareler tahmin edicilerinin belirsizliğine olası bir çözüm olarak bir Ridge regresyon tahmincisi (RR) yaratarak geliştirilmiştir. Varyansı ve ortalama kare tahmincisi genellikle daha önce türetilen en küçük kare tahmincilerinden daha küçük olduğundan, bu daha kesin bir Ridge parametreleri tahmini sağlar. Bu model, kayıp fonksiyonunun lineer en küçük kareler fonksiyonu olduğu ve düzenlileştirmenin L_2-norm tarafından verildiği bir regresyon modelini çözer. Ayrıca Ridge Regresyon veya Tikhonov düzenlemesi olarak da bilinir.

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y_i})^2$$

SSE = En Küçük Kareler Toplamı

Bu doğrusal regresyon modelinin kabiliyetlerini ve başarılarını kullanarak daha ileri düzeye ve daha başarılı hale nasıl getirebiliriz çabaları sonucu doğan başka modelleme tekniğidir.

$$SSE_{L_2} = \sum_{i=1}^{n} (y_i - \hat{y_i})^2 + \lambda \sum_{j=1}^{P} {\beta_j}^2$$

λ = Ayar Parametresi Lambda

$\sum_{j=1}^{P} {\beta_j}^2$ = Ceza Terimi

- Aşırı öğrenmeye karşı dirençli.
- Yanlıdır fakat varyansı düşüktür. (Bazen yanlı modelleri daha çok tercih ederiz.)
- Çok fazla paremetre olduğunda EKK'ya göre daha iyidir.
- Çok boyutluluk lanetine karşı çözüm sunar.
- Çoklu doğrusal bağlantı problemi olduğunda etkilidir.
- Tüm değişkenler ile model kurar. İlgisiz değişkenleri modelden çıkarmaz, katsayılarını sıfıra yaklaştırır.

- λ kritik roldedir. İki terimin (formüldeki) göreceli etkilerini kontrol etmeyi sağlar.
- λ için iyi bir değer bulunması önemlidir. Bunun için CV yöntemi kullanılır.[24]

2.7.7. Lasso Regresyon

Amaç hata kareler toplamını minimize eden katsayıları bu katsayılara bir ceza uygulayarak bulmaktır. Ridge Regresyondan ayrı olarak bu katsayılara ceza işlemini biraz daha abartarak katsayıların cezalarını onları sıfır yapacak şekilde uygulamaktadır. Böylece değişken seçimi yapmaktadır. Lasso regresyonu bir düzenlileştirme tekniğidir. Daha doğru bir tahmin için regresyon yöntemleri üzerinde kullanılır. Bu model büzülme kullanır. Büzülme, veri değerlerinin ortalama olarak merkezi bir noktaya doğru küçüldüğü yerdir. Lasso prosedürü basit, seyrek modelleri (yani daha az parametreli modelleri) teşvik eder. Lasso, başlangıçta doğrusal regresyon modelleri için formüle edilmiştir. Bu basit durum, tahmin edici hakkında önemli miktarda bilgi verir. Bunlar, sırt regresyonu ve en iyi alt küme seçimi ile ilişkisini ve kement katsayısı tahminleri ile sözde yumuşak eşikleme arasındaki bağlantıları içerir. Bu özel regresyon türü, yüksek düzeyde çoklu bağlantı gösteren modeller için veya değişken seçimi/parametre eleme gibi model seçiminin belirli kısımlarını otomatikleştirmek istediğinizde çok uygundur.

$$SSE_{L_1} = \sum_{i=1}^{n} (y_i - \hat{y_i})^2 + \lambda \sum_{j=1}^{P} |\beta_j|$$

- Ridge regresyonun ilgili-ilgisiz tüm değişkenleri modelde bırakma dezavantajını gidermek için önerilmiştir.
- Lasso'da katsayıları sıfıra yaklaştırır.
- Fakat L1 normu λ yeteri kadar büyük olduğunda bazı katsayıları sıfır yapar. Böylece değişken seçimi yapmış olur.
- X'nın doğru seçilmesi çok önemlidir, burada da CV kullanılır.
- Ridge ve Lasso yöntemleri birbirinden üstün değildir.

2.7.7.1. λ Ayar Parametresinin Belirlenmesi

- λ'nın sıfır olduğu yer en küçük karelerdir. Hata kareler toplamını minimum yapan X'yı arıyoruz
- λ için belirli değerleri içeren bir küme seçilir ve her birisi için cross validation test hatası hesaplanır.

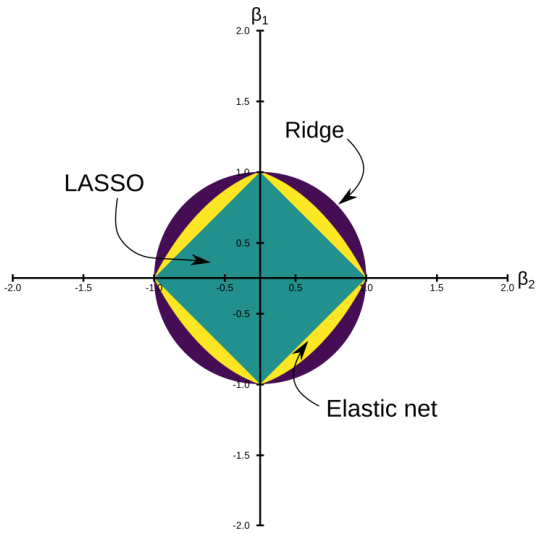
- En küçük cross validation'ı veren λ ayar parametresi olarak seçilir.
- Son olarak seçilen bu λ ile model yeniden tüm gözlemlere fit edilir. [25]

2.7.7. ElasticNet Regresyon

İstatistikte ve özellikle, doğrusal veya lojistik regresyon modellerinin uydurulmasında, ElasticNet, ridge ve lasso yöntemlerinin L_1 ve L_2 cezalarını doğrusal olarak birleştiren düzenli bir regresyon yöntemidir. ElasticNet yöntemi, lassonun sınırlamalarını iyileştirir. ElasticNet prosedürü, doygunluğa kadar "n" sayıda değişkenin dahil edilmesini sağlar. Değişkenler yüksek oranda ilişkili gruplar ise, lasso bu tür gruplardan bir değişken seçme ve gerisini tamamen görmezden gelme eğilimindedir. ElasticNet yönteminin tahmin edicisini bulma prosedüründe, iki aşama hem lasso hem de regresyon tekniklerini içerir. İlk önce ridge regresyon katsayılarını bulur ve ardından katsayıların bir kement büzülmesini kullanarak ikinci adımı gerçekleştirir. Dolayısıyla bu yöntem, katsayıları iki tür büzülmeye tabi tutar. ElasticNet'in naif versiyonundan gelen çift büzülme, öngörülebilirlikte düşük verimliliğe ve yüksek yanlılığa neden olur.

$$SSE_{L_1} = \sum_{i=1}^{n}(y_i - \hat{y_i})^2 + \lambda_1 \sum_{j=1}^{P} {\beta_j}^2 + \lambda_2 \sum_{j=1}^{P} |\beta_j|$$

- Elastik ağ yöntemi, değişken seçimi ve düzenlemeyi aynı anda gerçekleştirir.
- Elastik ağ tekniği, boyutsal verilerin kullanılan numune sayısından fazla olduğu durumlarda en uygundur.
- Gruplamalar ve değişken seçimi, elastik ağ tekniğinin kilit rolleridir.[26]



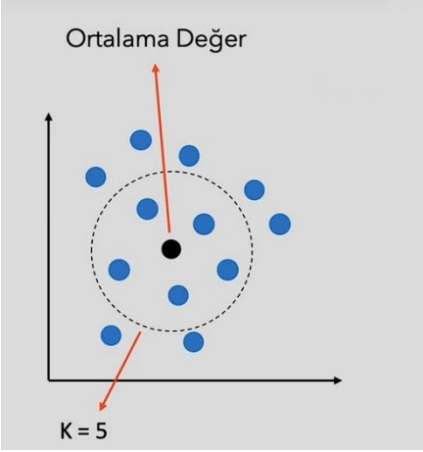
Şekil 2.3. Lasso, Ridge ve Elastic Net Kesişimi [52].

2.8. DOĞRUSAL OLMAYAN REGRESYON

Doğrusal olmayan model karmaşıktır ve aynı zamanda doğru sonuçlar verir. Analiz, sağlanan veri kümesine dayalı olarak değişkenler arasındaki ilişkiyi gösteren bir eğri(vektör) geliştirir. Büyük esneklik sunan model, senaryoya en uygun eğriyi oluşturabilir. Bu ilişki, zaman ve nüfus arasında bağlantı kurmaktan yatırımcı duyarlılığına ve bunun borsa getirileri üzerindeki doğrusal olmayan etkisine kadar her şey olabilir. Doğrusal olmayan regresyonda, deneysel veriler bir modele eşlenir ve eğrisel olan doğrusal olmayan bir ilişkide değişkenleri (bağımlı ve bağımsız) temsil eden matematiksel fonksiyon oluşturulur ve optimize edilir. Esnek bir form olarak kabul edilmektedir. Doğrusal olmayan modeller geliştirmek için kullanılan algoritma örnekleri, Levenberg-Marquardt doğrusal olmayan en küçük kareler ve Gauss-Newton algoritmalarıdır.[27]

2.8.1. K- En Yakın Komşu (KNN)

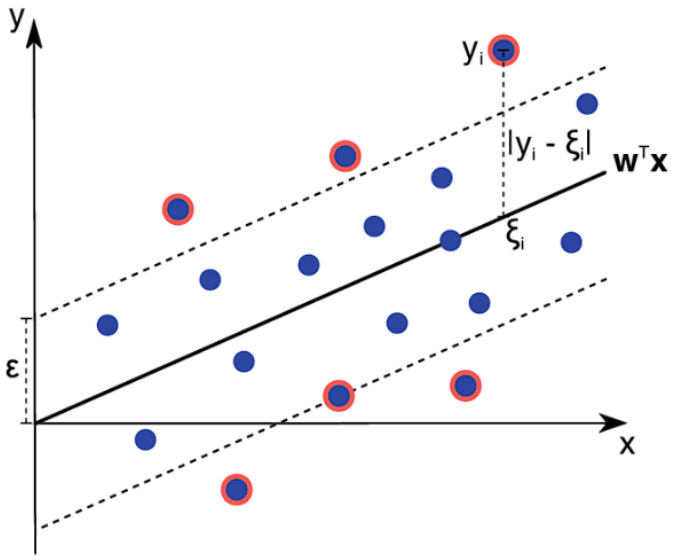
KNN algoritması, son derece basit, uygulaması kolay olan bir denetimli (supervised) makine öğrenmesi algoritmasıdır. KNN algoritması hem regresyon hem de classification için kullanılır. K en yakın komşu (KNN) algoritması, adından da anlaşılacağı üzere, komşularına bakarak tahminlemede bulunan bir algoritmadır. KNN algoritmasında, benzer olan şeyler birbirine yakındır varsayımı geçerlidir. Genel itibari ile, benzer olan sınıfların birbiri ile yakın mesafede oldukları gözlenir. KNN algoritması da bu gözleme dayanarak, yeni gelecek tahminler bu noktalara yakınlığa göre tahminlenir. Ölçülen mesafe nedir? Evet Machine learning algoritmalarında farklı mesafe ölçümleri kullanılmaktadır. Temel olarak Öklid mesafesi (Euclidean distance) kullanılsa da daha farklı ölçümlerde mevcuttur. Kalıcı olması bakımıyla "Bana arkadaşını söyle sana kim olduğunu söyleyeyim" atasözüyle de eşleştirilebilen bir algoritmadır.[28]



Şekil 2.4. KNN Çalışma Prensibi Geometrik Gösterim [53].

2.8.2. Destek Vektör Regresyonu (SVR)

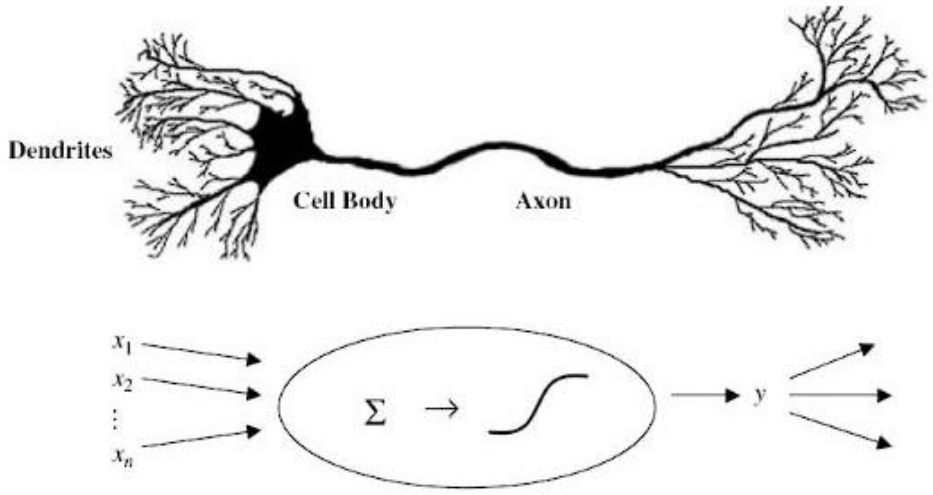
Destek Vektör Makinesi, algoritmayı karakterize eden tüm ana özellikleri (maksimum marj) koruyarak bir regresyon yöntemi olarak da kullanılabilir. Destek Vektör Regresyonu (SVR), sınıflandırma için SVM ile aynı prensipleri kullanır, sadece birkaç küçük farklılık vardır. Her şeyden önce, çıktı gerçek bir sayı olduğundan, sonsuz olasılıklara sahip olan eldeki bilgiyi tahmin etmek çok zor hale gelir. Gerileme durumunda, problemden zaten talep etmiş olacak olan SVM'ye yaklaşık olarak bir tolerans marjı (epsilon) ayarlanır. Ancak bu gerçeğin yanı sıra, daha karmaşık bir neden daha var, algoritma daha karmaşık, bu nedenle dikkate alınması gerekiyor. Bununla birlikte, ana fikir her zaman aynıdır: hatayı en aza indirmek, marjı en üst düzeye çıkaran hiperdüzlemi bireyselleştirmektir. [29]



Şekil 2.5. Destek Vektör Regresyonun Geometrik Gösterimi [54].

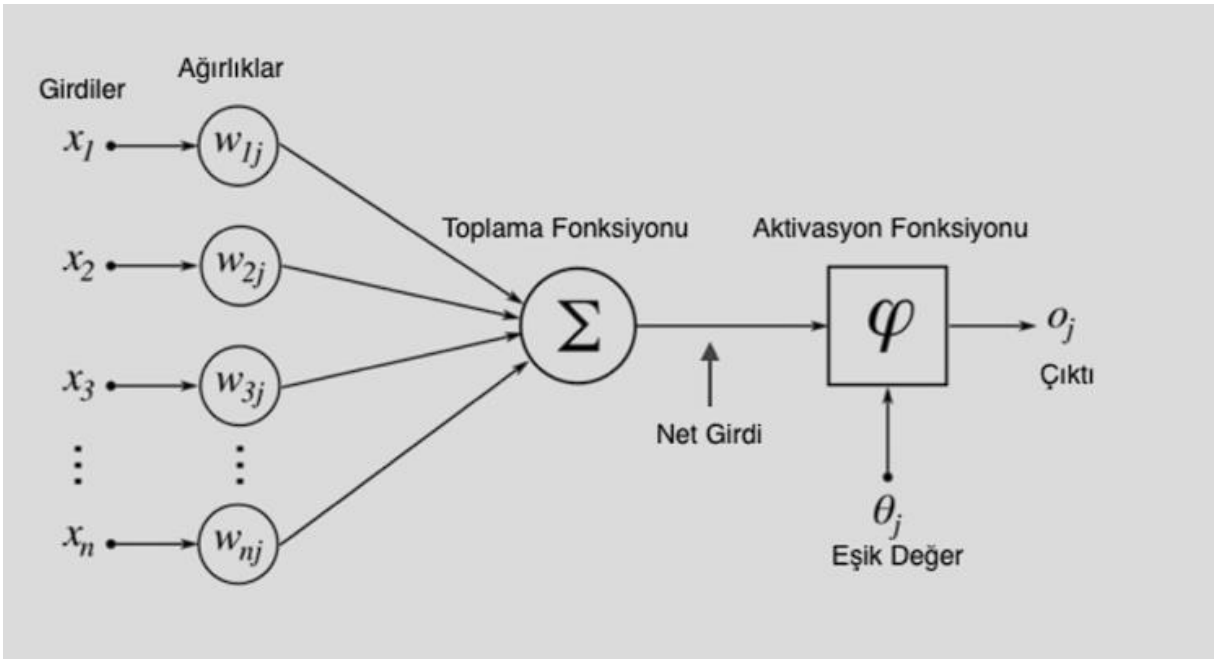
2.8.3. Yapay Sinir Ağları

Yapay sinir ağlarının regresyon problemlerine uyarlanmış bölümüdür. Insan beyninin bilgi işleme şeklini referans alan sınıflandırma ve regresyon problemleri için kullanılabilen kuvvetli makine öğrenmesi algoritmalarından birisidir.[30]



Şekil 2.6. Yapay Sinir Ağlarının Sinir Ağlarıyla Eşleştirme Görseli [55].

Sinir Sistemi	**Yapay Sinir Ağı**
Nöron	İşlem Elemanı
Dentrit	Toplama Fonksiyonu
Hücre Gövdesi	Aktivasyon Fonksiyonu
Akson	Eleman Çıkışı
Sinaps	Ağırlıklar



Şekil 2.7. Yapay Sinir Ağlarının İşlem Şeması [56].

Yapay sinir ağları formülü

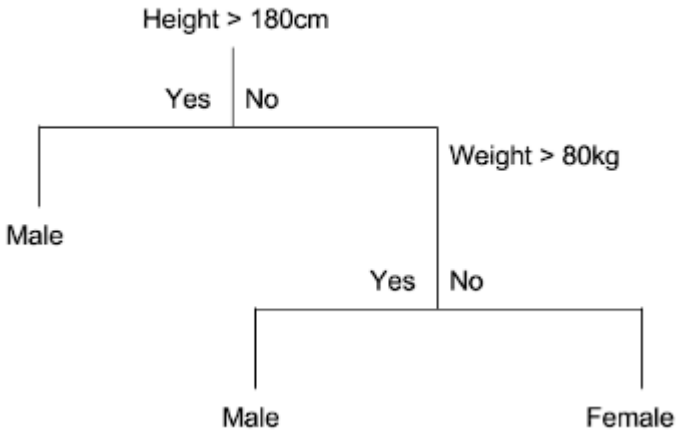
$$h_k(x) = g\left(\beta_j + \sum_{j=1}^{P} x_j \beta_{jk}\right)$$

$$g(u) = \frac{1}{1 + e^{-u}}$$

$$f(x) = \gamma_0 + \sum_{k=1}^{H} \gamma_k h_k$$

2.8.4. Regresyon Ağaçları (CART)

Amaç veri seti içerisindeki karmaşık yapıları basit karar yapılarına dönüştürmektir. Heterojen veri setleri belirlenmiş bir hedef değişkene göre homojen alt gruplara ayrılır. CART, 1984 yılında Leo Breiman, Jerome Friedman, Richard Olshen ve Charles Stone tarafından regresyon görevi için tanıtıldı. Ek olarak, diğer etiketlenmiş değişkenleri destekleyen bir değişken aramaya yardımcı olan bir tahmin modelidir. Daha açık olmak gerekirse, ağaç modelleri bir grup "if-else" sorusu sorarak sonucu tahmin eder.[31]



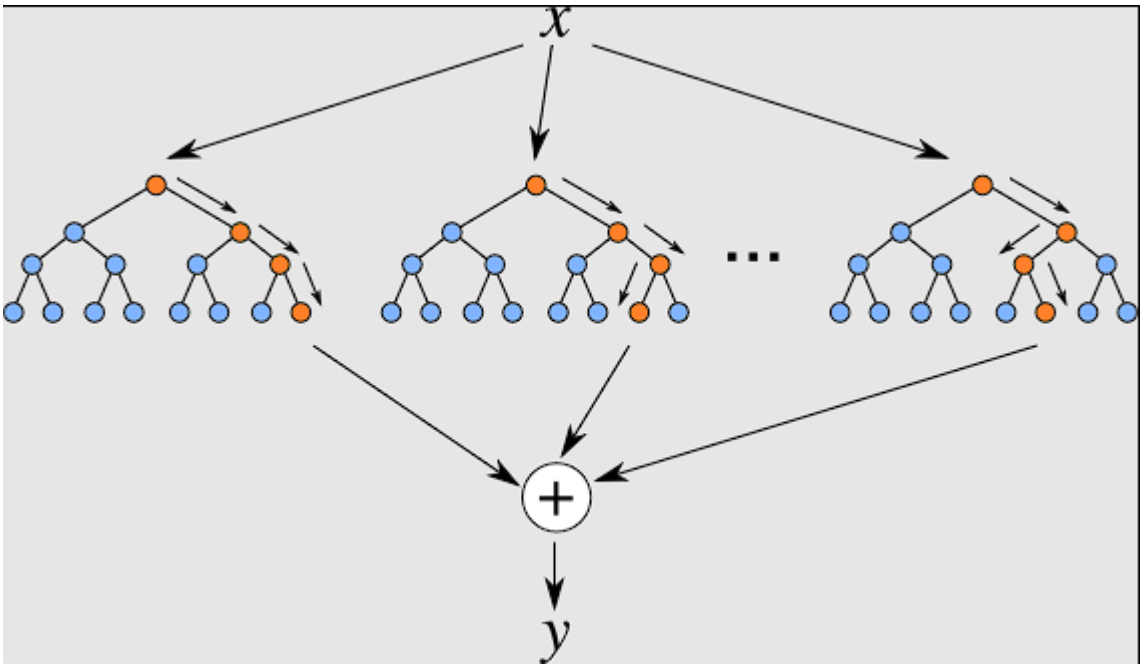
Şekil 2.8. CART Ağaçlandırma Sistemi [57].

2.8.5. Bagging

Temeli bootstrap yöntemi ile oluşturulan birden fazla karar ağacının ürettiği tahminlerin bir araya getirilerek değerlendirilmesine dayanır.

2.8.6. Random Forests (RF)

Random Forest, Sınıflandırma ve Regresyon problemlerinde yaygın olarak kullanılan bir denetimli makine öğrenmesi algoritmasıdır. Farklı örnekler üzerinde karar ağaçları oluşturur ve regresyon durumunda sınıflandırma ve ortalama için çoğunluk oyu alır.Random Forest Algoritmasının en önemli özelliklerinden biri, regresyon durumunda olduğu gibi sürekli değişkenleri ve sınıflandırma durumunda olduğu gibi kategorik değişkenleri içeren veri setini işleyebilmesidir .Rastgele orman, Leo Breiman ve Adele Cutler'ın ticari markası olan ve tek bir sonuca ulaşmak için birden fazla karar ağacının çıktısını birleştiren, yaygın olarak kullanılan bir makine öğrenme algoritmasıdır. Kullanım kolaylığı ve esnekliği hem sınıflandırma hem de regresyon problemlerini ele aldığı için benimsenmesini hızlandırdı.[32]



Şekil 2.9. Random Forest Ağaçlandırma Şeması [58].

2.8.7. Gradient Boosting Machines (GBM)

Gradyan güçlendirme, diğerleri arasında regresyon ve sınıflandırma görevlerinde kullanılan bir makine öğrenimi tekniğidir. Tipik olarak karar ağaçları olan zayıf tahmin modelleri topluluğu

şeklinde bir tahmin modeli verir. Bir karar ağacı zayıf öğrenen olduğunda, elde edilen algoritmaya gradyan destekli ağaçlar denir; genellikle rastgele ormandan daha iyi performans gösterir. Gradyan destekli bir ağaç modeli, diğer artırma yöntemlerinde olduğu gibi aşamalı bir şekilde oluşturulur, ancak diğer yöntemleri, rastgele bir optimizasyonun optimizasyonuna izin vererek genelleştirir. Adaboost'un sınflandırma ve regresyon problemlerine kolayca uyarlanabilen genelleştirilmiş versiyonudur. Artılar üzerine tek bir tahminsel model formunda olan modeller serisi kurulur. Boosting yöntemi zayıf öğrenicileri bir araya getirip güçlü bir öğrenici ortaya çıkarmak fikrine dayanır. Adaptive boosting (AdaBoost) Zayıf sınıflandırıcıların bir araya gelerek güçlü bir sınıflandırıcı oluşturması fikrini hayata geçiren algoritmadır.[33]

Özellikler:

- Gradient boosting tek bir tahminsel model formunda olan modeller serisi oluşturur.
- Seri içerisindeki bir model serideki bir önceki modelin tahmin artıklarının/hatalarının (residuals) üzerine kurularak(fit)oluşturulur.
- GBM diferansiyellenebilen herhangi bir kayıp fonksiyonunu optimize edebilen Gradient descent algoritmasını kullanmakta.
- GB birçok temel öğrenici tipi (base learner type) kullanabilir. (Trees, linear terms, splines, ...)
- Cost fonksiyonları ve link fonksiyonları modifiye edilebilirdir.
- Boosting + Gradient Descent

2.8.8. eXtreme Gradient Boosting (XGBoost)

XGBoost, GBM'in hız ve tahmin performansını arttırmak üzere optimize edilmiş; ölçeklenebilir ve farklı platformlara entegre edilebilir halidir.

Extreme Gradient Boosting (XGBoost), gradyan artırma algoritmasının verimli ve etkili bir şekilde uygulanmasını sağlayan açık kaynaklı bir kitaplıktır.

Geliştirilmesinden ve ilk piyasaya sürülmesinden kısa bir süre sonra, XGBoost, makine öğrenimi yarışmalarında bir dizi sorun için çözüm kazanmanın en önemli yöntemi ve çoğu zaman anahtar bileşeni haline geldi.

Regresyon tahmini modelleme problemleri, dolar miktarı veya yükseklik gibi sayısal bir değerin tahmin edilmesini içerir. XGBoost, doğrudan regresyon öngörücü modelleme için kullanılabilir.[34]

- R, Python, Hadoop, Scala, Julia ile kullanılabilir.
- Ölçeklenebilirdir.
- Hızlıdır.
- Tahmin başarısı yüksektir.
- Birçok kaggle yarışmasında başarısını kanıtlamıştır.

2.8.9. Light Gradient Boosting Machines (Light GBM)

Light GBM, XGBoost'un eğitim süresi performansını arttırmaya yönelik geliştirilen bir diğer GBM türüdür. LightGBM, histogram tabanlı çalışan bir algoritmadır. Sürekli değere sahip olan değişkenleri kesikli (discrete bin) hale getirerek hesaplama maliyetini azaltır. Karar ağaçlarının eğitim süresi yapılan hesaplama ve dolayısıyla bölünme sayısı ile doğru orantılıdır. Bu yöntem sayesinde hem eğitim süresi kısalmakta hem de kaynak kullanımı düşmektedir. Karar ağaçlarında öğreniminde seviye odaklı (level-wise or depth-wise) veya yaprak odaklı(leaf-wise) olarak iki strateji kullanılabilir. Seviye odaklı stratejide ağaç büyürken ağacın dengesi korunur. Yaprak odaklı stratejide ise kaybı azaltan yapraklardan bölünme işlemi devam eder. LightGBM bu özelliği sayesinde diğer boosting algoritmalarından ayrılmaktadır. Model yaprak odaklı strateji ile daha az hata oranına sahip olur ve daha hızlı öğrenir. Ancak yaprak odaklı büyüme stratejisi veri sayısının az olduğu durumlarda modelin aşırı öğrenmeye yatkın olmasına sebebiyet verir. Bu nedenle algoritma büyük verilerde kullanılmak için daha uygundur.[35]

2.8.10. Category Boosting (CatBoost)

CatBoost, Yandex'in yakın zamanda açık kaynaklı bir makine öğrenimi algoritmasıdır. Google'ın TensorFlow ve Apple'ın Core ML'si gibi derin öğrenme çerçeveleriyle kolayca entegre olabilir. İşletmelerin bugün karşılaştığı çok çeşitli sorunları çözmeye yardımcı olmak için çeşitli veri türleriyle çalışabilir. Üstüne üstlük, sınıfının en iyisi doğruluk sağlar.[36]

Özellikle iki şekilde güçlüdür:

- Tipik olarak diğer makine öğrenimi yöntemlerinin gerektirdiği kapsamlı veri eğitimi olmadan son teknoloji ürünü sonuçlar verir ve
- Birçok iş sorununa eşlik eden daha açıklayıcı veri biçimleri için kullanıma hazır güçlü destek sağlar. "CatBoost" adı "Cat egory" ve "Boost ing" olmak üzere iki kelimeden gelmektedir .

Bu kitaplık, gradyan artırma kitaplığına dayandığından, " Güçlendirme ", gradyan artırma makine öğrenimi algoritmasından gelir. Gradient boosting, sahtekarlık algılama, öneri öğeleri, tahmin gibi birden çok türde iş zorluğuna yaygın olarak uygulanan güçlü bir makine öğrenimi algoritmasıdır ve aynı zamanda iyi performans gösterir. Ayrıca, büyük miktarda veriden öğrenmesi gereken doğrusal regresyon modellerinin aksine, nispeten daha az veri ile çok iyi sonuçlar verebilir.

2.9. DOĞRUSAL REGRESYON ve DOĞRUSAL OLMAYAN REGRESYON HAKKINDAKİ BAZI ÇALIŞMALAR

Doğrusal regresyon modellemesi kullanılarak Erzurum kuzey çevre yolunda kaza tahmin modeli oluşturulmuştur. Oluşturulan doğrusal regresyon modeli bağımlı ve bağımsız değişkenler olarak ayrıştırılıp bunun üzerine regresyon analizi yapılmıştır. Karayolu güvenliği ile ve haritalardan aldıkları bilgiler ile yaptıkları modelleme ile bir sonuca ulaşmaya çalışmışlardır [40].

Eczacılık anlamında büyüme gösterip ilaçların ne kadar faydalı olabilmesi için yapılmış bir çalışmadır. Yapılan çalışmada doğrusal regresyon çözümlemesi ve ilaçların bir tablet üzerinde ne kadar değer gözlemlediği hakkında çalışma yapılmıştır. Doğrusal regresyon formülasyonu işlenmiştir. Ön kestirim ve güven aralıkları hesaplamaları yapılıp buna duyarlı olarak tahmin işlemleri yapılmıştır [41].

Doğrusal regresyon onun nasıl işlediği incelenip nedenselliğine ile bakılmıştır. Doğrusal regresyon hangi durumlarda kullanılabileceği tartışılmıştır. Bu derginin asıl amacı regresyon modelinin istatiksel olarak önemli bulunması tek başına neden sonuç ilişkisini açıklamayacağını anlaşılır hale getirmektir [42].

Doğrusal olmayan regresyon ne olduğunu anlatmayı hedeflemiştir. Doğrusal olmayan regresyon bağımlı değişikliğin ile bir dizi bağımsız değişken arasındaki ilişkinin doğrusal olmayan bir modeli bulunmakta kullanılan bir yöntemdir doğrusal regresyon ondan farklı olarak doğrusal olmayan regresyon modellerinin bağımsız ve bağımlı değişkenler arasında keyfi ilişkiler ile tahmin edilebilmesini sağlar [43].

Doğrusal regresyon ile doğrusal olmayan regresyon modellerinin karşılaştırılması yapılmaktadır. Doğrusal olmayan modellerin doğrusal resesyonun aksine uydurma önemsiz olmayan varsayımlara dayandığını söyler. Parametre tahmini, ideal olarak optimum parametre tahminlerinin belirlenmesine yol açan yinelemeli bir süreci içeren en küçük kareler kriterinin bazı varyantları kullanılarak gerçekleştirilir. Bu yazıda göğüs hastalıkları alanından işlenmiş bir örnek ele alınmıştır [44].

Random Forest'ın tarihçesini anlatmaktadır. Genel amaçlı olarak regresyon ve sınıflandırmada son derece başarılı olduğu aktarılmaktadır. Birkaç rastgele karar ağacını birleştiren ve tahminlerini ortalama alarak toplayan yaklaşım değişken sayısının gözlem sayısından çok daha fazla olduğu ortamlarda mükemmel performans gördüğü kanıtlanmıştır. Büyük ölçekli problemlere uygulanabilecek kadar çok yönlü olduğu söylenmiştir. Bu makalede random forest için en son teorik ve metodolojik gelişmeler gözden geçirilmektedir [45].

KNN algoritmasını anlatmaktadır. Sınıflandırma veya regresyon için anlaşılması çok basit bir algoritma olduğu söylenmektedir önemli bir noktanın test aşamasında tüm eğitim verilerini tutan tembel bir veri algoritması olduğu söylenmektedir. Bu algoritma için popülasyon boyutu çok önemlidir çünkü popülasyon boyutu ne kadar büyürse ya ös yürütme hızını ve büyük bellek gereksinimlerine ihtiyaç olunabilir [46].

Elektrik yükü, ekonomik faaliyetler veya iklim döngüsel doğası nedeniyle bazen mevsimsel eğilim gösterir. Mevsimsel elektrik yükü tahminini SVR modeli kullanılarak yapılmaya çalışmışlardır. Aynı zamanda SVR ile yapay sinir ağlarını birleştirilmişlerdir. Bunun sebebi geçmiş bilgileri kullanmayı tekrarlamaya odaklanan yararlı bir birleşimin olmasıdır. Bu modelleme birleşiminin elektrik yüklerinin tahmini için umut verici bir alternatif olduğunu dayandırılmaktadır [47].

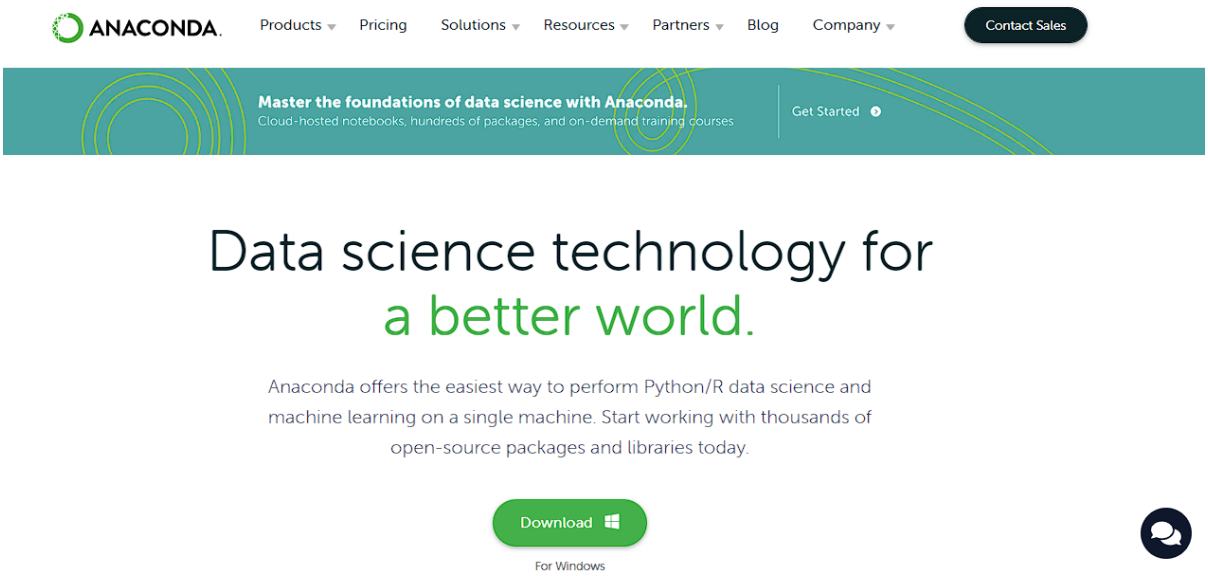
Sınıflandırma problemlerinde kullanılan boosting yöntemi regresyon problemlerinde ne tür bir sonuç doğurabileceği düşünülerek böyle bir makale yazılmıştır. Regresyon problemlerinde boosting uygulaması çok az araştırılmıştır. Bu makalede regresyon

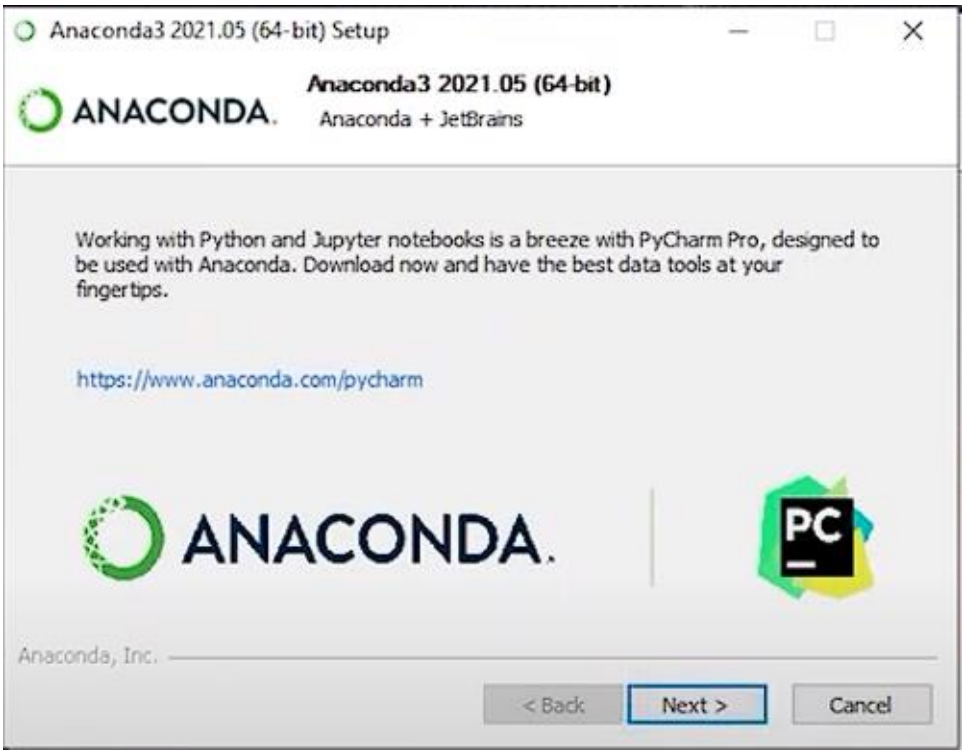
problemleri için bir boosting yöntemi geliştirilmeye çalışılmıştır. Regresyon problemi bir sınıflandırma problemi olarak gösterilerek boosting yöntemi için bayes sınıflandırıcısının yorumlanabilir bir formu uygulandı. Makalede boosting sistemi uygulanmış bayes regresyon modelinin performansını diğer regresyon prosedürleriyle karşılaştırılıyor [48].

3. UYGULAMA

3.1. ANACONDA KURULUMU

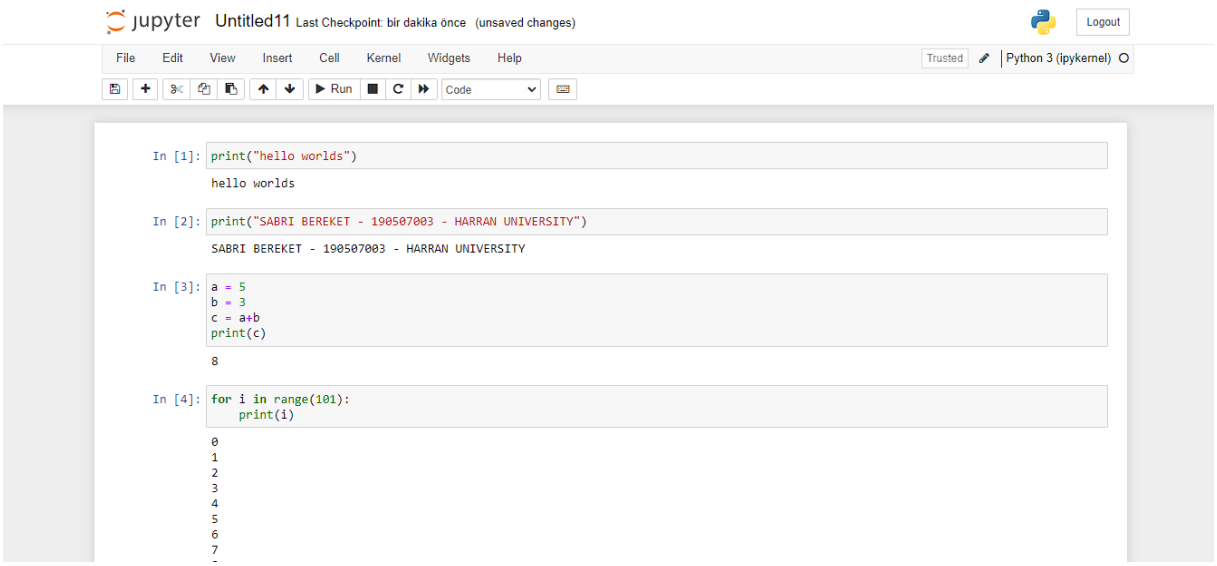
Anaconda [37] websitesinden Anaconda3 adlı uygulamayı kurulum işlemlerimizi başlatmak için indirip bilgisayarımıza kuruyoruz.





3.2. JUPYTER NOTEBOOK İLE PYTHON ÇALIŞMALARI

Python kodlama dili üzerine kod parçacıklarını öğrenme amacıyla python kullanarak kodlama çalışmaları yapıldı.



3.3. VERİ SETİ HİKAYELERİ

Çalışmalar boyunca kullanacağımız iki veri seti vardır.

3.3.1. Advertising

Advertising [38] adlı veri setinde bir şirketin televizyon, radyo ve gazetede vermiş oldukları reklamlara göre yapmış oldukları satışların verisini barındırmaktadır.

TV = Televizyon

Radio = Radyo

Newspaper = Gazete

Sales = Satışlar

3.3.2. Hitters

Hitters [39] adlı veri setinde 1986 ve 1987 sezonlarından Major League Baseball verileridir.

Hits = 1986'daki isabet sayısı

HmRun = 1986'daki ev koşusu sayısı

Runs = 1986'daki koşu sayısı

RBI = 1986'da yapılan koşu sayısı

Walks = 1986'daki yürüyüş sayısı

Years = Büyük liglerdeki yıl sayısı

CAtBat = Kariyeri boyunca vuruş sayısı

CHits = Kariyeri boyunca isabet sayısı

CHmRun = Kariyeri boyunca yaptığı ev koşusu sayısı

CRuns = Kariyeri boyunca koşu sayısı

CRBI = Kariyeri boyunca yapılan koşu sayısı

CWalks = Kariyeri boyunca yürüyüş sayısı

League = Seviyeleri A ve N olan 1986'nın sonunda oyuncunun ligini gösteren bir faktör

Division = Seviyeleri E ve W olan 1986'nın sonunda oyuncunun bölümünü gösteren bir faktör

PutOuts = 1986'daki satış sayısı

Assists = 1986'daki asist sayısı

Errors = 1986'daki hata sayısı

Salary = 1987 açılış gününde binlerce dolar olarak yıllık maaş

NewLeague = 1987'nin başında seviyeleri olan A ve oyuncunun ligini gösteren bir faktör

3.3. BASİT DOĞRUSAL REGRESYON

3.3.1. Modelleme

Uyarıları yoksaymak için öncelikle uyarıları filtreliyoruz sonrasında veri setimizi [] pandas aracılığıyla okutup tablo halinde gösteriyoruz.

Kod Parçacığı 1.

```
from warnings import filterwarnings
filterwarnings('ignore')

import pandas as pd
ad = pd.read_csv("Advertising.csv", usecols = [1,2,3,4])
df = ad.copy()
df.head()
```

Kod Çıktısı 1.

	TV	radio	newspaper	sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9

Veri seti hakkında bilgi ediniyoruz. Veri seti bilgilendirmesi, açıklanması, birbiriyle olan ilişkilerini gözlemliyoruz.

```
df.info()
df.describe().T
df.isnull().values.any()
df.corr()
```

Kod Çıktısı 2.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   TV         200 non-null    float64
 1   radio      200 non-null    float64
 2   newspaper  200 non-null    float64
 3   sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
```

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
TV	200.0	147.0425	85.854236	0.7	74.375	149.75	218.825	296.4
radio	200.0	23.2640	14.846809	0.0	9.975	22.90	36.525	49.6
newspaper	200.0	30.5540	21.778621	0.3	12.750	25.75	45.100	114.0
sales	200.0	14.0225	5.217457	1.6	10.375	12.90	17.400	27.0

```
df.isnull().values.any()
```

```
False
```

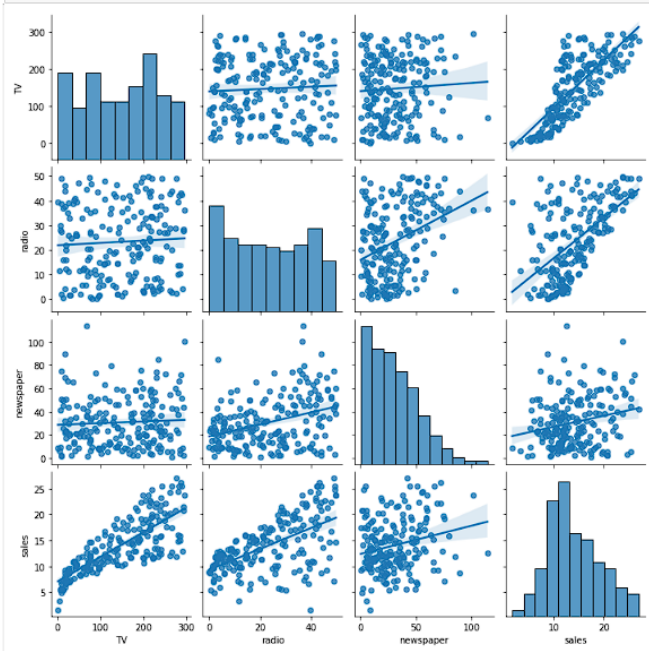
```
df.corr()
```

	TV	radio	newspaper	sales
TV	1.000000	0.054809	0.056648	0.782224
radio	0.054809	1.000000	0.354104	0.576223
newspaper	0.056648	0.354104	1.000000	0.228299
sales	0.782224	0.576223	0.228299	1.000000

Veriseti değişkenlerinin birbirleriyle olan ilişkilerini görselleştirerek gösteriyoruz.

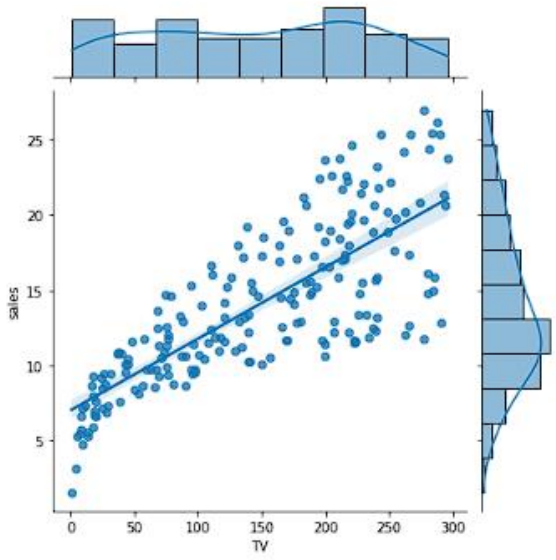
```
1.  import seaborn as sns
2.  sns.pairplot(df, kind  ="reg");
3.  sns.jointplot(x = "TV", y = "sales", data = df, kind = "reg")
```

```
import seaborn as sns
sns.pairplot(df, kind  ="reg");
```



Spesifik olarak televizyon ile satışların orantısını inceliyoruz ve bunu her biri için ayrı ayrı yaparak gözlemleyebiliyoruz.

```
sns.jointplot(x = "TV", y = "sales", data = df, kind = "reg")
<seaborn.axisgrid.JointGrid at 0x1d6eed8d7c0>
```



Basit doğrusal regresyonun çalışma prensibinden dolayı tek bir değişken seçiyoruz. Bu değişken veri setimizdeki “TV” alanıdır. Basit doğrusal regresyon uygulamasını TV alanı üzerine inşa edeceğiz. Modelleme işlemi için statsmodels kütüphanesini kullanacağız. X parametresine matris işlemi uygulamak için yeni sütun ekliyoruz. Y parametresine bağımlı değişkenimiz olan sales kısmını tanımlıyoruz.

```
1. import statsmodels.api as sm
2. X = df[["TV"]]
3. X = sm.add_constant(X)
4. y = df["sales"]
```

Model kurma işlemini gerçekleştiriyoruz. Sonrasında modelimizi fit edip modelimizi görme işlemini gerçekleştiriyoruz.

```
1. lm = sm.OLS(y,X)
2. model = lm.fit()
3. model.summary()
```

OLS Regression Results

Dep. Variable:	sales	**R-squared:**	0.612
Model:	OLS	**Adj. R-squared:**	0.610
Method:	Least Squares	**F-statistic:**	312.1
Date:	Sun, 20 Nov 2022	**Prob (F-statistic):**	1.47e-42
Time:	13:50:47	**Log-Likelihood:**	-519.05
No. Observations:	200	**AIC:**	1042.
Df Residuals:	198	**BIC:**	1049.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	7.0326	0.458	15.360	0.000	6.130	7.935
TV	0.0475	0.003	17.668	0.000	0.042	0.053

Omnibus:	0.531	**Durbin-Watson:**	1.935
Prob(Omnibus):	0.767	**Jarque-Bera (JB):**	0.669
Skew:	-0.089	**Prob(JB):**	0.716
Kurtosis:	2.779	**Cond. No.**	338.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Statsmodels kütüphanesinden yararlanarak “TV” ve “Sales” arasında OLS (en küçük kareler) Regresyonu oluşturduk. Bu regresyonu oluştururken dataframeimizi kullandık. Regresyon işleminden sonra model isminde bir değişken oluşturduk ve işlem sonuçlarını bu değişkene atadık.

R-squared sonucunda gördüğümüz 0.612 bağımsız değişkenin bağımlı değişkendeki değişkenliği açıklama başarısıdır. Televizyon değişkeni satış değişkeninde yer alan değişkenliğin %60'ını açıklayabilmektedir.

Adj. R-squared değeri ise R karenin her parametre eklenmesine karşı olan duyarlılığını törpüleyen, düzenleyen ve duyarlılığı daha az olan bir metriktir.

F-istatistiği modelin anlamlılığının anlaşılması için kurulan test istatistiğidir.

Prob (F-istatistik) modelin anlamlılığının anlaşılması için kurulan test istatistiğinin p value değeridir.

Model parametlerini görüyoruz. Bu şekilde en başta anlattığımız β_0 ve β_1 değerlerini görüyoruz.

```
In [10]: model.params
Out[10]: const    7.032594
         TV       0.047537
         dtype: float64
```

Katsayıların güven aralıklarına erişiyoruz.

```
In [11]: model.conf_int()
```

Out[11]:

	0	1
const	6.129719	7.935468
TV	0.042231	0.052843

Modelin anlamlılığına ilişkin istatistiklere erişiyoruz.

```
In [12]: model.f_pvalue
Out[12]: 1.4673897001947095e-42
```

Modelin anlamlılığına ilişkin model değerlendirme istatistiklerine ulaşıyoruz. Bu şekilde hata kareler ortalamasına erişmiş olduk. Modelin başarısıyla ilgili ilk istatistiğe eriştik. Sales

değişkeninin ortalaması 14 iken hata kareler ortalamasına baktığımızda birim başına yapacağı böyle bir hata çok kötü bir modelleme olduğunu gösteriyor.

```
In [13]: model.mse_model
Out[13]: 3314.6181668686486
```

Modelin tahmin ettiği değerlere erişiyoruz.

```
In [14]: model.fittedvalues[0:5]
Out[14]: 0    17.970775
         1     9.147974
         2     7.850224
         3    14.234395
         4    15.627218
         dtype: float64
```

Gerçek değerleri getirip göz ile karşılaştırmasını yapıyoruz.

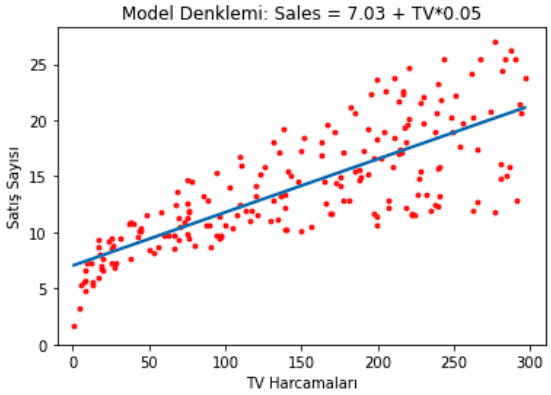
```
In [15]: y[0:5]
Out[15]: 0    22.1
         1    10.4
         2     9.3
         3    18.5
         4    12.9
         Name: sales, dtype: float64
```

Kurduğumuz modelin denklemini oluşturuyoruz.

```
In [16]: print("Sales = " +  str("%.2f" % model.params[0]) + " + TV" + "*" + str("%.2f" % model.params[1]))
         Sales = 7.03 + TV*0.05
```

Modelimizin görsel anlamda neyi ifade ettiğini gösteriyoruz.

```
In [17]: g = sns.regplot(df["TV"], df["sales"], ci=None, scatter_kws={'color':'r', 's':9})
         g.set_title("Model Denklemi: Sales = 7.03 + TV*0.05")
         g.set_ylabel("Satış Sayısı")
         g.set_xlabel("TV Harcamaları")
         import matplotlib.pyplot as plt
         plt.xlim(-10,310)
         plt.ylim(bottom=0);
```



3.3.2. Tahmin

Tahmin işlemini sağlıklı yapmak için kurduğumuz modeli sklearnde tekrardan kurup bunun üzerinden devam ediyoruz.

```
1. from sklearn.linear_model import LinearRegression
2. X = df[["TV"]]
3. y = df["sales"]
4. reg = LinearRegression()
5. model = reg.fit(X, y)
6. model.intercept_
7. model.coef_
8. model.score(X,y)
9. model.predict(X)[0:10]
```

Modelin tahmin etmesi uygulaması için girilen tahmin edilmesi gereken örnek değeri denklemde yerine yerleştirip sonucu bize çıktı oluşturucaktır.

```
In [22]: model.predict([[30]])
Out[22]: array([8.45869276])
```

Kurduğumuz model ile tahmin etme işlemini gerçekleştirmiş olduk.

3.4. ÇOKLU DOĞRUSAL REGRESYON

3.4.1. Modelleme ve Tahmin

```
1. import pandas as pd
2. ad = pd.read_csv("Advertising.csv", usecols = [1,2,3,4])
3. df = ad.copy()
4. df.head()
5. from sklearn.model_selection import train_test_split, cross_val_score,
   cross_val_predict
6. X = df.drop("sales", axis = 1)
7. y = df["sales"]
8. X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20,
   random_state= 42)
9. X_train.shape
10. y_train.shape
11. X_test.shape
12. y_test.shape
13. training = df.copy()
14. training.shape
15. ## Statsmodels
16. lm = sm.OLS(y_train, X_train)
17. model = lm.fit()
18. model.summary()
19. model.summary().tables[1]
20. ## scikit-learn model
21. lm = LinearRegression()
22. model = lm.fit(X_train, y_train)
23. model.intercept_
24. model.coef_
25. ## Tahmin
26. Model denklemi:
27. Sales = 2.97 + TV0.04 + radio0.18 + newspaper*0.002
28. Örneğin 30 birim TV harcaması, 10 birim radio harcamasi, 40 birimde gazete harcaması
    olduğunda satışların tahmini değeri ne olur?
29. yeni_veri = [[30], [10],[40]]
30. yeni_veri = pd.DataFrame(yeni_veri).T
```

Out[71]:

OLS Regression Results

Dep. Variable:	sales	**R-squared (uncentered):**	0.982
Model:	OLS	**Adj. R-squared (uncentered):**	0.982
Method:	Least Squares	**F-statistic:**	2935.
Date:	Sun, 20 Nov 2022	**Prob (F-statistic):**	1.28e-137
Time:	18:18:30	**Log-Likelihood:**	-336.65
No. Observations:	160	**AIC:**	679.3
Df Residuals:	157	**BIC:**	688.5
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
TV	0.0531	0.001	36.467	0.000	0.050	0.056
radio	0.2188	0.011	20.138	0.000	0.197	0.240
newspaper	0.0239	0.008	3.011	0.003	0.008	0.040

Omnibus:	11.405	**Durbin-Watson:**	1.895
Prob(Omnibus):	0.003	**Jarque-Bera (JB):**	15.574
Skew:	-0.432	**Prob(JB):**	0.000415
Kurtosis:	4.261	**Cond. No.**	13.5

$$Sales = 2.97 + TV0.04 + radio0.18 + newspaper * 0.002$$

3.4.2. Model Tuning (Doğrulama)

Model doğrulama işleminde k katlı cross validation gibi yöntemlerle yapmak istediğimiz durum elde ettiğimiz hataları daha doğru hatalar olarak elde etmeye çalışmak çabasıdır.

```
1.  df.head()
2.
3.  X = df.drop('sales', axis=1)
4.  y = df["sales"]
5.  X_train, X_test, y_train, y_test = train_test_split(X, y,
6.                                                      test_size=0.20,
7.                                                      random_state=144)
8.  lm = LinearRegression()
9.  model = lm.fit(X_train, y_train)
10. np.sqrt(mean_squared_error(y_train, model.predict(X_train)))
11. np.sqrt(mean_squared_error(y_test, model.predict(X_test)))
12. model.score(X_train, y_train)
13. cross_val_score(model, X_train, y_train, cv = 10, scoring = "r2").mean()
14.
15. np.sqrt(-cross_val_score(model,
16.                 X_train,
17.                 y_train,
18.                 cv = 10,
19.                 scoring = "neg_mean_squared_error")).mean()
20. np.sqrt(-cross_val_score(model,
21.                 X_test,
```

```
                y_test,
                cv = 10,
                scoring = "neg_mean_squared_error")).mean()
```

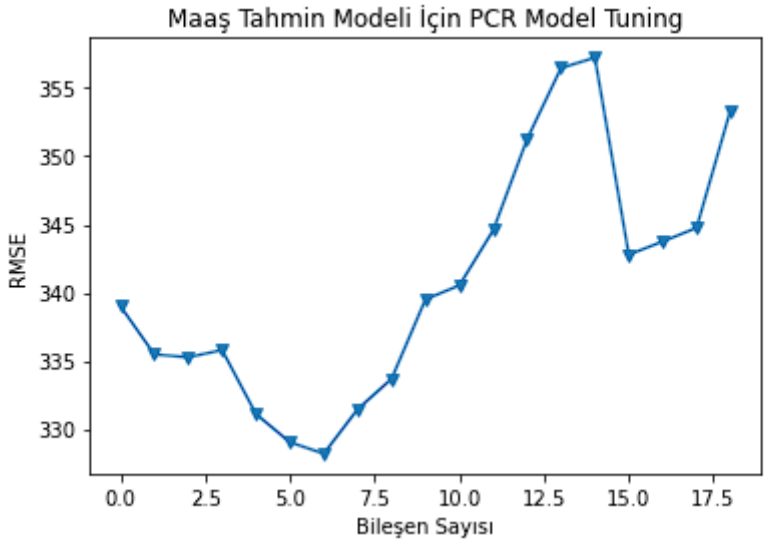
3.5. PCR

```
import pandas as pd
import numpy as np
hit = pd.read_csv("Hitters.csv")
df = hit.copy()
df = df.dropna()
df.head()
df.info()
df.describe().T
dms = pd.get_dummies(df[['League', 'Division', 'NewLeague']])
dms.head()
y = df["Salary"]
X_ = df.drop(["Salary","League","Division","NewLeague"], axis = 1).astype("float64")
X_.head()
X = pd.concat([X_, dms[["League_N", "Division_W","NewLeague_N"]]], axis = 1)
X.head()
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.25,
                                                    random_state=42)

print("X_train", X_train.shape)
print("y_train",y_train.shape)
print("X_test",X_test.shape)
print("y_test",y_test.shape)
training = df.copy()
print("training", training.shape)
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale
pca = PCA()
X_reduced_train = pca.fit_transform(scale(X_train))
X_reduced_train[0:1,:]
np.cumsum(np.round(pca.explained_variance_ratio_, decimals = 4)*100)[0:5]
lm = LinearRegression()
pcr_model = lm.fit(X_reduced_train, y_train)
pcr_model.intercept_
pcr_model.coef_
## Tahmin
y_pred = pcr_model.predict(X_reduced_train)
y_pred[0:5]
np.sqrt(mean_squared_error(y_train, y_pred))
df["Salary"].mean()
r2_score(y_train, y_pred)
pca2 = PCA()
X_reduced_test = pca2.fit_transform(scale(X_test))
y_pred = pcr_model.predict(X_reduced_test)
np.sqrt(mean_squared_error(y_test, y_pred))
## Model Tuning
lm = LinearRegression()
pcr_model = lm.fit(X_reduced_train[:,0:10], y_train)
y_pred = pcr_model.predict(X_reduced_test[:,0:10])
print(np.sqrt(mean_squared_error(y_test, y_pred)))
from sklearn import model_selection
cv_10 = model_selection.KFold(n_splits = 10,
                              shuffle = True,
                              random_state = 1)

lm = LinearRegression()
RMSE = []
```

```
for i in np.arange(1, X_reduced_train.shape[1] + 1):
    score = np.sqrt(-1*model_selection.cross_val_score(lm,
                                                       X_reduced_train[:,:i],
                                                       y_train.ravel(),
                                                       cv=cv_10,
                                                       scoring='neg_mean_squared_error').mean())
    RMSE.append(score)
plt.plot(RMSE, '-v')
plt.xlabel('Bileşen Sayısı')
plt.ylabel('RMSE')
plt.title('Maaş Tahmin Modeli İçin PCR Model Tuning');
lm = LinearRegression()
pcr_model = lm.fit(X_reduced_train[:,0:6], y_train)
y_pred = pcr_model.predict(X_reduced_train[:,0:6])
print(np.sqrt(mean_squared_error(y_train, y_pred)))
y_pred = pcr_model.predict(X_reduced_test[:,0:6])
print(np.sqrt(mean_squared_error(y_test, y_pred)))
```



3.6. PLS

```
hit = pd.read_csv("Hitters.csv")
df = hit.copy()
df = df.dropna()
ms = pd.get_dummies(df[['League', 'Division', 'NewLeague']])
y = df["Salary"]
X_ = df.drop(['Salary', 'League', 'Division', 'NewLeague'], axis=1).astype('float64')
X = pd.concat([X_, dms[['League_N', 'Division_W', 'NewLeague_N']]], axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

from sklearn.cross_decomposition import PLSRegression, PLSSVD
pls_model = PLSRegression().fit(X_train, y_train)
pls_model.coef_
## Tahmin
X_train.head()
pls_model.predict(X_train)[0:10]
y_pred = pls_model.predict(X_train)
np.sqrt(mean_squared_error(y_train, y_pred))
r2_score(y_train, y_pred)
y_pred = pls_model.predict(X_test)
```

```
np.sqrt(mean_squared_error(y_test, y_pred))
## Model Tuning
#CV
cv_10 = model_selection.KFold(n_splits=10, shuffle=True, random_state=1)
#Hata hesaplamak için döngü
RMSE = []
for i in np.arange(1, X_train.shape[1] + 1):
    pls = PLSRegression(n_components=i)
    score = np.sqrt(-1*cross_val_score(pls, X_train, y_train, cv=cv_10,
scoring='neg_mean_squared_error').mean())
    RMSE.append(score)
#Sonuçların Görselleştirilmesi
plt.plot(np.arange(1, X_train.shape[1] + 1), np.array(RMSE), '-v', c = "r")
plt.xlabel('Bileşen Sayısı')
plt.ylabel('RMSE')
plt.title('Salary');
pls_model = PLSRegression(n_components = 2).fit(X_train, y_train)
y_pred = pls_model.predict(X_test)
np.sqrt(mean_squared_error(y_test, y_pred))
```

3.7. Ridge Regresyon

```
hit = pd.read_csv("Hitters.csv")
df = hit.copy()
df = df.dropna()
ms = pd.get_dummies(df[['League', 'Division', 'NewLeague']])
y = df["Salary"]
X_ = df.drop(['Salary', 'League', 'Division', 'NewLeague'],
axis=1).astype('float64')
X = pd.concat([X_, dms[['League_N', 'Division_W', 'NewLeague_N']]], axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.25,
                                                    random_state=42)

from sklearn.linear_model import Ridge
ridge_model = Ridge(alpha = 0.1).fit(X_train, y_train)
ridge_model
ridge_model.coef_
10**np.linspace(10,-2,100)*0.5
lambdalar = 10**np.linspace(10,-2,100)*0.5
ridge_model = Ridge()
katsayilar = []
for i in lambdalar:
    ridge_model.set_params(alpha = i)
    ridge_model.fit(X_train, y_train)
    katsayilar.append(ridge_model.coef_)


ax = plt.gca()
ax.plot(lambdalar, katsayilar)
ax.set_xscale('log')
plt.xlabel('Lambda(Alpha) Değerleri')
plt.ylabel('Katsayılar/Ağırlıklar')
plt.title('Düzenlileştirmenin Bir Fonksiyonu Olarak Ridge Katsayıları');

## Tahmin
y_pred = ridge_model.predict(X_test)
np.sqrt(mean_squared_error(y_test, y_pred))
## Model Tuning
lambdalar = 10**np.linspace(10,-2,100)*0.5
lambdalar[0:5]
from sklearn.linear_model import RidgeCV
ridge_cv = RidgeCV(alphas = lambdalar,
                   scoring = "neg_mean_squared_error",
```

```
                    normalize = True)
ridge_cv.fit(X_train, y_train)
ridge_cv.alpha_
ridge_tuned = Ridge(alpha = ridge_cv.alpha_,
                    normalize = True).fit(X_train,y_train)
np.sqrt(mean_squared_error(y_test, ridge_tuned.predict(X_test)))
```

3.8. Lasso Regresyon

```
hit = pd.read_csv("Hitters.csv")
df = hit.copy()
df = df.dropna()
ms = pd.get_dummies(df[['League', 'Division', 'NewLeague']])
y = df["Salary"]
X_ = df.drop(['Salary', 'League', 'Division', 'NewLeague'],
axis=1).astype('float64')
X = pd.concat([X_, dms[['League_N', 'Division_W', 'NewLeague_N']]], axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.25,
                                                    random_state=42)

from sklearn.linear_model import Lasso
lasso_model = Lasso(alpha = 0.1).fit(X_train, y_train)
lasso_model
lasso_model.coef_
lasso = Lasso()
lambdalar = 10**np.linspace(10,-2,100)*0.5
katsayilar = []
for i in lambdalar:
    lasso.set_params(alpha=i)
    lasso.fit(X_train, y_train)
    katsayilar.append(lasso.coef_)

ax = plt.gca()
ax.plot(lambdalar*2, katsayilar)
ax.set_xscale('log')
plt.axis('tight')
plt.xlabel('alpha')
plt.ylabel('weights')
## Tahmin
lasso_model.predict(X_test)
y_pred = lasso_model.predict(X_test)
np.sqrt(mean_squared_error(y_test, y_pred))
## Model Tuning
from sklearn.linear_model import LassoCV
lasso_cv_model = LassoCV(alphas = None,
                         cv = 10,
                         max_iter = 10000,
                         normalize = True)

lasso_cv_model.fit(X_train,y_train)
lasso_cv_model.alpha_
lasso_tuned = Lasso(alpha = lasso_cv_model.alpha_)
lasso_tuned.fit(X_train, y_train)
y_pred = lasso_tuned.predict(X_test)
np.sqrt(mean_squared_error(y_test, y_pred))
```

3.9. ElasticNet Regresyon

```
hit = pd.read_csv("Hitters.csv")
df = hit.copy()
df = df.dropna()
```

```
ms = pd.get_dummies(df[['League', 'Division', 'NewLeague']])
y = df["Salary"]
X_ = df.drop(['Salary', 'League', 'Division', 'NewLeague'],
axis=1).astype('float64')
X = pd.concat([X_, dms[['League_N', 'Division_W', 'NewLeague_N']]], axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.25,
                                                    random_state=42)


from sklearn.linear_model import ElasticNet
enet_model = ElasticNet().fit(X_train, y_train)
enet_model.coef_
enet_model.intercept_
## Tahmin
enet_model
enet_model.predict(X_test)
y_pred = enet_model.predict(X_test)
np.sqrt(mean_squared_error(y_test, y_pred))
r2_score(y_test, y_pred)

## Model Tuning
from sklearn.linear_model import ElasticNetCV
enet_cv_model = ElasticNetCV(cv = 10, random_state = 0).fit(X_train, y_train)
enet_cv_model.alpha_
enet_cv_model
enet_tuned = ElasticNet(alpha = enet_cv_model.alpha_).fit(X_train,y_train)
y_pred = enet_tuned.predict(X_test)
np.sqrt(mean_squared_error(y_test, y_pred))
```

3.10. KNN

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split,
GridSearchCV,cross_val_score
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
from sklearn.preprocessing import scale
from sklearn import model_selection
from sklearn.tree import DecisionTreeRegressor, DecisionTreeClassifier
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import BaggingRegressor

from warnings import filterwarnings
filterwarnings('ignore')

# KNN

hit = pd.read_csv("Hitters.csv")
df = hit.copy()
df = df.dropna()
dms = pd.get_dummies(df[['League', 'Division', 'NewLeague']])
y = df["Salary"]
X_ = df.drop(['Salary', 'League', 'Division', 'NewLeague'],
axis=1).astype('float64')
X = pd.concat([X_, dms[['League_N', 'Division_W', 'NewLeague_N']]], axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.25,
                                                    random_state=42)
```

```
27.
28. knn_model = KNeighborsRegressor().fit(X_train, y_train)
29. knn_model
30. knn_model.n_neighbors
31. knn_model.effective_metric_
32.
33. ## Tahmin
34. y_pred = knn_model.predict(X_test)
35. np.sqrt(mean_squared_error(y_test, y_pred))
36. RMSE = []
37. for k in range(10):
38.     k = k+1
39.     knn_model = KNeighborsRegressor(n_neighbors = k).fit(X_train, y_train)
40.     y_pred = knn_model.predict(X_train)
41.     rmse = np.sqrt(mean_squared_error(y_train,y_pred))
42.     RMSE.append(rmse)
43.     print("k =" , k , "için RMSE değeri: ", rmse)
44.
45. ## Model Tuning
46. from sklearn.model_selection import GridSearchCV
47. knn_params = {'n_neighbors': np.arange(1,30,1)}
48. knn = KNeighborsRegressor()
49. knn_cv_model = GridSearchCV(knn, knn_params, cv = 10)
50. knn_cv_model.fit(X_train, y_train)
51. knn_cv_model.best_params_["n_neighbors"]
52. RMSE = []
53. RMSE_CV = []
54. for k in range(10):
55.     k = k+1
56.     knn_model = KNeighborsRegressor(n_neighbors = k).fit(X_train, y_train)
57.     y_pred = knn_model.predict(X_train)
58.     rmse = np.sqrt(mean_squared_error(y_train,y_pred))
59.     rmse_cv = np.sqrt(-1*cross_val_score(knn_model, X_train, y_train, cv=10,
60.                                          scoring                              =
    "neg_mean_squared_error").mean())
61.     RMSE.append(rmse)
62.     RMSE_CV.append(rmse_cv)
63.     print("k =" , k , "için RMSE değeri: ", rmse, "RMSE_CV değeri: ", rmse_cv )
64.
65. knn_tuned=KNeighborsRegressor(n_neighbors                                     =
    knn_cv_model.best_params_["n_neighbors"])
66. knn_tuned.fit(X_train, y_train)
67. np.sqrt(mean_squared_error(y_test, knn_tuned.predict(X_test)))
```

3.11. SVR

```
1.  hit = pd.read_csv("Hitters.csv")
2.  df = hit.copy()
3.  df = df.dropna()
4.  dms = pd.get_dummies(df[['League', 'Division', 'NewLeague']])
5.  y = df["Salary"]
6.  X_ = df.drop(['Salary', 'League', 'Division', 'NewLeague'], axis=1).astype('float64')
7.  X = pd.concat([X_, dms[['League_N', 'Division_W', 'NewLeague_N']]], axis=1)
8.  X_train, X_test, y_train, y_test = train_test_split(X, y,
9.                                                      test_size=0.25,
10.                                                     random_state=42)
11.
12. X_train = pd.DataFrame(X_train["Hits"])
```

```
X_test = pd.DataFrame(X_test["Hits"])
from sklearn.svm import SVR
svr_model = SVR("linear").fit(X_train, y_train)
svr_model.predict(X_train)[0:5]
print ("y = {0} + {1} x".format(svr_model.intercept_[0],
                                svr_model.coef_[0][0]))

X_train["Hits"][0:1]
-48.69756097561513 + 4.969512195122093*91
y_pred = svr_model.predict(X_train)
plt.scatter(X_train, y_train)
plt.plot(X_train,y_pred, color = "r")
from sklearn.linear_model import LinearRegression
lm_model = LinearRegression().fit(X_train, y_train)
lm_pred  = lm_model.predict(X_train)
print("y = {0} + {1} x".format(lm_model.intercept_, lm_model.coef_[0]))
-8.814095480334572 + 5.1724561354706875*91

plt.scatter(X_train, y_train, alpha=0.5, s=23)
plt.plot(X_train, lm_pred, 'g')
plt.plot(X_train, y_pred, color='r')

plt.xlabel("Atış Sayısı(Hits)")
plt.ylabel("Maaş (Salary)")

## Tahmin
print ("y = {0} + {1} x".format(svr_model.intercept_[0], svr_model.coef_[0][0]))
svr_model.predict([[91]])
y_pred = svr_model.predict(X_test)
np.sqrt(mean_squared_error(y_test, y_pred))
svr_model

## Model Tuning
svr_model
svr_params = {"C": np.arange(0.1,2,0.1)}
svr_cv_model = GridSearchCV(svr_model, svr_params, cv = 10).fit(X_train,y_train)
pd.Series(svr_cv_model.best_params_)[0]
svr_tuned = SVR("linear",
                C = pd.Series(svr_cv_model.best_params_)[0]).fit(X_train, y_train)

y_pred = svr_tuned.predict(X_test)
np.sqrt(mean_squared_error(y_test, y_pred))
np.sqrt(mean_squared_error(y_test, y_pred))
```

3.12. YSA (Çok Katmanlı Algılayıcı)

```
hit = pd.read_csv("Hitters.csv")
df = hit.copy()
df = df.dropna()
dms = pd.get_dummies(df[['League', 'Division', 'NewLeague']])
y = df["Salary"]
X_ = df.drop(['Salary', 'League', 'Division', 'NewLeague'],
axis=1).astype('float64')
X = pd.concat([X_, dms[['League_N', 'Division_W', 'NewLeague_N']]], axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.25,
                                                    random_state=42)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
```

```
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
from sklearn.neural_network import MLPRegressor
mlp_model = MLPRegressor(hidden_layer_sizes = (100,20)).fit(X_train_scaled, y_train)
mlp_model
mlp_model.n_layers_
mlp_model.hidden_layer_sizes

## Tahmin
y_pred = mlp_model.predict(X_test_scaled)
np.sqrt(mean_squared_error(y_test, y_pred))
## Model Tuning
mlp_model
mlp_params = {'alpha': [0.1, 0.01,0.02,0.005],
              'hidden_layer_sizes': [(20,20),(100,50,150),(300,200,150)],
              'activation': ['relu','logistic']}

mlp_cv_model = GridSearchCV(mlp_model, mlp_params, cv = 10)
mlp_cv_model.fit(X_train_scaled, y_train)
mlp_cv_model.best_params_
mlp_tuned = MLPRegressor(alpha = 0.02, hidden_layer_sizes = (100,50,150))
mlp_tuned.fit(X_train_scaled, y_train)
y_pred = mlp_tuned.predict(X_test_scaled)
np.sqrt(mean_squared_error(y_test, y_pred))
```

3.13. CART

```
hit = pd.read_csv("Hitters.csv")
df = hit.copy()
df = df.dropna()
dms = pd.get_dummies(df[['League', 'Division', 'NewLeague']])
y = df["Salary"]
X_ = df.drop(['Salary', 'League', 'Division', 'NewLeague'],
axis=1).astype('float64')
X = pd.concat([X_, dms[['League_N', 'Division_W', 'NewLeague_N']]], axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.25,
                                                    random_state=42)


X_train = pd.DataFrame(X_train["Hits"])
X_test = pd.DataFrame(X_test["Hits"])
cart_model = DecisionTreeRegressor(min_samples_split = 2)
?cart_model
cart_model.fit(X_train, y_train)
X_grid = np.arange(min(np.array(X_train)),max(np.array(X_train)), 0.01)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_grid, cart_model.predict(X_grid), color = 'blue')
plt.title('CART REGRESON AĞACI')
plt.xlabel('Atış Sayısı(Hits)')
plt.ylabel('Maaş (Salary)') ;
#!pip install skompiler
from skompiler import skompile
print(skompile(cart_model.predict).to('python/code'))

## Tahmin
x = [91]
(345.2011551724138 if x[0] <= 117.5 else ((((1300.0 if x[0] <= 118.5 else
    641.0) if x[0] <= 122.5 else 1468.5236666666667) if x[0] <= 125.5 else
    621.9679230769232) if x[0] <= 143.0 else (958.6111111111111 if x[0] <=
    150.5 else 2460.0) if x[0] <= 151.5 else 499.1666666666667 if x[0] <=
    157.5 else 892.5402413793104) if x[0] <= 225.5 else 1975.0)
cart_model.predict(X_test)[0:5]
```

```
cart_model.predict([[91]])
y_pred =cart_model.predict(X_test)
np.sqrt(mean_squared_error(y_test, y_pred))
## Model Tuning
cart_model = DecisionTreeRegressor()
cart_model.fit(X_train, y_train)
y_pred = cart_model.predict(X_test)
np.sqrt(mean_squared_error(y_test, y_pred))
cart_params = {"min_samples_split": range(2,100),
               "max_leaf_nodes": range(2,10)}
cart_cv_model = GridSearchCV(cart_model, cart_params, cv = 10)
cart_cv_model.fit(X_train, y_train)
cart_cv_model.best_params_
cart_tuned = DecisionTreeRegressor(max_leaf_nodes = 9, min_samples_split = 37)
cart_tuned.fit(X_train, y_train)
y_pred = cart_tuned.predict(X_test)
np.sqrt(mean_squared_error(y_test, y_pred))
```

3.14. Bagged Trees Regresyon

```
hit = pd.read_csv("Hitters.csv")
df = hit.copy()
df = df.dropna()
dms = pd.get_dummies(df[['League', 'Division', 'NewLeague']])
y = df["Salary"]
X_ = df.drop(['Salary', 'League', 'Division', 'NewLeague'],
axis=1).astype('float64')
X = pd.concat([X_, dms[['League_N', 'Division_W', 'NewLeague_N']]], axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.25,
                                                    random_state=42)

bag_model = BaggingRegressor(bootstrap_features = True)
bag_model.fit(X_train, y_train)
bag_model.n_estimators
bag_model.estimators_
bag_model.estimators_samples_
bag_model.estimators_features_
bag_model.estimators_[1]

## Tahmin

y_pred = bag_model.predict(X_test)

np.sqrt(mean_squared_error(y_test, y_pred))

iki_y_pred = bag_model.estimators_[1].fit(X_train, y_train).predict(X_test)

np.sqrt(mean_squared_error(y_test, iki_y_pred))

yedi_y_pred = bag_model.estimators_[4].fit(X_train, y_train).predict(X_test)

np.sqrt(mean_squared_error(y_test, yedi_y_pred))

# Model Tuning
bag_model = BaggingRegressor(bootstrap_features = True)
bag_model.fit(X_train, y_train)
bag_params = {"n_estimators": range(2,20)}
bag_cv_model = GridSearchCV(bag_model, bag_params, cv = 10)
bag_cv_model.fit(X_train, y_train)
bag_cv_model.best_params_
bag_tuned = BaggingRegressor( n_estimators = 14, random_state = 45)
bag_tuned.fit(X_train, y_train)
y_pred = bag_tuned.predict(X_test)
```

```
44. np.sqrt(mean_squared_error(y_test, y_pred))
```

3.15. Random Forest

```
1.  hit = pd.read_csv("Hitters.csv")
2.  df = hit.copy()
3.  df = df.dropna()
4.  dms = pd.get_dummies(df[['League', 'Division', 'NewLeague']])
5.  y = df["Salary"]
6.  X_ = df.drop(['Salary', 'League', 'Division', 'NewLeague'],
    axis=1).astype('float64')
7.  X = pd.concat([X_, dms[['League_N', 'Division_W', 'NewLeague_N']]], axis=1)
8.  X_train, X_test, y_train, y_test = train_test_split(X, y,
9.                                                      test_size=0.25,
10.                                                     random_state=42)
11. from sklearn.ensemble import RandomForestRegressor
12. rf_model = RandomForestRegressor(random_state = 42)
13. rf_model.fit(X_train, y_train)
14.
15. ## Tahmin
16. rf_model.predict(X_test)[0:5]
17. y_pred = rf_model.predict(X_test)
18. np.sqrt(mean_squared_error(y_test, y_pred))
19.
20. ## Model Tuning
21. rf_params = {'max_depth': list(range(1,10)),
22.             'max_features': [3,5,10,15],
23.             'n_estimators' : [100, 200, 500, 1000, 2000]}
24.
25. rf_model = RandomForestRegressor(random_state = 42)
26. rf_cv_model = GridSearchCV(rf_model,
27.                            rf_params,
28.                            cv = 10,
29.                             n_jobs = -1)
30. rf_cv_model.fit(X_train, y_train)
31. rf_cv_model.best_params_
32. rf_tuned = RandomForestRegressor(max_depth  = 8,
33.                                  max_features = 3,
34.                                  n_estimators =200)
35.
36. rf_tuned.fit(X_train, y_train)
37. y_pred = rf_tuned.predict(X_test)
38. np.sqrt(mean_squared_error(y_test, y_pred))
39. Importance = pd.DataFrame({"Importance": rf_tuned.feature_importances_*100},
40.                           index = X_train.columns)
41. Importance.sort_values(by = "Importance",
42.                        axis = 0,
43.                        ascending = True).plot(kind ="barh", color = "r")
44.
45. plt.xlabel("Değişken Önem Düzeyleri")
```

3.16. Gradient Boosting Machines

```
1.  from sklearn.ensemble import GradientBoostingRegressor
2.  gbm_model = GradientBoostingRegressor()
3.  gbm_model.fit(X_train, y_train)
4.  ##Tahmin
5.  y_pred = gbm_model.predict(X_test)
6.  np.sqrt(mean_squared_error(y_test, y_pred))
```

```
##Model Tuning
gbm_params = {
    'learning_rate': [0.001, 0.01, 0.1, 0.2],
    'max_depth': [3, 5, 8,50,100],
    'n_estimators': [200, 500, 1000, 2000],
    'subsample': [1,0.5,0.75],
}
gbm = GradientBoostingRegressor()
gbm_cv_model = GridSearchCV(gbm, gbm_params, cv = 10, n_jobs = -1, verbose = 2)
gbm_cv_model.fit(X_train, y_train)
gbm_cv_model.best_params_
gbm_tuned = GradientBoostingRegressor(learning_rate = 0.1,
                                      max_depth = 5,
                                      n_estimators = 200,
                                      subsample = 0.5)

gbm_tuned = gbm_tuned.fit(X_train,y_train)
y_pred = gbm_tuned.predict(X_test)
np.sqrt(mean_squared_error(y_test, y_pred))
Importance = pd.DataFrame({"Importance": gbm_tuned.feature_importances_*100},
                          index = X_train.columns)
```

3.17. XGBoost

```
hit = pd.read_csv("Hitters.csv")
df = hit.copy()
df = df.dropna()
dms = pd.get_dummies(df[['League', 'Division', 'NewLeague']])
y = df["Salary"]
X_ = df.drop(['Salary', 'League', 'Division', 'NewLeague'],
axis=1).astype('float64')
X = pd.concat([X_, dms[['League_N', 'Division_W', 'NewLeague_N']]], axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.25,
                                                    random_state=42)


#!pip install xgboost

import xgboost as xgb

DM_train = xgb.DMatrix(data = X_train, label = y_train)
DM_test = xgb.DMatrix(data = X_test, label = y_test)

from xgboost import XGBRegressor

xgb_model = XGBRegressor().fit(X_train, y_train)

## Tahmin

y_pred = xgb_model.predict(X_test)
np.sqrt(mean_squared_error(y_test, y_pred))

## Model Tuning

xgb_model

xgb_grid = {
    'colsample_bytree': [0.4, 0.5,0.6,0.9,1],
    'n_estimators':[100, 200, 500, 1000],
    'max_depth': [2,3,4,5,6],
    'learning_rate': [0.1, 0.01, 0.5]
}

```

```
40.
41. xgb = XGBRegressor()
42.
43. xgb_cv = GridSearchCV(xgb,
44.                        param_grid = xgb_grid,
45.                        cv = 10,
46.                        n_jobs = -1,
47.                        verbose = 2)
48.
49.
50. xgb_cv.fit(X_train, y_train)
51.
52. xgb_cv.best_params_
53.
54. xgb_tuned = XGBRegressor(colsample_bytree = 0.9,
55.                           learning_rate = 0.01,
56.                           max_depth = 5,
57.                           n_estimators = 1000)
58.
59. xgb_tuned = xgb_tuned.fit(X_train,y_train)
60.
61. y_pred = xgb_tuned.predict(X_test)
62. np.sqrt(mean_squared_error(y_test, y_pred))
```

3.18. Light GBM

```
1.  hit = pd.read_csv("Hitters.csv")
2.  df = hit.copy()
3.  df = df.dropna()
4.  dms = pd.get_dummies(df[['League', 'Division', 'NewLeague']])
5.  y = df["Salary"]
6.  X_ = df.drop(['Salary', 'League', 'Division', 'NewLeague'],
    axis=1).astype('float64')
7.  X = pd.concat([X_, dms[['League_N', 'Division_W', 'NewLeague_N']]], axis=1)
8.  X_train, X_test, y_train, y_test = train_test_split(X, y,
9.                                                      test_size=0.25,
10.                                                     random_state=42)
11.
12.
13. #!pip install lightgbm
14.
15. from lightgbm import LGBMRegressor
16.
17. #conda install -c conda-forge lightgbm
18.
19. from lightgbm import LGBMRegressor
20.
21. lgbm = LGBMRegressor()
22. lgbm_model = lgbm.fit(X_train, y_train)
23.
24. ## Tahmin
25.
26. y_pred = lgbm_model.predict(X_test,
27.                              num_iteration = lgbm_model.best_iteration_)
28.
29. np.sqrt(mean_squared_error(y_test, y_pred))
30.
31. ## Model Tuning
32.
33. lgbm_model
34.
35. lgbm_grid = {
36.     'colsample_bytree': [0.4, 0.5,0.6,0.9,1],
37.     'learning_rate': [0.01, 0.1, 0.5,1],
```

```
    'n_estimators': [20, 40, 100, 200, 500,1000],
    'max_depth': [1,2,3,4,5,6,7,8] }

lgbm = LGBMRegressor()
lgbm_cv_model = GridSearchCV(lgbm, lgbm_grid, cv=10, n_jobs = -1, verbose = 2)

lgbm_cv_model.fit(X_train, y_train)

lgbm_cv_model.best_params_

lgbm_tuned = LGBMRegressor(learning_rate = 0.1,
                           max_depth = 7,
                           n_estimators = 40,
                          colsample_bytree = 0.6)

lgbm_tuned = lgbm_tuned.fit(X_train,y_train)

y_pred = lgbm_tuned.predict(X_test)

np.sqrt(mean_squared_error(y_test, y_pred))
```

3.19. CatBoost

```
hit = pd.read_csv("Hitters.csv")
df = hit.copy()
df = df.dropna()
dms = pd.get_dummies(df[['League', 'Division', 'NewLeague']])
y = df["Salary"]
X_ = df.drop(['Salary', 'League', 'Division', 'NewLeague'],
axis=1).astype('float64')
X = pd.concat([X_, dms[['League_N', 'Division_W', 'NewLeague_N']]], axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.25,
                                                    random_state=42)


!pip install catboost

from catboost import CatBoostRegressor

catb = CatBoostRegressor()
catb_model = catb.fit(X_train, y_train)

## Tahmin

y_pred = catb_model.predict(X_test)
np.sqrt(mean_squared_error(y_test, y_pred))

## Model Tuning

catb_grid = {
    'iterations': [200,500,1000,2000],
    'learning_rate': [0.01, 0.03, 0.05, 0.1],
    'depth': [3,4,5,6,7,8] }

catb = CatBoostRegressor()
catb_cv_model = GridSearchCV(catb, catb_grid, cv=5, n_jobs = -1, verbose = 2)

catb_cv_model.fit(X_train, y_train)

catb_cv_model.best_params_

catb_tuned = CatBoostRegressor(iterations = 200,
                               learning_rate = 0.01,
```

```
                                depth = 8)

catb_tuned = catb_tuned.fit(X_train,y_train)

y_pred = catb_tuned.predict(X_test)
np.sqrt(mean_squared_error(y_test, y_pred))
```

4. SONUÇ

Yaptığımız araştırmalar ve çalışmalarda görüldüğü üzere makine öğrenmesinin bir dalı olan regresyonlar hayatımızın her noktasında var olması gereken hayatımızı kolaylaştıran ve ileriye dayalı tahmin yapmakta çok önemli rol oynamaktadır. Evren üzerinde oluşan ve oluşturulan her noktada veri vardır. Her şeyin verilerden oluştuğunu ve oluşan veriler üzerinde analiz yapmak ileriye dönük tahminler yapmak hayatımızı çok kolaylaştırdığı gibi hayatımızda yeni bir çağ açmak için de büyük bir faktör olacağını makine öğrenimi algoritmaları göstermektedir. Analizin yapılması, bağlantıları görülmesi, çıkarım yapılması gibi birçok faktörün insan gücüyle çok uzun süreçler ve yanlışlık yapılma potansiyelinin çok görülmesine karşın bilgisayarda kurduğumuz makine öğrenimi algoritmaları, regresyon modelleri, sınıflandırma modelleri ile çok hızlı ve hata payının çok aza indirilmesi ile çok büyük adım atılmıştır. Bu raporda bu durum incelenip detaylandırılmıştır. Yapılan çalışmalar sonucunda gördüğümüz sonuçlar gözle görülebilir farklar yaratmıştır.

KAYNAKÇA

[1] https://www.astera.com/type/blog/data-manipulation-tools/

[2] Emre, İlkim Ecem, and Çiğdem Selçukcan Erol. "Veri Analizinde İstatistik mi Veri Madenciliği mi?." Bilişim Teknolojileri Dergisi 10.2 (2017): 161-167.

[3] Oğuzlar, Ayşe. "Veri ön işleme." Erciyes Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi 21 (2003).

[4] ŞEKER, Şadi Evren. "Veri Ön İşleme (Data PreProcessing)."

[5] https://www.datasciencearth.com/herkes-icin-veri-okuryazarligi

[6] https://www.techtarget.com/searchbusinessanalytics/definition/data-visualization

[7] https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python

[8] https://numpy.org/

[9] https://www.geeksforgeeks.org/numpy-in-python-set-1-introduction/

[10] https://pypi.org/project/pandas/

[11] McKinney, Wes. "pandas: a foundational Python library for data analysis and statistics." Python for high performance and scientific computing 14.9 (2011): 1-9.

[12] https://www.section.io/engineering-education/seaborn-tutorial/

[13] https://www.activestate.com/resources/quick-reads/what-is-pyplot-in-matplotlib/

[14] https://en.wikipedia.org/wiki/Statsmodels

[15] Garreta, Raul, and Guillermo Moncecchi. Learning scikit-learn: Machine learning in python. Packt Publishing Ltd, 2013.

[16] https://www.analyticsvidhya.com/blog/2015/01/scikit-learn-python-machine-learning-tool/

[17] https://www.ibm.com/tr-tr/cloud/learn/machine-learning

[18] https://azure.microsoft.com/tr-tr/resources/cloud-computing-dictionary/what-is-machine-learning-platform

[19] https://www.ibm.com/topics/linear-regression#:~:text=Resources,What%20is%20linear%20regression%3F,is%20called%20the%20independent%20variable.

[20] https://en.wikipedia.org/wiki/Linear_regression

[21] https://www.veribilimiokulu.com/basit-dogrusal-regresyon/

[22] https://www.scribbr.com/statistics/multiple-linear-regression/#:~:text=What%20is%20multiple%20linear%20regression,variables%20using%20a%20straight%20line.

[23] https://rpubs.com/esobolewska/pcr-step-by-step

[24] Hilt, Donald E.; Seegrist, Donald W. (1977). Ridge, a computer program for calculating ridge regression estimates

[25] https://www.statisticshowto.com/lasso-regression/

[26] https://analyticsindiamag.com/hands-on-tutorial-on-elasticnet-regression/

[27] https://tr.wikipedia.org/wiki/Do%C4%9Frusal_olmayan_regresyon

[28] https://datasciencebook.ca/regression1.html

[29] https://en.wikipedia.org/wiki/Support_vector_machine

[30] http://uc-r.github.io/ann_regression

[31] https://machinelearningmastery.com/classification-and-regression-trees-for-machine-learning/

[32] https://levelup.gitconnected.com/random-forest-regression-209c0f354c84

[33] https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab

[34] https://en.wikipedia.org/wiki/XGBoost

[35] https://www.veribilimiokulu.com/lightgbm/

[36] https://en.wikipedia.org/wiki/CatBoost

[37] https://www.anaconda.com/

[38] Games, G., Witten, D., Hastie, T., and Tibshirani, R. (2013) An Introduction to Statistical Learning with applications in R, www.StatLearning.com, Springer-Verlag, New York.

[39] Games, G., Witten, D., Hastie, T., and Tibshirani, R. (2013) An Introduction to Statistical Learning with applications in R, www.StatLearning.com, Springer-Verlag, New York.

[40] Iğdır Üni. Fen Bilimleri Enst. Der. / Iğdır Univ. J. Inst. Sci. & Tech. 3(1): 79-84, 2013

[41] Öner, L., Alpar, R,. (1988). Regresyon Çözümlemesi. Fabad Farm. 13,556-565,1988

[42] Kılıç, S. (2013). Doğrusal regresyon analizi . Journal of Mood Disorders , 3 (2) , 90-92 . DOI: 10.5455/jmood.20130624120840

[43] Gallant (1975) Nonlinear Regression, The American Statistician, 29:2, 73-81, DOI: 10.1080/00031305.1975.10477374

[44] Baty, F., Ritz, C., Charles, S., Brutsche, M., Flandrois, J.-P., & Delignette-Muller, M.-L. (2015). A Toolbox for Nonlinear Regression in R: The Package nlstools. Journal of Statistical Software, 66(5), 1–21. https://doi.org/10.18637/jss.v066.i05

[45] Biau, Gérard & Scornet, Erwan. (2015). A Random Forest Guided Tour. TEST. 25. 10.1007/s11749-016-0481-7.

[46] Song, Yunsheng & Liang, Jiye & Lu, Jing & Zhao, Xingwang. (2017). An efficient instance selection algorithm for k nearest neighbor regression. Neurocomputing. 10.1016/j.neucom.2017.04.018.

[47] Chen, Yongbao & Xu, Peng & Chu, Yiyi & Li, Weilin & Wu, Yuntao & Ni, Lizhou & Bao, Yi & Wang, Kun, 2017. "Short-term electrical load forecasting using the Support Vector Regression (SVR) model to calculate the demand response baseline for office buildings," Applied Energy, Elsevier, vol. 195(C), pages 659-670.

[48] Ridgeway, G., Madigan, D. & Richardson, T.S.. (1999). Boosting Methodology for Regression Problems. Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics, in Proceedings of Machine Learning

[50] https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machinelearning-and-how-to-deal-with-it-6803a989c76

[51] https://www.veribilimiokulu.com

[52] https://analyticsindiamag.com/hands-on-tutorial-on-elasticnet-regression/

[53] https://www.veribilimiokulu.com

[54] https://www.researchgate.net/figure/Support-vector-regression-SVR-Illustration-of-an-SVR-regression-function-represented_fig12_248396465

[55] http://veribilimci.org/yapay-sinir-aglari-ysa-nedir-bolum-1/

[56] https://www.veribilimiokulu.com

[57] https://www.codingninjas.com/codestudio/library/classification-and-regression tree-algorithm

[58] https://levelup.gitconnected.com/random-forest-regression-209c0f354c84