T.C.
BALIKESİR ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
MATEMATİK ANABİLİM DALI

MAKİNE ÖĞRENMESİ ALGORİTMALARI VE UYGULAMALARI

HALİL SEZGİN CAN

YÜKSEK LİSANS TEZİ

Jüri Üyeleri : **Prof. Dr. Fırat ATEŞ** **(Tez Danışmanı)**
Prof. Dr. Sebahattin İKİKARDEŞ
Dr. Öğr. Üyesi Kadriye Filiz BALBAL

BALIKESİR, OCAK – 2026

ETİK BEYAN

Balıkesir Üniversitesi Fen Bilimleri Enstitüsü Tez Yazım İlkelerine uygun biçimde tarafımca hazırlanmış **"MAKİNE ÖĞRENMESİ ALGORİTMALARI VE UYGULAMALARI"** başlıklı bu tezde;

- Tüm bilgileri ve verileri akademik dürüstlük ilkeleri doğrultusunda elde ettiğimi,
- Kullanılan veriler ile elde edilen sonuçlarda herhangi bir değişiklik ya da müdahale yapmadığımı,
- Tüm bilgileri ve sonuçları bilimsel araştırma etiğine uygun olarak sunduğumu,
- Yararlanılan tüm kaynaklara uygun biçimde atıfta bulunup kaynakçada gösterdiğimi

beyan eder; aksi bir durumun ortaya çıkması hâlinde doğacak tüm yasal sorumlulukları kabul ettiğimi bildiririm.

Halil Sezgin CAN

ÖZET

MAKİNE ÖĞRENMESİ ALGORİTMALARI VE UYGULAMALARI

YÜKSEK LİSANS TEZİ
HALİL SEZGİN CAN
BALIKESİR ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ
MATEMATİK ANABİLİM DALI
(TEZ DANIŞMANI: PROF. DR. FIRAT ATEŞ)

BALIKESİR, OCAK - 2026

Bu tez çalışmasının 1. Bölüm giriş bölümü olup makine öğrenmesi ile ilgili çalışmaları ile ilgili bilgiler sunmaktadır. Tezimizin 2. bölümünde makine öğrenmesi çeşitleri ele alınarak denetimsiz öğrenme, denetimli öğrenme ve pekiştirmeli öğrenme kavramları teorik açıdan açıklanmıştır. Denetimsiz öğrenme alt başlıklarında ilişkilendirme, Apriori algoritması, kümeleme, hiyerarşik kümeleme, K-Means kümeleme ve K-En Yakın Komşuluğu yöntemleri detaylı şekilde tanıtılmıştır. Tezin üçüncü bölümde, farklı makine öğrenmesi algoritmaları tanıtılmış ve basit doğrusal regresyon, çoklu doğrusal regresyon, polinom regresyon, destek vektör makine regresyonu, karar ağacı regresyonu, rastgele orman regresyonu, Ridge, Lasso ve Elastic Net regresyon modelleri istatistiksel temel kavramlarla birlikte ayrıntılı olarak incelenmiştir. Dördüncü bölümde ise, doğrusal regresyon modeli yardımıyla belirli aralıklarda $f(x) = e^{-x^2}$, $f(x) = \sqrt{x^4 + 1}$, $f(x) = cosx$ fonksiyonlarının yaklaşık polinom karşılıkları hesaplanmış, bu süreçte trapez yöntemi referans alınarak model performansları değerlendirilmiştir. Son olarak, farklı makine öğrenmesi regresyon modelleri kullanılarak altın ons fiyatlarının tahmini gerçekleştirilmiş, modellerin tahmin doğrulukları karşılaştırılmış ve elde edilen bulgular yorumlanmıştır. Elde edilen sonuçlar, makine öğrenmesi algoritmalarının hem matematiksel hesaplamalarda hem de finansal öngörülerde etkin biçimde kullanılabileceğini göstermektedir. Bu tez, ilgili literatüre katkı sunarak gelecekteki çalışmalara yol göstermesi amacıyla hazırlanmıştır.

Anahtar Kelimeler: Altın ons fiyat tahmini, Makine öğrenmesi, Regresyon modelleri

Bilim Kod / Kodları: 20401 Sayfa Sayısı: 85

ABSTRACT

MACHINE LEARNING ALGORITHMS AND APPLICATIONS
MSC THESIS
HALIL SEZGIN CAN
BALIKESIR UNIVERSITY INSTITUTE OF SCIENCE
MATHEMATICS
(SUPERVISOR: PROF. DR. FIRAT ATES)

BALIKESİR, JANUARY - 2026

In this thesis, firstly, types of machine learning were addressed theoretically, unsupervised learning, including supervised learning, and reinforcement learning. In the context of unsupervised learning, methods such as association, Apriori algorithm, clustering, hierarchical clustering, K-Means clustering and K-Nearest Neighbors were introduced in detail. In the second stage, various machine learning algorithms were presented and examined comprehensively with their statistical fundamentals. These contain simple linear regression, multiple linear regression, polynomial regression, support vector machine regression, decision tree regression, random forest regression, Ridge, Lasso and Elastic Net regression models. In the third stage, approximate polynomial representations of the functions $f(x) = e^{-x^2}$, $f(x) = \sqrt{x^4 + 1}$, $f(x) = cosx$ within specific intervals were calculated using the linear regression model, and model performances were evaluated based on the trapezoidal method as a reference. Finally, the prediction of gold ounce prices was conducted using different machine learning regression models. The prediction accuracies of the models were compared and the results were interpreted. The findings demonstrate that machine learning algorithms can be effectively utilized in both mathematical computations and financial forecasting. This thesis aims to contribute to the related literature and to guide future research in this field. Finally, different machine learning regression models are used to predict gold ounce prices, the prediction accuracy of the models are compared and the findings are interpreted. The results obtained show that machine learning algorithms can be used effectively in both mathematical calculations and financial forecasting. This thesis is intended to contribute to the related literature and guide future studies.

Keywords: Gold ounce price forecasting, Machine learning, Regression models
Science Code / Codes : 20401 Pages Number : 85

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (DEVAM)

ŞEKİL LİSTESİ

Sayfa

TABLO LİSTESİ

Sayfa

ÖNSÖZ

Tez çalışmaları aşamasında karşılaştığım tüm durumlarda benimle yakından ilgilenen danışman hocam sayın Prof. Dr. Fırat ATEŞ'e, çalışmalarımda beni yakından takip eden sayın Prof. Dr. Eylem TOKSOY'a ve akademik olarak çeşitli yönlendirmeler ile bana destek olan sayın Prof. Dr. Sebahattin İKİKARDEŞ'e değerli ilgi ve destekleri için teşekkürlerimi sunarım.

Bu çalışmayı tamamlayabilmemde benden maddi manevi desteğini esirgemeyen ve emeği geçen eşim Tuğba'ya, biricik kızım Gaye'ye, babam Abdullah'a, annem Şükran'a ve kardeşim Elif 'e sonsuz teşekkürlerimi sunarım.

Balıkesir, 2026 **Halil Sezgin CAN**

1. GİRİŞ

Makine öğrenmesi, algoritmalar geliştirmek için kullanılan, yapay zekanın alt dalı olan ve sistemlerin performanslarını geliştirebilecekleri matematik ve istatistik bilimleri kullanılarak veri analizleri ile tahminlerde bulunan bilgisayar ile modellenmesidir. Makine öğrenmesi, insan zekasının belirli bir kısımlarını taklit etmeye yakın davranışlar sergiler. Alpaydın'a (2010) göre, makine öğrenmesi, elimizde var olan örnek verilerinden ya da geçmiş deneyimlerinden yararlanarak bir performans ölçütünü en iyi duruma getirecek şekilde programlanması sürecidir.

Makine öğrenmesinin en güzel tarafı, sistemlerin kendi içinde öğrenmelerine izin vermesidir. Girilen verileri yorumlar, çıktı alır ve sonuçlara göre sınıflandırma yapar. Finans, sağlık, bankacılık, turizm, tarım, ulaşım, perakende, müşteri hizmetleri akıllı kartlar, online arama, doğal dil işleme, öneri sistemleri, kişisel güvenlik veri güvenliği vb. gibi alanlar makine öğrenmesinden faydalanmaktadır. Makine öğrenmesi üç önemli durum üzerinde yoğunlaşır. Bunlar; veri, model ve öğrenmedir. Veri, makine öğrenmesi için en önemli hususların başında gelmektedir. Elde edilen verilerin doğru işlenmesi, makine öğrenmesinin tahmininde çok önemli rol oynamaktadır. Verilerin anlamlı olması modellerin eğitmesine daha fazla yardımcı olacaktır. Makine öğrenmesinde üç öğrenme tekniği vardır. Bu teknikler denetimli, denetimsiz ve pekiştirmeli öğrenmedir. Bu öğrenme türlerinin açıklanması tezin devamında detaylı bir şekilde yer verilecektir. Makine öğrenmesinde matematik bilimi çok önemli bir yere sahiptir. Verilerin işlenmesi sürecinden modellerin oluşturulmasına kadar uzanan süreçte matematik sıklıkla kullanılır. Matematik biliminin alt dalları olan Kalkülüs, Olasılık, Analitik Geometri, Cebir, Vektör, Matris, Optimizasyon gibi alanlar makine öğrenmesinin temelini oluşturmaktadır.

Makine öğrenmesi, yapay zekanın alt dalı olduğunu belirtmiştik. Bu bölümde yapay zeka ve makine öğrenmesinin tarihi gelişimini inceleyeceğiz. Yapay zeka kavramı, teknolojik bir terim olmasının ötesinde, popüler kültür ve özellikle bilim kurgu sineması tarafından şekillendirilmiş zengin bir imgesele sahiptir. Bu bağlamda, yapay zeka denildiğinde kolektif hafızada yer eden temsiller arasında; 2001: Bir Uzay Macerası filmindeki HAL 9000, Yıldız Savaşları (Star Wars) evreninden C-3PO, Uzay Yolu (Star Trek) dizisindeki android Data ve Türkiye'de Aşk adıyla gösterime giren Her filmindeki Samantha gibi karakterler yer almaktadır. Bu karakterler, yapay zekanın toplumsal algısını ve geleceğine yönelik

beklentileri büyük ölçüde etkilemiştir. Yapay zeka ve makine öğrenimi sık sık birbirlerinin yerine kullanılabiliyor ancak genel anlamda bakacak olursak makine öğrenimi yapay zekadan evrildi diyebiliriz. Google Trend grafiklerine baktığımızda, Yapay Zeka'nın (AI) Eylül 2015 dilimlerinde makine öğrenimi sözcüğü yayılana kadarki sürece kadar popüler bir arama olduğunu görebiliriz. Makine öğrenimi yapay zekanın tek olmasa uygulamalarından biridir. Makine öğrenmesi yapay zekanın bir alt dalı ise, derin öğrenme de makine öğrenmesinin bir alt dalıdır. Bu durumu yapay zekâ > makine öğrenimi > derin öğrenme bu şekilde şematize edebiliriz. Derin öğrenme modeli ilk olarak 2000'lerde Igor Aizenburg ve aynı alanda araştırma yapan arkadaşları tarafından Yapay Sinir Ağları (ANNs) kullanıldı. Özetleyecek olursak, derin öğrenme, makine öğrenimini kullanmak için bir yöntemdir (Çelti, C. 2022).

Şimdi yapay zekanın tarihsel sürecine dair açıklamalar yapalım. Bu kısımda açıklamar ile ilgili detaylı ele alınan içeriklere "https://www-lightsondata-com.translate.goog/the-history of%20machinelearning/?_x_tr_sl=en&_x_tr_tl=tr&_x_tr_hl=tr&_x_tr_pto=tc" adresinden ulaşılabilir. Yapay sinir ağının bilinen en eski matematiksel modeli 1943'te Walter Pitts ve Warren McCulloch tarafından çalışıldı. Çalıştıkları alan o zamana göre çok etkileyeci olan sinirsel aktivitelerdir. Günümüzde halen standart olarak kabul edilen bu modelleme McCulloch-Pitts nöronları olarak bilinir. 1950 yıllarına geldiğimzde II. Dünya Savaşı'ndaki şifre kırıcılığıyla bilinen İngiliz matematikçi Alan Turing karşımıza çıkıyor. 1950 yıllardaki "Makinelerin İşleyişi ve Zekâ" adlı makalesi en bilinenlerinden. Turing Testi ile bir bilgisayarın düşünüp düşünemeyeceğini test edebilen test tasarlıyor. 1952 yılında Arthur Samuel makine öğrenimi tabirini ilk ortaya atan kişi olmakla beraber makine öğreniminin öncüsü olarak bilinir. IBM'deki Poughkeepsie Laboratuvarına katılarak Samuel öncelikli bilgisayar programını tasarları. Bu programla dama oyununu daha iyi öğrenebilmesi adına en iyi yolları öğrenmeyi içeriyordu. 1957 yılında derin sinir ağlarının temelleri Frank Rosenblatt tarafından atıldı. "Perceptron: Algılayan ve Tanıyan bir Otomaton isimli yayınında, optik bir beynin algısal süreçlerine yakın olacak bir şekilde, bilgi kalıpları arasındaki benzerlikleri veya kimlikleri tanımayı öğrenecek elektronik veya elektromekanik bir sistem oluşturacağını belirtiyordu. 1959 yıllında nörofizyolojist ve Nobel ödüllü David H. Hubel ve Torsten Wiesel birincil görme korteksinde iki tip hücre meydana geldiğini keşfetti. Bu buluşunda biyolojik gözlemler ve yapay sinir ağlarından yararlanmıştır. 1960 yılında Henry J. Kelley, "Optimal Uçuş Rotaları için İrtifa Kuramı" yazısını yayımladı. Bu yazısında sinir ağları ve yapay zeka kavramlarından sıkça bahsedilmiştir. 1965 yılında

Matematikçi Ivakhnenko ve Lapa'nın da entegre olduğu meslektaşları, çalışan derin öğrenme ağları üzerinde çalıştılar. Ivakhnenko, modellerin tam otomatik yapısal ve parametrik optimizasyonunu içeren bilgisayar tabanlı matematiksel modelleri için tümevarım algoritmaları ailesi geliştirdi ve bunu sinir ağlarına uyguladı. Ivakhnenko 1971'de GMDH'yi test ederek 8 katmanlı bir derin ağ oluşturabilmiş ve Alpha diye bilinen bilgisayar tanımlama sistemindeki öğrenme sürecini başarılı bir şekilde gösterebilmişti. 1979-80 yıllarında yapay sinir ağlarının görsel görüntüleri nasıl tanıyacağı öğretiliyor bilgisayarlar. Bunun öncüsü ise Kunihiko Fukushima el yazısı karakteri ve örüntü tanıma görevlerinde kullandı. 1982 yılında Hoofield, kendi ismini taşıyan bir sistem tanıttı ve Hopfield Ağları, bağlantı kurulabilir bellek sistemleri olarak faliyet gösteren tekrarlayan sinir ağları şeklinde günümüze kadar ulaştı. 1985 yılında Terry Sejnowski, NETtalk'u oluşturdu. Program İngilizce kelimeleri neredeyse bir çocuüun öğrenme biçimine benzer şekilde edinmiş ve metinden konuşmaya dönüştürme süreci boyunca kendi performansını sürekli geliştirmiştir. 1986 yılında bu alanda şekili tanıma ve kelime tahminlerinde ilerlemeler kaydedildi. 1989 yıllarında el yazması rakamları okuyan yani o yazıları tahmin eden makineler tasarlandı. 1989 yılında Christopher Watkins, Q öğrenme üzerine çalışmalar yaptı. 1993 yılında bir çok derin öğrenme ismi altında derin öğrenmenin katmanlar üzerinde çalışmaları hızlandı. 1995 yılında Destek Vektör Makineleri ki halen günümüzde algoritmaları kullanılıyor bu yıllarda tasarlandı. 1997 yılında tekrar eden sinir ağı yapısı olan uzun kısa-süreli hafıza öne sürüldü. 1998 yılında Gradyan tabanlı öğretim kullanıldı. 2009 yılında Fei-Fei Li, 2009'da ImageNet'i önderlik etti. 2019 yılı itibariyle, 14 milyondan fazla etiketli görselin eğitimciler, araştırmacıların ve öğrencilerin ulaşımına açık olduğu çok fazla ve ücreti olmayaan bir veri tabanı. 2011 ve 2012 yılları arasında Alex Krixhevsky, AlexNet ile birçok yarış kazandı. 2012 yılında Kedi Deneyi denendi. 2014 yılında DeepFace ve Üretken Çekişmeli Ağlar kullanıldı. 2016 yılında şirketler kuvvetli makine öğrenimi ve derin öğrenme türleri ve çözümleri sunabildi. Bunlara ek olarak eğlence ve oyun alanında şunlar yapıldı.

Yapay zekanın gelişim tarihi, özellikle satranç ve Go gibi insan zekasının zirvesi kabul edilen strateji oyunlarında elde ettiği başarılarla şekillenmiştir. Bu alandaki ilk atılımlardan biri, 1992'de kendi kendine tavla oynamayo öğrenen TD – Gammon programı olsa da, asıl sembolik zafer 1997'de IBM Deep Blue'nun satranç ustası Garry Kasparov'a karşı galip gelmesiyle kazanılmıştır. Bu başarıyı yaklaşık yirmi yıl sonra, çok daha karmaşık bir oyun olan Go'da elde edilen zaferler izlemiştir. Google DeepMind'in AlphaGo programı, makine

öğrenimi ve ağaç araması yöntemlerini kullanarak önce 2016'da Lee Sedol'ü, ardından 2017'de dünya şampiyonu Ke Jie'yi mağlup ederek makine zekasının ulaştığı sofistike düzeyi kanıtlamıştır. Strateji oyunlarının yanı sıra, yapay zeka doğal dili anlama ve yorumlama konusunda da önemli bir sınav vermiştir. 2011 yılında IBM Watson, popüler yarışma programı Jeopardy'de makine öğrenimi, doğal dil işleme ve bilgi çıkarma yeteneklerini birleştirerek insan rakiplerini yenmiş ve yapay zekanın farklı bir alandaki potansiyelini ortaya koymuştur.

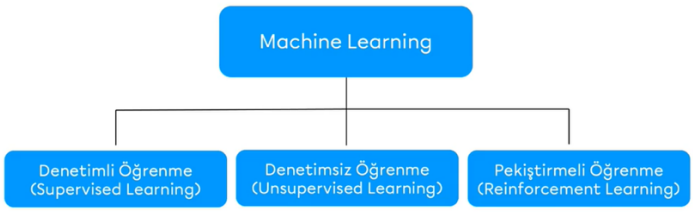
Yapay Zeka, Makine öğrenmesi ve Derin Öğrenme öğelerinin tarihçesini

- 1960 - Yüzeysel sinir ağları
- 1970 - Geri yayılımın ortaya çıkışı
- 1974 - İlk AI Kışı
- 1980 - Evrişimin ortaya çıkışı
- 1987 - İkinci AI Kışı
- 1990 - Gözetimsiz derin öğrenme
- 1990 - Gözetimli derin öğrenme yeniden moda
- 2006 - Modern derin öğrenme

şeklinde kısa bir şekilde toparlayabiliriz.

2. MAKİNE ÖĞRENMESİ ÇEŞİTLERİ

Bu bölümde Makine Öğrenmesinin alt dallarından olan Denetimli Öğrenme, Denetimsiz Öğrenme ve Pekiştirmeli Öğrenme konularına yönelik tanımlamalar yapacağız.



Şekil 2.1: Makine Öğrenmesi Çeşitleri

2.1 Denetimli Öğrenme

Denetimli öğrenme, veriler üzerinde etiket kullanarak modeller geliştiren öğrenme türüdür. Denetimli öğrenme yaklaşımında kullanılan etiketli veri setleri, her bir girdi örneğinin, karşılığı olan doğru çıktı ile birlikte sunulduğu veri kümeleridir. Bu sayede algoritma, girdiler ile çıktılar arasındaki temel örüntüyü modelleyerek bir tahmin fonksiyonu geliştirmeyi öğrenir. Bu öğrenme verilerin manuel olarak etiketlenmesi anlamına gelir. Bu etiketlenmede amaç girdilerin çıktılar ile eşleştirilmesidir. Denetimli öğrenmeyi somutlaştırmak adına bir sınıftaki öğretmen-öğrenci ilişkisinden bahsedebiliriz. Burada öğretmenler için her bir öğrenci aslında birer etiket gibi düşünülebilir. Bu modelde bir görev ve çok sayıda manuel veri içerir. Bu verileri eğitirken 3 kategoride değerlendirir. Bunlar; eğitim verileri, test verileri ve doğrulama verileridir (Rani, V., Nabi, S. T., Kumar, M., Mittal, A., and Kumar, K. 2023).

Denetimli öğrenme, makine öğrenmesi üzerinde en öne çıkan modellerden birisidir. Bunu önemli kılan kritik özelliği ise girdiler ve çıktılar arasındaki kurduğu ilişkilerden yararlanmasıdır. Verilerin kategorize edilmesi bu modelin verileri sınıflandırmasına çok yardımcı olur. Algoritmanın esas amacı girdiler ve çıktılar arasında ilişki kurup sonuçları tahmin etmesidir. Elde edilen verilerin fazlalığı makine öğreniminin tahmindeki hassasiyeti ile doğru orantılıdır. Denetimli öğrenme, veri gurubundaki durumları etiketleyip yeni gelecek olan durumları bu etiketlere göre genelleme yapabiliyor. Bu etiketler veriler üzerinde eğitilir sonrasında girdi ve çıktı arasındaki ilişkiye bağlı olarak algoritma tahmin yapar. Makine öğrenmesi ile bir tahmin modelinin nasıl çalıştığı, bir teknik makalenin

okunma sayısını tahmin etme problemi üzerinden örneklendirilebilir. Bu senaryoda, makalenin kelime sayısı, yayınlanma zamanı veya yazarın takipçi sayısı gibi özellikler bağımsız değişken (girdi) olarak kullanılırken, hedeflenen bağımlı değişken (çıktı) ise makalenin alacağı okuyucu sayısıdır. Bir makine öğrenmesi algoritması, bu girdi ve çıktı verilerinden oluşan bir eğitim setini analiz ederek aralarındaki temel örüntüyü ve fonksiyonel ilişkiyi öğrenir. Bu öğrenme süreci tamamlandığında, model, gelecekteki bir makalenin özelliklerine (girdilerine) bakarak okunma sayısını başarılı bir şekilde öngörebilir.

Denetimli öğrenme, etketli veri kümelerinden yararlanarak girdi ve çıktı değişkenleri arasında bir eşleme fonksiyonu oluşturma sürecidir. Bu süreçte tahmin edilecek çıktının yapısı, problemin regresyon ve sınıflandırma olarak kategorize edilmesini belirler. Çıktı değişkeninin nicel ve sürekli olduğunu (örneğin, bir konutun piyasa değeri, hava durumu sıcaklığı veya bir öğrencinin sınav notu gibi) durumlarda problem, bir regresyon analizi olarak ele alınır. Buna karşılık, çıktı değişkeni erkek/kadın veya doğru/yanlış gibi nitel ve ayrık kategorilerden oluşuyorsa, bu bir sınıflanma sürecidir.

Regresyonun temel amacı, bağımsız değişkenlerdeki değişimlerin bağımlı değişkeni nasıl etkilediğini modelleyerek geleceğe yönelik sayısal öngörülerde bulunmaktır. Değişkenler arasındaki istatiksel ilişkiyi modellemek üzerine geliştirilmiş yaygın regresyon algortimalarından bazıları şu şekildedir:

- Basit Doğrusal Regresyon (Simple Linear Regression)
- Çoklu Doğrusal Regresyon (Multiple Linear Regression)
- Polinom Regresyonu (Polynomial Regression)
- Lojistik Regresyon (Logistic Regression)
- Destek Vektör Regresyonu (Support Vector Regression)
- Karar” Ağacı Regresyonu (Decision Tree Regression)
- Rastgele Orman Regresyonu (Random Forest Regression)

Şimdi regresyon modeli sınıflandırma kısmından bahsedelim. Sınıflandırma veri kümesini farklı değişkenlere dayalı olarak sınıflara ayırmayı amaçlayan bir algoritmadır. Girdi verilerini ayrık değerdeki çıktı verisine ilişkileyen problemlere uygulanmaktadır. Örneğin göğüs kanseri tahmini, bir resmin kedi içerip içermediği, cinsiyet tahmini vb. Yani

bir şeyleri kategorize etmek istiyorsak kullanabiliriz. Sınıflandırma algoritmalarından bazıları şu şekilde sıralanabilir :

- Lojistik Regresyon (Logistic Regression)
- K-En Yakın Komşuluğu (K-Nearest Neighbours)
- Destek Vektör Makineleri (Support Vector Machines)
- Çekirdek Destek Vektör Makineleri (Kernel SVM)
- Naive Bayes
- Karar Ağacı Sınıflandırması (Decision Tree Classification)
- Rastgele Orman Sınıflandırması (Random Forest Classification)

Denetimli öğrenmenin iki ana kategorisi olan regresyon ve sınıflandırma arasındaki fark, pratik iş dünyası seneryalarını üzerinden somutlaştırılabilir. Bu bağlamda, bir şirketin karşılaşabileceği iki farklı problem ele alınabilir. İlk senerya, şirketin büyük bir envanterindeki ürünlerinin önümüzdeki üç aylık periyotta satış hacmini öngörme ihtiyacıdır. Bu durum, bir regresyon durumudur. Zira buradaki hedeflenen çıktı, satılacak ürün adedi gibi sayısal ve sürekli bir değerdir. Model, geçmiş verilere dayanarakbelirli bir aralıkta herhangi bir sayısal değeri tahmin etmeye çalışır. İkinci senaryo ise, müşteri hesaplarını analiz ederek bir hesabın güvenlik ihlaline uğrayıp uğramadığını tespit etme görevidir. Bu durum ise bir sınıflandırma durumudur. Bu senaryoda modelden beklenen çıktı, güvenlik ihlali var veya güvenlik ihlali yok gibi önceden tanımlanmış iki ayrık kategorilerden biridir. modelin görevi, her bir hesabı bu sınıflardan birine doğru şekilde atamaktır.

2.2 Denetimsiz Öğrenme

Denetimsiz öğrenme etiketlenmemiş veriler üzerinde ilişkiyi algoritmaya öğretip bunu ortaya çıkarmasını sağlayan öğrenme çeşididir. Bu öğrenme için ek açıklamalar, etiket gibi durumlar gerekli değildir. Denetimsiz öğrenmede en önemli kullanılan yol kümelemedir. Sisteme girilen verileri önceden sınıflandırmaz veya etiketlemez. Bunların dışında verileri kendisi analiz ederek keşif yapar. Denetimsiz öğrenme, denetimli öğrenme aksine benzerlikleri, gruplamaları ve bağlantıları bularak verilerin iç yapılarını ortaya çıkarır. Verilerin anlam kazanması algoritmanın temel aldığı nokta korelasyon, benzerlik ve bağımlılıktır. Denetimsiz öğrenme, segmentasyon, veri keşfi ve anomali tespiti (örneğin, sahtekarlık tespiti, müşteri segmentasyonu) gibi alanlarda kullanılır (Çelti, C. 2022).
Denetimsiz Öğrenme Algoritmaları aşağıdaki gibidir.

- K-Means Kümeleme
- Hiyerarşik Kümeleme
- DBSCAN (Yoğunluk Tabanlı Kümeleme)
- PCA (Temel Bileşen Analizi)
- t-SNE (t-dağıtılmış stokastik komşu yerleştirme)
- Autoencoder (Oto Kodlayıcı)

Denetimsiz öğrenme, etiketlenmiş veri setleri içerisindeki gizli örüntüleri ve içsel yapıları keşfetmeyi amaçlayan bir makine öğrenmesi durumudur. Bu durum altında, veriyi anlamlandırmak için kullanılan iki temel bakış açısı öne çıkar. Bunlar kümeleme ve ilişkilendirmedir. Kümeleme, veri noktaları arasındaki benzerliklere göre doğal gruplara ayırma işlemidir. İlişkilendirme ise, büyük veri setlerindeki öğeler arasında sıkça birlikte meydana gelen ilişkileri veya kuralları ortaya çıkarmaya odaklanır. Bu iki temel yaklaşım, denetimsiz öğrenmenin temelini oluşturur (Rani, V., Nabi, S. T., Kumar, M., Mittal, A., and Kumar, K. 2023).

Bu kısımda yer alan detaylı bilgilere (https://medium.com/machine-learning-türkiye/adım-adım-makine-öğrenmesi-bölüm-3-denetimsiz-öğrenme-nedir-f890ada49a40) kaynağından ulaşılabilir.

2.2.1 İlişkilendirme

İlişkilendirme analizi, genellikle Birliktelik Kuralı Madenciliği (Association Rule Mining) olarak bilinen bir teknik olup, büyük veri setleri içerisindeki değişkenler arasında var olan örtük ilişkileri ve kalıpları keşfetmeyi hedefler. Bu prosedür, özellikle işlemsel (transactional) veritabanları gibi büyük veri ambarlarında sıkça birlikte ortaya çıkan öğe kümelerini (frequent itemsets) ve bu birliktelikleri tanımlayan kuralları ortaya çıkarır.

Bu tekniğin en bilinen ve en yaygın uygulama alanı Pazar Sepeti Analizi (Market Basket Analysis)'dir. Bu analiz sayesinde, perakendecilik ve e-ticaret gibi sektörlerde hangi ürünlerin genellikle birlikte satın alındığı gibi değerli bilgiler elde edilir. Elde edilen bulgular, mağaza içi ürün yerleşimi, çapraz satış (cross-selling) stratejileri ve kullanıcılara yönelik kişiselleştirilmiş ürün tavsiye sistemleri gibi ticari ve stratejik kararların alınmasında kullanılır.

2.2.2 Apriori Algoritması

Birliktelik kuralı madenciliğinde kullanılan en temel ve öncü yaklaşımlardan biri Apriori algoritmasıdır. Bu algoritma, veri setinde sık tekrar eden öğe kümelerini (frequent itemsets) aşamalı olarak belirleme prensibine dayanır.

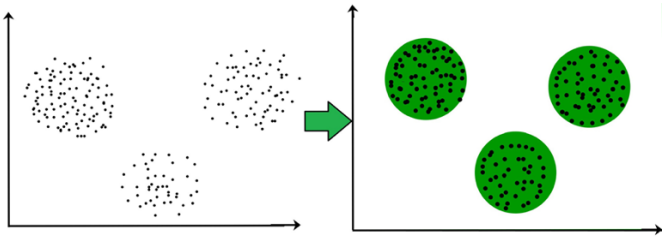
Algoritmanın adı, Latince "önsel" anlamına gelen ve temelini oluşturan Apriori Prensibi'nden gelir. Bu prensibe göre, bir öğe kümesi eğer sık tekrar ediyorsa, o kümenin tüm alt kümeleri de sık tekrar etmek zorundadır. Bu "önsel bilgi", algoritmanın daha nadir görülen kombinasyonları verimsiz bir şekilde aramasını engelleyerek hesaplama verimliliği sağlar.

Pratikte bu algoritma, temel olarak "hangi ürünlerin birlikte satın alındığı" sorusuna yanıt bulur. Örneğin, bir e-ticaret platformunda yapılan analiz sonucunda "filtre kahve alan müşterilerin büyük bir kısmının aynı zamanda kahve kupası da satın aldığı" kuralı keşfedilebilir. Bu bilgi, filtre kahveyi sepetine ekleyen bir müşteriye otomatik olarak kahve kupası önerilmesini sağlayarak çapraz satış (cross-selling) fırsatları yaratır ve satış hacmini artırmayı hedefler.

2.2.3 Kümeleme

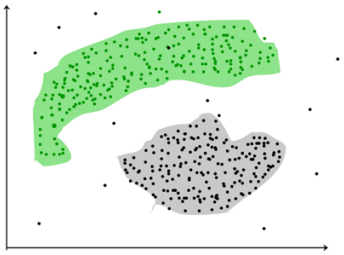
Kümeleme (Clustering), denetimsiz öğrenmenin temel yöntemlerinden biri olup, etiketlenmemiş bir veri setindeki doğal gruplanmaları ortaya çıkarmayı amaçlar. Bu yöntemin temel amacı, veri noktalarını, grup içi benzerliği (intra-cluster similarity) maksimize edecek ve gruplar arası benzerliği (inter-cluster similarity) minimize edecek şekilde kümelere ayırmaktır. Diğer bir deyişle, aynı kümedeki veri noktalarının birbirine çok benzer, farklı kümelerdeki veri noktalarının ise birbirinden oldukça farklı olması hedeflenir (Çelti, C. 2022).

Pek çok kümeleme algoritmasında, oluşturulacak küme sayısı (genellikle "k" hiperparametresi olarak anılır) kullanıcı tarafından önceden belirlenebilir. Bu parametre, verinin hangi ayrıntı düzeyinde veya granüleritede gruplanacağını kontrol etme imkânı sunar.



Şekil 2.2: Kümelenmiş veri noktaları

Şekil 2.2 'te yer alan grafikte yapılan kümeleme analizi sonucunda, veri setinin üç farklı gruba ayrıldığı tespit edilmiştir. Kümeleme algoritmalarının çıktısı sıklıkla dairesel veya küresel (spherical) formda tasavvur edilse de, bu bir genelleme olup zorunluluk teşkil etmez. Özellikle K-Ortalamalar (K-Means) gibi merkez tabanlı (centroid-based) algoritmalar bu tür geometrik şekillere eğilimliyken, yoğunluk tabanlı (density-based) algoritmalar gibi farklı yaklaşımlar, düzensiz veya iç içe geçmiş formlarda kümeleri de ortaya çıkarabilir. Nitekim, gerçek dünya verileri nadiren bu tür idealize edilmiş ve net bir şekilde ayrılmış küresel yapılar sergiler; bu nedenle farklı küme geometrilerini tanıyabilen algoritmaların seçimi, analizin başarısı için kritik önem taşır.



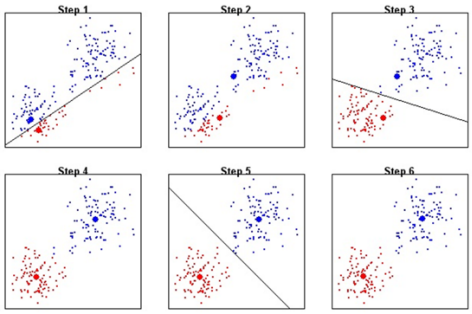
Şekil 2.3:Kümeleme

2.2.4 K-Means Kümeleme

K-Ortalamalar (K-Means), en yaygın kullanılan bölümlemeli (partitioning) kümeleme algoritmalarından biridir ve yinelemeli bir yaklaşımla çalışır. Algoritmanın adındaki "K" terimi, veri setinin ayrıştırılacağı küme sayısını ifade eden ve kullanıcı tarafından önceden belirlenmesi gereken bir hiperparametredir. Algoritma, K adet küme merkezinin (sentroid) başlangıçta rastgele atanmasıyla başlar. Ardından, iki adım yinelemeli olarak tekrar edilir:

1. Atama Adımı: Her bir veri noktası, kendisine en yakın küme merkezine (genellikle Öklid mesafesi kullanılarak) atanır.
2. Güncelleme Adımı: Her bir küme merkezi, o kümeye atanan tüm veri noktalarının ortalaması alınarak konumu yeniden hesaplanır.

Bu atama ve güncelleme döngüsü, küme merkezlerinin konumları sabitlenene veya veri noktalarının küme üyelikleri daha fazla değişmeyene kadar devam eder. Bu duruma algoritmanın yakınsaması (convergence) denir ve bu noktada en optimize edilmiş küme yapısına ulaşıldığı kabul edilir.



Şekil 2.4: K-Means Kümeleme

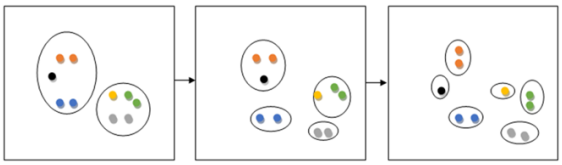
Şekil 2.4 üzerinden örnekle anlatmaya çalışalım. K-Ortalamalar (K-Means) algoritmasının çalışma süreci, küme sayısının k=2 olarak belirlendiği bir örnek üzerinden incelenebilir.

Algoritmanın ilk adımında (Step 1), Şekil X'te mavi ve siyah renklerle gösterilen iki adet başlangıç küme merkezi (sentroid) rastgele veya belirli bir yöntemle seçilir. Ardından, her bir veri noktası, kendisine en yakın olan sentroide atanarak ilk kümeleme gerçekleştirilir. Bu atama işleminin tamamlanmasının ardından, her iki küme merkezinin konumu, kendilerine atanan veri noktalarının yeni ortalaması olarak güncellenir; bu durum sentroidlerin yerlerinin değişmesiyle sonuçlanır. Bu atama ve güncelleme döngüsü, küme üyeliklerinde herhangi bir değişiklik olmayana kadar, yani model yakınsayana (converge) kadar tekrarlanır.

2.2.5 Hiyerarşik Kümeleme

Hiyerarşik kümeleme, veri noktalarını tek bir küme bölümlemesi yerine, iç içe geçmiş bir kümeler hiyerarşisi oluşturarak gruplandıran bir denetimsiz öğrenme yöntemidir. Bu yöntemin iki temel yaklaşımı bulunmaktadır. Birleştirici yaklaşım en yaygın kullanılan bu

yaklaşım, her veri noktasını başlangıçta kendi kümesi olarak kabul eder ve her adımda birbirine en yakın iki kümeyi aşamalı olarak birleştirir. Tüm veri noktaları tek bir kümede birikene kadar devam eder. Diğer yaklaşım ise Bölücü yaklaşımdır. Bu yaklaşım ise tüm veri setini tek bir küme olarak başlatır ve her adımda en heterojen kümeyi aşamalı olarak ikiye böler. Hiyerarşik kümelemenin çıktısı, bu birleşme veya bölünme sürecini görselleştiren ve dendrogram adı verilen ağaç yapılı bir diyagramdır. Dendrogram, farklı benzerlik seviyelerindeki küme yapılarını inceleme olanağı tanır ve istenilen bir seviyeden "kesilerek" belirli sayıda küme elde edilebilir. Her küme farklıdır ve veriler büyük ölçüde birbirini andırır (Şekil 2.5).

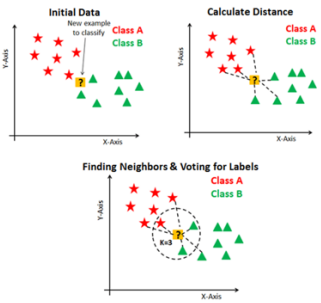


Şekil 2.5: Hiyerarşik Kümeleme

2.2.6 K-En Yakın Komşuluğu (K-NN)

K-En Yakın Komşu (K-NN), denetimli öğrenme problemlerinde, özellikle sınıflandırma görevlerinde kullanılan, parametrik olmayan (non-parametric) ve örneklem tabanlı (instance-based) temel algoritmalardan biridir. Algoritmanın çalışma prensibi, sınıflandırılacak yeni bir veri noktasının etrafındaki "K" adet en yakın komşusunun sınıf etiketlerine dayanır. Sınıflandırma süreci şu adımları izler. İlk olarak yeni veri noktası ile eğitim setindeki tüm veri noktaları arasındaki uzaklık (genellikle Öklid mesafesi) hesaplanır. Sonrasında uzaklık metriklerine göre en yakın "K" adet komşu belirlenir. Son olarak yeni veri noktası, bu "K" komşu arasında en sık tekrar eden sınıfa atanır. Bu işleme çoğunluk oylaması (majority voting) denir.

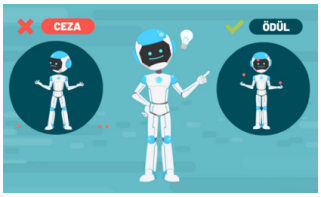
"K" değeri, modelin performansını doğrudan etkileyen ve kullanıcı tarafından belirlenen bir hiperparametredir. K-NN, "tembel öğrenme" (lazy learning) algoritması olarak da bilinir; çünkü modelin ezberlemesi gereken bir fonksiyon yoktur ve belirgin bir eğitim aşaması bulunmaz. Bununla birlikte, her bir yeni tahmin için eğitim setindeki tüm noktalara olan uzaklığı hesaplaması gerektiğinden, büyük veri setlerinde tahmin (prediction) süresi oldukça maliyetli olabilir.



Şekil 2.6: K-En Yakın Komşuluğu

2.3 Pekiştirmeli Öğrenme

Pekiştirmeli Öğrenme (Reinforcement Learning), bir ajanın (agent) bir çevre (environment) içerisinde deneme-yanılma yoluyla en uygun davranış biçimini öğrendiği bir makine öğrenmesi paradigmasıdır (Kumar, S. and Bhatnagar, V., 2022). Bu süreçte ajan, belirli bir durumdayken (state) bir eylemde (action) bulunur. Bu eylemin sonucunda çevreden sayısal bir ödül (reward) veya ceza (penalty) sinyali alır ve yeni bir duruma geçer. Ajanın temel amacı, tek bir hamlenin anlık getirisini değil, uzun vadede toplam ödülünü maksimize edecek bir politika (policy), yani bir eylem stratejisi geliştirmektir. Bu yaklaşıma klasik bir örnek olarak, dama gibi strateji oyunlarını oynamayı öğrenen bir program verilebilir. Bu senaryoda program (ajan), oyun tahtası (çevre) üzerinde hamleler (eylemler) yapar. Oyunun kazanılması pozitif bir ödülle sonuçlanırken, kaybedilmesi negatif bir ödül (ceza) anlamına gelir. Program, binlerce oyun oynayarak hangi hamle dizisinin zafere götürdüğünü, yani en yüksek kümülatif ödülü sağlayan politikayı öğrenir ve zamanla uzmanlaşır (Çelti, C. 2022).



Şekil 2.7: Pekiştirmeli Öğrenme

3. MAKİNE ÖĞRENMESİ ALGORİTMALARI

3.1 Giriş

Giriş kısmında belirttiğimiz üzere makine öğrenmesi 2 alt basamakta inceleniyor. Bunlar denetimli öğrenme ve denetimsiz öğrenme olarak. Denetimli öğrenme kendi içinde regresyon ve sınıflandırma olarak ikiye ayrılıyor. Denetimsiz öğrenme ise alt dalında kümeleme yöntemi yer almaktadır. Tezimizin bu bölümünde makine öğrenmesinin alt dalından birisi olan gözetimli öğrenme modelleri inceleyeceğiz. Bunları sırasıyla Basit Doğrusal Regresyon (Simple Linear Regression), Çoklu Doğrusal Regresyon (Multiple Linear Regression), Polinom Regresyon (Polinomial Regression), Destek Vektör Makine Regresyon (Support Vector Machine Regression), Karar Ağacı Regresyon (Decision Tree Regression), Rastgele Orman Regresyon (Random Forest Regression), Ridge Regresyon ve Lasso Regresyon modelleri hakkında detaylı bilgiler verip bu modellere uygun örnekler üzerinden uygulamalar yapacağız. Bu bölümde yer alan bilgilere, (Lichtenberg, J. M. and Şimşek, Ö. (2016), Mahaboob, B., Praveen, J. P., Appa Rao, B. V., Harnath, Y., Narayana, C., and Prakash, G. B. 2020, Arslankaya, S. ve Toprak, Ş. , 2021) kaynaklarından ulaşılabilir.

3.2 Basit Doğrusal Regresyon (Simple Linear Regression)

Bir veri grubundaki değerlerin önce noktaya dönüştürülüp *x*-*y* koordinat sistemi üzerine yerleştirildikten sonra o noktalara en yakın geçebilecek fonksiyonların oluşturmasına Basit Doğrusal Regresyon (Simple Linear Regression) denir. Bu regresyon modelinde en çok kullanılan yöntem küçük kareler yöntemidir. (x_i, y_i) noktaları için $i = 1,2, \dots, n$ ele alalım. Bu kısımda her bir y_i lerin x_i lere bağlı olduğunu x_i değerlerine karşılık y_i lerin değiştiğini varsayalım. Her $i = 1,2, \dots, n$ için $y_i = f(x_i)$ olacak şekilde bir fonksiyon tanımlayıp, hata payını ele aldığımızda bazı eşitliklerin sağlanmadığını kabul edelim. Bu kısımdaki amacımız bizi en az hatayla en doğru fonksiyona ulaştırmak olacaktır. Basit Doğrusal Regresyon (Simple Linear Regression) modelinde $f(x) = ax + b$ fonksiyonu kullanılır ve burada bulunması gerekenler a ve b parametreleridir (Lichtenberg, J. M. and Şimşek, Ö. 2016).

3.3 Çoklu Doğrusal Regresyon (Multiple Linear Regression)

Bir önceki bölümde bağımsız değişken sayısının tek olduğu basit doğrusal regresyon yöntemini inceledik. Bu bölümde ise bağımsız değişken sayısı birden fazla olacak şekilde çoklu doğrusal regresyon modelini inceleyeceğiz. Bu modeli incelemeden önce ihtiyacımız olan bazı temel istatistiki kavramlara değinelim.

3.3.1 Beklenen Değer

İstatistiksel bir deneyin bir sonraki tekrarlanan denemesinin sonucu olma olasılığı en yüksek olan değere beklenen değer denir. Bir deneyin rastgele denemesinde beklenen sonucu belirlemek için, tüm olası sonuçların olasılığı beklenen değer hesaplamalarına dahil edilir. Beklenen değer, veri kümesindeki verilerin ağırlıklı ortalamasını bulmak için tüm olası sonuçları ve bunların gerçekleşme olasılıklarını kullanır. X kesikli rassal değişken, $P(x)$ ise olasılık dağılımının x teki değeri olmak üzere X in beklenen değeri

$$E(X) = \sum_{x} xP(x) \tag{3.1}$$

denklemi ile gösterilir. Ayrıca X rassal değişkeni sürekli olduğunda ise X in beklenen değeri

$$E(X) = \int_{-\infty}^{\infty} xP(x) \tag{3.2}$$

denklemi ile ifade edilir.

3.3.2 Varyans

Bir veri setindeki tüm verilerin, veri setinin ortalamasına olan uzaklıklarının ortalamasına varyans denir. Standart sapma σ olmak üzere, varyans ifadesinin gösterimi σ^2 şeklindedir. Varyans, bir veri setindeki her bir noktanın, o setin aritmetik ortalamasından ne kadar saptığını gösteren temel bir dağılım ölçüsüdür. Bu değer, her bir veri noktasının ortalamadan farkının karesi alınıp bu karelerin toplanması ve bulunan toplamın, veri setindeki eleman sayısına bölünmesiyle hesaplanır. Ayrıca veri kümesinde bulunan gözlemlerin tamamına ana kütle denir ve μ ile gösterilir. Varyans ifadesinin denklemi $x_1, x_2, \ldots\ x_n$ ler gözlem değerleri olmak üzere

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n} \tag{3.3}$$

şeklindedir.

3.3.3 Kovaryans

İki değişkenin birlikte değişimlerinin ölçüsüne kovaryans denir. *X* ve *Y* rassal değişkenler olmak üzere kovaryans $Cov(X,Y)\ ya\ da\ Kov(X,Y)$ ile gösterilir ve

$$Cov(X,Y) = E\big[(X - E(X)) - \big(Y - E(Y)\big)\big] = E(XY) - E(X)E(Y) \tag{3.4}$$

eşitliklerine sahiptir. Eğer *X* ve *Y* rastgele değişkenleri bağımsız ise $E(XY) = E(X)E(Y)$ olacağından $Cov(X,Y) = 0$ olur. Ancak $Cov(X,Y) = 0$ olması *X* ve *Y* rastgele değişkenlerinin bağımsız olduğu anlamına gelmez. Bu durunda iki değişken arasındaki doğrusal ilişkinin sıfır olduğu sonucuna ulaşılır **(**Mahaboob, B., Praveen, J. P., Appa Rao, B. V., Harnath, Y., Narayana, C., and Prakash, G. B. 2020).

3.3.4 Rassal Hata

Rassal hata, düzeltilemez ve ölçme sonuçlarına nereden, hangi miktarda, nasıl karıştığı bilinmeyen hatalara denir. Hata kaynağı belli değildir. Güvenirliği düşürür. Müşteri anketlerini doldururken bazı müşterilerin soruları yanlış anlaması ve rastgele yanıtlar vermesi bu tip hatalara örnektir. Bir öğretmenin yazılı sınav kağıtlarını incelerken bazı soruların cevaplarını fark etmesi tesadüfi hatalara bir örnek teşkil eder. Şimdi çoklu doğrusal regresyon modeline dair yapılan çalışmaları verelim. Bu çalışmalar ile ilgili daha detaylı bilgilere (Maulud, D. H. ve Abdulazeez, A. M., 2020) kaynağından da ulaşılabilir. Şimdi

$X = \{x_{k1}, x_{k2}, \dots\ x_{ki} : k \in \{1,2, \dots, m\}, i \in \mathrm{Z}^{+}\}$ kümesi verilen değerler,

$Y = \{y_1, y_2, \dots, y_m\}$ kümesine gözlenen değerler,

$\alpha = \{\alpha_0, \alpha_1, \dots, \alpha_i\}$ regresyon katsayıları,

$E = \{\varepsilon_1, \varepsilon_2, \dots, \varepsilon_m\}$ rassal hatalar

olmak üzere bu regresyon modelinde kullanılan denklemimizi verebiliriz. Bu denklem

$$Y = \alpha_0 + \alpha_1 x_{k1} + \alpha_2 x_{k2} + \cdots + \alpha_i x_{ki} + E \tag{3.5}$$

şeklindedir (Maulud, D. H. ve Abdulazeez, A. M., 2020).

Böylece $\beta = \{\beta_1, \beta_2, \dots, \beta_m\}$ katsayılar vektörü olmak üzere

$$y_k = \alpha_0 + \alpha_1 x_{k1} + \alpha_2 x_{k2} + \cdots + \alpha_i x_{ki} + \varepsilon_k \tag{3.6}$$

$$y_k = \alpha_0 + \sum_{t=1}^{i} \alpha_t X_{kt} + \varepsilon_k \tag{3.7}$$

denklemiyle gösterilir. Çoklu doğrusal regresyon modelinde $E = \{\varepsilon_1, \varepsilon_2, \dots, \varepsilon_m\}$ rassal hataları için varsayımlar şu şekildedir:

(i) k = 1,2,....m için ε_k ların beklenen değeri sıfır ise y_k ların beklenen değeri $\alpha_0 + \alpha_1 x_{k1} + \alpha_2 x_{k1} + \cdots + \alpha_i x_{ki}$ şeklindedir.

(ii) k = 1,2,...m olmak üzere Var $(\varepsilon_k) = \sigma^2$ için Var $(y_k) = \sigma^2$ dir.

(iii) k = 2,3,...m için Cov $(\varepsilon_k, \varepsilon_l) = 0$ ise Cov $(y_k, \gamma_1) = 0$ dır.

Tezimizin bu kesiminde Çoklu Doğrusal Regresyon (Multiple Linear Regression) modelinin matris formuna dair çalışmalar yapacağız. Bunun için gözlemlerin her birinin oluşturduğu aşağıda verilen denklemleri düşünelim. Bu denklemler

$$y_1 = \alpha_0 + \alpha_1 x_{11} + \alpha_2 x_{12} + \cdots + \alpha_i x_{1i} + \varepsilon_1$$

$$y_2 = \alpha_0 + \alpha_1 x_{21} + \alpha_2 x_{22} + \cdots + \alpha_i x_{2i} + \varepsilon_2$$

$$\cdots$$

$$y_m = \alpha_0 + \alpha_1 x_{m1} + \alpha_2 x_{m2} + \cdots + \alpha_i x_{mi} + \varepsilon_m$$

şeklindedir. Bu denklemler kullanılarak

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1i} \\ 1 & x_{21} & x_{22} & \cdots & x_{2i} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & x_{m2} & \cdots & x_{mi} \end{bmatrix} \begin{bmatrix} \alpha_o \\ \alpha_1 \\ \vdots \\ \alpha_i \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_m \end{bmatrix}$$

matris formu elde edilir. Bu formda

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1i} \\ 1 & x_{21} & x_{22} & \cdots & x_{2i} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & x_{m2} & \cdots & x_{mi} \end{bmatrix}, \quad Y = \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_m \end{bmatrix}, \quad \alpha = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_i \end{bmatrix}, \quad E = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_m \end{bmatrix}$$

olarak alındığında $Y = X\alpha + E$ biçimi elde edilir. Burada

X matrisi $m\ x\ (i+1)$ boyutlu (Verilen değerlerin oluşturduğu matris)

Y matrisi $m\ x\ 1$ boyutlu (Gözlenen değerlerin oluşturduğu matris)

α matrisi $(i+1)x\ 1$ boyutlu (Regresyon katsayılarının oluşturduğu matris)

E matrisi $m\ x\ 1$ boyutlu (Rassal Hataların oluşturduğu matris)

şeklindedir **(**Mahaboob, B., Praveen, J. P., Appa Rao, B. V., Harnath, Y., Narayana, C., and Prakash, G. B. 2020).

3.1 Örnek: Bu kısımda Çoklu Doğrusal Regresyon modeline ait bir örnek inceleyeceğiz. Bunun için;

$$y = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 \tag{3.8}$$

çoklu doğrusal regresyon modeline ait denkleme dair aşağıda verilen çıktı değeri y, iki tahmin değeri X_1 ve X_2 değerlerin yer aldığı Tablo 3.1 düşünelim.

y	**X1**	**X2**
140	60	22
155	62	25
159	67	24
179	70	20
192	71	15
200	72	14
212	75	14
215	78	11

Tablo 3.1: Çoklu Doğrusal Regresyon Modeli Eğitim Seti.

Bu veri kümesine çoklu doğrusal regresyon modelini elde etmek için aşağıdaki adımları uygulayacağız. Bu adımlar

$$\sum x_1^2 = \sum X_1^2 - \frac{(X_1)^2}{n} = 38{,}767 - \frac{(555)^2}{8} = 263{,}875$$

$$\sum x_2^2 = \sum X_2^2 - \frac{(\sum X_2)^2}{n} = 2{,}823 - \frac{(145)^2}{8} = 194{,}875$$

$$\sum x_1 y = \sum X_1 y - \frac{\sum X_1 \sum y}{n} = 101{,}895 - \frac{555 * 1.452}{8} = 1.162{,}5$$

$$\sum x_2 y = \sum X_2 y - \frac{\sum X_2 \sum y}{n} = 25.364 - \frac{145 * 1.452}{8} = -953{,}5$$

$$\sum x_1 x_2 = \sum X_1 X_2 - \frac{\sum X_1 \sum x_2}{n} = 9.859 - \frac{555 * 145}{8} = -200{,}375$$

şeklindedir. Böylece regresyon toplamlarını veren hesaplamalar Tablo 3.2 da verilmiştir.

	y	X1	X2
0	140	60	22
1	155	62	25
2	159	67	24
3	179	70	20
4	192	71	15
5	200	72	14
6	212	75	14
7	215	78	11
Sum	1452	555	145
Mean	322,6667	123,3333	32,22222

	X_1^2	X_2^2	X1*y	X2*y	X1*X2
0	3600	484	8400	3080	1320
1	3844	625	9610	3875	1550
2	4489	576	10653	3816	1608
3	4900	400	12530	3580	1400
4	5041	225	13632	2880	1065
5	5184	196	14400	2800	1008
6	5625	196	15900	2968	1050
7	6084	121	16770	2365	858
Sum	38767	2823	101895	25364	9859

	X1	X2	X1*y	X2*y	X1*X2
Reg Sums	263,875	194,875	1162,5	-953,5	-200,375

Tablo 3.2: Çoklu Doğrusal Regresyon Modeli Sonuçları

Tablo 3.2 elde ettiğimiz verilerin toplamları kullanarak aşağıdaki denklemlerde kullanalım.

$$y = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 \quad (3.9)$$

denkleminde $\alpha_0, \alpha_1 \; ve \; \alpha_2$katsayılarını bulacağız. Bu katsayılar

$$\alpha_1 = \frac{[(\sum x_2^2)(\sum x_1 y) - (\sum x_1 x_2)(\sum x_2 y)]}{[(\sum x_1^2)(\sum x_2^2) - (\sum x_1 x_2)^2]}$$

$$\alpha_1 = \frac{[(194.875)(1162.5) - (-200.375)(-953.5)]}{[(263.875)(194.875) - (-200.375)^2]} = 3{,}148$$

$$\alpha_2 = \frac{[(\sum x_1^2)(\sum x_2 y) - (\sum x_1 x_2)(\sum x_1 y)]}{[(\sum x_1^2)(\sum x_2^2) - (\sum x_1 x_2)^2]}$$

$$\alpha_2 = \frac{[(263.875)(-953.5) - (-200.375)(1152.5)]}{[(263.875)(194.875) - (-200.375)^2]} = -1{,}656$$

$\alpha_0 =$ (181,5)-(3,148)(69,375)-(-1,656)(18,125)= -6,867

şeklinde bulunur. Bulunan bu katsayılar

$$y = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2$$

denkleminde yerine yazıldığında

$$y = -6{,}867 + 3{,}148 x_1 - 1{,}656 x_2$$

elde edilir.

3.2 Örnek: Çoklu Doğrusal Regresyon modelini somutlaştırmak adına veri grubu üzerinden bir örnek vereceğiz. 10 kişilik bir öğrenci grubu üzerinden alınan verilerde (https://www.kaggle.com/datasets) yaş, derse katılım durumu, haftalık çalışma saati, geçmiş dönem notları, diğer aktiviteler, aile desteği ve yılsonu notlarından oluşan veri grubunu inceledik. Bu veri grubu aşağıdaki şekilde gösterilmiştir.

StudentID	Name	Gender	Attendance Rate	StudyHoursPerWeek	PreviousGrade	ExtracurricularActivities	ParentalSupport	FinalGrade
1	John	Male	85	15	78	1	High	80
2	Sarah	Female	90	20	85	2	Medium	87
3	Alox	Male	78	10	65	0	Low	68
4	Michael	Male	92	25	90	3	High	92
5	Emma	Female	88	18	82	2	Medium	85
6	Olivia	Female	94	30	88	1	High	90
7	Daniel	Male	70	8	60	0	Low	62
8	Sophin	Female	85	17	77	1	Medium	78
9	James	Male	82	12	70	2	Low	72
10	Isabella	Female	91	22	86	3	High	88

Tablo 3.3: Çoklu Doğrusal Regresyon Modeli Veri Seti

Buradaki amacımız elimizde olan final sonuçlarını sisteme girmeden tahmin etmek olacak. Çoklu doğrusal regresyon modelinde bir önceki basit doğrusal regresyon modelindeki gibi kütüphaneler kullanılarak doğrusal regresyon modelini kullanacağız. Eğitim ve test seti olarak rastgele oluşan veri setinde bazı öğrencilerin notlarını bu regresyon modeli ile tahmin edeceğiz. Bu regresyon modelinin en önemli özelliği birden fazla faktörü ele alarak tahmin yapması olacaktır. Bu örneklem özelinde yazdığımız Python kodları ve elde ettiğimiz sonuçların görseli aşağıda gösterilmiştir.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset = pd.read_csv('studentperformance.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
print(X)
```

```
[['John' 'Male' 85 15 78 1 'High']
 ['Sarah' 'Female' 90 20 85 2 'Medium']
 ['Alex' 'Male' 78 10 65 0 'Low']
 ['Michael' 'Male' 92 25 90 3 'High']
 ['Emma' 'Female' 88 18 82 2 'Medium']
 ['Olivia' 'Female' 95 30 88 1 'High']
 ['Daniel' 'Male' 70 8 60 0 'Low']
 ['Sophia' 'Female' 85 17 77 1 'Medium']
 ['James' 'Male' 82 12 70 2 'Low']
 ['Isabella' 'Female' 91 22 86 3 'High']]
```

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [3])], remainder='passthrough')
X = np.array(ct.fit_transform(X))
print(X)
```

```
[[0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 85 15 78 1]
 [0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 1.0 90 20 85 2]
 [1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 78 10 65 0]
 [0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 92 25 90 3]
 [0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 88 18 82 2]
 [0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 95 30 88 1]
 [0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 70 8 60 0]
 [0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 1.0 85 17 77 1]
 [0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 82 12 70 2]
 [0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 91 22 86 3]]
```

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

```
[[70.18 68.  ]
 [77.65 72.  ]
 [83.11 85.  ]
 [88.44 88.  ]
 [85.53 87.  ]]
```

Şekil 3.1: Çoklu Doğrusal Regresyon Örneğinin Python Kodları

Yukarıdaki görselde ilk sütundaki veriler tahmin ikinci sütundaki veriler ise öğrencilerin gerçek sonuçları. Sırasıyla 3.sıradaki Alex, 9.sıradaki James, 5.sıradaki Emma, 10.sıradaki Isabellia, 2.sıradaki Sarah öğrencileri için 1.sırdaki sonuçlar tahmin 2.sıradaki sonuçlar ise gerçek sonuçlarıdır. Doğrusal regresyon modelini kullanarak gerçek değere yakın değerleri tahmin etmiş bulunmaktayız.

3.4 Polinom Regresyon (Polinomial Regression)

Bağımsız değişken x ile bağımlı değişken y arasındaki bağlantının bir polinom olarak modellendiği bir regresyon analizi biçimine polinom regresyonu denir.

Bu regresyon aşağıdaki

$$y = \alpha_o + \alpha_1 x + \alpha_2 x^2 + \alpha_3 x^3 + \cdots + \alpha_k x^k + \varepsilon \tag{3.10}$$

denklemi kullanılır. Burada y değerleri ile doğru üzerinde yer alan teorik değerleri $\hat{y}$ ifadesi ile göstereceğiz. Polinom regresyon modeli ile çoklu doğrusal regresyon denklemleri arasında bağıntı vardır. Dikkat edilirse doğrusal regresyon modelinde modelleme yapmak için 1.dereceden

$$y_k = \alpha_0 + \alpha_1 x_{k1} + \alpha_2 x_{k2} + \cdots + \alpha_i x_{ki} + \varepsilon_k \tag{3.11}$$

polinomunu kullanmıştık. Burada

$$x_{ki} = x^i$$

eşitliğini düşündüğümüzde

$$\hat{y}_i = \widehat{\alpha_o} + \widehat{\alpha_1} x_i + \widehat{\alpha_2} {x_i}^2 + \widehat{\alpha_3} {x_i}^3 + \cdots + \widehat{\alpha_k} {x_i}^k \tag{3.12}$$

bu bağlantı kurulmuş olur. Bu denklemden ve matrislerden yararlanarak en küçük kareler yöntemini kullanacağız. Şimdi en küçük kareler yönteminin tanımını yapalım (Sinha, P., 2013).

3.4.1 Polinom Regresyon Modelinde En Küçük Kareler Yöntemi

En Küçük Kareler yöntemi, bir regresyon modelindeki parametreleri (eğim ve kesim noktası gibi katsayıları) tahmin etmek için kullanılan temel bir istatistiksel tekniktir. Bu yöntemin temel prensibi, bağımlı değişkenin gözlemlenen gerçek değerleri ile model tarafından tahmin edilen değerler arasındaki farkların (yani artıkların veya hataların) kareleri toplamını minimize eden bir doğruyu en uygun "regresyon doğrusu" olarak belirlemektir. Bu yöntemde aşağıdaki denklem kullanılır.

$$\text{En Küçük Kareler Yöntemi} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{3.13}$$

Parametrelerin tahmini En Küçük Kareler Yöntemi ile yapacağız. Uygun regresyon denkleminin aşağıdaki gibi ele alalım.

$$\hat{y} = \widehat{\alpha_o} + \widehat{\alpha_1}x \quad + \widehat{\alpha_2}x^2 + \widehat{\alpha_3}x^3 + \cdots + \widehat{\alpha_k}x^k \tag{3.14}$$

en küçük kareler yöntemi veriler üzerinden elde ettiğimiz noktalar arasındaki uzaklıkların kareleri toplamını minimum yapan α katsayılarını bulma işlemi olduğunu hatırlatalım. En küçük kareler yöntemi uyguladığımızda

$$\mathrm{F} = \sum_{i=1}^{n} (y_i - (\widehat{\alpha_o} + \widehat{\alpha_1}x \quad + \widehat{\alpha_2}x^2 + \widehat{\alpha_3}x^3 + \cdots + \widehat{\alpha_k}x^k))^2 \tag{3.15}$$

denklemini elde etmiş olduk. Kısmı türevler alınıp sıfıra eşitlendiğinde

$$\frac{\partial F}{\partial \widehat{\alpha_0}} = \frac{\partial F}{\partial \widehat{\alpha_1}} = \frac{\partial F}{\partial \widehat{\alpha_2}} = \cdots = \frac{\partial F}{\partial \widehat{\alpha_k}} = 0$$

$$\frac{\partial F}{\partial \widehat{\alpha_0}} = \sum_{i=1}^{n} -2[y_i - (\widehat{\alpha_o} + \widehat{\alpha_1}x \quad + \widehat{\alpha_2}x^2 + \widehat{\alpha_3}x^3 + \cdots + \widehat{\alpha_k}x^k)] = 0$$

$$\frac{\partial F}{\partial \widehat{\alpha_1}} = \sum_{i=1}^{n} -2[y_i - (\widehat{\alpha_o} + \widehat{\alpha_1}x \quad + \widehat{\alpha_2}x^2 + \widehat{\alpha_3}x^3 + \cdots + \widehat{\alpha_k}x^k)x]$$

$$\cdots$$

$$\frac{\partial F}{\partial \widehat{\alpha_k}} = \sum_{i=1}^{n} -2[y_i - (\widehat{\alpha_o} + \widehat{\alpha_1}x \quad + \widehat{\alpha_2}x^2 + \widehat{\alpha_3}x^3 + \cdots + \widehat{\alpha_k}x^k)x^k]$$

ifadeleri elde edilir. Daha sonra eşitlikler sadeleştirilip toplam sembolü ifadeleri dağıtılır.

$$\sum_{i=1}^{n} y_i - \sum_{i=1}^{n} \widehat{\alpha_o} - \sum_{i=1}^{n} \widehat{\alpha_1}x \quad - \cdots - \sum_{i=1}^{n} \widehat{\alpha_k}x^k = 0$$

$$\sum_{i=1}^{n} y_i - \sum_{i=1}^{n} \widehat{\alpha_o}\, x - \sum_{i=1}^{n} \widehat{\alpha_1} x \quad x - \cdots - \sum_{i=1}^{n} \widehat{\alpha_k} x^k x = 0$$

$$\cdots$$

$$\sum_{i=1}^{n} y_i - \sum_{i=1}^{n} \widehat{\alpha_o}\, x^k - \sum_{i=1}^{n} \widehat{\alpha_1} x \quad x^k - \cdots - \sum_{i=1}^{n} \widehat{\alpha_k} x^k x^k = 0$$

sadeleştirilip toplam sembolü ifadeleri dağıtılır. Elde ettiğimiz bu ifadeleri düzenlediğimizde

$$\sum_{i=1}^{n} y_i = \sum_{i=1}^{n} \widehat{\alpha_o} - \sum_{i=1}^{n} \widehat{\alpha_1} x \quad - \cdots - \sum_{i=1}^{n} \widehat{\alpha_k} x^k$$

$$\sum_{i=1}^{n} y_i = \sum_{i=1}^{n} \widehat{\alpha_o}\, x - \sum_{i=1}^{n} \widehat{\alpha_1} x \quad x - \cdots - \sum_{i=1}^{n} \widehat{\alpha_k} x^k x$$

$$\ldots.$$

$$\sum_{i=1}^{n} y_i = \sum_{i=1}^{n} \widehat{\alpha_o}\, x^k - \sum_{i=1}^{n} \widehat{\alpha_1} x \quad x^k - \cdots - \sum_{i=1}^{n} \widehat{\alpha_k} x^k x^k$$

eşitlikleri elde edilir.

$$\sum_{i=1}^{n} \widehat{\alpha_o} = \widehat{\alpha_o} n$$

olduğu açıkça görülmektedir. Bu ifade denklemlerde yerine yazıldığında

$$\sum_{i=1}^{n} y_i = \widehat{\alpha_o} n - \widehat{\alpha_1} \sum_{i=1}^{n} x \quad - \cdots - \widehat{\alpha_k} \sum_{i=1}^{n} x^k$$

$$\sum_{i=1}^{n} x_i y_i = \widehat{\alpha_o} \sum_{i=1}^{n} x - \widehat{\alpha_1} \sum_{i=1}^{n} x^2 - \cdots - \widehat{\alpha_k} \sum_{i=1}^{n} x^{k+1}$$

$$\cdots$$

$$\sum_{i=1}^{n} x^k y_i = \widehat{\alpha_o} \sum_{i=1}^{n} x^k - \widehat{\alpha_1} \sum_{i=1}^{n} x^{k+1} - \cdots - \widehat{\alpha_k} \sum_{i=1}^{n} x^{2k}$$

eşitlikleri elde edilir. Elde ettiğimiz bu denklemlerin matris formunu aşağıda inceleyeceğiz. Bu matris formu

$$\begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} xy_i \\ \vdots \\ \sum_{i=1}^{n} x^k y_i \end{bmatrix} = \begin{bmatrix} n & \sum_{i=1}^{n} x & \sum_{i=1}^{n} x^2 & \cdots & \sum_{i=1}^{n} x^k \\ \sum_{i=1}^{n} x & \sum_{i=1}^{n} x^2 & \sum_{i=1}^{n} x^3 & \cdots & \sum_{i=1}^{n} x^{k+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{n} x^k & \sum_{i=1}^{n} x^{k+1} & \sum_{i=1}^{n} x^{k+2} & \cdots & \sum_{i=1}^{n} x^{2k} \end{bmatrix} \begin{bmatrix} \widehat{\alpha_o} \\ \widehat{\alpha_1} \\ \vdots \\ \widehat{\alpha_k} \end{bmatrix}$$

bu şekilde elde edilir. Elde ettiğimiz bu matris formu ile en küçük kareler doğrusu elde edeceğiz. En küçük kareler doğrusunun

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1 & \cdots & {x_1}^{n-1} \\ 1 & x_2 & x_2 & \cdots & {x_2}^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m & \cdots & {x_m}^{n-1} \end{bmatrix} \begin{bmatrix} \widehat{\alpha_o} \\ \widehat{\alpha_1} \\ \vdots \\ \widehat{\alpha_k} \end{bmatrix}$$

matrisini elde ettik. Bu matrisin her iki tarafını

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ x_1 & x_2 & x_3 & \cdots & x_m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ {x_1}^{n-1} & {x_2}^{n-1} & {x_3}^{n-1} & \cdots & {x_m}^{n-1} \end{bmatrix}$$

matrisi ile çarpalım. Çarpım sonucunda

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ x_1 & x_2 & x_3 & \cdots & x_m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ {x_1}^{n-1} & {x_2}^{n-1} & {x_3}^{n-1} & \cdots & {x_m}^{n-1} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ x_1 & x_2 & x_3 & \cdots & x_m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ {x_1}^{n-1} & {x_2}^{n-1} & {x_3}^{n-1} & \cdots & {x_m}^{n-1} \end{bmatrix} \begin{bmatrix} 1 & x_1 & x_1 & \cdots & {x_1}^{n-1} \\ 1 & x_2 & x_2 & \cdots & {x_2}^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m & \cdots & {x_m}^{n-1} \end{bmatrix} \begin{bmatrix} \widehat{\alpha_o} \\ \widehat{\alpha_1} \\ \vdots \\ \widehat{\alpha_k} \end{bmatrix}$$

matris eşitliğini elde ediyoruz. Yapmış olduğumuz bu transpoze işlemi sonrasında $(\widehat{\alpha_o}, \widehat{\alpha_1}, \dots, \widehat{\alpha_k})$ polinom katsayıları elde edilir. Matris ifadesinde polinom doğrusu için denklem

$$Y = X\hat{\alpha}$$

biçimindedir. X^T ifadesi X matrisinin transpozesidir. Buradan

$$X^T Y = X^T X\hat{\alpha}$$

eşitliği elde edilir. Polinom katsayıları

$$\hat{\alpha} = (X^T X)^{-1} X^T Y$$

formuyla elde edilir (Sinha, P., 2013).

3.4.2 Değişim Katsayısı

Bir veri grubunda verilen bir $y_1, y_2, \dots, y_n$ sayı dizisindeki elemanların aritmetik ortalaması

$$\bar{y}_i = \frac{\sum y_i}{n}, i = 1, \dots, n \tag{3.16}$$

şeklindedir. Ayrıca standart sapma, veri değerlerinin aritmetik ortalamadan farklarının karelerinin toplamının veri sayısı n-1'e bölümünün karekökü olup

$$S_x = \sum (y_i - \bar{y}_i)^2 \tag{3.17}$$

için

$$\sigma_y = \sqrt{\frac{S_x}{n-1}} \tag{3.18}$$

ile ifade edilir. Benzer şekilde varyans,

$$\sigma_y{}^2 = \frac{\sum(y_i - \bar{y}_i)^2}{n-1} \tag{3.19}$$

ile gösterilir. Değişim katsayısı

$$Değişim\ Katsayısı\ = \frac{\sigma_y}{\bar{y}_i}\ 100 \tag{3.20}$$

şeklinde gösterilir (Arslankaya, S. ve Toprak, Ş. , 2021).

3.4.3 Standart Hata ve Belirleme Katsayısı

Standart hata, bir istatistiğin aynı popülasyondan çekilebilecek farklı örneklemlerde ne kadar değişkenlik gösterebileceğinin bir ölçüsüdür. Daha teknik bir ifadeyle, bir istatistiğin örnekleme dağılımının standart sapmasıdır. Standart hata, örneklem aracılığıyla yapılan bir tahminin, bilinmeyen popülasyon parametresini ne kadar hassas bir şekilde temsil ettiğini gösterir. Daha düşük bir standart hata, tahminin daha güvenilir ve hassas olduğuna işaret eder. Bu standart hata ve belirleme katsayısı

$$Standart\ Hata\text{: } S_{\frac{y}{x}} = \sqrt{\frac{S_r}{n-(m+1)}} \tag{3.21}$$

$$\text{Belirleme Katsayısı: } r^2 = \frac{S_x - S_r}{S_x} \tag{3.22}$$

şeklinde gösterilir. Burada *m* değeri polinom derecesi, *n* değeri veri gurundaki girdi sayısıdır ve buradaki denklemler

$$S_r = \sum {\varepsilon_i}^2 = \sum \left(\hat{y}_i - \left(\widehat{\alpha_o} + \widehat{\alpha_1} x_i \quad + \widehat{\alpha_2} {x_i}^2 + \widehat{\alpha_3} {x_i}^3 + \cdots + \widehat{\alpha_k} {x_i}^k \right) \right)^2 \qquad (3.23)$$

şeklindedir (Arslankaya, S. ve Toprak, Ş. , 2021).

3.3 Örnek: Bu kısımla ilgili bir örnek üzerinde çalışalım. Polinom regresyon modelinde en genel halde verdiğimiz modeli 2.derece denklem üzerinde düşünelim. Bu denklemde

$$\hat{y}_i = \widehat{\alpha_o} + \widehat{\alpha_1} x_i \quad + \widehat{\alpha_2} {x_i}^2 + \varepsilon_i \qquad (3.24)$$

sabit değeri yalnız bıraktığımızda

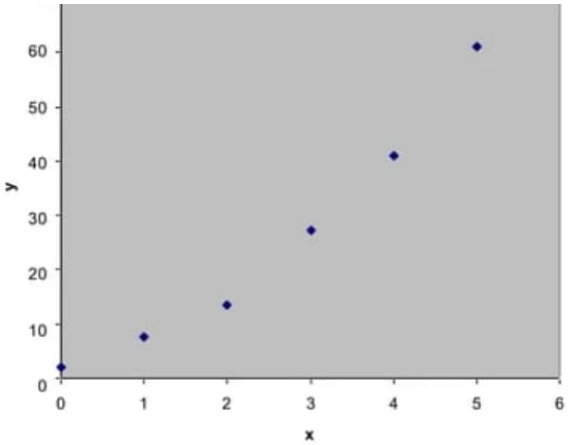
$$\varepsilon_i = \hat{y}_i - \widehat{\alpha_o} - \widehat{\alpha_1} x_i \quad - \widehat{\alpha_2} {x_i}^2 \qquad (3.25)$$

elde etmiş oluyoruz. Sabit sayıların kareleri toplamını

$$S_r = \sum {\varepsilon_i}^2 = \sum \left(\hat{y}_i - \widehat{\alpha_o} - \widehat{\alpha_1} x_i \quad - \widehat{\alpha_2} {x_i}^2 \right)^2 \qquad (3.26)$$

şeklindedir. Bu örnekte 2.dereceden polinom içerecek şekilde bir veri grubunu ele alalım.

x_i	y_i
0	2.1
1	7.7
2	13.6
3	27.2
4	40.9
5	61.1



Şekil 3.2: Polinom Regresyon Modeli Eğitim Seti

Elimizdeki bu veriler toplamda 6 adet olup ve $n = 6$ dır. Böylece elde ettiğimiz polinom regresyonuna ait matris gösterimi

$$\begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} {x_i}^2 y_i \end{bmatrix} = \begin{bmatrix} n & \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} {x_i}^2 \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} {x_i}^2 & \sum_{i=1}^{n} {x_i}^3 \\ \sum_{i=1}^{n} {x_i}^2 & \sum_{i=1}^{n} {x_i}^3 & \sum_{i=1}^{n} {x_i}^4 \end{bmatrix} \begin{bmatrix} \widehat{\alpha_0} \\ \widehat{\alpha_1} \\ \widehat{\alpha_2} \end{bmatrix}$$

şeklindedir. Bu matris gösteriminde verilerimizin değerlerini yerine yazdığımızda

$$\begin{bmatrix} 152{,}6 \\ 585{,}6 \\ 2488{,}8 \end{bmatrix} = \begin{bmatrix} 6 & 15 & 55 \\ 15 & 55 & 225 \\ 55 & 225 & 979 \end{bmatrix} \begin{bmatrix} \widehat{\alpha_0} \\ \widehat{\alpha_1} \\ \widehat{\alpha_2} \end{bmatrix}$$

Gauss eleminasyon yöntemi kullanılarak matristeki sabitler olan

$$\begin{bmatrix} \widehat{\alpha_0} \\ \widehat{\alpha_1} \\ \widehat{\alpha_2} \end{bmatrix} = \begin{bmatrix} 2{,}47857 \\ 2{,}35979 \\ 1{,}86071 \end{bmatrix}$$

Şeklinde bulunur. Elde edilen bu değerler denklemde yerine yazıldığında

$$y = 2{,}47857 + 2{,}35979x + 1{,}86071x^2$$

ifadesi elde edilir. Bu veri grubuna ait hesaplamamızdaki standart hata ve belirleme katsayısı

$$s_{y/x} = \sqrt{\frac{3.74657}{6-(2+1)}} = 1.12$$

x_i	y_i	$(y_i - \bar{y})^2$	$(y_i - a_0 - a_1x_i - a_2x_i^2)^2$
0	2.1	544.44	0.14332
1	7.7	314.47	1.00286
2	13.6	140.03	1.08158
3	27.2	3.1211	0.80491
4	40.9	239.22	0.61951
5	61.1	1272.1	0.09439
15	152.6	2513.4	3.74657

$$s_{y/x} = \sqrt{\frac{S_r}{n-(m+1)}}$$

Şekil 3.3: Standart Hata

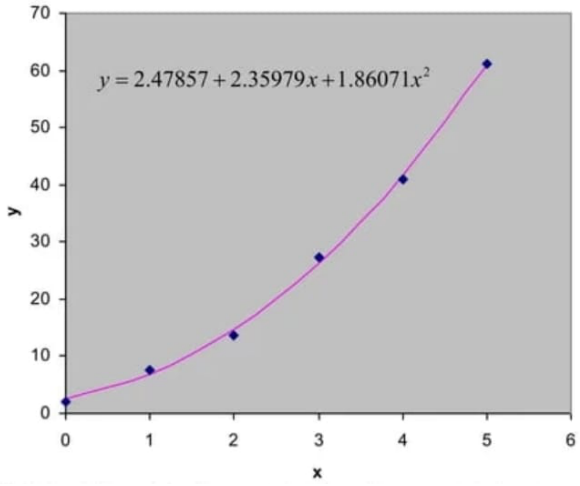
$$r^2 = \frac{2513.4 - 3.74657}{2513.4} = 0.99851$$

x_i	y_i	$(y_i - \bar{y})^2$	$(y_i - a_0 - a_1x_i - a_2x_i^2)^2$
0	2.1	544.44	0.14332
1	7.7	314.47	1.00286
2	13.6	140.03	1.08158
3	27.2	3.1211	0.80491
4	40.9	239.22	0.61951
5	61.1	1272.1	0.09439
15	152.6	2513.4	3.74657

$$r^2 = \frac{S_t - S_r}{S_t}$$

Şekil 3.4: Standart Hata ve Belirleme Katsayısı

şekildeki gibidir. Bu hesaplamalar sonucunda elde ettiğimiz polinom regresyon modeli grafiği Sy/x R square ne anlam veriyor.

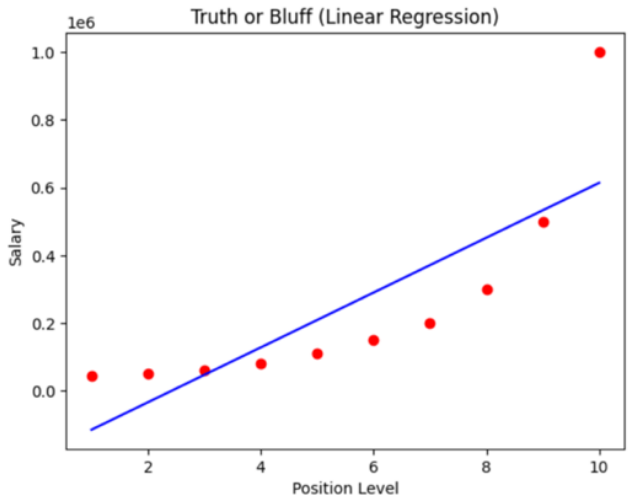


Şekil 3.5: Polinom Regresyon Modeli Grafiği

3.4 Örnek: Bu kısımda bir örnek üzerinde çalışalım. Polinom regresyonu modelini küçük veri grubunda örnek olarak tahmini değerlerimizi inceleyeceğiz. Bir önceki bölümdeki çoklu lineer regresyon modeline baz alacak olursak çok fazla benzer özellikleri mevcut. Bu iki regresyonu birbirinden ayıran en temel özellik yapılacak çizimin veya denklemin derecesinin 2 ve 2 den büyük olması. Çoklu doğrusal regresyon modelinde 1.dereceden bir denklem ile tahmini değerimize ulaşıyorduk ancak burada oluşturacağımız denklem derecesi 2 ve 2 den büyük derecelere sahip. Tahmin sürecinde Polinom regresyonu modelinde istediğimiz derece ile tahminimizi yapmamız mümkün. Elde ettiğimiz veriler 3 sütundan oluşuyor ancak biz bu veri grubunda 2 veri üzerinden tahmin yapacağız. Veri grubumuzda yaş, tecrübe ve gelir bilgileri mevcut. Biz yaş üzerinden gelir tahmini yapacağız. Aşağıdaki görselde bu veri grubuna ait bilgiler verilmiştir.

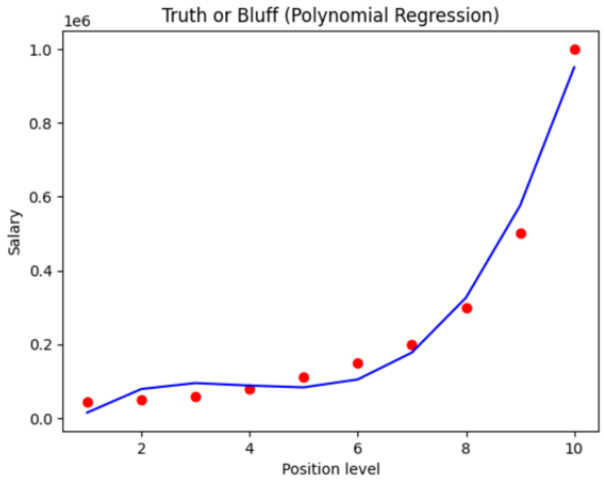
```
dataset = pd.read_csv('Position_Salaries.csv')
X = dataset.iloc[:, 1:-1].values
y = dataset.iloc[:, -1].values
```

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 3)
X_poly = poly_reg.fit_transform(X)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(X_poly, y)
plt.scatter(X, y, color = 'red')
plt.plot(X, lin_reg.predict(X), color = 'blue')
plt.title('Truth or Bluff (Linear Regression)')
plt.xlabel('Position Level')
plt.ylabel('Salary')
plt.show()
```



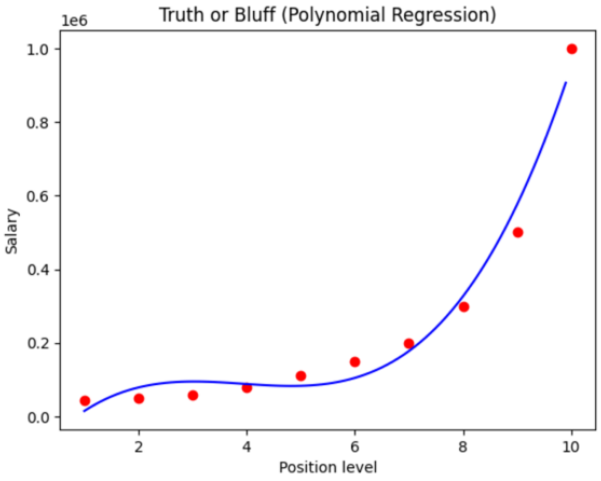
Şekil 3.6: Lineer Regresyon Modeli Grafiği

```
plt.scatter(X, y, color = 'red')
plt.plot(X, lin_reg_2.predict(poly_reg.fit_transform(X)), color = 'blue')
plt.title('Truth or Bluff (Polynomial Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```



Şekil 3.7: Polinom Regresyon Modeli Grafiği

```
X_grid = np.arange(min(X), max(X), 0.1)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X, y, color = 'red')
plt.plot(X_grid, lin_reg_2.predict(poly_reg.fit_transform(X_grid)), color = 'blue')
plt.title('Truth or Bluff (Polynomial Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```



Şekil 3.8: Polinom Regresyon Modeli Grafiği

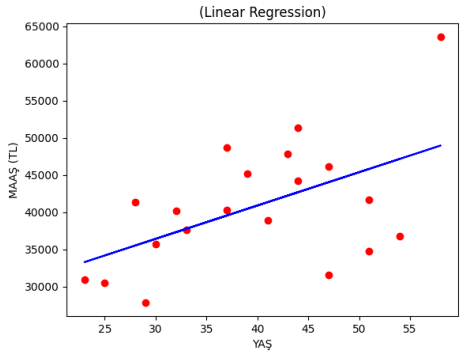
```
lin_reg.predict([[6.5]])
```

```
array([330378.78787879])
```

```
lin_reg_2.predict(poly_reg.fit_transform([[6.5]]))
```

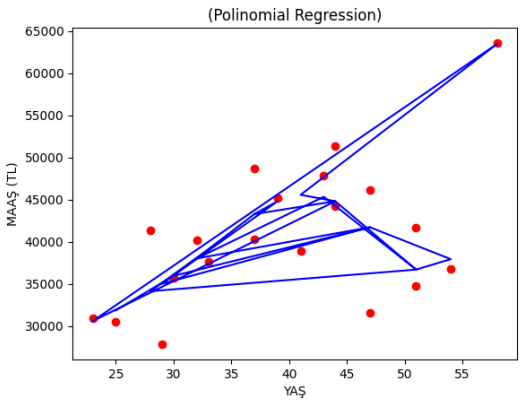
```
array([133259.46969697])
```

Bu veri grubunu Python programında gerekli kodlar yazılarak hem Polinom regresyonu hem de Çoklu doğrusal regresyon modelleri üzerinden tahminlerimizi yapacağız. Bu kısımda her iki modelinde kıyaslaması anlamında daha detaylı bilgiler elde edeceğiz. Gerekli kodlar yazıldıktan sonra elde ettiğimiz verileri grafiği Çoklu doğrusal modeli üzerinden aşağıdaki şekilde gösterilmiştir. Burada eldeki verilerle Çoklu doğrusal regresyon modeli kullanarak bir doğru denklemi üzerinden tahmini değerlerimize yaklaşmamıza olanak sağlıyor.



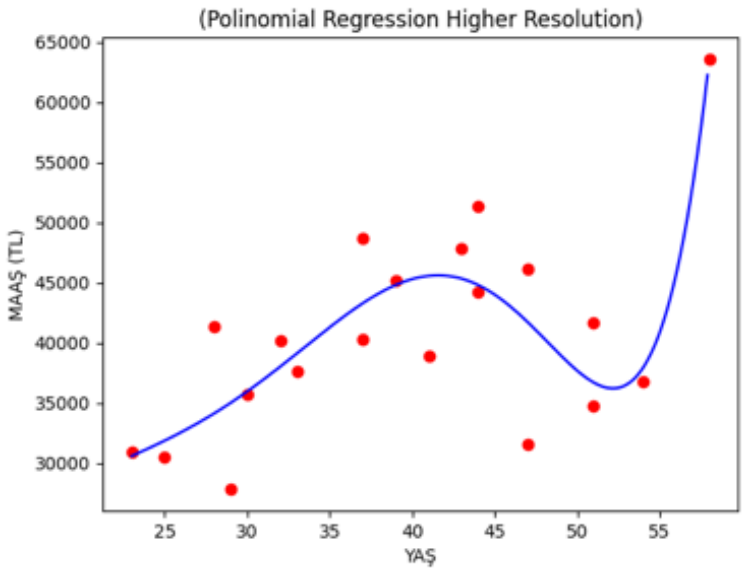
Şekil 3.9: Lineer Regresyon Modeli Grafiği

Aşağıdaki grafik polinom regresyonu modeli üzerinden bir grafik çizimine ait. Bu çizim çok karmaşık bir görüntüye sahip sebebi elimizdeki verilerin dağılımdan kaynaklı. Her zaman elimizdeki veriler düzgün, periyodik, sıralı olmayabilir.



Şekil 3.10: Polinom Regresyon Modeli Grafiği

Bir önceki grafik karmaşıklığından bahsetmiştik. Python programa dilinde Polinom regresyon modeliniz daha iyi eğitmek adına polinom regresyonu Hisger Resolution metodu kullanılarak elimizdeki verilere en yakın eğri denklemini ve grafiği oluşturması bize yardımcı olacaktır. Burada tahminlerimizi karmaşıklıktan çıkarıp daha iyi çözümlere sahip olmamız.

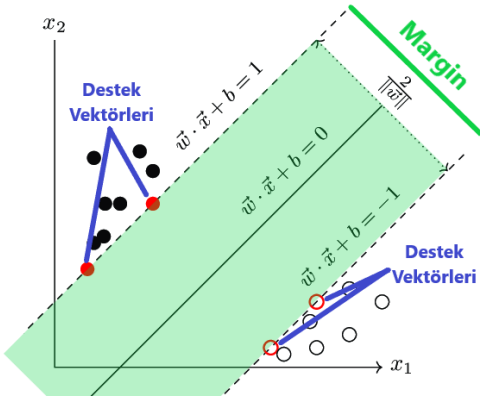


Şekil 3.11: Polinom Regresyon Modeli Grafiği

3.5 Destek Vektör Makine Regresyon (Support Vector Machine Regression)

Destek Vektör Makineleri Regresyon Modeli hem sınıflandırma hem de regresyon problemleri için kullanılabilen, güçlü ve popüler bir denetimli öğrenme algoritmasıdır. Destek Vektör Makineleri'nin temel amacı, farklı sınıflara ait veri noktalarını birbirinden ayırmak için en uygun hiperdüzlemi (hyperplane) bulmaktır. İki boyutlu bir öznitelik uzayında bu hiperdüzlem bir doğru iken, çok boyutlu uzaylarda bir düzlem veya daha yüksek boyutlu bir yapıdır.

Algoritmanın ayırt edici özelliği, bu ayırıcı hiperdüzlemi, her iki sınıfa en yakın veri noktaları arasındaki mesafeyi, yani marjı (margin), en büyük edecek şekilde konumlandırmasıdır. Hiperdüzlemin konumunu belirleyen ve marj üzerinde yer alan bu en yakın noktalara destek vektörleri (support vectors) denir. Algoritma adını, modelin yalnızca bu kritik veri noktalarına dayanarak inşa edilmesinden alır ve kullanmasında bu durum ele alınır.



Şekil 3.12: Destek Vektör Makineleri

Destek Vektör Makineleri Regresyon modelinde giriş özellik vektörleri *x* ve bunlara karşılık gelen sınıf etiketleri *y* den oluşan bir eğitim veri kümemiz vardır. Doğrusal hiper düzlem için denklem bu regresyon modelinde denklem

$$w^T.x + b = 0 \tag{3.27}$$

şeklinde yazılır. Bu denklemde w vektörü hiper düzlemin normal vektörünü temsil eder. Bu vektörü hiper düzleme dik olan yön olarak belirtilir. Denklemdeki b parametresi, hiper düzlemin w normal vektörü boyunca orijinden uzaklığını veya mesafesi olarak temsil edilir. Ayrıca bir veri noktası x_i ile karar sınırı arasındaki mesafe

$$d_i = \frac{w^T.x_i + b}{\|w\|} \tag{3.28}$$

şeklinde hesaplanır. Burada $\|w\|$ ağırlık vektörü *w* nun Öklid normunu temsil eder. Normal vektör *w* nin Öklid normu Doğrusal Destek Vektör Makineleri (DVM) sınıflandırıcı için

$$\hat{y} = \begin{cases} 1: & w^T.x + b \geq 0 \\ 0: & w^T.x + b < 0 \end{cases} \tag{3.29}$$

biçimindedir. Bu regresyon modelinde optimizasyon sabit marjlı doğrusal DVM sınıflandırıcı için

$$minimize_{w,b}\frac{1}{2}w^T.w = minimize_{w,b}\frac{1}{2}\|w\|^2$$

$$y_i(w^T.x_i + b) \geq 1 \text{ için } i = 1,2,3,\dots,m$$

biçimindedir. Bu ifadede *i* inci eğitim örneği için hedef değişken veya etiket t_i sembolüyle gösterilir. Ve negatif olaylar için $t_i = -1$ ($y_i = 0$ olduğunda) ve pozitif olaylar için $t_i = 1$ ($y_i = 1$ olduğunda). Çünkü kısıtlamayı karşılayan karar sınırını gerektiririz,

$$t_i(w^T.x_i + b) \geq 1$$

Yumuşak marjlı doğrusal DVM sınıflandırıcısı için,

$$minimize_{w,b}\frac{1}{2}w^T.w + C\sum_{i=1}^{m}\xi_i$$

$$y_i(w^T.x_i + b) \geq 1 - \xi_i \qquad \xi_i \geq 0 \text{ için } \quad i = 1,2,3,\dots,m$$

Çift problem durumu ise destek vektörleriyle ilgili Lagrange çarpanlarını bulmayı gerektiren optimizasyon probleminin çift problemi DVM'yi çözmek için kullanılabilir. Aşağıdaki çift amaç fonksiyonunu maksimize eden en iyi Lagrange çarpanları *α(i)*

$$maximize_{\alpha}: \frac{1}{2}\sum_{i\to m}\sum_{j\to m}\alpha_i\alpha_j t_i t_j K(x_i, x_j) - \sum_{i\to m}\alpha_i$$

burada,

$$w = \sum_{j\to m}\alpha_i t_i K(x_i, x) + b$$

$$t_i(w^T.x_i - b) = 1 \Leftrightarrow b = w^T.x_i - t_i$$

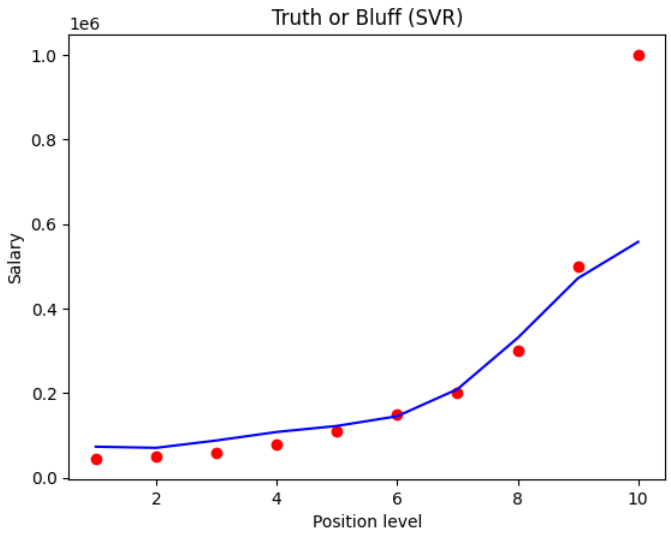
dir (Yacoub, M. H., Ismail, S. M., Saida, L. A., Madian, A. H., and Radwan, A. G. , 2024).

3.5 Örnek: Bu kısımda DVM için bir örnek üzerinde Python programı üzerinden uygulama inceleyeceğiz. Aşağıda bir firmaya ait pozisyonlar , bu pozisyonların karşılığına denk gelen maaşlar ve bu verileri kullanarak incelediğimiz Python kodları mevcuttur. Bu verileri kullanarak DVM regresyonu kullanarak tahmini inceleyeceğiz. Elde ettiğim verileri tablo halinde position,level ve salary sütunlarında ele aldım.

Position	Level	Salary
Business Analyst	1	45000
Junior Consultant	2	50000
Senior Consultant	3	60000
Manager	4	80000
Country Manager	5	110000
Region Manager	6	150000
Partner	7	200000
Senior Partner	8	300000
C-level	9	500000
CEO	10	1000000

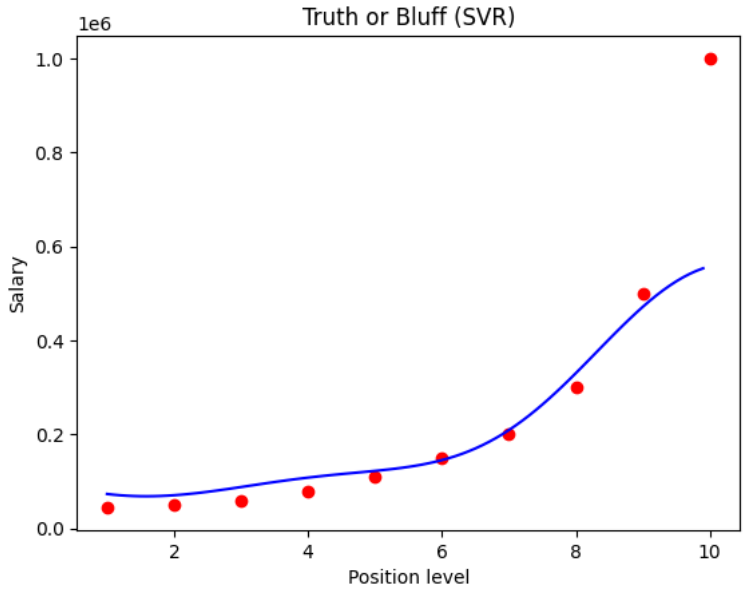
Tablo 3.4: DVM Örnek Veri Seti

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_y = StandardScaler()
X = sc_X.fit_transform(X)
y = sc_y.fit_transform(y)
```



Şekil 3.13: Destek Vektör Makineleri Grafiği

```
plt.title('Truth or Bluff (SVR)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```



Şekil 3.14: Destek Vektör Makineleri Modeli Grafiği

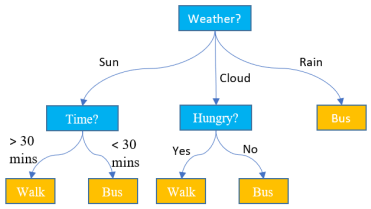
Verileri kullanarak elde ettiğimiz Destek Vektör Makineleri algoritmaları ile ettiğimiz grafiği yukarıdaki görselde göstermiş olduk. Bu grafiklerle birlikte grafiklere denk gelen denklemleri kullanarak farklı verilere karşılık gelecek tahminlerde bulunabiliriz.

3.6 Karar Ağacı Regresyon (Decision Tree Regression)

Karar ağacı regresyon modeli, makine öğrenmesinde tahmini modelleme görevleri için yaygın olarak kullanılan bir algoritmadır. Karar ağacı regresyon algoritmaları hem regresyon hem de sınıflandırma için kullanılır. Algoritmanın nasıl çalıştığını, tümevarım olduğunu, yapısını tanımlayan parametreleri ve avantajlarını ve sınırlamalarını inceleyeceğiz. Bir karar ağacı, bir veri kümesini geçmenin bir yolunu tanımlarken aynı zamanda beklenen sonuçlara ağaç benzeri bir yol tanımlamak için etkili bir algoritmadır. Bir ağaçtaki bu dallanma, kontrol ifadelerine veya değerlerine dayanır ve veri noktaları, belirli bir özelliğin değerine bağlı olarak bölme düğümünün her iki tarafında bulunur.

Bir karar ağacının yapısı, en önemli bölme özelliği olan bir kök düğüm tarafından tanımlanabilir. Dahili düğümler bir öznitelik üzerinde testlerdir. Örneğin, dahili bir düğümün bir kontrol ifadesi varsa, bu koşulu karşılayan veri noktaları bir tarafta, geri kalanı diğer taraftadır. Yaprak düğümleri, veri kümesinin temsil ettiği mevcut sınıflara aittir.

Karar ağacı indüksiyonunu bir örnek üzerinde inceleyelim. Yürüyüp yürümeyeceğine veya otobüse binip binmeyeceğine karar verme problemi için Karar Ağacı algoritmasını uygulayarak, bir kök düğüm, dahili düğümler, yaprak düğümleri seçerek ve ardından sınıf için bölme kriterlerini adım adım tanımlayarak bir karar ağacı geliştirebiliriz. En önemli düğümü kök düğüm olarak seçiyoruz. Örnek olarak hava tahminlerini göz önünde bulunduralım. Hava durumuna bağlı olarak zaman ve açlık gibi diğer koşullara bağlı olarak ağacımızı şu şekilde inşa edebiliriz (https://medium.com/swlh/decision-tree-regression-and-its-mathematical-implementation-58c6e9c5e88e).



Şekil 3.15: Karar Ağacı Modeli Örneği

Karar ağacı modelinde düğümleri nasıl böleceğimizi inceleyeceğiz. Bir öznitelik seçim ölçüsü, verilerin bireysel sınıflarla sonuçlanacak şekilde nasıl bölüneceğine karar vermede en yetenekli bölme kriterini seçmek için sezgiseldir. Öznitelik seçim ölçümleri, veri noktalarının bir karar ağacında belirli bir seviyede nasıl bölüneceğini tanımladıkları için bölme kuralları olarak da bilinir. Ölçü için en iyi puanı alan özellik, verilen veri noktaları için bölme özelliği olarak seçilir. Bu kısımda karar ağacı regresyon modelinde kullanılan bilgi kazancı hakkında incelemede bulunacağız. Bu özellik, ağacı daha iyi tanımlamak için gereken bilgi miktarı açısından bize bölme özelliğini sağlar. Bilgi kazancı, veri noktalarını ilgili bölümlere sınıflandırmak için gereken bilgileri en düşük seviyeye indirir ve bu bölümlerdeki en az rastgeleliği veya "kirliliği" yansıtır (https://fritz.ai/the-mathematics-behind-decision-trees/).

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i) \tag{3.30}$$

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \; x \; Info(D_j) \tag{3.31}$$

Yukarıdaki denklemdeki p_i, D veri kümesindeki rastgele bir kümenin C_i sınıfına ait olma ve $|C_i$,D|/|D| ile tahmin edilme olasılığıdır. Bilgi (D), D deki bir veri noktasının sınıfını/kategorisini tanımlamak için gereken ortalama bilgi miktarıdır. Bilgi kazancı şu şekilde hesaplanabilir,

$$Gain(A) = Info(D) - Info_A(D) \tag{3.32}$$

Kazanç Oranı ise bilgi kazancı ölçüsü, birçok sonuca sahip testlere yönelik önyargılardır. Bu nedenle, çok sayıda değere sahip özellikleri seçmeyi tercih eder. Kazanç oranı bu sorunu iyileştirme girişimidir. Kazan oranını veren formül aşağıdaki gibidir.

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \; x \; \log_2\left(\frac{|D_j|}{|D|}\right) \tag{3.33}$$

$$\text{GainRatio(A)} = \frac{Gain(A)}{SplitInfo(A)} \tag{3.34}$$

Gini indeksi şu şekilde hesaplanır.

$$Gini(D) = 1 - \sum_{i=1}^{m} {p_i}^2 \tag{3.35}$$

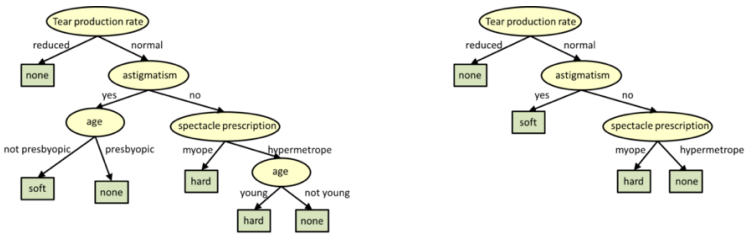
Burada p_i ifadesi D deki bir kümenin C_i sınıfına ait olma olasılığıdır ve $|C_i, D|/|D|$ ifadesi ile tahmin edilir. Toplam m sınıfı üzerinden hesaplanır. G ini endeksi, her özellik için ikili bir bölmeyi dikkate alır. Örneğin, yaşın üç olası değeri varsa, yani {düşük, orta, yüksek}, o zaman olası alt kümeler {düşük, orta, yüksek}, {düşük, orta}, {düşük, yüksek}, {orta,

yüksek}, {düşük}, {orta}, {orta}, {yüksek} ve {}'dir (hesaplamalarımızdaki güç ve üst kümeyi göz ardı ederek).

$$Gini_{(D)} = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2) \tag{3.36}$$

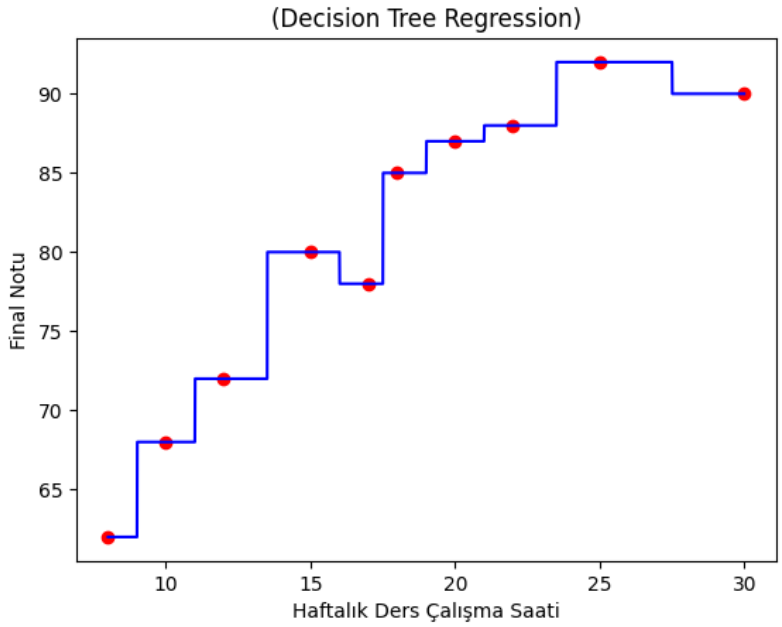
$$\Delta Gini(A) = Gini(D) - Gini_A(D) \tag{3.37}$$

Tamamen büyütülmüş bir karar ağacının dalları, genellikle eğitim verisindeki temel örüntülerin yanı sıra gürültü ve aykırılıklar gibi tesadüfi anormallikleri de yansıtır. Modelin bu şekilde eğitim setindeki her detayı ezberlemesi, aşırı uyuma (overfitting) ve yüksek varyansa yol açar. Bu durum, modelin eğitim verilerindeki küçük değişimlere karşı aşırı duyarlı olmasına ve bunun sonucunda tahminlerinde tutarsızlıklar göstermesine neden olur. Bu problemin çözümü için budama (pruning) teknikleri kullanılır (Mesarić, J. and Šebalj, D. 2016). Aşırı uyumun (overfitting) yarattığı kararsızlığı önlemek ve karar ağaçlarının genelleme performansını iyileştirmek için temel strateji budama (pruning) yapmaktır. Budama, istatistiksel olarak anlamlı olmayan veya çok az sayıda örnekle desteklenen ve bu nedenle modelin tahmin gücünü zayıflattığı düşünülen dalların ağaçtan çıkarılmasıyla modelin basitleştirilmesi işlemidir. Bu işlemin sonucunda, daha az karmaşık, daha hızlı ve daha kolay yorumlanabilir bir model elde edilir. En önemlisi, budanmış bir ağaç, eğitim setindeki gürültüden arındığı için genellikle görünmeyen test verileri üzerinde daha isabetli tahminler yapar. Bu amaçla, ağacın büyümesini erken durduran ön-budama (pre-pruning) veya tam büyümüş ağacı basitleştiren sonradan-budama (post-pruning) gibi farklı teknikler mevcuttur (https://medium.com/swlh/decision-tree-regression-and-its-mathematical-implementation-58c6e9c5e88e).



Şekil 3.16: Karar Ağacı Modeli Örneği

Bu kısımda ise Karar Ağacı Regresyon Modeli ile ilgili bir uygulama yapacağız. Bu örnek haftalık ders çalışma saatine karşılık gelen final sınav notlarının incelenmesi şeklinde olacak. Aşağıdaki grafikte temsili olarak ele aldığımız haftalık ders çalışma saatine karşılık final notları gösterilmiştir.



Şekil 3.17: Karar Ağacı Regresyon Modeli

Bu grafik üzerinden tahminlerimizi yapacağız. Aşağıdaki şekilde girdiğimiz haftalık çalışma saatine göre final notlarının tahminlerinin yer aldığı değerler yer almaktadır. Bu tahminleri elde ederken Decision Tree Regression modelini kullanarak elde ettik.

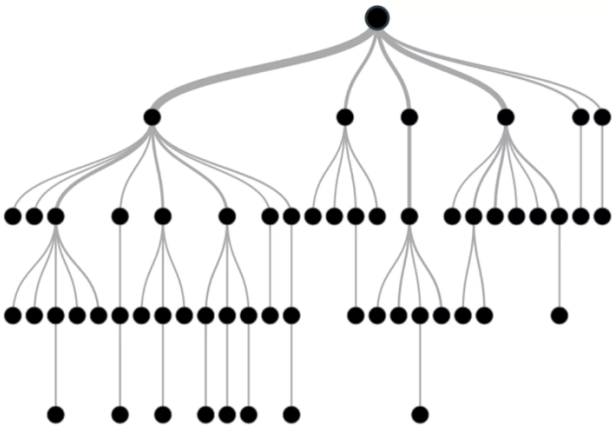
```
regressor.predict([[10]])
array([68.])

[133] regressor.predict([[16]])
array([80.])

[134] regressor.predict([[20]])
array([87.])

[135] regressor.predict([[5]])
array([62.])

[136] regressor.predict([[30]])
array([90.])
```

Şekil 3.18: Karar Ağacı Regresyon Modeli Sonuçları

3.7 Rastgele Orman Regresyon (Random Forest Regression)

Rastgele Orman Regresyon **(**Random Forest Regression), karmaşık sorunlara çözüm sağlamak için birçok zayıf sınıflandırıcıyı birleştiren, topluluk öğrenmeyi kullanan bir tekniktir. Rastgele Orman Regresyonu, çok sayıda karar ağacının tahminlerini birleştirerek tek bir ağacın aşırı uyum (overfitting) eğilimini ve kararsızlığını azaltan bir topluluk öğrenmesi tekniğidir. Model, her biri eğitim verisinin rastgele bir alt örneği üzerinde ve her bir düğümde özniteliklerin rastgele bir alt kümesi kullanılarak eğitilmiş yüzlerce veya binlerce karar ağacı oluşturur. Bir tahminleme (regresyon) görevi için, ormandaki her bir ağacın ürettiği sayısal tahminlerin ortalamasıalınarak nihai ve daha sağlam bir sonuç elde edilir. Bu yaklaşım, bireysel ağaçların hatalarını kolektif olarak dengeleyerek daha doğru tahminler yapılmasını sağlar. Bir örnek yardımıyla rastgele ormanları açıklayalım. Tek başına bir yolculuğa çıkmamız gerektiğini varsayalım. Bir tepe istasyonuna mı yoksa biraz macera yapmak için bir yere mi gitmek istediğinizden emin değiliz. Yani, arkadaşına git ve ona ne önerdiğini sor, diyelim ki arkadaş 1 (F1) zaten Kasım olduğu için bir tepe istasyonuna gitmeni söylüyor ve bu orada eğlenmek için harika bir zaman olacak, arkadaş 2 (F2) maceraya gitmeni istiyor. Benzer şekilde, tüm arkadaşlarınız size bir seyahate çıkabileceğiniz yerlerde önerilerde bulundu. Sonunda, ya seçtiğiniz bir yere gidebilir ya da arkadaşlarınızın çoğunun önerdiği bir yere karar verebilirsiniz.

Benzer şekilde Rastgele Orman Regresyon bir dizi karar ağacını eğitiyoruz ve maksimum oyalan sınıf, bir sınıflandırma problemiyse nihai sonuç ve bir regresyon sorunuysa ortalama oluyor. Karar ağaçları, bir dizi özellik tabanlı bölünmeden kaynaklanan tahminleri göstermek için bir ağaç yapısı gibi bir akış şeması kullanır. Bir kök düğümle başlar ve yapraklar tarafından verilen bir kararla sona erer (Edalı, M. 2020).



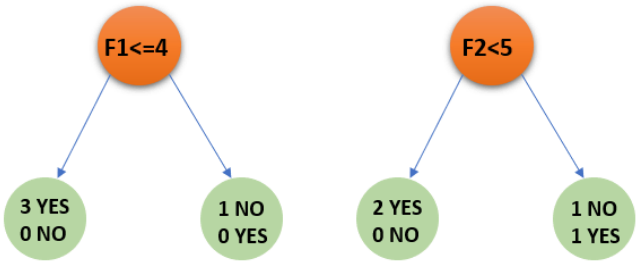
Şekil 3.18: Rastgele Orman Modeli

Kök düğümü, karar düğümü, yaprak düğümü olan 3 bileşenden oluşur. Nüfusun bölünmeye başladığı düğüme kök düğüm denir. Bir kök düğümü böldükten sonra elde ettiğimiz düğümlere karar düğümleri ve daha fazla bölmenin mümkün olmadığı düğüme yaprak düğümü denir. Bir veri kümesinde 100'lerce özellik olabilir, bu nedenle hangi özelliğin kök düğümümüz olacağına "Gini Endeksi" ifadesini kullanarak çözümleyeceğiz. Gini Endeksi; daha fazla bölmek için bir özellik seçmeyi, bu bölünmenin ne kadar saf olacağını bilmemiz gerekir. Saf bir alt bölünme, ya "evet" ya da "hayır" almanız gerektiği anlamına gelir (https://www.analyticsvidhya.com/blog/2021/10/an-introduction-to-random-forest-algorithm-for-beginners/). Aşağıdaki veri kümesini örnek alarak açıklayalım.

Feature 1	Feature 2	Output
2	7	Yes
3	3	Yes
6	5	No
4	1	Yes

Şekil 3.19: Rastgele Orman Modeli

Her özelliği kök düğümümüz olarak alarak böldükten sonra hangi çıktıyı elde edeceğimizi göreceğiz.



Şekil 3.20: Rastgele Orman Modeli

Kök düğümümüz olarak 1. özelliği aldığımızda saf bir bölünme elde ederken, 2. özelliği aldığımızda bölme saf değildir. Peki bu özel düğümün ne kadar safsızlığa sahip olduğunu nasıl bilebiliriz? Bu, "Gini Endeksi" yardımıyla anlaşılabilir.

Temel olarak veri kümemizin safsızlığını bilmemiz gerekiyor ve bu özelliği en düşük safsızlığı veren veya hangisinin en düşük Gini indeksine sahip olduğunu söyleyen kök düğüm olarak alacağız. Matematiksel olarak Gini indeksi şu şekilde yazılır.

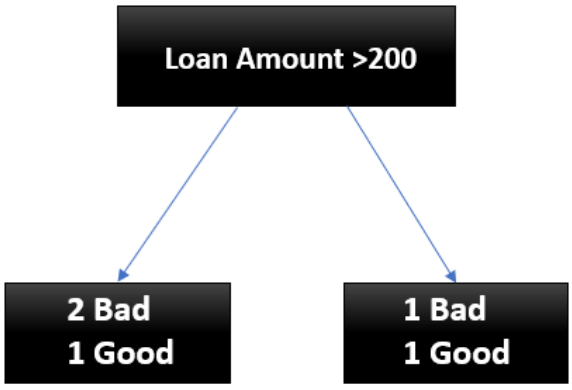
$$Gini\ Index = 1 - \sum_{i=1}^{n} (P_i)^2 = 1 - [(P_+)^2 + (P_-)^2] \tag{3.38}$$

Burada P_+ pozitif bir sınıfın olasılığı ve P_- negatif bir sınıfın olasılığıdır. Bu formülü bir oyuncak veri kümesi yardımıyla anlayalım.

ID	Loan Amount	Loan Status
1	100	Bad
2	200	Good
3	250	Bad
4	400	Good
5	300	Bad

Şekil 3.21: Rastgele Orman Modeli

Kredi Tutarını kök düğümümüz olarak alalım ve bölmeye çalışalım.



Şekil 3.22: Rastgele Orman Modeli

Sol bölmenin değerlerini elde ettiğimiz formüle koyarak

$$\text{Gini Index } = 1 - \sum_{i=1}^{n} (P_i)^2 = 1 - [(P_+)^2 + (P_-)^2]$$

$$= 1 - \left[\left(\frac{1}{3}\right)^2 + \left(\frac{2}{3}\right)^2\right]$$

$$= 1 - [0{,}1089 + 0{,}4356]$$

$= 1 - 0{,}5445 = 0{,}4555$

Değerini elde ederiz. Doğru bölme için Gini endeksi şöyle olacaktır,

$$\text{Gini Index } = 1 - \sum_{i=1}^{n} (P_i)^2 = 1 - [(P_+)^2 + (P\_)^2]$$

$$= 1 - \left[\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2\right]$$

$$= 1 - [0{,}25 + 0{,}25]$$

$$= 1 - 0{,}5 = 0{,}5$$

Şimdi bu bölünmenin toplam Gini Endeksi olan Ağırlıklı Gini Endeksini hesaplamamız gerekiyor. Bu ise;

$$\text{Weighted Gini Index } = \frac{3}{5} * 0{,}4555 + \frac{2}{5} * 0{,}5$$

$$= \frac{6}{10} * 0{,}4555 + \frac{4}{10} * 0{,}5$$

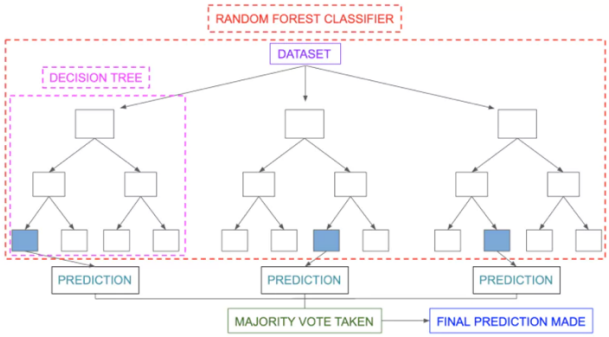
$$= 0{,}2733 + 0{,}2 = 0{,}4733$$

Şeklinde hesaplanır. Benzer şekilde, bu algoritma mümkün olan tüm bölmelerin Gini indeksini bulmaya çalışacak ve en düşük Gini indeksini verecek olan kök düğüm için bu özelliği seçecektir. En az Gini indeksi, az safsızlık anlamıda olur. Bölünmenin safsızlığını ölçmek için de kullanılan ifade "Entropi" değeridir. Entropi için matematiksel formül

$$E(S) = -p_{(+)} \log p_{(+)} - p_{(-)} \log p_{(-)} \tag{3.39}$$

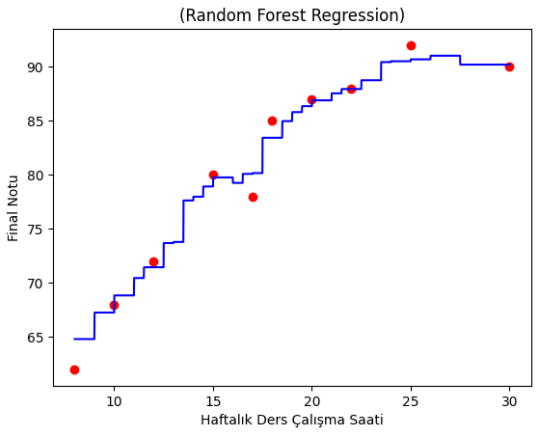
biçimindedir. Hesaplamalı olarak verimli olduğu için genellikle Gini indeksini kullanırız, burada entropide olduğu gibi logaritmik bir terim olmadığı için yürütme için daha kısa bir süre alır. Genellikle, logaritmik hesaplamalar yapmak istiyorsak biraz zaman alır. Bu nedenle birçok güçlendirme algoritması parametre olarak Gini indeksini

kullanır. Burada dikkat edilmesi gereken önemli bir şey daha, belirli bir düğümde her iki sınıfın eşit sayısı varsa, Gini indeksinin maksimum değerine sahip olacağıdır, bu da düğümün oldukça saf olmadığı anlamına gelir. Rastgele Orman Algoritmasında yer alan adımları şu şekilde belirtebiliriz. Önce orijinal verilerimizin alt kümelerini yapıyoruz. Satır örneklemesi ve özellik örneklemesi yapıp, burada değiştirme ile satır ve sütunları seçeceğimiz ve eğitim veri kümesinin alt kümelerini oluşturacağımız anlamına geliyor. Aldığımız her alt küme için ayrı bir karar ağacı oluşturuyoruz. Her karar ağacı bir çıktı verecektir. Nihai çıktı, bir sınıflandırma ise Çoğunluk Oylamasına ve bir regresyon modeli ortalamaya göre değerlendirilir (Biau, G. 2012).



Şekil 3.23: Rastgele Orman Modeli

Bu kısımda bir örnek vererek devam edelim. Bu regresyon modelini küçük bir veri grubu üzerinden uygulamasını yapacağız. Elimizdeki veri grubunda bir grup öğrencinin haftalık ders çalışma saatinin, final notları üzerinden haftalık ders çalışma saatine göre final notlarını tahmin edeceğiz. Aşağıdaki şekilde Rastgele Orman Regresyon modeli kullanılarak grafiği şekilde gösterilmiştir (Salama, M. K., 2024).



Şekil 3.24: Rastgele Orman Regresyon Modeli

Bu grafik üzerinden tahminlerimizi yapacağız. Aşağıdaki şekilde girdiğimiz haftalık çalışma saatine göre final notlarının tahminleri yer almaktadır. Bu tahminleri elde ederken Rastgele Orman Regresyon modelini kullanarak elde ettik.

Şekil 3.25: Rastgele Orman Regresyon Modeli Sonuçları

Rastgele Orman Regresyon modelini kullanarak elde ettiğimiz tahminleri grafikte göstermiş bulunduk.

3.8 Lasso Regresyon

Lasso regresyonu, küçültme (shrinkage) yöntemi kullanılarak yapılan bir düzenlileştirme (regularization) tekniğidir ve cezalandırılmış regresyon yöntemi olarak da bilinir. Lasso, "Least Absolute Shrinkage and Selection Operator" ifadesinin kısaltmasıdır. Hem düzenleştirme hem de model seçimi amacıyla kullanılır. Bir model L1 düzenlileştirme tekniğini kullanıyorsa, bu model Lasso regresyonu olarak adlandırılır. Bu küçültme tekniğinde, aşağıdaki denklemde belirlenen doğrusal modeldeki katsayılar, ortalama (mean) değerine doğru küçültülerek bir cezalandırma faktörü olan α (alfa) değeri ile düzenlenir.

$$L_{lasso}(\hat{\alpha}) = \sum_{i=1}^{n}(y_i - x_i'\hat{\alpha})^2 + \lambda \sum_{j=1}^{m}|\hat{\alpha}_j| \tag{3.40}$$

Alfa (α), modelde uygulanacak küçültme miktarını belirleyen ceza terimidir. Alfa sıfır olarak ayarlandığında, bu durum doğrusal regresyon modeline eşdeğer olur. Alfa değeri büyüdükçe, optimizasyon fonksiyonuna uygulanan ceza da artar. Bu nedenle, Lasso regresyonu katsayıları küçülterek modelin karmaşıklığını azaltır ve çoklu doğrusal bağlantıyı (multicollinearity) düşürmeye yardımcı olur. Alfa (α) sıfır ile sonsuz arasında

herhangi bir reel değer alabilir. Alfa değeri arttıkça, cezalandırma daha agresif hale gelir ve modelde daha fazla değişkenin etkisi azaltılır. Lasso regresyonunun önemli bir özelliği, katsayıları sıfıra doğru küçültmesi nedeniyle, veri setindeki daha az önemli değişkenleri ortadan kaldırmasıdır. Belirlenen alfa değeri, bazı değişkenlerin sıfıra indirgenmesine neden olarak otomatik bir değişken seçimi (feature selection) mekanizması oluşturur. Bu sayede, gereksiz girdiler elenir ve daha etkili bir model oluşturulur (Piri, M., 2023).

3.9 Ridge Regresyon

Lasso regresyonuna benzer şekilde Ridge regresyonu da katsayılara bir kısıtlama getirerek cezalandırma faktörü uygular. Ancak temel fark, Lasso regresyonunun katsayıların mutlak değerini (L1 normu) kullanmasına karşın, Ridge regresyonunun karelerini (L2 normu) kullanmasıdır (Piri, M., 2023).

$$L_{ridge}(\hat{\alpha}) = \sum_{i=1}^{n} (y_i - x_i'\hat{\alpha})^2 + \lambda \sum_{j=1}^{m} w_j \hat{\alpha}_j^2 \tag{3.41}$$

Bu nedenle Ridge regresyonu, L2 düzenlileştirme (L2 Regularization) olarak da bilinir ve özellikle tüm değişkenleri koruyarak aşırı öğrenmeyi (overfitting) önlemek için kullanılır (Kumar, S. and Bhatnagar, V., 2022).

3.10 Elastic Net Regresyon

Elastic Net regresyonu, Lasso (L1) ve Ridge (L2) regresyonlarının birleşimi olarak geliştirilen bir düzenlileştirme (regularization) yöntemidir. Lasso regresyonu, bazı katsayıları sıfıra çekerek değişken seçimi yaparken, Ridge regresyonu katsayıları sıfıra yaklaştırarak çoklu doğrusal bağlantıyı (multicollinearity) azaltır. Elastic Net, bu iki yöntemi birleştirerek hem değişken seçimi yapar hem de çoklu doğrusal bağlantıyı azaltır. Elastic Net, özellikle yüksek boyutlu verilerde ve birbirine yüksek korelasyona sahip (collinear) değişkenlerin bulunduğu durumlarda tercih edilir. Lasso'nun tamamen sıfıra indirgediği değişkenlerin, Ridge bileşeni sayesinde modelde bir miktar tutulmasını sağlar (Kumar, S. and Bhatnagar, V., 2022).

4. LİNEER REGRESYON MODELİ YARDIMIYLA BELİRLİ ARALIKLARDA BAZI FONKSİYONLARIN POLİNOM KARŞILIKLARI

Bu bölümde lineer regresyon modelleri kullanılarak $f(x) = e^{-x^2}$, $f(x) = \sqrt{x^4 + 1}$ ve $f(x) = cosx$ fonksiyonlarının belirli aralıklarda polinom karşılıkları bulunarak, bu karşılıların belirli integrali üzerine çalışmalar yapılmıştır. Yapılan çalışmaların sonuçları ayrıca Trapez Yöntemi, Simpson Yöntemi, Polinom Regresyon Modeli, Riemann Yaklaşımı ve Gauss-Kronrod yöntemleri ile karşılaştırılmıştır.

4.1 Trapez Yöntemi

Trapez yöntemi, belirli integralin yaklaşık hesaplanmasında kullanılan temel sayısal yöntemlerden biridir. Bu yöntemde, verilen fonksiyon belirli aralıklarda doğrusal bir fonksiyon ile yaklaşık olarak ifade edilir. İntegral hesabı yapılırken, fonksiyonun altındaki alan bir dizi yamuk (trapez) ile modellenir ve bu yamukların alanları toplanarak yaklaşık bir sonuç elde edilir. Bu yöntem, özellikle sürekli ve düzgün davranış gösteren fonksiyonlar için etkili bir yaklaşımdır. Ancak, fonksiyonun eğimi çok hızlı değiştiğinde veya doğrusal yaklaşımın yeterli olmadığı durumlarda, hata payı artabilir. Daha kesin sonuçlar elde etmek için alt aralıkların sayısı artırılabilir veya daha gelişmiş sayısal yöntemler tercih edilebilir.

Trapez yöntemi, sayısal analizde sıkça kullanılan ve hesaplaması kolay olan bir yöntemdir. Özellikle mühendislik ve fizik alanlarında, integral hesaplamalarında pratik bir çözüm sunarak birçok uygulamada kullanılmaktadır.

$$I = \int_a^b f(x)dx \tag{4.1}$$

Bu integralin yaklaşık değeri, aralığı eşit parçaya bölerek ve her bölümü bir trapez olarak ele alarak bulunur. Trapez Kuralı, h adım büyüklüğü ile şu şekilde tanımlanır:

$$h = \frac{b-a}{n}$$

Trapezlerin toplam alanı aşağıdaki formülle hesaplanır:

$$I \approx \frac{h}{2}[f(a) + 2\sum_{i=1}^{n-1} f(x_i) + f(b)] \tag{4.2}$$

Burada, a ve b integralin alt ve üst sınırlarını temsil eder. n aralığın kaç bölüme ayrıldığını gösterir. h adım büyüklüğü olup, her alt bölmenin genişliğini belirler. $x_i = a + ih$ alt bölme noktalarını gösterir. Bu yöntem, eşit aralıklı noktalar kullanarak fonksiyonun altındaki alanı yamuklarla yaklaşık hesaplamaktadır.

Trapez kuralının hata terimi, ikinci türev üzerinden hesaplanabilir. Teorik hata formülü şu şekildedir:

$$E_T = -\frac{(b-a)h^2}{12} f''(\xi) \tag{4.3}$$

Burada, $\xi(x_i), [a, b]$ aralığında bir noktadır. $f''(\xi)$ ikinci türev olup, fonksiyonun eğriliğini ifade eder. Bu formüle göre, trapez yönteminin hata büyüklüğü, adım büyüklüğünün karesi O(h^2) mertebesindedir. Yani, adım sayısı n artırıldıkça hata küçülmektedir.

Örneğin, $f(x) = e^{-x^2}$ fonksiyonunun [-2,2] aralığında integralini Trapez Yöntemi ile yaklaşık olarak hesaplayalım.

1. n=4 alarak adım genişliği hesaplanır:
2. $h = \frac{2-(-2)}{4} = 1$
3. Fonksiyon değerleri hesaplanır:
4. $f(-2) = e^4,\ f(-1) = e^1 = e, f(0) = e^0 = 1,\ f(1) = e^1 = e, f(2) = e^4$
5. Yaklaşık integral değeri:

$$\int_{-2}^{2} e^{-x^2} \mathrm{dx} \approx \frac{1}{2}[f(-2) + 2f(-1) + 2f(0) + 2f(1) + 2f(2)]$$

$$\approx \frac{1}{2}(0{,}0183 + 2(0{,}3679) + 2(1) + 2(0{,}3679 + 0{,}0183)$$

$$\approx \tfrac{1}{2}(0{,}0183 + 0{,}7358 + 2 + 0{,}7358 + 0{,}0183)$$

$$\approx 1{,}7541$$

Bu yöntem, sayısal integrasyon uygulamalarında yaygın olarak kullanılmakta olup, Newton-Cotes yöntemleri arasında en temel ve en hızlı yöntemlerden biridir (Çelik, E. 2025).

4.2 Gauss – Kronrod Yöntemi

Gauss kuadratür yöntemi, sayısal integrasyon alanında kullanılan, belirli integrallerin yaklaşık değerini yüksek doğrulukla hesaplamak için geliştirilmiştir. Bu yöntemi

$$\int_{-1}^{1} f(x)\mathrm{dx} \approx \sum_{i=1}^{n} w_i f(x_i) \tag{4.4}$$

şeklinde gösterilebilir (Orive, R., Pejčev, A. V., Spalević, M. M., ve Mihić, L. (2022)).

4.3 Lineer Regresyon Modeli Yardımıyla Belirli Aralıklarda $f(x) = e^{-x^2}$ Fonksiyonunun Polinom Karşılığı

Bu çalışmada, tek bir polinom modelinin uç bölgelerdeki sapmalarını önlemek amacıyla $f(x) = e^{-x^2}$ fonksiyonunun [-6,6] aralığı beş alt parçaya bölünmüş, her parçanın uzunluğunun 50 katı kadar nokta ile örnekleme yapılmış ve dördüncü dereceden polinom regresyonlar her bir alt aralık için ayrı ayrı eğitilmiştir. Dinamik örnekleme stratejisi sayesinde orta bölgede 200, uçlarda ise 100 nokta kullanılarak hem eğitim doğruluğu artırılmış hem de gereksiz hesaplama yükü azaltılmıştır. Parçalı model, merkezde orijinal eğriyle neredeyse kusursuz çakışma göstermiş, uç aralıklarda ise klasik Runge fenomeni yerine istikrarlı bir izleme sergilemiştir. Ortalama mutlak hata ve ortalama kare hatanın her parçada 10^{-3} mertebesinde kalması, yöntemimizin hem merkezî hem de uç bölgelerde yüksek doğruluk sunduğunu doğrulamıştır. Grafiksel karşılaştırmalar, parçalı polinom yaklaşımının geniş aralıkta dengeli bir tahmin sağladığını açıkça ortaya koymaktadır.

```
# 1. Parça aralıklarını ve polinom derecesini ayarla
pieces = [(-6, -4), (-4, -2), (-2, 2), (2, 4), (4, 6)]
degree = 4



piecewise_models = []
piecewise_polys = []



print("Parçalı Polinom Regresyon Denklem Çıktıları:\n")
```

```
for (a, b) in pieces:
    # 1.a. O aralığın verisini oluştur
    x_piece = np.linspace(a, b, 200).reshape(-1, 1)
    y_piece = np.exp(-x_piece**2)

    # 1.b. Polinom dönüşümü ve model eğitimi
    poly = PolynomialFeatures(degree=degree)
    X_poly_piece = poly.fit_transform(x_piece)
    model = LinearRegression()
    model.fit(X_poly_piece, y_piece)

    # Sakla
    piecewise_models.append(model)
    piecewise_polys.append(poly)

    # 1.c. Katsayıları al ve denklem stringi oluştur
    coefs = model.coef_.flatten()
    coefs[0] = model.intercept_
    denklem = f"{a} ≤ x ≤ {b} aralığı için:\n  f(x) = "
    for i, c in enumerate(coefs):
        if i == 0:
            denklem += f"{c:.6f}"
        else:
            sign = "+" if c >= 0 else "-"
            denklem += f" {sign} {abs(c):.6f}"
            if i == 1:
                denklem += "·x"
            else:
                denklem += f"·x^{i}"
    print(denklem + "\n")

# 2. Parçalı tahmin fonksiyonu
def piecewise_predict(x):
    x = np.asarray(x)
```

```
    y_pred = np.zeros_like(x, dtype=float)
    for (a, b), model, poly in zip(pieces, piecewise_models, piecewise_polys):
        mask = (x >= a) & (x <= b)
        if np.any(mask):
            Xp = poly.transform(x[mask].reshape(-1, 1))
            y_pred[mask] = model.predict(Xp).flatten()
    return y_pred

# 3. Karşılaştırma için geniş aralıkta test et
x_test = np.linspace(-6, 6, 800)
y_true = np.exp(-x_test**2)
y_pw   = piecewise_predict(x_test)

plt.figure(figsize=(12, 6))
plt.plot(x_test, y_true, label='Gerçek $e^{-x^2}$', linewidth=2)
plt.plot(x_test, y_pw, '--', label='Parçalı Polinom Regresyon', linewidth=2)
plt.title("Parçalı Polinom Regresyon ile Yaklaşım")
plt.xlabel("x")
plt.ylabel("f(x)")
plt.legend()
plt.grid(True)
plt.show()
```

Parçalı Polinom Regresyon Denklem Çıktıları

-6 ≤ x ≤ -4 aralığı için:

f(x) = 0,000056 + 0,000044·x + 0,000013·x^2 + 0,000002·x^3 + 0,000000·x^4

-4 ≤ x ≤ -2 aralığı için:

f(x) = 0,886009 + 1,095839·x + 0,505805·x^2 + 0,103209·x^3 + 0,007853·x^4

-2 ≤ x ≤ 2 aralığı için:

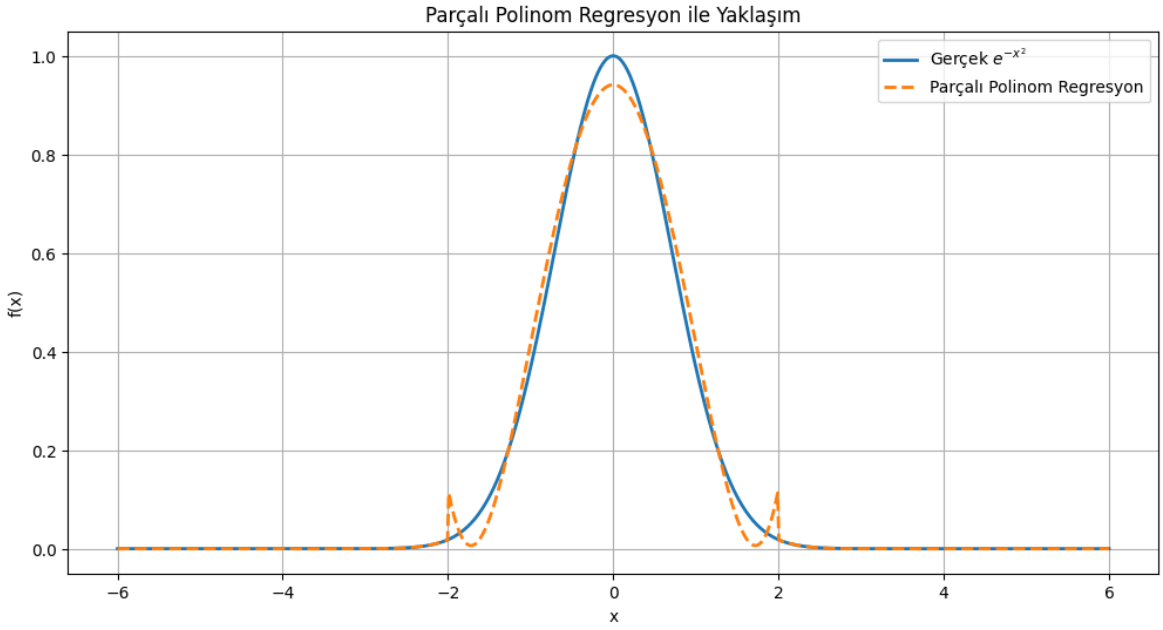
f(x) = 0,941368 + 0,000000·x – 0,631791·x^2 – 0,000000·x^3 + 0,106726·x^4

2 ≤ x ≤ 4 aralığı için:

f(x) = 0,886009 – 1,095839·x + 0,505805·x^2 – 0,103209·x^3 + 0,007853·x^4

4 ≤ x ≤ 6 aralığı için:

f(x) = 0,000056 – 0,000044·x + 0,000013·x^2 – 0,000002·x^3 + 0,000000·x^4



Şekil 4.1: Polinom Regresyon Modeli

Lineer $f(x) = e^{-x^2}$ fonksiyonuna en yakın olarak bulduğumuz

$$f(x) = 0{,}941368 + 0{,}000000 \cdot x - 0{,}631791 \cdot x^2 - 0{,}000000 \cdot x^3 + 0{,}106726 \cdot x^4$$

temel integral alma yöntemleri kullanarak elde ettiğimiz integral sonucu

$$\int_{-2}^{2} (0{,}941368 + 0{,}000000 \cdot x - 0{,}631791 \cdot x^2 - 0{,}000000 \cdot x^3 + 0{,}106726 \cdot x^4) dx$$

$$= 1{,}7618$$

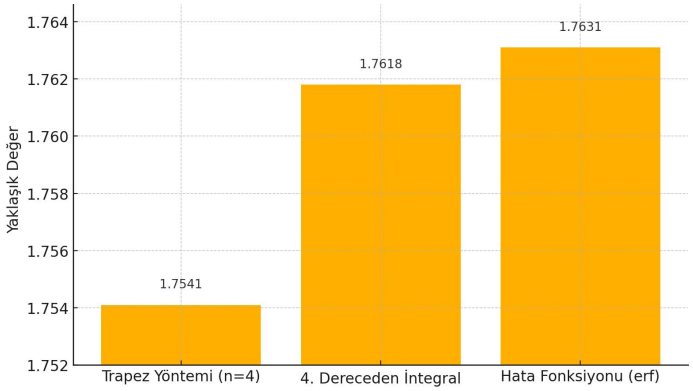
dir. Şimdi hata fonksiyonu kullanarak $\int_{-2}^{2} e^{-x^2} dx$ integralinin değerini bulalım. Bu integral, Gauss hata fonksiyonu (error function) ile ilişkilidir.

$$\int_{-2}^{2} e^{-x^2} dx = \sqrt{\pi}.\,\mathrm{erf}(2)$$

$$\sqrt{\pi} \approx 1{,}77245$$

$$\text{erf}\,(2) \approx 0{,}99532$$

$$\int_{-2}^{2} e^{-x^2}\text{dx} \approx 1{,}77245\,.\,0{,}99532 \approx 1{,}7631$$



Şekil 4.2: Yaklaşık Değer Karşılaştırma

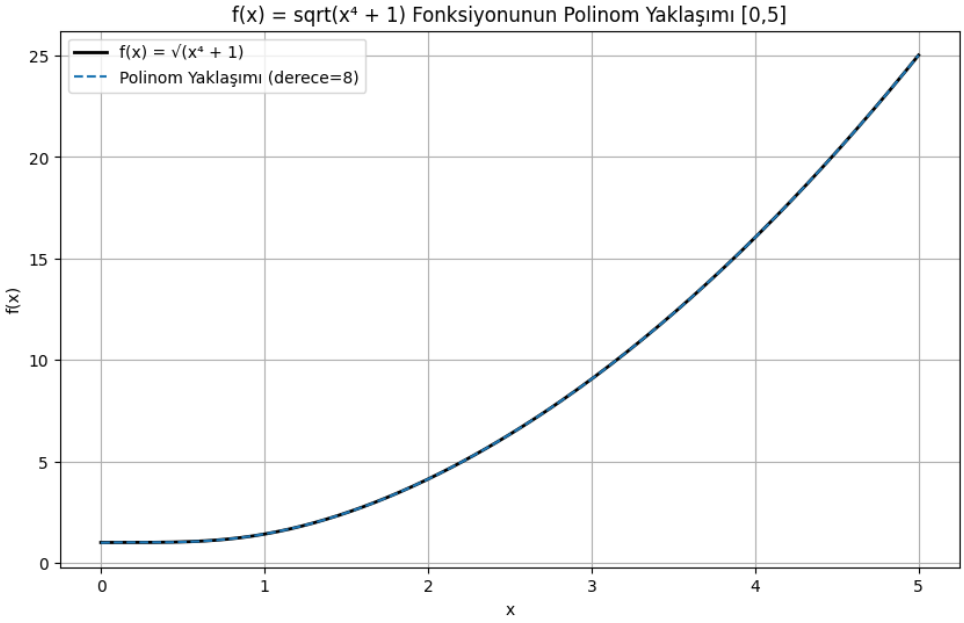
Yukarıdaki grafikte $\int_{-2}^{2} e^{-x^2}\text{dx}$ integraline ilişkin üç farklı yöntemin (Trapez, 4. dereceden integral ve hata fonksiyonu) yaklaşık değerleri karşılaştırmalı olarak görselleştirilmiştir. Görüldüğü gibi, bu yöntemlerin birbirine kadar yakın olduğunu da net bir şekilde gösteriyor.

4.4 Polinom Regresyon Modeli Yardımıyla Belirli Aralıklarda $f(x) = \sqrt{x^4 + 1}$ Fonksiyonunun Polinom Karşılığı

$f(x) = \sqrt{x^4 + 1}$ fonksiyonunu inceleyelim. [0,5] aralığındaki integralinin değerini bulmak için bu fonksiyona daha yakın bir polinom fonksiyonu bulup, bulacağımız fonksiyonun daha kolay bir şekilde integralini alabiliriz. O halde integrali zor alınabilen bir fonksiyonun integralini almak yerine o fonksiyona yakın bir fonksiyon bulup daha kolay bir şekilde integral almayı planlıyoruz. Temel amacımız integral değeri olarak tahminlerimizde yakın bir değere ulaşmak. Bunun için polinom regresyon modeli kullanarak Python programlama dilini kullanacağız. Bu model sayesinde $f(x) = \sqrt{x^4 + 1}$ fonksiyonuna yakın bir polinom fonksiyonu bulacağız. Aşağıda kodlarımız ve elde ettiğimiz yaklaşık fonksiyon yer almaktadır.

```
# 1. Veri Üret
x = np.linspace(0, 5, 200).reshape(-1, 1)
y = np.sqrt(x**4 + 1)
```

```
# 2. Polinom Regresyon (n=8)
degree = 8
model = make_pipeline(PolynomialFeatures(degree), LinearRegression())
model.fit(x, y)
y_pred = model.predict(x)

# 3. Katsayılar
coeffs = model.named_steps['linearregression'].coef_.flatten()
intercept = float(model.named_steps['linearregression'].intercept_)  # Hata buradaydı

# 4. Grafik
plt.figure(figsize=(10,6))
plt.plot(x, y, label='f(x) = √(x⁴ + 1)', color='black', linewidth=2)
plt.plot(x, y_pred, label=f'Polinom Yaklaşımı (derece={degree})', linestyle='--')
plt.title("f(x) = sqrt(x⁴ + 1) Fonksiyonunun Polinom Yaklaşımı [0,5]")
plt.xlabel("x")
plt.ylabel("f(x)")
plt.grid(True)
plt.legend()
plt.show()

# 5. Polinom Fonksiyonu Yazdır
print("Polinom Yaklaşım (n=8):")
terms = [f"{intercept:.6f}"]
for i, c in enumerate(coeffs[1:], start=1):
    if abs(c) > 1e-6:  # çok küçük katsayıları atla
        terms.append(f"{c:+.6f}·x^{i}")
print("P₈(x) ≈ " + " ".join(terms))
```



Şekil 4.3: Polinom Yaklaşım Grafiği

Polinom Yaklaşım (n=8):

$P_8(x) \approx 0{,}986667 + 0{,}304895 \cdot x^1 - 1{,}469446 \cdot x^2 + 2{,}611441 \cdot x^3 - 1{,}352253 \cdot x^4 + 0{,}399455 \cdot x^5 - 0{,}068527 \cdot x^6 + 0{,}006362 \cdot x^7 - 0{,}000247 \cdot x^8$

Elde ettiğimiz 8.dereceden polinomunun integral değeri ile $f(x) = \sqrt{x^4 + 1}$ fonksiyonunun [0,5] integral değerlerini karşılaştıracağız. İlk olarak $f(x) = \sqrt{x^4 + 1}$ fonksiyonunun [0,5] aralığında integralini Trapez Yöntemi ile yaklaşık olarak hesaplayalım.

1. n=10 alarak adım genişliği hesaplanır:
2. $h = \frac{5-0}{10} = 0{,}5$
3. Yaklaşık integral değeri:
4. $\int_0^5 \sqrt{x^4 + 1}\, dx \approx \frac{0{,}5}{2}[f(0) + 2f(0.5) + 2f(1) + \cdots + 2f(4.5) + 2f(5)]$
   $\approx 43{,}010915$

Şimdi $f(x) = \sqrt{x^4 + 1}$ fonksiyonunun [0,5] aralığında integralini Simpson Yöntemi ile yaklaşık olarak hesaplayalım.

$$\int_a^b f(x)\mathrm{dx} \approx \frac{h}{3}\left[f(x_0) + 4\sum_{i=1,3}^{n-1} f(x_i) + 2\sum_{i=2,4}^{n-2} f(x_i) + f(x_n)\right]$$

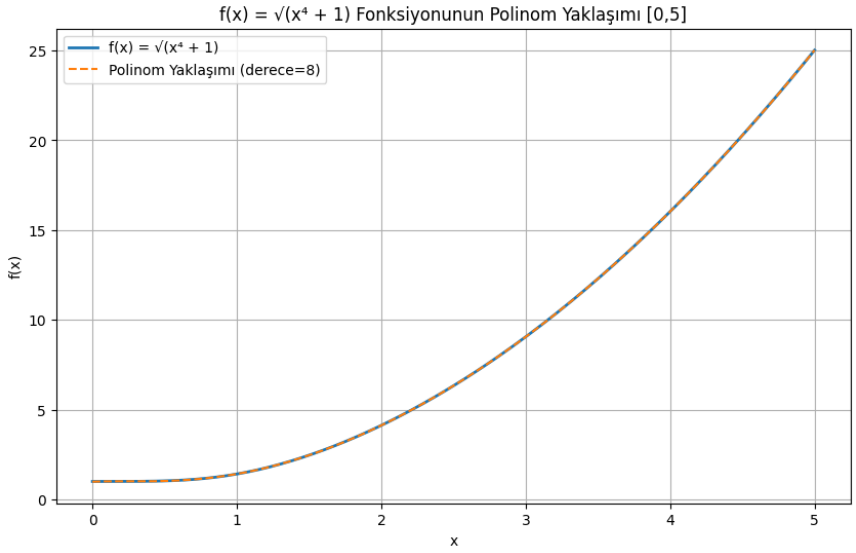
Simpson Yöntemini kullanarak elde ettiğimiz yaklaşık sonuç

$$\approx \frac{0.5}{3}[f(0) + 4f(0{,}5) + f(1{,}5) + \cdots + 2(f(1{,}0) + f(2{,}0) + \cdots) + f(5)]$$
$$\approx 42{,}803247$$

Şimdi $f(x) = \sqrt{x^4 + 1}$ fonksiyonunun [0,5] aralığında integralini polinom regresyon modeli ile analiz edip bu fonksiyona en yakın bir polinom tahmininde bulunacağız. Bu kodlar aşağıdaki gibidir.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline

# 1. Veri Üretimi
x = np.linspace(0, 5, 200).reshape(-1, 1)
y = np.sqrt(x**4 + 1)  # burada bırakıyoruz; düzeltme aşağıda

# 2. Polinom Regresyon (n=8)
degree = 8
model = make_pipeline(PolynomialFeatures(degree), LinearRegression())
model.fit(x, y)

# 3. Katsayıları Çek ve Düzelt
linear_reg = model.named_steps['linearregression']
# Eğer y 2B ise intercept_ bir dizi olur; onu tek float'a indiriyoruz:
intercept = linear_reg.intercept_
if isinstance(intercept, np.ndarray):
    intercept = float(intercept)  # veya intercept = intercept.item()
```

```
# coef_ da (1, n_features) şeklinde gelebilir; bunu 1B vektöre çeviriyoruz:
coeffs = linear_reg.coef_
coeffs = np.ravel(coeffs)

# 4. Tahmin ve Grafik
y_pred = model.predict(x)
plt.figure(figsize=(10, 6))
plt.plot(x, np.sqrt(x**4 + 1), label='f(x) = √(x⁴ + 1)', linewidth=2)
plt.plot(x, y_pred, '--', label=f'Polinom Yaklaşımı (derece={degree})')
plt.title("f(x) = √(x⁴ + 1) Fonksiyonunun Polinom Yaklaşımı [0,5]")
plt.xlabel("x")
plt.ylabel("f(x)")
plt.grid(True)
plt.legend()
plt.show()

# 5. Polinom Fonksiyonu Yazdır
print("Polinom Yaklaşım (n=8):")
terms = [f"{intercept:.6f}"]
for i, c in enumerate(coeffs[1:], start=1):
    if abs(c) > 1e-6:
        terms.append(f"{c:+.6f}·x^{i}")

print("P₈(x) ≈ " + " ".join(terms))
```



Şekil 4.4: Polinom Yaklaşım Grafiği

Polinom Yaklaşım (n=8):

$P_8(x) \approx 0{,}986667 + 0{,}304895 \cdot x^1 - 1{,}469446 \cdot x^2 + 2{,}611441 \cdot x^3 - 1{,}352253 \cdot x^4 + 0{,}399455 \cdot x^5 - 0{,}068527 \cdot x^6 + 0{,}006362 \cdot x^7 - 0{,}000247 \cdot x^8$

Bu $f(x) = \sqrt{x^4 + 1}$ fonksiyonuna polinom regresyon modeli kullanarak elde ettiğimiz yakın 8.dereceden polinom fonksiyonunun integral değerini bulalım.

$\int_0^5 (0{,}986667 + 0{,}304895 \cdot x^1 - 1{,}469446 \cdot x^2 + 2{,}611441 \cdot x^3 - 1{,}352253 \cdot x^4 + 0{,}399455 \cdot x^5 - 0{,}068527 \cdot x^6 + 0{,}006362 \cdot x^7 - 0{,}000247 \cdot x^8) dx$

$= 42{,}27565$

Şimdi $f(x) = \sqrt{x^4 + 1}$ fonksiyonunun [0,5] aralığında integralini Basit Riemann Yöntemi ile yaklaşık olarak hesaplayalım. n=1000 bölme sayısı kullanalım.

$$\int_0^5 \sqrt{x^4 + 1}\,dx \approx \sum_{i=1}^{n} f(x_i).\Delta x$$

$$\Delta x = \frac{5 - 0}{1000} = 0{,}005$$

$x_i = a + i.\Delta x = 0 + i.0.05\ (i = 1,2,\ldots,1000)$

$f(x_i) = \sqrt{x^4 + 1}$

$$\sum_{i=1}^{1000} \sqrt{x^4 + 1}$$

$$\sum_{i=1}^{1000} \sqrt{(x_i)^4 + 1} \text{ x } 0{,}005 \approx 42{,}8628$$

$\int_0^5 \sqrt{x^4 + 1}dx$ integrali, kapalı formda çözülemeyen bir integraldir. Bu nedenle, yüksek hassasiyetli sayısal integrasyon yöntemleri kullanılarak yaklaşık değerler elde edilir. Bu tür integrallerin hesaplanmasında yaygın olarak kullanılan yöntemlerden biri, SciPy kütüphanesinin quad fonksiyonudur. Bu fonksiyon, Gauss-Kronrod yöntemi gibi adaptif kuadratik integrasyon tekniklerini kullanarak yüksek doğruluklu sonuçlar üretir. Python'da aşağıdaki kod parçası kullanılarak bu integralin yaklaşık değeri hesaplanabilir.

```
from scipy.integrate import quad
import numpy as np

f = lambda x: np.sqrt(x**4 + 1)
sonuc, hata = quad(f, 0, 5)
print(sonuc)
```

Bu kod çalıştırıldığında, yaklaşık olarak 42,8027 değeri elde edilir. Bu değer, "Gauss-Kronrod yöntemi" olarak kabul edilen değerdir.

Yöntem	**Yaklaşık Sonuç**	**Gerçeğe Yakınlık**
Trapez Yöntemi	43,010915	~ %0,49 fazla
Simpson Yöntemi	42,803247	~ %0,001 fazla
Polinom Regresyon	42,275650	~ %1,23 eksik
Riemann Yaklaşımı	42,862800	~ %0,14 fazla
Gauss-Kronrod yöntemi	42,802700	– (Referans)

Bu çalışmada, $f(x) = \sqrt{x^4 + 1}$ fonksiyonunun [0, 5] aralığında belirli integrali, farklı sayısal yöntemler kullanılarak yaklaşık olarak hesaplanmıştır. Elde edilen sonuçlar, yüksek hassasiyetli sayısal integrasyon yöntemiyle (gerçek değer) elde edilen referans değere göre

karşılaştırmalı olarak analiz edilmiştir. Gerçek değer yaklaşık olarak 42,8027 olarak kabul edilmiştir. Trapez yöntemi ile elde edilen sonuç 43,010915 olup, gerçek değerden yaklaşık %0,49 oranında daha büyüktür. Bu fark, trapez yöntemi ile hesaplanan alanın, fonksiyonun konveks yapısı nedeniyle üstten yaklaşmasından kaynaklanmaktadır. Simpson yöntemi ile hesaplanan değer ise 42,803247 olup, gerçek değere yalnızca %0.001'lik bir sapma ile oldukça yakın bir sonuç üretmiştir. Bu durum, Simpson yönteminin parabolik yaklaşımı sayesinde daha yüksek doğruluk sağlamasından kaynaklanmaktadır. Polinom regresyon yöntemi kullanılarak elde edilen yaklaşık değer 42,275650 olup, gerçek değerden yaklaşık %1,23 oranında düşük kalmıştır. Bu fark, kullanılan regresyon modelinin veri dağılımını yeterince temsil edememesi veya düşük dereceli polinomun fonksiyonun yapısını tam olarak yakalayamamasından kaynaklanabilir. Diğer taraftan, basit Riemann toplamı ile sağ uç noktalar kullanılarak hesaplanan değer 42,862800 olup, gerçek değerin yaklaşık %0,14 üzerinde kalmıştır. Bu sapma, yöntemin temel doğası gereği her bir alt aralıkta fonksiyonun daha büyük olan sağ uç değerini esas alarak toplamı oluşturmasından ileri gelmektedir.

Sonuç olarak, incelenen yöntemler arasında Simpson yöntemi, doğruluğu açısından en güvenilir sonucu vermiştir. Polinom regresyon yaklaşımı ise, belirli ayarlamalar yapılmadığı takdirde bu tür fonksiyonlarda ciddi sapmalara neden olabilmektedir. Sayısal integrasyon gerektiren benzer problemlerde, yöntem seçimi fonksiyonun yapısına, istenen doğruluk seviyesine ve işlem maliyetine bağlı olarak dikkatli bir şekilde yapılmalıdır.

4.5 Lineer Regresyon Modeli Yardımıyla Belirli Aralıklarda $f(x) = cosx$ Fonksiyonunun Polinom Karşılığı

$f(x) = cosx$ fonksiyonunu inceleyelim. İntegralinin bazı rastgele alacağımız noktalardaki değerlerini bulmak için bu fonksiyona daha yakın bir polinom fonksiyonu bulup, bulacağımız fonksiyonun daha kolay bir şekilde integralini alabiliriz. O halde integrali zor alınabilen bir fonksiyonun integralini almak yerine o fonksiyona yakın bir fonksiyon bulup daha kolay bir şekilde integral almayı planlıyoruz. Temel amacımız integral değeri olarak tahminlerimizde yakın bir değere ulaşmak. Bunun için polinom regresyon modeli kullanarak Python programlama dilini kullanacağız. Bu model sayesinde $f(x) = cosx$ fonksiyonuna yakın bir polinom fonksiyonu bulacağız. Aşağıda kodlarımız ve elde ettiğimiz yaklaşık fonksiyon yer almaktadır.

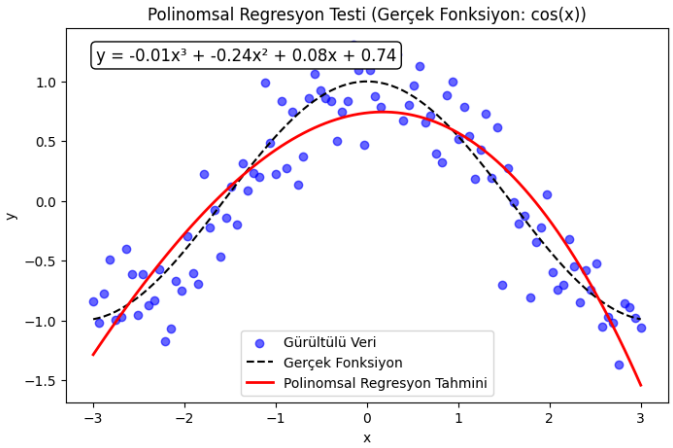
```
def polynomial_regression_test():
    np.random.seed(42)
```

```
x = np.linspace(-3, 3, 100).reshape(-1, 1)
# Gerçek fonksiyon: y = cos(x)

true_y = np.cos(x)

# Gürültü eklenmiş veri (cos(x) için küçük bir gürültü)

y = true_y + np.random.normal(0, 0.3, size=x.shape)

degree = 3
model = make_pipeline(PolynomialFeatures(degree), LinearRegression())
model.fit(x, y)
y_pred = model.predict(x)

lin_reg = model.named_steps['linearregression']
mse = mean_squared_error(true_y, y_pred)
r2 = r2_score(true_y, y_pred)

print("\n=== Polinomsal Regresyon (Derece {}) ===".format(degree))
print("Katsayılar:", lin_reg.coef_)
print("Intercept:", lin_reg.intercept_)
print("MSE: {:.2f}, R2: {:.2f}".format(mse, r2))

# Rastgele 10 örnek x değeri seçimi ve karşılaştırma

x_targets = np.sort(np.random.choice(x.flatten(), 10, replace=False))
print("\n   x      Gerçek y      Tahmin y")
for xt in x_targets:
    true_val = np.cos(xt)
    pred_val = model.predict(np.array([[xt]]))[0][0]
    print("{:5.2f}    {:10.2f}    {:10.2f}".format(xt, true_val, pred_val))

plt.figure(figsize=(8,5))
plt.scatter(x, y, color="blue", alpha=0.6, label="Gürültülü Veri")
```

```
    plt.plot(x, true_y, color="black", linestyle="--", label="Gerçek Fonksiyon")
    plt.plot(x, y_pred, color="red", linewidth=2, label="Polinomsal Regresyon Tahmini")
    plt.xlabel("x")
    plt.ylabel("y")
    plt.title("Polinomsal Regresyon Testi (Gerçek Fonksiyon: cos(x))")
    plt.legend()

    # Modelin tahmin formülü: y = (coef[0] + intercept) + coef[1]*x + coef[2]*x^2 + coef[3]*x^3

    effective_intercept = lin_reg.coef_[0][0] + lin_reg.intercept_[0]
    coef_x   = lin_reg.coef_[0][1]
    coef_x2  = lin_reg.coef_[0][2]
    coef_x3  = lin_reg.coef_[0][3]

    # Denklem metni (katsayılar 2 ondalık basamağa yuvarlanmıştır)

    equation_text = f"y = {coef_x3:.2f}x³ + {coef_x2:.2f}x² + {coef_x:.2f}x + {effective_intercept:.2f}"

    # Denklemi grafiğin üst kısmında görüntüleme

    plt.text(0.05, 0.95, equation_text, transform=plt.gca().transAxes, fontsize=12,
             verticalalignment='top', bbox=dict(boxstyle="round,pad=0.3", fc="white", ec="black", lw=1))

    plt.show()
if __name__ == '__main__': polynomial_regression_test()
```

```
=== Polinomsal Regresyon (Derece 3) ===
Katsayılar: [[ 0.         0.08496434 -0.23899448 -0.0141726 ]]
Intercept: [0.73696941] MSE: 0.04, R2: 0.93
```

x	Gerçek y	Tahmin y
-2.82	-0.95	-1.08
-2.58	-0.84	-0.83
-2.21	-0.60	-0.47
-2.15	-0.55	-0.41
-1.79	-0.22	-0.10
-1.18	0.38	0.33
-0.52	0.87	0.63
-0.09	1.00	0.73
0.39	0.92	0.73
2.76	-0.93	-1.14



Şekil 4.5: Polinom Regresyon Testi

5. MAKİNE ÖĞRENMESİ REGRESYON MODELLERİ KULLANILARAK ALTIN ONS FİYATLARININ TAHMİNİ

Tezimizin bu bölümünde, 20.10.2010 yılından 20.10.2025 yılına kadar olan süreçteki günlük altın fiyatlarının ons değerleri analiz edilmiştir. Analiz kapsamında, piyasa işlemlerinin gerçekleştiği hafta içi günlerine ait veriler tercih edilmiştir. Orijinal veri setinde gün içerisinde gerçekleşen en yüksek ve en düşük altın fiyatlarının aritmetik ortalaması, geometrik ortalaması, logaritmik ortalaması ve tipik ortalaması alınarak, analiz için sadeleştirilmiş bir veri seti oluşturulmuştur. Elde edilen bu sadeleştirilmiş veriler, çalışmanın tekrar edilebilirliğini sağlamak ve ileride gerçekleştirilecek benzer çalışmalar için kolaylık olması amacıyla "altın_ons_aritmetik.csv" dosyasında kaydedilmiştir.

Bu çalışma kapsamında gerçekleştirilen analizde, veri setine yönelik tahminleme işlemleri için çeşitli makine öğrenmesi algoritmaları uygulanmıştır. Kullanılan algoritmalar arasında Lineer Regresyon Modeli, Polinom Regresyon Modeli, Destek Vektör Makineleri Regresyon Modeli (SVR), Ridge Regresyon Modeli, Lasso Regresyon Modeli, ElasticNet Regresyon Modeli ve Bayes Regresyon Modeli mevcuttur. Python programlama dili aracılığıyla oluşturulan modeller, elde edilen altın fiyat verileri kullanılarak eğitilmiş ve modellerin tahmin performansları karşılaştırmalı olarak incelenmiştir. Bu sayede, algoritmaların veri setindeki eğilimleri yakalama yetkinlikleri ve tahmin doğrulukları değerlendirilmiştir. Aşağıda bu uygulamaya ait Python kodları ve kodların işlenmesi sonucu elde edilen sonuçlar ve grafikler mevcuttur.

İlk olarak altın ons tahimi yapmak için elde ettiğimiz verilerin günlük değerleri baz alınarak işlem yapılmıştır. Yapılan işlemler günlük değer içerisinde yer alan “Şimdi,Açılış, Yüksek ve Düşük” değerlerinin aritmetik ortalaması kullanılarak yapılan çalışmamız yer almaktadır. Bu verilere https://tr.investing.com/currencies/xau-usd-historical-data adresinden ulaşılır.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from datetime import timedelta
from math import sqrt, ceil
```

```
# Ek model importları
from sklearn.linear_model import LinearRegression, Ridge, Lasso, ElasticNet, BayesianRidge
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.svm import SVR
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.compose import TransformedTargetRegressor
from sklearn.model_selection import GridSearchCV

# Stil ayarını kontrol edelim:
if 'seaborn-whitegrid' in plt.style.available:
    plt.style.use('seaborn-whitegrid')
else:
    plt.style.use('default')

# Dosyanızın bulunduğu yolu belirtin:
file_path = 'altın_ons_aritmetik.csv'
df = pd.read_csv(file_path)



# Eğer CSV dosyanızda header yoksa:
# df.columns = ['Tarih', 'Fiyat']

# Tarih sütununu, günün önce geleceği biçimde (dayfirst=True) datetime formatına çevirin
# ve geçersiz tarihleri temizleyin
df['Tarih'] = pd.to_datetime(df['Tarih'], dayfirst=True, errors='coerce')
df = df.dropna(subset=['Tarih'])

# "Fiyat" sütunundaki European formatındaki sayısal veriler
# (örneğin "1.503,60": nokta binlik ayırıcı, virgül ondalık ayırıcı)
# önce nokta karakterleri kaldırılır, sonra virgül nokta ile değiştirilir ve float'a dönüştürülür.
```

```
df['Fiyat'] = df['Fiyat'].astype(str).str.replace('.', '', regex=False).str.replace(',', '.', regex=False)
df['Fiyat'] = pd.to_numeric(df['Fiyat'], errors='coerce')
df = df.dropna(subset=['Fiyat'])

# Tarih değerlerini sayısal değere (ordinal) dönüştürün
df['Tarih_Ordinal'] = df['Tarih'].apply(lambda date: date.toordinal())

# Özellik (X) ve hedef (y) değişkenlerini oluşturun
X = df[['Tarih_Ordinal']].values
y = df['Fiyat'].values

# Gelecek tahmin aralığını 60 gün olarak ayarlayın (grafik gösterimi için)
forecast_period = 60
last_date = df['Tarih'].max()
future_dates = pd.date_range(last_date + pd.Timedelta(days=1), periods=forecast_period)
all_dates = pd.date_range(df['Tarih'].min(), future_dates[-1])
all_dates_ordinal = np.array([d.toordinal() for d in all_dates]).reshape(-1, 1)

# Başlangıç için polinomial regresyonda kullanılacak başlangıç derecesi
initial_poly_degree = 3



# Kullanılacak regresyon modelleri (8 model)
# Burada Polynomial Regression modelini Ridge ile genişlettik.
models = {
    'Linear Regression': Pipeline([
        ('scaler', StandardScaler()),
        ('lr', LinearRegression())
    ]),
    'Polynomial Regression': TransformedTargetRegressor(
        regressor=Pipeline([
            ('scaler', StandardScaler()),
('poly', PolynomialFeatures()),   # Derece burada parametre olarak ayarlanacak.
```

```
('ridge', Ridge())                # Ridge regularizasyonu ekleniyor.
        ]),
Func=np.log,          # Hedef değişkenin logaritmasını alıyoruz.
Inverse_func=np.exp   # Tahmin sonrası üstel dönüşüm uyguluyoruz.
    ),
    'SVR': Pipeline([
        ('scaler', StandardScaler()),
        ('svr', SVR(kernel='rbf'))
    ]),
    'Ridge Regression': Pipeline([
        ('scaler', StandardScaler()),
        ('ridge', Ridge())
    ]),
    'Lasso Regression': Pipeline([
        ('scaler', StandardScaler()),
        ('lasso', Lasso())
    ]),
    'ElasticNet Regression': Pipeline([
        ('scaler', StandardScaler()),
        ('elasticnet', ElasticNet(random_state=0))
    ]),
    'Bayesian Ridge': Pipeline([
        ('scaler', StandardScaler()),
        ('bayesian', BayesianRidge())
    ])
}

# Hiperparametre gridlerini tanımlayalım:
param_grids = {
'Linear Regression': {},  # Parametre yok.
    'Polynomial Regression': {
        'regressor__poly__degree': [2, 3, 4],
        'regressor__ridge__alpha': [0.1, 1, 10]
    },
```

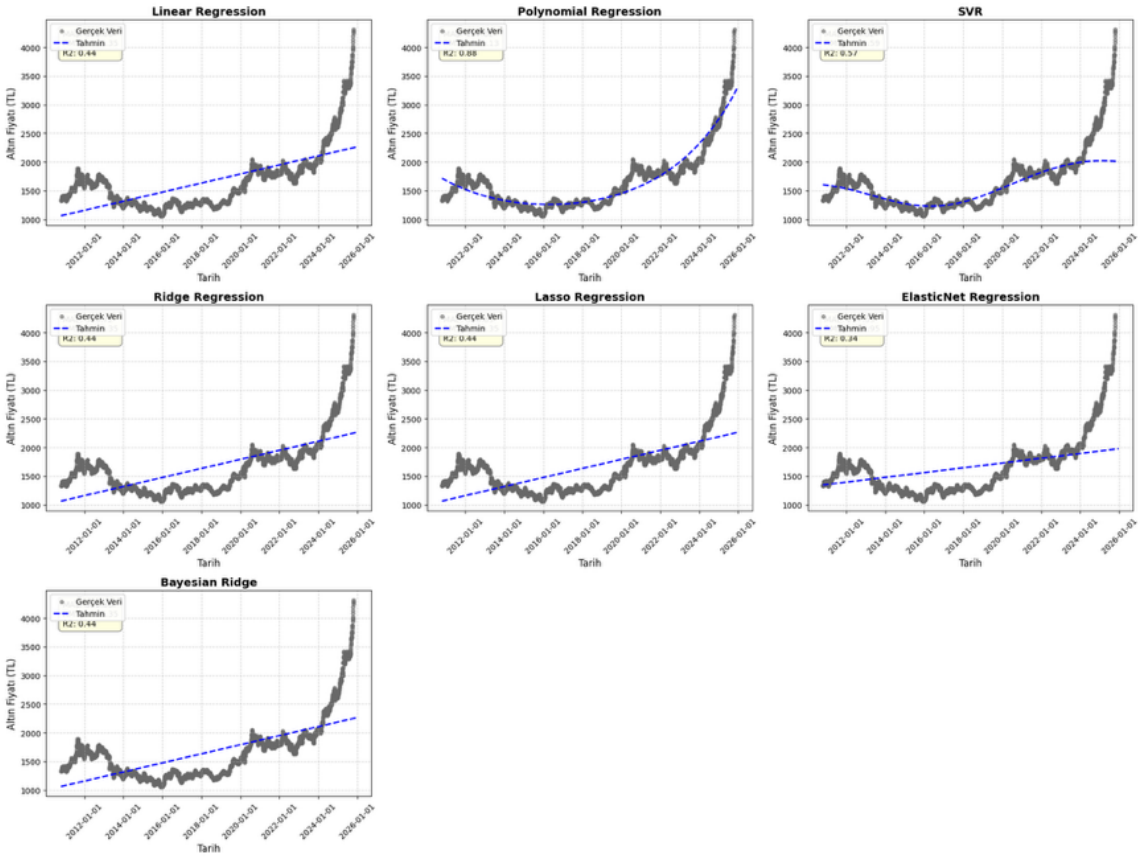
```
    'SVR': {
        'svr__C': [0.1, 1, 10],
        'svr__gamma': ['scale', 'auto']
    },
    'Ridge Regression': {
        'ridge__alpha': [0.1, 1, 10]
    },
    'Lasso Regression': {
'lasso__alpha': [0.001, 0.01, 0.1, 1].
    },
    'ElasticNet Regression': {
'elasticnet__alpha': [0.001, 0.01, 0.1, 1],.
        'elasticnet__l1_ratio': [0.1, 0.5, 0.9]
    },
    'Bayesian Ridge': {
        'bayesian__alpha_1': [1e-6, 1e-5],
        'bayesian__lambda_1': [1e-6, 1e-5]
    }
}

# Hiperparametre optimizasyonu: GridSearchCV ile en iyi parametreleri buluyoruz.
For name, model in models.items():.
    grid = param_grids.get(name, {})
If grid:  # Eğer grid boş değilse.
        gs = GridSearchCV(model, grid, cv=5, n_jobs=-1)
        gs.fit(X, y)
        best_model = gs.best_estimator_
        models[name] = best_model  # En iyi modeli güncelliyoruz
Print(f"{name} - Best Params: {gs.best_params_}").
    else:
        # Grid yoksa direkt eğitiyoruz
        model.fit(X, y)
        models[name] = model
```

```
# Hata metriklerini saklamak için dictionary oluşturun
error_metrics = {'Model': [], 'MAE': [], 'RMSE': [], 'R2': []}

# Grafik düzenini dinamik ayarlamak (3 sütun kullanıyoruz)
n_models = len(models)
cols = 3
rows = ceil(n_models / cols)
plt.figure(figsize=(20, rows * 5))

For i, (name, model) in enumerate(models.items(), 1):.
    # Her modelle yeniden eğitim
    model.fit(X, y)

    # Tüm tarih aralığı (gerçek veriler + gelecek 60 gün) için tahmin
    y_pred_all = model.predict(all_dates_ordinal)
    # Eğitim verisi üzerindeki tahminler (hata hesaplaması için)
    y_train_pred = model.predict(X)

    # Hata metriklerini hesapla
    mae = mean_absolute_error(y, y_train_pred)
    rmse = sqrt(mean_squared_error(y, y_train_pred))
    r2 = r2_score(y, y_train_pred)

    error_metrics['Model'].append(name)
    error_metrics['MAE'].append(mae)
    error_metrics['RMSE'].append(rmse)
    error_metrics['R2'].append(r2)

    ax = plt.subplot(rows, cols, i)
    # Gerçek verileri daha hafif bir renk ve daha küçük marker ile gösterelim
Ax.scatter(df['Tarih'], y, color='dimgray', s=20, alpha=0.6, label='Gerçek Veri', marker='o').
    # Tahmin çizgisini kesikli ve kalın olarak çizelim
    ax.plot(all_dates, y_pred_all, color='blue', lw=2, linestyle='--', label='Tahmin')
    ax.set_title(name, fontsize=14, fontweight='bold')
```

```
    ax.set_xlabel('Tarih', fontsize=12)
    ax.set_ylabel('Altın Fiyatı (TL)', fontsize=12)
    ax.legend(fontsize=10, loc='upper left', framealpha=0.9)
    # Tarih etiketleri için biçimlendirme
    ax.xaxis.set_major_locator(mdates.AutoDateLocator())
    ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
    ax.tick_params(axis='x', rotation=45)
    # Hata metriklerini kutucuk içinde gösterelim
    metrics_text = f"MAE: {mae:.2f}\nRMSE: {rmse:.2f}\nR2: {r2:.2f}"
Ax.text(0.05, 0.95, metrics_text, transform=ax.transAxes, fontsize=10,.
Verticalalignment='top',    bbox=dict(boxstyle="round,pad=0.5",    facecolor='lightyellow',
edgecolor='gray', lw=1)).
    ax.grid(True, linestyle='--', alpha=0.6)

plt.tight_layout()
plt.show()

# Hata metriklerini tablo olarak gösterin
error_df = pd.DataFrame(error_metrics)
Print("Regresyon Model Hata Metrikleri (Eğitim Verisi Üzerinden):").
print(error_df)



# ----------------------------------------------------------------
# Belirtilen gelecek yıllar için tahminler
# (2026, 2027, 2028, 2029, 2030, 2035, 2040, 2045, 2050)
# ----------------------------------------------------------------

# İstenen yılların 1 Ocak tarihlerini oluşturuyoruz
target_years = [2026, 2027, 2028, 2029, 2030, 2035, 2040, 2045, 2050]
target_dates = pd.to_datetime([f"{year}-01-01" for year in target_years])
target_dates_ordinal = np.array([d.toordinal() for d in target_dates]).reshape(-1, 1)
# Her modelin tahminlerini saklamak için dictionary
target_predictions = {}
```

```
For name, model in models.items():.
    target_predictions[name] = model.predict(target_dates_ordinal)
# Tahmin değerlerini tablo şeklinde bir DataFrame'e aktarın
target_predictions_df = pd.DataFrame(target_predictions, index=target_dates)
target_predictions_df.index = target_predictions_df.index.strftime('%Y-%m-%d')
target_predictions_df.index.name = 'Tarih'
Print("\nBelirtilen Gelecek Yıllar İçin Model Tahminleri:").
print(target_predictions_df)
```

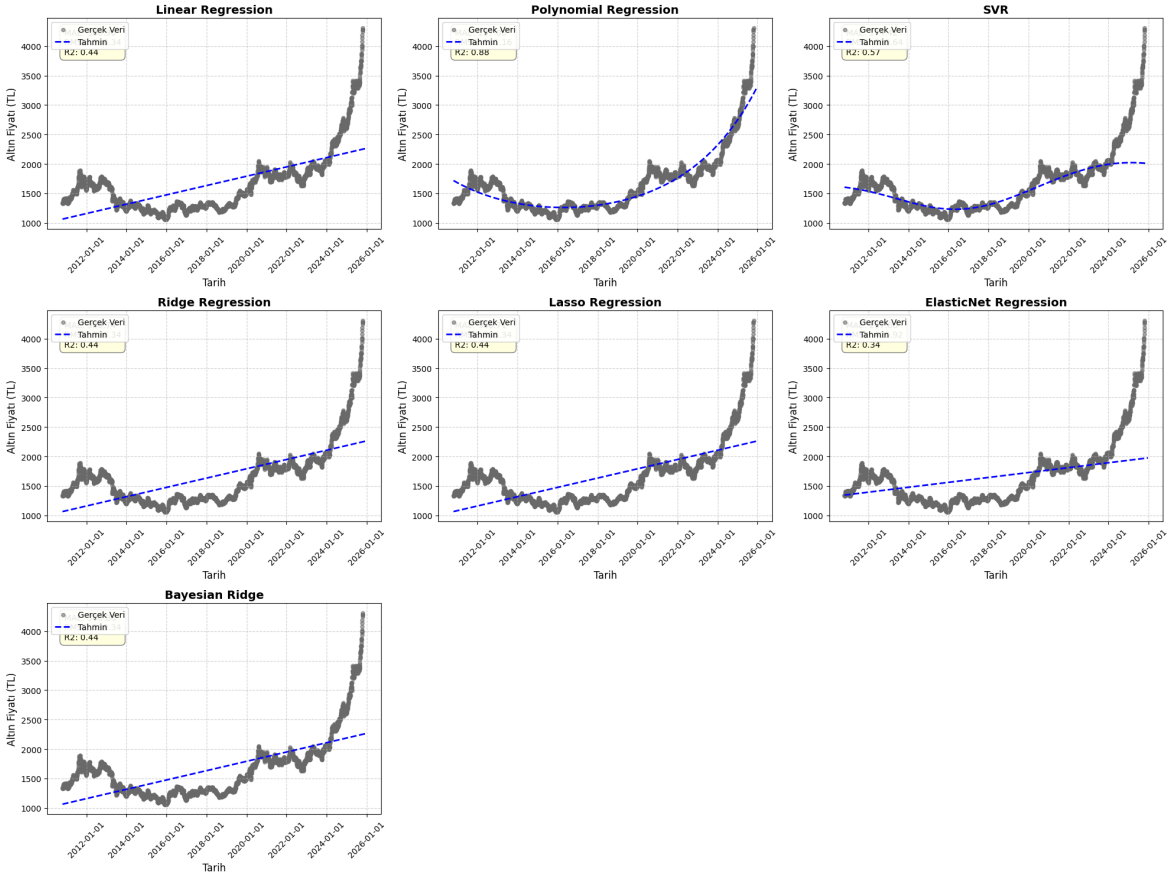
Bu kısımda ise verilerin tipik ortalaması kullanılarak yapılan çalışmamız yer alıyor. Bölümün en başında kullandığımız kodları kullanarak farklı değerler ve farklı grafikleri elde etmiş oluyoruz değişen veri setimizle.



Şekil 4.7: Regresyon Modelleri Karşılaştırma

```
Regresyon Model Hata Metrikleri (Eğitim Verisi Üzerinden):
                  Model         MAE        RMSE        R2
0     Linear Regression  300.885894  388.352376  0.437888
1  Polynomial Regression  137.024750  181.130098  0.877721
2                    SVR  174.319181  339.594875  0.570173
3       Ridge Regression  300.718301  388.353363  0.437885
4       Lasso Regression  300.694530  388.353663  0.437884
5  ElasticNet Regression  297.587665  420.946686  0.339572
6         Bayesian Ridge  300.864249  388.352392  0.437888


Belirtilen Gelecek Yıllar İçin Model Tahminleri:
            Linear Regression  Polynomial Regression          SVR  \
Tarih
2026-01-01        2265.587994           3.320302e+03  2006.458944
2027-01-01        2344.685825           4.094988e+03  1956.996038
2028-01-01        2423.783655           5.155140e+03  1887.742906
2029-01-01        2503.098192           6.629057e+03  1815.032571
2030-01-01        2582.196023           8.695403e+03  1752.343823
2035-01-01        2977.901882           4.598634e+04  1646.938017
2040-01-01        3373.607741           4.064740e+05  1644.709440
2045-01-01        3769.530307           6.014550e+06  1644.706128
2050-01-01        4165.236166           1.485453e+08  1644.706127

            Ridge Regression  Lasso Regression  ElasticNet Regression
\
Tarih
2026-01-01       2264.030092       2263.808584            1976.585559
2027-01-01       2342.925885       2342.675650            2018.203851
2028-01-01       2421.821678       2421.542716            2059.822142
2029-01-01       2500.933623       2500.625856            2101.554457
2030-01-01       2579.829416       2579.492923            2143.172748
2035-01-01       2974.524532       2974.044328            2351.378229
2040-01-01       3369.219649       3368.595733            2559.583710
2045-01-01       3764.130918       3763.363212            2767.903214
2050-01-01       4158.826034       4157.914617            2976.108695

            Bayesian Ridge
Tarih
2026-01-01     2265.387444
2027-01-01     2344.459266
2028-01-01     2423.531088
2029-01-01     2502.819545
2030-01-01     2581.891367
2035-01-01     2977.467112
2040-01-01     3373.042858
2045-01-01     3768.835238
2050-01-01     4164.410983
```

Bu kısımda ise verilerin ağırlıklı ortalaması kullanılarak yapılan çalışmamız yer alıyor. Bölümün en başında kullandığımız kodları kullanarak farklı değerler ve farklı grafikleri elde etmiş oluyoruz değişen veri setimizle.

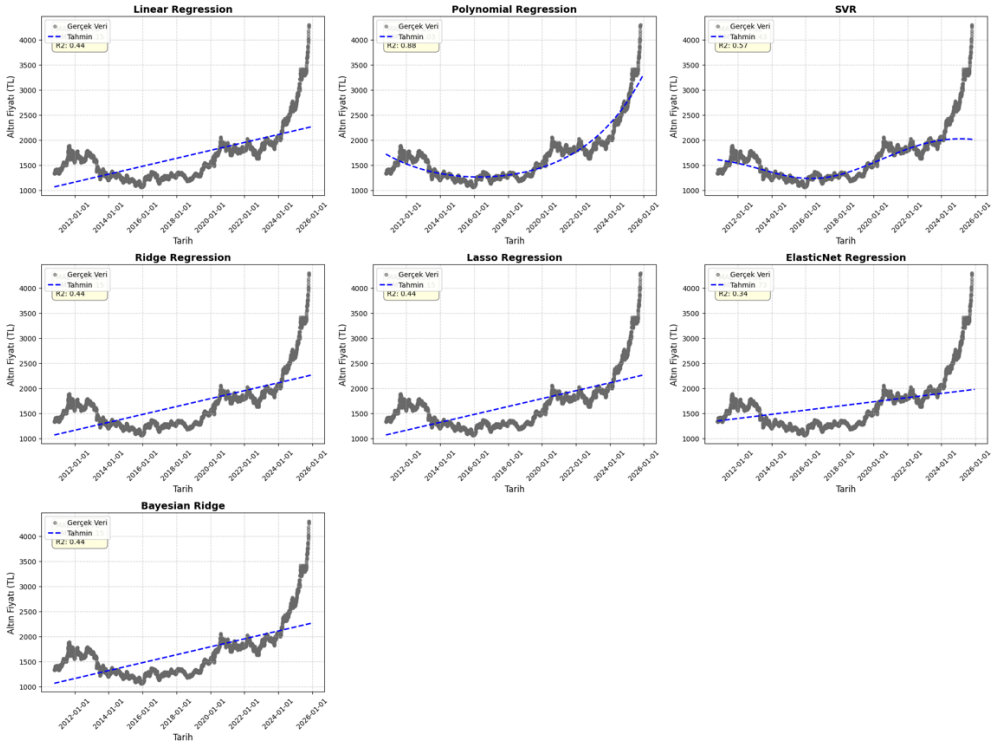


Şekil 4.8: Regresyon Modelleri Karşılaştırma

```
Regresyon Model Hata Metrikleri (Eğitim Verisi Üzerinden):
                   Model         MAE         RMSE        R2
0      Linear Regression  300.902912  388.343141  0.437754
1  Polynomial Regression  137.082012  181.155221  0.877652
2                    SVR  174.416408  339.635327  0.569948
3       Ridge Regression  300.736074  388.344127  0.437752
4       Lasso Regression  300.712348  388.344429  0.437751
5  ElasticNet Regression  297.601325  420.919727  0.339469
6         Bayesian Ridge  300.881390  388.343157  0.437754

Belirtilen Gelecek Yıllar İçin Model Tahminleri:
            Linear Regression  Polynomial Regression          SVR  \
Tarih
2026-01-01        2265.399922           3.320085e+03  2006.222225
2027-01-01        2344.474467           4.094711e+03  1956.772098
2028-01-01        2423.549012           5.154785e+03  1887.506094
2029-01-01        2502.840199           6.628600e+03  1814.778327
2030-01-01        2581.914744           8.694812e+03  1752.077404
2035-01-01        2977.504111           4.598424e+04  1646.663986
2040-01-01        3373.093478           4.064760e+05  1644.435565
2045-01-01        3768.899488           6.015054e+06  1644.432253
2050-01-01        4164.488854           1.485737e+08  1644.432253
```

```
            Ridge Regression  Lasso Regression  ElasticNet Regression
\
Tarih
2026-01-01       2263.842479       2263.620512            1976.482539
2027-01-01       2342.715045       2342.464292            2018.088575
2028-01-01       2421.587612       2421.308073            2059.694611
2029-01-01       2500.676268       2500.367864            2101.414636
2030-01-01       2579.548834       2579.211644            2143.020672
2035-01-01       2974.127756       2973.646557            2351.164842
2040-01-01       3368.706677       3368.081469            2559.309011
2045-01-01       3763.501688       3762.732392            2767.567169
2050-01-01       4158.080609       4157.167305            2975.711338

            Bayesian Ridge
Tarih
2026-01-01     2265.199322
2027-01-01     2344.247852
2028-01-01     2423.296382
2029-01-01     2502.561483
2030-01-01     2581.610013
2035-01-01     2977.069234
2040-01-01     3372.528455
2045-01-01     3768.204247
2050-01-01     4163.663468
```

Bu kısımda ise verilerin logaritmik ortalaması kullanılarak yapılan çalışmamız yer alıyor. Bölümün en başında kullandığımız kodları kullanarak farklı değerler ve farklı grafikleri elde etmiş oluyoruz değişen veri setimizle.



Şekil 4.9: Regresyon Modelleri Karşılaştırma

```
Regresyon Model Hata Metrikleri (Eğitim Verisi Üzerinden):
                   Model         MAE        RMSE        R2
0      Linear Regression  300.784631  388.151270  0.437879
1  Polynomial Regression  137.025024  181.026682  0.877732
2                    SVR  174.317966  339.434943  0.570126
3       Ridge Regression  300.617816  388.152256  0.437876
4       Lasso Regression  300.594135  388.152558  0.437875
5  ElasticNet Regression  297.522946  420.727558  0.339565
6         Bayesian Ridge  300.763132  388.151286  0.437879

Belirtilen Gelecek Yıllar İçin Model Tahminleri:
            Linear Regression  Polynomial Regression          SVR  \
Tarih
2026-01-01        2265.150988           3.319480e+03  2005.951817
2027-01-01        2344.206401           4.093773e+03  1956.394602
2028-01-01        2423.261813           5.153336e+03  1887.082022
2029-01-01        2502.533816           6.626357e+03  1814.354898
2030-01-01        2581.589229           8.691328e+03  1751.677983
2035-01-01        2977.082882           4.594792e+04  1646.335817
2040-01-01        3372.576535           4.059444e+05  1644.108994
2045-01-01        3768.286778           6.003294e+06  1644.105684
2050-01-01        4163.780431           1.481680e+08  1644.105684

            Ridge Regression  Lasso Regression  ElasticNet Regression
\
Tarih
2026-01-01       2263.593922       2263.371578            1976.303487
2027-01-01       2342.447405       2342.196226            2017.899453
2028-01-01       2421.300888       2421.020874            2059.495420
2029-01-01       2500.370408       2500.061480            2101.205348
2030-01-01       2579.223891       2578.886129            2142.801314
2035-01-01       2973.707343       2973.225327            2350.895108
2040-01-01       3368.190795       3367.564526            2558.988901
2045-01-01       3762.890284       3762.119683            2767.196656
2050-01-01       4157.373737       4156.458882            2975.290450

            Bayesian Ridge
Tarih
2026-01-01     2264.950538
2027-01-01     2343.979955
2028-01-01     2423.009372
2029-01-01     2502.255308
2030-01-01     2581.284725
2035-01-01     2976.648329
2040-01-01     3372.011933
2045-01-01     3767.592056
2050-01-01     4162.955660
```

Yukarıda yer alan çalışmalarda görüleceği gibi dört farklı ortalama türüyle (aritmetik, tipik, ağırlıklı ve logaritmik) altın ons tahminleri elde edilmiştir. Burada elde edilen altın ons fiyatı tahmin sonuçları incelendiğinde, modellerin genel performanslarının oldukça benzer olduğu, ancak Polynomial Regression modelinin tüm ortalama türlerinde açık ara en düşük hata değerlerine (MAE ≈ 137, RMSE ≈ 181, $R^2 \approx 0.878$) sahip olduğu görülmektedir. Bu durum,

polinom modelin verinin doğrusal olmayan yapısını en iyi yakalayan regresyon modeli olduğunu göstermektedir. Buna karşın, Lineer, Ridge, Lasso ve Bayesian Ridge regresyonlarının hata değerleri birbirine çok yakın olup (MAE ≈ 300, RMSE ≈ 388, R^2 ≈ 0.438), bu modellerin doğrusal varsayımlarının altın fiyatı gibi volatilitesi yüksek, trendli bir seriyi temsil etmede yetersiz kaldığı anlaşılmaktadır. ElasticNet Regression modeli ise daha yüksek RMSE ve düşük R^2 değeri (≈ 0.34) ile tahmin doğruluğu bakımından en zayıf sonuçları vermiştir. Geleceğe yönelik tahminlerde, Polynomial Regression modelinin 2030 yılı sonrasında aşırı büyüme göstermesi (örneğin 2050 için 1.48×10^8 USD gibi gerçek dışı bir tahmin üretmesi), modelin aşırı öğrenmeye (overfitting) yatkın olduğunu ve uzun dönem projeksiyonlarda güvenilirliğinin azaldığını ortaya koymaktadır. Buna karşın, Lineer, Ridge, Lasso ve Bayesian Ridge modelleri benzer ve istikrarlı eğilimler sergilemiş, 2050 yılı için 4150–4165 USD civarında yakınsayan değerler üretmiştir; bu modellerin uzun vadeli öngörüler açısından daha tutarlı olduğu söylenebilir. Ortalama türleri karşılaştırıldığında ise, dört yöntemin (aritmetik, tipik, ağırlıklı ve logaritmik) hata metrikleri arasında anlamlı bir fark gözlenmemekte, dolayısıyla verinin doğası gereği ortalama türünün model performansına belirgin bir katkı sağlamadığı sonucuna varılmaktadır. Bununla birlikte, logaritmik ortalama verisinde Polynomial modelin R^2 değerinin 0.8777 ile diğerlerine çok yakın kalması, logaritmik dönüşümün modeldeki varyansın azaltılmasında etkili olduğunu göstermektedir. Sonuç olarak, kısa dönemli ve doğrusal olmayan tahminlerde Polynomial Regression modeli en yüksek doğruluk performansını sergilerken, uzun dönemli ve kararlı projeksiyonlar için Ridge, Lasso veya Bayesian Ridge gibi düzenlileştirilmiş lineer modellerin tercih edilmesi metodolojik olarak daha rasyonel görünmektedir.

6. SONUÇ VE ÖNERİLER

Bu tez çalışmasında, makine öğrenmesi teknikleri incelenmiş; özellikle regresyon tabanlı algoritmalar kullanılarak hem matematiksel modelleme uygulamaları hem de finansal bir öngörü problemi olan altın ons fiyatlarının tahmini gerçekleştirilmiştir. Çalışma iki temel eksende ilerlemiştir: matematiksel fonksiyonların belirli aralıklarda polinom yaklaşımlarla modellenmesi ve finansal zaman serilerinin makine öğrenmesi regresyon modelleri ile tahmini.

Denetimli öğrenme kapsamında basit doğrusal regresyon, çoklu doğrusal regresyon, polinom regresyon, SVR, karar ağacı, rastgele orman, Lasso, Ridge ve Elastic Net regresyon modelleri teorik ve uygulamalı olarak incelenmiştir.

Regresyon modelleri ile oluşturulan polinomlar, belirli aralıklarda $f(x) = e^{-x^2}$ fonksiyonu, $f(x) = \sqrt{x^4 + 1}$ fonksiyonu ve $f(x) = cosx$ fonksiyonu gibi fonksiyonların yaklaşık çözümlerini üretmede başarı göstermiştir. Özellikle polinom regresyon modellerinin, fonksiyonların eğrisel yapısına daha uygun sonuçlar verdiği gözlemlenmiştir.
Lineer regresyon ile elde edilen polinomlar kullanılarak, integral ve alan hesaplamalarına yönelik yaklaşımlar sunulmuş; bu yaklaşımın klasik sayısal yöntemlere (örneğin Trapez yöntemi) göre avantajları tartışılmıştır.

2010–2025 yılları arasındaki günlük altın ons fiyatları kullanılarak oluşturulan veri seti üzerinde çeşitli regresyon algoritmaları uygulanmıştır. SVR, Random Forest ve Elastic Net modelleri, özellikle zaman serisi verisinin karşısında daha düşük hata oranları (MAE, RMSE) ve daha yüksek R^2 değerleri sunmuştur. Ridge ve Lasso gibi regülerleştirme temelli modellerin, veri setindeki çoklu doğrusal bağlantı (multicollinearity) problemlerine karşı dayanıklı sonuçlar ürettiği gözlemlenmiştir. Verilere uygulanan farklı ortalama yöntemleri (aritmetik, ağırlıklı, logaritmik vb.) model başarısını doğrudan etkilemiş; özellikle logaritmik ortalamaya dayalı veri seti ile SVR ve Elastic Net modelinde daha yüksek doğruluk elde edilmiştir.

Model Geliştirme: Gelecek çalışmalarda daha karmaşık ve derin öğrenme tabanlı modeller (örneğin LSTM, GRU gibi RNN tabanlı yapılar) ile altın fiyat tahmini yapılması, zaman bağımlılığını daha iyi yakalayabilir.

Veri Genişletme: Finansal veriler üzerinde çalışırken yalnızca altın fiyatı değil; döviz kurları, enflasyon oranları, merkez bankası kararları gibi makroekonomik göstergeler de modele dahil edilerek tahmin performansı artırılabilir.

Model Performans Analizi: Sadece MAE, RMSE ve R^2 değil, MAPE, RMSLE gibi farklı hata ölçütlerinin de kullanılması model değerlendirmesini daha kapsamlı hale getirecektir.

Fonksiyonel Yaklaşımlar: Regresyon modelleriyle yapılan polinomlaştırma işlemleri farklı matematiksel fonksiyon türlerine (trigonometrik, üstel, logaritmik, parametrik) uygulanarak sayısal analizde alternatif bir araç olarak kullanılabilir.

Uygulama Genişletme: Bu çalışmada kullanılan modeller farklı alanlara (örneğin enerji tüketimi tahmini, nüfus projeksiyonu, iklim verisi analizi) uyarlanarak disiplinlerarası katkılar sağlanabilir.

Eğitim ve Öğretim: Bu tür uygulamalı çalışmalar, lisansüstü düzeyde makine öğrenmesi eğitimi için örnek vaka çalışması niteliği taşımaktadır. Matematik ve veri bilimi arasında bir köprü kurularak öğrencilerin uygulama becerileri geliştirilebilir.

7. KAYNAKLAR

Edalı, M. (2020). Performance analysis of set partitioning formulations on the rule extraction from random forests. *Pamukkale University Journal of Engineering Sciences*, doi: 10.5505/pajes.2020.05926

Mesarić, J. and Šebalj, D. (2016). Decision trees for predicting the academic success of students.

Rani, V., Nabi, S. T., Kumar, M., Mittal, A., and Kumar, K. (2023). Self-supervised learning: A succinct review. *Archives of Computational Methods in Engineering*.

Mahaboob, B., Praveen, J. P., Appa Rao, B. V., Harnath, Y., Narayana, C., and Prakash, G. B. (2020). A study on multiple linear regression using matrix calculus. *Advances in Mathematics: Scientific Journal*, 9(7), 4863–4872. https://doi.org/10.37418/amsj.9.7.52

Biau, G. (2012). Analysis of a random forests model. *Journal of Machine Learning Research*, 13, 1063–1095

Fritz.ai. (2023). *Understanding the mathematics behind decision trees*. Erişim adresi: https://fritz.ai/the-mathematics-behind-decision-trees/ (Erişim tarihi: 30 Mayıs 2025)

Parashar, P. (2020). *Decision tree regression and its mathematical implementation*. Erişim adresi https://medium.com/swlh/decision-tree-regression-and-its-mathematical-implementation-58c6e9c5e88e (Erişim tarihi: 30 Mayıs 2025)

Schott, M. (2019). *Random forest algorithm for machine learning*. Capital One Tech. Erişim adresi: https://medium.com/capital-one-tech/random-forest-algorithm-for-machine-learning-c4b2c8cc9feb (Erişim tarihi: 30 Mayıs 2025)

Lichtenberg, J. M. and Şimşek, Ö. (2016). Simple regression models. *Proceedings of Machine Learning Research*, 58, 13–25

Yacoub, M. H., Ismail, S. M., Saida, L. A., Madian, A. H., and Radwan, A. G. (2024). Support vector machine reconfigurable hardware implementation on FPGA. *Franklin Open*, 7, 100115. https://www.sciencedirect.com/journal/franklin-open

Candan, H. (2021). Adım adım makine öğrenmesi bölüm 3: Denetimsiz öğrenme nedir? *Machine Learning Türkiye*. Erişim adresi: https://medium.com/machine-learning-türkiye/adım-adım-makine-öğrenmesi-bölüm-3-denetimsiz-öğrenme-nedir-f890ada49a40 (Erişim tarihi: 30 Mayıs 2025)

KAYNAKLAR DİZİNİ (DEVAM)

Anshul. (2021). *An introduction to random forest algorithm for beginners*. Analytics Vidhya. Erişim adresi: https://www.analyticsvidhya.com/blog/2021/10/an-introduction-to-random-forest-algorithm-for-beginners/ (Erişim tarihi: 30 Mayıs 2025)

GeeksforGeeks. (2025). *Support Vector Machine (SVM) Algorithm*. Erişim adresi: https://www.geeksforgeeks.org/machine-learning/support-vector-machine-algorithm/ (Erişim tarihi: 30 Mayıs 2025)

Sinha, P. (2013). Multivariate polynomial regression in data mining: Methodology, problems and solutions. *International Journal of Scientific and Engineering Research*, (Aralık). https://www.researchgate.net/publication/264425037 (Erişim tarihi: 30 Mayıs 2025)

Arslankaya, S. ve Toprak, Ş. (2021). Makine öğrenmesi ve derin öğrenme algoritmalarını kullanarak hisse senedi fiyat tahmini. *Uluslararası Mühendislik Araştırma ve Geliştirme Dergisi (UMAGD)*, 13(1), 178–192. https://doi.org/10.29137/umagd.771671

Maulud, D. H. ve Abdulazeez, A. M. (2020). A review on linear regression comprehensive in machine learning. *Journal of Applied Science and Technology Trends*, 1(4), 140–147

Piri, M. (2023). Review of regression algorithms. *Konferans Bildirisi*, [çevrimiçi]. https://www.researchgate.net/publication/373514743 (Erişim tarihi: 30 Mayıs 2025)

Salama, M. K. (2024). Optimization of regression models using machine learning: A comprehensive study with Scikit-learn. *International Uni-Scientific Research Journal*, (Eylül). https://doi.org/10.59271/s45500.024.0624.16

Kumar, S. and Bhatnagar, V. (2022). A review of regression models in machine learning.

LightsOnData. (2022). *The history of machine learning*. Erişim adresi: https://www.lightsondata.com/the-history-of-machine-learning/ (Erişim tarihi: 30 Mayıs 2025)

Çelik, E. (2025). Gerçek hayatta integral ve uygulamaları. *Uşak Üniversitesi Fen ve Doğa Bilimleri Dergisi (Usak University Journal of Science and Natural Sciences)*, 1, 92–110. https://doi.org/10.47137/usufedbid.1624019)

Orive, R., Pejčev, A. V., Spalević, M. M., ve Mihić, L. (2022). On the Gauss-Kronrod quadrature formula for a modified weight function of Chebyshev type. *Numerical Algorithms*. https://doi.org/10.1007/s11075-022-01325-8

Çelti, C. (2022). Makine Öğrenmesi Yöntemleri ve Bazı Uygulamaları, Yüksek Lisans Tezi, Haliç Üniversitesi, Lisansüstü Eğitim Enstitüsü, 93

8. ÖZGEÇMİŞ

Kişisel Bilgiler

Adı Soyadı : Halil Sezgin CAN

Doğum tarihi ve yeri :

e-posta :

Öğrenim Bilgileri

Derece	**Okul/Program**	**Yıl**
Y. Lisans	Çankaya Üniversitesi / Matematik-Bilgisayar	2018
Lisans	Uşak Üniversitesi /Matematik	2013
Lise	Kalaba Lisesi	2008

BİLDİRİLER

Can, H.S. (2025). "Makine Öğrenmesi Regresyon Modelleri Kullanılarak Altın Ons Tahmini", International Artificial Intelligence and Data Science Congress (ICADA 2025)