Bu kitaba sığmayan daha neler var!

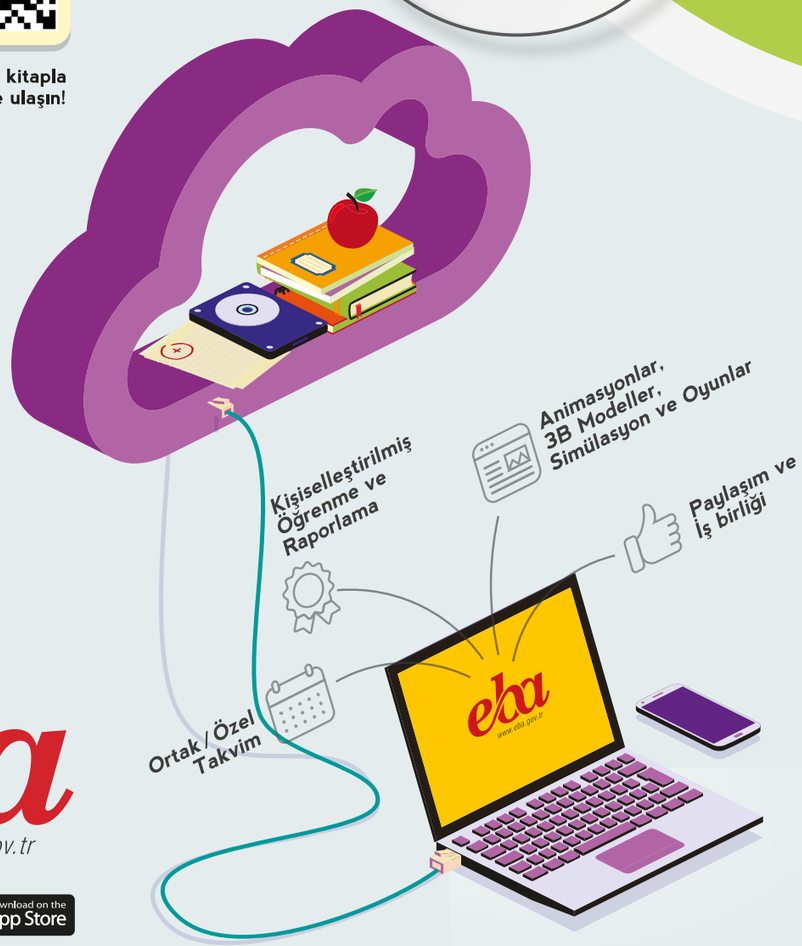
Karekodu okutun, bu kitapla ilgili EBA içeriklerine ulaşın!

ÖDS

ÖĞRENCİ/ÖĞRETMEN DESTEK SİSTEMİ

https://ods.eba.gov.tr

- Konu Anlatımlı Ders Videoları
- Soru Çözüm Videoları
- Ders Anlatım Videoları
- Çoktan Seçmeli Sorular



eba
www.eba.gov.tr

Google play | App Store

BU DERS KİTABI MİLLÎ EĞİTİM BAKANLIĞINCA ÜCRETSİZ OLARAK VERİLMİŞTİR.
PARA İLE SATILAMAZ.

ISBN: 978-975-11-7116-0

Bandrol Uygulamasına İlişkin Usul ve Esaslar Hakkında Yönetmelik'in 5'inci Maddesinin İkinci Fıkrası Çerçevesinde Bandrol Taşıması Zorunlu Değildir.

BİLİŞİM TEKNOLOJİLERİ ALANI

YAPAY ZEKÂ VE MAKİNE ÖĞRENMESİ

11-12

DERS MATERYALİ

MESLEKİ VE TEKNİK ANADOLU LİSESİ

BİLİŞİM TEKNOLOJİLERİ ALANI

11-12 DERS MATERYALİ

YAPAY ZEKÂ VE MAKİNE ÖĞRENMESİ

MESLEKİ VE TEKNİK
ANADOLU LİSESİ

BİLİŞİM TEKNOLOJİLERİ ALANI

YAPAY ZEKÂ VE MAKİNE ÖĞRENMESİ

11-12

DERS MATERYALİ

YAZARLAR

Dr. Murat ALTUN

Mustafa NACAR

Onur ÇAKAR

MİLLÎ EĞİTİM BAKANLIĞI YAYINLARI .....: 8350
YARDIMCI VE KAYNAK KİTAPLAR DİZİSİ .....: 2242

HAZIRLAYANLAR

Dil Uzmanı
Osman Nuri GÜVEN

Program Geliştirme Uzmanı
Ergül SIRKINTI

Ölçme ve Değerlendirme Uzmanı
Günay DURUCAN

Rehberlik Uzmanı
Gülşen YALIN

Görsel Tasarım Uzmanı
Tuğba SANCI

ISBN: 978-975-11-7116-0

Millî Eğitim Bakanlığının 24.12.2020 gün ve 18433886 sayılı oluru ile Meslekî ve Teknik Eğitim Genel Müdürlüğünce ders materyali olarak hazırlanmıştır.

İSTİKLÂL MARŞI

Korkma, sönmez bu şafaklarda yüzen al sancak;
Sönmeden yurdumun üstünde tüten en son ocak.
O benim milletimin yıldızıdır, parlayacak;
O benimdir, o benim milletimindir ancak.

Çatma, kurban olayım, çehreni ey nazlı hilâl!
Kahraman ırkıma bir gül! Ne bu şiddet, bu celâl?
Sana olmaz dökülen kanlarımız sonra helâl.
Hakkıdır Hakk'a tapan milletimin istiklâl.

Ben ezelden beridir hür yaşadım, hür yaşarım.
Hangi çılgın bana zincir vuracakmış? Şaşarım!
Kükremiş sel gibiyim, bendimi çiğner, aşarım.
Yırtarım dağları, enginlere sığmam, taşarım.

Garbın âfâkını sarmışsa çelik zırhlı duvar,
Benim iman dolu göğsüm gibi serhaddim var.
Ulusun, korkma! Nasıl böyle bir imanı boğar,
Medeniyyet dediğin tek dişi kalmış canavar?

Arkadaş, yurduma alçakları uğratma sakın;
Siper et gövdeni, dursun bu hayâsızca akın.
Doğacaktır sana va'dettiği günler Hakk'ın;
Kim bilir, belki yarın, belki yarından da yakın.

Bastığın yerleri toprak diyerek geçme, tanı:
Düşün altındaki binlerce kefensiz yatanı.
Sen şehit oğlusun, incitme, yazıktır, atanı:
Verme, dünyaları alsan da bu cennet vatanı.

Kim bu cennet vatanın uğruna olmaz ki feda?
Şüheda fışkıracak toprağı sıksan, şüheda!
Cânı, cânânı, bütün varımı alsın da Huda,
Etmesin tek vatanımdan beni dünyada cüda.

Ruhumun senden İlâhî, şudur ancak emeli:
Değmesin mabedimin göğsüne nâmahrem eli.
Bu ezanlar -ki şehadetleri dinin temeli-
Ebedî yurdumun üstünde benim inlemeli.

O zaman vecd ile bin secde eder -varsa- taşım,
Her cerîhamdan İlâhî, boşanıp kanlı yaşım,
Fışkırır ruh-ı mücerret gibi yerden na'şım;
O zaman yükselerek arşa değer belki başım.

Dalgalan sen de şafaklar gibi ey şanlı hilâl!
Olsun artık dökülen kanlarımın hepsi helâl.
Ebediyyen sana yok, ırkıma yok izmihlâl;
Hakkıdır hür yaşamış bayrağımın hürriyyet;
Hakkıdır Hakk'a tapan milletimin istiklâl!

Mehmet Âkif Ersoy

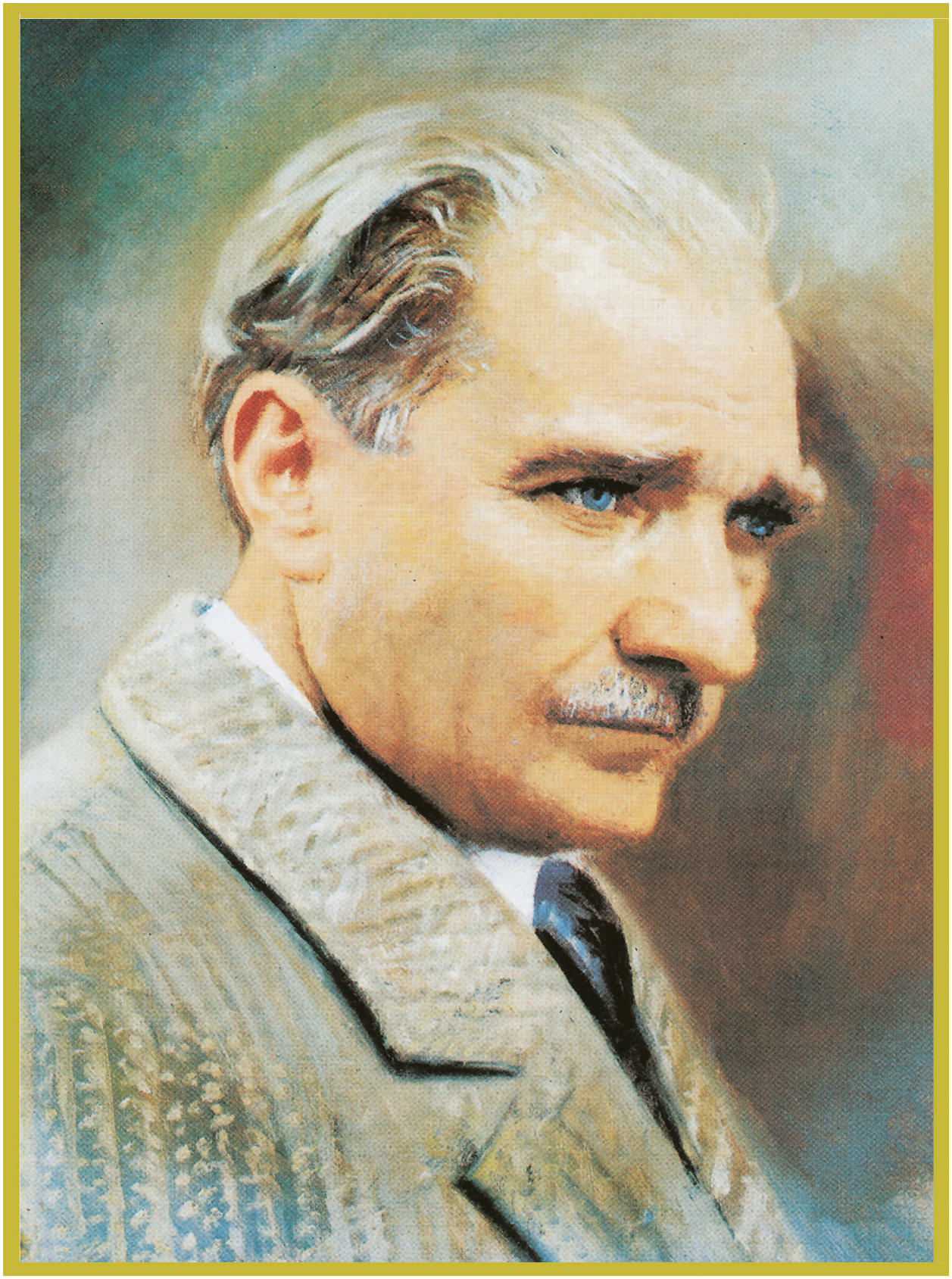
GENÇLİĞE HİTABE

Ey Türk gençliği! Birinci vazifen, Türk istiklâlini, Türk Cumhuriyetini, ilelebet muhafaza ve müdafaa etmektir.

Mevcudiyetinin ve istikbalinin yegâne temeli budur. Bu temel, senin en kıymetli hazinendir. İstikbalde dahi, seni bu hazineden mahrum etmek isteyecek dâhilî ve hâricî bedhahların olacaktır. Bir gün, istiklâl ve cumhuriyeti müdafaa mecburiyetine düşersen, vazifeye atılmak için, içinde bulunacağın vaziyetin imkân ve şeraitini düşünmeyeceksin! Bu imkân ve şerait, çok namüsait bir mahiyette tezahür edebilir. İstiklâl ve cumhuriyetine kastedecek düşmanlar, bütün dünyada emsali görülmemiş bir galibiyetin mümessili olabilirler. Cebren ve hile ile aziz vatanın bütün kaleleri zapt edilmiş, bütün tersanelerine girilmiş, bütün orduları dağıtılmış ve memleketin her köşesi bilfiil işgal edilmiş olabilir. Bütün bu şeraitten daha elîm ve daha vahim olmak üzere, memleketin dâhilinde iktidara sahip olanlar gaflet ve dalâlet ve hattâ hıyanet içinde bulunabilirler. Hattâ bu iktidar sahipleri şahsî menfaatlerini, müstevlîlerin siyasî emelleriyle tevhit edebilirler. Millet, fakr u zaruret içinde harap ve bîtap düşmüş olabilir.

Ey Türk istikbalinin evlâdı! İşte, bu ahval ve şerait içinde dahi vazifen, Türk istiklâl ve cumhuriyetini kurtarmaktır. Muhtaç olduğun kudret, damarlarındaki asil kanda mevcuttur.

Mustafa Kemal Atatürk



MUSTAFA KEMAL ATATÜRK

İÇİNDEKİLER

3. ÖĞRENME BİRİMİ

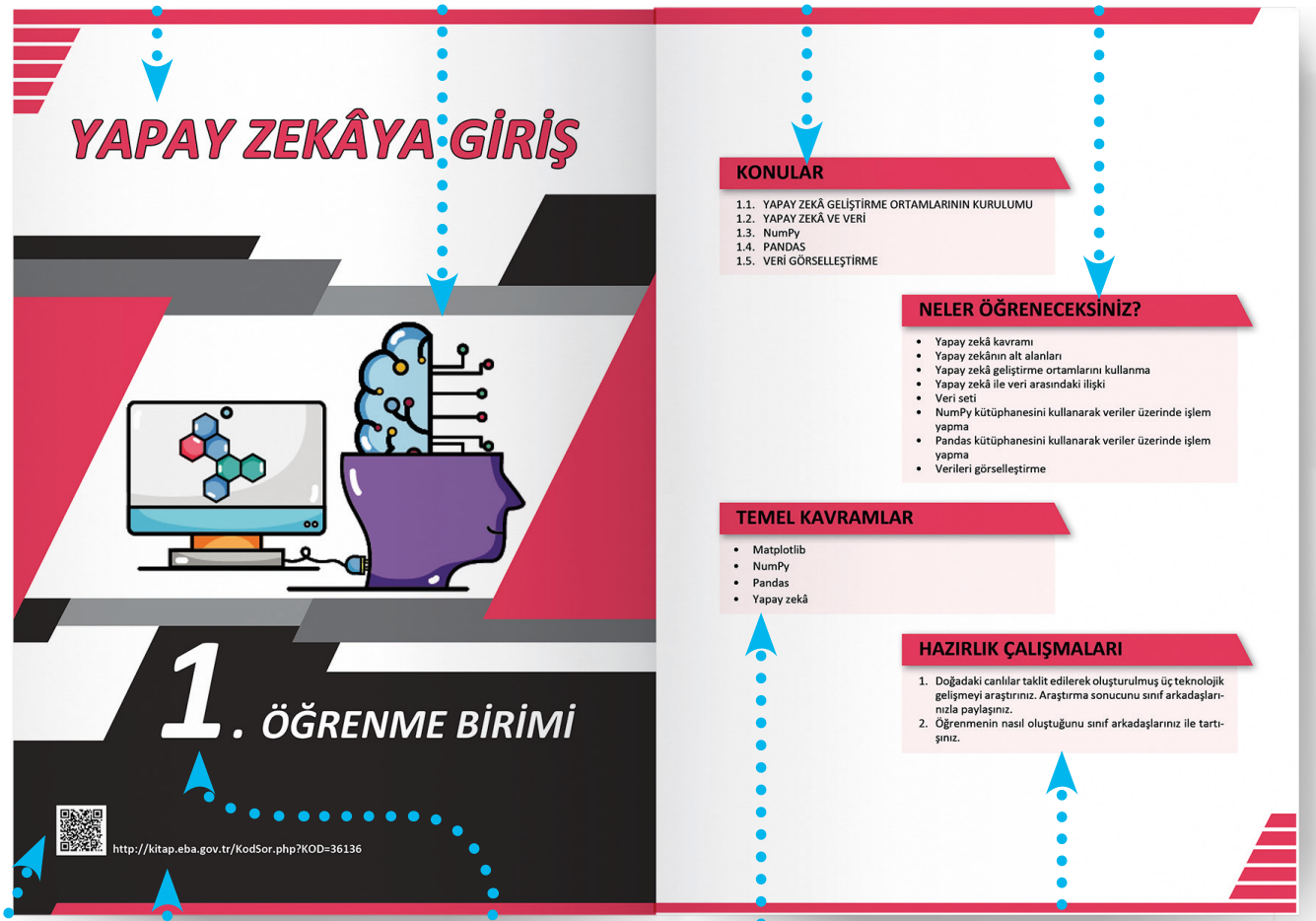
DERS MATERYALİNİN TANITIMI

Öğrenme birimi adını gösterir.

Öğrenme birimi görselini gösterir.

Öğrenme birimi konularını gösterir.

Öğrenme biriminde neler öğrenileceğini gösterir.



Öğrenme birimi linkini gösterir.

Öğrenme birimi numarasını gösterir.

Öğrenme biriminin temel kavramlarını gösterir.

Öğrenme biriminin başında yapılacak ön çalışmaları gösterir.

Öğrenme birimi karekodunu gösterir.

DERS MATERYALİNİN TANITIMI

Konu ile ilgili görselleri gösterir.

Örnek etkinliğini gösterir.

Öğrenme birimi adını ve numarasını gösterir.

Konu başlıklarını gösterir.

Uygulama etkinliğini gösterir.

Sıra sizde etkinliğini gösterir.

1. YAPAY ZEKÂ

Bilgisayar bir makinedir ve insanlar tarafından belirlenen görevleri yerine getirir. Yapay zekâ (Artificial Intelligence - AI) bir makinenin insanlar gibi algılama, öğrenme, akıl yürütme ve problem çözme gibi davranışlar sergilemesidir. Yapay zekâ terimindeki yapay kelimesi insanlar tarafından oluşturulan anlamına gelir. Yapay zekâ, insanın veya doğanın zekâ gerektiren özelliklerine benzerlik gösteren sistemler tasarlamaktır.

1.1. YAPAY ZEKÂ GELİŞTİRME ORTAMLARININ KURULUMU

Yapay zekâ geliştirmek için birden çok yazılım dili kullanılabilir. Bu yazılım dilleri içinden genellikle Python programlama dili kullanılır. Python programlama dili makine öğrenmesi, veri bilimi ve yapay zekâ alanında sağladığı avantajlar sayesinde daha çok tercih edilmektedir. Bu kitapta Python programlama dili kullanılarak yazılımlar geliştirilecektir.

1.1.1. Jupyter Notebook

Jupyter Notebook; açık kaynak kodlu, web tabanlı, etkileşimli bir geliştirme ortamıdır. Jupyter Notebook ile yazılan kodların çıktısı anında görülebilir, veriler görselleştirilebilir ve matematiksel denklemler oluşturulabilir. Python, PHP, R, C# gibi programlama dilleri kullanılarak uygulamalar geliştirilebilir.

Görsel 1.1: Yapay zekâ alt alanları

NOT !!! NumPy kütüphanesi genellikle np takma adı ile import edilir.

Tablo 1.1: Veri Kümesi Örneği

Ad	Maaş	Yaş	Cinsiyet	Şehir
Zeynep	6000	27	Kadın	Ankara
Ali	5000	23	Erkek	İstanbul
Ahmet	5500	24	Erkek	İzmir
Ayşe	6500	28	Kadın	Antalya

1. UYGULAMA

Anaconda kurulumu için aşağıdaki adımları takip ediniz.

1. Adım: Anaconda'nın kurulumu için tarayıcınızda https://www.anaconda.com/products/individual adresini yazınız. Açılan sayfada Görsel 1.2'de görülen kurulum dosyalarından işletim sisteminize uygun olan kurulum dosyasını indiriniz.

1. ÖRNEK

Küçük bir çocuğun ilk kez gördüğü bir hayvanı nasıl öğrendiğini düşününüz. Bu çocuk ilk kez bir kedi gördüğünde belki şaşıracak ve ne olduğunu soracaktır. "Bu bir kedi" yanıtını aldığında zihninde kediye ilişkin bir şablon oluşmaya başlar. Çocuk ilerleyen zamanlarda farklı renklerde, farklı türlerde ve farklı boyutlarda kediler ile karşılaştığında bunların da birer kedi olduğu birileri tarafından ona söylenir (Görsel 2.2).

SIRA SİZDE

Farklı bir görselleştirme yöntemi olarak keman (violin) grafiği kullanılabilir. Örnekte kind parametresi grafik türünü belirtmek için kullanılmaktadır ve argüman olarak violin verilmiştir. İnner parametresi grafik içinde veriyi göstermek için (çizgiler) kullanılmaktadır, argüman olarak stick verilmiştir. split parametresi farklı kategoriler için grafiği bölmek için kullanılmaktadır. Örnekte cinsiyete göre keman şeklinde bir görselleştirme yapılmıştır ve bu şekilde kategorilere ilişkin veri kolay bir şekilde karşılaştırılabilir.

1. ÖLÇME VE DEĞERLENDİRME

1. **Aşağıdakilerden hangisi yapay zekânın tanımıdır?**
   A) Kendi zekânızla programlama yapma
   B) İnsan zekâsını bilgisayara yerleştirme
   C) Oyun programlama ve oynama
   D) Bir makineyi akıllı yapma
   E) Bir bilgisayarı tasarlama

2. **Aşağıdakilerden hangisi yapay zekânın alt alanlarından biri değildir?**
   A) Makine Öğrenmesi
   B) Sinir Ağları
   C) Uzman Sistem
   D) Doğal Dil İşleme

Konuyla ilgili notları gösterir.

Tabloları gösterir.

Kitap adını gösterir.

Sayfa numaralarını gösterir.

Ölçme ve değerlendirme sorularını gösterir.

*** Bu ders materyalinde ölçü birimlerinin uluslararası kısaltmaları kullanılmıştır.**

YAPAY ZEKÂYA GİRİŞ

1. ÖĞRENME BİRİMİ

http://kitap.eba.gov.tr/KodSor.php?KOD=36136

KONULAR

1.1. YAPAY ZEKÂ KAVRAMLARI
1.2. VERİ SETİ KAVRAMLARI

NELER ÖĞRENECEKSİNİZ?

- Yapay zekâ kavramı
- Yapay zekânın alt alanları
- Yapay zekâ geliştirme ortamlarını kullanma
- Yapay zekâ ile veri arasındaki ilişki
- Veri seti
- NumPy kütüphanesini kullanarak veriler üzerinde işlem yapma
- Pandas kütüphanesini kullanarak veriler üzerinde işlem yapma
- Verileri görselleştirme

TEMEL KAVRAMLAR

- Matplotlib
- NumPy
- Pandas
- Yapay zekâ

HAZIRLIK ÇALIŞMALARI

1. Canlıların taklit edildiği teknolojiler hakkındaki bilgilerinizi arkadaşlarınızla paylaşınız.
2. İnsanlar nasıl öğrenir? Sınıfta arkadaşlarınızla tartışınız.

1.1. YAPAY ZEKÂ KAVRAMLARI

Bilgisayar bir makinedir ve insanlar tarafından belirlenen görevleri yerine getirir. Yapay zekâ [Artificial Intelligence (AI)] bir makinenin insanlar gibi algılama, öğrenme, akıl yürütme ve problem çözme gibi davranışlar sergilemesidir. Yapay zekâ terimindeki yapay kelimesi insanlar tarafından oluşturulan anlamına gelir. Yapay zekâ, insanın veya doğanın zekâ gerektiren özelliklerine benzerlik gösteren sistemler tasarlamaktır. Yapay zekâ, içinde birden çok alt alanı barındıran bir çatıdır. Bu alt alanlar Görsel 1.1'de verilmiştir.

Görsel 1.1: Yapay zekâ alt alanları

Makine Öğrenmesi (Machine Learning): Bilgisayara yüklenen verilerden ve kullanılan algoritmalardan bilgisayarın kendi kendine öğrenmesini sağlayan tekniktir. Verilerden yararlanarak sonucu tahmin etmek için sistem eğitilir. Makine öğrenmesi algoritmaları normal algoritmaların gelişmiş hâlidir. Açık bir şekilde program kodları yazmadan sonucu tahmin etmek için kullanılan algoritma ve veriler ile bir model oluşturulur. Bu model kullanılarak yapılan program akıllı hâle getirilir.

Doğal Dil İşleme (Natural Language Processing): Doğal dil işleme, makineler ile iletişim kurmak ve insan dilinden bilgi edinmek için kullanılan tekniktir. Bir makinenin insan ile etkileşiminden konuşulan dili okuma, çözümleme, anlama ve dilden anlam çıkarma sürecine **Doğal Dil İşleme** denir.

Bulanık Mantık (Fuzzy Logic): Bulanık mantık, belirsizliği olan problemlerin çözümü ile ilgilenen yapay zekâ tekniğidir. Klasik programlamada doğru-yanlış (true-false) olma durumundan farklı olarak ikilem durumlarında karar vermeye dayalı sistemlerde kullanılır.

Uzman Sistem (Expert System): Uzman sistem yapay zekâ tabanlı bilgisayar sistemidir. Bu bilgisayar sistemi öğrenir ve karşılık verir. Arayüz, çıkarım motoru ve bilgi tabanı olmak üzere üç kısımdan oluşur. Arayüz üzerinde gelen veriler çıkarım motoruna iletilir, çıkarım motoru bilgi tabanından yararlanarak sonuçlar üretir.

Robotik (Robotics): Robotik akıllı sistemlerin vücut bulmuş hâlidir. Robotlar; belirlenen eylemleri gerçekleştirerek sonuçlar üreten, gerçek dünyada hareket eden araçlardır.

Yapay Sinir Ağları (Artificial Neural Networks): Yapay Sinir Ağları insan beynindeki öğrenme süreçlerini taklit etmeye çalışan bir modeldir. Yapay Sinir Ağları katmanlı bir yapıya sahiptir. İlk katmanda girdiler sayesinde oluşturulan model üzerinde gizli katmanlar bulunur. Son katman üzerinden sonuçlar alınır. Gizli katmanların görevi sağlanan girdilerden bilgi çıkarımı yapmaktır.

Günlük hayatta, veri aramada, ürün satın almada, haberleşmede ve daha birçok yerde yapay zekâ uygulamaları kullanılır. Aşağıda yapay zekâ uygulamalarından bazılarına örnekler verilmiştir.

Otonom Araçlar: Kendi kendini süren araçlardır. Bu araçlar çevrelerindeki dünyayı algılayarak insan müdahalesi olmadan hareket edebilir. Araçlarda bulunan sensörler ile yapay zekâ teknolojisinin birleşiminden oluşur.

Sağlık Sektörü: Yapay zekâ teknolojisi birçok hastalığın teşhisi ve klinik kararları vermek için kullanılır. Hastalığın teşhisinde kalp ritimleri, kan değerleri, laboratuvar tahlil sonuçları vb. verilerden yararlanarak hastalığın tespitini ve tedavisini hızlandırır.

Konuşma Tanıma: İnsanların konuşmalarını tanıyarak yapılmak istenen işlemleri yerine getiren yapay zekâ uygulamalarıdır.

Görüntü İşleme: Resimlerdeki veya videolardaki nesneleri ve kişileri tanımlamasını ve anlamasını sağlayan yapay zekâ uygulamalarıdır.

Eğitim Alanı: Eğitim alanında yapay zekâ kullanılarak her öğrencinin düzeyine göre uygun ders programı sunarak başarının artırılmasını sağlayan yapay zekâ uygulamalarıdır.

Nesnelerin İnterneti: İnternete bağlı cihazlardan alınan büyük veriler analiz edilerek kullanılan algoritmalar ile sonuçlar elde edilen yapay zekâ uygulamalarıdır.

1.1.1. Yapay Zekâ Geliştirme Ortamlarının Kurulumu

Yapay zekâ geliştirmek için birden çok yazılım dili kullanılabilir. Bu yazılım dilleri içinden genellikle Python programlama dili kullanılır. Python programlama dili makine öğrenmesi, veri bilimi ve yapay zekâ alanında sağladığı avantajlar sayesinde daha çok tercih edilmektedir. Bu kitapta Python programlama dili kullanılarak yazılımlar geliştirilecektir.

Yapay zekâ geliştirmek için kullanılan diğer programlama dillerinden bazıları şunlardır:

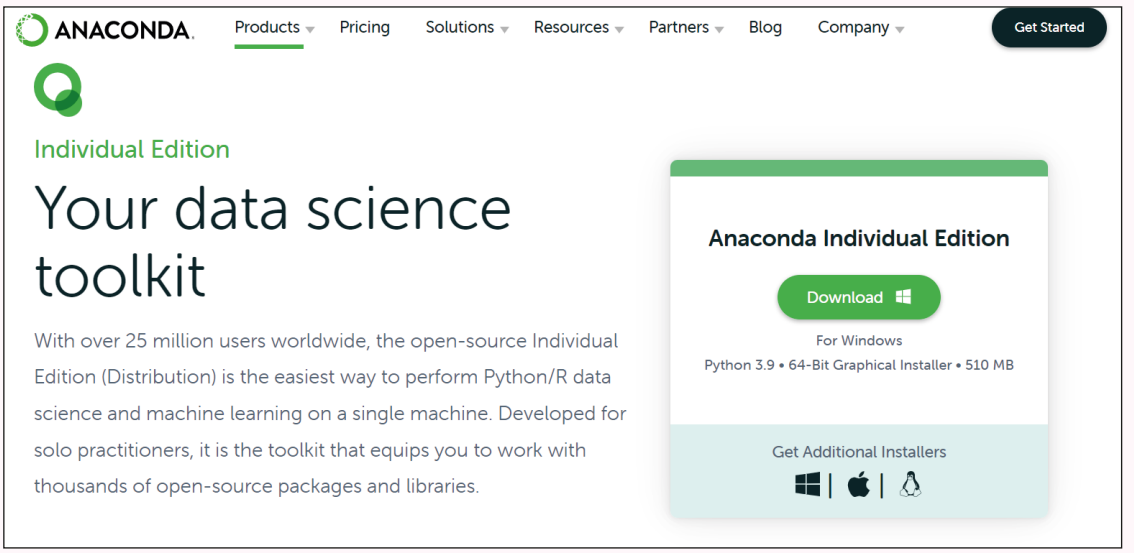
- R
- Java
- C++, C#
- Ruby
- Javascript
- Scala
- Lisp
- Prolog
- Julia
- Haskell
- GO

Yapay zekâ yazılımı geliştirmek için programlama dili seçildikten sonra geliştirme ortamı ve araçlar kurulmalıdır. Bu kitapta geliştirme ortamı için Anaconda kullanılacaktır. Anaconda kurulumu ile Bütünleşik Geliştirme Ortamları (IDE), araçlar ve kütüphaneler birlikte kurulur. Anaconda ile karmaşık yapılandırma ayarları ve bakımı daha kolay hâle getirilmiştir.

1. UYGULAMA

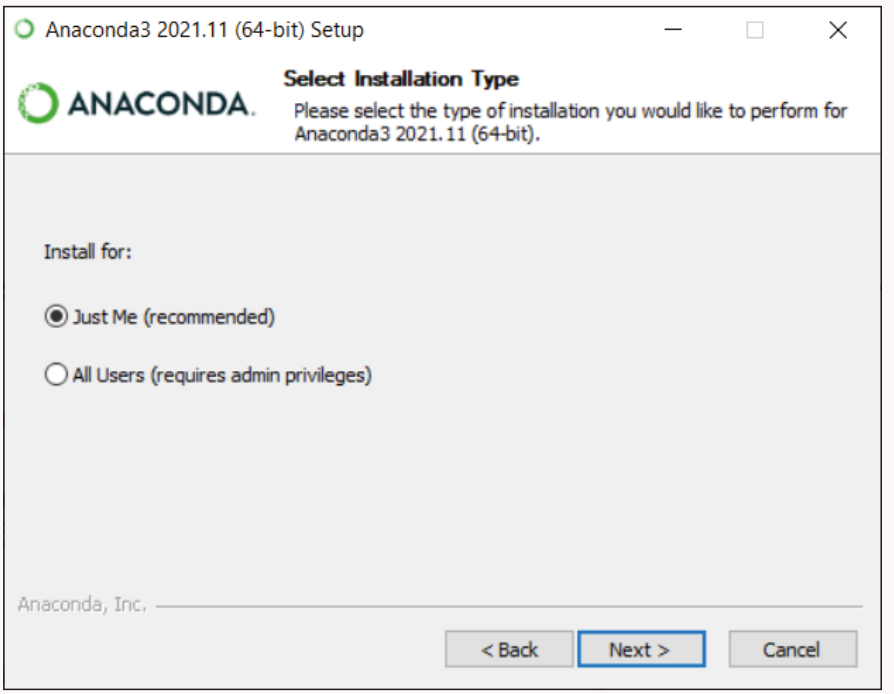
Yapay Zekâ ve Makine Öğrenmesi uygulamaları geliştirmek amacıyla kullanılan Anaconda Bütünleşik Geliştirme Ortamının kurulumu için aşağıdaki adımları takip ediniz.

1. Adım: Anaconda'nın kurulumu için tarayıcınızda https://www.anaconda.com/products/individual adresini yazınız. Açılan sayfada Görsel 1.2'de görülen kurulum dosyalarından işletim sisteminiz için uygun olanı indiriniz.



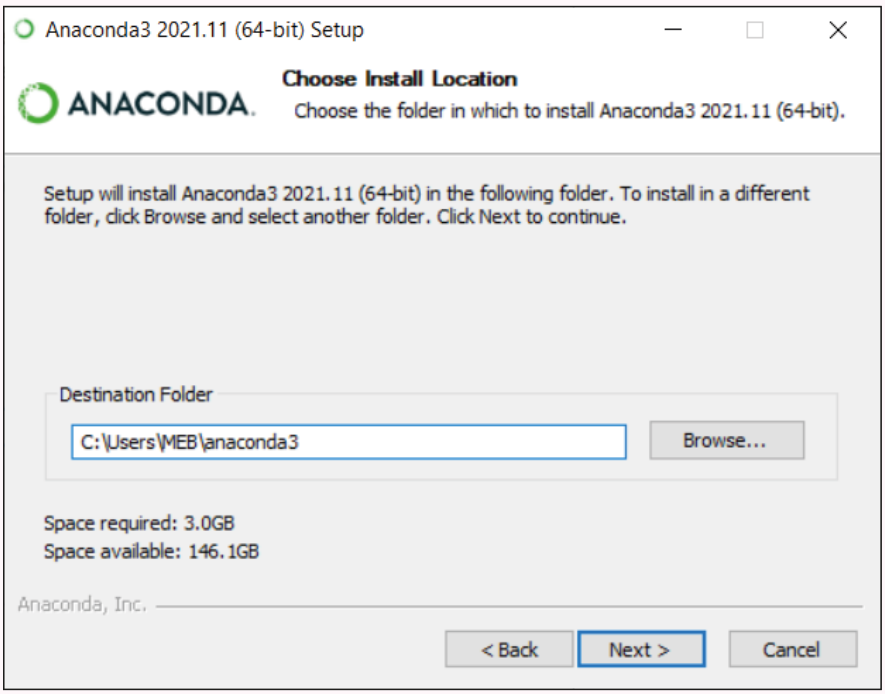
Görsel 1.2: Anaconda kurulum dosyasını indirme web sayfası

2. Adım: İndirilen kurulum dosyasını açınız. Lisans anlaşmasını kabul ettikten sonra Görsel 1.3'te gösterilen kurulum tipini seçiniz.



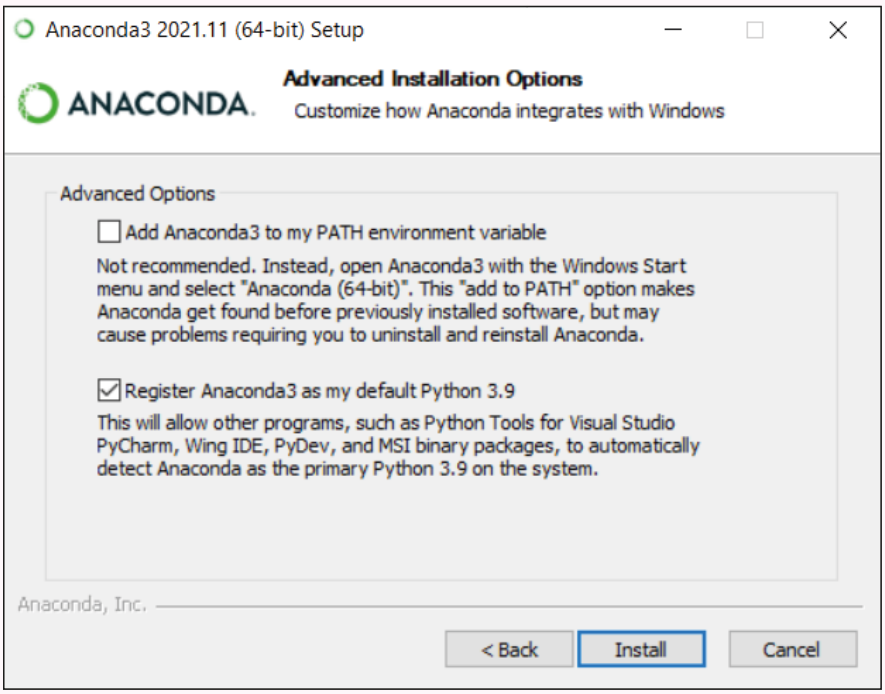
Görsel 1.3: Anaconda kurulum tipi seçimi

3. Adım: Görsel 1.4'te Anaconda uygulamasının kurulumunun yapılacağı klasörü belirleyiniz. Farklı bir klasöre kurulum yapılacak ise Browse butonunu tıklayarak farklı bir klasör seçiniz.



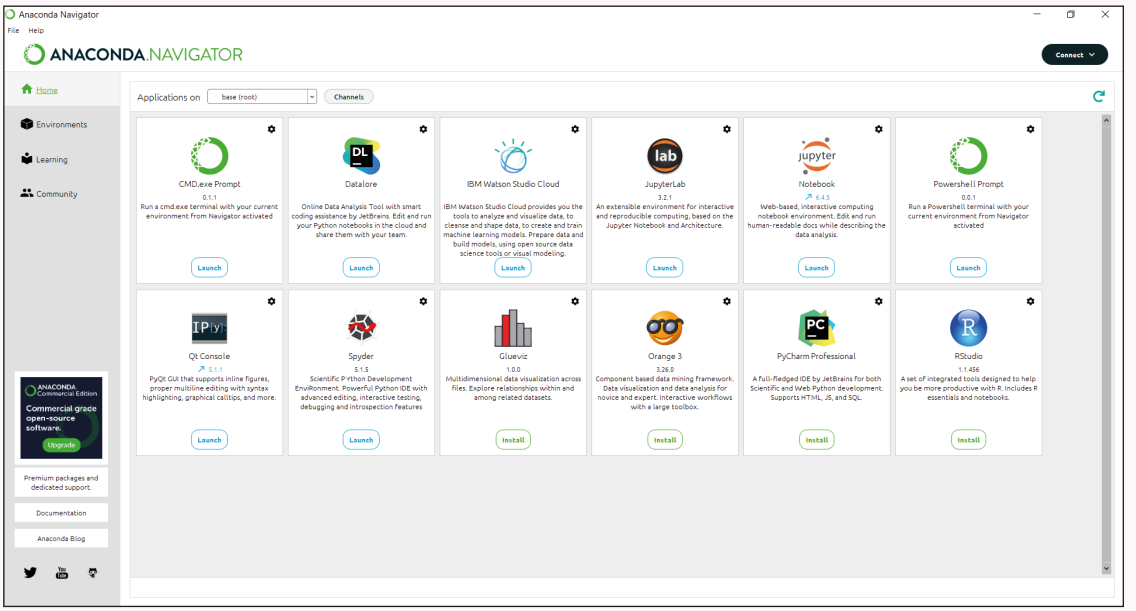
Görsel 1.4: Anaconda kurulum klasörünün seçimi

4. Adım: Gelişmiş kurulum ayarları penceresinden Görsel 1.5'teki gibi uygulamanın Path ayarları ve Python seçimi yapılabilir. Kurulum ayarlarını ön tanımlı olarak yapmak için Görsel 1.5'teki gibi seçimi işaretli bırakarak Install butonuna tıklayınız.



Görsel 1.5: Anaconda gelişmiş kurulum seçenekleri

5. Adım: Kurulum tamamlandıktan sonra Görsel 1.6'daki Anaconda Navigator uygulamasını açarak çalıştığını gözlemleyiniz.



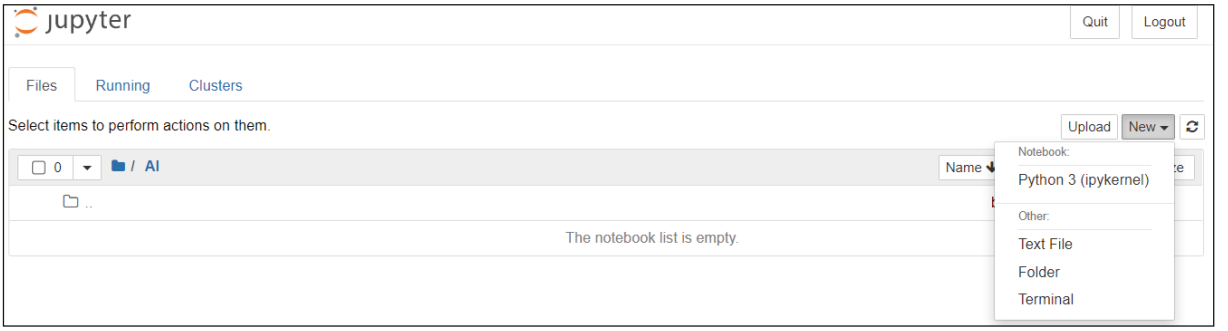
Görsel 1.6: Anaconda Navigator uygulaması

Jupyter Notebook; açık kaynak kodlu, web tabanlı, etkileşimli bir geliştirme ortamıdır. Jupyter Notebook ile yazılan kodların çıktısı anında görülebilir, veriler görselleştirilebilir ve matematiksel denklemler oluşturulabilir. Python, PHP, R, C# gibi programlama dilleri kullanılarak uygulamalar geliştirilebilir.

Jupyter Notebook açıldığında Görsel 1.7'deki gibi üç sekme içeren ekran ile karşılaşılır. "Files" sekmesi geçerli dizindeki dosya ve klasörleri görüntülemek için kullanılır. "Running" sekmesi çalışmakta olan Jupyter Notebook dosyalarını gösterir. "Clusters" sekmesi interaktif paralel hesaplama yapmada kullanılır.

Görsel 1.7: Jupyter Notebook uygulaması

Görsel 1.8'de açılan uygulama sayfasında New butonuna tıklandığında yeni bir Notebook oluşturma işlemi yapılır.



Görsel 1.8: Jupyter Notebook dosyası oluşturma

Jupyter Notebook uygulamasında otomatik olarak IPython çekirdeği yüklü olarak gelir. Çekirdek, Jupyter uygulamaları ve kullanıcı arabirimleriyle etkileşime giren programlama diline özgü yapıdır. Görsel 1.9'da üç ana çekirdek gösterilmiştir.

Görsel 1.9: Anaconda Navigator uygulaması

Kitapta Python programlama dili kullanılacağı için Notebook uygulamalarında IPython çekirdeği tercih edilecektir. Görsel 1.10'da oluşturulan Notebook dosyalarında kullanılan komutlar gösterilmiştir.

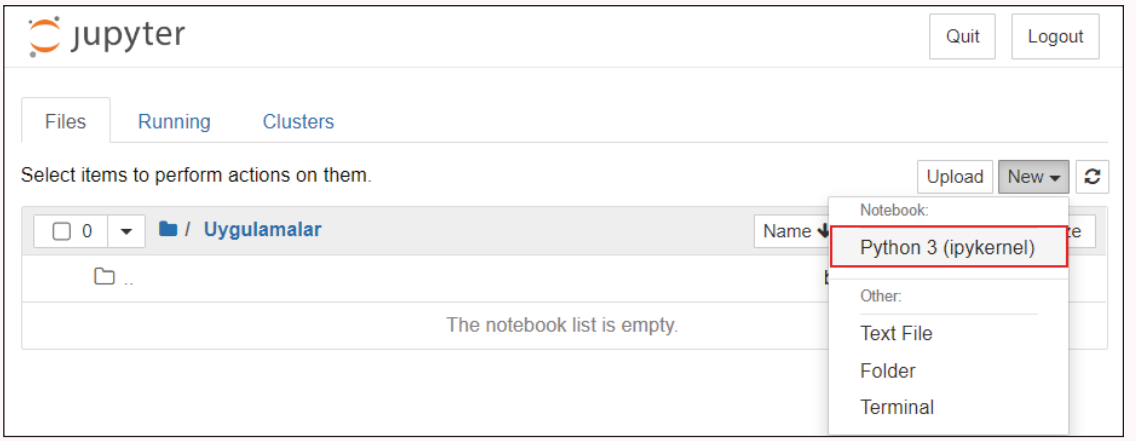
Görsel 1.10: Jupyter Notebook komutları

1. Kaydet
2. Alta Hücre Ekle
3. Hücre Kes
4. Hücre Kopyala
5. Hücre Yapıştır
6. Hücreyi Yukarı Taşı
7. Hücreyi Aşağı Taşı
8. Seçili Hücreyi Çalıştır
9. Çekirdeği Durdurma
10. Çekirdeği Yeniden Başlatma
11. Çekirdeği Yeniden Başlatma ve Tekrar Başlatma
12. Yazım Türünü Belirleme
13. Komut Paleti
14. Mevcut Çekirdek
15. Çekirdek Durumu
16. Notebook Sunucusu Çıkışı

2. UYGULAMA

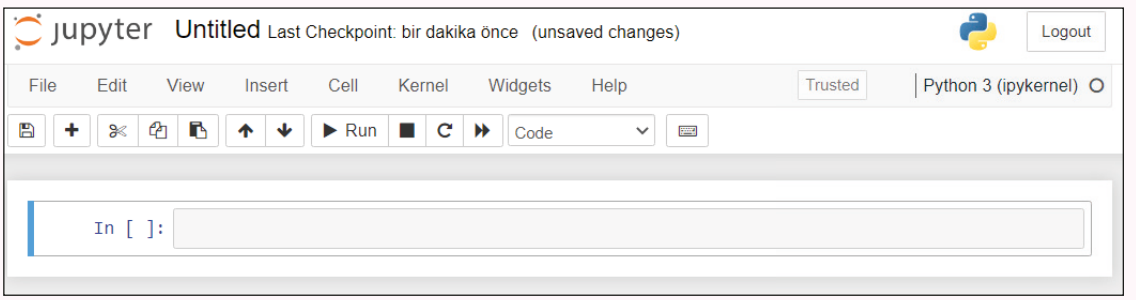
Aşağıdaki işlem basamaklarını takip ederek yeni bir not defteri oluşturma işlemlerini gerçekleştiriniz.

1. Adım: Jupyter Notebook dosyası oluşturmak için Görsel 1.11'de belirtildiği gibi New butonuna tıkladıktan sonra Python 3'ü seçiniz.



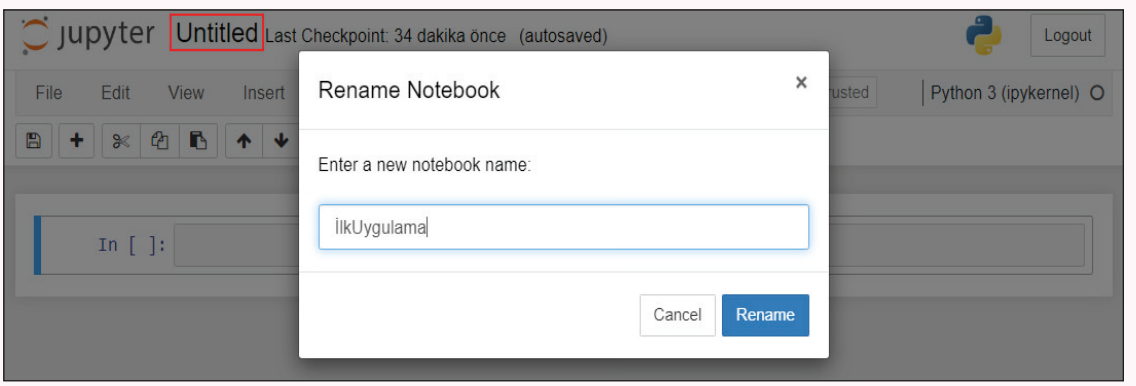
Görsel 1.11: Jupyter Notebook oluşturma

2. Adım: Görsel 1.12'de yeni sekmede oluşturulan yeni Jupyter Notebook'u gözlemleyiniz.



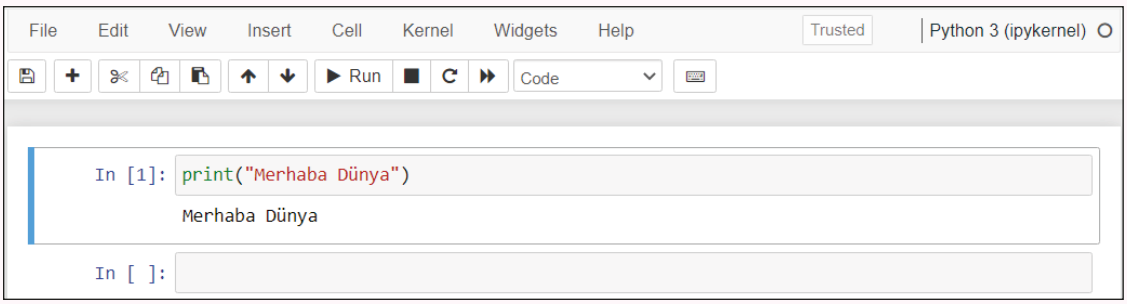
Görsel 1.12: Yeni Jupyter Notebook dosyası

3. Adım: Jupyter Notebook dosyasını yeniden adlandırmak için Görsel 1.13'te ekranın üst kısmındaki Untitled (Başlıksız) kısmına çift tıklayarak açılan pencerede dosya ismini değiştiriniz.



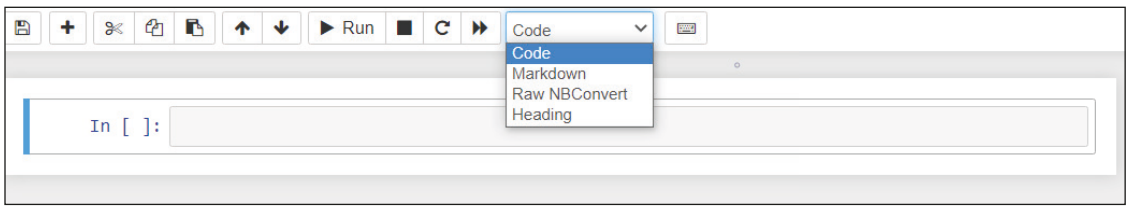
Görsel 1.13: Jupyter Notebook dosya adını değiştirme

4. Adım: Dosya adını değiştirdikten sonra Görsel 1.14'teki gibi ilk hücreye kodlarını yazıp Run (Çalıştır) butonuna tıklayarak veya Shift+Enter tuşlarına basarak seçili olan hücreyi çalıştırınız.



Görsel 1.14: Jupyter Notebook hücreleri

Jupyter Notebook'ta Görsel 1.15'te gösterildiği gibi kullanılan dört hücre türü vardır.



Görsel 1.15: Jupyter Notebook hücre türleri

Kod (Code) Hücresi: Kullanılan programlama diline bağlı olarak programlama dilinin kodlarının yazıldığı alandır. Yazılan kodlar çalıştırıldığında yazı, resim, çizim veya Html tablosu olarak çıktı verir.

Markdown (İşaretleme) Hücresi: Dosyada görsel olarak belirtilmek istenen yazı, başlık, bağlantı, sıralı / sırasız liste, bağlantı ve tablo içerikleri göstermek için kullanılır.

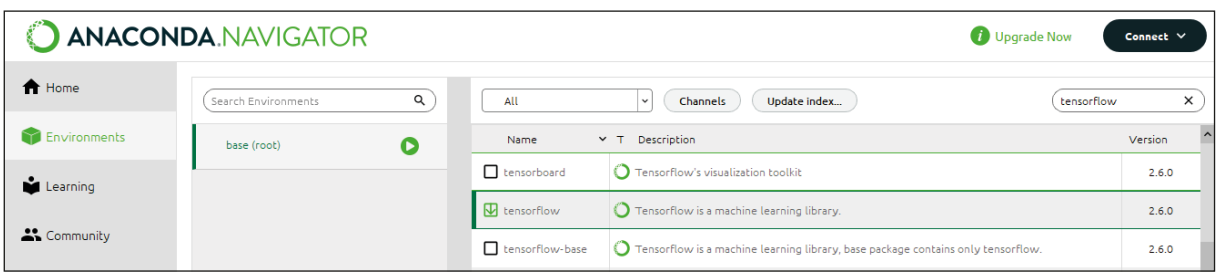
Raw (İşlenmemiş) Hücresi: Herhangi bir derleme işlemi yapılmadan programlama dili tarafından değerlendirilmeden doğrudan çıktı yazılan hücrelerdir.

Heading (Başlık) Hücresi: Başlık eklemek için kullanılan hücrelerdir.

1.1.2. Geliştirme Ortamına Paket Yükleme ve Kaldırma

Anaconda ortamına uygulama geliştirirken bazen önceden oluşturulmuş paketlerin kurulmasına ihtiyaç duyulabilir. Anaconda geliştirme ortamında hem grafik arayüzünü hem de komut satırını kullanarak paket yükleme işlemi gerçekleştirilebilir. Mevcut Anakonda ortamına paket eklemenin birden çok yolu vardır. Bu yollar şunlardır:

1. Yol: Anaconda Navigator kullanarak paket ekleme işlemini gerçekleştirmek yaygın olarak kullanılan yoldur. Anaconda Navigator uygulaması açıkken Environments sekmesine geçiş yapılarak Görsel 1.16'da gösterildiği gibi grafik arayüzü kullanılarak kurulacak olan paketlerin araması yapılabilir. Kurulacak paketler seçildikten sonra Apply butonuna tıklanarak paketlerin ortama kurulması sağlanır.



Görsel 1.16: Anaconda Navigator paket kurulumu

2. Yol: Paketleri kurmanın başka bir yolu da Anaconda Komut İstemi kullanılarak kurulum yapılmasıdır. Anaconda Komut İstemi penceresi yönetici olarak açıldıktan sonra belirlenen sanal ortama aşağıdaki komut kullanılarak kurulum yapılabilir.

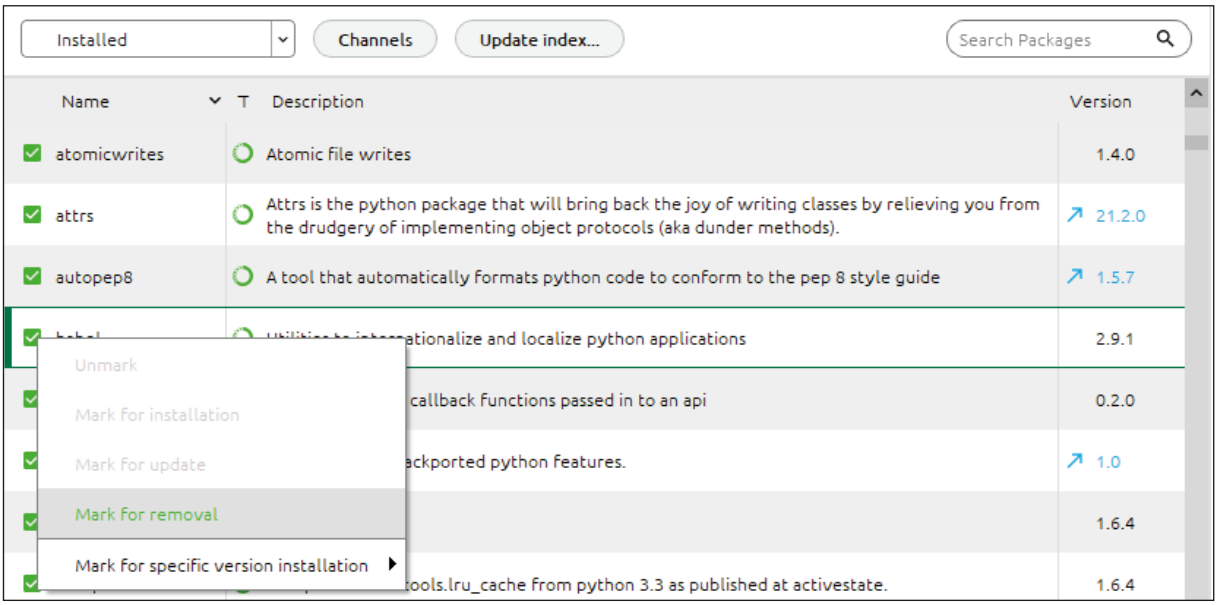
```
conda install opencv
```

3. Yol: Kurulması istenen paket Anaconda ortamında veya Anaconda Navigator aracılığıyla bulunamıyorsa kurulması istenen paket **pip** gibi bir paket yöneticisi ile bulunabilir ve aşağıdaki komut kullanılarak kurulabilir.

```
pip install scikit-learn
```

Yüklü olan paketlerin sanal ortamdan kaldırılması için Anaconda geliştirme ortamında hem grafik arayüzünü hem de komut satırını kullanarak paket kaldırma işlemi gerçekleştirilebilir. Bu işlemler için izlenebilecek yollar şunlardır:

1. Yol: Anaconda Navigator uygulaması açıkken Environments sekmesine geçiş yapılarak **açılır listeden** Installed menü ögesi seçilerek yüklü olan paketler listelenir. Kaldırılmak istenen paketler Görsel 1.17'de olduğu gibi paket ismine tıklanarak açılan pencerede "Mark for removal" seçeneği seçildikten sonra Apply butonuna tıklanarak kaldırma işlemi gerçekleştirilir.



Görsel 1.17: Anaconda Navigator paket kaldırma

2. Yol: Paketleri kaldırma işlemi Anaconda Komut İstemi kullanılarak yapılabilir. Kaldırılmak istenen paket adı **conda uninstall paket_adı** komutu kullanılarak kaldırılabilir.

3. Yol: Paketleri kaldırma işlemi pip paket yöneticisi kullanılarak yapılabilir. Komut istemi penceresi açılarak aktif sanal ortamda kaldırılmak istenen paket adı **pip uninstall paket_adı** komutu kullanılarak kaldırılabilir.

1.1.3. Yapay Zekâ Çevrimiçi Geliştirme Ortamları

Yapay zekâ uygulamaları geliştirmede Anaconda gibi platformlar yerine çevrimiçi platformlar da kullanılır. Bu çevrimiçi platformların avantajı gerekli yazılım ortamlarının ve kütüphanelerin kurulmasına gerek duyulmamasıdır. Bu çevrimiçi geliştirme ortamlarından biri Google Colab'dır. Google Colab diğer adıyla Colaboratory, bulut tabanlı ücretsiz bir geliştirme ortamıdır.

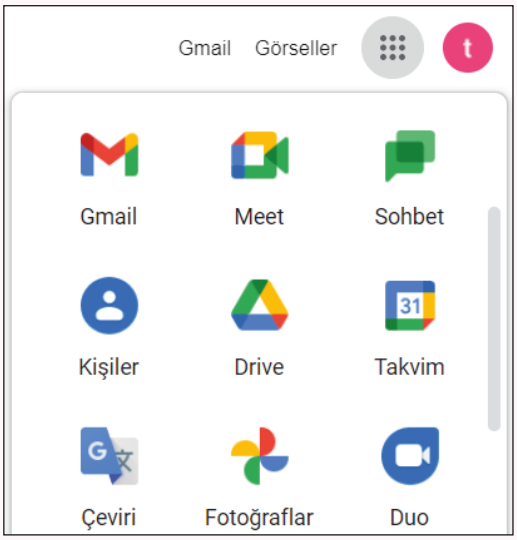
SIRA SİZDE

Yapay Zekâ ve Makine Öğrenmesi alanında sıkça kullanılan açık kaynak kütüphanelerini ve özelliklerini araştırınız. Edindiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

3. UYGULAMA

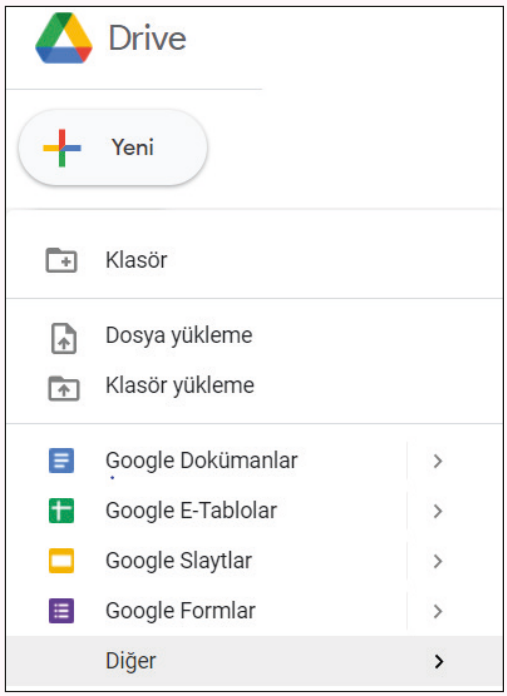
Bulut tabanlı yapay zekâ geliştirme ortamını oluşturma işlemlerini verilen adımlara göre yapınız.

1. Adım: Gmail hesabınız ile giriş yaptıktan sonra Görsel 1.18'deki menüyü açarak Drive ikonuna tıklayınız.



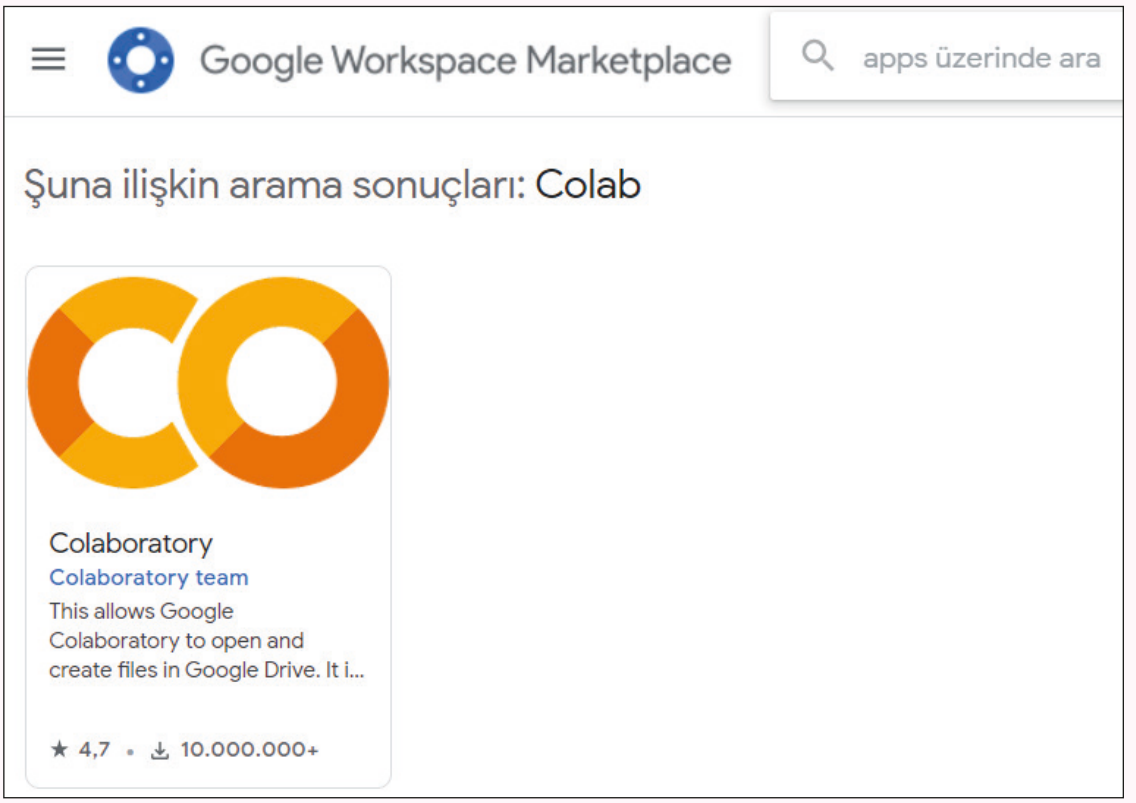
Görsel 1.18: Uygulamalar menüsü

2. Adım: Drive geçiş yaptıktan sonra Görsel 1.19'daki Yeni butonuna tıklayarak açılan menüde eğer Google Colab mevcut değil ise Diğer menüsünü tıklayınız.



Görsel 1.19: Drive Yeni menüsü

3. Adım: Açılan menüde "Daha Fazla uygulama bağla" linkine tıklayarak Görsel 1.20'de açılan pencerede Colaboratory uygulamasını bularak ekleme işlemini yapınız.

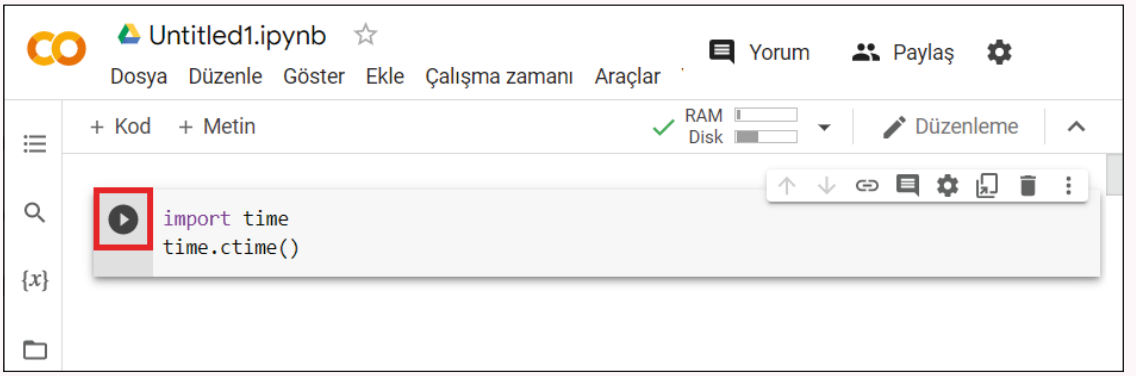


Görsel 1.20: Colab uygulamasını ekleme

4. Adım: Aşağıdaki Python kod parçasını yazınız.

```
import time
time.ctime()
```

5. Adım: Yazılan kodu çalıştırmak için Görsel 1.21'de gösterilen kod hücresinin sol tarafındaki butona tıklayınız.

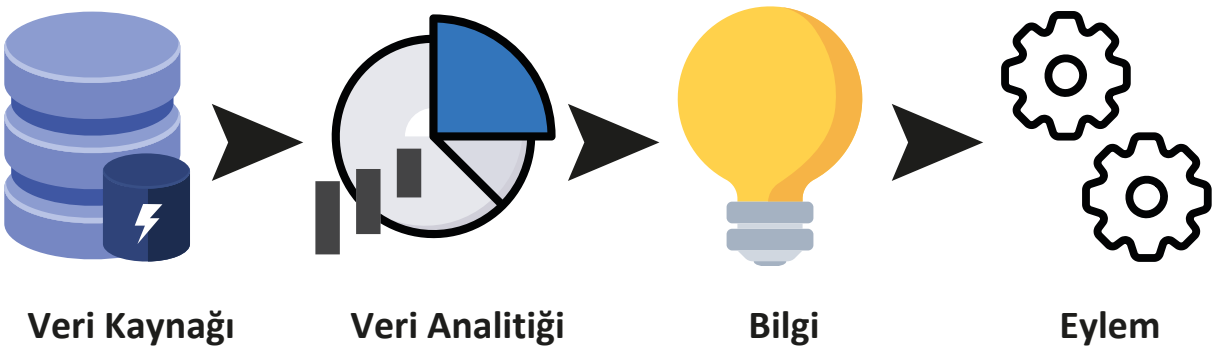


Görsel 1.21: Colab kod parçasının çalıştırılması

1.2. VERİ SETİ KAVRAMLARI

Yapay zekâ; verileri sistematik bir şekilde ve onlardan öğrenmek için kullanır. Veriler, yapay zekâ modellerinin temel bileşenidir. Yapay zekâ teknolojileri verileri işlemek, verilerdeki kalıpları tanıyarak bilgisayarları belirli görevleri yerine getirmek ve eğitmek için kullanır.

Veriler günümüzde büyük öneme sahiptir. Verilerden elde edilen bilgiler istatistik, matematik, bilgisayar bilimleri, yapay zekâ vb. birçok alanda kullanılır. Verilerden bilgi elde etmek için sistemleri, süreçleri ve bilimsel yöntemleri kullanarak veri ile ilgili her türlü konuyu inceleyen bilim dalına **veri bilimi** denir. Veri bilimi ile uğraşan, veriden faydalı bilgi çıkarma sürecini yöneten kişiye de **veri bilimci** denir. Görsel 1.22'de veri biliminin süreçleri gösterilmiştir.



Görsel 1.22: Veri bilimi süreçleri

Yapay zekâ insan zekâsında olduğu gibi bir öğrenme gerçekleştirebilmek için verilere ihtiyaç duyar. Örnek olarak bir emlakçı ev satışlarında tahminde bulunabilmek için satılan evlerin fiyatlarını bilmelidir. Bir bilgisayar programı olan yapay zekânın da tahminde bulunabilmesi için veriler gereklidir. Yapay zekâ alanında veriler temelde dört gruba ayrılır.

Sayısal veriler boy, kilo, maaş, yaş gibi tam veya ondalık sayılardan oluşan veri türleridir. Bu tür veriler matematiksel hesaplama, istatistiksel grafik oluşturma işlemlerinde kullanılır.

Kategorik veriler cinsiyet, sosyal sınıf, doğum yeri, yaşadığı ülke gibi metinsel verilerden oluşur. Kategorik veriler yapay zekâ ve makine öğrenmesinde benzer özellikler gruplama ve filtreleme yapmakta çokça kullanılır.

Zaman serileri zaman aralığında belirli noktaları kaydetmek için kullanılır. Kaydedilen zaman noktalarını gün, hafta, ay ve yıl olarak herhangi bir zaman değerine göre karşılaştırma ve sıralama işlemlerinde kullanılır.

Metin verileri sözcükler, cümleler ve paragraflardan oluşan metinsel verilerdir. Yapay zekâ içinde bu veriler kelime sıklığı, metin sınıflandırma ve cümle analizi amaçları için kullanılır.

Bu temel dört sınıfın haricinde görüntü işleme için resim ve video, konuşma tanıma için ses verileri de bulunmaktadır.

1.2.1. Veri Seti

Veri seti (Dataset) temel olarak verilerin belirli bir sırayla düzenlendiği veri topluluğu dosyasıdır. Veri setleri bir diziden veri tabanı tablosuna kadar her şey olabilir. Veri setleri genel olarak CSV veya elektronik tablo formatında, satırlar ve sütunlardan oluşan bir tablo olarak düzenlenen tek bir dosyadır. Bazı durumlarda veri setleri farklı formatta birden çok dosyadan oluşabilir.

Veri setlerinde verilerin elde edilmeleri şu şekildedir:

- Bilgisayarlar veya makinelerden elde edilen veriler
- Anketler yoluyla toplanan veriler
- İnsanların gözlemlerinden elde edilen veriler
- Web siteleri veya API'ler aracılığı ile elde edilen veriler

Veri setleri kategorik olarak şu şekildedir:

- Makine öğrenmesi ve veri analizi projeleri için genel veri setleri
- Bilgisayarlı görme (computer vision) projeleri için video ve resimler
- Doğal Dil İşleme (Natural Language Processing) projeleri için metinler
- Konuşma ve ses tanıma projeleri için sesler

Bir veri seti içindeki veri türleri şunlardır:

- **Sayısal Veri:** Kişilerin yaşları, ev fiyatları, sıcaklık vb.
- **Kategorik Veri:** Cinsiyet, renkler, evet / hayır, şehirler vb.
- **Sıralı Veri:** Zayıf / orta / iyi / pekiyi, çocuk / genç / orta yaşlı / ihtiyar, evet / hayır vb.

Makine öğrenmesi ve yapay zekâ projelerinde modelleri eğitmek için büyük miktarda verilere ihtiyaç duyulur. Bir makine öğrenmesi ve yapay zekâ projesinde verilerin toplanması ve hazırlanması en önemli kısımdır. İyi hazırlanmamış bir veri seti yapay zekâ projelerinde doğru sonuçların elde edilmemesine neden olmaktadır. Tablo 1.1'de bir veri setinin örneği verilmiştir.

Tablo 1.1: Veri Seti Örneği

Ad	Maaş	Deneyim	Cinsiyet	Şehir
Zeynep	13000 TL.	8	Kadın	Ankara
Ali	11500 TL.	5	Erkek	İstanbul
Ahmet	10000 TL.	3	Erkek	İzmir
Ayşe	15000 TL.	10	Kadın	Antalya

1.2.2. Açık Veri Setleri

Bu veri setleri halka açık bir şekilde bilgilendirme amacı ile oluşturulmuş olup makine öğrenmesi ve yapay zekâ uygulamalarında kullanılabilir. Bu veri setleri şunlardır:

Devletlerin Veri Setleri

- ABD devlet kurumları açık veri setleri https://data.gov
- Birleşik Krallık hükümeti açık veri setleri https://data.gov.uk/
- Avrupa Birliği açık veri setleri https://data.europa.eu/en
- İstanbul Büyükşehir Belediyesi açık veri setleri https://data.ibb.gov.tr
- Türkiye İstatistik Kurumu açık veri setleri https://data.tuik.gov.tr

UCI Makine Öğrenmesi Deposu

Bu depoda makine öğrenimi topluluğu tarafından makine öğrenimi ve yapay zekâ uygulamaları için yaygın olarak kullanılan veri setleri bulunur. UCI makine öğrenimi deposundan yararlanmak için https://archive.ics.uci.edu/ml/index.php bağlantı adresi kullanılır.

Kaggle

En çok kullanılan ve en yaygın olarak bilinen veri kaynağıdır. İçinde istatistiksel veriler, metin, ses ve bilgisayar görüşüne kadar çok sayıda veri seti bulunur. Kaggle veri setleri için https://www.kaggle.com/datasets bağlantı adresi kullanılır.

AWS Veri Setleri

Herkese açık olan veri setleri aranabilir, erişilebilir, paylaşılabilir ve indirilebilir. AWS veri kaynakları için https://registry.opendata.aws bağlantı adresi kullanılır.

Google Dataset Search

Kullanıcıların web üzerinde binlerce veri havuzuna yüklenen çok çeşitli veri setlerinden arama yapmasına olanak tanır. Arama motorunda bulunan tüm veriler istediğiniz amaç için kullanılamaz, bu nedenle lisansları ve kullanım kısıtlamaları kontrol edilmelidir. Google veri seti arama motoru için https://toolbox.google.com/datasetsearch bağlantı adresi kullanılır.

Microsoft Research Open Data

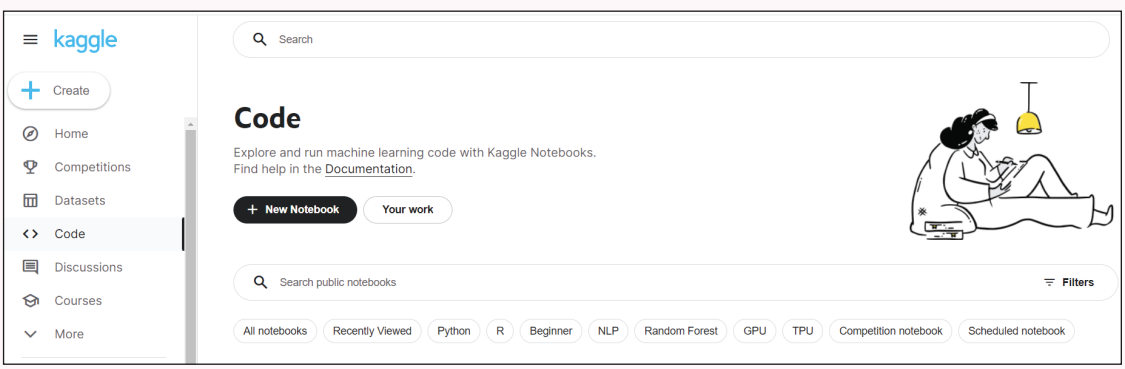
Microsoft tarafından ücretsiz sunulan veri seti deposudur. Bu depo içinde görüntü işleme, doğal dil işleme ve veri bilimi için kullanılacak çeşitli veri setleri bulunur. Microsoft tarafından sunulan veri setlerini kullanmak ve indirmek için https://msropendata.com bağlantı adresi kullanılır.

4. UYGULAMA

Kaggle Notebook uygulamasını kullanmak için aşağıdaki adımları takip ediniz.

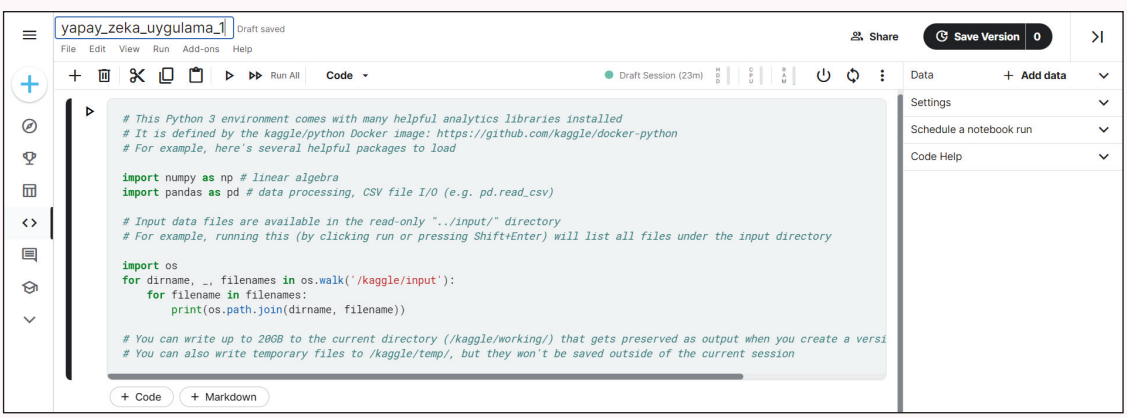
1. Adım: Tarayıcınızdan adres satırına www.kaggle.com yazarak Kaggle web sitesini görüntüleyiniz.

2. Adım: Yeni bir Notebook oluşturmak için Görsel 1.23'te Code linkine tıkladıktan sonra New Notebook butonuna tıklayınız.



Görsel 1.23: Kaggle ile yeni Notebook oluşturma

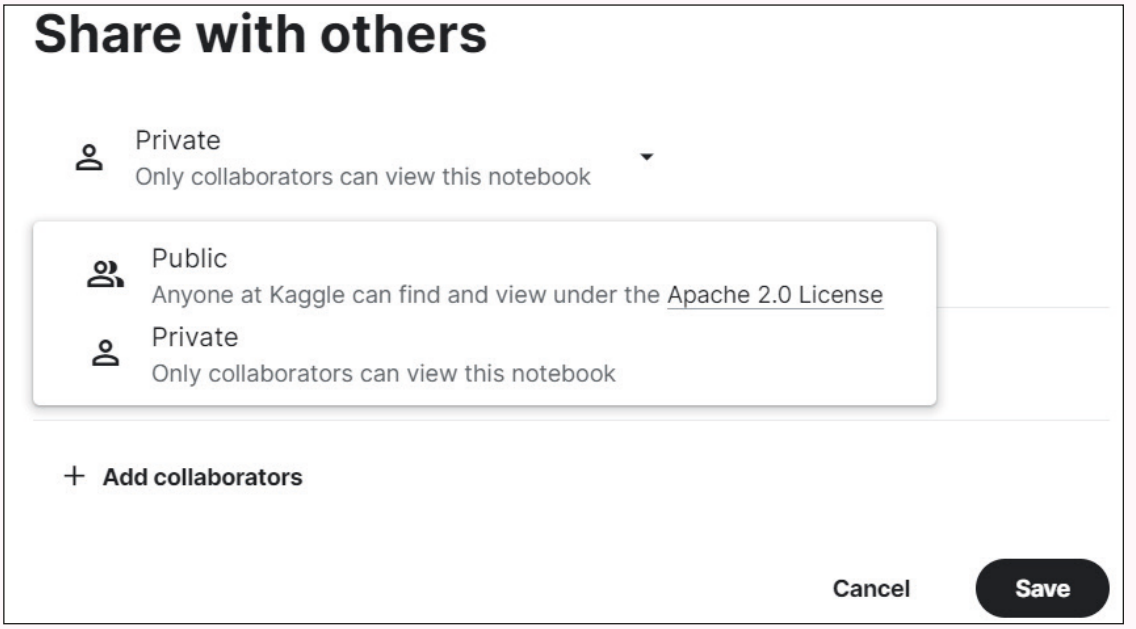
3. Adım: Notebook oluşturma işleminden sonra bulut kaynakları kullanılarak Notebook oluşturulacaktır. Görsel 1.24'te Notebook sayfasının sol üst köşesine tıklayarak yeniden isim veriniz.



Görsel 1.24: Notebook sayfası

4. Adım: Notebook dosyasını herkese açık hâle getirmek veya belirli kişileri dâhil etmek için Görsel 1.24'te gösterilen **Share** butonuna tıklayınız.

3. Adım: Görsel 1.25'te gösterildiği gibi Notebook paylaşımını herkese açık yapmak istiyorsanız Public, ortak çalışanları dâhil etmek istiyorsanız Private seçeneğini seçiniz. Add collaborators seçeneği ile ortak çalışanları belirleyebilirsiniz.

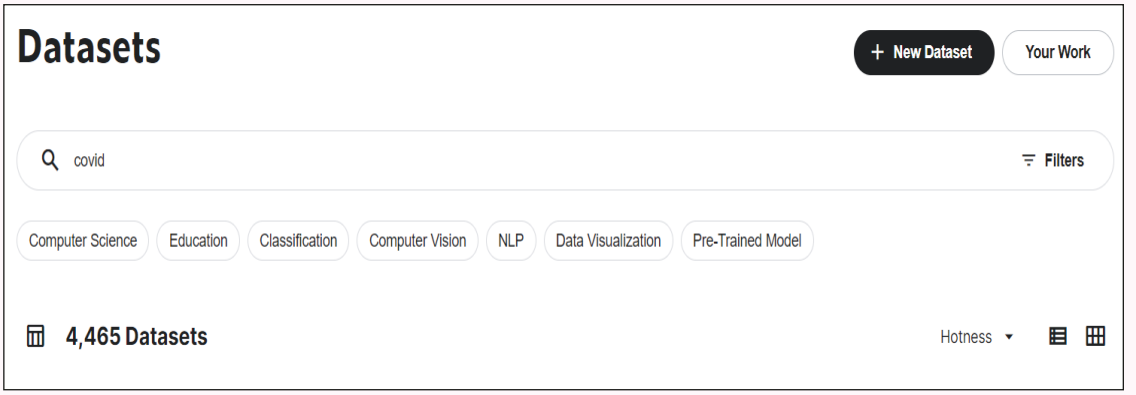


Görsel 1.25: Notebook paylaşım penceresi

5. UYGULAMA

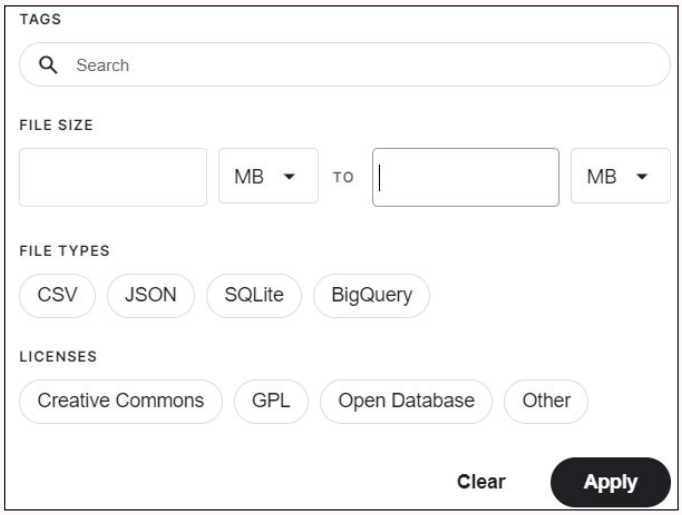
Kaggle veri setleri arama, diğer kullanıcılar tarafından oluşturulmuş veri setlerine erişim sağlama ve Notebook içine aktarma işlemleri için aşağıdaki adımları takip ediniz.

1. Adım: Bir veri seti aramak için Datasets linkine tıkladıktan sonra anahtar kelimeleri Görsel 1.26'da gösterildiği açılan sayfada arama alanına yazınız.



Görsel 1.26: Veri seti arama

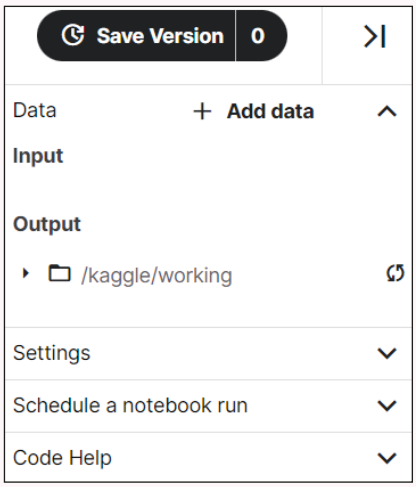
2. Adım: Arama işlemini filtrelemek için arama alanının sağ tarafında bulunan Filters butonuna tıkladıktan sonra Görsel 1.27'de açılan pencereden filtreleme yapabilirsiniz.



Görsel 1.27: Arama filtreleme

3. Adım: Veri setleri listesinden bir veri seti seçildiğinde açılan sayfada Data Explorer bölümüne geçerek veri setlerinin içeriğini ve yapısını inceleyiniz.

4. Adım: Seçilen veri setini bilgisayarınıza indirmek için Download butonuna tıklayınız. Seçilen veri setini Kaggle Notebook içinde kullanmak için Notebook oluşturduktan sonra Görsel 1.28'de Add data butonuna tıklayarak ekleme işlemini gerçekleştiriniz.



Görsel 1.28: Notebook içine veri seti ekleme

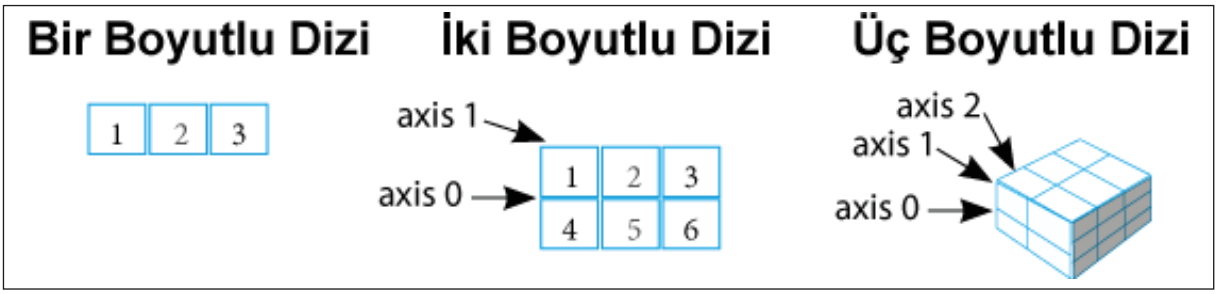
SIRA SİZDE

Küçük gruplar oluşturarak belirlediğiniz bir kategoride açık veri setlerinin bulunduğu platformlardan veri seti indirme işlemini gerçekleştiriniz. İndirdiğiniz veri setini inceleyerek hakkındaki bilgileri diğer gruplar ile paylaşınız.

1.2.3. NumPy Kütüphanesi

NumPy (Numerical Python-Sayısal Python) açık kaynak kodlu bilimsel hesaplamalar için kullanılan bir Python kütüphanesidir. NumPy kütüphanesinin temelinde ndarray adından diziler bulunur. Bu diziler Python programla dilinde kullanılan dizilerden daha hızlı ve işlevseldir.

Görsel 1.29'da farklı boyutlarda NumPy dizileri gösterilmiştir.



Görsel 1.29: NumPy dizileri

6. UYGULAMA

NumPy kütüphanesi import etme ve NumPy dizisi oluşturma işlemi gerçekleştirmek için aşağıdaki işlem adımlarını takip ediniz.

1. Adım: Jupyter Notebook uygulamasını açınız. Açılan Notebook uygulamasındaki ilk hücreye aşağıdaki kodları yazarak NumPy kütüphanesini import ediniz.

NOT !!! NumPy kütüphanesi genellikle np takma adı ile import edilir.

```
import numpy as np
```

2. Adım: NumPy dizisi oluşturmak için aşağıdaki kodlamaları yapınız.

```
np.array([1,2,3])
```

3. Adım: Bir NumPy dizisini atama işlemi için aşağıdaki kodlamaları yapınız.

```
a=np.array([1,2,3])
```

NumPy dizilerinin avantajı içindeki verilere göre kendisini tek tip olarak ayarlamasıdır. Aşağıdaki örnekte dizi içine eklenen bir ondalık sayı, tam sayı olan diğer elemanları da ondalık sayı hâline dönüştürmektedir.

```
dizi=np.array([1,2,3,3.14])
dizi
```

NumPy dizilerini tanımlama sırasında dizideki tüm elemanları tek tipe dönüştürme işlemi gerçekleştirilebilir.

```
dizi=np.array([1,2,3,3.14],dtype=int)
dizi
```

7. UYGULAMA

1, 2 ve 3 boyutlu dizi oluşturma işlemi gerçekleştirmek için aşağıdaki işlem adımlarını takip ediniz.

1. Adım: NumPy kütüphanesini import ettikten sonra aşağıdaki kodları yazarak bir boyutlu dizi oluşturunuz.

```
birboyut=np.array([1,2,3])
birboyut
```

2. Adım: İki boyutlu dizi oluşturmak için aşağıdaki kodlamaları yapınız.

```
ikiboyut = np.array([[1,2,3], [4,5,6]])
ikiboyut
```

3. Adım: Üç boyutlu dizi oluşturmak için aşağıdaki kodlamaları yapınız.

```
ucboyut= np.array([[[1,2,3], [4,5,6]], [[1,2,3], [4,5,6]]])
ucboyut
```

- **NumPy İlk Yer Tutucular**

Dizinin elemanları başlangıçta çoğu zaman bilinmez. NumPy'da daha sonra doldurmak amacıyla yer tutucular kullanılır. NumPy yer tutucu içeriği olan diziler oluşturmak için çeşitli işlevler sunar.

8. UYGULAMA

Farklı şekillerde NumPy yer tutucu dizisi oluşturma işlemlerini gerçekleştirmek için aşağıdaki işlem adımlarını takip ediniz.

1. Adım: Aşağıdaki kodlarda NumPy **zeros** fonksiyonunu kullanarak sıfırlardan oluşan NumPy dizisi oluşturunuz.

```
np.zeros(5)
```

```
np.zeros((4,3))
```

```
np.zeros((4,3),dtype=int)
```

2. Adım: Aşağıdaki kodlarda NumPy **ones** fonksiyonunu kullanarak birlerden oluşan NumPy dizisi oluşturunuz.

```
np.ones(5)
```

```
np.ones((4,3))
```

```
np.ones((4,3),dtype=int)
```

3. Adım: Aşağıdaki kodlarda NumPy **full** fonksiyonunu kullanarak belirlenen sayıdan oluşan NumPy dizisi oluşturunuz.

```
np.full((4,3),10)
```

4. Adım: Aşağıdaki kodlarda NumPy **arrange** fonksiyonunu kullanarak başlangıç, bitiş ve artış değerlerinden oluşan NumPy dizisi oluşturunuz.

```
np.arange(1,30,2)
```

5. Adım: Aşağıdaki kodlarda NumPy **linspace** fonksiyonunu kullanarak belirli bir aralıkta eşit aralıklı sayılardan oluşan NumPy dizisi oluşturunuz.

```
np.linspace(0, 10, num=5)
```

6. Adım: Aşağıdaki kodlarda NumPy **random** fonksiyonunu kullanarak belirli bir aralıkta rastgele sayılardan oluşan NumPy dizisi oluşturunuz.

```
np.random.randint(0,10,5)
```

```
np.random.randint(0,10,(4,3))
```

- **NumPy Dizi İncelemesi**

NumPy dizi incelemesi temel olarak dizinin şeklini, boyutunu, elemanlarının boyutunu öğrenme işlemleridir. Oluşturulan NumPy dizisi hakkından detaylı bilgi edinmek için kullanılan özellikler şunlardır:

- shape
- ndim
- size
- dtype
- itemsize

9. UYGULAMA

NumPy dizilerinin incelenmesinde kullanılan özelliklerle işlem yapmak için aşağıdaki işlem adımlarını takip ediniz.

1. Adım: NumPy shape özelliğini kullanarak bir dizinin her bir boyutundaki eleman sayısını öğrenme ve diziyi yeniden boyutlandırma işlemini gerçekleştiriniz.

```
a=np.array([[1,2,3],[4,5,6]])
a.shape
```

2. Adım: NumPy ndim özelliğini kullanarak bir dizinin boyutunu öğrenme işlemini gerçekleştiriniz.

```
b=np.array([1,2,3,4,5])
b.ndim
```

```
c=np.array([[1,2,3,4],[5,6,7,8]])
c.ndim
```

3. Adım: NumPy size özelliğini kullanarak bir dizinin toplam eleman sayısını öğrenme işlemini gerçekleştiriniz.

```
d=np.array([1,2,3,4,5])
d.size
```

4. Adım: NumPy dtype özelliğini kullanarak bir dizinin elemanlarının veri tipini öğrenme işlemini gerçekleştiriniz.

NOT !!! NumPy dizisinde tüm elemanlar aynı veri tipine sahiptir.

```
e= np.array([1,2,3,4,5])
e.dtype
```

```
f=np.array([1.1,2.2,3.3])
f.dtype
```

5. Adım: NumPy itemsize özelliğini kullanarak bir dizinin her bir elemanının bayt cinsinden değerini öğrenme işlemini gerçekleştiriniz.

```
g=np.array([1,2,3])
g.itemsize
```

- **NumPy İndeksleme ve Dilimleme**

NumPy dizilerinin elemanları; seçme, indeksleme veya dilimleme yoluyla erişilebilir ve değiştirilebilir.

İndeksleme, bir dizinin elemanlarına erişmek için kullanılır. NumPy dizilerinin indeks numaraları 0 ile başlar. Bir boyutlu dizilerde indeks numarası tek olurken çok boyutlu dizilerde satır, sütun veya düzlem indeks numaraları kullanılır.

10. UYGULAMA

Oluşturulan NumPy dizilerinde indeksleme işlemleri yapmak için diğer sayfadaki işlem adımlarını takip ediniz.

1. Adım: Bir boyutlu NumPy dizisi oluşturarak dizi elemanlarına erişim için indeksleme işlemlerini yapınız.

```
a = np.array([1,2,3,4])
a[0]
a[1]
a[2]=99
```

2. Adım: İki boyutlu NumPy dizisi oluşturarak dizi elemanlarına erişim için indeksleme işlemlerini yapınız.

```
b= np.array([[1,2,3], [4,5,6], [7,8,9]])
b[2][1]
b[2,1]
b[1,2]=99
```

3. Adım: Üç boyutlu NumPy dizisi oluşturarak dizi elemanlarına erişim için indeksleme işlemlerini yapınız.

```
c = np.array([[[1,2,3], [4,5,6]], [[7,8,9], [10,11,12]]])
c[0, 1, 2]
c[1, 1, 1]=99
```

4. Adım: NumPy dizilerinde negatif indeks numarası kullanarak dizinin sonundan itibaren seçme işlemi yapınız.

```
a[-2]
b[-1,-2]
```

Dilimleme, NumPy dizisinde bir indeks numarasından diğer indeks numarasına kadar dizi elemanları seçme işlemidir. Kullanımı **[başlangıç : bitiş]** veya **[başlangıç : bitiş : adım]** şeklindedir. Başlangıç değeri belirtilmezse indeks numarası 0 olarak kabul edilir. Bitiş değeri belirtilmezse indeks numarası dizinin eleman sayısı olarak kabul edilir. Adım değeri belirtilmezse 1 olarak kabul edilir.

11. UYGULAMA

Oluşturulan bir boyutlu NumPy dizisinden dilimleme ile dizi elemanlarını seçme işlemi gerçekleştirmek için aşağıdaki işlem adımlarını takip ediniz.

1. Adım: 10 ile 30 arasında sayılardan oluşan bir NumPy dizisi oluşturunuz.

```
x=np.arange(10,30)
```

2. Adım: Oluşturulan dizinin 0 ile 4 indeks numarası arasındaki elemanlara erişim işlemini yapınız.

```
x[0:4]
x[:4]
```

3. Adım: Oluşturulan dizinin 4 indeks numaralı elemanından son elemana kadar erişim işlemini yapınız.

```
x[4:]
```

4. Adım: Oluşturulan dizinin 0 indeks numaralı elemanından başlayarak indeks numarası ikişer artarak 10 indeks numaralı elemana kadar erişim işlemini yapınız.

```
x[0:10:2]
```

12. UYGULAMA

Oluşturulan iki boyutlu NumPy dizisinden dilimleme ile dizi elemanlarını seçme işlemi gerçekleştirmek için aşağıdaki işlem adımlarını takip ediniz.

1. Adım: İki boyutlu bir dizi oluşturunuz.

```
y=np.random.randint(0,10,(4,4))
```

2. Adım: Oluşturulan dizinin ikinci satırının tüm sütunlarını seçme işlemini yapınız.

```
 y[1,:]
```

3. Adım: Oluşturulan dizinin ikinci sütununun tüm satırlarını seçme işlemini yapınız.

```
y[:,1]
```

4. Adım: Oluşturulan dizinin ilk iki satırı ile ilk üç sütununu seçme işlemini yapınız.

```
y[0:2,0:3]
```

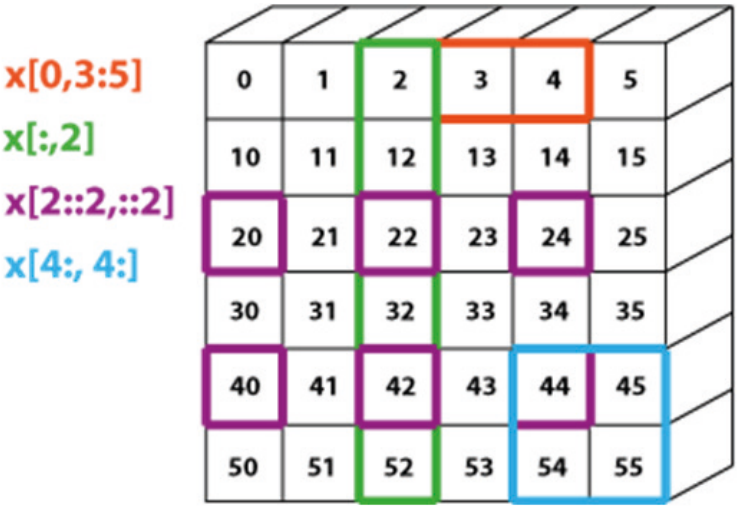
5. Adım: Oluşturulan dizinin ilk iki sütununu seçme işlemini yapınız.

```
y[:,0:2]
```

6. Adım: Oluşturulan dizide 4 sayısından büyük değere sahip olan dizi elemanları için mantıksal seçim işlemini yapınız.

```
y[y>4]
```

Görsel 1.30'da farklı dilimleme ve indeksleme tekniklerine kısa bir genel bakış gösterilmektedir.



Görsel 1.30: NumPy dilimleme ve indeksleme

- **NumPy Yeniden Boyutlandırma**

Reshape (yeniden şekillendirme) NumPy dizisini yeniden boyutlandırmak için kullanılan bir fonksiyondur.

13. UYGULAMA

Bir boyutlu NumPy dizisini yeniden boyutlandırma ile iki boyutlu hâle getirme işlemini gerçekleştirmek için aşağıdaki işlem adımlarını takip ediniz.

1. Adım: Aşağıdaki kodlarda bir boyutlu NumPy dizisi oluşturma işlemini gerçekleştiriniz.

```
dizi=np.arange(1,10)
dizi
dizi.ndim
```

2. Adım: Oluşturulan diziyi iki boyutlu hâle dönüştürmek için kodlamaları yapınız.

```
yenidizi2d=dizi.reshape(3,3)
yenidizi2d
```

3. Adım: Oluşturulan diziyi üç boyutlu hâle dönüştürmek için kodlamaları yapınız.

```
yenidizi3d=dizi.reshape(3,3,1)
yenidizi3d
```

4. Adım: Oluşturulan üç boyutlu veya iki boyutlu bir diziyi bir boyutlu hâle dönüştürmek için kodlamaları yapınız.

```
yenidizi3d.reshape(-1)
yenidizi2d.reshape(-1)
```

- **NumPy Dizileri Birleştirme, Ayrıştırma ve Sıralama**

Concatenation (birleştirme) aynı şekle sahip iki veya daha fazla NumPy dizisini birleştirmek için kullanılır.

14. UYGULAMA

İki NumPy dizisini birleştirme işlemini gerçekleştirmek için aşağıdaki işlem adımlarını takip ediniz.

1. Adım: Aşağıdaki kodlarda bir boyutlu NumPy dizisi oluşturma işlemini gerçekleştiriniz.

```
x=np.array([1,2,3])
y=np.array([4,5,6])
t=np.concatenate([x,y])
```

2. Adım: Aşağıdaki kodlamalarda iki adet NumPy dizisi oluşturarak bu dizileri birleştirme işlemini gerçekleştiriniz.

```
z=np.array([7,8,9])
t=np.concatenate([t,z])
```

3. Adım: Aşağıdaki kodlamalarda iki boyutlu NumPy dizilerini birleştirme işlemini gerçekleştiriniz.

```
a=np.array([[1,2,3],[4,5,6]])
b=np.array([[7,8,9],[10,11,12]])
c=np.concatenate([a,b])
```

4. Adım: Aşağıdaki kodlamalarda iki boyutlu NumPy dizilerini sütun bazında birleştirme işlemini gerçekleştiriniz.

NOT !!! axis değeri 0 ise satırları, 1 ise sütunları temsil eder. Ön tanımlı olarak 0 değerindedir.

```
d=np.concatenate([a,b],axis=1)
d
```

Split (ayrıştırma) fonksiyonu bir NumPy dizisini birden çok alt diziye bölmek için kullanılır. Üç parametre ile kullanılır. İlk parametre ayrıştırılacak olan diziyi, ikinci parametre hangi indeks numaralı elemanların ayrıştırılacağını, üçüncü parametre ise satır veya sütun bazında bölümleme yapılacağını belirtir.

15. UYGULAMA

NumPy dizilerini ayrıştırma işlemlerini gerçekleştirmek için aşağıdaki işlem adımlarını takip ediniz.

1. Adım: Aşağıdaki kodlamalarda bir NumPy dizisi oluşturunuz. Split fonksiyonu ile oluşturulan diziyi üç parçaya ayırınız.

```
x=np.array([11,12,13,14,15,16,17,18,19])
np.split(x,3)
```

2. Adım: Aşağıdaki kodlarda indeks numaralarını kullanarak diziyi belirli aralıkta ayrıştırma işlemini gerçekleştiriniz.

```
np.split(x,[2,6])
```

3. Adım: Aşağıdaki kodlarda ayrıştırılan alt dizileri farklı dizilere aktarma işlemini gerçekleştiriniz.

```
a,b,c=np.split(x,[2,6])
a
b
c
```

4. Adım: Aşağıdaki kodlamalarda iki boyutlu bir dizi oluşturarak oluşturulan bu dizinin ayrıştırma işlemini gerçekleştiriniz.

```
y=np.arange(16).reshape(4,4)
y
```

5. Adım: Aşağıdaki kodlarda ayrıştırılan iki boyutlu diziyi satır bazında alt dizilere aktarma işlemini gerçekleştiriniz.

```
d,e=np.vsplit(y,2)
d
e
```

6. Adım: Aşağıdaki kodlarda ayrıştırılan iki boyutlu diziyi sütun bazında alt dizilere aktarma işlemini gerçekleştiriniz.

```
f,g=np.hsplit(y,2)
f
g
```

Sort (sıralama) fonksiyonu bir NumPy dizisini sıralamak için kullanılır. Sıralama dizi elemanlarında sayısal veya alfabetik olarak gerçekleşir.

16. UYGULAMA

Bir NumPy dizisinin elemanlarını sıralama işlemlerini gerçekleştirmek için aşağıdaki işlem adımlarını takip ediniz.

1. Adım: Aşağıdaki kodlarda 0 ile 100 arasında rastgele 10 sayıdan oluşan bir boyutlu NumPy dizisi oluşturma işlemini gerçekleştiriniz.

```
x=np.random.randint(1,100,10)
```

2. Adım: Aşağıdaki kodlarda oluşturulan diziyi sıralama işlemini gerçekleştiriniz.

```
np.sort(x)
```

3. Adım: Sıralama işleminin kalıcı olması için aşağıdaki kodlamaları yapınız.

```
x.sort()
```

4. Adım: İki boyutlu dizi sıralaması için aşağıdaki kodlamaları yazınız.

```
y=np.random.randint(1,100,[4,4])
```

5. Adım: Oluşturulan diziyi satır bazında sıralamak için aşağıdaki kodlamaları yazınız.

```
np.sort(y,axis=1)
```

6. Adım: Oluşturulan diziyi sütun bazında sıralamak için aşağıdaki kodlamaları yazınız.

```
np.sort(y,axis=0)
```

- **NumPy Operatörler, Matematiksel ve İstatistiksel Fonksiyonlar**

NumPy dizilerinde aritmetik ve mantıksal operatörlerin kullanımı ile diziler üzerinde aritmetik ve mantıksal işlemler gerçekleştirilebilir.

17. UYGULAMA

NumPy dizileri üzerinde aritmetiksel ve mantıksal operatörlerin kullanım işlemlerini yapmak için aşağıdaki işlem adımlarını takip ediniz.

1. Adım: Bir NumPy dizisi oluşturunuz.

```
x=np.arange(1,11)
x
```

2. Adım: Oluşturulan dizide aşağıdaki aritmetiksel operatörleri kullanarak işlem yapınız.

```
x+5
x-10
x/2
x*3
x**2
```

3. Adım: Oluşturulan dizide belirtilen koşula göre elemanları alma işlemini yapınız.

```
x[x>4]
x[x<=7]
x[x==5]
```

NumPy matematiksel problemleri çözebilecek çok sayıda fonksiyon bulundurur. Bu matematiksel fonksiyonlar trigonometrik, aritmetik, logaritmik vb. fonksiyonları içerir.

18. UYGULAMA

Aritmetik operatör kullanmak yerine matematiksel fonksiyonlar kullanarak işlemler yapmak için aşağıdaki işlem adımlarını takip ediniz.

1. Adım: İki adet NumPy dizisi oluşturunuz.

```
a=np.array([10,20,30])
b=np.array([1,2,3])
```

2. Adım: NumPy add fonksiyonunu kullanarak toplama işlemlerini yapınız.

```
np.add(a,5)
np.add(a,b)
```

3. Adım: NumPy subtract fonksiyonunu kullanarak çıkarma işlemlerini yapınız.

```
np.subtract(a,5)
np.subtract(a,b)
```

4. Adım: NumPy multiply fonksiyonunu kullanarak çarpma işlemlerini yapınız.

```
np.multiply(a,5)
np.multiply(a,b)
```

5. Adım: NumPy power fonksiyonunu kullanarak üs alma işlemlerini yapınız.

```
np.power(a,2)
np.power(a,b)
```

19. UYGULAMA

NumPy trigonometrik fonksiyonları ile işlemler yapmak için aşağıdaki işlem adımlarını takip ediniz.

1. Adım: Değerleri verilen bir NumPy dizisi oluşturunuz.

```
c = np.array([0,30,60,90,120,150,180])
```

2. Adım: Oluşturulan dizinin sinüs değerlerini bulma işlemini yapınız.

```
np.sin(c*np.pi/180)
```

3. Adım: Oluşturulan dizinin kosinüs değerlerini bulma işlemini yapınız.

```
np.cos(c*np.pi/180)
```

4. Adım: Oluşturulan dizinin tanjant değerlerini bulma işlemini yapınız.

```
np.tan(c*np.pi/180)
```

NumPy dizileri içindeki verilerin analiz edilmesi için istatistiksel fonksiyonlar kullanılır. Bu fonksiyonlar bir dizi içindeki en büyük, en küçük değerleri bulma veya standart sapma, varyans gibi temel istatistiksel fonksiyonlardır.

20. UYGULAMA

Bir dersten sınava giren on öğrencinin üç dersten aldıkları notlar üzerinden NumPy istatistiksel fonksiyonlarının kullanım işlemini yapmak için aşağıdaki işlem adımlarını takip ediniz.

1. Adım: 0 ile 100 arasında rastgele sayılardan oluşan 10x3 bir NumPy dizisi oluşturunuz.

```
x=np.random.randint(0,101,([10,3]))
```

2. Adım: Oluşturulan dizinin elemanları içinde en büyük ve en küçük değerleri bulmak için max ve min fonksiyonlarını kullanınız.

```
x.max()
x.min()
```

3. Adım: Oluşturulan dizinin tamamının veya belirtilen eksenlerinin toplamını bulmak için sum fonksiyonunu kullanınız.

```
x.sum()
x.sum(axis=0)
x.sum(axis=1)
```

4. Adım: Oluşturulan dizinin elemanları içinde belirli eksende en büyük ve en küçük değerleri bulmak için max ve min fonksiyonlarını kullanınız.

```
x.max(axis=0)
x.max(axis=1)
x.min(axis=0)
x.min(axis=1)
```

5. Adım: Oluşturulan dizinin tamamının ve belirtilen eksenlerinin ortalamasını bulmak için mean fonksiyonunu kullanınız.

```
x.mean()
x.mean(axis=0)
x.mean(axis=1)
```

6. Adım: Oluşturulan dizinin ortalamaya göre ne kadar değişkenlik gösteren varyansını bulmak için var fonksiyonunu kullanınız.

```
x.var()
x.var(axis=0)
x.var(axis=1)
```

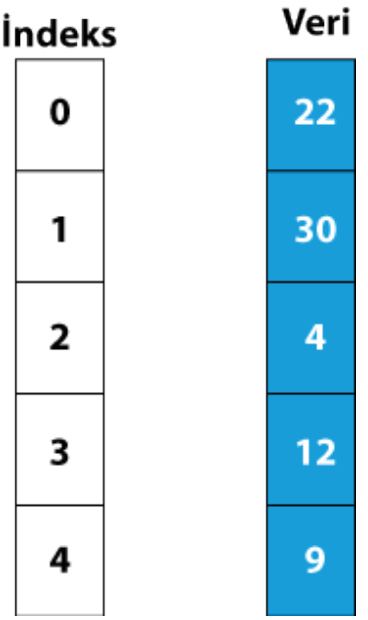
7. Adım: Oluşturulan dizinin ortalamaya göre ne kadar saptığını gösteren standart sapmasını bulmak için std fonksiyonunu kullanınız.

```
x.std()
x.std(axis=0)
x.std(axis=1)
```

1.2.4. Pandas Kütüphanesi

Pandas veri analizi için kullanılan açık kaynak kodlu bir Python kütüphanesidir. Pandas kütüphanesi CVS dosyaları, Excel dosyaları veya veri tabanı tabloları gibi yapılandırılmış verilerle işlem yapmak için kullanışlıdır. Pandas temel olarak Seriler ve DataFrame olmak üzere iki veri tipi kullanır.

Seriler, bir boyutlu etiketli indekslenmiş verilerdir. Seri içindeki veriler aynı türe sahiptir. Görsel 1.31'de bir serinin yapısı gösterilmiştir.



Görsel 1.31: Pandas serisinin yapısı

21. UYGULAMA

Pandas kütüphanesini yükleme ve Pandas serisi oluşturma işlemlerini yapmak için aşağıdaki işlem adımlarını takip ediniz.

1. Adım: Pandas kütüphanesini yükleme işlemini yapınız.

NOT !!! NumPy dizisinde tüm elemanlar aynı veri tipine sahiptir.

```
import pandas as pd
```

2. Adım: Pandas serisine eklemek için bir boyutlu dizi oluşturunuz.

```
data = ["Merve","Yusuf", "Ayşe","Zehra"]
```

3. Adım: Oluşturulan diziden Pandas kütüphanesini kullanarak bir seri oluşturunuz.

```
s1 = pd.Series(data)
s1
```

4. Adım: Sayılardan oluşan bir Pandas serisi oluşturunuz.

```
s2=pd.Series([1,2,3,4])
s2
```

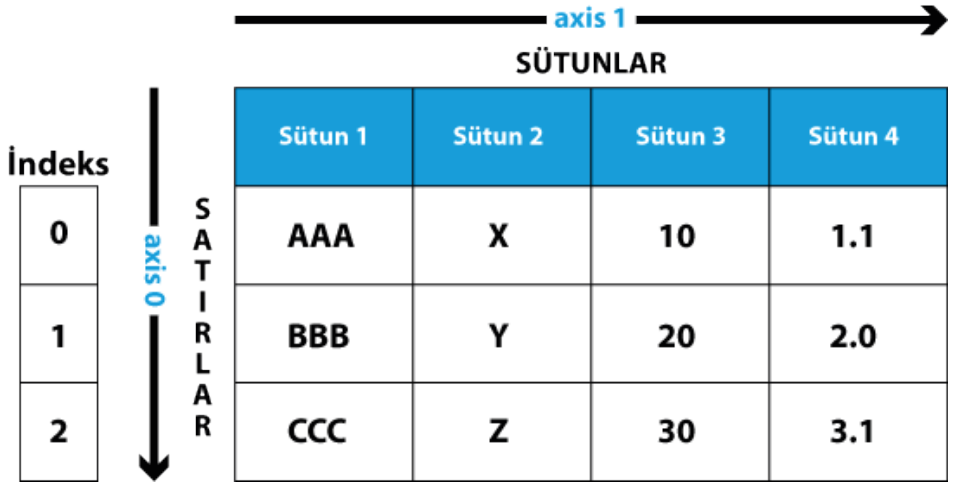
5. Adım: Karakterlerden oluşan bir NumPy dizisinden Pandas serisi oluşturunuz.

```
dizi = np.array(['b','i','l','i','ş','i','m'])
s3=pd.Series(dizi)
s3
```

6. Adım: Python sözlüğü üzerinden seri oluşturma işlemini yapınız.

```
dict={"tr":"Türkiye","fr":"Fransa","de":"Almanya"}
s4=pd.Series(dict)
s4
```

Pandas DataFrame, sütun ve satırlardan oluşan iki boyutlu veri tablolarıdır. Görsel 1.32'de bir DataFrame yapısı gösterilmiştir.



Görsel 1.32: DataFrame yapısı

22. UYGULAMA

Pandas DataFrame oluşturma işlemleri yapmak için aşağıdaki işlem adımlarını takip ediniz.

1. Adım: Bir Python listesi oluşturunuz.

```
liste=[1,2,3,"a","b","c"]
```

2. Adım: Oluşturulan listeyi kullanarak bir Pandas DataFrame oluşturunuz.

```
df=pd.DataFrame(liste)
```

3. Adım: Oluşturulan DataFrame'de sütun için etiket oluşturma işlemini yapınız.

```
df=pd.DataFrame(liste, columns=["Veriler"])
```

4. Adım: İki boyutlu bir liste oluşturunuz.

```
data = [['Ali',10],['Ayşe',12],['Mehmet',13]]
```

5. Adım: Oluşturulan iki boyutlu listeden bir DataFrame oluşturma işlemini yapınız.

```
df = pd.DataFrame(data,columns=['Ad','Yaş'])
```

6. Adım: Bir liste sözlüğünden DataFrame oluşturma işlemini yapınız.

```
data = {'Ad':['Ali', 'Ayşe', 'Mehmet'],'Yaş':[10,12,13]}
df = pd.DataFrame(data)
df
```

7. Adım: DataFrame indeks değerlerini değiştirme işlemini yapınız.

```
df = pd.DataFrame(data,index=['Ogrenci1','Ogrenci2',
     'Ogrenci3'])
df
```

- **Pandas Seri / DataFrame İncelemesi**

Oluşturulan Seri / DataFrame hakkında bilgi alabilmek için veriler üzerinde inceleme işlemleri gerçekleştirilir.

23. UYGULAMA

Pandas Seri / DataFrame inceleme işlemleri yapmak için aşağıdaki işlem adımlarını takip ediniz.

1. Adım: 1 ile 100 arasında rastgele sayılardan oluşan iki boyutlu bir NumPy dizisi oluşturunuz.

```
data=np.random.randint(1,100,size=(3,3))
```

2. Adım: Oluşturulan NumPy dizisinden Pandas DataFrame oluşturunuz.

```
df=pd.DataFrame(data, columns=["x1","x2","x3"])
```

3. Adım: Oluşturulan DataFrame'in indeks, veri türü ve bellek bilgilerini görüntüleme işlemlerini yapınız.

```
df.info()
```

4. Adım: Oluşturulan DataFrame'in satır ve sütun bilgilerini görüntüleme işlemlerini yapınız.

```
df.axes
```

5. Adım: Oluşturulan DataFrame'in boyut bilgilerini görüntüleme işlemini yapınız.

```
df.shape
```

6. Adım: Oluşturulan DataFrame'in boyutunu görüntüleme işlemini yapınız.

```
df.ndim
```

7. Adım: Oluşturulan DataFrame'in eleman sayısını görüntüleme işlemini yapınız.

```
df.size
```

8. Adım: Oluşturulan DataFrame'in içindeki eleman değerlerini görüntüleme işlemini yapınız.

```
df.values
```

9. Adım: Oluşturulan DataFrame'in ilk 5 satırını görüntüleme işlemini yapınız.

```
df.head()
```

10. Adım: Oluşturulan DataFrame'in son 5 satırını görüntüleme işlemini yapınız.

```
df.tail()
```

- **Pandas Seri / DataFrame Seçimi**

Pandas, verilerin seçim işleminde esnek birçok yöntem sunar. Seçme işlemi ile veriler içinden istenilen kısımlar alınarak gösterilebilir veya yeni bir veri yapısı oluşturulabilir.

24. UYGULAMA

Pandas DataFrame'lerin elemanları üzerinde seçme işlemini yapmak için aşağıdaki işlem adımlarını takip ediniz.

1. Adım: Dört adet 0 ile 100 arasında rastgele sayılardan oluşan NumPy dizilerini oluşturunuz.

```
d1=np.random.randint(0,100,size=10)
d2=np.random.randint(0,100,size=10)
d3=np.random.randint(0,100,size=10)
d4=np.random.randint(0,100,size=10)
```

2. Adım: Oluşturulan NumPy dizilerinden bir Python sözlüğü oluşturunuz.

```
dict={"seri1":d1,"seri2":d2,"seri3":3,"seri4":d4}
```

3. Adım: Oluşturulan sözlükten bir Pandas DataFrame oluşturunuz.

```
df=pd.DataFrame(dict)
```

4. Adım: Oluşturulan DataFrame'in 2 indeks numaralı satırından başlayarak 4 indeks numaralı satıra kadar olan verileri seçme işlemini yapınız.

```
df[2:4]
```

5. Adım: Oluşturulan DataFrame'in ilk sütununu seçme işlemini yapınız.

```
df["seri1"]
```

6. Adım: DataFrame'in ilk sütununu özellik olarak seçme işlemini yapınız.

```
df.seri1
```

7. Adım: Oluşturulan DataFrame'in birden fazla sütununu seçme işlemini yapınız.

```
df[["seri1","seri4"]]
```

8. Adım: Oluşturulan DataFrame'de iloc komutunu kullanarak konuma göre satır seçme işlemini yapınız.

```
df.iloc[0]
df.iloc[1]
df.iloc[-1]
df.iloc[0:3]
```

9. Adım: Oluşturulan DataFrame'de iloc komutunu kullanarak konuma göre sütun seçme işlemini yapınız.

```
df.iloc[:,0]
df.iloc[:,1]
df.iloc[:,-1]
```

10. Adım: Oluşturulan DataFrame'de iloc komutunu kullanarak konuma göre hem satır hem de sütun seçme işlemini yapınız.

```
df.iloc[0:3]
df.iloc[:, 0:2]
df.iloc[[0,2,6,8], [0,2,3]]
df.iloc[0:5, 2:4]
```

11. Adım: Oluşturulan DataFrame'de loc komutunu kullanarak satır seçme işlemini yapınız.

```
df.loc[0]
df.loc[1]
df.loc[0:3]
```

12. Adım: Oluşturulan DataFrame'de loc komutunu kullanarak etikete göre sütun seçme işlemini yapınız.

```
df.loc[:,"seri1"]
```

13. Adım: Oluşturulan DataFrame'de loc komutunu kullanarak hem satır hem de sütun seçme işlemini yapınız.

```
df.loc[2:5,"seri1":"seri3"]
```

- **Pandas Seri / DataFrame Birleştirme**

Pandas iki veya daha fazla Seri / DataFrame birleştirmek için merge, join ve concat metotlarını kullanır. Bu metotların kullanım farklılıkları şunlardır:

- concat (), DataFrame'leri satırlar veya sütunlar arasında birleştirmek için kullanılır.
- merge (), ortak sütunlar veya indeksler üzerindeki verileri birleştirmek için kullanılır.
- join (), bir anahtar sütun veya bir indeks üzerinden verileri birleştirmek için kullanılır.

25. UYGULAMA

Rastgele sayılardan oluşturulan iki DataFrame nesnesini concat metodunu kullanarak birleştirme işlemi yapmak için aşağıdaki işlem adımlarını takip ediniz.

1. Adım: İki adet rastgele sayılardan oluşan 4x3'lük NumPy dizisi oluşturunuz.

```
rastgele1=np.random.randint(1,20,size=(4,3))
rastgele2=np.random.randint(20,40,size=(4,3))
```

2. Adım: Oluşturulan NumPy dizilerinden DataFrame oluşturunuz ve sütun adlarını değiştiriniz.

```
df1=pd.DataFrame(rastgele1,columns=["x","y","z"])
df2=pd.DataFrame(rastgele2,columns=["x","y","z"])
```

3. Adım: Oluşturulan DataFrame nesnelerini concat metodunu kullanarak birleştirme işlemini yapınız.

```
df=pd.concat([df1,df2])
df
```

4. Adım: Birleştirme işlemi sonucunda indeks numaralarını düzenlemek için concat metodunun ignore_index parametresini kullanınız.

```
df=pd.concat([df1,df2], ignore_index=True)
df
```

5. Adım: Farklı sütun ismine sahip olan DataFrame nesnelerini birleştirme işlemi için oluşturulan ikinci DataFrame nesnesinin sütun ismini değiştiriniz.

```
df2.columns=["x","y","a"]
df=pd.concat([df1,df2],ignore_index=True)
df
```

6. Adım: İki DataFrame nesnesinde ortak olan sütun adlarına göre birleştirme işlemi için concat metodunun join parametresini kullanınız.

```
df=pd.concat([df1,df2],ignore_index=True, join="inner")
df
```

26. UYGULAMA

İki DataFrame nesnesini merge metodunu kullanarak birleştirme işlemi yapmak için aşağıdaki ve diğer sayfadaki işlem adımlarını takip ediniz.

1. Adım: İki adet DataFrame oluşturunuz.

```
sol = pd.DataFrame({
        "anahtar": ["A0", "A1", "A2", "A3"],
        "X": ["X0", "X1", "X2", "X3"],
        "Y": ["Y0", "Y1", "Y2", "Y3"],
    })
```

```
sag = pd.DataFrame({
        "anahtar": ["A0", "A1", "A2", "A3"],
        "Z": ["Z0", "Z1", "Z2", "Z3"],
        "T": ["T0", "T1", "T2", "T3"],
    })
```

2. Adım: Oluşturulan DataFrame nesnelerinin anahtar adındaki sütun üzerinden merge metodunu kullanarak birleştirme işlemini yapınız.

```
birlesim = pd.merge(sol, sag,on="anahtar")
birlesim
```

3. Adım: Oluşturulan sag isimli DataFrame nesnesini anahtar sütunundaki A3 değerini A4 olarak değiştirip birleştirme işlemini tekrar yaparak birleştirme sonucunu gözlemleyiniz.

```
sag.iloc[[3],[0]]="A4"
birlesim = pd.merge(sol, sag,on="anahtar")
birlesim
```

4. Adım: Birleştirme işleminde anahtar sütundaki bilgiler üzerinden soldaki DataFrame nesnesine göre how argümanı ile yapınız.

```
birlesim = pd.merge(sol, sag,on="anahtar",how="left")
birlesim
```

5. Adım: Birleştirme işlemini anahtar sütundaki bilgiler üzerinden sağdaki DataFrame nesnesine göre how parametresi ile yapınız.

```
birlesim = pd.merge(sol, sag,on="anahtar",how="right")
birlesim
```

27. UYGULAMA

Join metodunu kullanarak birleştirme işlemi yapmak için aşağıdaki ve diğer sayfadaki işlem adımlarını takip ediniz.

1. Adım: İki adet DataFrame oluşturunuz.

```
df1 = pd.DataFrame(
    {"A": ["A0", "A1", "A2"], "B": ["B0","B1","B2"]},
     index=["K0", "K1", "K2"])
df1
```

```
df2 = pd.DataFrame(
    {"C": ["C0", "C2", "C3"], "D": ["D0","D2","D3"]},
     index=["K0", "K2", "K3"])
df2
```

2. Adım: Oluşturulan DataFrame nesnelerinin birleştirme işlemini join metodunu kullanarak yapınız.

```
birlesim = df1.join(df2)
birlesim
```

3. Adım: Oluşturulan DataFrame nesnelerini join metodunun how=”outer” parametresini kullanarak birleştirme işlemini yapınız.

```
birlesim = df1.join(df2, how="outer")
birlesim
```

4. Adım: Pandas how parametresinin değerini inner olarak değiştirerek sonuçları gözlemleyiniz.

```
birlesim = df1.join(df2, how="inner")
birlesim
```

- ## Verileri İçe ve Dışa Aktarma

Farklı veri kaynaklarındaki verileri Pandas DataFrame içine aktarmak için Pandas kütüphanesi içinde bir dizi okuyucu fonksiyonu bulunur. Tablo 1.2’de okuma işlemi için kullanılan fonksiyonlar gösterilmiştir.

Tablo 1.2: Pandas Okuyucu Fonksiyonları

Fonksiyon Adı	İşlevi
pd.read_csv(dosya adı)	CSV dosyasındaki verileri içe aktarır.
pd.read_excel(dosya adı)	Excel dosyasındaki verileri içe aktarır.
pd.read_sql(sorgu, bağlantı)	SQL tablo veya veri tabanından verileri okur.
pd.read_json(json dizisi)	URL adresi veya dosyadan json formatında verileri okur.
pd.read_html(url)	HTML tablosundaki verileri içe aktarır.

Bir DataFrame içindeki verileri csv, excel, sql veya json olarak dışa aktarmak için Tablo 1.3’te verilen Pandas kütüphanesi fonksiyonları kullanılır.

Tablo 1.3: Pandas Yazıcı Fonksiyonları

Fonksiyon Adı	İşlevi
df.to_csv(dosya adı)	DataFrame içindeki verileri csv dosyasına aktarır.
df.to_excel(dosya adı)	DataFrame içindeki verileri excel dosyasına aktarır.
df.to_sql(tablo adı, baglantı)	DataFrame içindeki verileri sql tablosuna aktarır.
df.to_json(dosya adı)	DataFrame içindeki verileri json dosyasına aktarır.

28. UYGULAMA

CSV uzantılı bir veri setini Pandas DataFrame nesnesine dönüştürme işlemi yapmak için aşağıdaki işlem adımlarını takip ediniz.

1. Adım: Açık veri setlerinin bulunduğu depolardan indirilen veya bilgisayarınızda bulunan CSV dosyasından DataFrame oluşturunuz.

```
veri=pd.read_csv("abc.csv")
vs
```

2. Adım: Oluşturulan DataFrame nesnesinin ilk 5 satırını ve son 5 satırını görüntüleyiniz.

```
veri.head()
```

```
veri.tail()
```

3. Adım: Oluşturulan DataFrame nesnesinin kopyasını alma işlemini yapınız.

```
df=veri.copy()
```

4. Adım: Oluşturulan DataFrame nesnesinin boyutunu görüntüleyiniz.

```
df.shape
```

5. Adım: Oluşturulan DataFrame nesnesinin özetini görüntüleyiniz.

```
df.info()
```

6. Adım: Oluşturulan DataFrame nesnesi hakkındaki istatistiksel bilgileri görüntüleyiniz.

```
df.describe()
```

7. Adım: Oluşturulan DataFrame nesnesinin sütun bilgilerini görüntüleyiniz.

```
df.columns
```

8. Adım: Oluşturulan DataFrame nesnesinin belirtilen sütun bilgilerini görüntüleyiniz.

```
df[“AAAA”]
```

9. Adım: Oluşturulan DataFrame nesnesinin belirtilen satırı veya satırları kaldırma işlemini yapınız.

```
df.drop(1)
```

```
df.drop([0,1,2])
```

10. Adım: Oluşturulan DataFrame nesnesinin belirtilen sütun veya sütunları kaldırma işlemini yapınız.

```
df.drop('AAAA', axis=1)
```

29. UYGULAMA

Belirli aralıktaki Türk Hava Yollarına ait hisse senedi değerlerinden bir DataFrame nesnesi oluşturmak ve oluşturulan DataFrame nesnesini dışa aktarma işlemini yapmak için aşağıdaki işlem adımlarını takip ediniz.

1. Adım: Anaconda ortamına Anaconda Navigator veya komut istemini kullanarak pandas_datareader paketini kurunuz.

```
conda install pandas_datareader
```

2. Adım: Jupyter Notebook sayfasına gerekli kütüphaneleri dâhil ediniz.

```
import pandas as pd
import pandas_datareader.data as pdr
```

3. Adım: Çekilecek olan verilerin başlangıç ve bitiş tarihlerini belirleyiniz.

```
start_date = "2020-01-1"
end_date = "2022-5-15"
```

4. Adım: Türk Hava Yollarının hisse senedi fiyat verilerini bir DataFrame'de depolamak için datareader metodunu kullanınız.

```
df = pdr.DataReader(name="THYAO.IS", data_source='yahoo',
                    start=start_date, end=end_date)
df
```

5. Adım: Oluşturulan DataFrame'i csv dosyası olarak dışa aktarım işlemini yapınız.

```
df.to_csv('data.csv')
```

6. Adım: Oluşturulan DataFrame'i Excel dosyası olarak dışa aktarım işlemini yapınız.

```
df.to_excel('data1.xlsx')
```

7. Adım: Oluşturulan DataFrame'in sadece bir sütununu HTML dosyası olarak dışa aktarım işlemini yapınız.

```
df[["Open"]].to_html('data2.html')
```

- **Pandas Gruplama ve Kümeleme**

Pandas, groupby fonksiyonu ile verilerin gruplanmasına ve bu gruplar üzerinde işlevler yürütülmesine olanak tanır. Gruplanan verilerinin istenen özelliklerine göre kümelemek için aggregate fonksiyonu kullanılır.

30. UYGULAMA

Pandas groupby fonksiyonunu kullanarak bir DataFrame üzerinde gruplama işlemini yapmak için aşağıdaki işlem adımlarını takip ediniz.

1. Adım: Kaggle açık veri setleri sunan internet sitesi üzerinden "Iris" olarak arama yaparak veri seti dosyasını indiriniz.

2. Adım: İndirilen veri seti dosyasını Pandas DataFrame'e aktarma işlemini yapınız.

```
data=pd.read_csv('iris.csv')
data
```

3. Adım: DataFrame ile ilgili şekli, sütun sayısı, dizinleri ve diğer bilgileri anlamak için info fonksiyonunu kullanınız.

```
data.info()
```

4. Adım: Türlere göre gruplamak için groupby() yöntemini kullanarak verileri gruplandırınız.

```
data.groupby("Species")
```

5. Adım: Gruplanan türlerin sayısını öğrenmek için count fonksiyonunu kullanınız.

```
data.groupby("Species").count()
```

6. Adım: Gruplanan türlerin ortalamasını öğrenmek için mean fonksiyonunu kullanınız.

```
data.groupby("Species").mean()
```

7. Adım: Gruplanan türlerin toplamını öğrenmek için sum fonksiyonunu kullanınız.

```
data.groupby("Species").sum()
```

8. Adım: Gruplanan türlerin birden fazla özelliğine göre gruplamak için aggregate fonksiyonunu kullanınız.

```
data.groupby("Species").agg([min,max,np.std])
```

9. Adım: Gruplanan türlerin sadece belirli sütununa göre seçim yapmak için sütun kısmını seçme işlemini yapınız.

```
data.groupby("Species")["SepalLengthCm"].
aggregate([max,min])
```

1.2.5. Veri Görselleştirme

Veri analizinin en önemli kısımlarından biri, verilerdeki temel anlamı iletmek için görselleştirme oluşturmaktır. Veri görselleştirme; verileri daha iyi anlamaya yardımcı olmak, aykırı değerleri veya eksik verileri belirlemek için grafikler oluşturma işlemidir. Kitabın bu bölümünde verileri görselleştirme işlemleri için matplotlib kütüphanesi kullanılacaktır.

Matplotlib plot olarak adlandırılan iki boyutlu grafikler çizmek için kullanılan çapraz platform bir Python kütüphanesidir.

31. UYGULAMA

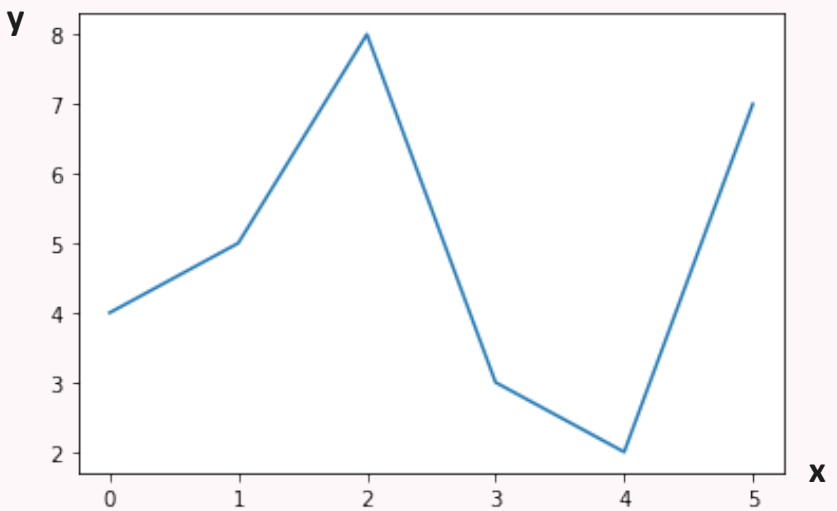
Matplotlib kütüphanesi kullanarak verilerden grafik oluşturma işlemini yapmak için aşağıdaki adımları takip ediniz.

1. Adım: Jupyter Notebook sayfasına matplotlib kütüphanesinde grafik çizdirmek için pyplot modülünü plt takma adı ile yükleyiniz.

```
import matplotlib.pyplot as plt
```

2. Adım: Sayılardan bir liste oluşturarak liste değerlerinin grafiğini oluşturunuz (Görsel 1.33'te y ekseninde liste değerleri, x ekseninde indeks numaraları gösterilmiştir.).

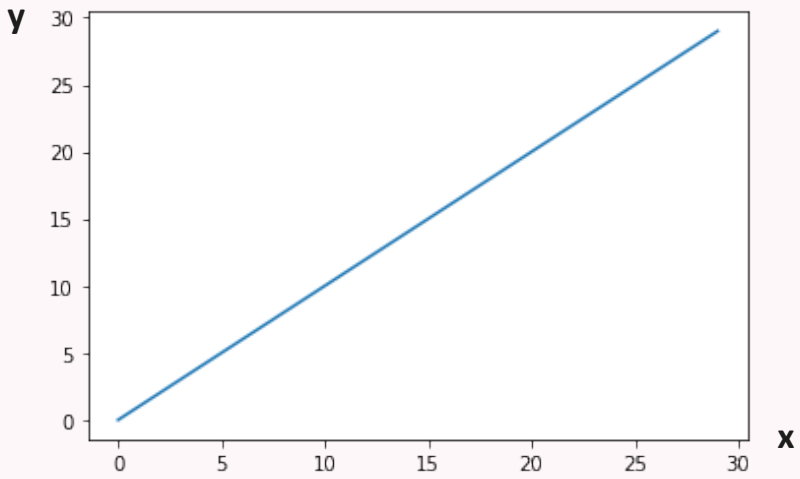
```
liste = [4, 5, 8, 3, 2, 7]
plt.plot(liste)
plt.show()
```



Görsel 1.33: Liste değerlerinin tek çizgi ile gösterilmesi

3. Adım: NumPy dizisinden oluşan verilerin grafiksel gösterimini yapınız (Görsel 1.34'te arrange metodu ile grafik oluşturma gösterilmiştir.).

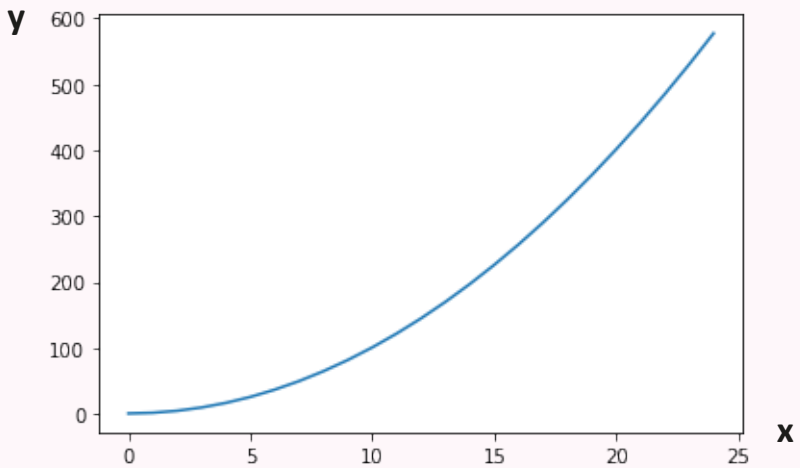
```
import numpy as np
dizi = np.arange(30)
plt.plot(dizi)
plt.show()
```



Görsel 1.34: NumPy dizi değerlerinin tek çizgi ile gösterilmesi

4. Adım: $y = x^2 + 1$ İkinci dereceden bir denklemin grafiğini oluşturunuz (Görsel 1.35'te denklemin grafiği gösterilmiştir.).

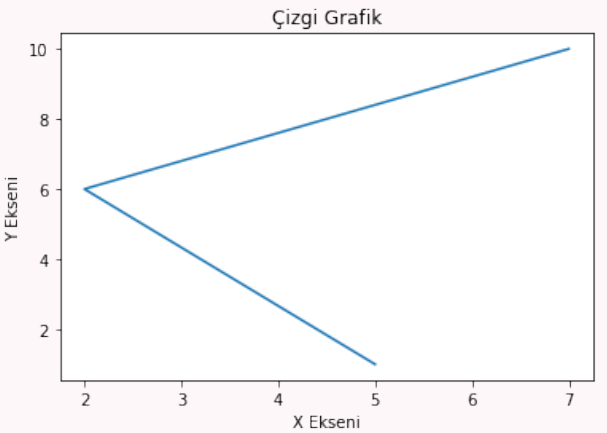
```
x=np.arange(25)
y=x**2+1
plt.plot(x,y)
plt.show()
```



Görsel 1.35: Denklem grafiği

5. Adım: Oluşturulan grafiğe başlık ve etiket ekleme işlemini yapınız (Görsel 1.36'da başlık ve etiketler gösterilmiştir.).

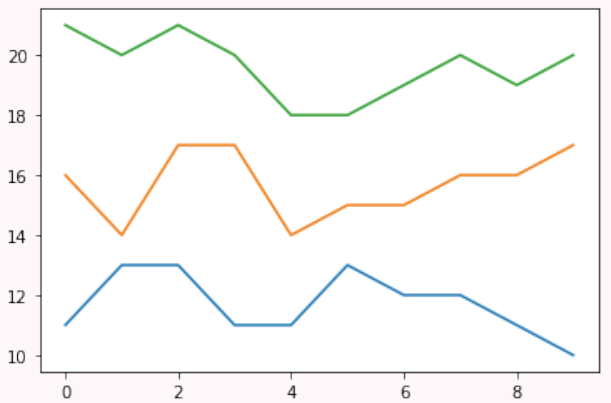
```
x = [5, 2, 7]
y = [1, 6, 10]
plt.plot(x, y)
plt.title('Çizgi Grafik')
plt.ylabel('Y Ekseni')
plt.xlabel('X Ekseni')
plt.show()
```



Görsel 1.36: Grafiğe başlık ve etiket ekleme

6. Adım: Birden çok grafik değerinin gösterilmesi işlemini yapınız (Görsel 1.37'de birden çok değerin grafik gösterimi verilmiştir.).

```
x = np.arange(0,10)
y0 = np.random.randint(10,14,10)
y1 = np.random.randint(14,18,10)
y2 = np.random.randint(18,22,10)
plt.plot(x,y0)
plt.plot(x,y1)
plt.plot(x,y2)
plt.show()
```



Görsel 1.37: Çoklu grafik

- **Çizim Tipleri**

Python matplotlib ile oluşturulabilecek çeşitli grafik tipleri bulunur. Bu grafik tiplerinden bazıları şunlardır:

Çubuk Grafik (Bar Graph): Verileri farklı kategoriler arasında karşılaştırmak ve belirli süre içinde değişiklikleri izlemek için kullanılır.

Histogram Grafik (Histogram Graph): Histogram bir dağılımı göstermek için kullanılan grafik türüdür.

Dağılım Grafik (Scatter Graph): Değişkenleri karşılaştırmak için kullanılan grafik türüdür. Bir değişkeninin diğeri ile ilişki kurmak için ne kadar etkilendiğini göstermek amacı ile kullanılır.

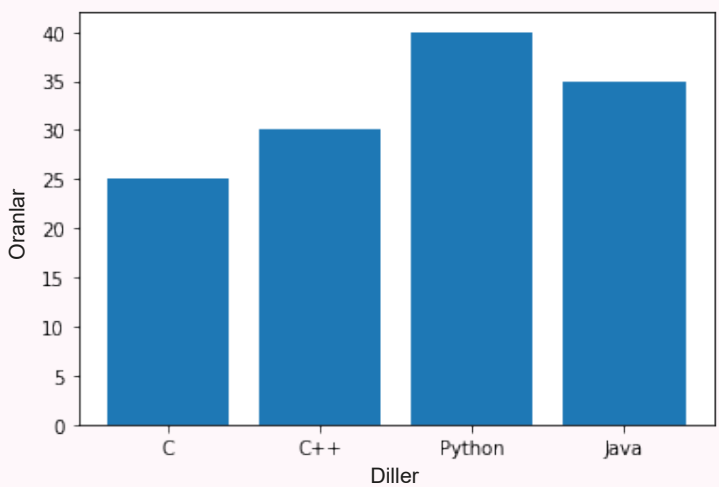
Pasta Grafik (Pie Graph): Pasta grafiği dilimlerden oluşan ve her dilimin yüzde veya oran olarak bir kategoriyi temsil ettiği grafik türüdür.

32. UYGULAMA

Matplotlib kütüphanesi kullanarak verilerin farklı grafik türlerinde gösterim işlemini yapmak için aşağıdaki adımları takip ediniz.

1. Adım: Verileri matplotlib bar metodu ile çubuk grafik şeklinde gösterme işlemini yapınız (Görsel 1.38'de verilerin çubuk grafik gösterimi verilmiştir.).

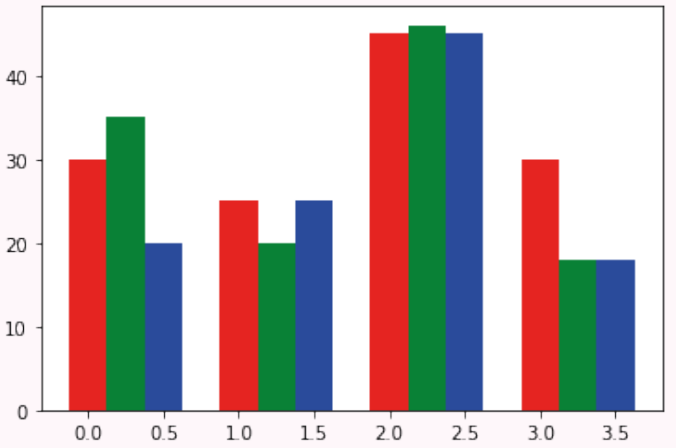
```
diller = ['C', 'C++', 'Python', 'Java']
oranlar = [25,30,40,35]
plt.xlabel("Diller")
plt.ylabel("Oranlar")
plt.bar(diller,oranlar)
plt.show()
```



Görsel 1.38: Çubuk grafik

2. Adım: Çubuk grafiğinde çubukların genişliklerini ve konumlarını değiştirerek birden fazla çubuk grafik oluşturma işlemini yapınız (Görsel 1.39'da birden fazla çubuk grafik oluşturma gösterilmiştir.).

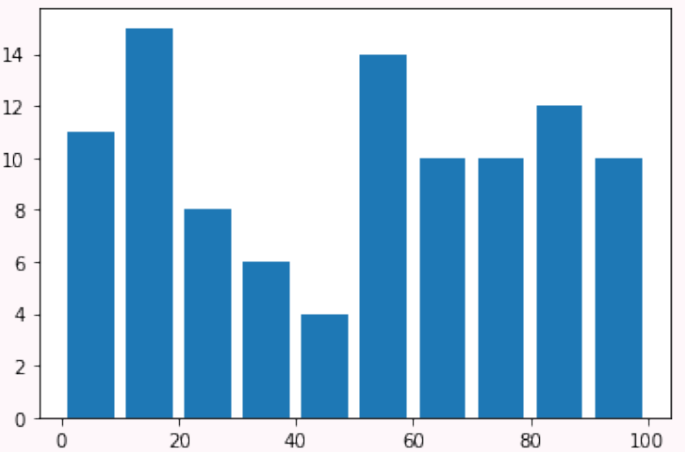
```
data = [[30, 25, 45, 30],
[35, 20, 46, 18],
[20, 25, 45, 18]]
x= np.arange(4)
plt.bar(x + 0.00, data[0], color = 'red', width = 0.25)
plt.bar(x + 0.25, data[1], color = 'green', width = 0.25)
plt.bar(x + 0.50, data[2], color = 'blue', width = 0.25)
plt.show()
```



Görsel 1.39: Çoklu çubuk grafik

3. Adım: 0 ile 100 arasında rastgele 100 adet sayıdan oluşan değerlerin dağılımını histogram grafik kullanarak yapınız (Görsel 1.40'ta değerlerin dağılımı histogram grafik olarak gösterilmiştir.).

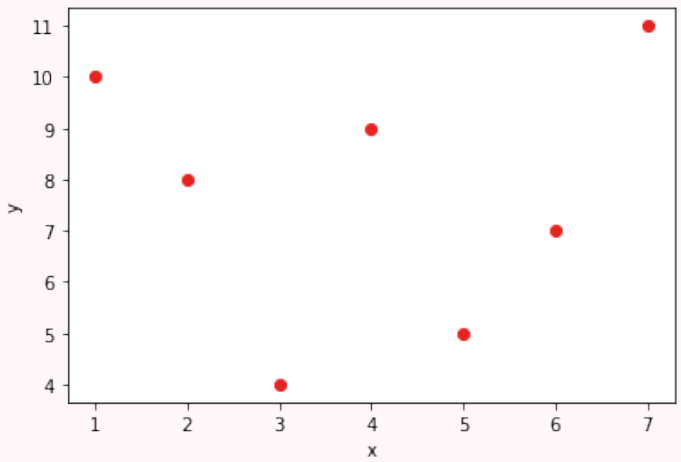
```
data=np.random.randint(0,100,100)
aralik=np.arange(0,110,10)
plt.hist(data, aralik, rwidth=0.8)
plt.show()
```



Görsel 1.40: Histogram grafik ile değerlerin dağılımı

4. Adım: Veriler arasındaki ilişkiyi scatter grafik kullanarak gösteriniz (Görsel 1.41'de scatter grafik ile veriler arasında ilişki gösterilmiştir.).

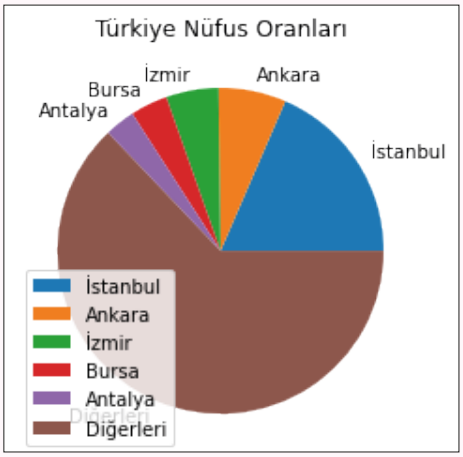
```
x = [ 1 , 2 , 3 , 4 , 5 , 6 , 7 ]
y = [ 10 , 8 , 4 , 9 , 5 , 7 , 11 ]
plt.scatter(x, y, label= 'skitscat' , color= 'red' )
plt.xlabel( 'x' )
plt.ylabel( 'y' )
plt.show()
```



Görsel 1.41: Dağılım grafik ile veriler arası ilişki

5. Adım: Türkiye'deki şehirlerin nüfus oranlarını pasta grafik olarak gösterme işlemini yapınız (Görsel 1.42'de değerlerin pasta grafiği gösterilmiştir.).

```
iller=["İstanbul","Ankara","İzmir","Bursa","Antalya",
       "Diğerleri"]
oran=[18.5,6.7,5.2,3.7,3,62.9]
plt.pie(oran, labels = iller)
plt.title( 'Türkiye Nüfus Oranları' )
plt.legend()
plt.show()
```



Görsel 1.42: Pasta grafik ile verilerin gösterimi

1. ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyunuz ve doğru seçeneği işaretleyiniz.

1. Aşağıdakilerden hangisi yapay zekânın tanımlarından biridir?

A) Kendi zekânızla programlama yapma
B) İnsan zekâsını bilgisayara yerleştirme
C) Oyun programlama ve oynama
D) Bir makineyi akıllı yapma
E) Bir bilgisayarı tasarlama

2. Aşağıdakilerden hangisi yapay zekânın alt alanlarından biri değildir?

A) Makine Öğrenmesi
B) Sinir Ağları
C) Uzman Sistem
D) Doğal Dil İşleme
E) Veri Tabanı Yönetim Sistemi

3. Aşağıdaki programlama dillerinden hangisi yapay zekâ için yaygın olarak kullanılır?

A) Python
B) Perl
C) Php
D) Go
E) Asp.Net

4. NumPy dizilerini kullanabilmek için aşağıdaki kodlardan hangisi kullanılır?

A) import numpy as np
B) import pyplot as plt
C) import pandas as pd
D) import math as np
E) import python as pt

5. NumPy dizisi oluşturmak için kullanılması gereken doğru söz dizimi aşağıdakilerden hangisidir?

A) np.createArray([1, 2, 3, 4, 5])
B) np.array([1, 2, 3, 4, 5])
C) np.object([1, 2, 3, 4, 5])
D) np.numpy([1, 2, 3, 4, 5])
E) np.Series([1, 2, 3, 4, 5])

6.
```
isimler = np.array(["Ali" , "Ayşe" , "Ahmet" , "Fatma" , "Zeynep"])
isimler[2] = "Ömer"
isimler
```

Verilen kodlar çalıştırıldığında ekran çıktısı aşağıdakilerden hangisi olur?

A) ['Ali', 'Ayşe', 'Ahmet', 'Fatma', 'Zeynep']
B) ['Ali', 'Ayşe', 'Ahmet', 'Ömer', 'Zeynep']
C) ['Ali', 'Ömer', 'Ahmet', 'Fatma', 'Zeynep']
D) ['Ömer', 'Ayşe', 'Ahmet', 'Fatma', 'Zeynep']
E) ['Ali', 'Ayşe', 'Ömer', 'Fatma', 'Zeynep']

7. **NumPy dizilerinde boyut sayısı bilgisine erişmek için aşağıdakilerden hangisi kullanılır?**

A) size B) dtype C) ndim D) shape E) itemsize

8. array([1, 2, 3, 4, 5, 6, 7, 8, 9])

Verilen çıktıyı elde edebilmek için aşağıdaki kodlardan hangisi kullanılır?

A) import numpy as np
np.arange(1,10)
B) import numpy as np
np.arange(0,10)
C) import numpy as np
np.array(1,10)
D) import numpy as np
np.array(0,10)
E) import numpy as np
np.arange(1,9)

9.
```
import numpy as np
x = np.array([1, 2, 3])
y = np.array([4, 5, 6])
np.concatenate([x,y])
```

Verilen kodların çıktısı aşağıdakilerden hangisidir?

A) array([[1, 2, 3], [4, 5, 6]])
B) array([1, 2, 3, 4, 5, 6])
C) array([4, 5, 6,1,2,3])
D) array([[4, 5, 6],[1,2,3]])
E) array([[1,2,3,1,2,3],[4,5,6,4,5,6]])

10.
```
x=np.array([[1, 0, 4, 2, 1], [7, 6, 1, 5, 3],[1, 9, 5, 8, 2]])
x[0:1]
```

Verilen kodların çıktısı aşağıdakilerden hangisidir?

A) array([[1, 9, 5, 8, 2]])
B) array([7, 6, 1, 5, 3])
C) array([[1, 0, 4, 2, 1]])
D) array([1, 0, 4, 2, 1])
E) array([[4, 1, 5]])

11. **Bir DataFrame'de axes=0 ifadesi aşağıdakilerden hangisini ifade eder?**

A) Index
B) Row
C) Column
D) Values
E) Size

12. **Bir pandas DataFrame'i oluştururken sütun isimlerini belirtmek için aşağıdaki parametrelerden hangisi kullanılır?**

A) index
B) size
C) variable
D) columns
E) rows

13. Bir csv dosyasını DataFrame'e yüklemek için aşağıdakilerden hangisi kullanılır?

A) read_csv()
B) ReadFile()
C) Readcsv()
D) to_csv
E) read_file()

14. Bx = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])
x['b']

Verilen kodların çıktısı aşağıdakilerden hangisidir?

A) 1 B) 2 C) 3 D) 4 E) 5

15. Bir data isminde DataFrame nesnesinde ilk üç satırını ve üç sütununu seçmek için aşağıdakilerden hangisi kullanılır?

A) data.iloc[0:3,0:3]
B) data.iloc[0:2,0:2]
C) data.iloc[1:3,1:3]
D) data.loc[0:3,0:3]
E) data.loc[1:3,1:3]

16. Aşağıdakilerden hangisi grafik çizimi için gerekli bir kütüphanedir?

A) plot
B) draw
C) numpy
D) pandas
E) matplotlib

17. Grafikleri görüntülemek için aşağıdaki kodlardan hangisi kullanılır?

A) plt.draw()
B) plt.chart()
C) plt.display()
D) plt.show()
E) plt.graph()

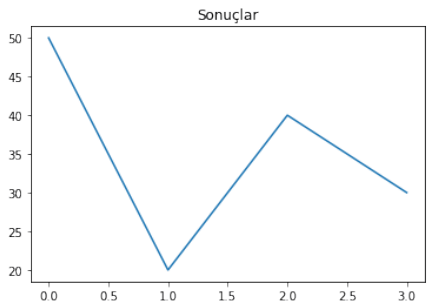
18. Oluşturulan bir grafiğe başlık eklemek için aşağıdaki kodlardan hangisi kullanılır?

A) plt.title("Başlık")
B) plt.title(Başlık)
C) title("Başlık")
D) plt.label("Başlık")
E) plt.head("Başlık")

19. Aşağıdakilerden hangisi bir grafik çeşidi değildir?

A) Çubuk
B) Yıldız
C) Histogram
D) Dağılım
E) Pasta

20.



Bu grafiğin oluşturduğu kodlama aşağıdakilerden hangisidir?

A)
```
x = np.array([0, 1, 2, 3])
y = np.array([50, 20, 40, 30])
plt.plot(x,y)
plt.title("Sonuçlar")
plt.show()
```

B)
```
x = np.array([0, 1, 2, 3])
y = np.array([50, 20, 40, 30])
plt.plot(x,y, "Sonuçlar")
plt.show()
```

C)
```
x = np.array([0, 1, 2, 3])
y = np.array([50, 20, 40, 30])
plt.plot("Sonuçlar", x,y)
plt.show()
```

D)
```
x = np.array([0, 1, 2, 3])
y = np.array([50, 20, 40, 30])
plt.bar("Sonuçlar", x,y)
plt.show()
```

E)
```
x = np.array([0, 1, 2, 3])
y = np.array([50, 20, 40, 30])
plt.plot(y,x)
plt.title("Sonuçlar")
plt.show()
```

MAKİNE ÖĞRENMESİ

2. ÖĞRENME BİRİMİ

http://kitap.eba.gov.tr/KodSor.php?KOD=36137

KONULAR

2.1. MAKİNE ÖĞRENMESİ KAVRAMI VE TEMELLERİ
2.2. MAKİNE ÖĞRENMESİ İLE İLGİLİ KÜTÜPHANELER
2.3. MAKİNE ÖĞRENMESİ İÇİN KULLANILAN ALGORİTMALAR
2.4. REGRESYON ANALİZİ VE TÜRLERİ
2.5. DİĞER GÖZETİMLİ ÖĞRENME ALGORİTMALARI

NELER ÖĞRENECEKSİNİZ?

- Makine öğrenmesinin temel kavramları ve temelleri
- Makine öğrenmesi türleri
- Makine öğrenmesi süreci
- Sınıflandırma, kestirim ve kümeleme kavramları
- Makine öğrenmesinde kullanılacak yazılımlar
- Model başarısını ölçme yöntemleri
- Makine öğrenmesinde kullanılan kütüphaneler
- Makine öğrenmesinde kullanılan algoritmalar
- Model geçerliliği
- Uygun makine öğrenmesi algoritmasının seçimi ve kullanımı

TEMEL KAVRAMLAR

- Yapay öğrenme
- Makine öğrenmesi
- Denetimli (Gözetimli) öğrenme
- Denetimsiz (Gözetimsiz) öğrenme
- Bağımlı ve bağımsız değişkenler
- Makine öğrenmesi süreci
- Model değerlendirme
- Makine öğrenmesi algoritmaları

HAZIRLIK ÇALIŞMALARI

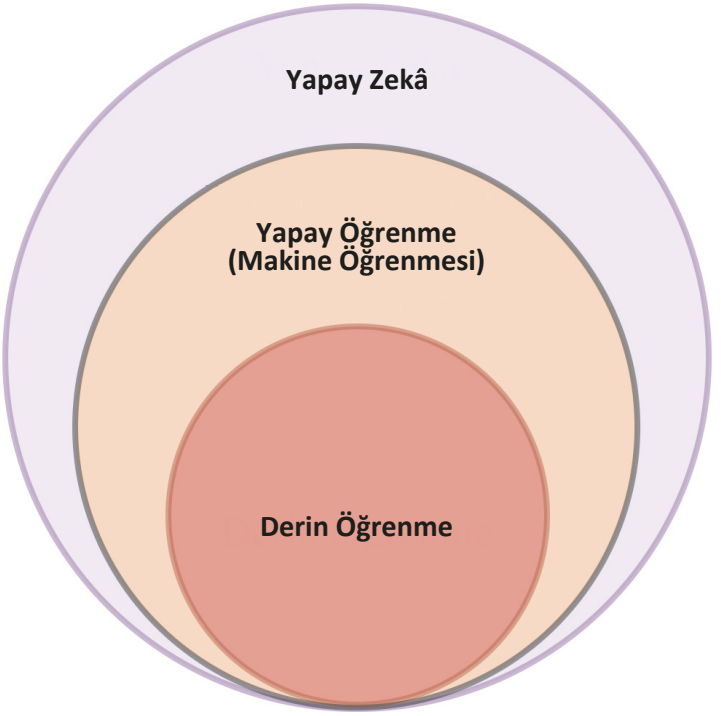
1. Makineler insanlar gibi öğrenebilir mi? Düşüncelerinizi arkadaşlarınızla paylaşınız.
2. Yapay öğrenme kavramı ile ilgili bilgilerinizi arkadaşlarınızla paylaşınız.

2.1. MAKİNE ÖĞRENMESİ KAVRAMI VE TEMELLERİ

Bu bölümde makine öğrenmesi konusuna giriş yapılarak konuyla ilgili temel kavramlara yer verilmiştir.

2.1.1. Makine Öğrenmesi

Türkçede yaygın olarak makine öğrenmesi şeklinde kullanılan terimin orijinali İngilizce Machine Learning'tir. İnsan öğrenmesini taklit edebilen her türlü öğrenmeye **yapay öğrenme** (makine öğrenmesi) denir. Makine öğrenmesi yapay zekânın alt alanı olarak yapay zekânın kullanıldığı her alanda kullanılır (Görsel 2.1).

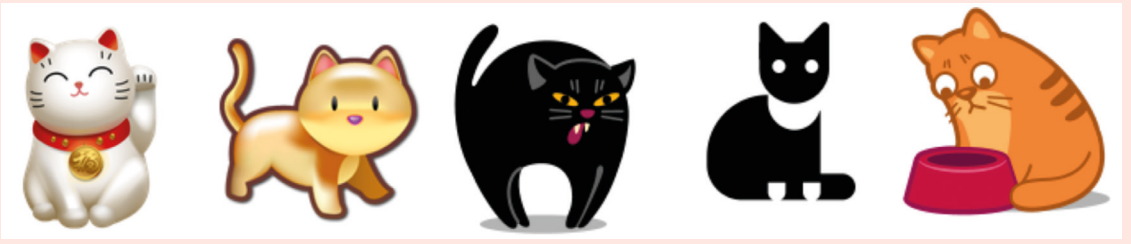


Görsel 2.1: Yapay öğrenme

Makine öğrenmesi, yapay zekânın içinde bir alan olarak yer alır. Makine öğrenmesini anlamak için insan öğrenmesini anlamak gerekir.

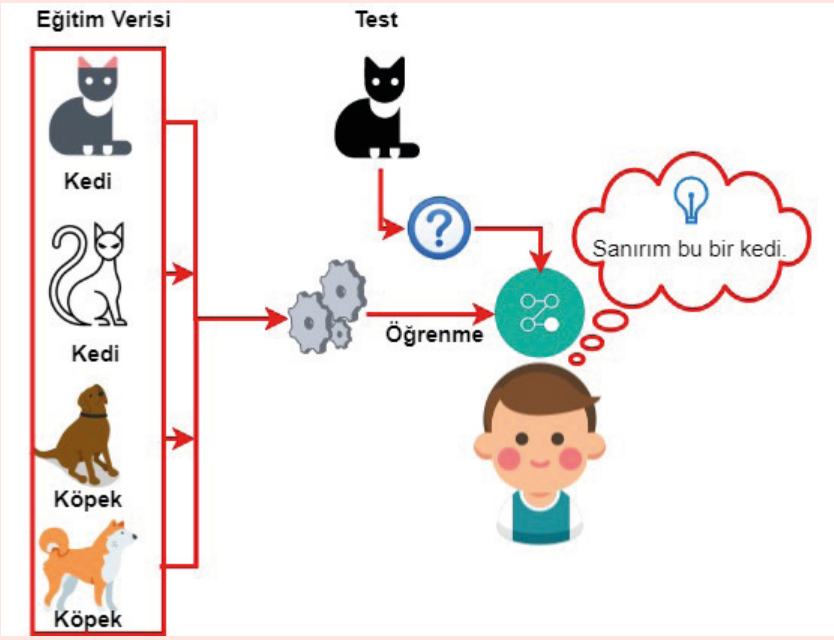
1. ÖRNEK

Küçük bir çocuk ilk kez kedi gördüğünde belki şaşıracak ve ne olduğunu soracaktır. "Bu bir kedi" yanıtını aldığında zihninde kediye ilişkin bir şablon oluşmaya başlar. Çocuk ilerleyen zamanlarda farklı renklerde, farklı türlerde ve farklı boyutlarda kediler ile karşılaştığında bunların da birer kedi olduğu birileri tarafından ona söylenir (Görsel 2.2).



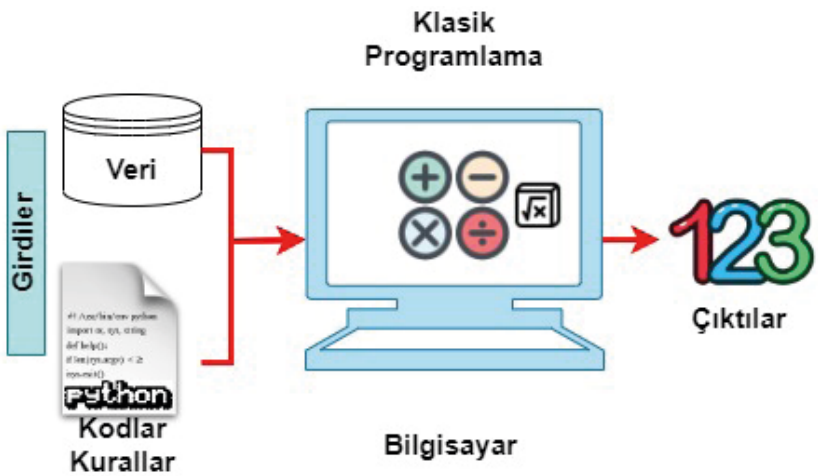
Görsel 2.2: Kedi türleri

Çocuğun gördüğü örnekler çoğaldıkça zihninde kedilere ilişkin ortak özelliklerin olduğu ayak sayısı, kuyruğu, kulaklarının şekli ve bıyıkları gibi bir şablon oluşmaya başlayacaktır. Bu özellikler, kedinin diğer hayvanlardan ayrılmasını sağlayan belirgin özellikleridir. Küçük çocuk, yeni bir kedi gördüğünde kafasındaki kedi şablonuyla ne derece eşleştiğini kontrol ederek bunun bir kedi olup olmadığına karar verecektir (Görsel 2.3). Bu öğrenme süreci kendiliğinden gerçekleştiği için çocuklar bir liste veya şablon oluşturmak için özel bir zaman ayırmamaktadır.



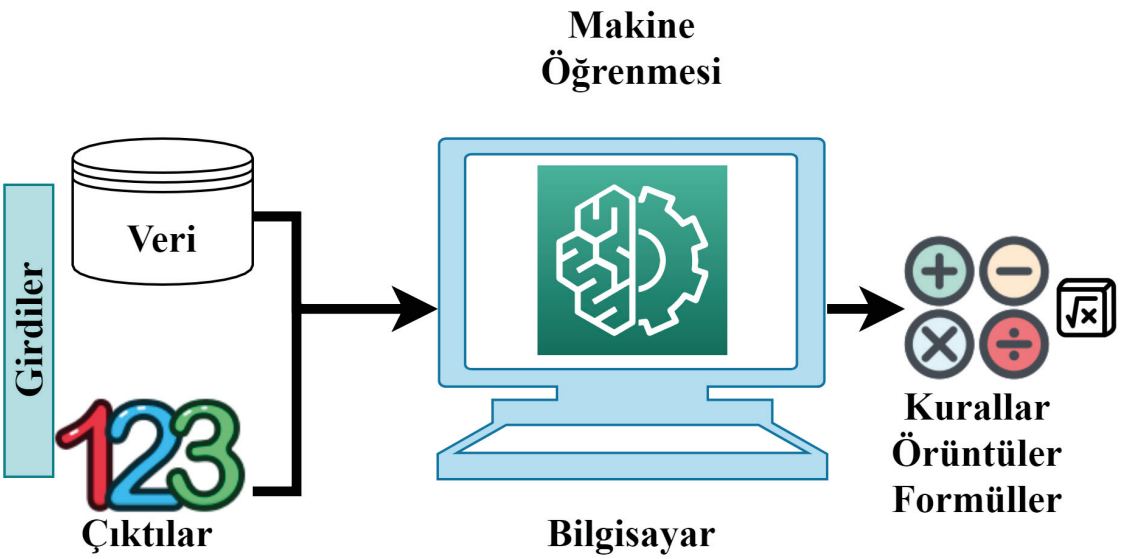
Görsel 2.3: Öğrenme süreci

Örnekte ele alınan öğrenme sürecinin yapay öğrenmede nasıl gerçekleştiği şöyle incelenebilir. Kedi ve köpek olarak etiketlenmiş (sınıflandırılmış) fotoğraflardan oluşan veri seti kullanılarak bir makine öğrenmesi algoritması (bir model) eğitilir. Makine öğrenmesi algoritması bu fotoğraflardan sınıfları betimleyen özellikleri çıkarır. Yeni verilen bir fotoğrafın belirlenen sınıf özellikleriyle eşleşme derecesine göre bir sınıflandırma yapar (Örnek: Bu bir kedi.). Makine öğrenmesini daha iyi anlamak için klasik programlamayla karşılaştırmak daha yararlı olabilir (Görsel 2.4). Klasik programlamada programcılar tüm olası durumları göz önünde bulundurarak koşullu ifadeler yazarlar. Programlar, bu kodlar (kurallar) doğrultusunda veriyi işler ve çıktı üretir. Yazılan komutlar dışında herhangi bir işlem gerçekleşmez.



Görsel 2.4: Klasik programlama

Makine öğrenmesinde ise veri ve çıktılar, makine öğrenmesi algoritmaları tarafından işlenerek veri ve çıktı arasındaki olası bağlantılar, ilişkiler, şablonlar, örüntüler ve kurallar ortaya çıkarılır (Görsel 2.5). Klasik programlamada sadece kodlama doğrultusunda belirli sonuçlar elde edilirken makine öğrenmesinde sonucu veri-çıktı belirler.

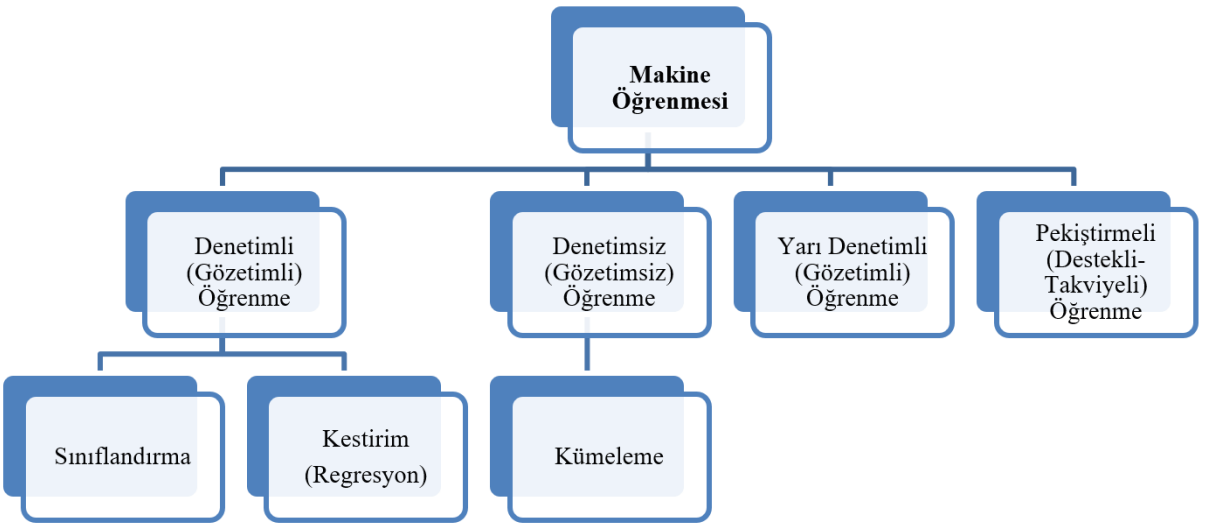


Görsel 2.5: Makine öğrenmesi

Klasik programlamayla binlerce satırda gerçekleştirilebilecek ve her veri seti için yeni kodlar, yeni kurallar eklenmesini gerektirebilecek işlemler, makine öğrenmesi yaklaşımıyla daha kolay yapılır. Klasik programlamanın tek sınırlılığı çok fazla kod yazmayı gerektirmesi değildir. Klasik programlamada tüm girdileri ve bunları işlemek için gerekli kuralları ve kodları önceden belirlemek de gerekir ki bu her zaman mümkün olmayabilir.

2.1.2. Makine Öğrenmesi Türleri

Makine öğrenmesi modelleri öğrenme türlerine göre dört gruba ayrılır (Şema 2.1).



Şema 2.1: Makine öğrenmesi

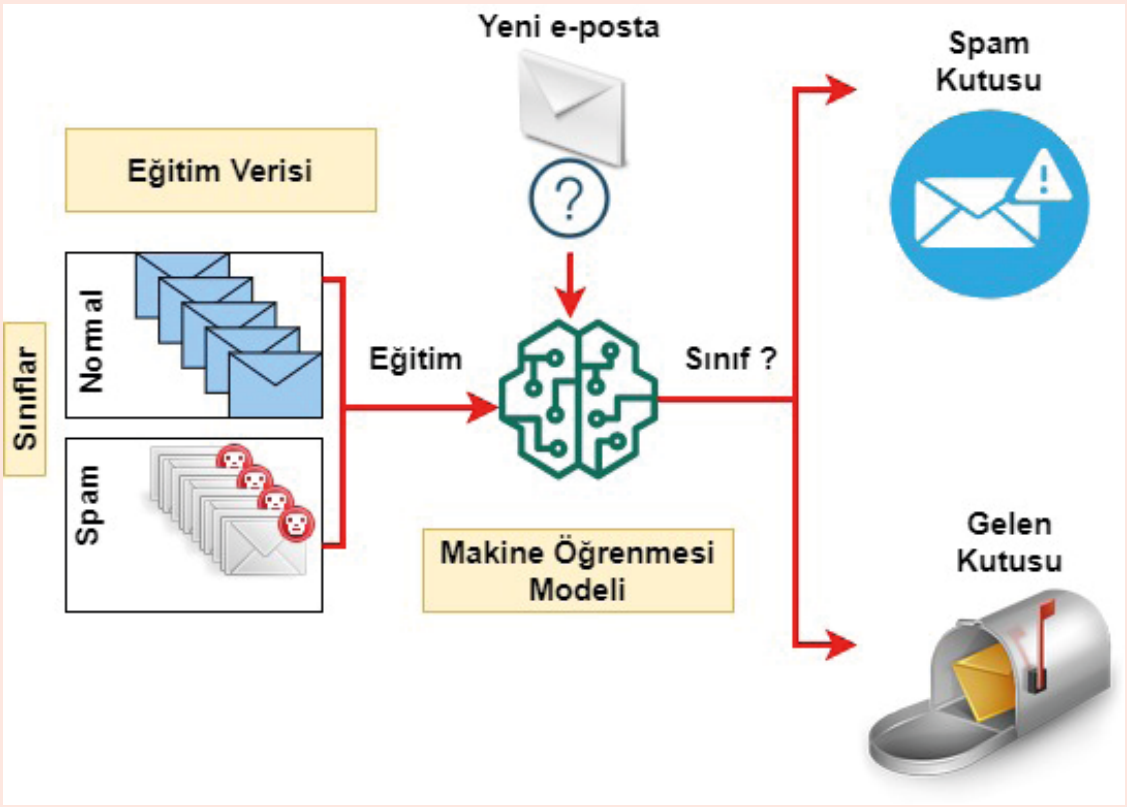
2.1.2.1. Denetimli (Gözetimli) Öğrenme

Denetimli öğrenme veriden çıktıları elde edecek fonksiyonları, kuralları, şablonları bulmak için kullanılır. Denetimli öğrenmede veri setinin durumuna göre ya bir sınıf aitliği (etiket) ya da bir bağımlı sonuç değişkeni

(sayısal değer) yer alır. Daha önceki örneklerde çocuğa gösterilen hayvanların kedi veya köpek (etiketler) olduğunun söylenmesi denetimli öğrenmeye bir örnektir. Denetimli öğrenmede veri setinde sınıflara ait etiket veya sonuç değeri bulunur. Denetimli öğrenme, sınıflandırma problemleri için kullanılır. Aşağıda bu tür sınıflandırma problemlerine ilişkin örnekler verilmiştir.

2. ÖRNEK

E-posta servislerinin bir hizmet olarak sunduğu istenmeyen e-posta (spam) filtreleme işlemi bir sınıflandırma problemidir (Görsel 2.6). E-posta servisi tarafından önceden etiketlenmiş normal ve istenmeyen e-posta örnekleri (etiketli veri) tarafından bir makine öğrenmesi algoritması ile özellikler çıkarılır, kurallar öğrenilir ve bir model oluşturulur. Yeni bir e-posta alındığında eğitilen model bu e-postayı sınıflandırarak istenmeyen kutusuna veya gelen kutusuna iletir.



Görsel 2.6: İstenmeyen e-posta sınıflandırma

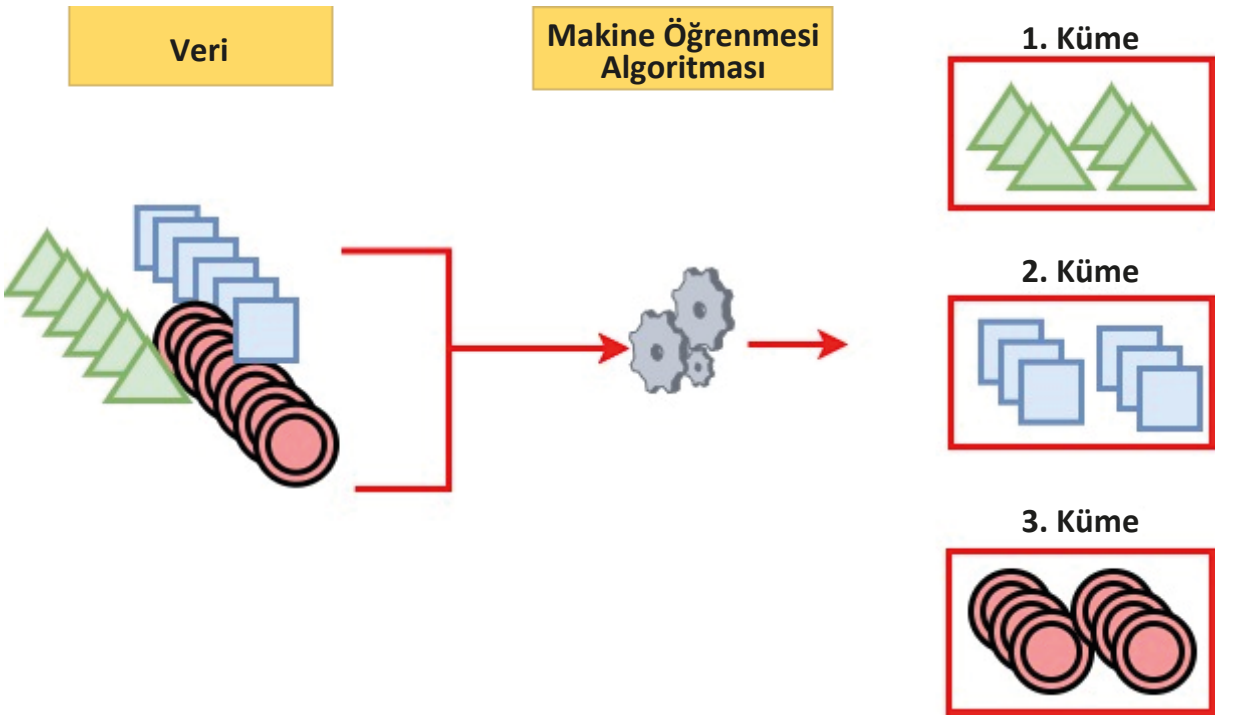
Denetimli öğrenme yalnızca sınıflandırma problemlerinde değil aynı zamanda kestirim (regresyon) için kullanılır. Kestirim, verideki bazı özellikler (bağımsız değişken) ile bir sonuç (bağımlı değişken) arasındaki bağıntı veya formülleri keşfetmek ve yeni veride sonuç değişkenini tahmin için kullanılabilir. Kestirim bir bağıntıdaki sabitleri belirlemek için kullanılırken tahmin ise belirlenen bağıntı kullanılarak değişkenin değerinin bulunmasıdır. Kestirim ve tahmin terimlerinin literatürde genellikle aynı anlamda kullanıldığını da belirtmek gerekir. Sonuç bu kez bir sınıf etiketi değil sayısal bir değerdir.

3. ÖRNEK

Elinizde ikinci el araba fiyatlarının bulunduğu bir liste olduğunu düşününüz. Listede arabanın özelliklerinin (marka, model, durumu, renk, kilometre ve sıfır aracın fiyatı) ve fiyat sütununun olduğunu varsayınız. Arabanın özellikleriyle fiyatı arasındaki bağıntıları bulabilir misiniz? Önceki tecrübelerinize dayanarak tahminde bulunabileceğiniz fiyatı bir kenara koyunuz. Listeyi inceleyerek diğer tüm özellikler aynıyken markanın satış fiyatını değiştirdiğini keşfedebilirsiniz. Bu durum sizi şaşırtmayacaktır. Peki, aracın yalnızca rengi, fiyatını belirleyebilir mi? Kilometresi düşük bir aracın aynı özelliklerdeki kilometresi yüksek bir araçtan daha pahalı olması beklenebilir. Listeyi inceleyerek ikinci el bir aracın fiyatı üzerinde etki eden özellikleri belirleyebilirsiniz. Size sorulan ikinci el bir aracın fiyatını listeyi kullanarak tahmin edebilir misiniz? Bu tür kestirim için de denetimli öğrenme kullanılmaktadır. Aracın özelikleriyle arabanın fiyatı arasında bir bağıntı, bir formül elde etmek için bir makine öğrenmesi algoritması eldeki liste (veri) ile eğitilerek bir model geliştirilir. Geliştirilen model kullanılarak girdi olarak özellikleri verilen ikinci el bir aracın fiyatı için bir tahmin yapılabilir.

2.1.2.2. Denetimsiz (Gözetimsiz) Öğrenme

Denetimsiz öğrenmede çıktılar yoktur. Bu yaklaşım, veriyi özelliklere göre gruplamak (kümelemek) için kullanılır. Denetimsiz öğrenmede bir neden sonuç ilişkisi aranmaz. Denetimli öğrenmede olduğu gibi eğitim işlemi de yer almaz. Veriyi etiketlemek genellikle insanlar tarafından yapılan zahmetli bir iştir. En büyük veri kaynağı internettir. İnternette veri çoğunlukla sosyal medya içeriği, metin, ses ve görüntü şeklindedir ve etiketlenmemiştir. Başka bir ifadeyle bir görsel vardır ancak ne olduğuna (etiket) ilişkin bir veri yoktur. Aynı şekilde bir ses dosyası vardır ancak bu sesin hangi canlıya ait olduğu belirtilmemiştir. Bu tür durumlarda denetimsiz öğrenme ile bir veri setinde benzer ögeler (örnekler) kümelenir (Görsel 2.7). Denetimsiz öğrenmede veri setinde etiket veya sonuç değişkeni yer almaz. Veri setindeki örneklerin ait oldukları sınıflar belirli değildir. Örnekte veri seti, makine öğrenmesi algoritması tarafından veri özelliklerine göre üç kümeye ayrılmıştır. Kümeler belirlendikten sonra her bir kümeden belirli sayıda örnek incelenerek kümelerin içeriği hakkında (Örnek: 1. Kümede üçgenler yer almaktadır.) bilgi edinilebilir.



Görsel 2.7: Denetimsiz öğrenme

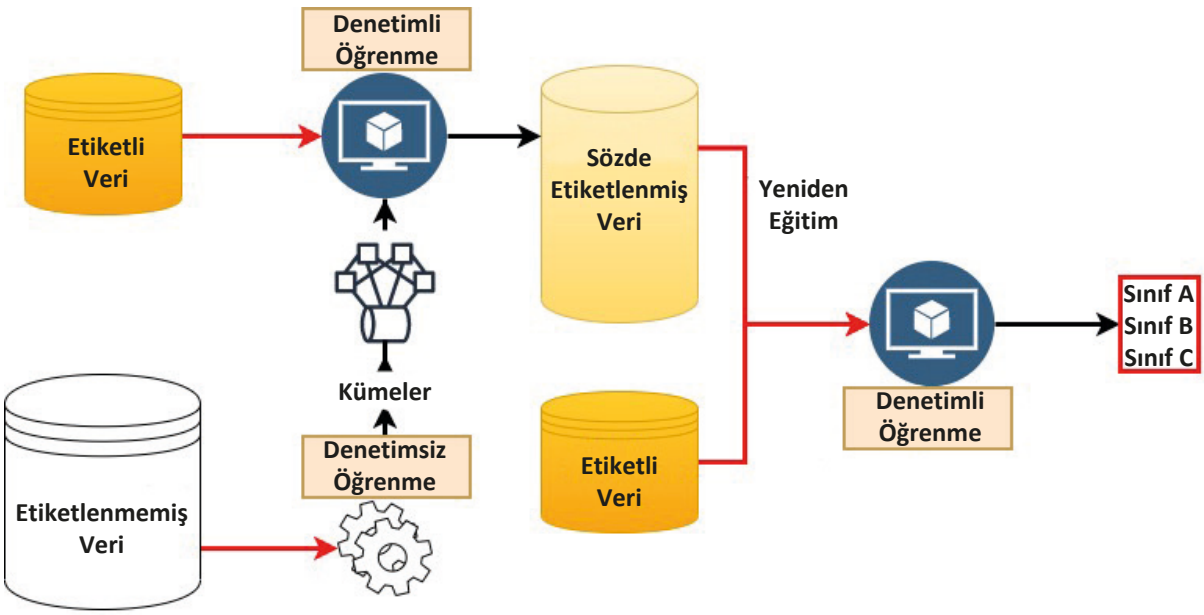
Denetimsiz makine öğrenmesinin kullanıldığı alanlardan biri de öneri sistemleridir.

4. ÖRNEK

Eğlence platformlarının bir filmi, bir müziği nasıl önerdiğini hiç düşündünüz mü? Sosyal medyada karşınıza çıkan arkadaşlık, grup önerileri ya da elektronik alışveriş sırasında sepetinize bir ürün ekledikten sonra başka ürünler önerilmesini (belki de tam aradığınız ürünler) düşününüz. Sizinle benzer aktivitelere sahip kişilerle aynı grupta yer almanız ve benzer zevklere sahip olabileceğiniz düşünülerek sizinle aynı kümede bulunan diğer kullanıcıların izledikleri filmler veya dinledikleri müzikler size de önerilecektir. Elbette diğer kullanıcıların yorumlarını ve puanlarını değerlendiren ek modellemeler de yer alabilir. Alışveriş örneğinden gidilirse (Sepet analizi olarak da adlandırılır.) birliktelik modeliyle A ürününü alan müşterilerin belirli bir oranının B ürününü de aldığı belirlenebilmektedir. Böylece A ürününü alan müşterilere B ürünü de önerilmektedir.

2.1.2.3. Yarı Denetimli (Yarı Gözetimli) Öğrenme

Yarı denetimli öğrenme hem denetimli hem de denetimsiz öğrenmenin birlikte kullanıldığı, içinde az miktarda etiketlenmiş veri ve çok miktarda etiketlenmemiş verinin bulunduğu veri kümeleri için kullanılan öğrenme türüdür (Görsel 2.8). Bu öğrenme türü; sınıflandırma, regresyon ve tahmin için kullanılabilir. Etiketlenmiş veri kullanılarak elde edilen bilgiler, etiketlenmemiş veri üzerinden denetimsiz öğrenme ile elde edilen kümeleri tanımlamak için kullanılabilir. Kümeler tanımlandıktan sonra sözde etiketlenmiş veriler ve etiketli veri kullanılarak denetimli öğrenme ile model yeniden eğitilir. Etiketlenmiş az sayıda veri ve çok sayıda etiketlenmemiş veri bulunan bilgisayarlı görü, doğal dil işleme gibi alanlar yarı denetimli öğrenmenin en çok kullanıldığı alanlardır.

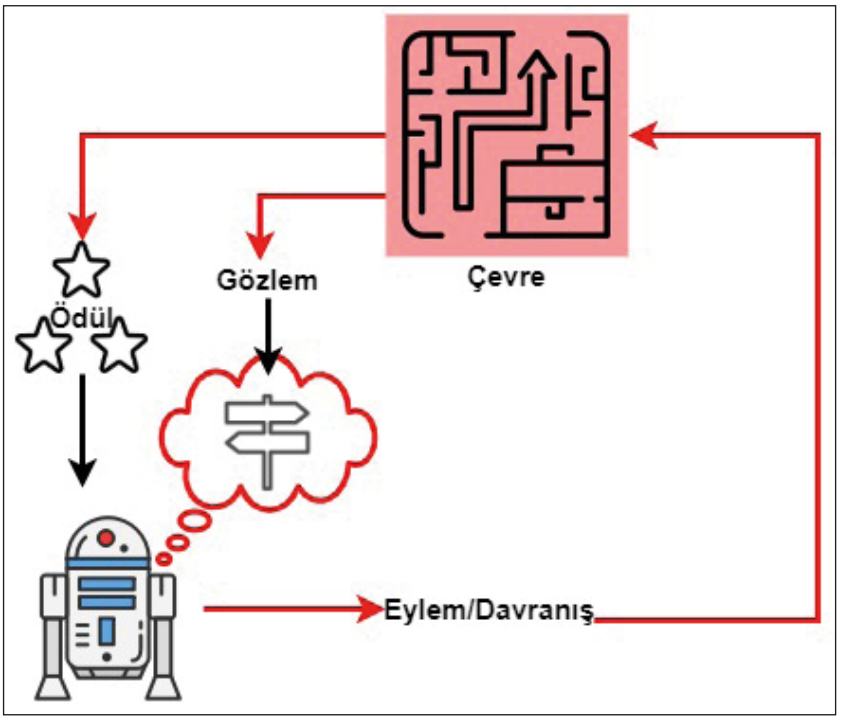


Görsel 2.8: Yarı denetimli öğrenme

2.1.2.4. Pekiştirmeli Öğrenme

Pekiştirmeli öğrenme, diğer öğrenme türlerinden farklı olarak deneme-ödül motivasyonuna dayanan bir öğrenme sürecidir. Bir ortamdaki en yüksek ödülü toplamak için gerekli adımları bir sistemin deneme-yanılma yoluyla bulması temeline dayanır. Bir robot, labirentten çıkışı bulmak için yaptığı eylemler sonucunda ödül

alıyorsa (bir oyunda skorun artması gibi) çevreyi gözlemler ve hareketlerine ona göre devam eder (Görsel 2.9). Bu durumda robota hangi yolun daha iyi olduğunu gösterecek bir gözetmen olmadığı (veya sonuç etiketleri yer almadığı) için denetimli öğrenmeden ayrılır. Pekiştirmeli öğrenme, etiketli verinin ve ortam hakkında yeterli bilginin olmadığı durumlarda kullanılmaktadır. Özellikle robotik, oyunlar ve kişisel öneriler pekiştirmeli öğrenmenin kullanıldığı alanlardır.



Görsel 2.9: Pekiştirmeli öğrenme süreci

2.1.3. Makine Öğrenmesine İlişkin Terimler

Bağımlı Değişken (Çıktı): Makine öğrenmesindeki çıktıdır. Bu bir sınıf etiketi, bir tahmin veya kümeleme olabilir.

Bağımsız Değişken: Kendisi herhangi bir değişkenden etkilenmeyen ancak bağımlı değişken üzerinde etkisi olan (veya olduğu düşünülen) değişken veya değişkenlerdir.

Makine Öğrenmesi Algoritması: Makine öğrenmesi için bir kod yazarken izlenen prosedürlerdir. Farklı makine öğrenmelerinin farklı algoritmaları bulunmaktadır.

Model: Makine öğrenmesi algoritmasının bir program veya kod aracılığıyla veri üzerinde çalıştırılması (eğitilmesi) ile oluşur. Bir model makine öğrenmesi algoritması tarafından öğrenileni temsil eder.

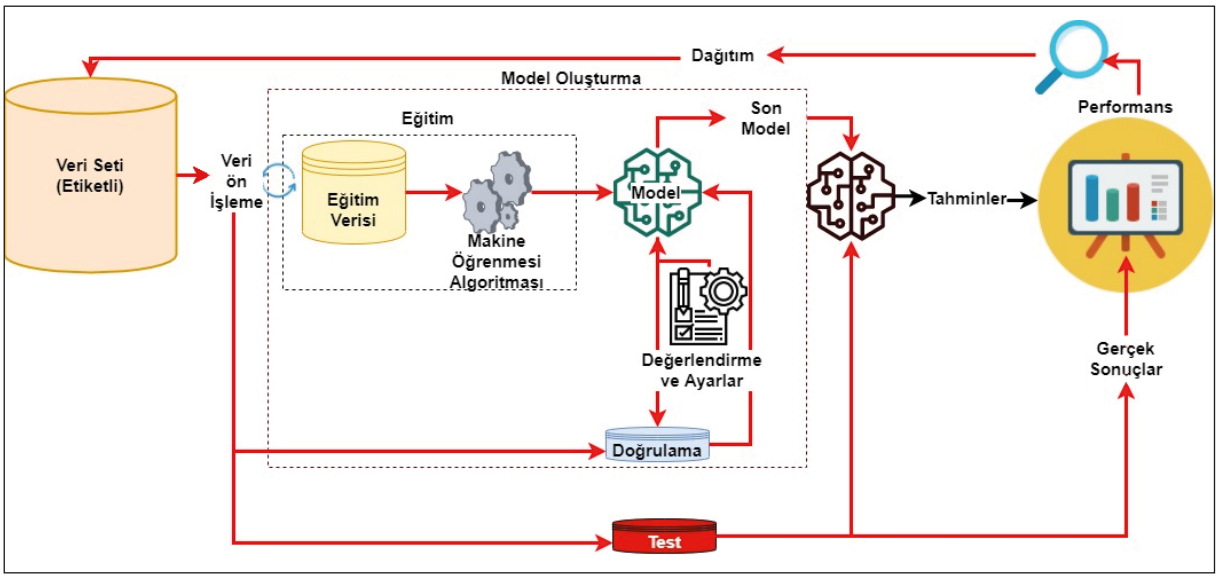
Eğitim Verisi (Training Data): Denetimli öğrenmede modeli eğitmek için kullanılan veri setidir.

Doğrulama Verisi (Validation Data): Oluşturulan modelin geçerliliğinin değerlendirilmesi ve model üzerinde gerekli hiper parametre ayarlarının yapılması için kullanılan veri setidir.

Test Verisi (Test Data): Bir modelin gerçek veri üzerinde işe yarayıp yaramadığının (doğru fiyat tahmini, doğru sınıflandırma gibi) kontrol edilmesi için kullanılan veri setidir. Bir model, eğitimde kullanılan veri seti ile test edilirse bu işlem, modelin performansı hakkında yanıltıcı sonuçlar verebilir.

2.1.4. Makine Öğrenmesi Süreci

Makine öğrenmesi süreci Görsel 2.10'da verilmiştir. Denetimli bir makine öğrenmesi sürecini temsil eden görsele göre önce veri seti alınır ve veri ön işleme adımlarında incelenerek gerekli işlemler yapılır. Veri ön hazırlık aşaması; verinin keşfedildiği, temizlendiği, eksik verilerin ele alındığı, gerekli dönüşümlerin yapıldığı, boyutların indirgendiği (özellik çıkarımı) aşamadır. Veri ön işlemenin amacı, veri setini model oluşturma aşamasında algoritmanın kullanımına uygun hâle getirmektir. Bu aşamada veri görselleştirme veri keşfi için çok yararlı olabilir. Veri ön işleme aşaması, model geliştirme aşamasına bağlı olarak tekrarlanabilir.



Görsel 2.10: Makine öğrenmesi süreci

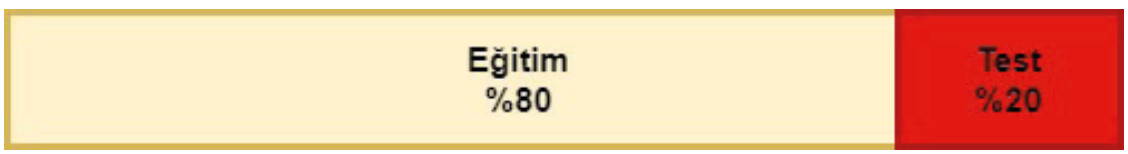
Veri seti ön işleme aşamasından sonra veri seti; eğitim, doğrulama ve test için ayrılmalıdır. Bunun için iki ana yaklaşım vardır:

- **Eğitim-doğrulama ve test:** Bu yaklaşımda veri seti üç parçaya ayrılır. Genel eğilim eğitim için %60, doğrulama için %20 ve test için %20 ayrılmasıdır (Görsel 2.11).

Görsel 2.11: Eğitim-doğrulama-test

Makine öğrenmesi algoritması eğitim verisini kullanarak bir model geliştirir. Doğrulama verisi, modeli değerlendirmek ve ayarlamak için kullanılır. Doğrulama verisindeki etiketler modelin tahminleri ile karşılaştırılarak değerlendirilir. Bu aşamada modelin hiper parametreleri ayarlanmalıdır. Gerekli ayarlar yapılarak modelin tahmin başarısı artırılmaya çalışılır. Yapılan ayarlamalar başarı oranını artırabildiği gibi kritik değerler aşıldığında başarı oranını düşürebilir. Başarı oranının en yüksek olduğu hiper parametre değerleri bulunmalıdır. Bu ayarlamalar yapıldıktan sonra modelin son hâli test verisi üzerinde test edilir. Test verisindeki etiketleri tahmin etmedeki başarı oranı modelin performansını verir.

- **Çapraz doğrulama:** Bu yaklaşımda veri seti yaklaşık olarak 80/20 oranında ayrılır. Veri setinin %80'i çapraz doğrulama, %20'si ise final testi için kullanılır (Görsel 2.12).



Görsel 2.12: Eğitim-test oranı

Çapraz doğrulama yapılırken eğitim seti n alt kümeye ayrılır. Her defasında (n-1) alt küme eğitim için kullanılırken bir tanesi doğrulama için kullanılır. Her alt küme doğrulama için bir kez kullanılana kadar işlem tekrarlanır. Bu sayede eğitim verisinin tamamı hem eğitim hem de test için kullanılmış olur.

5. ÖRNEK

Görsel 2.13'te n = 5 için 5 adımlı bir çapraz doğrulama yapılır. Eğitim verisi 5'e bölünmüştür. Birinci adımda ilk parça doğrulama için ayrılarak eğitim setinin geri kalanı (4/5) model oluşturulurken eğitim için kullanılmıştır. Eğitim işlemi bittikten sonra doğrulama veri seti için etiketler veya sayısal değerler tahmin edilir. Tahmini değerler ile doğrulama veri setindeki gerçek etiketler veya sayısal değerler karşılaştırılarak performans değerleri hesaplanır. Bu işlem her bir adımda veri setinin farklı bir parçası doğrulama, geri kalanı eğitim için kullanılarak devam eder. Beşinci adımın tamamlanmasıyla eğitim setinin tamamı hem eğitim hem de doğrulama için kullanılmış olur (Görsel 2.13).

1. Adım	**Doğrulama**			1. Performans puanı
2. Adım		**Doğrulama**		2. Performans puanı
3. Adım			**Doğrulama**	3. Performans puanı
4. Adım			**Doğrulama**	4. Performans puanı
5. Adım			**Doğrulama**	5. Performans puanı

Görsel 2.13: Çapraz doğrulama

Modelin doğrulama performansı hesaplanırken ortalama performans puanı hesaplanır.

Ortalama Performans Puanı = (1. Performans Puanı + 2. Performans Puanı + 3. Performans Puanı + 4. Performans Puanı + 5. Performans Puanı) / 5

Modelin performansı değerlendirilerek model içinde gerekli parametreler değiştirilebilir. Ayarlamalar yapılarak güncellenen model tıpkı **Eğitim-doğrulama ve test** yaklaşımında olduğu gibi test edilerek performansı tekrar değerlendirilir ve uygulamaya geçirilir. Makine öğrenmesi modelleri veriyi açıklayacak bir model bulmak için kullanılır. Bulunan açıklamalar (modeller) tekrar veriye dönmeyi gerektirebilir. Makine öğrenmesi süreci dairesel bir süreçtir ve uygulamadan sonra veri setine dönmek gerekebilir. Farklı algoritmalar için veri setlerinde farklı ön işleme süreçleri yürütmek gerekebilir. Geliştirilen modeller test aşamasında istenen düzeyde bir başarıya ulaşamayabilir. Bu yüzden tekrar veriye dönerek veri setini incelemek gerekebilir. Özelikle eğitim aşamasında kullanılan veri setinin (örneklem) bütünü temsil etmesi çok önemlidir. Tersi durumda model istenildiği gibi çalışmayabilir.

2.1.5. Makine Öğrenmesi Performansının Ölçülmesi

Makine öğrenmesinde sınıflama veya kestirim gibi farklı amaçlar için farklı algoritmalar kullanılır. Eğitim işlemi, algoritmaları ve veri setini kullanarak bir model geliştirme (eğitim) işlemidir. Geliştirilen modellerin başarısını ölçmek modeli değerlendirebilmek için gereklidir. Sınıflandırma ve sayısal tahmin olmak üzere iki farklı performans ölçüm yöntemi bulunur.

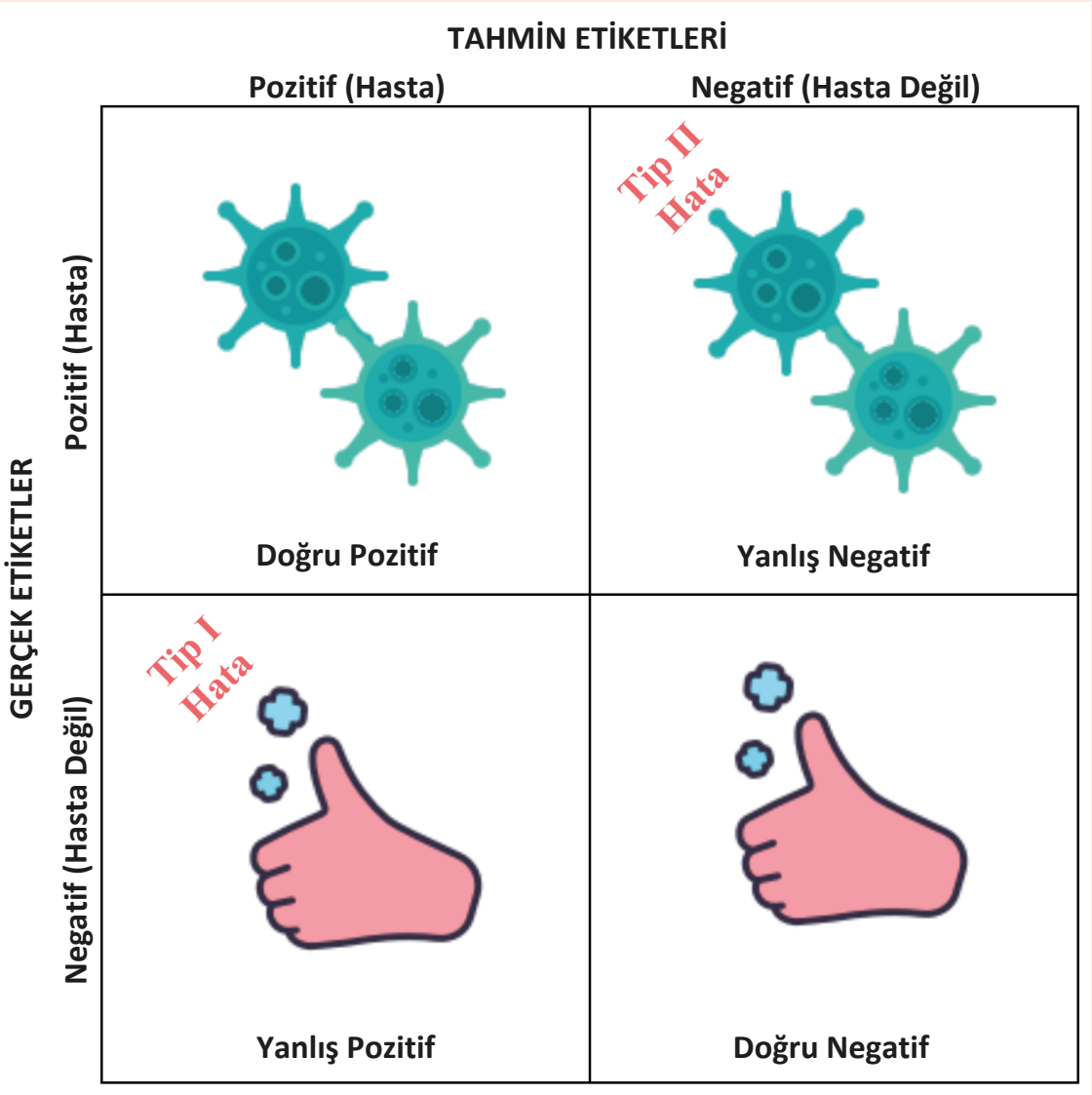
2.1.5.1. Sınıflandırma Performansının Ölçülmesi

Sınıflandırma işlemlerinde veri setindeki sınıf etiketleri tahmin edilmektedir.

6. ÖRNEK

Kişilerin özellikleri ve röntgen görüntüleri kullanılarak COVID Testi yapan bir makine öğrenmesi modeli. Çeşitli belirtilerden veya röntgen resimlerinden hastalık sınıflandırması yapan bir model

oluşturduğunuzu düşününüz. Modelinizi hazırladınız ve doğrulama aşamasında veya test aşamasında modelinizi değerlendirmek istiyorsunuz. Test veri setinde hasta veya hasta değil olarak etiketli kişilerin model tarafından doğru olarak sınıflandırılmasını beklersiniz. Makine öğrenmesinin tahminlerini gerçek etiketlerle karşılaştırmanız gerekir. Test setinde "hasta" olarak etiketlenmiş bir kişiye ait girdiler verildiğinde "hasta" olduğunu tahmin etmesini beklersiniz ancak her zaman böyle olmaz. Modeller tıpkı insanlar gibi (böyle işlemlerde insanlardan çok daha fazla) hatalı sınıflandırmalar yapabilir. COVID+ sınıflandırması yapan bir modelin hata matrisi (confusion matris) Görsel 2.14'te verilmiştir.



Görsel 2.14: COVID+ sınıflandırması

Sınıflandırma için geliştirilen bir model örnekteki gibi ikili sınıflandırmada dört farklı tahmin üretebilir. Gerçek hasta olan bir kişi (COVID+) model tarafından "hasta" olarak tahmin edilirse buna **doğru pozitif**, gerçekte hasta olmasına rağmen "hasta değil" olarak sınıflandırılırsa buna **yanlış negatif** denir. Gerçekte "hasta olmayan" bir kişi model tarafından "hasta" olarak sınıflandırılırsa buna **yanlış pozitif**, gerçekte "hasta olmayan" bir kişi "hasta değil" olarak tahmin edilirse buna **doğru negatif** denir. Yanlış pozitif ve yanlış negatif sırasıyla Tip I ve Tip II hata olarak adlandırılır. Sınıflandırma metriklerinin hesaplanmasına ilişkin formüller Tablo 2.1'de verilmiştir.

Tablo 2.1: Hata Matrisi Hesaplama

		Tahmin Sınıfları		
		Pozitif	**Negatif**	
Gerçek Sınıflar	**Pozitif**	**Doğru Pozitif (DP)** 15 vaka	**Yanlış Negatif (YN)** Tip II Hata 12 vaka	**Duyarlılık (Sensitivity True Positivite Rate, Recall)** $\frac{DP}{(DP+YN)}$
	Negatif	**Yanlış Pozitif (YP)** Tip I Hata 5 vaka	**Doğru Negatif (DN)** 68 vaka	**Seçicilik (Specificity, True Negative Rate, Selectivity)** $\frac{DN}{(DN+YP)}$
		Kesinlik/Hassasiyet (Precision) $\frac{DP}{(DP+YP)}$	**Negatif Tahmin Değeri** $\frac{DN}{(DN+YN)}$	**Doğruluk (Accuracy)** $\frac{DP+DN}{(DP+DN+YP+YN)}$

- DP (Doğru Pozitif).
- DN (Doğru Negatif).
- YP (Yanlış Pozitif).
- YN (Yanlış Negatif).

Sınıflandırmada kullanılan bazı metrikler şunlardır: Doğruluk, kesinlik, negatif tahmin değeri, duyarlılık, seçicilik ve F1 skoru. Ele alınan sınıflandırma problemine göre metriklerin önemi değişir.

- **Doğruluk**

  Model tarafından doğru tahmin edilen örnek sayısının (doğru pozitif + doğru negatif) toplam örnek sayısına oranıdır.

  Örnek: Testte, 100 tahminde toplam 83 adet doğruysa doğruluk %83 olarak hesaplanır.

- **Kesinlik**

  Pozitif tahminlerin kesinliğini belirtmektedir. Gerçekte pozitif olan örneklerin pozitif olarak tahmin edilenlere oranıdır.

  Örnek: Testte 100 örnekten 20'si hasta olarak tahmin edilmiştir. Ancak bu tahminlerden 15'i gerçekten hastadır. Kesinlik oranı 15/20'den %75 olarak hesaplanır.

- **Negatif Tahmin Değeri**

  Gerçekte negatif olan örneklerin negatif olarak tahmin edilenlere oranıdır.

  Örnek: Testte 100 örnekten 80'i hasta değil olarak tahmin edilmiştir. Ancak bu tahminlerden 12'si aslında hastadır. Negatif tahmin değeri 68/80'den %85 olarak hesaplanır.

- **Duyarlılık**

  Pozitif olarak doğru tahmin edilen örneklerin (DP) gerçekte toplam pozitif olanlara oranıdır.

7. ÖRNEK

Özellikle bulaşıcı bir hastalık veya kanser için vakaların yanlışlıkla hasta değil olarak tahmin edilmesi hayati bir sorundur. Böyle durumlarda duyarlılık değerleri önemli hâle gelir.

Örneğin gerçekte toplam 27 hasta vardır. Model, 27 hastanın 15'ini hasta olarak tahmin ederken 12'sini hasta değil olarak tahmin etmiştir. Modelin duyarlılığı %55.6 olarak hesaplanır.

Seçicilik

Negatif olarak doğru tahmin edilen örneklerin (DN) gerçekte toplam negatif olanlara oranıdır.

Örnek: Gerçekte hasta olmayan toplam 73 kişi vardır. Model bunlardan 68'ini doğru tahmin ederken 5'ini ise hasta olarak tahmin etmiştir. Modelin seçiciliği %93.2 olarak hesaplanır.

F1 Skoru

Dengesiz sınıf dağılımına sahip veri setleri için kullanılan bir metriktir. Modelin F1 skoru %63.8 olarak hesaplanır.

$$\frac{2DP}{(2DP+YP+YN)}$$

8. ÖRNEK

On binde bir görünen bir hastalığı erken teşhis etmek için bir makine öğrenmesi modeli geliştirdiniz. Bu model 100.000 veriden oluşan test verisinin tamamını "hasta değil" olarak tahmin ederse doğruluğu %99.9 olur ancak model aslında hiçbir hastaya doğru teşhis koymamıştır. Böyle bir durumda tek başına doğruluk metriğinin anlamı kalmaz. Kesinlik değeri %0.0 olarak hesaplanır.

Negatif tahmin değeri %99.9 olarak hesaplanır.

Seçicilik değeri %100 olarak hesaplanır.

Duyarlılık değeri ise %0.0 olarak hesaplanır.

Modelin başarısını değerlendirmek için hangi metriklerin incelenmesi gerektiği çok önemlidir. Örnekte verilen bir problem durumu için kesinlik ve duyarlılık değerlerinin önemi ön plana çıkmaktadır. Geliştirilen bir modelin sınıflandırma başarısı değerlendirilirken bu tür durumlara dikkat edilmelidir.

2.1.5.2. Kestirim (Regresyon) Performansının Ölçülmesi

Veri setindeki etiketin sınıflandırma dışında sayısal bir değeri olduğu durumlarda farklı metrikler kullanılır.

9. ÖRNEK

İkinci el araç fiyatlarını tahmin etmek için bir makine öğrenmesi modeli geliştirdiğinizi düşününüz. Modelinizin ne kadar başarılı olduğunu nasıl ölçersiniz? Modelin başarısı, tahmini fiyatlar ile araçların gerçek fiyatlarının karşılaştırılması ile belirlenir. Bunun için farklı metrikler vardır:

R^2, RMSE, MAE, MSE ve MAPE.

Metrikleri hesaplamak için Tablo 2.2'deki değerler kullanılmıştır.

Tablo 2.2: Model Performansında Hata Hesaplama

	Gerçek Fiyat (y)	*Tahmin (y')*	*Fark (e_t) (y-y')*	*Mutlak Fark*	e_t / y	*Farkın Karesi* e_t^2	$y'-\bar{y}$	$(y'-\bar{y})^2$
1	110	120	10	10	0,09	100	-42,5	1806,25
2	90	85	-5	5	0,05	25	-77,5	6006,25
3	50	70	20	20	0,4	400	-92,5	8556,25
4	320	330	10	10	0,03	100	167,5	28056,25
5	70	60	-10	10	0,14	100	-102,5	10506,25
6	340	320	-20	20	0,05	400	157,5	24806,25
7	110	120	10	10	0,09	100	-42,5	1806,25
8	105	110	5	5	0,04	25	-52,5	2756,25
9	280	250	-30	30	0,10	900	87,5	7656,25
10	150	200	50	50	0,33	2500	37,5	1406,25
Ort	162,5	166,5	***Toplam***	170	1,35	4650	-	93362,50

Terimler

e_t : Hata terimi (gerçek değer ile tahmin arasındaki fark)
y : Gerçek değer
t : İndis numarası (örneklemdeki sırası)
n : Örneklem büyüklüğü
y' : Tahmin değeri
$\bar{y}$: Gerçek değerlerin ortalaması
p : Bağımsız değişken sayısı

MSE: Ortalama Kare Hata (Mean Squared Error)

Tahmin edilen değerlerle gerçek değerler arasındaki farkın karelerinin ortalaması alınarak hesaplanır. MSE aşağıdaki formülle hesaplanır.

$$MSE = \frac{1}{n}\sum_{t=1}^{n} e_t^2$$

Değerler yerine konulduğunda Ortalama Kare Hata değeri 465 olarak hesaplanır.

$$MSE = \frac{4650}{10}$$

R^2 Değeri

$$R^2 = 1 - \frac{\sum_{t=1}^{n} e_t^2}{\sum_{t=1}^{n} (y - \bar{y})^2}$$

Formüle göre Tablo 2.1.'de modelin R2 değeri

$$R^2 = 1 - \frac{4650}{93362,5} = 0,95$$

MAE: Ortalama Mutlak Hata (Mean Absolute Error)

Tahmin edilen değerlerle gerçek değerler arasındaki farkın mutlak değerlerinin ortalaması alınarak hesaplanır. MAE aşağıdaki formülle hesaplanır.

$$MAE = \frac{1}{n}\sum_{t=1}^{n} |e_t|$$

Değerler yerine konulduğunda Ortalama Mutlak Hata değeri 17 olarak hesaplanır.

$$MAE = \frac{170}{10}$$

MAE değeri ne kadar düşük olursa modelin doğruluğu o kadar yüksektir.

MAE aykırı değerlere karşı daha iyi sonuçlar verir.

MAPE: Ortalama Mutlak Yüzde Hata (Mean Absolute Percentage Error)

Modeldeki hataların gerçek tahminlere oranının yüzdelik değeridir. Bir tahmin yapıldığında bu tahmin ile gerçek değerdeki farkın (hata) gerçek değere oranı yüzdelik hata değerini verir. Gerçek değeri 50 olan bir aracın değeri 70 olarak tahmin edilirse mutlak yüzde hata (70 -50) / 50 % 40 olarak bulunur. Gerçek değer %40 hatayla tahmin edilmiş olur. Bu şekilde tüm değerler için hesaplanarak ortalaması bulunur.

$$\text{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n} \left|\frac{e_t}{y_t}\right|$$

Değerler yerine konulduğunda Ortalama Kare Hata 13,5 olarak hesaplanır. Bunun anlamı modelin gerçek değerleri ortalama %13,5 hata ile tahmin edebildiğidir.

$$\text{MAPE} = \%\frac{1,35}{10}$$

RMSE: Root Mean Squared Error (Kök Ortalama Kare Hata)

MSE değerinin karekökü alınarak hesaplanır.

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$$

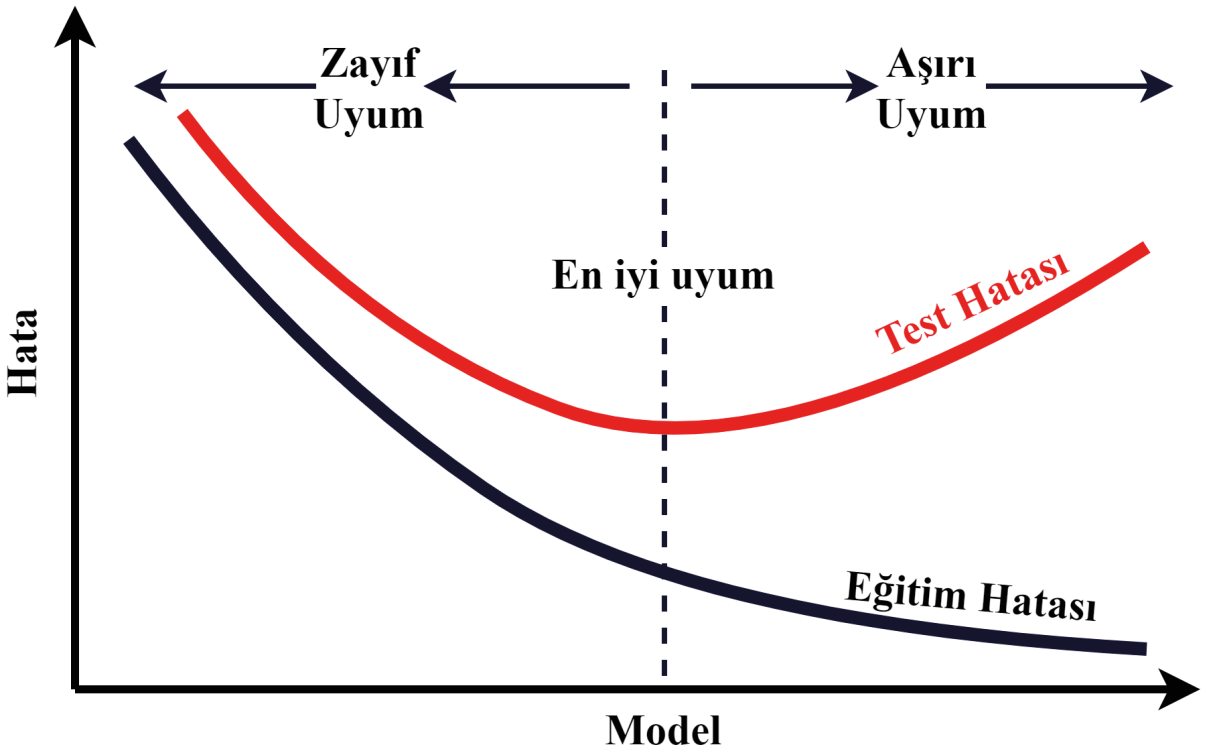
Değerler yerine konulduğunda RMSE değeri 21,56 olarak hesaplanır.

$$RMSE = \sqrt{\frac{4650}{10}}$$

2.1.5.3. Aşırı Öğrenme ve Zayıf Öğrenme

Bir modelin hem eğitim verisindeki hem de test verisindeki hata oranı yüksekse bu modelin yetersiz olduğu söylenebilir. Buna **zayıf öğrenme** (yetersiz öğrenme, underfitting) denir.

Test veri setindeki tahmin hatası eğitim veri setinden oldukça yüksekse buna **aşırı öğrenme** (ezberleme, overfitting) denir (Görsel 2.15).



Görsel 2.15: Aşırı ve zayıf öğrenme

Overfitting: Model eğitim esnasında aşırı uyum göstermiş veriyi ezberlemiştir. Bundan kaçınmak için öz nitelik sayısının azaltılması (feature seletion), yeni veri ekleme, Regulation (düzenleme) droupout, early learning vb. yöntemler kullanılabilir.

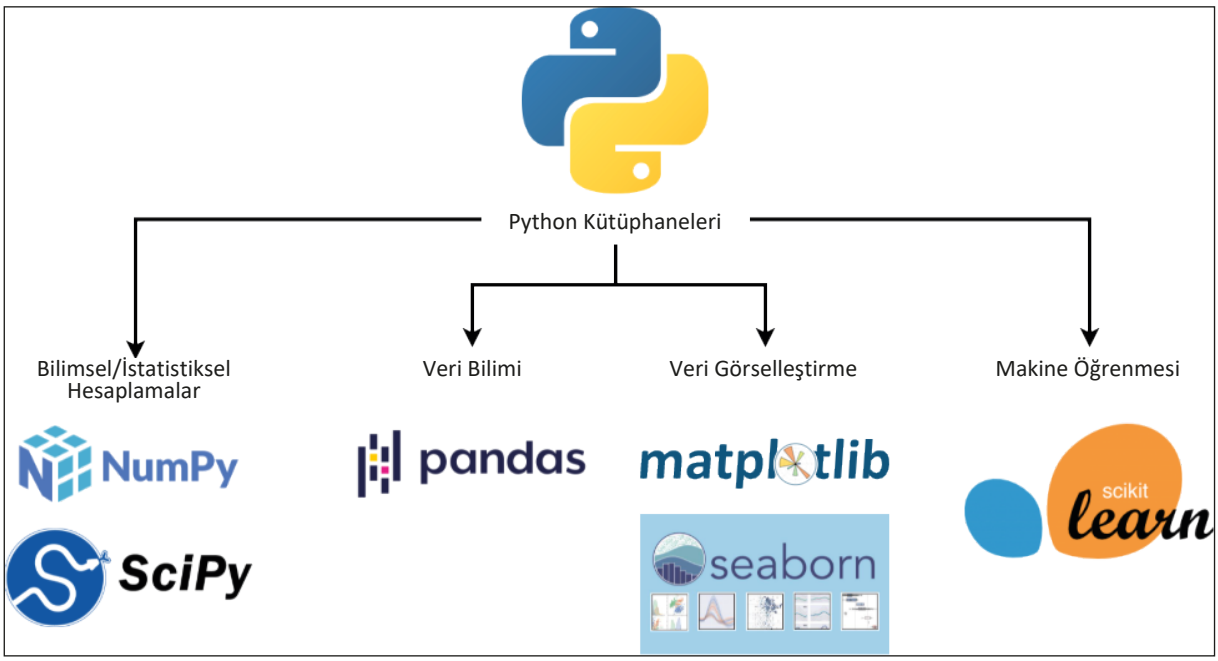
Zayıf öğrenme ile karşılaşıldığında farklı makine öğrenmesi algoritmaları kullanılarak yeni modeller geliştirilir. Aşırı öğrenmeden kaçınmak için çapraz doğrulama yöntemi veya farklı doğrulama seti kullanılması yararlı olacaktır. En iyi uyum, eğitim ve test veri setindeki hata oranın düşük olduğu alandır.

2.1.6. Makine Öğrenmesi İçin Gerekli Yazılımlar

Bu bölümde makine öğrenmesi uygulamaları geliştirmek için kullanılabilecek yazılımlar ve platformlar tanıtılmaktadır. Makine öğrenmesi için farklı programlama dilleri ve platformlar kullanılır. Bu öğrenme birimindeki uygulamalar için yaygın olarak kullanılan platformlardan **Google Colab Not Defteri** kullanılmıştır. Programlama dili olarak da **Python** seçilmiştir. Platform ve programlama dili; açık kaynak kodlu, işlevsel ve ücretsiz olmaları nedeniyle tercih edilmiştir. Python; geniş topluluk desteği, kütüphane zenginliği ve işlevselliği nedeniyle özellikle son yıllarda programlama, makine öğrenmesi ve yapay zekâ alanlarında tercih edilen bir programlama dilidir. Öğrenme birimindeki uygulamalarda bu yazılımlar kullanılmıştır.

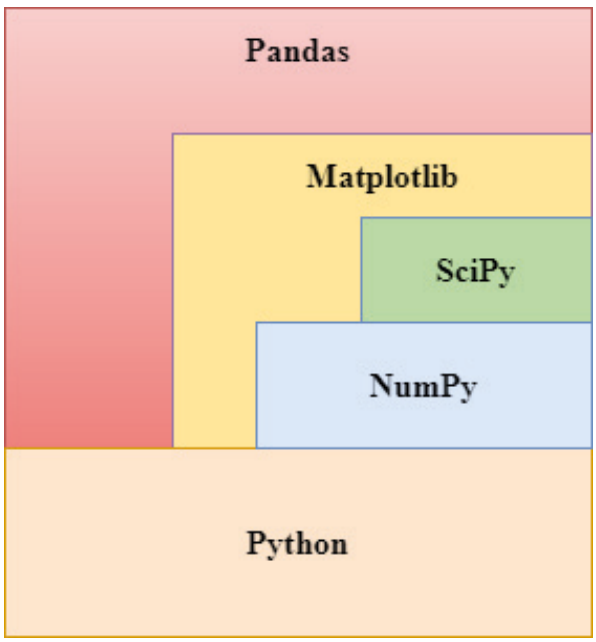
2.2. MAKİNE ÖĞRENMESİ İLE İLGİLİ KÜTÜPHANELER

Python'da bilimsel / istatistiksel hesaplamalar, veri bilimi, veri görselleştirme ve makine öğrenmesi uygulamalarında sık kullanılan kütüphaneler Görsel 2.16'da verilmiştir.



Görsel 2.16: Veri bilimi ve makine öğrenmesi ile ilgili kütüphaneler

Makine öğrenmesi için kullanılan kütüphaneler, bilimsel / istatistiksel, veri bilimi, veri görselleştirme ve makine öğrenmesi başlıkları altında toplanabilir. Bu kütüphaneler birbirleriyle ilişkilidir ve kütüphaneler birbiri üzerine inşa edilmiştir. Temelde ise Python yer alır (Görsel 2.17).

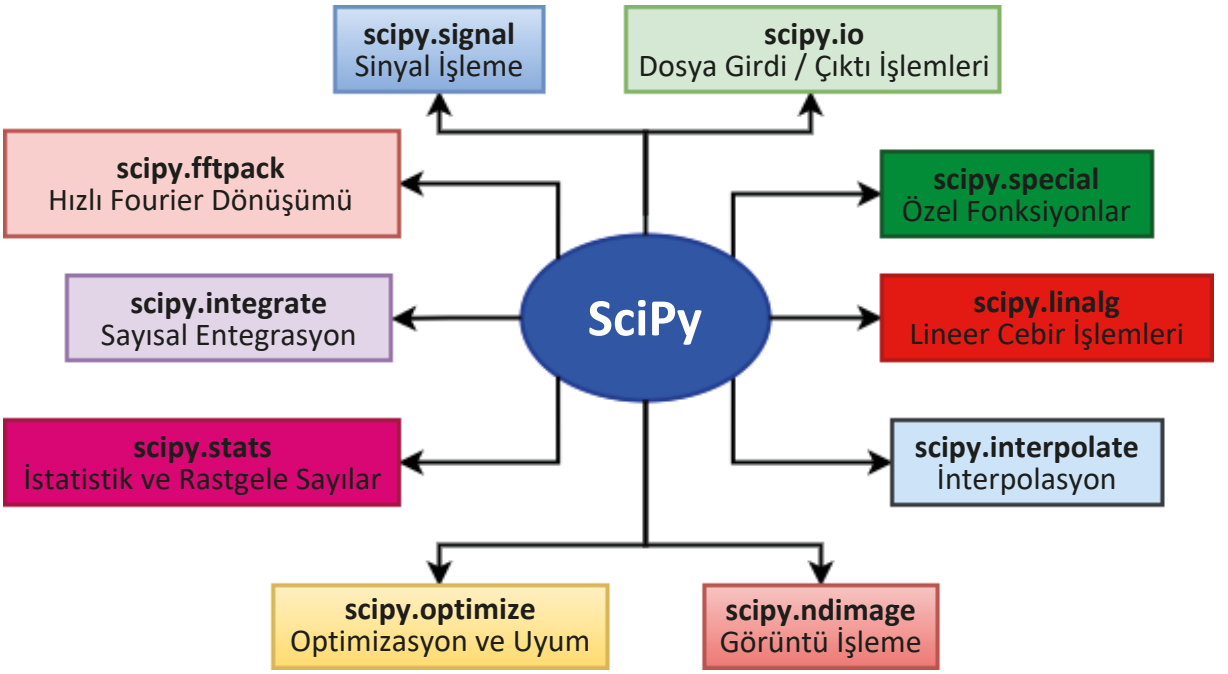


Görsel 2.17: Makine öğrenmesinde kullanılan kütüphaneler

NOT !!! Scikit-learn bir SciPy eklentisi olduğu için ayrıca yer verilmemiştir. Seaborn ise matplotlib tabanlı bir kütüphanedir.

2.2.1. SciPy Kütüphanesi

SciPy, Scientific Python'ın kısaltmasıdır. Bilimsel Python anlamına gelen SciPy, bilimsel hesaplama araçları, sayısal algoritmalar, sinyal işleme, optimizasyon ve istatistik araçları içerir. SciPy, NumPy üzerine inşa edilmiş ve daha gelişmiş bir kütüphanedir. SciPy içindeki başlıca alt modüller ve kullanım amaçları aşağıdaki gibidir (Görsel 2.18).



Görsel 2.18: SciPy kütüphanesi

2.2.2.1. SciPy Kurulumu ve İçe Aktarılması

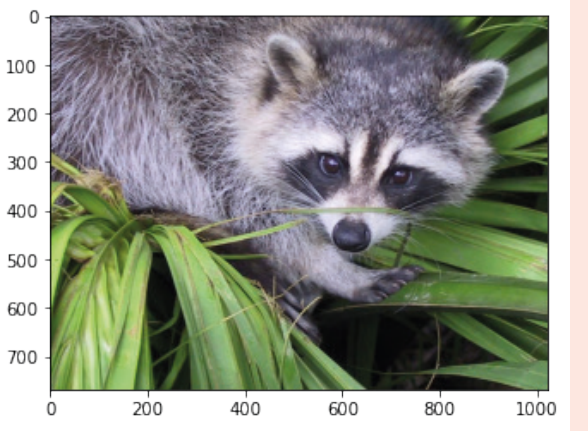
Sistemde SciPy kurulu değilse Python üzerinden **pip install scipy** komutu ile kurulur. Jupyter Notebook üzerinde kurulum yapmak için **!pip install scipy** komutu kullanılır. Anaconda platformu üzerinde Jupyter Notebook ile çalışılıyorsa SciPy kütüphanesi kurulu olarak gelir.

10. ÖRNEK

```
import scipy
print(scipy.__version__)
```

Çıktı (Görsel 2.19)

```
1.4.1
```



Görsel 2.19: SciPy örnek resim

11. ÖRNEK

```
from scipy import misc
img = misc.face()
# veri türünü yazdırma.
print(type(img))
# diziyi yazdırma.
print(img)
```

Çıktı

```
<class 'numpy.ndarray'>
[[[121 112 131]
[138 129 148]
[153 144 165]
...
[119 126 74]
[131 136 82]
[139 144 90]]]
```

Örnekte görüleceği üzere SciPy kütüphanesinde yer alan resim bir NumPy dizisi olarak tutulmaktadır. Dizinin özelliklerine bakılarak daha çok bilgi alınabilir.

12. ÖRNEK

```
print("Dizinin boyutu:",img.ndim)
```

Çıktı

Dizinin boyutu: 3

13. ÖRNEK

```
print("Dizi veri tipi:",img.dtype)
```

Çıktı

Dizi veri tipi: uint8

```
print("Dizi şekil:",img.shape)
```

Çıktı

```
Dizi şekil: (768, 1024, 3)
```

Kodların çıktısı incelendiğinde resmin 768 satır x 1024 sütundan oluşan üç boyutlu bir dizi olduğu görülür. Her öge 3 elemandan (RGB) oluşan bir listeden oluşmaktadır. Bu sayılar kırmızı, yeşil ve mavi renkler için 0-255 aralığında bir sayıdan oluşmaktadır.

```
Dizinin ögeleri:
   R G B
   [138 129 148]
```

SciPy üzerinde scikit-learn ve scikit-image gibi bilimsel eklenti paketleri geliştirilmiştir. Makine öğrenmesi uygulamaları geliştirirken scikit-learn kütüphanesi yaygın olarak kullanıldığından o bölümde daha fazla örneğe yer verilmiştir.

2.2.2. Seaborn Kütüphanesi

Seaborn, Matplotlib tabanlı bir Python gelişmiş veri görselleştirme kitaplığıdır. Seaborn kullanılarak estetik ve bilgilendirici grafikler çizilir. Seaborn, Pandas veri yapılarıyla bütünleşik olarak çalışır. Veri dağılımını görmede, veriyi keşfetmede, estetik ve bilgilendirici grafikler oluşturmada çok kullanışlıdır.

2.2.2.1. Seaborn Kurulumu ve İçe Aktarılması

Sistemde Seaborn kurulu değilse Python üzerinden **pip install seaborn** komutu ile kurulur. Jupyter Notebook üzerinde kurulum yapmak için **!pip install seaborn** komutu kullanılır. Anaconda platformu üzerinde Jupyter Notebook ile çalışılıyorsa Seaborn kütüphanesi kurulu olarak gelir. Seaborn kütüphanesi içinde hazır veri setleri bulunur. Örnekte Seaborn kütüphanesi, gerekli diğer kütüphanelerin içe aktarılması ve Seaborn'de bulunan örnek veri setleri listelenmiştir.

14. ÖRNEK

```
# Seaborn kütüphanesinin içe aktarılması.
import seaborn as sns
# Matplotlib kütüphanesinin içe aktarılması.
import matplotlib.pyplot as plt
# Seaborn hazır veri setleri.
print(sns.get_dataset_names())
# Uyarıların kapatılması.
import warnings
warnings.filterwarnings('ignore')
```

Çıktı

```
['anagrams', 'anscombe', 'attention', 'brain_networks',
 'car_crashes', 'diamonds', 'dots', 'exercise', 'flights', 'fmri',
 'gammas', 'geyser', 'iris', 'mpg', 'penguins', 'planets', 'tips',
 'titanic']
```

Görselleştirme örneklerinde Seaborn'de bulunan tips veri seti kullanılmıştır. Tips, bir restoranda müşterilerin verdikleri bahşişlere ilişkin bilgilerin tutulduğu veri setidir. 2. örnekte bir veri seti yüklenerek incelenmiştir.

15. ÖRNEK

```
# Veri çerçevesi tanımlanıyor.
bahsis_veri= sns.load_dataset('tips')
# Arka planda Pandas'ı kullanmaktadır.
print(type(bahsis_veri))
print(bahsis_veri.head())
```

Çıktı

```
<class 'pandas.core.frame.DataFrame'>
```

	total_bill	tip	sex	day	time	size
0	16.99	1.01	Female	Sun	Dinner	2
1	10.34	1.66	Male	Sun	Dinner	3
2	21.01	3.50	Male	Sun	Dinner	3
3	23.68	3.31	Male	Sun	Dinner	2
4	24.59	3.61	Female	Sun	Dinner	4

Veri setindeki sütunların adları Türkçeye çevrilerek verinin anlaşılması sağlanabilir.

16. ÖRNEK

```
# column sözlükte anahtar olarak eski kolon adları, değer olarak
yeni kolon anahtarları verilir.
# inplace = True ile değişikliklerin veri çerçevesinde kalıcı
olması sağlanır.

bahsis_veri.rename(columns={'total_bill'  :  'fatura_toplami',
                            'tip'         :  'bahsis',
                            'sex'         :  'cinsiyet',
                            'gum'         :  'sakız',
                            'day'         :  'gun',
                           'time'         :  'zaman',
                           'size'         :  'kisi_sayisi'
                    }, inplace=True)
```

Sütun isimlerinin değiştiği görülebilir.

17. ÖRNEK

```
print(bahsis_veri.head())
```

Çıktı

	fatura_toplami	bahsis	cinsiyet	sakız	gun	zaman	kisi_sayisi
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

Veri seti hakkında bilgi almak için **info** metodu kullanılır.

18. ÖRNEK

```
print(bahsis_veri.info())
```

Çıktı

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
#        Column         Non-Null Count      Dtype
---      -------        --------------      ------
0    fatura_toplami      244 non-null      float64
1        bahsis          244 non-null      float64
2       cinsiyet         244 non-null     category
3         sakız          244 non-null     category
4          gun           244 non-null     category
5         zaman          244 non-null     category
6     kisi_sayisi        244 non-null       int64
dtypes: category(4), float64(2), int64(1)
memory usage: 7.3 KB
None
```

Veri setinde eksik veri olmadığı görülmektedir. Fatura toplamı ve bahşiş ondalık sayı, kişi sayısı ise tam sayı veri tipindedir. Cinsiyet, sakız, gün ve zaman ise kategorik veri tipindedir. Kategoriler verideki grupları ifade eder. Örnek: Cinsiyet Female, Male. Female=Kadın, Male=Erkek. Veri setindeki sütunların açıklaması aşağıda verilmiştir.

```
fatura_toplami: Fatura miktarı
bahsis        : Bahşiş miktarı
cinsiyet      : Müşterinin cinsiyeti (Female, Male)
gun           : Gün adları (Thur, Fri, Sat, Sun )
zaman         : Yemek zamanı: Öğle, Akşam (Lunch, Dinner)
kişi_sayisi   : Yemekteki kişi sayısı
```

describe metodu kullanılarak veri setindeki sayısal özeliklere (sütun, kolon) ait tanımlayıcı istatistikler görülebilir ve veri hakkında bilgi edinilebilir.

19. ÖRNEK

```
print(bahsis_veri.describe())
```

Çıktı

	fatura toplamı	bahsis	kisi_sayisi
count	244.000000	244.000000	244.000000
mean	19.785943	2.998279	2.569672
std	8.902412	1.383638	0.951100
min	3.070000	1.000000	category
25%	13.347500	2.000000	category
50%	17.795000	2.900000	category
75%	24.127500	3.562500	1.000000
max	50.810000	2.000000	int64

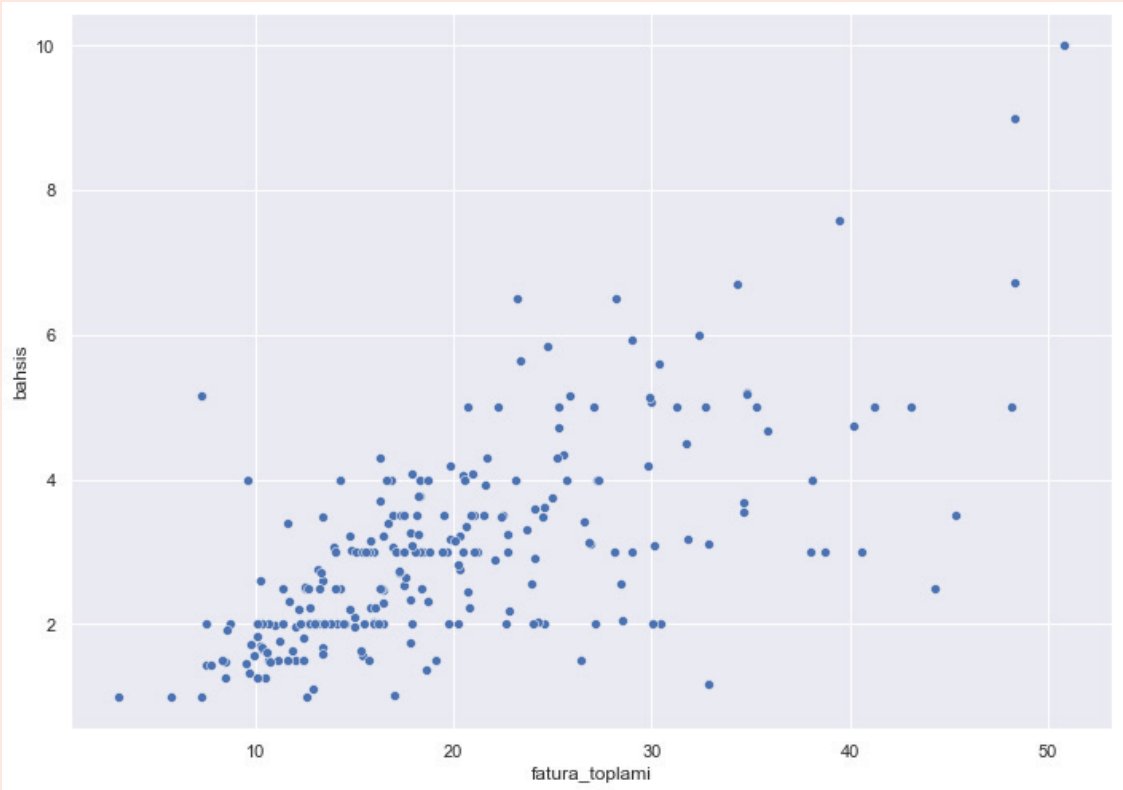
2.2.2.2. Seaborn ile Veri Görselleştirme

Bahşiş ve fatura toplamı **scatter** (saçılım) grafiğini çizmek için **scatterplot** metodu kullanılır. **data** parametresine veri çerçevesi, x ve y parametrelerine değer olarak sütun adları verilir.

20. ÖRNEK

```
sns.scatterplot( data = bahsis_veri, x = 'fatura_toplami',
                 y = 'bahsis')
```

Çıktı (Görsel 2.20)



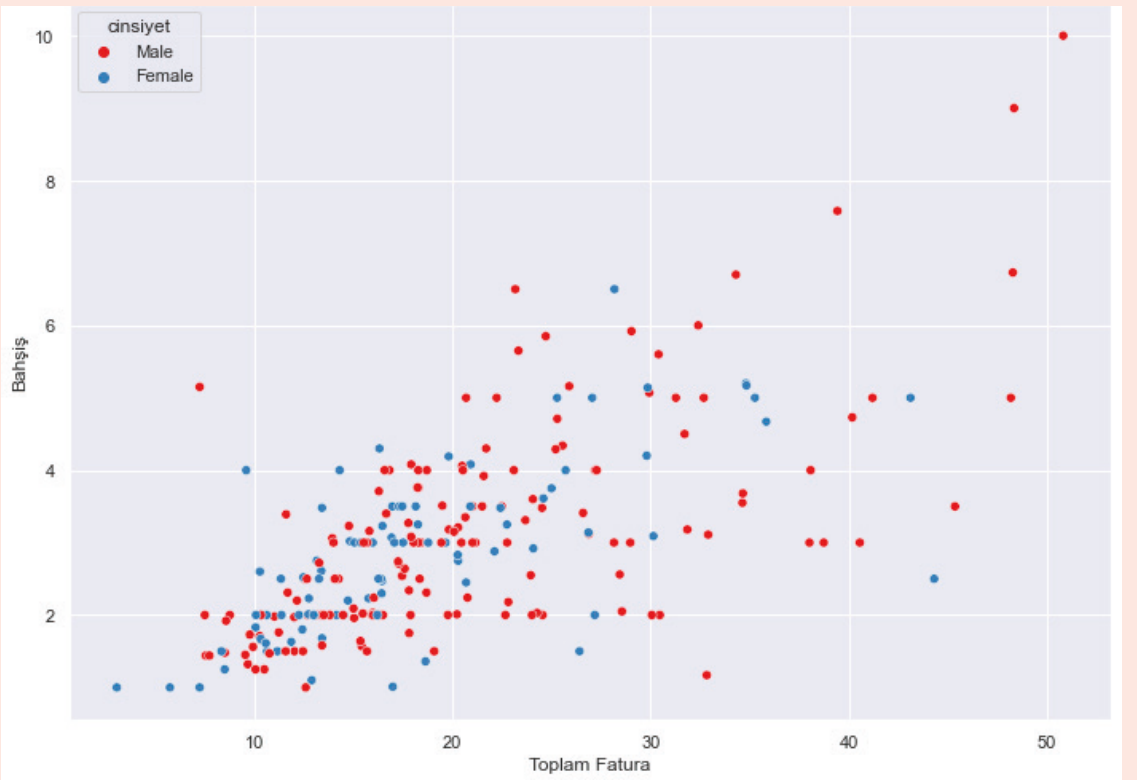
Görsel 2.20: Bahşiş fatura toplamı saçılım grafiği

Çizim alanında eksenlerin rengi, ızgaranın (grid) etkin olup olmadığı ve diğer estetik görünümler **set_style** metodu ile belirlenir. Seaborn'e seçilebilecek stiller; **darkgrid, whitegrid, dark, White** ve **ticks** olarak sıralanabilir. Scatterplot metodunda palette parametresi figürler için renk paleti belirlemek amacıyla kullanılır. **set** parametresinde **rc** grafiğin boyutunu belirlemek için kullanılır. **color** parametresi ise grafikteki çizgilerin ve noktaların rengini belirtmek için kullanılır. **hue** parametresi veriyi kategorik olarak gruplandırmak için kullanılır. Cinsiyete göre bahşiş toplam fatura saçılım grafiği örnekteki gibi çizdirilebilir.

21. ÖRNEK

```
# stil seçme.
sns.set_style("whitegrid")
# Grafik boyutu.
sns.set(rc={'figure.figsize':(11.7,8.27)})
# Grafik seçme.
sns.scatterplot( data = bahsis_veri, x = 'fatura_toplami',
y = 'bahsis', hue = 'cinsiyet', color = 'b', palette = 'Set1',)
# sns.regplot(x='fatura_toplami', y='bahsis', data=bahsis_veri
marker='+', scatter_kws={'alpha':0.1})
# y ekseni başlığı.
plt.ylabel("Bahşiş")
# x ekseni başlığı.
plt.xlabel("Toplam Fatura")
# Grafiğin başlığı.
plt.title('Cinsiyete Göre Bahşiş Fatura Toplamı Saçılım Grafiği')
# Seaborn kullanılarak oluşturulan grafiklerin gösterilmesi.
plt.show()
```

Çıktı (Görsel 2.21)



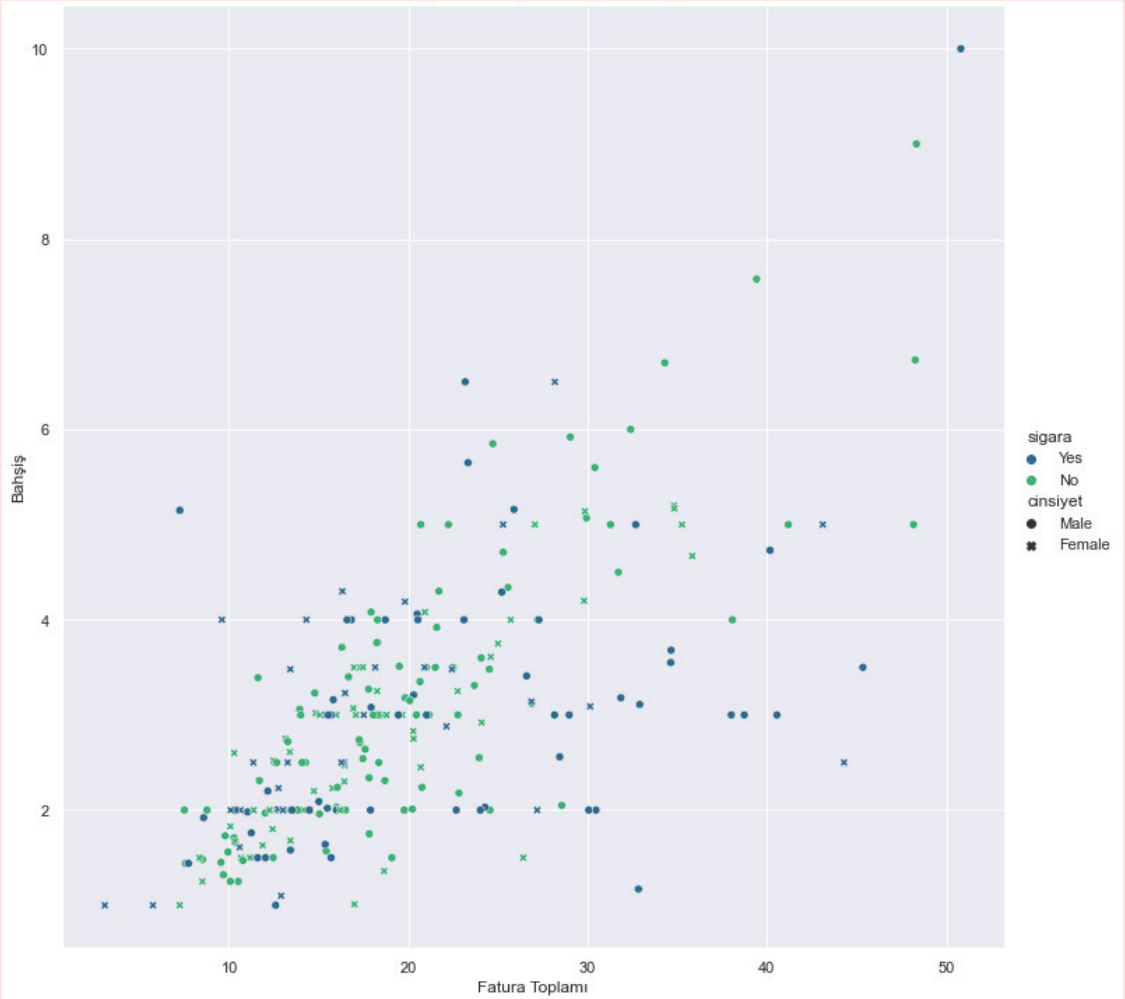
Görsel 2.21: Cinsiyete göre bahşiş fatura toplamı saçılım grafiği

relplot metodu özellikler arası ilişkilerin görselleştirilmesinde **scatterplot**tan daha esnektir. height grafik büyüklüğünü belirlemek için kullanılır. **style** ise kategorik olarak **hue**'dan başka ikinci bir kategorik gösterim olanağı verir.

22. ÖRNEK

```
sns.relplot(x = "fatura_toplami", y = "bahsis", hue = "sakız",
style = "zaman", data = bahsis_veri, palette = 'viridis',
height=10)
plt.ylabel("Bahşiş")
plt.xlabel("Fatura Toplamı")
plt.title('Bahşiş - Fatura Toplamı Saçılım Grafiği')
plt.show()
```

Çıktı (Görsel 2.22)



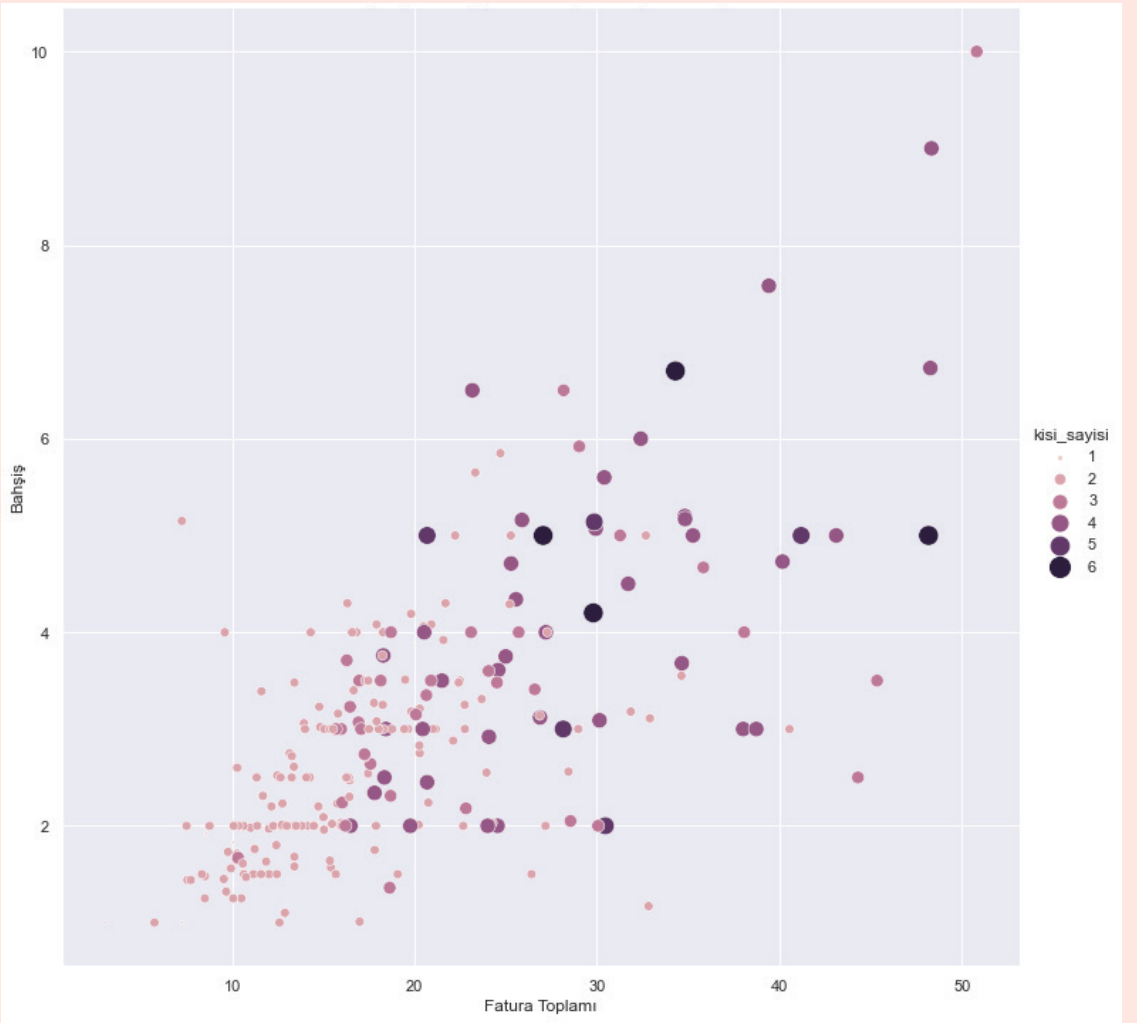
Görsel 2.22: Bahşiş fatura toplamı (Cinsiyet)

sizes parametresi sayısal büyüklüğü göstermek için kullanılmaktadır. Bahşiş–fatura toplamı saçılımı kişi sayısı büyüklüğünde 23. örnekteki gibi görselleştirilebilir.

23. ÖRNEK

```
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.relplot(x = "fatura_toplami", y = "bahsis",
size = "kisi_sayisi", sizes = (5, 200), hue = "kisi_sayisi",
height = 10, data = bahsis_veri)
plt.ylabel("Bahşiş")
plt.xlabel("Fatura Toplamı")
plt.title('Kişi Sayısı Bahşiş-Fatura Toplamı Saçılım Grafiği')
plt.show()
```

Çıktı (Görsel 2.23)



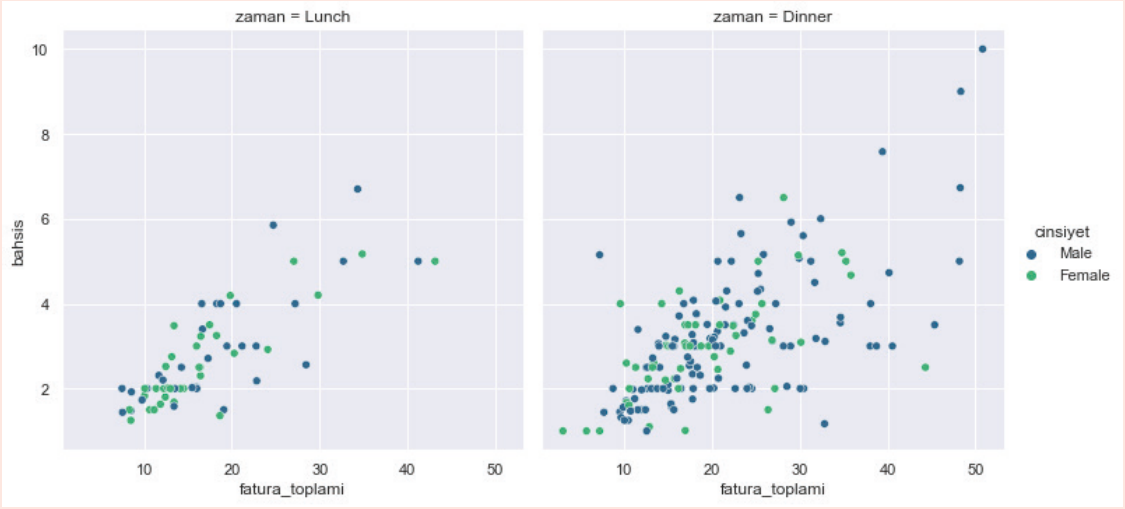
Görsel 2.23: Kişi sayısı bahşiş–fatura toplamı saçılım grafiği

relplot metodunda **col** parametresi kullanılarak kategorilere göre alt grafikler oluşturulabilir.

24. ÖRNEK

```
sns.relplot(x = ”fatura_toplami”, y = ”bahsis”, hue = ”cinsiyet”,
col = ”zaman”, data = bahsis_veri, palette = ’viridis’)
plt.show()
```

Çıktı (Görsel 2.24)



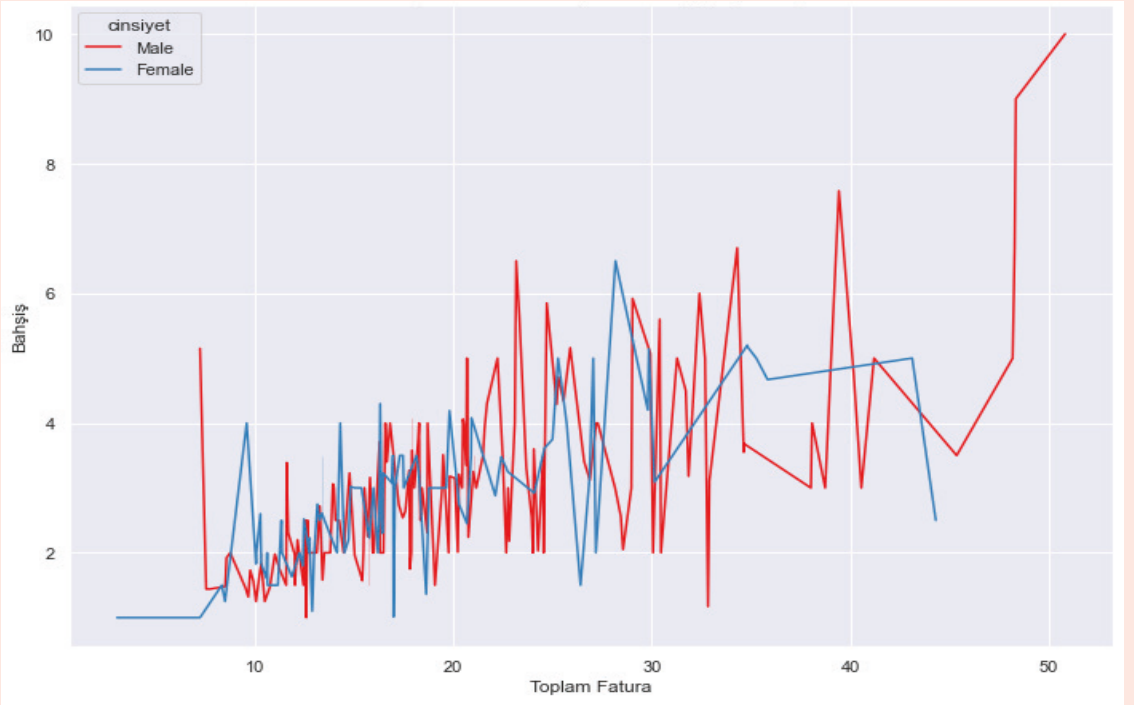
Görsel 2.24: Bahşiş-fatura toplamı cinsiyet-zaman saçılım grafiği

lineplot metodu çizgi grafiği çizmek için kullanılır. Cinsiyete göre fatura toplamı–bahşiş çizgi grafiği örnekteki gibi çizilebilir.

25. ÖRNEK

```
sns.lineplot(x = ‘fatura_toplami’, y = ‘bahsis’,
hue = ’cinsiyet’, color = ’b’, data = bahsis_veri,
palette = ‘Set1’)
plt.xlabel(“Toplam Fatura”)
plt.ylabel(“Bahşiş”)
plt.title(“Cinsiyete Göre Fatura Toplamı-Bahşiş Çizgi Grafiği”)
plt.show()
```

Çıktı (Görsel 2.25)

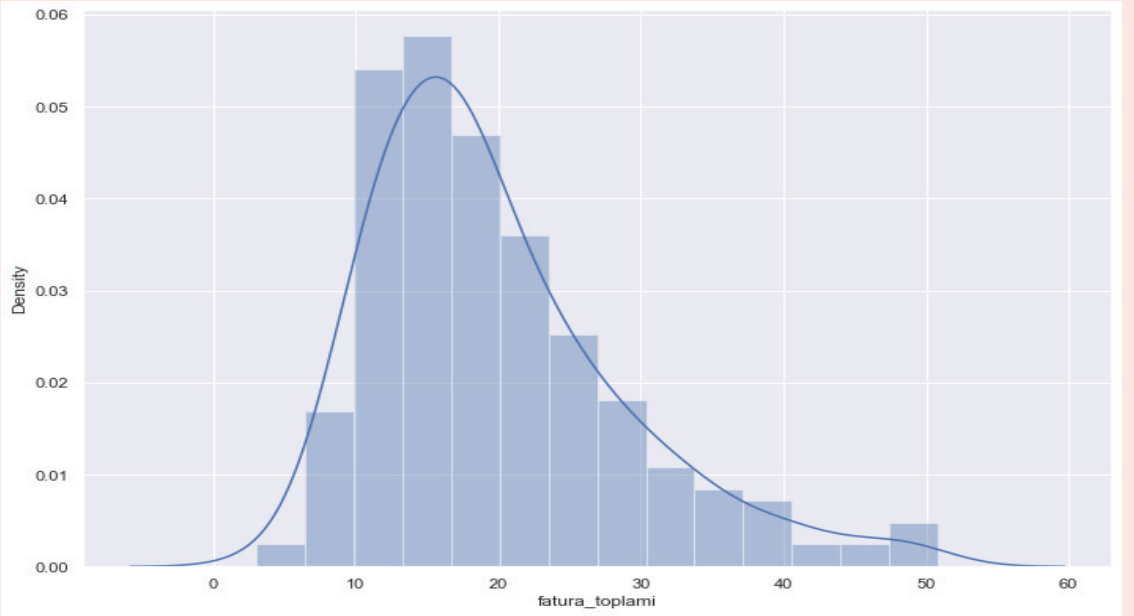


Görsel 2.25: Cinsiyete göre fatura toplamı-bahşiş çizgi grafiği

26. ÖRNEK

```
sns.distplot(bahsis_veri['fatura_toplami'])
```

Çıktı (Görsel 2.26)



Görsel 2.26: Fatura toplamı histogram ve KDE eğrisi

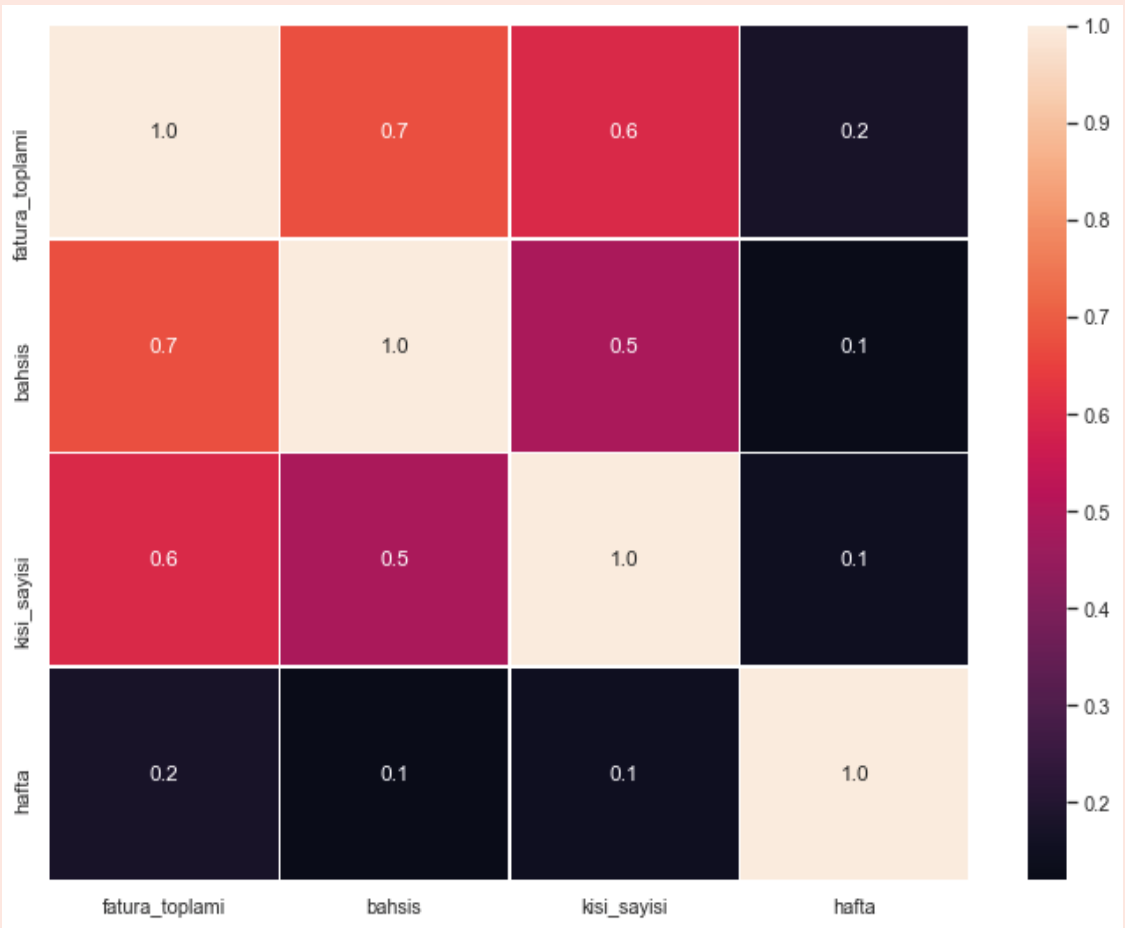
Korelasyon ve Isı Haritası

Isı haritası değişkenler arasındaki korelasyonun görselleştirilmesi için kullanılır. Korelasyon iki değişken arasındaki doğrusal ilişkinin gücünü ve yönünü belirtir. Korelasyon -1 ile +1 arası değerler alır. Korelasyon 0 ise iki değişken arasında ilişki yoktur. Korelasyonun 1 olması tam ilişkiyi gösterir. Korelasyon değerinin artı olması değişkenlerin birlikte azalıp birlikte arttığını ifade eder. Korelasyon değerinin eksi olması zıt yönde bir ilişki olduğunu, değişkenlerden biri artarken diğerinin azaldığını ifade eder. Değişkenler arasında korelasyon örnekteki gibi görselleştirilebilir.

27. ÖRNEK

```
sns.heatmap(bahsis_veri.corr(), annot = True, linewidths = .5,
fmt = '.1f')
plt.show()
```

Çıktı (Görsel 2.27)



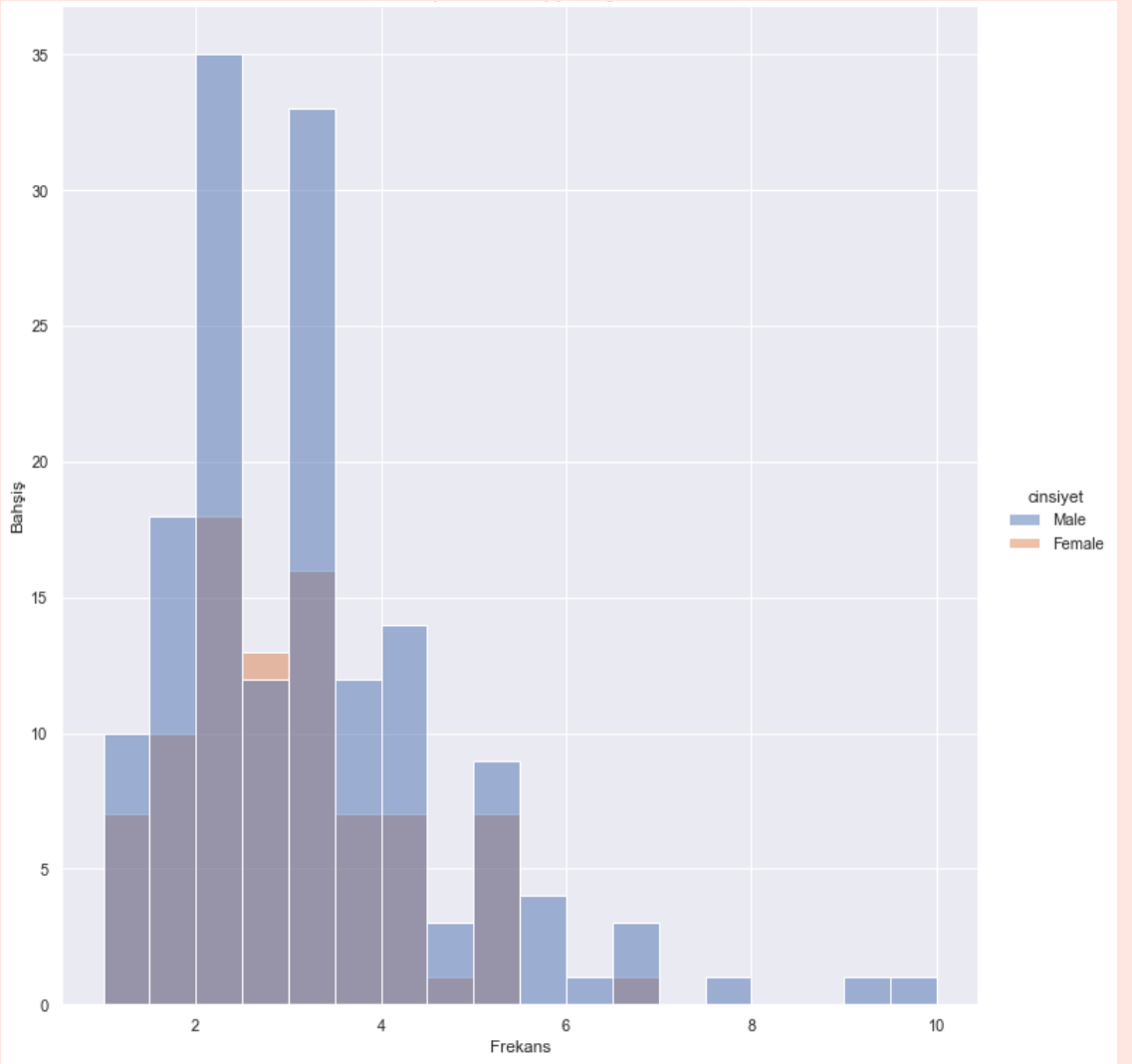
Görsel 2.27: Korelasyon ısı haritası

Cinsiyete göre bahşiş histogramı örnekteki gibi çizilebilir. Histogram ile bahşiş dağılımı cinsiyete göre sütunlar ile görselleştirilmiştir.

28. ÖRNEK

```
sns.displot(bahsis_veri, x = "bahsis", hue = "cinsiyet",
height = 10)
plt.xlabel("Frekans")
plt.ylabel("Bahşiş")
plt.title("Cinsiyete Göre Bahşiş Histogram")
plt.show()
```

Çıktı (Görsel 2.28)



Görsel 2.28: Cinsiyete göre bahşiş histogramı

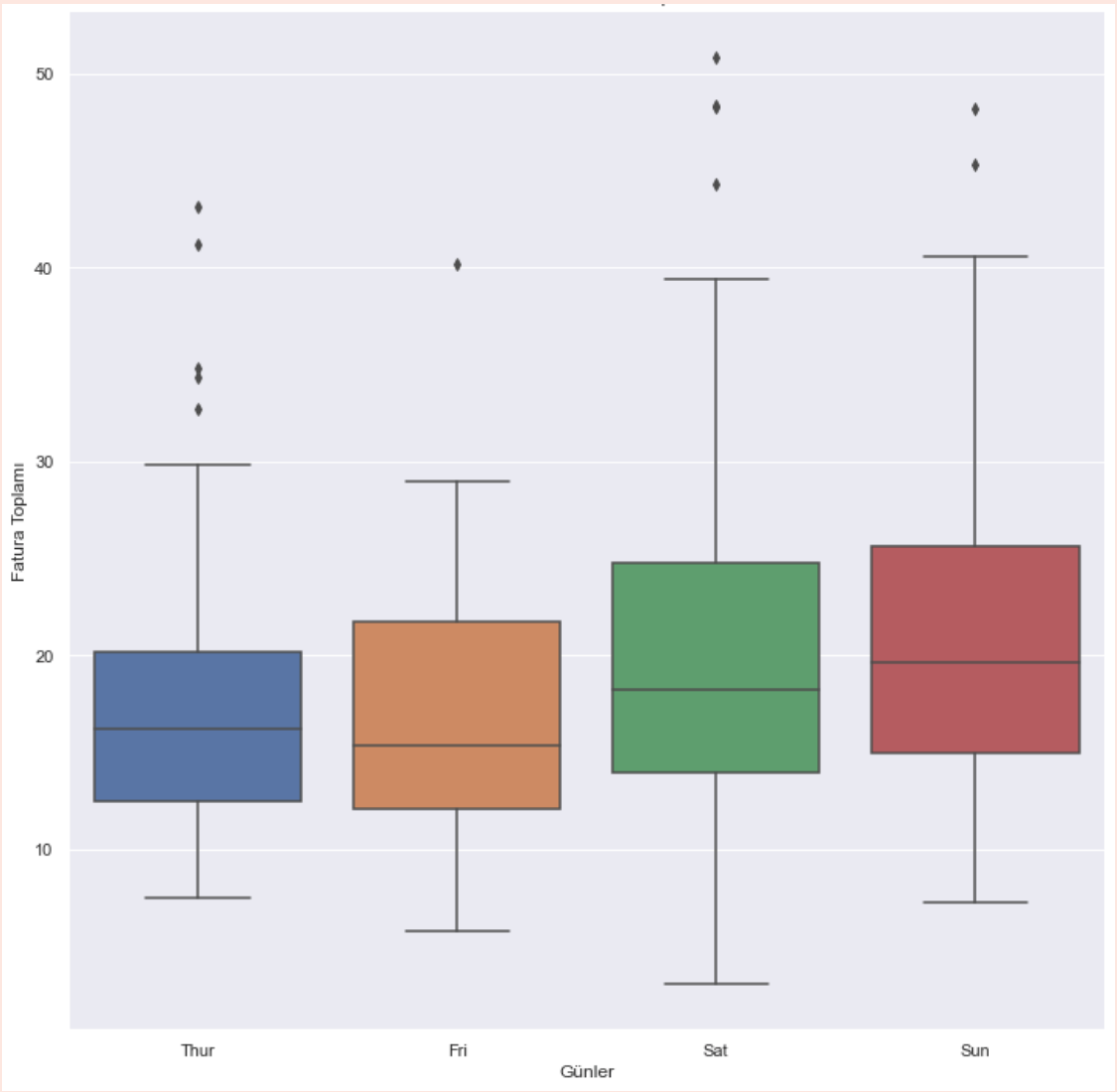
Kategorik özellikler için grafik seçiminde **catplot** metodunda **kind** parametresine **strip, swarm, box, violin, box, point, bar** veya **count** gibi argümanlar verilebilir.

Günlere göre fatura toplamı kutu grafiği örnekteki gibi çizilebilir.

29. ÖRNEK

```
# Günlere göre bahşiş.
sns.catplot(x = "gun", y = "fatura_toplami", kind = "box",
data = bahsis_veri, height = 10)
plt.xlabel("Günler")
plt.ylabel("Fatura Toplamı")
plt.title("Günlere Göre Fatura Toplamı")
plt.show()
```

Çıktı (Görsel 2.29)



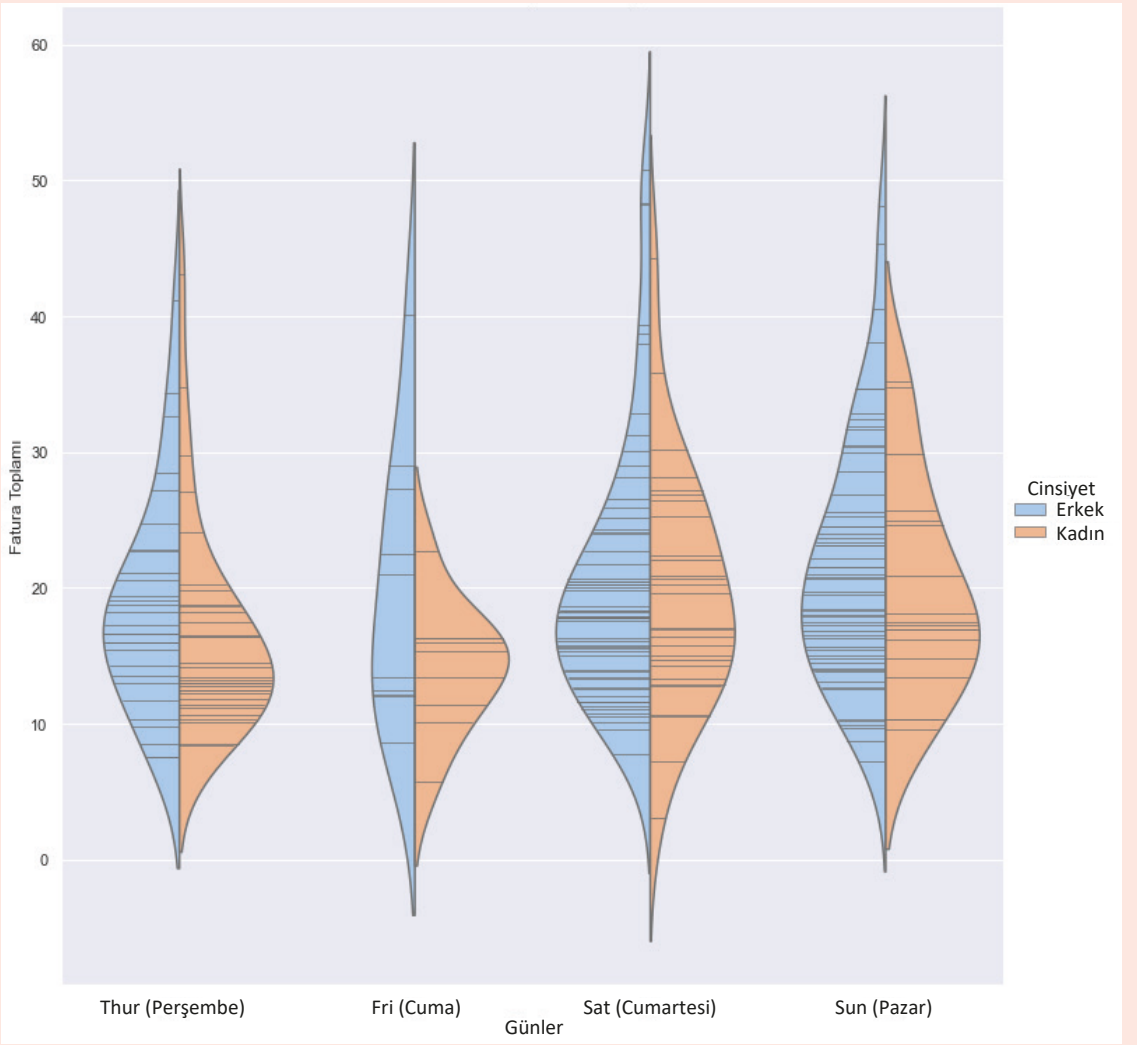
Görsel 2.29: Günlere göre bahşiş kutu diyagramı

Farklı bir görselleştirme yöntemi olarak keman (**violin**) grafiği kullanılabilir. Örnekte **kind** parametresi grafik türünü belirtmek için kullanılmaktadır ve argüman olarak violin verilmiştir. **İnner** parametresi grafik içinde veriyi göstermek için (çizgiler) kullanılmaktadır, argüman olarak **stick** verilmiştir. **split** parametresi farklı kategoriler için grafiği bölmek için kullanılmaktadır. Örnekte cinsiyete göre keman şeklinde bir görselleştirme yapılmıştır ve bu şekilde kategorilere ilişkin veri kolay bir şekilde karşılaştırılabilir.

30. ÖRNEK

```
sns.catplot(x = "gun", y = "fatura_toplami", hue = "cinsiyet",
kind = "violin", inner = "stick", split = True, palette="pastel",
data = bahsis_veri, height = 10)
plt.xlabel("Günler")
plt.ylabel("Fatura Toplamı")
plt.title("Günlere Göre cinsiyet-Fatura Toplamı")
plt.show()
```

Çıktı (Görsel 2.30)



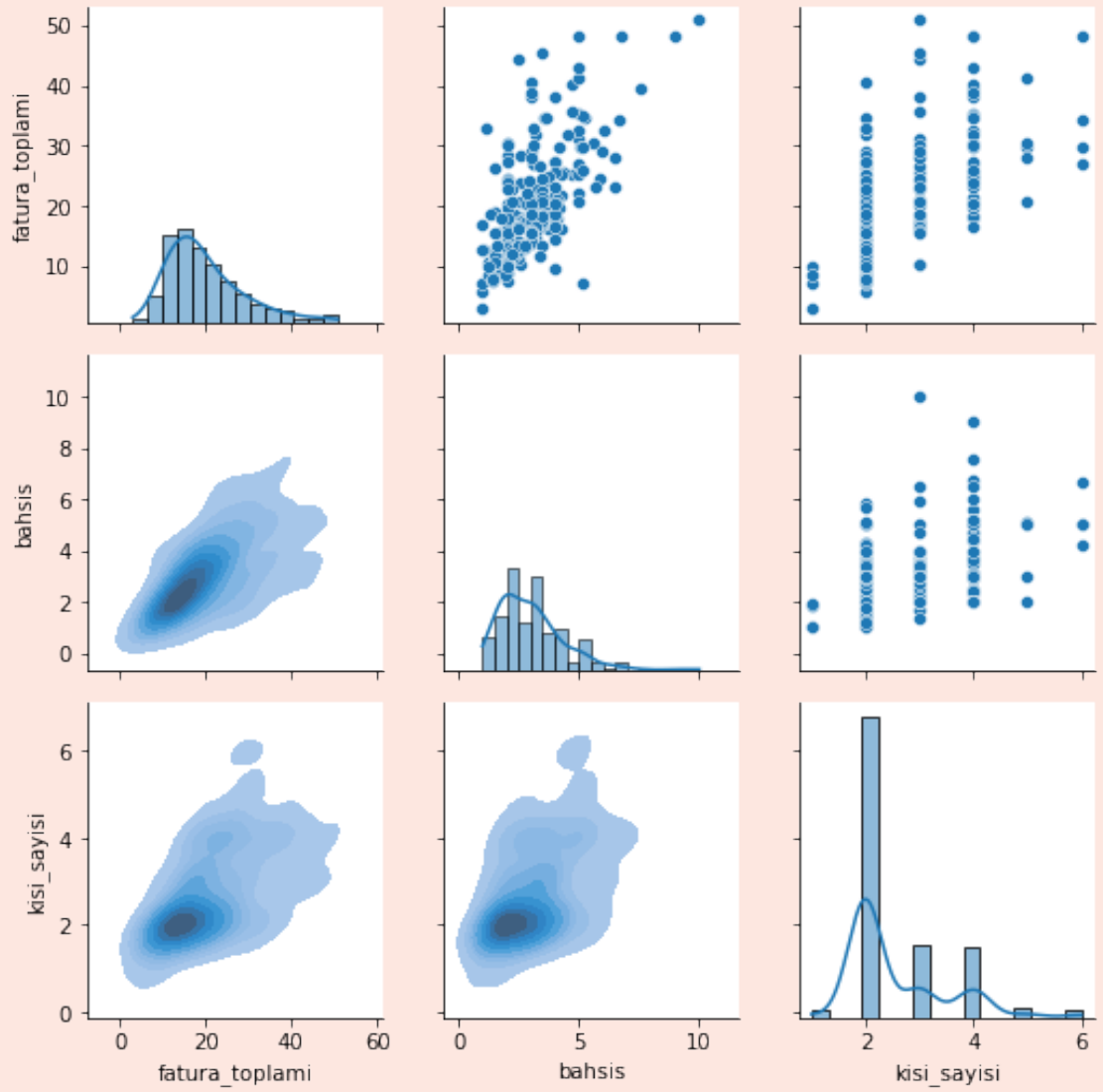
Görsel 2.30: Günlere göre cinsiyet fatura toplamı

Birçok özelliğin birbiriyle ilişkisini birden fazla grafikle görselleştirmek için **PairGrid** metodu kullanılır.

31. ÖRNEK

```
sg = sns.PairGrid(bahsis_veri)
# Digonalın üstünde kalanlar saçılım grafiği olsun.
g.map_upper(sns.scatterplot)
# Digonalın altında kalanlar yoğunluk grafiği olsun.
g.map_lower(sns.kdeplot, fill = True)
# Diogonaldeki grafikler histogram olsun.
g.map_diag(sns.histplot, kde = True)
```

Çıktı (Görsel 2.31)



Görsel 2.31: Çoklu grafik

SIRA SİZDE

Aynı veri seti üzerinde keman (violin) grafiğini kullanarak veri görselleştirme yapınız.

2.2.3. Scikit-learn

Python için temel makine öğrenmesi kütüphanesidir. Scikit-learn kütüphanesi kullanılarak makine öğrenmesi süreci uçtan uca yapılabilir. Scikit-learn kütüphanesinin başlıca NumPy, Pandas, SciPy, Matplotlib ve Seaborn gibi bağımlılıkları bulunur. Scikit-learn bünyesinde bu kütüphaneleri kullanarak makine öğrenmesi aşamalarında gerekli işlemler için modüller sunar.

2.2.3.1. Scikit-learn Kurulumu ve İçe Aktarılması

Sistemde Scikit-learn kurulu değilse Python üzerinden **pip install scikit-learn** komutu ile kurulur. Jupyter Notebook üzerinde kurulum yapmak için **!pip install scikit-learn** komutu kullanılır. Anaconda platformu üzerinde Jupyter Notebook ile çalışılıyorsa Scikit-learn kütüphanesi kurulu olarak gelir. Scikit-learn kütüphanesi örnekte gösterildiği gibi içe aktarılarak versiyonu yazdırılabilir.

32. ÖRNEK

```
import sklearn
print ("Versiyon:", sklearn.__version__)
```

Çıktı

```
Versiyon: 0.23.2
```

NOT !!!

Bu bölümde makine öğrenmesi sürecinde kullanılan temel kütüphanelere yer verilmiştir. Scikit-learn ve diğer kütüphanelere ilişkin uygulamalara makine öğrenmesi süreci içinde yer verilecektir.

2.2.3.2. Makine Öğrenmesi Süreci

Makine öğrenmesi uygulamaları genellikle beş aşamalı bir süreç olarak değerlendirilir. Bu süreçler şunlardır:

- Veri toplama
- Veri ön işleme
- Model oluşturma
- Değerlendirme
- Dağıtım

Veri Toplama

Veri toplama, makine öğrenmesi sürecinde kullanılacak verinin bir veya daha fazla veri kaynağından alınarak (genellikle) yapısal bir veri seti hâline getirilmesi aşamasıdır. İnternet üzerinde çok büyük veri yığınları bulunmaktadır ancak bu veri yığınları yapısal (belirli bir formata uygun) değildir. Yapısal olmayan veriyi işleme için doğal dil işleme, metin madenciliği, büyük veri analizi gibi farklı teknikler kullanılması veya verinin belirli bir formata (veri modeli) dönüştürülmesi gerekir. Yapısal veri, farklı dosya türlerinde depolanmakla birlikte uygulamalarda çoğunlukla CSV formatında dosyalar kullanılmaktadır. Geliştiriciler kendi veri setlerini kullanabilecekleri gibi farklı kişi, kurum ve kuruluşların kullanıma sunduğu veri setlerini de kullanabilirler. Makine öğrenmesi ve veri analizinde kullanılan kütüphaneler pratik yapılabilmesi için örnek veri setleri (toy datasets) içerir. Veri setleri ve makine öğrenmesi uygulamaları için internet toplulukları ve etkin platformlar bulunmaktadır. Kaggle; veri setlerine ulaşmak, makine öğrenmesi uygulamaları geliştirmek ve düzenlenen yarışmalara katılmak için kullanılan popüler bir platformdur. Kaggle'da binlerce veri seti ve makine öğrenmesi uygulaması çalışması, kullanıcılara ücretsiz olarak sunulur.

Veri Ön İşleme

Veri setinin incelenerek modelleme için uygun hâle getirilmesi sürecidir.

Kütüphanelerin Yüklenmesi ve İçe Aktarılması

Makine öğrenmesi ve veri bilimi için temel kütüphanelerin yüklendiği aşamadır. Bu aşamada ihtiyaca göre NumPy, Pandas, Matplotlib, Seaborn ve Scikit-learn kütüphaneleri içe aktarılır. Kütüphaneler sistemde yüklü değilse kurulmalıdır. Modülleri içindeki alt modüllerin ve metotların çakışmaması ve kullanılmayacak modüllerin yer kaplamaması için **from** … **import** … ifadesinin kullanılarak sadece gerekli modüllerin ve alt modüllerin yüklenmesi tavsiye edilir.

33. ÖRNEK

```
# Kütüphanelerin yüklenmesi.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# Scikit-learn veri setlerinin, sınıflandırıcıların ve performans
metriklerinin içe aktarılması.
from sklearn import datasets, svm, metrics
from sklearn.model_selection import train_test_split
# Uyarıların kapatılması.
import warnings
warnings.filterwarnings('ignore')
```

Veri Setinin Yüklenmesi

Veri seti ile ilgili işlemler doğrudan veya dolaylı olarak **Pandas** ve **NumPy** kütüphaneleri kullanılarak yapılır. **Scikit-learn** veya **Seaborn** kütüphaneleri veri yükleme metotlarına sahiptir. Bu metotlar da arka planda **Pandas**, **NumPy** kütüphanelerini veya onları kullanan kütüphaneleri kullanır. Veri setini yüklemek için farklı modüller kullanılabilir. İris veri seti, makine öğrenmesi pratikleri için kullanılan popüler bir veri setidir. Veri setinde İris çiçeğinin üç türüne (virginica, versicolor ve setosa) ait dört niteliğe ilişkin ölçüler bulunur. Dört nitelik; alt yaprak uzunluğu (sepal-length), alt yaprak genişliği (sepal-with), üst yaprak uzunluğu (pedal-length), üst yaprak genişliği (pedal-width) şeklindedir. Veri setinde dört özelliğe ait ölçüler santimetre olarak verilmiştir. Veri setinde bu dört özelliğin yanında çiçeğin türüne ilişkin species (tür) kolonu bulunmaktadır. Örnekte Seaborn kütüphanesinde hazır olarak gelmekte olan İris veri seti yüklenmektedir.

34. ÖRNEK

```
# Seaborn kütüphanesinin içe aktarılması.
import seaborn as sns
# Hazır veri setinin yüklenmesi.
iris = sns.load_dataset('iris')
# Veri setinin ilk satırlarını gösterme.
print (iris.head())
# Nesnenin tipi.
print(type(iris))
print(type(iris))
```

Çıktı

```
   sepal_length  sepal_width  petal_length  petal_width  species
0           5.1          3.5           1.4          0.2   setosa
1           4.9          3.0           1.4          0.2   setosa
2           4.7          3.2           1.3          0.2   setosa
3           4.6          3.1           1.5          0.2   setosa
4           5.0          3.6           1.4          0.2   setosa

<class 'pandas.core.frame.DataFrame'>
```

Pandas ile web üzerindeki veri dosyalarına erişilebilir. Örnekte web üzerindeki iris.csv dosyası Pandas veri çerçevesi olarak yüklenmiştir.

35. ÖRNEK

```
iris = pd.read_csv('https://raw.githubusercontent.com/mwaskom/seaborndata/master/iris.csv')
iris.head()
```

Çıktı

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Scikit-learn içinde yer alan iris veri seti, örnekteki gibi yüklenerek bir **Pandas** veri çerçevesine (data frame) aktarılabilir.

36. ÖRNEK

```
# Kütüphanelerin yüklenmesi.
from sklearn.datasets import load_iris
import pandas as pd
# Veri seti bir numpy dizisi.
veri = load_iris()
print(type(veri.data))
# dataframe (veri çerçevesi) oluşturuluyor.
df = pd.DataFrame(veri.data, columns = veri.feature_names)
df.head()
```

Çıktı

```
<class 'numpy.ndarray'>
```

sepal_length (cm)	sepal_width (cm)	petal_length (cm)	petal_width (cm)
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2

Veri Setinin İncelenmesi

Veri setini incelemek için **head, tail, describe, info** ve **sample** metotları kullanılabilir.

Pandas veri çerçevesine yüklenen iris veri seti incelenebilir.

37. ÖRNEK

```
# veri tipleri ve eksik veri hakkında bilgi alma.
iris.info()
```

Çıktı

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
#        Column          Non-Null Count      Dtype
---      -------         ---------------     ------
0     sepal_length       150 non-null       float64
1      sepal_width       150 non-null       float64
2     petal_length       150 non-null       float64
3      petal_width       150 non-null       float64
4        species         150 non-null       float64
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

38. ÖRNEK

```
# Veri seti özet istatistikler.
iris.describe()
```

Çıktı

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

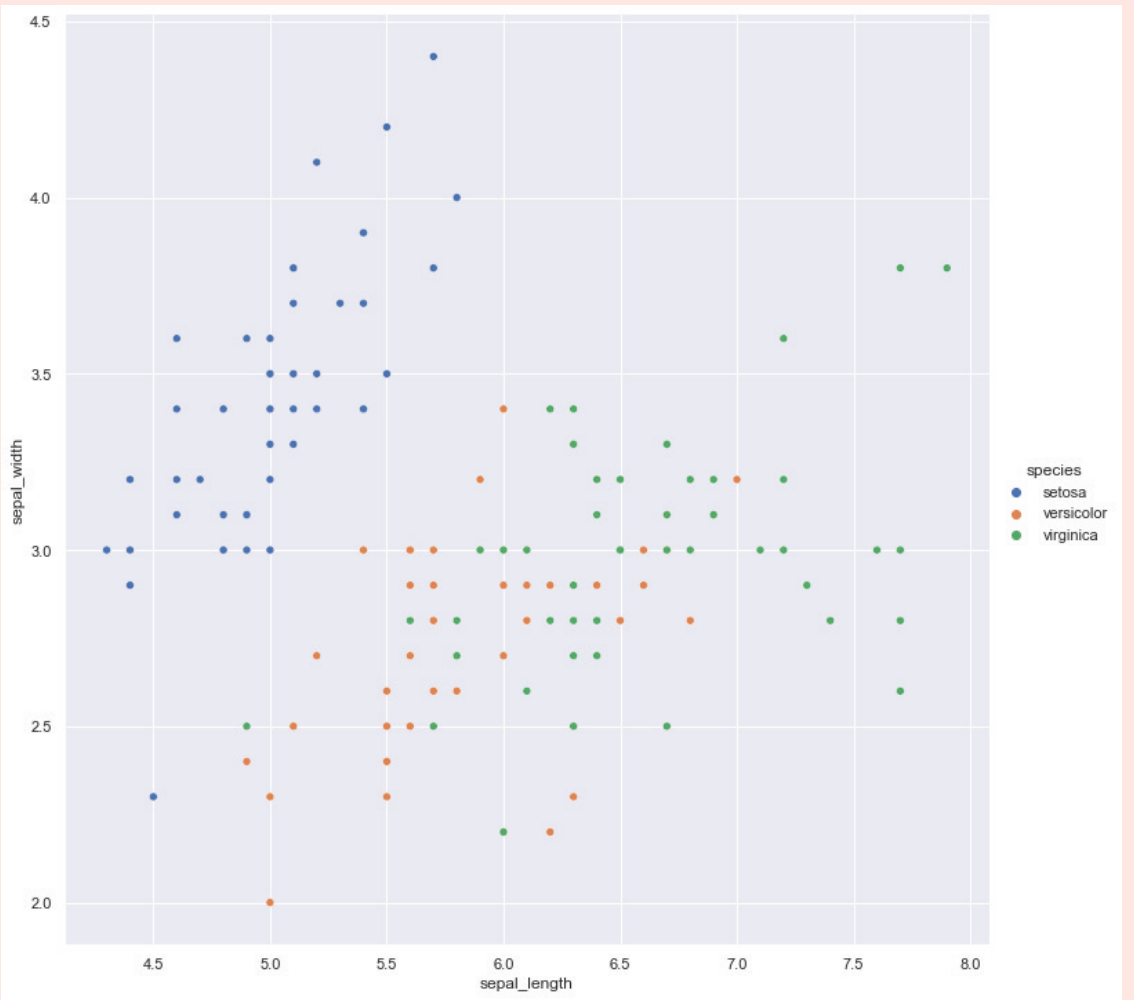
Veri setini incelemek için Seaborn kütüphanesi kullanılarak veri görselleştirme yapılabilir. Veri görselleştirme ile veri setinin anlaşılması kolaylaştırılır. Makine öğrenmesi için kullanılacak yöntemlerin seçiminde yardımcı olur.

39. ÖRNEK

```
# Türlere göre veri görselleştirme.
sns.set_theme() # Tema seçimi.
sns.relplot( data = iris, x = 'sepal_length', y = 'sepal_width',
hue='species', height=10)
```

Çıktı (Görsel 2.32)

```
 #        Column         Non-Null Count      Dtype
---      -------        ---------------     ------
 0     sepal_length       150 non-null      float64
 1      sepal_width       150 non-null      float64
 2     petal_length       150 non-null      float64
 3      petal_width       150 non-null      float64
 4        species         150 non-null      float64
```

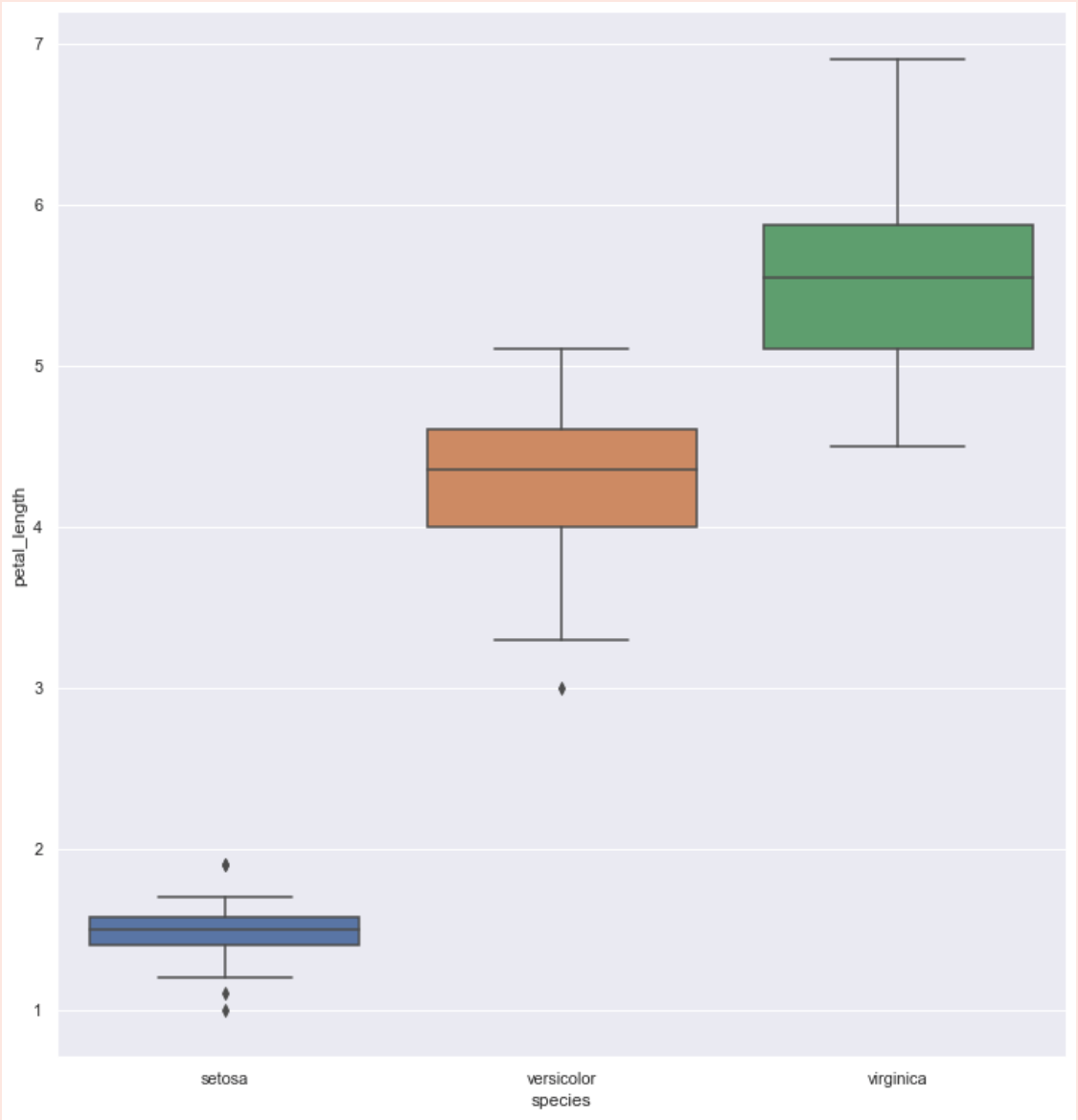


Görsel 2.32: İris veri seti türlerine göre çanak yaprağı saçılım grafiği

40. ÖRNEK

```
# boxplot ile bir özellik petal_lengh özelliğinin incelenmesi.
sns.catplot(x = "species", y = "petal_length", kind = "box",
data = iris, height = 10)
```

Çıktı (Görsel 2.33)



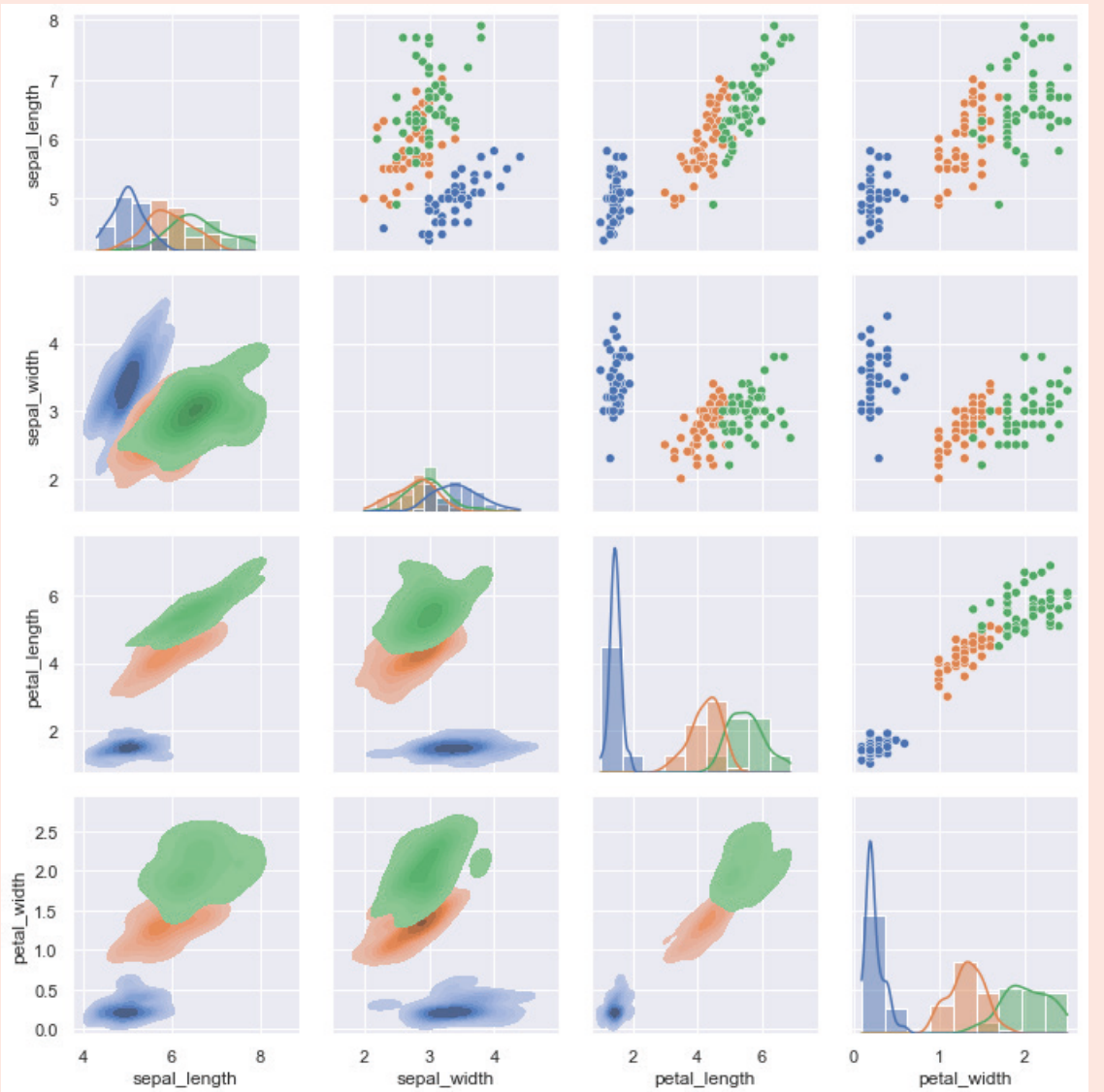
Görsel 2.33: İris veri seti türlerine göre taç yaprak uzunluğu kutu grafiği

Benzer şekilde diğer özellikler için de veri görselleştirme işlemleri yapılarak veri setindeki özellikler arasındaki ilişkiler incelenebilir. Özelliklerin türlere göre ikili görselleştirmeleri tek bir grafikte çoklu olarak yapılabilir.

41. ÖRNEK

```
g = sns.PairGrid(iris, hue = "species")
# diogonalın üstü saçılım grafikleri ile dolu olsun.
g.map_upper(sns.scatterplot)
# Diogonalın altı yoğunluk grafiği olsun.
g.map_lower(sns.kdeplot, fill = True)
# Diogonaldeki grafikler histogram olsun.
g.map_diag(sns.histplot, kde = True)
```

Çıktı (Görsel 2.34)



Görsel 2.34: İris eşleştirme tablo (pair grid) grafiği

Özellik Seçimi ve Eksik Veriler ile İlgili İşlemler

Bir veri seti yüklenerek incelendikten sonra uygulamada kullanılmayacak özellikler belirlenerek Pandas kütüphanesinde anlatılan metotlar (**drop**, **iloc** ve **loc**) kullanılarak veya kolon adları belirtilerek veri setinden çıkarılabilir. Makine öğrenmesi algoritmaları eksik veriyle çalışmaz. Bu yüzden veri seti inceleme işleminde

eksik veri belirlenirse işlem yapılması gerekir. Eksik veri ile ilgili işlemler yapılırken farklı kütüphanelerin modüllerinden yararlanılır. Eksik veri ile ilgili iki tür işlem yapılabilir. Birinci yol eksik veri olan satırların veri setinden çıkarılmasıdır.

42. ÖRNEK

```
# Orijinal veri seti https://www.kaggle.com/'dan uyarlanmıştır.
# veri setinin yüklenmesi.
df_boy_kilo = pd.read_csv('data/500_kisi_boy_kilo_eksik_veri.csv',
sep = ';')
df_boy_kilo.head()
```

Çıktı

	Cinsiyet	Boy	Kilo
0	NaN	140.0	129.0
1	E	140.0	152.0
2	E	140.0	79.0
3	K	140.0	NaN
4	E	140.0	52.0

Örnek veri setinde 500 kişinin cinsiyet, boy ve kilo verisi bulunmaktadır. Veri setindeki satırlarda eksik veri (NaN) bulunmaktadır.

43. ÖRNEK

```
df_boy_kilo.info()
```

Çıktı

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 3 columns):
#        Column         Non-Null Count      Dtype
---      -------        ---------------     ------
0       Cinsiyet         462 non-null       object
1         Boy            450 non-null       float64
2        Kilo            450 non-null       float64
dtypes: float64(2), object(1)
memory usage: 11.8+ KB
```

info metoduyla veri seti hakkında bilgi alındığında cinsiyet sütununda 500-462=38 eksik veri olduğu, boy ve kilo sütunlarında ise 500-450=50 eksik veri bulunduğu görülmektedir. Boy santimetre olarak kilo ise kg olarak verilmiştir. Boy ve kilo numerik (**float64**), cinsiyet ise kategorik (nesne) veri tipindedir.

isnull ve sum metotları birlikte kullanılarak eksik veri sayısı belirlenebilir.

44. ÖRNEK

```
# Eksik veri sayısı.
df_boy_kilo.isnull().sum()
```

Çıktı

```
Cinsiyet 38
Boy 50
Agirlik 50
dtype: int6
```

Veri setindeki sayısal özelliklere ait özet istatistikler **describe** metodu kullanılarak incelenebilir.

45. ÖRNEK

```
# Veri setine ilişkin özet istatistikler.
df_boy_kilo.describe()
```

Çıktı

	Boy	Agirlik
count	450.000000	450.000000
mean	169.893333	105.975556
std	16.387626	32.282327
min	140.000000	50.000000
25%	156.000000	80.000000
50%	170.500000	106.000000
75%	184.000000	136.000000
max	199.000000	160.000000

Pandas kütüphanesinden **dropna** metodu kullanılarak eksik verinin bulunduğu satırlar çıkarılabilir.

46. ÖRNEK

```
# Eksik verilerin çıkarılması.
df_boy_kilo_dna=df_boy_kilo.dropna()
# Aynı veri seti üzerinde değiştirilmek istenirse.
# df_boy_kilo.dropna(inplace=True)
df_boy_kilo_dna.info()
```

Çıktı

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 362 entries, 1 to 497
Data columns (total 3 columns):
#        Column         Non-Null Count      Dtype
---      -------        ---------------     ------
0       Cinsiyet         462 non-null       object
1         Boy            450 non-null       float64
2         Kilo           450 non-null       float64

dtypes: float64(2), object(1)
memory usage: 11.3+ KB
```

Veri setindeki eksik veri çıkarıldıktan sonra satır x, sütun sayısı **shape** metodu ile görülebilir.

47. ÖRNEK

```
# Veri setinin şekli.
df_boy_kilo_dna.shape
```

Çıktı

```
(362,3)
```

Veri setinden eksik veriler çıkarıldığında 362 satırlık ve 3 sütunluk bir veri çerçevesi elde edilir. Görüldüğü gibi veri setinden eksik veri çıkarıldığında doğal olarak veri seti eksilir. Bu durumu önlemek için eksik verinin giderilmesinde ikinci bir yol olarak eksik verinin yerine yeni bir değer ataması yapılabilir. Değer ataması yapılırken yeni değer sabit bir sayı olabileceği gibi ortalama değer veya bir yöntemle hesaplanmış değer de atanabilir. Pandas ile sayısal özelliklerde eksik veri yerine sabit değer atanması için **fillna** metodu kullanılır.

48.ÖRNEK

```
# Pandas ile eksik verinin yerine sabit değer atanması.
df_boy_kilo["Boy"].fillna(170)
# Aynı veri çerçevesi üzerinde değiştirilmek istenirse.
# df_boy_kilo["Boy"].fillna(170, inplace=True)
```

Çıktı

```
0 140.0
1 140.0
2 140.0
3 140.0
4 140.0
...
```

```
495 198.0
496 198.0
497 199.0
498 170.0
499 199.0
Name: Boy, Length: 500, dtype: float64
```

Pandas ile kategorik özelliklerde eksik veri yerine sabit değer atanması.

49. ÖRNEK

```
# Kategorik veride eksik varsa.
# Eksik cinsiyet bilgisi E olarak atanır.
df_boy_kilo["Cinsiyet"].fillna('E')
# Aynı veri seti üzerinde değiştirilmek istenirse.
# df_boy_kilo["Boy"].fillna(170, inplace=True)
```

Pandas ile sayısal özelliklerde eksik veri yerine sabit değer atanması.

50. ÖRNEK

```
# sayısal değerlere ortalama değer atanması.
df_boy_kilo.fillna(df_boy_kilo.mean())
# inplace = True parametresi eklenirse değişiklik veri setine
uygulanır.
```

Çıktı

	Cinsiyet	Boy	Agirlik
0	NaN	140.000000	129.000000
1	E	140.000000	152.000000
2	E	140.000000	79.000000
3	K	140.000000	105.975556
4	E	140.000000	52.000000
...	...	...	...
495	E	198.000000	136.000000
496	K	198.000000	50.000000
497	E	199.000000	156.000000
498	E	169.893333	99.000000
499	NaN	199.000000	92.000000

500 rows × 3 columns

Boy ve ağırlık değerlerinin tutulduğu sütunlarda eksik veri yerine ortalama değerler (mean boy = 169.893333 agirlik = 105.975556) atanmıştır.

mean metodu yerine **median** metodu kullanılırsa ortalama değer yerine ortanca değer atanır.

51. ÖRNEK

```
# sayısal değerlere ortanca değer ataması.
df_boy_kilo.fillna(df_boy_kilo.median())
# inplace = True parametresi eklenirse değişiklik veri setine
uygulanır.
```

Çıktı

Cinsiyet	Boy	Agirlik
NaN	140.0	129.0
E	140.0	152.0
E	140.0	79.0
K	140.0	106.0
E	140.0	52.0
...	...	...
E	198.0	136.0
K	198.0	50.0
E	199.0	156.0
E	170.5	99.0
NaN	199.0	92.0

500 rows × 3 columns

Boy ve ağırlık değerlerinin tutulduğu sütunlarda eksik veri yerine ortalama değerler (mean boy = 170.5 agirlik = 106) atanmıştır.

Kayıp veri işlemleri için **Scikit-learn** kütüphanesindeki **Imputer** sınıfı kullanılır. Imputer sınıfında **strategy**, **missing_values** ve **axis** olmak üzere üç önemli özellik bulunur. Scikit-learn kütüphanesi kullanılarak eksik veri değer atama stratejileri, sayısal ve kategorik özellikleri veri yapılarından dolayı değişiklikler gösterir. Bu yüzden eksik veri ile ilgili işlem yaparken sayısal ve kategorik özellikler ayrı olarak ele alınmalıdır.

strategy özelliği, eksik verinin yerine atanacak değerin belirlenmesinde kullanılacak yöntemi seçmek için kullanılır. Sayısal özellikler için ortalama (strategy=mean), ortanca (strategy=median), sabit değer atama (Örnek: **strategy** ='constant', fill_value=0) kullanılır. Kategorik özellikler için ise en çok tekrarlanan (strategy =most_frequent) değerin veya sabit değer (strategy='constant', fill_value='E') atanması kullanılır.

missing_values özelliği kayıp veriyi tanımlamak için kullanılır. Veri yoksa **np.nan** kullanılır. Eksik veriyi ifade etmek için farklı bir değer kullanıldıysa (örnek: yok, eksik) belirtilmelidir.

axis ise yapılacak işlemlerin satır veya sütunlar bazında işleme alınmasını sağlamak için kullanılır. Sütun için 1, satır için 0 değeri verilmelidir.

Eksik veri tamamlanırken Imputer sınıfının **fit, transform** ve **fit_transform** metotları kullanılır.

fit metodu belirtilen strateji doğrultusunda eksik veri için atanacak değerin sütunlardan hesaplanmasını sağlar.

transform ise hesaplanan değer atanarak eksik verinin dönüştürülmesini sağlar.

fit_transform metodu ise **fit** ve **transform** metodunun yaptığı işlemleri birlikte yapar.

52. ÖRNEK

```
# Scikit-learn kütüphanesinden Imputer modülü kullanılır.
# Kayıp veriye ortalama değeri atama.
from sklearn.impute import SimpleImputer
imp = SimpleImputer(missing_values=np.nan, strategy = 'mean')
# Sayısal kolonlar listeye atanır.
numerik_ozellikler=['Boy', 'Agirlik']
# sayısal sütunların ortalama değerini öğrenir ve stratejiye göre
öğrenir ve stratejiye göre değeri NAN değerlere uygular.
df_boy_kilo_imp1 = imp.fit_transform(df_boy_kilo[numerik_ozellikler])
df_boy_kilo_imp1
```

Çıktı

```
array([[140.        , 129.        ],
       [140.        , 152.        ],
       [140.        , 79.         ],
       [140.        , 105.97555556],
       [140.        , 52.         ],

       [198.        , 136.        ],
       [198.        , 50.         ],
       [199.        , 156.        ],
       [169.89333333, 99.         ],
       [199.        , 92.         ]])
```

Eksik veriye sabit değer atamak.

53. ÖRNEK

```
# Eksik veriye sabit bir değer atamak.
# boy sütununda NAN değerlere sabit değer 165 olarak belirlenmiştir.
imp = SimpleImputer(missing_values = np.nan, strategy = 'constant',
fill_value = 165)
df_boy_kilo.Boy = imp.fit_transform(df_boy_kilo['Boy'].values.
reshape(-1,1))[:,0]
df_boy_kilo.Boy.head()
```

Çıktı

```
0 140.0
1 140.0
2 140.0
3 140.0
4 140.0
Name: Boy, dtype: float64
```

Kategorik eksik veriyi tamamlamak.

54. ÖRNEK

```
# En çok tekrarlanan değer de atanabilir.
# imp = SimpleImputer(missing_values = np.nan, strategy = 'most_
frequent')
df_boy_kilo.Cinsiyet = imp.fit_transform(df_boy_kilo['Cinsiyet'].
values.
reshape(-1,1))[:,0]
df_boy_kilo.Cinsiyet.head()
```

Çıktı

```
0 E
1 E
2 E
3 K
4 E
Name: Cinsiyet, dtype: object
```

Eksik veri üzerinde işlemler yapılarak veri seti, makine öğrenmesinin sonraki aşamaları için hazırlanmalıdır. Veri setinin büyüklüğü, eksik veri sayısı, veri türü ve verinin dağılımı gibi özelliklere bakılarak eksik verinin giderilmesi için örneklerde gösterildiği gibi yöntemler belirlenir ve uygulanır. Bu yöntemler dışında eksik verinin tamamlanması için regresyon ve kümeleme yöntemleri de kullanılabilir. Örnekte eksik veri silinerek veri seti üzerine kaydedilmiştir.

55. ÖRNEK

```
import pandas as pd
# kategorik verinin dönüştürülmesi.
# Orijinal veri seti https://www.kaggle.com/'dan uyarlanmıştır.
# veri setinin yüklenmesi.
df_boy_kilo = pd.read_csv('data/500_kisi_boy_kilo_eksik_veri.
csv', sep = ';')
df_boy_kilo.head()
# Eksik verilerin çıkarılması.
df_boy_kilo.dropna(inplace = True)
print(df_boy_kilo.info())
print (df_boy_kilo)
```

Çıktı

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 362 entries, 1 to 497
Data columns (total 3 columns):
#        Column         Non-Null Count      Dtype
---      -------        ---------------     ------
0       Cinsiyet         362 non-null       object
1         Boy            362 non-null       float64
2       Agirlik          362 non-null       float64
```

```
dtypes: float64(2), object(1)
memory usage: 11.3+ KB
None
          Cinsiyet         Boy         Agirlik
1            E          140.0          52.0
2            E          140.0          79.0
4            E          140.0          52.0
5            E          140.0          143.0
6            K          140.0          76.0
...        ...            ...           ...
492          E          198.0          50.0
494          E          198.0          109.0
495          E          198.0          136.0
496          K          198.0          50.0
497          E          199.0          156.0
[362 rows x 3 columns]
```

Kategorik Verinin Dönüştürülmesi ve Dummy (Kukla) Değişkenlerin Çıkarılması

Makine öğrenmesi algoritmaları kategorik veriyi doğrudan kullanamaz. Makine öğrenmesi algoritmalarını kullanabilmek için kategorik verinin sayısal olarak kodlanması (dönüştürülmesi) gerekir. Bunun için iki yöntem kullanılmaktadır: Label encoding ve one hot encoding. Label encoding yönteminde kategorik verideki eşsiz her değer için bir tam sayı ataması yapılır.

Species	Label Encoding
setosa	0
versicolor	1
virginica	2
setosa	0

56. ÖRNEK

```
# kütüphanelerin içe aktarılması.
from sklearn.preprocessing import LabelEncoder
import pandas as pd
# iris veri setinin yüklenmesi.
iris = pd.read_csv('https://raw.githubusercontent.com/mwaskom/
seaborndata/master/iris.csv')
# indis numarası 0,50 ve 100 olan 3 satırın yazdırılması.
print(iris.loc[[0,50,100],:])
# labelencoder nesnesi etiketleri belirlemek için kullanılır.
label_encoder = LabelEncoder()
# species altında benzersiz değerlerin (kategori etiketlerinin)
bulunması.
```

```
print(iris['species'].unique())
# species kolonundaki etiketleri kodlama.
iris['species']= label_encoder.fit_transform(iris['species'])
# indis numarası 0,50 ve 100 olan 3 satırın yazdırılması.
print(iris.loc[[0,50,100],:])
print(iris['species'].unique())
```

```
    sepal_length sepal_width petal_length petal_width   species
0            5.1         3.5          1.4         0.2    setosa
50           7.0         3.2          4.7         1.4  versicolor
100          6.3         3.3          6.0         2.5   virginica
['setosa' 'versicolor' 'virginica']
    sepal_length sepal_width petal_length petal_width   species
0            5.1         3.5          1.4         0.2         0
50           7.0         3.2          4.7         1.4         1
100          6.3         3.3          6.0         2.5         2
[0 1 2]
```

One hot encoding yöntemi ise kategorik verideki eşsiz her değer bir sütuna dönüştürülerek ilgili sütuna 1 değerinin diğer sütunlara ise 0 değerinin atandığı kodlama çeşididir.

Species	setosa	versicolor	virginica
setosa	1	0	0
versicolor	0	1	0
virginica	1	0	1
setosa	1	0	0

57. ÖRNEK

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
import pandas as pd
# iris veri setinin yüklenmesi.
iris = pd.read_csv('https://raw.githubusercontent.com/mwaskom/
seaborndata/master/iris.csv')
# indis numarası 0,50 ve 100 olan 3 satırın yazdırılması.
print(iris.loc[[0,50,100],:])
# labelencoder nesnesi etiketleri belirlemek için kullanılır.
ohe_encoder = OneHotEncoder()
# species altında benzersiz değerlerin (kategori etiketlerinin)
bulunması.
print(iris['species'].unique())
# species kolonundaki etiketleri kodlama
iris_ohe = iris['species'].values.reshape(-1,1)
print("Yeniden boyutlandırıldı:", iris_ohe.shape)
```

```
# Belirle ve sayısal değerlere dönüştür.
y = ohe_encoder.fit_transform(iris_ohe)
# diziye çevrilir.
y = y.toarray()
sonuc_y = pd.DataFrame(data = y, index = range(150),
columns = ['setosa', 'versicolor', 'virginica'])
print(sonuc_y)
# veri setine ekleme.
veriseti_fnl = pd.concat([iris.iloc[:,0:3], sonuc_y], axis = 1)
print(veriseti_fnl.loc[[0,50,100],:])
```

```
    sepal_length sepal_width petal_length petal_width   species
0            5.1         3.5          1.4         0.2      setosa
50           7.0         3.2          4.7         1.4  versicolor
100          6.3         3.3          6.0         2.5   virginica
['setosa' 'versicolor' 'virginica']
(150, 1)
       setosa  versicolor  virginica
0         1.0         0.0        0.0
1         1.0         0.0        0.0
2         1.0         0.0        0.0
3         1.0         0.0        0.0
4         1.0         0.0        0.0
..        ...         ...        ...
145       0.0         0.0        1.0
146       0.0         0.0        1.0
147       0.0         0.0        1.0
148       0.0         0.0        1.0
149       0.0         0.0        1.0
[150 rows x 3 columns]
     sepal_   sepal_   petal_
     length    width   length   setosa  versicolor  virginica
0       5.1      3.5      1.4     1.0         0.0        0.0
50      7.0      3.2      4.7     0.0         1.0        0.0
100     6.3      3.3      6.0     0.0         0.0        1.0
```

One hot encoding dönüşümünde tür kategorisi 3 sütuna dönüştürülmüştür. Tür için üç sütunun ikisinin değerinin 0 olması diğerinin 1 olması anlamına gelir. Bu durum kukla değişken olarak adlandırılan duruma yol açar. Kukla değişken, makine öğrenmesi modellerinde sonuçları olumsuz etkileyebileceği için veri setinden çıkarılmalıdır.

Örnekte; kukla değişkeni engellemek için final veri setinde setosa, versicolor ve virginica kolonlarından birinin çıkarılması gerekir.

Veri Setinin Bölünmesi

Denetimli öğrenme modellerinde (sınıflandırma ve regresyon) veri seti, eğitim ve test seti olarak iki parçaya ayrılır. Eğitim seti modeli eğitmek (parametreleri ayarlamak) için kullanılırken test seti ise modelin performansını test etmek için kullanılır. Kaggle gibi platformlarda veri seti eğitim ve test olarak ayrı dosyalar hâlinde verilebilir. Denetimsiz öğrenme modellerinde (kümeleme) ise veri seti bölünmez. Modelin eğitimi için veri setini bölerken eğitim ve test için verinin homojen olarak dağılmasını sağlamak gerekir. Sınıf etiketleri oranları eğitim ve test verisi için benzer olmalıdır. Sınıf etiketleri eşit dağılmazsa model bir sınıfa ait özellikleri diğer sınıflara göre daha iyi öğrenir. Model bu şekilde eğitilirse test verisinde farklı sınıflara ait tahminlerde başarı düşecektir. Denetimli modellerde veri seti eğitim ve test olarak ayrılmasına ek olarak veri setindeki sütunlar özellikler, bağımsız değişkenler (X) ve bağımlı değişken, etiket, sonuç (y) olarak ayrılır. Denetimli öğrenme modellerinde veri seti dört parça olarak ele alınır.

58. ÖRNEK

```
#kütüphanelerin içe aktarılması.
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
#Uyarıların kapatılması.
import warnings
warnings.filterwarnings("ignore")
#iris veri setinin yüklenmesi.
iris = pd.read_csv('https://raw.githubusercontent.
com/mwaskom/seaborndata/master/iris.csv')
#Veri setinin bölünmesi.
X_train, X_test, y_train, y_test = train_test_split(
iris.loc[:, ['petal_length', 'petal_width', 'sepal_width',
'sepal_length' ]],
iris.loc[:, 'species'], test_size = 0.33, random_state = 0)
```

X_train: Eğitim setindeki özelliklerdir. İris veri setinde çanak ve taç yaprağa ait uzunluk ve genişlik değerlerinin bulunduğu sütunlardır.

X_test: Test setindeki özelliklerdir. **X_train** ile aynı sütunlar bulunur. Model eğitildikten sonra modeli test için kullanılan test veri setindeki özelliklerdir.

y_train: Modeli eğitmek için kullanılan sınıf etiketlerinin bulunduğu sütundur. **X_train** özelliklerine karşılık gelen tür sınıf etiketleridir. Örnek olarak **setosa, versicolor ve virginica** verilebilir.

y_test: Modeli test etmek için kullanılan sınıf etiketlerinin bulunduğu sütundur. **y_test** özelliklerine karşılık gelen sınıf etiketleridir. Modelin **X_test** veri seti üzerindeki tahminlerini karşılaştırmak için kullanılır.

test_size: Veri setinin ne kadarının test olarak ayrılacağını belirtmek için kullanılan parametredir. Varsayılan değeri 0,25 tir (%25). Genellikle 0,20-0,33 aralığında bir değer kullanılır.

random_state: Veri setinden rastgele olarak veri kümesi seçilmesini sağlar. Bir sayı verildiğinde o sayı için hep aynı rastgele veri kümesi oluşturulmasını sağlar. Bu sayede modelin uygulamasında ve performans metriklerinde hep aynı sonuçlar alınır.

Veriyi Standartlaştırma, Normalizasyon ve Ölçeklendirme

Veri setindeki özelliklere ait sayısal değerler birbirinden çok farklı bir skalada olabilir. Örnek olarak bir veri setinde bir sütunda çalışma süresi yıl olarak verilmektedir. Ücret değeri ise dört haneli sayılar olarak verilmektedir. Bir veri seti içinde aykırı (aşırı uç) değerler de olabilir. Bir makine öğrenmesi modeli eğitilirken

bu değerler dönüştürülmeden kullanılırsa iyi bir model geliştirilmesi güçleşir. Bazı makine öğrenmesi algoritmaları bu tür değerlerden çok olumsuz etkilenir. Modelin sayısal olarak büyük değerlerden daha fazla etkilenme riski bulunmaktadır. Bu tür olası sorunları engellemek için değerleri belirli bir aralığa ölçeklendirme işlemleri yapılmalıdır.

Normalleştirme : Değerlerin 0-1 arasındaki değerlere eşleşmesidir.

Standartlaştırma : Değerlerin ortalama değeri 0, standart sapması 1 olacak şekilde normal dağılıma eşleştirilmesidir.

59. ÖRNEK

```
#Standartlaştırma için kütüphane içe aktarılıyor.
from sklearn.preprocessing import StandardScaler
#Yeni bir nesne oluşturuluyor.
sc = StandardScaler()
#Eğitim setindeki özelliklerin dönüştürülmesi.
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

İris veri seti için özelliklerin standartlaştırılması eğitim performans değerini artırırken test performans değerini düşürür.

Normalleştirme için **sklearn.preprocessing** modülünden **normalize** nesnesi kullanılır.

60. ÖRNEK

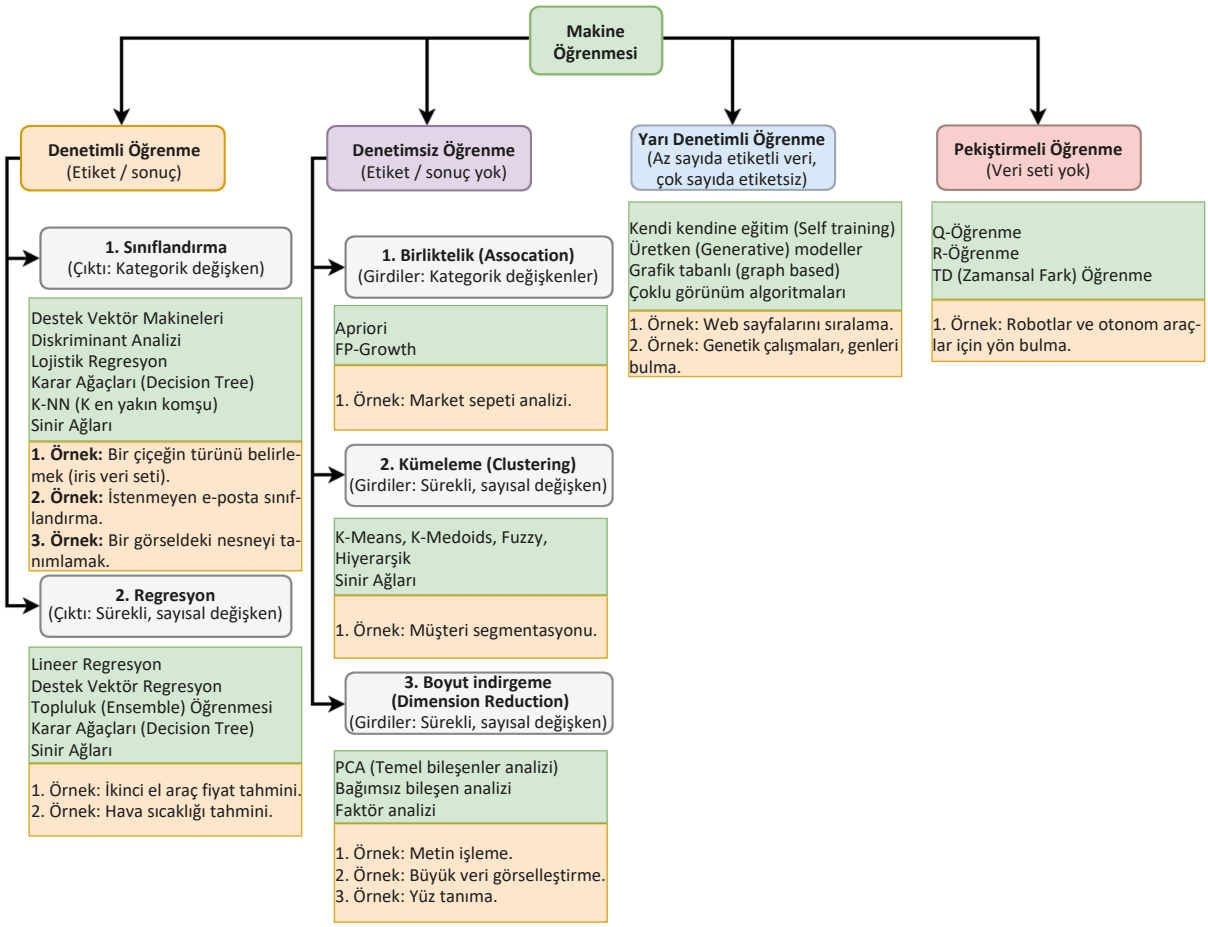
```
#kütüphaneler içe aktarılıyor.
from sklearn.preprocessing import normalize
#normalleştirme işleminin yapılması.
X_train = normalize(X_train)
X_test = normalize(X_test)
```

SIRA SİZDE

Boston ev fiyatları veri seti üzerinde Scikit-learn kütüphanesiyle veri ön işleme işlemlerini yapınız.

2.3. MAKİNE ÖĞRENMESİ İÇİN KULLANILAN ALGORİTMALAR

Makine öğrenmesi algoritmaları denetimli, denetimsiz, yarı denetimli ve pekiştirmeli olmak üzere dört sınıfa ayrılır. Makine öğrenmesi süreci bir probleme çözüm bulmak amacıyla yürütülür. Üzerinde çalışılan problem durumlarına ve eldeki veri setine göre makine öğrenmesi modeli ve algoritma seçilmelidir. Bu amaçla makine öğrenmesi türüne göre model görselinden yararlanılır (Görsel 2.35).



Görsel 2.35: Makine öğrenmesi türüne göre model seçimi

2.3.1. Denetimli (Gözetimli) Öğrenme Algoritmaları

Denetimli öğrenme modelleri sınıflandırma ve regresyon problemlerinin çözümünde kullanılır. Sınıflandırma bir örneğin ait olduğu sınıfı, kategoriyi belirlemek için kullanılır. Veri setindeki etiket, iki kategoriden oluşuyorsa **ikili sınıflandırma** (binary classification), ikiden fazla kategoriden oluşuyorsa **çok sınıflı sınıflandırma** (multi class clasification) olarak adlandırılır. Sınıf adlarının her birine **etiket** denir. Sınıflandırma için kullanılacak veri setlerinde sınıf etiketlerinin bulunduğu bir sütun olmalıdır. Regresyonda ise bu sütun sürekli sayısal bir değer olmalıdır. Denetimli öğrenme algoritmalarının seçiminde örnekteki gibi bir yol izlenebilir. İris veri setindeki tür sütunu çiçeğin türüne (sınıfına) ait kategorik bir veri içerir. Setosa, versicolor ve virginica çiçeğin türünü (sınıfını) belirten etiketlerdir. Veri setinde tür sütununda üç kategori bulunduğu için çok sınıflı sınıflandırmaya örnek olarak verilebilir. Çiçeğe ait özellikler (sütunlar) çanak yaprak uzunluğu, çanak yaprak genişliği, taç yaprak uzunluğu ve taç yaprak genişliğidir. Bu özelliklere **bağımsız değişken** adı verilir. Çiçeğin türü ise bağımlı değişkendir.

2.3.2. Denetimli (Gözetimli) Öğrenme Modeli Seçimi

Veri seti eğitim ve test olmak üzere iki parçaya bölünür. Eğitim seti, özellikler (çanak yaprak uzunluğu, çanak yaprak genişliği, taç yaprak uzunluğu ve taç yaprak genişliği) ve sınıf etiketi (Setosa, versicolor ve virginica) olarak sütunlara ayrılır. Makine öğrenmesinde bir problem durumu için birden fazla model kullanılabilir.

Makine öğrenmesi türüne göre model seçimi görselinden yararlanılarak denetimli öğrenme modellerinden duruma uygun sınıflandırma veya regresyon modellerden biri seçilir (Görsel 2.35). Bu aşamada birden fazla algoritma eğitilerek test edilebilir.

Örnek: Sınıflandırma için K-NN, Regresyon için Lineer Regresyon. Seçilen model veri setindeki özellikler ve sınıf etiketleri / sonuç ile eğitilir. Model, özellikler ile sınıf / sonuç etiketleri arasındaki ilişkiyi öğrenir. Bu aşamada her algoritma farklı çalışır. Modelin eğitiminde tüm eğitim setinin hem eğitim hem de doğrulama amaçlı kullanılmasıyla (K katmanlı çapraz doğrulama tekniği) geçerliliği daha yüksek bir model elde edilebilir. Model içinde kullanıcı tarafından belirlenmesi gereken parametrelere **hiper parametre** denir.

Örnek: K-NN sınıflandırmasında K değeri gibi. Bu aşamada ızgara araması tekniği kullanılır. Bu teknikle yüksek doğrulama sonuçları elde etmek için ideal hiper parametre değerleri belirlenir.

61. ÖRNEK

```
#Sınıflandırıcı kütüphanesinin yüklenmesi.
#Bir sınıflandırıcı tanımlanıyor.
knn = KNeighborsClassifier(n_neighbors = 2, p = 2, metric = 'minkowski')
#Modelin eğitilmesi.
knn.fit(X_train, y_train)
```

Çıktı

```
KNeighborsClassifier(algorithm = 'auto', leaf_size = 30,
metric = 'minkowski', metric_params = None, n_jobs = None,
n_neighbors = 2, p = 2,weights = 'uniform')
```

Örnekte eğitim seti 10 eşit parçaya (fold) bölünmektedir. Her seferinde 9 parça, modeli eğitmek için kullanılırken 1 parça doğruluğu ölçmek için kullanılır. Bu işlem 10 defa tekrarlanarak doğruluk ölçüsü hesaplanır. Hesaplanan doğruluk ölçülerinin ortalaması modelin ortalama doğruluk ölçüsünü verir.

62. ÖRNEK

```
#Çapraz doğrulama nesnesi içe aktarılıyor.
from sklearn.model_selection import cross_val_score
#Eğitim setinin kaça bölüneceğini belirleme.
fold_sayisi = 10
#On kere doğrulama işlemi yapılarak sonuçlar bir diziye atanır.
dogruluklar = cross_val_score(estimator = knn, X = X_train,
y = y_train, cv = fold_sayisi)
#np.mean ile ortalama doğruluk hesaplanır.
print("Ortalama Doğruluk: ", np.mean(dogruluklar))
print("Doğrulukların Standart Sapması: ", np.std(dogruluklar))
```

Çıktı

```
Ortalama Doğruluk:                0.82
Doğrulukların Standart Sapması:   0.09797958971132714
```

2.3.3. Denetimsiz (Gözetimsiz) Öğrenme Algoritmaları

Denetimsiz öğrenmede bir sınıf / sonuç sütunu bulunmaz. Veri setinde bir hedef değişken bulunmaz. Denetimsiz öğrenme modelleri birliktelik, kümeleme ve boyut indirgeme problemlerinin çözümünde kullanılır.

Birliktelik: Özellikle **market sepeti analizi** adı verilen uygulamalarda hangi ürünlerin birlikte satıldığını belirlemek için kullanılır.

Örnek: E-ticaret sitelerindeki öneri sistemleri bu mantıkla çalışır.

Kümeleme: Denetimsiz öğrenme, etiketsiz örneklerin benzer özelliklerine göre gruplandırılmasıdır.

Örnek: Bankaların, kredi kartı kullanıcılarını gruplara ayırması.

Boyut indirgeme: Çok sayıda özelliği (boyut) temsil edecek daha az sayıda özellik elde etmek için kullanılır. Çok sayıda özellik veri işlemede ve model oluşturmada zaman ve kaynak kullanımını artırır. Veri setindeki özellikler arasında yüksek korelasyon olması model oluşturulmasını olumsuz etkiler. Bu tür nedenler, en az bilgi kaybıyla en az sayıda özelliği seçmeyi (boyut indirgeme) önemli hâle getirir.

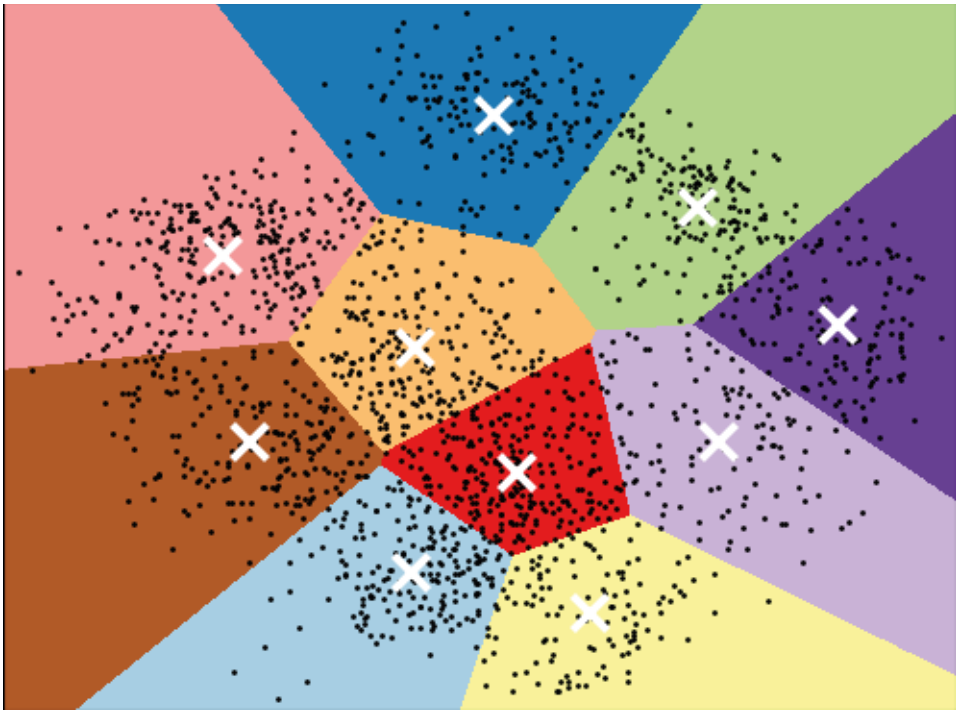
Örnek: Yüz tanıma, sinyal işleme ve konuşma tanıma.

2.3.4. Denetimsiz (Gözetimsiz) Öğrenme Modeli Seçimi

Veri seti eğitim ve test olarak bölünmez ve tek parça olarak kullanılır. Denetimsiz öğrenmede de bir problem durumu için birden fazla model kullanılabilir. Bu aşamada model seçimi görselinden yararlanılarak denetimsiz öğrenme modellerinden duruma uygun olarak birliktelik, kümeleme ve boyut indirgeme modellerinden biri seçilir (Görsel 2.35).

2.3.5. K-Means Kümeleme

K-Means, kümeleme için kullanılan denetimsiz bir öğrenme modelidir. K hiper parametresi veri setinin kaç kümeye ayrılacağını belirler. Benzer ögeleri aynı kümelere yerleştirirken birbirine benzemeyen ögeleri farklı kümelere ayırır. K-Means kümeleme algoritması aşağıdaki gibi çalışır. K sayıda rastgele küme merkezi belirlenir. Örnekler küme merkezlerine olan mesafelerine göre kümelendirilir. Her yeni örnekte küme merkezi yeniden hesaplanır. Küme merkezleri değişmeyene kadar 2. ve 3. adım tekrarlanır. Görsel 2.36'da örnek bir K-Means kümeleme sonucu gösterilmektedir. Nihai küme merkezleri çarpı ile işaretlenmiştir. Örnekte K=10 olarak belirlenmiştir.



Görsel 2.36: K-Means kümeleme

Örnek: Kümeleme için duruma göre K-Means ve hiyerarşik algoritmalarından biri kullanılabilir. Seçilen model, veri seti ile eğitilir. Model, kullanılan algoritmaya göre özelliklerle ilgili benzerlikleri ve farklılıkları

belirler. Bu aşamada sınıflandırma algoritmaları farklı çalışır. Model geliştirilirken tüm veri seti kullanılır. Model içinde kullanıcı tarafından belirlenmesi gereken parametrelere **hiper parametre** denir.

Örnek : K-Means kümelemedeki K değeri gibi. Bu aşamada ızgara araması tekniği kullanılır. Bu teknikle yüksek doğrulama sonuçları elde etmek için ideal hiper parametre değerleri belirlenir.

63. ÖRNEK

```
Bir kümeleme modelinin tanımlanması.
#kümeleme sınıfının içe aktarılması.
from sklearn.cluster import KMeans
#n_cluster hiper parametresi küme sayısını belirlemek için kulla-
nılır.
k_ortalama = KMeans(n_clusters = 3)
#Modelin veri kullanılarak eğitilmesi.
kumeler = k_ortalama.fit_predict(data)
```

2.3.6. Denetimli (Gözetimli) Öğrenme Modellerinde Değerlendirme

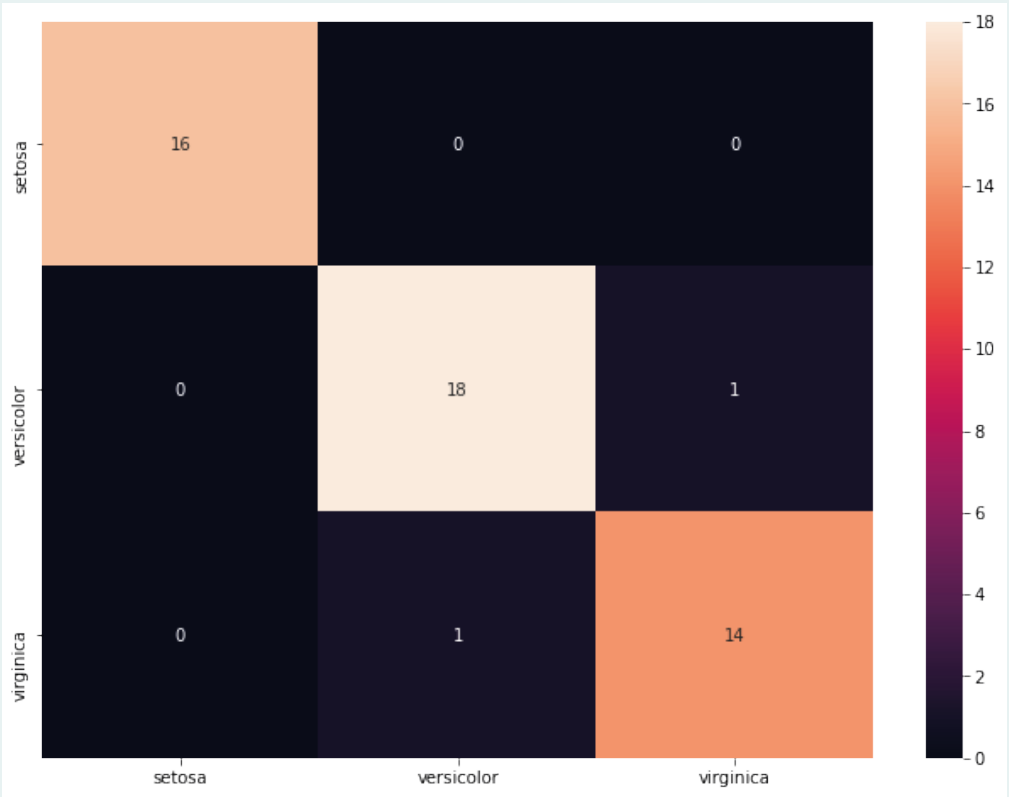
Test setindeki özellikler (sınıf etiketleri verilmeden) modelin sınıfları tahmin etmesi için kullanılır. Tahmin sınıfları ile gerçek sınıflar karşılaştırılır ve sınıflandırma metrikleri kullanılarak modelin başarısı ölçülebilir. Algoritmalar örneklerin hangi sınıfa ait olduğunu belirlemek için iki yöntem kullanır: Eşikleme ve yarışmalı sınıflandırma. Eşiklemede sınıf üyeliği için belirli bir eşik değeri vardır. İkili sınıflandırmada 0 ve 1 iki sınıf değeri olarak belirlendiğinde eşik değeri 0,5 olarak belirlenir. Örnek için tahmin edilen değer, eşik değerin altında ise 0 etiketli sınıfa ait, eşik değerin üstünde ise 1 etiketli sınıfa ait olarak tahmin edilir. Yarışmalı sınıflandırmada ise örneğin her bir sınıf için bir ait olma olasılığı hesaplanır. Bu değerlerden büyük olan sınıfa ait olduğu tahmin edilir.

64. ÖRNEK

```
import matplotlib.pyplot as plt
from sklearn import metrics
#Tahminler bir NumPy dizisi olarak atanıyor.
#knn.predict verilen test değerleri için sınıfları tahmin ediyor.
y_pred = knn.predict(X_test)
# hata matrisinin oluşturulması.
cm = metrics.confusion_matrix(y_test, y_pred)
# Veri çerçevesine dönüştürme.
cm_df = pd.DataFrame(cm,
index = ['setosa', 'versicolor', 'virginica'],
columns = ['setosa', 'versicolor', 'virginica'])
#11x8 büyüklüğünde bir grafik çizme.
plt.figure(figsize = (11,8))
sns.heatmap(cm_df, annot = True)
```

Çıktı (Görsel 2.37)

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8037af6ad0>
```



Görsel 2.37: Sınıflandırma matrisi grafiği

65. ÖRNEK

```
#Performansın değerlendirilmesi.
print ("Eğitim doğruluğu (Accuarcy %) {}".format(100*knn.score(
X_train, y_train)))
print ("Test doğruluğu (Accuarcy %) {}".format(100*knn.score(
X_test, y_test)))
```

Çıktı

```
Eğitim doğruluğu (Accuarcy %) 99.0
Test doğruluğu (Accuarcy %) 98.0
```

2.3.7. Denetimsiz (Gözetimsiz) Öğrenme Modellerinde Değerlendirme

Denetimsiz öğrenmeye örnek olarak kümeleme algoritması örneği ele alınmıştır. Birbirinden farklı örnekleri iyi ayırma derecesi ve benzer örnekleri gruplama derecesi performansı belirler. Kümeleme algoritmaları için en popüler ölçütler: Silhouette katsayısı, Davies-Bouldin Endeksi ve Dunn dizinidir.

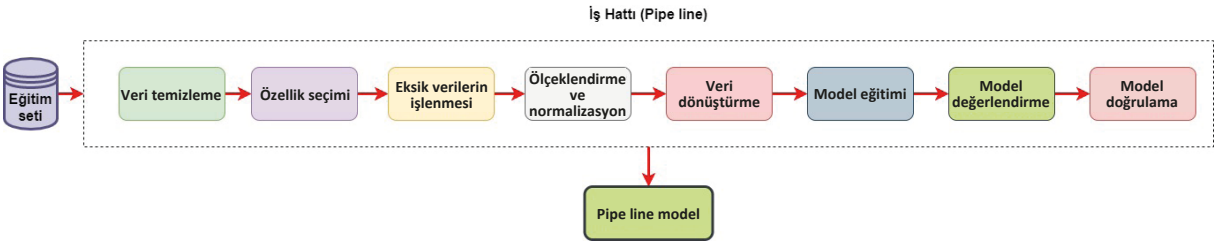
Dağıtım (Deployment)

Dağıtım, bir makine öğrenmesi modelinin yeni veri setlerinde ve farklı ortamlarda kullanılabilmesi anlamına gelir. Python ile geliştirilen bir makine öğrenmesi modeli Flask ve Heroku kullanılarak internete taşınabilir.

İş Hattı (Pipeline)

İş hattı, makine öğrenmesi sürecini otomatikleştiren bir yaklaşımdır. Scikit-learn kütüphanesinde pipeline modülü kullanılarak makine öğrenmesi sürecindeki aşamalar entegre biçimde tanımlanabilir.

Bir iş hattı Görsel 2.38'de gösterildiği gibidir.



Görsel 2.38: İş hattı süreci

Bir iş hattı süreci Python'da örnekteki gibi kodlanabilir. İş hattı süreci **pipeline** sınıfı kullanılarak tanımlanır.

1. UYGULAMA

Pipeline uygulamasını gerçekleştirmek için aşağıdaki işlem adımlarını takip ediniz

1. Adım: SVM modelini ve ölçeklendirme modellerini içe aktarınız.

```
#Model içe aktarılır.
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
```

2. Adım: Veri setini içe aktarınız. Veri setini eğitim ve test olarak bölmek amacıyla modülü içe aktarınız.

```
#Veri setleri modülü içe aktarılır.
from sklearn.datasets import make_classification
#Veri setini bölmek için kullanılan modül.
from sklearn.model_selection import train_test_split
```

3. Adım: İş hattı modülünü içe aktarınız. Veri setindeki özellikleri ve etiketleri ayırınız. Veri seti özelliklerini eğitim, hedef eğitim, özellikler test ve hedef test olarak ayırınız.

```
#Pipeline (iş hattı) içe aktarılır.
from sklearn.pipeline import Pipeline
#Özellikler (X) ve sınıf etiketleri (y) atanıyor.
X, y = make_classification(random_state = 0)
# Eğitim ve test verisi için özellikler ve etiketlerden veri seti
4 parçaya ayrılıyor.
X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state = 0)
```

4. Adım: Yeni bir iş hattı tanımlayınız. İş hattında ölçeklendirme için standart ölçeklendirmeyi seçiniz. Model olarak SVM altında Destek Vektör Sınıflandırmasını seçiniz.

```
pipe = Pipeline([('scaler', StandardScaler()), ('svc', SVC())])
# Başka bir tahminleyici (estiamtor) algoritma da seçilebilirdi.
# Model eğitilir.
pipe.fit(X_train, y_train)
#İş hattı adımları tanımlanır.
Pipeline(steps = [('scaler', StandardScaler()), ('svc', SVC())])
```

5. Adım: Modelin performans skoruna bakarak modeli değerlendiriniz.

```
pipe.score(X_test, y_test)
```

Çıktı

```
0.88
```

Sık kullanılan makine öğrenmesi algoritmaları K-NN, Karar Ağacı, Destek Vektör Makineleri (SVM), K-Means, Temel Bileşenler Analizi (PCA) ve Apriori şeklindedir. Regresyon analizi ve yöntemleri ayrı bir konu olarak ele alınmıştır.

SIRA SİZDE

Örnek veri seti üzerinde Scikit-learn kütüphanesiyle modelleme ve iş hattı tanımlama işlemlerini yapınız. Önceki çalışmalarınızdan yararlanınız.

2.4. REGRESYON ANALİZİ VE TÜRLERİ

Regresyon analizi özellik (bağımsız değişken) veya özelliklerle bir bağımlı değişken arasındaki ilişkiyi belirlemek için kullanılır. Regresyon için kıdem-ücret, çalışma süresi-alınan not ve kan değerleri-hastalık durumu gibi örnekler verilebilir. Regresyon analizi gözetimli öğrenme yöntemlerindendir. Doğrusal ve doğrusal olmayan regresyon modelleri bulunmaktadır.

2.4.1. Doğrusal ve Doğrusal Olmayan Regresyon

Doğrusal regresyon düz bir çizgiyle ifade edilebilen bir model türüdür. Basit doğrusal regresyon ve çoklu doğrusal regresyon olmak üzere iki tür doğrusal regresyon vardır. Doğrusal olmayan regresyon düz bir çizgi ile ifade edilemeyen modeller için kullanılır. Polinominal regresyon ve lojistik regresyon, doğrusal olmayan regresyon türleridir.

2.4.2. Basit Doğrusal Regresyon

Basit doğrusal regresyon, iki değişken arasındaki ilişkiyi tanımlamak için kullanılan doğrusal bir modelin tanımıdır. Basit doğrusal regresyonda değişkenler de bağımlı değişken ve bağımsız değişkendir. Doğrusal regresyonun amacı gerçek değerlere en yakın doğruyu çizmektir. Doğrunun gerçek değerlere uzaklıkları hesaplanır. Ortalama uzaklık değerleri (hata) en düşük doğrunun formülü bulunur. Doğrusal regresyon aşağıdaki gibi formüle edilebilir.

$$y=\beta_0+\beta_1 X+\epsilon$$

Burada;

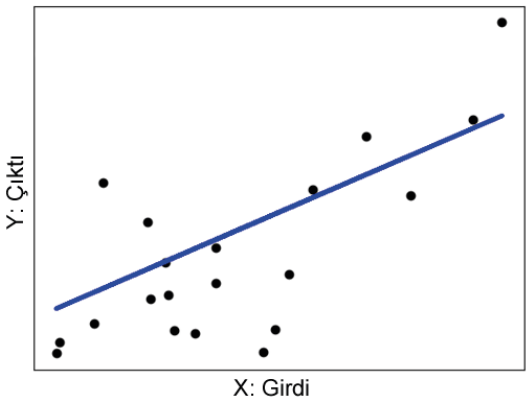
β_0 : Doğrunun y eksenini kestiği nokta ve regresyon sabitidir.

β_1 : Doğrunun eğimi veya katsayı

X : Bağımsız değişken değeri

ϵ : Hata değeri

Örnekte basit doğrusal regresyon doğrusu gösterilmektedir (Görsel 2.39).



Görsel 2.39: Basit doğrusal regresyon

NOT !!! Doğrusal regresyon ile çizilen doğru, gerçek değerlere en yakın şekilde geçmelidir.

2.4.3. Çoklu Doğrusal Regresyon

Birden fazla bağımsız değişkenin ve bir bağımlı değişkenin bulunduğu regresyon modeli çoklu doğrusal regresyon modelidir. Çoklu doğrusal regresyon modeli aşağıdaki gibi formüle edilebilir. Mesai süresi ve kıdem seviyesinin ücrete etkisi çoklu doğrusal regresyona örnek verilebilir. Çoklu doğrusal regresyon formülü aşağıdaki gibidir.

$$y= \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n + \epsilon$$

Burada;

β_0 : Regresyon sabiti

$\beta_1, \beta_2, \ldots, \beta_n$: Bağımsız değişkenlerin katsayıları

$X_1, X_2, \ldots, X_n$: Bağımsız değişkenlerin değerleri

ϵ : Hata değeri

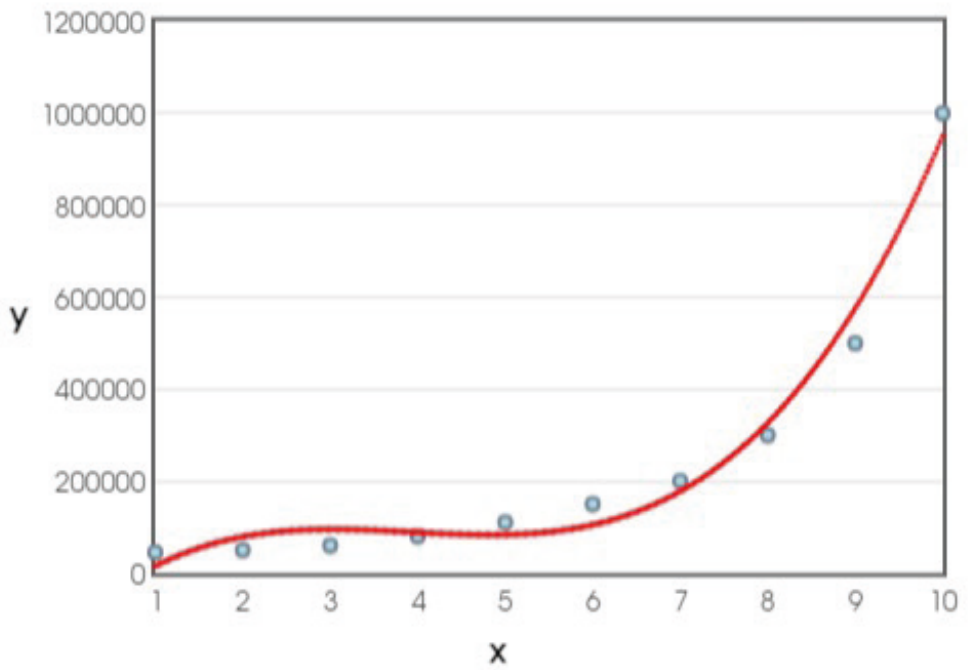
2.4.4. Polinominal Regresyon (Polynominal Regression)

Değişkenler arasındaki ilişkinin doğrusal olmadığı durumlarda Polinomsal Regresyon analizi yapılmaktadır. Polinomsal regresyon modellerinde denklemdeki parametre sayısı ile veri kümesindeki değişken sayısı doğrudan ilişkili olmayabilir. Doğrusal regresyondan farklı olarak bir doğru yerine bir polinom grafiği çizilmektedir. Polinom derecesi oluşturulan modelin uyumunu etkiler.

$$y= \beta_0 + \beta_1 X + \beta_2 X^2 + \ldots + b_n X^h + \epsilon$$

Burada h polinom derecesini göstermektedir. **İkinci derece polinoma kuadratik, üçüncü derece polinoma kübik ve dördüncü polinoma kuartik denir.**

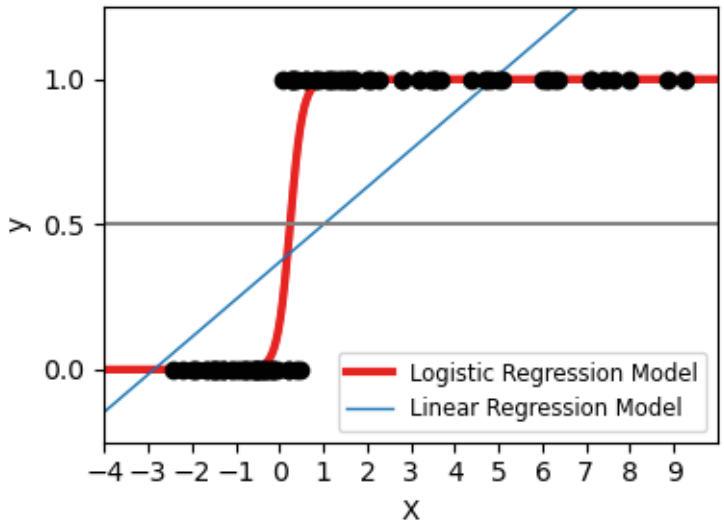
Örnekte pozisyon ile maaş arasındaki ilişki 4. dereceden bir polinom denklemi ile görselleştirilmiştir (Görsel 2.40).



Görsel 2.40: Polinominal regresyon

2.4.5. Lojistik Regresyon (Lojistic Regression)

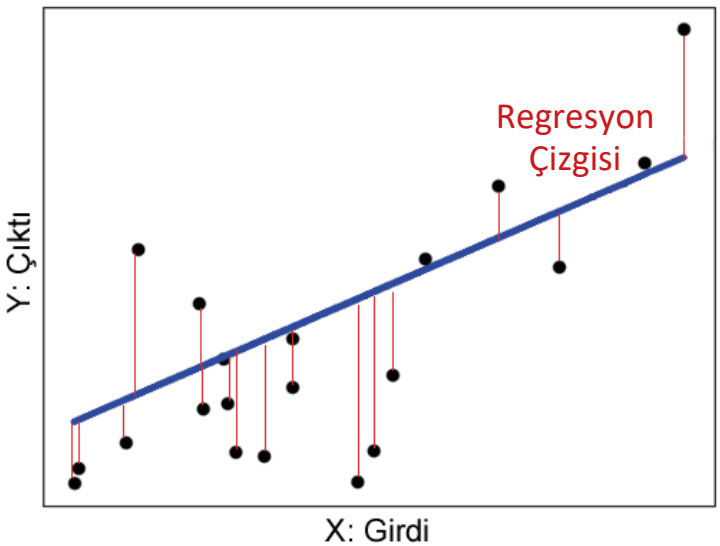
Sınıflandırma için kullanılan denetimli bir makine öğrenmesi algoritmasıdır. Bir veya daha fazla bağımsız değişken ile bağımlı değişken arasındaki ilişkiyi modellemek için kullanılır. Lojistik regresyon ikili sınıflandırma için bir regresyon fonksiyonu oluşturur. Bu fonksiyon ikili bir sınıf için 0 ve 1 çıktıları üretir. Lojistik regresyona cinsiyet tahmini (erkek, kadın) ve hastalık durumu (pozitif, negatif) örnek verilebilir. Örnekte lojistik regresyon ve lineer regresyon arasındaki fark grafiksel olarak gösterilmektedir (Görsel 2.41).



Görsel 2.41: Lojistik regresyon

2.4.6. Tahmin Modellerinin Değerlendirilmesi En Küçük Kareler Yöntemi

Regresyon yöntemlerin başarısı ortalama kare hata (MSE) ölçütü ile karşılaştırılır. En küçük kareler yöntemi ortalama kare hatanın (MSE) minimize edilmesine dayalıdır. En küçük kareler yönteminde hatayı minimize edecek regresyon çizgisini elde etmek amaçlanmaktadır. Örnekte gerçek değerlerin regresyon çizgisine uzaklıkları gösterilmiştir. Her bir regresyon modeli değerlendirilirken gerçek verilerin regresyon çizgisine olan uzaklıklar toplanarak bir değer elde edilir. Toplam uzaklıkları minimize edecek bir doğruyu elde etmek için regresyon formülünde sabit ve katsayı değerleri değiştirilir. En küçük kareler yöntemiyle en uyumlu model bulunur (Görsel 2.42).



Görsel 2.42: Lojistik regresyon

2.4.7. Eşikleme ve Yarışmalı Sınıflandırma

Regresyon gibi tahminlerde kullanılan modellerin sınıflandırma yapabilmesi için alınan sonuçlarda eşikleme yapılır. Etiketler 0 ve 1 ise eşikleme 0,5 olur. Sınıflamada 0,5 ve üstü 1 etiketi olarak tahmin edilirken 0,5 altı ise 0 etiketi olarak tahmin edilir. Veri setinde ikiden çok etiket varsa sonuç yarışmalı sınıflandırma yapılarak belirlenir. Yarışmalı sınıflandırmada sınıf olasılık değeri büyük olan sınıf yarışta diğerlerini geçerek tahmin sınıfı olur.

Korelasyon ve Regresyon İlişkisi

Regresyon değişkenler arasındaki neden-sonuç ilişkisini araştırırken korelasyon iki değişken arasında ilişkinin yönünü ve şiddetini ölçer. Yani regresyonda iki değişken biri diğerine bağımlıyken korelasyonda iki değişkenin varlığı birbirinden bağımsızdır. Örneğin "yemek yeme" ile "kilo alma" arasındaki ilişki regresyona örnek verilebilecekken "günlük uyku süresi" ile "TV izleme süresi" arasındaki ilişki korelasyona örnek verilebilir.

2.4.8. R^2 ile Regresyon Modellerinin Değerlendirilmesi

Doğrusal modellerde en çok kullanılan metriklerden biridir. Aşağıdaki formülle hesaplanır. Bir bağımsız değişkenle bir bağımlı değişkenin tahmin edildiği modellerde kullanılır. Genel olarak R^2 en yüksek 1 değerini alabilir. Genel olarak R^2 değeri ne kadar büyük olursa model o kadar başarılıdır.

Düzeltilmiş R^2 : Modelde bağımsız değişken sayısı birden fazlaysa R^2 değeri yanıltıcı olabilir. Bu tür modellerde Düzeltilmiş R^2 değeri kullanılır.

n = 10, p = 3 için modelin düzeltilmiş R^2 değeri aşağıdaki formülle hesaplanır.

$$R^2_{Ad} = 1 - (1 - R^2)\frac{n-1}{n-p-1}$$

Değerler yerine konduğunda R^2_{Ad} değeri 0,925 olarak hesaplanmaktadır.

$$R^2_{Ad} = 1 - (1 - 0{,}95)\frac{10-1}{10-3-1} = 0{,}925$$

2.4.9. Regresyon Uygulamaları

Bu bölümde Python'da yapılmış Regresyon uygulamalarına yer verilmiştir.

2. UYGULAMA

Basit doğrusal regresyon uygulaması yapmak için aşağıdaki işlem adımlarını takip ediniz.

Uygulama veri setinde 10 adet özellik ve bir adet hedef değişkeni (diyabet değeri) yer almaktadır. Bu uygulamada diyabete etkisi özelliklerden biri seçilerek bir regresyon modeli oluşturulmaktadır. Oluşturulan regresyon modeli verilen değere göre bir diyabet değeri tahmini yapmaktadır.

1. Adım: Regresyon modeli ve değerlendirme metrikleri ile ilgili kütüphaneleri içe aktarınız.

```
# Temel kütüphaneler içe aktarılır.
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
# Scikit-learn kütüphanesinden data setler ve lineer model içe
aktarılır.
from sklearn import datasets, linear_model
# Model değerlendirme metrikleri içe aktarılır.
from sklearn.metrics import mean_squared_error, r2_score
```

2. Adım: Veri setini özellikler ve hedef sütunu olarak ayırınız. Veri seti hakkında bilgi alınız.

```
#Veri setindeki özellikler ve hedef sütun ayrılır.
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y = True)
# Veri seti hakkında bilgi alınır.
pd.DataFrame (data = diabetes_X).info()
# Yalnız bir özellik seçiliyor.
diabetes_X = diabetes_X[:, np.newaxis, 2]
```

Çıktı

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 442 entries, 0 to 441
Data columns (total 10 columns):
 #     Column     Non-Null Count      Dtype
---    -------    ---------------     ------
 0       0         442 non-null      float64
 1       1         442 non-null      float64
 2       2         442 non-null      float64
 3       3         442 non-null      float64
 4       4         442 non-null      float64
 5       5         442 non-null      float64
 6       6         442 non-null      float64
 7       7         442 non-null      float64
 8       8         442 non-null      float64
 9       9         442 non-null      float64

dtypes: float64(10)
memory usage: 34.7 KB
```

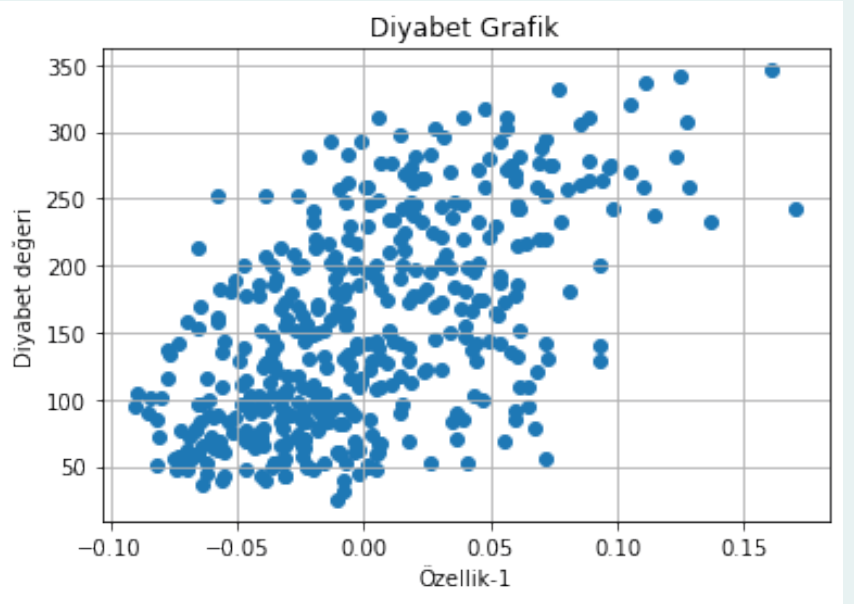
3. Adım: Veri setini eğitim ve veri seti olarak ayırınız. Veri setindeki 422 kaydı (ilk kayıttan son 20 kayda kadar) eğitim seti olarak son 20 kaydı ise test seti olarak belirleyiniz.

```
# Veri seti eğitim ve test olarak bölünür.
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]
# Hedef değişken eğitim ve test olarak bölünür.
diabetes_y_train = diabetes_y[:-20]
diabetes_y_test = diabetes_y[-20:]
```

4. Adım: Özellik sütunu ve diyabet değeri arasındaki ilişkiyi görebilmek için veri görselleştirme saçılım grafiği oluşturunuz. Grafikte özellik ile diyabet arasında doğrusal bir ilişki olduğunu gözlemleyiniz.

```
# Veri görselleştirme ile özellik ile diyabet arasındaki ilişki
görselleştirilir.
plt.scatter(diabetes_X, diabetes_y)
plt.title("Diyabet Grafik")
plt.xlabel("Özellik-1")
plt.ylabel("Diyabet değeri")
plt.grid(True)
plt.show()
```

Çıktı (Görsel 2.43)



Görsel 2.43: Saçılım grafiği

5. Adım: Model için bir regresyon nesnesi oluşturunuz. Eğitim seti kullanarak modeli eğitiniz.

```
# Yeni doğrusal regresyon nesnesi oluşturulur.
regr = linear_model.LinearRegression()
# Eğitim seti kullanılarak model eğitilir.
regr.fit(diabetes_X_train, diabetes_y_train)
```

Çıktı

```
LinearRegression()
```

6. Adım: Eğitilen modelin test veri setindeki özelliklerini kullanarak tahminler yapınız.

```
# Test seti kullanılarak tahminler yapılır.
diabetes_y_pred = regr.predict(diabetes_X_test)
diabetes_y_pred
```

Çıktı

```
array([225.9732401 , 115.74763374, 163.27610621, 114.73638965,
       120.80385422, 158.21988574, 236.08568105, 121.81509832,
       99.56772822, 123.83758651, 204.73711411,  96.53399594,
       154.17490936, 130.91629517,  83.3878227 , 171.36605897,
       137.99500384, 137.99500384, 189.56845268,  84.3990668 ])
```

7. Adım: Modele ait sabit (bias) ve katsayıları görüntüleyiniz.

```
# Regresyon denklemi sabit ve katsayılar.
print('Sabit (bias): \n', regr.intercept_)
print('Katsayı: \n', regr.coef_)
```

Çıktı

```
Sabit (bias):
    152.91886182616167
Katsayı:
    [938.23786125]
```

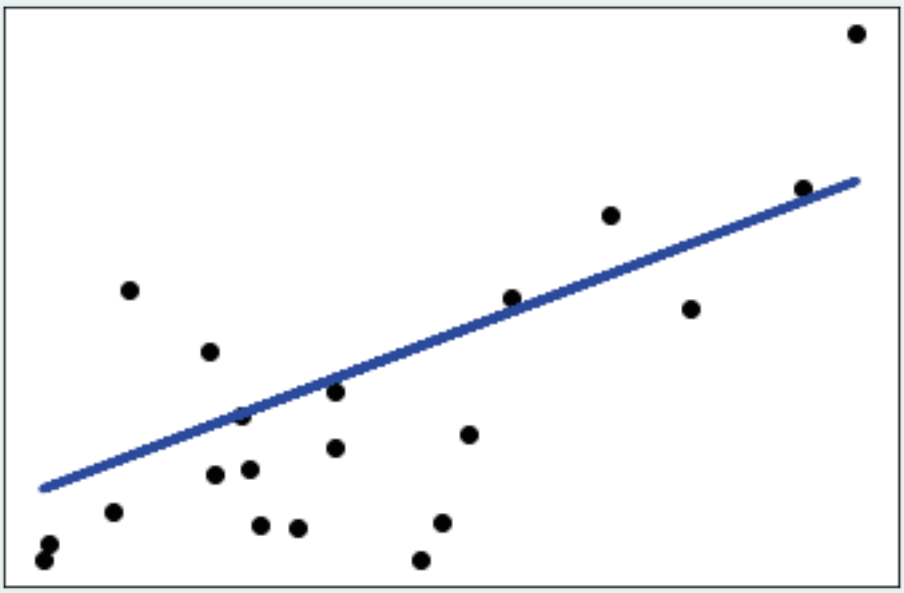
Sabit ve katsayıları kullanarak regresyon formülü oluşturulabilir. Bu formülle tahminler hesaplanabilir.

```
diyabet_degeri = 152.92+(938.24).ozellik
```

8. Adım: Gerçek değerler ile hesaplanan regresyon çizgisini görselleştiriniz.

```
# Gerçek değerler.
plt.scatter(diabetes_X_test, diabetes_y_test, color = "black")
# Regresyon doğrusu.
plt.plot(diabetes_X_test, diabetes_y_pred, color = "blue",
linewidth = 3)
plt.xticks(())
plt.yticks(())
plt.show()
```

Çıktı (Görsel 2.44)



Görsel 2.44: Lineer regresyon

9. Adım: Model performansını değerlendirmek için ortalama kare hata ve R kare değerlerini hesaplayınız.

```
# Model Değerlendirme.
# Ortalama Kare Hata.
print("Ortalama Kare Hata: %.2f" % mean_squared_error(diabetes_y_
test, diabetes_y_pred))
# R Kare: 1 mükemmel tahmin.
print("R Kare: %.2f" % r2_score(diabetes_y_test, diabetes_y_
pred))
```

Çıktı

```
Ortalama Kare Hata: 2548.07
R Kare: 0.47
```

3. UYGULAMA

Çoklu doğrusal regresyon uygulamasını gerçekleştirmek için aşağıdaki işlem adımlarını takip ediniz.

Bu uygulamada Kaggle'da bulunan konut fiyatı veri seti kullanılmıştır. Bu veri setinde konutun alan, oda sayısı, bina yaşı gibi özellikleri ve fiyatı (bağımlı değişken) sütun olarak bulunmaktadır. Oluşturulan model, özellikler verildiğinde bir konut fiyatı hesaplayarak tahmin etmektedir.

1. Adım: Regresyon modeli ve değerlendirme metrikleri ile ilgili kütüphaneleri içe aktarınız.

```
# Temel kütüphaneler içe aktarılır.
import numpy as np
import pandas as pd
from sklearn import linear_model
```

2. Adım: Veri setini yükleyiniz ve veriye göz atınız.

```
# Veri seti yüklenir.
data = pd.read_csv('../content/sample_data/homeprices.csv')
# Veriye göz atılır.
data.head()
```

Çıktı

	area	bedrooms	age	price
0	2600	3.0	20	550000
1	3000	4.0	15	565000
2	3200	NaN	18	610000
3	3600	3.0	30	595000
4	4000	5.0	8	760000

3. Adım: Veri setinin daha iyi anlaşılabilmesi için sütun adlarını Türkçeye çeviriniz ve veri setine göz atınız.

```
# Sütun adları Türkçeye çevrilmiştir.
data.rename(columns = {'area':'alan', 'bedrooms':'oda_sayisi',
'age':'yasi', 'price':'fiyati'}, inplace = True)
# Veri setine göz atılır.
data.head()
```

Çıktı

	alan	oda_sayisi	yasi	fiyati
0	2600	3.0	20	550000
1	3000	4.0	15	565000
2	3200	NaN	18	610000
3	3600	3.0	30	595000
4	4000	5.0	8	760000

4. Adım: Veri seti hakkında bilgi alarak veri türlerini ve eksik verileri belirleyiniz.

```
# Veri türleri ve eksik veri bilgisi görüntülenir.
data.info()
# Eksik veri ortanca değer ile tamamlanır.
# Veri setinin özelliğine göre farklı yöntemler kullanılabilir.
median_oda_sayisi = data['oda_sayisi'].median()
median_oda_sayisi
```

Çıktı

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 4 columns):
 #     Column      Non-Null Count      Dtype
---    -------     ---------------     ------
 0      alan         6 non-null        int64
 1   oda_sayisi      5 non-null       float64
 2      yasi         6 non-null        int64
 3     fiyati        6 non-null        int64

dtypes: float64(1), int64(3)
memory usage: 320.0 bytes
```

5. Adım: Eksik veri veri setinden çıkarılır veya tamamlanır. Bu aşamada eksik veriyi ortanca değer ile tamamlayınız. Oda sayısı ortanca değer 4 olarak hesaplanmıştır. Eksik veri 4 olarak tamamlanır.

```
# Eksik veri ortanca değer ile tamamlanır.
# Veri setinin özelliğine göre farklı yöntemler kullanılabilir.
median_oda_sayisi = data['oda_sayisi'].median()
median_oda_sayisi
data.oda_sayisi = data.oda_sayisi.fillna(median_oda_sayisi)
data
```

Çıktı

```
Ortalama Kare Hata: 2548.07
R Kare: 0.47
       alan        oda_sayisi        yasi        fiyati
0      2600            3.0           20         550000
1      3000            4.0           15         565000
2      3200            4.0           18         610000
3      3600            3.0           30         595000
4      4000            5.0            8         760000
5      4100            6.0            8         810000
```

6. Adım: Bir çoklu doğrusal regresyon modeli oluşturunuz ve veri setini kullanarak eğitiniz. Veri seti 6 örnekten oluştuğu için eğitim için tamamı kullanılmıştır. Veri setindeki **alan, oda_sayisi** ve **yasi** özellik sütunlarıdır. **fiyati** sütunu ise hedef sütunudur.

```
# Çoklu doğrusal regresyon modeli eğitilir.
# Özellik olarak alan, oda_sayisi ve yasi sütunları seçilir.
# Tahmin edilecek değişken olarak fiyat sütunu belirlenir.
reg = linear_model.LinearRegression()
reg.fit(data[['alan', 'oda_sayisi', 'yasi']], data.fiyati)
```

Çıktı

```
LinearRegression()
```

7. Adım: Modele ait sabit (bias) ve katsayıları görüntüleyiniz.

```
# Regresyon denklemi sabit ve katsayılar.
print('Sabit (bias): \n', reg.intercept_)
print('Katsayılar: \n', reg.coef_)
```

Çıktı

```
Sabit (bias):
   221323.00186540396
Katsayılar:
   [112.06244194 23388.88007794 -3231.71790863]
```

Sabit ve katsayıları kullanarak regresyon formülü oluşturulabilir. Bu formülle tahminler hesaplanabilir.

```
fiyati = 221323 + (112.06.alan ) + (23388.88.oda_sayisi) – (3231.72.yasi)
```

8. Adım: Modeli kullanarak veri setindeki özellikler ile fiyat tahminleri yapınız.

```
# Model kullanılarak tahminler yapılır.
reg_pred = reg.predict(data[['alan', 'oda_sayisi', 'yasi']])
```

9. Adım: Model performansını değerlendirmek için ortalama kare hata ve R kare değerlerini hesaplayınız.

```
# Model değerlendirme metrikleri içe aktarılır.
from sklearn.metrics import mean_squared_error, r2_score
# Ortalama Kare Hata
print("Ortalama Kare Hata: %.2f" % mean_squared_error(data.
fiyati, reg_pred))
# R Kare: 1 mükemmel tahmin.
print("R Kare: %.2f" % r2_score(data.fiyati, reg_pred))
```

Çıktı

```
Ortalama Kare Hata: 446305128.22
R Kare: 0.96
```

10. Adım: Yeni bir tahmin için modeli kullanınız. 3000 metrekare alana sahip 3 odalı 40 yaşında bir konut için fiyat tahmini yapılır.

```
reg.predict([[3000,3,40]])
```

Çıktı

```
array([498408.25158031])
```

4. UYGULAMA

Polinominal regresyon uygulamasını gerçekleştirmek için aşağıdaki işlem adımlarını takip ediniz.

Bu uygulamada Kaggle'da bulunan pozisyon maaş veri seti kullanılmıştır. Veri setinde pozisyon, pozisyon düzeyi (bağımsız değişkenler) ve maaş (bağımlı değişken) sütunları bulunmaktadır. Oluşturulan model verilen pozisyon değerine göre maaşı tahmin etmektedir.

1. Adım: Regresyon modeli, veri görselleştirme ve değerlendirme metrikleri ile ilgili kütüphaneleri içe aktarınız.

```
# Temel kütüphaneler içe aktarılır.
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
import matplotlib.pyplot as plt
import numpy as np
```

```
# Veri görselleştirme.
import seaborn as sns
plt.style.use('fivethirtyeight')
# Uyarılar kapatılır.
import warnings
warnings.filterwarnings('ignore')
# Performans değerlendirme metrikleri.
from sklearn.metrics import r2_score # r kare
```

2. Adım: Veri setini yükleyiniz ve veri setine göz atınız.

```
# Veri seti yüklenir.
df = pd.read_csv("../content/sample_data/Position_Salaries.csv")
# Veri seti yazdırılır.
print (df)
```

Çıktı

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

3. Adım: Veri setinin daha iyi anlaşılabilmesi için sütun adlarını Türkçeye çeviriniz ve veri setine göz atınız.

```
# Veri setinin daha iyi anlaşılması için sütun isimleri Türkçeye
çevrilir.
df.columns = ['pozisyon', 'pozisyon_seviyesi','maas']
# Yeni sütun adları ile veri seti tekrar görüntülenir.
df.head()
```

Çıktı

	pozisyon	pozisyon_seviyesi	maas
0	Business Analyst	1	45000
1	Junior Consultant	2	50000
2	Senior Consultant	3	60000
3	Manager	4	80000
4	Country Manager	5	110000

4. Adım: Veri setindeki **pozisyon_seviyesi** bağımsız değişken, **maas** ise bağımlı değişkendir.

```
# pozisyon_seviyesi bağımsız değişken olarak belirlenir.
X = df.iloc[:,1:2].values
# maas bağımlı değişken olarak belirlenir.
y = df.iloc[:,2].values
```

5. Adım: İlk olarak basit doğrusal regresyon modelini tanımlayınız ve modeli eğitiniz.

```
# Lineer regresyon modeli tanımlanır ve model eğitilir.
lin_reg = LinearRegression()
lin_reg.fit(X,y)
```

6. Adım: Bu aşamada ikinci dereceden bir polinominal regresyon modeli oluşturunuz ve modeli eğitiniz.

```
poly_reg2 = PolynomialFeatures(degree = 2)
# Polinom derecesi 2 olarak belirlenir.
X_poly = poly_reg2.fit_transform(X)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(X_poly,y)
```

Çıktı

```
LinearRegression()
```

7. Adım: Üçüncü dereceden bir polinominal regresyon modeli oluşturunuz ve modeli eğitiniz.

```
poly_reg3 = PolynomialFeatures(degree=3)
# Polinom derecesi 3 olarak belirlenir.
X_poly3 = poly_reg3.fit_transform(X)
lin_reg_3 = LinearRegression()
lin_reg_3.fit(X_poly3,y)
```

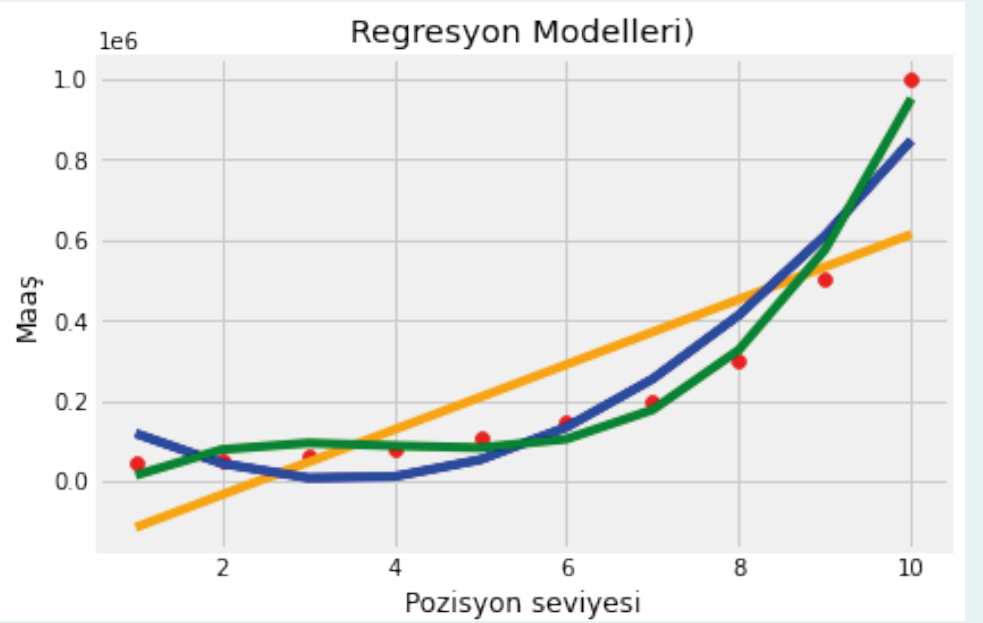
Çıktı

```
LinearRegression()
```

8. Adım: Modeller kullanarak tahminler yapınız.

```
# Tahminler
y_tahmin_lineer = lin_reg.predict(X)
y_tahmin_poly2 = lin_reg_2.predict(poly_reg2.fit_transform(X))
y_tahmin_poly3 = lin_reg_3.predict(poly_reg3.fit_transform(X))
```

9. Adım: Modellerin veri setindeki maaş değerlerini tahmin etmek için kullanınız. Veri görselleştirme ile regresyon modellerini inceleyiniz (Görsel 2.45).



Görsel 2.45: Regresyon modelleri

Kırmızı noktalar gerçek veri noktalarını temsil eder. Turuncu doğru doğrusal regresyon modelini, mavi çizgi polinom ikinci derece polinominal regresyon modelini ve yeşil çizgi de üçüncü derece polinominal regresyon modelini temsil eder. Üçüncü derece polinomal regresyon modelinin gerçek veri değerlerini daha iyi tahmin ettiği görülmektedir.

10. Adım: Model performansını değerlendirmek için R kare değerlerini hesaplayınız. R Kare'de 1 değeri tam uyumu gösterir. Regresyon modelleri incelendiğinde polinominal regresyon modellerinin daha iyi performans gösterdiği görülmektedir.

```
# R Kare Değeri.
print("Doğrusal Regresyon R Kare Değeri: %.2f" % r2_score(y,y_
tahmin_lineer))
print("İkinci derece polinomsal regresyon R kare değeri: %.2f" %
r2_score(y,y_tahmin_poly2))
print("Üçüncü derece polinomsal regresyon R Kare değeri: %.2f" %
r2_score(y,y_tahmin_poly3))
```

Çıktı

```
Doğrusal Regresyon R Kare Değeri: 0.67
İkinci derece polinomsal regresyon R kare değeri: 0.92
Üçüncü derece polinomsal regresyon R Kare değeri: 0.98
```

NOT !!! R kare değeri modelin uyumluluğunu gösteren bir ölçüttür. R Kare değeri 1'e yaklaştıkça modelin uyumu artar.

11. Adım: Modelleri kullanarak bir maaş tahmini yapınız.

```
# Bir çalışanın maaşını tahmin etmek.
print("Lineer Regresyon Tahmini: ", lin_reg.predict([[6.5]]))
print("İkinci Derece Pol.Regresyon Tahmini: ",lin_reg_2.
predict(poly_reg2.fit_transform([[6.5]])))
print("İkinci Derece Pol. Tahmini: ", lin_reg_3.predict(poly_
reg3.fit_transform([[6.5]])))
```

Çıktı

```
Lineer Regresyon Tahmini: [330378.78787879]
İkinci Derece Pol.Regresyon Tahmini: [189498.10606061]
İkinci Derece Pol.Tahmini: [133259.46969697]
```

5. UYGULAMA

Lojistik regresyon uygulamasını gerçekleştirmek için aşağıdaki işlem adımlarını takip ediniz.

Bu uygulamada Kaggle'da bulunan Pima Indians Diabetes veri seti kullanılmıştır. Veri setinde, birkaç tıbbi özellik (bağımsız) değişken ve bir hedef (bağımlı) değişken olan **sonuc** bulunmaktadır. Bağımsız değişkenler; hastanın sahip olduğu gebelik sayısı, vücut kitle indeksi (**bmi**), insülin düzeyi ve yaşı gibi özelliklerdir. Sonuç özelliği diyabet olup olmama durumudur.

1. Adım: Temel kütüphaneleri ve modülleri içe aktarınız.

```
# Temel kütüphaneler içe aktarılır.
import numpy as np # Lineer cebir kütüphanesi.
import pandas as pd # Veri işleme.
```

```
# Veri görselleştirme.
import seaborn as sns
%matplotlib inline
from matplotlib import pyplot as plt
from matplotlib import style

# Algoritmalar.
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

# Model performans değerlendirme metrikleri.
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from random import randrange
import warnings # Uyarılar kapatılır.
warnings.filterwarnings("ignore")
```

2. Adım: Veri setini yükleyiniz ve görüntüleyiniz.

```
# Veri seti yüklenir.
diyabet_df= pd.read_csv("../content/sample_data/diabetes.csv")
diyabet_df.head()
```

Çıktı

```
Lineer Regresyon Tahmini: [330378.78787879]
İkinci Derece Pol.Regresyon Tahmini: [189498.10606061]
İkinci Derece Pol.Tahmini: [133259.46969697]
```

Pregnan-cies	Glucose	Blood Pressure	Skin Thickness	Insulin	BMI	Diabetes Pedigree Function	Age	Out-come	
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

3. Adım: Eğitim veri setini daha iyi anlayabilmek için sütun adlarını Türkçeye çeviriniz.

```
diyabet_df.rename(columns = {'Pregnancies':'gebelik',
'Glucose':'glikoz', 'BloodPressure':'kan_basinci',
'SkinThickness':'cilt_kalinligi', 'Insulin':'insulin',
'BMI':'vki', 'DiabetesPedigreeFunction':'diyabet_fonk',
'Age':'yas', 'Outcome':'sonuc'}, inplace = True)
diyabet_df.head()
```

gebelik	glikoz	kan_ basinci	cilt_ kalinligi	insulin	vki	diyabet_ fonk	yas	sonuc	
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

4. Adım: Veri seti hakkında bilgi alınız. Veri setinde eksik veri bulunmadığını onaylayınız.

```
diyabet_df.info()
```

Çıktı

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):

 #        Column          Non-Null Count      Dtype
---       -------         ---------------     ------
 0        gebelik          768 non-null       int64
 1        glikoz           768 non-null       int64
 2      kan_basinci        768 non-null       int64
 3    cilt_kalinligi       768 non-null       int64
 4       insulin           768 non-null       int64
 5         vki             768 non-null      float64
 6     diyabet_fonk        768 non-null      float64
 7         yas             768 non-null       int64
 8        sonuc            768 non-null       int64

dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

5. Adım: Veri setindeki bağımsız değişkenleri ve bağımlı değişkenleri ayırınız. Veri setinin %70'ini eğitim ve %30'unu test olarak ayırınız ve sadece değerleri alınız.

```
X = diyabet_df[['gebelik', 'glikoz', 'kan_basinci',
'cilt_kalinligi', 'insulin', 'vki', 'diyabet_fonk',
'yas' ]] # bağımsız değişkenler.
y = diyabet_df[['sonuc']] # bağımlı değişken.
# Veri seti eğitim ve test olarak ayrılır.
X_train,X_test,y_train,y_test = train_test_
split(X,y,test_size = 0.3, random_state = 0)
# Veri setinde sadece değerler alınır.
diyabet_df = diyabet_df.values
diyabet_df
```

Çıktı

```
array([[  6.  , 148.  ,  72.  , ...,  0.627,  50.  ,   1.  ],
       [  1.  ,  85.  ,  66.  , ...,  0.351,  31.  ,   0.  ],
       [  8.  , 183.  ,  64.  , ...,  0.672,  32.  ,   1.  ],
       ...,
       [  5.  , 121.  ,  72.  , ...,  0.245,  30.  ,   0.  ],
       [  1.  , 126.  ,  60.  , ...,  0.349,  47.  ,   1.  ],
       [  1.  ,  93.  ,  70.  , ...,  0.315,  23.  ,   0.  ]])
```

6. Adım: Lojistik regresyon nesnesini oluşturunuz ve modeli eğitiniz.

```
logistic_model = LogisticRegression()
logistic_model.fit(X_train,y_train)
```

Çıktı

```
LogisticRegression()
```

7. Adım: Modeli kullanarak tahminler yapınız. Hata matrisi ve sınıflandırma raporu kullanarak modelin performansını değerlendiriniz.

```
# Test seti kullanılarak tahminler yapılır.
tahminler = logistic_model.predict(X_test)

# Performans Değerlendirmesi.
print("Hata Matrisi")
matris = confusion_matrix(y_test,tahminler)
print(matris)
print("\nSınıflandırma Raporu")
rapor = classification_report(y_test,tahminler)
print(rapor)
lr_accuracy = accuracy_score(y_test, tahminler)
print('Lojistik Regresyon model doğruluğu: {:.2f}%'.format(lr_
accuracy*100))
```

Çıktı

```
Hata Matrisi
  [[141  16]
  [ 35  39]]

Sınıflandırma Raporu
              precision    recall  f1-score   support

           0       0.80      0.90      0.85       157
           1       0.71      0.53      0.60        74

    accuracy                           0.78       231
   macro avg       0.76      0.71      0.73       231
weighted avg       0.77      0.78      0.77       231

Lojistik Regresyon model doğruluğu: 77.92%
```

Model test verisindeki örneklerden %77,92'sini doğru olarak sınıflandırmıştır. %30'u test olarak ayrılır ve sadece değerler alınır.

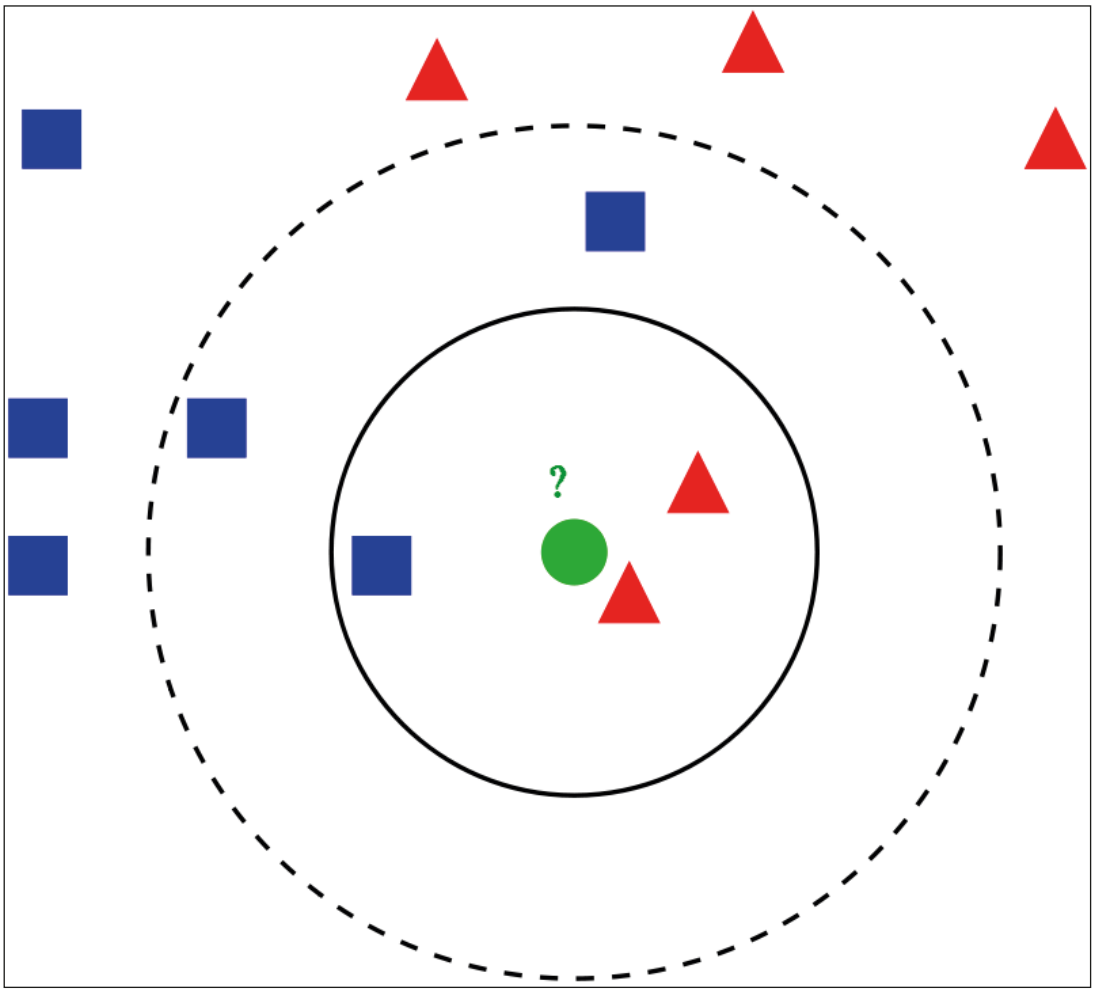
2.5. DİĞER GÖZETİMLİ ÖĞRENME ALGORİTMALARI

Bu bölümde regresyon dışında sık kullanılan gözetimli öğrenme algoritmalarına ilişkin uygulamalara yer verilmiştir.

2.5.1. K En Yakın Komşu (K-Nearest Neighbors-K-NN)

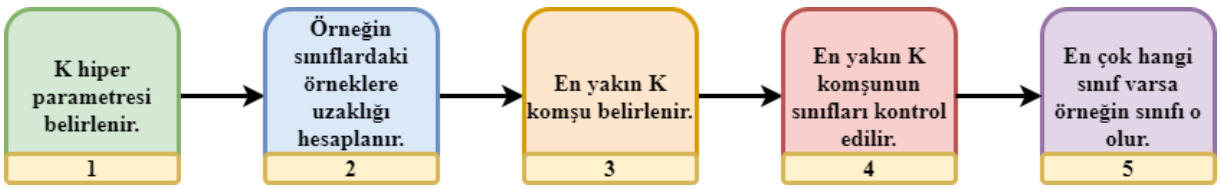
Sınıflandırma tahminleri için kullanılan denetimli bir makine öğrenmesi algoritmasıdır.

Yeni örneğin mevcut örneklere göre uzaklığı hesaplanır. Örneğin K sayıda yakın komşuya uzaklığı hesaplanarak örneğin hangi sınıfa ait olduğu belirlenir (Görsel 2.46).



Görsel 2.46: Sınıflandırma

K-NN algoritmasının aşamaları görseldeki gibi sıralanabilir (Görsel 2.47).



Görsel 2.47: K-NN algoritması

6. UYGULAMA

K En Yakın Komşu (K-Nearest Neighbors-K-NN) uygulamasını yapmak için aşağıdaki işlem adımlarını takip ediniz.

Bu uygulamada iris veri seti kullanılarak bir sınıflama uygulaması yapılmaktadır. Veri kümesi dört öznitelikten oluşur: sepal-width, sepal-length, petal-width ve petal-length. Bunlar, belirli iris bitkisi türlerinin özellikleridir. Amaç, bu bitkilerin ait olduğu sınıfı tahmin etmektir. Veri kümesinde üç sınıf vardır: Iris-setosa, Iris-versicolor ve Iris-virginica.

1. Adım: Temel kütüphaneleri ve modülleri içe aktarınız.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

2. Adım: UCI'den iris veri setini yükleyiniz. Veri setindeki sütunların adlarını belirleyiniz. Pandas data frame kullanarak veri setini yükleyiniz. head() metodunu kullanarak veri setine göz atınız.

```
# iris veri seti url adresinden çekilmektedir.
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/
iris/iris.data"

# Veri setindeki sütunların adları belirlenir.
sutunlar = ['sepal_uzunluk', 'sepal_genislik', 'petal_uzunluk',
'petal_genislik', 'sinif']

# Veri dosyası yüklenir.
iris_df = pd.read_csv(url, names = sutunlar)

# Veri seti kontrol edilir.
iris_df.head()
```

Çıktı

	sepal_ uzunluk	sepal_ genislik	petal_ uzunluk	petal_ genislik	sinif
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

3. Adım: Veri türlerini ve eksik verileri kontrol ediniz. Veri setinde eksik veri bulunmadığını onaylayınız.

```
# Veri seti hakkında bilgi alınır.
iris_df.info()
```

Çıktı

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
```

```
#         Column          Non-Null Count       Dtype
---       -------         ---------------      ------
0     sepal_uzunluk       150 non-null       float64
1     sepal_genislik      150 non-null       float64
2     petal_uzunluk       150 non-null       float64
3     petal_genislik      150 non-null       float64
4          sinif          150 non-null       object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

4. Adım: Veri setindeki özellikler ve etiket sütununu ayırınız. **sepal_uzunluk**, **sepal_genislik**, **petal_uzunluk** ve **petal_genislik** sütunları özelliklerdir. **sinif** ise etiket sütunudur.

```
# Veri setindeki özellikler ve hedef sütun ayrılır.

X = iris_df.iloc[:, :-1].values
y = iris_df.iloc[:, 4].values
```

5. Adım: Veri setinin %80'i eğitim %20'si ise test için kullanılmaktadır.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_
size = 0.20, random_state = 60)

# random_state parametresine değer verildiğinde hep aynı alt
kümeyi seçmesini sağlar.
```

6. Adım: Model eğitilirken özelliklerdeki değerlerin modeli domine etmesini engellemek için ölçeklendirme yapınız.

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

7. Adım: KNeighborsClassifier içe aktarınız. Bir sınıflandırıcı oluşturunuz. n_neighbors parametresine farklı değerler verilebilir. Bu uygulamada en yakın komşu sayısı 5 olarak belirlenmiştir. Modelin performansına göre bu parametre değiştirilebilir.

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5)
classifier.fit(X_train, y_train)
```

Çıktı

```
KNeighborsClassifier()
```

8. Adım: Modeli kullanarak tahminler yapınız.

```
y_pred = classifier.predict(X_test)
print (y_pred) # Tahminler yazdırılır.
```

Çıktı

```
['Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa'
 'Iris-versicolor' 'Iris-setosa' 'Iris-versicolor' 'Iris-setosa'
 'Iris-setosa' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica'
 'Iris-virginica' 'Iris-setosa' 'Iris-versicolor' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-setosa']
```

9. Adım: Model performansının ve sınıflandırma performansının değerlendirilmesi için hata matrisini (confusion matrix) kullanınız. Hata matrisi kesinlik, geri çağırma ve doğruluk skoru verir. Sklearn kütüphanesi kullanılarak classification_report ve hata matrisi elde edilebilir.

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

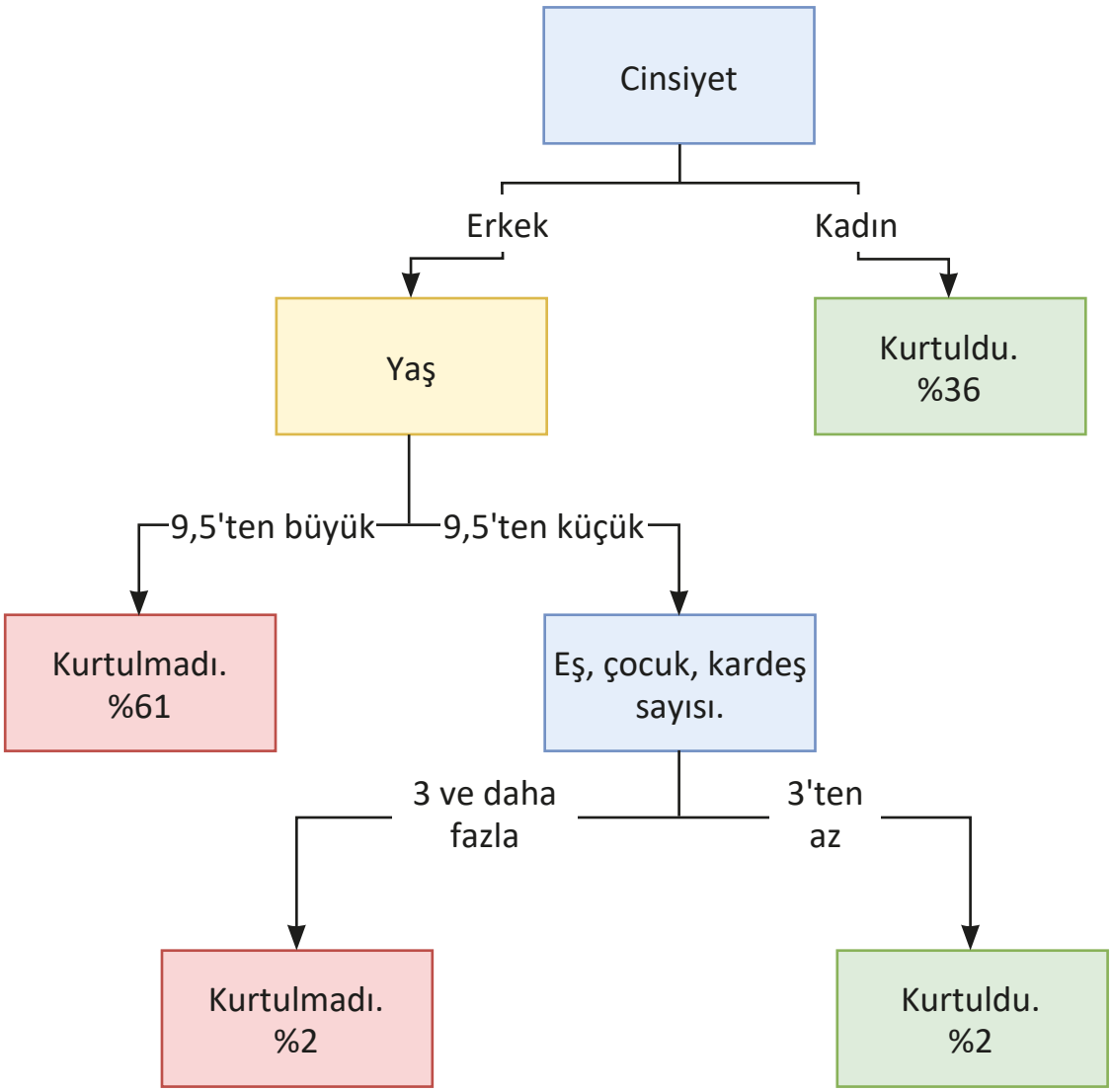
Çıktı

```
[[13  0  0]
 [ 0 10  0]
 [ 0  2  5]]

                precision    recall  f1-score   support

   Iris-setosa       1.0      1.00      1.00        13
Iris-versicolor      0.83      1.00      0.91        10
Iris-virginica       1.00      0.71      0.83         7

      accuracy                           0.93        30
     macro avg       0.94      0.90      0.91        30
  weighted avg       0.94      0.93      0.93        30
```

Model değerlendirildiğinde sadece 2 örneğin yanlış sınıflandırıldığı görülmektedir. Modelin doğruluğu %93,33 olarak hesaplanmıştır. Model performansı metrikler incelenerek daha ayrıntılı olarak değerlendirilebilir.

2.5.2. Karar Ağacı (Decision Tree)

Sınıflandırma ve regresyon tahminleri için kullanılan denetimli bir makine öğrenmesi algoritmasıdır. Bağımsız değişkenler ile bağımlı değişken arasındaki ilişki bir ağaç yapısı ve onun dallarıyla modellenir. Karar ağaçları hem sayısal hem de kategorik veride çalışır. Karar ağaçları dallanırken kategorik değişkenler için entropi ve gini değerleri sürekli sayısal değişkenler için ise en küçük kareler yöntemidir. Karar ağaçlarının dal sayısı modelin aşırı uyumuna neden olabilir. Örnekte Titanic kazası sonucu insanların kurtulma durumlarını modelleyen bir karar ağacı görselleştirilmiştir (Görsel 2.48). Karar ağacındaki dallar veri setindeki özelliklere ilişkin şartlara bağlı birinin kazadan kurtulup kurtulmadığını modellemektedir.



Görsel 2.48: Titanik kazası karar ağacı

7. UYGULAMA

Karar Ağacı uygulamasını yapmak için aşağıdaki işlem adımlarını takip ediniz.

Bu uygulamada bir banknotun dört farklı özelliğine bakarak onun gerçek olup olmadığını tahmin eden bir model geliştirilmektedir.

1. Adım: Temel kütüphaneleri ve modülleri içe aktarınız.

```
# Temel kütüphaneler içe aktarılır.
import pandas as pd
import numpy as np
# Veri görselleştirme.
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn import tree
```

2. Adım: Veri setini yükleyiniz. Veri setinde **varyans, carpiklik, basiklik ve entropi** banknotun özellikleridir. **sinif** ise etiket sütunudur.

```
# Veri setinin yüklenmesi.
banknot_df=pd.read_csv('/content/sample_data/data_banknote_
authentication.txt', sep = ',', header = None,
names = ["varyans","carpiklik", "basiklik", "entropi", "sinif" ])

# Veri seti kontrol edilir.
banknot_df.head()
```

Çıktı

	varyans	carpiklik	basiklik	entropi	sinif
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0

3. Adım: Veri türleri ve eksik veri hakkında bilgi alınız. Veri setinde eksik veri bulunmadığını onaylayınız.

```
# Veri seti hakkında bilgi alınır.
banknot_df.info()
```

Çıktı

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1372 entries, 0 to 1371
Data columns (total 5 columns):
 #         Column          Non-Null Count       Dtype
---        -------         ---------------      ------
 0         varyans          1372 non-null       float64
 1        carpiklik         1372 non-null       float64
 2        basiklik          1372 non-null       float64
 3        entropi           1372 non-null       float64
 4         sinif            1372 non-null        int64

dtypes: float64(4), int64(1)
memory usage: 53.7 KB
```

4. Adım: X, veri setindeki nitelikleri, y (sinif sütunu) ise etiketleri içerir. Veri setini eğitim ve test olmak üzere ikiye bölünüz. Veri setinin %80'ini eğitim, %20'sini test için kullanınız.

```
X = banknot_df.drop('sinif', axis = 1) # sinif sütunu çıkarılır
ve özellikler seçilir.
y = banknot_df['sinif'] # Etiketler belirlenir.
# Veri seti eğitim ve test olarak bölünür.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_
size = 0.20, random_state = 60)
# random_state parametresine değer verildiğinde hep aynı alt
kümeyi seçmesini sağlar.
```

5. Adım: Eğitim verisi kullanarak modeli eğitiniz.

```
# Modelin eğitimi.
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)
```

Çıktı

```
DecisionTreeClassifier()
```

6. Adım: DecisionTreeClassifier sınıflandırıcısı oluşturunuz. Eğitim veri seti kullanarak modeli eğitiniz ve tahminleri yazdırınız.

```
y_pred = classifier.predict(X_test)
print (y_pred) # Tahminler yazdırılır.
```

Çıktı

```
[0 0 0 0 0 0 1 0 1 0 1 1 0 0 1 0 0 1 0 1 0 1 0 1 1 1 0 1 0 0 0 1
 0 1 0 1 0 1 1 1 1 0 0 1 0 0 1 0 1 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0
 1 1 1 0 1 1 1 0 1 0 1 1 0 0 0 0 0 0 1 0 1 0 1 1 1 1 1 1 0 0 1 0
 0 1 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 0 0 1 0 0 0 0 1 0 0 1 0 0 1 1
 0 1 0 0 1 1 0 1 0 1 1 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 1 0 0 0 1 1
 1 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 1  0 0 0 0 0 1 0
 0 1 0 0 1 1 1 1 1 1 1 0 1 0 1 0 1 0 0 0 1 0 1 1 0 0 0 0 1 1  1 0
 0 0 0 0 0 1 0 1 1 0 0 1 0 1 0 0 0 0 1 0 1 0 0 0 0 1 1 1 0 0 0 0
 0 0 0 1 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0]
```

7. Adım: Oluşturulan modelin performansını değerlendirmek için hata matrisi ve sınıflama raporu oluşturunuz.

Çıktı

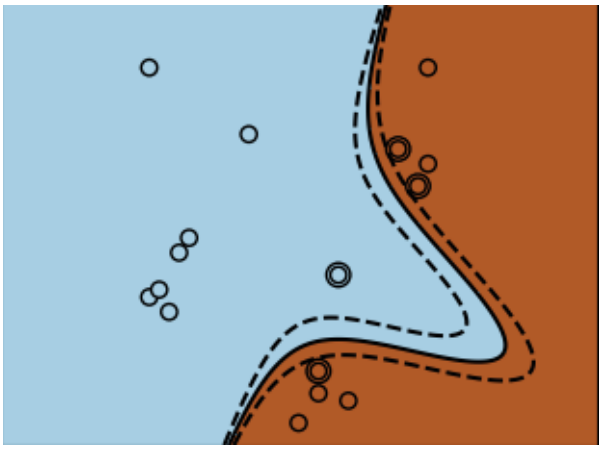
```
[[153   2]
 [  2 118]]
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       155
           1       0.98      0.98      0.98       120

    accuracy                           0.99       275
   macro avg       0.99      0.99      0.99       275
weighted avg       0.99      0.99      0.99       275
```

Model performansının değerlendirilmesi sınıflandırma performansının değerlendirilmesi için hata matrisi (confusion matrix) kullanılır. Hata matrisi kesinlik, geri çağırma ve doğruluk skoru verir. Sklearn kütüphanesi kullanılarak **classification_report** ve hata matrisi elde edilebilir. Model performansı incelendiğinde sadece 4 örneğin yanlış sınıflandırıldığı (%98,5 doğruluk) görülmektedir. Model performansı metrikler incelenerek daha ayrıntılı olarak değerlendirilebilir.

2.5.3. Destek Vektör Makineleri (SVM)

Sınıflandırma ve regresyon tahminleri için kullanılan denetimli bir makine öğrenmesi algoritmasıdır. Doğrusal ve doğrusal olmayan veri setlerinde SVM ile sınıflandırma yapılabilir. SVM, eğitim veri setindeki herhangi bir noktadan iki sınıf arasında bir karar sınırı bulan vektör uzayı tabanlı bir algoritmadır. İki sınıfa ait örnekler birbirinden en uygun şekilde destek vektörleri ile ayrılır.

Örnekte iki sınıfı ayıran destek vektörleri görülmektedir (Görsel 2.49).



Görsel 2.49: Destek vektör makinesi örneği

8. UYGULAMA

Destek Vektör Makineleri uygulamasını yapmak için aşağıdaki işlem adımlarını takip ediniz.

Bu uygulamada elma portakal veri seti kullanılmıştır. Veri setinde agirlik, boyut ve sinif etiketi (elma, portakal) sütunları bulunmaktadır. Bir SVM modeli eğitilerek genişlik ve boyut değerleri verilen bir meyvenin türü tahmin edilmektedir.

1. Adım: Kütüphaneleri yükleyiniz ve veri setini içe aktarınız. Veri setini kontrol ediniz.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Pandas kütüphanesi içe aktarılır.
import pandas as pd
# Veri dosyası yüklenir.
data = pd.read_csv("../content/sample_data/apples_and_oranges.
csv")
# Veri seti kontrol edilir.
data.head()
```

Çıktı

	Weight	Size	Class
0	69	4.39	orange
1	69	4.21	orange
2	65	4.09	orange
3	72	5.85	apple
4	67	4.70	orange

2. Adım: Veri setinin daha iyi anlaşılabilmesi için sütun adlarını Türkçeye çeviriniz. Veri setini kontrol ediniz.

```
sutunlar = ['agirlik', 'boyut', 'sinif']
data.columns = sutunlar
# Veri seti kontrol edilir.
data.head()
```

Çıktı

```
    agirlik   boyut    sinif
0      69      4.39    orange
1      69      4.21    orange
2      65      4.09    orange
3      72      5.85     apple
4      67      4.70    orange
```

3. Adım: Veri seti hakkında bilgi alınız. Veri setinde eksik veri bulunmadığını onaylayınız.

```
# Veri seti hakkında bilgi alınır.
data.info()
```

Çıktı

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40 entries, 0 to 39
Data columns (total 3 columns):
#         Column          Non-Null Count      Dtype
---       -------         ---------------     ------
0         agirlik           40 non-null        int64
1          boyut            40 non-null       float64
2          sinif            40 non-null       object

dtypes: float64(1), int64(1), object(1)
memory usage: 1.1+ KB
```

4. Adım: X veri kümesinin **agirlik** ve **boyut** sütunlarını (nitelikleri), y ise etiketlerini (**sinif**) içerir. Veri setini eğitim ve test olmak üzere ikiye bölünüz. Veri setinin %70'ini eğitim, %30'unu test için kullanınız.

```
from sklearn.model_selection import train_test_split
X = data.drop(['sinif'], axis = 'columns') # sinif sütunu atılır.
y = data.drop(['agirlik','boyut'], axis = 'columns') # agirlik
ve boyut sütunları atılır.
# Veri seti eğitim ve test olarak bölünür.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_
size = 0.3, random_state = 1)
print ("Eğitim veri setindeki veri sayısı:", len(X_train))
print ("Test veri setindeki veri sayısı:", len(X_test))
```

Çıktı

```
Eğitim veri setindeki veri sayısı: 28
Test veri setindeki veri sayısı: 12
```

5. Adım: Bir SVM modeli oluşturup eğitim seti kullanarak modeli eğitiniz.

```
# Destek vektör sınıflayıcı içe aktarılır.
from sklearn.svm import SVC

# Bir model oluşturulur.
model = SVC(kernel='rbf')

# Eğitim veri seti kullanılarak model eğitilir.
model.fit(X_train,y_train)
SVC(C = 1.0, cache_size = 200, class_weight = None, coef0 = 0.0,
  decision_function_shape = 'ovr', degree = 3, gamma = 'auto_deprecated',
  kernel = 'rbf', max_iter = -1, probability = False,
  random_state = None, shrinking = True, tol = 0.001, verbose = False)
```

Çıktı

```
SVC(gamma='auto_deprecated')
```

6. Adım: Model kullanarak tahminler yapınız.

```
y_pred = model.predict(X_test)
print (model.predict([[74, 5.3]]))
print (model.predict([[62, 3.0]]))
```

Çıktı

```
['apple']
['orange']
```

7. Adım: Modelin performansını değerlendirmek için hata matrisi ve sınıflandırma raporu kullanınız.

```
from sklearn.model_selection import train_test_split
X = data.drop(['sinif'], axis = 'columns') # sinif sütunu atılır.
y = data.drop(['agirlik','boyut'], axis = 'columns') # agirlik
ve boyut sütunları atılır.
# Veri seti eğitim ve test olarak bölünür.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_
size = 0.3, random_state = 1)
print ("Eğitim veri setindeki veri sayısı:", len(X_train))
print ("Test veri setindeki veri sayısı:", len(X_test))

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Çıktı

```
[[6 0]
 [1 5]]

              precision    recall  f1-score   support

           0       0.99      0.99      0.99       155
           1       0.98      0.98      0.98       120


    accuracy                           0.99       275
   macro avg       0.99      0.99      0.99       275
weighted avg       0.99      0.99      0.99       275
```

Model değerlendirildiğinde sadece 1 örneğin yanlış sınıflandırıldığı görülmektedir. Modelin doğruluğu %91,66 olarak hesaplanmıştır. Model performansı metrikler incelenerek daha ayrıntılı olarak değerlendirilebilir.

2.5.4. Topluluk Öğrenmesi (Ensemble Learning)

Topluluk öğrenmesi, kolektif öğrenme olarak da adlandırılır. Topluluk öğrenmesinde birden fazla model bir heyet gibi bir araya getirilerek karar vermek için kullanılır. Kendi başlarına çok iyi sonuçlar vermeyen bu temel modellerin (zayıf öğrenciler) birlikte kullanılmasıyla güçlü bir kolektif öğrenici, karar verici elde etmek amaçlanır. Bu sayede birden fazla zayıf öğrenici bir arada kullanılarak sınıflandırma ve regresyon için doğruluk oranları daha yüksek tahminler yapılabilir. Topluluk öğrenmesi modelleri torbalama (bagging), yükseltme (boosting) ve yığma (stacking) olarak üçe ayrılır.

Torbalama: Her seferinde eğitim setinin farklı bir alt kümesi alınarak temel modeller (homojen zayıf öğreniciler) paralel olarak eğitilir. Geliştirilen modellerin tahminleri kullanılarak sınıflandırma için oylama yoluyla regresyon için ise tahminlerin ortalaması yoluyla sonuç tahmin edilir. Rastgele Orman (Random Forest) torbalama algoritmalarına örnek olarak verilebilir.

Yükseltme: Eğitim setinin bir alt kümesiyle bir model geliştirilir. Sonraki model torbalamadaki gibi paralel bir şekilde bağımsız olarak geliştirilmez. İlk modelde hatalı tahminlere daha fazla ağırlık verilerek sonraki zayıf öğrenicinin bu eksikleri gidermesi sağlanır. İşlem bu şekilde tekrarlanarak öğrenici daha güçlü bir öğreniciye yükseltilir. AdaBoost, Gradient Boosting ve XGBoost yükseltme algoritmalarına örnek verilebilir.

Yığma: Torbalama ve yükseltmeden farklı olarak bir üst öğrenici (meta öğrenici) geliştirme mantığına dayanır. Sonuç tahmini, temel modellerin tahminlerini girdi olarak kullanan bir üst öğrenici model tarafından yapılır. Scikit-learn kütüphanesi içinde **sklearn.ensemble** modülü altında topluluk öğrenmesi modellerine erişilebilir.

2.5.4.1. Rastgele Orman (Random Forest)

Rastgele Orman algoritmasında eğitim setinin alt kümelerinde birden fazla karar ağacının rastgele özellikler alt kümesiyle eğitilmesi esasına dayanır. Sınıflandırma ve regresyon için karar ağaçlarının kolektif olarak karar vermesini sağlayan bir torbalama yöntemidir. Denetimli bir makine öğrenmesi algoritmasıdır. Karar ağaçlarında karşılaşılan aşırı uyum sorunun üstesinden gelmeye olanak verir. GradientBoost ve XGBoost gibi karar ağaçlarını kullanan daha gelişmiş yükseltme (boosting) yöntemleri de bulunur.

9. UYGULAMA

Rastgele orman uygulamasını yapmak için aşağıdaki işlem adımlarını takip ediniz.

Bu uygulamada Melbourne ev fiyatları veri seti kullanılmıştır. Veri setinde eve ait özellikler (banyo sayısı, mutfak sayısı, satış yılı vb.) ve hedef sütunu (satış fiyatı) yer almaktadır.

1. Adım: Uygulamada gerekli kütüphaneleri içe aktarınız.

```
# Temel kütüphaneler içe aktarılır.
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# Model değerlendirme metrikleri içe aktarılır.
from sklearn.metrics import mean_squared_error, r2_score
# Veri görselleştirme.
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.tree import plot_tree
```

2. Adım: Veri setini yükleyiniz ve kontrol ediniz.

```
# veri seti yüklenir.
melb_df = pd.read_csv("../content/sample_data/melb_data.csv")

# Veri seti kontrol edilir.
melb_df.head()
```

3. Adım: Veri seti hakkında bilgi alınız, veri türlerini ve eksik verileri kontrol ediniz. Veri setinde 21 sütun bulunmaktadır. Veri setinde 20 özellik ve hedef sütunu vardır. Veri setinde eksik veri bulunmaktadır.

```
# Veri seti hakkında bilgi alınır.
melb_df.info()
```

Çıktı

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13580 entries, 0 to 13579
Data columns (total 21 columns):
 #         Column          Non-Null Count      Dtype
---       -------         ---------------     ------
 0    Suburb               13580 non-null      object
 1    Address              13580 non-null      object
 2    Rooms                13580 non-null       int64
 3    Type                 13580 non-null      object
 4    Price                13580 non-null      float64
 5    Method               13580 non-null      object
 6    SellerG              13580 non-null      object
 7    Date                 13580 non-null      object
 8    Distance             13580 non-null      float64
 9    Postcode             13580 non-null      float64
 10   Bedroom2             13580 non-null      float64
 11   Bathroom             13580 non-null      float64
 12   Car                  13518 non-null      float64
 13   Landsize             13580 non-null      float64
 14   BuildingArea          7130 non-null      float64
 15   YearBuilt             8205 non-null      float64
 16   CouncilArea          12211 non-null      object
 17   Lattitude            13580 non-null      float64
 18   Longtitude           13580 non-null      float64
 19   Regionname           13580 non-null      object
 20   Propertycount        13580 non-null      float64

dtypes: float64(12), int64(1), object(8)
memory usage: 2.2+ MB
```

4. Adım: Veri setinden eksik veri satırlarını çıkarınız. X, veri setindeki nitelikleri, y fiyat sütununu içerir. Veri setinde çok fazla özellik bulunduğu için özelliklerden bazılarını seçiniz. Veri setini eğitim ve test olmak üzere ikiye bölünüz. Veri setinin %80'ini eğitim, %20'sini test için kullanınız.

```
# veri seti yüklenir.
melb_df = pd.read_csv("../content/sample_data/melb_data.csv")

# Veri seti kontrol edilir.
melb_df.head()


# Eksik satırlar veri setinden çıkarılır.
melb_df = melb_df.dropna(axis = 0)

y = melb_df.Price # hedef sütunu seçilir.
melbourne_ozellikleri=['Rooms','Bathroom','Landsize','BuildingArea',
'YearBuilt', 'Lattitude', 'Longtitude']
X = melb_df[melbourne_ozellikleri]

# Veri seti eğitim ve test olarak bölünmektedir.
from sklearn.model_selection import train_test_split
train_X, val_X, train_y, val_y = train_test_split(X, y, test_
size = 0.2, random_state = 0)
```

5. Adım: Scikit-learn'de bir karar ağacına benzer bir rastgele orman modeli oluşturunuz. **RandomForestRegressor** modelini içe aktarınız. Modeli eğitip tahminler yapınız.

```
# Veri seti hakkında bilgi alınır.
melb_df.info()

from sklearn.ensemble import RandomForestRegressor
forest_model = RandomForestRegressor()
forest_model.fit(train_X, train_y)
melb_preds = forest_model.predict(val_X)
```

6. Adım: Hedef sayısal bir değer olduğu için model performansının değerlendirilmesinde ortalama kare hata (MSE) ve R Kare değerlerini kullanınız.

```
# Ortalama Kare Hata.
print("Ortalama Kare Hata: %.2f" % mean_absolute_error(val_y,
melb_preds))

# R Kare: 1 mükemmel tahmin.
print("R Kare: %.2f" % r2_score(val_y, melb_preds))
```

Çıktı

```
Ortalama Kare Hata: 193489.41
R Kare: 0.70
```

Değerlendirme: Örnek veri setlerini kullanarak yapacağınız gözetimli öğrenme uygulaması aşağıda verilen kontrol listesi kullanılarak değerlendirilecektir.

GÖZETİMLİ ÖĞRENME UYGULAMASI KONTROL LİSTESİ				
SINIF	**NO**	**ÖĞRENCİ ADI SOYADI**	**DEĞERLENDİRME TARİHİ**	
Ölçütler			**Evet**	**Hayır**
1. Gözetimli öğrenme uygulaması için uygun kütüphaneleri dosyaya dahil etti.				
2. Veri setini yükledi.				
3. Veri seti hakkında bilgi almak için gerekli kodu yazdı.				
4. Veri setinde özellikleri ve çıktıyı ayırdı.				
5. Veri setini eğitim ve test verisi olarak ikiye böldü.				
6. Problemin amacına uygun modeli seçerek oluşturdu.				
7. Uygun metrikleri kullanarak sonucu değerlendirdi.				
"Hayır" olarak işaretlenen ölçütler için ilgili konuları tekrar ediniz.				

2. ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyunuz ve doğru seçeneği işaretleyiniz.

1. I. İstatistiksel ve matematiksel yöntemleri içermektedir.
II. Tüm problem durumlarında en iyi sonucu verir.
III. Yapay öğrenme olarak da adlandırılır.
IV. Yapay zekânın alt alanıdır.

Makine öğrenmesi modelleri ile ilgili yukarıdaki numaralanmış ifadelerden hangileri kullanılır?

A) Yalnız III B) II ve IV C) I, III ve IV D) II, III ve IV E) I, II, III ve IV

2. Aşağıdakilerin hangisinde makine öğrenmesi türleri doğru olarak verilmiştir?

A) Denetimli, Denetimsiz, Güçsüz ve Güçlü
B) Denetimsiz, Yarı Denetimsiz ve Güçlendirilmiş
C) Basit Seviye, Orta Seviye ve Üst Seviye
D) Denetimli, Denetimsiz, Yarı Denetimli ve Pekiştirmeli
E) Yarı Denetimli, Denetimli, Denetimsiz ve Yarı Denetimsiz

3. I. Denetimli öğrenme, kestirim ve sınıflandırma için kullanılır.
II. Denetimli öğrenmede veri setinde etiket sütunu bulunmalıdır.
III. Denetimsiz öğrenmede veri setinde etiket sütunu gerekmez.
IV. Kümeleme denetimsiz bir makine öğrenmesi modelidir.
V. Regresyon denetimsiz bir makine öğrenmesi modelidir.

Makine öğrenmesi modelleri ile ilgili yukarıdaki numaralanmış ifadelerden hangileri doğrudur?

A) Yalnız I B) I ve II C) I ve III D) I, II ve III E) I, II, III ve IV

4. Hastalara ait değerlerin bulunduğu bir veri setini kullanarak verilen değerlerle insanların hasta olup olmadığını sınıflayan bir model geliştirmek için aşağıdaki makine öğrenmesi türlerinden hangisi kullanılır?

A) Denetimli B) Denetimsiz C) Karışık D) Pekiştirmeli E) Yarı denetimli

5. Makine öğrenmesi türlerine ilişkin aşağıdaki ifadelerden hangisi doğrudur?

A) Doğal dil işleme, denetimsiz öğrenmenin kullanıldığı alanlardan biridir.
B) Etiketli verinin az, etiketsiz verinin çok olduğu durumlarda denetimli öğrenme tercih edilir.
C) Pekiştirmeli öğrenme etiketli veri ve ortam hakkında yeterli bilgi olduğunda tercih edilir.
D) Sepet analizi, pazarlamada kullanılan denetimsiz makine öğrenmesi modellerinden biridir.
E) Yarı denetimli öğrenme pekiştirmeli ve denetimsiz öğrenmenin karışımıdır.

6. Makine öğrenmesinin temel kavramlarıyla ilgili aşağıdakilerden hangisi doğrudur?

A) Bağımlı değişken, çıktıya etki ettiği düşünülen özelliklerdir.
B) Denetimli öğrenmede bağımlı bir değişken bulunmaz.
C) Denetimsiz öğrenmede etiketli veri kullanılmaz.
D) Pekiştirmeli öğrenmede etiketsiz veri kullanılır.
E) Model, verinin algoritmaya uydurulması işlemidir.

7. **Makine öğrenmesi süreci ile ilgili aşağıdakilerden hangisi doğrudur?**

A) Modelin iyi çalışması için test kümesi eğitim kümesinden büyük olmalıdır.
B) Çapraz doğrulama yöntemi denetimli bir makine öğrenmesi türüdür.
C) Çapraz doğrulamada veri seti bölünerek tüm parçalar eğitim ve test için kullanılır.
D) Denetimli öğrenme modelinde geliştirilen model eğitim verisi kullanılarak test edilir.
E) Sınıflandırma ve regresyon performanslarını ölçmek için aynı metrikler kullanılır.

8.
I. T1 Skoru
II. Duyarlılık
III. Kesinlik
IV. Seçicilik

Yukarıdaki numaralanmış metriklerden hangileri sınıflandırma performansının ölçülmesi için kullanılır?

A) II ve III B) II ve IV C) I, II ve III D) I, II ve IV E) II, III ve IV

9.
I. MAPE
II. MAE
III. R2
IV. RMSE

Yukarıdaki numaralanmış metriklerden hangileri kestirim performansının ölçülmesi için kullanılır?

A) I, II, III B) I, III ve IV C) III ve IV D) II, III ve IV E) I, II, III ve IV

10. **Aşağıdakilerden hangisi Python kodlarını yazmak için kullanılan etkileşimli bir Python not defteridir?**

A) IDLE B) Jupyter C) Mars D) Neptun E) Spyder

11. **Aşağıdakilerden hangisi diziler ve cebirsel işlemler için kullanılan temel kütüphanedir?**

A) Matplotlib B) NumPy C) Pandas D) Python E) SciPy

12. **Aşağıdaki komutlardan hangisi not defterinde bir paket kurmak için kullanılmalıdır?**

A) install B) pip install C) setup lib D) !install E) !pip install

13.
```
import numpy as np
dizi2D = np.array([[1,0,1],[2,1,0]])
print ( dizi2D.sum(axis = 0))
```

Verilen kodun çıktısı aşağıdakilerden hangisidir?

A) 4 B) 5 C) 8 D) [1 1 3] E) [3 1 1]

14. **0-1 aralığında ondalıklı 2x3'lük bir dizi oluşturmak için aşağıdaki kod satırlarından hangisi kullanılmalıdır?**

A) np.random.rand(2,3)
B) np.random.rand(3,2)
C) np.random.random(2,3)
D) np.random.random(3,2)
E) np.rand.random(2,3)

15. **Aşağıdakilerden hangisi grafik çizmek için kullanılan kütüphanedir?**

A) Django B) matplotlib C) PyQt5 D) Requests E) Scarpy

16. Pandas ile ilgili aşağıda verilen bilgilerden hangisi doğrudur?

A) "head" ile veri setinin başındaki satırlar listelenir.
B) Bir seri hakkında bilgi almak için "size" metodu kullanılır.
C) "shape" bir serinin boyutunu görmek için kullanılır.
D) "ndim" bir serinin şeklini görmek için kullanılır.
E) "long" serinin uzunluğunu görmek için kullanılır.

17. Pandas kütüphanesiyle ilgili bir veri setinde eksik veri durumunu görmek için aşağıdakilerden hangisi kullanılmalıdır?

A) count B) describe C) info D) shape E) unique

18. Pandas kütüphanesiyle ilgili veri setindeki sütunlarla ilgili işlemler için aşağıdaki ifadelerden hangisi doğrudur?

A) "column_names" sütun adlarını verir.
B) Birden fazla sütun seçilebilir.
C) "iloc" ile sütun adları kullanılır.
D) "loc" ile indis numaraları kullanılır.
E) Sütun adları değiştirilemez.

19. I. "label encoding" kullanılabilir.
II. "one hot encoding" kullanılabilir.
III. İşlem yapılmazsa bazı algoritmalar kullanılmaz.
IV. Kukla değişkenler ortaya çıkar.

Veri setindeki kategorik verinin dönüştürülmesi ile ilgili verilen ifadelerden hangileri doğrudur?

A) Yalnız III B) I ve III C) I, II ve III D) I, III ve IV E) I, II, III ve IV

20. Seaborn kütüphanesinde ısı haritası için aşağıdaki metotlardan hangisi kullanılır?

A) catplot B) displot C) heatmap D) plt E) relplot

21. Pandas kütüphanesinde eksik veriyi çıkarmak için aşağıdaki metotlardan hangisi kullanılmalıdır?

A) dropna B) drop_nan C) miss_values D) resize E) shape

22. Scikit-learn kütüphanesinde eksik veri ile ilgili işlemleri yapmak için aşağıdaki modüllerden hangisi kullanılmalıdır?

A) DataNone B) Decoding C) Encoding D) MissingValues E) SimpleImputer

23. Seaborn kütüphanesiyle ilgili aşağıdakilerden hangisi doğrudur?

A) "hue" parametresi grafiğin büyüklüğünü belirler.
B) "height" parametresi grafiği kategorik olarak gruplar.
C) "palette" parametresi sütun adlarını gösterir.
D) "kind" parametresi grafik türünü belirler.
E) "label" parametresi grafiğin renk paletini belirler.

24.
```
from sklearn.preprocessing import normalize
```

Verilen kod ve içe aktarılan modül ile ilgili aşağıdaki ifadelerden hangisi doğrudur?

A) "normalize" modülü "sklearn" kütüphanesinin üstünde yer alır.
B) Makine öğrenmesi modelini belirlemek için kullanılır.
C) Veri setindeki değerleri 0-100 aralığındaki bir skalada yeni değerlere dönüştürür.
D) Veri setindeki değerlerin ortalamasını 0, standart sapmasını 1'e eşler.
E) Scikit-learn kütüphanesinden normalize modülünü içe aktarmak için kullanılır.

25. İkinci el araçlara ilişkin özellikleri ve fiyatlarını kullanarak ikinci el bir aracın fiyatını tahmin etmek için hangi makine öğrenmesi modeli kullanılmalıdır?

A) Denetimli öğrenme \ kestirim
B) Denetimli öğrenme \ sınıflandırma
C) Denetimli öğrenme \ kümeleme
D) Denetimsiz öğrenme \ kümeleme
E) Yarı denetimli öğrenme \ kestirim

26. Bir markette hangi ürünlerin birlikte satıldığını belirlemek ve müşteriler için bir öneri sistemi geliştirmek için hangi makine öğrenmesi modeli kullanılmalıdır?

A) Denetimli öğrenme \ market analizi
B) Denetimli öğrenme \ pazar analizi
C) Denetimsiz öğrenme \ birliktelik analizi
D) Denetimsiz öğrenme \ boyut indirgeme
E) Denetimsiz öğrenme \ kümeleme

27. Çapraz doğrulamada veri kümesinin kaç parçaya bölüneceğini belirtmek için aşağıdaki parametrelerden hangisine değer verilmelidir?

A) cv B) estimator C) fold D) fold_number E) score

28. Modelin bir veri için sınıflandırma veya kestirim yapmasını sağlamak için aşağıdaki metotlardan hangisi kullanılır?

A) class B) estimate C) predict D) regression E) test

29.
I. Değerlendirme
II. Veri toplama
III. Model oluşturma
IV. Veri ön işleme
V. Dağıtım

Numaralandırılarak verilen makine öğrenmesi sürecinin aşamaları aşağıdakilerin hangisinde doğru olarak sıralanmıştır?

A) I, II, III, V, IV B) II, I, IV, III, V C) II, I, IV, V, III D) II, IV, III, I, V E) IV, II, III, V, I

30. Makine öğrenmesi sürecinde iş hattı kullanmak için aşağıdaki modüllerden hangisi içe aktarılmalıdır?

A) automation B) blue_line C) line D) pipeline E) work_flow

31.
I. Destek Vektör Makineleri
II. Doğrusal Regresyon
III. K-NN
IV. Sıkıştırma Algoritmaları

Yukarıdakilerden hangileri makine öğrenmesi algoritmasıdır?

A) Yalnız II B) I ve II C) I, II ve III D)I, II ve IV E) I, II, III ve IV

32.
I. Denetimli bir makine öğrenmesi türüdür.
II. Üçten fazla sınıf olmalıdır.
III. K hiper parametresi en yakın komşu sayısıdır.
IV. Sınıflandırma yapmak için kullanılmaktadır.

K-NN ile ilgili yukarıdaki numaralanmış ifadelerden hangileri doğrudur?

A) III ve IV B) I, II ve III C) I, II ve IV D) I, III ve IV E) I, II, III ve IV

33. Karar ağaçları ile ilgili aşağıdaki ifadelerden hangisi doğrudur?

A) Dal sayısı modelin uyumunu etkiler.
B) Dal sayısı tek sayı olmalıdır.
C) Denetimsiz öğrenme türündedir.
D) Sadece kestirim yapmak için kullanılır.
E) Sadece sayısal veri türünde çalışır.

34. K-Means ile ilgili aşağıdakilerden hangisi doğrudur?

A) Algoritmada küme merkezleri sabittir.
B) Kümeleme amacıyla kullanılır.
C) Denetimli makine öğrenmesi türüdür.
D) Küme sayısı tek olmalıdır.
E) Hiper parametresi yoktur.

35. Aşağıdakilerden hangisi sadece sınıflandırma için kullanılan bir algoritmadır?

A) Destek vektör makineleri
B) Doğrusal regresyon
C) Karar ağaçları
D) K-NN
E) Rastgele orman

36. Aşağıdakilerden hangisi bir topluluk öğrenmesi modelidir?

A) Apriori
B) K-Means
C) Naive Bayes
D) Polinomsal regresyon
E) Rastgele orman

37. Aşağıdakilerden hangisi en çok kullanılan makine öğrenmesi algoritmalarından biri değildir?

A) Doğrusal Regresyon
B) Lojistik Regresyon
C) K-NN
D) Destek Vektör Makineleri (SVM)
E) Sıkıştırma Algoritmaları

38. Doğrusal Regresyon (Lineer Regression) ile ilgili aşağıdaki cümlelerden hangisi yanlıştır?

A) Regresyon tahminleri için kullanılan denetimli bir makine öğrenmesi algoritmasıdır.
B) Bağımsız değişkenleri kullanarak bir hedef tahmin değerini modeller.
C) Bağımlı değişken veya değişkenler (X) kullanılarak bağımsız değişkeni (y) tahmin etmek için kullanılır.
D) Değişkenler arası ilişki olmasa da kullanılabilir.
E) Basit doğrusal regresyon y=b0+b1x1 şeklinde formüle edilir.

YAPAY SİNİR AĞLARI

3. ÖĞRENME BİRİMİ

http://kitap.eba.gov.tr/KodSor.php?KOD=36138

KONULAR

3.1. YAPAY SİNİR AĞLARI TEMEL KAVRAM VE UYGULAMALARI
3.2. YAPAY SİNİR AĞLARI ÇEŞİTLERİ VE KATMANLARI

NELER ÖĞRENECEKSİNİZ?

- Yapay sinir ağlarının nasıl uygulandığını açıklama
- Yapay sinir ağları çeşitleri
- Farklı katmanlara sahip ağları açıklama
- Sinir ağı yapıları
- Görüntü işleme kütüphanelerini kullanma
- Görüntü işlemeyle ilgili uygulamalar

TEMEL KAVRAMLAR

- Yapay sinir ağları
- Derin öğrenme
- Python
- Colaboratory
- Nesne tanıma
- Sınıflandırma

HAZIRLIK ÇALIŞMASI

1. İnsan beynini taklit eden bir sistemin avantajları ve dezavantajları neler olabilir? Arkadaşlarınızla değerlendiriniz.

3.1. YAPAY SİNİR AĞLARI TEMEL KAVRAM VE UYGULAMALARI

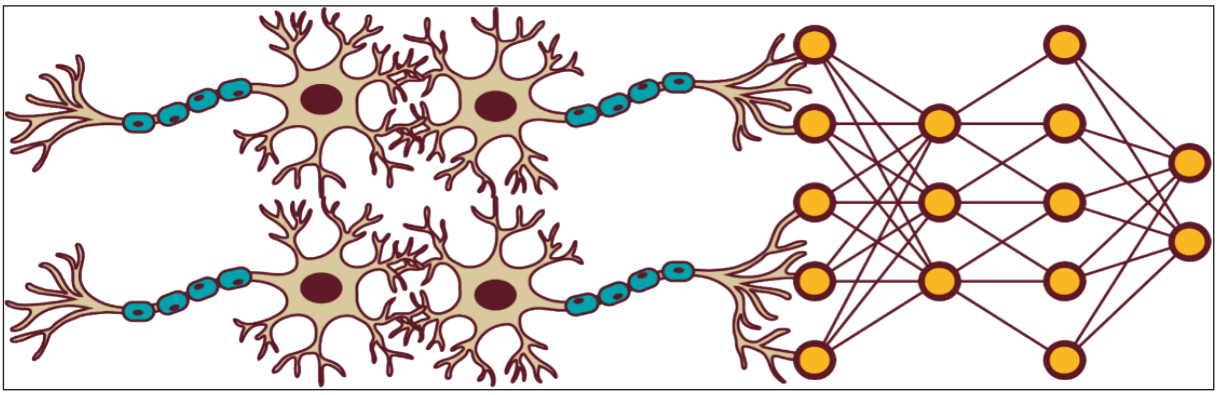
Yapay sinir ağları, insan beyninin biyolojik sinir ağları çalışma prensibinin matematiksel olarak modellenmesini hedef alan, algoritmalar ile kodlanmış, önceden öğrenilmiş ya da sınıflandırılmış veri setlerini kullanarak yeni bilgiler oluşturabilen, karar verebilen ve sınıflandıran yazılım uygulamalarıdır.

Yapay sinir ağları insan beynindeki biyolojik sinir ağlarının yapısını, öğrenme, hatırlama, genelleme ve karar verme kabiliyetlerini taklit eder. Yapay sinir ağlarında öğrenme örnekler ve veri setleri kullanılarak gerçekleştirilir.

3.1.1. Yapay Sinir Ağı

Yapay sinir ağlarının öğrenmesi sırasında giriş çıkış verileri verilerek matematiksel fonksiyonlar ile kurallar koyulur.

İlk yapay sinir ağı modeli 1943 yılında bir sinir hekimi olan Warren McCulloch ve bir matematikçi olan Walter Pitts tarafından "Sinir Aktivitesinde Düşüncelere Ait Bir Mantıksal Hesap (A Logical Calculus of Ideas Immanent in Nervous Activity)" başlıklı makale ile ortaya çıkarılmıştır (Görsel 3.1).



Görsel 3.1: Biyolojik sinir hücresi ve yapay sinir hücresi

3.1.2. Yapay Sinir Ağının Avantajları

- Öğrenme kabiliyeti vardır.
- Farklı öğrenme algoritmalarıyla öğrenebilir.
- Görülmemiş çıktılar için sonuçlar üretebilir.
- Algılamaya yönelik olaylarda kullanılabilir.
- Örüntü tanıma ve sınıflandırma yapabilir.
- Eksik örüntüleri tamamlayabilir.
- Kendi kendini öğrenebilme yetenekleri vardır.
- Hata toleransına sahiptir.

3.1.3. Yapay Sinir Ağının Dezavantajları

- Bir problem için geliştirilen yapay sinir ağı modelinin bir başka probleme uyarlanması zor olabilir.
- Örnek veri seti yoksa eğitim mümkün değildir. Örnek veri seti olsa da ağ mimarisi ve öğrenme kuralından öğrenme gerçekleştirilemeyebilir ya da öğrenmenin hata toleransı yüksek olabilir.
- Probleme uygun ağ yapısının belirlenmesi genellikle deneme yanılma yoluyla yapılır. Bu da zaman kaybına neden olur.
- Ağın eğitiminin ne zaman bitirileceğine karar vermek için kesin bir yöntem yoktur.
- Paralel işlem yapma yeteneğine sahip yüksek kaynak gerektiren donanımlar üzerinde çalıştırılmalıdır.

3.1.4. Biyolojik Sinir Ağı

İnsandaki biyolojik sinir ağı, nöron isimli bilgi işleme yeteneği olan sinir hücrelerinin bir araya gelmesinden oluşur. Nöronlar (sinir hücreleri) birbirleri ile bir araya gelip bağlanarak nörolojik fonksiyonları yerine getirir. Bilim insanları tarafından yapılan araştırmalar sonucu beyinde 100 milyar adet nöron olduğu tahmin edilmekte, bununla birlikte bir nöron başka nöronlarla 50.000-250.000 arasında bağlantı yapabilmekte ve beyinde 6x1013'ten fazla sayıda bağlantı bulunduğu tahmin edilmektedir.

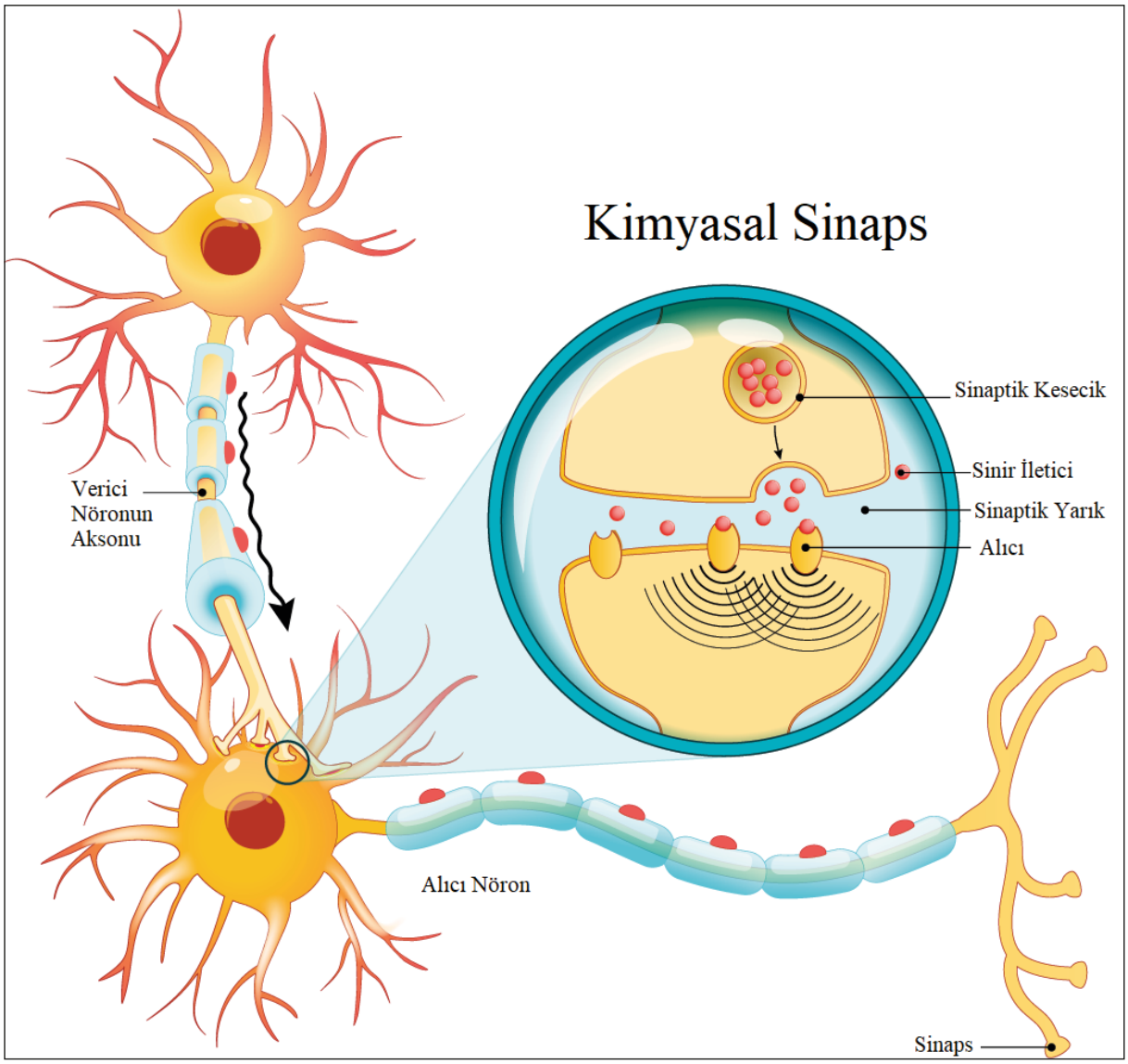
Biyolojik sinir ağları insanın bütün davranışlarını ve çevresini anlamasını sağlar. Biyolojik sinir ağları beş duyu organından gelen bilgilerle geliştirdiği algılama ve anlama mekanizmalarını kullanarak olaylar arasındaki ilişkileri öğrenir. İnsan beyninin değişik bölgeleri değişik fonksiyonları yerine getirir. Duyu organlarından gelen sinyallerle okunan bilgiler, sinir sistemi aracılığıyla beyne taşınır ve beynin oluşturduğu kararlar da yine sinir sistemi tarafından vücudun organlarına sinyallerle eylem olarak gönderilir (Görsel 3.2).

Nöronlar, işlevlerine göre 3 ana gruba ayrılır.

Duyusal Nöronlar: Dış ortamdan koku, tat, dokunma, görme ve ses vasıtasıyla aldıkları bilgiyi beyne iletir.

Motor Nöronlar: Kasların kasılmasını kontrol ederek hareketi sağlar.

Ara Nöronlar: Sinirsel devreleri oluşturarak duyusal nöronlar, motor nöronlar ve merkezî sinir sistemi arasındaki iletişimi sağlar.



Görsel 3.2: Neurotransmitter geçişi ile hücrelerin uyarılması

Bir sinir hücresinden diğer sinir hücresine sinyallerin sinaps üzerinden taşınması neurotransmitter isimli kimyasal maddelerle sağlanır. Alınan kimyasal madde aracılığıyla sinyal hücrenin içindeki elektrik potansiyel değerine (eşik) ulaşırsa akson içine bir işaret gönderir. Buna **hücrenin uyarılması** denir.

Sinir hücresinin işlevi, sinaps aracılığıyla dentrite ulaşan uyarıları alarak bunlardan aksiyon potansiyeli oluşturmak veya oluşturmamaktır. Aksonlar boyunca iletim yapılır. Daha sonra çıkış terminallerinde dentrit uçlarından elde edilen sensör verileri çekirdekte ağırlıklandırılarak akson boyunca iletilir ve başka sinir hücresine bağlanır. Bu şekilde sinirler arası iletişim sağlanır.

İnsan beyninde öğrenme 3 şekilde olur.

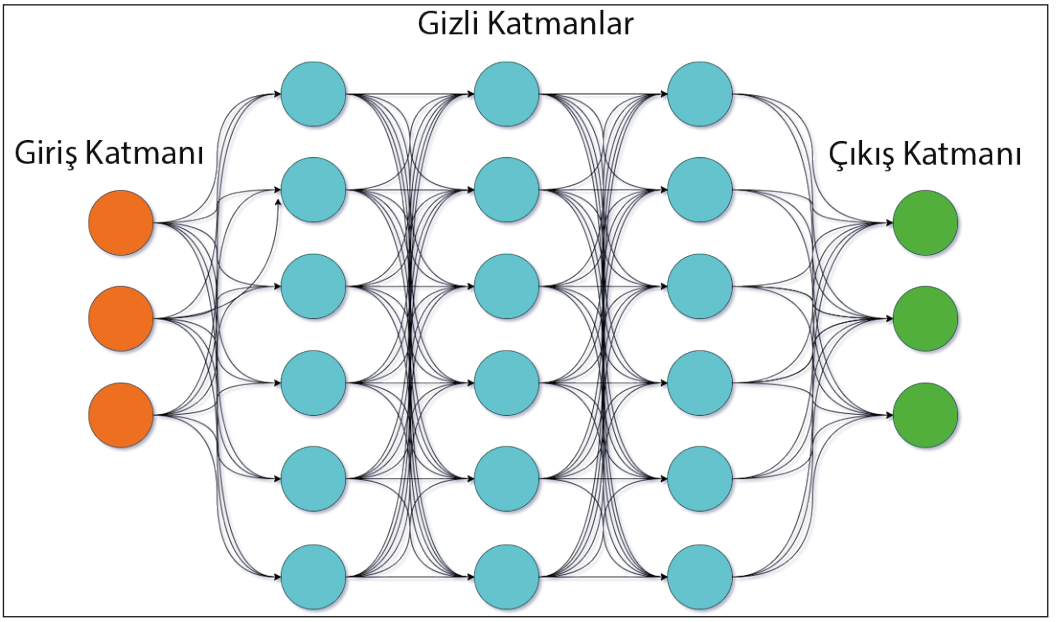
- Yeni aksonlar üreterek öğrenme
- Aksonların uyarılmasıyla öğrenme
- Mevcut aksonların güçlerini değiştirerek öğrenme

3.1.5. Yapay Sinir Ağının Yapısı

Yapay sinir ağları yapay sinir hücrelerinin birbirine bağlanmasıyla oluşan yapılardır (Görsel 3.3). Yapay sinir ağları üç katmanda incelenir (Görsel 3.3).

1. Giriş Katmanı
2. Ara (Gizli) Katmanlar
3. Çıkış Katmanı

Bilgiler ağa girdi katmanından iletilir. Bilgiler, ara katmanlarda işlenerek yani ağırlık değerleri elde edilerek çıktı katmanına gönderilir. Yapay sinir ağının girdiler ile doğru çıktıları üretebilmesi için ağırlıkların doğru değerlerinin olması gerekir.



Görsel 3.3: Yapay sinir ağı katmanları

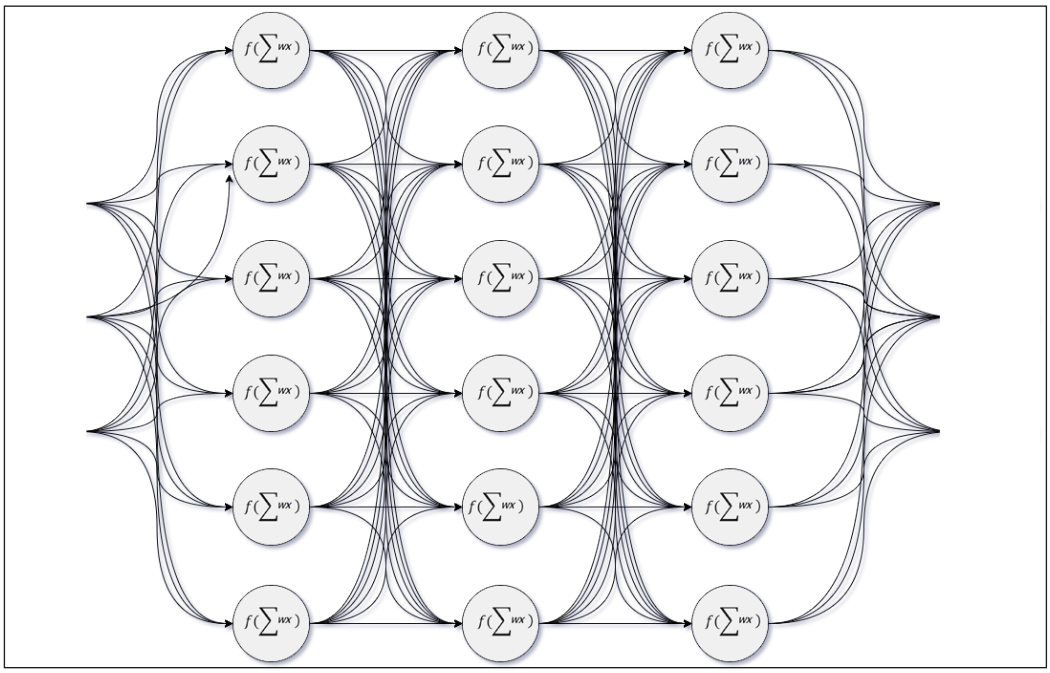
Sonuç olarak bu katmanlı yapıda giriş değerleri, ara katman ve çıkış değerleri aşağıdaki gibi elde edilir.

- Giriş değerleri olarak veri seti kullanılır.
- Yapay sinir ağı, giriş katmanından veri setini aldıktan sonra başlangıç için verilecek ağırlık değerlerine göre bir çıktı üretecektir ve çıktıların doğruluğuna göre uygun ağırlık değerlerini bulana kadar işlemleri tekrarlayacaktır. Ara (gizli) katman değerleri bu ağırlıklar ile belirlenir.
- Kabul edilebilir dereceye ulaşan çıktılar elde edildiğinde eğitim durur ve çıktılar oluşturulur.

Yapay sinir ağlarındaki ağırlıkların belirlenmesi ve çıktıların oluşturulması belirli bir matematiksel model ile elde edilir.

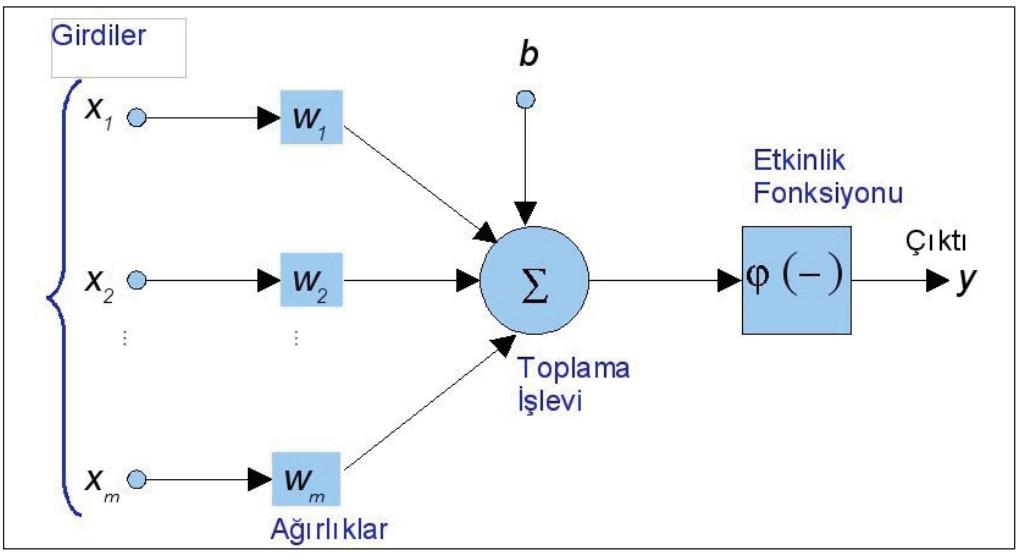
3.1.6. Sinir Hücresinin Matematiksel Modeli

İnsandaki bir sinir hücresinin matematiksel modeli aşağıdaki şekilde gösterilebilir. Burada yapay sinir ağının esas görevi, girdi veri seti olarak kendisine gelen verilere karşılık Görsel 3.3'te verildiği gibi bir çıktı üretebilmektir. Bu işlemin yapılabilmesi için ağ daha önceden belirlenmiş eğitim veri setleriyle eğitilir. Bu aşamadan sonra yapay sinir ağı genelleme yapabilecek ve karar verebilecek seviyeye kavuşur. En sonunda da çıktıları üretir.



Görsel 3.4: Sinir hücrelerinin matematiksel modeli

Görsel 3.4'teki her bir matematik fonksiyonun çalıştığı nöron Görsel 3.5'te gösterildiği gibi çalışır.



Görsel 3.5: Bir sinir hücresinin matematiksel modeli

3.1.7. Matematiksel Model Hesap İşlemi

Dentrit ismi verilen ara katman yolları boyunca ağırlıklar mevcuttur ve bu dentritlere giren bir başka nörondan da gelmiş olabilecek bir giriş değeri (x_1, x_2, x_3,… x_m) vardır. Bu parametrelerle hesaplama adım adım yapılır.

1. Giriş değeri ve dentritteki ağırlık(w_1) çarpıldıktan sonra(w_1x_1) sinir hücresine iletilir.
2. Tüm dentritlerden gelen ağırlık ile giriş çarpımları toplanır. Yani ağırlıklı toplama işlemi yapılır.
3. Ardından bir bias(b) ile toplandıktan sonra aktivasyon fonksiyonu ardından çıkışa aktarılır.
4. Bu çıkış nihai çıkış olabileceği gibi bir başka hücrenin girişi olabilir.
5. Her bir sinir hücresi aynı şekilde hesaplanır ve bunlar birbirine seri ya da paralel şekilde bağlanır.
6. Matematiksel olarak ağırlıklar ile girişler çarpılır ve artı bir bias eklenir. Böylelikle basit bir matematiksel model elde edilir.

Yapay sinir ağları matematiksel modelde ağırlıkları ve çıkışları bulmak için yapılan en önemli işlem; modelin en iyi derecede vereceği **w** (ağırlık parametresi) ve **b** (bias değeri) parametrelerinin hesabını yapmaktır.

3.1.8. Yapay Sinir Hücresi Bölümleri

Bir yapay sinir hücresi beş bölümden oluşur.

3.1.8.1. Girdiler

Girdiler, nöronlara gelen verilerdir. Bu girdilerden gelen veriler biyolojik sinir hücrelerinde olduğu gibi toplanmak üzere nöron çekirdeğine gönderilir.

3.1.8.2. Ağırlıklar

Yapay sinir hücresine gelen bilgiler, girdiler üzerinden çekirdeğe ulaşmadan önce geldikleri bağlantıların ağırlığıyla çarpılarak çekirdeğe iletilir. Bu sayede girdilerin üretilecek çıktı üzerindeki etkisi ayarlanabilir.

3.1.8.3. Toplama Fonksiyonu (Birleştirme Fonksiyonu)

Toplama fonksiyonu, bir yapay sinir hücresine ağırlıklarla çarpılarak gelen girdileri toplayarak o hücrenin net girdisini hesaplayan fonksiyondur.

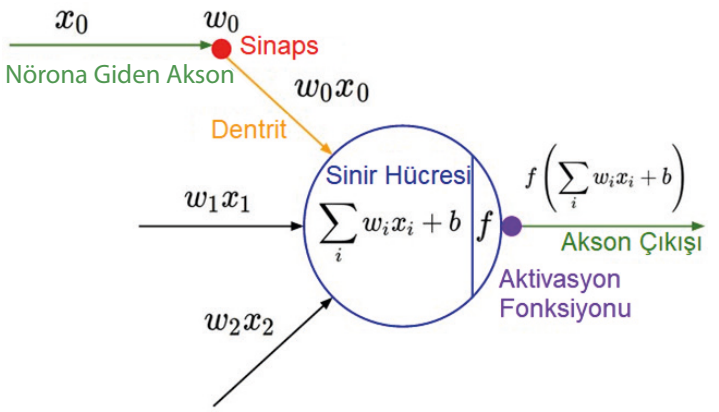
3.1.8.4. Aktivasyon Fonksiyonları

Önceki katmandaki tüm girdilerin ağırlıklı toplamını alan ve daha sonra bir çıkış değeri (tipik olarak doğrusal olmayan) üreten ve bir sonraki katmana geçiren fonksiyondur. Yapay sinir ağlarına komplex verilerin öğretilebilmesi için aktivasyon fonksiyonları gereklidir. Aktivasyon fonksiyonlarının amacı weight (ağırlık) ve bias değerlerini ayarlamaktır.

L = x.w + b işlemi yapıldıktan sonra bu değer bir aktivasyon fonksiyonundan geçirilir. Bu fonksiyonlar tensorlere etki eder ve aslında bu nöronların tetiklenmesi olarak da adlandırılabilir.

3.1.8.5. Aktivasyon Fonksiyonuna Neden İhtiyaç Duyulur?

Yapay sinir ağlarına doğrusal olmayan gerçek dünya özelliklerini tanıtmak için aktivasyon fonksiyonuna ihtiyaç duyulur. En temel anlamda basit bir yapay sinir ağında x girdiler, w ağırlıklar olarak tanımlanır ve ağın çıkışına aktarılan sonuca **f** (x) yani aktivasyon işlemi uygulanır. Daha sonra bu sonuç çıkış ya da bir başka katmanın girişi olur (Görsel 3.6).



Görsel 3.6: Aktivasyon fonksiyonu

Aktivasyon fonksiyonu uygulanmazsa çıkış sinyali basit bir doğrusal fonksiyon olur. Aktivasyon fonksiyonu kullanılmayan bir yapay sinir ağı, sınırlı öğrenme gücüne sahip olacaktır. Sinir ağının doğrusal olmayan durumları da öğrenmesi en önemli amaç olduğuna göre sinir ağına öğrenmesi için görüntü, video, yazı ve ses gibi karmaşık gerçek dünya bilgileri de verilir. Çok katmanlı derin yapay sinir ağları bu sayede verilerden anlamlı özellikleri öğrenebilir.

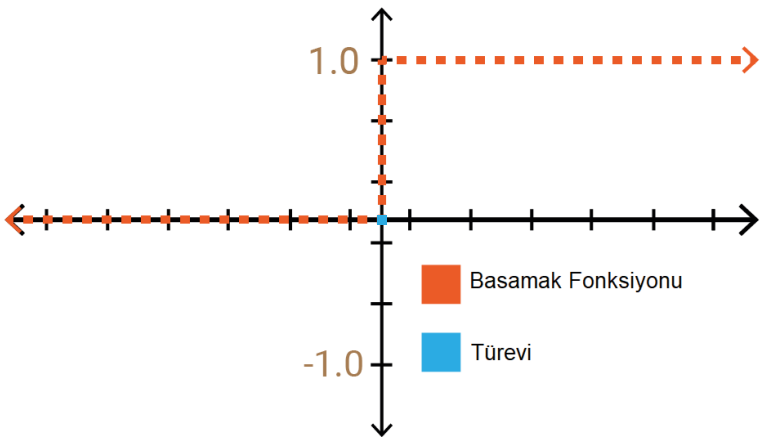
Ağırlıklar ile ilgili hata değerlerini hesaplamak için yapay sinir ağında hatanın geriye yayılımı algoritması uygulanır. Aktivasyon fonksiyonunu uygulamak için giriş değerleri ile ağırlıklar çarpılır, bias ile toplanır ve aktivasyon fonksiyonu uygulanır.

3.1.8.6. Aktivasyon Fonksiyonu Çeşitleri

Bu bölümde yapay sinir ağlarında kullanılan aktivasyon fonksiyonlarına yer verilmiştir.

- **Basamak (Step) Fonksiyonu**

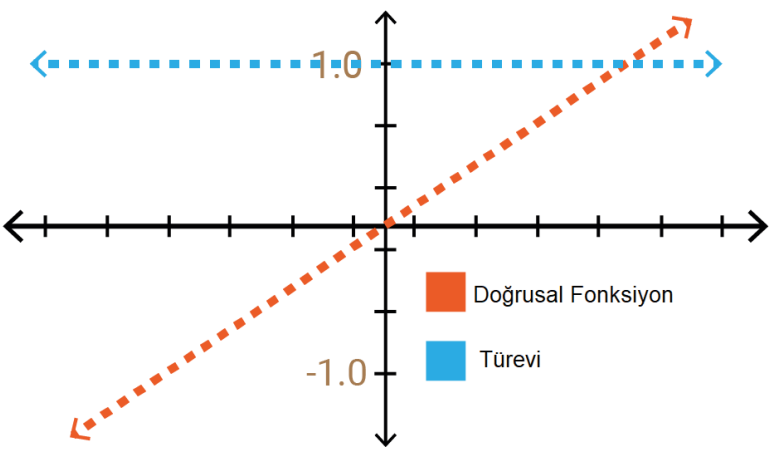
İkili değer alan bir fonksiyondur ve sonuç olarak ikili sınıflayıcı olarak kullanılır (Görsel 3.7). Bu sebeple çıkış katmanlarında sınıflandırmak için tercih edilir. Gizli katmanlarda basamak fonksiyonun türevi öğrenme değeri olmadığı için kullanılması tavsiye edilmez.



Görsel 3.7: Basamak fonksiyonu ve türevi

- **Doğrusal (Linear) Fonksiyon**

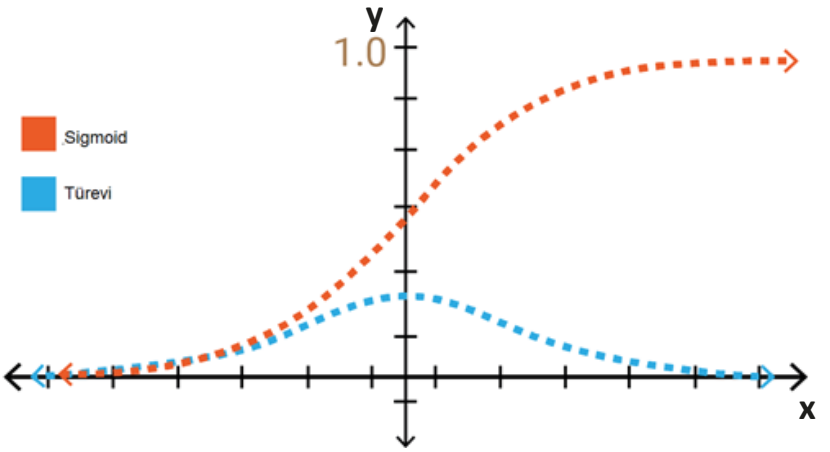
Bir dizi aktivasyon değeri üretir ve bunlar basamak fonksiyonundaki gibi ikili değerler değildir (Görsel 3.8). Kesinlikle bir kaç sinir hücresini birbirine bağlamaya izin verir. Bu fonksiyonun türevinin sabit olması önemli bir sorundur. Çünkü türevi hep sabit bir değer çıkıyorsa öğrenme işlemi gerçekleşmez. Ayrıca tüm katmanlarda doğrusal fonksiyon kullanıldığında giriş katmanı ile çıkış katmanı arasında hep aynı doğrusal sonuca ulaşılır ve nöronlar yani ara katmanlar işlevsiz kalır. Böylelikle en önemli amaç olan çok katmanda çalışma özelliği kaybedilir.



Görsel 3.8: Doğrusal fonksiyon ve türevi

- **Sigmoid Fonksiyonu**

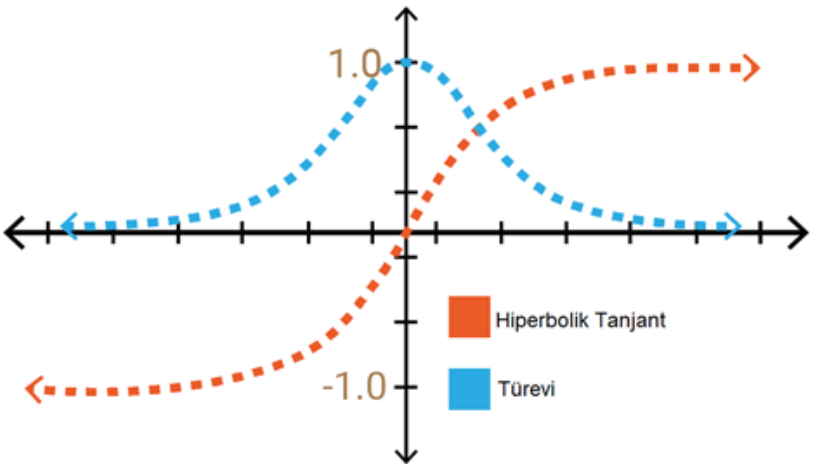
Eskiden çok kullanılan bu fonksiyon aldığı değerleri 0 ve 1 arasında tutar ve bu fonksiyon doğrusal değildir (Görsel 3.9). Yüksek bir değer geldiğinde 1'e yakın olurken, düşük bir değer geldiğinde 0'a yakın olur. Bu durumda 1'i ve 0'ı geçen bir değer olmaz. Türevlenebilir bir fonksiyon olması sebebiyle öğrenme olayı bu fonksiyon kullanıldığında gerçekleşebilir. Grafik incelendiğinde x, -2 ile +2 arasında iken, y değerleri hızlı şekilde değişir. x'te yapılan küçük değişimler y için büyük olacaktır. Bu durum iyi bir sınıflandırıcı olarak kullanılabileceği anlamına gelir. Bu fonksiyonun bir diğer güzel yanı da doğrusal fonksiyonda olduğu gibi (-sonsuz, +sonsuz) ile karşılaşıldığında her zaman (0,1) aralığında değer üretmesidir. Yani aktivasyon değeri kullanılabilir.



Görsel 3.9: Sigmoid fonksiyonu türevi

- **tanh (Hiperbolik Tanjant) Fonksiyonu**

Hiperbolik tanjant "tanh"n sigmoid'e çok benzer. İkisinde de sıkıştırma bulunur fakat tanh oluşan değerleri [-1,1] arasında tutar (Görsel 3.10). Hiperbolik tanjant fonksiyonunda Sigmoid fonksiyonuna göre daha iyi sonuçlar elde edilebilir çünkü 0 odaklıdır. Sigmoid fonksiyonuna çok benzer bir yapıya sahiptir. Ancak fonksiyonun aralığı bu kez (-1,+1) olarak tanımlanır. Sigmoid fonksiyonuna göre avantajı ise türevinin daha dik olması yani daha çok değer alabilmesidir. Bu daha hızlı öğrenme ve sınıflama işlemi için daha geniş aralığa sahip olmasından daha verimli olacağı anlamına gelir. Ama yine fonksiyonun uçlarında gradyanların ölmesi problemi devam etmektedir.

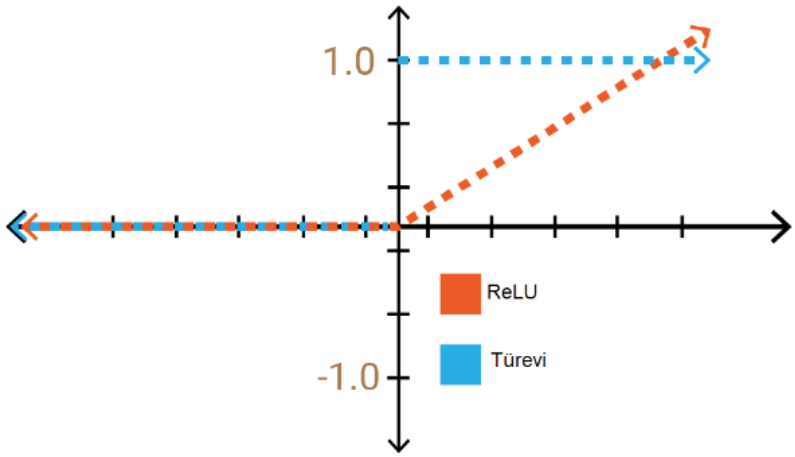


Görsel 3.10: Hiperbolik tanjant fonksiyonu türevi

- **ReLu Fonksiyonu**

En yaygın kullanılan aktivasyon fonksiyonudur (Görsel 3.11). ReLu fonksiyonunun en önemli özelliği biyolojik nörona benzerliğidir. Gelen değerlerin pozitif mi negatif mi olduğuna baktıktan sonra gelen değer negatif ise işlem sonucunu 0 verir ancak gelen değer pozitifse sıkıştırma ya da değiştirme işlemi uygulamaz, olduğu gibi geçer. Bilgisayarlar tanh ve sigmoid fonksiyonlarında karışık hesaplamalar yaparken ReLu fonksiyonunda

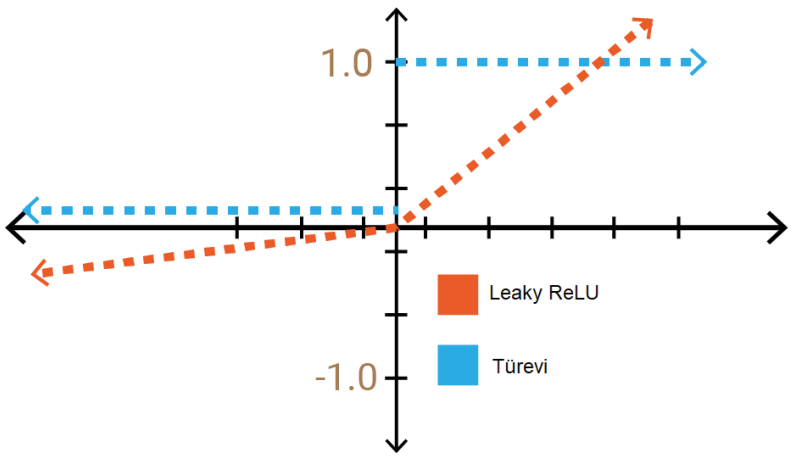
yalnızca pozitiflik negatiflik durumuna bakar. Bu sebeple bilgisayar bu denklemi çok daha hızlı hesaplar. ReLu fonksiyonu iyi bir tahmin edicidir ama aynı zamanda nöron ölümü de vardır. ReLu fonksiyonunun kombinasyonları ile herhangi başka bir fonksiyona da yakınsamak mümkündür. Bu durumda "Çok katmanlı yapıda kullanılabilir." denilebilir.



Görsel 3.11: ReLu fonksiyonu ve türevi

- **Sızıntı (Leaky ReLu) Fonksiyonu**

Leaky ReLu fonksiyonu ölü nöronları ortadan kaldırmak için geliştirilmiş bir fonksiyondur (Görsel 3.12). Negatif bir değer geldiğinde çok küçük bir sayı döndürerek nöronların ölmesinin önüne geçer. Ayrıca hesaplama sayısının artmasına rağmen diğer fonksiyonlara göre oldukça hızlı çalışır.



Görsel 3.12: Leaky ReLu fonksiyonu ve türevi

- **Softmax Fonksiyonu**

Sigmoid fonksiyonuna çok benzer. Sigmoid fonksiyonunda olduğu gibi sınıflayıcı olarak kullanıldığında oldukça iyi bir performans sergiler. En önemli farkı sigmoid fonksiyonu gibi ikiden fazla sınıflama yapılması gereken durumlarda özellikle derin öğrenme modellerinin çıkış katmanında tercih edilmektedir. 0-1 aralığında değerler üreterek girdinin belirli sınıfa ait olma olasılığının belirlenmesini sağlar. Yani olasılıksal bir yorumlama gerçekleştirir.

3.1.8.7. Hangi Aktivasyon Fonksiyonu Kullanılmalıdır?

Sıklıkla kullanılan sigmoid fonksiyonunun yerini artık ReLu fonksiyonu almıştır. Çok sınıflayıcı fonksiyonlar olarak geniş aralıkta aktive olması dolayısıyla hiperbolik tanjant kullanılabilir ya da model biraz daha yavaş öğrenebilmeye uygun ise sigmoid fonksiyonu kullanılabilir. Sinir ağı çok derinse ve işlem hacmi de bir o kadar önemli ise ReLu fonksiyonu tercih edilebilir. ReLu fonksiyonundaki gradyanların ölmesi sorununa çözüm olarak Leaky ReLu fonksiyonu kullanılmaya karar verilebilir.

Aktivasyon fonksiyonu seçerken aşağıda belirtilen ölçütlere dikkat edilmelidir (Görsel 3.13).

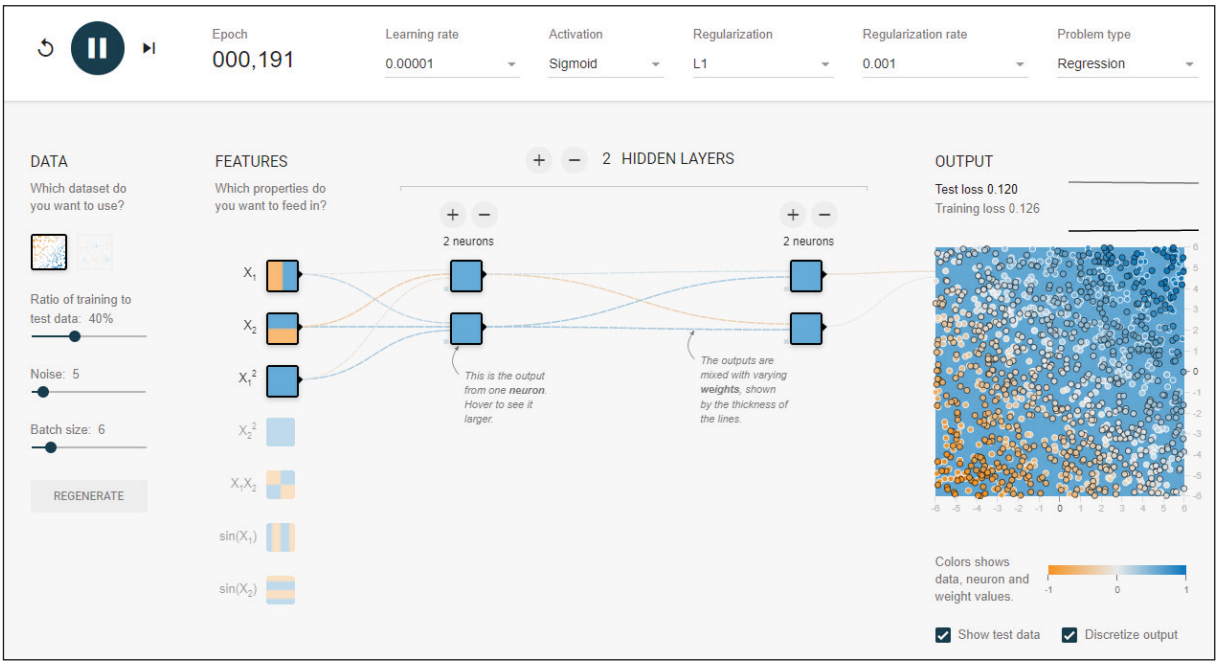
- Ağın kolay ve hızlı yakınsaması ilk ölçüt olabilir.
- ReLu hız bakımından avantajlıdır. Gradyanların ölmesi göze alınmalıdır. Genellikle çıkış değil ara katmanlarda kullanılır.
- Gradyanların ölmesi problemine ilk çözüm Leaky ReLu fonksiyonu olabilir.
- Derin öğrenme modelleri için ReLu fonksiyonu ile denemelere başlanabilir.
- Çıkış katmanlarında genellikle Softmax kullanılır.

En önemli sonuç, kendi eğiteceğiniz ağa göre uygun olan aktivasyon fonksiyonunun deneyerek seçilmesidir.

AKTİVASYON FONKSİYON	DENKLEM	ARALIK
Doğrusal Fonksiyon	$f(x) = x$	$(-\infty, \infty)$
Basamak Fonksiyonu	$f(x) = \begin{cases} 0 \; için \; x < 0 \\ 1 \; için \; x \geq 0 \end{cases}$	$\{0, 1\}$
Sigmoid Fonksiyon	$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$	$(0, 1)$
Hiperbolik Tanjant Fonksiyonu	$f(x) = \tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$	$(-1, 1)$
ReLU	$f(x) = \begin{cases} 0 \; için \; x < 0 \\ x \; için \; x \geq 0 \end{cases}$	$[0, \infty)$
Leaky (Sızıntı) ReLU	$f(x) = \begin{cases} 0.01 \; için \; x < 0 \\ x \; için \; x \geq 0 \end{cases}$	$(-\infty, \infty)$
Swish Fonksiyonu	$f(x) = 2x\sigma(\beta x) = \begin{cases} \beta = 0 \; için \; f(x) = x \\ \beta \to \infty \; için \; f(x) = 2\max(0, x) \end{cases}$	$(-\infty, \infty)$

Görsel 3.13: Aktivasyon fonksiyonlarının matematiksel ifadeleri

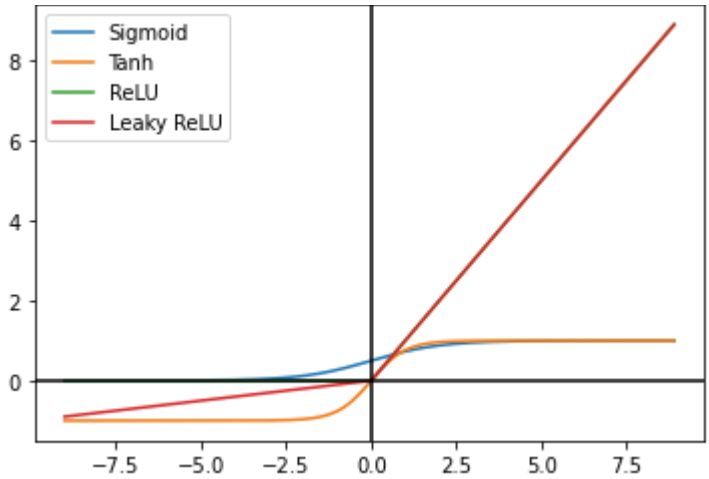
https://playground.tensorflow.org/ sitesi kullanılarak canlı grafikler, fonksiyonlar ve amaç seçilerek uygun olan sonuç kullanıcı için gözlemlenebilir (Görsel 3.14).



Görsel 3.14: Tarayıcıda yapay sinir ağı eğitimi

3.1.8.8. Aktivasyon Fonksiyonlarının Grafiklerinin Colab'ta Çizdirilmesi

Aşağıdaki kod bloku Colab'ta çalıştırılınca elde edilecek aktivasyon fonksiyonu grafikleri incelenebilir (Görsel 3.15).



Görsel 3.15: Aktivasyon fonksiyonları grafikleri

```
# Hesaplama için gerekli kütüphanelerin import edilmesi.
import math
import matplotlib.pyplot as plt
import numpy as np

# sigmoid, tanh, re, lr fonksiyonlarının tanımlamalarının matematik-
# sel olarak yazılması.

def sigmoid(x):
    a = []
    for i in x:
        a.append(1/(1+math.exp(-i)))
    return a

def tanh(x, derivative=False):
    if (derivative == True):
        return (1 - (x ** 2))
    return np.tanh(x)

def re(x):
    b = []
    for i in x:
        if i<0:
            b.append(0)
        else:
            b.append(i)
    return b

def lr(x):
    b = []
    for i in x:
        if i<0:
            b.append(i/10)
        else:
            b.append(i)
    return b
```

```
# Aktivasyon fonksiyonları grafikleri için oluşturulacak aralıkların
# belirlenmesi.

x = np.arange(-9., 9., 0.1)
sig = sigmoid(x)
tanh = tanh(x)
relu = re(x)
leaky_relu = lr(x)

#Fonksiyonların ekrana çizilmesi ve gösterilmesi.
line_1, = plt.plot(x,sig, label='Sigmoid')
line_2, = plt.plot(x,tanh, label='Tanh')
line_3, = plt.plot(x,relu, label='ReLU')
line_4, = plt.plot(x,leaky_relu, label='Leaky ReLU')

plt.legend(handles=[line_1, line_2, line_3, line_4])
plt.axhline(y=0, color='k')
plt.axvline(x=0, color='k')
plt.show()
```

3.1.8.9. Çıktılar

Aktivasyon fonksiyonundan çıkan değer hücrenin çıktı değeri olur. Her hücrenin birden fazla girdisi olmasına rağmen bir tek çıktısı olur. Bu çıktı istenilen sayıda hücreye bağlanabilir.

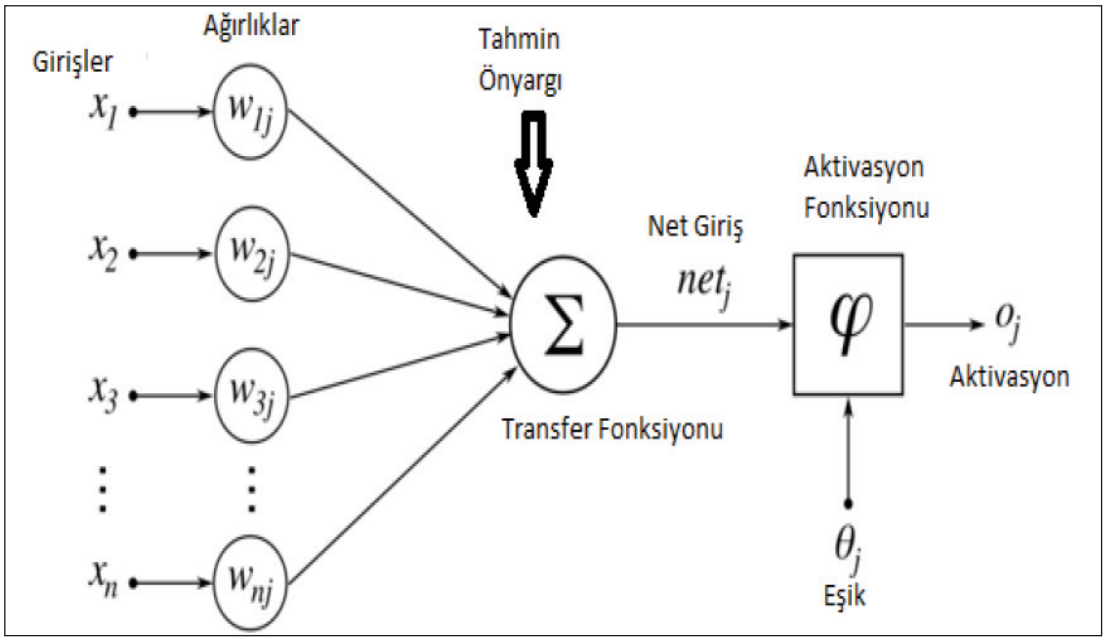
3.1.9. Biyolojik Sinir Sistemi Elemanları ve Yapay Sinir Sisteminde Karşılıkları

Biyolojik sinir hücresi ve yapay sinir ağı benzetimleri Görsel 3.16'da verilmiştir. Biyolojik sinir sistemi elemanları ve yapay sinir sisteminde karşılıkları ise Tablo 3.1'de verilmiştir. Burada biyolojik sinir sistemi kısımlara ayrılmış ve her bir elemana karşılık yapay sinir ağı sisteminde bir karşılığı verilmiştir.

Tablo 3.1: Biyolojik Sinir Sistemi Elemanları ve Yapay Sinir Sisteminde Karşılıkları

Biyolojik Sinir Sistemi	Yapay Sinir Sistemi
Nöron	İşlemci Elemanı
Dentrit	Toplama Fonksiyonu
Hücre Gövdesi	Transfer Fonksiyonu
Aksonlar	Yapay Nöron Çıkışı
Sinapslar	Ağırlıklar

Görsel 3.16'da görüldüğü gibi bir hücreye n tane veri girişi yapılmaktadır (Xn veri girişi). Girilen veriler ağırlıklarla çarpılarak tüm veriler toplanır ve sonra bias (önyargı) eklenir. Bunun sonucunda kesin sonuç elde edilir. Net girdi aktivasyon fonksiyonundan geçirilir ve bir veri çıktısı elde edilir.



Görsel 3.16: Çıkışın elde edilmesi

3.1.10. Yapay Sinir Ağları Kullanım Alanları

Yapay sinir ağları uygulamaları genelde sınıflandırma, tahmin, ilişkilendirme, veri yorumlama ve veri filtreleme işlemlerinde kullanılır.

3.1.10.1. Tahmin

Bu prensipte çalışan yapay sinir ağları girdi değerinden çıktıları tahmin etme üzerine çalışır. Altın ons fiyatının tahmini buna örnek olarak verilebilir.

3.1.10.2. Veri Filtreleme

Veri filtreleme algoritmasıyla kodlanan yapay sinir ağları, toplanan veriler arasından en işe yarayan verileri kullanır.

3.1.10.3. Sınıflandırma

Girdi değerlerini sınıflandırarak sistemin daha hızlı sonuca ulaşmasına etkide bulunan yapay sinir ağlarında kullanılır.

3.1.10.4. Veri Yorumlama

Önceden eğitilen yapay sinir ağı verilerini analiz eder. Bir durum hakkında bu girdiler sayesinde yeni yorumlamalar yapılabilir.

3.1.10.5. Veri İlişkilendirme

Öğrendiği bilgilerle konuları ilişkilendirir ve bunun sonucunda ortaya çıkan eksik bilgileri tamamlar.

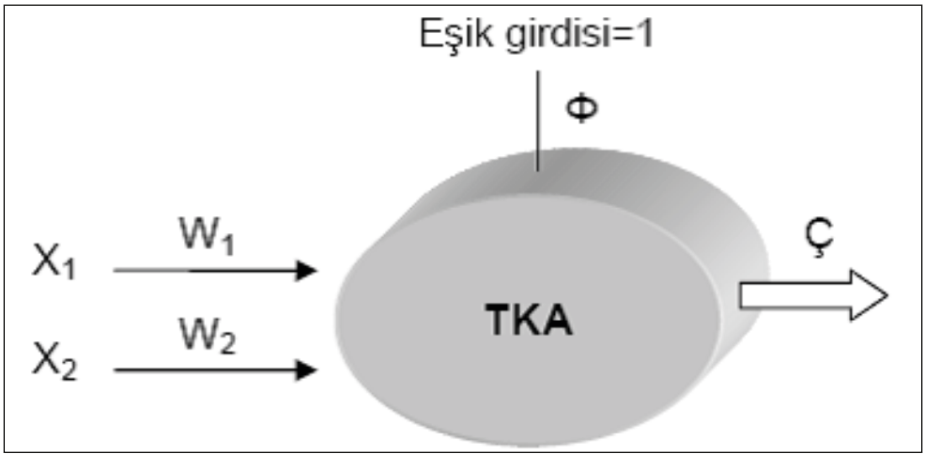
3.2. YAPAY SİNİR AĞLARI ÇEŞİTLERİ VE KATMANLARI

Yapay sinir ağları yapılarına göre dört farklı gruba ayrılır. Bu gruplar aşağıda sıralanmıştır.

- Tek Katmanlı Algılayıcılar
- Çok Katmanlı Algılayıcılar
- İleri Beslemeli Yapay Sinir Ağları
- Geri Beslemeli Yapay Sinir Ağları

3.2.1. Tek Katmanlı Algılayıcılar

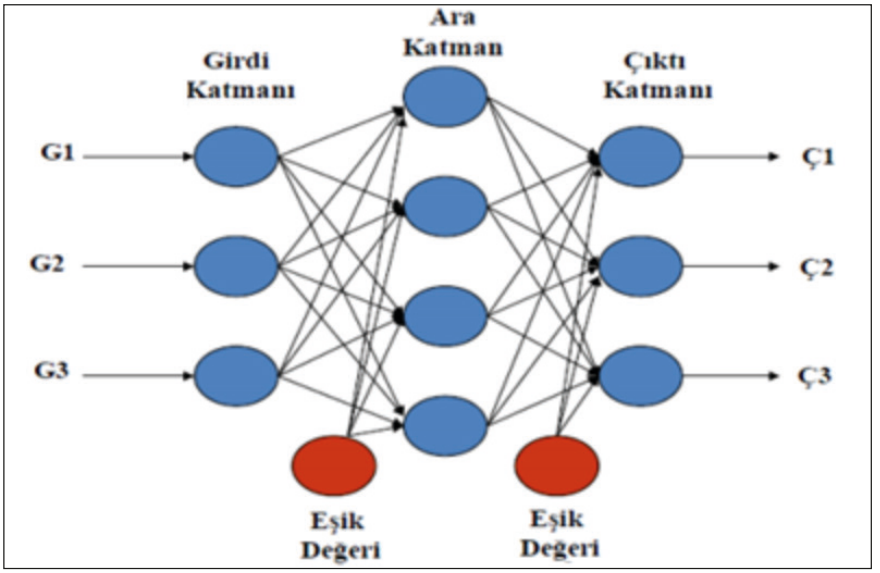
Tek katmanlı yapay sinir ağları sadece girdi ve çıktıdan oluşur. Tek katmanlı algılayıcılarda çıktı fonksiyonu doğrusaldır ve 1 veya -1 değerlerini alır. Eğer çıktı 1 ise birinci sınıf, -1 ise ikinci sınıf olur (Görsel 3.17).



Görsel 3.17: Tek katmanlı algılayıcı modeli

3.2.2. Çok Katmanlı Algılayıcılar

Doğrusal olmayan aktivasyon fonksiyonu kullanan birçok nöronun belli bir seviye ile bağlandığı yapıya **çok katmanlı algılayıcılar** denir. Çok katmanlı algılayıcılar diğer yöntemler fayda sağlamadığı için ortaya çıkmıştır (Görsel 3.18).



Görsel 3.18: Çok katmanlı algılayıcı modeli

3.2.3. İleri Beslemeli Yapay Sinir Ağları

İleri beslemeli yapay sinir ağlarının yapısında nöronlar, girişten çıkışa doğru tek yönlü katmanlar şeklinde dizilmiştir ve aynı katmanda bulunan nöronlar arasında bağlantı bulunmaz. Nöronlar arasında sadece kendinden sonraki ya da önceki katmanlara bağ bulunur. Yapay sinir ağının girişine gelen bilgiler bir değişime uğratılmadan ara katmana diğer adıyla gizli katmandaki hücrelere iletilir. Ardından çıkış katmanından işlenerek geçer ve çıkışa aktarılır. Yani tüm süreç girişten çıkışa doğru gerçekleşir ve sonlanır.

3.2.4. Geri Beslemeli Yapay Sinir Ağları

Geri beslemeli yapay sinir ağlarında çıkış ve ara katmanlardaki çıkışların, giriş birimlerine veya bir önceki ara katmanlara geri beslendiği bir yapay sinir ağı yapısıdır. Geri beslemeli yapay sinir ağlarında ileri beslemeli

ağlardan farklı olarak bir nöronun çıktısı sadece kendinden sonra gelen nöron katmanına girdi olarak verilmez. Kendinden önceki katmanda veya kendi katmanında bulunan herhangi bir nörona girdi olarak bağlanabilir. Böylece, girişler hem ileri yönde hem de geri yönde aktarılır. Bu yapısı ile geri beslemeli yapay sinir ağları doğrusal olmayan dinamik bir hareket oluşturur. Geri besleme özelliğini kazandıran bağlantıların bağlanış şekline göre; aynı yapay sinir ağıyla farklı davranışta ve yapıda geri beslemeli yapay sinir ağları elde edilebilir. Geri besleme aynı anda girişleri de etkilediği için özellikle tahmin uygulamaları için uygulanabilir.

3.2.5. Derin Öğrenme Nedir?

Derin öğrenme, yapay zekâ ve makine öğrenmesi çalışma alanlarının oldukça önemli bir dalı hâline gelmiştir. Özellikle son yıllarda, farklı derin öğrenme yöntemleri öneren çalışmaların ve mevcut yöntemleri değişik problemler üzerinde uygulayan çalışmaların sayıları hızla artar.

Derin öğrenmenin makine öğrenmesinden en önemli farkı öğrenme aşamasında çok katmanlı yapay sinir ağlarının kullanılmasıdır. Derin öğrenmeden daha verimli sonuçlar elde edilmesinin sebebi de çok katmanlı yapay sinir ağları teknikleridir. Derin öğrenme metotları, genel anlamda yapay sinir ağları (YSA-artificial neural networks) çalışmaları üzerine geliştirilmiştir. Yapay sinir ağları çalışmalarından farklı olarak daha fazla sayıda gizli nöron ve katman üzerine kuruludur.

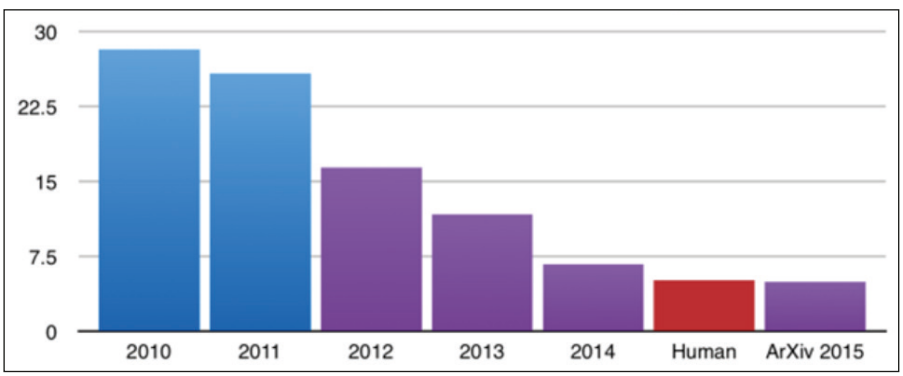
Başta gelen derin öğrenme uygulamaları aşağıda listelenmiştir.

1. Dil modelleme ve doğal dil işleme
2. Konuşma ve ses işleme
3. Bilgi erişimi
4. Nesne tanıma ve bilgisayarlı görü
5. Çok modlu ve çok görevli öğrenme

3.2.6. Derin Öğrenmenin Diğer Yöntemlerden Farkları

Alışılagelmiş makine öğrenme yöntemleri ile bir model tanımlama veya makine öğrenimi sistemi kurmak için öncelikle özellik vektörünün yani; gerçek dünyada ele alınamayacak kadar karışık olabilen verilerin makinelerin anlayabileceği şekilde basitleştirilmiş bir hâlinin çıkarılması gerekir. Özellik vektörünün çıkarılması için alanında uzman kişilere ihtiyaç duyulur. Bu işlemler hem çok zaman alır hem de uzmanı çok meşgul eder. Bu sebeple bu teknikler, ham bir veriyi ön işlem yapmadan ve uzman yardımı olmadan işleyemez. Derin öğrenme yöntemi, makine öğrenimi alanında çalışanların uzun yıllar boyunca uğraştığı bu sorunu ortadan kaldırarak büyük ilerleme sağlamıştır. Çünkü derin ağlar geleneksel makine öğrenmesi ve görüntü işleme tekniklerinin aksine öğrenme işlemini ham veri üzerinde yapar. Ham veriyi işlerken gerekli bilgiyi farklı katmanlarda oluşturmuş olduğu temsillerle elde eder.

Derin öğrenme ilk defa 2012 yılında bilim dünyasında çok büyük yankı oluşturmuştur. Dünyada nesne tanımlama kategorisinde en büyük yarışma olan Büyük Ölçekli Görsel Tanıma Yarışması (ImageNet) (Competition 2012) o yıl derin öğrenmede temel mimari kabul edilen Evrişimsel Sinir Ağı (ESA) ile kazanılır (Görsel 3.19). Bu sonuç ile derin öğrenmede çok büyük bir yükseliş olur. Yarışmada %26,1 olan Top-5 hata oranı %15,3 gibi bir orana düşürülür. Derin öğrenmedeki ilerlemeler Top-5 hata oranını %3,6'ya kadar düşürmüştür.



Görsel 3.19: Büyük Ölçekli Görsel Tanıma Yarışması(ImageNet)(Competition 2012) sonuçları

3.2.7. Yapay Sinir Ağları Uygulamaları İçin Gerekli Donanımlar

Yapay sinir ağlarında (YSA) gizli katmanların, ara katmanların kullanılmasıyla bellek hafızası ve hesaplama için işlemci gücü kapasitesi artmıştır. Gizli katman sayısının artırılmasıyla derinleştirilen yapay sinir ağları, daha büyük belleklere sahip çok daha hızlı bilgisayar ihtiyacını meydana getirir.

Örneğin İmageNet yarışması için kullanılan veri setinin 1,2 milyon olduğu düşünülürse kullanılacak bellek hafızasının önemi ortaya çıkar. Aynı şekilde birden fazla gizli katmana sahip bir ağın eğitilmesi esnasında geriye yayılım algoritmasının kullanılması için yapılan hesaplamalar paralel işlemciler ile daha hızlı gerçekleştirilebilir. Bu sebeple derin ağların eğitimi için Central Processing Unit (CPU) yerine genel amaçlı kullanılmak üzere ortaya çıkan Graphic Processing Unit (GPU) kullanılır.

3.2.8. Yapay Sinir Ağları Uygulamaları İçin Kullanılan Programlama Dilleri

Yapay zekâ son yıllarda çok hızlı gelişerek yaşamın çoğu alanına girmeye başlar. Sürekli yeni gelişmelerin yaşandığı yapay zekâ ve alt dalları olan makine öğrenmesi, derin öğrenme ve yapay sinir ağları alanlarında konuya göre hangi programlama dili ile uygulama geliştirileceğine dair çok önemli bir konu hâline gelir. Yapay zekâ ve alt dallarında uygulama geliştirmek için herhangi bir dil kullanılabilir? Bu konuda bir tek dil yerine, geliştirilmek istenen uygulama özelliklerine göre yani ihtiyaca göre dil seçmek en doğrusudur.

3.2.8.1. Python

Python, DRY [Don’t Repeat Yourself (kendinizi tekrar etmeyin.)] ve RAD (hızlı uygulama geliştirme) üzerine odaklanmış bir dildir. 1990'ların başlarında geliştirilen Python, ölçeklenebilirlik, uyarlanabilirlik ve öğrenme kolaylığı gibi özellikleriyle en hızlı büyüyen programlama dillerinden biri hâline gelmiştir.

Python; mobil uygulama, web uygulaması, veri bilimi veya yapay zekâ olsun her tür projeyi mümkün kılan yüzlerce kitaplığa sahiptir. Örneğin bilimsel hesaplama için "NumPy", makine öğrenimi için "Pybrain", gelişmiş hesaplama için "Scipy" ve yapay zekâ için "AIMA" kütüphaneleri mevcuttur.

Python'ın bütünsel dil tasarımı, düşük seviyeli ve yüksek seviyeli programlama dengesi, modüler programlama ve test çerçeveleri onu diğer dillerden farklı kılar. Bir diğer avantajı ise hızlı prototiplemedir.

3.2.8.2. C++

C++ diğer dillerden daha hızlıdır. Donanım düzeyinde iletişim kurma yeteneği, kod yürütme süresini iyileştirmeye olanak tanır. Zamana duyarlı yapay zekâ projeleri için son derece kullanışlıdır. Sinir ağlarında bulunanlar gibi istatistiksel AI yaklaşımı için kullanılabilir.

Daha hızlı yürütme süresi ve OOP ilkeleriyle C++, kendisini AI programları için iyi bir aday yapar. Aslında, makine öğrenimi ve derin öğrenme kitaplıklarının büyük bir kısmı C/C++ ile yazılmıştır ve aynısı için API'ler ve diğer programlama dilleri için sarmalayıcı sunar.

Çalışma zamanı ve performans üzerinde kontrol sahibi olunmak isteniyorsa C++ iyi bir seçimdir. Şablonların kullanımı daha güvenlidir (tür güvenliği) ve API'leri genelleştirmek için daha iyi bir yol sağlar. Şablonlar çoğu şeyi basitleştirebilecek güçlü bir teknik olsa da kullanımlarının ne zaman uygun olduğuna karar vermek için daha fazla zaman ve deneyim gerektirir.

3.2.8.3. Lisp

Lisp, 'Yapay Zekâ' terimini ortaya atan Dr. John MaCarthy tarafından oluşturulan en eski (1958'de geliştirilmiştir.) ve önde gelen dillerden biridir. Günümüzde çok kullanılmasa da dil hem esnek hem de genişletilebilirdir.

Lambda Calculus hesaplaması için geliştirilen dil, başlangıcından bu yana çok gelişim göstermiştir. Dil, bilgisayar biliminde özyineleme, dinamik yazma, üst düzey işlevler, otomatik depolama yönetimi, kendi kendini barındıran derleyici ve ağaç veri yapısı gibi birçok fikri tanıtmıştır.

Lisp, sembollerle çok iyi hesaplayan bir programın uygulanmasını desteklediği için yapay zekâ yazılımı geliştirmek için kullanılır. Sembolik ifade ve bunlarla hesaplama, Lisp kullanımının iyi olduğu alanlardır.

Ayrıca Lisp, bir makro sistemden verimli kod üretebilen iyi geliştirilmiş bir derleyiciden ve hashtable'lar ve dinamik boyutlu listeler dâhil olmak üzere bir koleksiyon türleri kitaplığından oluşur.

3.2.8.4. Java

Java programlama diliyle ilgili en iyi özellik, Java destekli tüm platformlarda çalışacak tek bir uygulama sürümü oluşturulmasına olanak tanıyan Java Sanal Makinesi [Java Virtual Machine (JVM)] teknolojisidir. Ayrıca Java programlama dilinin diğer güçlü yönleri şeffaflık, sürdürülebilirlik ve taşınabilirliktir. Java'da yapay zekâ uygulamalarını geliştirmenin faydaları büyük ölçekli projeleri desteklemesi, daha iyi kullanıcı etkileşimi, hata ayıklama kolaylığı, kolaylaştırılmış görselleştirme, Swing ve Standard Widget Toolkit'in dâhil edilmesidir. En büyük avantajı ise çok yönlülüğüdür.

Java'da geliştirilen iyi bilinen uygulamalardan bazıları şunlardır:

- Makine öğrenimi ve veri madenciliğine adanmış WEKA makine öğrenimi paketi
- Sinir ağlarını tasarlamak, eğitmek ve test etmek için JOONE sinir motoru
- ALICE (yapay dilsel internet bilgisayar varlığının kısaltması), doğal dil işleme sohbet robotları
- Robocode, Java programlama ilkelerini öğrenmek için açık kaynaklı bir oyun

3.2.8.5. Prolog

Prolog, hesaplamalı dil bilim ve yapay zekâ ile ilişkili bir mantık programlama dili ve anlamsal çıkarım motorudur. Teorem kanıtlama, sayısal olmayan programlama, doğal dil işleme ve genel olarak AI için yaygın olarak kullanılan esnek ve güçlü bir çerçeveye sahiptir.

Biçimsel mantığı olan bildirimsel bir dildir. AI geliştiricileri, önceden tasarlanmış arama mekanizması, belirsizliği, geri izleme mekanizması, özyinelemeli doğası, üst düzey soyutlama ve kalıp eşleştirmesi için buna değer verir.

Prolog, yapılandırılmış nesneler ve bunlar arasındaki ilişkileri içeren problemler için çok uygundur. Prolog dilinde nesneler arasındaki uzamsal ilişkileri ifade etmek daha kolaydır. Örneğin mavi üçgenin arkasında yeşil bir üçgen vardır. Genel bir kural belirtmek de kolaydır. Eğer A nesnesi kişiye B nesnesinden daha yakınsa ve B nesnesi C'den daha yakınsa, o zaman A, C'den daha yakın olmalıdır. Prolog dilinin doğası, gerçekleri ve kuralları uygulamayı basit ve anlaşılır kılar. Gerçek ise Prolog dilindeki her şey bir gerçek veya bir kuraldır. Binlerce gerçek ve kurala sahip olduğunuzda bile veri tabanını sorgulamanıza olanak tanır. Prolog dili, grafiksel kullanıcı arabirimi, yönetim ve ağ uygulamalarının geliştirilmesini destekler. Sesli kontrol sistemleri ve doldurma şablonları gibi projeler için çok uygundur.

3.2.9. Google Colaboratory Kullanımı

Colab veya diğer adıyla "Colaboratory" uygulaması tarayıcıda Python dilinde uygulama geliştirmeyi ve çalıştırmayı sağlar. Jupiyter notebook gibi çalışan Colab Google'ın ücretsiz hizmetidir. Bu uygulamanın bazı özellikleri aşağıda belirtilmiştir.

- Hiç yapılandırma gerektirmez.
- GPU'lara ücretsiz erişim imkânı sunar.
- Kolay paylaşım imkânı sunar.
- Kullanmak için sadece Google hesabının olması yeterlidir.

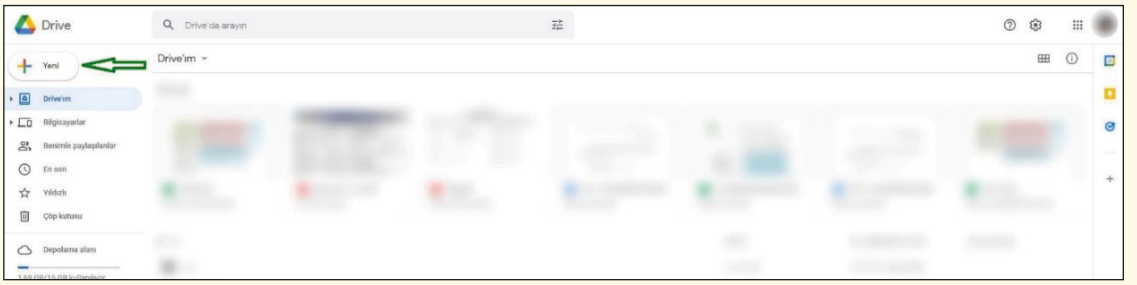
Öğrenci, veri bilimci ya da yapay zekâ araştırmacısı gibi tüm kullanıcı düzeylerinde Colab işleri oldukça kolaylaştırır. Büyük verilerle yapay zekâ uygulamaları geliştirirken hızlı olmak için amazon gibi bulut servislerden faydalanılması ya da çok güçlü kişisel bilgisayarın olması gerekir. Google Colab ücretsiz olarak GPU ve tensör işlem birimi olan TPU'yu kullanma olanağı sağlar.

Colab üzerinde Python ile uygulamalar yazılarak, tensorflow, opencv ve keras gibi kütüphaneler kullanılarak veri analizleri yapılabilir.

1. UYGULAMA

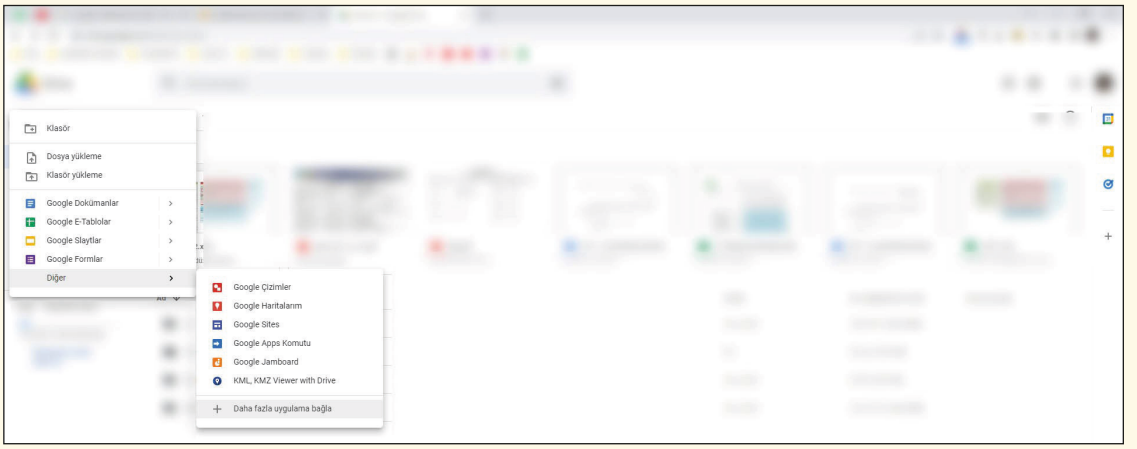
Colab uygulamasını açmak için aşağıdaki işlem adımlarını takip ediniz.

1. Adım: Google Drive'da yeni klasör açınız (Görsel 3.20).



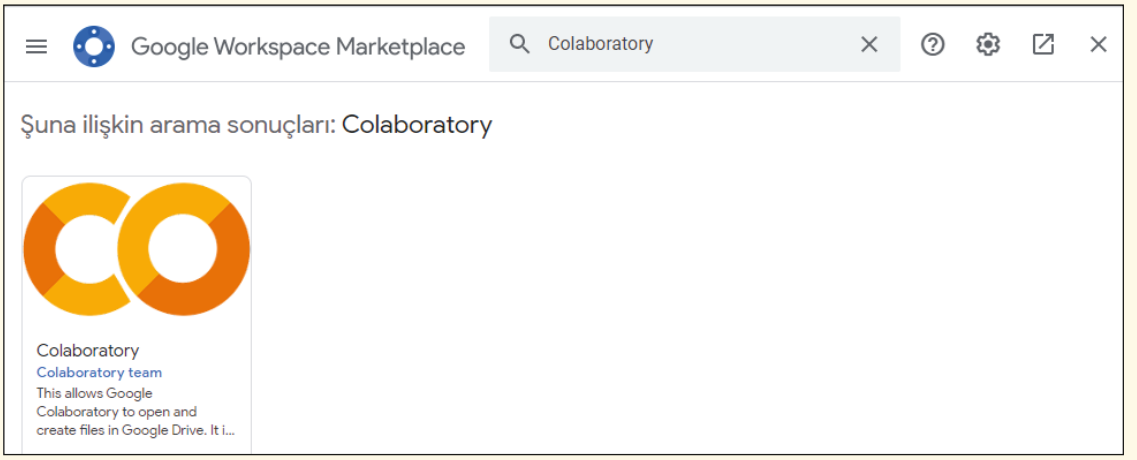
Görsel 3.20: Colab için Google drive'da klasör açma

2. Adım: Uygulamayı ilk defa açıyorsanız diğer menüsünden daha fazla uygulama bağla seçeneğini tıklayınız (Görsel 3.21).



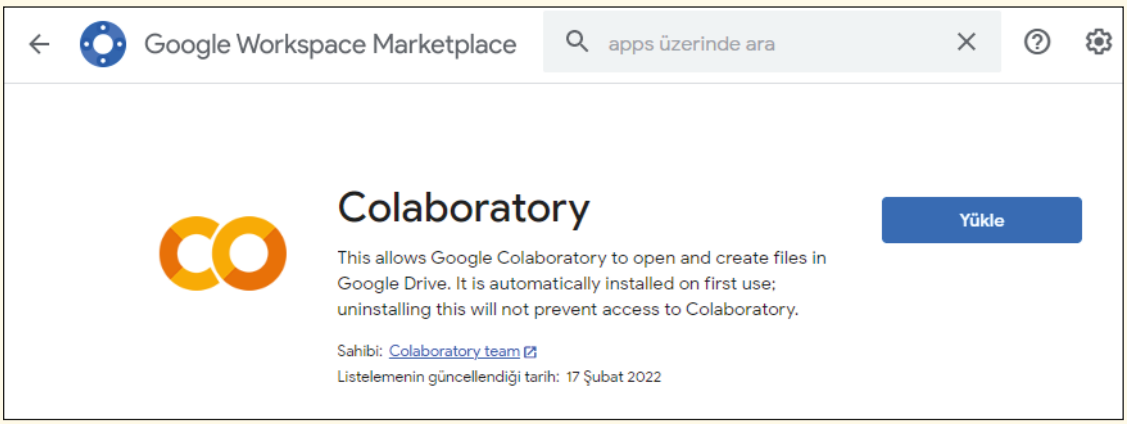
Görsel 3.21: Colab için Google drive'da uygulama bağlama

3. Adım: Uygulama arama kutusuna Colaboratory yazıp arayınız (Görsel 3.22).



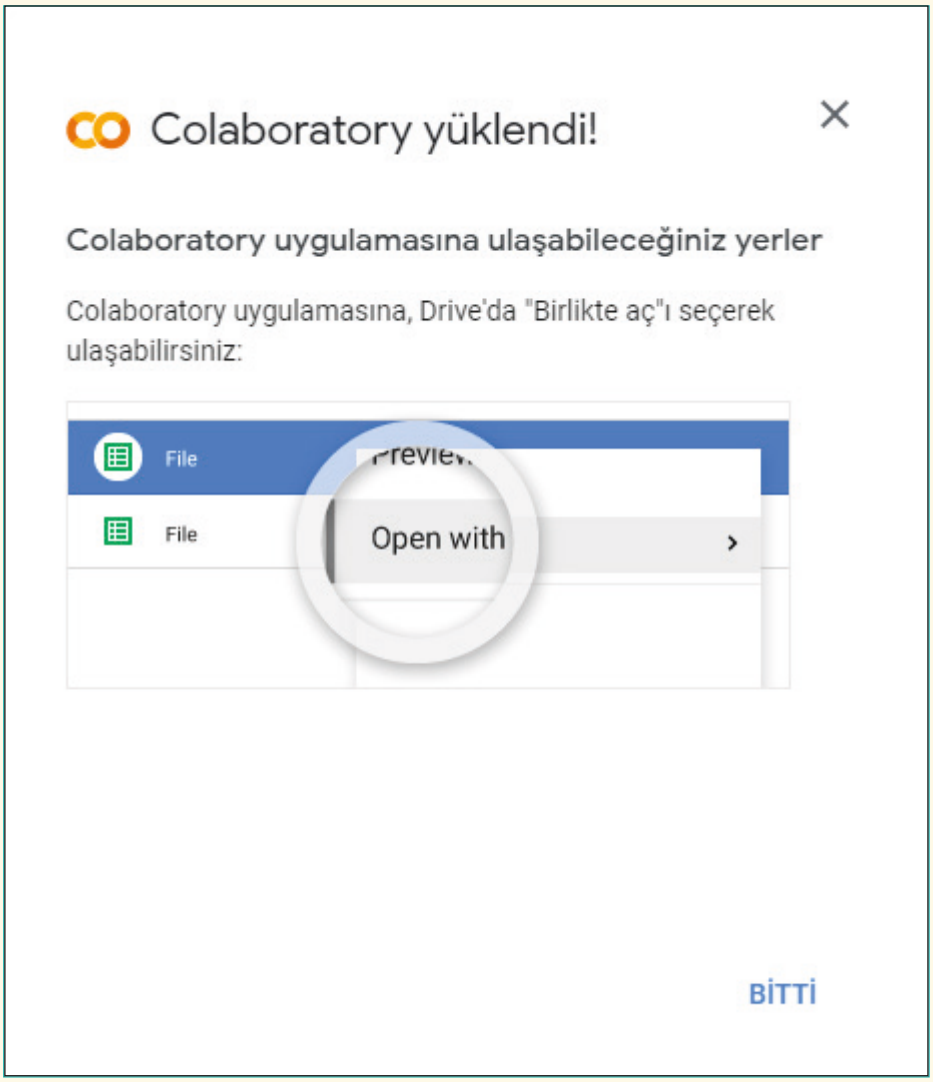
Görsel 3.22: Colab kurulumu için arama işlemi

4. Adım: Çıkan sonuçlardan Colab'ı yükleyiniz (Görsel 3.23).



Görsel 3.23: Colab yükleme

5. Adım: Yükleme işleminin Görsel 3.24'teki gibi tamamlandığından emin olunuz (Görsel 3.24).

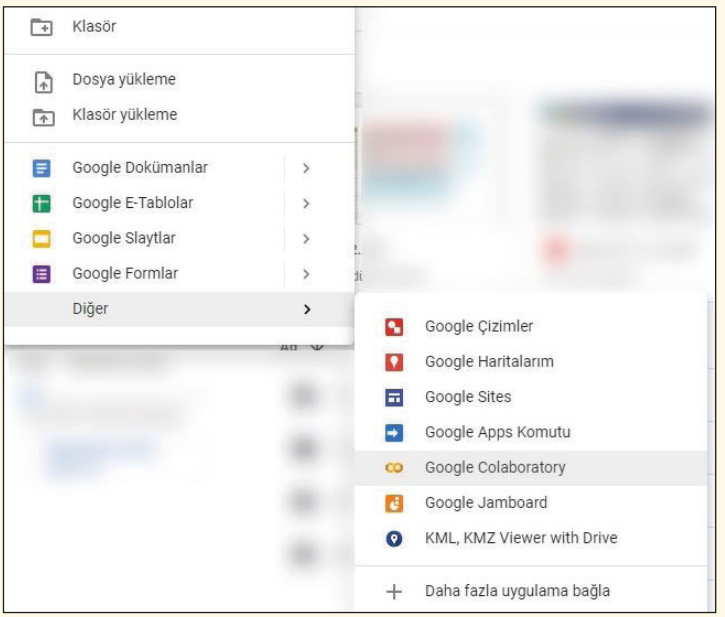


Görsel 3.24: Colab kurulumunun tamamlanması

2. UYGULAMA

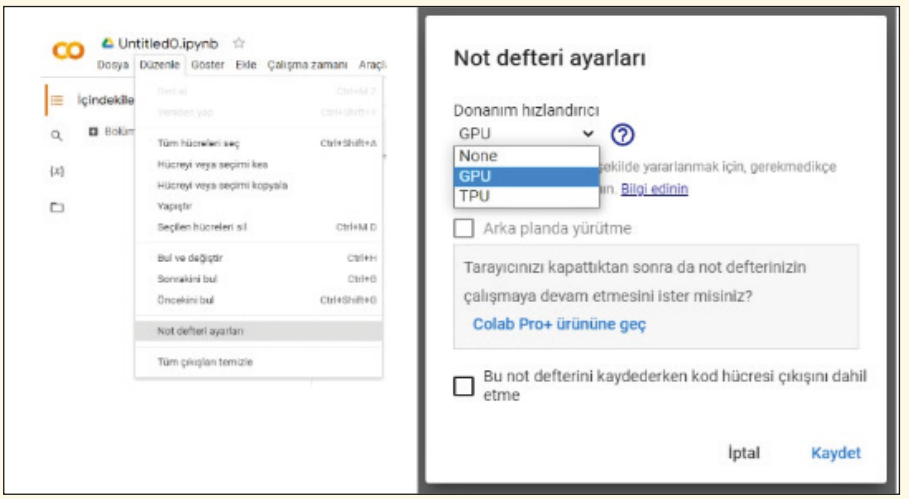
Colab ücretsiz GPU hizmetinden faydalanmak için aşağıdaki işlem adımlarını takip ediniz.

1. Adım: Colab uygulamasını açınız (Görsel 3.25).



Görsel 3.25: Colab ücretsiz GPU hizmetinden faydalanma

2. Adım: Notebook üzerinde Edit >> Notebook settings yolunu izleyerek Görsel 3.26'daki ekrana ulaşınız. Bu ekranda Hardware accelerator'u GPU olacak şekilde değiştiriniz. Kaydedip ekrandan çıkınız.



Görsel 3.26: Colab ayarları

3. Adım: Bu aşamadan sonra Google Colaboratory ile ücretsiz Tesla K80 GPU üzerinde veri analiz çalışmalarını sürdürebilirsiniz.

SIRA SİZDE

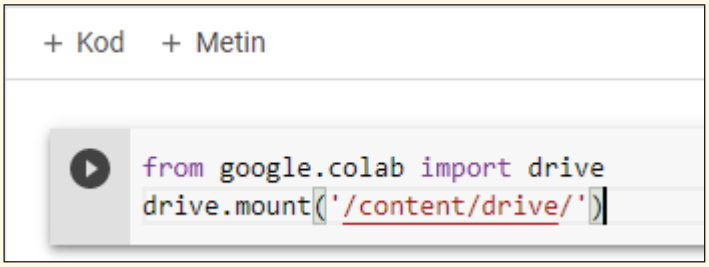
Colab ücretsiz GPU hizmetinden faydalanmak için gerekenleri kendi Colab hesabınızda yapınız.

3. UYGULAMA

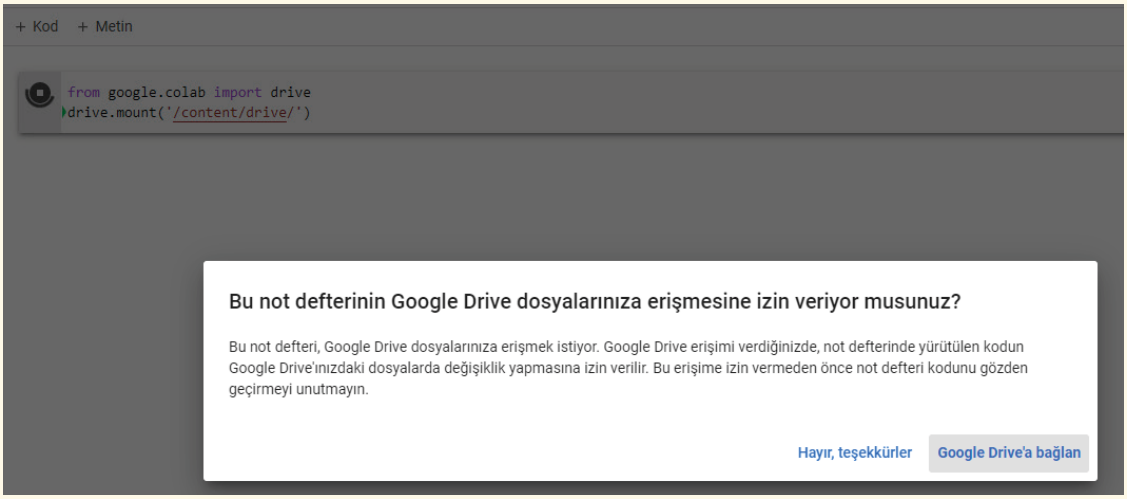
Google Colab ile Google Drive hesabını bağlamak için aşağıdaki işlem adımlarını takip ediniz.

Google Colab'tan tam anlamda faydalanabilmek ve verimli kullanmak için Google Drive hesabı ile Colab bağlanmalıdır. Bağlama işlemi gerçekleştirilmezse kodlar çalıştırıldığında farklı hata mesajları ile karşılaşılabilir. Bu işlem için izin işlemleri yapılmalıdır.

1. Adım: Görsel 3.27'de yer alan kod satırlarını çalıştırınız. Kod satırları çalıştırıldığında Google sizden doğrulama isteyecektir. Böylelikle Drive hesabınız ile bağlantı kurmuş olacaksınız. İsterseniz '/content/drive/' kısmının devamına Google Drive içinde oluşturduğunuz yeni bir klasörün uzantısını dâhil edebilirsiniz. Oynat çalıştır düğmesine tıklayarak Görsel 3.28'deki ekran ile karşılaşıp karşılaşmadığınızı kontrol ediniz.

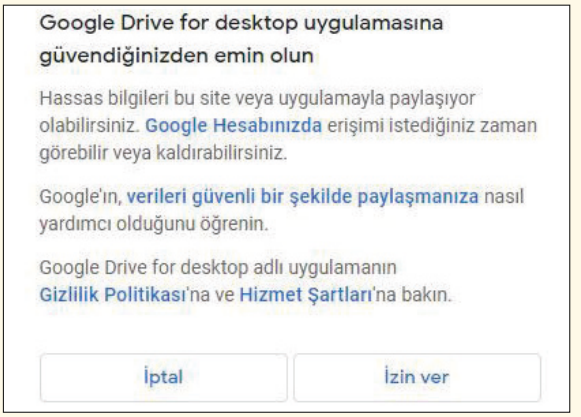


Görsel 3.27: Colab ile Google Drive hesabını bağlama



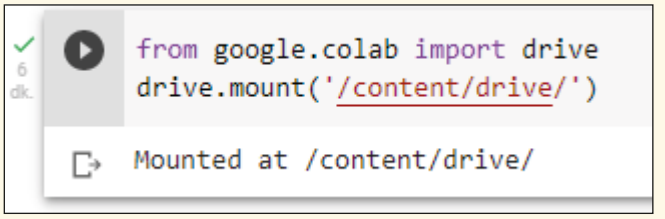
Görsel 3.28: Colab ile Google Drive hesabını bağlama

2. Adım: Google Drive'a bağlan düğmesine tıkladıktan sonra izin ver seçeneği ile karşılaşacaksınız. Görsel 3.29'da görülen ekranda Google Drive izin ver düğmesine tıklayınız.



Görsel 3.29: Colab ile Google Drive hesabını bağlama

3. Adım: Herhangi bir sorun oluşmadıysa Colab not defterinizin Drive hesabınıza bağlanıp bağlanmadığını kontrol ediniz (Görsel 3.30).



Görsel 3.30: Colab ile Google Drive hesabını bağlama

4. Adım: Bir hata ile karşılaşırsanız tam olarak bağlamak için Görsel 3.31'de görülen kodları çalıştırınız.

```
!apt-get install -y -qq software-properties-common python-software-properties module-init-tools
!add-apt-repository -y ppa:alessandro-strada/ppa 2>&1 > /dev/null
!apt-get update -qq 2>&1 > /dev/null
!apt-get -y install -qq google-drive-ocamlfuse fuse
from google.colab import auth
auth.authenticate_user()
from oauth2client.client import GoogleCredentials
creds = GoogleCredentials.get_application_default()
import getpass
!google-drive-ocamlfuse -headless -id={creds.client_id} -secret={creds.client_secret} < /dev/null 2>&1 | grep URL
vcode = getpass.getpass()
!echo {vcode} | google-drive-ocamlfuse -headless -id={creds.client_id} -secret={creds.client_secret}
```

Görsel 3.31: Colab ile Google Drive hesabını bağlama

5. Adım: Kodlar çalıştırıldığında Görsel 3.32'de görülen doğrulama sonuç ekran ile karşılaşırsınız.

Görsel 3.32: Colab ile Google Drive hesabını bağlama

6. Adım: Çalıştırılan kodun altında çıkan linke tıklayarak doğrulama kodunuzu alınız. Doğrulama kodunu Enter verification code yazan yere yapıştırıp Enter tuşuna basınız. Karşınıza toplamda 2 kere kutucuk çıkacaktır. İki kutucuğa da doğrulama kodunuzu yapıştırıp Enter tuşuna basınız. İşlemleri tamamlandığınızda Görsel 3.33'te gösterilen bir sonuçla karşılaşıp karşılaşmadığınızı kontrol ediniz.

```
!apt-get install -y -qq software-properties-common python-software-properties module-init-tools
!add-apt-repository -y ppa:alessandro-strada/ppa 2>&1 > /dev/null
!apt-get update -qq 2>&1 > /dev/null
!apt-get -y install -qq google-drive-ocamlfuse fuse
from google.colab import auth
auth.authenticate_user()
from oauth2client.client import GoogleCredentials
creds = GoogleCredentials.get_application_default()
import getpass
!google-drive-ocamlfuse -headless -id={creds.client_id} -secret={creds.client_secret} < /dev/null 2>&1 | grep URL
vcode = getpass.getpass()
!echo {vcode} | google-drive-ocamlfuse -headless -id={creds.client_id} -secret={creds.client_secret}

Please, open the following URL in a web browser: https://accounts.google.com/o/oauth2/auth?client_id=32555940559.apps.googleusercontent.com&redirect_uri=urn
.....
Please, open the following URL in a web browser: https://accounts.google.com/o/oauth2/auth?client_id=32555940559.apps.googleusercontent.com&redirect_uri=urn
Please enter the verification code: Cannot retrieve auth tokens.
Failure("Unexpected error response: {\n  \"error\" : \"invalid_grant\",\n  \"error_description\" : \"Code was already redeemed.\"\n}")
```

Görsel 3.33: Colab ile Google Drive hesabını bağlama

7. Adım: Görsel 3.33 ekranı ile karşılaştıysanız Colab ile Drive'ınız bağlı durumda demektir. Mevcut Drive içeriğindeki klasörlerinizi listelemek için aşağıdaki kod blokunu çalıştırabilirsiniz (Görsel 3.34).

```
!ls '/content/drive/MyDrive'
```

Görsel 3.34: Colab ile Google Drive hesabını bağlama

SIRA SİZDE

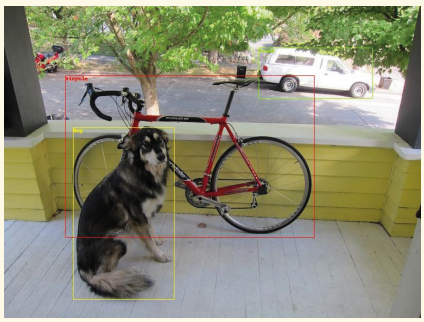
Google Colab hesabı oluşturarak Google drive hesabınız ile bağlı duruma getiriniz.

3.2.10. Uygulamalar

Bu bölümde yapay sinir ağları kullanılarak geliştirilen uygulamalar yer almaktadır.

4. UYGULAMA

Yolo v4 teknolojisi kullanılarak özelleşmiş nesne algılama ve nesne sınıflandırma için izlenmesi gereken işlemler adım adım diğer sayfadaki uygulamada gerçekleştirilecektir (Görsel 3.35). Yolo v4 modelinin, yerel makinelerde eğitilmesi iş-leminin bir hayli zorlu olması sebebiyle Google Colab üzerinde eğitilmesi ele alınacaktır. Kod satırları Colab not defterinin +kod bölümüne eklenip çalıştırılarak uygulama için gereken kütüphaneler dâhil edilir.



Görsel 3.35: Yolo v4 ve OpenCv ile nesne sınıflandırma

```
# bağımlılıkları ve kütüphaneleri dâhil etmek için aşağıdaki kod satırları çalıştırılır.
from IPython.display import display, Javascript, Image
from google.colab.output import eval_js
from google.colab.patches import cv2_imshow
from base64 import b64decode, b64encode
import cv2
import numpy as np
import PIL
import io
import html
import time
import matplotlib.pyplot as plt
%matplotlib inline
```

YOLOv4 algılamalarını gerçekleştirmek için bu yapay sinir ağı eğitiminde oldukça popüler olan AlexeyAB'nin darknet deposu kullanılacaktır. Bu depoyu kullanmak için aşağıdaki kodlar yazılıp çalıştırılır.

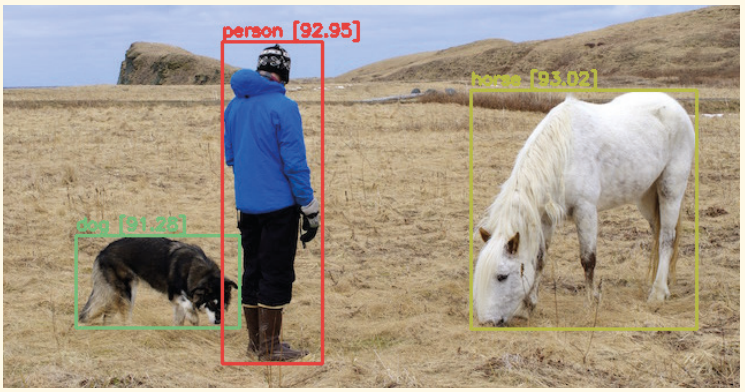
```
# darknet reposunu github'tan çekmek için aşağıdaki kod çalıştırılır.
!git clone https://github.com/AlexeyAB/darknet

# GPU opencv ve libsoyu aktif etmek için aşağıdaki kod satırları yazılır.
%cd darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
!sed -i 's/LIBSO=0/LIBSO=1/' Makefile

# darkneti oluşturup kullanmak için aşağıdaki kod satırı çalıştırılır.
!make

# Google sunucularından 80 sınıfı (nesneyi) algılamak için önceden eğitilmiş, ölçeklenmiş yolov4 ağırlıkları dosyasını almak için aşağıdaki kod çalıştırılır.
wget --load-
cookies /tmp/cookies.txt "https://docs.google.com/
uc?export = download&confirm = $(wget --quiet --save-
cookies/tmp/cookies.txt --keep-session-
cookies --no-check-
certificate 'https://docs.google.com/uc?export =
download&id = 1V3vsIaxAlGWvK4Aar9bAiK5U0QFttKwq' -O-
| sed -rn 's/.*confirm = ([0-9A-Za-z_]+).*/\1\n/p')&id
= 1V3vsIaxAlGWvK4Aar9bAiK5U0QFttKwq" -O yolov4-
csp.weights && rm -rf /tmp/cookies.txt
```

YOLOv4'ü Python koduyla kullanmak için fonksiyonları Colab'a aktararak darknet.py içinde bulunan önceden oluşturulmuş fonksiyonlardan bazıları kullanılacaktır. Fonksiyon tanımlarını ayrıntılı olarak görmek için darknet.py dosyasını inceleyebilirsiniz.

```
# nesne algılamaları ve darknet fonksiyonlarını içe aktarmak için
aşağıdaki kodlar çalıştırılır.
from darknet import *

# YOLOv4 mimarisini yapay sinir ağına eklemek için aşağıdaki kod
çalıştırılır.
network, class_names, class_colors = load_network("cfg/yolov4-
csp.cfg", "cfg/coco.data", "yolov4-csp.weights")
width = network_width(network)
height = network_height(network)

# görüntüde algılamayı çalıştırmak ve darknet yardımcı fonk-
siyonlarını yüklemek için aşağıdaki kod çalıştırılır.
def darknet_helper(img, width, height):
  darknet_image = make_image(width, height, 3)
  img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img_resized = cv2.resize(img_rgb, (width, height),
                           interpolation=cv2.INTER_LINEAR)

# nesne sınıflandırma işleminde sınırlayıcı kutuları uygun boyuta
dönüştürmede görüntü oranlarını almak için aşağıdaki kod çalış-
tırılır.
  img_height, img_width, _ = img.shape
  width_ratio = img_width/width
  height_ratio = img_height/height

# run model on darknet style image to get detections.
copy_image_from_bytes(darknet_image, img_resized.tobytes())
detections = detect_image(network, class_names, darknet_image)
free_image(darknet_image)
return detections, width_ratio, height_ratio
```

Yapay sinir ağı modelinin başarıyla yüklendiğinden emin olmak ve bir test görüntüsü üzerinde doğru tespitler yapmak için test çalışması gerçekleştirilecektir (Görsel 3.36).

```
# depoyla birlikte gelen person.jpg görüntüsü üzerinde test
çalıştırılır.
image = cv2.imread("data/person.jpg")
detections, width_ratio, height_ratio = darknet_helper(image,
width, height)

for label, confidence, bbox in detections:
   left, top, right, bottom = bbox2points(bbox)
   left, top, right, bottom = int(left * width_ratio), int(top *
   height_ratio), int(right * width_ratio), int(bottom * height_
   ratio)
   cv2.rectangle(image, (left, top), (right, bottom), class_
   colors[label], 2)
   cv2.putText(image,"{} [{:.2f}]".format(label,float(confidence)),
                  (left, top - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
                   class_colors[label], 2)
cv2_imshow(image)
```



Görsel 3.36: Nesne sınıflandırma test sonucu görüntüsü

Aşağıda, sonraki adımlarda farklı görüntü türleri arasında kolayca dönüştürme yapmak için kullanılacak tanımlanmış birkaç yardımcı işlev verilmiştir.

```
# JavaScript nesnesini bir OpenCV görüntüsüne dönüştürme fonk-
siyonu için aşağıdaki kod satırları çalıştırılır.
def js_to_image(js_reply):
  """
  Params:
        js_reply: JavaScript object containing image from webcam
  Returns:
         img: OpenCV BGR image
  """
```

```
# base64
image_bytes = b64decode(js_reply.split(',')[1])

# görsel baytları NumPy dizisine dönüştürmek için aşağıdaki kod kullanılır.
jpg_as_np = np.frombuffer(image_bytes, dtype=np.uint8)

# NumPy dizisini opencv görüntüsüne çözmek için aşağıdaki kod satırı kullanılır.
img = cv2.imdecode(jpg_as_np, flags=1)

return img

# Nesne sınıflandırmada tespit işlemlerinde OpenCV Dikdörtgen sınırlayıcı kutu görüntüsünü video akışına yerleştirebilecek base64 bayt dizisine dönüştürme fonksiyonu.
def bbox_to_bytes(bbox_array):
  """
  Params:
          bbox_array: Numpy array (pixels) containing rectangle to overlay on video stream.
  Returns:
        bytes: Base64 image byte string
  """

# diziyi pil görüntüsüne dönüştürme işlemi.
bbox_PIL = PIL.Image.fromarray(bbox_array, 'RGBA')
iobuf = io.BytesIO()

# dönüş için bbox ı png olarak biçimlendirilir.
bbox_PIL.save(iobuf, format = 'png')

# dönüş dizisi hazırlanır.
bbox_bytes = '{}'.format((str(b64encode
(iobuf.getvalue()), 'utf-8')))

return bbox_bytes
```

YOLOv4'ü web kamerasından alınan görüntüler üzerinde çalıştırmak oldukça basittir. Google Colab'ın çeşitli görevleri gerçekleştirmesi için çeşitli yararlı kod işlevlerine sahip Kod Parçacıkları içindeki kod kullanılır.

Bilgisayarınızın web kamerasını kullanmak için JavaScript kodunu çalıştıran Camera Capture için kod parçacığı kullanılır. Kod parçacığı bir web kamerası fotoğrafı çekecek ve daha sonra nesne algılama için YOLOv4 modeline geçecektir.

Aşağıda, JavaScript kullanarak web kamerası resmini çekme ve ardından üzerinde YOLOv4 çalıştırma kodları bulunmaktadır.

```
def take_photo(filename='photo.jpg', quality=0.8):
  js = Javascript('''
    async function takePhoto(quality) {
      const div = document.createElement('div');
```

```
      const capture = document.createElement('button');
      capture.textContent = 'Capture';
      div.appendChild(capture);

      const video = document.createElement('video');
      video.style.display = 'block';
      const stream = await navigator.mediaDevices.getUserMedia
      ({video: true});

      document.body.appendChild(div);
      div.appendChild(video);
      video.srcObject = stream;
      await video.play();

     // Video ögesine sığdırmak için çıktıyı yeniden boyutlandırma
     kod satırı.
      google.colab.output.setIframeHeight(document.
      documentElement.scrollHeight, true);

     // Sınıflandırılacak görsel için tıklayarak görüntüyü yaka-
     lama kod bloku.
      await new Promise((resolve) => capture.onclick = resolve);
      const canvas = document.createElement('canvas');
      canvas.width = video.videoWidth;
      canvas.height = video.videoHeight;
      canvas.getContext('2d').drawImage(video, 0, 0);
      stream.getVideoTracks()[0].stop();
      div.remove();
      return canvas.toDataURL('image/jpeg', quality);
    }
    ''')
display(js)

# fotoğraf verilerinin alınması.
data = eval_js('takePhoto({})'.format(quality))

# OpenCV formatındaki görüntüyü alın.
img = js_to_image(data)

# web kamerası görüntüsünde darknet yardımcısını almak için
gereken kod satırı.
detections, width_ratio, height_ratio = darknet_helper(img, width,
height)

# Algılanan nesneler arasında dolaşarak ve bunları web kamerası
görüntüsüne çizdirmek için gereken kodlar.
for label, confidence, bbox in detections:
    left, top, right, bottom = bbox2points(bbox)
    left, top, right, bottom = int(left * width_ratio), int(top
* height_ratio), int(right * width_ratio), int(bottom * height_
ratio)
```

```
    cv2.rectangle(img, (left, top), (right, bottom), class_colors
[label], 2)
    cv2.putText(img, "{} [{:.2f}]".format(label, float(confidence)),
                (left, top - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
                class_colors[label], 2)

# görseli kaydet.
  cv2.imwrite(filename, img)
  return filename

# web kamerası üzerinde görüntüyü yakalamak için gereken kodlar.
try:
  filename = take_photo('photo.jpg')
  print('Saved to {}'.format(filename))

# Az önce çekilen son fotoğrafı göster.
  display(Image(filename))
except Exception as err:

# Kullanıcının web kamerası yoksa oluşacak hata kodlarını yazdır.

# Sayfaya erişmesi için izin kodları.
  print(str(err))

# Web kamerasını girdi olarak kullanarak canlı video akışınızı
düzgün bir şekilde oluşturmak için kullanılan JavaScript kodları.
def video_stream():
  js = Javascript('''
    var video;
    var div = null;
    var stream;
    var captureCanvas;
    var imgElement;
    var labelElement;
    var pendingResolve = null;
    var shutdown = false;

  function removeDom() {
      stream.getVideoTracks()[0].stop();
      video.remove();
      div.remove();
      video = null;
      div = null;
      stream = null;
      imgElement = null;
      captureCanvas = null;
      labelElement = null;
       }
```

```
function onAnimationFrame() {
  if (!shutdown) {
   window.requestAnimationFrame(onAnimationFrame);
     }
  if (pendingResolve) {
   var result = "";
   if (!shutdown) {
        captureCanvas.getContext('2d').
        drawImage(video, 0, 0, 640, 480);
        result = captureCanvas.toDataURL('image/jpeg', 0.8)
        }
        var lp = pendingResolve;
        pendingResolve = null;
        lp(result);
     }
   }
async function createDom() {
  if (div !== null) {
   return stream;
     }
  div = document.createElement('div');
  div.style.border = '2px solid black';
  div.style.padding = '3px';
  div.style.width = '100%';
  div.style.maxWidth = '600px';
  document.body.appendChild(div);
  const modelOut = document.createElement('div');
  modelOut.innerHTML = "<span>Status:</span>";
  labelElement = document.createElement('span');
  labelElement.innerText = 'No data';
  labelElement.style.fontWeight = 'bold';
  modelOut.appendChild(labelElement);
  div.appendChild(modelOut);
  video = document.createElement('video');
  video.style.display = 'block';
  video.width = div.clientWidth - 6;
  video.setAttribute('playsinline', '');
  video.onclick = () => { shutdown = true; };
  stream = await navigator.mediaDevices.getUserMedia(
          {video: { facingMode: "environment"}});
  div.appendChild(video);
  imgElement = document.createElement('img');
  imgElement.style.position = 'absolute';
  imgElement.style.zIndex = 1;
  imgElement.onclick = () => { shutdown = true; };
  div.appendChild(imgElement);
```

```
    const instruction = document.createElement('div');
    instruction.innerHTML =
          '<span style="color: red; font-weight: bold;">' +
          'When finished, click here or on the video to stop this
demo</span>';
    div.appendChild(instruction);
    instruction.onclick = () => { shutdown = true; };

    video.srcObject = stream;
    await video.play();

    captureCanvas = document.createElement('canvas');
    captureCanvas.width = 640; //video.videoWidth;
    captureCanvas.height = 480; //video.videoHeight;
    window.requestAnimationFrame(onAnimationFrame);
    return stream;
    }
    async function stream_frame(label, imgData) {
      if (shutdown) {
        removeDom();
        shutdown = false;
        return '';
      }
      var preCreate = Date.now();
      stream = await createDom();
      var preShow = Date.now();

      if (label != "") {
        labelElement.innerHTML = label;
      }

      if (imgData != "") {
        var videoRect = video.getClientRects()[0];
        imgElement.style.top = videoRect.top + "px";
        imgElement.style.left = videoRect.left + "px";
        imgElement.style.width = videoRect.width + "px";
        imgElement.style.height = videoRect.height + "px";
        imgElement.src = imgData;
      }
      var preCapture = Date.now();
      var result = await new Promise(function(resolve, reject) {
        pendingResolve = resolve;
      });
      shutdown = false;

      return {'create': preShow - preCreate,
              'show': preCapture - preShow,
              'capture': Date.now() - preCapture,
              'img': result};
    }
    ''')
```

```
  display(js)

def video_frame(label, bbox):
  data = eval_js('stream_frame("{}", "{}")'.format(label, bbox))
  return data

# web kamerasından video görüntü akışını başlat.
video_stream()

# label for video.
label_html = 'Capturing...'

# initialze bounding box to empty.
bbox = ''
count = 0
while True:
    js_reply = video_frame(label_html, bbox)
    if not js_reply:
        break

    # JS yanıtını OpenCV görüntüsüne dönüştür.
    frame = js_to_image(js_reply["img"])

    # sınırlayıcı kutu için şeffaf kaplama oluştur.
    bbox_array = np.zeros([480,640,4], dtype=np.uint8)

    # video karesinde darknet yardımcı fonksiyonunu getir.
    detections, width_ratio, height_ratio = darknet_helper
         (frame, width, height)

    # Algılamalar arasında dolaşarak, bunları şeffaf bindirme gö-
    rüntüsüne çizdirmek.
    for label, confidence, bbox in detections:
      left, top, right, bottom = bbox2points(bbox)
      left, top, right, bottom = int(left * width_ratio), int(top
* height_ratio), int(right * width_ratio), int(bottom * height_
ratio)
     bbox_array = cv2.rectangle(bbox_array, (left, top), (right,
bottom), class_colors[label], 2)
     bbox_array = cv2.putText(bbox_array, "{} [{:.2f}]".format
(label, float(confidence)),
                 (left, top - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
                  class_colors[label], 2)

     bbox_array[:,:,3] = (bbox_array.max(axis = 2) > 0 ).astype(int)
* 255

    # convert overlay of bbox into bytes.
    bbox_bytes = bbox_to_bytes(bbox_array)

    # update bbox so next frame gets new overlay.
    bbox = bbox_bytes
```

5. UYGULAMA

Bu uygulamada, el yazısı rakamları tanımak için Tensorflow kullanılarak bir Evrişimsel Sinir Ağı (CNN) modeli oluşturulacaktır (Görsel 3.37). Evrişimsel Sinir Ağı (CNN veya ConvNet), bir girdi görüntüsünü alabilen, görüntüdeki çeşitli yönlere, nesnelere önem derecesine göre öğrenilebilir ağırlıklar ve önyargılar atayan ve birini diğerinden ayırt edebilen bir Derin Öğrenme algoritmasıdır.

Görsel 3.37: El yazısı veri seti

Kullanılacak kütüphaneler ve bağımlılıklar aşağıda belirtilmiştir.

Tensorflow: ML modellerini geliştirmek ve eğitmek için kullanılır.

Matplotlib: Verileri çizmek için kullanılır.

Seaborn: Karışıklık matrisi çizmek için kullanılır.

NumPy: Lineer cebir işlemleri için kullanılır.

Pandas: Bir tabloda eğitim / test verilerini görüntülemek için kullanılır.

Math: Karekök vb. hesaplamak için kullanılır.

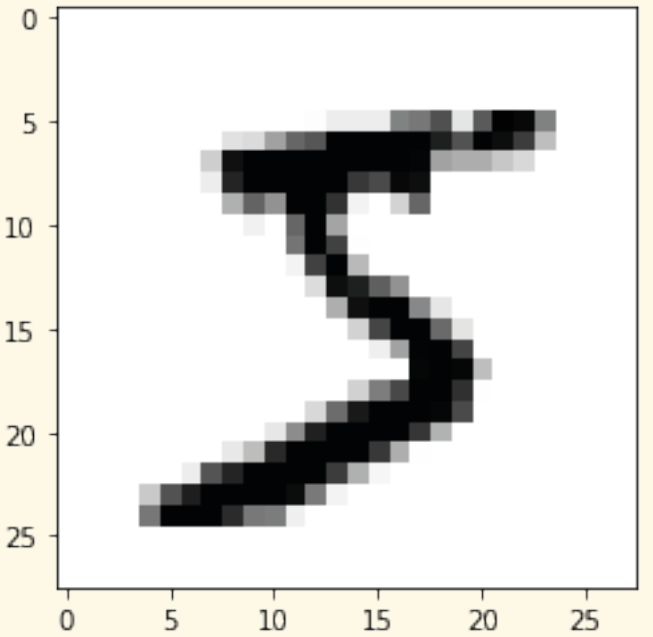
Datetime: Günlük klasör adları oluşturmak için kullanılır.

```
# Tensorflow versiyon v2 seçilir.
%tensorflow_version 2.x

#kütüphaneler dâhil edilir.
import tensorflow as tf
import matplotlib.pyplot as plt
import seaborn as sn
import numpy as np
import pandas as pd
import math
import datetime
import platform
print('Python version:', platform.python_version())
print('Tensorflow version:', tf.__version__)
print('Keras version:', tf.keras.__version__)
#daha sonra hata ayıklamak için tensorboard kullanılır.
# TensorBoard notdefteri uzantısı yüklenir.
%load_ext tensorboard
```

Eğitim veri seti, 0'dan 9'a kadar elle yazılmış rakamların 60000 tane 28x28 px görüntüsünden oluşur. Test veri seti 10000 28x28 piksel görüntüden oluşur.

```
mnist_dataset = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist_dataset.load_data()
print('x_train:', x_train.shape)
print('y_train:', y_train.shape)
print('x_test:', x_test.shape)
print('y_test:', y_test.shape)

# Görüntü parametreleri daha sonra veriyi yeniden şekillendirme
ve model eğitimi için kullanılacak sabitlere kaydedilir.
(_, IMAGE_WIDTH, IMAGE_HEIGHT) = x_train.shape IMAGE_CHANNELS = 1

print('IMAGE_WIDTH:', IMAGE_WIDTH);
print('IMAGE_HEIGHT:', IMAGE_HEIGHT);
print('IMAGE_CHANNELS:', IMAGE_CHANNELS);
```

Veri kümesindeki her görüntünün nasıl göründüğü aşağıda açıklanmıştır. 28x28'lik bir tam sayı matrisidir (0'dan 255'e kadar). Her tam sayı, bir pikselin rengini temsil eder. Bu sayı matrisi aşağıdaki gibi çizilebilir (Görsel 3.38).



Görsel 3.38: Örnek sayı matrisi

Rakamların nasıl yazıldığını anlamak için biraz daha eğitim örneği yazdırılabilir (Görsel 3.39).

```
numbers_to_display = 25
num_cells = math.ceil(math.sqrt(numbers_to_display))
plt.figure(figsize=(10,10))
for i in range(numbers_to_display):
    plt.subplot(num_cells, num_cells, i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(x_train[i], cmap=plt.cm.binary)
    plt.xlabel(y_train[i])
plt.show()
```

Görsel 3.39: Eğitim örneği

Verileri Yeniden Şekillendirme

Evrişim katmanlarını kullanmak için verileri yeniden şekillendirmek ve ona bir renk kanalı eklemek gerekir. Şu anda görüldüğü gibi her basamağın bir (28, 28) şekli vardır. Bu da 0 ila 255 arasında 28x28 renk değerleri matrisi olduğu anlamına gelir. Onu (28, 28, 1) şeklinde yeniden şekillendirmek gerekir. Böylece her biri piksel potansiyel olarak birden fazla kanala sahip olabilir (Kırmızı, Yeşil, Mavi gibi).

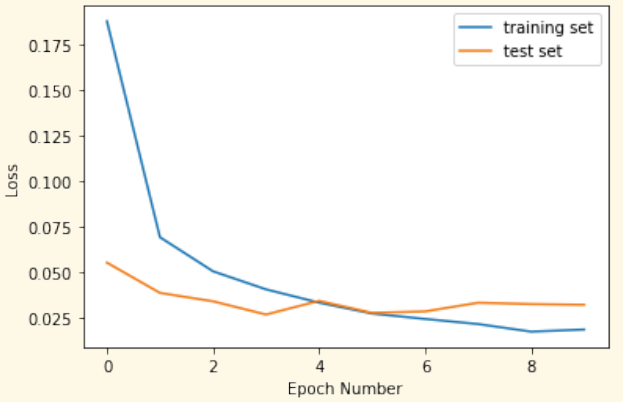
```
mnist_dataset = tf.keras.datasets.mnist
x_train_with_chanels = x_train.reshape(
    x_train.shape[0],
    IMAGE_WIDTH,
    IMAGE_HEIGHT,
    IMAGE_CHANNELS
)

x_test_with_chanels = x_test.reshape(
    x_test.shape[0],
    IMAGE_WIDTH,
    IMAGE_HEIGHT,
    IMAGE_CHANNELS
)
print('x_train_with_chanels:', x_train_with_chanels.shape)
print('x_test_with_chanels:', x_test_with_chanels.shape)

# Normalizasyondan sonra renk kanal değerlerini görmek için 0.
görüntüden sadece bir satır kontrol edilir.
x_train_normalized[0][18]
```

Modeli oluştururken, Sıralı Keras modeli kullanılacaktır. Ardından iki çift Convolution2D ve MaxPooling2D katman olacaktır. MaxPooling katmanı, ortalama almak yerine bir bölgedeki maksimum değerleri kullanan bir tür alt örnekleme işlevi görür. Bundan sonra çok boyutlu parametreleri vektöre dönüştürmek için Flatten katmanı kullanılacaktır. Las katmanı, 10 Softmax çıkışlı bir yoğun katman olacaktır. Çıktı, ağ tahminini temsil eder. 0. çıkış, giriş basamağının 0 olma olasılığını temsil eder, 1. çıktı, giriş basamağının 1 olma olasılığını temsil eder ve bu şekilde devam eder.

```
# Model derlenir.
adam_optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)

model.compile(
    optimizer=adam_optimizer,
    loss=tf.keras.losses.sparse_categorical_crossentropy,
    metrics=['accuracy']
)

# Model eğitilir.
log_dir=".logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-
%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,
histogram_freq=1)

training_history = model.fit(
    x_train_normalized,
    y_train,
    epochs=10,
    validation_data=(x_test_normalized, y_test),
    callbacks=[tensorboard_callback]
)
```

Eğitim sırasında kayıp fonksiyonunun nasıl değiştiği görülebilir. Her bir sonraki iterasyonda eğitimde daha da küçülmesi beklenir (Görsel 3.40).

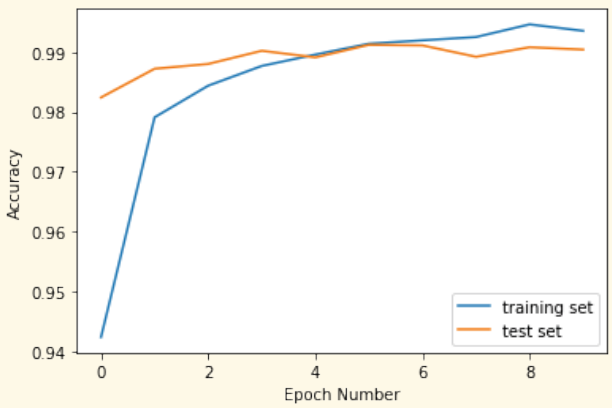
```
plt.xlabel('Epoch Number')
plt.ylabel('Loss')
plt.plot(training_history.history['loss'], label='training set')
plt.plot(training_history.history['val_loss'], label='test set')
plt.legend()
```



Görsel 3.40: Test ve eğitim setinde epok sayılarının değişimi

```
plt.xlabel('Epoch Number')
plt.ylabel('Accuracy')
plt.plot(training_history.history['accuracy'], label='training set')
plt.plot(training_history.history['val_accuracy'], label='test set')
plt.legend()
```

Modelin doğruluğu eğitim setinde ve test setinde karşılaştırılmalı ki doğruluğu değerlendirilebilsin (Görsel 3.41). Modelin her iki sette de benzer performans göstermesi bekleniyor. Bir test setindeki performans, bir eğitim setine kıyasla düşük olacaksa bu modelin fazla takıldığının ve "yüksek varyans" sorunu olduğunun bir göstergesi olacaktır.



Görsel 3.41: Test ve eğitim setinde epok sayılarının değişimi

```
%%capture
train_loss, train_accuracy = model.evaluate(x_train_normalized,
y_train)
print('Training loss: ', train_loss)
print('Training accuracy: ', train_accuracy)
%%capture
validation_loss, validation_accuracy = model.evaluate(x_test_
normalized, y_test)
print('Validation loss: ', validation_loss)
print('Validation accuracy: ', validation_accuracy)
```

Modeli kaydetme işleminde, tüm model bir HDF5 dosyasına kaydedilir. Dosyanın .h5 uzantısı, modelin Keras formatında HDF5 dosyası olarak kaydedilmesi gerektiğini belirtir. Bu modeli ön uçta kullanmak için ana beni oku dosyasında belirtildiği gibi tensorflowjs_converter kullanarak onu (bu not defterinde daha sonra) Javascript anlaşılabilir biçime (.json ve .bin dosyalarıyla tfjs_layers_model) dönüştürülmelidir.

Modeli kullanarak tahminler yapmak yani rakam tanıma işlemi ve yeni eğitilen modeli kullanmak için tahmin(predict) yöntemi çağrılmalıdır.

```
predictions_one_hot = loaded_model.predict([x_test_normalized])
print('predictions_one_hot:', predictions_one_hot.shape)
```

Her tahmin 0'dan 9'a kadar her sayı için bir tane olmak üzere 10 olasılıktan oluşur. En yüksek olasılığa sahip rakam seçilmelidir. Çünkü modelin en çok güvendiği rakam bu rakam olacaktır.

```
# olasılık dizileri biçimindeki tahminler.
# En yüksek olasılıklı tahminleri çıkartarak gerçekte hangi ra-
kamların tanındığı tespit edilir.
predictions = np.argmax(predictions_one_hot, axis=1)
pd.DataFrame(predictions)
pd.DataFrame(predictions_one_hot)
```

	0
0	7
1	2
2	1
3	0
4	4
...	...
9995	2
9996	3
9997	4
9998	5
9999	6

10000 rows × 1 columns

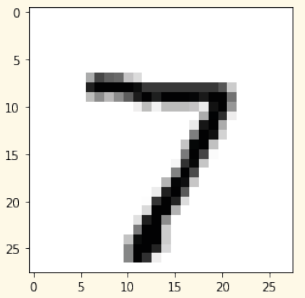
Görsel 3.42: Tahminler

Görsel 3.42 incelendiğinde model, test setindeki ilk örneğin 7 olduğunu tahmin eder.

```
#sonucu 7 olarak basıyor.
print(predictions[0])
```

Modelin tahmininin doğru olup olmadığını görmek için bir test setinden ilk görüntü yazdırılıp kontrol edilebilir (Görsel 3.43).

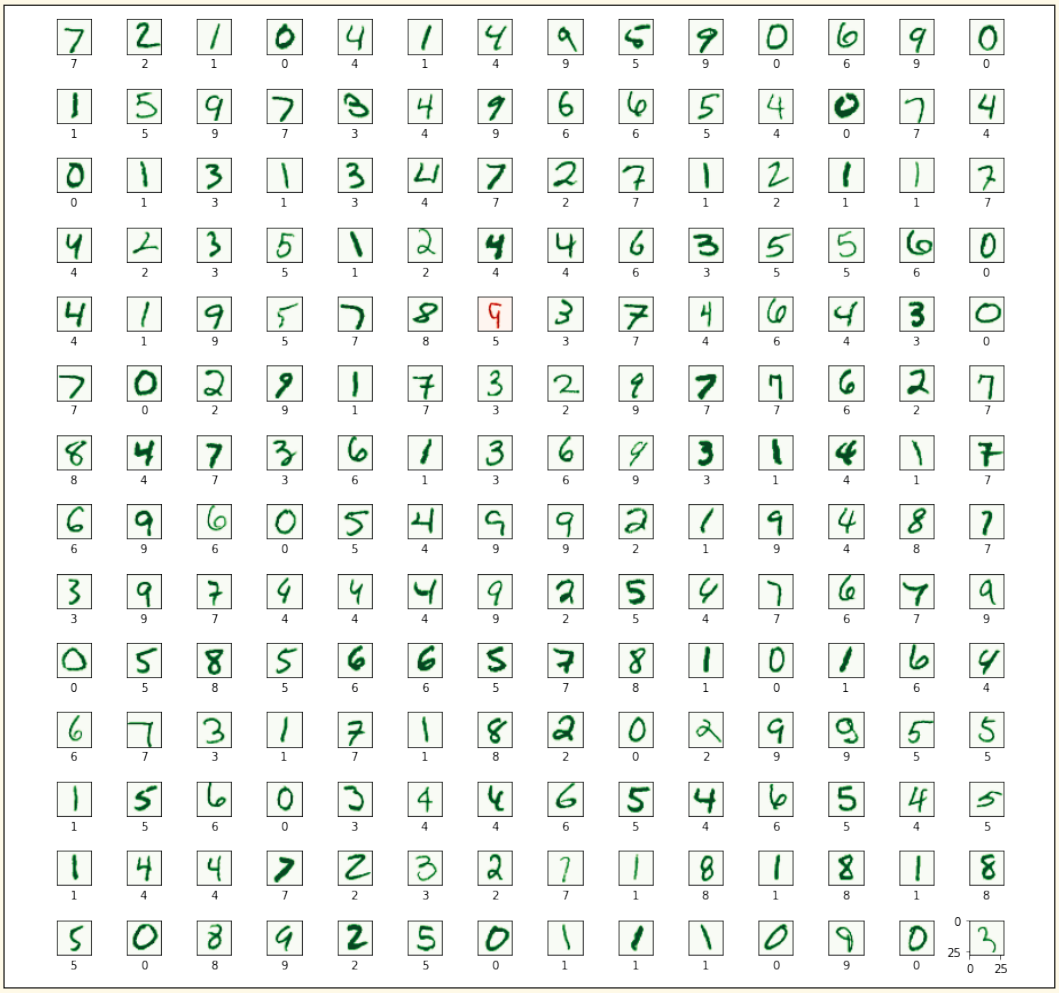
```
plt.imshow(x_test_normalized[0].reshape((IMAGE_WIDTH, IMAGE_HEIGHT)),
cmap=plt.cm.binary)
plt.show()
```



Görsel 3.43: Tahmin sonucu

Modelin doğru bir tahmin yaptığı ve 7 rakamını başarıyla tanıdığı görülebilir. Modelin nasıl performans gösterdiğini ve nerede hata yaptığını görmek için biraz daha test örneği ve karşılık gelen tahmin yazdırılabilir (Görsel 3.44).

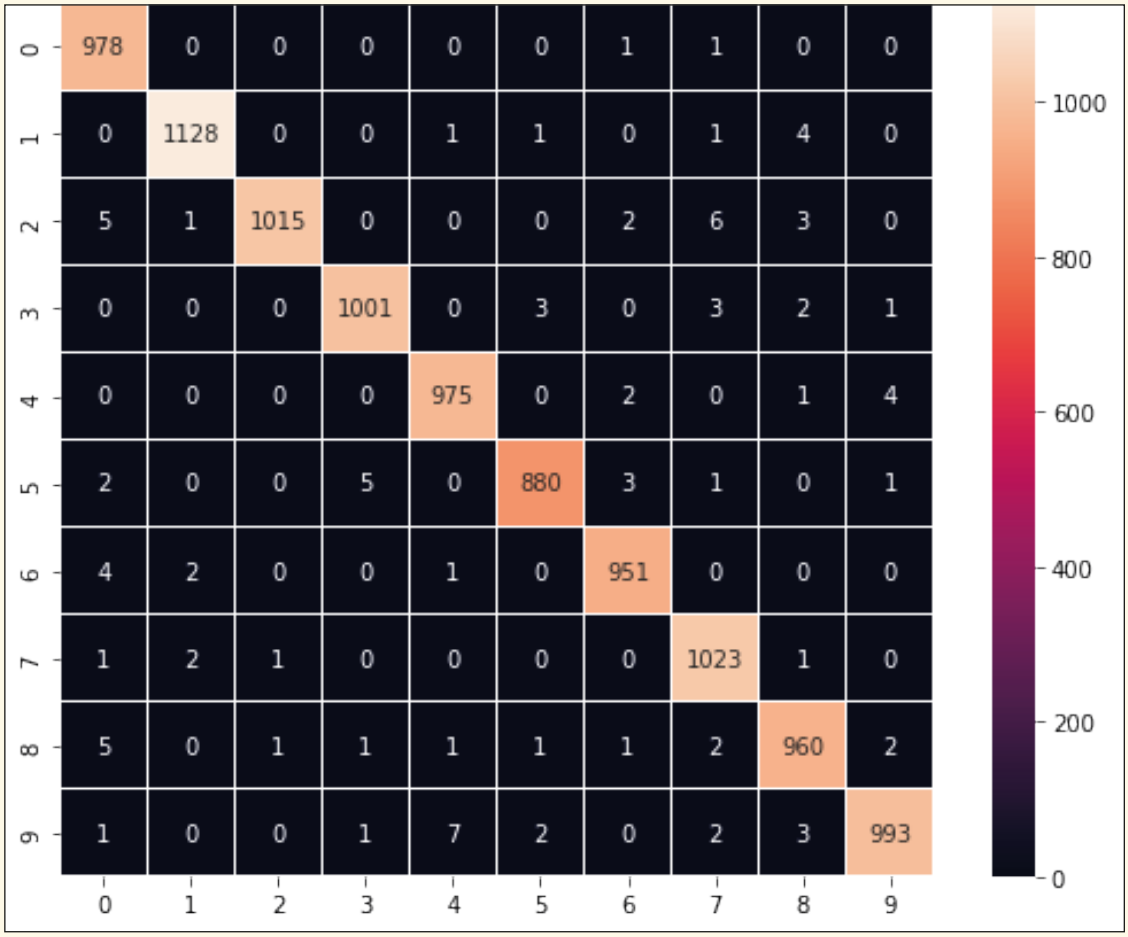
```
numbers_to_display = 196
num_cells = math.ceil(math.sqrt(numbers_to_display))
plt.figure(figsize=(15, 15))
for plot_index in range(numbers_to_display):
    predicted_label = predictions[plot_index]
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    color_map = 'Greens' if predicted_label == y_test[plot_index]
else 'Reds'
    plt.subplot(num_cells, num_cells, plot_index + 1)
    plt.imshow(x_test_normalized[plot_index].reshape((IMAGE_
    WIDTH, IMAGE_HEIGHT)), cmap=color_map)
    plt.xlabel(predicted_label)
plt.subplots_adjust(hspace=1, wspace=0.5)
plt.show()
```



Görsel 3.44: Diğer tahmin sonuçları

Karışıklık matrisi çizerek, model tarafından hangi sayıların iyi tanındığını ve modelin genellikle hangi sayıları doğru tanımak için karıştırdığını gösterir. Modelin gerçekten iyi performans gösterdiği görülebilir ancak bazen (10000 üzerinden 28 kez) 5 rakamını 3 ile veya 2 rakamını 3 ile karıştırabilir (Görsel 3.45).

```
confusion_matrix = tf.math.confusion_
matrix(y_test, predictions)
f, ax = plt.subplots(figsize=(9, 7))
sn.heatmap(
    confusion_matrix,
    annot=True,
    linewidths=.5,
    fmt="d",
    square=True,
    ax=ax
)
plt.show()
```



Görsel 3.45: Diğer tahmin sonuçları

6. UYGULAMA

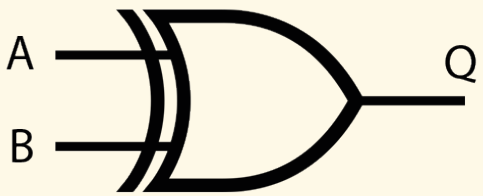
XOR Kapısının Çözümünü Öğrenecek Basit Bir Sinir Ağı Uygulaması

Xor kapısının çözümü yapay sinir ağları kullanılarak derin öğrenme ile çözülebilir . Bu uygulamada 2 adet giriş katmanı, 2 adet gizli katman ve 1 adet de çıkış katmanı bulunacaktır. Burada giriş ve gizli katman sayıları değiştirilebilir fakat çıkış Xor problemi çözümü olacağından tek olacaktır. Öğrenme algoritması olarak geri yayılım algoritması kullanılacaktır.

Matris matematiği için NumPy, çizimler için matplotlib ve yazdırmak için sys kütüphanelerini içe aktarınız.

```
import numpy as np # For matrix math
import matplotlib.pyplot as plt # For plotting
import sys # For printing
```

XOR mantık kapısı, verilen girişlerin sayısı tek olduğunda sonucu doğru, çift olduğunda ise yanlış döndürür (Görsel 3.46).



XOR Doğruluk Tablosu		
A	**B**	**Q**
0	0	0
0	1	1
1	0	1
1	1	0

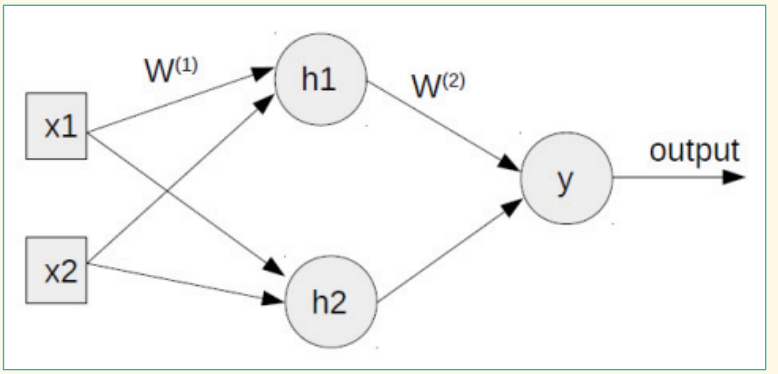
Görsel 3.46: Xor mantık kapısı ve doğruluk tablosu

Bu mantık kapısı sonuçlarını doğrulayacak aşağıdaki eğitim veri setini ve etiketlerini çalıştırınız.

```
# Eğitim verileri.
X = np.array([
    [0, 1],
    [1, 0],
    [1, 1],
    [0, 0]    ])
# Eğitim verileri için etiketler.
y = np.array([
    [1],
    [1],
    [0],
    [0]       ])
X ve y yazarak çalıştırıldığında giriş ve çıkış dizileri
oluşturulur.
Giriş, gizli ve çıkış katmanı sayıları belirtilir.
num_i_units = 2 # 2 tane giriş katmanı birimi
num_h_units = 2 # 2 tane gizli katman birimi
num_o_units = 1 # 1 tane çıkış katmanı birimi
```

```
# Dereceli geri yayılma (gradient descent) için
öğrenme oranı 0.01 olarak belirlenir.
learning_rate = 0.01

# Uyum göstermeye yardımcı parametre 0 olarak belirlenir.
reg_param = 0

# Geri yayılma algoritması için en fazla iterasyon sayısı 5000
olarak belirlenir. İterasyon sayısını değiştirerek sonuçları
gözlemleyebilirsiniz.
max_iter = 5000

# Eğitim örneği sayısı 4 olarak belirlenir.
m = 4
```

Yukarıdaki sayılar yapay sinir ağlarının doğru tahminler yapmak için öğrenmesi gereken sayılardır. Giriş katmanından gizli katmana yapılan bağlantılar için ağırlıklar ve önyargılar (bias) aşağıdaki gibi hesaplanır (Görsel 3.47).



Görsel 3.47: XOR problemi yapay sinir ağı yapısı

Yukarıda görülen çok katmanlı yapay sinir ağı yapılarında önce ileri doğru ağın çıktısı hesaplanır. Daha sonra geriye doğru hesaplama ile ağırlıklar değiştirilir. Eğitim sırasında sürekli yinelemeler ile bu işlemleri uygulayarak gerekli ağırlıklar ve çıkış değeri elde edilir.

```
np.random.seed(1)
W1 = np.random.normal(0, 1, (gizli katman sayısı, giriş katmanı
sayısı)) # 2x2
W2 = np.random.normal(0, 1, (çıkış_katmanı sayısı, gizli katman
sayısı)) # 1x2

B1 = np.random.random((gizli_katman sayısı, 1)) # 2x1
B2 = np.random.random((çıkış_katmanı sayısı, 1)) # 1x1
```

Normal dağılım kullanılarak ağırlıklar oluşturulacaktır.

W1 yazılıp çalıştırılır.

```
array([[ 1.62434536, -0.61175641], [-0.52817175, -1.07296862]])
```

Ağırlıkları elde edilir.

W2 yazılıp çalıştırılır.

```
array([[ 0.86540763, -2.3015387 ]])
```

Ağırlıkları elde edilir.

B1 yazılıp çalıştırılır.

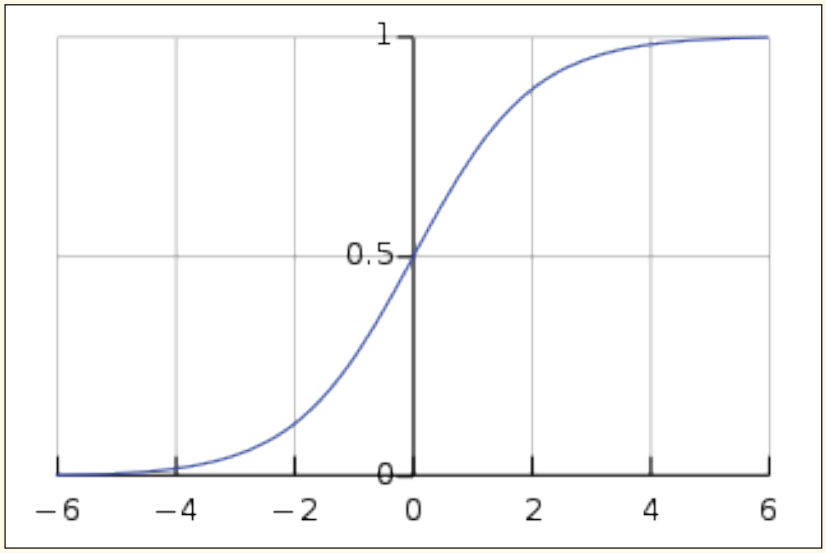
```
array([[0.41919451],0.6852195 ]])
```

Önyargılar elde edilir.
B2 yazılıp çalıştırılır.

```
array([[0.20445225]])
```

Önyargılar elde edilir.

Sigmoid aktivasyon fonksiyonu ile herhangi bir giriş 0-1 arasında herhangi bir değere eşlenir (Görsel 3.48).



Görsel 3.48: Sigmoid fonksiyonu grafiği

Aşağıda, uygulama true olarak ayarlanırsa giriş değerinin Sigmoid türevi (sigmoid işlevinin türevi) döndürecek bir boole eklenir. Bu değer daha sonra geri yayılımda kullanılır.

```
def sigmoid(z, derv=False):
    if derv: return z * (1 - z)
    return 1 / (1 + np.exp(-z))
```

İleri yayılım kodları aşağıdaki gibi yazılır ve çalıştırılır.

```
def forward(x, predict=False):
    a1 = x.reshape(x.shape[0], 1) # Getting the training example
as a column vector.

    z2 = W1.dot(a1) + B1 # 2x2 * 2x1 + 2x1 = 2x1
    a2 = sigmoid(z2) # 2x1

    z3 = W2.dot(a2) + B2 # 1x2 * 2x1 + 1x1 = 1x1
    a3 = sigmoid(z3)

    if predict: return a3
    return (a1, a2, a3)
```

Aşağıdaki değişkenler, ağırlıkları ve sapmaları güncellemek için kullanılacak olan gradyanları içerir.

```
dW1 = 0 # W1 türevi
dW2 = 0 # W2 türevi

dB1 = 0 # B1 türevi
dB2 = 0 # B2 türevi

# Geri yayılım işlemlerinde yapay sinir ağı maliyetini kaydeden
paramatre.
cost = np.zeros((max_iter, 1))
```

Yapay sinir ağını eğitmek için aşağıdaki kodlar yazılır.

```
def train(_W1, _W2, _B1, _B2):
    for i in range(max_iter):
        c = 0

        dW1 = 0
        dW2 = 0

        dB1 = 0
        dB2 = 0

        for j in range(m):
            sys.stdout.write("\rIteration: {} and {}".format(i
            + 1, j + 1))

            # Forward Prop.
            a0 = X[j].reshape(X[j].shape[0], 1) # 2x1

            z1 = _W1.dot(a0) + _B1 # 2x2 * 2x1 + 2x1 = 2x1
            a1 = sigmoid(z1) # 2x1

            z2 = _W2.dot(a1) + _B2 # 1x2 * 2x1 + 1x1 = 1x1
            a2 = sigmoid(z2) # 1x1
```

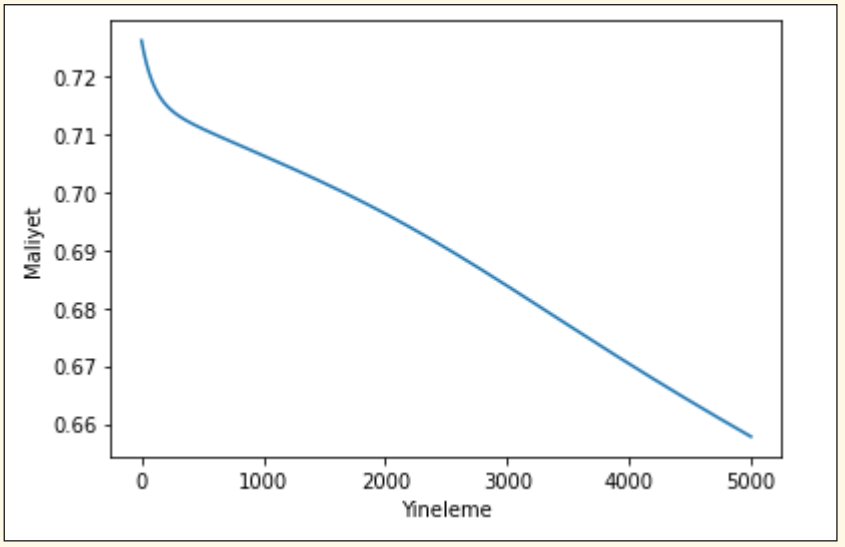
```
            # Back prop.
            dz2 = a2 - y[j] # 1x1
            dW2 += dz2 * a1.T # 1x1 .* 1x2 = 1x2

            dz1 = np.multiply((_W2.T * dz2), sigmoid(a1,
            derv=True)) # (2x1 * 1x1) .* 2x1 = 2x1
            dW1 += dz1.dot(a0.T) # 2x1 * 1x2 = 2x2

            dB1 += dz1 # 2x1
            dB2 += dz2 # 1x1

            c = c + (-(y[j] * np.log(a2)) - ((1 - y[j]) *
            np.log(1 - a2)))
            sys.stdout.flush()

        _W1 = _W1 - learning_rate * (dW1 / m) + ( (reg_param /
        m) * _W1)
        _W2 = _W2 - learning_rate * (dW2 / m) + ( (reg_param /
        m) * _W2)

        _B1 = _B1 - learning_rate * (dB1 / m)
        _B2 = _B2 - learning_rate * (dB2 / m)
        cost[i] = (c / m) + (
            (reg_param / (2 * m)) *
            (
                np.sum(np.power(_W1, 2)) +
                np.sum(np.power(_W2, 2))
            )
        )
    return (_W1, _W2, _B1, _B2)
```

Ağırlıklar ve önyargılar eğitim(train) fonksiyonuna gönderilerek sonuçlar alınır.

```
W1, W2, B1, B2 = train(W1, W2, B1, B2)

W1=array([[ 1.32260643, -0.42332921],[-1.4336158 , -1.67239068]])
W2=array([[ 0.25299514, -2.21317287]])
B1=array([[ 0.37348207],[-0.22080277]])
B2=array([[0.24523225]])
```

İterasyon arttıkça yapay sinir ağı eforunun yani maliyetin nasıl düştüğünü gösteren grafik çizdirilir (Görsel 3.49).

```
# Eksenler farklı elemanlara atanır.
plt.plot(range(max_iter), cost)

# x ekseni iterasyon(yineleme) olarak atanır.
plt.xlabel("yineleme ")

# y ekseni maliyet olarak atanır.
plt.ylabel("maliyet ")

# Çizim gösterilir.
plt.show()
```



Görsel 3.49: Yineleme maliyet grafiği

Siz de iterasyon sayısını arttırarak maliyetin düştüğünü gözlemleyebilirsiniz.

3. ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyunuz ve doğru seçeneği işaretleyiniz.

1. Aşağıdakilerden hangisi yapay sinir ağlarının özelliklerindendir?

A) Özetleme B) Hızlandırma C) Onaylama D) Kaydetme E) Sınıflandırma

2. Aşağıdakilerden hangisi derin öğrenme uygulamalarıyla geliştirilmez?

A) Bilgi erişimi
B) Veri tabanı işlemleri
C) Konuşma ve ses işleme
D) Nesne tanıma ve bilgisayarlı görü
E) Dil modelleme ve doğal dil işleme

3. Aşağıdakilerden hangisi yapılarına göre yapay sinir ağları gruplarından biri değildir?

A) Tek katmanlı algılayıcılar
B) Çok katmanlı algılayıcılar
C) İleri beslemeli yapay sinir ağları
D) Geri beslemeli yapay sinir ağları
E) Yukarı beslemeli yapay sinir ağları

4. Aşağıdakilerden hangisi bir aktivasyon fonksiyonu değildir?

A) Relu B) Tanh C) Sigmoid D) Toplama E) Doğrusal

5. Aşağıdaki yapay sinir ağlarından hangisi doğrusal olmayan aktivasyon fonksiyonu kullanamaz?

A) Tek katmanlı B) Çok katmanlı C) Gizli katmanlı D) İleri beslemeli E) Geri beslemeli

6. Ağın temel birimi aşağıdakilerden hangisidir?

A) Beyin B) Nükleus C) Nöron D) Akson E) Dentrit

7. Dendritlerin fonksiyonuna ne ad verilir?

A) Reseptör B) Verici C) Hem alıcı hem de verici
D) Aktifleme E) İletme

8. Sinapsta sinyal iletimi sürecinin özelliği aşağıdakilerden hangisidir?

A) Fiziksel B) Kimyasal C) Hem fiziksel hem de kimyasal
D) Elektriksel E) Manyetik

9. Nöronda kimyasal reaksiyonlar nerede gerçekleşir?

A) Dendritler B) Akson C) Sinapslar D) Nükleus E) Çekirdek

10. Aşağıdakilerden hangisi akson'un amacıdır?

A) Reseptör B) Verici C) İletim D) Seçme E) Sınıflandırma

KAYNAKÇA

- Bengio, Y. (2009). Learning deep architectures for AI. Foundations and Trends in Machine Learning 2(1), 1-127.
- Bengio, Y., P. Lamblin, D. Popovici and H. Larochelle (2007). "Greedy layer-wise training of deep networks." In NIPS'2006 . 14, 19, 200, 323, 324, 530, 532.
- Cho, Y., & Saul, L. K. (2009). "Kernel methods for deep learning". Advances in Neural Information Processing Systems, 342-350.
- Collobert, R., & Weston, J. (2008). "A unified architecture for natural language processing: Deep neural networks with multitask learning". International Conference on Machine Learning (ICML), 160-167.
- Deng, L., & Yu, D. (2014). "Deep learning: methods and applications". Foundations and Trends in Signal Processing, 7(3–4), 197-387.
- Doğan, O. (2016). Yapay Sinir Ağları
- Hinton, G. E., S. Osindero and Y.-W. Teh (2006). "A fast learning algorithm for deep belief nets." Neural computation 18(7): 1527–1554.
- Krizhevsky, A., I. Sutskever and G. Hinton (2012). "ImageNet classification with deep convolutional neural networks." In NIPS'2012 . 23, 24, 27, 100, 200, 371, 456, 460.
- LeCun, Y. (1987). Modèles connexionistes de l'apprentissage, Université de Paris VI. 18, 504, 517.
- Lecun, Y., Bengio, Y., & Hinton, G. (2015). "Deep learning". Nature 521(7553), 436.
- Lee, D. H. (2013). "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks". Workshop on Challenges in Representation Learning, ICML (3), 2
- McCulloch, W. S. and W. Pitts (1943). "A logical calculus of the ideas immanent in nervous activity." The Bulletin of Mathematical Biophysics 5(4): 115–133.
- Minsky, M. L. a. P., S. A. (1969). "Perceptrons." MIT Press, Cambridge. 15.
- Montufar, G. F. (2014). "Universal approximation depth and errors of narrow belief networks with discrete units." Neural computation 26(7): 1386–1407.
- Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., & Ng, A. Y. (2011). "Multimodal deep learning". International Conference on Machine Learning, 689-696.
- Pascanu, R., Ç. Gülçehre, K. Cho and Y. Bengio (2014). "How to construct deep recurrent neural networks." In ICLR'2014 . 19, 199, 265, 398, 399, 400, 412, 462.
- Ranzato, M., C. Poultney, S. Chopra and Y. LeCun (2007). "Efficient learning of sparse representations with an energy-based model." In NIPS'2006 . 14, 19, 509, 530, 532.
- Ravì, D., Wong, C., Deligianni, F., Berthelot, M., Andreu-Perez, J., Lo, B., & Yang, G. Z. (2017). "Deep learning for health informatics". IEEE journal of Biomedical and Health Informatics 21(1), 4-21.
- Rosenblatt, F. (1958). "The perceptron: A probabilistic model for information storage and organization in the brain." Psychological review 65(6): 386–408.
- Rosenblatt, F. (1962). Principles of Neurodynamics. Spartan, New York. 15, 27.
- Rumelhart, D. E., G. E. Hinton and R. J. Williams (1986). "Learning representations by back-propagating errors." Nature 323(6088): 533–536.
- Rumelhart, D. E., J. L. McClelland and T. P. R. Group (1986). Parallel Distributed Processing: Explorations in the Microstructure of Cognition, MIT Press, Cambridge. 17
- Widrow, B. and M. E. Hoff (1960). "Adaptive switching circuits." Adaptive switching circuits. In 1960 IRE WESCON Convention Record, volume 4, pages 96–104. IRE, New York. 15, 21, 24, 27.
- Efe, Abadoğlu, & Kaynak, 1999; Efe & Kaynak, 1999; Narendra & Parthasarathy, 1990
- [1]https://www.linkedin.com/pulse/yapay-sinir-a%C4%9Flar%C4%B1-ve-tek-katmanl%C4%B1-a%C4%9Flarda-%C3%B6%C4%9Frenme-tanju-do%C4%9Fan/
- [2]https://colab.research.google.com/notebooks/welcome.ipynb?hl=tr#scrollTo=5fCEDCU_qrC0

GENEL AĞ KAYNAKÇASI

- https://bilgeis.net/
- https://jupyter.org/install
- https://www.kaggle.com/
- https://matplotlib.org/
- https://numpy.org/
- https://pandas.pydata.org/
- https://pandas.pydata.org/docs/
- https://scikit-learn.org/stable/
- https://seaborn.pydata.org/
- https://stanford.edu/~shervine/l/tr/teaching/cs-229/cheatsheet-machine-learning-tips-and-tricks
- https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205
- https://www.datasciencearth.com/boosting-algoritmalari/
- https://www.scipy.org/
- https://www.veribilimiokulu.com/
- http://www.veridefteri.com/

GÖRSEL KAYNAKÇASI

http://kitap.eba.gov.tr/karekod/Kaynak.php?KOD=3174

CEVAP ANAHTARLARI

1. ÖLÇME VE DEĞERLENDİRME							
1	D	**6**	E	**11**	B	**16**	E
2	E	**7**	C	**12**	D	**17**	D
3	A	**8**	A	**13**	A	**18**	A
4	A	**9**	B	**14**	B	**19**	B
5	B	**10**	C	**15**	A	**20**	A

2. ÖLÇME VE DEĞERLENDİRME							
1	D	**11**	B	**21**	A	**31**	D
2	D	**12**	E	**22**	E	**32**	D
3	E	**13**	E	**23**	D	**33**	B
4	A	**14**	A	**24**	E	**34**	B
5	D	**15**	B	**25**	A	**35**	D
6	C	**16**	A	**26**	C	**36**	E
7	C	**17**	C	**27**	A	**37**	D
8	E	**18**	B	**28**	C	**38**	B
9	E	**19**	E	**29**	D		
10	B	**20**	C	**30**	D		

3. ÖLÇME VE DEĞERLENDİRME									
1	E	**2**	B	**3**	E	**4**	D	**5**	A
6	C	**7**	A	**8**	B	**9**	C	**10**	C