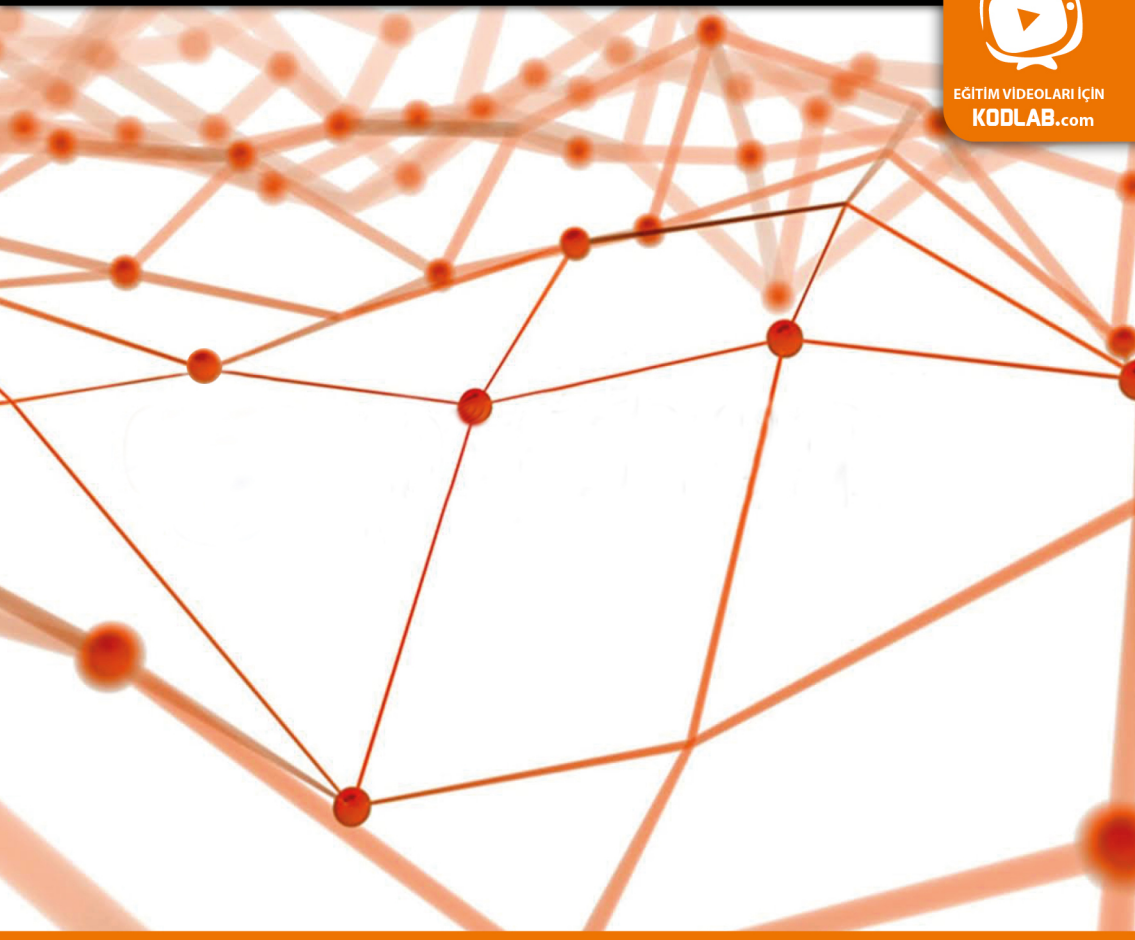
Caner ERDEN

EĞİTİM VİDEOLARI İÇİN
KODLAB.com



PYTHON
İLE VERİ MADENCİLİĞİ

KODLAB®
www.kodlab.com

PYTHON

İLE VERİ MADENCİLİĞİ

Caner ERDEN

KODLAB®
Yayın Dağıtım Yazılım ve Eğitim
Hizmetleri San. ve Tic. Ltd. Şti.

KODLAB® 270

PYTHON İLE VERİ MADENCİLİĞİ

Caner ERDEN

ISBN 978-625-7440-17-2
Yayıncılık Sertifika No: 13206

1. Baskı: 2021

Genel Yayın Yönetmeni
Gizem Atlı

Editör
İrem Soylu

Grafik Tasarım
Tamer Takmaz

Satış ve Pazarlama
Can Üstünel

Baskı: İnkılâp Kitabevi Yayın San. Tic. A.Ş. Sertifika No: 44066

KODLAB Yayın Dağıtım Yazılım ve Eğitim Hizmetleri San. ve Tic. Ltd. Şti.
15 Temmuz Mh. 1481. Sk. No : 44/A Bağcılar / İSTANBUL

tel: 0(212) 514 55 66
0(212) 514 66 61

web: www.kodlab.com
e-posta: bilgi@kodlab.com

Caner ERDEN

Caner Erden, hâlen Sakarya Uygulamalı Bilimler Üniversitesi (SUBÜ), Uygulamalı Bilimler Fakültesinde doktor öğretim üyesi olarak görev yapmaktadır. Dr. Erden, aynı zamanda SUBÜ yapay zekâ uygulama ve araştırma merkezinde araştırmacı olarak çalışmaktadır. Lisans eğitimini 2010 yılında İstanbul Ticaret Üniversitesi Endüstri Mühendisliği Bölümünde, yüksek lisans eğitimini ise 2013 yılında İstanbul Üniversitesi Endüstri Mühendisliği Bölümünde tamamlamıştır. Dr. Erden, iş hayatına 2012-2020 yılları arasında Sakarya Üniversitesi Endüstri Mühendisliği Bölümünde araştırma görevlisi olarak başlamıştır. Aynı bölümden 2019 yılında doktorasını "Dinamik Bütünleşik Süreç Planlama, Çizelgeleme Ve Teslim Tarihi Belirleme" başlıklı doktora teziyle almıştır. Doktora mezuniyetinden sonra Yöneylem Araştırması, Benzetim, Veri Madenciliği, Tedarik Zinciri Yönetimi ve Lojistik Yönetiminde Çok Kriterli Karar Verme Yöntemleri derslerini lisans ve yüksek lisans seviyesinde vermektedir. Dr. Erden'in araştırma alanları arasında proses planlama, makine çizelgeleme, meta sezgisel algoritmalar, optimizasyon, makine öğrenmesi ve veri madenciliği uygulamaları bulunmaktadır.

Eşime, kızıma ve oğluma…

ÖN SÖZ

Endüstride ve akademide son zamanların en çok ilgi çeken ve popüler konusu olan veri madenciliği üniversitelerde birçok mühendislik bölümünde bir dönemlik bir ders olarak gösterilmektedir. Bazı bölümlerde bu ders seçimli ders olarak gösterilse de özellikle bilgisayar ve endüstri mühendisliği bölümlerinde veri madenciliği dersi temel derslerden birisi olarak görülmektedir. Bu kitap, endüstri mühendisliği bölümünde verilen veri madenciliği dersinde sunulan ders notlarından hazırlanmıştır. Kitabın her bir bölümü 2-3 saatlik bir derste gösterilebilecek içeriklerden oluşmaktadır.

Veri madenciliği birçok disiplinle düşünülmesi gereken matematik ve istatistiğin bir alt bilimi olarak görülebilir. O nedenle veri madenciliği biliminden önce matematik, istatistik ve bilgisayar bilimlerinin temellerinin bilinmesi zaruridir. Kitap, bilgisayar programlamanın detay yapılarından bir bölüm olarak bahsetmektedir. Ancak kitap, okuyucunun bilgisayar programlama anlamında temel bilgilere sahip olduğu varsayılarak hazırlanmıştır. Temel düzeyde bilgiden kasıt, algoritmaların akış diyagramlarının bilinmesi, değişken tanımlama, fonksiyon tanımlama, metot çağırma gibi süreçlere aşinalıktır. Kitapta programlama dili olarak diğer dillere göre araştırmacılar tarafından en fazla tercih edilen dil olan Python belirlenmiştir. Veri madenciliği uygulamaları için geliştirilmiş paket programlar bulunsa da daha esnek ve araştırmacının hemen hemen bütün istediklerini yapabilmesi nedeniyle programlama dilleri hala uygulama geliştirmede en fazla kullanılan araçlardandır. Bu kitapta Python programlama dili içerisindeki güçlü kütüphanelerden olan NumPy, Pandas, Matplotlib, Scikit-Learn gibi kütüphanelerin verimli kullanımı, dönüşümleri ve veri işleme yöntemleri gibi özelliklerine ağırlık verilmiştir.

Veri madenciliğinin kalbinde veri vardır. Bir altın madencisinin altını kazarak çıkarması gibi veri madenciliğinde de veri içerisinde kullanılabilir olan bilgi ortaya çıkarılmaya çalışılır. Bu amaç doğrultusunda veri madenciliği disiplini içerisinde birçok algoritma ve teknik bulunmaktadır. Veri madenciliğinde kullanılan teknikler ile kimi zaman mevcut verilerden gelecekte olması muhtemel olayları tahmin etmek kimi zaman da veri içerisindeki gizli örüntü ve bilgileri çıkarmak amaçlanır. Kullanılan bu teknikler genellikle karmaşık matematiksel bağlantılara ve denklemlere sahiptir. Bu kitapta tekniklerin derin ve karmaşık matematiğinden ziyade uygulama boyutuna ağırlık verilecektir. Çünkü endüstri mühendisliği bölümünde lisans düzeyinde okutulacak bir ders için uygulamaların daha fazla gösterilmesi uygun olacaktır. Eğer ders matematik, istatistik veya bilgisayar mühendisliği gibi bölümlerde verilecek olursa veri madenciliği tekniklerinin daha fazla matematik ve bilgisayar programlama detaylarını içerecek şekilde ele alınması gerekebilir. Ya da veri madenciliği dersi lisansüstü bir düzeyde verilecek olursa tekniklerin arkasında yatan matematik, istatistik ve bilgisayar bilimleri alanları daha detaylı incelenebilir. Birçok endüstri mühendisi veri madenciliğinin uygulanması ile ilgilenir. Var olan tekniklerden uygun birisini kullanarak veri analizi ve tahmini çalışmalarını gerçekleştirir. Bu nedenle bu kitapta veri madenciliğinin uygulanması için gerekli çalışmalara daha çok yer verilmiştir.

Kitaptaki anlatım veri madenciliğine giriş seviyesinde olan öğrenciler ve araştırmacılar için bilgiler taşımaktadır. Veri madenciliği serüvenine Python ile Veri Madenciliği kitabı ile başladığınız için teşekkür ederim. Faydalı olması dileğiyle...

İÇİNDEKİLER

BÖLÜM 1

VERİ MADENCİLİĞİ KONUSUNA GİRİŞ

Bu bölümde veri madenciliği konusuna kısa giriş yapılacak ve ardından uygulama geliştirme ortamlarının tanıtımı yapılacaktır. Ayrıca uygulama geliştirme ortamı olarak seçilen Anaconda programının nasıl yükleneceğinden ve nasıl kullanılabileceğinden bahsedilecektir.

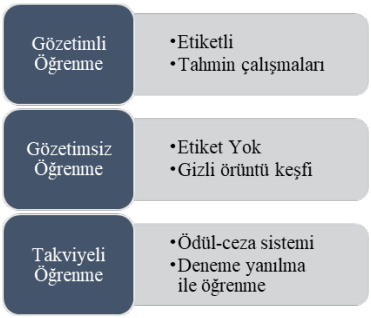
Veri Madenciliği

Kısa ve genel tanım olarak **veri madenciliği,** veri üzerinde madencilik yapmak demektir. Veri madenciliğinde veri setleri içerisinde bulunan ve genellikle hemen görülemeyen bilgilerin ortaya çıkarılarak veriye dayalı karar verme ve gelecekteki olayları tahmin etme çalışmaları gerçekleştirilir. Karar verme ve tahmin çalışmaları için geliştirilmiş birçok teknik vardır. Bu teknikler kullanılarak geliştirilen modeller veri madenciliğinin bir çıktısıdır. Veri madenciliği modelleri ekonomi, sağlık, eğitim, psikoloji, yönetim ve organizasyon gibi birçok disiplinde daha iyi kararlar vermek ve daha iyi tahminler yapabilmek amacıyla kullanılmaktadır. Veri madenciliği günümüzde verilerin çok hızlı bir şekilde üretilmesi, depolanması ve tüketilmesi nedeniyle şirketlerin başarısı için hayati bir noktaya gelmiştir. Üretilen bu veriler genellikle sensörler, bilgisayar sistemleri gibi otomatik sistemler vasıtasıyla alınır. Alınan veriler genellikle ilk hâliyle bir anlam ifade etmez. Verilerin anlamlı bilgilere dönüştürülmesi için bazı süreçlere ihtiyaç vardır. Verilerin işlenmesi ve bilgiye dönüştürülmesi için yüksek işlemci gücüne sahip bilgisayarların kullanılması gereklidir. Ancak bu durum günümüzde önemli bir sorun teşkil etmemektedir. Çünkü şimdiki günlük bilgisayarlarda dahi yeterli güce sahip işlemcilere sahip olunabilmektedir. İşte verilerden anlamlı bilgiler çıkarılması süreçlerinin tamamı yine veri

madenciliğinin bir başka tanımı olarak söylenebilir. Şirketler bu anlamlı bilgileri veri madenciliği modellerini kullanarak üretirler. Aldıkları kararları ve gerçek dünyadaki problemlerin çözüm yöntemlerini, ürettikleri bilgiler ışında araştırarak daha sağlıklı bir yönetim gerçekleştirmiş olurlar.

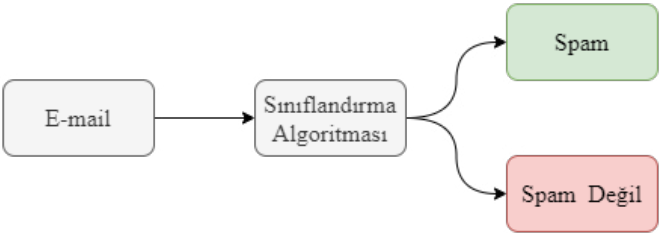
Veriden anlamlı bilgi üretebilmek için neler nasıl yapılmalıdır, sorusunun cevabı veri madenciliğinin temel sorularındandır. Bu sorunun cevabı için öncelikle eldeki problemin **iyi tanımlanmış** olması gerekir. Problemin tanımında, *eldeki verilerden çıkarımlar ile gelecekte olabilecek durumları nasıl tahmin edebiliriz,* sorusunun nedenleri verilmelidir. Yine *problem tanımında kurulan modelin varsayımları nelerdir, sınırlılıkları, avantajları ve dezavantajları nelerdir,* gibi soruların da cevapları verilmelidir. Sonuç olarak veri madenciliği modelini uyguladıktan sonra nasıl bir çıktı almanın hedeflendiği yine problemin tanımlaması aşamasında belirtilmelidir. Veri madenciliği modellerinde çıktı genellikle bir tahmin sonucu ya da modellenmiş bir veri setidir. Tahmin probleminde *mevcut veriler içerisinde nasıl bir ilişki kurulabilir, bu ilişkiden faydalanılarak gelecekteki veriler nasıl tahmin edilebilir ve tahmin edilen veriler ne kadar kalitelidir,* gibi sorulara cevap aranır. Örneğin bir sağlık kuruluşunda bir hastaya uygulanan tedavi, sonra gelecek aynı şikayet sahibi hastalara da uygulanırsa aynı sonuçlar alınabilir mi, diye bakıldığında **bir tahmin problemi** ele alınmış olunur. İlgili sağlık kuruluşunda hastaların durumlarını gösteren veriler bir yerde saklanırsa bu veriler kullanılarak bir veri madenciliği çalışması yapılabilir. Çalışma sonucunda çıktıların ne kadar kaliteli olduğunu göstermek için modelin test edilmesi gereklidir. Bu aşamada, model verilen verileri ne kadar açıklayabiliyor ve gelecekteki verileri ne kadar doğru tahmin edebiliyor diye bakılır. Modelin testi ise modele verilen ilk verilerden farklı olmalıdır. Böylece iki adet veri seti ile çalışmak gerektiği ortaya çıkmaktadır. Bu iki veri setinden modelin eğitimi için kullanılan sete **eğitim**, modelin testi için kullanılan sete ise **test veri seti** denilir.

Veri madenciliğinde diğer önemli kavram ise yapay öğrenme kavramıdır. Makine öğrenmesi olarak da bilinen bu kavram ile önceki verilerden gelecekteki veriler tahmin edilir. Yapay öğrenmenin temel olarak üç çeşidi bulunmaktadır. Bunlar şekilde de gösterildiği gibi **gözetimli öğrenme** (supervised learning), **gözetimsiz öğrenme** (unsupervised learning) ve **takviyeli öğrenme** (reinforcement learning) çeşitleridir.



Şekil 1-1: Yapay öğrenme çeşitleri

Gözetimli öğrenmeye verilebilecek en güzel örneklerden birisi spam e-posta örneğidir. **Şekil 1-2**'de bir e-posta geldiğinde spam ya da spam değil diye iki çeşide ayrıldığı gösterilmektedir. Daha önceden gelen e-postalar veri madenciliği algoritmasına verilir. Algoritmanın görevi yeni gelecek e-postaların spam olup olmadığını belirlemektir. Bunun için geçmişteki e-postalardan hangilerinin spam olduğunun veya spam olmadığının belirtilmesi gereklidir. Algoritma, mevcut veriler içerisindeki ilişkileri çözerek yeni gelecek e-postanın sınıfına karar verecektir. Bu problem literatürde sınıflandırma problemi olarak geçer. Sınıflandırma problemlerinde bir durumun hangi sınıfa ait olduğuna karar verilir. Spam e-posta örneğinde 2 adet sınıf bulunur ve bu tip problemler **ikili çıktıya sahip problemler** olarak isimlendirilirler. İkiden fazla sınıf olması durumunda çok çıktılı sınıflandırma problemleri söz konusu olur. Eğer çıktı değişkeni örnekteki gibi kesikli verilerden değil de sürekli verilerden oluşursa problem regresyon problemine dönüşür. Regresyon problemlerinde sürekli verilerin tahmini yapılır. Örneğin bir evin satış fiyatının tahmini için bir regresyon modeli kurulabilir. Çünkü evin satış fiyatı sürekli bir veridir. Veri tipleri konusunda daha detaylı bilgi üçüncü bölümde verilecektir.



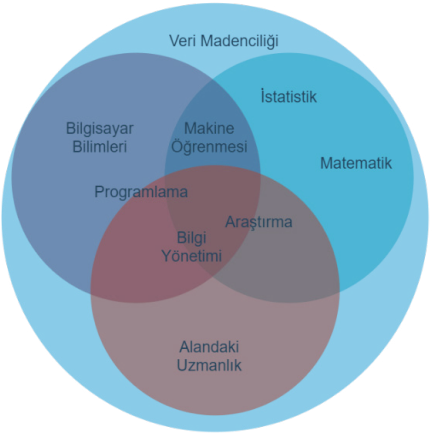
Şekil 1-2: Spam e-posta örneği

Diğer yapay öğrenme çeşidi de gözetimsiz öğrenmedir. Bu öğrenme çeşidinde, algoritma önceki verilerin sınıflarını bilmeden çalışır. Yani başlangıçta algoritmaya verilen bir hedef değişken yoktur. Algoritma, veri seti içerisindeki örüntülerden, bağlantılardan veya çeşitli ilişkilerden yola çıkarak bir süreç çalıştırır. Gözetimsiz öğrenme tipine örnek olarak **kümeleme** ve **boyut azaltma** çalışmaları verilebilir. Kümelemede benzer veriler bir araya getirilerek bir grup oluşturulur ve etiketler belirlenmeye çalışılır. Makine öğrenmesi bölümünde bu alanlarla ilgili daha detaylı bilgiler verilecektir.

Genel anlamda veri madenciliği bu şekilde özetlenebilir. Veri madenciliği alanında kendini geliştirmek isteyen araştırmacılara aşağıdaki gibi öneriler sunulacaktır.

Veri Madenciliği Yol Haritası

Veri madenciliği ile ilgili internette bulunan kaynaklar araştırıldığında nereden başlanılacağına dair bir kafa karışıklığı söz konusu olabilir. İnternette paylaşılan gerekli gereksiz bilgilerden bir yol haritası çıkarmak ve odaklı çalışmak mümkün değildir. Burada anlatılanlar belirli bir aşamada öğrenilirse veri madenciliği alanına giriş yapılmış olunur. Ancak yine de bir veri madencisi olmak için öğrenmek yeterli değildir. Öğrenilen bilgilerin uygulamaları da bir o kadar önemli olacaktır. Toplamda 8 adımdan oluşan yol haritasının hazırlanmasında kullanılan yaklaşım **Şekil 1-3**'teki diyagramda gösterilmiştir. Şekilde veri madenciliği biliminde; bilgisayar bilimleri, matematik ve istatistik bilgisi ve alanda uzmanlık alanları 3 önemli bileşen olarak gösterilmiştir.



Şekil 1-3: Veri madenciliği bilgi alanları[1]

Bilgisayar Bilimleri

Veri madenciliği araştırmacıları genellikle bilgisayar mühendisliği ya da bilgisayar bilimleri ile ilgilenen araştırmacılardır. Çünkü bilgisayar programlama bilgisi veri madenciliğinde önemli bir yer tutar. Her ne kadar günümüzde birçok paket program ile veri madenciliği uygulamaları geliştirilebiliyor olsa da veri madenciliği uygulamalarının bazı noktalarında araştırmacının kod yazma ihtiyacı olacaktır. Programlama dilleri arasında en önde gelen seçenek Python olmakla birlikte R, Java, JavaScript, Julia, MATLAB gibi diller ile de önemli sayıda çalışma yapılmaktadır.

Matematik ve İstatistik Bilgisi

Matematik bilgilerinin içerisine temel matematik (calculus), lineer cebir, diferansiyel denklemler gibi konular girmektedir. Veri madenciliği algoritmalarının temelinde bahsedilen matematik konularının olduğu bilinmelidir. Aynı şekilde istatistik bilgisi de veri madenciliği çalışmalarının olmazsa olmazıdır. Veri madenciliği çalışmaları istatistiğin bilgisayarda uygulanması olarak görülmektedir[2]. **Cao** veri bilimi için aşağıdaki denklemi paylaşmıştır[3].

Veri bilimi

= (istatistik + bilişim + bilgisayar + iletişim + sosyoloji + yönetim) | (veri + ortam + düşünme).

Veri madenciliği için kullanılacak istatistik konuları, hipotez testleri[4], sınıflandırma ve kümeleme, regresyon analizi, zaman serisi analizi, istatistiksel modelleme, çıkarımsal istatistik gibi konulardan oluşmaktadır[5]. Belirtilen konular dışında istatistik ile ilgili ne kadar derin bilgi sahibi olunursa veri madenciliği çalışmalarına o kadar faydası olacaktır. Çünkü yeni yöntemlerin geliştirilebilmesi istatistik ve matematik bilimlerindeki bilgi birikimine sıkı sıkıya bağlıdır.

Veri Önişleme

Verilere takla attırabilme aşamasıdır. **Veri manipülasyonu** olarak da bilinir. Veri önişleme adımı için Python kütüphanelerinden NumPy ve Pandas kütüphanelerinin iyi bilinmesi önerilir. Ayrıca herkesin günlük ortalama işlerinde sıklıkla kullandığı Microsoft Excel de bu alanda yardımcı olacaktır. Gerçek hayattaki verilerin analiz aşamasına getirilebilmesi için veri önişleme aşaması da önemli bir alan olarak görülmelidir. Veri önişleme çalışmaları hakkında detaylı bilgi ilerleyen bölümlerde verilecektir.

Veri Görselleştirme

Veri görselleştirme çalışmaları verileri özet hâlde sunmak için kullanılır. Bu alanda özellikle Tableau (*https://www.tableau.com/*), Datawrapper (*https://www.datawrapper.de/*), D3.js (*https://d3js.org/*), Matplotlib, Seaborn, Bokeh vb. gibi programlar üst düzey denilebilecek görselleştirme araçları sunmaktadır.

Veri Tabanı Yönetimi

Veri madenciliği çalışmalarında veri tabanı veya veri ambarlarından veri okunabilmeli ve veri tabanlarına veri kaydedebiliyor olunmalıdır. SQL, mongo, Postgre gibi veri tabanı yönetimlerinin bilinmesi önemlidir. Günümüzde SQL veri tabanları yeterli gelmemektedir. Verilerin boyutları, hacimleri ve kaynakları hızlı bir şekilde artmaktadır. Bu nedenle büyük verileri işleyebilecek araçların kullanılmasına ihtiyaç olacaktır. Bu alanda Hadoop Platformu ve Apache Spark gibi platformlar kullanılabilir.

Makine Öğrenmesi

Veri madenciliğinin yol haritasında sırada makine öğrenmesi algoritmaları gelmektedir. Makine öğrenmesinde veya yapay öğrenmede gözetimli öğrenme, gözetimsiz öğrenme ve takviyeli öğrenme kavramları önemli bir yer tutmaktadır. Günümüzde yapay sinir ağları ile geliştirilen **derin öğrenme** çalışmaları en yoğun çalışma alanlarındandır. Derin öğrenme ile özellikle görüntü işleme konusunda ve yine güncel çalışma alanlarından sürücüsüz otomobiller gibi ürünler üretilebilmektedir. Bunun yanı sıra literatürde bilinen makine öğrenmesi algoritmalarından karar ağaçlarını, destek vektör makinelerini, lojistik regresyonu, regresyon analizleri de bu kapsamdan değerlendirilmektedir.

Problem Çözme

Veri madenciliğinin süreci bir problem üzerinde düşünerek başlar. O nedenle uygulamalara gerçek hayat problemini çözmeye odaklanarak başlamak gerekir. Öğrenilen algoritma ve yöntemler gerçek hayattaki problemlerin çözümü için kullanıldığı ölçüde değerlidir. Bu anlamda Kaggle *https://www.kaggle.com/* gibi veri madenciliği çalışmalarının tartışıldığı ve yapıldığı ortamların takip edilmesi gerekmektedir.

Tekrar ve Takip

Bilgisayar bilimlerinde güncel gelişmeleri takip ve tekrar etmek gerekir. Takip edilebilecek bazı adresler aşağıda listelenmiştir.

- **Cebir, Matematik Alanında Eğitim:** *https://tr.khanacademy.org/math/algebra*
- **Veri madenciliği için listelenmiş 10 adet kitap**: *https://hackr.io/blog/data-science-books*
- **Python programlama dili öğrenimi için kitap:** Learning to program with Python 3[6]
- **Takip edilebilecek blog sayfası**: Towards Data Science *https://towardsdatascience.com/*

Veri madenciliği için çeşitli YouTube kanalı önerileri:

- **3Blue1Brown** *https://www.youtube.com/channel/UCYO_jab_esuFRV4b17AJtAw*
- **Simplilearn** *https://www.youtube.com/user/Simplilearn*
- **sentdex** *https://www.youtube.com/c/sentdex/playlists*
- **StatQuest** *https://youtube.com/playlist?list=PLblh5JKOoLUK0FLuzwntyYI10UQFUhsY9*

Soru - Cevap - Tartışma siteleri:

- **Stack Overflow** - *https://stackoverflow.com/*
- **Quora** *https://www.quora.com/*
- **GitHub**: *https://github.com/*

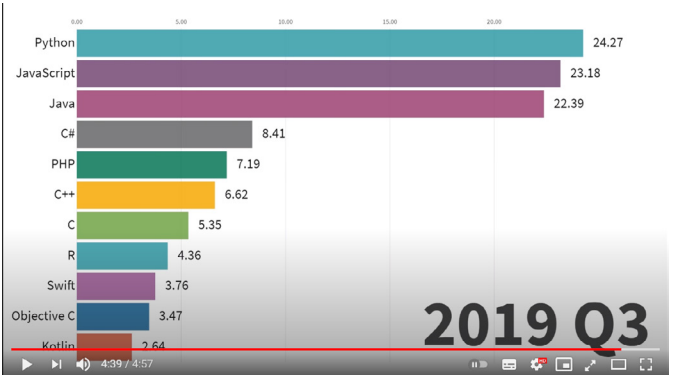
Yol haritası hazırlanırken yararlanılan diğer kaynaklar [7–9] referanslarında verilmiştir.

Gerekli Programlar

Bu kitaptaki tüm örnek uygulamalar Python programlama dili ile geliştirilecektir. Algoritmaların geliştirilmesi için NumPy, Pandas, Matplotlib, Scikit-Learn gibi kütüphaneler kullanılacaktır. Bu bölümde kitaptaki anlatımları takip etmek isteyen kişiler için bilgisayara kurulması gereken programlardan ve bu programların neden seçildiğinden bahsedilecektir.

Neden Python Programlama Dili?

Öncelikle neden Python programlama dilinin seçildiğinden bahsedilmelidir. Python programlama dili günümüzde en çok tercih edilen programlama dilidir. Python programlama dilinin kullanım yüzdelerinin nasıl arttığını görüntülemek için şu video takip edilebilir *https://www.youtube.com/watch?v=Og847HVwRSI*. Ayrıca videodan bir görüntü **Şekil 1-4**'te verilmiştir. Buna göre 2019 yılı üçüncü çeyreğinde GitHub depolarında kullanılan programlama dillerinin kullanım yüzdelerine bakıldığında Python %24 ile ilk sırada yer almıştır.



Şekil 1-4: 2019 yılı üçüncü çeyrekte programlama dillerinin kullanım oranları

Python programlama dilinin yaygın kullanımının nedenleri şu şekilde sıralanabilir:

1. Uygulaması ve öğrenmesi kolaydır.
2. Çok sayıda kullanan ve destekleyen topluluklarına sahiptir.
3. Tanınmış kurumsal sponsorlardan destek vardır.
4. Yüzlerce Python paketi ve kütüphanesi açık erişim olarak sunulmaktadır.
5. Çok yönlülük, verimlilik, güvenilirlik ve hız özelliklerini barındırmaktadır.
6. Büyük veri, makine öğrenmesi ve bulut bilişim alanlarında kullanım kolaylığı sağlar.
7. En çok tercih edilen programlama dillerinden birisidir.
8. Python programlama dilinin yapısı esnektir.
9. Akademide ve endüstride Python kullanımı yaygınlaşmaktadır.
10. Otomasyon ile çalışabilme özelliği bulunur.

Tüm bu nedenlerden dolayı günümüzde veri madenciliği için kullanılması gereken programlama dilinin Python olması en mantıklı seçimlerden birisi olacaktır. Bununla birlikte, bazı durumlarda paket programlar ile veri madenciliği uygulamaları geliştirmek daha kısa zaman alabilir. Bir programlama dili ile araştırmacının kendi yazdığı kodlar üzerinde çalışma geliştirmesinin en önemli artısı çalışmanın istenildiği kadar esnek hâle getirilmesidir. Ayrıca daha önceden bahsedilen tüm avantajlar ile yazılan kodlar sayesinde veri madenciliği uygulaması yapmak araştırmacılara uygulamaları istedikleri gibi müdahale imkânı kazandıracaktır.

Python programlama dili gibi veri madenciliğinde sıklıkla kullanılan başka programlama dilleri de mevcuttur. Bir veri madenciliği uzmanı programlama dillerinin hepsine hakim olamayabilir. Programlama dilinin yapısına, yazım kurallarına veya ortamına hakim olmayınca ortaya gereksiz zaman kayıpları çıkabilir. Her ne kadar Python ilk sırada da gelse veri madenciliğinde kullanılan diğer programlama dillerine örnek olarak **JavaScript, R, Julia, Scala, Swift, MATLAB, Octave** programlama dilleri verilebilir.[10]

Bu kitapta Python programlama dili ile veri madenciliği uygulamalarının aşamalarından ve tekniklerinden bahsedilecektir. Kitapta öncelikle veri madenciliği için öğrenilmesi tavsiye edilen temel Python kütüphanelerinin kullanımı ile ilgili bilgi verilecektir. Bu kütüphaneler **NumPy, Pandas, Scikit-Learn** ve **Matplotlib** kütüphaneleridir. Bunlarla birlikte görsel ara yüzü ile kullanım kolaylığı sağlayan Orange paketinin kullanımı da gösterilecektir. Kitabın ana amacı programlama dili yapısını ya da kütüphanelerin sağladıkları tüm kavramları ve imkânları göstermek ve anlatmak değildir. Bunun yerine, veri madenciliği uygulamalarında kullanılacak şekilde programlama dili ve içerisindeki kütüphaneler nasıl uyarlanabilir diye bakılacaktır. Bir metafor ile söylemek gerekirse bu kitapta arabanın kullanılması öğretilecektir, araba motorunun nasıl çalıştığı ya da motorun mühendislik hesaplaması gerektiren bilgiler gösterilmeyecektir. Araba motorunun nasıl çalıştığını ince hatları ile bilinmeyebilir ama yine de iyi bir sürücü olunabilir. Çoğu kişi motor, kaporta, iç aksam, elektrik devreleri vb. gibi alanlarda teknik bilgilere sahip değildir ama herkesin araba kullanmaya ihtiyacı var. Her ne kadar motor bilgisi üst düzey olursa arabaya daha özenli bakılması söz konusu olsa da ortalama bir kullanıcı için bu kadar detay bilginin gereksiz olduğu söylenebilir. Bu metaforda olduğu gibi kitaptaki ana motivasyonda programlama dilinin incelikleri değil, veri madenciliğinin nasıl adapte edileceğini göstermek olacaktır.

Python programlama dilinin kurulumu için birden fazla yöntem bulunmaktadır. Veri madenciliği uygulamalarında kullanmak için Python kurulumunu Anaconda programı ile yapmak daha kolay bir seçenek olacaktır. Kitapta tercih edilen çalışma ortamı takip eden bölümde verilmiştir.

Çalışma Ortamı

Bu kitapta kullanılan tüm kodlar Python 3 programlama diliyle **Jupyter** ortamında geliştirilecektir. Jupyter ortamı sayesinde interaktif bir kod geliştirme ortamına sahip olunabilir. Yani yazılan kodların çıktılarını interaktif bir şekilde görülebilir. Jupyter ve Python 3 (şu anki versiyon 3.8) kurulumu ayrı ayrı yapılabileceği gibi Anaconda programı sayesinde tüm ihtiyaç duyulan programlar tek seferde kurulabilir. Diğer türlü, her programın ayrı ayrı kurulması gereklidir. Bireysel programların kurulumu için önce Python 3 bilgisayara kurulur. Ardından Jupyter ortamı ve diğer kullanılacak paketler kurularak kurulumlar tamamlanır. Anaconda programı ile tüm paketlere tek seferde ulaşılır. Anaconda ile ayrıca R, JavaScript gibi programlama dillerinde de yazılım geliştirilebilir. Anaconda dağıtımı ile bilgisayara kurulan paketler ve programlar şunlardır:

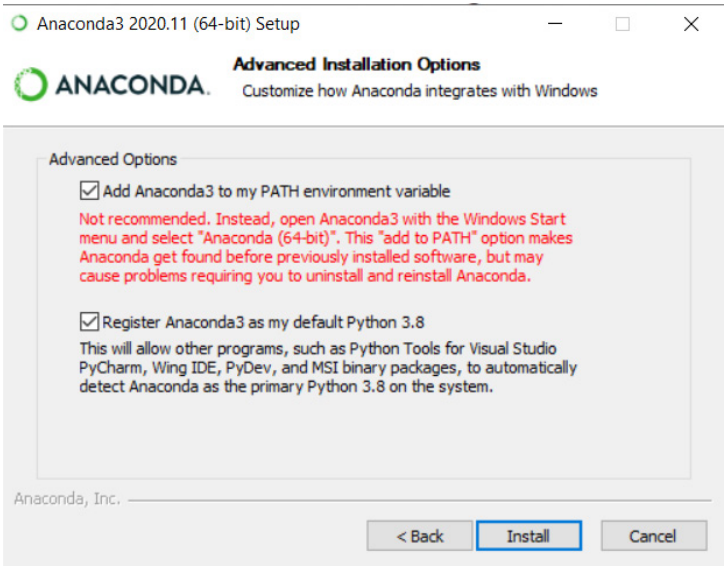
1. Jupyter notebook, Spider, Visual Studio Code gibi çalışma editörleri
2. Paket yönetimi için Conda yazılımı
3. Veri madenciliği için kullanılabilecek Python paketleri
4. Python 3 programlama dili

Anaconda Programının Kurulumu

Anaconda programı **Continuum Analytics** şirketi tarafından ücretsiz olarak sunulan bir programdır[11]. Anaconda programını kurabilmek için *https://www.anaconda.com/products/individual#Downloads* adresindeki sayfadan işletim sistemine göre seçim yapılır. Windows, Mac ve Linux için kurulum aşamaları farklılık göstermektedir. Bu kitapta Windows ortamında kurulumun nasıl yapıldığı adım adım gösterilecektir. Grafikte Anaconda yükleyicisi için indirme linkinin yeri gösterilmiştir. 64-Bit kurulum için 457 MB'lık bellek alanı gerekmektedir.

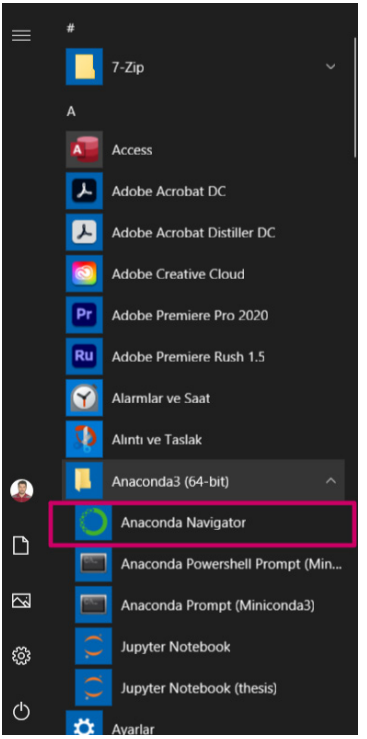
Şekil 1-5: Anaconda yükleyicileri (64-Bit Windows için 457 MB)

İlgili yükleme dosyası (.exe) indirildikten sonra kurulum gerçekleştirilir. Kurulumda Anaconda programının Windows yönetici haklarına sahip olmasını sağlamak ilerde kullanım açısından kolaylık sağlayacaktır. O nedenle şekilde görüldüğü gibi "**Add Anaconda 3 to my PATH environment variable**" seçeneğini işaretlemek faydalı olacaktır. Bu seçenek işaretlendikten sonra Anaconda komutlarını komut istemcisinde (Command Prompt) kullanabilir olur. O nedenle kitapta anlatılacak bazı durumlar için bu özelliğin aktif hâle gelmesi gerekmektedir.



Şekil 1-6: Anaconda programına Windows yetkili kullanıcı izni verme

Anaconda programını yükledikten sonra **Başlat>>Anaconda3** (64-bit) klasörü içerisindeki Anaconda Navigatör açılarak yüklenen paketler, programlar ve belgelendirmeler hakkında bilgi alınabilir.



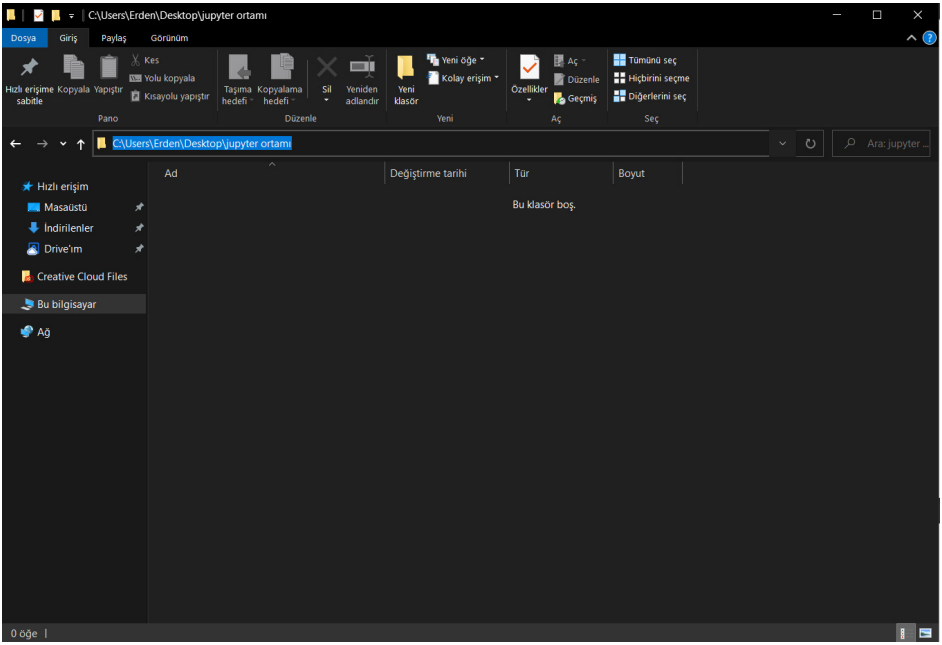
Şekil 1-7: Anaconda Navigatör programının açılması

Jupyter notebook ortamını açarak artık uygulama geliştirilebilir. Bunun için iki farklı yöntemden bahsedilecektir. Bunlar:

1. İstenilen klasör içerisinde Jupyter notebook ortamı açmak
2. Jupyter notebook ortamını varsayılan klasörde açmak

Birinci Yöntem: İstenilen klasörde Jupyter notebook ortamı açmak

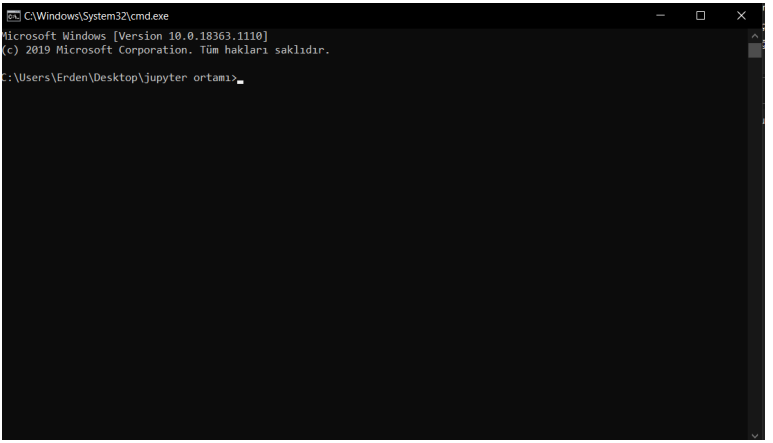
Bu yöntem sayesinde klasör içerisine bir Jupyter notebook ortamı oluşturulur. Eğer bir klasörde çalışma yapılacaksa ve bütün çalışma dosyalar bir klasör içerisindeyse bu yöntem ile kolayca içerisinde olunan klasör için kod geliştirilebilir. Bunun için, adres satırına **`cmd`** yazılır. Bu sayede komut istemcisi belirtilen klasörde açılmış olur.



Şekil 1-8: Adres satırından komut istemcisine giriş

Klasör İçerisinde Jupyter Ortamı Açmak

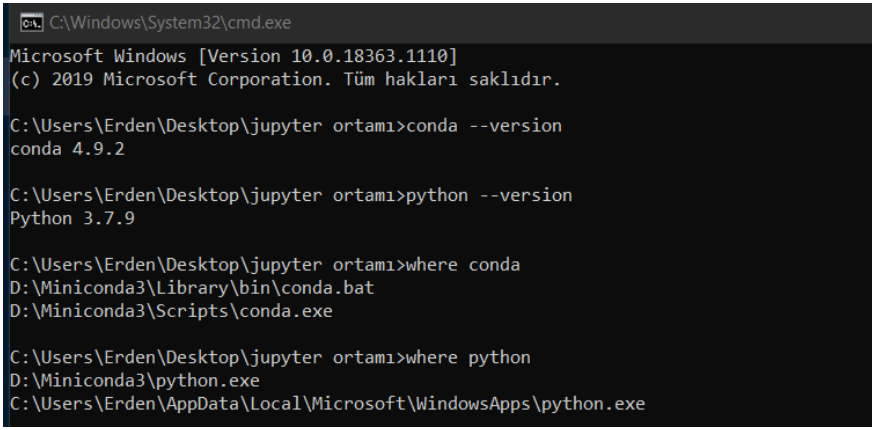
Komut istemcisi açıldıktan sonra Anaconda programına Windows yetkili kullanıcı yetkisi verildiği için **Conda** komutları kullanılabilir.



Şekil 1-9: Komut istemcisini açma

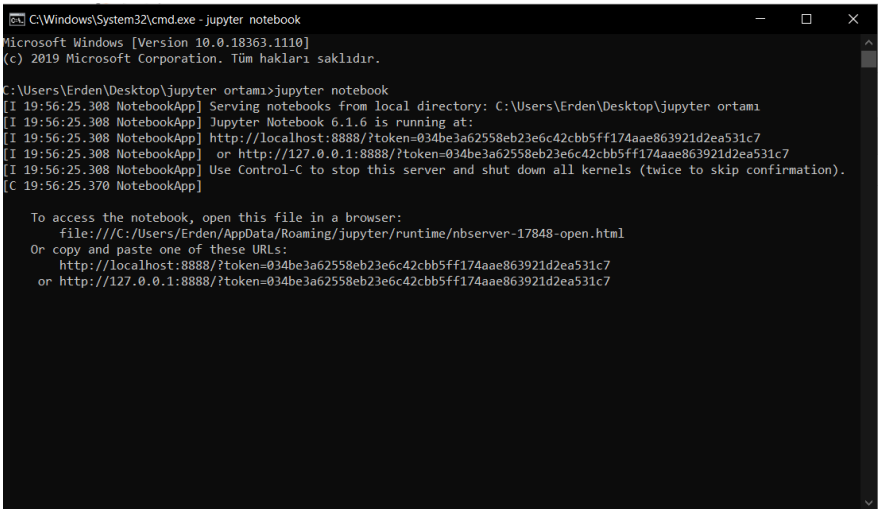
Kurulumların doğru yapılıp yapılmadığını kontrol etmek için aşağıdaki komutları yazarak kontrol sağlanabilir. Bu komutlar ile elde edilen çıktıları bilmek gerekebilir. İleride **paket yüklemekle** ilgili ya da **paketlerdeki versiyon hataları** ile ilgili problem yaşandığında buradaki bilgileri kullanmak gerekecektir.

```
conda --version # Conda versiyonunu sorgulamak için
python --version # Python versiyonunu sorgulamak için
where conda # Conda'nın kurulu olduğu dizini bulmak için
where python # Python'un kurulu olduğu dizini bulmak için
```



Şekil 1-10: Conda ile komut istemcisinde sorgulama yapma

Bu adımdan sonra komut istemcisine Jupyter notebook komutunu yazarak Jupyter notebook ortamına gidilebilir. Jupyter notebook ortamı açılırken komut istemcisinde aşağıdaki bilgiler verilir.



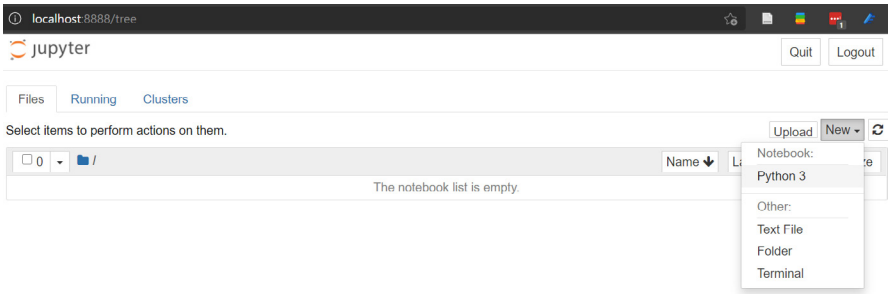
Şekil 1-11: Jupyter notebook ortamı açma

```
Microsoft Windows [Version 10.0.18363.1110]
(c) 2019 Microsoft Corporation. Tüm hakları saklıdır.
C:\Users\Erden\Desktop\jupyter ortamı>jupyter notebook
```

```
[I 19:56:25.308 NotebookApp] Serving notebooks from local directory: C:\
Users\Erden\Desktop\jupyter ortamı
[I 19:56:25.308 NotebookApp] Jupyter Notebook 6.1.6 is running at:
[I 19:56:25.308 NotebookApp] http://localhost:8888/?token=034be3a62558eb23
e6c42cbb5ff174aae863921d2ea531c7
[I 19:56:25.308 NotebookApp]  or http://127.0.0.1:8888/?token=034be3a62558
eb23e6c42cbb5ff174aae863921d2ea531c7
[I 19:56:25.308 NotebookApp] Use Control-C to stop this server and shut
down all kernels (twice to skip confirmation).
[C 19:56:25.370 NotebookApp]
    To access the notebook, open this file in a browser:
        file:///C:/Users/Erden/AppData/Roaming/jupyter/runtime/nbserver-
17848-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=034be3a62558eb23e6c42cbb5ff174aae8639
21d2ea531c7
     or http://127.0.0.1:8888/?token=034be3a62558eb23e6c42cbb5ff174aae8639
21d2ea531c7
```

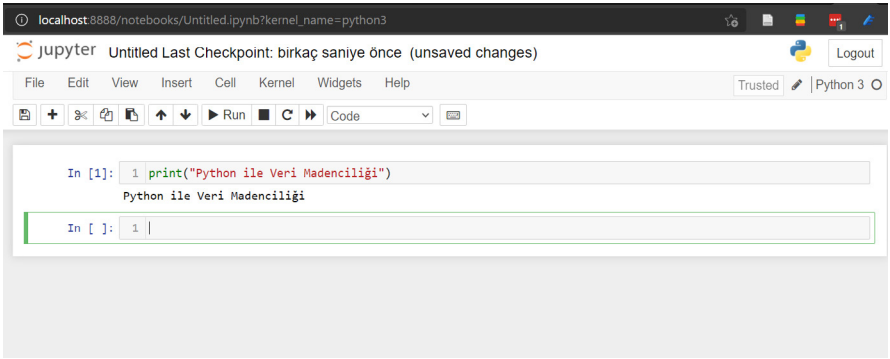
Bilgilerde kısaca çalışma klasöründe bir **localhost** oluşturulduğu ve içerisine Jupyter notebook ortamı üretildiği belirtilmektedir. Eğer tarayıcı komut istemcisinden açılmaz ise komut istemcisinde yazan *http://localhost:8888/?token=034be3a62558eb23e6c42cbb5ff174aae863921d2ea531c7* ifadesi bir tarayıcı adresine yazılarak açılırsa işlem tamamlanmış olur.

Belirtilen klasörde **New** seçeneğinden **Python 3** seçilerek klasörde bir Jupyter notebook ortamı açılmış olur.



Şekil 1-12: Python 3 dosyası açma

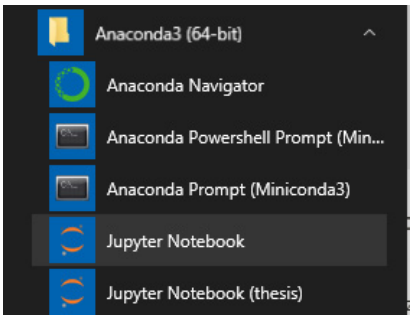
Notebook ortamı ilk açıldığında ismi "**Untitled**" olarak gelir. Notebook ortamının ismini değiştirmek ilk adımdır. Notebook ortamını bir sonraki bölümde daha detaylı tanıtılacaktır. Şimdilik artık bir notebook ortamı kurulduğuna göre ilk kod parçacığı print ("**Python ile Veri Madenciliği**") ile ekrana yazdırılarak kod geliştirme işlemine başlanabilir hâle gelinir. Birinci hücreye komut yazıldıktan sonra klavyeden **CTRL+Shift** ile veya ekrandan **Run** seçeneği seçilerek şekilde görüldüğü gibi hücre çalıştırılır.



Şekil 1-13: print() fonksiyonu ile ekrana yazdırma

İkinci Yöntem: Varsayılan klasörde Jupyter notebook ortamı açmak

Bu yöntem, eğer çalışma klasörü belli değilse ve yeni bir çalışmaya başlanıyorsa kullanılabilir. Jupyter notebook çalışma ortamı şekildeki gibi **Başlat** menüsünden direkt açılabileceği gibi Windows arama kısmına **cmd** yazılarak komut istemcisine Jupyter notebook yazılarak da açılabilir. Diğer adımlar birinci yöntemde olduğu gibi ilerletilebilir.

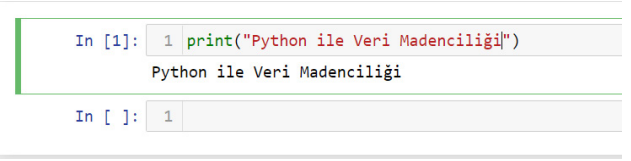


Şekil 1-14: Varsayılan klasörde Jupyter notebook dosyası açma

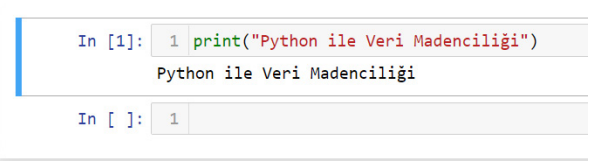
Jupyter Notebook Ortamı ile Alıştırmalar

Jupyter notebook dosyası içerisinde istenilen sayıda hücre oluşturulabilir. Notebook dosyasındaki hücreler içerisine kod parçacıkları yazılabileceği gibi normal metin de yazılabilir. Hatta içerisine **markdown** dili kullanılarak görüntü, tablo, denklem, liste vb. gibi özellikler de eklenebilir. Yani markdown yazılacakları Microsoft Word içerisinde yazılıyormuş gibi düşünülebilir. Şu adresteki blog yazısında Jupyter notebook dosyasında kullanılacak birçok özellikten bahsedilmiştir. *https://medium.com/analytics-vidhya/the-ultimate-markdown-guide-for-jupyter-notebook-d5e5abf728fd*. Bu kitapta da birkaç uygulama ile sıklıkla kullanılan Jupyter notebook özelliklerinden bahsedilecektir.

Jupyter notebook hücreleri seçildiğinde sol kısmında şekilde görüldüğü gibi mavi bir çubuk çıkar. Hücre seçildikten sonra **Enter** tuşuna basılınca hücre sol çubuğu yeşile döner. Yeşil çubukta kod parçacığı yazılabilir, mavi çubukta ise kısa yollar kullanılabilir durumdadır.



Şekil 1-15: Hücrede kod yazmanın aktif olması durumu



Şekil 1-16: Hücrenin aktif olmama durumu

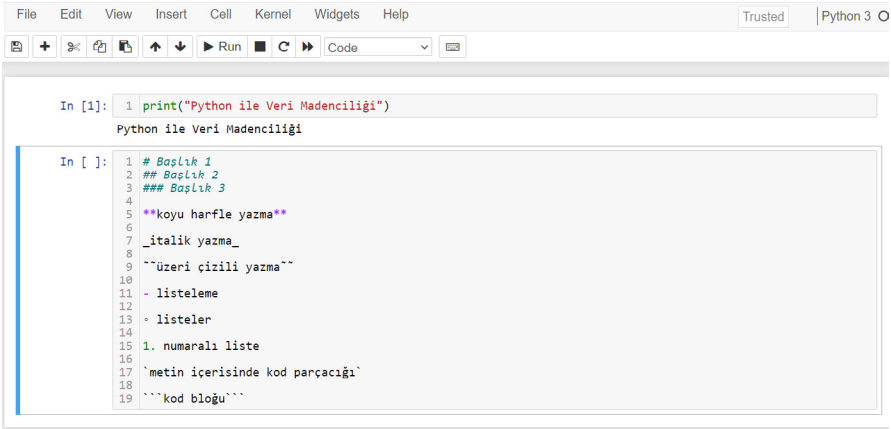
Hücre mavi çubuğa getirildikten sonra aşağıdaki tablodaki klavye kısa yolları kullanılabilir. Klavye kısa yolları kod geliştirirken ciddi zaman kazançları sağlamaktadır.

Kısa yollar	İşlem
a	Yukarı bir hücre ekler.
b	Aşağı bir hücre ekler.
dd	Hücreyi siler.
enter	Hücre düzenleme moduna geçer.
shift +enter	Hücre çalıştırılır.
esc	Hücre in aktif moda geçer.
y	Kod parçacığına dönüştürür.
m	Markdown parçacığına dönüştürür.
shift + j	Şimdiki ve sıradaki hücreyi seç.
shift + m	Seçili hücreleri birleştir.
shift + minus	Hücreyi böler.

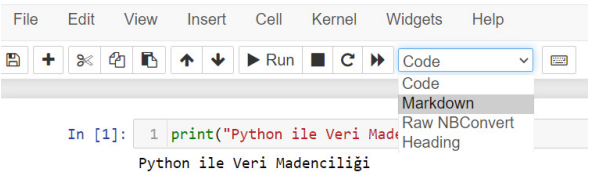
Tablo 1.1: Bazı Jupyter notebook kısayolları

Markdown Kullanımı

Markdown işaretleme dili (Markup) olarak bilinir. HTML gibi bir yapısı vardır. 2004 yılında John Gruber ve Aaron Swartz tarafından geliştirilmiştir[12,13]. Markdown dili kısaca kolay okuma ve yazma için geliştirilmiş yalın metin formatı olarak tanımlanabilir. HTML içerisinde bir başlık tanımlanacağı zaman **<h1>Başlık<\h1>** yapısı kullanılırken Markdown içerisinde **#** yazdıktan sonra boşluk bırakılması yeterli gelmektedir. Bu nedenle kullanım kolaylığı sağlamaktadır. Jupyter notebook hücreleri içinde Markdown özelliklerinin tamamını şekilde gösterildiği gibi kullanılabilir. Bu özellikleri Markdown olarak kullanabilmek için **Code** hücresini **Markdown** hücresine dönüştürmek gereklidir.

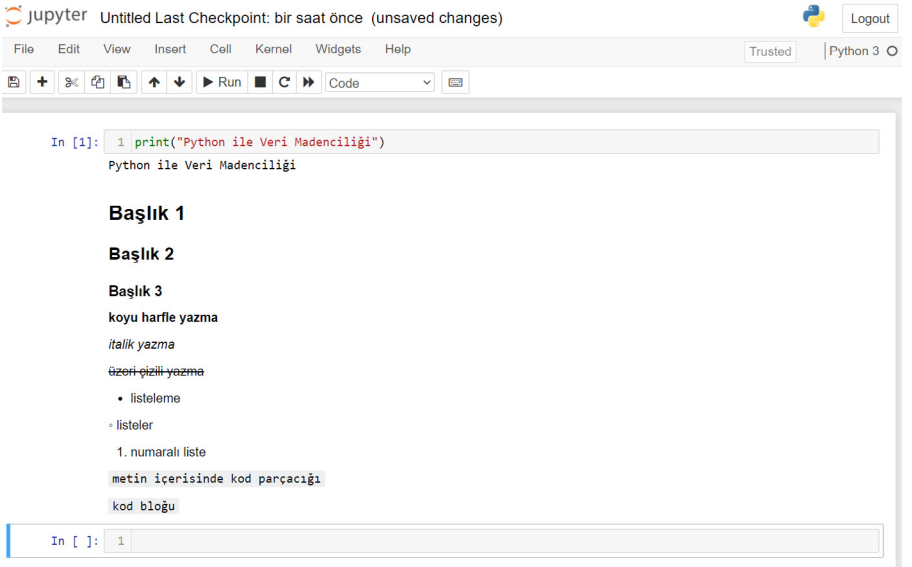


Şekil 1-17: Jupyter notebook içerisinde Markdown kullanımı



Şekil 1-18: Kod hücresini Markdown hücresine çevirme

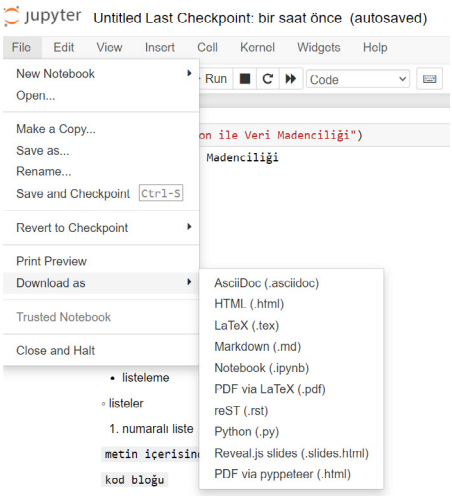
Hücreyi Markdown hücresine çevirdikten sonra çalıştırınca şekildeki gibi bir görüntü elde edilir. Şekilde görüldüğü gibi Markdown hücresi çalıştırıldıktan sonra sol tarafta köşeli parantezle numaralandırılmamıştır. Bu hücreler kod hücresi olarak geçmez. Kod için bir **belgelendirme** olarak bilinir. Bu özelliği sayesinde kodlar arasına bilgilendirmeler daha zengin hâle gelir. Markdown dili sayesinde belgelendirme oluşturma, içindekiler tablosu, sunum dosyası, denklem editörü gibi birçok özellik kullanılabilir. Bu özellikler ile alakalı detaylı Markdown kullanımı için şu adres ziyaret edilebilir: *https://www.markdownguide.org/getting-started/*



Şekil 1-19: Markdown hücresini çalıştırma

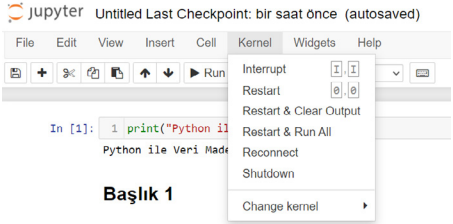
Jupyter Notebook Menüleri

Jupyter notebook ortamındaki menüler temel bir menü yapısındadır. **File** menüsünde standart menülerin yanında "**Download as**" seçeneği bulunur. Bu seçenek sayesinde geliştirilen kodlar (.py, .pdf, .md gibi) değişik formatlarda kaydedilebilir. Şekilde görülen bu özellik ile geliştirilen kodlar başka ortamlarda kolaylıkla paylaşılabilir. Böylece yazılan kodlar ile belgelendirme kısmı da tek bir dosya hâlinde diğer araştırmacılara gönderilebilir.



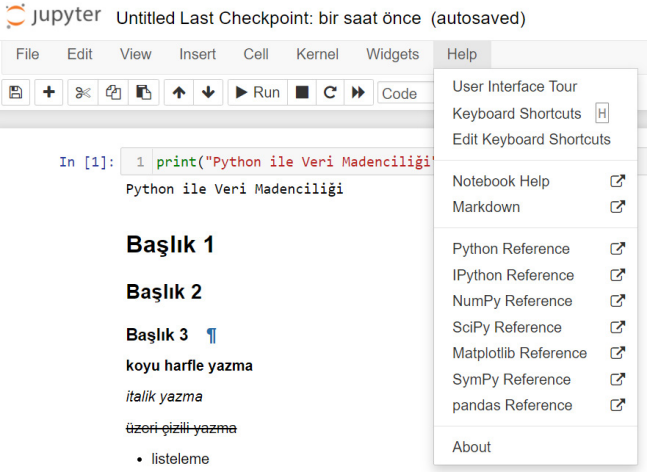
Şekil 1-20: Jupyter notebook menülerinde Download as özelliği

Kernel menüsünde localhostta çalıştırılan notebook ortamı durdurabilir, yeniden başlatabilir, değiştirebilir veya tüm çıktılarını silinebilir. Bu özellik sayesinde bazen kodlarda yaşanacak sorunlarla karşılaşıldığında kernelin yeniden başlatılması gerekebilir. **Change Kernel** seçeneği ile var olan başka bir ortama geçilebilir. Bu kitaptaki anlatımda tek kernel ile çalışıldığı varsayılacaktır.

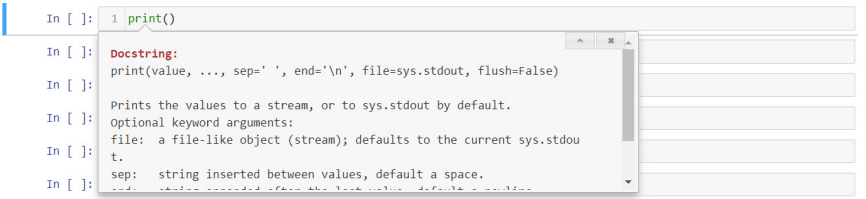


Şekil 1-21: Jupyter notebook Kernel menüsü

Yardım menüsünde Jupyter notebook kullanımı ile sıklıkla kullanılan kütüphaneler ile ilgili de dokümantasyonlara ulaşılabilir. Şekildeki referanslara ulaşabilmek için internet bağlantısı gerekmektedir. Bunlar dışında yazılan kod içerisindeki bir fonksiyon ya da metodun yardım dokümanına ulaşabilmek için fonksiyonu yazıp parantez koyduktan sonra **Shift+Tab** tuş kombinasyonuna basılması gerekir. Yardım dokümanının tamamına ulaşmak için **Shift+Tab** kombinasyonundan sonra tekrar **Tab** tuşuna basılabilir. Örnek olarak **`print`** fonksiyonu için bir yardım dokümanı şekilde gösterilmiştir.



Şekil 1-22: Yardım menüsü



Şekil 1-23: Fonksiyonlar için yardım dokümantasyonuna ulaşma

Diğer Veri Madenciliği Araçları

Python programlama diline alternatif olarak düşünebilecek bazı veri madenciliği araçlarından kısaca bahsedilecek olursa aşağıdaki örnekler verilebilir. Kimi araçlar kullanıcının uygulamaları kendi kodları ile geliştirmesini beklerken kimi araçlar menüler ve butonlar sayesinde kolay bir kullanım imkânı sunar.

Orange: Veri bilimi için kullanılabilecek sürükle bırak özelliği olan bir yazılımdır. *https://orange.biolab.si/* adresinden indirilebilir durumdadır. Anaconda Navigatör içerisinde Orange paketinin kurulumu için yönlendirmeler bulunur. Veri görselleştirme, sınıflandırma, kümeleme gibi birçok veri madenciliği görevlerini yerine getirir. Gerekli durumlarda Python kodları ile yazılıma müdahale edilebilmektedir.

Şekil 1-24: Orange logo

Weka: Yeni Zelanda'daki Waikato Üniversitesindeki bir grup akademisyen tarafından JavaScript kodları ile geliştirilmiştir. İçerisinde çok geniş veri madenciliği algoritmaları bulunmaktadır. İnternet sayfası *https://www.cs.waikato.ac.nz/ml/weka/* adresinde yer almaktadır. Geniş bir kullanıcı ve destekleyici kitlesi vardır.

Şekil 1-25: Weka logo

Veri madenciliği görevlerinden veri önişleme, sınıflandırma, regresyon, kümeleme analizleri, veri görselleştirme gibi görevleri yerine getirebilir. Ara yüzü ve kullanımı kolaylıkla öğrenilebilir.

Rapidminer: Veri bilimi veya veri ile ilgilenen kişiler için ücretsiz imkânlar sunar. Yaklaşık 40.000 global şirket bu uygulamayı kullanır. İnternet sayfası *https://rapidminer.com/* adresinde bulunmaktadır.

Şekil 1-26: Rapidminer Logosu

Veri hazırlama, makine öğrenmesi, derin öğrenme, metin madenciliği gibi birçok alanda algoritmalar sunmaktadır. Ücretsiz ve açık kaynak bir yazılımdır. Ara yüzü ve kullanımı kolaydır. Sürükle bırak şeklinde çalışır. Arzu eden kullanıcılar R ve Python programlama diliyle kendi kodlarını geliştirebilirler.

Rapidminer içerisindeki stüdyo sayesinde verilerin görselleştirilmesi ve sunulması, analizler yapılması, örüntü keşfi gibi çalışmalar hızlı bir şekilde yapılır. Ücretsiz versiyonunda 10.000 satır veriye kadar izin vermektedir ki bu kadar satır veri çoğu uygulama için yeterli olacaktır.

Knime: Diğer bir açık kaynak makine öğrenmesi platformudur. *https://www.knime.com/* adresinde ana sayfası yer alır. Baştan sonra veri bilimi projeleri geliştirmek için tasarlanmıştır. Daha önce belirtilen veri madenciliği görevlerinin hemen hemen tamamını yerine getirmektedir.

Şekil 1-27: Knime logo

Diğer veri madenciliği araçları şu şekilde listelenebilir:

- Advanced miner
- Alteryx
- Apache Mahout
- Apache Spark
- Azure ML
- IBM SPSS
- Qlik
- R-Programlama
- SAS Enterprise Miner
- Sisense
- Teradata

Özet

Bu bölümde veri madenciliği ile ilgili giriş seviyesinde bilgi verildikten sonra kitapta kullanılacak programlar gösterilmiştir. Kısaca tekrarlamak gerekirse Anaconda programı tek başına gerekli tüm yüklemeleri yapmaktadır. Anaconda programı ile yüklenen **Jupyter notebook ortamı** ile nasıl çalışılacağı gösterilerek bu bölüm tamamlanmıştır.

Kaynaklar

1. "Battle of the Data Science Venn Diagrams", KDnuggets (2021).
2. Ceri, S. "On the role of statistics in the era of big data: A computer science perspective", Statistics & Probability Letters, 136, pp. 68–72 (2018).
3. Cao, L. "Data Science: A Comprehensive Overview", ACM Comput. Surv., 50(3), pp. 1–42 (2017).
4. "Multiple Testing Problems in Pharmaceutical Statistics | Taylor & Francis Group", *https://www.taylorfrancis.com/books/multiple-testing-problems-pharmaceutical-statistics-alex-dmitrienko-ajit-tamhane-frank-bretz/e/10.1201/9781584889854.*
5. Weihs, C. and Ickstadt, K. "Data Science: the impact of statistics", Int J Data Sci Anal, 6(3), pp. 189–194 (2018).
6. L Halterman, R. Learning to Program with Python (2011).
7. Passmore, H. A. "To Learn Data Science Better, Use SCIENCE!", *https://towardsdatascience.com/to-learn-data-science-better-use-science-ffaae3d56bea.*
8. "Data Science Roadmap", *https://medium.com/@ArtisOne/data-science-roadmap-2020-b256fb948404.*
9. Metwalli, S. A. "A Learning Path To Becoming a Data Scientist", *https://towardsdatascience.com/a-learning-path-to-becoming-a-data-scientist-56c5c2e8ae3f.*
10. "Top 6 Data Science Programming Languages 2021 [Hand-Picked]", *https://www.upgrad.com/blog/data-science-programming-languages/.*
11. "Anaconda | Individual Edition", *https://www.anaconda.com/products/individual.*
12. *"Wayback Machine", https://web.archive.org/.*
13. "Daring Fireball: Markdown", *https://daringfireball.net/projects/markdown/.*

BÖLÜM 2

PYTHON PROGRAMLAMA DİLİ İÇİN HIZLI BİR KURS

Bu bölümde hızlı bir şekilde Python programlama dilinin yapısından ve özelliklerinden bahsedilecektir. Burada temel seviyede Python programlama diline başlamak için gerekli bilgileri öğrenebilirsiniz. Eğer ortalama seviyede bir Python bilgisine ve tecrübesine sahipseniz bu bölümü geçebilirsiniz. Burada anlatılanlar sadece bir giriş anlamı taşıyacaktır.

Bu bölüm daha önce hiç bilgisayar programlama çalışması yapmamış kişiler için hazırlanmıştır. Eğer daha önce bir eğitiminiz varsa ya da program geliştirdiyseniz bu eğitim sizin için basit kalabilir. Ancak yine de Python programlama dilini öğrenmeye başlamak için bir temel edinebilirsiniz. Daha detaylı anlatımlar için kaynaklar şu şekilde listelenebilir:

1. **Türkçe kaynak**
   a. İstihza[1]
   b. Yeni Başlayanlar için Python[2]
2. **İngilizce kaynak**
   a. Learn Python Hard Way[3]
   b. Learning Python[4]
3. **YouTube kanalları**
   a. Sentdex[5]
   b. Techwithtim[6]
   c. Telusko[7]

4. İnternet Sayfaları

a. Towards Data Science[8]

b. Real Python[9]

c. Python Anywhere[10]

Python programlama dilinde yaşanan sorunlar ile ilgili soru sorulabilecek sayfalar şu şekilde listelenebilir:

1. **Stack Overflow** *https://stackoverflow.com/*
2. **GitHub** *https://github.com/*
3. **Python forum** *https://www.python.org/community/forums/*
4. **Google Groups** *https://groups.google.com/forum/#!forum/python-ideas*
5. **Paket belgelendirmeleri** *https://docs.scipy.org/doc/numpy/reference/*
6. **Facebook, Twitter, Linkedin vb. sosyal medyalar.**

Jupyter Hakkında

Bu bölümdeki gösterimler birinci bölümde anlatılan Python kurulumu üzerinden götürülecektir. Jupyter notebook hakkında hem notların hem de kodların bir arada tutulmak için kullanılabilecek başarılı bir araç olduğundan bahsedilmişti. Jupyter notebook veri madenciliği kodlarının geliştirildiği alanda kendisini kanıtlamıştır ve birçok geliştirici tarafından kullanılmaktadır. Jupyter notebook programını açmak için Anaconda programını kurulduktan sonra **Başlat >> Anaconda3 >> Jupyter** notebook seçeneğinden açılabilir.

Gömülü Fonksiyonlar

Python kurulumu ile gelen ve sonradan kurmanıza gerek kalmayan fonksiyonlardır. Tüm fonksiyonlar alfabetik olarak aşağıdaki tabloda listelenmiştir.[11]

abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

Tablo 2-1: Python dilindeki gömülü fonksiyonlar

Print() fonksiyonu

print() fonksiyonu içerisine yazılan bilgiyi **ekrana yazdırır**.

Ekrana **print()** fonksiyonu ile yazdırılan bilgiler sayesinde programın nasıl çalıştığı hakkında bilgi alınarak debug işlemi de gerçekleştirilebilir. **Print()** fonksiyonu hakkında yardım almak istendiğinde aşağıdaki gibi **help()** fonksiyonu kullanılabilir.

```
print("Python ile Veri Madenciliği")
print("o----")
print("*"*10)
```

```
Python ile Veri Madenciliği
o----
**********
```

```
help(print)
```

```
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

print() fonksiyonunun parametrelerinden örneğin **sep** parametresi verilen kelimeler arasına istenilen karakterin yazılmasını sağlar.

```
print("Python","ile", "Veri", "Madenciliği",sep="*")
```

```
Python*ile*Veri*Madenciliği
```

print() fonksiyonunun **\n** özelliği ile yazdırılacak metin alt satıra yazdırılabilir.

```
print("Python\nile\nVeri\nMadenciliği")
```

```
Python
ile
Veri
Madenciliği
```

\t karakteri bir **tab** boşluk bırakarak ekrana yazdırır.

```
print("Python\tile\tVeri\tMadenciliği")
```

```
Python          ile     Veri    Madenciliği
```

Format **print()** özelliği ile "**Mehmet Yılmaz**" isminde bir hastanın bilgilerini ekrana yazdırmak için **{}** özelliği kullanılır.

```
print("Bugün hastaneye {} isminde {} yaşında bir hasta
gelmiştir.".format("Mehmet Yılmaz",30))
```

```
Bugün hastaneye Mehmet Yılmaz isminde 30 yaşında bir hasta gelmiştir.
```

```
print("Hastanın {} derece ateşi vardır.".format(37.6))
```

```
Hastanın 37.6 derece ateşi vardır.
```

Input() Fonksiyonu

Kullanıcıdan veri almak için kullanılır.

```
isim = input("isminiz nedir? ")
# + işareti birleştirmek içindir (boşluk bırakmaz)
print("Merhaba " + isim)
```

```
isminiz nedir? Caner
Merhaba Caner
```

```
yas = input("Yaşınız kaç? ")
# , işareti de birleştirme için kullanılır arada boşluk vardır.
print(isim, yas, "yaşındadır.")
```

```
Caner 30 yaşındadır.
```

Girilen yıla göre yaş hesaplama istenirse aşağıdaki gibi bir hata alınır. Bu hatanın giderilmesi için **değişken dönüşümünün** yapılması gerekir.

```
dogum_yili = input("Doğum yılınızı giriniz >> ")
yas = 2019 - dogum_yili
print("Yaşınız ", yas)
```

```
Doğum yılınızı giriniz >> 1989
---------------------------------------------------------------------------
TypeError Traceback (most recent call last)
<ipython-input-3-adda66b69295> in <module>
      1 dogum_yili = input("Doğum yılınızı giriniz >> ")
----> 2 yas = 2019 - dogum_yili
      3 print("Yaşınız ", yas)

TypeError: unsupported operand type(s) for -: 'int' and 'str'
```

Değişken Dönüşümü

Input() her değişkeni **str** olarak alır. Bu nedenle değişken dönüşümü yapmamız gerekir. **int()** fonksiyonu girdiyi tam sayıya (**int**) çevirir. **float()** küsuratlı, **str()** metine çevirir.

```
print("Caner "+"Erden")
```

Caner Erden

```
dogum_yili = int(dogum_yili)
yas = 2019 - dogum_yili
print("Yaşınız ", yas)
```

Yaşınız 30

```
type(dogum_yili)
```

int

```
dogum_yili = int(input("Doğum yılınızı giriniz >> "))
print("yasiniz", 2019-dogum_yili)
```

yasiniz 30

Örnek olarak **TL dolar dönüşümü** yapılmak istenirse aşağıdaki gibi yapılabilir.

```
TL_miktar = float(input("Elinizdeki TL miktarını giriniz\t\t: "))
dolar_miktari = TL_miktar*0.2
print("Elinizdeki TL'nin Dolar karşılığı\t:", dolar_miktari)
```

Elinizdeki TL miktarını giriniz : 100
Elinizdeki TL'nin Dolar karşılığı : 20.0

Değişken Tipleri

Python programlama dilinde yer alan standart değişken tipleri aşağıdaki gibi listelenebilir:

- **Sayılar (integer, float, complex):** Nümerik verilerin saklandığı veri tipidir. Genel olarak üç çeşit sayı tipinden bahsedilebilir. Bunlar tamsayılar (integer - 10, 20 gibi), küsuratlı sayılar (float - 1,5 15,50 gibi) ve karmaşık sayılar (complex – 4+3i gibi) sayı tipleridir.
- **Metinler (String):** Tırnak içerisinde yazılan ve karakterlerden oluşan dizilerdir. String içerisinde listelerde olduğu gibi karakterler arasında işlemler yapılabilir. String içerisindeki herhangi bir karaktere ulaşmak için "**[**" köşeli parantez kullanılır.

- **Listeler (lists):** Listeler Python programlama dilinde üzerinde en fazla durulması gereken veri tipidir. Listenin içerisinde virgüller ile ayrılmış çeşitli değişken tiplerinde elemanlar bulunur. Listeler ile ilgili detaylı bilgi NumPy paketi ile verilecektir.
- **Demet (Tuple):** Listelere benzer yapıdadır. Listelerden farkı demetlerde elemanların değerleri veya sırası değiştirilemezdir. Demet içerisindeki elemanlar sadece okunabilir. Listelerden farklı olarak "`(`" ile kullanılır.
- **Sözlük (dictionary):** Sözlükler ikili bilgi tutarlar. İlk gelen bilgi anahtar ikinci gelen bilgi değer olarak bilinir. "`{`" ile kullanılır.

Değişkenlerin tipleri görüntülenmek istendiğinde `type()` fonksiyonundan yararlanılır. Aşağıda bu alanda gerçekleştirilen uygulamalara örnekler verilmiştir.

```
# Tam sayılar
yas = 10
print(yas)
print(type(yas))
```

```
10
<class 'int'>
```

```
help(int)
```

```
Help on class int in module builtins:
class int(object)
| int([x]) -> integer
| int(x, base=10) -> integer |
| Convert a number or string to an integer, or return 0 if no arguments
| are given. If x is a number, return x.__int__(). For floating point
| numbers, this truncates towards zero. |
| If x is not a number or if base is given, then x must be a string,
| bytes, or bytearray instance representing an integer literal in the | given base. The literal can be preceded by '+' or '-' and be surrounded
| by whitespace. The base defaults to 10. Valid bases are 0 and 2-36.
| Base 0 means to interpret the base from the string as an integer literal.
```

```
# Float Küsüratlı değerler
ates_derecesi = 36.9
print(type(ates_derecesi))
```

```
<class 'float'>
```

```
help(float)
```

```
Help on class float in module builtins:
class float(object)
| float(x=0, /) |
| Convert a string or number to a floating point number, if possible. |
| Methods defined here: |
| __abs__(self, /)
| abs(self) |
| __add__(self, value, /)
| Return self+value. |
| __bool__(self, /) | self != 0 |
| __divmod__(self, value, /)
| Return divmod(self, value). |
| __eq__(self, value, /) | Return self==value.
```

```
# String Metinsel değişkenler
isim_soyisim = "Mehmet Yılmaz"
print(type(isim_soyisim))
```

```
<class 'str'>
```

```
# bool değişkenleri Doğru Yanlış değerleridir
oksuruk_var_mi = True
print(type(oksuruk_var_mi))
```

```
<class 'bool'>
```

```
# Tek ' ile tanımlama
kitap_adi = 'Python' ile Veri Madenciliği'
```

```
File "<ipython-input-18-93d0bdf2d8e5>", line 2 kitap_adi = 'Python' ile Veri Madenciliği' ^ SyntaxError: invalid syntax
```

```
kitap_adi="Python' ile Veri Madenciliği"
```

```
# Dilimleme yapmak için
print(kitap_adi[:6]) # ilk 6 karakteri getirir.
```

```
Python
```

```
print(kurs_adi[6:]) # Altıncı karakter ve sonrasını getirir.
```

```
' ile Veri Madenciliği
```

```
print(kitap_adi[6:12]) # Altıncı karakter ve sonrasında 12. karaktere kadar getirir.
```

```
' ile
```

```
# bir metindeki karakter sayısını ölçmek için
print(len(kitap_adi))
```

```
28
```

```
# büyük ya da küçük karaktere dönüştürme
print(kitap_adi.upper())
```

```
PYTHON' ILE VERI MADENCILIĞI
```

```
print(kitap_adi.lower())
```

```
python' ile veri madenciliği
```

```
# Kitap adı değişmiyor
print(kurs_adi)
```

```
Python' ile Veri Madenciliği
```

```
# Karakterin kaçıncı sırada olduğunu bulmak için
print(kurs_adi.find('V'))
```

```
12
```

```
# bir yerdeki karakterleri değiştirmek için
print(kitap_adi.replace("Python", "R"))
```

```
R' ile Veri Madenciliği
```

```
# bir takım karakterler var mı diye bakmak [in] operatörü
print('Python' in kitap_adi)
```

```
True
```

```
# center
print(kitap_adi.center(50,"*"))
# split fonksiyonu
print('1+2+3+4+5'.split('+'))
# strip fonksiyonu
" Python ile veri madenciliği ".strip()
# join
'+'.join("1234")
```

```
**********Mühendishane'nin Python Eğitimi**********
['1', '2', '3', '4', '5']
'1+2+3+4'
```

Aritmetik İşlemler

Bu bölümde aritmetik işlemler ile ilgili uygulama örnekleri gösterilecektir. Python programlama dili ve Jupyter ortamı hesap makinesi yerine kullanılabilecek derecede pratiklik sağlamaktadır.

```
# Toplama çıkarma
print(10+3)
# çarpma, üs alma
print(10**3)
# bölme, modül alma
print(10/3)
print(10//3)
print(10%3)
```

```
13
1000
3.3333333333333335
3
1
```

Aritmetik işlemlerdeki operasyon öncelikleri aşağıdaki gibidir.

- parantez içi
- Üssel ifadeler
- çarpma veya bölme
- toplama veya çıkarma

```
x = 30+4*5
y = 3**2*4-10*2
z = (9+2)**2+3
print(x)
print(y)
print(z)
```

```
50
16
124
```

```
# mutlak değer
print(abs(-2.6))
```

```
2.6
```

Math Modülü

Matematiksel işlemlerin olduğu modül Python programlama dilinin Standart Kütüphanesi içinde hangi modüllerin olduğunu *https://docs.python.org/3/library/* adresinden incelenebilir.

```
# Math modülünü çağırıyoruz.
import math
# modül içindeki tüm fonksiyonlar
dir(math)
```

```
['__doc__', '__loader__', '__name__', '__package__', '__spec__',
 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil',
 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod',
'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma',
'log', 'log10', 'log1p', 'log2', 'modf',
 'nan', 'pi', 'pow', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']
```

```
x = 2.9
y=math.ceil(x)
print(y)
z = math.floor(x)
print(z)
pi=math.pi
print(pi)
kok=math.sqrt(16)
print(kok)
print(math.inf)
print(math.factorial(5))
```

```
3
2
3.141592653589793
4.0
inf
120
```

Pratik Önerisi

Bahsedilen konularda pratik yapmak için Practice Python (*https://www.practicepython.org/*) adresindeki örnekler çözülebilir. Örneğin **egzersiz 1** ve **6** çözülebilir. Ya da "Kullanıcıdan alınan yarıçap ölçüsü (float) ile çemberin çevresini ve alanını hesaplayan ve ekrana yazdıran bir program yazınız." sorusuna cevap aranabilir.

Fonksiyonlar

Fonksiyonların görevi, karmaşık işlemleri bir araya toplayarak, bu işlemleri tek adımda yapmamızı sağlamaktır. Fonksiyonlar çoğu zaman, yapmak istediğimiz işlemler için bir şablon vazifesi görür. Fonksiyonları kullanarak, bir veya birkaç adımdan oluşan işlemleri tek bir isim altında toplanabilir. Python'daki '**fonksiyon**' kavramı başka programlama dillerinde '**rutin**' veya '**prosedür**' olarak adlandırılır. Gerçekten de fonksiyonlar rutin olarak tekrar edilen görevleri veya prosedürleri tek bir ad/çatı altında toplayan araçlardır.[1]

- Her fonksiyonun bir adı bulunur ve fonksiyonlar sahip oldukları bu adlarla anılır. (**`print`** fonksiyonu, **`open`** fonksiyonu, **`type`** fonksiyonu, **`input`** fonksiyonu, **`len`** fonksiyonu vb.)
- Şekil olarak, her fonksiyonun isminin yanında birer parantez işareti bulunur. (**`open()`**, **`print()`**, **`input()`**, **`len()`** vb.)
- Bu parantez işaretlerinin içine, fonksiyonlara işlevsellik kazandıran bazı parametreler yazılır. (**`open(dosya_adı)`**, **`print("Merhaba Dünya!")`**, len("Kahramanmaraş") vb.
- Fonksiyonlar farklı sayıda parametre alabilir. Örneğin **`print()`** fonksiyonu toplam 256 adet parametre alabilirken **`input()`** fonksiyonu yalnızca tek bir parametre alır.
- Fonksiyonların isimli ve isimsiz parametreleri vardır. **`print()`** fonksiyonundaki sep, end ve file parametreleri isimli parametrelere örnekken, mesela **`print("Merhaba Dünya!")`** kodunda **Merhaba Dünya!** parametresi isimsiz bir parametredir. Aynı şekilde **`input("Adınız: ")`** gibi bir kodda **`Adınız:`** parametresi isimsiz bir parametredir.
- Fonksiyonların, isimli ve isimsiz parametreleri dışında, bir de varsayılan değerli parametreleri vardır. Örneğin **`print()`** fonksiyonunun **`sep`**, **`end`** ve **`file`** parametreleri varsayılan değerli parametrelere birer örnektir. Eğer bir parametrenin varsayılan bir değeri varsa, o parametreye herhangi bir değer vermeden de fonksiyonu kullanabiliriz. Python bu parametrelere, belirli değerleri ön tanımlı olarak kendisi atayacaktır. Tabii eğer istersek, varsayılan değerli parametrelere kendimiz de başka birtakım değerler verebiliriz.

```
# Kullanıcıdan alınan bir yarıçapa göre çemberin çevresini hesaplayın
yaricap = int(input("Çmberin yarıçapını giriniz..."))
cemberin_cevresi = 2*3.14*yaricap
print("Yarıçapı {} olan çemberin çevresi {}'dir'".format(yaricap,cemberin_
cevresi))
```

```
Çemberin yarıçapını giriniz...
20 Yarıçapı 20 olan çemberin çapı 125.60000000000001'dir'
```

```
def birinci_fonksiyonum():
    print('Merhaba Dünya!',"...",sep=' ',end=" ")

print('type: {}'.format(birinci_fonksiyonum))

birinci_fonksiyonum()  # fonksiyonu çağır
```

```
type: <function birinci_fonksiyonum at 0x0000012705160C18>
Merhaba Dünya! ...
```

```
type(birinci_fonksiyonum)
```

```
function
```

```
# Fonksiyonun içerisine parametrelerin girilmesi
def f1(a="girdi1",b="girdi2"):
    print(a+' ' +b)
f1(a="birinci_deger", b="ikinci_deger")
```

```
birinci_deger ikinci_deger
```

```
# Bir parametrenin değerinin değiştirilmesi
f1(b="b değeri")
```

```
girdi1 b değeri
```

```
# Çıktı dönderen fonksiyon yazılımı
def donusumlu_fonksiyon(x):
    y = 3*x+5
    return y
donusumlu_fonksiyon(10)
```

```
35
```

```
def f2(x):
    y = 3*x + 5
    return y
f2(5)
```

```
20
```

Argümanlar

Fonksiyonlara girdi olarak verilir. Örnek uygulamalar aşağıda verilmiştir.

```
def selamla(isim1="Caner", isim2=""):
    print('Merhaba {} {}'.format(isim1,isim2))
# isim1 verilmez ise varsayılan alınır.
selamla(isim2="Erden")
```

Merhaba Caner Erden

```
# isim1 ve isim2 birlikte verilebilir.
selamla(isim1="Caner", isim2="Erden")
```

Merhaba Caner ve Erden!

```
# Argümanlar tanımlı çağrılırsa yerleri değişebilir.
selamla(isim2="Erden",isim1="Caner")
```

Merhaba Caner ve Erden!

```
# return değeri fonksiyonun döndüreceği değeri belirler.

def kucuk_harf_yaz(original):
    modified = original.strip().lower()
    return modified

bozuk_string = '  MeRhaBa DÜnYA  '
duzgun = kucuk_harf_yaz(bozuk_string)
print('duzgun: {}'.format(duzgun))
```

duzgun: merhaba dünya

```
kucuk_harf_yaz("   KıiaöŞ ppğO    ")
```

'kıiaöş ppğo'

Keywordler

```
def toplama_cikarma(birinci, ikinci, ucuncu):
    return birinci + ikinci - ucuncu

print(toplama_cikarma(3, 2, 1))

print(toplama_cikarma(birinci=3, ikinci=2, ucuncu=1))
```

```
# karışık sırada
print(toplama_cikarma(ucuncu=1, ikinci=3, birinci=2))

# argüman ismini yazmadan da olur
print(toplama_cikarma(3, ucuncu=1, ikinci=2))
```

```
4
4
4
4
```

Docstrings

Fonksiyonlar hakkında **bilgi verilen yer**dir. Örneğin hangi parametre ne işe yarıyor ve fonksiyonun özellikleri nelerdir gibi bilgiler burada verilir.

```
help(print)
```

```
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

```
# Fonksiyon tanımlandıktan sonra ilk satırda üç tırnak ile verilir.
def print_sum(val1, val2):
    """Verilen iki değişkeni toplayan fonksiyon."""
    print('sum: {}'.format(val1 + val2))

print(help(print_sum))
```

```
Help on function print_sum in module __main__:

print_sum(val1, val2)
    Verilen iki değişkeni toplayan fonksiyon.

None
```

Eğer yardım dokümanında daha detaylı bilgi vermek istenirse aşağıdaki gibi bir kullanım gerçekleştirilebilir.

```
# Daha detaylı yardım yazısı
def calculate_sum(val1, val2):
    """Verilen iki değeri toplar ve toplam değeri döndürür.

    Args:
        val1: Birinci değer.
        val2: İkinci değer.

    Returns:
        val1 ve val2 değerlerinin toplamını döndürür.

    """
    return val1 + val2
```

```
help(calculate_sum)
```

```
Help on function calculate_sum in module __main__:

calculate_sum(val1, val2)
    Verilen iki değeri toplar ve toplam değeri döndürür.

    Args:
        val1: Birinci değer.
        val2: İkinci değer.

    Returns:
        val1 ve val2 değerlerinin toplamını döndürür.
```

```
help(calculate_sum)
```

```
Help on function calculate_sum in module __main__:

calculate_sum(val1, val2)
    Verilen iki değeri toplar ve toplam değeri döndürür.

    Args:
        val1: Birinci değer.
        val2: İkinci değer.

    Returns:
        val1 ve val2 değerlerinin toplamını döndürür.
```

pass Durumu

pass ifadesi eğer fonksiyon bir şey **yapmayacaksa** kullanılır. Daha sonra doldurulmak üzere fonksiyona yazılabilir.

```
def my_function(some_argument):
    pass

def my_other_function():
    pass
```

Modüller ve Paketler

Modül bir Python kaynak kodudur örneğin .py dosyası şeklinde kaydedilen kodlar bir modüldür. Paket ise **__init__.py** dosyasına sahip olan bir dizindir. İçerisinde çeşitli modüller ve diğer paketler bulunabilir. Örneğin **Meyve1.py** dosyası içerisindeki fonksiyonu kullanmak için aşağıdaki yöntemler izlenebilir.

```
# Meyve1.py dosyası
def fonksiyon_meyve(x_meyve):
    y_meyve = x_meyve + 3
    return y_meyve
```

Aşağıdaki kodlarda önce Meyve1.py dosyası tamamen içeri alınmıştır. Bu durumda Meyve1.py dosyasındaki fonksiyonu çağırmak için "**.**" konulmuştur. İkinci durumda ise **Meyve1.py** içerisindeki **fonksiyon_meyve** fonksiyonu direkt olarak içeri alınmıştır. Bu nedenle fonksiyon ismi direkt kullanılabilmektedir. Birinci kullanımda Meyve1.py dosyasındaki tüm fonksiyonlar içeri alınmışken ikinci durumda sadece **fonksiyon_meyve** içeri alındığı için birinci durumun programda kapladığı yer artacaktır. Yani ihtiyaç duyulmayan fonksiyonun içeri alınmaması için ikinci kullanım tavsiye edilir.

Python programlarının tamamını tek bir dosyada yazmak karmaşıklığa neden olabilir. Paket ve modüllerin kullanım amaçları şu şekilde listelenebilir:

- Kodun bakım ve geliştirilmesi için,
- Tekrar kullanım için,
- İsim ve fonksiyonları ayrı yerde tutmak için,
- Bir sınıfı ya da fonksiyonu bulmak daha kolay olduğu için,

Paket İçeri Alma (Importing)

Eğer **Meyve** isminde bir sınıf varsa ve bu sınıfı **meyve.py** dosyasında tutulursa proje dosyasına almak için aşağıdaki adımlar uygulanabilir.

```
# meyve.py dosyası
class Meyve:
    def __init__(self, isim, renk):
        self.isim = isim
        self.renk = renk
```

Aşağıdaki kodlar ile **meyve.py** dosyasındaki sınıfa ulaşarak yeni bir sınıf oluşturulabilir.

```
from meyve import Meyve

# Kullanımı
myve = Meyve()
myve = meyve.Meyve("Muz", "Sarı")
meyve1 = meyve.Meyve
meyve1.renk = "Sarı"
meyve1.renk
```

```
'Sarı'
```

Buradaki anlatımın üzerine modüller ile ilgili detaylı bilgi almak için şu adresi (*https://realpython.com/python-modules-packages/*) takip edebilirsiniz.

Listeler

Listeler Python programlama dilindeki en fazla kullanılan veri tipidir. Hemen hemen tüm kodlarda listelerden faydalanılır. Listelerin içerisinde çeşitli veri tiplerinde veriler olabilir. Listeler ile alakalı önemli özellikler şu şekilde listelenebilir.[12]

- Listeler sıralıdır.
- Listeler rasgele nesneler içerebilir.
- Liste öğelerine dizine göre erişilebilir.
- Listeler rasgele derinliğe iç içe olabilir.
- Listeler kesilebilir.
- Listeler dinamiktir.

Örnek liste kullanımları aşağıda gösterilmiştir.

```
bos_liste = []
print('boş liste: {}, type: {}'.format(bos_liste, type(bos_liste)))
```

boş liste: [], type: <class 'list'>

```
sayi_listesi = [1,2,6,7]
karisik_liste = [0.2, 5, 'Python', 'Veri', 'Madenciliği', '!']
print('uzunluk: {} ve {}'.format(len(sayi_listesi), len(karisik_liste)))
```

uzunluk: 4 ve 6

```
# Listenin bir elemanını değiştirme
sayi_listesi[0] = 8
sayi_listesi
```

[8, 2, 6, 7]

Listeler içerisindeki değerlere ulaşmak için "[]" parantezler kullanılır.

```
my_list = ['Python','Veri', 'Madenciliği']
print(my_list[0]) # İlk elaman
# not: Python'da listeler 0'dan başlar.
print(my_list[2]) # 2. eleman
```

Python
Madenciliği

```
koordinatlar = [[12.0, 13.3], [0.6, 18.0], [88.0, 1.1]]  # iki boyutlu bir  dizi
print('birinci koordinatlar: {}'.format(koordinatlar[0]))
print('ikinci koordinatlar: {}'.format(koordinatlar[0][1]))
```

birinci koordinatlar: [12.0, 13.3]
ikinci koordinatlar: 13.3

```
my_list = [0, 1, 2, 3, 4, 5]
my_list[0] = 99
print(my_list)
```

```
# birinci değeri kaldırma
del my_list[0]
# kaldırdıktan sonra
print(my_list)
```

[99, 1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]

```
# ilk elemanı silme
del sayi_listesi[0]
sayi_listesi
```

```
[2, 6, 7]
```

```
# Yeni bir elemanı listenin sonuna ekleme
sayi_listesi.append(7)
sayi_listesi
```

```
[2, 6, 7, 7]
```

```
# Sayı listesinin eleman sayısı
len(sayi_listesi)
```

```
4
```

```
# Sayı listesinin son elemanını getir
sayi_listesi.pop()
```

```
7
```

```
# Bir değer listenin içerisinde var mıdır?
2 not in sayi_listesi
```

```
False
```

```
yazilim_dilleri = ['Java', 'C++', 'Go', 'Python', 'JavaScript']
if 'Python' in yazilim_dilleri:
    print('evet Python bulunmuştur!')
```

```
evet Python bulunmuştur!
```

```
if 6 not in [1, 2, 3, 7]:
    print('6 numarası yoktur!')
```

```
6 numarası yoktur!
```

```
my_list = ['Python', 'Veri', 'Madenciliği', 'Hoşgeldiniz']
my_list.remove('Hoşgeldiniz')
print(my_list)
```

```
# Eğer değerin listede olup olmadığından emin değilsek;
if 'Java' in my_list:
```

```
    my_list.remove('Java')
else:
    print('Java bu listede yoktur.')
```

```
['Python', 'Veri', 'Madenciliği']
Java bu listede yoktur.
```

```
# listelerin sıralanması
my_list = [10, 15, 20, 3, 4, 5]
my_list.sort()
my_list
```

```
[3, 4, 5, 10, 15, 20]
```

```
# Büyükten küçüğe sıralama için
my_list.sort(reverse=True)
my_list
```

```
[20, 15, 10, 5, 4, 3]
```

```
# sorted() fonksiyonu listeyi bozmadan sıralama gerçekleştirir
sorted_list = sorted(my_list)
my_list
```

```
[20, 15, 10, 5, 4, 3]
```

```
sorted_list
```

```
[3, 4, 5, 10, 15, 20]
```

```
# iki listenin birleştirilmesi
birinci_liste = [1,2,3]
ikinci_liste = [10,11,12]
ucuncu_liste = birinci_liste + ikinci_liste
ucuncu_liste
```

```
[1, 2, 3, 10, 11, 12]
```

```
# Liste değerlerinin toplanarak yazılması
for i in birinci_liste:
    for j in ikinci_liste:
        sonuc = i + j
        print(sonuc)
```

```
11
12
```

```
13
12
13
14
13
14
15
```

```
# listelerin uzatılması
first_list = ['et', 'tavuk']
second_list = ['patatesler',1 ,3]
first_list.extend(second_list)
print('birinci: {}, ikinci: {}'.format(first_list, second_list))
```

```
birinci: ['et', 'tavuk', 'patatesler', 1, 3], ikinci: ['patatesler', 1, 3]
```

```
# listenin ters çevrilmesi
my_list = ['a', 'b', 'et']
my_list.reverse()
print(my_list)
```

```
['et', 'b', 'a']
```

```
# iki boyutlu liste
iki_boyutlu = [[0,1,2],[10,11,12],[4,5,6]]
# iki boyutlu listenin elemanlarını yazdırma
for satir in iki_boyutlu:
    for karakter in satir:
        print(karakter)
```

```
0
1
2
10
11
12
4
5
6
```

```
# listeleri birbirine eşitleme
sayi_listesi = [1,2,3]
sayi_listesi2 = sayi_listesi
sayi_listesi2
```

```
[1, 2, 3]
```

```
# ikinci sayı listesinde bir eleman değişince birinci listede de  değişiklik olur.
sayi_listesi2[2] = 5
sayi_listesi
```

```
[1, 2, 5]
```

```
# eğer birinci listenin etkilenmesini istemezsek;
sayi_listesi3 = sayi_listesi[:]
sayi_listesi3
```

```
[1, 2, 5]
```

```
sayi_listesi3[1] = 3
sayi_listesi3
```

```
[1, 3, 5]
```

```
# birinci sayı listesi değişmemiştir.
sayi_listesi
```

```
[1, 2, 5]
```

```
sayi_listesi = [1, 2, 3]
# listeye liste ekleme
sayi_listesi.append([10, 20])
# listenin 3. elemanı yeni eklenen liste oldu.
sayi_listesi
```

```
[1, 2, 3, [10, 20]]
```

```
# yeni eklenen listenin birinci elemanına ulaşmak için
sayi_listesi[3][0]
```

```
10
```

```
# üçüncü elemena ekleme yapılırsa
sayi_listesi[3].append(["x","y","z"])
# üçüncü elemanın içerisinde bir liste daha oluşturulur
sayi_listesi
```

```
[1, 2, 3, [10, 20, ['x', 'y', 'z']]]
```

Koşullu Durumlar (if, elif, else)

Karar verme durumlarında eğer (**if**) ifadelerinin kullanımı Python programlama dilinde diğer dillere göre daha kolay bir yapı vardır. Öncelikle **bool** değişken tipinden tekrar bahsetmekte fayda vardır.

```
# bool değişkeni
print('True veya False değerlerinin değişken tipi: {}'.format(type(True)))
```

```
True veya False değerlerinin değişken tipi: <class 'bool'>
```

```
# 0 --> False, 1--> True demektir.
print('0: {}, 1: {}'.format(bool(0), bool(1)))
print('boş liste: {}, dolu liste: {}'.format(bool([]), bool(['Python'])))
```

```
0: False, 1: True boş liste: False,
dolu liste: True
```

```
print('1 == 0: {}'.format(1 == 0)) # İlk değer ikinci değere eşitse True değilse False yazdırır.
print('1 != 0: {}'.format(1 != 0)) # Eşit değilse True eşitse False,
print('1 > 0: {}'.format(1 > 0)) # Büyükse True değilse False,
print('1 > 1: {}'.format(1 > 1)) # Büyükse True değilse False,
print('1 < 0: {}'.format(1 < 0)) # Küçükse True değilse False,
print('1 < 1: {}'.format(1 < 1)) # Küçükse True değilse False
print('1 >= 0: {}'.format(1 >= 0)) # Büyük eşitse True değilse False,
print('1 >= 1: {}'.format(1 >= 1)) # Büyük eşitse True değilse False,
print('1 <= 0: {}'.format(1 <= 0)) # Küçük eşitse True değilse False,
print('1 <= 1: {}'.format(1 <= 1)) # Küçük eşitse True değilse False,
```

```
1 == 0: False
1 != 0: True
1 > 0: True
1 > 1: False
1 < 0: False
1 < 1: False
1 >= 0: True
1 >= 1: True
1 <= 0: False
1 <= 1: True
```

```
# and, or, not kullanımı
kedi_var_mi = True
kopek_var_mi = False
```

```
bos_liste = []
pi_sayisi = 3.14
# and kullanımı
print('Hem kedi hem köpek sahibi misiniz?: {}'.format(kedi_var_mi and
kopek_var_mi))
# or kullanımı
print('Köpek ya da kedi sahibi misiniz?: {}'.format(kedi_var_mi or kopek_
var_mi))
# not kullanımı
print('Kedi sahibi değil misiniz?: {}'.format(not kedi_var_mi))
```

```
Hem kedi hem köpek sahibi misiniz?: False
Köpek ya da kedi sahibi misiniz?: True
Kedi sahibi değil misiniz?: False
```

```
# if kullanımı
durum = True
if durum:
    print('durum doğru')

if not durum:
    print('durum doğru değil')
```

```
durum doğru
```

```
bos_liste = []
# Liste boş olmadığı zaman çalışır.
if bos_liste:
    print('Boş liste 1 değeri döndermediği için buraya girmez')
```

```
# if-elif-else
not_degeri = 85
if not_degeri >= 90:
    print('AA aldınız')
elif 90 > not_degeri > 80:
    print('BB aldınız')
# diğer tüm durumlar için
else:
    print('diğer...')
```

```
BB aldınız
```

Mantıksal sorgular ile ilgili daha detaylı bilgi için şu adresi (*https://realpython.com/python-conditional-statements/*) ziyaret edebilirsiniz.

Döngüler

Tekrarlayan işlemler için döngüler kullanılır. En sık kullanılan döngü **for** döngüsüdür. Kullanım örnekleri aşağıda verilmiştir.

```
# döngüler
for i in range(3):
    print(i)
```

```
0
1
2
```

```
# listenin elemanları içerisinde dönmek
liste = [1, 2, 3, "Python", "Veri", "Madenciliği"]
for eleman in liste:
    print(eleman)
```

```
1
2
3
Python
Veri
Madenciliği
```

```
# break durumu döngüyü durdurur.
for eleman in liste:
    # Veriye kadar yazdırmak istersek
    if eleman == "Veri":
        break
    print(eleman)
```

```
1
2
3
Python
```

```
# continue durumu döngüyü atlatır.
for eleman in liste:
    # Veri elemanını yazdırmak istemezsek
    if eleman == "Veri":
        continue
    print(eleman)
```

```
1
```

```
2
3
Python
Madenciliği
```

```
# enumerate ifadesi listenin elemanlarını numaralandırır.
for i, eleman in enumerate(liste):
    print('i: {}, eleman: {}'.format(i, eleman))
```

```
i: 0, eleman: 1
i: 1, eleman: 2
i: 2, eleman: 3
i: 3, eleman: Python
i: 4, eleman: Veri
i: 5, eleman: Madenciliği
```

```
# Sözlüklerde döngüler
sozluk = {'kedi_var_mi': True, 'pi_sayisi': 3.14, 'isim': 'Caner'}
for eleman in sozluk:
    print(eleman)
```

```
kedi_var_mi
pi_sayisi
isim
```

```
# anahtar ve değeri göstermek istersek
for anahtar, eleman in sozluk.items():
    print('{}={}'.format(anahtar, eleman))
```

```
kedi_var_mi=True
pi_sayisi=3.14
isim=Caner
```

```
# range durumu belirtilen sayıya kadar olan sayıları getirir.
for sayi in range(3):
    print(sayi)
```

```
0
1
2
```

```
# iki sayı arasındaki sayıları getirmek için
for sayi in range(3, 6):
    print(sayi)
```

```
3
```

```
4
5
```

```
for sayi in range(2, 9, 3):  # 2 den başla 9 a kadar 3 er 3 er atlayarak
    print(sayi)
```

```
2
5
8
```

Python Modülleri

Python programlama dilindeki önemli modüllerden birisi de **os** modülüdür.[13] os modülü genel olarak dosya ve klasör işlemlerini kolaylık sağlamaktadır. Bu modülün kullanımına ilişkin örnekler aşağıda paylaşılmıştır.

```
# os paketin çağrılması
import os
print(os.getcwd()) # çalışma klasörünü yazar
print(os.name) # işletim sistemimizi yazar 'nt' windows
print(os.sep) # dosya ayırıcısını yazar ('/, \\) vb
```

```
g:\Drive'ım\my_scripts\python-egitim
nt
\
```

```
os.chdir() # çalışma klasörünü değiştirir
os.listdir() # klasördeki dosyaları yazdırır
os.curdir #os.curdir, bir işletim sisteminde, o anda içinde bulunulan  dizini temsil eden karakter dizisi ne ise onun değerini barındırır. Bu değer çoğu işletim sisteminde '.' adlı karakter dizisidir
os.startfile() #os modülü içindeki startfile() adlı fonksiyonun görevi bilgisayarımızda bulunan herhangi bir dosyayı, ilişkilendirilmiş olduğu programla açmaktır.
os.mkdir() #os modülünün mkdir() fonksiyonu yeni dizinler oluşturabilmemizi sağlar.
```

```
# datetime modülü
import datetime as dt

simdiki_zaman = dt.datetime.now()
print('yerel zaman: {}'.format(simdiki_zaman))

# Gün ay yıl gibi ayırmak için:
print('{} {} {} {} {} {}'.format(simdiki_zaman.year, simdiki_zaman.month,
```

```
                                    simdiki_zaman.day, simdiki_zaman.hour,
                                    simdiki_zaman.minute, simdiki_zaman.
second))

print('tarih: {}'.format(simdiki_zaman.date()))
print('saat: {}'.format(simdiki_zaman.time()))
```

```
yerel zaman: 2021-03-19 00:08:01.899815
2021 3 19 0 8 1
tarih: 2021-03-19
saat: 00:08:01.899815
# strftime() tarihi formatlamak için kullanılır.
formatted1 = simdiki_zaman.strftime('%Y/%m/%d-%H:%M:%S')
print(formatted1)
```

```
formatted2 = simdiki_zaman.strftime('tarih: %Y-%m-%d time:%H:%M:%S')
print(formatted2)
```

```
2021/03/19-00:08:01
tarih: 2021-03-19 time:00:08:01
```

```
# strptime() stringi datetime objesine çevirmek için kullanılır.
tarih = dt.datetime.strptime('2000-01-01 10:00:00', '%Y-%m-%d %H:%M:%S')
print('tarih: {}'.format(tarih))
```

```
tarih: 2000-01-01 10:00:00
```

```
# timedelta tarih hesaplamaları yapmak için kullanılır.
yarin = simdiki_zaman + dt.timedelta(days=1)
print('yarin : {}'.format(yarin))

delta = yarin - simdiki_zaman
print('yarin - simdiki_zaman = {}'.format(delta))
print('gün: {}, saniye: {}'.format(delta.days, delta.seconds))
print('toplam saniye: {}'.format(delta.total_seconds()))
```

```
yarin : 2021-03-20 00:08:01.899815
yarin - simdiki_zaman = 1 day, 0:00:00
gün: 1, saniye: 0
toplam saniye: 86400.0
```

```
# random modülü rassal sayı üretmek için kullanılır.
import random

rand_int = random.randint(1, 100)
```

```
print('1-100 arası rastgele sayılar: {}'.format(rand_int))

rand = random.random()
print('0-1 arası rastgele sayılar: {}'.format(rand))
```

```
1-100 arası rastgele sayılar: 61
1.1 arası rastgele sayılar: 0.8649539941761258
```

```
# seed değerini ayarlayarak aynı rassal sayılar üretebiliriz.
import random
random.seed(5)  # seed i ayarla
# 10 tane rastgele sayı yazdıralım
for _ in range(10):
    print(random.random())
```

```
0.6229016948897019 0.7417869892607294 0.7951935655656966 0.9424502837770503
0.7398985747399307 0.922324996665417 0.029005228283614737 0.46562265437810535
0.9433567169983137 0.6489745531369242
```

```
# re modülü düzenli ifadeler için kullanılır.
import re
ifade = 'aldajasd as*1unada'
# "r" raw format demektir
aranan_metin = r'(as)'
eslesen = re.search(aranan_metin, ifade)
print('eslesen: {}'.format(eslesen))
print('eslesen.group(): {}'.format(eslesen.group()))
# ifadenin içerisinde sayı var mı?
sayi_eslestir = r'[0-9]'
eslesen_sayilar = re.findall(sayi_eslestir, ifade)
print('sayilar: {}'.format(eslesen_sayilar))
```

```
eslesen: <re.Match object; span=(5, 7), match='as'>
eslesen.group(): as
sayilar: ['1']
```

```
# txt dosyası okuma
with open('dosya.txt', 'r') as dosya_ac:
    for satir in dosya_ac:
        print(satir.strip())
```

```
Merhaba!
Herkese
```

```
# dosyaya yazma
with open("dosya2.txt", 'w') as dosya_ac2:
    dosya_ac2.write('Merhaba ikinci dosya.')
```

Özet

Bu bölümde Python programlama diline hızlı bir giriş yapılmıştır. Bahsedilen konuların tüm detayları verilmeden giriş seviyesinde bilgiler ile bölüm tamamlanmıştır. Burada gösterilen bilgiler kitaptaki kodların takibi için yeterli gelecektir. Ancak bilgisayar programcılığı için yeterli gelmeyecektir.

Kaynaklar

1. "YazBel Belgeleri", https://belgeler.yazbel.com/.
2. "Yeni Başlayanlar İçin Python", http://www.idefix.com/Kitap/Yeni-Baslayanlar-Icin-Python/Egitim-Basvuru/Bilgisayar/urunno=0000000694609?gclid=CjwKCAiA8rnfBRB3EiwAhrhBGvtC3-OQEYO058lov4qtFRvJ5hyM1XbF80JosZXy4SBKxn1nGtQKoBoC33EQAvD_BwE.
3. Shaw, Z. A. Learn Python the Hard Way: A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code, Addison-Wesley (2013).
4. Lutz, M. Learning Python, 5th Edition, Fifth edition, O'Reilly Media, Beijing (2013).
5. "Learning to program with Python 3 (py 3.7) - YouTube", https://www.youtube.com/.
6. "Python Programming Tutorials - YouTube", https://www.youtube.com/.
7. "Python Tutorial for Beginners (Full Course) - YouTube", https://www.youtube.com/.
8. "Towards Data Science", https://towardsdatascience.com/.
9. Python, R. "Python Tutorials – Real Python", https://realpython.com/.
10. "Host, run, and code Python in the cloud: PythonAnywhere", https://www.pythonanywhere.com/.
11. "Built-in Functions — Python 3.9.2 documentation", https://docs.python.org/3/library/functions.html.
12. Python, R. "Lists and Tuples in Python – Real Python", https://realpython.com/python-lists-tuples/.
13. "os — Miscellaneous operating system interfaces — Python 3.9.2 documentation", https://docs.python.org/3/library/os.html.

BÖLÜM 3

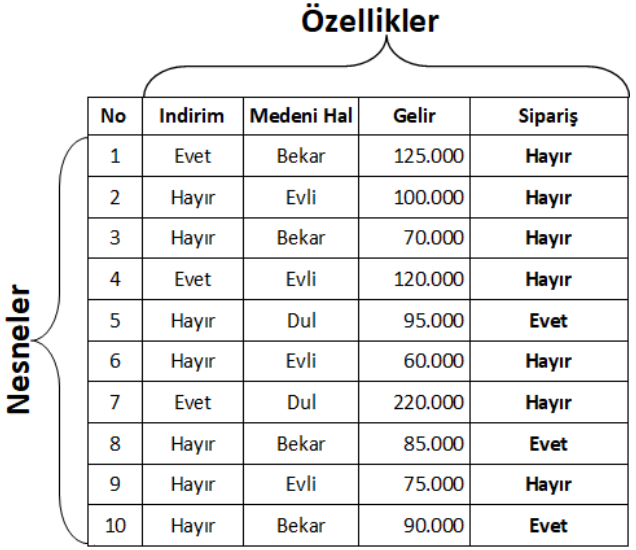
VERİ HAZIRLAMA VE ÖN İŞLEME İŞLEMLERİ

Bu bölümde **veri hazırlama** ve **veri önişleme** konuları anlatılacaktır. Bu bölümdeki konular iyi bir veri madenciliği uygulaması için gerekli ve önemli konulardır. Kısaca belirtmek gerekirse bu bölümde, verinin tanımı yapılacak, değişik veri tipleri anlatılacak ve veri madenciliğinde kullanılan veri ön işleme yöntemlerinden bahsedilecektir.

Veri Nedir?

Veri madenciliğinin kalbinde veri vardır. O nedenle verinin ne olduğunu ya da ne olmadığını iyi bilmek gerekir. Verinin birçok alanda birçok tanımı olabilir. Veri, veri madenciliğinde özellik ve veri nesnelerinden oluşan bir koleksiyon olarak tanımlanabilir. Veri seti ise birtakım verinin bir birlikte oluşturduğu bir koleksiyon olarak söylenebilir. Veri setinde bulunan her bir gözlem değerine **veri, kayıt, nokta, durum, örnek, gözlem, girdi** gibi isimler verilir. Veri madenciliğinde gözlem değerleri genellikle satırlarda gösterilir. Sütunlara ise gözlemin etkilendiği **özellikler** (attributes) ya da **değişkenler** (variables) yazılır. Özellikler veriyi açıklayan veya bir karakteristiğini ortaya koyan bilgi demektir. Bu nedenle özelliklere karakteristik, değişken, boyut, alan olarak da isim verilebilir. Örneğin bir insanın göz rengi o insanın bir özelliği olarak verilebilir. Ya da insanın boyu, kilosu, ya da yaşı yine o insanın özelliğini ya da karakteristiğini gösterir. Belirtilen bu gösterimler aşağıdaki görselde bulunan tabloda gösterilmiştir. Tabloda bir alışveriş marketine gelen 10 müşteri ile ilgili bilgiler bulunmaktadır. Veri setinde sonuç olarak müşterinin sipariş verip vermediği gösterilmiştir. Bu nedenle bu veri seti bir **etiketli**, yani **sınıflı** bir veri setidir. Sipariş özelliği diğer özelliklerden farklıdır, sonuç ya da karar özelliği olarak veri setinde yer almıştır.

Bu özelliğe kitapta **karar değişkeni** de denilecektir.



No	Indirim	Medeni Hal	Gelir	Sipariş
1	Evet	Bekar	125.000	**Hayır**
2	Hayır	Evli	100.000	**Hayır**
3	Hayır	Bekar	70.000	**Hayır**
4	Evet	Evli	120.000	**Hayır**
5	Hayır	Dul	95.000	**Evet**
6	Hayır	Evli	60.000	**Hayır**
7	Evet	Dul	220.000	**Hayır**
8	Hayır	Bekar	85.000	**Evet**
9	Hayır	Evli	75.000	**Hayır**
10	Hayır	Bekar	90.000	**Evet**

Şekil 3-1: Özellikler ve nesneler

Veri seti oluşturmak için veriler çeşitli kaynaklardan alınabilir. Örneğin bir SQL dosyası olarak işletmenin veri tabanından alınabilir. Ya da internet sitesinden **.csv** dosyası hâlinde indirilebilir.

Veri Toplama

Veri toplama yöntemleri aşağıdaki gibi söylenebilir.

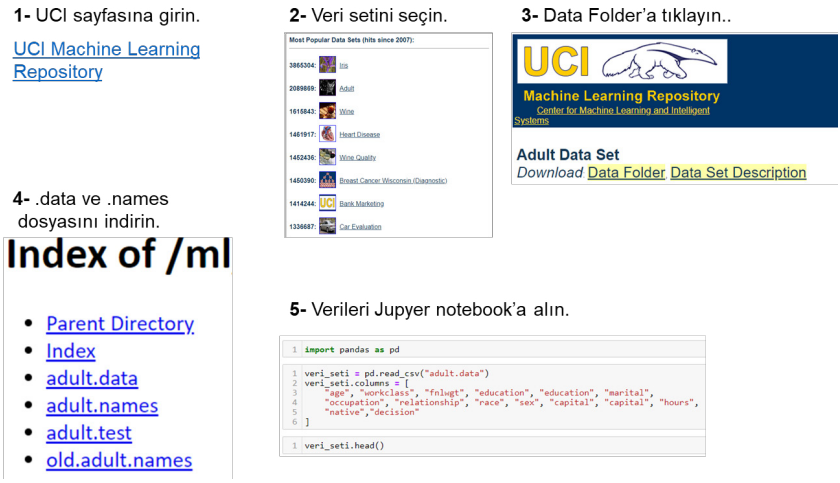
- **Gözlem**: Bir olay, durum hakkında gözlem veya deney yaparak veriler elde edilebilir.
- **Anket & Soru Formları**: Özellikle sosyal bilimlerde kullanılan bir yöntemdir. Oluşturulan anket soruları katılımcılara sorulur ve alınan cevaplara göre veriler toplanır. Bu yöntemde anketler yüz yüze yapılabileceği gibi internetten veya telefon ile de yapılabilir.
- **Veri seti yayınlayan siteler**: Veri setlerini depolayan ve hizmete sunan sayfalar sayesinde de veriler toplanabilir. Örneğin Kaggle, UCI veri seti depolamaktadır.
- **Açık kaynak verileri (GitHub, Kaggle, Scraping Websites, Parsing Data)**: İnternette açık kaynaklı olan veriler için internetten veri çekme (web scraping)

uygulamaları ile veri alınabilir. Alınan dosyalar CSV, Metin Dosyası, HTML, XML, Excel, JSON dosyaları halinde saklanarak veri madenciliği

- **Veri tabanları**: İlişkisel veri tabanlarında depolanan verilerin SQL kodları ile toplanması sağlanabilir.

Veri madenciliği çalışmalar için veri setlerinin hazır bulunduğu en fazla kullanılan depolardan birisi **UCI makine öğrenmesi** (UCI Machine Learning Repository) deposudur. UCI deposuna veriler, California Üniversitesi tarafından sağlanmaktadır. Şu anda UCI deposunda değişik alanlarda 587 adet veri seti bulunmaktadır. Bu kısımda örnek bir veri setinin UCI veri seti deposundan indirilerek kullanıma hazır hale getirilmesi gösterilecektir. Örnek olarak "**Adult**" veri seti için bir gösterim yapılacaktır.

Öncelikle *https://archive.ics.uci.edu/ml/index.php* adresinden incelenecek veri seti sayfasına gidilir. Ardından "**Data Folder**" içerisindeki "**adult.data**" ve "**adult.names**" dosyaları indirilerek bir klasöre koyulur. Klasör içerisinde bir Jupyter notebook açılarak veri seti kullanıma hazır hâle getirilmiş olur. İlgili adımlar şekilde gösterilmiştir.



Şekil 3-2: UCI deposundan veri setlerinin indirilip kullanımı

Özelliklerine Göre Veri Tipleri

Veri analizinde göz önünde bulundurulması gereken bir diğer durum da verinin tipidir. Veri tipleri genel olarak 4 grupta toplanabilir. Bunlar:

1. **Nominal**: Bu tip veriler arasında eşitlik ya da eşitsizlik söz konusudur. Nominal veriler olarak paket programlarda geçtiği için burada da nominal olarak isimlendirilmiştir. Nominal veri tipi daha fazla ifadesi ile kullanılmazlar. Örneğin, TC kimlik no, ID, göz rengi vb. Nominal veriler eğer 2 değer alabiliyorsa bu veri tipine **ikili veri tipi** (binary variable) denilir.
2. **Sıralama Ölçeği (Ordinal)**: Veriler arasında bir büyüklük küçüklük ilişkisi varsa veri tipine **sıralama tipi** denilir. Bu tip veriler daha fazla ifadesi ile kullanılabilir. Örneğin uzunluk (uzun, orta boy, kısa), eğitim derecesi (ilkokul, ortaokul, lise) vb.
3. **Aralıklı (interval)**: Aralıklı veri tipinde bir döngü söz konusudur ve 0 noktası olmama durumunu ifade etmez. Örneğin sıcaklığın Fahrenhayt veya Celcius ile ölçülmesi interval veri tipine örnektir. Çünkü 0 derece sıcaklığın olmadığını ifade etmez. Ya da 100 derece sıcaklık 50 derece sıcaklığın 2 katıdır diyemeyiz. Bu nedenle interval veri tipinde oransallık söz konusu değildir.
4. **Oransal (Ratio)**: İnterval veri tipine benzer şekildedir. Sadece burada başlangıç noktası 0'dır ve 0 noktası yokluk ifade eder. Örneğin sıcaklığın Kelvin ile ölçülmesi, uzunluk (cm) ölçüsü vb. 0 cm uzunluk yokluk ifade ettiği için oransal veri tipine örnektir.

Sayılabilir Olmasına Göre Veri Tipleri

Bu gruplandırmanın dışında verileri sayılabilir olma özelliklerine göre 2'ye ayırabiliriz.

1. **Sayılabilir (kantitatif)**: Verilerin nümerik olarak ifade edilebilmesi demektir. Diğer bir adı da **nümerik veriler**dir. Aralıklı veya oransal veri tipi bu gruba girer. Sayılabilir veriler kendi içerisinde kesikli ve sürekli olarak 2'ye ayrılabilir. Kesikli veriler, belirli bir aralıkta sonlu sayıda değer alabilen verilerdir. Örneğin bir bankaya bir saat içerisinde gelen müşteri sayısı kesikli veridir. Sürekli veriler ise belirli bir aralıktaki tüm değerleri alabilen verilerdir. Örneğin bir kişinin boy ölçüsü 1,75 m olabilir. Ya da 1,85231 m olabilir. Yani bir aralıktaki bütün değerleri alabilir. O nedenle uzunluk veri tipi sürekli sayılabilir bir veri tipidir.
2. **Sayılamayan (kalitatif)**: Nümerik olarak ifade edilemezler. **Kategorik veriler** olarak da bilinir. Nominal, ikili değişkenler veya ordinal veriler bu gruba girer.

Özet olarak, veri tipleri aşağıdaki tabloda gösterildiği gibi verilebilir.

	Veri Tipi	**Açıklama**	**Örnek**
Kategorik (Kalitatif) Veriler	Nominal	Eşitlik ya da eşitsizlik durumu söz konusu (=, ≠)	PK Kodu, göz rengi, cinsiyet vb.
	Ordinal	Aralarında büyüklük küçüklük ilişkisi var (<,>,≤,≥)	Not (AA, BA, BB), Kalite Derecesi (İyi, Orta, Kötü)
Nümerik (Kantitatif) Veriler	Interval	Toplama çıkarma gibi işlemler yapılabilir (+, -)	Takvim günleri, Santigrat ya da Fahrenhayt gibi sıcaklık ölçüsü
	Ratio	İki değer arasındaki oran anlamlıdır. (\times, /)	Kelvin sıcaklığı, yaş, ağırlık, uzunluk gibi.

Tablo 3-1: Veri tipleri

Veri analizinden önce veri setindeki veri tiplerini ortaya koymak incelemek son derece önemlidir. Çünkü ileride veri madenciliği çalışmasında bazı veriler için yeniden kodlama yapmak gerekecek ve ordinal veriler aralarında büyüklük ilişkisinin olduğu veriler olduğu için **1-2-3** gibi kodlanabilirken nominal veriler aralarında büyüklük ilişkisi olmadığı için **0-1** olarak kodlanabilecektir. Örneğin göz rengi değişkenindeki mavi değeri 1 ya da 0 olarak kodlanacak. Yani göz renginin mavi olması veya olmaması durumu söz konusu olacak. Eğer göz rengi mavi için 1, kahverengi için 2, siyah için 3 olarak kodlanırsa modelin yanlış yorumlanması tehlikesi bulunur. Çünkü göz renkleri arasında büyüklük ilişkisi yoktur. Aralarında eşitlik ya da eşitlik olmama durumu söz konusudur. Bu nedenle göz rengi için yeni değişkenler oluşturarak her bir renk için 0-1 yeniden kodlaması yapılır. Bu konu ileride **one-hot encoding** diye geçecek ve daha detaylı şekilde gösterilecektir. Veriler arasındaki dönüşüm için aşağıdaki tablo kullanılabilir.

Kategorik	Nominal	Her bir değer başka bir değerle değiştirilir.
Ordinal	Değerler sıralama olacak şekilde değiştirilir.	İyi, orta, kötü değerlerinin 3,2,1 ile değiştirilmesi
Nümerik	Interval	Yeni_değer=a × eski_değer+b
Ratio	Yeni_değer = a × eski_değer	Metre ve feet arasındaki dönüşüm

Tablo 3-2: Veri dönüşümleri

Temel Tanımlayıcı İstatistikler

Veri ön işlemeden önce verilerin daha iyi anlaşılması için temel birtakım istatistiksel işlemlerin yapılması gerekir. Tanımlayıcı kavramı yerine kimi kaynaklarda **betimleyici** kavramı da kullanılmaktadır. İngilizcede bu kavram "**descriptive**" olarak bilinir. Tanımlayıcı istatistik ile veri seti üzerinde bir çıkarım gerçekleştirilmez. Bu kısımda öncelikle ana kütle ve örneklem üzerinde durmakta fayda vardır.

- **Ana kütle**: Gözlem yapılan olay hakkındaki tüm durumların ortaya koyulduğu gruptur.

- **Örneklem**: Ana kütle içerisinden seçilen bir gruptur.

Merkezî Eğilim Ölçüleri

Verinin orta noktası ile ilgilenir. Örnek olarak mod, medyan, ortalama gibi ölçüler verilebilir.

Ortalama: Ortalama olarak genellikle aritmetik ortalama kullanılır. Aritmetik ortalamadan başka geometrik, harmonik ortalama olarak da verilebilir. Aritmetik ortalamada verilerin toplamı veri sayısına bölünür. Aritmetik ortalama aşağıdaki formül ile hesaplanabilir. Kitapta ortalama denildiğinde aritmetik ortalama kastedilecektir.

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N} = \frac{x_1 + x_2 + \cdots + x_N}{N}$$

Veriler arasında ağırlıkların olması durumunda ortalama aşağıdaki gibi hesaplanır.

$$\bar{x} = \frac{\sum_{i=1}^{N} w_i x_i}{N \sum i = 1^N w_i} = \frac{w_{1x_1} + w_2 x_2 + \cdots + w_N x_N}{w_1 + w_2 + \cdots + w_N}$$

Medyan: Veriler küçükten büyüğe sıralandığında ortada kalan sayıya medyan adı verilir.

Mod: Veri içerisinde en fazla tekrar eden değere mod adı verilir.

Merkezi eğilim ölçüsü olarak ortalama, mod ya da medyandan hangisinin kullanılacağı farklı durumlarda değişkenlik gösterebilir. Örneğin eğer veri setindeki değerler arasında uçurum denilebilecek farklar varsa aritmetik ortalama yerine medyan kullanılmalıdır gibi birtakım kurallar vardır.

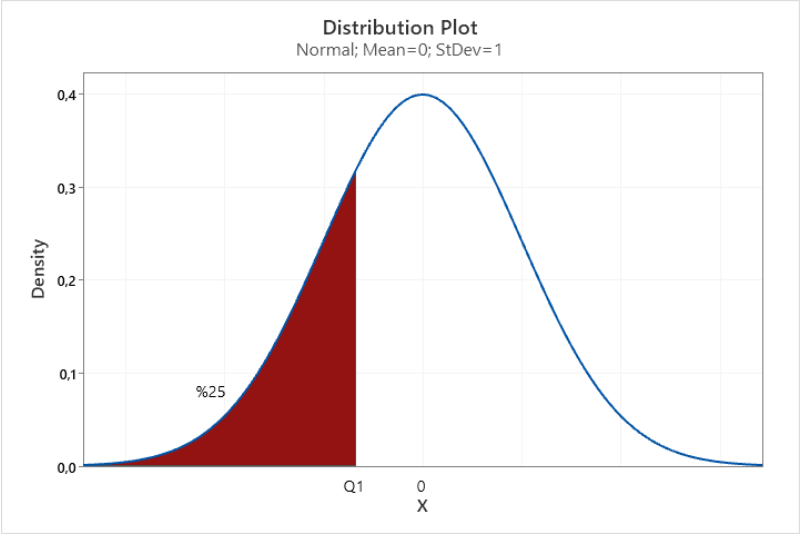
Örnek verilmesi gerekirse, 9, 10, 10, 11 ve 0, 10, 10, 20 gibi iki seri olduğunu varsayılsın. İki seri merkezi eğilim ölçüleri ile açıklanmak istenirse iki serinin de ortalamasının 10 olduğu görülür. Bu noktada analiz yetersiz kalır. Bu iki seriyi birbirinden ayırt edebilmek için dağılım ölçülerine ihtiyaç duyulur.

Dağılım Ölçüleri

Verinin nasıl dağıldığını gösterir. Örnek olarak aralık, varyans, standart sapma gibi ölçüler verilebilir.

Aralık (Ranj): Verinin maksimum ve minimum değerleri arasındaki değerleridir.

Kartiller (Quartiles): Sıralanmış verilerin aynı oranda bölünmesi ile elde edilir. Örneğin verileri 4'e bölündüğünde birinci çeyrek birinci kartil (Q_1), ikinci kartil (Q_2), Q_3 ve Q_4 değerleri elde edilir. Q_2 değeri aynı zamanda medyan değerini verir. Aşağıdaki şekilde kartiller gösterilmiştir. Q_3 ile Q_1 değerleri arasındaki fark çeyrekler arası fark (**IQR**) değeri olarak bilinir ve aşağıdaki formül ile hesaplanır.



Şekil 3-3: Kartillerin gösterimi

$$IQR=Q_3-Q_1$$

Varyans ve Standart Sapma

Verinin dağılımına bakmak için sadece IQR değerine bakmak yeterli değildir. Verilerin dağılımı ile alakalı diğer ölçüler varyans ve standart sapma ölçüleridir.

Standart sapma: Verinin ortalamadan ne kadar sapıldığının ölçüsüdür. Standart sapma 0 ise gözlem değerleri ortalamadan sapmamıştır denir. Küçük standart sapma değeri gözlem değerlerinin ortalamadan çok sapmadığı anlamına gelir. Standart sapma σ ile gösterilir. Varyans σ^2 ise standart sapmanın karesidir ve aşağıdaki formülde gösterildiği gibi hesaplanır.

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

Örneğin verilerin 30,36,47,50,52,52,56,60,63,70,70,110 olarak verilen bir dizi için standart sapma ve varyans değerlerinin hesabı aşağıdaki gibi yapılır:

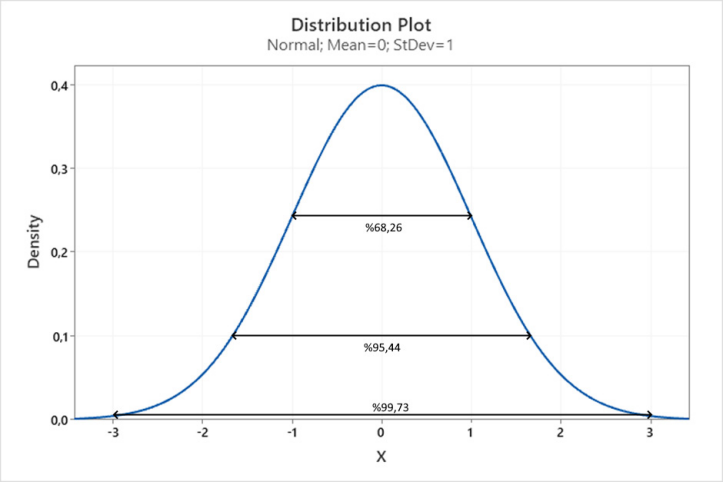
$$\bar{x} = \frac{30 + 36 + 47 + 50 + 52 + 52 + 56 + 60 + 63 + 70 + 70 + 110}{12} = 696/12 = 58$$

$$\sigma^2 = \frac{1}{12}\left(30^2 + 36^2 + \cdots + 110^2\right) - 58^2 = 379{,}17$$

Normal Dağılım

En yaygın kullanılan dağılım tipidir. Bir gözlem değişkenine ilişkin yeterli sayıda ölçüm yapılırsa verilerin normal dağılıma uygun dağılacağı varsayılır. Bu nedenle istatistik biliminde önemli bir yer tutar. **Çan eğrisi** olarak da bilinir. Eğrinin altında kalan alan olayın olma olasılığını gösterir. Standart bir normal dağılımda ortalama 0, standart sapma 1 olarak belirlenir ve -3 ile +3 arasındaki değerler %99 oranındadır. Ortalamadan 1 standart sapma sapmanın oranı da %68 olarak ölçülmüştür. Bu ölçümler aşağıdaki grafikte verilmiştir. Buna göre;

- μ- σ ile μ+ σ arasındaki veriler toplam verinin %68'dir.
- μ-2 σ ile μ+2 σ arasında %95 veri bulunur.
- μ-3 σ to μ+3 σ arasında %99.7 veri bulunur.



Şekil 3-4: Normal dağılım özellikleri[1]

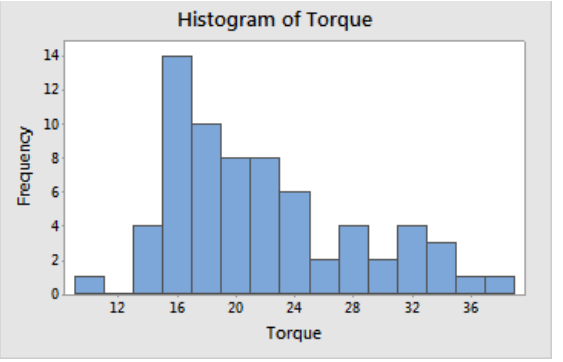
XN(μ,σ) şeklinde dağılan bir rassal değişken ve Z standart normal dağılımı yani ortalaması 0 standart sapması 1 olan bir değişken olursa Z değişkeni aşağıdaki formül ile hesaplanabilir.

$$Z = \frac{X - \mu}{\sigma}$$

Denklemdeki Z değeri ayrıca **z-skoru** olarak da bilinir. Bu formül bize standart normal dağılıma uyan bir örneklem oluşturabileceğimizi gösterir. Bir serinin dağılımı hakkında bilgi alabilmek için serinin grafiğini çizdirmek iyi bir yoldur. Bu bölümde çizilebilecek grafikler ile ilgili kısa bilgilendirme yapılacaktır. Daha detaylı bilgi **Matplotlib ile veri görselleştirme** bölümünde gösterilecektir.

Kutup Grafikleri

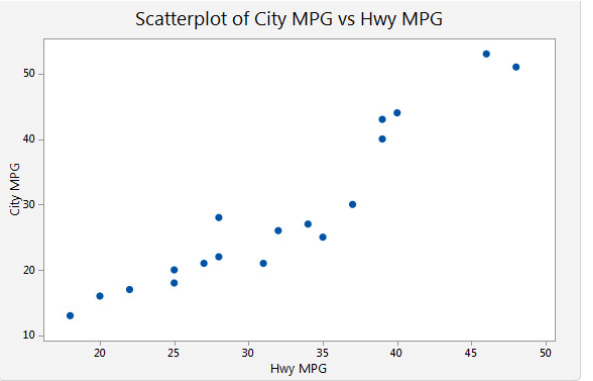
Veri hakkında bilgi almak için öncelikle uygun bir grafikle verinin analizinin yapılması gereklidir. Kutup grafikleri (histogram) ile verilerin dağılımı görülebilir. Genellikle X eksenindeki değerler verinin aldığı değerleri Y ekseninde ise o değerlerin sayısı gösterilir. Örnek bir **histogram grafiği** aşağıda gösterilmiştir.



Şekil 3-5: Histogram grafiği örneği

Dağılım Grafiği

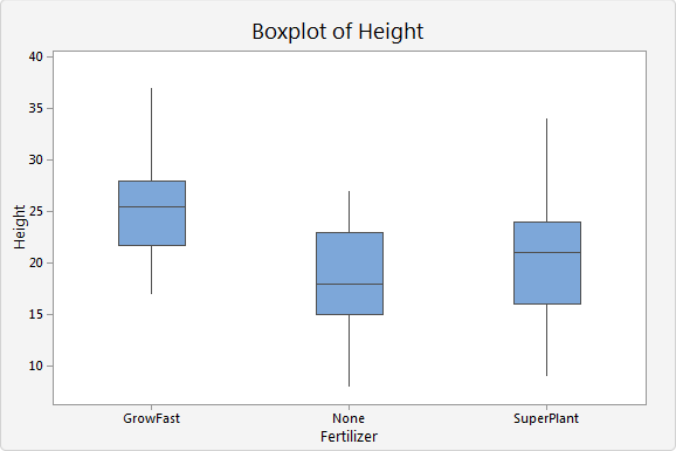
İki değişken arasındaki ilişkinin gösterilmesi için dağılım grafiği çizilir. Dağılım grafiği İngilizcede **scatter plot** olarak bilinir. Dağılım grafiği ile her **x** değerine karşılık gelen y değeri grafik üzerinde gösterilir. Bu grafik sayesinde aykırı noktalar rahatlıkla görülebilir. Örnek bir **dağılım grafiği** aşağıdaki şekilde gösterilmiştir.



Şekil 3-6: Dağılım grafiği örneği

Kutu Grafikleri

Verinin dağılımını gösteren diğer bir grafik türüdür. Değişkenin maksimum, minimum noktası, medyan değeri ya da Q_1, Q_3 değerleri ya da IQR değerleri gibi bilgileri içeren bir grafik türüdür. Kutu grafikleri aykırı veri analizinde kullanılabilir. Aykırı veriler belirlerken ortalamadan 1,5IQR uzaklıktaki veriler aykırı veri olarak belirlenebilir. Aynı şekilde 1,5 IQR değerinden küçük veriler de aykırı veri olarak belirlenebilir. Bu işlemler için kutu grafiği (boxplot grafiği) çizdirmekte fayda vardır. Aşağıdaki örnek kutu grafiğinde minimum, maksimum nokta, Q_1,Q_2,Q_3 gibi değerler işaretlenmiştir.

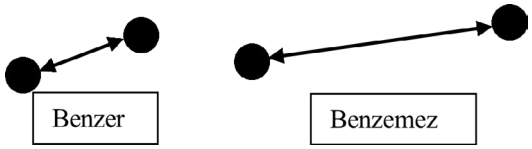


Şekil 3-7: Kutu grafiği örneği

Benzerlik Ölçüleri

Veri madenciliği uygulamalarında özellikle kümeleme çalışmalarında, aykırı veri analizinde, en yakın komşuluk sınıflandırmasında veriler arasındaki uzaklıkların veya yakınlıkların hesaplanması gereklidir. Bu işlem için veriler arasında **benzerlik** veya **benzeşmezlik ilişkileri** kurulur. Örneğin bir market sahibi müşterilerini kümelere ayırmak istediğinde müşteriler arasında benzerlik ölçüsü belirlemelidir. Bu ölçü sayesinde marketin müşterileri arasında benzer ya da benzemez yorumları yapabilmesi mümkün olur. Benzerlik ölçüsü sayesinde iki müşterinin birbirine ne kadar benzediği ortaya koyulur ve ardından belirlenen bir eşik değeri ile müşteriler arasına çizgiler çizilerek müşteriler birbirinden ayrılmış olur. Örneğin, market için kaliteyi ön plana alan müşteri portföyü ya da fiyatı öncelik hâline getiren müşteri portföyü gibi kümeler oluşturulabilir.

Benzerlik ölçüsü 0 ile 1 arasında bir değer alacak şekilde hesaplanır. Ardından ölçünün 1'e yakın olması verilerin birbirine benzediği anlamına gelir. Benzerlik ölçüsünün tam tersi olacak şekilde uzaklık ölçüsü de belirlenebilir. Bu durumda da ölçünün değeri 0'a yakın çıkması iki verinin benzer olduğu sonucuna götürür. Bu durumun detayları aşağıdaki şekilde gösterilmiştir.



Şekil 3-8: Benzerlik ölçüleri

Veriler arasında benzerliklerin hesaplanması için öncelikle veri matrisine ihtiyaç duyulur. Veri matrisi aşağıdaki gibi oluşturulur.

Veri Matrisi: $\begin{bmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{bmatrix}$

Uzaklık Matrisi: $\begin{bmatrix} 0 & d(1,2) & \cdots & d(1,p) \\ d(1,2) & 0 & \cdots & d(2,p) \\ \vdots & \vdots & \ddots & \vdots \\ d(n,1) & d(n,2) & \cdots & 0 \end{bmatrix}$

Benzerlik Ölçüsü: $s(i,j) = 1 - d(i,j)$

Burada, p adet özellikten oluşan bir veri seti bulunmaktadır. d(i,j) i. veri ile j. veri arasındaki **mesafeyi (distance)** gösterir. s(i,j) ise i. veri ile j. veri arasındaki **benzerliği (similarity)** gösterir.

Nümerik Verilerin Uzaklığı

Veri madenciliğinde esas olarak nümerik veriler ile ilgilenilir. Bu nedenle nümerik verilerin **uzaklığını** incelemek gereklidir. Örneğin aşağıda bilgileri verilen kişilerin birbirlerine olan benzerliklerinin hesaplanması için nümerik verilerin uzaklığı hesaplamalarının yapılması gereklidir. Örnek olarak aşağıdaki tabloda bulunan 3 kişinin hangilerinin birbirine daha fazla benzediği ile ilgili ilk bakışta Hasan ve Hüseyin'in birbirine daha fazla benzediği söylenebilir. Bu durumun veriler arasındaki uzaklık ölçüleri ile de hesaplamak istenirse nümerik verilerin uzaklık ölçülerinden faydalanılabilir.

	Yaş	Gelir
Hasan	30	4000 TL
Hüseyin	28	3500 TL
Mahmut	59	12000 TL

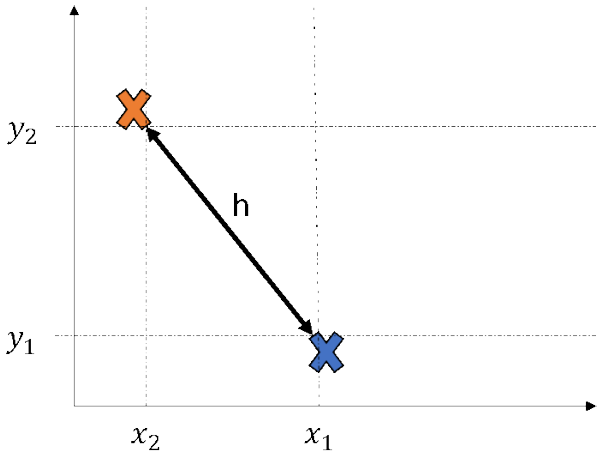
Tablo 3-3: Örnek bir uygulama veri seti

Nümerik verilerin uzaklık hesabı için farklı yöntemler vardır. Bu yöntemler ile ilgili bilgiler aşağıda verilmiştir.

Öklid Uzaklığı: En sık kullanılan uzaklık ölçüsüdür. Geometriden bilinen 2 nokta arasındaki mesafe mesela **öklid ölçüsü** ile ifade edilir. Her bir boyutun değerin kendi içerisinde farkı alınır ardından karesi alınarak toplanır. Bu toplamların karekökü iki veri arasındaki mesafeyi verir. Böylece veriler arasındaki mesafelerin negatif olmasının önüne geçilmiş olunur. Genel formülü aşağıda verilmiştir.

$$d\big((x_1, x_2, \dots, x_n), (y_1, y_2, \dots, y_n)\big) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Aşağıdaki grafikte öklid mesafesi gösterilmiştir.



Şekil 3-9: Örnek Öklid Mesafesi

Öklid mesafesinin hesaplanması için çeşitli yöntemler olmakla birlikte Python üzerinde **Scipy** kütüphanesinden faydalanılarak da hesaplanabilir.

```
from scipy.spatial import distance
hasan = [30, 4000]
huseyin = [28, 3500]
mahmut = [59, 12000]
hh_oklid_mesafesi = distance.euclidean(hasan, huseyin)
hm_oklid_mesafesi = distance.euclidean(hasan, mahmut)
mh_oklid_mesafesi = distance.euclidean(mahmut, huseyin)
print("Hüseyin ile Hasan arasındaki uzaklık: {}, \nHüseyin ile Mahmut
arasındaki uzaklık: {}, \nMahmut ile Hüseyin arasındaki uzaklık: {} \n".
format(hh_oklid_mesafesi, hm_oklid_mesafesi, mh_oklid_mesafesi))
```

```
Hüseyin ile Hasan arasındaki uzaklık: 500.00399998400013,
Hüseyin ile Mahmut arasındaki uzaklık: 8000.052562327325,
Mahmut ile Hüseyin arasındaki uzaklık: 8500.056529223792
```

Manhattan Uzaklığı: Diğer ölçüt ise Manhattan uzaklığıdır. İsminin Manhattan olmasının nedeni Manhattan şehrinde uzaklıkların yatay ve dikeydeki bloklar ile tarif edilmesidir. Örneğin 2 blok yukarı, 3 blok sağa gibi tarifler yapılabilir. Aslında öklid uzaklığının bir formudur. Öklid uzaklığında karesi alınan ve sonra karekök içerisine alınan ifade burada üssü alınmayacak şekilde mutlak değer içerisinde yazılır. Buna $L_1 norm$ denilir. Öklid uzaklığına ise $L_2 norm$ olarak bilinir. Manhattan uzaklığı aşağıdaki formül ile uzaklık hesabı yapılabilir.

$$d\big((x_1, x_2, \dots, x_n), (y_1, y_2, \dots, y_n)\big) = \sum_{i=1}^{n} |x_i - y_i|$$

Manhattan uzaklığı hesabının özellikleri şu şekilde verilebilir:

1. Negatif olmama koşulu: $d(i,j) \geq 0$
2. Aynı şehirde olma: $d(i,i) = 0$
3. Simetrik olma koşulu: $d(i,j) = d(j,i)$
4. Üçgen eşitsizliği: $d(i,j) \leq d(i,k) + d(k,j)$

Eğer bu 4 özellik sağlanırsa ölçüt metriktir yorumu yapılır. Örneğin x_1=(1,2) ve x_2=(3,5) olursa Öklid ve Manhattan uzaklıkları aşağıdaki gibi hesaplanır.

Öklid uzaklığı : $\sqrt{(2^2 + 3^2)} = 3{,}61$

Manhattan Uzaklığı : $2 + 3 = 5$

Manhattan mesafesini hesaplamak için Scipy kütüphanesi içerisindeki cityblock metodu kullanılabilir.

```
# Manhattan mesafesi Scipy içerisinde cityblock olarak tanımlanır.
hh_manhattan_mesafesi = distance.cityblock(hasan, huseyin)
hm_manhattan_mesafesi = distance.cityblock(hasan, mahmut)
mh_manhattan_mesafesi = distance.cityblock(mahmut, huseyin)
print(hh_manhattan_mesafesi)
print(hm_manhattan_mesafesi)
print(mh_manhattan_mesafesi)
```

```
502
8029
8531
```

Minkowski Uzaklığı: Öklid ve Manhattan uzaklığının genelleştirilmiş hâlidir. $L_n\ norm$ olarak da bilinir. N boyuttaki mesafeyi ölçmek için aşağıdaki formülden yararlanılır.

$$d\big((x_1, x_2, \dots, x_n),\ (y_1, y_2, \dots, y_n)\big) = \left(\sum_{i=1}^{n} (x_i - y_i)^r\right)^{1/r}$$

Minkowski uzaklığını hesaplamak için Scipy kütüphanesi içerisindeki **`minkowski`** metodu kullanılabilir. Bu metodun içerisine parametre olarak **`p=1`** yazılırsa Manhattan, **`p=2`** yazılırsa Öklid uzaklığı, **`p=3+`** yazılırsa Chebyshev uzaklığı elde edilebilir. Bu nedenle genelleştirilmiş bir uzaklık ölçüsüdür.

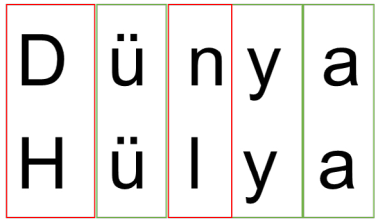
```
# Manhattan mesafesi Scipy içerisinde cityblock olarak tanımlanır.
hh_minkowski_mesafesi = distance.minkowski(hasan, huseyin, p=1)
hm_minkowski_mesafesi = distance.minkowski(hasan, mahmut, p=1)
mh_minkowski_mesafesi = distance.minkowski(mahmut, huseyin, p=1)
print(hh_minkowski_mesafesi)
print(hm_minkowski_mesafesi)
print(mh_minkowski_mesafesi)
```

```
502.0
8029.0
8531.0
```

Hamming Uzaklığı: Aynı boyuttaki listelerin veya metinsel ifadelerin uzaklığını ölçer. Örneğin "Dünya" ve "Hülya" kelimelerinin ne kadar benzer olduğunu harflerinin benzerliği ile ölçülmek istenebilir. Bu durumda hangi harf sırasıyla diğer kelimede varsa o kelimelerin daha benzer olduğu söylenir. Bu iki kelimede 2 harf benzemez iken 3 harf benzemektedir. Bu nedenle Hamming uzaklığı 2/5 olarak hesaplanır.

```
distance.hamming(list("Dünya"),list("Hülya"))
```

```
0.4
```



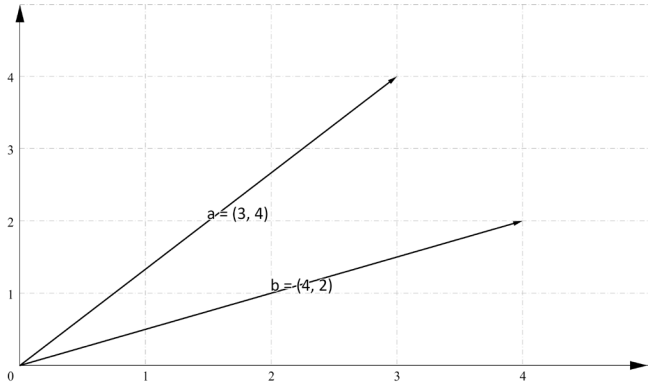
Şekil 3-10: Hamming uzaklığı için örnek metinler

Kosinüs Benzerliği

Kosinüs benzerliği öklid uzaklığının çok boyutlu durumlarda yaşadığı sorunları gidermek için geliştirilmiştir. Kosinüs benzerliğinde verilerin oluşturduğu vektörlerin arasındaki açının kosinüsüne bakılır. Vektörler arasındaki açının kosinüsü $(cos(\theta))$ benzerlik ölçüsü olarak kullanılır. İki vektör $\left(\vec{a} = \binom{3}{4}, \vec{b} = \binom{4}{2}\right)$ arasındaki açının kosinüsünü hesaplamak için öncelikle vektörlerin uzunlukları $\left(\|\vec{a}\|\|\vec{b}\|\right)$ aşağıdaki formülde gösterildiği gibi hesaplanır.

$$\|\vec{a}\| = \sqrt{a_1^2 + a_2^2} = \sqrt{3^2 + 4^2} = 5$$

$$\|\vec{b}\| = \sqrt{b_1^2 + b_2^2} = \sqrt{4^2 + 2^2} = 2\sqrt{5}$$



Şekil 3-11: İki vektör arasındaki Kosinüs benzerliği

Ardından iki vektör arasındaki açının kosinüsü benzerlik ölçümü için hesaplanır. Nokta çarpımda vektörün bileşenleri birbiri ile çarpılarak toplanır. İki vektörün arasındaki nokta çarpım sonucu aşağıdaki formül ile hesaplanır.

$$\vec{a} \cdot \vec{b} = a_1 b_1 + a_2 b_2$$
$$= (3 \times 4) + (4 \times 2) = 20$$

Nokta çarpım sonucu ve vektörlerin uzunlukları bulunduktan sonra iki vektör arasındaki açının kosinüsü aşağıdaki formül ile bulunabilir.

$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \times \|\vec{b}\| \times cos(\theta)$$

$$cos(\theta) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \times \|\vec{b}\|} = \frac{20}{5 \times 2\sqrt{5}} = \frac{2\sqrt{5}}{5} = 0{,}89$$

Birbirleriyle aynı güzergahta olan vektörlerin aralarındaki açı 0 olduğu için benzerliği de $cos(0) = 1$ olarak hesaplanır. Yani aynı güzergahta olan vektörlerin birbirinin aynısı olduğu sonucuna ulaşılır. Tamamen zıt yöndeki vektörlerin aralarındaki açı 180 olduğu için benzerliği de $cos(180) = -1$ olarak hesaplanır. Bu durumda, kosinüs benzerliğinin maksimum 1 minimum -1 değerini alabildiği söylenir.

Verilen a, b vektörleri için Python kodları ile kosinüs benzerliğini hesaplayabilmek için aşağıdaki kodlar kullanılabilir.

```
from scipy.spatial import distance
a_vektoru = (3,4)
b_vektoru = (4,2)
a_b_uzaklik = distance.cosine(a_vektoru,b_vektoru)
a_b_benzerlik = 1- a_b_uzaklik

print("a ve b vektörleri arasındaki benzerlik: ", a_b_benzerlik)
```

```
a ve b vektörleri arasındaki benzerlik:  0.8944271909999159
```

Kosinüs benzerliği özellikle iki belgenin karşılaştırılması için kullanılabilir. Bir belge binlerce özellikten oluşabilir. Her bir özellik bir kelimeyi belirtebilir. Örneğin metin madenciliği çalışmalarında birçok kelime ile aynı anda ilgilenmek gerekir. Bu durumda da mevcut ölçüler yetersiz kalabilir. İki dokümanın ortak kelimelerine yoğunlaşacak şekilde oluşturulan ölçüte **Kosinüs Benzerlik ölçütü** denilir. Örneğin metin madenciliğinde birçok kelime ile aynı anda ilgilenmek gereklidir. Bu nedenle klasik mesafe ölçütleri yetersiz kalabilir. İki

dokümanın ortak özelliklerine yoğunlaşan bir ölçüte ihtiyaç vardır. Bu ihtiyaç kosinüs benzerliği ölçütü ile giderilebilir.

Birinci Belge İçeriği = "**Sakarya Üniversitesi Endüstri Mühendisliği Bölümü**"

İkinci Belge İçeriği = "**Sakarya Uygulamalı Bilimler Üniversitesi Makine Mühendisliği Bölümü**"

Üçüncü Belge İçeriği = "**İstanbul Üniversitesi Endüstri Mühendisliği Bölümü**"

Verilen üç içerikle ilgili kosinüs benzerlik ölçüsü kullanılarak hangi iki içeriğin birbirine daha yakın olduğu hesaplanabilir. Bu belgelerin kıyaslanması için öncelikle kullanılan kelimeler tablo üzerinde gösterilir.

	İstanbul	Sakarya	Üniversite	Uygulamalı	Bilimler	Makine	Endüstri	Mühendislik	Bölüm
Belge1	0	1	1	0	0	0	1	1	1
Belge2	0	1	1	1	1	1	0	1	1
Belge3	1	0	1	0	0	0	1	1	1

Tablo 3-4: Belgelerin tablo hâli

Belge1 ve 2 arasındaki nokta çarpım sonucu= $0\times0+1\times1+1\times1+\cdots+1\times1=4$

Belge1 ve 3 arasındaki nokta çarpım sonucu= $0\times1+1\times0+0\times0+\cdots+1\times1=3$

Belge2 ve 3 arasındaki nokta çarpım sonucu= $0\times1+1\times0+1\times1+\cdots+1\times1=3$

Belge1 için uzunluk hesabı : $\sqrt{0^2+1^2+\cdots+1^2}=5$

Belge2 için uzunluk hesabı: $\sqrt{0^2+1^2+\cdots+1^2}=7$

Belge3 için uzunluk hesabı: $\sqrt{1^2+0^2+\cdots+1^2}=5$

Belge1 ve 2 arasındaki kosinüs benzerliği: $\frac{4}{5\times7}=0{,}11$

Belge1 ve 3 arasındaki kosinüs benzerliği: $\frac{3}{25}=0{,}12$

Belge2 ve 3 arasındaki kosinüs benzerliği: $\frac{3}{5\times7}=0{,}08$

Buradan çıkan hesaplamalara göre birinci belge ile üçüncü belgenin en fazla benzediği sonucuna ulaşılır. İkinci belge ile üçüncü belge ise birbirine en az benzeyen belgelerdir, denilir.

Jaccard Benzerliği: Metin verilerindeki benzerliğin ölçüsü olarak kullanılan bir diğer ölçüt de Jaccard benzerliğidir. Kimi kaynaklarda **Jaccard katsayısı** (Jaccard coefficient) olarak da geçer. Jaccard benzerliği genellikle asimetrik ikili özelliklerin arasındaki mesafenin hesabında kullanılır. Bu noktada Jaccard

benzerlik ölçüsünden önce simetrik ve asimetrik özelliklerin tanımını vermek gereklidir.[2] İkili veriler 0 veya 1 değerini alırlar. Eğer bir özellik veride mevcutsa 1 mevcut değilse 0 değerini alır. Örneğin değişken sigara kullanımı ise 1 sigara kullanma durumunu 0 ise sigara kullanmama durumunu gösterir.

Simetrik İkili Özellikler: Durumun varlığı ile yokluğu eşit derecede önemli ise bu tip ikili özelliklere simetrik ikili özellik denilir. Yani durumun 0 değeri ya da 1 değerini alması eşit önemdedir. Örneğin cinsiyet değişkeni için erkek için 0 kadın için 1 kodlamasını yapılırsa kadın ve erkek durumlarının simetrik bir özellik olduğu söylenebilir.

Asimetrik İkili Özellikler: Simetrik ikili özelliğin tersine durumların eşit önemde olmadığı durumlarda söz konusudur. Örneğin doğru/yanlış, pozitif/negatif, evet/hayır tipindeki veriler asimetrik ikili özelliklerdir. Çünkü 0 kodu durumun yokluğunu ifade ederken 1 kodu durumun varlığını nitelendirir. Aralarında asimetrik bir durum vardır.

Simetrik ve asimetrik ikili özellik durumuna göre Jaccard benzerlik ölçüsünün hesabı değişecektir. Jaccard katsayısı için aşağıdaki tablo referans olarak kullanılabilir. q, r, s, t değerleri eşleşen durum sayılarını göstermektedir. Örneğin birinci verinin 1 ikinci verinin 1 olduğu durumların sayısı q ile gösterilmiştir.

		İkinci veri	
		1	0
Birinci Veri	1	q	r
	0	s	t

Tablo 3-5: Jaccard benzerliği hesabında ikili verilerin referans tablosu

Simetrik ikili özellikler için: $\frac{r+s}{q+r+s+t}$

Asimetrik ikili özellikler için: $\frac{r+s}{q+r+s} \rightarrow s(i,j) = 1 - d(i,j)$

Jaccard benzerlik ölçütüne örnek olarak aşağıdaki tabloda bulunan verileri verebiliriz. Tabloda bir bölümdeki öğrencilerin dersleri geçip geçmedikleri gösterilmiştir. Bu durumda hangi öğrencilerin birbirine daha yakın oldukları Jaccard benzerlik ölçüsü ile hesaplanabilir.

İsim	**Cinsiyet**	**Ders 1**	**Ders 2**	**Ders 3**	**Ders 4**
Hasan	E	G	G	G	K
Hüseyin	E	G	K	G	K
Ayşe	K	K	G	K	G

Tablo 3-6: Öğrencilerin ders durumları

Öncelikle özelliklerin simetrik mi asimetrik mi olduğuna bakılır. Cinsiyet özelliği simetrik bir özelliktir. Diğer özellikler ise asimetrik özelliklerdir. Buna göre tablo 0-1 olarak yeniden kodlanırsa:

İsim	Cinsiyet	Ders 1	Ders 2	Ders 3	Ders 4
Hasan	0	1	1	1	0
Hüseyin	0	1	0	1	0
Ayşe	1	0	1	0	1

Tablo 3-7: Yeniden kodlanan tablo

Öğrencilerin ikili olarak aralarındaki Jaccard benzerlik hesabı için **referans tablolar** aşağıdaki gibi oluşturulabilir. Simetrik özellik olan cinsiyet bu hesaplamaya katılmamıştır. Simetrik özellikler için ayrıca bir hesaplama yapılabilir, ancak bu örnekte sadece bir özellik olduğu için benzerlik ölçüsü hesaplanması doğru olmayacaktır.

		Hüseyin	
		1	0
Hasan	1	q=2	r=1
	0	s=0	t=1

Tablo 3-8: Hasan Hüseyin için referans tablo

		Ayşe	
		1	0
Hasan	1	q=1	r=2
	0	s=1	t=0

Tablo 3-9: Hasan Ayşe için referans tablo

		Ayşe	
		1	0
Hüseyin	1	q=0	r=2
	0	s=2	t=0

Tablo 3-10: Hüseyin Ayşe için referans tablo

Jaccard uzaklık hesabı aşağıdaki gibi hesaplanır.

$$d(\text{Hasan, Hüseyin}) = \frac{r+s}{q+r+s} = \frac{1+0}{2+1+0} = 0{,}33 \rightarrow s(\text{Hasan, Hüseyin}) = 0{,}67$$

$$d(\text{Hasan, Ayşe}) = \frac{2+1}{1+2+1} = 0{,}75 \rightarrow s(\text{Hasan, Ayşe}) = 0{,}25$$

$$d(\text{Hüseyin, Ayşe}) = \frac{2+2}{0+2+2} = 1 \rightarrow s(\text{Hüseyin, Ayşe}) = 0$$

Çıkan sonuçlara göre benzerliği (s) en fazla olan ikili Hasan ve Hüseyin olmuştur. Hasan ve Ayşe'nin verilerinin hiç benzemediği ($d(\text{Hüseyin, Ayşe}) = 1$ olduğu için) görülmüştür.

Bu uygulamanın kodları Python ile aşağıdaki gibi oluşturulabilir.

```
hasan_gecme_durumu = (1, 1, 1, 0)
huseyin_gecme_durumu = (1, 0, 1, 0)
ayse_gecme_durumu = (0, 1, 0, 1)
hasan_huseyin_benzerlik = 1 - distance.jaccard(hasan_gecme_durumu,
                                               huseyin_gecme_durumu)
hasan_ayse_benzerlik = 1 - distance.jaccard(hasan_gecme_durumu,
                                            ayse_gecme_durumu)
huseyin_ayse_benzerlik = 1 - distance.jaccard(huseyin_gecme_durumu,
                                              ayse_gecme_durumu)

print(
    "Hasan ve Hüseyin in geçme durumları arasındaki benzerlik: {},\n Hasan
ve Ayşe nin geçme durumları arasındaki benzerlik: {},\n Hasan ve Ayşe nin
geçme durumları arasındaki benzerlik: {}"
    .format(hasan_huseyin_benzerlik, hasan_ayse_benzerlik,
            huseyin_ayse_benzerlik))
```

```
Hasan ve Hüseyin in geçme durumları arasındaki benzerlik:
0.6666666666666667,
 Hasan ve Ayşe nin geçme durumları arasındaki benzerlik: 0.25,
 Hasan ve Ayşe nin geçme durumları arasındaki benzerlik: 0.0
```

Nominal Verilerin Uzaklık Hesabı

Nominal bir veri birden fazla kategoride olabilir. Örneğin göz rengi yeşil, siyah, mavi gibi renkler olabilir. Nominal veriler için kategori sayısına m ve nesneyi tanımlayan toplam özellik sayısına p denilirse aşağıdaki formül iki verinin arasındaki uzaklığı verir.

$$d(i,j) = \frac{p-m}{p}$$

Örneğin aşağıdaki tabloda 4 kişiye ait 3 farklı özellik bulunmaktadır. Özellik-1 nominal bir özelliktir. Özellik-2 ise ordinal bir özelliktir. Bu nedenle Özellik-1 için uzaklık hesabı aşağıdaki gibi yapılabilir. Özellik-2 için ordinal özelliklerin uzaklık hesabı bir sonraki başlıkta incelenmiştir.

No	Özellik-1	Özellik-2	Özellik-3
1	A	Çok İyi	45
2	B	Kötü	22
3	C	İyi	64
4	A	Çok iyi	28

Tablo 3-11: Nominal verilerin uzaklık hesabı için örnek veriler

Uzaklık Matrisi: $\begin{bmatrix} 0 & - & - & - \\ d(2,1) & 0 & - & - \\ d(3,1) & d(3,2) & 0 & - \\ d(4,1) & d(4,2) & d(4,3) & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & - & - & - \\ 1 & 0 & - & - \\ 1 & 1 & 0 & - \\ 0 & 1 & 1 & 0 \end{bmatrix}$

Benzerlik Matrisi: $s(i,j) = 1 - d(i,j) = \frac{m}{p}$

Hesaplanan uzaklık matrisine göre 1 değerini alan uzaklıklar iki verinin aralarında bir benzerlik olmadığını gösterir. Değer olarak 0 değerini alanlar ise verilerin birbirine benzediğini gösterir. Buna göre $d(4,1) = 0$ olduğundan dördüncü ve birinci veriler benzerdir denilir.

Nominal veriler için literatürde yer alan birçok yöntem vardır. Bu yöntemler ile ilgili karşılaştırmalı çalışma Li ve Li tarafından 2010 yılında gerçekleştirilmiştir.[3] Bu yöntemler **çakışma metriği** (overlap metric), **minimum risk metriği** (minimum risk metric) ve **değer farkı metriği** (value difference metric) olarak verilmiştir. Ayrıca kategorik ve nominal verilerin uzaklığı konusunda yapılan birkaç çalışma da şu şekilde verilebilir.[4,9]

Çakışma Metriği: Çakışma metriğinin hesaplanması için kullanılacak denklem aşağıda verilmiştir.[3] Denklemde geçen n toplam özellik sayısını, $a_i(x)$ ve $a_i(y)$ ise i özelliğinin değerini göstermektedir. Bu nedenle eğer $a_i(x)$=$a_i(y)$ olursa $\delta(a_i(x), a_i(y)) = 0$ olur. Tersi durumda ise 1 değerini alır.

$$d(x,y) = \sum_{i=1}^{n} \delta\big(a_i(x), a_i(y)\big)$$

Değer Farkı Metriği: Standfill ve Waltz tarafından geliştirilmiştir.[10] Bu yöntemin kullandığı formül aşağıda verilmiştir. Formülde geçen C nominal verilerdeki sınıf sayısını, $P\big(c|a_i(x)\big)$ koşullu olasılığı göstermektedir.

$$\sum_{i=1}^{n} \sum_{c=1}^{C} \left| P\big(c|a_i(x)\big) - P\big(c|a_i(y)\big) \right|$$

Ordinal Veriler için Uzaklık Hesabı

Ordinal veriler aralarında sıralama ve büyüklük-küçüklük ilişkisi söz konusudur. Bu nedenle nominal verilerden farklı değerlendirilirler. Örneğin kalitenin kötü, orta, iyi diye kategorize edilmesi ordinal verilere örnektir. Örneğin küçük orta, büyük diye kategorize etme işlemi yapıldığında ordinal bir veri tipi tanımlanmış olur. Nominal verilerdekine benzer şekilde m ordinal verilerin alabileceği değer sayısı olursa, aşağıdaki adımları uygulayarak ordinal veriler için uzaklık hesabı yapılabilir.

1. Özellikler arasında $1,2,\dots,m$ şeklinde sıralama yap.
2. Veriler arasında normalizasyon yap.
3. Nümerik veriler için yapılan uzaklık hesaplamalarını gerçekleştir.

Nominal veriler için verilen tabloda Özellik-2'nin ordinal veri tipinde veri olduğunu görebiliriz. Ordinal veri uzaklığını hesaplamak istersek, $m = 3 \rightarrow$ (Çok iyi: 3, İyi: 2,Kötü:1) dönüşümü yapılır. Yapılan yeniden kodlama ile Özellik-2 değerleri sırasıyla 3, 1, 2, 3 değerlerini alır. Normalizasyon yapılınca Çok İyi: 1, İyi: 0,5, Kötü: 0 olur. Sonuç ise aşağıda gösterildiği gibi çıkar.

$$d(i,j) = \begin{bmatrix} 0 & - & - & - \\ 1 & 0 & - & - \\ 0{,}5 & 0{,}5 & 0 & - \\ 0 & 1 & 0{,}5 & 0 \end{bmatrix}$$

$d(1,2)\ ve\ d(4,2) = 1$ olduğu için en uzak veriler 1-2 ve 4-2 sonucu çıkarılır. Burada benzerlik ölçüsü olarak $s(i,j) = 1 - d(i,j)$ formülü kullanılır.

Veri Setleri

Veri seti daha önce belirtildiği gibi özelliklerden ve nesnelerden oluşan tablo şeklinde düşünülebilir. Veri madenciliğinde kullanılan veri setleri çok farklı kaynaklardan alınabilir ve birçok açıdan kategorilere ayrılabilir. Veri setlerine örnek olarak şunlar verilebilir.[2]

- **Kayıt veri seti**: Bir gözleme ait verilerin tutulduğu verilerden oluşan veri setidir. Burada her yeni girdi yeni bir olay olduğunda sisteme eklenir.
- **Veri Matrisi**: Eğer veri objeleri sayısal verilerden oluşursa veri matrisi oluşur denebilir. $n \times m$ boyutuna sahiptir.
- **Doküman veri seti**: Her belge bazı terimleri kaç defa içerdiğine göre bir tabloda tutulabilir. Bu tip veri setine belge ya da belge verisi denir. Belgelerin birbirine ne kadar benzediği ile alakalı çalışmalar için kullanılan veri setleridir.
- **İşlemler veri seti**: Özel bir kayıt verisidir. Her kayıt birkaç elemandan oluşan setleri tutar. Örneğin bakkal dükkanının satılan ürünleri sıraladığı liste bir işlemler veri setidir.
- **Grafik veri seti**: Molekül yapısı ya da internet sayfalarının gösterilmesi gibi birbirleri ile bağlantılı verilerin gösterildiği veri setleridir.

Veriyi Analize Hazır Hâle Getirme

Veriler ile çalışma detaylı ve zorlu bir süreçtir. Detaylı süreçlerin başlangıcı verinin analize hazır hâle getirilmesi aşamasında gerçekleşir. Veriler elde edildikten sonra genellikle gerçek veriler üzerinde istenmeyen, eksik veya hatalı girdiler bulunur. Kimi zaman olmaması gereken veriler, kimi zaman yanlış girilmiş veriler kimi zaman da sensör ya da okuyucu hatasından dolayı alınamayan veriler ile ilgilenmek veri madenciliğinin ilgilendiği bir alandır. Kirli olarak gelen verilerin temizlenmesi ve birtakım işlemlerden geçirilmesi gerekir. Bu işlemlere veri önişleme aşamaları denilir. Bu başlıkta veri ön işleme aşamalarından bahsedilecektir.

Veri Temizleme

Veri temizleme aşaması veri madenciliğinde önemli bir yer tutar. Çünkü veri toplamadaki herhangi bir hata ya da eksiklik analizi de doğrudan etkileyecektir. Bu aşamada **verinin kalitesi** kavramından bahsetmek gerekir. Kalitenin tanımı kullanıldığı alana göre çok değişkenlik gösterir. Genel olarak kalite teknik özelliklere uygunluk olarak tanımlanabilir. Veride kalite ölçütleri şu şekilde sıralanabilir:

doğruluk

tutarlılık

güvenilirlik

öngörülebilirlik

Bazen yanlış girilmiş veriler karşımıza çıkabilir. Örneğin yaş değerinin 330 olduğu veriler olabilir ya da 30 Şubat tarihinin girildiği veriler olabilir. Bu veriler veri kalitesi ile ilgilidir ve yapılacak olan analizlere doğrudan etkisi vardır. Bu verilerin yanlış girilmesinin birçok nedeni olabilir. Verileri giren kişi yanlış girmiş olabilir ya da hatalı yazım olmuş olabilir. Ya da veriler kaymış olabilir. Öncelikle yanlış girilen verinin tespit edilmesi, ardından yanlış girilen verinin neden yanlış girildiğinin incelenmesi gerekir. Mesela yaş değeri 330 olarak girildiyse muhtemelen 30 girilecek veri 2 kez 3 değerine basıldığı için 330 girilmiş olabilir. Bu şekilde diğer yanlış veriler ile alakalı da değerlendirmeler yapmak gerekir.

Öngörülebilirlik ise verinin ne kadar kolay anlaşılabilir olduğu ile alakalıdır. Örneğin bir veri tabanında birçok hatalı girdinin olduğunu düşünün bu durumda diğer verilerin de güvenirliği zedelenecektir.

Güvenilirlik de veri kalitesini etkileyen diğer bir faktördür. Kullanıcıların verilen veriye ne kadar güvendiğini gösterir. Örneğin yaşı 30 girilmiş birinin doğum tarihinin 1980 girilmesi.

Bu özelliklere sahip veriler kaliteli olarak nitelendirilir. Ancak gerçek hayattaki veriler her zaman doğru, tutarlı, güvenilir veya öngörülebilir değildir. Kalitesiz veriler söz konusu olduğunda ortaya aşağıdaki sorunlar çıkabilir:

Gürültülü veri: Uç ya da aykırı verilerin veri setine girmesi ile gürültülü veriler ortaya çıkar. Örnek olarak ses kalitesi düşük olan bir telefondan alınan ses frekansları verileri verilebilir.

Eksik veri: Tüm değişkenlere ait verilerden herhangi birisinde eksik veri yer alabilir. Bu durum kimi zaman bir sensör arızasından kimi zaman da verinin girilmemesinden kaynaklanabilir. Ya da konuşma verileri düşünülürse, konuşmadan yazıya dönüştürülürken bazı kelimeler alınamayabilir. Bu durumda da eksik veriler söz konusu olur. Eksik verilerin bir yöntem ile doldurulması gerekebilir. Ya da yöntem olarak eksik verilerin göz ardı edilmesi söz konusu olabilir. Bu durumda da veri kaybına neden olunur. Bu nedenle eksik verilerin doğru bir yöntem ile doldurulması daha doğru bir çözüm olacaktır. Eksik verilerin doldurulması ile ilgili literatürde birçok yöntem geliştirilmiştir.

Eksik verileri elle doldur: Eksik verileri bir uzmanın doldurması işlemidir. Genellikle küçük sayıdaki verinin eksik olması durumunda kullanılır. Örneğin ankette soruların bir kısmını cevaplamayan kişilerin verilerinin doldurulması bu yönteme örnektir.

Global bir değişken ile değiştir: Veri setindeki değerler ile karışmayacak şekilde eksik verilerin özel bir değer ile doldurulması işlemidir. Örneğin "Bilinmeyen", ya da nümerik bir veride "-1" değeri vermek.

Merkezi bir ölçü kullanarak değiştir: Bu durumda eksik olan veriler ortalama, mod ya da medyan ile değiştirilebilir.

Aynı sınıfa ait merkezi bir ölçü ile değiştir: Bir sınıftaki ortalama ile değiştir.

Olası bir değer ile değiştir: Bir analiz sonucunda belirlenen bir değerle değiştirme yöntemidir. Örneğin regresyon analizi, en yakın komşuluk sınıflandırma analizi ya da karar ağaçları ile bulunan değerler ile eksik verilerin değiştirilmesi.

Öncesi veya sonrası ile değiştir: Bu yöntemde eksik veriler önceki veya sonraki takip eden değer ile doldurulur. Genellikle zaman serisi veri setlerinde kullanılmaktadır. Örneğin bir günlük veri içerisinde bazı saatlik veriler eksik kaldıysa bir saat önceki değer ile veri doldurulabilir.

Python ile Eksik Veri İşlemleri: Python programlama dilinde boş veya hatalı görülen değerler için `NaN` veya `None` ifadesi kullanılır. Kimi programlama dillerinde Null olarak kullanılan ifade Python programlama dilinde bulunmamaktadır. `NaN` ifadesi NumPy kütüphanesi içerisinde yer alan bir ifadedir ve "**IEEE 754 floating point representation of Not a Number (NaN).**" olarak tanımlanır.[11] `None` ile `NaN` değişkenleri arasında küçük farklar olsa da Pandas kütüphanesinde iki veri tipinde de eksik veri operasyonları yapılabilir. `NaN` veri tipinde nümerik hesaplamalar yapılabilir ancak `None` veri tipinde aritmetik işlemlerde hata ile karşılaşılır. Bu durumlar ile ilgili uygulamalar aşağıda verilmiştir.

```
import numpy as np
import pandas as pd
# None veri tipi
type(None)
```

```
NoneType
```

```
# np.NaN veri tipi
type(np.nan)
```

```
# nan ile aritmetik işlemler yapılabilir.
print(np.nan+1)
```

```
nan
```

```
# None ile aritmetik işlemler yapılırsa hata ile karşılaşılır.
print(None+1)
```

```
TypeError Traceback (most recent call last) <ipython-input-6-7e4f7026c7a7> in <module>()
1 # None ile aritmetik işlemler yapılırsa hata ile karşılaşılır.
----> 2 print(None+1)
TypeError: unsupported operand type(s) for +: 'NoneType' and 'int'
```

```
# pandas içerisinde bir seri oluşturalım.
# Serileri daha sonra daha detaylı inceleyeceğiz şimdilik bir veri
# dizisi olarak bilebilirsiniz
eksik_veri_serisi = pd.Series(data=np.array([None, np.nan, 1, 2,
"Merhaba"]))
```

isnull(), isna() fonksiyonları eksik verileri bulmak için kullanılır. Eksik verilerde **True** değeri dönderilir. Eksik olmayan verilerde **False** değeri dönderilir. **notnull()** fonksiyonu ise eksik verinin olduğu yerleri **False** olarak gönderir. Bu fonksiyonlar ile ilgili uygulamalar aşağıda gösterilmiştir:

```
eksik_veri_serisi.isna()
```

```
0    True
1    True
2   False
3   False
4   False
dtype: bool
```

```
# Kaç tane eksik veri var.
eksik_veri_serisi.isnull().sum()
```

```
2
```

```
# Kaç tane eksik olmayan veri var.
eksik_veri_serisi.notnull().sum()
```

```
3
```

```
# Eksik verileri gösterme.
eksik_veri_serisi[eksik_veri_serisi.isnull()]
```

```
0    None
1     NaN
dtype: object
```

```
# Eksik olmayan verileri getir.
eksik_veri_serisi[eksik_veri_serisi.notnull()]
```

```
2         1
3         2
4    Merhaba
dtype: object
```

Eksik veriler tespit edildikten ve gösterildikten sonra eksik verilerle ilgilenme aşamasına geçilebilir. Aşağıda bu konuda örnekler gösterilmiştir. Kullanılan fonksiyonları; **`dropna(), fillna()`** fonksiyonlarıdır. Bu fonksiyonlara ilişkin bilgiler **`help()`** fonksiyonu ile alınabilir. Aşağıda bu duruma örnek gösterilmiştir. Bu şekilde diğer fonksiyonların kullanımı ile ilgili detaylı bilgi alınabilir.

```
help(pd.DataFrame.fillna)
```

```
Help on function fillna in module pandas.core.frame:
fillna(self, value=None, method=None, axis=None, inplace=False, limit=None, downcast=None) ->
'Optional[DataFrame]'
    Fill NA/NaN values using the specified method.

    Parameters
    ----------
    value : scalar, dict, Series, or DataFrame
        Value to use to fill holes (e.g. 0), alternately a
        dict/Series/DataFrame of values specifying which value to use for
        each index (for a Series) or column (for a DataFrame).  Values not
        in the dict/Series/DataFrame will not be filled. This value cannot
        be a list.
    method : {'backfill', 'bfill', 'pad', 'ffill', None}, default None
        Method to use for filling holes in reindexed Series
        pad / ffill: propagate last valid observation forward to next valid
        backfill / bfill: use next valid observation to fill gap.
    axis : {0 or 'index', 1 or 'columns'}
        Axis along which to fill missing values.
    inplace : bool, default False
        If True, fill in-place. Note: this will modify any
        other views on this object (e.g., a no-copy slice for a column in a
```

```
    DataFrame).
limit : int, default None
    If method is specified, this is the maximum number of consecutive
    NaN values to forward/backward fill. In other words, if there is
    a gap with more than this number of consecutive NaNs, it will only
    be partially filled. If method is not specified, this is the
    maximum number of entries along the entire axis where NaNs will be
    filled. Must be greater than 0 if not None.
downcast : dict, default is None
    A dict of item->dtype of what to downcast if possible,
    or the string 'infer' which will try to downcast to an appropriate
    equal type (e.g. float64 to int64 if possible).

Returns
-------
DataFrame or None
    Object with missing values filled or None if ``inplace=True``.
```

```
# dropna(): Eksik verileri düşürür.
eksik_veri_serisi.dropna()
```

```
2         1
3         2
4    Merhaba
dtype: object
```

```
# Eksik verilerin doldurulması
# fillna() herhangi bir değerle doldurabiliriz.
eksik_veri_serisi_eksi_birli = eksik_veri_serisi.fillna(value=-1)
eksik_veri_serisi
```

```
0      None
1       NaN
2         1
3         2
4    Merhaba
dtype: object
```

```
eksik_veri_serisi_eksi_birli
```

```
0        -1
1        -1
2         1
3         2
4    Merhaba
dtype: object
```

```
eksik_veri_serisi_dusurulmus = eksik_veri_serisi.dropna()
eksik_veri_serisi_dusurulmus
```

```
2      1
3      2
4   Merhaba
dtype: object
```

```
eksik_veri_serisi
```

```
0       None
1        NaN
2          1
3          2
4    Merhaba
dtype: object
```

Uygulamalardan da görülebileceği gibi fonksiyonlar kalıcı etki oluşturmamaktadır. Kalıcı etki oluşturabilmesi için yani veri setinin değişmesi için **inplace=True** parametresinin yazılması gereklidir. Ya da bunun yerine daha güvenilir bir yöntem olan her değişiklik için farklı bir değişken atamasının yapılması önerilmektedir. Çünkü ileride orijinal veri setine ulaşılmak istenirse orijinal veri seti hızlı bir şekilde getirilebilir. Daha gelişmiş eksik veri işlemleri için gerçek bir veri seti olan **titanic** veri seti ile gerçekleştirilen çalışma aşağıdadır. Titanic veri seti bu bağlantıdan direkt çağrılabileceği gibi bilgisayara indirildikten sonra klasörden de çağrılabilir. Titanic veri setindeki eksik veri işlemleri için aşağıdaki kodlar geliştirilmiştir:

```
veri_seti = pd.read_csv(
    "https://gist.githubusercontent.com/michhar/2dfd2de0d4f8727f873422c5d9
59fff5/raw/fa71405126017e6a37bea592440b4bee94bf7b9e/titanic.csv"
)
veri_seti.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Veri setindeki ilk 5 satır gösterilmiş oldu. Bu veri seti literatürde bilinen ve sıklıkla çalışılan bir veri setidir. Veri seti ile ilgili detaylı bilgi veri setinin sayfasından alınabilir. O nedenle verilerin özellikleri ile ilgilenilmeden eksik veri işlemlerinin gösterilmesi amacıyla aşağıdaki kodlar gösterilecektir.

```
# Boş verilerin sayısına bakalım.
veri_seti.isna().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

```
# Yaş verilerinde anormal bir veri var mı diye bakalım.
veri_seti.Age.unique()
```

```
array([22. , 38. , 26. , 35. , nan, 54. , 2. , 27. , 14. , 4. , 58. , 20. , 39. , 55. , 31. , 34. , 15. , 28. , 8. , 19. , 40. , 66. , 42. , 21. , 18. , 3. , 7. , 49. , 29. , 65. , 28.5 , 5. , 11. , 45. , 17. , 32. , 16. , 25. , 0.83, 30. , 33. , 23. , 24. , 46. , 59. , 71. , 37. , 47. , 14.5 , 70.5 , 32.5 , 12. , 9. , 36.5 , 51. , 55.5 , 40.5 , 44. , 1. , 61. , 56. , 50. , 36. , 45.5 , 20.5 , 62. , 41. , 52. , 63. , 23.5 , 0.92, 43. , 60. , 10. , 64. , 13. , 48. , 0.75, 53. , 57. , 80. , 70. , 24.5 , 6. , 0.67, 30.5 , 0.42, 34.5 , 74. ])
```

```
# Tekrar eden değerlerin sayısını bulmak için
veri_seti.Cabin.value_counts()
```

```
G6             4
B96 B98        4
C23 C25 C27    4
F2             3
C22 C26        3
              ..
A16            1
C148           1
E38            1
D47            1
C86            1
Name: Cabin, Length: 147, dtype: int64
```

```
# Kabin özelliğinde anormal bir veri var mı diye bakalım
veri_seti.Cabin.unique()
```

```
array([nan, 'C85', 'C123', 'E46', 'G6', 'C103', 'D56', 'A6',
       'C23 C25 C27', 'B78', 'D33', 'B30', 'C52', 'B28', 'C83', 'F33',
       'F G73', 'E31', 'A5', 'D10 D12', 'D26', 'C110', 'B58 B60', 'E101', 'F E69', 'D47', 'B86', 'F2', 'C2', 'E33',
'B19', 'A7', 'C49', 'F4', 'A32', 'B4', 'B80', 'A31', 'D36', 'D15', 'C93', 'C78', 'D35',
       'C87', 'B77', 'E67', 'B94', 'C125', 'C99', 'C118', 'D7', 'A19',
       'B49', 'D', 'C22 C26', 'C106', 'C65', 'E36', 'C54',
       'B57 B59 B63 B66', 'C7', 'E34', 'C32', 'B18', 'C124', 'C91', 'E40', 'T', 'C128', 'D37', 'B35', 'E50',
'C82', 'B96 B98', 'E10', 'E44', 'A34', 'C104', 'C111', 'C92', 'E38', 'D21', 'E12', 'E63', 'A14',
       'B37', 'C30', 'D20', 'B79', 'E25', 'D46', 'B73', 'C95', 'B38',
       'B39', 'B22', 'C86', 'C70', 'A16', 'C101', 'C68', 'A10', 'E68',
       'B41', 'A20', 'D19', 'D50', 'D9', 'A23', 'B50', 'A26', 'D48',
       'E58', 'C126', 'B71', 'B51 B53 B55', 'D49', 'B5', 'B20', 'F G63', 'C62 C64', 'E24', 'C90', 'C45', 'E8',
'B101', 'D45', 'C46', 'D30', 'E121', 'D11', 'E77', 'F38', 'B3', 'D6', 'B82 B84', 'D17', 'A36', 'B102', 'B69',
'E49', 'C47', 'D28', 'E17', 'A24', 'C50', 'B42',
       'C148'], dtype=object)
```

```
# Eksik verileri fillna() fonksiyonu ile dolduralım.
veri_seti.fillna(value=-1).head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	-1	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/ O2. 3101282	7.9250	-1	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	-1	S

`Fillna()` komutu ile bir takım parametreler kullanılarak değişik **doldurma** yöntemleri kullanılabilir. Örnekler aşağıda gösterilmiştir.

```
# Belirli sütunları doldurmak istersek
veri_seti.fillna({'Cabin':'G6', 'Age':-1}).tail()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	G6	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	-1.0	1	2	W./C. 6607	23.45	G6	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	G6	Q

```
# axis = 0 ise satırlarda, 1 ise sütunlarda işlem
veri_seti.fillna(method='ffill', axis="columns").head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	7.250	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	7.925	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	8.050	S

`Dropna()` fonksiyonu ile eksik verilerin veri setinden **kaldırılması** işlemleri yapılabilir. Bu durumlara örnek aşağıda gösterilmiştir.

```
# Cabin ve Age özelliklerindeki boş verilerin kaldırılması
veri_seti.dropna(subset=['Cabin', 'Age']).isnull().sum()
```

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       2
dtype: int64
```

```
# how= any demek satırda herhangi bir özelliğin değeri boşsa o satır komple silinir.
veri_seti.dropna(how="any").isnull().sum()
```

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
```

```
SibSp        0
Parch        0
Ticket       0
Fare         0
Cabin        0
Embarked     0
dtype: int64
```

```
# how=all demek eğer tüm verileri eksik olan varsa onları sil demek.
veri_seti.dropna(how="all").isnull().sum()
```

```
PassengerId     0
Survived        0
Pclass          0
Name            0
Sex             0
Age           177
SibSp           0
Parch           0
Ticket          0
Fare            0
Cabin         687
Embarked        2
dtype: int64
```

- **Aykırı veri**: Veri setinin karakteristiğini yansıtmayan ve büyük ihtimalle bir hata sonucu veri setine dahil edilen verilerdir. Aykırı verilerin elenmesi veya analizi ile ilgili örnek bir çalışma **titanic** veri seti üzerinde aşağıdaki gibi gerçekleştirilmiştir.

```
# Veri setindeki yaş özelliği ile ilgilenelim.
# Öncelikle yaş değişkeninin eksik verilerini dolduralım.
veri_seti.Age.head()
```

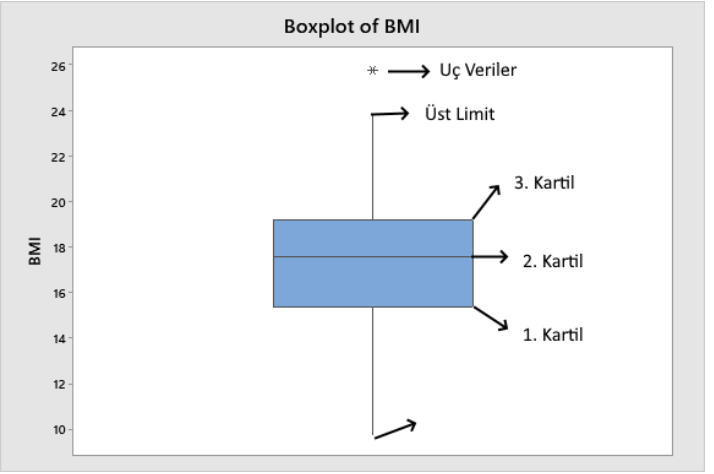
```
0    22.0
1    38.0
2    26.0
3    35.0
4    35.0
Name: Age, dtype: float64
```

```
yas_dolu_veri_seti= veri_seti.Age.fillna(value=veri_seti.Age.mean())
```

```
# Bir verilerimizi inceleyelim.
yas_dolu_veri_seti.describe()
```

```
count   891.000000
mean     29.699118
std      13.002015
min       0.420000
25%      22.000000
50%      29.699118
75%      35.000000
max      80.000000
Name: Age, dtype: float64
```

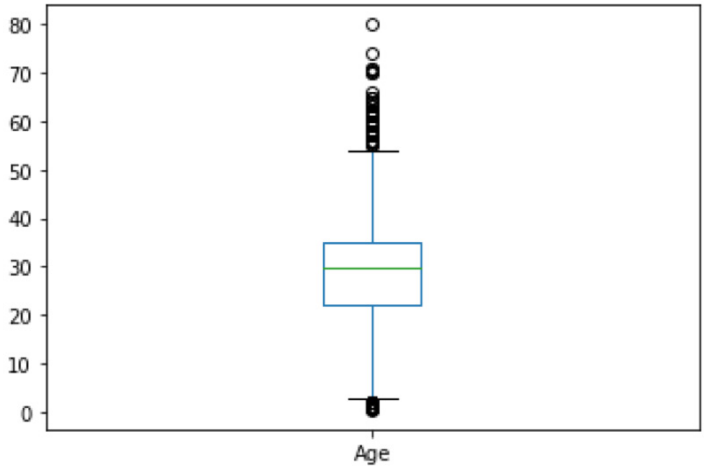
Yaş verileri incelendiğinde %75'lik kısım ile maksimum değer arasında ciddi bir fark olduğu görülebilir. Ya da %25'lik değer ile minimum değer arasında yine ciddi fark görülebilir. Bu durumda aykırı verilerin veri setinde olması söz konusu olabilir. Aykırı verilerin görüntülemek için kutu grafiklerinden yararlanılabilir. Genel anlamda kutu grafiklerinin okunması için aşağıdaki görsel kullanılabilir.



Şekil 3-12: Kutu grafiğinin okunması

```
# box plot ile aykırı veri olup olmadığını görebiliriz
yas_dolu_veri_seti.plot.box()
```

```
<AxesSubplot:>
```



Kutu grafiğinde görüldüğü gibi veri setinde aykırı veriler bulunmaktadır. Bu verilerin veri setinden kaldırılması veya başka bir yöntem ile ayıklanması gerekmektedir. Aykırı verilerin tespiti için iki farklı yöntemden bahsedilebilir. Bunlardan birincisi çeyrekler arası mesafedir. Bu mesafede öncelikle Q_1 ve Q_3 değerleri belirlenir. Daha sonra çeyrekler arası mesafe ile aykırı verilerin aralığı gösterilir.

```
Q1 = yas_dolu_veri_seti.quantile(0.25)
Q3 = yas_dolu_veri_seti.quantile(0.75)
IQR = Q3-Q1
print("Q1: {}, Q3: {}, IQR: {}".format(Q1, Q3, IQR))
```

```
Q1: 22.0, Q3: 35.0, IQR: 13.0
```

```
alt_sinir = Q1-(IQR*1.5)
ust_sinir = Q3+(IQR*1.5)
print(alt_sinir)
print(ust_sinir)
```

```
2.5
54.5
```

```
# üst sinirdan küçük olan değerler
yas_dolu_veri_seti[yas_dolu_veri_seti>ust_sinir]
```

```
11    58.0
15    55.0
```

```
33    66.0
54    65.0
94    59.0
96    71.0
116   70.5
152   55.5
170   61.0
174   56.0
…
```

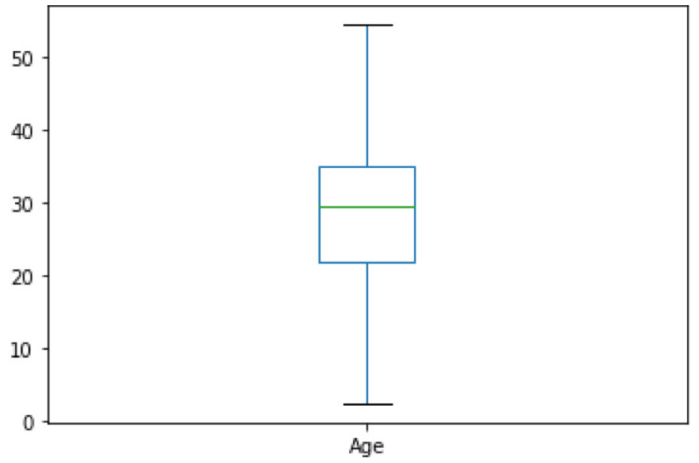
```
# alt sınırın altındaki alt sınıra, üst sınırın üzerindekiler üst sınıra eşitlendi.
yas_dolu_veri_seti[yas_dolu_veri_seti<alt_sinir] = alt_sinir
yas_dolu_veri_seti[yas_dolu_veri_seti>ust_sinir] = ust_sinir
yas_dolu_veri_seti.describe()
```

```
count   891.000000
mean     29.376817
std      12.062035
min       2.500000
25%      22.000000
50%      29.699118
75%      35.000000
max      54.500000
Name: Age, dtype: float64
```

Bu işlemler yapıldıktan sonra tekrar kutu grafiği çizildiğinde aykırı verilerin olmadığı görülür.

```
yas_dolu_veri_seti.plot.box()
```

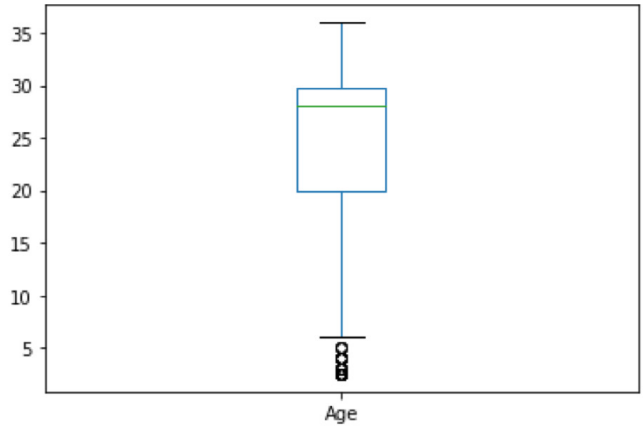
```
<AxesSubplot:>
```



Burada aykırı verilerin tespiti için çeyrekler arası mesafenin kullanıldığı bir örnek gösterilmiş oldu. İkinci yöntem olarak **z-skoru** yöntemi söylenebilir. Z-skoru yönteminden önce verilerin normal dağılıma uygun dağıldığı varsayımı yapılır. Eğer veriler normal dağılıma uymuyorsa z-skoru yöntemi kullanılmamalıdır.

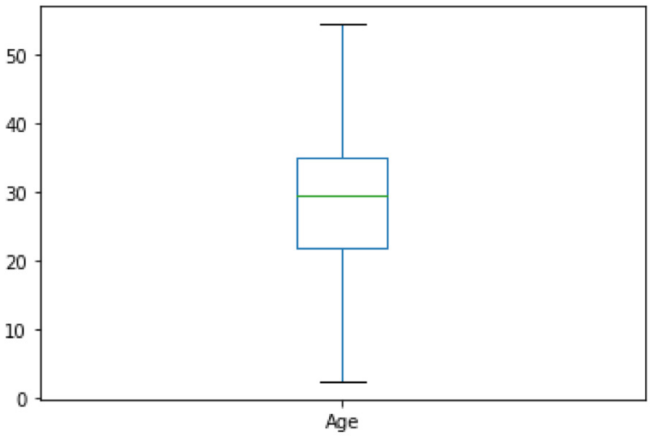
```
# 3 standart sapma ve üzerindekiler kaldırılırsa
yas_dolu_veri_seti[yas_dolu_veri_seti<yas_dolu_veri_seti.std()*3].plot.
box()
```

```
<AxesSubplot:>
```




```
# z skoruna göre kırpılmış veri seti
yas_dolu_veri_seti[(yas_dolu_veri_seti>yas_dolu_veri_seti.std()*-3)].plot.
box()
```

```
<AxesSubplot:>
```



- **Tekrarlı veri**: Veri setinde aynı olarak geçen birden fazla verinin olması durumunda ortaya çıkar. Örneğin bir adet girilmesi gereken e-posta adresi bilgisinin birden fazla girilmesi durumu. Tekrarlı verilerin temizlenerek veri setinin kalitesinin artırılması gerekmektedir.
- **Yanlış veri**: Bazı durumlarda mümkün olmayan değerlerin veri olarak girildiği durumlar olabilir. Bu durumda yanlış veriler ile ilgilenilmesi gerekir. Örneğin hava kalitesi parametrelerinin kaydedildiği bir veri setinde karbondioksit miktarının negatif değer girilmesi durumunda yanlış veri söz konusu olur. Çünkü karbondioksit miktarının negatif değer alması mümkün değildir. En az 0 değeri alabilir.

Veri Birleştirme

Verilerin kalitesini artırmak için bir dizi metotlar uygulanabilir. Bunlardan birisi de **verileri birleştirmek**tir. Verileri birleştirmekteki amaç daha stabil bir veriye sahip olmaktır. Örnek olarak gün ve saat olarak verilen özelliklerin tek özellik olan tarih özelliğinde birleştirilmesi işlemi verilebilir. Ya da şehir olarak verilen konum bilgisinin bölge olarak değiştirilmesi işlemi de örnek olarak söylenebilir.

Özellik İndirgeme

Bir veri setinde ne kadar fazla özellik varsa analizin daha sağlıklı gerçekleştirilmesi o kadar zorlaşır. Çünkü veri boyutu azalınca modellerin karmaşıklığı daha azalır. Aynı zamanda bilgisayarın işlemci gücü ve işlem zamanları da özelliklerin **miktarına** bağlı olarak değişecektir. Bu nedenlerle olabildiğince az özellik ile çalışarak doğruluğu daha yüksek modeller kurmak veri madenciliğinde önemli bir aşamadır.

Özellik indirgeme çalışmalarına bir örnek olarak **birinci etmen analizi** (*principle component analysis*) verilebilir. Özellik indirgeme çalışmalarının amacı; gereksiz olabilecek özellikleri analizden çıkarmaktır. Bazı özellikler birbirine çok benzeyebilir, bu durumda birbirine benzeyen özelliklerden birinin elenmesi gerekir. Aynı şekilde analizde alakasız olabilecek özelliklerin de veri analizinden çıkarılması gerekir. Örneğin öğrencilerin bilgilerinin tutulduğu veri setinde, öğrenci numarasının analizde kullanılması mümkün olmayacağı için setten çıkarılması gibi.

Örneklem alma

Özellik indirgeme ile gözlem sayısında gereksiz olabilecek derecede büyük veri seti ile çalışmak da **analizi kötü yönde etkiler**. Gözlem sayısını indirme çalışmalarına genel olarak örneklem alma çalışmaları denilebilir. Örneklem

alma çalışmaları bazen veride var olan bilgileri kaybetmemize de neden olabilir. İstatistik biliminde örneklem üzerinden yapılan çıkarımların ana kütleye yayılması önemli bir problem alanı doğurur. Veri madenciliğinde de kimi zaman verilerin tamamını analizde kullanmak çok zor olabilir. Bu durumda veri içerisinde ana kütleyi temsil edecek örneklem seçilmesi gereklidir.

Kesikli ya da İkili Hâle Getirme

Sürekli veriler ile çalışmak veri madenciliğinde zorluk doğurur. Bu nedenle sürekli veriler **kesikli** hâle getirilebilir. Böylece sınırsız sayıdaki seçenek sonlu sayıya indirilebilir. Örnek olarak maaş değişkeninin sürekli hâlde verildiği varsayılsın. Veri setine göre maaş değişkeni içerisinde 1000-2000 TL arasına 1, 2000-5000 TL arasına 2, 5000-10000 TL arasına 3 denirse maaş değişkeni kesikli hale getirilmiş olur.

Kategorik verilerin yeniden kodlaması (encoding) için Pandas içerisindeki **`get_dummies()`** fonksiyonu kullanılabilir. Ya da Scikit-Learn içerisinde de yine Label Encoder, One Hot Encoder gibi özellikler bulunmaktadır. Aşağıda **`get_dummies()`** fonksiyonu ile ilgili bir örnek gösterilmiştir. Örnekte görüldüğü gibi cinsiyet değişkeni erkek ve kadın değerleri alırken **`get_dummies()`** fonksiyonu ile erkek sütunu içerisine 0-1 kodlaması gerçekleştirilmiştir. Böylece kişinin erkek olması durumunda 1 kadın olması durumunda 0 yazması sağlanmıştır.

```
cinsiyet = pd.get_dummies(veri_seti.Sex, drop_first=True)
cinsiyet.head()
```

```
   male
0     1
1     0
2     0
3     0
4     1
```

```
veri_seti.Sex.head()
```

```
0     male
1   female
2   female
3   female
4     male
Name: Sex, dtype: object
```

Kategorik verilerin nümerik veriler olarak kullanılabilmesi için ikili veri tipine dönüştürülmesi gerekir. İkili veri tipinde özellik 0 ya da 1 değerini alabilir. Bu

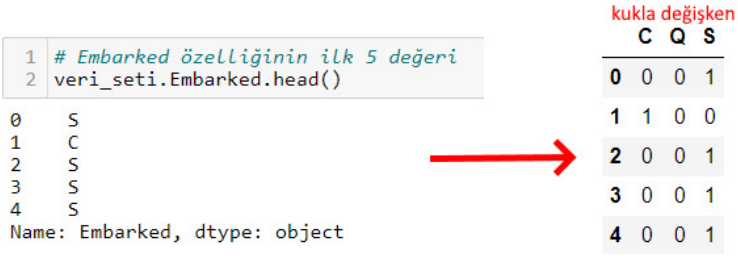
işlemin yapılabilmesi için **kukla değişken** eklenmesi gerekir. Buna İngilizcede "**dummy variables**" denilir. Örneğin Embarked özelliği için kukla değişkenler oluşturularak ikili gösterimler elde edilebilir. Aşağıda bu konuyla ilgili bir örnek gösterilmiştir.

```
# Embarked özelliğinin ilk 5 değeri
veri_seti.Embarked.head()
```

```
0  S
1  C
2  S
3  S
4  S
Name: Embarked, dtype: object
```

```
# kukla değişkenler ile birlikte
pd.get_dummies(veri_seti.Embarked).head()
```

	C	Q	S
0	0	0	1
1	1	0	0
2	0	0	1
3	0	0	1
4	0	0	1



Şekil 3-13: Kukla değişkenler

Ölçeklendirme

Veri setindeki veriler değişik ölçeklerde olabilir. Bu durumda verilerin benzer ölçeklere gelebilmesi için veri ön işleme çalışmaları yapılır. Örneğin yaş değişkeni 20-80 arasında değişirken maaş değişkeni 5000-30000 arasında değişebilir. Bu da veri analizinde istenmeyen zorluklar oluşturabilir. Verilerin benzer ölçekte sunulması ile öğrenme aşamasında daha hızlı bir şekilde öğrenme gerçekleştirilebilir. O nedenle verilerin analizden önce

ölçeklendirilmesi, normalleştirilmesi veya standardize edilmesi gerekmektedir. Böylece veri madenciliği çalışmasının daha iyi bir performansa sahip olması beklenir. Ölçeklendirmede normalizasyon ve standardizasyon bazı durumlarda birbirleri yerine kullanılsa da aslında farklı işlemlerdir. Ancak birbirine benzedikleri söylenebilir. Çünkü bu veri önişleme çalışmalarında verinin dönüşümü sağlanarak veri noktaları değiştirilir. Fark olarak, normalizasyon aşamasında verinin aralığı genellikle [0,1] arasına gelecek şekilde değiştirilirken standardizasyonda verinin dağılım şekli değiştirilmektedir. Ayrıca normalizasyon aşamasında verilerin ölçeği değiştirildiği için **aykırı verilerin ortadan kaybolması** riski de vardır.

Normalizasyon

Bu aşamada veriler belirli bir aralıkta olacak hâle getirilir. Örneğin verileri [0,1] aralığında ölçeklendirmek için Scikit-learn kütüphanesi içerisindeki **`minmax`** fonksiyonu kullanılabilir. Bu ölçeklendirme işlemi için her bir değer minimum değerden (x_{min}) çıkarılır. Ardından ölçeklendirilmiş değer $\widehat{x_i}$, maksimum değerin (x_{max}) minimum değerden farkına bölünerek aşağıdaki gibi hesaplanır.

$$\widehat{x_i} = \frac{x_i - x_{min}}{x_{max} - x_{min}}$$

Aşağıda normalizasyon için bir kod örneği gösterilmiştir. Normalizasyon işlemleri **Pandas** kütüphanesi içerisindeki işlemler ile yazılabileceği gibi daha sonraki bölümlerde gösterilecek olan **Scikit-learn** kütüphanesinin hazır fonksiyonları ile de gerçekleştirilebilir.

```
# Min-max normalizasyonu
mm_normalize_edildi = (veri_seti.Age - veri_seti.Age.min()) / (
    veri_seti.Age.max() - veri_seti.Age.min())
mm_normalize_edildi.head()
```

```
0   0.271174
1   0.472229
2   0.321438
3   0.434531
4   0.434531
Name: Age, dtype: float64
```

Standardizasyon (z-score Normalizasyon)

Standardizasyonda öncelikle ortalaması μ standart sapması σ olan bir *z* skoru hesaplanır. Aşağıdaki formül kullanılarak standardizasyon işlemi gerçekleştirilir. Genellikle normal dağılımın [-3,3] arasında değiştiği varsayılır.

$$z_i = \frac{x_i - \mu_i}{\sigma_i}$$

Standardizasyon işleminin gerçekleştirilmesi için verilerin normal dağılıma uyduğu varsayılır. Bu varsayımla birlikte *z* skoru bulunduktan sonra tüm verilerin yeni değerleri atanmış olur. Örnek bir çalışma aşağıda gösterilmiştir.

```
# standardizasyon
standardize_edildi = (veri_seti.Age - veri_seti.Age.mean())/veri_seti.Age.
std()
standardize_edildi.head()
```

```
0 -0.530005
1  0.571430
2 -0.254646
3  0.364911
4  0.364911
Name: Age, dtype: float64
```

Özet

Bu bölümde veri hakkında genel bilgilendirmede bulunulmuştur. Veri madenciliğinin kalbinde veri vardır. Bu nedenle çeşitli açılardan veri ile ilgili yapıların anlaşılması önem arz etmektedir.

Kaynaklar

1. “Normal Distribution”, https://www.six-sigma-material.com/Normal-Distribution.html.
2. Han, J., Pei, J., and Kamber, M. Data Mining: Concepts and Techniques, Elsevier (2011).
3. Li, C. and Li, H. “A Survey of Distance Metrics for Nominal Attributes”, JSW, 5(11), pp. 1262–1269 (2010).
4. Li, C., Jiang, L., and Li, H. “Local value difference metric”, Pattern Recognition Letters, 49, pp. 62–68 (2014).
5. Zhang, Y. and Cheung, Y.-M. “A New Distance Metric Exploiting Heterogeneous Interattribute Relationship for Ordinal-and-Nominal-Attribute Data Clustering”, IEEE Trans. Cybern., pp. 1–14 (2020).
6. Bock, H.-H. and Diday, E. Analysis of Symbolic Data: Exploratory Methods for Extracting Statistical Information from Complex Data, Springer Science & Business Media (2012).
7. Goodall, D. W. “A new similarity index based on probability”, Biometrics, pp. 882–907 (1966).
8. Le, S. Q. and Ho, T. B. “An association-based dissimilarity measure for categorical data”, Pattern Recognition Letters, 26(16), pp. 2549–2557 (2005).
9. Ahmad, A. and Dey, L. “A method to compute distance between two categorical values of same attribute in unsupervised learning for categorical data set”, Pattern Recognition Letters, 28(1), pp. 110–118 (2007).
10. Stanfill, C. and Waltz, D. “Toward memory-based reasoning”, Communications of the ACM, 29(12), pp. 1213–1228 (1986).
11. “Constants — NumPy v1.21.dev0 Manual”, https://numpy.org/devdocs/reference/constants.html.

BÖLÜM 4

NUMPY İLE BİLİMSEL HESAPLAMALAR VE VERİ DÖNÜŞÜMLERİ

Bu bölümde veri madenciliğinde özellikle veri önişleme aşamasında sıklıkla kullanılan **NumPy** paketi hakkında bilgi verilecektir. **Nümerik Python (NumPy)** kütüphanesi bilimsel hesaplamalar için en çok kullanılan kütüphanelerden birisidir. İçerisindeki çok boyutlu (multidimensional array) desteği ile birçok lineer cebir işlemi kolaylıkla yapılabilir durumdadır.

NumPy dizileri ile Python içerisindeki listelere benzer şekilde bir kullanıma sahiptir. Ancak listeler ile yapılamayan çok boyutlu diziler için NumPy çok daha kullanışlı bir yapı sunmaktadır. Bu nedenle veri madenciliği için gerekli hesaplamalar için sıklıkla NumPy kullanılmaktadır. İlk bölümde bahsedilen şekilde bir **Anaconda** kurulumu gerçekleştirildiyse aşağıda gösterildiği gibi Jupyter notebook üzerinden NumPy sürümü kontrol edilebilir. Kitapta kullanılan NumPy versiyonu aşağıda gösterildiği gibidir:

```
import numpy as np
np.__version__
```

```
'1.19.2'
```

Böylece NumPy kütüphanesini çağırmak için genel kullanım da gösterilmiş oldu. NumPy paketi np olarak kısaltılarak çağrılmıştır. NumPy içerisindeki tüm metot ve sınıflara **`np.`** kısa yolu ile ulaşılabilir olacaktır. Örneğin **`np.`** yazdıktan sonra **Tab** tuşuna basıldığında aşağıdaki gibi bir görüntü elde edilir. Bu şekilde tüm metot ve sınıflar ile ilgili kısa yol görüntülenmiş olur.

Ayrıca jupyter notebook içerisinde NumPy yardım dokümantasyonuna ulaşmak için **`np?`** komutu kullanılabilir. Aşağıda NumPy kütüphanesine ait yardım dokümanı gösterilmiştir.

Bir diğer yardım dokümanına erişme yöntemi de **`<shift>+<tab>`** tuş kombinasyonudur. **`np`** yazdıktan sonra **shift+tab** tuş kombinasyonuna basılınca açılır bir pencerede yardım dokümanına ulaşabiliriz. **Shift+tab** kombinasyonundan sonra **Shift** tuşunu bırakmadan **Tab** tuşuna bir kez daha basılırsa tüm yardım dokümanı açılacak şekilde pencere genişletilebilir. Burada anlatıldığı gibi NumPy ve içerisindeki metotlar hakkında yardım belgelerine ulaşılabilir.

Bu bölümde veri madenciliğinde özellikle veri önişleme aşamasında NumPy kullanımı gösterilecektir. NumPy kütüphanesi bilimsel hesaplamalar için temel bir kütüphanedir. Bu nedenle NumPy kullanımı ile ilgili örnekler yeterli genişlikte verilmiştir. NumPy, sonraki bölümlerde bahsedilecek olan Pandas, Matplotlib ve Scikit-Learn kütüphaneleri de NumPy üzerine bina edilmiştir. Ayrıca Scipy kütüphanesinde optimizasyon, istatistik gibi alanlardaki fonksiyonlar yazılırken NumPy kütüphanesinin çok boyutlu dizilerinden faydalanılmıştır.[1]

Python Dilinde Listeler

Listeler Python programlama dilinde önemli bir yere sahiptir. NumPy dizilerine geçmeden önce Python programlama dili içerisindeki veri tiplerinden tekrar bahsetmekte fayda var.

Python ile gelen listeler **değiştirilebilir** (mutable) ve **sıralanabilir** bir grup elemandan oluşur. Listeler ile string, tuple ve range nesneleri de bir grup objeyi ifade eder. String veri tipinde de listelerde olduğu gibi bir sıralama söz konusudur. String verilerde harfler bir araya gelerek metinleri oluşturur. String içerisindeki harfler **değiştirilemezdir** (immutable). Bu nedenle listelerde yapılabilen bazı operasyonlar string veriler üzerinde yapılamaz. Ancak indeksleme ya da dilimleme gibi operasyonlar değiştirilebilir ve değiştirilemez veri tiplerinde ortak olarak gerçekleştirilebilir. Bu metotlara örnek olarak **`len()`**, **`min()`**, **`max()`** gibi fonksiyonlar verilebilir. Python gömülü listeler ile alakalı örnekler bir önceki bölümde verilmişti, o nedenle NumPy dizileri ile devam edilecektir.

NumPy Dizileri

NumPy kütüphanesinin temelinde çok boyutlu diziler vardır. Python programlama dilinde gömülü olarak gelen liste veri tipinden farklı olarak NumPy dizilerindeki elemanlar aynı veri tipindedir. NumPy içerisindeki en önemli veri yapısı "**ndarray**"dir. Bu sınıf hakkında bilgi almak için aşağıdaki kodlar kullanılabilir:

```
np.ndarray?
# shape: tuple olarak dizinin şekli hakkında bilgi verir.
# size: dizinin büyüklüğü hakkında bilgi verir. Kaç adet eleman var bilgisi.
# Ndim: Dizinin kaç boyutlu olduğu hakkında bilgi verir.
# nbytes: Verilerin ne kadar yer tuttuğunu söyler.
# dtype: Tutulan elemanların hangi tipte veri olduğunu söyler.
veriler = np.array([1,2,3])
veriler.shape
```

```
(3,)
```

```
veriler.size
```

```
3
```

```
veriler.ndim
```

```
1
```

```
veriler.nbytes
```

```
12
```

```
veriler.dtype
```

```
dtype('int32')
```

```
# dtype parametresi değiştirilebilir.
np.array([1,2,4], dtype=np.float)
```

```
array([1., 2., 4.]
```

```
# veri tipi dönüşümü aşağıdaki gibi yapılabilir.
veriler = np.array(veriler, dtype=float)
```

```
dtype('float64')
```

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

```
2
```

```
veriler_iki_boyutlu.size
```

```
6
```

```
veriler_iki_boyutlu.shape
```

```
(2, 3)
```

```
veriler_iki_boyutlu.ndim
```

```
2
```

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

```
veriler_uc_boyutlu.shape
# 1 satır 3 sütündan oluşan 3 adet dizi
```

```
(1, 3, 3)
```

Bir dizi oluşturulduktan sonra veri tipi atama yöntemi ile değiştirilemez. NumPy içerisindeki **astype()** fonksiyonu ile değişkenlerin tipi değiştirilebilir.

NumPy ile çok boyutlu dizi oluşturmak için de birtakım fonksiyonlar vardır. Bu fonksiyonlar sayesinde istenilen büyüklükte ve boyutta diziler oluşturulabilir. Bu fonksiyonlardan bazıları **zeros(), ones(), full(), arange(), linspace(), random()** fonksiyonlarıdır. Belirtilen fonksiyonlara ilişkin örnek kullanımlar aşağıda verilebilir:

```
# 10 adet 0'dan oluşan dizi
print(np.zeros(10, dtype=int))
# 3 satır 5 sütun içerisinde 1'lerden oluşan dizi
print(np.ones((3, 5), dtype=float))
# 3.14'lerden oluşan dizi
print(np.full((3, 5), 3.14))
# Birim matris
print(np.eye(3))
```

```
[0 0 0 0 0 0 0 0 0 0]
[[1. 1. 1. 1. 1.] [1. 1. 1. 1. 1.] [1. 1. 1. 1. 1.]]
```

```
[[3.14 3.14 3.14 3.14 3.14] [3.14 3.14 3.14 3.14 3.14] [3.14 3.14 3.14 3.14 3.14]]
[[1. 0. 0.] [0. 1. 0.] [0. 0. 1.]]
```

Gösterilen şekilde olduğu gibi diziler istenilen formatta, boyutta ve büyüklükte oluşturulabilir. **arange()** fonksiyonu ile belirli düzenlere sahip diziler üretilebilir. Örneğin 0 ile 5 arasındaki tam sayılardan oluşan bir dizi oluşturmak istenirse aşağıdaki kodlar kullanılabilir:

```
# [0,5) arasındaki tam sayılar
print(np.arange(5))
# [3,7) arasındaki tam sayılar
print(np.arange(3,7))
# [0, 10) arasındaki tam sayıları 3'er artması
print(np.arange(0,10,3))
```

```
[0 1 2 3 4]
[3 4 5 6]
[0 3 6 9]
```

Oluşturulan dizilerin boyutlarında değişiklik yapılmak istenirse **reshape()** fonksiyonu kullanılabilir.

```
# [0,9) arasındaki tam sayıların 3 satır 3 sütunda yazılması
np.arange(0,9).reshape(3,3)
```

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

arange() fonksiyonuna benzer bir fonksiyon olan **linspace()** fonksiyonu sayesinde kaç adet elemanın hangi aralıkta üretileceğini belirlenebilir.

```
# [0,9) arasındaki tam sayılardan eşit aralıkta 3 sayı getirme
np.linspace(0,9,3)
```

```
array([0. , 4.5, 9. ])
```

Rassal Veri Oluşturma

Dizinin elemanları rassal veriler olması istenirse NumPy random modülü kullanılabilir. Bu modülün içerisinde birçok dağılıma uygun veri üretilmesi için fonksiyonlar bulunmaktadır. **`Seed()`** fonksiyonu ile rassal sayı üreteci sabit bir sayıya ayarlanarak her seferinde üretilen rassal sayıların aynı olması sağlanabilir. Örnek bir kullanım aşağıda gösterilmiştir:

```
# rassal sayı üretecinin sabitlenmesi
np.random.seed(0)
# 1 ile 10 arasında 3 adet tam sayı üret
r1 = np.random.randint(low=1, high=10,size=3)
print(r1)
# 4 adet 0-1 arasında sayı üret
r2 = np.random.random(4)
print(r2)
```

```
[6 1 4]
[0.85794562 0.84725174 0.6235637  0.38438171]
```

Normal dağılım, üstel dağılım vb. dağılımlara uygun veriler üretmek için aşağıdaki kodlar kullanılabilir:

```
# Belirli şekilde sayı üret
# 3 satır 5 sütundan oluşan sayılar
np.random.seed(0)
r3 = np.random.randint(10, size=(3,5))
print(r3)
# ortalaması 10 standart sapması 2 olan normal dağılıma uyan sayı üret
r4 = np.random.normal(10,2,size=(10,))
print(r4)
```

```
[[5 0 3 3 7]
 [9 3 5 2 4]
 [7 6 8 8 1]]
[10.28808714 12.90854701 11.52207545 10.24335003 10.88772647 10.66734865
 12.98815815  9.58968347 10.6261354   8.29180852]
```

Dizilerde İndeksleme ve Dilimleme

Python listelerinde olduğu gibi NumPy dizilerinde de **köşeli parantez** kullanımı ile dizinin elemanlarına ulaşılabilir. Eğer dizi birden fazla boyutta ise köşeli parantezler artırılmalıdır.

```
x1 = np.random.randint(10, size=6) # Tek boyutlu dizi
x2 = np.random.randint(10, size=(3, 4)) # iki boyutlu dizi
x3 = np.random.randint(10, size=(3, 4, 5)) # üç boyutlu dizi
print(x1[0]) # tek boyutlu dizinin ilk elemanı
print(x1[-1]) # sondan ilk eleman
print(x1[-2]) # sondan ikinci eleman
```

```
9
3
7
```

Dizi içerisinde belirli aralıktaki verileri **almak** için "**:**" işareti kullanılır.

```
# x[start:stop:step] ile dizi içerisinde dilimleme yapılabilir.
x = np.arange(10)  # 0-10 arasındaki sayılar
print(x[:5])  # ilk 5 eleman
print(x[5:])  # 5 elemandan sonrası
print(x[4:7])  # 4. ve 7 elemanlar arası
print(x[::2])  # tüm elemanlar iki iki artır
print(x[::-1])  # elemanları ters çevir
print(x[5::-2])  # 5 ten sonrasını bir sonraki olacak şekilde ters çevir
```

```
[0 1 2 3 4]
[5 6 7 8 9]
[4 5 6]
[0 2 4 6 8]
[9 8 7 6 5 4 3 2 1 0]
[5 3 1]
```

Benzer işlem iki boyutlu dizide yapılırsa:

```
print(x2[:2, :3])  # 2 satır, 3 sütun
print(x2[:3, ::2])  # ilk 3 satır, sonraki olacak şekilde sütunlar
print(x2[::-1, ::-1])  # satırları ve sütunları ters çevir
print(x2[:, 0])  # ilk sütundaki tüm satırlar
print(x2[0, :])  # ilk satırdaki tüm sütunlar
```

```
[[2 7 2]
 [0 4 5]]
```

```
[[2 2]
 [0 5]
 [6 4]]
[[1 4 8 6]
 [5 5 4 0]
 [0 2 7 2]]
[2 0 6]
[2 7 2 0]
```

İç içe dizilerde ise dikkatli davranmak gereklidir.

```
# iç içe diziler(nested) için dilimleme işlemleri
ic_ice = np.array([1,2,[3,4,5, [6,7,8]]])
print(ic_ice[0])
print(ic_ice[2])
```

```
1
[3, 4, 5, [6, 7, 8]]
```

```
print(ic_ice[3])
# dizi aslında 3 elemanlı 3. eleman da bir dizi.
```

```
IndexError                              Traceback (most recent call last)
<ipython-input-79-e7d710525efa> in <module>()
----> 1 print(ic_ice[3])
      2 # dizi aslında 3 elemanlı 3. eleman da bir dizi.
IndexError: index 3 is out of bounds for axis 0 with size 3
```

```
# bu dizi de 4 elemanlı 4. elemanı da bir dizi.
ic_ice[2][3]
```

```
[6, 7, 8]
```

```
ic_ice[2][3][2]
```

```
8
```

Dilimleme işlemi için satır, sütunları ayrı olarak gösterilebilir. Satır, sütun dilimlemesi için **`[satır başlangıç: satır bitiş, sütun başlangıç: sütun bitiş]`** yapısı kullanılabilir. Yani köşeli parantez içerisine bir virgül koyulur. Virgülün sol tarafına satır dilimleme bilgisi sağ tarafına da sütun dilimleme bilgisi girilebilir. Bu kapsamda gerçekleştirilen örnek çalışma aşağıda verilmiştir.

```
x  # tüm veriler
```

```
array([[5, 0, 3, 3],
       [7, 9, 3, 5],
       [2, 4, 7, 6]])
```

```
x[0] # birinci satır veriler
```

```
array([5, 0, 3, 3])
```

```
x[2,:] # üçüncü satır tüm sütunlardaki veriler
```

```
array([2, 4, 7, 6])
```

```
x[:,2] # tüm satırdakiler ve üçüncü sütundaki veriler
```

```
array([3, 3, 7])
```

```
x[[0,2],[1]] # birinci ve üçüncü satırdaki ve ikinci sütundaki veriler
```

```
array([0, 4])
```

```
x[1:3,1:3] # ikinci ve üçüncü satırdaki ikinci ve üçüncü sütundaki veriler
```

```
array([[9, 3],
       [4, 7]])
```

x			
5	**0**	**3**	**3**
7	**9**	**3**	**5**
2	**4**	**7**	**6**

x[0]			
5	**0**	**3**	**3**
7	9	3	5
2	4	7	6

x[2,:]			
5	0	3	3
7	9	3	5
2	**4**	**7**	**6**

x[:,2]			
5	0	**3**	3
7	9	**3**	5
2	4	**7**	6

x[[0,2],[1]]			
5	0	3	3
7	9	3	5
2	4	7	6

x[1:3,1:3]			
5	0	3	3
7	9	3	5
2	4	7	6

Mantıksal Sınama ve Maskeleme (Fancy Indexing)

Dizilerde dilimleme işlemlerinin yanı sıra istenilen şartları sağlayan verilere de ulaşılmak istenebilir. Bu işlem için mantıksal sınamalar kullanılmalıdır.

```
x < 5   # 5'ten küçük olanlara True yazar
```

```
array([[False,  True,  True,  True],
       [False, False,  True, False],
       [ True,  True, False, False]])
```

```
x[x < 5]   # 5 ten küçük olanları getir
```

```
array([0, 3, 3, 3, 2, 4])
```

```
x % 3 == 0 # 3 ile bölünebilen yerlere doğru yazdırır
```

```
array([[False,  True,  True,  True],
       [False,  True,  True, False],
       [False, False, False,  True]])
```

```
x[x % 3 == 0]  # 3 ile bölünebilen sayıları getirir.
```

```
array([0, 3, 3, 9, 3, 6])
```

Dizilerde Aritmetik İşlemler

Çok boyutlu dizilerin oluşturulması ve çeşitli işlemlerinden sonra **aritmetik işlemler**e bakılabilir. NumPy dizileri üst düzey denilebilecek matematiksel işlemler için fonksiyonlar ve verimli bir yapı sunmaktadır. Dizilerde genel aritmetik işlemlere örnekler aşağıda verilmiştir:

```
x = np.arange(4)
print("x =", x)
print("x + 5 =", x + 5)
print("x - 5 =", x - 5)
print("x * 2 =", x * 2)
print("x / 2 =", x / 2)
print("x // 2 =", x // 2)   # kaç sefer var
print("-x = ", -x)
print("x ** 2 = ", x**2)   # karesi
print("x % 2 = ", x % 2)   # mod bölümünden kalan
```

```
x = [0 1 2 3]
x + 5 = [5 6 7 8]
x - 5 = [-5 -4 -3 -2]
x * 2 = [0 2 4 6]
x / 2 = [0.  0.5 1.  1.5]
x // 2 = [0 0 1 1]
-x =  [ 0 -1 -2 -3]
x ** 2 =  [0 1 4 9]
x % 2 =  [0 1 0 1]
```

```
-(0.5 * x + 1)**2 # dizideki elemanları 0.5 ile çarp 1 ekle, karesini al - ile çarp
```

```
array([-1.  , -2.25, -4.  , -6.25])
```

```
abs(-(0.5 * x + 1)**2)  # mutlak değer
```

```
array([1.  , 2.25, 4.  , 6.25])
```

```
print(np.sum(x))  # dizideki elemanların toplamı
print(np.min(x))
print(np.max(x))
print(np.std(x))  # standart sapma
print(np.var(x))  # varyans
print(np.argmin(x))  # minimum değerin indexi
print(np.argmax(x))
print(np.median(x))
print(np.percentile(x, q=25)) # birinci çeyrekteki değer
```

```
6
0
3
1.118033988749895
1.25
0
3
1.5
0.75
```

Matris işlemleri NumPy dizilerinin önemli özelliklerindendir. "*****" operatörü ile iki dizi çarpma işlemine tabi tutulduğunda eğer dizilerin boyutları aynı ise karşılıklı elemanlar çarpılmaktadır. Farklı boyutlardaki diziler için tamamlamalar yapılmaktadır. Matris çarpım ya da nokta çarpım için ise **`np.dot()`** fonksiyonunu kullanılabilir.

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

```
A*B
```

```
array([[ 4,  8, 12],
       [ 5, 10, 15],
       [ 6, 12, 18]])
```

```
np.dot(A,B)
```

```
array([32])
```

Şekil Dönüşümleri

Dizilerde istenilen şekillerde kayıt elde etmek için **reshape()** fonksiyonundan yararlanılır. Örneğin A dizisi, bir satır üç sütundan oluşan bir yapıda iken üç satır bir sütundan oluşacak şekle gelmesi istenirse aşağıdaki gibi **reshape** fonksiyonu kullanılabilir.

```
# Şekil dönüşümleri
A.reshape(3,1)
```

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

```
B
```

```
array([[4],
       [5],
       [6]])
```

```
B.transpose() # sütunları satır satırları sütuna çevirir.
```

```
array([[4, 5, 6]])
```

İki dizinin birleştirilmesi için **vstack()** ve **hstack()** fonksiyonları kullanılır. **vstack()** dikey eksende birleştirme sağlar, **hstack()** ise yatay eksende birleştirme sağlar.

```
C = np.zeros((2, 2))
D = np.ones((2, 2))
```

```
np.vstack((C, D)) # D dizisini C dizisinin altına koyar
```

```
array([[0., 0.],
       [0., 0.],
       [1., 1.],
       [1., 1.]])
```

```
np.hstack((C,D)) # D dizisini C dizisinin sağ tarafına koyar
```

```
array([[0., 0., 1., 1.],
       [0., 0., 1., 1.]])
```

İkiden fazla dizinin birleştirilmesi için **column_stack()** ve **row_stack()** fonksiyonları kullanılabilir.

```
E = np.eye(2)
np.column_stack((C, D, E)) # sütunlarda birleştirir
```

```
array([[0., 0., 1., 1., 1., 0.],
       [0., 0., 1., 1., 0., 1.]])
```

```
np.row_stack((C, D, E)) # satırlarda birleştirir
```

```
array([[0., 0.],
       [0., 0.],
       [1., 1.],
       [1., 1.],
       [1., 0.],
       [0., 1.]])
```

NumPy Dizilerinin Kopyalanması

NumPy dizilerinin yapılan işlemlerden sonra bozulmaması isteniyorsa dizilerin kopyalanması doğru bir yaklaşım olacaktır. "=" operatörü ile atama yapılması dizilerin birinde yapılan değişikliğin diğerini de etkilemesi anlamına gelir.

```
a1 = np.array([1, 2, 3])
b1 = a1
b1
```

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

```
a1[0] = 99 # a1'deki değişiklik b1'i etkileyecektir.
b1
```

```
array([99,  2,  3])
```

Bu durumun olmaması istenirse **copy()** fonksiyonu kullanılmalıdır. Bu fonksiyon sayesinde birbirinden bağımsız diziler oluşturulabilir.

```
# copy işlemi
a1 = np.array([1, 2, 3])
b1 = a1.copy()
a1[0] = 99
b1 # a1'deki değişiklik b1'i etkilemeyecektir.
```

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

Dizilerin Okunması ve Dışarı Aktarılması

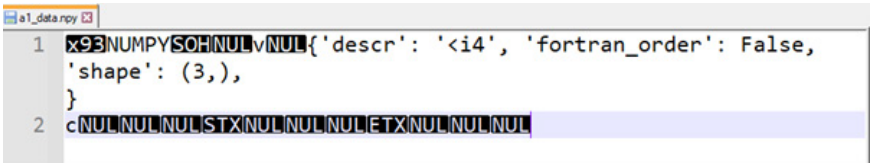
NumPy kütüphanesinin önemli özelliklerinden birisi de dışarıdan veri okumada ve dışarıya veri kaydetmedeki sağladığı kolaylıklardır. Özellikle büyük veri setleri ile çalışmalarda verilerin notebook dosyasında tutulması istenmez. Bunun yerine veri dosyaları klasörde tutulur ve oradan çağrılması sağlanır. NumPy **load** ve **save** fonksiyonları ile dosya okuma ve yazma işlemlerini gerçekleştirir.

```
np.save("a1_data", a1) # bu şekilde .npy dosyası kaydı gerçekleştirilir.
```

```
okunan_a1 = np.load("a1_data.npy")
okunan_a1
```

```
array([99,  2,  3])
```

Bu şekilde yapılan bir kayıt sonucunda **.npy** dosyası NumPy tarafından düzgün bir şekilde okunabilir. Ancak not defteri ile npy dosyasını açmak istediğimizde aşağıdaki gibi bir görüntü alırız.



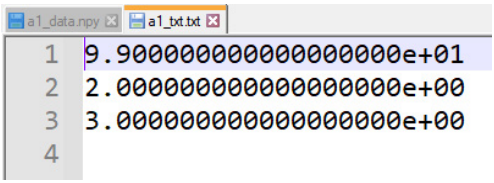
Şekil 4-1: .npy dosyası görüntülenmesi

Daha yaygın bir formatta verilerin kaydedilmesi ile not defterindeki sorun çözülecektir. Örneğin veriler **.csv** veya **.txt** dosyası olarak kaydedilebilir. Dizileri txt formatında kaydetmek için **savetxt()** ve geri çağırmak için **loadtxt()** fonksiyonları kullanılabilir.

```
np.savetxt("a1_txt.txt", a1)
a1_txt_okunan = np.loadtxt("a1_txt.txt")
a1_txt_okunan
```

```
array([99., 2., 3.])
```

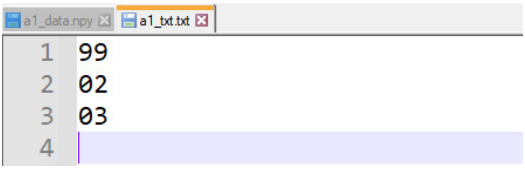
Bu şekilde bir kayıt sonucunda txt dosyası not defteri ile açıldığında şu görüntü elde edilir.



Şekil 4-2: Dosyaya kayıt görüntüsü

Daha düzgün bir kayıt için **savetxt** fonksiyonundaki **fmt** parametresini değiştirmek gerekecektir.

```
np.savetxt("a1_txt.txt", a1, fmt='%.2i') # iki basamaklı tam sayı formatı
```



Şekil 4-3: 2 basamaklı tam sayı kaydı

Eğer klasörde bulunan bir csv dosyasını NumPy dizisi hâlinde çağırmak istenirse **genfromtxt()** fonksiyonu kullanılabilir. Örneğin **titanic** verileri NumPy dizisi şeklinde kaydedilmek istenirse aşağıdaki gibi kodlar kullanılabilir.

```
titanic_verileri = np.genfromtxt("titanic.csv", delimiter=',', names=True)
```

```
titanic_verileri[0]
```

```
(1., 0., 3., nan, nan, nan, 22., 1., 0., nan, 7.25, nan, nan)
```

Görüldüğü gibi NumPy eksik olan veya sayısal olmayan (not a number) veriler yerine '**nan**' değerini yazdı. Bu nedenle csv dosyalarının alınması ile ilgili bir sonraki bölümde gösterilecek olan Pandas kütüphanesinin kullanılması önerilmektedir.

Özet

Bu bölümde **NumPy** kütüphanesinin genel kullanım örnekleri gösterilmiştir. Ayrıca bu bölümde kitabın ilerleyen bölümlerinde kullanılabilecek olan bazı fonksiyonlar ve özelliklerden bahsedilmiştir. Burada gösterilen kullanımlar bilimsel bir hesaplamada kullanılabilecek yapılardır. Veri madenciliği görevlerinin veri analizi kısmında kullanılacak bu fonksiyon ve özelliklerin detaylı bir şekilde bilinmesi önerilmektedir. Bir sonraki bölümde anlatılacak olan Pandas kütüphanesi NumPy dizilerini kullanmaktadır. Yani Pandas içerisindeki verileri NumPy dizisi hâline getirip üzerinde işlem yapabileceğiz. O nedenle Pandas veya diğer kütüphanelerin özelliklerini NumPy kütüphanesi ile düşünmek gereklidir.

Kaynaklar

1. Harris, C. R., Millman, K. J., van der Walt, S. J., et al. “Array programming with NumPy”, Nature, 585(7825), pp. 357–362 (2020).

BÖLÜM 5

PANDAS İLE VERİ SETİ İŞLEMLERİ

NumPy kütüphanesinden sonra detaylı gösterilecek olan ikinci kütüphane Pandas kütüphanesidir. NumPy kütüphanesindeki n boyutlu diziler önemli fonksiyonlar sunsa da veri madenciliğinde verilerin genellikle tablo şeklinde tutulması nedeniyle tablolar üzerinde işlemlerin kolaylaşacağı bir kütüphaneye ihtiyaç vardır. Pandas kütüphanesi açık kaynak kodlu bir kütüphanedir. Özellikle veri analizi için hızlı, güçlü ve esnek bir yapı sunar.[1] Kitabın önceki bölümlerinde veri önişleme aşamaları ile ilgili Pandas pratikleri gösterilmişti. Bu bölümde daha detaylı bir Pandas kullanımından bahsedilecektir. Ayrıca bu bölümde Pandas ile veri okuma, kaydetme gibi işlemler gösterilecektir.

Virgülle Ayrılmış Veri Seti(.csv) Dosyaları

Öncelikle virgülle ayrılmış veri setlerinden ve bu dosyaların özelliklerinden bahsetmek faydalı olacaktır. Çünkü genel kullanımda şu anda verilerin .csv dosyaları halinde tutulduğunu görmekteyiz. Virgülle ayrılmış veri setlerinde sütunlar birbirinden virgül ile ayrılmıştır. Yeni satıra gelen veriler bir satır boşluk bırakılarak yazılır. Örnek bir csv dosyası aşağıda gösterilmiştir. Bu dosyada çalışanlara ilişkin bilgiler virgülle ayrılmış veriler şeklinde tutulmuştur. İlgili veri seti "**employees.csv**" olarak klasöre kaydedilmiştir. İlerleyen kısımlarda kullanılacaktır.

```
First Name,Gender,Start Date,Last Login Time,Salary,Bonus %,Senior Management,Team

Douglas,Male,8/6/1993,12:42 PM,97308,6.945,true,Marketing
```

Thomas,Male,3/31/1996,6:53 AM,61933,4.17,true,

Maria,Female,4/23/1993,11:17 AM,130590,11.858,false,Finance

Jerry,Male,3/4/2005,1:00 PM,138705,9.34,true,Finance

Larry,Male,1/24/1998,4:47 PM,101004,1.389,true,Client Services

Dennis,Male,4/18/1987,1:35 AM,115163,10.125,false,Legal

Ruby,Female,8/17/1987,4:20 PM,65476,10.012,true,Product

,Female,7/20/2015,10:43 AM,45906,11.598,,Finance

Angela,Female,11/22/2005,6:29 AM,95570,18.523,true,Engineering

Frances,Female,8/8/2002,6:51 AM,139852,7.524,true,Business Development

Louise,Female,8/12/1980,9:01 AM,63241,15.132,true,

Julie,Female,10/26/1997,3:19 PM,102508,12.637,true,Legal

Brandon,Male,12/1/1980,1:08 AM,112807,17.492,true,Human Resources

Gary,Male,1/27/2008,11:40 PM,109831,5.831,false,Sales

Virgülle ayrılmış dosyalar veri madenciliğinde en fazla kullanılan dosyalardır. Bu dosyalar incelenirken aşağıdaki dört özelliğe dikkat edilmesi gerekir.

1. **Verilerin birbirinden ayrılması için kullanılan karakter:** Bu karakter genellikle **virgül(,)**'dür. Fakat kimi zaman Tab boşluğu ya da tire(-) ile ayrılan dosyalar ile de karşılaşabiliriz. Ayıran karakter İngilizcede "**seperator**" veya "**delimeter**" olarak geçer.
2. **Ondalık sayı ayıracı için kullanılan karakter:** Genellikle nokta kullanılmaktadır. Ancak bazı durumlarda standardın dışında virgül ile ayrılan ondalık ayıraçlarından da bahsedilebilir. Ondalık sayı ayıracı İngilizcede "**decimal**" olarak geçer.
3. **Veri setinde sütun isimlerinin olup olmaması:** Genellikle ilk satırda veri setinin sütun isimleri yer almaktadır. Bazı durumlarda sütun isimlerinin olmaması ile karşılaşılabilir. Bu durumda sütun isimlerinin dışarıdan eklenmesi gereklidir.
4. **İndeks sütununun olup olmaması:** Genellikle ilk sütunda yer alan veriler indeks sütununu göstermektedir. İndeks sütununda kimi zaman verilerin numarasını kimi zaman meydana geliş zamanını gösterir. Eğer indeks sütunu dosyada tanımlanmadıysa veri seti Pandas içerisine alındığında otomatik indeks sütunu oluşturulacaktır.

Virgülle ayrılmış dosyaları açmak için bir metin okuyucu ya da MS Excel gibi tablo uygulamaları yeterlidir. Virgülle ayrılmış dosyaların dışında tablo şeklindeki verilerin kaydedildiği diğer yaygın formatlar **.xlsx, .txt** olarak söylenebilir. Okunacak dosyanın uzantısına göre Pandas içerisinde özelleştirilmiş fonksiyonlar bulunmaktadır. Örneğin csv dosyalarının oluşturulması için **read_csv()**, Excel dosyalarının okunması için **read_excel()**, HTML verilerinin okunması için **read_html()**, **JSON** (JavaScript Object Notation) dosyaları için **read_json()** gibi fonksiyonlar bulunmaktadır.

Aşağıda gösterilen kodlar ile csv dosyası Pandas içerisine alınabilir.

```
import pandas as pd
calisan_verileri = pd.read_csv("employees.csv")
calisan_verileri.head()
```

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	8/6/1993	12:42 PM	97308	6.945	True	Marketing
1	Thomas	Male	3/31/1996	6:53 AM	61933	4.170	True	NaN
2	Maria	Female	4/23/1993	11:17 AM	130590	11.858	False	Finance
3	Jerry	Male	3/4/2005	1:00 PM	138705	9.340	True	Finance
4	Larry	Male	1/24/1998	4:47 PM	101004	1.389	True	Client Services

Bu örnekte varsayılan olarak gelen parametreler ile veri setini Pandas içerisine almak mümkün olabildi. Csv dosyasının durumuna göre header, names, decimal, sep gibi otuzdan fazla parametrelerin ayarlanması gerekecektir. **read_csv()** fonksiyonu yerine **read_table()** fonksiyonu da kullanılabilir. Bu durumda parametrelerin aşağıdaki gibi verilmesi gerekir.

```
calisan_verileri_tablo = pd.read_table("employees.csv", sep=',',
decimal='.')
calisan_verileri_tablo.head()
```

Csv dosyası alındıktan sonra ilk görülmek istenen bilgilerden birisi de kaç adet veri ve özellik olduğu bilgisidir. Bunun için **shape()** fonksiyonu satır ve sütun sayısını gösterir. Bu komutlar çalıştırıldığında ilk elde edilen sayı satır sayısıdır. İkinci gösterilen sayı da sütun sayısını gösterir. Bu örnekte 1000 adet kişinin 8 adet özelliğinden oluşan bir veri setinin olduğu görülebilir.

```
calisan_verileri.shape
```

(1000, 8)

Csv dosyaları Pandas içerisine alındıktan ve işlem yapıldıktan sonra tekrar klasöre kaydedilmek istenebilir. Bu durumda **to_csv()** fonksiyonu ile yeniden klasöre kayıt yapılabilir.

```
calisan_verileri.to_csv("calisan_veriler.csv")
```

NumPy kütüphanesindeki en önemli veri yapısının diziler olduğu söylenmişti. Pandas kütüphanesinde üç önemli veri yapısından (Pandas objesinden) bahsedilebilir. Bunlar **Series** (Seri), **DataFrame** (Veri Çerçevesi) ve **Pandas İndeksleri** veri yapılarıdır.

Pandas Serileri

Pandas içerisindeki tek boyutlu veri yapılarıdır. NumPy kütüphanesindeki dizilere ya da Python'da gömülü olarak gelen listelere benzer. Dizilerden farkı serilerin içerisinde ayrıca indeks değerinin de olmasıdır.

```
seri_ornegi = pd.Series([1,2,3,5,8]) # Seriler tek boyutlu diziler gibidir.
seri_ornegi
```

```
0  1
1  2
2  3
3  5
4  8
dtype: int64
```

```
print(seri_ornegi.values) # serideki elemanlara ulaşmak için
print(seri_ornegi.index) # serideki indeks değerleri
```

```
[1 2 3 5 8]
RangeIndex(start=0, stop=5, step=1)
```

Pandas serilerine eğer indeks değerleri belirtilmediyse örnekte görüldüğü gibi sıfırdan başlayarak eleman sayısı kadar artırılan bir dizi olarak indeks değerleri verilmektedir. Eğer indeks değerleri verilmek istenirse aşağıdaki yapı kullanılabilir:

```
seri_ornegi = pd.Series([1,2,3,5,8], index=['a','b','c','d','e'])
seri_ornegi['c'] # serinin elemanlarına indeks değeri ile ulaşılır
```

Serideki elemanlar arasında NumPy dizilerinde olduğu gibi dilimleme, indeksleme işlemleri yapılabilir.

```
print(seri_ornegi[1:3])
```

```
b 2
c 3
dtype: int64
```

Veri Çerçevesi

Veri çerçevesi (DataFrame) Pandas kütüphanesinin en önemli çekirdek objesidir. Birden fazla serinin bir araya gelmesi ile oluştuğu söylenebilir. Satır ve sütunlardan oluşan veri yapılarıdır. Yani tablo verileri olarak söylenebilir. Aşağıda gösterildiği gibi oluşturulabilir:

```
# dataframe
import numpy as np
veri_cercevesi = pd.DataFrame(np.random.rand(3, 2),
                  columns=['sutun 1', 'sutun 2'],
                  index=['a', 'b', 'c'])
```

```
     sutun 1 sutun 2
a    0.3775890.364145
b    0.4873180.126421
c    0.0760780.042498
```

```
# başka veri çerçevesi oluşturma yöntemi
plaka_kodlari = {'Istanbul': '34', 'Sakarya': '54', 'KahramanMaraş': '46'}
nufus_sayilari = {
    'Istanbul': 20000000,
    'Sakarya': 1100000,
    'KahramanMaraş': 1000000
}
sehirler = pd.DataFrame({'plakalar':plaka_kodlari,'nufus':nufus_sayilari})
sehirler
```

```
                plakalar  nufus
Istanbul        34              20000000
Sakarya         54              1100000
KahramanMaraş   46        1000000
```

Birinci örnekte **veri_cercevesi** değişkeninin içerisine iki sütun oluşturulmuştur. Satır olarak söylenebilecek indeks değerleri de a, b, c olarak verilmiştir. Veriler ise NumPy rassal sayıları olarak üretilmiş oldu.

İkinci örnekte ise veri çerçevesinin verileri sözlük yapısı hâlinde verilmiştir. Bu durumda indeks bilgileri sözlük içerisindeki anahtar değer içerisindeki değerler ise anahtarların karşısındaki değerler olarak söylenebilir.

Pandas İndeksleri

Seri ve veri çerçevelerinden sonra Pandas içerisindeki diğer veri yapısı **indeksler**dir. İndeksler değiştirilemez objelerdir. Tablodaki indeksler bir veriye ulaşmada kullanılan alanlardır. İndekslerin kendine özel yapısına ilişkin özellikler aşağıda gösterilmiştir:

```
indeks_1 = pd.Index([1, 2, 3, 4, 5])
indeks_2 = pd.Index([2, 4, 6])
```

```
indeks_1[2] = 3 # indeks elemanları değiştirilemezdir.
```

```
--------------------------------------------------------------------------- TypeError Traceback (most recent call last)
<ipython-input-44-649110818a02> in <module>() ----> 1 indeks_1[2] = 3 # indeksler değiştirilemezdir.
D:\Miniconda3\lib\site-packages\pandas\core\indexes\base.py in __setitem__(self, key, value)
4275 @final
4276 def __setitem__(self, key, value):
-> 4277 raise TypeError("Index does not support mutable operations")
4278
4279 def __getitem__(self, key):
TypeError: Index does not support mutable operations
```

```
print(indeks_1.intersection(indeks_2)) # kesişim kümesi
print(indeks_1.union(indeks_2)) # birleşim kümesi
print(indeks_1.difference(indeks_2)) # fark kümesi
```

```
Int64Index([2, 4], dtype='int64')
Int64Index([1, 2, 3, 4, 5, 6], dtype='int64')
Int64Index([1, 3, 5], dtype='int64')
```

İndeksler birleşim, keşişim veya fark kümeleri gibi işlemler sayesinde bir arada kullanılabilirler. Böylece iki veri çerçevesinin birleştirilmesi, keşişim kümelerinin oluşturulması gibi işlemler için bu özellikler faydalı olacaktır.

İndeksleme, Seçme, Dilimleme, Filtreleme İşlemleri

Pandas seri ve veri çerçevelerindeki istenilen satır veya sütuna ulaşabilmek için NumPy dizilerinde olduğu gibi dilimleme işlemlerine **alternatif** olarak **loc** ve **iloc** operatörleri kullanılabilir. Kısaca söylenmesi gerekirse **iloc** operatörü satırları numaraları ile çağırmak için kullanılır. **loc** operatörü ise verileri etiketleri veya mantıksal sınamaları ile çağırmada kullanılır. Örneğin **iloc** operatörü **veri.iloc[<satır sayıları>, <sütun sayıları>]** şeklinde çalışmaktadır. Bu kullanımlar NumPy dizilerindeki dilimleme ile benzerlik göstermektedir. Eğer bir Pandas veri çerçevesindeki verilere direkt köşeli parantezler ile ulaşılmak istenirse sorun yaşanabilir. Pandas güncel versiyonunda **iloc** ve **loc** kullanımını temel almaktadır.

```
sehirler.iloc[1,0] # ikinci şehrin(Sakarya)'nın birinci özellğini(plakasını) getirir
```

```
'54'
```

```
sehirler.iloc[:,0] # tüm satırlar birinci sütun
```

```
Istanbul        34
Sakarya          54
KahramanMaraş    46
Name: plakalar, dtype: object
```

```
sehirler.iloc[:2, :2] # ilk 2 satır, ilk 2 sütun
```

	plakalar	nufus
Istanbul	34	20000000
Sakarya	54	1100000

```
sehirler.iloc[[0, 1], 1] # Belirli satır ve sütunları getirebiliriz
```

```
Istanbul    20000000
Sakarya      1100000
Name: nufus, dtype: int64
```

İndeksleme ile ilgili diğer kullanım tarzı da **loc** operatörü ile gerçekleştirilebilir. **loc** ile satırları etiketleri veya indeks değerleri ile getirilebilir. Aynı zamanda mantıksal sorgulamalar da yapılabilir. **loc** da **iloc** gibi **veri.loc[<satır sayıları>, <sütun sayıları>]** şeklinde çalışır. **iloc** kullanımına göre daha fazla dikkatli kullanmak gerekebilir. Çünkü verilerin etiketlerine ulaşılarak çalışılması gerekir.

```
print(sehirler.loc['Istanbul']) # istanbul'a ait bilgileri getirir.
```

```
plakalar        34
nufus       20000000
Name: Istanbul, dtype: object
```

```
print(sehirler.loc['Istanbul','nufus']) # istanbul'un nüfusunu getirir.
```

```
20000000
```

```
print(sehirler.loc[['Istanbul','Sakarya'],'nufus']) # istanbul ve sakarya'nın nüfusunu getirir.
```

```
Istanbul    20000000
Sakarya      1100000
Name: nufus, dtype: int64
```

```
print(sehirler.loc[['Istanbul','Sakarya'],['nufus','plakalar']]) # nüfus ve plakaları getirir.
```

```
          nufus    plakalar
Istanbul  20000000    34
Sakarya   1100000     54
```

`loc` operatörü ile aynı zamanda **mantıksal sınamalar** da yapılabilir. Örneğin nüfusu bir milyondan fazla olan şehirler getirilmek istenirse aşağıdaki gibi bir yapı kullanılabilir:

```
sehirler.loc[sehirler.nufus>1000000]
```

```
          plakalar    nufus
Istanbul      34      20000000
Sakarya       54      1100000
```

```
sehirler.loc[sehirler.plakalar=='34'] # plakası 34 olan şehri getir
```

```
          plakalar    nufus
Istanbul      34      20000000
```

Yukarıdaki örnekte gösterildiği gibi bir sütunun değerlerine ulaşabilmek için "`.`" İşareti kullanılabilir. "`.`" İşaretinin kullanılabilmesi için sütun isminde Türkçe karakter veya boşluk karakteri bulunmaması gereklidir. Eğer sütunda boşluk karakteri varsa aşağıdaki gibi köşeli parantezlerden (`[]`) oluşan yapı tercih edilebilir.

```
sehirler['nufus']
```

Veri atamaları için ise "=" operatörü kullanılır. Örneğin İstanbul şehrinin nüfusu 25000000 olarak **güncellemek** istenirse hem `loc` hem de `iloc` ile aşağıdaki gibi güncellenebilir:

```
sehirler.iloc[0,1] = 25000000
sehirler.loc['Istanbul', 'nufus'] = 25000000
```

Yeni bir sütun ilave etmek için sütun ismini yazdıktan sonra indeks değerlerine karşılık gelen değerler girilebilir:

```
sehirler['bolge'] = ['Marmara', 'Marmara', 'Akdeniz'] # bolge isminde bir sütun oluşturur ve içerisine verileri girer.
sehirler
```

	plakalar	nufus	bolge
Istanbul	34	25000000	Marmara
Sakarya	54	1100000	Marmara
KahramanMaraş	46	1000000	Akdeniz

Pandas Kütüphanesindeki Bazı Önemli Fonksiyonlar

value_counts(): Tek değerleri çıkararak tek değerlerden kaç adet olduğunu gösteren fonksiyondur. Çıktı olarak Pandas serisi gönderir. İçerisine sunulacak parametrelere göre değerlerin normalize edilmesi, sıralanması, eksik verilerin düşürülmesi gibi özellikleri de bulunmaktadır. Çalışan veri setinde hangi takımdan kaç kişinin olduğunu görüntülemek için bu fonksiyondan yararlanılabilir.

```
calisan_verileri.Team.value_counts() # Hangi takımda kaç kişi var?
```

```
Client Services         106
Finance                 102
Business Development    101
Marketing                98
Product                  95
Sales                    94
Engineering              92
Human Resources          91
Distribution             90
Legal                    88
Name: Team, dtype: int64
```

```
calisan_verileri.Team.value_counts(normalize=True) # normalize edildiğinde
```

```
Client Services          0.110763
Finance                  0.106583
Business Development     0.105538
Marketing                0.102403
Product                  0.099269
Sales                    0.098224
Engineering              0.096134
Human Resources          0.095089
Distribution             0.094044
Legal                    0.091954
Name: Team, dtype: float64
```

describe(): Pandas veri çerçevesi veri yapısının içerisinde istatistiksel bilgiler ile ilgili bir önemli fonksiyonlar bulunmaktadır. Bunlardan birisi `describe()` fonksiyonudur. Bu fonksiyon sayesinde hızlı bir şekilde veri sayıları, ortalama, standart sapma gibi temel istatistikler verilebilir. Aynı zamanda örneğin `mean()` fonksiyonu ile satır veya sütunlarda ayrı ayrı ortalamalar gösterilebilir.

```
calisan_verileri.describe()
```

	Salary	Bonus %
count	1000.000000	1000.000000
mean	90662.181000	10.207555
std	32923.693342	5.528481
min	35013.000000	1.015000
25%	62613.000000	5.401750
50%	90428.000000	9.838500
75%	118740.250000	14.838000
max	149908.000000	19.944000

```
calisan_verileri.mean(0) # sütunlara göre ortalama
```

```
Salary              90662.181000
Bonus %                10.207555
Senior Management       0.501608
dtype: float64
```

```
calisan_verileri.mean(1) # satırlara göre ortalamalar
```

```
0    48657.4725
1    30968.5850
2    65300.9290
3    69357.1700
4    50502.6945
```

```
      ...
995   66249.8275
996   21205.8375
997   48457.7105
998   30255.9925
999   64979.5845
Length: 1000, dtype: float64
```

Salary özelliğinin en küçük, en büyük ve ortalamalarının gösterilmesi istenirse aşağıdaki gibi tek tek de çalıştırılabilir.

```
print (calisan_verileri.Salary.min(), calisan_verileri.Salary.max(),
calisan_verileri.Salary.mean())
```

```
35013 149908 90662.181
```

Unique(): Bir özellikteki tek olarak geçen değerler getirilmek istenirse **`unique()`** fonksiyonu kullanılabilir. Özellikle veri temizleme aşamasında yazım hatası olup olmadığının kontrolü için **`unique()`** fonksiyonu ile tarama yapmakta fayda vardır.

```
calisan_verileri.Team.unique() # Var olan takımları liste olarak getirir.
```

```
array(['Marketing', nan, 'Finance', 'Client Services', 'Legal', 'Product', 'Engineering', 'Business
Development', 'Human Resources', 'Sales', 'Distribution'], dtype=object)
```

groupby(): SQL sorgu dilinde çalışan **`groupby`** fonksiyonu benzer şekilde Pandas içerisinde de çalışabilir.

```
calisan_verileri.groupby('Team').sum() # takımlara ait istatistikler
```

	Salary	Bonus %
Team		
Business Development	9278498	1067.810
Client Services	9351789	1112.481
Distribution	7965042	865.408
Engineering	8672766	962.595
Finance	9406387	1039.061
Human Resources	8275952	909.443
Legal	7858718	908.409
Marketing	8862688	1014.638
Product	8423223	930.191
Sales	8664303	950.990

```
calisan_verileri.groupby('Team').Team.count() # hangi takımda kaç kişi var?
```

```
Team
Business Development    101
Client Services          06
Distribution             90
Engineering              92
Finance                 102
Human Resources          91
Legal                    88
Marketing                98
Product                  95
Sales                    94
Name: Team, dtype: int64
```

apply(): Örneğin tüm çalışanların maaşlarına zam yapılmak istenirse `apply()` fonksiyonunda yararlanılabilir. Bu fonksiyon ile herhangi bir işlem sütunlarda yapılabilir. Yani burası Excel'deki fonksiyonlar gibi düşünülebilir. Excel'de yazılacak fonksiyonlar `apply()` fonksiyonu sayesinde Pandas içerisinde de yazılabilir.

```
calisan_verileri.Salary.apply(lambda x:x+1000) # Maaşlara 1000'lik bir artış
gerçekleştirildi.
```

```
0     98308
1     62933
2    131590
3    139705
4    102004
     ...
995  133483
996   43392
997   97914
998   61500
999  130949
Name: Salary, Length: 1000, dtype: int64
```

Map(): `Map()` fonksiyonu sayesinde birtakım veriler istenilen değerlere dönüşümü sağlanabilir. `Apply()` fonksiyonunda olduğu gibi bu fonksiyonun içerisinde de kullanıcı tanımlı fonksiyonlar kullanabiliriz. Örneğin maaş değişkeninin değerleri ortalama değerden çıkarılarak yeni bir sütunda yazılmak istenirse aşağıdaki yapı kullanılabilir.

```
calisan_verileri['ortalamadan_sapma']= calisan_verileri.Salary.map(lambda
x: x - calisan_verileri.Salary.mean() )
calisan_verileri.head()
```

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team	ortalamadan_sapma
0	Douglas	Male	8/6/1993	12:42 PM	97308	6.945	True	Marketing	6645.819
1	Thomas	Male	3/31/1996	6:53 AM	61933	4.170	True	NaN	-28729.181
2	Maria	Female	4/23/1993	11:17 AM	130590	11.858	False	Finance	39927.819
3	Jerry	Male	3/4/2005	1:00 PM	138705	9.340	True	Finance	48042.819
4	Larry	Male	1/24/1998	4:47 PM	101004	1.389	True	Client Services	10341.819

Kullanıcı tanımlı fonksiyonlar yazılarak **apply()** fonksiyonu içerisinde çağrılabilir. Örneğin aynı takımda yer alan kişilerin maaşları kendi içerisinde ölçeklendirmek istenirse aşağıdaki yapı kullanılabilir.

```
def olcekle(x):
    x['Salary'] /= x['Salary'].sum()
    return x
```

calisan_verileri.groupby(by='Team').apply(olcekle).head()

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team	ortalamadan_sapma
0	Douglas	Male	8/6/1993	12:42 PM	0.010980	6.945	True	Marketing	6645.819
2	Maria	Female	4/23/1993	11:17 AM	0.013883	11.858	False	Finance	39927.819
3	Jerry	Male	3/4/2005	1:00 PM	0.014746	9.340	True	Finance	48042.819
4	Larry	Male	1/24/1998	4:47 PM	0.010801	1.389	True	Client Services	10341.819
5	Dennis	Male	4/18/1987	1:35 AM	0.014654	10.125	False	Legal	24500.819

Yukarıdaki örnekte veriler öncelikle takımlarına göre gruplanmıştır. Ardından olcekle fonksiyonu ile kendi toplamlarına bölünecek şekilde ölçeklendirilmiştir.

sort_values(): İstenilen sütuna göre sıralama yapılmasını sağlar. Örneğin aylık ücretlere göre sıralama yapılmak istenirse aşağıdaki yapı kullanılabilir.

```
calisan_verileri.sort_values(by='Salary').head()
```

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team	ortalamadan_sapma
576	Michael	Male	7/30/1993	5:35 PM	35013	14.879	False	Product	-55649.181
238	Kevin	Male	3/25/1982	7:31 AM	35061	5.128	False	Legal	-55601.181

82	Steven	Male	3/30/1980	9:20 PM	35095	8.379	True	Client Services	-55567.181
63	Matthew	Male	1/2/2013	10:33 PM	35203	18.040	False	Human Resources	-55459.181
650	Cynthia	Female	7/5/1986	1:24 AM	35381	11.749	False	Finance	-55281.181

Sort_values() fonksiyonu varsayılan olarak artan sıralama yapmaktadır. Eğer azalan sıra yapılmak istenirse **ascending** parametresi **False** olarak ayarlanmalıdır.

```
calisan_verileri.sort_values(by='Salary', ascending=False).head()
```

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team	ortalamadan_sapma
644	Katherine	Female	8/13/1996	12:21 AM	149908	18.912	False	Finance	59245.819
429	Rose	Female	5/28/2015	8:40 AM	149903	5.630	False	Human Resources	59240.819
828	Cynthia	Female	7/12/2006	8:55 AM	149684	7.864	False	Product	59021.819
186	NaN	Female	2/23/2005	9:50 PM	149654	1.825	NaN	Sales	58991.819
160	Kathy	Female	3/18/2000	7:26 PM	149563	16.991	True	Finance	58900.819

Birden fazla sütuna göre sıralama yapılmak istenirse bunun için de **by** parametresinin içerisine liste şeklinde sütun isimleri verilmelidir.

```
calisan_verileri.sort_values(by=['Salary', 'Bonus %'], ascending=False).
head()
```

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team	ortalamadan_sapma
644	Katherine	Female	8/13/1996	12:21 AM	149908	18.912	False	Finance	59245.819
429	Rose	Female	5/28/2015	8:40 AM	149903	5.630	False	Human Resources	59240.819
828	Cynthia	Female	7/12/2006	8:55 AM	149684	7.864	False	Product	59021.819
186	NaN	Female	2/23/2005	9:50 PM	149654	1.825	NaN	Sales	58991.819
160	Kathy	Female	3/18/2000	7:26 PM	149563	16.991	True	Finance	58900.819

rename(): İndeks veya özellik isimlerini değiştirmede kullanılır. Örneğin '**Bonus %**' olarak verilen sütun arada boşluk olmadan yazılmak istenirse;

```
calisan_verileri.rename(columns= {'Bonus %': 'Bonus'}).head()
```

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus	Senior Management	Team	ortalamadan_sapma
0	Douglas	Male	8/6/1993	12:42 PM	97308	6.945	True	Marketing	6645.819
1	Thomas	Male	3/31/1996	6:53 AM	61933	4.170	True	NaN	-28729.181
2	Maria	Female	4/23/1993	11:17 AM	130590	11.858	False	Finance	39927.819
3	Jerry	Male	3/4/2005	1:00 PM	138705	9.340	True	Finance	48042.819
4	Larry	Male	1/24/1998	4:47 PM	101004	1.389	True	Client Services	10341.819

insert(): Yeni bir özellik tabloya eklendiğinde tablonun sonuna eklenir. `insert()` fonksiyonu ile istenilen bir konuma yeni bir özellik eklenebilir.

```
yeni_sutun = np.random.randint(1000)
calisan_verileri.insert(4,'yeni_sutun', yeni_sutun)
```

sample(): Veri seti içerisinde istenilen kadar rassal veri seçmek için bu fonksiyon kullanılabilir.

```
calisan_verileri.sample(2)
```

	First Name	Gender	yeni_sutun	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team	ortalamadan_sapma
77	Charles	Male	747	9/14/2004	8:13 PM	107391	1.260	True	Marketing	16728.819
389	Sharon	NaN	747	10/28/1984	5:59 PM	97635	10.413	True	Client Services	6972.819

nunique(): Özellik içerisindeki verilerin tek olan elemanlarının sayısını gösterir. Örneğin kaç adet takım olduğu bilgisini almak için bu fonksiyon kullanılabilir.

```
calisan_verileri.nunique()
```

```
First Name          200
Gender                2
yeni_sutun            1
Start Date          972
Last Login Time     720
Salary              995
Bonus %             971
Senior Management     2
Team                 10
ortalamadan_sapma   995
dtype: int64
```

select_dtypes(): Belirli veri tiplerine sahip verileri seçmek için kullanılır. **include** parametresi ile içerilecek veri tipleri **exclude** parametresi ile çıkarılacak veri tipleri söylenebilir.

```
calisan_verileri.select_dtypes(include='float').head()
```

	Bonus %	ortalamadan_sapma
0	6.945	6645.819
1	4.170	-28729.181
2	11.858	39927.819
3	9.340	48042.819
4	1.389	10341.819

Özet

Bu bölümde Pandas paketi ile alakalı örneklere yer verilmiştir. Daha detaylı gösterimler yapılabilirdi. Ayrıca Pandas içerisinde grafik çizimi ile alakalı fonksiyonlardan da bahsedilebilirdi. Ancak bu kitapta grafik çizimleri için Matplotlib kütüphanesinin imkânlarından faydalanacağı için o kısımlar gösterilmemiştir. Bir sonraki bölümde veri görselleştirilmesi ile ilgili olan **Matplotlib kütüphanesinin** genel kullanımı gösterilecektir.

Kaynaklar

1. Reback, J., McKinney, W., Jbrockmendel, et al. Pandas-Dev/Pandas: Pandas 1.2.4, Zenodo (2021).

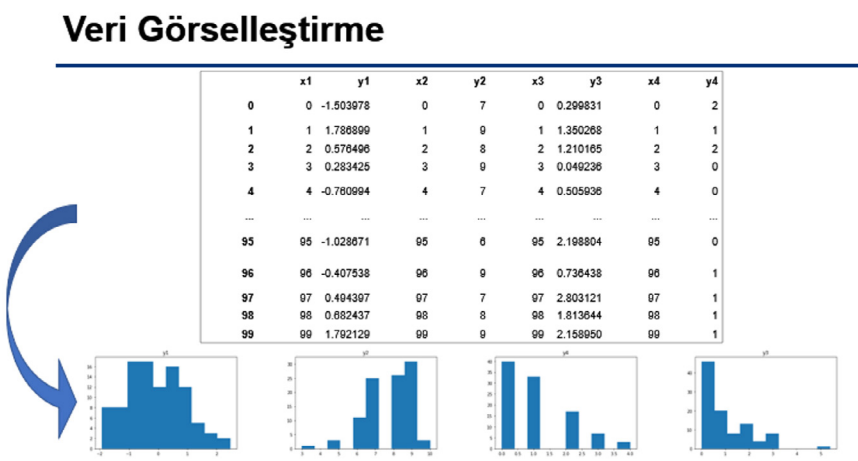
BÖLÜM 6

MATPLOTLİB İLE VERİ GÖRSELLEŞTİRME

Python ile veri madenciliği uygulamaları geliştirmek için kullanılacak bir diğer kütüphane **Matplotlib** kütüphanesidir. Matplotlib kütüphanesi veri görselleştirme kütüphanesi olarak bilinir. Verilerin görselleştirilmesi ise veri analizi aşamasını kolaylaştıran bir süreçtir. Bu bölümde verilerin doğru şekilde görselleştirilmesi ve görüntülenmesi için kullanılabilecek birtakım yöntemlerden bahsedilecek ve Matplotlib kütüphanesi tanıtılacaktır.

Matplotlib Kütüphanesi

Matplotlib, 2 ve 3 boyutlu grafiklerin çizilmesi için geliştirilmiş, ana sayfası (*http://matplotlib.org*) adresinde bulunan, özellikle bilimsel ve mühendislik alanlarındaki veri görselleştirmelerinde sıklıkla kullanılan bir kütüphanedir. **Veri görselleştirme** ihtiyacı, verinin olduğu her alanda geçerlidir. Çünkü büyük veri kümelerindeki veriyi nitelendiren özelliklere ve ölçülere genel çerçeveden bakılmak istendiğinde görsel araçlara ihtiyaç duyulur. Örneğin bir veri setinin dağılımı ile ilgili histogram grafikleri hızlı bir izlenim imkânı sunar. Aşağıdaki görselde tablo ile sunulamayacak bilgilerin veri görselleştirme ile göze hitap edecek şekilde sunulması örneği gösterilmiştir. Tablodaki veriler hakkında özet bilgiyi alabilmek ve veri dağılımlarını görüntüleyebilmek için veri görselleştirmesi adımları uygulanabilir.



Şekil 6-1: Veri görselleştirme

Matplotlib kütüphanesi NumPy üzerine kurulmuş bir kütüphanedir. Bu nedenle Matplotlib kütüphanesinin çağrılmasından önce NumPy kütüphanesinin de yüklü kütüphaneler arasında olması gereklidir. Kısaca Matplotlib kütüphanesinin **önemli avantajları** şu şekilde sunulabilir:

- İnteraktif grafikler sayesinde grafik üzerinde yakınlaşma, uzaklaşma, güncelleştirme özellikleri kullanılabilir.
- Baskı kalitesi yüksek grafikler .svg, .pdf gibi formatlarda kaydedilebilir.
- Grafiğe normal metinler ile matematiksel ifadeler de LaTeX metinleri olarak yazılabilir.
- Yaygın kullanımı ve geniş kullanım rehberleri vardır.
- Statik, hareketli ve interaktif görselleştirmeler sunabilir.

Eğer Matplolib kütüphanesi bilgisayarda kurulu değilse aşağıdaki komutlar ile kurulumu gerçekleştirilebilir.

```
!conda install matplotlib
```

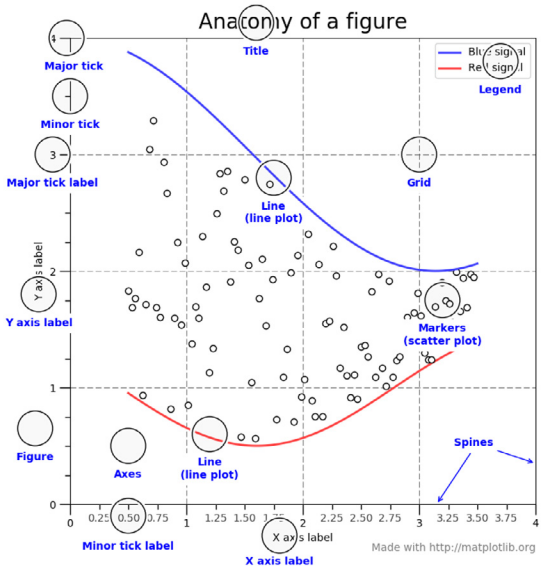
Matplotlib Grafiklerine Giriş

Matplotlib ile görselleştirmelere geçmeden önce Matplotlib katmanlarından bahsetmekte fayda vardır. Matplotlib 3 temel katmandan oluşur. Bunlar sınıfların ve objelerin tutulduğu **geri katman (backend)**, grafik çizdirmek için gerekli bileşenlerin olduğu **çizim katmanı (artist)**, hesaplama ve görselleştirme araçlarının bulunduğu **script** katmanlarıdır.

Klasik bir Matplotlib kütüphane çağrılma yöntemi aşağıda verilmiştir.

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

Bu kod pyplot isimli modülün çağrılmasını sağlar. Bu modül sayesinde Matplotlib grafikleri, eksenleri ve aksisleri çizdirilebilir. Aşağıdaki görselde pyplot ile çizilen figürlerin objeleri gösterilmiştir. Figür, içerisinde eksenlerin olduğu görseli ifade eder. Birden fazla eksenden oluşabilir. Eksenler ise aksislerden oluşur. Aksisler 2 boyutlu bir sistemde 2 adet değişkenin verilerinden oluşan bir yapıdır.



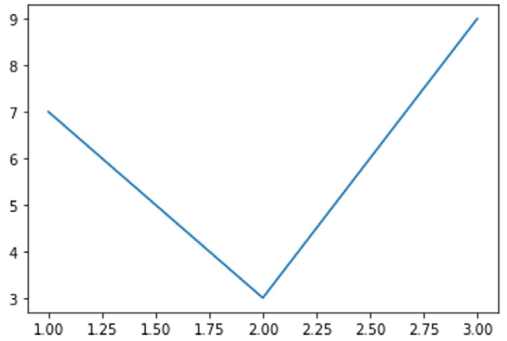
Şekil 6-2: Matplotlib figürlerinin anatomisi[1]

Aşağıdaki kodlar ile Matplotlib kütüphanesi çağrılmış olunur. Aynı zamanda NumPy ve Pandas kütüphanelerinin de başlangıçta çağrılmasında fayda vardır.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# from matplotlib import pyplot as plt
```

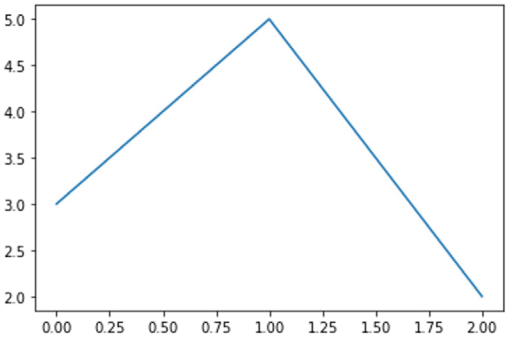
Basit bir Matplotlib grafiği kodları aşağıda paylaşılmıştır. **`plt.plot()`** ile x değerlerine karşılık gelen y değerleri çizdirilebilir.

```
plt.plot([1,2,3],[7,3,9]) # (1,7), (2,3), (3,9) noktalarını birleştirir
```



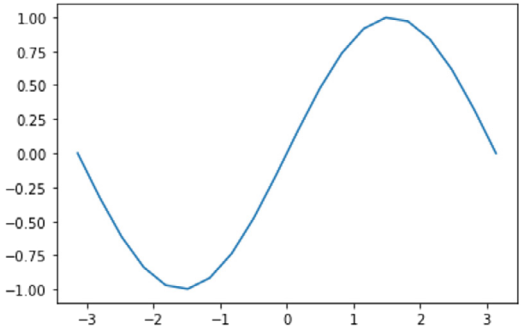

```
plt.plot([3, 5, 2])  # x ekseni verilmez ise varsayılan olarak 0-1-2... gelir
```

```
[<matplotlib.lines.Line2D at 0x23aee955198>]
```




```
x = np.linspace(-np.pi, np.pi, 20)  # -pi ile +pi arasında 20 adet değer
plt.plot(x, np.sin(x))
```

```
[<matplotlib.lines.Line2D at 0x23aeeb11a58>]
```



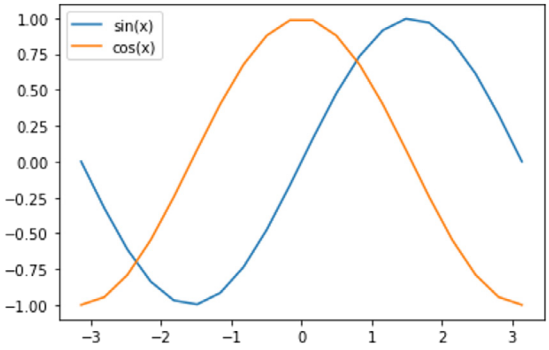
Grafiklere Özellik Ekleme

Az önce çizdirilen grafikler x, y eksenleri ve alınan değerlerin birleştirilmesi ile oluşturulmuştur. Matplotlib kütüphanesinde eğer bir özellik tanımı yapılmaz ise grafikler yalın hâlde gelir. Ancak grafikler yalın hâlde anlamsızdır. x, y eksenlerinin neyi gösterdiği, grafiğin başlığının ne olduğu, verilerin neler olduğu grafik üzerinde gösterilmelidir.

label özelliği verilerin neler olduğunu gösterir. label özelliğinin grafik içerisinde gösterilmesi için **`plt.legend()`** fonksiyonu eklenmiştir.

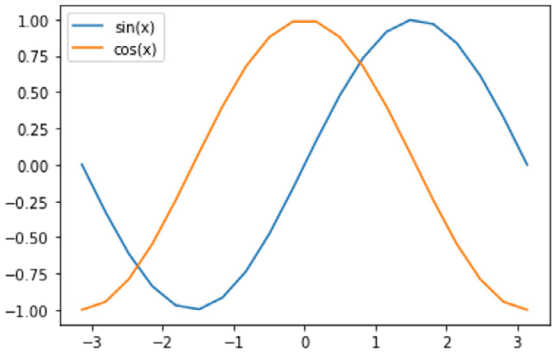
```
plt.plot(x,np.sin(x), label = 'sin(x)')
plt.plot(x,np.cos(x), label = 'cos(x)')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x23aee9f6b70>
```

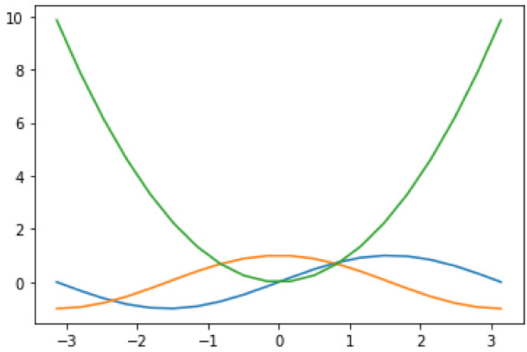


`legend()` fonksiyonu içerisine tanımlanacak parametreler ile etiketlerin konumu değiştirilebilir.

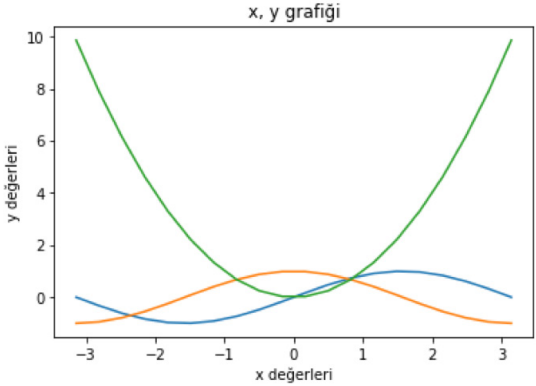
```
plt.plot(x,np.sin(x), label = 'sin(x)')
plt.plot(x,np.cos(x), label = 'cos(x)')
plt.legend(loc='best'); # upper left upper right
```




```
# Ya da aynı figür üzerinde şekil çizmek istersek
plt.plot(x, np.sin(x), x, np.cos(x), x, x**2);  # ; fonksiyon yazmak istemezse
```

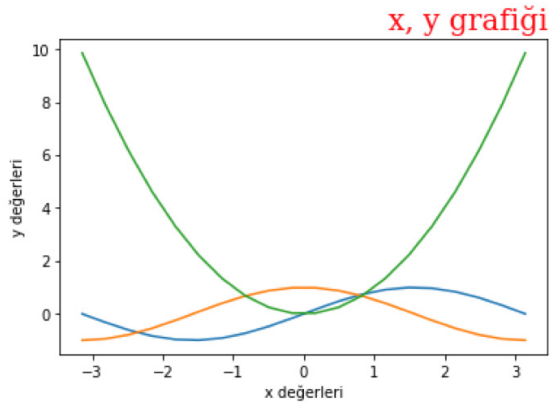



```
# xlabel, ylabel ve title
plt.plot(x, np.sin(x), x, np.cos(x), x, x**2)
plt.xlabel("x değerleri")
plt.ylabel("y değerleri")
plt.title("x, y grafiği");
```



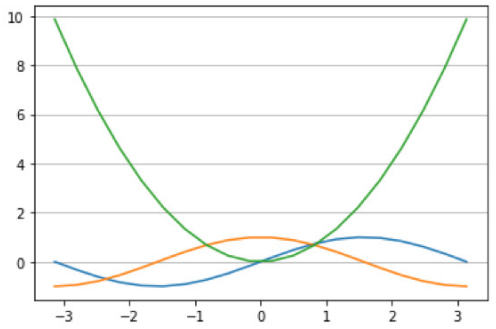

```
# Etiketlerin fontlarını değiştirme
plt.plot(x, np.sin(x), x, np.cos(x), x, x**2)
plt.xlabel("x değerleri")
plt.ylabel("y değerleri")
plt.title("x, y grafiği",
          fontdict={
              'family': 'serif',
              'color': 'red',
              'size': 20
          },
          loc='right')
```

```
Text(1.0, 1.0, 'x, y grafiği')
```

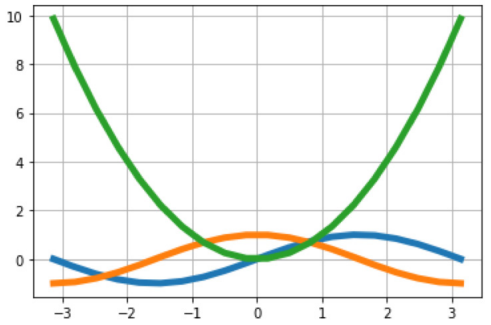


Fontlarla ilgili daha detaylı bilgi şu adresten alınabilir: *https://matplotlib.org/stable/gallery/text_labels_and_annotations/fonts_demo_kw.html*

```
# grafik içerisinde ızgaraları gösterme
plt.plot(x, np.sin(x), x, np.cos(x), x, x**2)
plt.grid(True, axis='y') # axis='x' sadece x ekseninde ızgaralar gösterir
```




```
# çizgilerin kalınlıklarının ayarlanması
plt.plot(x, np.sin(x), x, np.cos(x), x, x**2, linewidth=5);
plt.grid()
```




```
# çizgi stilleri
plt.plot(x, np.sin(x), linestyle='dashed', label='sin', marker = 'x')
plt.plot(x, np.cos(x), linestyle='--', label='cos', marker= 'p')
plt.plot(x, x**2, label='kare', linestyle='-.')
plt.grid()
```

```
plt.legend()
```

```
<matplotlib.legend.Legend at 0x23aeffd4748>
```

Matplotlib kütüphanesindeki grafiklerin tüm özelliklerini değiştirmek mümkündür. Diğer parametreler için şu adresten bilgi alınabilir: *https://matplotlib.org/devdocs/api/_as_gen/matplotlib.pyplot.plot.html*

Çoklu Figürler Çoklu Matplotlib Grafikleri

Bazı durumlarda birden fazla grafik bir figür içerisinde verilmek istenebilir. Böylece tek seferde okuyucuya birden fazla durumla ilgili özet bilgi sunulmuş olunur. Daha önceden belirtilen **Figure** ve **Axes** kavramları kullanılarak Matplotlib içerisinde birden fazla grafik **`subplot()`** fonksiyonu ile yerine getirilebilir. **`Subplot()`** fonksiyonu ile istenilen sayıda satır ve sütunda grafikler figürlere eklenebilir.

Subplot fonksiyonundan önce matplotlib kütüphanesindeki iki farklı yaklaşımdan bahsetmekte fayda vardır.

Matlab Tipi Pyplot ile Grafik Çizdirme

Pyplot ile çizdirilen grafik bir hücre için değiştirilebilir bir grafiktir. Başka hücrede yeni bir pyplot objesi eklenirse yeni grafikler çizdirilebilir. MATLAB programındakine benzer şekilde grafik çizdirme yöntemidir. Daha önce gösterilen grafikler Pyplot ile çizdirilen grafiklerdir.

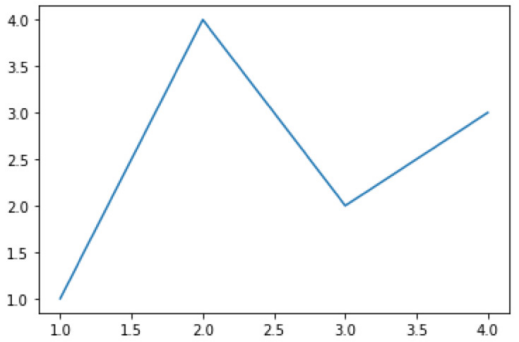
Nesne Yönelimli Şekilde Grafik Çizdirme

Matplotlib kütüphanesinin önemli özelliklerinden birisi de nesne yönelimli programlamaya izin vermesidir. Çizim alanı anlamına gelen **Figure** objesi Matplotlib kütüphanesinin tepesinde yer alır. Diğer elemanlar ise **Artist** diye bilinir ve aşağıdaki görselde gösterilen mavi yerler Artist olarak geçmektedir. Artistler **Axes**'ten (Eksenler) oluşur. Eksenler ise **Axis** öğelerinden oluşur. Eksenler (Axes) grafikleri tutan objelerdir. Bir grafik çizdirebilmek için en az bir eksenin figür içerisine eklenmesi gerekmektedir. Axis'ler ise x ve y aksisi olabilir ya da üçüncü boyut varsa z axisi de devreye girer. Bu nedenle x veya y'deki limitleri veya çizgileri ayarlamak için axis özelliğini kullanmak gerekecektir. Matplotlib ile daha gelişmiş ve pratik grafik çizimleri için bu hiyerarşiyi anlamak faydalı olacaktır.

Ayrıca nesne yönelimli yapı kullanılarak çok daha hızlı ve temiz grafikler çizilebilir. Hızlı grafik çizimleri için MATLAB stili grafik çizdirmek önerilse de daha detaylı işler için nesne yönelimli yapı önerilmektedir.

```
# basit bir matplotlib grafiği
fig, ax = plt.subplots()  # Tek eksenden oluşan bir figür oluşturmak için
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])  # Eksen içerisine bir grafik çizdirmek için
```

```
[<matplotlib.lines.Line2D at 0x23aeff0a780>]
```




```
print(type(fig))
```

```
<class 'matplotlib.figure.Figure'>
```

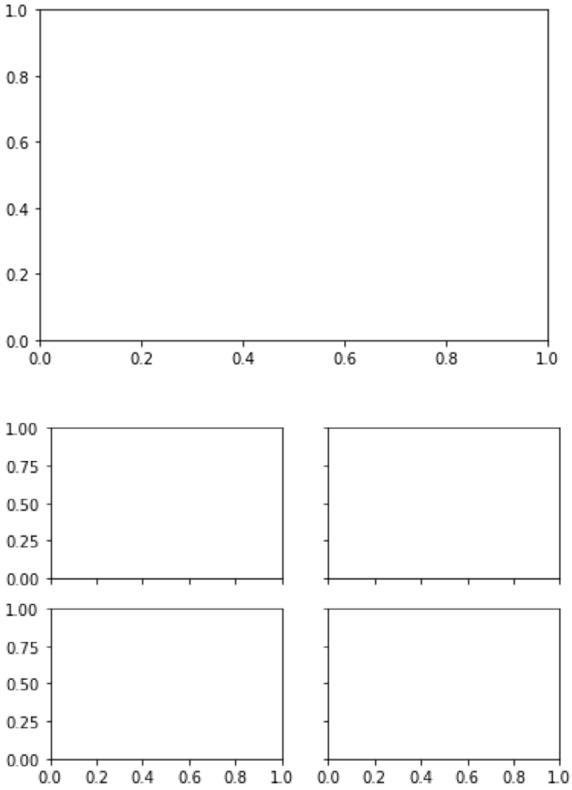
```
help(fig)
```

```
Help on Figure in module matplotlib.figure object:
```

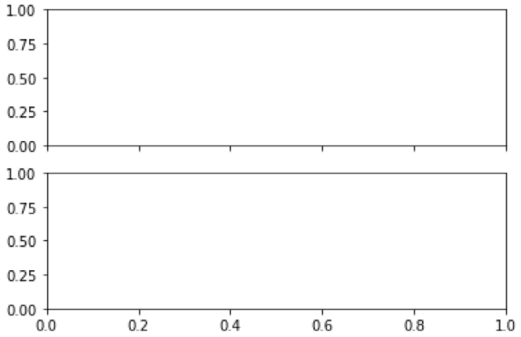
```
class Figure(matplotlib.artist.Artist)
 |  Figure(figsize=None, dpi=None, facecolor=None, edgecolor=None,
linewidth=0.0, frameon=None, subplotpars=None, tight_layout=None,
constrained_layout=None)
 |
 |  The top level container for all the plot elements.
 |
 |  The Figure instance supports callbacks through a *callbacks* attribute
 |  which is a `.CallbackRegistry` instance.  The events you can connect
to
 |  are 'dpi_changed', and the callback will be called with ``func(fig)``
where
 |  fig is the `Figure` instance.
```

```
 |
 |  Attributes
 |  ----------
fig = plt.figure()  # Eksenin olmadığı bir figür
fig, ax = plt.subplots()  # Tek eksenden oluşan bir figür
fig, axs = plt.subplots(2, 2, sharex= True, sharey=True)  # 2x2 li bir grafik
```

<Figure size 432x288 with 0 Axes>

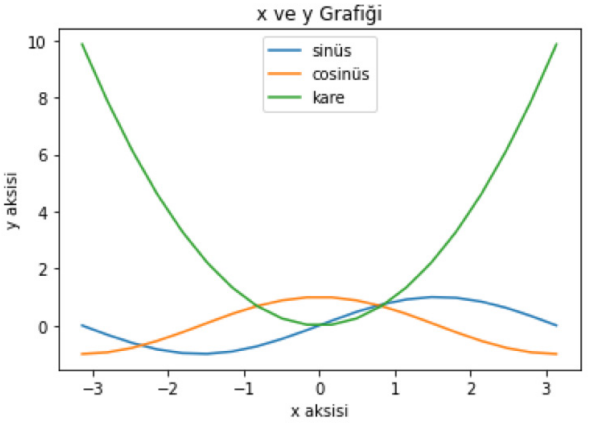



```
fig, axs = plt.subplots(2, 1, sharex=True)  # 2x1 li bir grafik ve ortak x axis i
```




```
# Nesne yönelimli grafikler oluşturmak için aşağıdaki yapı kullanılabilir.
fig, ax = plt.subplots()  # Bir eksenden oluşan bir figür oluşturduk.
ax.plot(x, np.sin(x), label='sinüs')  # Eksenin içerisine grafik çizdirmek için
ax.plot(x, np.cos(x), label='cosinüs')  # Aynı eksene başka grafik çizdirmek için
ax.plot(x, x**2, label='kare')  # istenilen kadar grafik aynı eksene eklenebilir.
ax.set_xlabel('x aksisi')  # Eksene x etiketi koymak için
ax.set_ylabel('y aksisi')  # set_ylabel.
ax.set_title("x ve y Grafiği")  # Grafiğe başlık eklemek için.
ax.legend()  # Etiketleri göstermek için.
```

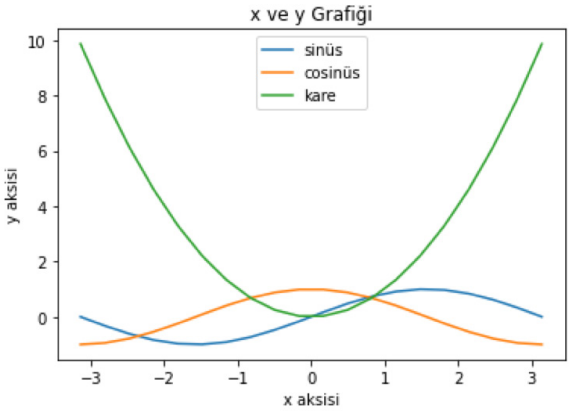
```
<matplotlib.legend.Legend at 0x23aeede28d0>
```




```
# ya da aynı işlemi daha önce gösterdiğimiz gibi de yapabilirdik.

plt.plot(x, np.sin(x), label='sinüs')  # Eksenin içerisine grafik çizdirmek için
```

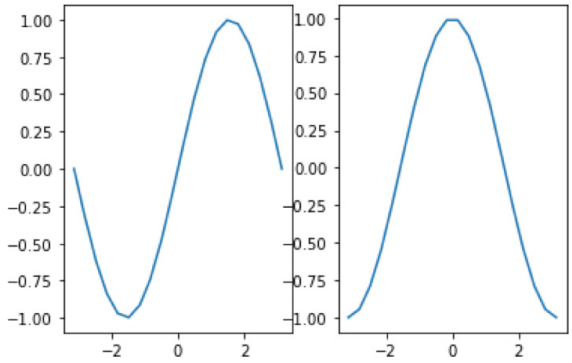
```
plt.plot(x, np.cos(x), label='cosinüs')  # Aynı eksene başka grafik çizdirmek için
plt.plot(x, x**2, label='kare')  # istenilen kadar grafik aynı eksene eklenebilir.
plt.xlabel('x aksisi')  # set_xlabel değişti
plt.ylabel('y aksisi')  # set_ylabel değişti.
plt.title("x ve y Grafiği")  # set_title değişti.
plt.legend()  # Etiketleri göstermek için.
```

```
<matplotlib.legend.Legend at 0x23af06fa898>
```



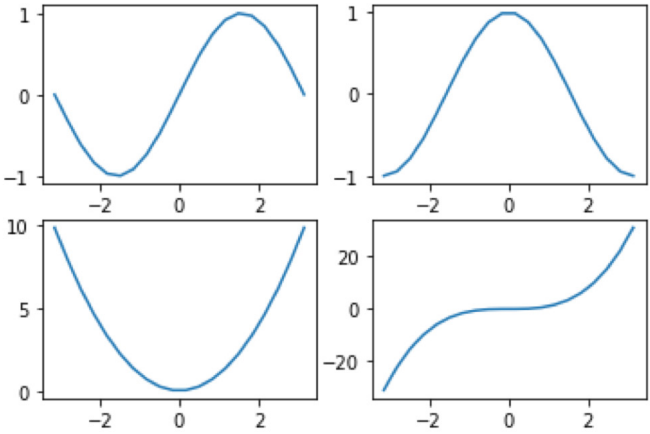

```
fig, ax = plt.subplots(1, 2) # bir satır iki sütundan oluşan iki grafik ekledik.
ax[0].plot(x, np.sin(x)) # ilk eksene grafik çizdirmek için
ax[1].plot(x, np.cos(x)) # ikinci eksene grafik çizdirdik.
```

```
[<matplotlib.lines.Line2D at 0x23af060b550>]
```



```
fig, ax = plt.subplots(2, 2) # iki satır iki sütundan oluşan iki grafik ekledik.
ax[0,0].plot(x, np.sin(x), label='sin') # birinci satır birinci sütundaki grafik
ax[0][1].plot(x, np.cos(x)) # ikinci eksene grafik çizdirdik.
ax[1][0].plot(x, x**2)
ax[1][1].plot(x, x**3)
```

```
[<matplotlib.lines.Line2D at 0x23af09c14a8>]
```




```
type(ax) # ax nesnesi bir NumPy dizisidir. İçerisinde Matplotlib nesneleri bulunmaktadır
```

```
numpy.ndarray
```

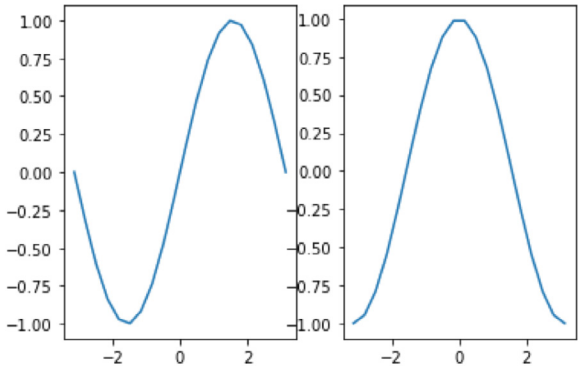
```
type(ax[0][0])
```

```
matplotlib.axes._subplots.AxesSubplot
```

MATLAB stili pyplot grafik çizdirmede **birden fazla eksende grafik** çizdirilmek istenirse aşağıdaki yapı kullanılabilir.

```
plt.subplot(121) # Bir satır iki sütunlu bir yapıda birinci grafiği çizdir.
plt.plot(x, np.sin(x))
plt.subplot(122) # İkinci grafik
plt.plot(x, np.cos(x))
```

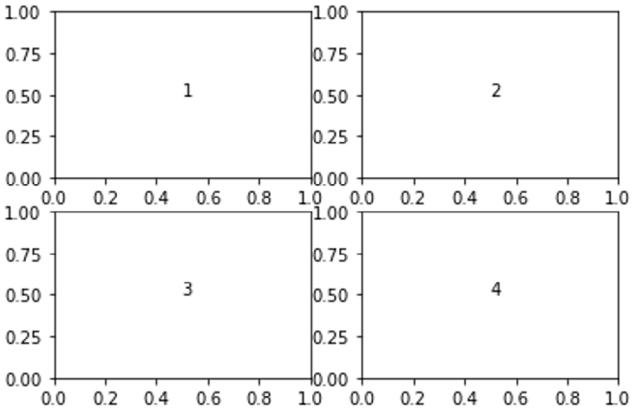
```
[<matplotlib.lines.Line2D at 0x23af0a504e0>]
```



Izgaralar ile Subplot Çizdirme

Yaygın kullanım için benzer ölçülerde eksenler ile çalışmak her ne kadar yeterli olsa da Matplotlib istenilen boyutlarda eksenler oluşturmak için imkan sunmaktadır. **`plt.subplot()`** fonksiyonu satır, sütun ve indeks değerlerinden oluşan 3 tam sayı değeri almaktaydı. Tam sayı değerleri eksenin kapsayacağı ızgara dilimi ile ilgili bilgiyi vermemektedir.

```
for i in range(1,5):
    plt.subplot(2,2,i) # 2 satır 2 sütun eksenlerin i. ekseni
    plt.text(0.5, 0.5, str(i))
```



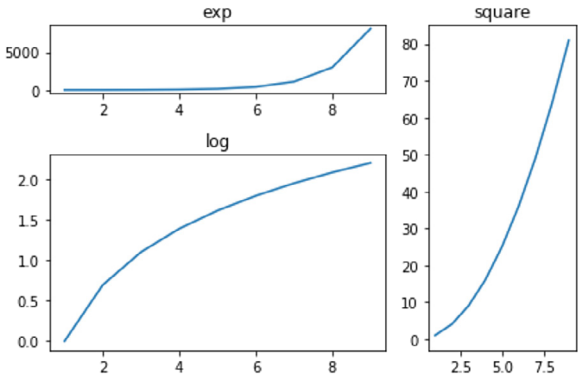
Birden fazla subplot çizdirmek için **`plt.subplots()`** fonksiyonu kullanılır. Bu durumda ızgara içerisindeki tüm konumlar kullanılabilir hâle gelir. Yukarıdaki örneği ortak x ve y için yapmak istersek aşağıdaki yapıyı kullanabiliriz.

plt.subplot2grid() Fonksiyonu

Izgaraların içerisine istenilen boyutlarda grafikler eklemek için **subplot2grid()** fonksiyonu kullanılabilir. Bu fonksiyon sırasıyla **shape, location, rowspan, colspan** parametrelerini alır.

- **shape:** satır ve sütun değerleri alır. Birinci girdi satır sayısını ikinci girdi sütun sayısını gösterir.
- **loc:** 2 tam sayı alır. Izgara içerisinde nereye yerleştirileceğini belirler.
- **rowspan:** Sağ tarafa doğru ne kadar genişleyeceğini belirtir.
- **colspan:** Aşağı doğru ne kadar genişleyeceğini belirler.

```
a1 = plt.subplot2grid((3,3),(0,0),colspan = 2) # 3x3 lük bir ızgarada 1. satır birinci sütunda iki sütun git.
a2 = plt.subplot2grid((3,3),(0,2), rowspan = 3)
a3 = plt.subplot2grid((3,3),(1,0),rowspan = 2, colspan = 2)
x = np.arange(1,10)
a2.plot(x, x*x)
a2.set_title('square')
a1.plot(x, np.exp(x))
a1.set_title('exp')
a3.plot(x, np.log(x))
a3.set_title('log')
plt.tight_layout()
plt.show()
```



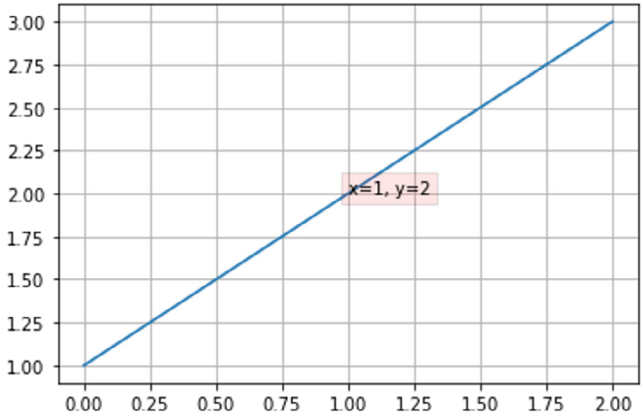
Grafiklere Farklı Elemanlar Ekleme

Grafikler içerisinde işaretleyici, metin, şekil gibi elemanlar eklenmek istenirse aşağıdaki yapılar kullanılabilir.

Metin Ekleme

Daha önceden gösterilen **`xlabel(), ylabel(), title()`** gibi fonksiyonlar figür içerisine metin eklemekte kullanılmıştı. Bu fonksiyonlara benzer şekilde **`text()`** fonksiyonu istenilen bir koordinata metin eklenmesini sağlar.

```
plt.plot([1,2,3])
plt.text(1,2,"x=1, y=2", bbox={'facecolor': 'red', 'alpha': 0.1})
plt.grid() # Konumları daha rahat görebilmek için
```

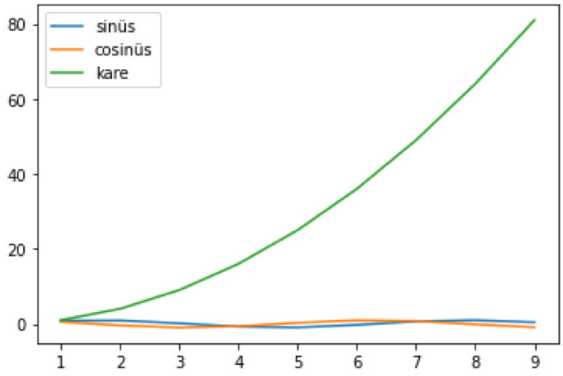


Legend() Fonksiyonu

Daha önce kullanılan bu fonksiyon ile daha önceden etiket verilmemiş olan grafiklere sırası ile etiket verilebilir. Konumu ile ilgili **`loc`** parametresine (best, upper right, upper left, right, center, lower center) gibi değerler verilebilir.

```
plt.plot(x, np.sin(x))
plt.plot(x, np.cos(x))
plt.plot(x, x**2)
plt.legend(['sinüs', 'cosinüs', 'kare'], loc = 'upper left')
```

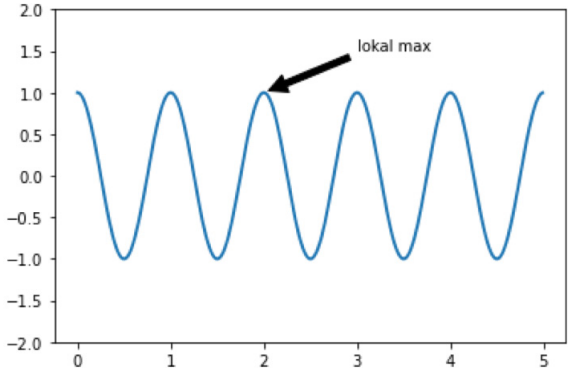
```
<matplotlib.legend.Legend at 0x23af1daf4e0>
```



Annotate Fonksiyonu

Grafik içerisinde bir noktaya dikkat çekmek için kullanılır.

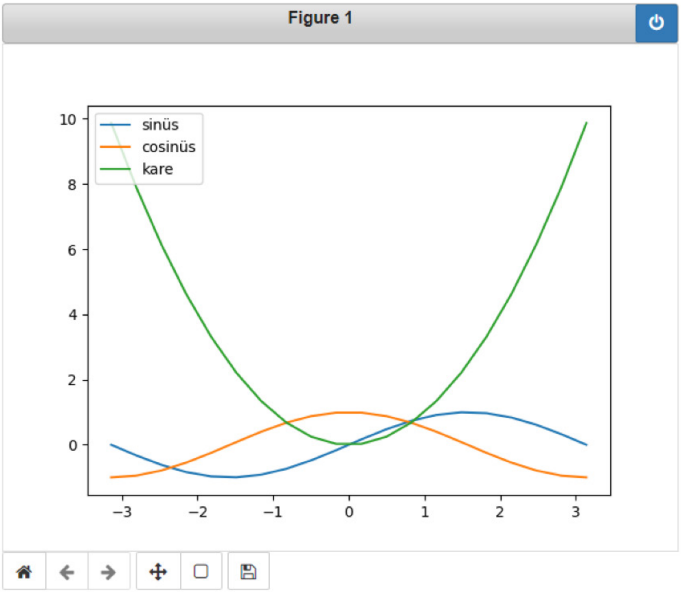
```
fig = plt.figure()
ax = fig.add_subplot(111)

t = np.arange(0.0, 5.0, 0.01)
s = np.cos(2*np.pi*t)
line, = ax.plot(t, s, lw=2)

ax.annotate('lokal max', xy=(2, 1), xytext=(3, 1.5),
            arrowprops=dict(facecolor='black', shrink=0.05),
            )

ax.set_ylim(-2,2)
plt.show()
```



Bazı Sihirli Fonksiyonlar

Jupyter notebooklarda kullanılabilecek bir takım **sihirli fonksiyonlar (magic functions)** ile kolay bir kullanım sağlanabilir. Matplotlib kütüphanesinde kullanılabilecek sihirli fonksiyonlardan bazıları şu şekildedir.

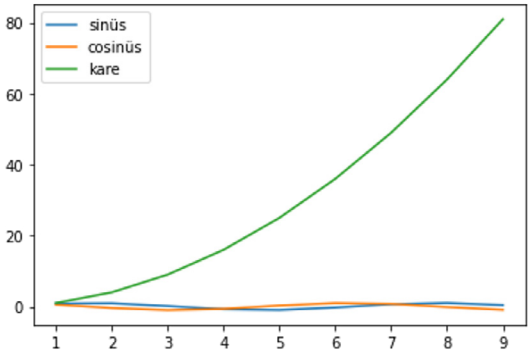
- **`%matplotlib inline`** sihirli komutu ile görsellerin notebook içerisinde gösterilmesi sağlanır.
- **`%matplotlib notebook`** interaktif grafikler için kullanılır.
- **`%matplotlib qt`** interaktif kullanımlar içindir.




```
%matplotlib inline
```

```
plt.plot(x, np.sin(x))
plt.plot(x, np.cos(x))
plt.plot(x, x**2)
plt.legend(['sinüs', 'cosinüs', 'kare'], loc = 'upper left')
```

```
<matplotlib.legend.Legend at 0x23af23f94e0>
```




```
# Diğer sihirli fonksiyonları listelemek için
%lsmagic
```

```
Available line magics:
%alias %alias_magic %autocall %automagic %autosave %bookmark %cd %clear %cls %colors
%config %connect_info %copy %ddir %debug %dhist %dirs %doctest_mode %echo %ed
%edit %env %gui %hist %history %killbgscripts %ldir %less %load %load_ext %loadpy
%logoff %logon %logstart %logstate %logstop %ls %lsmagic %macro %magic %matplotlib
%mkdir %more %notebook %page %pastebin %pdb %pdef %pdoc %pfile %pinfo %pinfo2
%popd %pprint %precision %profile %prun %psearch %psource %pushd %pwd %pycat
%pylab %qtconsole %quickref %recall %rehashx %reload_ext %ren %rep %rerun %reset
%reset_selective %rmdir %run %save %sc %set_env %store %sx %system %tb %time %timeit
%unalias %unload_ext %who %who_ls %whos %xdel %xmode

Available cell magics:
%%! %%HTML %%SVG %%bash %%capture %%cmd %%debug %%file %%html %%javascript
%%js %%latex %%perl %%prun %%pypy %%python %%python2 %%python3 %%ruby
%%script %%sh %%svg %%sx %%system %%time %%timeit %%writefile

Automagic is ON, % prefix IS NOT needed for line magics.
```

```
# Örneğin whos sihirli fonksiyonu kullanılan değişkenleri listeler
%whos
```

```
Variable        Type        Data/Info
------------------------------------------------------------
json            module      <module 'json' from 'D:\\<...>\lib\\json\\__init__.py'>
np              module      <module 'numpy' from 'D:\<...>ges\\numpy\\__init__.py'>
pd              module      <module 'pandas' from 'D:<...>es\\pandas\\__init__.py'>
plt             module      <module 'matplotlib.pyplo<...>\\matplotlib\\pyplot.py'>
x               ndarray     20: 20 elems, type `float64`, 160 bytes
yapf_reformat   function    <function yapf_reformat at 0x00000191089AC7B8>
```

Grafiklerin Kaydedilmesi

Eğer çizilen grafikler eğitim amaçlı veya Jupyter notebook içerisinde gösterim amaçlı kullanılacaksa Jupyter notebook içerisinde bırakılabilir. Ancak grafikler, başka bir yerde; örneğin yayında, raporda, dokümanda kullanılacaksa **dışarı aktarılması** gerekecektir. Böylece çözünürlüğü yüksek grafikler üretilerek dokümanlarda kullanılabilir. Eğer çizilen grafiklerin kodları da saklanmak isteniyorsa **%save** sihirli fonksiyonu kullanılabilir.

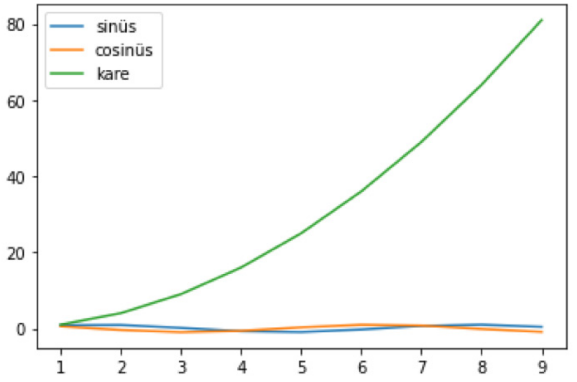
Daha fazla sihirli fonksiyonlar için şu adres ziyaret edilebilir: *https://ipython.readthedocs.io/en/stable/interactive/magics.html*

```
%save?
Docstring:
Save a set of lines or a macro to a given filename.

Usage:
  %save [options] filename n1-n2 n3-n4 ... n5 .. n6 ...

Options:

  -r: use 'raw' input.  By default, the 'processed' history is used,
  so that magics are loaded in their transformed version to valid
  Python.  If this option is given, the raw input as typed as the
  command line is used instead.

  -f: force overwrite.  If file exists, %save will prompt for overwrite
  unless -f is given.

  -a: append to the file instead of overwriting it.

This function uses the same syntax as %history for input ranges,
then saves the lines to the filename you specify.

It adds a '.py' extension to the file if you don't do so yourself, and
it asks for confirmation before overwriting existing files.

If `-r` option is used, the default extension is `.ipy`.
File:      d:\miniconda3\lib\site-packages\ipython\core\magics\code.py
```

```
% save sin_cos_grafigi 63 # 115. çalıştırılan hücredeki kodları kaydetmek için,
plt.plot(x, np.sin(x))
plt.plot(x, np.cos(x))
plt.plot(x, x**2)
plt.legend(['sinüs', 'cosinüs', 'kare'], loc = 'upper left')
```

File `sin_cos_grafigi.py` exists. Overwrite (y/[N])? y The following commands were written to file `sin_cos_grafigi.py`: get_ipython().magic('save sin_cos_grafigi 115 # 115. çalıştırılan hücredeki kodları kaydetmek için') plt.plot(x, np.sin(x)) plt.plot(x, np.cos(x)) plt.plot(x, x**2) plt.legend(['sinüs', 'cosinüs', 'kare'], loc = 'upper left')

<matplotlib.legend.Legend at 0x23af24e4c50>




```
% load sin_cos_grafigi.py # yazılan kodların geri çağrılması için
```

savefig() Fonksiyonu

Matplotlib kütüphanesinde grafikleri resim olarak kaydetmek, grafikleri göstermek kadar kolaydır. Grafikleri **jpeg, png** gibi formatlarda kaydetmek için `savefig()` fonksiyonu kullanılabilir. `savefig()` fonksiyonu içerisine parametre olarak grafiğin yolu ve ismi verilmelidir, bu parametre fname olarak verilir. fname dışında kullanılabilecek diğer bazı parametreler aşağıda verilmiştir.

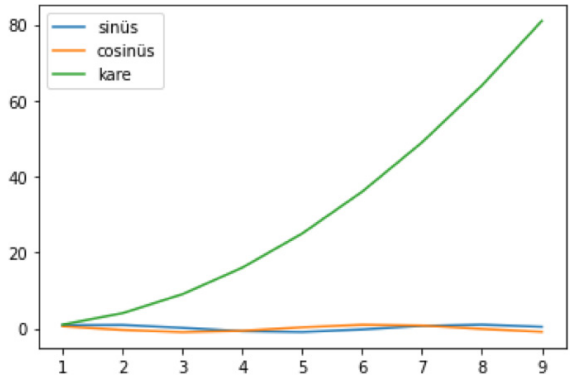
- **dpi=None:** Görüntünün çözünürlüğü, örneğin 300 dpi (inç başına düşen nokta sayısı)

Şekil 6-3: Farklı dpi değerleri

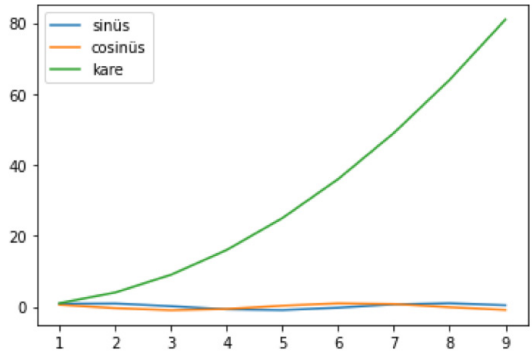
- **transparent=False:** `True` değeri verildiğinde grafiğin arka planı transparan hâle getirilir.
- **bbox_inches=None**: Grafik çevresindeki sınırlar ile ilgilidir. **`bbox_inches='tight'`** genellikle ideal sonuçlar verir.

PNG formatı internet kullanımları ve çizim grafikleri için daha uygundur. JPG formatı ise daha sıkıştırılmış bir resim formatıdır. Bu nedenle daha küçük boyutlarda görüntülerin kaydedilmesini sağlar. Eğer daha küçük boyutlu resimler ile çalışılmak isteniyorsa JPG formatı tercih edilebilir. SVG formatı ise vektörel görüntülerde kullanılan ve diğer formatlara göre boyutu yüksek bir seçimdir.

```
plt.plot(x, np.sin(x))
plt.plot(x, np.cos(x))
plt.plot(x, x**2)
plt.legend(['sinüs', 'cosinüs', 'kare'], loc = 'upper left')
plt.savefig('sin_cos_grafigi.png', dpi=300)
```



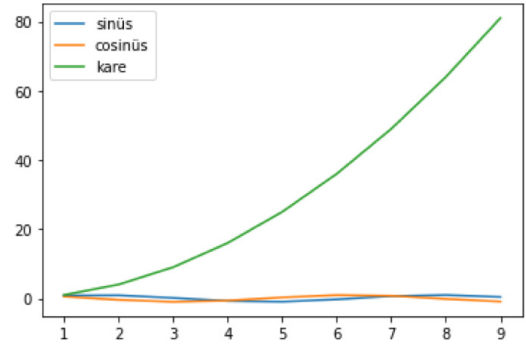

```
plt.plot(x, np.sin(x))
plt.plot(x, np.cos(x))
plt.plot(x, x**2)
plt.legend(['sinüs', 'cosinüs', 'kare'], loc = 'upper left')
plt.savefig('sin_cos_grafigi_transparent.png', dpi=300, transparent=True)
```



Renk kodları ile hızlı bir şekilde grafikteki renkler değiştirilebilir. Renk kodları aşağıda paylaşılmıştır.

- **r(red)**
- **g(green)**
- **b(blue)**
- **c(cyan)**
- **k(black)**
- **w(white)**

```
    plt.plot(x, np.sin(x))
    plt.plot(x, np.cos(x))
    plt.plot(x, x**2)
    plt.legend(['sinüs', 'cosinüs', 'kare'], loc = 'upper left')
    plt.savefig('sin_cos_grafigi_transparent.png', dpi=300,
transparent=True, facecolor='b') # ön renk mavi
```



Grafik Çeşitleri

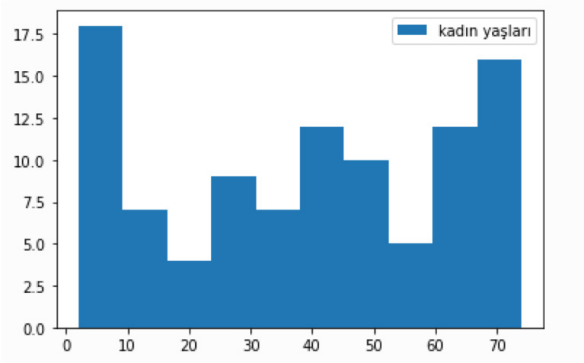
Matplotlib içerisinde hemen hemen bilindik tüm grafik çeşidi çizdirilebilir. Burada grafik çeşitlerine örnek olarak aşağıdaki çeşitler gösterilecektir.

Histogram

x aksis'indeki verilerin **dikdörtgen** şeklinde çizdirildiği grafik çeşididir. Bu nedenle kesikli verilerle kullanılması gerekmektedir. Eğer sürekli veriler söz konusu ise sürekli veriler **bins** denilen çerçevelere ayrılır. `hist()` fonksiyonu ile çizdirilebilir. `hist()` fonksiyonu `n, bins, patches` gibi parametreler alır. Yani histogram ile gruplu verilerin grafiği çizilebilir. Örneğin kadın ve erkekler ilişkin yaşlar histogram grafiği olarak aşağıdaki gibi çizdirilebilir.

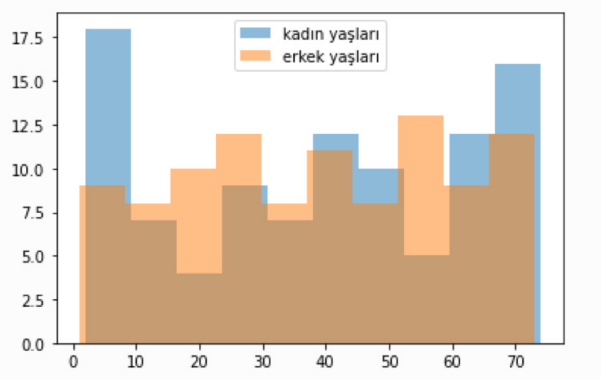
```
kadin_yaslar = np.random.randint(low=1,high=75,size=100)
plt.hist(kadin_yaslar,label='kadın yaşları')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1910c5141d0>
```



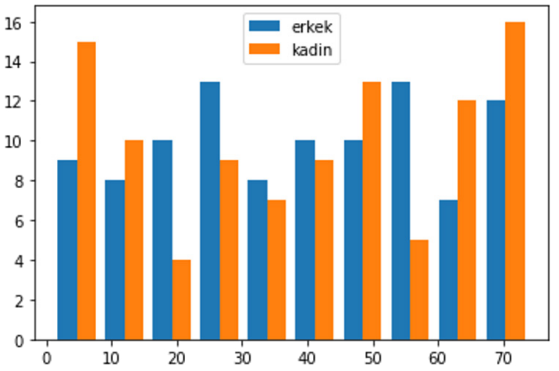

```
erkek_yaslar = np.random.randint(low=1,high=75,size=100)
plt.hist(kadin_yaslar,label='kadın yaşları', alpha=0.5)
plt.hist(erkek_yaslar,label='erkek yaşları', alpha=0.5)
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1910c8f7f98>
```



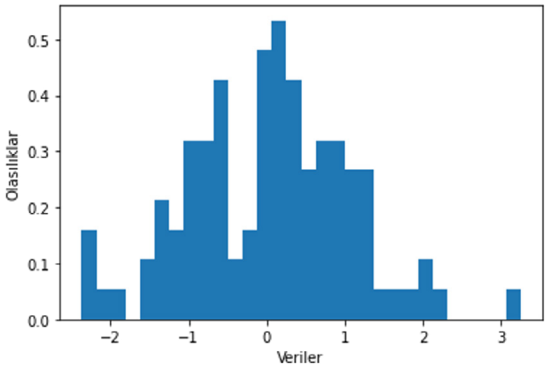

```
plt.hist([erkek_yaslar, kadin_yaslar], label=['erkek', 'kadin'])
plt.legend()
```

<matplotlib.legend.Legend at 0x1910c46f9b0>

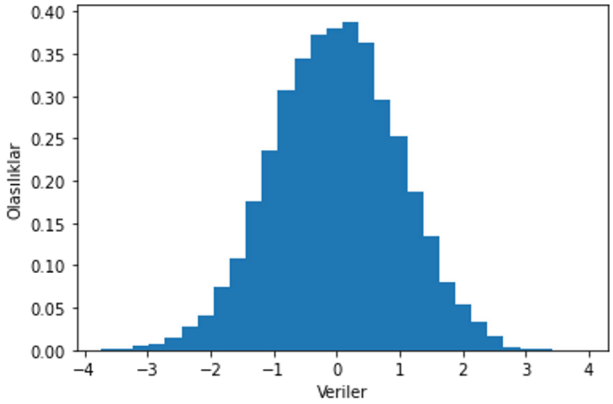


Aynı şekilde histogram grafikleri olasılık yoğunluk fonksiyonlarının çizdirilmesinde de kullanılabilir. Örneğin normal dağılıma uyan bir veri dizisinin grafiği aşağıdaki çizdirilebilir.

```
np.random.seed(12345)
x_normal = np.random.normal(size=100) # normal dağılıma uyan 100 veri
plt.hist(x_normal, density=True, bins=30)  # `density=False` sayıları alacaktı. Oranları aldık
plt.ylabel('Olasılıklar')
plt.xlabel('Veriler');
```




```
np.random.seed(12345)
x_normal = np.random.normal(size=10000) # normal dağılıma uyan 10000 veri
plt.hist(x_normal, density=True, bins=30)  # `density=False` sayıları  alacaktı.
Oranları aldık
plt.ylabel('Olasılıklar')
plt.xlabel('Veriler');
```

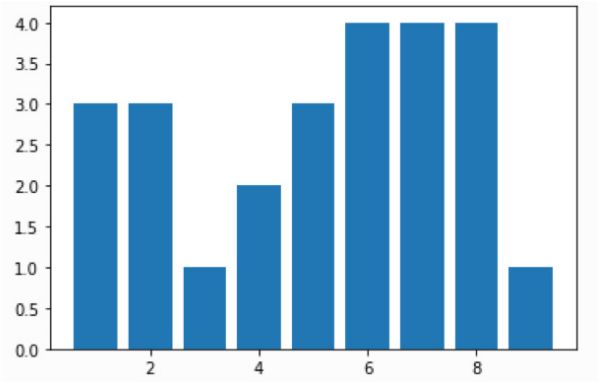


Çubuk Grafikleri

Yaygın bir kullanıma sahiptir. Histogram grafiklerinden farklı yanı x aksisindeki değerlerin kategorik değil, **nümerik** değerler almasıdır. **`bar()`** fonksiyonu ile çizim yapılabilir. Diğer grafiklerdeki benzer parametreler burada da geçerlidir. Örneğin **`align`** parametresi ile verinin çubuğun ortasından geçmesi istendiği durumda kullanılabilir. Ya da **`orientation`** parametresi ile çubuklar dikey eksende mi olsun yatay eksende mi, bununla ilgili ayarlama için kullanılabilir. Genellikle ölçüm değerinin nasıl değiştiği görülmek için kullanılır. x değerlerine karşılık gelen y değerlerine kadar bir çubuk çizilir.

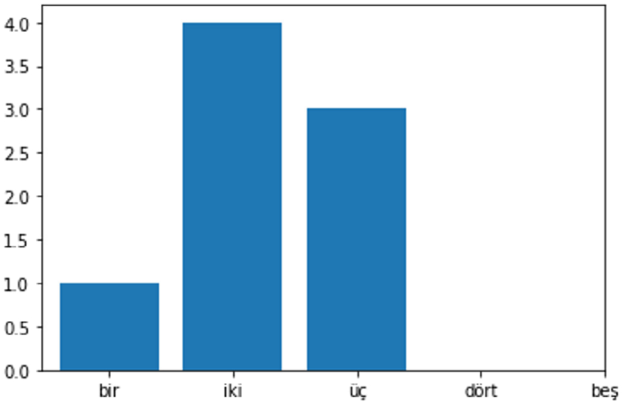
```
x1_koordinatlari = np.random.randint(10, size=20)
y1_koordinatlari = np.random.randint(5,size=20)

x2_koordinatlari = np.random.randint(10, size=20)
y2_koordinatlari = np.random.randint(5,size=20)

plt.bar(x1_koordinatlari,y1_koordinatlari,label='grup 1')
```

```
<BarContainer object of 20 artists>
```



Bunun yanı sıra çubuk grafikleri kategorik veriler için de kullanılabilir. Bunun için;

```
x_bar = np.random.randint(5,size=5)
plt.bar(x_bar, np.arange(5))
plt.xticks(np.arange(5),['bir','iki','üç','dört','beş']) # kategoriler ayrıca verilebilir.
```



Diğer grafik çeşitleri şu şekilde verilebilir:

- pie pasta grafikleri
- scatter dağılım grafikleri
- boxplot kutu grafikleri

Stil Dosyaları ile Çalışmak

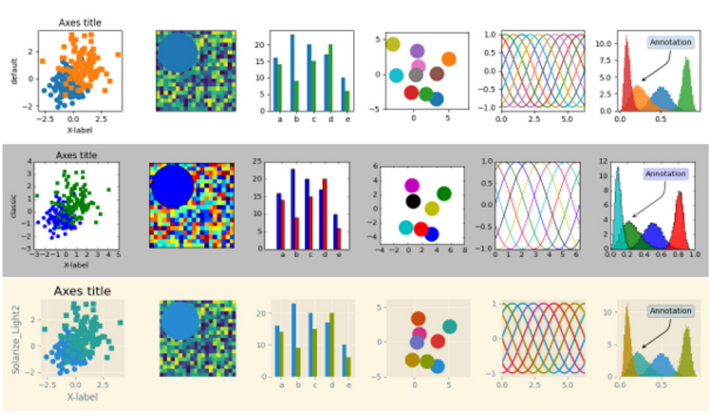
Matplotlib kütüphanesinde bir dokümanda düzenli bir şekilde renk ve görsel seçenekleri kullanmak için stiller kullanılmaktadır. Kütüphane içerisindeki stil dosyaları **`plt.style.available`** komutu ile gösterilebilir. Tercih edilen stil **`plt.style.use()`** komutu ile seçilebilir. Bir dokümanda çizilen grafiklerin renkleri, yazı fontları, font büyüklükleri, ızgara şekilleri gibi tasarım özelliklerinin benzer şekilde olması beklenir.

Diğer stiller ile ilgili bilgi şu adresten alınabilir: *https://matplotlib.org/stable/gallery/style_sheets/style_sheets_reference.html*

```
plt.style.available
```

```
['Solarize_Light2',
 '_classic_test_patch',
 'bmh',
 'classic',
 'dark_background',
 'fast',
 'fivethirtyeight',
 'ggplot',
```

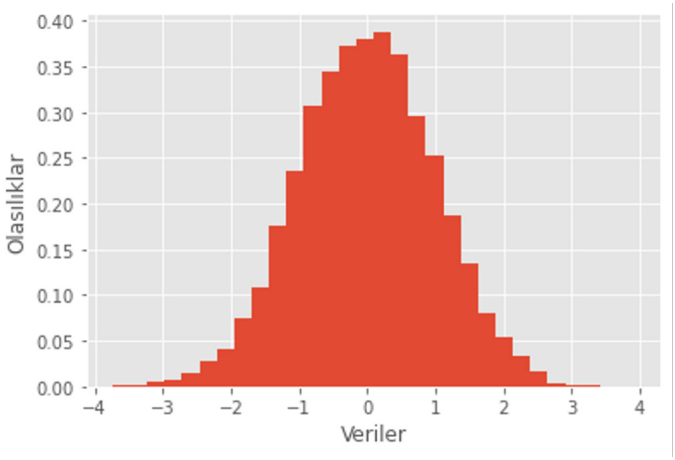
'grayscale',
'seaborn',
'seaborn-bright',
'seaborn-colorblind',
'seaborn-dark',
'seaborn-dark-palette',
'seaborn-darkgrid',
'seaborn-deep',
'seaborn-muted',
'seaborn-notebook',
'seaborn-paper',
'seaborn-pastel',
'seaborn-poster',
'seaborn-talk',
'seaborn-ticks',
'seaborn-white',
'seaborn-whitegrid',
'tableau-colorblind10']



Şekil 6-4: Bazı stil örnekleri

```
plt.style.use('ggplot')
```

```
np.random.seed(12345)
x_normal = np.random.normal(size=10000) # normal dağılıma uyan 10000 veri
plt.hist(x_normal, density=True, bins=30)  # `density=False` sayıları alacaktı. Oranları aldık
plt.ylabel('Olasılıklar')
plt.xlabel('Veriler')
```



Özet

Bu bölümde veri görselleştirmede kullanılan programlardan olan Matplotlib kütüphanesi incelenmiştir. Veri madenciliğinde verilerin görsel şekilde sunulması önem arz etmektedir. Bu nedenle veri görselleştirme araçlarından etkili bir şekilde yararlanmak gereklidir. Bu bölümde anlatılan görselleştirme teknikler ilerleyen bölümlerde kullanılacak olan tekniklerdir.

Kaynaklar

1. “Usage Guide — Matplotlib 3.4.1 documentation”, https://matplotlib.org/stable/tutorials/introductory/usage.html#sphx-glr-tutorials-introductory-usage-py.

BÖLÜM 7

SCIKIT-LEARN VE MAKİNE ÖĞRENMESİ

Bu bölümde genellikle makine öğrenmesi kütüphanesi olarak bilinen **Scikit-Learn** kütüphanesinden bahsedilecektir. Makine öğrenmesi uygulamaları için Pandas, NumPy ve Matplotlib kütüphaneleri birlikte kullanılarak bir model geliştirilebilir. Ancak bu durumda makine öğrenmesi uygulaması için fonksiyonları tekrar tekrar yazmak gereklidir. Scikit-Learn kütüphanesi, makine öğrenmesi süreçlerini içerdiği fonksiyonlar ve matematiksel hesaplamaları sayesinde pratik birtakım yöntemler sunmaktadır. Bu kütüphane veri bilimi süreçlerini kolay bir şekilde yerine getirebilmek için tasarlanmıştır. Bu bölümde öncelikle makine öğrenmesi hakkında teorik bilgi verilecektir. Ardından Scikit-Learn kütüphanesi ile makine öğrenmesi konularına bakılacaktır.

Makine Öğrenmesine Giriş

Makine öğrenmesi ya da yapay öğrenme geçmiş verilerden faydalanarak gelecekteki verilerin **öngörülmesi,** yani geçmiş verilerden öğrenmesi anlamına gelmektedir. Günümüzde verilerin toplanması ve depolanmasından ziyade veriden bilgi üretme çalışmaları daha değerli bir konuma sahiptir. Yani verinin dönüşümü, yapısal olmayan verilerin yapısal formata getirilmesi, verinin kullanılabilir hâle getirilmesi, gizli bilgilerin ortaya çıkarılması çalışmaları ön plana çıkmaktadır. Makine öğrenmesi yapay zekânın uygulama alanlarından birisidir. Makine öğrenmesinin daha spesifik bir alanı ise **derin öğrenme**dir. Derin öğrenmede görüntü işleme, ses tanıma, akıllı robotlar gibi alanlarda çalışmalar gerçekleştirilir. Makine öğrenmesi uygulamalarında ise genellikle veri setlerinden bilgi çıkarılmaya çalışılır.

Makine öğrenmesi uygulaması için e-postaların spam veya spam olmama olarak iki ayrılması örnek olarak verilebilir. Daha önceden gönderilen ve kullanıcılar tarafından spam olarak işaretlenen veya işaretlenmeyen e-postalardan çıkarım yapılarak yeni e-postalar için bir sınıflandırma yapılabilir. Bu tip çalışmalar sınıflandırma veya gözetimli öğrenme çalışmaları olarak bilinmektedir. Sınıflandırma çalışmalarında geçmişteki verilerin etiketleri ya da hedef değişkeninin değerlerinin bilindiği varsayılır. Hedef değişkenin değerleri bilinmiyor ve tahmin edilmeye çalışılıyorsa bu tip çalışmalar da **gözetimsiz öğrenme** çalışmaları olarak bilinmektedir.

Makine Öğrenmesinde Temel Kavramlar

Daha önce veri madenciliğinin kalbinde veri vardır, denilmişti. Makine öğrenmesinin en önemli kavramı ise **öğrenme** tanımıdır. Yapay öğrenmede insanlar ya da hayvanların öğrenme mekanizmaları taklit edilerek makineler üzerinde uygulanmak istenir. Makine öğrenmesindeki esas amaç genelleştirme kabiliyeti olan modeller ortaya koymaktır. Örneğin 100 hastaya ait hastalık verilerinin hastaneye yeni gelecek kişilerde de aynı özelliklerin olması durumunda hasta olup olmadığına bakılabilir. Bu işleme **genelleştirme** işlemi denilir. Hasta veri seti örneğinde olduğu gibi eğer veri seti çıktı değişkenine sahipse bu öğrenme tipine **gözetimli öğrenme** denilir. Gözetimli öğrenmede sınıflandırma çalışmaları gerçekleştirilir. Veri setindeki gözlemlerin her biri girdi değişkenlerine (x) ve çıktı değişkenine (y) sahiptir. Dolayısıyla, sınıflandırma çalışmalarında x noktalarıyla y noktası açıklanmaya çalışılır.

Makine öğrenmesi modellerinde ayrıca çıktı değişkenin sayılabilir ya da sayılamaz olması ile de ilgilenilir. Eğer **sayılabilir** (kantitatif) çıktı değişkeni varsa bu durumda regresyon çalışmaları gerçekleştirilir. Eğer çıktı değişkeni **sayılamaz** (kalitatif) veri tipinde ise bu durumda sınıflandırma çalışmaları gerçekleştirilir. Regresyon çalışmalarında girdi değişkeni kategorik veya nümerik veri tipinde olabilir. Ancak çıktı değişkeninin nümerik veri tipinde olması gereklidir. Çıktı değişkeninin nümerik olması halinde eğer sınıflandırma çalışması yapılmak istenirse veriler kategorik hâle dönüştürülerek işlem yapılabilir. Örneğin çıktı değişkeni 1-60 arasındaki yaşlardan oluşuyorsa 0-20 çocuk, 20-40 genç, 40 ve sonrası orta yaşlı denilerek bir kategorizasyon yapılabilir. Böylece problem sınıflandırma problemi olarak ele alınabilir. Bu bölümde makine öğrenmesi yaklaşımları gösterilecek ve regresyon analizi ile ilgili detaylı inceleme bir sonraki bölümde verilecektir.

Sınıflandırma çalışmaları için kullanılacak veri setini oluşturmak veya elde etmek kolay olmaz. Çünkü bazı durumlarda çıktı değişkeninin bilinmesi maliyetli olabilir. Örneğin bir hastalığa ait veri setinde çıktı değişkeni kişinin hasta/hasta değil durumları olsun. Bu durumda hastalık teşhisi gerçekleştikten sonra veri seti oluşabilecektir. Yani veri setindeki her kişi için hastalık teşhisinin yapılmış olması gereklidir. Bu da hastalık teşhisi için zaman ve para maliyetini getirir. Bu nedenle veri setinde çıktı değişkeninin olmaması durumunda da kullanılabilecek modellere ihtiyaç vardır. Bu modeller **gözetimsiz öğrenme** modelleri olarak bilinir. Gözetimsiz öğrenme modellerinde kullanılan veri setlerine çıktı değişkeni olmadığı için etiketsiz veri setleri de denilebilir. Gözetimsiz öğrenmenin en önemli örneği **kümeleme** çalışmalarıdır. Kümeleme çalışmalarında, daha önce veriler arasındaki uzaklık ve yakınlık ölçüleri ile veri noktaları arasındaki yakınlıklar ortaya koyularak benzer veriler aynı gruplara atanır. Gözetimsiz öğrenme çalışmaları bir keşif çalışmasıdır. Bu keşif çalışmasında veriler arasındaki ilişkiler, yakınlıklar veya uzaklıklar önemli yer tutar.

Diğer bir öğrenme çeşidi ise **yarı gözetimli öğrenme**dir. Bu çeşitte hem gözetimli hem de gözetimsiz öğrenmedeki avantajlardan yararlanılır. Küçük miktarda etiketli, büyük miktarda etiketsiz verilerin olması durumunda yarı gözetimli öğrenme modelleri kullanılabilir. Bunun dışında takviyeli öğrenme çeşidi de yine makine öğrenmesindeki diğer bir öğrenme modelidir. Takviyeli öğrenme modelleri daha çok oyunlarda kullanılan bir modeldir. Bu modelde ödül-ceza sistemleri ile öğrenmelerin daha iyi hâle gelmesi sağlanır.

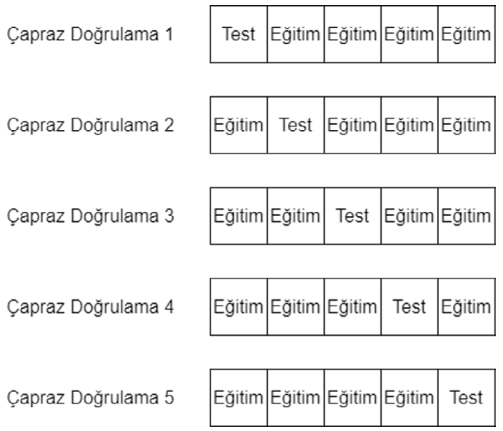
Eğitim ve Test Setleri

Gözetimli öğrenmede kullanılan veri seti öğrenmenin gerçekleştirildiği eğitim seti ve öğrenmenin test edildiği test seti olarak ikiye ayrılmaktadır. Birinci set eğitim seti, burada algoritmanın öğrenme gerçekleştireceği veriler bulunur. Sınıflandırmada özellikler ve hedef değişkenlerin olduğu bir settir. Test veri seti ise yine aynı şekilde gözlem değerlerinden oluşur. Ancak bu set eğitime girmez. Onun yerine eğitilen algoritmanın ne kadar iyi çalıştığı ile ilgili kullanılır. Eğitim ve test setleri birbirinden bağımsız olmalıdır. Yani ortak veri içermemelidir. Çünkü ortak veri olması durumunda eğitim setinin ezberlenmesi durumu söz konusu olabilir. Makine öğrenmesinde karşılaşılan önemli problemlerden birisi modelin ezberlemesidir. İngilizcede ezberleme olayına "**over-fitting**" denilir. Türkçede aşırı öğrenme olarak da isimlendirilebilir. Model eğitim sürecinde ezberler ise performansı yeni gelecek veriler için daha kötü olacaktır. Ezberlemeden kaçınmak için neler yapılacağı ile ilgili detaylı bilgi daha sonra verilecektir.

Bu iki set dışında ayrıca üçüncü bir setten de bahsedilebilir. O da validasyon veya doğrulama setidir. Bu setin kullanım amacı modelin parametrelerini düzenlemektir. Modelin parametrelerine İngilizcede **hyperparameters** denilir. Algoritmanın performansı test seti üzerinden değerlendirilir, performans değerlendirmesi doğrulama seti üzerinden yapılmamalıdır. Genel olarak eğitim, test ve doğrulama setlerinin ayrımı için bir oran vermek doğru değildir. Her model için farklılık arz edebilir. Bir oran söylemek şart ise, eğitim seti için %50-60, test seti için %20-30 ve doğrulama seti için %20-30 aralığında veriler kullanılabilir denilebilir.

Çapraz Doğrulama

Algoritmanın performansının test edilebilmesi için kullanılabilecek bir yöntemdir. Bu yöntemde veri seti belirli sayıda parçaya bölünür. Bölünen parçalardan bir parça test veri seti olarak kullanılır. Diğer parçalar eğitim veri seti olarak kullanılır. Bu sayede tüm parçalar kullanılacak şekilde eğitimin performansı test edilmiş olunur. Parça sayısına İngilizcede "**fold**" denilir. Yani "**10-folds cross validation**" 10 parçalı çapraz doğrulama demektir. Aşağıdaki görselde çapraz doğrulamaya ilişkin bir şekil paylaşılmıştır. Görüldüğü gibi eşit parçalara bölünen veri setinde tüm parçalar test amacıyla kullanılacak şekilde bir doğrulama gerçekleştirilmektedir.

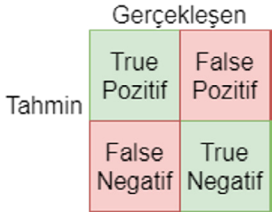


Şekil 7-1: Çapraz doğrulamada kullanılan eğitim ve test setleri

Performans Ölçütleri

Eğitim setinde gerçekleşen öğrenmenin test setinde öğrenme derecesi ile performans ölçütleri hesaplanmaktadır. Ele alınan makine öğrenmesi uygulamasına göre performans ölçütleri farklılık göstermektedir. Örneğin kategorik hedef değişkeninin olduğu sınıflandırma problemlerinde genel olarak kullanılan performans ölçütleri için şu örnekler verilebilir.

Karışıklık matrisi sınıfların doğru tahmin edilip edilmediği hakkında bilgi veren bir matristir. Aşağıdaki görselde gösterildiği gibi **TP** (True Pozitive) doğru tahmin edilen pozitif verilerin sayısını, **FP** (False Positive) yanlış tahmin edilen negatif sayısını, **FN** (False Negative) yanlış tahmin edilen negatif değerlerin, **TN** (True Negative) ise doğru tahmin edilen negatif değerlerin sayısını göstermektedir.



Şekil 7-2: Karışıklık matrisi

Karışıklık matrisi ile hesaplanacak performans ölçütlerinden ilki doğruluk skorudur. İngilizcede **Accuracy Score** olarak geçer. Toplam doğru tahminlerin sayısının toplam sayıya bölünerek hesaplanır. En fazla kullanılan performans ölçüsü olmasına rağmen tek başına kullanılması yeterli olmayabilir. Eğer çıktı değerlerinin sayısı birbirinden çok farklı ise doğruluk skoru iyi sonuçlar vermez. Bu durumda kesinlik, duyarlılık ve F1 skoru ölçütlerine bakılabilir. **Kesinlik** (precision) değerinde pozitif tahmin edilenlerin tüm pozitifler içerisindeki oranı hesaplanır. Kesinlik skoru ile FP yani yanlış tahmin edilen pozitif değerlere dikkat çekilir. Diğer ölçüt ise **duyarlılık** (recall) performans ölçüsüdür. Bu ölçütte ise yanlış tahmin edilen negatif değerlere dikkat çekilir. İki ölçütün bir arada kullanıldığı F1 skoru ise diğer bir performans ölçüsüdür. Performans ölçütlerinin formülasyonu aşağıda verilmiştir.

$$doğruluk = TP + TN \,/\, TP + FP + TN + FN$$

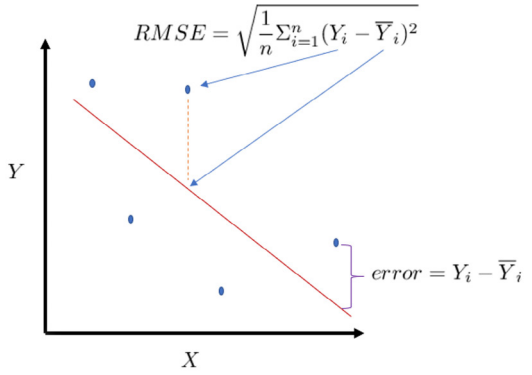
$$kesinlik = TP/(TP + FP)$$

$$duyarlılık = TP/(TP + FN)$$

$$F1 = 2 \times (kesinlik \times duyarlılık)/(kesinlik + duyarlılık)$$

Regresyon Analizinde Performans Ölçütleri

Regresyon analizinde hedef değişkenler sürekli veriler olduğu için karışıklık matrisi kullanılamaz. Regresyon analizinde gerçekleşen ile tahmin edilen arasındaki farka **hata terimi** denilir. Hata terimi aşağıdaki görseldeki gibi gösterilebilir:



Şekil 7-3: Regresyon analizinde hatalar

Regresyonda kullanılan performans ölçütlerine aşağıda yer verilmiştir.

Ortalama Mutlak Hata(Mean Absolute Error): Tahminlerin gerçek değerlerden mutlak farkının ortalamasıdır.

Ortalama Mutlak Hata: $MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}|$

Ortalama Karesel Hata (Mean Squared Error): $MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$

Ortalama Karesel Hataların Karekökü (Root Mean Squared Error): $RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}$

R^2 **Skoru (Determinasyon Katsayısı):** $1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y})^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$

Makine Öğrenmesi Projeleri için Uygulanabilecek Adımlar

Makine öğrenmesindeki temel kavramlardan ve örnek çalışmalardan bahsettikten sonra bir makine öğrenmesi projesi geliştirilirken uygulanabilecek adımlara yer verilebilir.

Makine öğrenmesinin ilk aşamasında **problemin ortaya koyulması** vardır. Bu aşama bilimsel bir çalışma yapılırken genellikle **ilk adım** olarak bilinir. Bu aşamada problemin çözümündeki amaçlardan, problemin çerçevesinden, sınırlılıklarından ve kapasitesinden bahsedilebilir. Örneğin spam e-postaların sınıflandırılması probleminde amaç gelen kutusuna gelecek e-postaların filtrelenmesi ve spam e-postaların spam klasörüne gitmesini sağlamaktır. Projenin sınırlılıklarına örnek olarak metin içeren e-postalar ya da 50 kelimenin üzerinde metin içeren e-postalar gibi kısıtlamalar eklenebilir. Bu bilgilerin ortaya koyulması ile problemin tanımlanması aşaması geçilmiş olunur.

İkinci aşamada **verilerin elde edilmesi** aşaması vardır. Daha önceki bölümlerde verilerin elde edilebileceği ücretli ve ücretsiz kaynaklardan bahsedilmişti. Kimi durumda veri setinin internet kaynaklarından bulunması mümkün olmayabilir. Proje için özel veri seti üretmek, veri toplamak, verileri istenilen formatta kaydetmek bu aşamada yerine getirilmesi gereken işlemlerden olabilir.

Diğer aşamada veri setinin istatistiksel yöntemler ile analiz edilmesi aşaması vardır. Proje için belirlenen veya oluşturulan ilk veri seti analize başlamadan önceki haliyle **ham veri seti (raw dataset)** olarak isimlendirilir. Ham veri setinin analize hazır hâle getirilmesi için daha önce bahsedilen veri önişleme süreçlerinin çalıştırılması gerekmektedir. Ardından veri seti içerisinde anlamlı ilişkilerin ortaya çıkarılması için korelasyon gibi ölçütlerin ortaya koyulması gerekebilir. Eğer bir regresyon çalışması yapılacaksa bağımsız değişkenler arasındaki korelasyonun düşük bağımsız değişken ile bağımlı değişken arasındaki korelasyonun ise yüksek olması istenir. Bu çerçevede bir analiz gerçekleştirilerek bazı bağımsız değişkenlerin analizden çıkarılması sağlanabilir. Ayrıca özellikler setindeki hedef değişkenle alakasız olduğu düşünülen değişkenler de sistemden çıkarılmalıdır. Spam e-posta örneğinde e-postaları gönderen kişilerin cinsiyetine bakmak gereksizdir. Cinsiyet özelliğinin veri setinden ve özellikler kümesinden çıkarılması başta düşünülmesi gereken bir süreç olmalıdır. Bunun gibi varsa başka alakasız özellikler bunların tespit edilerek analizden çıkarılması öğrenmenin kalitesine olumlu yönde etki edecektir. Bu aşamada özellik çıkarımı, özellik mühendisliği gibi yöntemlerden

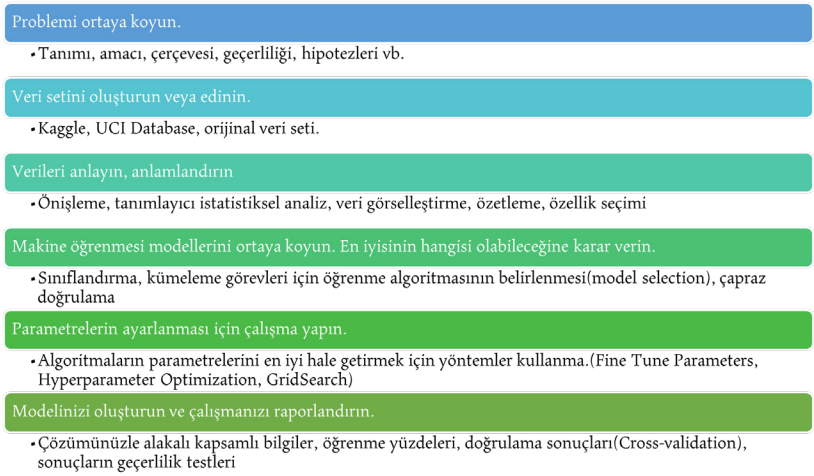
faydalanılmalıdır. Özellikler setine karar verildikten sonra ise veri setinin eğitim ve test seti ya da gerekli görülürse doğrulama veri seti olarak ayrılması işlemine geçilebilir.

Sonraki aşamada **makine öğrenmesi algoritmasının ortaya koyulması** var. Modelin seçilmesi ile modelin diğer modeller ile kıyaslanması, modelin performans değerlendirmesi gibi süreçler bu aşamada yerine getirilir.

Modeldeki parametrelerin ayarlanması, optimizasyonu süreçleri, modelin seçilmesinden sonra gerçekleştirilecek aşamadır. Bu aşamada parametre optimizasyonu veya hiperparametrelerin belirlenmesi işlemleri için literatürde geliştirilmiş yöntemlerden faydalanılabilir. Bu yöntemlere örnek olarak "**Grid Search**" verilebilir. Bu yöntem dışında çeşitli optimizasyon algoritmaları da kullanılabilir. Optimizasyon algoritmaları problemin karmaşıklığına göre sezgisel veya deterministik yöntemler olabilir.

Son aşamada ise projenin raporlanması ve sonuçlarının uygun şekilde sunulması işlemleri vardır. Bu aşamada gerekli istatistiksel testler, çapraz doğrulama sonuçları, sonuçların geçerlilik testleri gibi analizlerin verilmesi gerekir.

Yukarıda verilen aşamalar ayrıca şekilde özetlenmiştir. Makine öğrenmesi projeleri için bu aşamalar yerine getirilebilir. Tabii bu aşamalar şart olarak söylenemez. Bu aşamalar projeden projeye değişiklik gösterebilir.



Şekil 7-4: Makine öğrenmesi uygulama adımları

Başarısızlık Nedenleri

Makine öğrenmesi çalışmaları birçok denemeyi, deneme ve yanılmayı, tekrarlı çalışmaları, stresli işlemleri kapsayan zor bir iştir. Genel anlamda makine öğrenmesinin zorlukları veri setinden kaynaklı veya seçilen modelden kaynaklı olabilir. Veri setinin doğru seçilmemesi, eksik, hatalı, yanlış veriler içermesi gibi sorunlar makine öğrenmesi çalışmasını da zora sokabilir. Aynı şekilde veri seti doğru şekilde elde edilse de ardından uygulanan modelle ilgili sorunlar ortaya çıkabilir. Makine öğrenmesinde karşılaşılan zorluklar şu şekilde verilebilir.

- Yeterli verinin olmaması
- Verilerin kaliteli olmaması
- Doğru algoritmanın seçilmemesi
- Problemin makine öğrenmesi ile çözülmesinin mantıklı olmaması
- Aşırı öğrenme(Overfitting)
- Az Öğrenme (Underfitting)

Scikit-Learn Kütüphanesi Hakkında

Scikit-Learn kütüphanesine geçmeden önce Scikit-Learn ile kullanılan diğer kütüphaneler hakkında aşağıdaki bilgiler verilebilir. Verilen kütüphaneler, bilimsel çalışmalarda sıklıkla kullanıldığı için **SciKits (Science Kits – Bilim Araçları)** olarak bilinmektedir.[1] SciKits kütüphaneleri aşağıdaki gibi listelenebilir.

NumPy: N-Boyutlu diziler ve hesaplama kolaylığı sağlayan fonksiyonları sayesinde önemli avantajlar sunmaktadır.

Pandas: Seriler ve veri çerçeveleri veri yapıları ile tablo formatındaki verilerin manipülasyonu ve analizi için önemli avantajlar sunmaktadır.

Matplotlib: Verilerin görselleştirilmesi için kullanılmaktadır.

SciPy: Bilimsel hesaplamalar için kullanılan Python kütüphanesidir.

Scikit-Learn kütüphanesi açık kaynak kodlu bir Python kütüphanesidir. İlk olarak 2007 yılında Google Summer of Code etkinliğinde David Cournapeau tarafından geliştirilmiştir. Şu anda Google, the Python Software Foundation ve INRIA tarafından desteklenen 30'a yakın çalışanı ile geliştirilmesi devam etmektedir. Temel olarak NumPy ve SciPy kütüphaneleri üzerine kurulmuştur. Kütüphanenin ana sayfası *http://scikit-learn.org/* adresinde bulunmaktadır. Ana sayfada verildiği gibi Scikit-Learn kütüphanesi şu şekilde tanımlanabilir.

- Tahmine dayalı veri analizi için basit ve verimli araçlar
- Herkes tarafından erişilebilir ve çeşitli bağlamlarda yeniden kullanılabilir.
- NumPy, SciPy ve Matplotlib üzerine inşa edilmiştir.
- Açık kaynak kodlu, ticari olarak kullanılabilir - BSD lisansına sahip.

Yine kendi sayfasından Scikit-Learn kütüphanesinin çalıştığı alanlar şu şekilde özetlenebilir.

Uygulama Alanı	Tanım	Uygulamalar	Algoritmalar
Sınıflandırma	Nesnenin hangi kategoriye ait olduğunu belirleme.	İstenmeyen posta algılama, görüntü tanıma.	en yakın komşular, rastgele orman ve daha fazlası...
Regresyon	Bir nesneyle ilişkili sürekli değerli bir özniteliği tahmin etme.	İlaç tepkisi, hisse senedi fiyatları.	en yakın komşular, rastgele orman ve daha fazlası...
Kümeleme	Benzer nesnelerin kümeler halinde otomatik gruplandırması.	Müşteri segmentasyonu, Gruplandırma deneme sonuçları	k-Means, spectral clustering, mean-shift,
Boyut Azaltma	Dikkate alınması gereken rastgele değişkenlerin sayısını azaltma.	Görselleştirme, Artan verimlilik	k-Means, feature selection, non-negative matrix factorization
Model Seçme	Parametreleri ve modelleri karşılaştırma, doğrulama ve seçme.	Parametre ayarlama ile geliştirilmiş doğruluk	grid search, çapraz doğrulama, metrikler
Veri Önişleme	Özellik ayıklama ve normalleştirme.	Metin gibi giriş verilerini makine öğrenimi algoritmalarıyla kullanmak üzere dönüştürme.	önişleme, özellik çıkarımı

Tablo 7-1. Scikit-Learn Uygulama Alanları

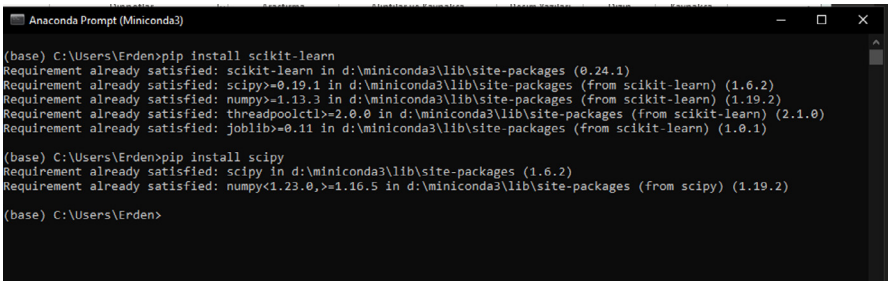
Scikit-Learn kütüphanesinin öne çıkmasının en önemli nedenlerinden birisi de iyi belgelendirilmiş olmasıdır. Makine öğrenmesi metotları kimi zaman karmaşık matematiksel ifadeler içerebilir. Scikit-Learn olabildiğince basit bir şekilde makine öğrenmesi metotlarını sunmaktadır. Böylece ileri düzey bir matematiğe ihtiyaç duymadan da makine öğrenmesi uygulaması geliştirmek mümkün olabilmektedir.

Bu özellikleri sayesinde Scikit-Learn birçok önemli işletme tarafından kullanılmaktadır. Örneğin Spotify, Scikit-Learn kütüphanesi ile müşterilerine yeni müzikler önermektedir. Geçmişteki müzik dinlemeleri ve müzik türleri

hakkındaki veriler kullanılarak dinleyicilere daha önce dinlemedikleri ve dinlemek isteyebilecekleri müzikler önerilir. Diğer bir örnek de not alma uygulaması olan **Evernote** uygulamasıdır. Evernote da Scikit-Learn kütüphanesini sıklıkla projelerinde kullanmaktadır.

Scikit-Learn Kütüphanesinin Yüklenmesi

Scikit-Learn kütüphanesini bilgisayara yüklemek için aşağıdaki şekilde gösterilen komutların komut istemcisine veya **Anaconda Prompt** içerisine yazılması gereklidir. Ancak Anaconda yüklemesi gerçekleştirilince Scikit-Learn kütüphanesi de yüklü gelmektedir. O nedenle yeniden yüklenmesine gerek yoktur. Eğer farklı bir ortamda çalışılacaksa ortam içerisine Scikit-Learn kütüphanesinin eklenmesi için bu yöntem kullanılabilir.



Şekil 7-5: Scikit-Learn Kütüphanesinin Anaconda Prompt ile Yüklenmesi

Yükleme doğrulandıktan sonra Jupyter notebook içerisinde aşağıdaki kodlar yardımıyla Scikit-Learn kütüphanesi çağrılabilir:

```
import sklearn as sk
import numpy as np
import pandas as pd
sk.__version__
```

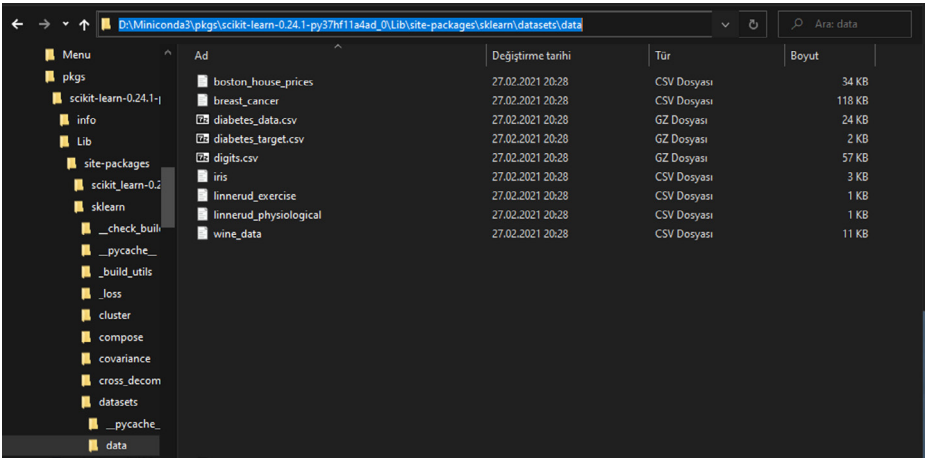
```
'0.24.1'
```

Kitapta kullanılan versiyon **0.24.1** versiyonudur. İlerleyen kısımlardaki bazı kodlar farklı versiyonlarda doğru çalışmayabilir.

Scikit-Learn Veri Setleri

Daha önce veri madenciliğinin kalbinde veri setleri var denmişti. Veri madenciliği yöntemlerinin uygulanabilmesi için veri setlerine ihtiyaç vardır. Veri setleri kaynaklarından birisi de Scikit-Learn veri setleridir. Bu kaynağın içerisinde de bilinen ve sıklıkla kullanılan veri setleri yer almaktadır. Yani Scikit-Learn veri setleri genellikle çok ön işleme aşaması gerektirmeyen, analize hazır veri setleridir. Makine öğrenmesinde kullanılan veri setleri çoğunlukla 2 boyutludur yani tablo formatındadır. Tablodaki her satır bir veriyi temsil eder. Sütunlar ise özellikleri gösterir. Aşağıdaki görselde Boston veri seti için satırlar ve sütunlar gösterilmiştir.

Veri setleri Scikit-Learn kütüphanesi ile gelmektedir. Eğer kütüphane yüklemesi tamamlandıysa veri setleri de bilgisayara yüklenmiş olacaktır. Veri setleri bilgisayarda csv dosyaları şeklinde bulunabilir. Bu işlem için izlenmesi gereken yol "**ANACONDA YÜKLEME KLASÖRÜ"\pkgs\scikit-learn-0.24.1-py37hf11a4ad_0\Lib\site-packages\sklearn\datasets\data**".



Şekil 7-6: Veri setlerine bilgisayardan ulaşma

Scikit-Learn veri setleri "**toy**" veri setleri olarak bilinir. Yani makine öğrenmesine yeni giriş yapacak kişiler için önerilen veri setlerinden bahsedilmektedir. Bu veri setleri hakkında kısaca bilgi verilmesi gerekirse;

Boston Ev Fiyatları (boston_house_prices.csv): Binanın yaşı, dairenin oda sayısı, binanın bulunduğu bölgedeki suç oranı gibi birtakım özelliklere bakılarak Boston eyaletindeki evlerin fiyatlarının bulunduğu veri setidir. Bu özellikler kullanılarak bir evin fiyatı tahmin edilmeye çalışılır.

- **Meme Kanseri (breast_cancer.csv):** Bir tıp veri setidir.
- **Diyabet (diabetes_data.csv):** Diyabet hastalığı veri setidir.
- **Zambak Çiçeği (iris.csv):** Zambak çiçeğinin türlerinin olduğu veri setidir.
- **Şarap Verileri (wine_data.csv):** Şarap verilerine ait veri setidir.

Bu veri setleri Jupyter notebooka çağrılırken **sklearn.datasets** içerisinden çağrılır. Aşağıdaki yöntem ile tüm veri setleri Jupyter notebook içerisine alınmış olunur.

```
from sklearn import datasets
boston = datasets.load_boston()
type(boston)
```

sklearn.utils.Bunch

Veri setlerinin kendilerine ait "**load_VERİSETİ()**" şeklinde bir yapısı mevcuttur. Dolayısıyla veri seti çağrılırken "**load_boston()**" ile çağrıldı. Aynı şekilde eğer meme kanseri veri seti çağrılmak istenseydi "**load_breast_cancer()**" fonksiyonu ile çağrılması gerekirdi.

```
breast_cancer = datasets.load_breast_cancer()
type(breast_cancer)
```

sklearn.utils.Bunch

Görüldüğü gibi veri setlerinin veri yapısı Bunch olarak gösterilmektedir. Bunch Python sözlüklerinin Scikit-Learn kütüphanesindeki karşılığı olarak görülebilir. Daha önce bahsedildiği gibi sözlük içerisinde **anahtar:değer** ilişkileri bulunmaktadır. Sözlük içerisindeki **anahtar(keys) değerleri** girilerek karşılığında gelecek değerler alınır. Bu sözlüğü parçalayarak veri setinin içeriğinden bahsedilmek gerekirse aşağıdaki kodlar kullanılabilir:

```
# Bunch içerisinde hangi anahtarlar var?
boston.keys()
```

dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])

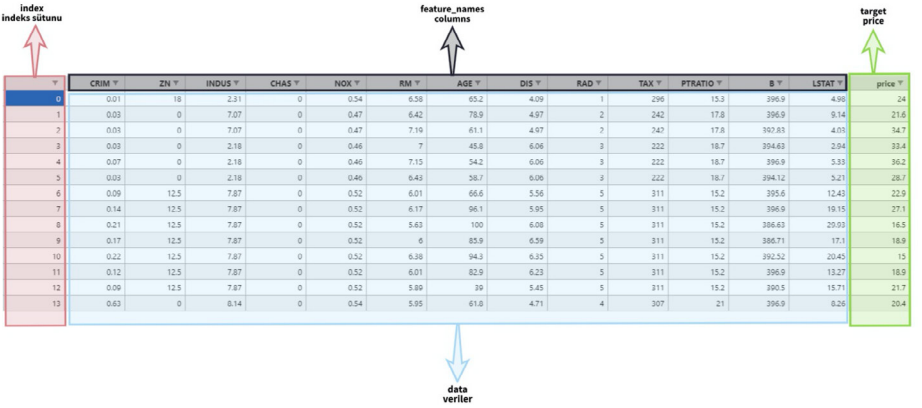
```
# Anahtarların veri yapıları neler?
type(boston.data),type(boston.target),type(boston.DESCR),type(boston.
feature_names)
```

(numpy.ndarray, numpy.ndarray, str, numpy.ndarray)

Boston veri seti için geçerli olan bu anahtarlar diğer veri setleri için de geçerlidir. Anahtarlar genellikle NumPy dizisi veri tipinde tutulmuştur. Anahtarlar hakkında kısaca bilgiler aşağıda verilmiştir.

data (boston.data): Bir NumPy dizisi şeklinde verilerin tutulduğu anahtardır. NumPy dizilerinde sunulan avantajlardan faydalanılarak incelenebilir.

target(boston.target): Hedef değişkeninin tutulduğu anahtardır. Yine NumPy dizisi şeklinde tutulur. Data ve target veri yapıları veri setini oluşturacak şekilde ayarlanmıştır. Aşağıdaki görselde Scikit-Learn kütüphanesinde geçtiği şekilde veri setlerinde kullanılan kavramlar tekrar gösterilmiştir.



	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	price
0	0.01	18	2.31	0	0.54	6.58	65.2	4.09	1	296	15.3	396.9	4.98	24
1	0.03	0	7.07	0	0.47	6.42	78.9	4.97	2	242	17.8	396.9	9.14	21.6
2	0.03	0	7.07	0	0.47	7.19	61.1	4.97	2	242	17.8	392.83	4.03	34.7
3	0.03	0	2.18	0	0.46	7	45.8	6.06	3	222	18.7	394.63	2.94	33.4
4	0.07	0	2.18	0	0.46	7.15	54.2	6.06	3	222	18.7	396.9	5.33	36.2
5	0.03	0	2.18	0	0.46	6.43	58.7	6.06	3	222	18.7	394.12	5.21	28.7
6	0.09	12.5	7.87	0	0.52	6.01	66.6	5.56	5	311	15.2	395.6	12.43	22.9
7	0.14	12.5	7.87	0	0.52	6.17	96.1	5.95	5	311	15.2	396.9	19.15	27.1
8	0.21	12.5	7.87	0	0.52	5.63	100	6.08	5	311	15.2	386.63	29.93	16.5
9	0.17	12.5	7.87	0	0.52	6	85.9	6.59	5	311	15.2	386.71	17.1	18.9
10	0.22	12.5	7.87	0	0.52	6.38	94.3	6.35	5	311	15.2	392.52	20.45	15
11	0.12	12.5	7.87	0	0.52	6.01	82.9	6.23	5	311	15.2	396.9	13.27	18.9
12	0.09	12.5	7.87	0	0.52	5.89	39	5.45	5	311	15.2	390.5	15.71	21.7
13	0.63	0	8.14	0	0.54	5.95	61.8	4.71	4	307	21	396.9	8.26	20.4

Şekil 7-7: Scikit-Learn Veri Setlerinin Yapısı

Boston veri setindeki veriler X değişkenine, hedef değişkeni de y değişkenine kaydedilebilir.

```
# X ve y değişkenleri
X = boston.data
y = boston.target
```

```
# Veri setinin pandas DataFrame e alınması
import pandas as pd
pd.options.display.float_format = '{:,.1f}'.format # ondalık kısım 1 karakterli
boston_veri_seti = pd.DataFrame(X, columns=boston.feature_names)
boston_veri_seti.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.0	18.0	2.3	0.0	0.5	6.6	65.2	4.1	1.0	296.0	15.3	396.9	5.0
1	0.0	0.0	7.1	0.0	0.5	6.4	78.9	5.0	2.0	242.0	17.8	396.9	9.1
2	0.0	0.0	7.1	0.0	0.5	7.2	61.1	5.0	2.0	242.0	17.8	392.8	4.0
3	0.0	0.0	2.2	0.0	0.5	7.0	45.8	6.1	3.0	222.0	18.7	394.6	2.9
4	0.1	0.0	2.2	0.0	0.5	7.1	54.2	6.1	3.0	222.0	18.7	396.9	5.3

Özet

Scikit-Learn kütüphanesi makine öğrenmesinin önde gelen kütüphanelerindendir. Birçok makine öğrenmesi araştırmacısı sadece Scikit-Learn kütüphanesini kullanarak işlerini halletmektedir. Bu nedenle kitabın bundan sonraki bölümlerinde yine Scikit-Learn kütüphanesi kullanımı gösterilecektir. Bu bölümle birlikte Scikit-Learn kütüphanesinin kullanımına bir giriş yapılmıştır.

Kaynaklar

1. Buitinck, L., Louppe, G., Blondel, M., et al. “API design for machine learning software: experiences from the scikit-learn project”, ECML PKDD Workshop: Languages for Data Mining and Machine Learning, pp. 108–122 (2013).

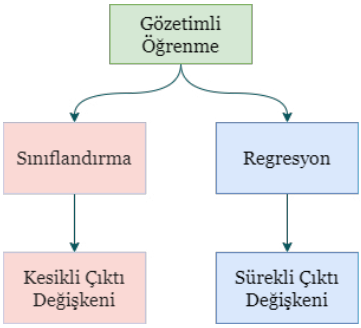
BÖLÜM 8

DOĞRUSAL REGRESYON MODELİ

Bu bölümde kitapta ilk çalışılacak model olan regresyon modelleri incelenecektir. Bu bölümün birinci kısmında regresyon analizinin temel kavramlarından ve temel hesaplamalarından bahsedilecektir. Ardından Scikit-Learn kütüphanesi ile regresyon uygulaması gösterilecektir.

Doğrusal Regresyon Modelleri

Regresyon modelleri makine öğrenmesi çalışmalarına başlayacak kişilerin ilk öğrendiği modellerdendir. Çünkü regresyon modelleri yapay sinir ağlarının da temelini oluşturur. Dolayısıyla derin öğrenme konusunun anlaşılması için de yine gerekli olacak bir modeldir. Regresyon modelleri makine öğrenmesi bölümünde belirtildiği gibi gözetimli öğrenme çeşidi içerisinde hedef değişkenin **sürekli** olduğu durumlarda geçerlidir. Bu durum aşağıdaki şekilde gösterilmiştir:



Şekil 8-1: Regresyon ve sınıflandırma modelleri

Regresyon modellerinde bağımlı değişken (y-çıktı, sonuç, hedef, etkilenen değişken), bağımsız değişkenler (x-girdi, neden, faktör, etkileyen değişken) tarafından açıklanır. Regresyon analizindeki cevabı aranan soru *bağımlı değişkenin bağımlı değişkene bağlı koşullarını nasıl tahmin edebiliriz* sorusudur. Bu koşullar bir neden sonuç ilişkisi doğuracaktır. O nedenle regresyon analizinde neden olan değişken x değişkeni sonuç olan değişken y değişkeni olarak söylenebilir. Regresyon modeli matematiksel olarak aşağıdaki gibi ifade edilebilir:

$$y = \beta_0 + \sum_{i=1}^{n} \beta_i \times x_i + hata(e)$$

Denklemde geçen y değişkeni tek boyutludur yani scaler formda bir dizidir. X değişkeni ise birden fazla boyuttan (**scaler dizilerden**) oluşabilir. Bu nedenle $x=[x_1,x_2,\ldots,x_n]$ denilebilir. n bağımsız değişkenlerin sayısını gösterir. Denklemdeki hata ($\mathbf{e}_i$) ise hataları göstermektedir. β_0 kesme terimidir (**intercept**), diğer parametreler ise (β_i) ise modeldeki katsayıları (**coefficent**) gösterir. hata (**e**) değerlerinde ise tahmin ile gerçekleşen arasındaki fark değerleri tutulur.

Makine öğrenmesinde bahsedilen öğrenme aşaması regresyon analizinde kesme terimi ve diğer parametrelerin yani $\beta_0,\beta_1,\ldots\beta_n$ değerlerinin belirlenmesi demektir. Bu katsayılar belirlendikten sonra değişkenler arasındaki ilişki ortaya konmuş olunur. Ardından yeni gelecek veriler için model çalıştırılarak tahmin değerlerine ulaşılabilir.

Regresyon analizinde 4 temel varsayım söz konusudur. Bunlar:

- **Doğrusallık**: X ile y değişkeni arasında lineer bir ilişkinin olması durumudur.
- **Varyansların eşitliği**: X değişkenin her değeri için hesaplanan hataların varyansının eşit olması durumudur.
- **Bağımsız Olma**: Gözlemlerin birbirinden bağımsız olması durumdur.
- **Normallik**: Hata terimindeki değerlerin normal dağılıma uyduğu durumdur.

Bu 4 varsayımın gerçekleştirildiği durumlarda regresyon modelleri kullanılabilir. Regresyonun temel kullanım alanı değişkenler arasındaki neden sonuç ilişkilerinin ortaya çıkarılmasıdır. Bu problem birçok alanda karşılaşılabilecek bir problemdir. Örneğin satış departmanında çalışan bir kişi kullandıkları reklam bütçelerinin satışları etkileyip etkilemediğini ölçmek isteyebilir. Diğer bir örnek olarak da hastalıkların tedavileri verilebilir. Acaba bir ilacın bir hastalığa iyi geldiği söylenebilir mi? Bu sorunun cevabı için de regresyon modelleri kullanılabilir.

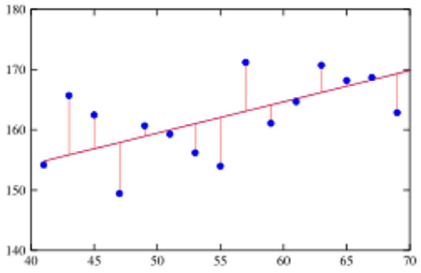
Tedavi uygulanan kişilerin tedavi uygulanmayan diğer hastalara göre daha hızlı iyileşme sürelerine sahip olduklarının ortaya koyulması gerekir. Eğer bir etki söz konusu ise, bağımsız değişkenin katsayısı (β_1) değerinin sıfırdan farklı olması beklenir. Katsayının sıfır değeri veya yakın değer alması ilişkinin olmadığını veya zayıf olduğunu gösterir. Bu örneklerde olduğu gibi regresyon analizi neden sonuç ilişkilerinin ortaya koyulmak istendiği birçok makine öğrenmesi alanında kullanılabilen ve en başta kullanılan modellerdendir.

Katsayıların Tahmini

Regresyon modellerinde yeni gelen verilerin çıktı değerlerinin belirlenebilmesi için katsayıların tahmin edilmesi gerekir. Tahmini yapılan denklem daha önce görülmemiş çıktılar anlamına gelen **şapka** işareti ile gösterilir. Şapkalı değerler gerçek değerleri değil, tahmini yapılan değerleri gösterir.

$$\hat{y} = \beta_0 + \sum_{i=1}^{n} \beta_i \times x_i$$

Regresyon modellerinde amaç hata değerinin minimize edilmesidir. Aşağıdaki şekilde mavi noktalar ile veri noktaları gösterilmiştir. Mavi noktalardan regresyon modeline doğru indirilen çizgiler **hataları** göstermektedir. Şekilde gösterilen modelde bir adet bağımsız değişken ile bir adet bağımsız değişkenin arasındaki bir regresyon modeli söz konusudur. Bu tip modellere **basit doğrusal regresyon modelleri** denilir.

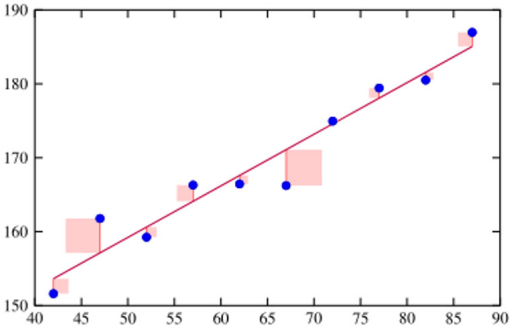


Şekil 8-2: Basit lineer regresyon modeli

Basit doğrusal regresyon modelinde toplamda iki adet parametre (β_0,β_1) bulunur. Bu parametrelerin tahmin edilmesi ile en az hataya sahip olan model keşfedilir. Parametrelerin tahmini için birçok yöntemden bahsedilebilir. İstatistiksel çalışmalarda bu yöntemlerden bazıları **en küçük kareler metodu** ve **maksimum olabilirlik yöntemleri**dir.

En Küçük Kareler Metodu

En küçük kareler (EKK) metodu en basit ve en fazla kullanılan tahmin edici olarak bilinir. Çünkü EKK uygulaması kolaydır ve birçok problemde uygulanabilir bir yöntemdir. Bu yöntemde $\beta_0 \beta_1$ değerleri gösterilen karelerin alanlarını en aza indirgemek için ayarlanır. Grafikte gösterilen kırmızı alanların en az olduğu zaman regresyon çizgisi ortaya çıkarılmış olur.



Şekil 8-3: En küçük karelerin gösterimi

En küçük kareler yönteminde bir minimizasyon problemi çözülür. Minimize edilecek olan değer hataların karelerinin toplamıdır. Aşağıdaki denklemde hataların kareleri gösterilmiştir.

$$\sum e^2 = \sum \left(Y_i - \widehat{Y}_i\right)^2 = \sum (Y_i - b_0 - b_1 X_i)^2 = min$$

Bu denklemin minimum noktasının olabilmesi için birinci türevinin sıfır olması gerekir. Buna göre b_0 ve b_1 için ayrı ayrı türev alıp sıfıra eşitlediğimizde aşağıdaki eşitliklere ulaşılır.

$$\sum Y_i = b_0(n) + b_1 \sum X_i \qquad \sum Y_i X_i = b_0 \sum x_i + b_1 \sum X_i^2$$

b_0 ve b_1 çekildiğinde;

$$b_0 = \frac{(\sum X^2 \sum Y - \sum X \sum XY)}{(n\sum X^2 - (\sum X)^2)} \qquad b_1 = \frac{(n\sum XY - \sum X \sum Y)}{(n\sum X^2 - (\sum X)^2)}$$

olur ve $\widehat{Y}_i = b_0 + b_1 X_i$ denkleminde yerine yazarak istenilen değerler için tahmin gerçekleştirilebilir.

Örnek: Aşağıdaki tabloda yapılan reklam giderlerine karşılık elde edilen satış rakamları verilmiştir. Reklam giderleri ile satışlar arasındaki regresyon modeli oluşturmak istenirse ilk adımda bağımlı ve bağımsız değişkenlerin belirlenmesi gereklidir. Bu örnekte bağımlı değişken (y) satışlar, bağımsız değişken ise reklam giderleridir.

Yıl	Reklam Giderleri	Satışlar
2001	7	223
2002	11	215
2003	15	233
2004	22	264
2005	26	305
2006	28	316
2007	31	320

Tablo 8-1: Reklam giderleri ve satış rakamları

Modeldeki parametrelerin tahmini için gerekli değerler X_i^2 ve X_iY_i bilgileridir. Bu değerler aşağıdaki tabloda hesaplanmıştır.

Y_i	X_i	X_i^2	X_iY_i
223	7	49	1561
215	11	121	2365
233	15	225	3495
264	22	484	5808
305	26	676	7930
316	28	784	8848
320	31	961	9920
1876	140	3300	39927

Tablo 8-2: Regresyon hesaplama tablosu

$b_0 = (\sum X^2 \sum Y - \sum X \sum XY)/(n\sum X^2 - (\sum X)^2)$ $b_1 = (n\sum XY - \sum X \sum Y)/(n\sum X^2 - (\sum X)^2)$ değerleri için aşağıdaki tablodaki hesaplamalar yapılır:

$$b_0 = \big(3300(1876) - 140(39927)\big)/(7(3300) - (140)^2) = 171{,}72$$
$$b_1 = \big(7(39927) - 140(1876)\big)/(7(3300) - (140)^2) = 4{,}841$$
$$y_i = 171{,}72 + 4{,}814x_i$$

Sapmalar ile Tahmin

EKK yöntemi aynı zamanda sapmalar cinsinden de ifade edilebilir. Sapmalar, **ortalamalardan farkları** göstermektedir. Bu nedenle $x_i = X_i - \bar{X}$ ve $y_i = Y_i - \bar{Y}$ dönüşümlerinin yapılması gereklidir. $\bar{X}$, $\bar{Y}$ ile değişkenlerin ortalamaları gösterilmektedir. Küçük x ile gösterilen ilgili noktanın ortalamadan farkını yani sapmayı göstermektedir. Bu durumda $\sum Y_i = b_0(n) + b_1 \sum X_i$ denklemindeki her iki tarafı da n ile bölünürse $\bar{Y} = b_0 + b_1\bar{X}$ denklemi oluşur. Regresyon denklemi ile alt alta yazılırsa:

$$Y_i = \beta_0 + \beta_1 X_i + e_i$$

$$\bar{Y} = b_0 + b_1\bar{X}$$

$$y_i = b_1 x_i + e_i$$

Bu modelin b_1'e göre türevi alındığında $b_1 = (\Sigma xy) \backslash (\Sigma x^2)$ eşitliği bulunur. Bulunan b_1 değeri $\bar{Y} = b_0 + b_1\bar{X}$ denkleminde yerine koyularak b_0 değerine ulaşılır. Az önce verilen reklam ve satış arasındaki regresyon modeli sapmalar ile tahmin edilmek istenirse aşağıdaki hesaplamaların yapılması gerekir:

Y_i	X_i	x_i	y_i	$x_i y_i$	x_i^2
223	7	-13	-45	585	169
215	11	-9	-53	477	81
233	15	-5	-35	175	25
264	22	2	-4	-8	4
305	26	6	37	222	36
316	28	8	48	384	64
320	31	11	52	572	121
1876	**140**	**0**	**0**	**2407**	**500**

Tablo 8-3: Sapmalarla tahmin hesaplamaları

$$b_1 = \left(\sum xy\right) / \left(\sum x^2\right) = 2407/500 = 4{,}8$$

$$\bar{Y} = b_0 + b_1\bar{X}$$
$$= 268 - 4{,}8(20) = 171{,}7$$

Determinasyon Katsayısı

Tahmin edilen bir regresyonun genel başarısı yüzdelik bir derece olarak determinasyon katsayısı ile ölçülür. R^2 ile gösterilen determinasyon katsayısı, basit regresyon için bağımlı ve bağımsız değişkenler arasındaki basit korelasyon katsayısıdır.

$R^2=(\Sigma xy)^2/(\Sigma x^2 \Sigma y^2)$ olur. $b_1=(\Sigma xy)/(\Sigma x^2)$ olduğu için $R^2=b_1(\Sigma xy)/(\Sigma y^2)$ ile determinasyon katsayısı hesaplanır. R^2 en az sıfır, en fazla bir değerini alabilir. Bire yakın değerler modelin verileri iyi açıkladığı anlamına gelir. Benzer şekilde sıfıra yakın değerler iyi bir model kurulmadığını gösterir.

Basit Doğrusal Regresyon Scikit-Learn Uygulaması

Gösterilen hesaplamalar NumPy kütüphanesinin sunduğu imkânlar ile geliştirilebilir. Ancak bu işlemler için kolaylık sunan Scikit-Learn kütüphanesinin fonksiyonlarından yararlanmak mantıklı olacaktır.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression  # lineer regresyon fonksiyonu
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_
error  # Performans ölçümleri için gerekli.
from sklearn.model_selection import train_test_split
```

```
# Veriler
reklam_giderler = np.array([7,11,15,22,26,28,31]) # x
satislar = np.array([223,215,233,264, 305,316,320]) # y
```

```
# Verilerin pandasa dönüştürülmesi
df = pd.DataFrame({'reklam_giderleri': reklam_giderler, 'satislar':
satislar})
df
```

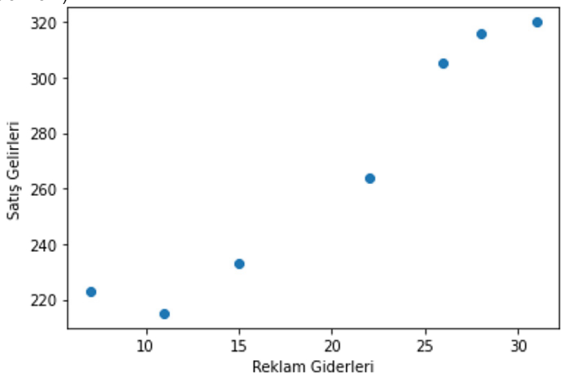
	reklam_giderleri	satislar
0	7	223
1	11	215
2	15	233
3	22	264
4	26	305
5	28	316
6	31	320

```
# İndex sütununu yıllara göre verelim.
df.index = ['2001', '2002', '2003', '2004', '2005', '2006', '2007']
df
```

	reklam_giderleri	satislar
2001	7	223
2002	11	215
2003	15	233
2004	22	264
2005	26	305
2006	28	316
2007	31	320

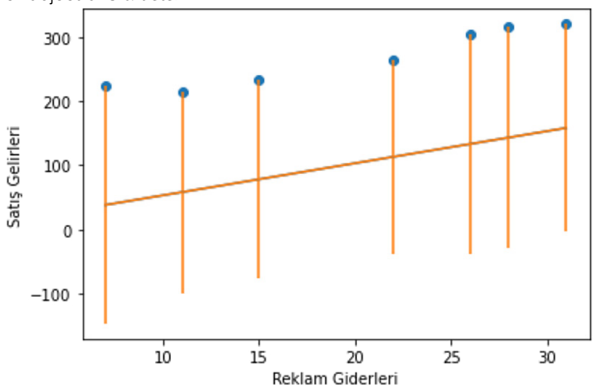
```
# Verilerin dağılım grafiği
plt.scatter(x=reklam_giderler,y=satislar)
plt.xlabel('Reklam Giderleri')
plt.ylabel('Satış Gelirleri')
```

```
Text(0, 0.5, 'Satış Gelirleri')
```



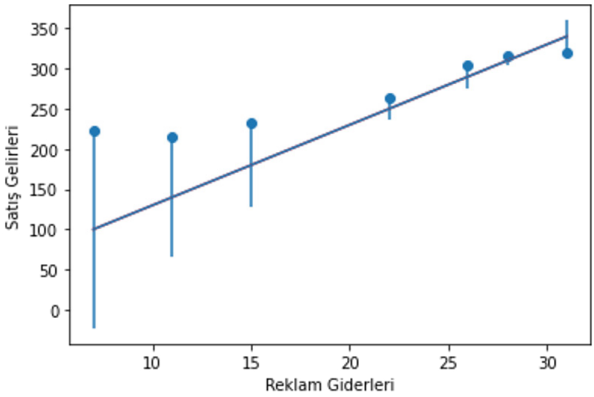
```
# deneme amaçlı reklam giderleri*5 + 3 denkleminin grafiği
plt.scatter(x=reklam_giderler,y=satislar)
plt.xlabel('Reklam Giderleri')
plt.ylabel('Satış Gelirleri')
plt.plot(reklam_giderler, reklam_giderler*5+3)
plt.errorbar(reklam_giderler, reklam_giderler*5+3, yerr=reklam_
giderler*5+3-satislar)
```

```
<ErrorbarContainer object of 3 artists>
```




```
# uzak kaldığı için *10 +30 deneyelim.
plt.scatter(x=reklam_giderler,y=satislar)
plt.xlabel('Reklam Giderleri')
plt.ylabel('Satış Gelirleri')
plt.plot(reklam_giderler, reklam_giderler*10+30, color='red')
plt.errorbar(reklam_giderler, reklam_giderler*10+30, yerr=reklam_
giderler*10+30-satislar)
```

```
<ErrorbarContainer object of 3 artists>
```



Rastgele çizilen çizginin hatalarının fazla olacaktır. O nedenle bir öğrenme algoritması ile verilerin öğrenilmesi çalışması yapılabilir.

`LinearRegression()` fonksiyonunun bazı parametreleri aşağıdaki gibi verilebilir:

- **fit_intercept bool, default=True:** Bu fonksiyon model için kesişme noktasının hesaplanıp hesaplanmayacağını söyler. `False` olarak ayarlanırsa, hesaplamalarda hiçbir kesişme kullanılmaz. Yani kesme terimi kullanılmaz.
- `normalize` giriş değişkenlerinin normalleştirilip normalleştirilmeyeceğine karar veren bir doğru/yanlış mantıksal sınamasıdır ve varsayılan olarak Yanlış değerini alır.

Modeli oluşturduktan sonra kullanılacak fonksiyonlar aşağıda verilmiştir:

- **fit():** Parametrelerin hesaplanmasını sağlar. Modele öğren demektir.
- **transform():** Modelin veri setine uygulanmasını sağlar.
- **fit_transform():** Fit ve Transform fonksiyonlarını birlikte sunar.

```
lineer_model = LinearRegression()
# beta_0 ve beta_1 i belirle
lineer_model.fit(reklam_giderler, satislar)
```

```
ValueError: Expected 2D array, got 1D array instead:
array=[ 7 11 15 22 26 28 31].
Reshape your data either using array.reshape(-1, 1) if your data has a single feature or array.reshape(1, -1) if it contains a single sample.
```

Yukarıdaki gibi bir hata alındığında düşünülmesi gereken `reshape` komutu ile değişkenlerin şeklini değiştirmek olacaktır.

```
# Hatanın çözümü için X'in sütun sütun verilmesi gerekiyor.
reklam_giderler = reklam_giderler.reshape(-1,1)
# beta_0 ve beta_1 i belirle
lineer_model.fit(reklam_giderler, satislar)
```

```
# beta_0,1 i yazdır
print('beta_0= {} beta_1 = {}'.format(lineer_model.intercept_,
                                      lineer_model.coef_[0]))
```

```
beta_0= 171.72 beta_1 = 4.814
```

```
# Modelin r2 skoru
lineer_model.score(reklam_giderler, satislar)
```

```
0.9305571795695471
```

Öğrenme performansından memnun olunduğunda tahmin aşamasına geçilebilir. Tahmin için kullanılacak olan fonksiyon **`predict()`** fonksiyonudur.

```
lineer_model.predict(reklam_giderler)
```

```
array([205.418, 224.674, 243.93 , 277.628, 296.884, 306.512, 320.954])
```

```
# Gerçek değerler
satislar
```

```
array([223, 215, 233, 264, 305, 316, 320])
```

```
# Hatalar (Errors)
satislar_tahmin = lineer_model.predict(reklam_giderler)
hatalar = satislar_tahmin - satislar
hatalar
```

```
array([-17.582,   9.674,  10.93 ,  13.628,  -8.116,  -9.488,   0.954])
```

```
# Hataların toplamı
hatalar.sum().round()
```

```
0.0
```

```
# hataların karesi
hatalarin_karesi = np.square(hatalar)
hatalarin_karesi
```

```
array([309.126724,  93.586276, 119.4649  , 185.722384,  65.869456,
       90.022144,   0.910116])
```

```
# Ortalama Karesel Hatalar (MSE - Mean Squared Error)
mse = np.mean(hatalarin_karesi)
mse
```

```
123.52885714285706
```

```
# Scikit-learn ile MSE
mean_squared_error(satislar, satislar_tahmin)
```

```
123.52885714285706
```

```
# Ortalama Mutlak Hatalar (MAE - Mean Absolute Error)
mae = np.mean(np.abs(hatalar))
```

```
10.053142857142856
```

```
# Scikit-learn ile MAE
mean_absolute_error(satislar, satislar_tahmin)
```

```
10.053142857142856
```

```
# Ortalama Karesel Hataların Karekökü (RMSE - Root Mean Squared Error)
rmse = np.sqrt(mse)
rmse
```

```
11.114353653850369
```

```
# Scikit-learn ile RMSE
mean_squared_error(satislar, satislar_tahmin, squared=False)
```

```
11.114353653850369
```

Varyans, gerçekleşen değerlerin tahmin edilen değerlerin ortalamasından ne kadar saptığını gösterir. r^2 skoru ile hesaplanır seviyesi hesaplanır. MSE ile alakalıdır.

```
varyans = np.var(hatalar)
varyans
```

```
123.52885714285706
```

score(): Gerçek değerler ile tahmin değerleri arasındaki hataları karşılaştırır. r^2 skoru bağımsız değişkenin öngörebildiği bağımlı değişkendeki varyans oranı diğer tanımla; (model tarafından açıklanan toplam varyans) / toplam varyans olarak verilebilir.

$$R^2 = 1 - \frac{\sum e_i^2}{\sum \left(Y - \overline{Y}\right)^2}$$

```
r2_skoru = 1 - (hatalarin_karesi /
                ((satislar - satislar.mean())**2).sum()).sum()
r2_skoru
```

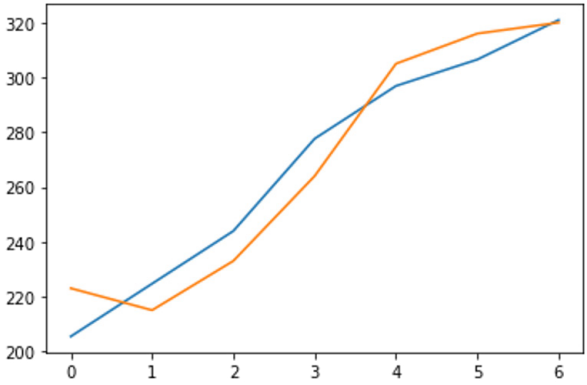
```
0.9305571795695471
```

```
# Scikitlearn içerisinde r2 score
r2_score(satislar, satislar_tahmin)
```

```
0.9305571795695471
```

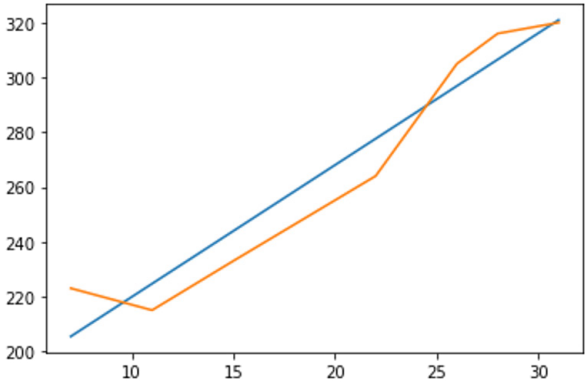
```
# Sonuçların görselleştirilmesi
plt.plot(satislar_tahmin)
plt.plot(satislar)
```

```
[<matplotlib.lines.Line2D at 0x163d9fe2860>]
```



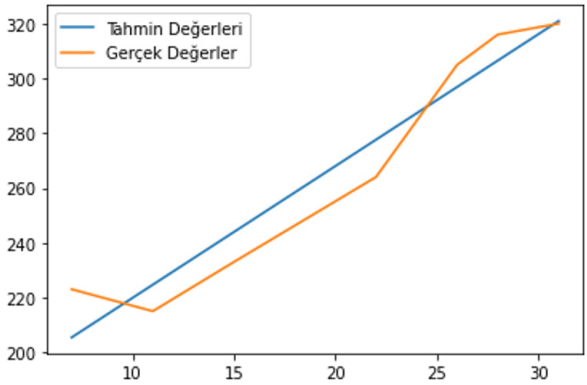

```
# x değerlerini verelim.
plt.plot(reklam_giderler, satislar_tahmin)
plt.plot(reklam_giderler, satislar)
```

```
[<matplotlib.lines.Line2D at 0x163da122e80>]
```



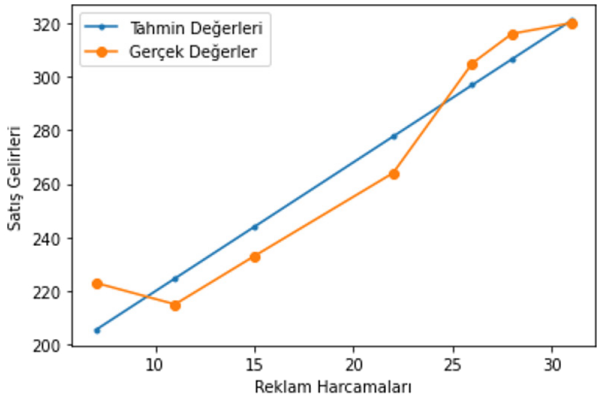
```
# etiketlerin eklenmesi
plt.plot(reklam_giderler, satislar_tahmin, label='Tahmin Değerleri')
plt.plot(reklam_giderler, satislar, label='Gerçek Değerler')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x163da1a3470>
```



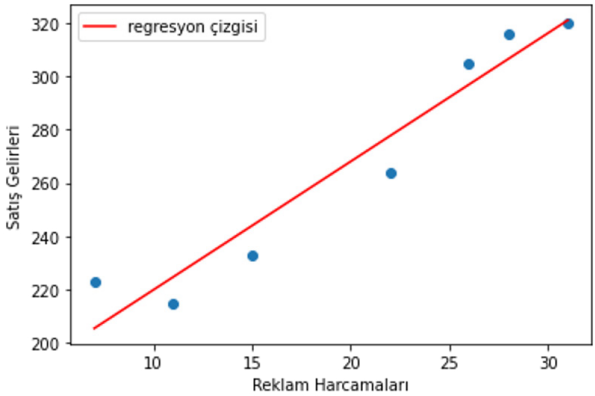

```
# Marker eklenmesi
plt.plot(reklam_giderler, satislar_tahmin, label='Tahmin Değerleri',
marker='.')
plt.plot(reklam_giderler, satislar, label='Gerçek Değerler', marker='o')
plt.xlabel('Reklam Harcamaları')
plt.ylabel('Satış Gelirleri')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x163da20bd30>
```



```
# regresyon çizgisi
plt.scatter(x=reklam_giderler,y=satislar)
plt.plot(reklam_giderler, reklam_giderler*lineer_model.coef_[0]+lineer_
model.intercept_, color='red', label='regresyon çizgisi')
plt.xlabel('Reklam Harcamaları')
plt.ylabel('Satış Gelirleri')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x163da3f7f28>
```



Gradyan Azalma Algoritması

Doğrusal regresyonda parametrelerin tahmini için EKK yöntemi sınırlı bir imkân sunar. Hata fonksiyonunun tahmin değerleri ile gerçek değerler arasındaki farkları gösterdiği söylenmişti. Hata fonksiyonunun minimum olduğu nokta modelin en iyi olduğu durumdur denilmişti. EKK yöntemi küçük problemler için bir çözüm sunsa da problemin karmaşıklığı arttıkça yetersiz hâle gelmektedir. Bu durumda kullanılacak bir algoritma **gradyan azalma algoritması** (gradient descent algorithm)dır. Gradyan azalma algoritması iteratif adımlar ile minimum noktaya erişmeye çalışır. Algoritma aşağıdaki adımlardan oluşmaktadır:

1- Rastgele bir başlangıç noktası belirle: Basit doğrusal regresyonda β_0 ve β_1 değerleri rassal olarak belirlenir.

2- Noktanın kayıp fonksiyonunu hesapla: Kayıp fonksiyonu olarak ortalama **karesel hataların karekökü** (root mean squared error) kullanılabilir.

$$RMSE = \sqrt{\left(\left(\sum \left(Y_i - \hat{Y}_i\right)^2\right)/n\right)}$$

3- Belirlenen noktanın türevini alarak eğimini hesapla.

$$\frac{d}{d\beta_1} = \frac{2}{n}\sum - x_i(y_i - (\beta_0 + \beta_1 x_i))$$

$$\frac{d}{d\beta_0} = \frac{2}{n}\sum - (y_i - (\beta_0 + \beta_1 x_i)$$

4- Adım boyutunu kullanarak ve yeni parametreleri hesapla.

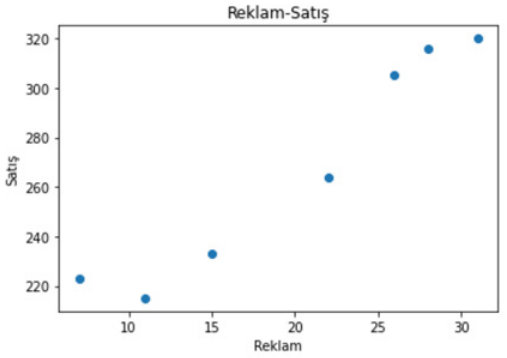
$$\beta_i^k = \beta_i^{(k-1)} - \Delta\beta_i$$

5- 2-4 arasını tekrar et.

İlgili adımlar örnekle aşağıda gösterilmiştir.

Y_i	X_i
223	7
215	11
233	15
264	22
305	26
316	28
320	31
1876	140

Tablo 8-4: Gradyan azalma algoritması örnek veri seti



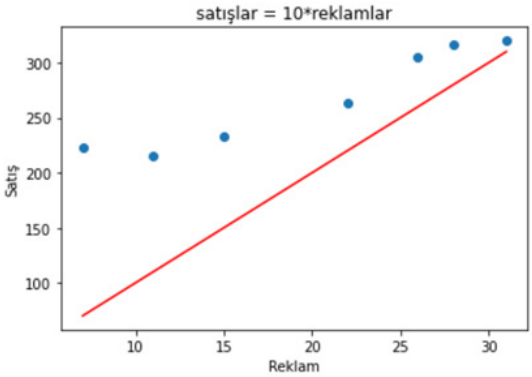
Şekil 8-4: Veri setinin grafiği

Satışlar=β_0+β_1×Reklam

İlk olarak rassal (0,1) parametre değerleri ile başlanırsa model aşağıdaki gibi kurulur:

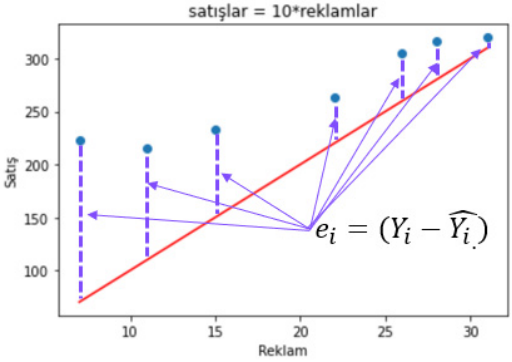
$$\text{Satışlar} = 0 + 10 \times \text{Reklam}$$

Modele ait regresyon grafiği çizildiğinde aşağıdaki gibi bir grafik elde edilir. Kırmızı çizgi regresyon çizgisini, mavi noktalar ise veri noktalarını göstermektedir.



Şekil 8-5: satışlar=10 reklamlar grafiği

Hata fonksiyonunu(RMSE)'yi hesaplayabilmek için tahmin değerlerine ihtiyaç duyulur. Hatalar aşağıdaki grafikte gösterilmiştir.



Şekil 8-6: Hataların gösterimi

```
In [29]:
satislar = np.array([223, 215, 233, 264, 305, 316, 320])
reklam = np.array([7, 11, 15, 22, 26, 28, 31])

In [30]:
y_prediction = 10*reklam
(y_prediction-satislar)

Out[30]: array([-153, -105,  -83,  -44,  -45,  -36,  -10])

In [31]:
np.mean((y_prediction-satislar)**2)

Out[31]: 6668.571428571428

In [32]:
RMSE = np.sqrt(np.mean((y_prediction-satislar)**2))
RMSE

Out[32]: 81.66132149660223
```

Şekil 8-7: RMSE hesabı

Görüldüğü gibi kesme terimi 0 olarak alındığında hata fonksiyonunun değeri 81,66 olarak bulunmuştur. Burada eğim katsayısı sabit tutularak **kesme teriminin hesabı** gösterilecektir. Eğim katsayısının hesaplanması için de benzer işlemler yapılabilir.

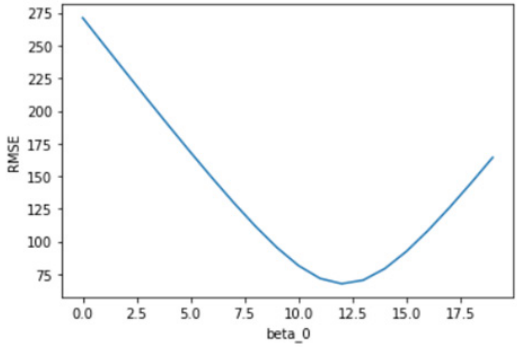
Bu aşamada eğer belirli aralıktaki tüm değerlerin denendiği bir çözüm istenirse aşağıdaki kodlar sayesinde bu değere ulaşılabilir.

```
In [62]:
RMSE_list = []
beta_0_list = []
for i in range(0, 20, 1):
    y_prediction = i * reklam
    RMSE = np.sqrt(np.mean((y_prediction - satislar)**2))
    RMSE_list.append(RMSE)
    beta_0_list.append(i)

In [63]:
RMSE_list

Out[63]: [271.29846505805585,
 250.3329211841132,
 229.50630243447097,
 208.86017195380126,
 188.45385945333447,
 168.37458240482735,
 148.7548318542964,
 129.80314766159233,
 111.85960588421287,
 95.49420326461107,
 81.66132149660223,
 71.83910594416625,
 67.79380502671317,
 70.52659073002182,
 79.34013576278493,
 92.51254741153456,
 108.4672432448749,
 126.1529683700365,
 144.93742492143684,
 164.4445195195024]
```

Şekil 8-8: RMSE listesi

Bu hata fonksiyonlarının grafiği aşağıdaki gibi olur:

```
In [67]:  1 plt.plot(beta_0_list,RMSE_list)
          2 plt.xlabel('beta_0')
          3 plt.ylabel('RMSE')

Out[67]: Text(0, 0.5, 'RMSE')
```



Şekil 8-9: beta0 grafiği

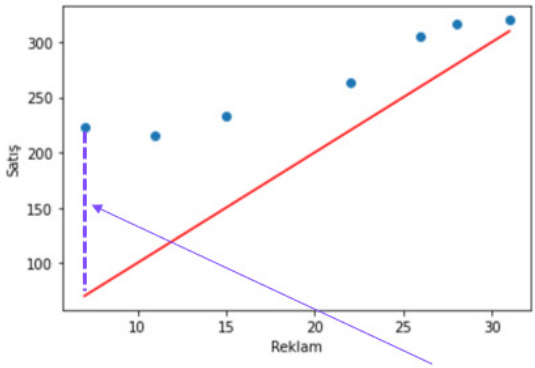
Hataların minimum olduğu kesme terimi değeri ise 12 olarak belirlenebilir.

```
In [68]:  1 beta_0_list[np.argmin(np.array(RMSE_list))]

Out[68]: 12
```

Şekil 8-10: beta0 listesi

Bu çözüm problem karmaşıklığı büyüdükçe işe yaramaz hâle gelir. Bu nedenle sezgisel yöntemler kullanarak karmaşık problemlerde de çözüme ulaşmak gereklidir. Bu yöntemlerden birisi **gradyan azalma algoritması**dır. Adım boyutunun hesabı için aşağıdaki işlemler yapılmalıdır:

Eğim katsayısının sabit tutulduğu durumda hataların karesinin denklemi şekilde gösterildiği gibi verilebilir.



$$e_1^2 = \left(223 - (\beta_0 + 10 * 7)\right)^2$$

Şekil 8-11: Adım boyutu hesabı

Bu hesaplamalar tüm noktalar için yapıldığında aşağıdaki denklem ortaya çıkar:

$$\sum e_i^2 = \left(223 - (\beta_0 + 10 * 7)\right)^2 + \left(215 - (\beta_0 + 10 * 11)\right)^2 + \left(233 - (\beta_0 + 10 * 15)\right)^2 + \left(264 - (\beta_0 + 10 * 22)\right)^2 + \left(305 - (\beta_0 + 10 * 26)\right)^2 + \left(316 - (\beta_0 + 10 * 28)\right)^2 + \left(320 - (\beta_0 + 10 * 31)\right)^2$$

Bu ifadenin türevi alındığında çözüm aşağıda verilmiştir. Bu tip basit türevlerin alınması için şu adreste bulunan hesaplama aracı kullanılabilir: *https://www.derivative-calculator.net/*

$(223-(x+10*7))^2+(215-(x+10*11))^2+(233-(x+10*15))^2+(264-(x+10*22))^2+(305-(x+10*26))^2+(316-(x+10*28))^2+(320-(x+10*31))^2$

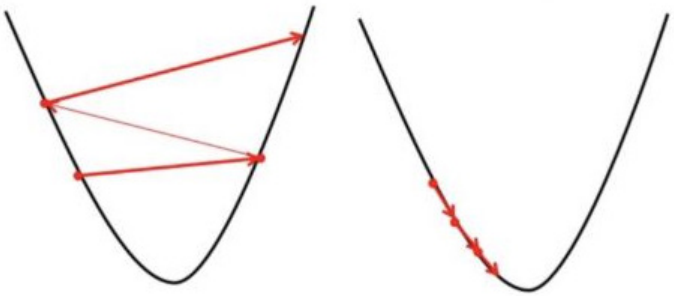
Sadeleştirme yapıldığında çözüm 14x-952 olarak bulunur. β_0=0 olduğundan eğim katsayısı -952 olarak gelir. Adım boyu = eğim * öğrenme katsayısı (λ) = -952*0,01=-9,52 olarak hesaplanır. Yeni β_0 önceki β_0 dan adım boyunun çıkarılması ile elde edilir.

$$\text{yeni}\beta_0 = \text{eski}\beta_0 - (\text{adımboyu})$$

$$\text{yeni}\beta_0 = 0 - (-9{,}52)$$

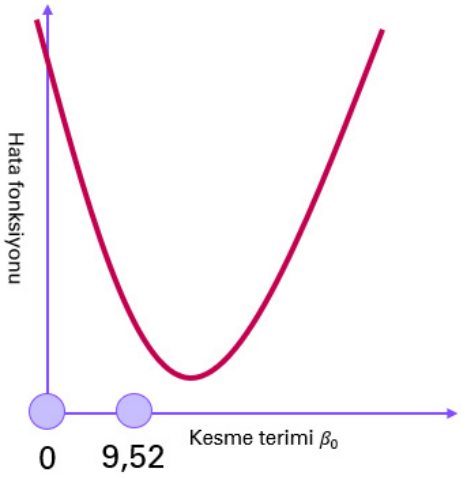
$$\text{yeni}\beta_0 = 9{,}52$$

olarak bulunacaktır. Formülde kullanılan öğrenme katsayısı önemli bir parametredir. Bu parametrenin yüksek olması durumunda hızlı bir şekilde en iyi noktaya ulaşılabilir, ama bu durumda en iyi noktanın atlanması riski de vardır. Küçük öğrenme katsayısı ile çalışılırsa bu kez en iyi noktanın atlanması riski azalır ancak bu durumda da daha yavaş bir şekilde en iyi noktaya ulaşılır. Bu durum aşağıdaki görselde gösterilmiştir.



Şekil 8-12: Küçük ve büyük öğrenme katsayısı

Yeni durumdaki hata fonksiyonu ve kesme terimi arasındaki grafik aşağıdaki gibi olur.



Şekil 8-13: Adım boyutunun güncellenmesi

Yeni durumda hata fonksiyonu değeri hesaplanırsa aşağıdaki gibi **73,92** değerine ulaşılır. Artık daha iyi bir hata fonksiyonuna erişilmiştir. Bu adımlar istenilen iterasyon sayısınca tekrar edilerek en iyi parametreler ortaya koyulabilir.

```
In [70]:
1 beta_0_952 = 9.52
2 y_prediction_952 = 9.52 + 10*reklam
3 np.mean((y_prediction_952-satislar)**2)
4 RMSE_952 = np.sqrt(np.mean((y_prediction_952-satislar)**2))
5 RMSE_952

Out[70]: 73.92213354991473
```

Şekil 8-14: RMSE 952 hesabı

Verilen örnekte sadece bir adet minimum nokta mevcuttur. Bu nedenle bu noktaya ulaşmak daha fazla minimumların olduğu durumlara göre daha kolaydır. Bazı problemler de **lokal minimum noktalarda** mevcut olabilir. Bu durumda global minimum noktaya ulaşabilmek için sezgisel algoritmanın olasılıksal hareketlerine ihtiyaç duyulur. Gradyan azalma algoritmasında stokastik gradyan azalma ile karmaşıklığı artan problemlere çözüm üretilir. Stokastik gradyan algoritması kodlar üzerinde gösterilecektir.

Regülarizasyon

Makine öğrenmesi çalışmalarında başarısızlık nedenlerinden birisi de modelin aşırı veya az öğrenmesi sorunlarıdır. Girdi sayısının fazla veri sayısının ise az olduğu durumlarda **aşırı öğrenme** sorunu söz konusu olabilir. Bu tip durumların çözümü için geliştirilen yöntemlerden birisi **regülarizasyon**dur. Bu yöntemde etkisi az olan parametrelerin sıfıra yakınlaştırılması sağlanarak girdi değişkeninin modelden çıkarılması işlemi gerçekleştirilir. Regresyon analizinde sıklıkla kullanılan regülarizasyon yöntemleri ridge regresyon ve LASSO olarak bilinir.

Ridge regresyon aynı zamanda **L2 Regresyon** olarak da geçer. Ridge regresyon, aşırı yüksek olan model parametrelerini cezalandırır. Bunu yaparken hata terimlerinin karelerinin toplamına aşağıdaki gibi katsayının karesini ekler. Böylece yüksek katsayılara ceza verilmiş ve ceza fonksiyonu yüksek tutulduğunda cezalandırılmış olur. Lambda parametresi cezalandırmanın şiddetini belirler. Lambda sıfır olduğunda ridge regresyon, lineer regresyona döner ve problem **EKK** ile çözülebilir hâle gelir.

$$RSS_{ridge} = \sum_{i=1}^{n}(y_i - x_i^T\beta)^2 + \lambda\sum_{j=1}^{p}\beta_j^2$$

Diğer formda ise Least Absolute Shrinkage and Selection Operator (LASSO) regresyon bulunur. LASSO da aşağıdaki gibi bir yapıda hataların karelerini cezalandırır.

$$RSS_{LASSO} = \sum_{i=1}^{n}(y_i - x_i^T\beta)^2 + \lambda\sum_{j=1}^{p}\beta_j$$

Regülasyon ile ilgili kullanım örneği Regresyon Analizi Scikit-Learn Uygulaması içerisinde verilecektir.

Çoklu Regresyon Analizi ve Scikit-Learn Uygulaması

Önceki bölümde ele alınan veri seti basit doğrusal regresyon için uygun bir veri setiydi. Birden fazla bağımsız değişkenin olması durumunda çoklu regresyon analizinin gerçekleştirilmesi gerekir. Bu kısımda Scikit-Learn kütüphanesi ile bir regresyon modellerinin kurulması ve çözümleri gösterilecektir.

Boston Veri Seti Regresyon Uygulaması

Örnek veri seti olarak Scikit-Learn kütüphanesi içerisindeki oyuncak veri setlerinden Boston veri seti kullanılacaktır.[1] Boston veri setinde Boston'daki ev fiyatları ile ilgili bilgiler paylaşılmıştır. Bu veri seti ile ilgili temel bilgiler önceki bölümde verilmişti.

```
from sklearn import datasets
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_
error
pd.options.display.float_format = '{:,.1f}'.format # ondalık kısım 1
karaktere sahip olsun
boston = datasets.load_boston()
# X ve y değişkenleri
X = boston.data
```

```
y = boston.target
boston_veri_seti = pd.DataFrame(X, columns=boston.feature_names)
dfy = pd.DataFrame(y,columns=['price'])
dfX = pd.DataFrame(X, columns=boston.feature_names)
df = dfX.join(dfy)
```

Veri seti tanıtıldıktan sonra öğrenme aşamasına geçilebilir. Bu problemde hedef değişken sürekli bir değişken olduğu için regresyon analizi yapılabilir. Problemde birden fazla bağımsız değişken olduğundan dolayı aşağıdaki gibi bir model kurulabilir.

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

Denklemde Y değişkeni hedef değişkeni X değişkenleri ise özellik değişkenlerini göstermektedir. β katsayıları katsayıları sklearn içerisinde coef_, β_0 ise intercept_ olarak kaydedilir.

Analize geçmeden önce veri setinde eksik veri olup olmadığının kontrol edilmesi gerekir.

```
df.isnull().sum()
```

```
CRIM      0
ZN        0
INDUS     0
CHAS      0
NOX       0
RM        0
AGE       0
DIS       0
RAD       0
TAX       0
PTRATIO   0
B         0
LSTAT     0
price     0
dtype: int64
```

Eksik veri olmadığını gördükten sonra başka veri önişleme aşaması yapmadan analize geçilmesi için veri setinin eğitim ve test veri seti olarak ayrılması gereklidir.

```
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.20,
                                                    random_state=12345)
```

Doğrusal regresyon modeli bu aşamada kurulabilir. Öğrenme eğitim setleri üzerinde gerçekleştirilir. Aşağıda **fit** fonksiyonu ile modelin öğrenmesi sağlanmıştır.

```
model = LinearRegression()
model.fit(X_train,y_train)
```

Eğitim gerçekleştikten sonra regresyondaki katsayıları getirmek için aşağıdaki kodlar kullanılabilir.

```
print("katsayılar: ", model.coef_)
print("kesme erimi: ", model.intercept_)
print("katsayı sayısı: ", model.coef_.shape)
```

```
katsayılar:  [-1.05894430e-01  4.94929319e-02  2.83620587e-02  2.97406193e+00
 -1.72401630e+01  4.55694468e+00 -5.72640896e-03 -1.33306114e+00
  2.61699220e-01 -1.09652598e-02 -9.90192265e-01  8.04198293e-03
 -4.46205178e-01]
kesme terimi:  31.24666483422307
katsayı sayısı:  (13,)
```

Bulunan katsayılar 13 adet özelliğin ağırlıklarını göstermektedir. Kesme terimi değeri 31,24 olarak bulunmuştur. Bulunan parametrelere göre test veri seti üzerinde **tahmin işlemi** gerçekleştirilebilir.

```
y_predict = model.predict(X_test)
# Determinasyon katsayısını verir.
model.score(X,y)
```

```
0.7368643662107702
```

%73 oranında bir öğrenmenin gerçekleştiği görülmektedir. Bu öğrenmenin test verisi üzerinde ne kadar olduğuna bakmak için;

```
r2_score(y_test,y_predict)
```

```
0.6160762605146671
```

Bu sonuç bazı durumlarda kabul edilebilir bir sonuç olabilir. Ancak bu sonuç hangi setin eğitim hangi setin test olarak seçildiğine bağlı olarak değişkenlik gösterecektir. Daha güvenilir bir sonuç için çapraz doğrulama yapılması gerekir.

```
from sklearn.model_selection import cross_val_score
cv_scores = cross_val_score(model,X,y,cv=10)
np.mean(cv_scores)
```

```
0.2025289900605657
```

Çapraz doğrulama sonucunda görüldüğü gibi düşük bir performans elde edilmiştir. Bu nedenle ilk başta bulunan yüksek doğruluk seçilen eğitim ve test veri setlerine bağlıdır, sonucu çıkarılabilir. Bu durumu düzeltmek için **parametre optimizasyonu** yapmak gereklidir. Doğrusal regresyon modelindeki parametreler; **`fit_intercept=True`**, **`normalize=False`**, **`copy_X=True`**, **`n_jobs=None`**, **`positive=False`** olarak verilmiş.

```
from sklearn.model_selection import GridSearchCV
parameters = {'fit_intercept':(True, False), 'normalize':(True, False),
'copy_X':(True, False), 'positive':(True, False)}
grid = GridSearchCV(model,parameters, cv=10)
grid.fit(X_train, y_train)
print(grid.best_score_)
```

```
0.7323649027911181
```

Boston Veri Seti ile Uygulama 2

Bu bölümde tekrar Boston veri setini kullanılarak ayrı bir çoklu regresyon analizi yapılacaktır. Önceki örnekten farklı olarak bu çalışmada farklı bir yolla tanımlamalar ve yaklaşımlar ortaya koyulacaktır. Yani başka bir notebook dosyası ile çalışıldığı varsayılarak bazı kütüphaneler yeniden çağrılacaktır.

```
from sklearn.datasets import load_boston
boston = load_boston()
boston_X, boston_y = load_boston(return_X_y=True)
```

```
# Sütun isimleri
boston.feature_names
```

```
array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
       'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7')
```

```
# X değerlerini pandas çerçevesine dönüştürelim
boston_X = pd.DataFrame(boston_X, columns=boston.feature_names)
```

```
#  veri seti
boston_X.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

```
# veri setini inceleyelim
boston_X.info()
```

<class 'pandas.core.frame.DataFrame'> RangeIndex: 506 entries, 0 to 505 Data columns (total 13 columns): CRIM 506 non-null float64 ZN 506 non-null float64 INDUS 506 non-null float64 CHAS 506 non-null float64 NOX 506 non-null float64 RM 506 non-null float64 AGE 506 non-null float64 DIS 506 non-null float64 RAD 506 non-null float64 TAX 506 non-null float64 PTRATIO 506 non-null float64 B 506 non-null float64 LSTAT 506 non-null float64 dtypes: float64(13) memory usage: 51.5 KB

```
# Tanımlayıcı istatistikleri
boston_X.describe()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032	12.653063
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	7.141062
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.730000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	6.950000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	11.360000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	16.955000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.970000

```
# boş değer var mı
boston_X.isnull().sum()
```

```
CRIM      0
ZN        0
INDUS     0
CHAS      0
NOX       0
RM        0
AGE       0
DIS       0
RAD       0
TAX       0
PTRATIO   0
B         0
LSTAT     0
dtype: int64
```

```
# korelasyonlara bakalım.
boston_X.corr()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
CRIM	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247	0.352734	-0.379670	0.625505	0.582764	0.289946	-0.385064	0.455621
ZN	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	0.664408	-0.311948	-0.314563	-0.391679	0.175520	-0.412995
INDUS	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	-0.708027	0.595129	0.720760	0.383248	-0.356977	0.603800
CHAS	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	-0.099176	-0.007368	-0.035587	-0.121515	0.048788	-0.053929
NOX	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	-0.769230	0.611441	0.668023	0.188933	-0.380051	0.590879
RM	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	0.205246	-0.209847	-0.292048	-0.355501	0.128069	-0.613808
AGE	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	-0.747881	0.456022	0.506456	0.261515	-0.273534	0.602339
DIS	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	1.000000	-0.494588	-0.534432	-0.232471	0.291512	-0.496996
RAD	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	-0.494588	1.000000	0.910228	0.464741	-0.444413	0.488676
TAX	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	-0.534432	0.910228	1.000000	0.460853	-0.441808	0.543993
PTRATIO	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	-0.232471	0.464741	0.460853	1.000000	-0.177383	0.374044
B	-0.385064	0.175520	-0.356977	0.048788	-0.380051	0.128069	-0.273534	0.291512	-0.444413	-0.441808	-0.177383	1.000000	-0.366087
LSTAT	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339	-0.496996	0.488676	0.543993	0.374044	-0.366087	1.000000

```
# En yüksek mutlak 5 korelasyonu sıralayalım
for col in boston.feature_names:
    en_yuksek_degerler = abs(boston_X.corr()[col]).nlargest(
        n=5)  # en yüksek korelasyona sahip 5 değeri alalım
```

```
    print(en_yuksek_degerler)
    # eğer 0.75'ten büyük değer varsa yazdır.
    for index, value in en_yuksek_degerler.items():
        if 1 > value >= 0.75:
            print(index, col, "değişkenleri yüksek korelasyona sahip: ",
value)
```

Özellikler arasında yüksek korelasyona sahip özellikler, analizden çıkarılabilir. Çünkü bağımsız özelliklerin aralarındaki korelasyonun **düşük** olması gerekmektedir.[2] **NOX** ve **RAD** değişkenleri analizden çıkarılabilir.

```
boston_X = boston_X.drop(['NOX', 'RAD'], axis=1)
```

```
X_train, X_test, y_train, y_test = train_test_split(boston_X,
                                                    boston_y,
                                                    test_size=0.33,
                                                    random_state=42)
```

```
coklu_regresyon = LinearRegression()
coklu_regresyon.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
#katsayılar
print(coklu_regresyon.coef_.round(2))
print(coklu_regresyon.intercept_.round(2))
```

```
[-0.08 0.03 -0.05 3.45 4.16 -0.03 -1.18 -0. -0.69 0.01 -0.57]
19.11
```

```
boston_y_predicted = coklu_regresyon.predict(X_test)
```

```
mean_squared_error(y_test, boston_y_predicted, squared=False)
```

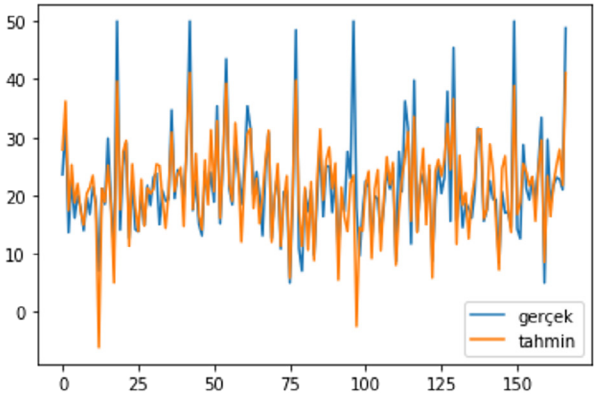
```
4.827980073358347
```

```
r2_score(y_test, boston_y_predicted)
```

```
0.6919945689994251
```

```
%matplotlib inline
plt.plot(y_test, label='gerçek')
plt.plot(boston_y_predicted, label='tahmin')
plt.legend()
```

<matplotlib.legend.Legend at 0x12e0e2f7108>



Özet

Doğrusal regresyon modelleri makine öğrenmesinde ilk akla gelen modellerdendir. Bu modeldeki kavramlar ve işlemler iyi bir şekilde kavranırsa diğer modellerin anlaşılması da kolaylaşacaktır. Bu nedenle bu bölümde detaylı bir şekilde modelin çalışma mantığı da gösterilmiştir.

Kaynaklar

1. "Boston Dataset", https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html.
2. "Why exclude highly correlated features when building regression model ?? | by Aishwarya V Srinivasan | Towards Data Science", https://towardsdatascience.com/why-exclude-highly-correlated-features-when-building-regression-model-34d77a90ea8e.

BÖLÜM 9

LOJİSTİK REGRESYON İLE SINIFLANDIRMA

Bu bölümde sınıflandırma algoritmalarından birisi olan **lojistik regresyon** modellerinden bahsedilecektir. Bir önceki bölümde gösterilen doğrusal regresyon modelleri hedef değişkeninin sürekli olduğu durumlarda çalışmaktaydı. **Sınıflandırma** modellerinde ise çıktı değişkenleri kesikli değerlerden oluşur.[1-3] Kesikli değişkenin tahmini için lojistik regresyon modelleri kullanılabilir. Konu anlatımının ardından lojistik regresyon için bir örnek uygulama bölüm sonunda gösterilecektir.

Lojistik Regresyon

Lojistik regresyonun anlaşılabilmesi için 5 adet müşteriden oluşan aşağıdaki veri seti dikkate alınabilir. Veri setinden önce kullanılacak olan kütüphaneler de ayrıca aşağıdaki kod hücresinde verilmiştir.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score
```

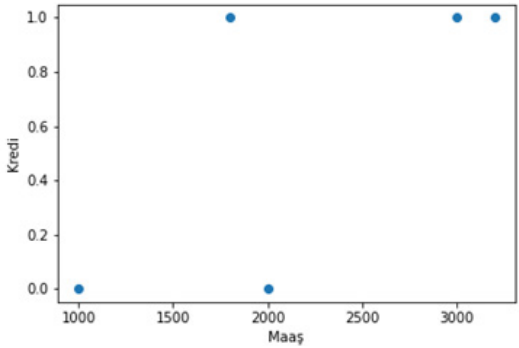
```
# Örnek veri seti
df = pd.DataFrame(data={
    'Maaş': [3000, 2000, 3200, 1000, 1800],
    'Kredi': [1, 0, 1, 0, 1]
})
df
```

	Maaş	**Kredi**
Müşteri 1	3000	1
Müşteri 2	2000	0
Müşteri 3	3200	1
Müşteri 4	1000	0
Müşteri 5	1800	1

Tablo 9-1: Lojistik regresyon örnek veri seti

Bu tabloya göre 5 müşteri için **maaş** değişkeni ve **kredi** değişkeni olduğu görülebilir. Bu verilerin bir bankanın müşterilerine ait olduğu varsayılırsa müşterilerin maaş değerlerine göre kredi alıp almadığı sonucuna bakılmak istenebilir. Banka müşterilerin yüksek maaşa sahipse kredi kullanmasını bekler. Tablo verilerinin grafiği çizilmek istenirse aşağıdaki gibi bir grafik elde edilir:

```
# dağılım grafiği
plt.scatter(df['Maaş'], df['Kredi'])
plt.xlabel('Maaş')
plt.ylabel('Kredi')
```

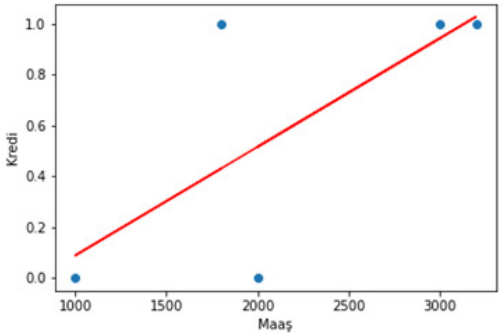


Şekil 9-1: Dağılım grafiği

Dağılım grafiğinden görülebileceği gibi maaş değişkenleri kredi değişkenini sadece **0 ve 1** değerlerinde kesmektedir. Bu problem doğrusal regresyon modeli ile çözülmek istenirse aşağıdaki grafikte gösterildiği gibi bir çizgi çizilmesi gerekecektir. Çizilen regresyon çizgisi iyi bir sonuç vermez. Çünkü çizginin 0,1 arasındaki ve bu aralık dışındaki değerleri de aldığı görülebilir. Örneğin regresyon çizgisi rastgele olarak alınan 5000 değeri için 1'in üzerinde değer alacaktır. Kredi değişkeninde ise 1'in üzerinde değerler yoktur. Bu nedenle regresyon çizgisinin 0 ile 1 aralığında sınırlandırılması gerekmektedir.

```
# doğrusal model
x = df['Maaş'].values.reshape(-1, 1)
y = df['Kredi']
model = LinearRegression()
model.fit(x, y)
```

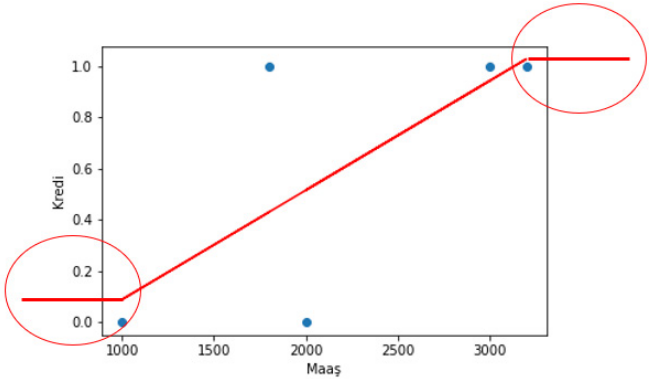
```
# regresyon çizgisi
beta_1, beta_0 = np.polyfit(df['Maaş'], df['Kredi'], 1)
plt.scatter(df['Maaş'], df['Kredi'])
plt.xlabel('Maaş')
plt.ylabel('Kredi')
plt.plot(x, beta_0 + beta_1 * x, color='red')
```



Şekil 9-2: Regresyon çizgisi ile beraber dağılım grafiği

Sınırlandırma işlemi 0'dan küçük değerlerin 0'a getirilmesi, 1'den büyük değerlerin ise 1'e getirilmesi ile yapılabilir. Aşağıdaki grafikte ilgili işlem görülebilir. Bu işleme göre en yüksek maaşı alan kişiden sonra gelecek maaş değerlerinin 1 değerini alması gerektiği söylenebilir. Aynı şekilde en

düşük maaş değerinden daha küçük değerler için kredi değişkeninin 0 değeri alacağı söylenebilir. Dikkat edilmesi gereken bir diğer durum da ara değerler içindir. Yani minimum ve maksimum değerler arasında kalan değerlerin kredi değişkeninin 0 ya da 1 değerlerinden hangisini alacağının incelenmesi gerekir. Bunu belirlemek için bir **eşik değeri** belirlenerek aşağı değerler 0, yukarı değerler 1 olarak kabul edilebilir. Ayrıca 0 ve 1 değerlerine yakınlık derecesine göre bir olasılık belirlenerek gelecek tüm değerlere 0 ya da 1 ataması yapılabilir.



Şekil 9-3: Lojistik regresyonun düzeltmeleri

Logit Fonksiyonu

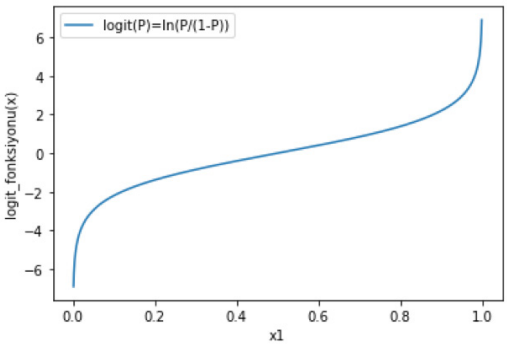
Lojistik fonksiyon bu noktada devreye girer anacak lojistik fonksiyondan önce türevlerinden olan ve literatürde bilinen diğer bir fonksiyon olan **logit** fonksiyonundan bahsedilebilir. Logit fonksiyonu aşağıdaki gibi verilebilir:

$$logit(P) = \ln\left(\frac{P}{1-P}\right)$$

Logit fonksiyonunda yer alan P değeri bir olasılığı göstermektedir. Bu nedenle 0<P<1 olmalıdır. P değeri bir olayın başarılı olma olasılığını gösterirken 1-P ise olayın başarısız olma olasılığını gösterir. P ile 1-P değerlerinin bölünmesinden odd değeri ortaya çıkar. Örneğin bir torbada 8 adet kırmızı top 2 adet mavi top varsa kırmızı top çekme olasılığı mavi top çekmekten 4 kat fazladır denilir. Bu hesaplamadaki 4 değeri, kırmızı çekme olasılığı 0,8 kırmızı çekmeme olasılığına yani 0,2'ye bölünerek bulunur. Yani logit fonksiyonu 0 ile 1 arasındaki değeri -sonsuz +sonsuz arasına getirmiş olur. Aşağıdaki kodlar ile logit fonksiyonu görselleştirilebilir.

```
def logit_fonksiyonu(x):
    return np.log(x / (1 - x))

x1 = np.linspace(0.001, 0.999, 1000)
plt.plot(x1, logit_fonksiyonu(x1), label="logit(P)=ln(P/(1-P))")
plt.xlabel("x1")
plt.ylabel("logit_fonksiyonu(x)")
plt.legend()
```

<matplotlib.legend.Legend at 0x2b84ddeb400>



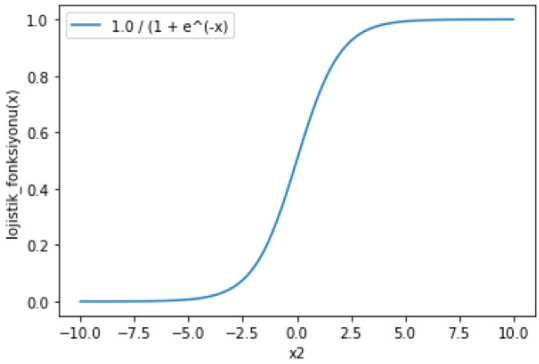
Logit fonksiyonunun eğrisinde girdi değerlerinin 0 ile 1 arasında değiştiği görülebilir. Lojistik regresyon analizinde tam tersine çıktı değişkenlerinin 0 ile 1 arasında değişmesi istenir. Bu durumun sağlanması için yapılan dönüşümlerin ardından ortaya **lojistik fonksiyonu (sigmoid fonksiyonu)** çıkar. Sigmoid fonksiyonu verilen herhangi bir değeri 0 ile 1 arasına getirir. Sigmoid fonksiyonunun alabileceği en yüksek değer 1 en düşük değer ise 0'dır. Aşağıdaki denklemde lojistik fonksiyonu verilmiştir.

$$S(x) = \frac{1}{1 + e^{(-x)}}$$

Aşağıdaki kodlar ile lojistik fonksiyon görselleştirilebilir. Lojistik fonksiyona verilen 0 değeri için fonksiyonun 0,5 değeri aldığı görülecektir. Ya da yüksek değerler için 1 düşük değerler için 0 değeri alacağı görülecektir.

```
def lojistik_fonksiyonu(x):
    return 1.0 / (1 + np.exp(-x))

x2 = np.linspace(-10, 10, 100)
plt.plot(x2, lojistik_fonksiyonu(x2), label="1.0 / (1 + e^(-x)")
plt.xlabel("x2")
plt.ylabel("lojistik_fonksiyonu(x)")
plt.legend()
```

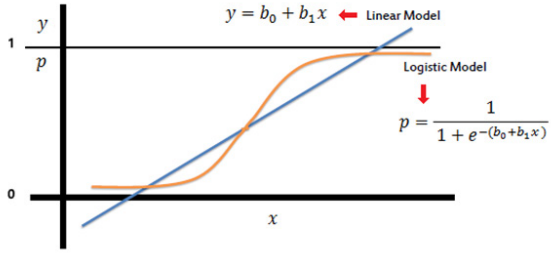
```
<matplotlib.legend.Legend at 0x2b84de25c50>
```



Grafikteki eğri regresyon çalışmalarındaki regresyon çizgisi olarak düşünülebilir. Basit lojistik regresyonda bir adet bağımsız değişken bulunur. Bu değişkenin olasılığı aşağıdaki formülle gösterilebilir.

$$P = 1/\left(1 + e^{(\beta_0 + \beta_1 \times x)}\right)$$

Verilen olasılık fonksiyonu içerisindeki katsayıların tahmin edilmesi ile lojistik regresyon tahminleri ortaya koyulur. Sonuç olarak aşağıdaki grafikte doğrusal regresyon ile lojistik regresyonun farklılaştığı durum gösterilmektedir. Grafiğe göre doğrusal regresyon çizgisinin kesikli değişkenlerde etkisiz kaldığı görülebilir. Lojistik model ise kesikli değişkenler için bir imkân sunmaktadır. Bu nedenle lojistik regresyon modelleri sınıflandırma problemleri için kullanılabilirdir.



Şekil 9-4: Lojistik regresyon ve doğrusal regresyon farkı

Python ile Örnek Lojistik Regresyon Çalışması

Bu örnekte cinsiyet, yaş ve maaş değişkenlerinin banka müşterisinin kredi alıp almadığını tespit etmekte kullanıldığını düşünülebilir. Örnek veri seti şu adresten indirilebilir: *https://drive.google.com/file/d/1gzsFVX2ucs9u7b3HMbGaDZBrru5rMFrB/view?usp=sharing*

```
dataset = pd.read_excel("kredi_veriseti.xlsx")
dataset.head()
```

	cinsiyet	yas	maas	kredi
0	Erkek	22	2200	0
1	Erkek	38	2300	0
2	Kadın	29	4600	0
3	Kadın	30	6000	0
4	Erkek	22	7900	0

```
#one hot encoding kategorik değişkenleri 0-1 değişkene dönüştürür.
from sklearn.preprocessing import OneHotEncoder
one_hot = OneHotEncoder(handle_unknown='ignore')
one_hot_cinsiyet = one_hot.fit_transform(dataset['cinsiyet'].values.
reshape(
    -1, 1)).toarray()
```

```
# Kategoriler
one_hot.categories_
```

```
[array(['Erkek', 'Kadın'], dtype=object)]
```

```
# yeni veri seti
one_hot_df = pd.DataFrame(one_hot_cinsiyet, columns=one_hot.categories_)
one_hot_df.head()
```

	Erkek	Kadın
0	1.0	0.0
1	1.0	0.0
2	0.0	1.0
3	0.0	1.0
4	1.0	0.0

```
# iki tablonun birleştirilmesi
one_hot_df2 = dataset.join(one_hot_df)
one_hot_df2.head()
```

cinsiyet	yas	maas	kredi	(Erkek,)	(Kadın,)	
0	Erkek	22	2200	0	1.0	0.0
1	Erkek	38	2300	0	1.0	0.0
2	Kadın	29	4600	0	0.0	1.0
3	Kadın	30	6000	0	0.0	1.0
4	Erkek	22	7900	0	1.0	0.0

```
# cinsiyet düşürülebilir.
one_hot_df2.drop('cinsiyet', axis=1, inplace=True)
one_hot_df2.head()
```

	yas	maas	kredi	(Erkek,)	(Kadın,)
0	22	2200	0	1.0	0.0
1	38	2300	0	1.0	0.0
2	29	4600	0	0.0	1.0
3	30	6000	0	0.0	1.0
4	22	7900	0	1.0	0.0

```
# one-hot işlemi pandas getdummies fonksiyonu ile de yapılabilir.
df_getdummy = pd.get_dummies(dataset, columns=['cinsiyet'])
```

```
# x ve y değişkenleri
X = one_hot_df2.drop('kredi', axis=1)
y = one_hot_df2['kredi']
```

```
# train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.33,
                                                    random_state=42)
```

```
# ölçeklendirme
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

```
# modelleme
model = LogisticRegression(random_state=0)
# Grid Search CV sklearn içerisinde parametre optimizasyonu için fonksiyon var.

model.fit(X_train, y_train)
```

```
# tahmin sonuçları
y_pred = model.predict(X_test)
```

```
# parametre tahminleri
print(model.intercept_, model.coef_)
```

```
[-1.13035642] [[ 1.84356965 1.09078635 0.03543114 -0.03543114]]
```

```
y_pred
```

```
array([0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0],
      dtype=int64)
```

```
y_test.values
```

```
array([0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0],
      dtype=int64)
```

```
# performans değerlendirmesi
confusion_matrix(y_true=y_test, y_pred=y_pred)
```

```
array([[78,  2],
       [19, 33]], dtype=int64)
```

```
Doğruların sayısı 78+33 = 111 yanlışların sayısı 2+19 = 21

accuracy_score(y_true=y_test, y_pred=y_pred)
```

```
0.8409090909090909
```

```
print(classification_report(y_true=y_test, y_pred=y_pred))
```

	precision	recall	f1-score	support
0	0.80	0.97	0.88	80
1	0.94	0.63	0.76	52
accuracy			0.84	132
macro avg	0.87	0.80	0.82	132
weighted avg	0.86	0.84	0.83	132

Precision(Kesinlik) — Tahminlerinizin yüzde kaçı doğru? Kesinlik, bir sınıflandırıcının aslında negatif olan bir örneği pozitif olarak etiketlememe yeteneğidir. Her sınıf için, gerçek pozitiflerin gerçek pozitif ve yanlış pozitiflerin toplamına oranı olarak tanımlanır.

Precision = TP/(TP + FP)

Recall — Pozitif durumların yüzde kaçı tahmin edildi?

Recall = TP/(TP+FN)

F1 score — Pozitif tahminlerin yüzde kaçı doğru?

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

Support Destek, belirtilen veri kümesindeki sınıfın gerçek durumlarının sayısıdır. Eğitim verilerindeki dengesiz destek, sınıflandırıcının rapor edilen puanlarındaki yapısal zayıflıkları gösterebilir ve örnekleme veya yeniden dengeleme ihtiyacını gösterebilir. Destek, modeller arasında değişmez, bunun yerine değerlendirme sürecini teşhis eder. Ölçütler ile ilgili detaylı bilgi şu adresten alınabilir: *https://www.scikit-yb.org/en/latest/api/classifier/classification_report.html*

Özet

Bu bölümde doğrusal regresyondan sonra sınıflandırma problemleri için geliştirilen lojistik regresyondan ve lojistik regresyondaki hesaplama detaylarından bahsedilmiştir. Lojistik fonksiyonu aslında yapay sinir ağlarında da kullanılan matematikteki önemli fonksiyonlardandır. Bu bölümde bu fonksiyonun önemi ortaya konmuş ve lojistik regresyon kullanımı gösterilmiştir. Ardından geliştirilen Python kodları ile bir uygulama üzerinde lojistik regresyon kullanılmıştır.

Kaynaklar

1. Wright, R. E. “Logistic regression.” (1995).
2. Hosmer Jr, D. W., Lemeshow, S., and Sturdivant, R. X. Applied Logistic Regression, John Wiley & Sons (2013).
3. Menard, S. Applied Logistic Regression Analysis, Sage (2002).

BÖLÜM 10

KARAR AĞAÇLARI İLE SINIFLANDIRMA

Bu bölümde incelenecek olan **karar ağaçları**, sınıflandırma algoritmaları arasında sıklıkla tercih edilen algoritmalardandır. Karar ağaçlarının kullanımı, okuması ve analizi diğer algoritmalara göre daha kolaydır. Bu nedenle karar ağaçları makine öğrenmesinde akla gelen ilk algoritmalardandır. Bu bölümde karar ağaçları konusunda geçen kavramlardan ve yapılardan bahsedildikten sonra bir uygulama üzerinde karar ağaçlarının Python programlama dili ile nasıl çalıştırılabileceği gösterilecektir.

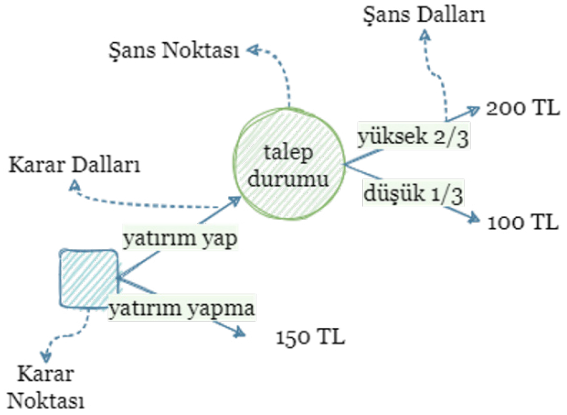
Karar Ağaçları

Karar ağaçları adından da anlaşılabileceği gibi ağaç şeklinde düşünülebilecek bir yapıya sahiptir. Yukarıdan aşağıya doğru bir okuma ile dallar ve düğümler olarak değerlendirilebilir. Dallar okunurken **if-else kuralları** işletilir. Düğümlerde bir özellik gösterilir. Düğümlere diğer bir tabirle **kriterler** denilebilir. Karar ağaçları ile temsil edilen yöntemlere **ağaç tabanlı öğrenme yöntemleri** denilir. Gözetimli öğrenme algoritmaları içerisinde yer verilen algoritmalar arasında ağaç tabanlı öğrenme algoritmaları önemli bir yer kaplar. Regresyon ya da sınıflandırma problemleri çözümünde karar ağaçları kullanılabilir.

Karar ağaçları sadece makine öğrenmesinde değil aynı zamanda karar vermede kullanılan bir yöntem olarak da bilinir. Karar ağaçları ile karar alma süreçlere aşağıdaki örnekler verilebilir.

- Decision Tree Analysis - Decision Skills from MindTools.com
- Decision Trees for Decision Making (hbr.org)
- What is a Decision Tree? How to do Decision Tree Analysis | Gliffy

Karar vermede kullanılan karar ağaçlarında karar noktaları ve karar dalları bulunur. **Karar noktaları** kareler ile karar noktaları ise oklar ile gösterilir. Diğer düğüm noktası ise **şans noktaları**dır. Daireler ile gösterilir ve bu düğümler belirli olasılıklar ile gerçekleşmesi beklenen olayları gösterir. Şans noktalarından çıkan oklara ise **şans dalları** denilir. Aşağıda karar vermede kullanılan bir karar ağacı gösterilmiştir. Örnek karar ağacında yatırım yapma ve yapmama durumları gösterilmiştir. Eğer kişi yatırım yapma kararını alırsa şans noktasında talebin yüksek olup olmaması durumu ortaya çıkar. Talebin yüksek olma ihtimali 2/3 iken düşük olma ihtimali 1/3 olarak verilmiştir. Eğer talep yüksek olursa yatırım yapılması durumunda yatırımcının parası 200 TL'ye çıkmaktadır. Eğer yatırımcı yatırım yapar ve talep düşük olursa yatırımcının parası 100 TL'ye düşmektedir. Bu durumda şans noktasındaki değerler hesaplanırsa $200 \times \frac{2}{3} + 100 \times \frac{1}{3} = \frac{500}{3}$ olur. Yani yatırım yapılması durumunda beklenen paranın yaklaşık 167 TL olmasını beklenir. Yatırım yapılmaması durumunda 150 TL'lik bir gelir beklenildiği için yatırım yapma kararının daha mantıklı olduğu sonucuna ulaşılır.

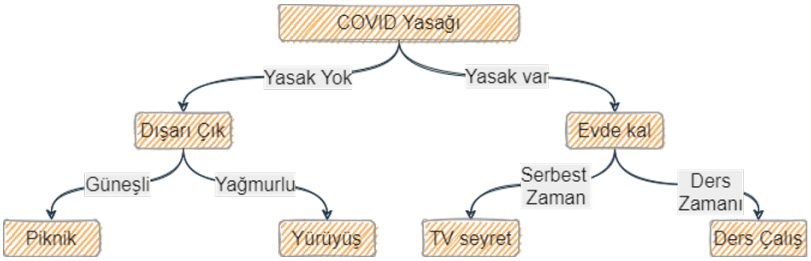


Şekil 10-1: Örnek bir karar ağacı

Örnekte gösterildiği gibi karar ağaçlarındaki temel mantık **mümkün** tüm durumların karar ağacında gösterilmesidir. Hiyerarşik bir yapıda karar verme süreçleri ve noktaları karar ağaçları üzerinde okunabilir veya gösterilebilir.

Aşağıdaki karar ağacında ise sadece mümkün durumlar gösterilmiş ve durumlara göre alınacak kararlar belirtilmiştir. Karar ağacında COVID yasaklarına göre verilecek kararlar gösterilmiştir. Bu kararlar dışsal faktörlere bağlıdır. Eğer COVID yasakları varsa evde kalma kararı verilecek, yoksa dışarı

çıkma kararı verilecek. Ardından eğer hava güneşli ise piknik kararı verilecek, yağmurlu ise yürüyüş kararı verilecektir.



Şekil 10-2: Dışarı çıkma kararı karar ağacı örneği

Şekillerde gösterildiği gibi karar ağaçlarını oluşturmak ve okumak kolaydır. Bu şekilde alınan kararlara **karar ağacı ile alınmış kararlar** denilir. Örneklerde görüldüğü gibi karar ağaçlarında değişkenler kategorik veya sürekli değişkenler olabilir. Değişkenler kategorik değişkense karar ağaçları kategori sayısı kadar bölünür. Eğer kategoriler Evet, Hayır gibi ikili değişkense karar ağacı iki daldan oluşur. Ya da veriler sürekli veriler ise bu durumda bir **bölünme noktası** bulunarak karar ağaçları çizilebilir.

Makine Öğrenmesinde Karar Ağaçları

Daha önce belirtildiği gibi karar ağaçları hem sınıflandırma hem de regresyon için bir çözüm sunan **parametrik olmayan** bir sınıflandırma algoritmasıdır. Karar ağaçları, sınıflandırma problemleri için sınıflandırma ağaçları regresyon problemleri için regresyon ağaçları ile isimlendirilir. Ancak genel olarak bu isimlendirme **Classification And Regression Trees (CART)** olarak yapılır.[1] CART ile önceki bölümde bahsedilen karar ağaçları yapısı ile bir öğrenme gerçekleştirilir.

Karar ağaçları oluşturmada kullanılan bazı CART algoritmaları aşağıdaki gibi verilebilir. Verilen bilgilerle ilgili daha detaylı bilgi Scikit-learn kütüphanesinin şu adresteki dokümantasyonundan alınabilir: *https://scikit-learn.org/stable/modules/tree.html#tree-algorithms-id3-c4-5-c5-0-and-cart*

- **ID3 (Iterative Dichotomiser 3):** Algoritma, her düğümdeki kategorik hedefler için en büyük bilgi kazancını sağlayacak kategorik özelliği bularak çok yollu bir ağaç oluşturur. Ağaçlar maksimum boyutlarına kadar büyütülür ve ardından ağacın verilmemiş verilere genelleme yapma yeteneğini geliştirmek için genellikle bir budama adımı ağaca uygulanır.[2]

- **C4.5:** ID3'ün sonraki versiyonudur. C4.5 sürekli özellik değerini kesikli bir aralık kümesine bölümleyen ayrı bir özelliği (sayısal değişkenlere dayalı) dinamik olarak tanımlayarak özelliklerin kategorik olması gerektiren kısıtlamayı kaldırmıştır. C4.5, eğitilen ağaçları (yani ID3 algoritmasının çıktısını) **if-then** kuralları kümesine dönüştürür.
- **C5.0:** Bu tip algoritmaların en son sürümüdür. Daha az bellek kullanır ve daha doğru sınıflandırmalar için C4.5'ten daha küçük kural kümesi oluşturur.

Golf Örneği

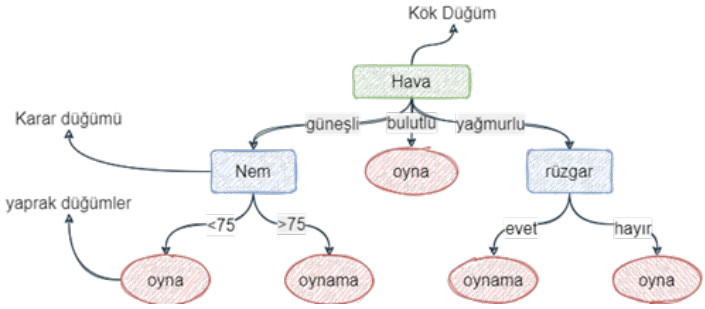
Golf veri seti örneği birçok araştırmacının kullandığı, ilk kez Quinlan tarafından dışarıya çıkıp golf oynayıp oynamama kararı ile ilgili bir ver seti örneğidir.[3] Golf oynamayı düşünen kişi hava durumunun güneşli, bulutlu ve yağmurlu olma durumlarına bakıyor. Kişi, hava durumu ile birlikte sıcaklık derecesine nem oranına ve rüzgar olup olmama durumlarına göre golf oynayıp oynamayacağına karar veriyor. Aşağıdaki tabloda örneğe ilişkin veri seti verilmiştir.

hava	sıcaklık C	nem (%)	rüzgar	karar
güneşli	23,9	70	evet	oyna
güneşli	26,7	90	evet	oynama
güneşli	29,4	85	hayır	oynama
güneşli	22,2	95	hayır	oynama
güneşli	20,6	70	hayır	oyna
bulutlu	22,2	90	evet	oyna
bulutlu	28,3	78	hayır	oyna
bulutlu	17,8	65	evet	oyna
bulutlu	27,2	75	hayır	oyna
yağmurlu	21,7	80	evet	oynama
yağmurlu	18,3	70	evet	oynama
yağmurlu	23,9	80	hayır	oyna
yağmurlu	20,0	80	hayır	oyna
yağmurlu	21,1	96	hayır	oyna

Tablo 10-1: Golf oynama veri seti

Bu veri setine göre hareket etmesi gereken bir kişi örneğin bir sonraki seferde golf oynayıp oynamaması gerektiğine karar verirken 4 farklı faktöre bakacaktır. *Peki makine öğrenmesi ile bu veri setine göre bir öğrenme gerçekleştirilebilir mi?* Bu sorunun cevabı karar ağaçları ile verilebilir. Karar ağaçlar durumları analiz ederek **if-else** kuralları ile bir yapı ortaya koyar. Ardından daha önce görülmemiş bir veri verildiğinde hangi kararın alınması gerektiğini söyler.

Örnek veri setine bakılacak olunursa ilk özellik olan hava özelliği eğer güneşli ise kimi zaman oyna (2/5 oranında) kimi zaman oynama (3/5) kararının çıktığı görülür. Bu nedenle ikinci özellik olan sıcaklık özelliğine de bakılması gereklidir. Ya da nem özelliğine bakmak gereklidir. Nem özelliğine bakıldığında şu seçeneklerden söz edilebilir. {güneşli, 70 → Oyna}, {güneşli, 90→oynama}, {güneşli, 85→oynama}, {güneşli, 95→oynama}, {güneşli, 70→oyna}. Bu durumda kural olarak güneşli havada nem yüzdesi 75'ten küçükse oyna büyükse oynama kararı çıkabilir. Hava özelliğinin bulutlu değerini alması durumunda ise oyna kararı verildiği görülebilir. Eğer hava durumu yağmurlu ise rüzgar varsa oynama, yoksa oyna kararının verildiği görülebilir. Bu durumlar karar ağaçlarına dökülürse aşağıdaki gibi bir yapı kurulabilir.



Şekil 10-3: Golf örneği için karar ağaçları

Örnekte sıcaklık değişkeni ele alınmamıştır. Kimi zaman okuması ve değerlendirmesi kolaylık sunsun diye bazı özellikler alınmayabilir. Karar ağaçları hava durumunun güneşli, bulutlu, yağmurlu olması gibi kategorik değişkenlerle çalıştığı gibi nem gibi sürekli değişkenler ile de çalışabilir. Sürekli değişkenlerde **dallanma**nın, yani bir sonraki aşamaya geçmenin nereden itibaren olacağı ile ilgili çalışma yapılması gereklidir. Sürekli verilerde dallanmalara yapılırken veri setindeki tüm değerlerin alındığından emin olunmalıdır. Kategorik değişkenlerde ise dallanmanın neye göre olacağı bellidir. Örneğin hava değişkeni için güneşli, bulutlu ve yağmurlu olarak dallanma gerçekleştirilmiştir.

Karar ağaçları oluşturulduktan sonra karar ağaçlarındaki yapı kullanılarak **tahmin işlemi** gerçekleştirilebilir. Örneğin yeni bir günde bulutlu hava durumu olması durumunda karar direkt olarak oyna olacağı için golf oynama kararı verilecektir. Ya da yağmurlu bir günde rüzgar yoksa oyna kararı yine verilecektir. Karar ağaçlarının en önemli özelliklerinden birisi de çıkarım yapma yeteneğinin olmasıdır. Gözetimli öğrenmede makinenin insan gibi öğrenmesini

sağlama çalışmalar gerçekleştirilir, denilmişti. Bir insan nasıl ki bir çıkarım yaparken zihninde birtakım kurallar çalıştırıyorsa bu kuralları makinelere de yaptırma çalışmaları makine öğrenmesi için önemli çalışmalardır. Örneğin bütün kedilerin 4 ayağı vardır, Kıpır da bir kedidir; öyleyse Kıpır'ın da 4 ayağı vardır, demek bir çıkarım yapmaktır. Ya da tüm kediler balık sever, Karabaş da balık sever; öyleyse Karabaş da kedidir, diye bir çıkarım yapılabilir. Bu durumda yanlış bir çıkarım yapılmış olunur. Çünkü balık sevmek kedi olmanın tek ve yeter şartı değildir. Karabaş'ın kedi olmadığının ortaya koyulması için başka özelliklere de ihtiyaç vardır. **Genelleme** yaparak bazen hata yapılabilir. Bir genelleme örneği olarak 100 kedinin gözlemlendiği düşünülsün. 100 kedinin tamamına da tavuk maması verilsin. 100 kedi de tavuk mamasını yerse daha önce gözlem yapılmamış kedinin de tavuk maması yiyeceği varsayılabilir. İşte karar ağaçlarındaki öğrenme de bu şekildedir. Tabii ki yeni gelecek olan kedinin tavuk maması yememe ihtimali de vardır. Ancak öğrenme algoritmasındaki hiçbir kedinin tavuk mamasını sevmemezlik yapmadığı için genelleme yaparak tüm kedilerin tavuk maması seveceğini söylemek gereklidir. Bu söylem hatalar içerebilir.

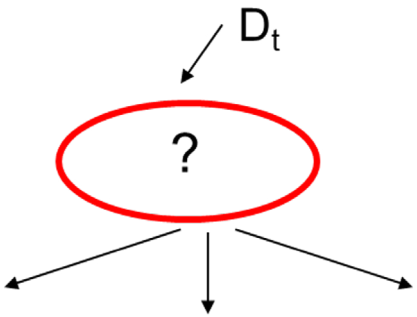
Golf örneğinde karar ağaçları oluşturulurken soldan başlanarak ilk önce hava durumu özelliği alınmıştı. Ardından rastgele başka bir özelliğe bakılarak devam edilmişti. Burada karar ağaçlarındaki en önemli sorun ortaya çıkmaktadır. Bu sorun da hangi özelliğin seçilmesi gerektiği sorunudur. Bu sorunun çözümü için rastgele çözümler yerine literatürde daha geçerli bilimsel yöntemler geliştirilmiştir. Kullanılan yönteme göre oluşturulan karar ağaçları da değişkenlik gösterecektir. Karar ağaçları arasında **performans kıyaslaması** yapılarak en iyi karar ağacı modeli tespit edilebilir. Örnekte en önemli karar kuralının hava durumunun bulutlu olması olduğu görülebilir. Çünkü bulutlu hava durumunda direkt olarak karar verilebiliyor. Bu gibi özelliklerin ortaya koyularak dallanma yapılacak olan düğümlerin belirlenmesi gereklidir. Dallanma düğümlerinin belirlenmesi için geliştirilen yöntemlerden birincisi **entropi** yöntemidir. Dallanma kriterleri ile ilgili aşağıdaki gruplandırma yapılabilir.

a. **Entropiye Dayalı Algoritmalar**
b. **Sınıflandırma ve Regresyon ağaçları**
c. **Bellek Tabanlı Sınıflandırma Algoritmaları**

Hunt Algoritması

Karar ağacı oluşturmada kullanılan algoritmalardandır. Pang-Ning Tan ve ark.[4] tarafından 2005 yılında geliştirilmiştir. Kısaca bahsedilmesi gerekirse, D bir takım özelliklerden ve özelliklere ait değerlerden oluşan bir veri seti olsun. A ise özellikleri ve test kriterini tutan bir liste olsun. A listesi örneğin yaş değişkeni için "**yaş>50**" gibi kriterleri tutan bir liste olacaktır. "**yaş**" özelliği "**>50**" ise kriteri göstermektedir. Hunt algoritması aşağıdaki adımları gerçekleştirir:

- N adında bir düğüm oluştur. Bu düğümdeki veri seti $\mathbf{D_N}$ olur. Başlangıçta bu veri seti tüm veri setidir.
- Eğer $\mathbf{D_t}$, yani bir özelliğin değerlerinin tamamı bir sınıfa aitse bu düğümü yaprak düğüm yap.



Şekil 10-4: Hunt Algoritması D_t

- Eğer A listesi boşsa $\mathbf{D_t}$ listesinde eleman varsa bu düğümü de yaprak düğüm yap.
- Eğer Dt'nin içerisindeki değerlerden birden fazla sınıfa ait olan veriler varsa burada özellik seçim metodunu kullan ve bir sonraki özellik için olabilecek en iyi özelliği belirle. Belirlenen en iyi özelliği kullanıldığı için A listesinden çıkar.
- Bu adımları tüm veriler açıklanıncaya kadar devam ettir.

Hunt algoritmasında görüldüğü gibi **iki** önemli durum vardır: Bunlardan birincisi en iyi özelliğin **nasıl** seçileceği sorusudur. İkincisi ise kriterin **belirlenmesi** sorusudur. Eğer özellik ikili bir özellikse örneğin Evet, Hayır gibi sadece iki değer alabiliyorsa bu durumda kriter oluşturma işlemi sadece evetse bir dal, hayırsa bir dal şeklinde yapılır. Eğer özellikte ikiden fazla değer varsa, yani özellik nominal ve ordinal bir özellikse bu durumda çok noktalı bölmeler de yapılabilir. Ya da çok noktalı bölmeler ikili hâlde de bölünebilir. Bu durumda

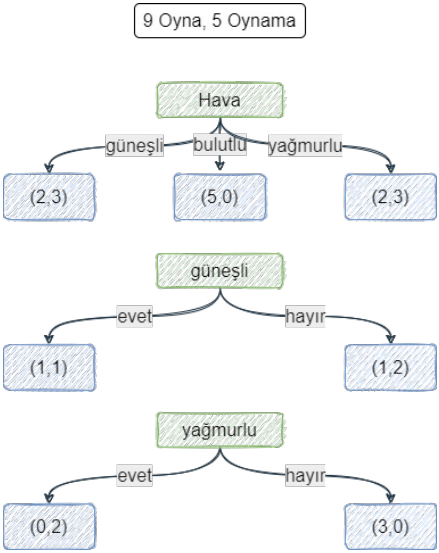
örneğin 3 kategorisi olan bir nominal değişken için 3 adet farklı **ikili bölmeler** söz konusu olacaktır. Eğer değişken sürekli veri tipinde ise bu durumda "<" ">" gibi eşitsizlikler kullanılarak aralıklar belirlenmeye çalışılır. İkinci soru olan en iyi özelliğin belirlenmesi için ise çeşitli algoritmalar geliştirilmiştir. Algoritmaların temelinde dallanmadan önceki durum ile sonraki durumun kıyaslanması vardır. Bu kıyas sonucunda daha iyi bir durum daha saf bir duruma geçilirse algoritma geçişi onaylar, geçilmezse geçiş onaylanmaz. Bu konu ile ilgili detaylar ilerleyen kısımlarda verilecektir.

Tekrar golf oynama örneği ele alınarak gösterilecek olursa, takip kolaylığı açısından hava durumu, rüzgar ve karar değişkenlerinin verilmesi iyi olacaktır.

hava	**rüzgar**	**karar**
güneşli	evet	oyna
güneşli	evet	oynama
güneşli	hayır	oynama
güneşli	hayır	oynama
güneşli	hayır	oyna
bulutlu	evet	oyna
bulutlu	hayır	oyna
bulutlu	evet	oyna
bulutlu	hayır	oyna
yağmurlu	evet	oynama
yağmurlu	evet	oynama
yağmurlu	hayır	oyna
yağmurlu	hayır	oyna
yağmurlu	hayır	oyna

Tablo 10-2: Golf oynama veri seti

Hunt algoritması öncelikle oyna ve oynama sayılarının hesaplanması ile başlar. Veri setinde 9 adet oyna, 5 adet oynama kararı vardır. Bu nedenle bir dallanma gerekmektedir. Dallanma yapılacak özellik olarak hava durumu seçilirse, güneşli hava durumunda 2 oyna, 3 oynama kararı bulutlu özelliğinde 5 oyna 0 oynama, yağmurlu özelliğinde 2 oyna 3 oynama kararı olduğu görülecektir. Bu durumda güneşli ve yağmurlu değerlerinde tekrar bir dallanmaya gitmek gerekir. Bu dallanma da rüzgar evet, hayır için yapılırsa aşağıdaki gibi bir karar ağacına ulaşılır:



Şekil 10-5: Oluşturulan karar ağaçları

Karar Ağaçları Algoritmaları

Karar ağaçları yardımıyla sınıflandırma işlemlerini yerine getirmek üzere Quinlan tarafından birçok algoritma geliştirilmiştir. Bu algoritmalarda **yukarıdan-aşağı** (top-down: kökten alt dallara doğru) ve **greedy search** (sonuca en yakın durum) teknikleri kullanılır. Bunlardan birisi olan ID3 algoritması, entropi ve bilgi kazanımı (Information Gain) üzerine inşa edilmiştir. Burada ID3 algoritmasındaki kavramlardan bahsedilmiş ve ilgili hesaplamalar gösterilmiştir.

Entropi

Entropi, bir sistemdeki belirsizlik ölçüsü olarak bilinir. Olasılık dağılımına sahip mesajları üreten S kaynağının entropisi şu şekildedir:

$$P = P_1, P_2, \ldots, P_n$$

$$H(S) = -\sum_{i=1}^{n} p_i log_2 p_i$$

H(S) burada entropi değerini gösterir. Yani bir veri kümesinde düzen varsa entropi değeri 0 olur. Ne kadar fazla düzensizlik varsa entropi değeri o kadar yüksek olur. Örneğin S=a_1 ,a_2 ,a_3 deney kümesini ifade etsin. a_1 ,a_2 ,a_3 olayları

sırasıyla ½, 1/3 ve 1/6 olma olasılıklarına sahipse bu olaylarının belirsizlikleri şöyle hesaplanır:

$$H(S) = -1/2log_2 1/2, -1/3log_2 1/3, -1/6log_2 1/6 = 1{,}4591$$

Entropi hesaplamaları yapılabilmesi için logaritma alma kurallarının bilinmesi gereklidir. Kısaca logaritma alma kurallarından bahsetmek gerekirse,

Çarpım kuralı: $log_b(x \times y) = log_b(x) + log_b(y)$

Bölme kuralı: $log_b(x/y) = log_b(x) - log_b(y)$

Üs alma kuralı: $log_b(x^y) = y \times log_b(x)$

Taban değiştirme kuralı: $log_b(c) = 1/log_c(b)$

Logaritma değiştirme kuralı: $log_b(x) = log_c(x) / log_c(b)$

Entropi hesabı Windows Bilimsel Hesap Makinesi ile hesaplanmak istenirse hesap makinesinde 2 tabanında logaritma olmadığı için logaritma değiştirme kuralından yararlanılarak **ln** fonksiyonu kullanılabilir. Örneğin $log_2(1/2) \rightarrow ln(1/2)/ln2$ olarak yazılabilir. Yani formül aşağıdaki gibi olur:

$$log_2(x) = ln(x)/ln(2)$$

Şekil 10-6: Windows hesap makinesi ile entropi hesaplama

Benzer hesaplama Python programlama dilinde de yapılabilir. Bunun için **math** kütüphanesinin çağrılması gerekmektedir.

```
from math import log
log?
```

Docstring:
log(x, [base=math.e]) Return the logarithm of x to the given base. If the base not specified, returns the natural logarithm (base e) of x.

Type: builtin_function_or_method

```
# Örnek logaritma 2 tabanında 4
log(4,2)
```

2.0

Entropi hesaplamasına bir diğer örnek ise şu şekilde verilebilir. Aşağıda sekiz elemanlı S kümesini göz önüne alınsın. $S=\{evet,evet,hayır,hayır,hayır,hayır,hayır,hayır\}$

Olasılıklar, iki adet ''evet'' değeri için,

$$P_1 = 2/8 = 0{,}25$$

Diğer altı adet ''hayır'' değeri için,

$$P_2 = 6/8 = 0{,}75$$

S için toplam entropi şu şekilde elde edilir:

$$H(S) = -2/8log_2 2/8 - 6/8log_2 6/8 = 0{,}81128$$

Yukarıdaki hesaplamanın Python kodları aşağıda verilmiştir:

```
(-2/8)*log((2/8),2)-((6/8)*log((6/8),2))
```

0.8112781244591328

Entropi ile Dallanmaya Başlanacak Özelliklerinin Belirlenmesi

Karar ağaçlarının oluşturulması esnasında dallanmaya hangi nitelikten başlanacağı önem taşımaktadır denilmişti. Çünkü sınırlı sayıda kayıttan oluşan bir eğitim kümesinden yararlanarak olası tüm ağaç yapılarını ortaya çıkarmak ve içlerinden en uygun olanı seçerek ondan başlamak kolay değildir. Bu sorunun çözümü için entropi değerleri kullanılabilir. Şimdi tekrar golf oynama örneğini ele alarak entropi hesaplamaları gösterilecektir. Golf örneğindeki veri seti ID3 algoritmasının kategorik veriler ile çalışması nedeniyle aşağıdaki değiştirilmiştir.

hava	sıcaklık C	nem (%)	rüzgar	karar
güneşli	ılık	düşük	evet	oyna
güneşli	sıcak	yüksek	evet	oynama
güneşli	sıcak	orta	hayır	oynama
güneşli	ılık	yüksek	hayır	oynama
güneşli	soğuk	düşük	hayır	oyna
bulutlu	ılık	yüksek	evet	oyna
bulutlu	sıcak	düşük	hayır	oyna
bulutlu	soğuk	düşük	evet	oyna
bulutlu	sıcak	düşük	hayır	oyna
yağmurlu	soğuk	orta	evet	oynama
yağmurlu	soğuk	düşük	evet	oynama
yağmurlu	ılık	orta	hayır	oyna
yağmurlu	soğuk	orta	hayır	oyna
yağmurlu	soğuk	yüksek	hayır	oyna

Tablo 10-3: Kategorik golf oynama veri seti

Karar ağacının dallanmasına hava değişkeninden başlanırsa başka bir karar ağacı sıcaklık değişkeninden başlanırsa başka bir karar ağacı ortaya çıkacaktır. *Peki hangi karar ağacı daha iyi sonuçlar verecektir?* Daha iyi sonuçtan kasıt karar ağacının **olabildiğince az** sayıda düğüm ve daldan oluşmasını sağlamaktır. Bunun kararını verebilmek için entropi hesaplamaları yapılması gerekir. Entropi hesabı özellikler için de yapılabilir. Özellikler için kullanılacak olan entropi formülü aşağıda verilmiştir.[5]

$$E(T,X) = \sum_{c \in X} P(c)E(c)$$

Burada C, belirli bir düğümdeki verilerin sınıflarının sayısını P ise olasılık anlamına gelir. E(T,X), T seviyesinde X koşulu uygulandıktan sonra verilerin entropi değerini temsil eder. Başlangıç durumunda karar sınıfında 2 değer bulunmaktadır. Oyna kararı için 9 örnek varken oynamama kararı için 5 örnek bulunmaktadır. Bu durumda başlangıç entropisi aşağıdaki gibi hesaplanır.

$$E_{başlangıç} = -\left(\frac{5}{14}\right) \times \log_2\left(\frac{5}{14}\right) - \left(\frac{9}{14}\right) \times \log_2\left(\frac{9}{14}\right) = 0{,}94$$

Başlangıç entropisi hesaplandıktan sonra her özelliğin aldığı değere göre bir olasılık hesabının yapılması gerekmektedir. Aşağıdaki 3 tabloda hava özelliğinin aldığı değerlere göre karar değişkeninin durumları gösterilmiştir.

hava	karar
güneşli	oyna
güneşli	oynama
güneşli	oynama
güneşli	oynama
güneşli	oyna

hava	karar
bulutlu	oyna
bulutlu	oyna
bulutlu	oyna
bulutlu	oyna

hava	karar
yağmurlu	oynama
yağmurlu	oynama
yağmurlu	oyna
yağmurlu	oyna
yağmurlu	oyna

$$E_{gunesli} = -(2/5)log_2(2/5) - (3/5)log_2(3/5) = -(2/5)(-1{,}32) - (3/5)(-0{,}73) = 0{,}97$$

$$E_{bulutlu} = -(4/4)log_2(4/4) = 0$$

$$E_{yagmurlu} = -(3/5)log_2(3/5) - (2/5)log_2(2/5) = 0{,}97$$

Güneşli ve yağmurlu özelliğinin entropisi eşit ve 0,97 olarak hesaplanmıştır. Bulutlu özelliğinin entropisi ise 0 olarak gelmiştir. Bu durumda hava özelliğinin yeni entropisi hesaplanan özelliklerin ağırlıklı ortalaması olarak gelecektir. Güneşli özelliği 14 özellik içerisinde 5 sefer, bulutlu özelliği 4 sefer, yağmurlu özelliği de 5 sefer geçtiği için ağırlıklı ortalama formülü aşağıdaki gibi yazılabilir.

$$E(oyna, hava) = P(güneşli) * E_{güneşli} + P(bulutlu) * E_{bulutlu} + P(yağmurlu) * E_{yağmurlu}$$

$$E_{hava} = \frac{5}{14} \times E_{gunesli} + \frac{4}{14} \times E_{bulutlu} + \frac{5}{14} \times E_{yağmurlu} = 0{,}6928$$

$E_{başlangıç}$ ile E_{hava} arasındaki fark hava özelliğinin seçilmesiyle kazanılacak olan bilgiyi gösterir. Buna **bilgi kazancı (information gain)** denir. Bilgi kazanımı, bir veri setini bir özellik üzerinde böldükten (Örneğin E(oyna, hava)) sonra tüm entropiden (E(oyna)) çıkarmaya dayanır. Entropinin küçük değer içermesi durumunda özelliğin önemi Decision Tree algoritması ID3 için artmaktadır. Diğer taraftan 1'e yaklaştıkça özelliğinin önemi azalır. Ancak bilgi kazanımında olay tam tersidir ve bu açıdan entropinin tersi gibi düşünülebilir. Karar ağacı inşa edilirken en yüksek değerleri bilgi kazanımına sahip özellik seçilir.

Hava özelliği için kazanç:

$$Information\ Gain = E_{başlangıç} - E_{hava} = 0{,}94 - 0{,}69 = 0{,}25$$

Yani hava özelliğinin seçilmesi durumunda elde edilecek olan bilgi kazancı 0,25 olarak hesaplanmıştır. İşte hangi özelliğin seçileceği ile ilgili karar en fazla bilgi kazancının olduğu duruma göre verilecektir. Ardından tekrarlı bir şekilde son özellik de belirlenene kadar bu süreç devam ettirilir.

Daha pratik bir hesaplama için frekans tablolarından yararlanılabilir. Aşağıdaki gösterimde sıcaklık, nem ve rüzgar özellikleri için frekans tabloları oluşturulmuştur. Frekans tabloları oluşturulduktan sonra her satırdaki değerler logaritma 2 tabanındaki değeri ile çarpılarak birbirinden çıkarılır ve toplam sütunundaki değerlerin entropi ile çarpım değerleri eklenir. Ardından toplam örnek sayısına bölünerek entropi değeri elde edilir.

Hava özelliği için;

	oyna	**oynama**	**toplam**
güneşli	2	3	5
bulutlu	4	0	4
yağmurlu	3	2	5
toplam	9	5	14

$$E_{hava} = \frac{-2\log_2 2 - 3\log_2 3 - 4\log_2 4 - 0\log_2 0 - 3\log_2 3 - 2\log_2 2 + 5\log_2 5 + 4\log_2 4 + 5\log_2 5}{14} = 0{,}69$$

```
(-2*log(2,2)-3*log(3,2)-4*log(4,2)-3*log(3,2)-2*log(2,2)+5*log(5,2)+4*log(
4,2)+5*log(5,2))/14
```

0.69

Sıcaklık Özelliği için;

	oyna	**oynama**	**toplam**
ılık	3	1	4
sıcak	2	2	4
soğuk	4	2	6
toplam	9	5	14

$$E_{sıcaklık} = \frac{-3\log_2 3 - 1\log_2 1 - 2\log_2 2 - 2\log_2 2 - 4\log_2 4 - 2\log_2 2 + 4\log_2 4 + 4\log_2 4 + 6\log_2 6}{14} = 0{,}91$$

Nem Özelliği için;

	oyna	**oynama**	**toplam**
düşük	5	1	6
orta	2	2	4
yüksek	2	2	4
toplam	9	5	14

$$E_{nem} = \frac{-5\log_2 5 - 1\log_2 1 - 2\log_2 2 - 2\log_2 2 - 2\log_2 2 - 2\log_2 2 + 6\log_2 6 + 4\log_2 4 + 4\log_2 4}{14} = 0{,}85$$

Rüzgar Özelliği için;

	oyna	oynama	toplam
evet	3	3	6
hayır	6	2	8
toplam	9	5	14

$$E_{rüzgar} = \frac{-3\log_2 3 - 3\log_2 3 - 6\log_2 6 - 2\log_2 2 + 6\log_2 6 + 8\log_2 8}{14} = 0{,}89$$

Bulunan değerlere göre kazançlar aşağıdaki gibi hesaplanır.

Kazanç(karar, hava) = E(karar) – E(karar, hava)

Kazanç (karar, hava) = 0,94-0,69=0,25 *

Kazanç (karar, sıcaklık) = 0,94-0,91=0,03

Kazanç (karar, nem) = 0,94-0,85=0,09

Kazanç (karar, rüzgar) = 0,94-0,89=0,05

Olduğu için dallanmaya hava özelliğinden başlanır sonucu çıkarılır. Bunun gibi her bir dallanmada ortaya çıkan hesaplamalar ile dallanma yapılacak özellik seçilir ve karar ağaçları oluşturulmuş olunur.

Gini Indeks

Entropi ve bilgi kazanımı ile birlikte kullanılan diğer bir metot ise **Gini indeks** değerlerinin hesaplanması ile oluşturulan metottur. Gini değerleri aşağıdaki formülde gösterildiği gibi hesaplanır.

$$Gini = 1 - \sum_{i=1}^{c} p_i^2$$

Formüldeki c bir gruptaki kategori sayısını göstermektedir. P ise o sınıfın olma olasılığını gösterir. Gini indeks veri setindeki karmaşıklığı gösterir. En düşük sıfır değeri alabilir sıfır değerini aldığında sınıflandırmanın **aynı** olduğu anlamı çıkar. Bu nedenle düşük gini indekse sahip olan özellik seçilir. Alt gruplar için gini indeks hesabı için aşağıdaki formül kullanılır:

$$G_j = 1 - \sum_{i=1}^{K} \left(\frac{f_{ij}}{N_j}\right)^2$$

Aşağıda golf oynama veri seti için gini indeks hesaplamaları gösterilmiştir. Golf oynama durumunda 2 farklı karar verilmişti. Oyna durumu sayısı 9 olduğu için olasılık 9/14, oynamama durumu sayısı 5 olduğu için olasılığı 5/14 olarak verilebilir. Bu durumda gini indeks aşağıdaki gibi hesaplanır.

$$G_{başlangıç} = 1 - \left(\left(\frac{5}{14}\right)^2 + \left(\frac{9}{14}\right)^2\right) = 0{,}4591$$

Hava özelliğinde 3 durum vardır, bu durumların olasılıkları sırasıyla 5/14, 4/14, 5/14 olarak verilmiştir. Frekans tablosu aşağıdaki şekilde verilmiştir.

	oyna	**oynama**	**toplam**
güneşli	2	3	5
bulutlu	4	0	4
yağmurlu	3	2	5
toplam	9	5	14

Adım 1: Her bir özellik için değerlerin karesini alıp topla ve satır sayısına böl.

Güneşli: $\frac{2^2+3^2}{5} = 2{,}6$

Bulutlu: $\frac{4^2+0^2}{4} = 4$

Yağmurlu: $\frac{3^2+2^2}{5} = 2{,}6$

Adım 2: gini indeks formülünü kullanarak hava özelliğinin gini indeks değerini hesapla.

$$G_{hava} = 1 - \frac{2{,}6 + 4 + 2{,}6}{14} = 0{,}34$$

Adım 3: Gini indeksteki azalma miktarını hesapla.

$$= G_{başlangıç} - G_{hava} = 0{,}4591 - 0{,}34 = 0{,}1191$$

Adım 4: Diğer özellikler için de benzer hesaplamalar yapılarak azalma miktarlarındaki en yüksek değere göre bir dallanma özelliği belirle.

Adım 5: Tüm özellikler bitene kadar adımları tekrar ettir.

Golf Oynama Veri Setinin Scikit-Learn ile Modellenmesi

Bu çalışmada Scikit-Learn kütüphanesi kullanılarak golf oynama veri seti modellemesi gerçekleştirilecektir. Verisetindeki özellikler ve özelliklerin kategorileri aşağıda gösterilmiştir.

- hava
  - güneşli
  - bulutlu
  - yağmurlu
- sicaklik:
  - ılık
  - sıcak
  - soğuk
- nem:
  - düşük
  - yüksek
  - orta
- ruzgar:
  - evet
  - hayır

Scikit-Learn içerisindeki karar ağaçları oluşturmada kullanılabilecek olan fonksiyonlar şunlardır.

- **tree.export_text**: Ağaç yapısını metin olarak göstermek için,
- **tree.plot_tree**: Matplotlib kütüphanesi ile ağacı görselleştirmek için,
- **tree.export_graphiz**: Graphiz programı ile daha kaliteli ağaç görselleri almak için,
- **dtreebviz**: Graphviz ve dtreeviz ile ağaç görseli oluşturmak için.

```
import pandas as pd
import numpy as np
from sklearn import tree
```

Veri seti aşağıdaki gibi Excel dosyasından okunmuştur. Veri seti şu adresten indirilebilir: *https://drive.google.com/file/d/1x_UrcZZVrlLNhKOPVelESnyjHE86BIXP/view?usp=sharing*

```
# Veri setindeki ilk 5 satırı aşağıda gösterdik.
veri_seti = pd.read_excel("golf_oynama_veri_seti.xlsx")
veri_seti.head()
```

	hav	sicaklik	nem	ruzgar	karar
0	gunesli	ilik	dusuk	evet	oyna
1	gunesli	sicak	yuksek	evet	oynama
2	gunesli	sicak	orta	hayir	oynama
3	gunesli	ilik	yuksek	hayir	oynama
4	gunesli	soguk	dusuk	hayir	oyna

```
# veri setindeki değişkenlerin veri tipleri
veri_seti.dtypes
```

```
hava        object
sicaklik    object
nem         object
ruzgar      object
karar       object
dtype:      object
```

X ve y Değişkenlerinin Belirlenmesi

Golf oynama kararında karar değişkeni y değişkenlerini diğer sütunlar ise x değişkenini oluşturacaktır.

```
ozellikler = ["hava", "sicaklik", "nem", "ruzgar"]
X = veri_seti.loc[:, ozellikler]
y = veri_seti.loc[:, 'karar']
print(X)
print(y)
```

	hava	sicaklik	nem	ruzgar
0	gunesli	ilik	dusuk	evet
1	gunesli	sicak	yuksek	evet
2	gunesli	sicak	orta	hayir
3	gunesli	ilik	yuksek	hayir
4	gunesli	soguk	dusuk	hayir
5	bulutlu	ilik	yuksek	evet
6	bulutlu	sicak	dusuk	hayir
7	bulutlu	soguk	dusuk	evet

```
8     bulutlu   sicak   dusuk   hayir
9     yagmurlu  soguk   orta    evet
10    yagmurlu  soguk   dusuk   evet
11    yagmurlu  ilik    orta    hayir
12    yagmurlu  soguk   orta    hayir
13    yagmurlu  soguk   yuksek  hayir
0     oyna
1     oynama
2     oynama
3     oynama
4     oyna
5     oyna
6     oyna
7     oyna
8     oyna
9     oynama
10    oynama
11    oyna
12    oyna
13    oyna
Name: karar, dtype: object
```

Verilerin değişken tipleri kategorik olarak görülmektedir. Scikit-Learn kütüphanesi karar ağaçlarını oluştururken nümerik verilere ihtiyaç duymaktadır. Bu nedenle kategorik verilerin nümerik verilere dönüştürülmesi gerekmektedir. Başka programlarda direkt kategorik veriler ile de devam edilebilir. Ancak Scikit-Learn içerisinde karar ağaçlarının kullanımı için nümerik hâle dönüşüm gereklidir.

Kategorik verilerin nümerik verilere dönüştürülmesi için kullanılacak olan yöntem **One-hot encoding** yöntemidir. Bu yönteme göre her bir kategorik değer için yeni bir sütun oluşturulur. Yeni sütundaki değerler 0-1 ikili değerlerini alabilirler. One-hot encoding aşağıdaki gibi yapılabilir:

```
# rüzgar özelliği evet hayır tipinde olduğu için
# evet:1 hayır:0 olarak yeniden kodlanır.
X['ruzgar'] = X['ruzgar'].map({'evet': 1, 'hayir': 0})
```

```
# diğer özellikler onehot encoding ile kodlanır.
X_encoded = pd.get_dummies(data=X, columns=['hava', 'sicaklik', 'nem'])
```

```
# yeniden kodlanan veri seti aşağıdaki gibidir.
X_encoded
```

	ruzgar	hava_ bulutlu	hava_ gunesli	hava_ yagmurlu	sicaklik_ ilik	sicaklik_ sicak	sicaklik_ soguk	nem_ dusuk	nem_ orta	nem_ yuksek
0	1	0	1	0	1	0	0	1	0	0
1	1	0	1	0	0	1	0	0	0	1
2	0	0	1	0	0	1	0	0	1	0
3	0	0	1	0	1	0	0	0	0	1
4	0	0	1	0	0	0	1	1	0	0
5	1	1	0	0	1	0	0	0	0	1
6	0	1	0	0	0	1	0	1	0	0
7	1	1	0	0	0	0	1	1	0	0
8	0	1	0	0	0	1	0	1	0	0
9	1	0	0	1	0	0	1	0	1	0
10	1	0	0	1	0	0	1	1	0	0
11	0	0	0	1	1	0	0	0	1	0
12	0	0	0	1	0	0	1	0	1	0
13	0	0	0	1	0	0	1	0	0	1

```
# Karar özelliğinde oyna:1 oynama:0 olarak yeniden kodlanır.
y_encoded = y.map({'oyna': 1, 'oynama': 0})
```

Yeniden kodlanan veri seti karar ağaçları ile modellenmeye hazır hâle gelmiştir. Tekrar x ve y atanırsa, aşağıdaki gibi karar ağaçları oluşturulabilir.

```
# karar ağacının oluşturulması
karar_agaci = tree.DecisionTreeClassifier(random_state=12345)
karar_agaci = karar_agaci.fit(X_encoded, y_encoded)
```

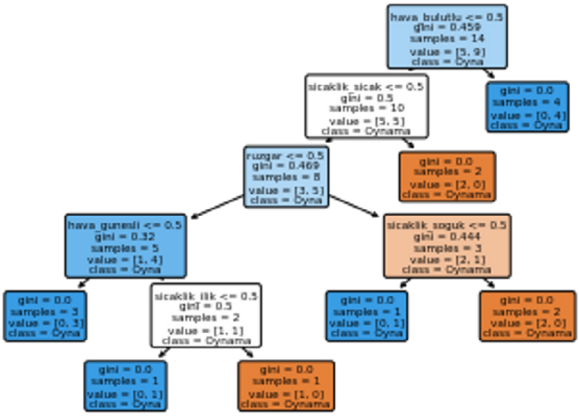
```
# karar ağaçlarını metin şeklinde yazdırmak için
print(tree.export_text(karar_agaci))
```

```
|--- feature_1 <= 0.50
|   |--- feature_5 <= 0.50
|   |   |--- feature_0 <= 0.50
|   |   |   |--- feature_2 <= 0.50
|   |   |   |   |--- class: 1
|   |   |   |--- feature_2 >  0.50
|   |   |   |   |--- feature_4 <= 0.50
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- feature_4 >  0.50
|   |   |   |   |   |--- class: 0
|   |   |--- feature_0 >  0.50
|   |   |   |--- feature_6 <= 0.50
|   |   |   |   |--- class: 1
|   |   |   |--- feature_6 >  0.50
```

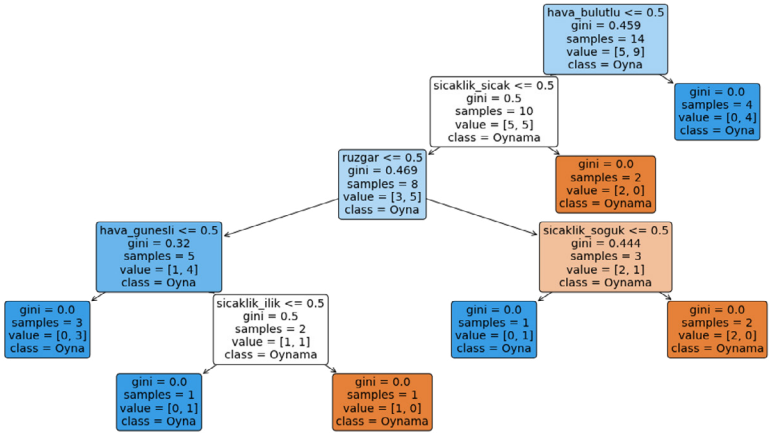
```
|  |  |  |  |--- class: 0
|  |--- feature_5 >  0.50
|  |  |--- class: 0
|--- feature_1 >  0.50
|  |--- class: 1
```

Karar ağaçlarının görselleştirmesi için Scikit-Learn içerisinde Matplotlib ile birlikte çalışan **`plot_tree`** fonksiyonu bulunmaktadır. Bu fonksiyon ile ilgili detaylı bilgi şu adresten alınabilir: *https://scikit-learn.org/stable/modules/generated/sklearn.tree.plot_tree.html*

```
tree.plot_tree(karar_agaci,
               filled=True,
               rounded=True,
               class_names=['Oynama', 'Oyna'],
               feature_names=X_encoded.columns)
```




```
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(10, 5))
_ = tree.plot_tree(karar_agaci,
                   filled=True,
                   rounded=True,
                   class_names=['Oynama', 'Oyna'],
                   feature_names=X_encoded.columns)
```



Karar ağacını okumamız gerekirse şunları söyleyebiliriz. Eğer bulutlu <=0.5 ise yani hava bulutlu değilse sol tarafa, eğer bulutlu ise sağ tarafa gidecek. Bulutlu ise 4 durum söz konusu 4'ü de oyna seçeneğinde olduğu için mavi düğümde gösterildiği gibi oyna seçeneğinin olduğu sonucuna varılır.

Eğer bulutlu değilse yani sola giderse sıcaklık değişkeni sıcak mı diye bakılır. Sıcak ise sağa, yani turuncu düğüme girilir. Turuncu düğümde hava bulutlu değil ve sıcak olması durumunda oynama kararı verilir. Eğer hava bulutlu değil ve sıcak değilse rüzgar özelliğine bakılır.

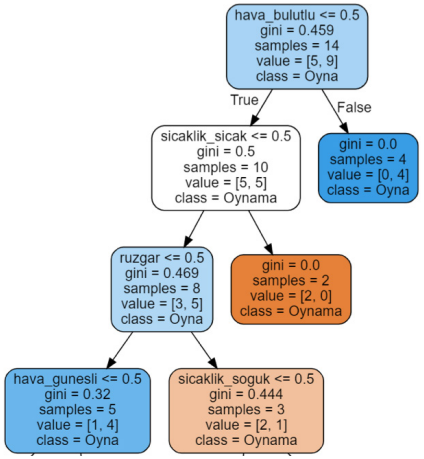
```
# Görseli kayıt edebilmek için,
fig.savefig("karar_agaci.png")
```

Graphviz kütüphanesi ile görselleştirme yapabilmek için öncelikle **`pip install graphviz`** ile kütüphanenin yüklenmesi gerekir. Graphviz kütüphanesine karar ağacının nokta formatı gönderilir. Graphviz hakkında daha detaylı bilgi için şu adres ziyaret edilebilir.

```
import graphviz
```

```
nokta_veriler = tree.export_graphviz(karar_agaci,
                                     filled=True,
                                     rounded=True,
                                     class_names=['Oynama', 'Oyna'],
                                     feature_names=X_encoded.columns)
```

```
grafik = graphviz.Source(nokta_veriler, format='png')
grafik
```

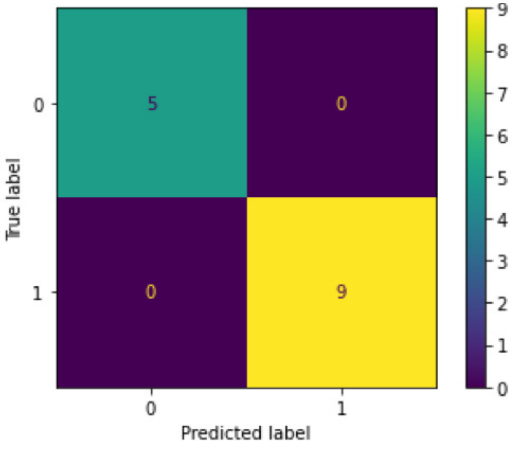


Yapılan sınıflandırmanın kalitesi

```
from sklearn.metrics import plot_confusion_matrix
```

```
plot_confusion_matrix(karar_agaci, X_encoded, y_encoded)
```

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x19357214c50>



Öğrenme gerçekleştikten sonra karar ağacı kullanılarak yeni verilerin tahmin edilmesi işlemi yapılabilir. Bu uygulamada eğitim ve test seti olarak ayrım yapılmamıştır. Tüm veri seti eğitimde kullanılmıştır. Öğrenme gerçekleştikten sonra modele yeni veriler elle girilecektir. Elle girilen verilerin öğrenme gerçekleşirken kullanılan girdi düzeninde olmasına dikkat edilmelidir. Bu düzenin sağlanması için tekrar **`pandas`** **`get_dummies`** fonksiyonunun kullanılması gerekecektir.

```
# Öğrenmeden sonra tahmin değerlerinin oluşturulması
# gunesli  ilik  dusuk  evet

yeni_veri = [1, 0, 1, 0, 1, 0, 0, 1, 0, 0]
karar_agaci.predict([yeni_veri])
```

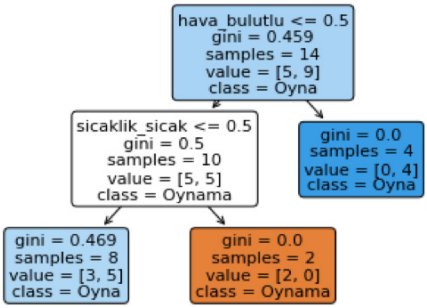
```
array([1], dtype=int64)
```

Karar özelliğinde oyna:1 oynama:0 olarak yeniden kodlanır. y_encoded=y.map({'oyna': 1, 'oynama':0}) olduğu için yeni verilerin oyna kararını vereceği tahmini yapılmıştır.

Karar ağaçlarının dallanma derinliği sınırlanmaz ise **aşırı öğrenme** söz konusu olabilir. Ayrıca fazla derinlikli karar ağaçlarının okunması ve anlaşılması da zor olacaktır. Bu nedenle **`max_depth=2`** parametresi ile derinliği 2 olan bir karar ağacı çizdirmek istenebilir. Aşağıda bununla ilgili bir örnek gösterilmiştir:

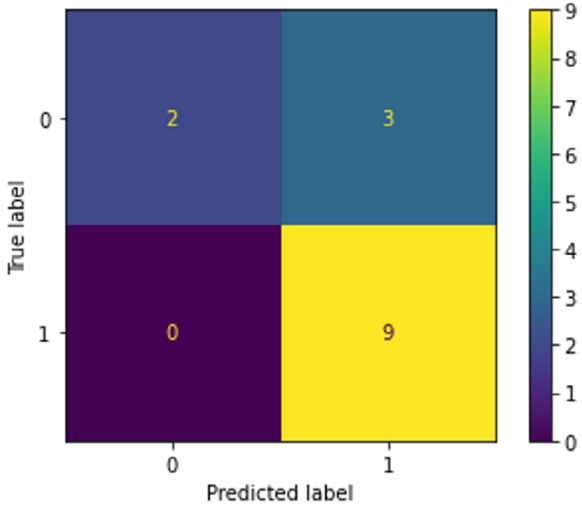
```
# Daha kısa bir karar ağacının oluşturulması
karar_agaci2 = tree.DecisionTreeClassifier(random_state=12345, max_
depth=2)
karar_agaci2 = karar_agaci2.fit(X_encoded, y_encoded)
```

```
tree.plot_tree(karar_agaci2,
               filled=True,
               rounded=True,
               class_names=['Oynama', 'Oyna'],
               feature_names=X_encoded.columns)
```




```
plot_confusion_matrix(karar_agaci2, X_encoded, y_encoded)
```

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x193573a6cc0>



Görüldüğü gibi karar ağacının daha kısa olması durumunda öğrenme oranı düşmüştür. Eğer derinlik sayısı sınırlandırılmaz ise karar ağacı tüm eğitim setini öğrenebilir. Ancak bu durum makine öğrenmesinde **aşırı öğrenme** olarak geçer. Eğitim setinin çok iyi öğrenilmesi durumunda yeni verilere uyum kabiliyeti düşebilir. Bu sorunun üstesinden gelmek için ise makine öğrenmesinde kullanılan aşırı öğrenmeyi engelleyecek yöntemler kullanılmalıdır.

LabelEncoder: Daha önce gösterilen One-Hot Encoding yerine Scikit-Learn içerisindeki **LabelEnconding** özelliği de kullanılarak veriler sayısal hale getirilebilir. `LabelEncoder` kategorik değişkenlerinin her bir değerine ayrı bir

numara verir. Fakat nominal veriler için **LabelEncoder** kullanılması durumunda kategori değerlerine değerler kullanıldığı için yeni bir sorun ortaya çıkar. Nümerik değerler kategoriler hakkındaki ilişkileri gösterir. Örneğin güneşli için 1, bulutlu için 2 denilirse bulutlu değerinin güneşli özelliğinden daha üstün olduğu anlamı çıkabilir. Bu yaklaşım düşük, orta, yüksek gibi kategorik ancak ordinal olan veri tipleri için kullanılabilir. Burada LabelEncoder'ın kullanımı gösterilecek ve yapılan yanlış yoruma da değinilecektir.

```
from sklearn.preprocessing import LabelEncoder
```

```
# tüm özellikler için ayrı birer encoder ayarlanır.
le_hava = LabelEncoder()
le_sicaklik = LabelEncoder()
le_nem = LabelEncoder()
le_ruzgar = LabelEncoder()
le_karar = LabelEncoder()
```

```
veri_seti2 = veri_seti.copy()
```

```
veri_seti2['le_hava'] = le_hava.fit_transform(veri_seti['hava'])
veri_seti2['le_sicaklik'] = le_sicaklik.fit_transform(veri_
seti['sicaklik'])
veri_seti2['le_nem'] = le_nem.fit_transform(veri_seti['nem'])
veri_seti2['le_ruzgar'] = le_ruzgar.fit_transform(veri_seti['ruzgar'])
veri_seti2['le_karar'] = le_karar.fit_transform(veri_seti['karar'])
```

```
veri_seti2.head()
```

	hava	sicaklik	nem	ruzgar	karar	le_hava	l e _ sicaklik	le_nem	l e _ ruzgar	l e _ karar
0	gunesli	ilik	dusuk	evet	oyna	1	0	0	0	0
1	gunesli	sicak	yuksek	evet	oynama	1	1	2	0	1
2	gunesli	sicak	orta	hayir	oynama	1	1	1	1	1
3	gunesli	ilik	yuksek	hayir	oynama	1	0	2	1	1
4	gunesli	soguk	dusuk	hayir	oyna	1	2	0	1	0

```
le_karar_agaci = tree.DecisionTreeClassifier()
```

```
le_karar_agaci.fit(le_X, le_y)
```

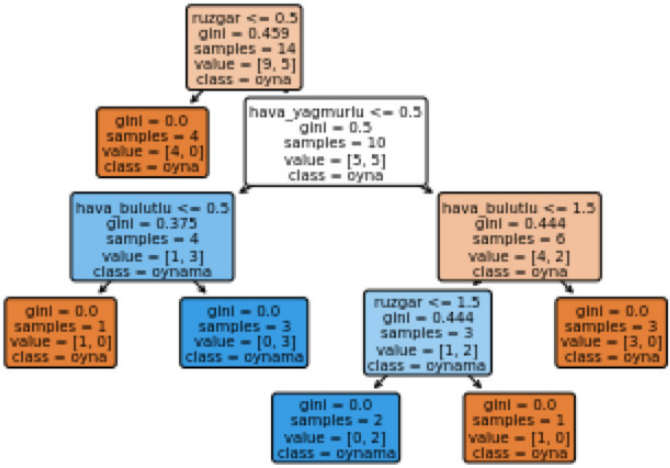
```
le_karar_agaci.get_params()
```

```
{'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'random_state': None,
 'splitter': 'best'}
```

```
le_karar.classes_
```

```
array(['oyna', 'oynama'], dtype=object)
```

```
tree.plot_tree(le_karar_agaci,
               filled=True,
               rounded=True,
               class_names=le_karar.classes_,
               feature_names=X_encoded.columns)
```



```
le_X = veri_seti2.drop(
    ['hava', 'sicaklik', 'nem', 'ruzgar', 'karar', 'le_karar'], axis=1)
le_y = veri_seti2.loc[:, ['le_karar']]
le_X
```

	le_hava	le_sicaklik	le_nem	le_ruzgar
0	1	0	0	0
1	1	1	2	0
2	1	1	1	1
3	1	0	2	1
4	1	2	0	1
5	0	0	2	0
6	0	1	0	1
7	0	2	0	0
8	0	1	0	1
9	2	2	1	0
10	2	2	0	0
11	2	0	1	1
12	2	2	1	1
13	2	2	2	1

Görüldüğü gibi rüzgar özelliği için **one-hot encoding** de olduğu gibi 0,5'ten küçük ve büyük olanlar diye bir ayrım yapıldı. Bu özellik için bir sorun yoktur, ancak hava özelliği için yorum ve açıklama yapmak mümkün değildir. Bu nedenle **`LabelEncoder`**'in kullanılması durumunda bu tip hatalar ile karşılaşılabilir. Bunun yerine Scikit-Learn içerisindeki **`OnehotEncoder`** kullanılması doğru olacaktır. Scikit-Learn içerisindeki diğer encoder ile ilgili detaylı bilgi şu adresten alınabilir *https://contrib.scikit-learn.org/category_encoders/index.html*.

Encoder'lar aşağıda listelenmiştir.

- **`LabelEncoder`** – Etiketler için (0,1,2 şeklinde)
- **`OrdinalEncoder`** – Sıralı veriler için (0,1,2, şeklinde)
- **`Label Binarizer`** – Hedef değişkenin kodlanması için (0,1 gibi)
- **`OneHotEncoder`** - Özellik değişkenleri için(her kategori için bir sütun oluşturur.)

Scikit-Learn Kütüphanesi ile Karar Ağaçları Uygulaması

Bu bölümde ise sürekli değişken tipindeki özelliklerin olduğu bir veri seti üzerinde karar ağaçları uygulaması gösterilecektir. Karar ağaçlarında sınıflandırma **iki** aşamadan oluşur. Birinci aşamada **öğrenme** gerçekleşir. İkinci aşamada **tahmin modeli** çalıştırılır. Tahmin edilen değerler ile test değerlerinin ne kadar doğru olup olmadığı ile ilgili olarak sınıflandırma kalitesi ortaya çıkar.

Bu uygulamada özellik seçim ölçütlerinden Information Gain (Bilgi Kazanımı), Gain Ratio (Kazanım Oranı), Gini Index (Gini Indeksi) ölçütleri kullanılacaktır.

Bu uygulamada Karar Ağaçları Sınıflandırmasının Python Scikit-learn paketi ile nasıl kullanılacağı gösterilecektir. Örnekte ele alınan veri seti **iris veri seti**dir. Veri seti hakkında detaylı bilgi şu adresten alınabilir: *https://archive.ics.uci.edu/ml/datasets/iris*

```
# paketlerin çağrılması
import numpy as np
import pandas as pd
from sklearn import tree
from sklearn import metrics
from sklearn import datasets
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
```

```
# değişkenlerin belirlenmesi
iris = datasets.load_iris()
X = iris.data
y = iris.target
```

```
# normalizasyon
scaler = preprocessing.StandardScaler()
scaler.fit(X)
X = scaler.transform(X)
```

```
# eğitim ve test setleri
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.3,
                                                    random_state=0)
```

```
# karar ağacının oluşturulması
karar_agaci = tree.DecisionTreeClassifier(criterion='entropy', random_
state=0)
```

```
# Öğrenme süreci
karar_agaci.fit(X_train, y_train)
```

```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

```
# doğruluk değeri
metrics.accuracy_score(y_train, karar_agaci.predict(X_train))
```

```
1.0
```

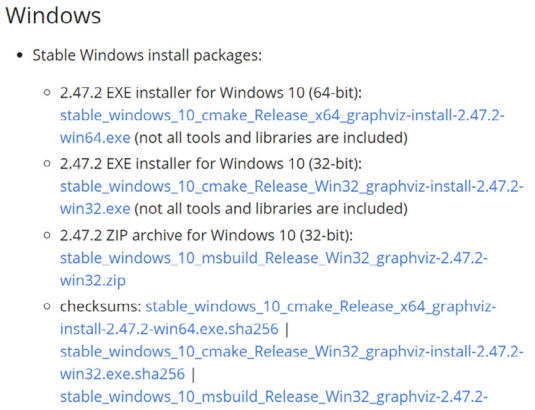
```
# karışıklık matrisi
confusion_matrix(y,karar_agaci.predict(X))
```

```
array([[50,  0,  0],
       [ 0, 49,  1],
       [ 0,  0, 50]], dtype=int64)
```

Veri seti düzgün bir veri seti olduğu için iyi bir öğrenme gerçekleştirilmiştir. 150 değerden 149'u doğru tahmin edilmiştir. Karar ağaçlarının görselleştirilmesi için bir takım kütüphanelere ihtiyaç olacaktır. Scikit-Learn kütüphanesi karar ağaçlarını **.dot** formatında vermektedir. Bu dosyayı alıp görselleştirecek olan kütüphane GraphViz kütüphanesidir. Kütüphanenin bilgilerine şu adresten ulaşılabilir: *http://www.graphviz.org/*. Yükleme yapmak için aşağıdaki kodlar Jupyter notebook üzerinde çalıştırılabilir.

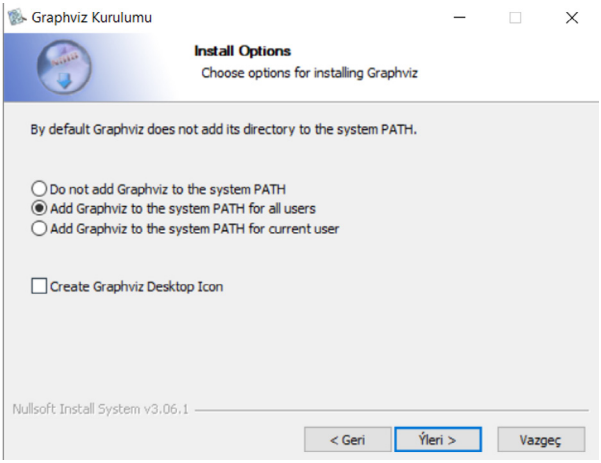
```
!pip install graphviz
```

GraphViz uygulamasını yüklemek için indirme sayfasından istenilen işletim sistemine göre seçim yapılmalıdır.



Şekil 10-7: GraphViz indirme

İndirmeden sonra kurulum yapılırken yönetici izninin aşağıda gösterildiği gibi verilmesi gerekmektedir.



Şekil 10-8: GraphViz yükleme

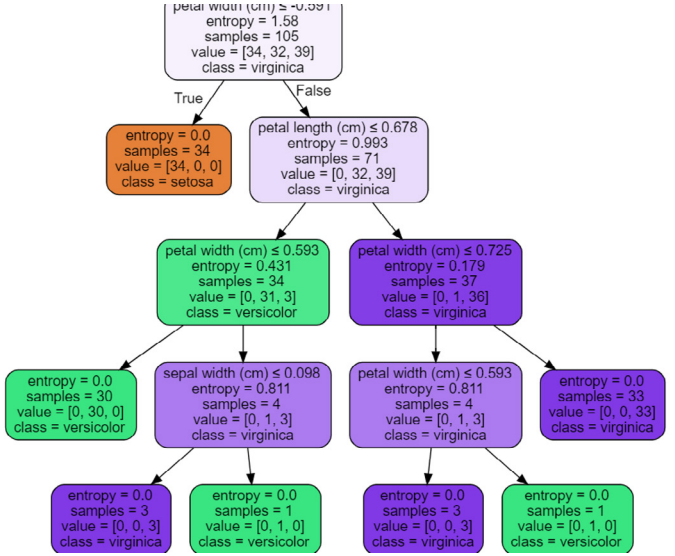
Eğer yönetici izni sonradan verilecekse aşağıdaki gibi verilebilir:

```
import os
# https://graphviz.gitlab.io/_pages/Download/Download_windows.html
os.environ["PATH"] += os.pathsep + 'C:/Program Files/Graphviz/bin/'
```

Ardından görselleştirme işlemi yapılabilir.

```
# karar ağaçlarının oluşturulması
import graphviz
dot_data = tree.export_graphviz(karar_agaci,
                                out_file=None,
                                feature_names = iris.feature_names,
                                class_names = iris.target_names,
                                filled=True,
                                rounded=True,
                                special_characters = True)
graph = graphviz.Source(dot_data)
graph.render('iris')
dot_data = tree.export_graphviz(karar_agaci,
                                out_file=None,
                                feature_names = iris.feature_names,
                                class_names = iris.target_names,
```

```
                                filled=True,
                                rounded=True,
                                special_characters = True)
graph = graphviz.Source(dot_data)
graph
```



Özet

Bu bölümde karar ağaçları hakkında teorik bilgi verilmiştir. Karar ağaçlarının oluşturulmasında kullanılan hesaplamalardan bahsedilmiştir. Ardından karar ağaçlarının Scikit-Learn ile uygulaması gösterilmiştir.

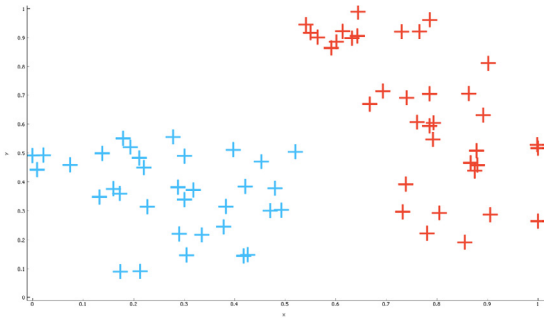
Kaynaklar

1. Breiman, L., Friedman, J., Stone, C. J., et al. Classification and Regression Trees, CRC press (1984).
2. Utgoff, P. E. “ID5: an incremental ID3”, In Machine Learning Proceedings 1988, Elsevier, pp. 107–120 (1988).
3. Quinlan, J. R. C4. 5: Programs for Machine Learning, Elsevier (2014).
4. Tan, P.-N., Steinbach, M., and Kumar, V. “Introduction to Data Mining, Addison”, ed: Boston, MA USA: Wesley Longman, Publishing Co., Inc (2005).
5. Shannon, C. E. “A mathematical theory of communication”, The Bell system technical journal, 27(3), pp. 379–423 (1948).

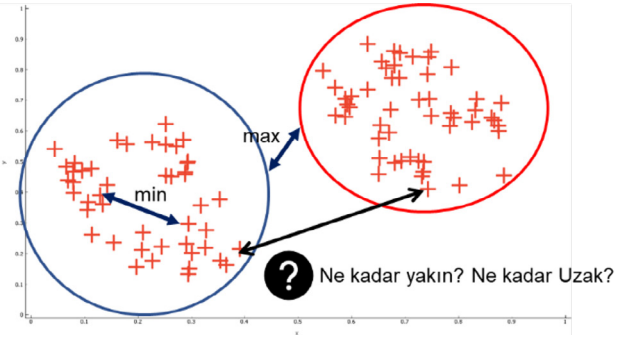
BÖLÜM 11

K-MEANS ALGORİTMASI İLE KÜMELEME

Şimdiye kadar gösterilen algoritmalar gözetimli öğrenme algoritmalarıydı. Makine öğrenmesi içerisinde en fazla kullanılan yöntemler **gözetimli öğrenme algoritmaları**dır. Tekrar hatırlanacak olursa, gözetimli öğrenme algoritmalarında hedef değişkenlerin etiketli olma durumları söz konusuydu. Kümeleme algoritmalarında ise hedef değişkenin etiketlerin belli olmadığı durumlar incelenir. Aşağıdaki görsellerde gözetimli öğrenme ile gözetimsiz öğrenme arasındaki fark görsel olarak gösterilmiştir. Bu bölümde kümeleme algoritmasının nasıl çalıştığı ve bir örnek algoritma olarak K-Means algoritması gösterilecektir. Ardından Python ile kümeleme algoritmasının uygulaması anlatılacaktır.



Şekil 11-1: Gözetimli öğrenmede verilerin etiketli olması durumu



Şekil 11-2: Gözetimsiz öğrenmede verilerin etiketleri bilinmemekte

Birinci görselde başlangıçta hangi verilerin hangi etikete sahip olduğu bilinmektedir. Daha önce yapılan çalışmalarda olduğu gibi iki veriyi birbirinden ayırabilmek için veriler arasına bir çizgi çizilebilir. Diğer görselde ise iki veri tipinin de turuncu renkte olduğu gösterilmiştir. Bu durumda yapılacak yorumlardan birisi veriler arasında bir grubun olmadığı olabilir. Ancak bu yorum doğru olmayabilir. Çünkü veriler arasında bir gruplaşmadan bahsetmek mümkündür. Bu grupların meydana çıkarılması gerekmektedir. Grupların ortaya çıkarılması ile küme içerisindeki mesafelerin **minimum** olması, kümeler arasındaki mesafenin ise **maksimum** olması istenir. Ancak bu şekilde kaliteli bir kümeleme algoritması çalıştırılabilir.

Kümeleme Analizleri

Kümelemede, veriler anlamlı birtakım **gruplara (kümelere)** ayrılır. Yani benzer gruplar bir araya getirilmeye çalışılır. Bu nedenle kümeleme çalışmaları için daha önce gösterilen verilerin uzaklık yakınlık hesaplarının önemi ortaya çıkmaktadır. Verilerin ne kadar benzeyip ne kadar benzemediğini gösterebilmek için benzerlik ölçülerinden yararlanılmalıdır. Kimi durumlarda kümeleme çalışmaları verilerden özet bilgiler çıkarmak amacıyla da kullanılabilir. Kümeleme sayesinde verilerin boyut azaltma çalışmaları gerçekleştirilebilir. Bu çalışmalara örnek olarak **birincil etmen analizi** verilebilir. Kümeleme çalışmaları ile bunun dışında birçok alanda karşılaşılabilir.[1] Bu çalışmalara birkaç örnek vermek gerekirse aşağıdaki örnekler verilebilir.

- Benzer özellikteki müşterilerin belirlenerek reklam kampanyalarının müşteri özelliklerine göre ayrı belirlemek
- İnternetteki benzer dokümanları gruplamak (Benzerlik raporları gibi)

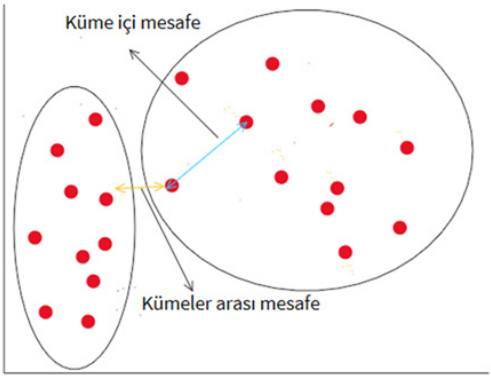
- Benzer özellikteki protein dizilimlerinin ortaya çıkarılması (COVID19 hangi virüsün protein dizisine benziyor gibi)
- Benzer özellikteki şarkıları bir araya getirerek bir kategorizasyon yapmak

Kümeleme çalışmalarının zorluklarından birisi kümeleme çalışmalarında bir keşif çalışması yapılmasıdır. Yani kümelemede doğrulama yapılacak verilerin olmaması nedeniyle analizinde zorluklar yaşanabilir. Yapılan kümelemenin doğrululuğunu kıyaslayacak imkân yoktur.[2] Bu nedenle veri madenciliği açısından anlamlı olmayan sonuçlar çıkabileceği ihtimali vardır. Kümeleme çalışmalarında cevaplanması gereken zor sorular aşağıdaki gibi özetlenebilir:

1. *Gruplama yaparken verilerin ne kadar benzediğine nasıl karar verilecektir?*
2. *Kümelemenin performansı nasıl ölçülecek, performans ne kadar doğru ve verimlidir?*
3. *Veri setini kaç kümeye ayrılacak?*

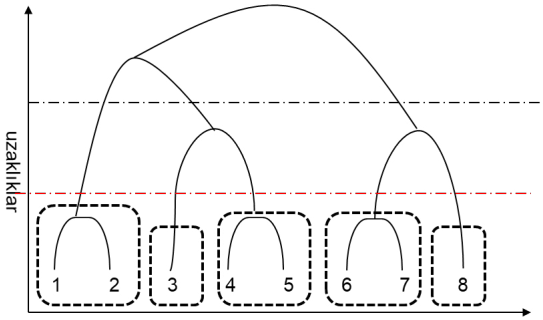
Kümeleme Çeşitleri

Kümeleme çalışmaları **iki** başlık altında incelenebilir. Bunlardan birisi parçalayarak **kümeleme (Partitional Clustering)**, ikincisi hiyerarşik olarak kümelemedir. Aşağıdaki görsellerde parçalayarak kümeleme örneği gösterilmektedir. Parçalayarak kümelemede merkez tabanlı veya yoğunluk tabanlı kümeleme işlemleri yapılır.



Şekil 11-3: Parçalayarak kümeleme

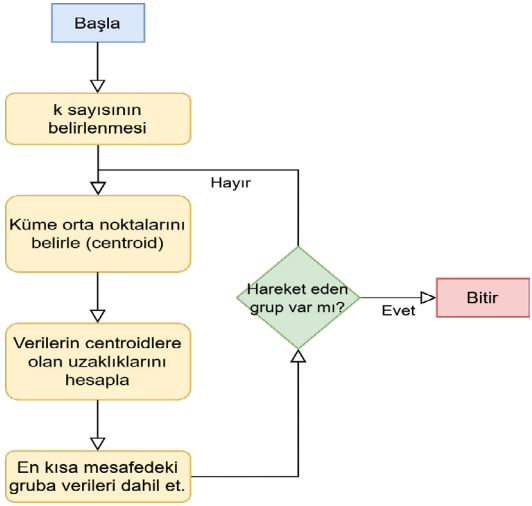
Aşağıdaki görselde ise hiyerarşik kümeleme gösterilmektedir. Hiyerarşik kümelemede dendrogramlar kullanılarak kümeleme yapılır. Bu iki kümeleme çeşidi ile ilgili örnek sonraki bölümde verilecektir.



Şekil 11-4: Hiyerarşik olarak kümeleme örneği

K-Means Kümeleme Algoritması

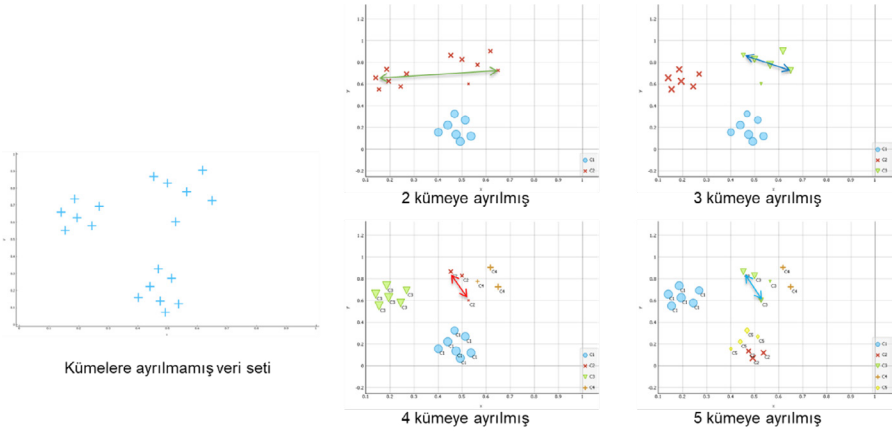
Kümelemede en fazla kullanılan algoritmalardan birisi K-Means algoritmasıdır. Merkez nokta üzerinde bir kümeleme işlemi yapar ve problemdeki verileri kesin kümelere ayırır. K-Means algoritmasında öncelikle K belirlenir. K sayısı küme sayısını göstermektedir. Aşağıdaki akış şemasında K-Means algoritmasının adımları gösterilmiştir. K sayısı belirlendikten sonra ikinci aşamada kümelerin orta noktaları, yani **centroidleri** bulunur. Ardından algoritma üçüncü aşamada verileri en yakın orta noktadaki kümeye atama yapar. Belirlenen uzaklık ölçüsüne göre bir hesaplama yapılır. Sonraki adımda orta noktaların yerleri değiştirilir. Aynı şekilde veriler en yakın orta noktaya atanır. Algoritma önceki adımın sonraki adımdan farkı olmadığı durumda sonlandırılır.



Şekil 11-5: K-Means Algoritmasının Akış Şeması

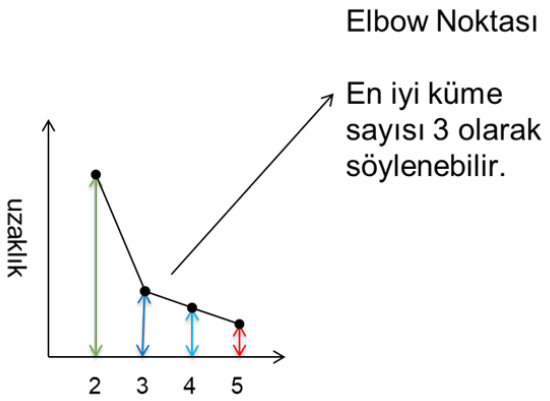
K Değerinin Belirlenmesi-Elbow Yöntemi

Görüldüğü gibi K-Means algoritmasının önemli aşamalarından birisi K değerinin belirlenmesidir. Elbow yani dirsek yöntemi K sayısının belirlenmesi için geliştirilmiş bir yöntemdir. Dirsek yönteminde öncelikle K sayısı için denenecek olan değer sayısına karar verilir. Örnek olarak ele alınacak çalışmada 2-3-4-5 değerleri denenecektir. Ardından en iyi K değeri gösterilecektir. Aşağıdaki görsellerde değişik sayıda kümelere ayrılan veri seti gösterilmektedir. Dirsek yönteminde bu grafikler çıkarıldıktan sonra aynı gruptaki en uzak noktalar belirlenir. Görselde oklar yardımıyla aynı grup içerisindeki en uzak iki nokta arasındaki mesafeler gösterilmiştir.



Şekil 11-6: Farklı sayıda kümelere ayrılmış olan veri seti

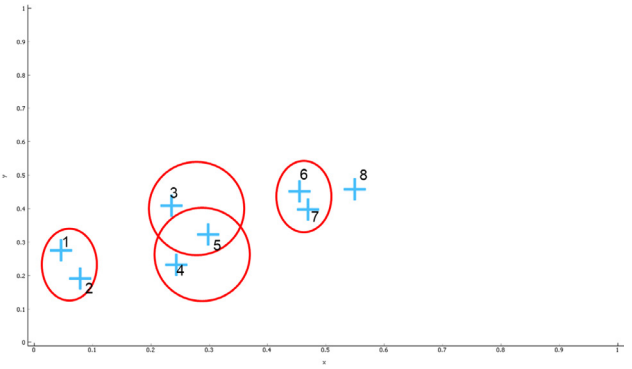
Uzaklıklar belirlendikten sonra uzaklıklar bir koordinat ekseninde sıralanır. Aşağıdaki görselde sıralanan uzaklıklar gösterilmiştir. Grafikten görüldüğü gibi dirsek oluşumu 3 sayısında olduğu için en iyi K sayısı 3'tür denir.



Şekil 11-7: Elbow noktası ve en iyi K değeri

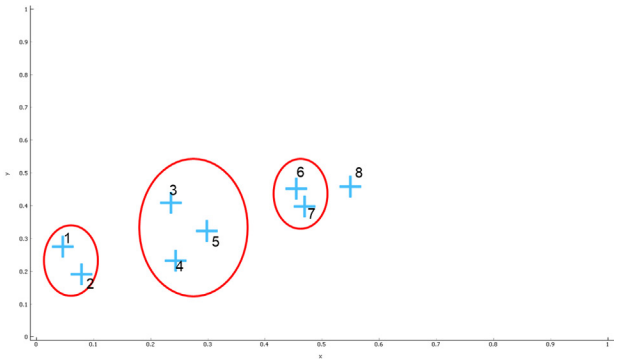
Hiyerarşik Kümeleme

Hiyerarşik kümeleme de kümeleme çalışmaları içerisindeki çalışmalardan birisidir. Hiyerarşik kümelemede öncelikle en yakın 2 nokta birbirleri ile birleştirilir. Aşağıdaki görselde 8 nokta için en yakın ikililer belirlenmiştir.



Şekil 11-8: En yakın ikili noktalar

Ardından görüldüğü gibi 3-5 ile 4-5 kesişim kümesine dâhil oldular. Bu durumda da iki grubun birleştirilmesi işlemi yapılır.



Şekil 11-9: Birleştirilen noktalar

Bu gruplandırmada 3 grubun olduğu görülebilir. Bir birleştirmenin daha olması durumunda uzak iki grubun birleştirilmesi söz konusu olacaktır. Bu nedenle birleştirmenin burada kesilmesi gereklidir. Birleştirme işlemlerinin nerede kesileceği ile ilgili **dendrogramlar** kullanılabilir.

K-Means Algoritması Uygulaması

K-Means algoritmasının adımlarından sonra bu bölümde Python üzerinde K-Means algoritmasını gösterilecektir. K-Means algoritması Scikit-Learn kütüphanesi içinde yer verilen bir algoritmadır. O nedenle kullanması ve çağırması nispeten kolaydır.

Bu örnekte veri seti olarak iyi bilenen **iris veri seti** kullanılmıştır. Veri seti hakkında detaylı bilgi şu adresten alınabilir: *https://archive.ics.uci.edu/ml/datasets/iris* Ayrıca iris veri seti Scikit-Learn kütüphanesinin sunduğu bir veri setidir. O nedenle veri setler içerisinden çağrılabilir.

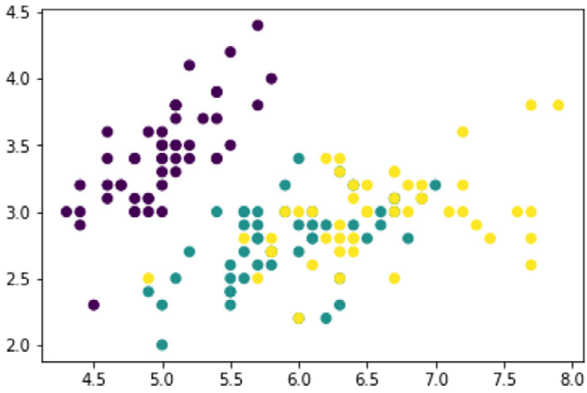
Öncelikle gerekli kütüphaneler çağırılmıştır.

```
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import cluster
```

Aşağıdaki kodlar sayesinde iris veri setinde hızlı bir şekilde X ve y değişkenlerinin tanımı yapılabilir.

```
iris = load_iris()
X, y = load_iris(return_X_y=True)
```

```
plt.scatter(X[:,0], X[:,1], c = iris.target)
```



Iris veri setinin dağılım grafiği yukarıdaki gibi olacaktır. **Fit()** fonksiyonu ile model aşağıdaki gibi eğitilir. Eğitildikten sonra bulunan centroidler yani orta noktalar bulunur. Aynı zamanda bulunan etiketler de aşağıdaki gibi gösterilebilir.

```
kmeans = cluster.KMeans(n_clusters=3, n_jobs=4, random_state=42)
kmeans.fit(X)
```

```
kmeans.cluster_centers_.round(2)
```

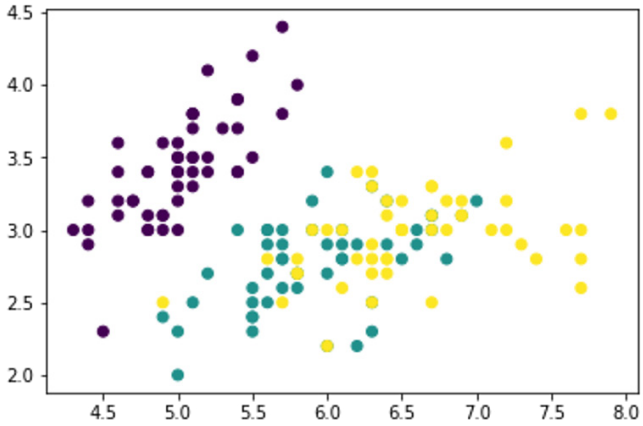
```
array([[5.9,  2.75,  4.39,  1.43],
       [5.01, 3.43,  1.46,  0.25],
       [6.85, 3.07,  5.74,  2.07]])
```

```
kmeans.labels_
```

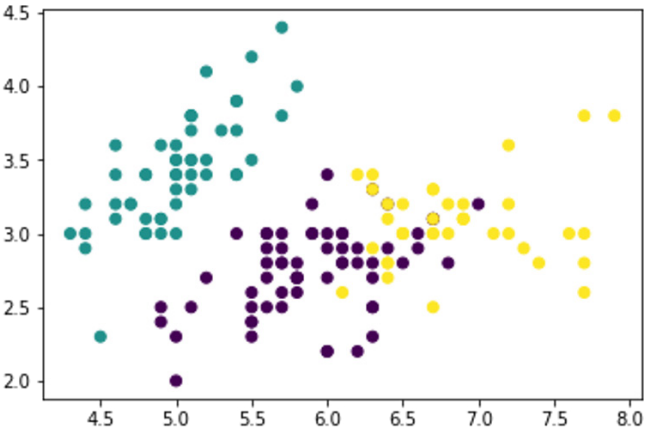
```
array([ 1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  0,  0,  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  2,  0,  2,  2,  2,  2,  0,  2,  2,  2,
        2,  2,  2,  0,  0,  2,  2,  2,  2,  0,  2,  0,  2,  0,  2,  2,  0,  0,  2,  2,  2,  2,
        2,  0,  2,  2,  2,  2,  0,  2,  2,  2,  0,  2,  2,  2,  0,  2,  2,  0])
```

Verilere bir etiket ataması için kullanılacak olan fonksiyon **predict** fonksiyonudur. Bu fonksiyon sonucunda elde edilen tahmin değerleri ve gerçek değerlere ilişkin dağılım grafikleri aşağıdaki gibi verilebilir.

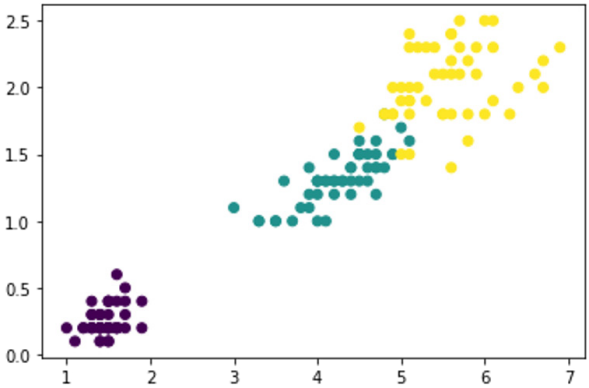
```
# Gerçek değerler
plt.scatter(X[:,0], X[:,1], c=y)
```



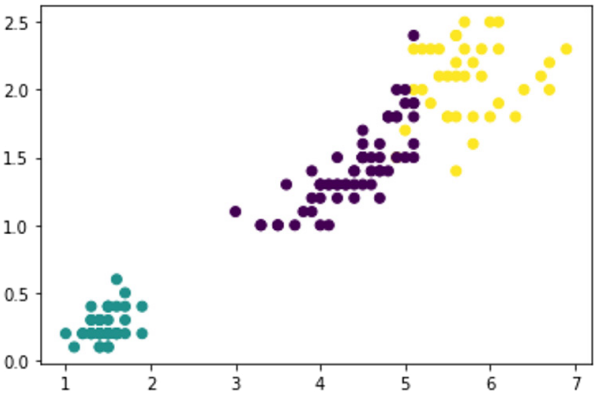

```
# Kümeleme sonuçları
plt.scatter(X[:,0],X[:,1],c=kmeans.labels_)
```




```
plt.scatter(X[:,2], X[:,3], c=y)
```




```
plt.scatter(X[:,2],X[:,3],c=kmeans.labels_)
```



Özet

Kümeleme analizleri literatürde sınıflandırma çalışmalarından sonra üzerinde en fazla çalışma yapılan makine öğrenmesi alanıdır. Bu bölümde K-Means algoritması ile kümeleme çalışmalarının nasıl geliştirilebileceği ve Scikit-Learn kütüphanesinin bu uygulamalarda nasıl kullanılabileceği gösterilmiştir.

Kaynaklar

1. Likas, A., Vlassis, N., and Verbeek, J. J. “The global k-means clustering algorithm”, Pattern recognition, 36(2), pp. 451–461 (2003).
2. Hartigan, J. A. and Wong, M. A. “AK-means clustering algorithm”, Journal of the Royal Statistical Society: Series C (Applied Statistics), 28(1), pp. 100–108 (1979).

BÖLÜM 12

BİRLİKTELİK ANALİZLERİ

Bu bölümde **Orange Veri Madenciliği** aracı kullanılarak bir birliktelik analizi gerçekleştirilecektir. Birliktelik kuralları oluşturulurken Apriori algoritmasından yararlanılacaktır. Küçük bir veri seti üzerinde algoritma adımları gösterilecektir. Bu bölümde ele alınan örnekler şu adreste bulunan *"https://www-users.cs.umn.edu/~kumar001/dmbook/index.php"* kitaptan uyarlanmıştır.[1]

Birliktelik Kuralları

Birliktelik analizleri veri seti içerisinde gizli olan **if-else** kurallarını ortaya çıkarmaya çalışan çalışmalardır. Genellikle kategorik veriler ile iyi sonuçlar verir. En bilindik örneği **sepet analizleri**dir. Bunu dışında bioinformatiks, hastalık teşhisi, web madenciliği, metin madenciliği gibi geniş bir kullanım alanına sahiptir.

Sepet Analizi (Basket Analysis)

Sepet analizinde markette alışveriş yapan kişilerin aldığı ürünler bir listede tutularak, acaba hangi ürünler birlikte daha fazla satılıyor diye bakılır. Aşağıdaki tabloda 5 adetlik bir alışveriş listesi bulunmaktadır. Veri seti 5 farklı kişinin yaptığı alışverişler olarak görülebilir.

No	Alışverişler
1	Ekmek, Süt
2	Ekmek, Çay, Kahve, Yumurta
3	Süt, Çay, Kahve, Kola
4	Ekmek, Süt, Çay, Kahve
5	Ekmek, Süt, Çay, Kola

Tablo 12-1: Örnek alışveriş listesi

Birliktelik analizinde öncelikle ikili gösterim tablosu oluşturulur. Bu tablo sayesinde birliktelik analizinde yer alan değerler ikili şekilde gösterilebilir. Aşağıdaki tabloda alışveriş sepetinin **ikili** gösterimini verilmiştir. Aynı şekilde bu tabloda da her bir satır yeni bir alışverişi gösterir.

No	Ekmek	Süt	Çay	Kahve	Yumurta	Kola
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

Tablo 12-2: Alışveriş listesinin ikili hale getirilmesi

Sepet analizinde kullanılan bir takım kavramlar vardır. Bunlardan birisi destek sayısıdır. Destek sayısı bir ürünün alışveriş listesinde geçme sayısını gösterir. σ ile gösterilir. Örneğin σ({Süt, Çay, Kahve}) = 2. Süt, Çay ve kahvenin bir arada geçtiği durum sayısı 2 olduğu için destek sayısı da 2 olarak hesaplanmıştır. Diğer bir kavram ise destek oranıdır. Destek oranı ürün listesinin alışverişteki oranını gösterir. s ile gösterilir. Örneğin s({Süt, Çay, Kahve}) = 2/5 olarak hesaplanır. Verilen set alışveriş listesinde 2/5 oranında geçmiştir denilir.

Birliktelik kuralları X'ten Y'ye gidiş şeklinde verilebilir. Yani X'de neler olduğunda Y'ler neler olur buna bakılır. Bu gösterimde sağ ok işareti kullanılır. Yani birliktelik kuralları X→Y şeklideki kurallardır. X neler olduğunda Y'ler neler

olmuş diye kurallar okunur. Örneğin {Süt, Çay}→{Kahve} Süt ve çay alanlar Kahve de almışlardır denilir.

Birliktelik analizindeki kural performansına bakmak için destek ve güven sayılarına bakılır. Aşağıdaki örneklerde destek ve güven hesaplamaları gösterilmiştir.

- Destek(s): X ve Y'yi birlikte içeren alışveriş sayısının oranı

  s=(σ(Süt,Çay,Kahve))/n=2/5=0,4

- Güven(c): Y'deki ürünler X'i içeren alışverişlerde ne sıklıkta geçiyor.

  c=(σ(Süt,Çay,Kahve))/(σ(Süt,Çay))=2/3=0,6

{Süt,Çay} → {Kahve} (s=0.4, c=0.67)

{Süt, Kahve} → {Çay} (s=0.4, c=1.0)

{Çay, Kahve} → {Süt} (s=0.4, c=0.67)

{Kahve} → {Süt,Çay} (s=0.4, c=0.67)

{Çay} → {Süt,Kahve} (s=0.4, c=0.5)

{Süt} → {Çay,Kahve} (s=0.4, c=0.5)

Birliktelik Kurallarının Çıkartılması

Kural oluşturmada 2 aşamalı yaklaşım vardır. Birinci yaklaşımda frekans listeleri ve uygun (eşik değerinden büyük) tüm ürün listeleri oluşturulur. Kural oluşturmada en çok kullanılan algoritma **Apriori Algoritması**dır.

Apriori Algoritması

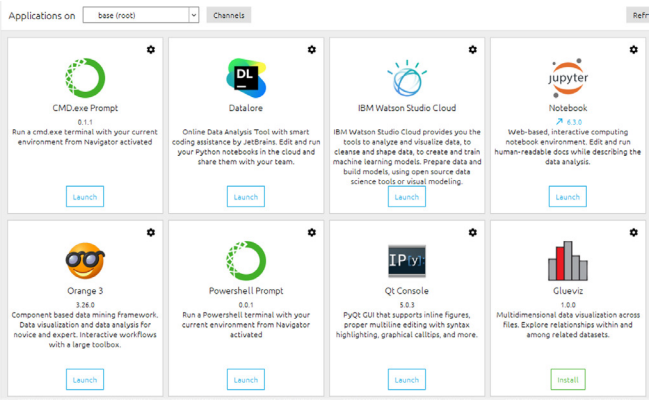
Veriler arasındaki **if-else** kurallarının ortaya çıkarılmasını sağlar. Algoritma kategorik veriler ile çalışabilir. Burada sepet analizi uygulaması bir kez de Apriori algoritması ile gösterilecektir.

Adım: Ürün listelerini frekanslar şeklinde yaz ve maksimum frekansa sahip ürünü belirle. Ürün sayısını eşik değeri ile çarp ve bulduğun değerin altındaki ürünleri çıkar. Adımlar aşağıdaki görselde verilmiştir.

Şekil 12-1: Apriori Algoritması Adımları

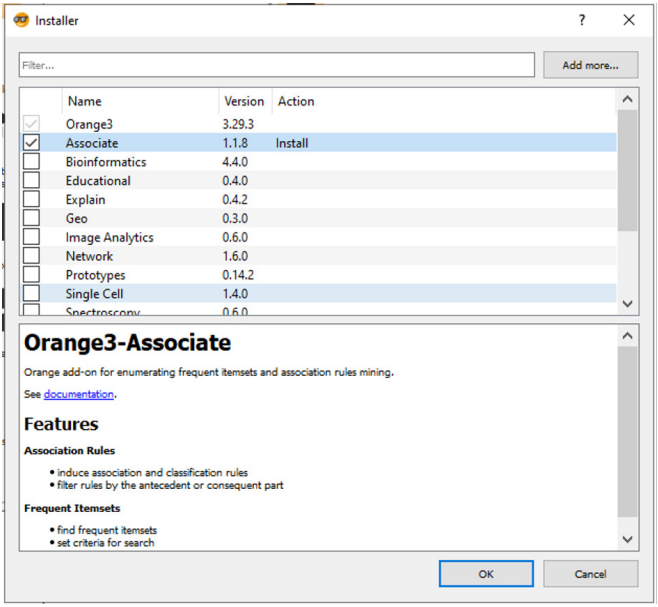
Orange Paketi ile Birliktelik Kuralları

Orange Aracı (*https://orange.biolab.si/*) açık kayak kodlu bir makine öğrenmesi ve veri görselleştirme aracıdır. Orange Aracı ile interaktif bir şekilde veri görselleştirmesi ve veri analizi yapılabilir. Orange Aracının sunduğu arayüz ile veri işlemleri kolaylıkla halledilebilir. Orange Aracı Anaconda üzerinden kurulabileceği gibi kendi sayfasından indirilerek de kurulabilir. Anaconda uygulamaları arasında aşağıdaki görselde gösterildiği gibi Orange Aracı da mecuttur. Eğer Orange 3 uygulamasının altında "**Install**" butonu çıkıyorsa yükleme yapılabilir, eğer "**Launch**" butonu çıkıyorsa yüklenmiş Orange Aracı açılabilirdir.



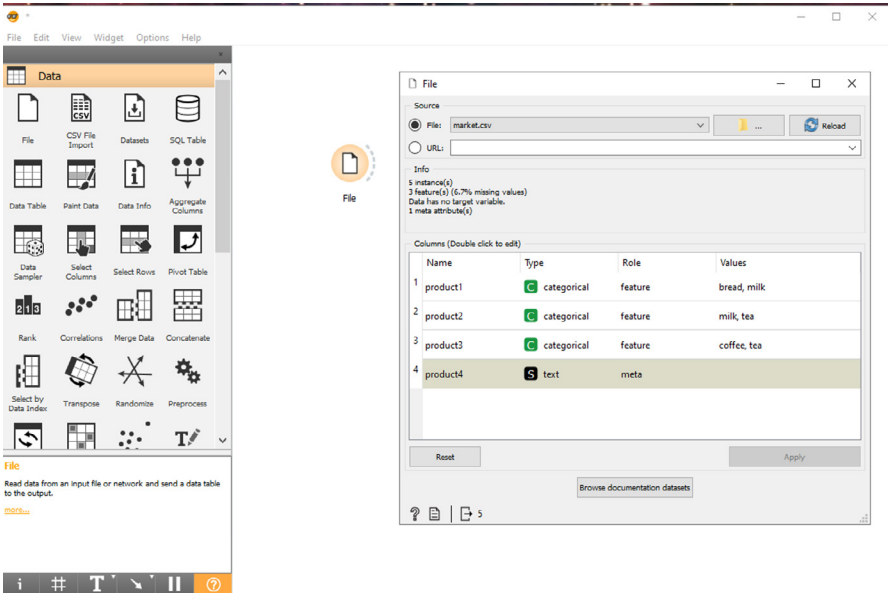
Şekil 12-2: Anaconda Navigator içerisinde Orange Aracı

Orange ile birliktelik kurallarının çalıştırılabilmesi için "**Associate!**" eklentisinin kurulması gerekmektedir. Bu yüklemenin yapılacağı yer **Options>>Add-ons** seçeneğidir. Add-ons seçeneğinde "**Associate!**" eklentisi kurularak birliktelik analizi gerçekleştirilebilir. Bu çalışmada kullanılacak olan veri seti şu adresten indirilebilir: *https://drive.google.com/file/d/1HLU03icAaOgwVkhTt-3_C1P4dFz8-np_/view.* Aşağıdaki görselde "**Associate!**" eklentisinin kurulumu gösterilmiştir. Bir eklenti yüklendikten sonra Orange Aracı yeniden başlatılmaya ihtiyaç duyar. O nedenle bir çalışma yapılmadan önce eklentilerin kurulması önerilmektedir.



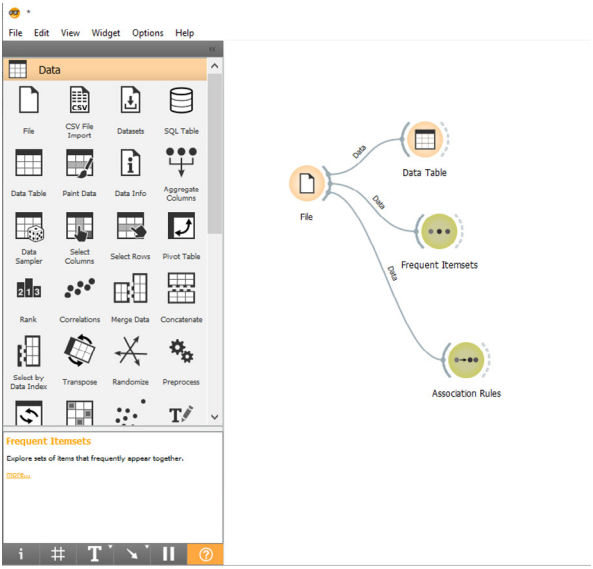
Şekil 12-3: Associate eklentisinin kurulması

İndirilen veri seti **File** seçeneğinden Orange Aracı ile açılır. Ve aşağıdaki gibi veri tablosuna, frekans modülüne bağlanır. Veri setine bağlandıktan sonra veri setinin görüntülenmesi için eklenen modül **çift** tıklanabilir. Sonuç olarak frekans elemanları ve birliktelik kuralları modülleri veri setine bağlanmış olunur. Veri setinin alınması ile ilgili görsel aşağıda paylaşılmıştır.



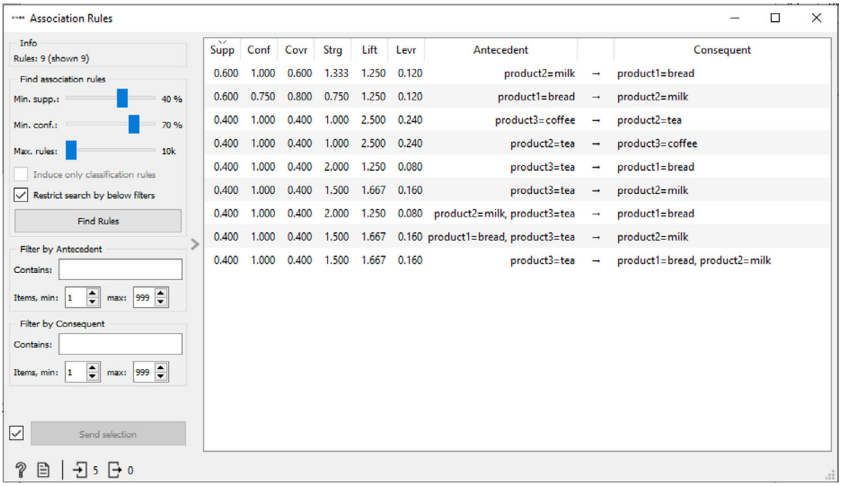
Şekil 12-4: Veri setinin Orange Aracına alınması

Veri seti aşağıdaki gibi hem veri setine hem frekans setine hem de birliktelik kurallarına bağlanmıştır. Orange Aracı sayesinde hızlı bir şekilde modüllerin birbirleri ile olan ilişkileri sürükle bırak, birbirine bağla adımları ile tanımlanabilir. Bu yönüyle Orange Aracı avantajlı kullanım imkânı sunmaktadır. Python ile bu bağlantıların tanımlanması için ayrı kod satırları yazılması gerektiğinden daha önce bahsedilmişti. Bu anlamda paket programların sunduğu avantajların da önemli olduğu söylenebilir.



Şekil 12-5: Orange Aracı ile Birliktelik Analizi

Ardından birliktelik modülünün içerisine girilir. Bu menü içerisinde çeşitli parametre ayarlamaları ile istenilen parametre değerleri ile birliktelik kuralları oluşturulmuş olunur. Aşağıdaki görselde örnek parametreler ile çıkarılan 9 adet kural gösterilmiştir.



Supp	Conf	Covr	Strg	Lift	Levr	Antecedent		Consequent
0.600	1.000	0.600	1.333	1.250	0.120	product2=milk	→	product1=bread
0.600	0.750	0.800	0.750	1.250	0.120	product1=bread	→	product2=milk
0.400	1.000	0.400	1.000	2.500	0.240	product3=coffee	→	product2=tea
0.400	1.000	0.400	1.000	2.500	0.240	product2=tea	→	product3=coffee
0.400	1.000	0.400	2.000	1.250	0.080	product3=tea	→	product1=bread
0.400	1.000	0.400	1.500	1.667	0.160	product3=tea	→	product2=milk
0.400	1.000	0.400	2.000	1.250	0.080	product2=milk, product3=tea	→	product1=bread
0.400	1.000	0.400	1.500	1.667	0.160	product1=bread, product3=tea	→	product2=milk
0.400	1.000	0.400	1.500	1.667	0.160	product3=tea	→	product1=bread, product2=milk

Şekil 12-6: Örnek çıkarılan birliktelik kuralları

Özet

Bu bölümde birliktelik analizleri ve Apriori Algoritması genel hatları ile gösterilmiştir. Birliktelik analizlerine henüz Scikit-Learn kütüphanesi içerisinde etkin bir şekilde yer verilmediğinden çalışma olarak Orange Aracı ile birliktelik analizi incelenmiştir.

Kaynaklar

1. Tan, P.-N., Steinbach, M., and Kumar, V. Introduction to Data Mining, Pearson Education India (2016).

BÖLÜM 13

MAKİNE ÖĞRENMESİ MODELLERİ UYGULAMALARI

Makine öğrenmesinde iki çeşit çalışma yapılabilir. Birisi **modelleme,** birisi de **tahmin** yapmadır. Bu kitapta yer alan öğrenme algoritmaları literatürdeki algoritmalara göre düşünüldüğünde çok küçük bir alanı kaplamaktadır. Makine öğrenmesi çalışmalarında önemli sorunlardan birisi de hangi algoritmanın veri setine daha uygun olduğu ve daha iyi performans vereceğinin ortaya koyulmasındaki zorluklardır. Bu bölümde örnek makine öğrenmesi uygulamaları kodları ile birlikte verilecektir.

Diyabet Hastalığı Tahmini Örneği

Bu bölümde şeker hastalığına ait bir veri seti üzerinde çalışarak bir kişinin şeker hastası olmaya ne kadar yakın olduğunu gösterilecektir. Kullanılan veri seti hakkında detaylı bilgi şu adresten alınabilir: *https://www.kaggle.com/uciml/pima-indians-diabetes-database*. Veri seti indirilip Jupyter notebook ile aynı klasöre koyulduktan sonra aşağıdaki kodlar ile bir uygulama geliştirilebilir. Buradaki uygulamaların herhangi bir akademik geçerliliği yoktur, pratiklik artırmak açısından takip edilebilecek uygulamalardır. Yani burada kitapta gösterilen fonksiyonlar ve yöntemlerin uygulamasının gösterilmesi birincil önceliktedir. Uygulamada kullanılan veri seti (*https://bit.ly/diabets-csv*) adresinden indirilebilir.

```
import numpy as np
import pandas as pd
```

```
df = pd.read_csv("diabetes.csv")
df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

```
# veri setinde eksik veri var mı
df.isnull().sum()
```

```
Pregnancies               0
Glucose                   0
BloodPressure             0
SkinThickness             0
Insulin                   0
BMI                       0
DiabetesPedigreeFunction  0
Age                       0
Outcome                   0
dtype: int64
```

Veri seti hakkında genel bilgi alındıktan sonra eksik veri olup olmadığına bakılmış ve eksik veri olmadığı görülmüştür. Ancak yine de veri setindeki ilk 5 satır incelendiğinde insülin değerlerinde 0 değerleri olduğuna dikkat edilmelidir. Bu durumda eksik verilerin 0 değeri olarak girildiği düşünülebilir. Bu durumun çözümü için hangi değerlerin sıfır olarak girildiğine bakılması gerekir.

```
df.eq(0).sum()
```

```
Pregnancies                 111
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                     500
dtype: int64
```

```
# Bu nedenle 0 değerlerinin nan olarak girilmesi gereklidir.
df[['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'Di
abetesPedigreeFunction','Age']] = df[['Glucose', 'BloodPressure',
'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction','Age']].
replace(0,np.NaN)
```

```
# Eksik verileri ortalama ile dolduralım.
df.fillna(df.mean(),inplace=True)
```

```
# Şimdi 0 olan değerleri görelim.
df.eq(0).sum()
```

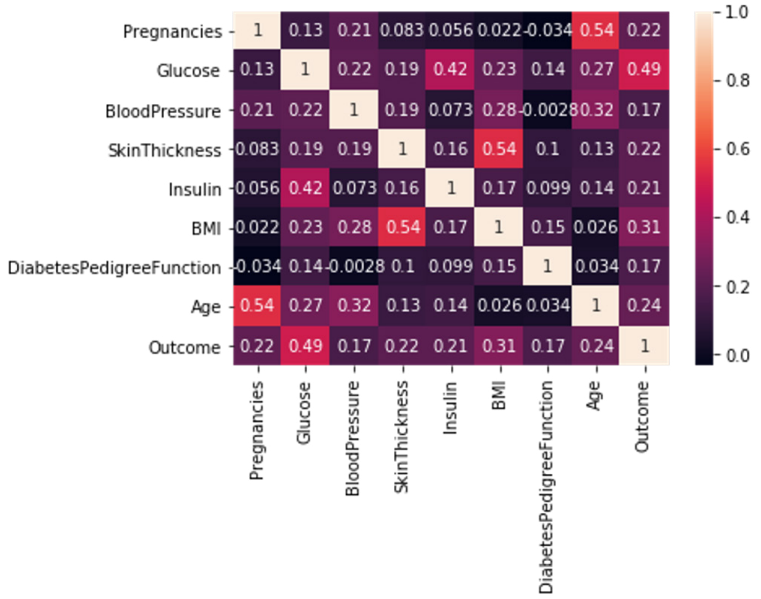
```
Pregnancies                 111
Glucose                       0
BloodPressure                 0
SkinThickness                 0
Insulin                       0
BMI                           0
DiabetesPedigreeFunction      0
Age                           0
Outcome                     500
dtype: int64
```

Özellik değişkenleri ve hedef değişkeni arasındaki ilişkilerin sayısal büyüklüğünü gösterebilmek için korelasyon değerlerine bakılmalıdır. Değerlere tablo olarak bakılabileceği gibi ısı haritası olarak da bakılabilir. Bunun için **Seaborn** kütüphanesinin çağrılması gereklidir.

```
# Korelasyon analizi
# Çıktı değişkenine hangi özelliklerin etki ettiğini korelasyon ile görebiliriz.
df.corr()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.127911	0.208522	0.082989	0.056027	0.021565	-0.033523	0.544341	0.221898
Glucose	0.127911	1.000000	0.218367	0.192991	0.420157	0.230941	0.137060	0.266534	0.492928
BloodPressure	0.208522	0.218367	1.000000	0.192816	0.072517	0.281268	-0.002763	0.324595	0.166074
SkinThickness	0.082989	0.192991	0.192816	1.000000	0.158139	0.542398	0.100966	0.127872	0.215299
Insulin	0.056027	0.420157	0.072517	0.158139	1.000000	0.166586	0.098634	0.136734	0.214411
BMI	0.021565	0.230941	0.281268	0.542398	0.166586	1.000000	0.153400	0.025519	0.311924
DiabetesPedigreeFunction	-0.033523	0.137060	-0.002763	0.100966	0.098634	0.153400	1.000000	0.033561	0.173844
Age	0.544341	0.266534	0.324595	0.127872	0.136734	0.025519	0.033561	1.000000	0.238356
Outcome	0.221898	0.492928	0.166074	0.215299	0.214411	0.311924	0.173844	0.238356	1.000000

```
# Korelasyon grafiği
import seaborn as sns
sns.heatmap(df.corr(), annot=True)
```



Korelasyon analizinden çıkan sonuca göre en yüksek korelasyona sahip 4 özellik yazdırılmak istenirse aşağıdaki gibi bir yapı kullanılabilir:

```
# en yüksek korelasyona sahip 4 özellik
df.corr().nlargest(4,'Outcome').index
```

```
Index(['Outcome', 'Glucose', 'BMI', 'Age'], dtype='object')
```

Görüldüğü gibi çıktı ile aralarında en yüksek korelasyona sahip olanlar Glucose, **BMI** ve **Age** özellikleridir. Analize sadece bu özellikler kullanılarak devam edilerek model şu şekilde kurulabilir:

```
from sklearn import linear_model
from sklearn.model_selection import cross_val_score
X = df[['Glucose','BMI','Age']]
y=df.iloc[:,8]
```

```
log_reg = linear_model.LogisticRegression()
log_reg_score = cross_val_score(log_reg,X,y,cv=10,scoring='accuracy').
mean()
```

```
log_reg_score
```

```
0.7669856459330144
```

```
result=[]
result.append(log_reg_score)
```

Yukarıdaki gibi lojistik regresyon uygulanması ile ilk skor elde edilmiş olunur. Bu skor 0,76 performansına sahiptir. Şimdilik bu skor bir listede tutulacak diğer algoritma sonuçları ile karşılaştırılmak üzere bekletilecektir. Aynı problem destek vektör makineleri ile modellenirse aşağıdaki gibi sonuçlara ulaşılır:

```
from sklearn import svm
linear_svm = svm.SVC(kernel='linear')
```

```
linear_svm_score = cross_val_score(linear_svm,X,y,cv=10,scoring='accura
cy').mean()
```

```
linear_svm_score
```

0.7656527

Destek vektör makineleri de **benzer** sayılabilecek sonuçlar vermiştir. Şimdi bu sonuçların kaydedilmesi ve daha sonradan da çalıştırılabilmesi için **pickle** kütüphanesi çağrılacaktır.

```
# modelin kaydedilmesi
import pickle
```

```
filename='diabets.sav'
```

```
log_reg.fit(X,y)
```

```
pickle.dump(log_reg,open(filename,'wb'))
```

```
# load model
loaded_model = pickle.load(open(filename,'rb'))
```

```
# prediction
Glucose = 55
BMI = 60
Age = 20
prediction = loaded_model.predict([[Glucose,BMI,Age]])
```

```
array([0], dtype=int64)
```

COVID19 Türkiye Verileri Veri Madenciliği Uygulama Örneği

Bu uygulama örneği COVID19 Türkiye verileri ile gerçekleştirilecektir. Gerçek veriler kullanılarak bir tahmin modeli geliştirilecek olup yapay sinir ağları ile model eğitilecektir. Veriler 11 Mart 2020 ile 21 Nisan 2020 tarihleri arasında Türkiye'de kaydedilen COVID19 rakamlarıdır. Özellikler olarak, tarih, test sayısı, vaka sayısı, yoğun bakımda yatan toplam hasta sayısı, entübe toplam hasta sayısı, iyileşen hasta sayısı ve vefat eden hasta sayıları alınmıştır. Veri seti **csv** hâlinde şu adresten indirilebilir: *https://drive.google.com/uc?export=view&id=11SJRu_ghaKQCmKTNP6avuSgOr_RqB7tx*

İlk önce ihtiyaç duyulan kütüphaneler çağrılır. Önceki bölümlerde geçmeyen **MLPRegressor** paketi içerisindeki fonksiyonlar ile yapay sinir ağları eğitilecektir.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from seaborn import heatmap
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.neural_network import MLPRegressor

plt.style.use('ggplot') # matplotlib stili için

csv_file_url = 'https://drive.google.com/uc?export=view&id=11SJRu_
ghaKQCmKTNP6avuSgOr_RqB7tx'
dataset = pd.read_csv(csv_file_url, sep=';',index_col=0)
```

Veri setindeki indeks sütunu aşağıdaki gibi tarihler ile değiştirilmiştir.

```
dataset.index = pd.to_datetime(dataset.index, format='%d.%m.%Y')
```

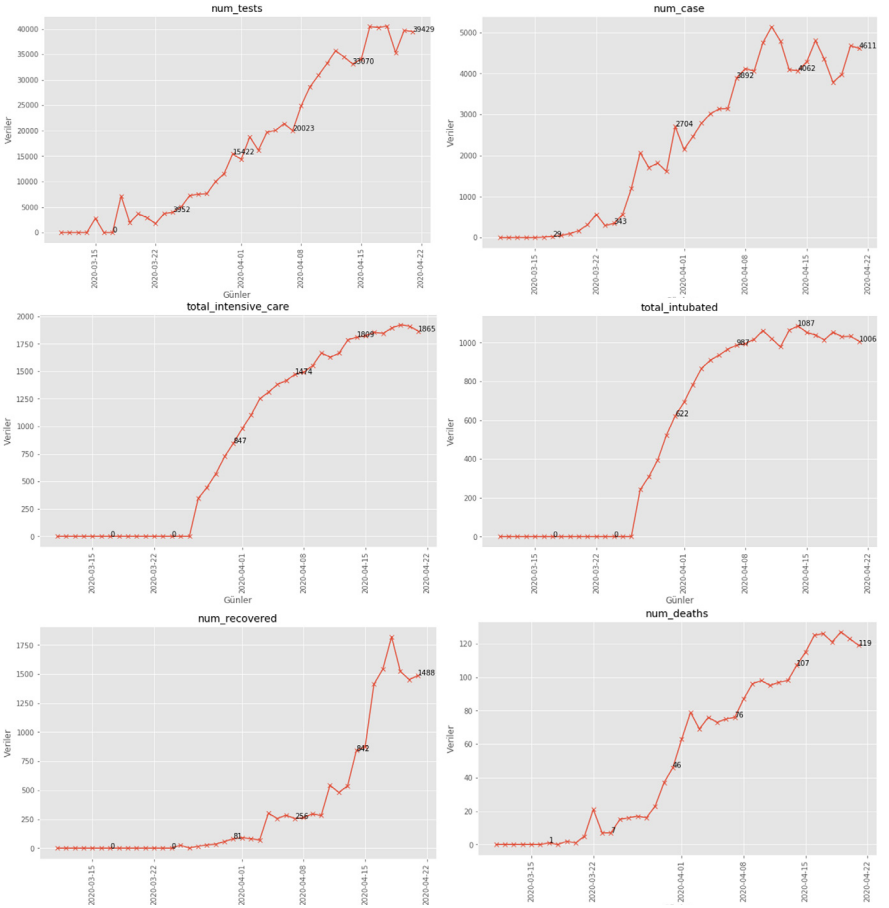
```
dataset.head(10)
```

	num_tests	num_case	total_intensive_care	total_intubated	num_recovered	num_deaths
date						
2020-03-11	0	1	0	0	0	0
2020-03-12	0	0	0	0	0	0
2020-03-13	0	4	0	0	0	0
2020-03-14	0	0	0	0	0	0
2020-03-15	2800	1	0	0	0	0
2020-03-16	0	12	0	0	0	0
2020-03-17	0	29	0	0	0	1
2020-03-18	7197	51	0	0	0	0
2020-03-19	1981	94	0	0	0	2
2020-03-20	3656	167	0	0	0	1

Veriler grafik üzerinde gösterilmek istenirse aşağıdaki yapı kullanılabilir. Burada her hafta gerçekleşen sayılar ayrıca grafiğe eklenmiştir.

```
for i, col in enumerate(dataset.columns.tolist()):
    plt.figure(figsize=(10,6))
```

```
x_axis= dataset.index.values
y_axis = dataset[col].values
plt.plot(x_axis, y_axis, label=col, marker='x')
plt.title(col)
plt.xlabel('Günler')
plt.ylabel('Veriler')
plt.xticks(rotation=90)
counter = 0
for j, k in zip(x_axis, y_axis):
    counter +=1
    if counter % 7 ==0:
        plt.annotate(str(k), xy=(j, k))
```



Şekil 13-1: COVID19 Türkiye verileri

İlk 9 günde problem olduğu için veriler alınamadığı için ilk 8 günlük veriler sonraya itelenmiştir.

```
dataset_shifted = dataset.iloc[9:,:]
```

```
dataset_shifted.head()
```

	num_tests	num_case	total_intensive_care	total_intubated	num_recovered	num_deaths
date						
2020-03-20	3656	167	0	0	0	1
2020-03-21	2953	311	0	0	0	5
2020-03-22	1758	566	0	0	0	21
2020-03-23	3672	293	0	0	0	7
2020-03-24	3952	343	0	0	0	7

```
dataset_shifted.describe()
```

	num_tests	num_case	total_intensive_care	total_intubated	num_recovered	num_deaths
count	33.000000	33.000000	33.000000	33.000000	33.000000	33.000000
mean	21255.484848	2890.878788	1107.484848	687.303030	452.848485	68.363636
std	13421.055955	1595.323728	724.413518	424.414559	571.175275	43.126136
min	1758.000000	167.000000	0.000000	0.000000	0.000000	1.000000
25%	7641.000000	1704.000000	445.000000	309.000000	28.000000	21.000000
50%	20023.000000	3135.000000	1381.000000	935.000000	256.000000	76.000000
75%	34090.000000	4117.000000	1786.000000	1021.000000	542.000000	98.000000
max	40520.000000	5138.000000	1922.000000	1087.000000	1822.000000	127.000000

Verilerde **ölçeklendirme** yapılması gereklidir. Ölçeklendirme için **`MinMaxScaler()`** kullanılmıştır.

```
scaler = MinMaxScaler()
dataset_scaled = pd.DataFrame(scaler.fit_transform(dataset_shifted),
                              columns=dataset_shifted.columns)
```

```
dataset_scaled = pd.DataFrame(dataset_scaled, columns=dataset.columns)
```

```
dataset_scaled.index = dataset_shifted.index
```

```
dataset_scaled.head()
```

	num_tests	num_case	total_intensive_care	total_intubated	num_recovered	num_deaths
date						
2020-03-20	0.048965	0.000000	0.0	0.0	0.0	0.000000
2020-03-21	0.030829	0.028968	0.0	0.0	0.0	0.031746
2020-03-22	0.000000	0.080266	0.0	0.0	0.0	0.158730
2020-03-23	0.049378	0.025347	0.0	0.0	0.0	0.047619
2020-03-24	0.056602	0.035405	0.0	0.0	0.0	0.047619

```
corr = dataset_scaled.corr()
```

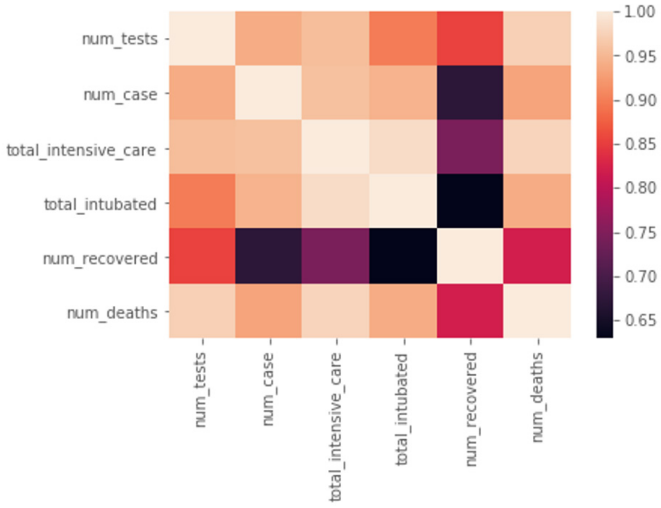
```
corr
```

	num_tests	num_case	total_intensive_care	total_intubated	num_recovered	num_deaths
num_tests	1.000000	0.939266	0.956330	0.899342	0.854870	0.973720
num_case	0.939266	1.000000	0.958863	0.946025	0.673708	0.931931
total_intensive_care	0.956330	0.958863	1.000000	0.984560	0.745196	0.977441
total_intubated	0.899342	0.946025	0.984560	1.000000	0.629068	0.939364
num_recovered	0.854870	0.673708	0.745196	0.629068	1.000000	0.820568
num_deaths	0.973720	0.931931	0.977441	0.939364	0.820568	1.000000

Korelasyonlara göre bir analiz yapılarak uygulamaya devam edilmiştir.

```
heatmap(corr,xticklabels=dataset.columns, yticklabels=dataset.columns)
```

<matplotlib.axes._subplots.AxesSubplot at 0x1c41d14d088>



Sonuç olarak X ve y değişkenleri aşağıdaki gibi belirlenmiştir:

```
X = dataset_scaled.iloc[:,[1,4]]
y =dataset_scaled.iloc[:,[2,3]]
```

```
mlp = MLPRegressor(hidden_layer_sizes=(75, ),
                   max_iter=100000,
                   learning_rate_init=0.05,
                   random_state=42)
mlp.fit(X.values, y.values)
```

```
MLPRegressor(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(75,), learning_rate='constant',
       learning_rate_init=0.05, max_fun=15000, max_iter=100000,
       momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
       power_t=0.5, random_state=42, shuffle=True, solver='adam',
       tol=0.0001, validation_fraction=0.1, verbose=False,
       warm_start=False)
```

```
y_predicted = mlp.predict(X.values)
```

```
mean_squared_error(y.values, y_predicted)
```

```
0.005340620984241113
```

```
r2_score(y.values, y_predicted)
```

```
0.9626789187506493
```

```
mlp.coefs_[1].round(2)
```

```
mlp.intercepts_
```

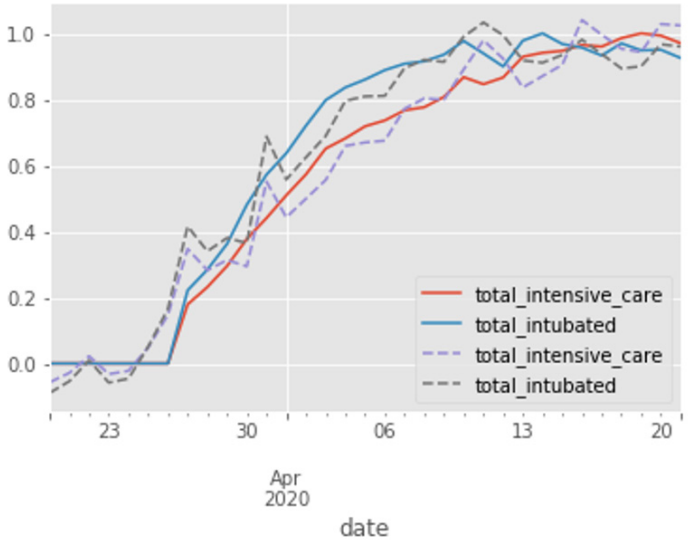
```
[array([-0.34807287,  0.07209708, -0.19825183, -0.3492062 ,  0.26917149,
       -0.1440081 , -0.0348794 , -0.32757356, -0.14647447, -0.04303863,
       -0.32960869,  0.15420971,  0.09536544, -0.45808058, -0.22873734,
       -0.12571818, -0.10005679, -0.17501375, -0.25638089, -0.25291298,
        0.07601236, -0.26988444, -0.29275101, -0.15269482, -0.10872383,
       -0.18179814, -0.07120347, -0.06323458, -0.12159667, -0.20236862,
       -0.08873115, -0.21579407, -0.10134007,  0.1360771 , -0.434648  ,
       -0.07260023, -0.13292528, -0.26869212, -0.18669255, -0.14412142,
```

```
       -0.22716687, -0.3873794 ,  0.36643258, -0.12961531, -0.08986814,
       -0.1490529 , -0.08988312, -0.14051884,  0.15220766, -0.33003645,
        0.08547389, -0.37202145, -0.50934092,  0.16347067, -0.20728203,
       -0.27401064, -0.22249466,  0.24160835, -0.27631943, -0.18936765,
       -0.29860145, -0.05992743, -0.05214584, -0.15393783, -0.07734605,
       -0.4461963 , -0.45719702, -0.03583762, -0.16223401,  0.16139983,
       -0.13234164, -0.19382516, -0.22684753, -0.3733633 , -0.13108529]),
 array([0.04962587, 0.29900824])]
```

Tahmin sonuçları elde edildikten sonra tahminler aşağıdaki gibi grafik üzerinde gösterilebilir.

```
ax = y.plot(linestyle='-')
pd.DataFrame(y_predicted, columns=y.columns, index=y.index).plot(ax=ax,
linestyle='--')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1c41ba1f788>
```



Özet

Bu bölümde iki adet farklı uçtan uca veri madenciliği uygulaması gösterilmiştir. Ele alınan problemler Python programlama diliyle yazılacak olan veri madenciliği uygulamalarına örnek olarak verilebilir. Bu uygulamaların problem özelinde farklı yaklaşımları söz konusu olabilir. Bu da uygulamaların adım adım farklılık gösterebileceği anlamına gelir.

BÖLÜM 14

BİBLİYOGRAFYA

Ahmad, Amir, and Lipika Dey. “A Method to Compute Distance between Two Categorical Values of Same Attribute in Unsupervised Learning for Categorical Data Set.” Pattern Recognition Letters 28, no. 1 (2007): 110–18.

Anaconda. “Anaconda | Individual Edition,” 2021. https://www.anaconda.com/products/individual.

KDnuggets. “Battle of the Data Science Venn Diagrams,” 2021. https://www.kdnuggets.com/battle-of-the-data-science-venn-diagrams.html/.

Bock, Hans-Hermann, and Edwin Diday. Analysis of Symbolic Data: Exploratory Methods for Extracting Statistical Information from Complex Data. Springer Science & Business Media, 2012.

“Boston Dataset,” 2021. https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html.

Breiman, Leo, Jerome Friedman, Charles J. Stone, and Richard A. Olshen. Classification and Regression Trees. CRC press, 1984.

Brownlee, Jason. “Classification And Regression Trees for Machine Learning.” Machine Learning Mastery (blog), 2016. https://machinelearningmastery.com/classification-and-regression-trees-for-machine-learning/.

“Built-in Functions — Python 3.9.2 Documentation,” 2021. https://docs.python.org/3/library/functions.html.

Buitinck, Lars, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, et al. “API Design for Machine Learning Software: Experiences from the Scikit-Learn Project.” In ECML PKDD Workshop: Languages for Data Mining and Machine Learning, 108–22, 2013.

Cao, Longbing. “Data Science: A Comprehensive Overview.” ACM Computing Surveys 50, no. 3 (2017): 1–42. https://doi.org/10.1145/3076253.

Ceri, Stefano. “On the Role of Statistics in the Era of Big Data: A Computer Science Perspective.” Statistics & Probability Letters 136 (2018): 68–72. https://doi.org/10.1016/j.spl.2018.02.019.

“Constants — NumPy v1.21.Dev0 Manual,” 2021. https://numpy.org/devdocs/reference/constants.html.

“Daring Fireball: Markdown,” 2021. https://daringfireball.net/projects/markdown/.

“Data Science Roadmap,” 2021. https://medium.com/@ArtisOne/data-science-roadmap-2020-b256fb948404.

Goodall, David W. “A New Similarity Index Based on Probability.” Biometrics, 1966, 882–907.

Han, Jiawei, Micheline Kamber, and Jian Pei. Data Mining: Concepts and Techniques. Haryana, India; Burlington, MA, 2011.

———. “Data Mining Concepts and Techniques Third Edition.” The Morgan Kaufmann Series in Data Management Systems 5, no. 4 (2011): 83–124.

Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, et al. “Array Programming with NumPy.” Nature 585, no. 7825 (2020): 357–62. https://doi.org/10.1038/s41586-020-2649-2.

Hartigan, John A., and Manchek A. Wong. “AK-Means Clustering Algorithm.” Journal of the Royal Statistical Society: Series C (Applied Statistics) 28, no. 1 (1979): 100–108.

Hosmer Jr, David W., Stanley Lemeshow, and Rodney X. Sturdivant. Applied Logistic Regression. Vol. 398. John Wiley & Sons, 2013.

“Host, Run, and Code Python in the Cloud: PythonAnywhere,” 2021. https://www.pythonanywhere.com/.

L Halterman, Richard. Learning to Program with Python, 2011.

Le, Si Quang, and Tu Bao Ho. “An Association-Based Dissimilarity Measure for Categorical Data.” Pattern Recognition Letters 26, no. 16 (2005): 2549–57.

“Learning to program with Python 3 (py 3.7) - YouTube,” 2021. https://www.youtube.com/.

Li, Chaoqun, Liangxiao Jiang, and Hongwei Li. “Local Value Difference Metric.” Pattern Recognition Letters 49 (2014): 62–68. https://doi.org/10.1016/j.patrec.2014.06.014.

Li, Chaoqun, and Hongwei Li. “A Survey of Distance Metrics for Nominal Attributes.” Journal of Software 5, no. 11 (2010): 1262–69. https://doi.org/10.4304/jsw.5.11.1262-1269.

Likas, Aristidis, Nikos Vlassis, and Jakob J. Verbeek. “The Global K-Means Clustering Algorithm.” Pattern Recognition 36, no. 2 (2003): 451–61.

Lutz, Mark. Learning Python, 5th Edition. Fifth edition. Beijing: O’Reilly Media, 2013.

Menard, Scott. Applied Logistic Regression Analysis. Vol. 106. Sage, 2002.

Metwalli, Sara A. “A Learning Path To Becoming a Data Scientist.” Medium, 2020. https://towardsdatascience.com/a-learning-path-to-becoming-a-data-scientist-56c5c2e8ae3f.

Taylor & Francis. “Multiple Testing Problems in Pharmaceutical Statistics | Taylor & Francis Group,” 2021. https://www.taylorfrancis.com/books/multiple-testing-problems-pharmaceutical-statistics-alex-dmitrienko-ajit-tamhane-frank-bretz/e/10.1201/9781584889854.

Six-Sigma-Material.com. “Normal Distribution,” 2021. https://www.six-sigma-material.com/Normal-Distribution.html.

Ömer Bağcı. Introduction to Python 3 Programming Tutorial, 2018. https://www.youtube.com/watch?v=eXBD2bB9-RA&list=PLQVvvaa0QuDeAams7fkdcwOGBpGdHpXln.

“Os — Miscellaneous Operating System Interfaces — Python 3.9.2 Documentation,” 2021. https://docs.python.org/3/library/os.html.

Passmore, Heather A. “To Learn Data Science Better, Use SCIENCE!” Medium, 2018. https://towardsdatascience.com/to-learn-data-science-better-use-science-ffaae3d56bea.

“Python Programlama Dili — Python 3 İçin Türkçe Kılavuz,” 2021. https://python-istihza.yazbel.com/.

“Python Programming Tutorials - YouTube,” 2021. https://www.youtube.com/.

Python, Real. “Lists and Tuples in Python – Real Python,” 2021. https://realpython.com/python-lists-tuples/.

———. “Python Tutorials – Real Python,” 2021. https://realpython.com/.

“Python Tutorial for Beginners (Full Course) - YouTube,” 2021. https://www.youtube.com/.

Quinlan, J. Ross. C4. 5: Programs for Machine Learning. Elsevier, 2014.

Reback, Jeff, Wes McKinney, Jbrockmendel, Joris Van Den Bossche, Tom Augspurger, Phillip Cloud, Simon Hawkins, et al. Pandas-Dev/Pandas: Pandas 1.2.4 (version v1.2.4). Zenodo, 2021. https://doi.org/10.5281/ZENODO.3509134.

Shannon, Claude E. “A Mathematical Theory of Communication.” The Bell System Technical Journal 27, no. 3 (1948): 379–423.

Shaw, Zed A. Learn Python the Hard Way: A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code. Addison-Wesley, 2013.

Stanfill, Craig, and David Waltz. “Toward Memory-Based Reasoning.” Communications of the ACM 29, no. 12 (1986): 1213–28.

Tan, Pang-Ning, Michael Steinbach, and Vipin Kumar. Introduction to Data Mining. Pearson Education India, 2016.

———. “Introduction to Data Mining, Addison.” Ed: Boston, MA USA: Wesley Longman, Publishing Co., Inc, 2005.

upGrad blog. “Top 6 Data Science Programming Languages 2021 [Hand-Picked],” 2021. https://www.upgrad.com/blog/data-science-programming-languages/.

Towards Data Science. “Towards Data Science,” 2021. https://towardsdatascience.com/.

“Usage Guide — Matplotlib 3.4.1 Documentation,” 2021. https://matplotlib.org/stable/tutorials/introductory/usage.html#sphx-glr-tutorials-introductory-usage-py.

Utgoff, Paul E. “ID5: An Incremental ID3.” In Machine Learning Proceedings 1988, 107–20. Elsevier, 1988.

“Wayback Machine,” 2021. https://web.archive.org/.

Weihs, Claus, and Katja Ickstadt. “Data Science: The Impact of Statistics.” International Journal of Data Science and Analytics 6, no. 3 (2018): 189–94. https://doi.org/10.1007/s41060-018-0102-5.

“Why Exclude Highly Correlated Features When Building Regression Model ?? | by Aishwarya V Srinivasan | Towards Data Science,” 2021. https://towardsdatascience.com/why-exclude-highly-correlated-features-when-building-regression-model-34d77a90ea8e.

Wright, Raymond E. “Logistic Regression.,” 1995.

“YazBel Belgeleri,” 2021. https://belgeler.yazbel.com/.

IDEFIX. “Yeni Başlayanlar İçin Python,” 2021. http://www.idefix.com/Kitap/Yeni-Baslayanlar-Icin-Python/Egitim-Basvuru/Bilgisayar/urunno=0000000694609?gclid=CjwKCAiA8rnfBRB3EiwAhrhBGvtC3-OQEYO058lov4qtFRvJ5hyM1XbF80JosZXy4SBKxn1nGtQKoBoC33EQAvD_BwE.

Zhang, Yiqun, and Yiu-Ming Cheung. “A New Distance Metric Exploiting Heterogeneous Interattribute Relationship for Ordinal-and-Nominal-Attribute Data Clustering.” IEEE Transactions on Cybernetics, 2020, 1–14. https://doi.org/10.1109/TCYB.2020.2983073.