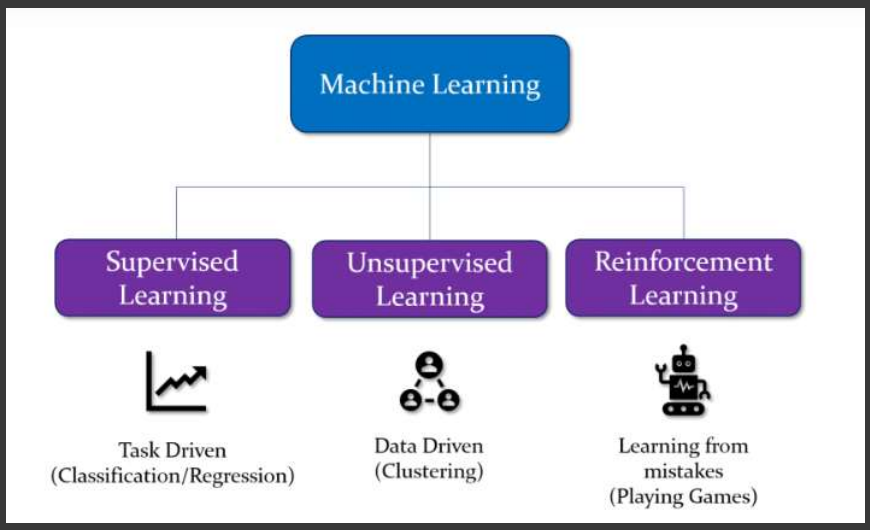
MAKİNE ÖĞRENMESİ: Sınıflandırma, Regresyon ve Kümeleme

Makine öğrenmesi 3 kategoride incelenebilir:

- Denetimli Öğrenme: Veri setlerinde etiketleme olan
- Denetimsiz Öğrenme: Veri setlerinde etiketleme olmayan
- Pekiştirmeli Öğrenme: Otonom araçlar, robotlar ile ilgilidir. Ödül-ceza sistemiyle robotun o sistemi öğrenmesi sağlanır.



HEDEFLER

- Makine öğrenimi çalışmaları gerçekleştirmek için popüler veri kümeleriyle scikit-learn kullanılabilir.
- Verileri görselleştirmek ve keşfetmek için Seaborn ve Matplotlib kullanılabilir.
- k-nearest neighbors sınıflandırması ve doğrusal regresyon ile denetimli makine öğrenimi gerçekleştirilebilir.
- Digits veri seti ile çoklu sınıflandırma gerçekleştirilebilir.
- Bir veri kümesi eğitim, test ve doğrulama kümelerine ayrılabilir.
- k-fold cross-validation ile model hiperparametreleri ayarlanabilir.
- Model performansı ölçülmelidir.
- Sınıflandırma tahmini isabetlerini ve ıskalamalarını gösteren bir karışıklık matrisi görüntülenebilir.
- California Housing veri kümesiyle çoklu doğrusal regresyon gerçekleştirilebilir.
- Iris ve Digits veri kümelerini iki boyutlu görselleştirmelere hazırlamak için PCA ve t-SNE ile boyutluluk azaltma gerçekleştirilebilir.
- K-means kümeleme ve Iris veri kümesiyle denetimsiz makine öğrenimi gerçekleştirilebilir.

Makine Öğrenimine Giriş

- Yapay zekanın en heyecan verici ve gelecek vaat eden alt alanlarından biridir.
- Büyük ve karmaşık bir konudur.

Makine Öğrenimi Nedir?

- Makinelerimizin (bilgisayarlarımızın) gerçekten öğrenmesini sağlayabilir miyiz? Bu sorunun cevabını araştırır.
- En önemli şey "veri"dir.

Tahmin

Aşağıdaki tahminlerin tümü bugün makine öğrenimi ile gerçekleşiyor:

- Hava tahminleri
- Kanser teşhisleri ve tedavi rejimlerini iyileştirme
- İş tahminleri
- Sahte kredi kartı satın alımlarını ve sigorta taleplerini tespit edin
- Evlerin muhtemelen hangi fiyatlara satılacağını, yeni filmlerin bilet satışlarını ve yeni ürün ve hizmetlerin beklenen gelirini tahmin etme
- Antrenörlerin ve oyuncuların daha fazla oyun ve şampiyonluk kazanmak için kullanabilecekleri en iyi stratejileri tahmin etme

Popüler Makine Öğrenimi Uygulamaları

Makine öğrenimi uygulamaları		
Anomali tespiti	Veri madenciliği sosyal medya (like Facebook, Twitter, LinkedIn)	Mortgage kredisi temerrütlerini tahmin etme
Chatbot'lar	Ekrandaki nesneleri algılama	Doğal dil çevirisi (İngilizce'den İspanyolca'ya, Fransızca'dan Japonca'ya vb.)
E-postaları spam veya spam değil olarak sınıflandırma	Verilerdeki kalıpları algılama	Tavsiye sistemleri ("bu ürünü alan kişiler aynı zamanda satın aldı...")

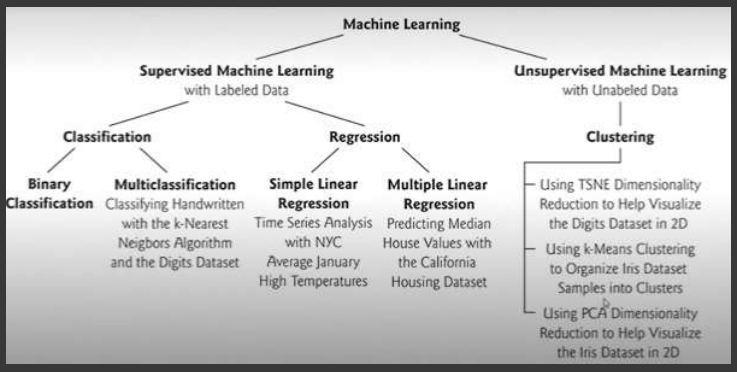
Makine öğrenimi uygulamaları		
Haber makalelerini spor, finans, politika vb. olarak sınıflandırmak	İlaç teşhisi	Sürücüsüz arabalar (genel olarak, otonom araçlar)
Bilgisayar görüşü ve görüntü sınıflandırması	Yüz tanıma	Duygu analizi (film incelemelerini olumlu, olumsuz veya tarafsız olarak sınıflandırmak gibi)
Kredi kartı dolandırıcılığı tespiti	Elyazısı tanıma	Spam filtreleme
Müşteri kaybı tahmini	Sigorta dolandırıcılığı tespiti	Hisse senedi fiyatı tahmini ve hava durumu tahmini gibi zaman serisi tahminleri
Veri sıkıştırma	Bilgisayar ağlarında izinsiz giriş tespiti	Ses tanıma
Veri keşfi	Pazarlama: Müşterileri kümelere ayırma	

Scikit-Learn

- En etkili makine öğrenimidir.
- Algoritmalarını tahmin ediciler olarak uygun şekilde paketler.
- Her biri kapsüllenmiştir, bu nedenle bu algoritmaların nasıl çalıştığına dair karmaşık ayrıntıları ve ağır matematiği görmezsiniz.
- Scikit-learn ve az miktarda Python kodu ile verileri analiz etmek, verilerden içgörüler çıkarmak ve en önemlisi tahminler yapmak için hızlı bir şekilde güçlü modeller oluşturmak.
- Verilerinizin bir alt kümesinde her bir modeli eğitmek için scikit-learn kullanılır, ardından modelin ne kadar iyi çalıştığını görmek için her modeli geri kalanı üzerinde test edilir.
- Modeller eğitildikten sonra, onları görmedikleri verilere dayalı tahminler yapmak üzere çalıştırılır.
- Daha çok ezberci işlerde kullanmış olduğunuz bilgisayar zeka özelliği kazanacaktır.

Projeniz için Hangi Scikit-Learn Tahmin Aracını Seçmelisiniz?

- Verileriniz üzerinde hangi modellerin en iyi performansı göstereceğini önceden bilmek zordur, bu nedenle genellikle birçok modeli dener ve en iyi performansı göstereni seçersiniz.
- Birçok modeli çalıştırmak ve en iyisini seçmek açısından popüler bir yaklaşımdır.
- Hangi modelin en iyi performansı gösterdiğini nasıl değerlendirebiliriz?
- Farklı türde veri kümeleri üzerinde birçok farklı modeli denemek isteyeceksiniz.
- Tahmin edicilerinde karmaşık matematiksel algoritmaların ayrıntılarını öğreneceksiniz, ancak deneyimle belirli veri kümeleri ve problem türleri için hangi algoritmaların en iyi olabileceğine aşina olacaksınız.
- Bu deneyimle bile, her yeni veri kümesi için en iyi modeli sezmeniz pek olası değildir.
- Yani scikit-learn, "hepsini denemenizi" kolaylaştırır.
- Modeller performanslarını raporlar, böylece sonuçları karşılaştırabilir ve en iyi performansa sahip modelleri seçebilirsiniz.



Denetimli Makine Öğrenimi (Supervised Machine Learning)

- Denetimli makine öğrenimi, sınıflandırma ve regresyon olmak üzere iki kategoriye ayrılır.
- Makine öğrenimi modellerini satırlardan ve sütunlardan oluşan veri kümeleri üzerinde eğitirsiniz.
- Her satır, bir veri örneğini temsil eder.
- Her sütun, o örneğin bir özelliğini temsil eder.
- Denetimli makine öğreniminde, her örneğin hedef adı verilen ("köpek" veya "kedi" gibi) ilişkili bir etiketi vardır.
- Bu, modellerinize sunduğunuz yeni veriler için tahmin etmeye çalıştığınız değerdir.

Veri Setleri (Data Sets)

- Her biri sınırlı sayıda özelliğe sahip az sayıda örneğe sahip bazı "toy" veri kümeleriyle çalışacaksınız.
- On binlerce örnek içeren zengin özelliklere sahip birkaç gerçek dünya veri kümesiyle de çalışılabilir.
- Büyük veri dünyasında, veri kümelerinde genellikle milyonlarca ve milyarlarca, hatta daha fazla örnek bulunur.
- Veri bilimi çalışmaları için çok sayıda ücretsiz ve açık veri seti mevcuttur.
- Scikit-learn gibi kitaplıklar, denemeniz için popüler veri kümelerini paketler ve çeşitli havuzlardan (openml.org gibi) veri kümelerini yüklemek için mekanizmalar sağlar.
- Dünya çapındaki hükümetler, işletmeler ve diğer kuruluşlar çok çeşitli konularda veri kümeleri sunar.
- Çeşitli makine öğrenimi teknikleri kullanarak birkaç popüler ücretsiz veri kümesiyle çalışılabilir.

Sınıflandırma (Classification)

- Scikit-learn ile birlikte gelen Digits veri setini analiz etmek için en basit sınıflandırma algoritmalarından biri olan k-nearest neighbors kullanacağız.
- Sınıflandırma algoritmaları, örneklerin ait olduğu ayrık sınıfları (kategorileri) tahmin eder.
- İkili sınıflandırma (Binary classification), bir e-posta sınıflandırma uygulamasında "spam" veya "spam değil" gibi iki sınıf kullanır.
- Çoklu sınıflandırma (Multi-classification), Digits veri kümesinde 0'dan 9'a kadar olan 10 sınıf gibi ikiden fazla sınıf kullanır.
- Film açıklamalarına bakan bir sınıflandırma şeması, onları "aksiyon", "macera", "fantastik", "romantizm", "tarih" ve benzeri olarak sınıflandırmaya çalışabilir.

Gerileme (Regression)

- Regresyon modelleri sürekli bir çıktı öngörür.
- Scikit-learn'ün LinearRegression tahmin aracını kullanarak basit bir lineer regresyon gerçekleştireceğiz.
- Ardından, scikit-learn ile birlikte gelen California Housing veri kümesiyle çoklu doğrusal regresyon gerçekleştirmek için bir LinearRegression tahmincisi kullanıyoruz.
- Ortalama oda sayısı, medyan ev yaşı, ortalama yatak odası sayısı ve medyan gelir gibi blok başına sekiz özelliği göz önünde bulundurarak ABD'deki bir ev bloğunun medyan ev değerini tahmin edeceğiz.
- Doğrusal Regresyon tahmincisi, varsayılan olarak, tek özellikli basit bir doğrusal regresyonla yapabileceğinizden daha karmaşık tahminler yapmak için bir veri kümesindeki tüm sayısal özellikleri kullanır.

Denetimsiz Makine Öğrenimi (Unsupervised Machine Learning)

- Ardından, kümeleme algoritmalarıyla denetimsiz makine öğrenimini tanıtacağız.
- Görselleştirme amacıyla Digits veri kümesinin 64 özelliğini ikiye sıkıştırmak için boyut indirgeme (scikit-learn'ün TSNE tahmincisi ile) kullanacağız.
- Bu, Digits verilerinin ne kadar iyi "kümelendiğini" görmemizi sağlayacaktır.
- Bu veri kümesi, postane bilgisayarlarının her bir harfi belirlenen posta koduna yönlendirmek için tanıması gerekenler gibi el yazısı rakamlar içerir.
- Her kişinin el yazısının benzersiz olduğu göz önüne alındığında, bu zorlu bir bilgisayarla görme sorunudur.
- Yine de, bu kümeleme modelini sadece birkaç satır kodla oluşturacağız ve etkileyici sonuçlar elde edeceğiz.
- Ve bunu, kümeleme algoritmasının iç işleyişini anlamak zorunda kalmadan yapacağız.
- Bu, nesne tabanlı programlamanın güzelliğidir.
- Açık kaynak Keras kitaplığını kullanarak güçlü derin öğrenme modelleri oluşturacağımız bir sonraki kısımda bu tür kullanışlı nesne tabanlı programlamayı yeniden göreceğiz.

K-Means Kümeleme ve İris Veri Kümesi (K-Means Clustering and the Iris Data Set)

- En basit denetimsiz makine öğrenimi algoritması olan k-means kümelemeyi sunacağız ve bunu yine scikit-learn ile birlikte gelen Iris veri kümesinde kullanacağız.
- Görselleştirme amacıyla Iris veri kümesinin dört özelliğini ikiye sıkıştırmak için boyut indirgeme (scikit-learn'ün PCA tahmincisi ile) kullanacağız.
- Veri kümesindeki üç İris türünün kümelenmesini göstereceğiz ve kümenin merkez noktası olan her kümenin ağırlık merkezinin grafiğini çizeceğiz.
- Son olarak, Iris veri kümesinin örneklerini etkili bir şekilde üç kümeye bölme yeteneklerini karşılaştırmak için birden çok kümeleme tahmincisi çalıştıracağız.
- k, normalde istenen küme sayısını belirtir.
- K-means, verileri o kadar çok kümeye bölmeye çalışarak çalışır.
- Birçok makine öğrenimi algoritmasında olduğu gibi, k-means yinelemelidir ve belirttiğiniz sayıyla eşleşmek için kümelerde kademeli olarak sıfırlanır.
- K-means kümeleme, etiketlenmemiş verilerde benzerlikler bulabilir.
- Bu, sonuçta denetimli öğrenme tahmincilerinin verileri işleyebilmesi için bu verilere etiket atamaya yardımcı olabilir.
- İnsanların etiketlenmemiş verilere etiket atamak zorunda kalmasının sıkıcı ve hataya açık olduğu ve dünyadaki verilerin büyük çoğunluğunun etiketlenmemiş olduğu göz önüne alındığında, denetimsiz makine öğrenimi önemli bir araçtır.

Büyük Veri ve Büyük Bilgisayar İşleme Gücü (Big Data and Big Computer Processing Power)

- Bugün mevcut olan veri miktarı zaten çok büyük ve katlanarak büyümeye devam ediyor.
- Dünyada son birkaç yılda üretilen veri, medeniyetin doğuşundan bu yana o ana kadar üretilen veri miktarına eşit.
- Genellikle büyük verilerden bahsederiz, ancak "büyük", verilerin ne kadar büyük hale geldiğini gerçekten açıklamak için yeterince güçlü bir terim olmayabilir.
- İnsanlar "Veriler içinde boğuluyorum ve bununla ne yapacağımı bilmiyorum" derdi.
- Makine öğrenimi ile artık "Beni büyük verilerle doldurun, böylece makine öğrenimi teknolojisini ve güçlü bilgi işlem yeteneklerini kullanarak içgörüler elde edip tahminlerde bulunabileyim" diyoruz.
- Bu, bilgi işlem gücünün patladığı ve bilgisayar belleği ile ikincil depolamanın kapasitesinin patladığı ve maliyetlerin önemli ölçüde düştüğü bir zamanda meydana geliyor.
- Bütün bunlar çözüm yaklaşımları hakkında farklı düşünmemizi sağlıyor.
- Artık bilgisayarları verilerden ve birçoğundan öğrenecek şekilde programlayabiliriz.
- Artık her şey verilerden tahmin yapmakla ilgili.

Scikit-Learn ile Birlikte Verilen Veri Setleri

- Scikit-learn ayrıca, https'de bulunan 20.000 veri kümesi gibi diğer kaynaklardan veri kümelerini yüklemek için yetenekler sağlar: https://openml.org.

Scikit-learn ile birlikte verilen veri kümeleri

"Toy" Veri Seti	Real-world Veri Seti
Boston ev fiyatları	Olivetti'nin yüzleri
Iris bitkileri	20 haber grubu metni
Diyabet	Vahşi yüz tanımada Etiketli Yüzler
El yazısı rakamların optik olarak tanınması	Orman örtüsü türleri
Linnerrud	RCV1
Wine recognition	Kddcup 99
Meme kanseri Wisconsin (teşhis)	California Housing

Tipik bir Veri Bilimi Çalışmasındaki Adımlar:

- Aşağıdakiler de dahil olmak üzere tipik bir makine öğrenimi vaka çalışmasının adımlarını uygulayacağız:
  - veri kümesi yüklenmesi
  - verileri pandas ve görselleştirmelerle keşfetme
  - verilerinizi dönüştürme (scikit-learn sayısal veriler gerektirdiğinden sayısal olmayan verileri sayısal verilere dönüştürme; bu bölümde "kullanıma hazır" veri kümeleri kullanıyoruz, ancak konuyu "Derin Öğrenme" bölümünde tekrar ele alacağız.)
  - eğitim ve test için verileri bölme
  - model oluşturma
  - modeli eğitmek ve test etmek
  - modeli ayarlama ve doğruluğunu değerlendirme
  - modelin daha önce görmediği canlı veriler üzerinde tahminler yapma
- Bunlar, makine öğrenimi için kullanmadan önce verilerinizi temizlemenin önemli adımlarıdır.

Örnek Olay (Case Study) : k-Nearest Neighbors ve Digits Veri Kümesi ile Sınıflandırma

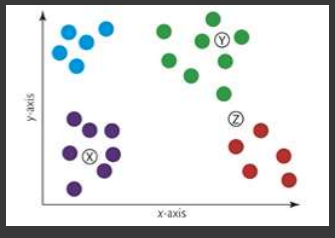
- Postayı verimli bir şekilde işlemek ve her mektubu doğru hedefe yönlendirmek için, posta hizmeti bilgisayarlarının el yazısı adları, adresleri ve posta kodlarını tarayabilmesi ve harfleri ve rakamları tanıyabilmesi gerekir.
- Scikit-learn, acemi programcıların bile bu tür makine öğrenimi sorunlarını yönetilebilir hale getirmesini sağlar.

Denetimli Makine Öğrenimi: Sınıflandırma (Supervised Machine Learning: Classification)

- Bir örneğin ait olduğu farklı sınıfı (kategoriyi) tahmin etmeye çalışmak
  - İkili sınıflandırma—iki sınıf (ör. "köpek" veya "kedi")
- Scikit-learn ile birlikte gelen Digits veri kümesi
  - 1797 tane elle yazılmış basamağı (0'dan 9'a) temsil eden 8'e 8 piksel görüntüler
- Hedef: Bir görüntünün hangi basamağı temsil ettiğini tahmin edin
  - Çoklu sınıflandırma—10 olası basamak (sınıflar)
- Etiketli verileri kullanarak bir sınıflandırma modeli eğitin; her basamağın sınıfını önceden bilin
- El yazısı rakamları tanımak için en basit makine öğrenimi sınıflandırma algoritmalarından biri olan k-nearest neighbors (k-NN) kullanacağız.

k-En Yakın Komşular Algoritması (k-Nearest Neighbors Algorithm) (k-NN)

- "mesafe" olarak en yakın k eğitim örneğine bakarak bir örneğin sınıfını tahmin edin.
- Dolu noktalar dört farklı sınıfı temsil eder:
  - A (mavi)
  - B (yeşil)
  - C (kırmızı)
  - D (mor)
- En çok "oyu" alan sınıf kazanır.
- Tek k değeri bağları önler, hiçbir zaman eşit sayıda oy yoktur.



Veri Kümesini load_digits Fonksiyonuyla Yükleme

- Digits örnekleri ve meta verileri içeren bir Bunch nesnesi döndürür.
- Bunch, veri kümesine özgü ek özniteliklere sahip bir sözlüktür.

```
1 from sklearn.datasets import load_digits
```

```
1 digits = load_digits()
```

Digits Veri Kümesinin Açıklamasını Görüntüleme

- Digits veri kümesi, UCI (University of California Irvine) ML elle yazılmış basamak veri kümesinin bir alt kümesidir.
  - Orijinal veri kümesi: 5620 örnekten 3823 tanesi eğitim için ve 1797 tanesi test içindir.
- Bir Bunch DESCR özniteliği, veri kümesinin açıklamasını içerir.
  - Her örnek, 0-16 piksel değerlerine sahip 8'e 8 görüntüyü temsil eden 64 özelliğe sahiptir.
  - Eksik değer yok (Eksik Özellik Değerleri)
- 64 özellik çok gibi görünebilir
  - Veri kümeleri yüzlerce, binlerce hatta milyonlarca özelliğe sahip olabilir.
  - Bunun gibi veri kümelerini işlemek, muazzam bilgi işlem yetenekleri gerektirebilir.

```
1 print(digits.DESCR)
```

```
.. _digits_dataset:

Optical recognition of handwritten digits dataset
--------------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 1797
    :Number of Attributes: 64
    :Attribute Information: 8x8 image of integer pixels in the range 0..16.
    :Missing Attribute Values: None
    :Creator: E. Alpaydin (alpaydin '@' boun.edu.tr)
    :Date: July; 1998

This is a copy of the test set of the UCI ML hand-written digits datasets
https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits

The data set contains images of hand-written digits: 10 classes where
each class refers to a digit.

Preprocessing programs made available by NIST were used to extract
normalized bitmaps of handwritten digits from a preprinted form. From a
total of 43 people, 30 contributed to the training set and different 13
to the test set. 32x32 bitmaps are divided into nonoverlapping blocks of
4x4 and the number of on pixels are counted in each block. This generates
an input matrix of 8x8 where each element is an integer in the range
0..16. This reduces dimensionality and gives invariance to small
distortions.

For info on NIST preprocessing routines, see M. D. Garris, J. L. Blue, G.
T. Candela, D. L. Dimmick, J. Geist, P. J. Grother, S. A. Janet, and C.
L. Wilson, NIST Form-Based Handprint Recognition System, NISTIR 5469,
1994.

.. topic:: References

  - C. Kaynak (1995) Methods of Combining Multiple Classifiers and Their
    Applications to Handwritten Digit Recognition, MSc Thesis, Institute of
    Graduate Studies in Science and Engineering, Bogazici University.
  - E. Alpaydin, C. Kaynak (1998) Cascading Classifiers, Kybernetika.
  - Ken Tang and Ponnuthurai N. Suganthan and Xi Yao and A. Kai Qin.
    Linear dimensionalityreduction using relevance weighted LDA. School of
    Electrical and Electronic Engineering Nanyang Technological University.
    2005.
  - Claudio Gentile. A New Approximate Maximal Margin Classification
    Algorithm. NIPS. 2000.
```

Örnek ve Hedef Büyüklüklerini Kontrol Etme

- Grup nesnesinin verileri ve hedef öznitelikleri NumPy dizileridir:
  - Veri dizisi: Her biri 0 (beyaz) ila 16 (siyah) değerlerine sahip 64 özelliğe sahip, piksel yoğunluklarını temsil eden 1797 örnek (rakamlı görüntüler)

  - Hedef dizi: Her görüntünün hangi basamağı temsil ettiğini gösteren görüntülerin etiketleri (sınıflar)

```
1 digits.target[::100]  # her 100. örneğin hedef değerleri
```

```
array([0, 4, 1, 7, 4, 8, 2, 2, 4, 4, 1, 9, 7, 3, 2, 1, 2, 5])
```

```
1 digits.data.shape # 1797 satır, 64 sütun
```

```
(1797, 64)
```

```
1 digits.target.shape
```

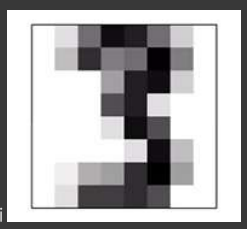
```
(1797,)
```

Örnek Digits Görüntüsü

- Görüntüler piksel cinsinden iki boyutludur:
  - genişlik
  - yükseklik
- Digits veri kümesinin Bunch nesnesinin bir resimsel özniteliği var.
  - Her öge, bir basamaklı görüntünün piksel yoğunluklarını temsil eden 8'e 8'lik bir dizidir.
- Scikit-learn, yoğunluk değerlerini NumPy tipi float64 olarak saklar.

```
1 digits.images[13]  # 13. dizinde örnek görüntü için gösterim
```

```
array([[ 0.,  2.,  9., 15., 14.,  9.,  3.,  0.],
       [ 0.,  4., 13.,  8.,  9., 16.,  8.,  0.],
       [ 0.,  0.,  0.,  6., 14., 15.,  3.,  0.],
       [ 0.,  0.,  0., 11., 14.,  2.,  0.,  0.],
       [ 0.,  0.,  0.,  2., 15., 11.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  2., 15.,  4.,  0.],
       [ 0.,  1.,  5.,  6., 13., 16.,  6.,  0.],
       [ 0.,  2., 12., 12., 13., 11.,  0.,  0.]])
```

digits.images[13] görselleştirmesi 

Verileri Scikit-Learn ile Kullanıma Hazırlama

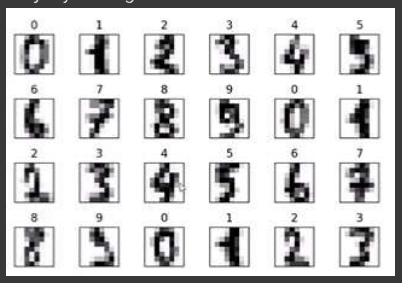
- Scikit-learn tahmin edicileri, örneklerin **iki boyutlu bir kayan noktalı değerler(floating-point) dizisinde** (veya listeler veya **pandas DataFrame listesi**) saklanmasını gerektirir:
  - Her satır bir örneği temsil eder.
  - Belirli bir satırdaki her sütun, o örnek için bir özelliği temsil eder.
- Çok boyutlu veri örnekleri tek boyutlu bir diziye düzleştirilmelidir.
- Kategorik özellikler için (örneğin, "spam" veya "spam değil" gibi dizeler), bu özellikleri **one-hot** kodlama olarak bilinen sayısal değerler halinde önceden işlemeniz gerekir. (daha sonra derin öğrenmede ele alınacaktır)
- load_digits, önceden işlenmiş verileri makine öğrenimi için hazır hale getirir.
- 8'e 8 dizi digits.images[13], 1'e 64 dizi digits.data[13]'e karşılık gelir:

```
1 digits.data[13]
```

```
array([ 0.,  2.,  9., 15., 14.,  9.,  3.,  0.,  0.,  4., 13.,  8.,  9.,
       16.,  8.,  0.,  0.,  0.,  0.,  6., 14., 15.,  3.,  0.,  0.,  0.,
        0., 11., 14.,  2.,  0.,  0.,  0.,  0.,  0.,  2., 15., 11.,  0.,
        0.,  0.,  0.,  0.,  0.,  2., 15.,  4.,  0.,  0.,  1.,  5.,  6.,
       13., 16.,  6.,  0.,  0.,  2., 12., 12., 13., 11.,  0.,  0.])
```

Verileri Görselleştirme

- Veri keşfi olarak adlandırılan verinize her zaman aşina olun.
- Veri setinin ilk 24 görüntüsünü Matplotlib ile görselleştirelim.
- El yazısı rakam tanıma probleminin ne kadar zor olduğunu görmek için birinci, üçüncü ve dördüncü sıralardaki 3'lerin görüntüleri arasındaki varyasyonları göz önünde bulundurun ve birinci, üçüncü ve dördüncü sıralardaki 2'lerin görüntülerine bakın.



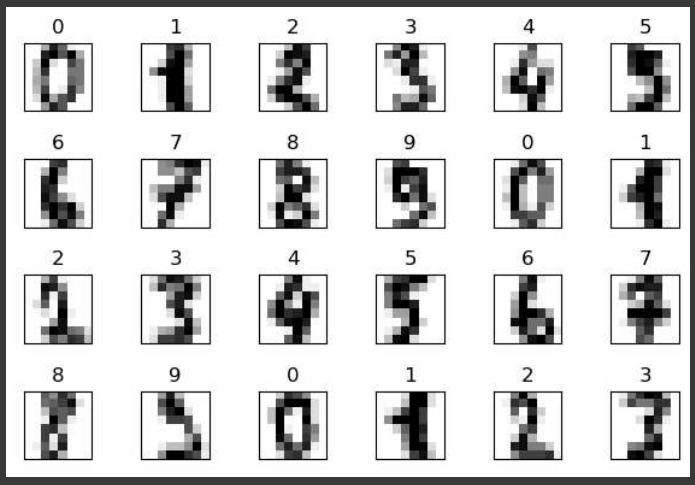
Diyagramı Oluşturma

- Renk haritası plt.cm.gray_r, beyaz için 0 olan gri tonlama içindir.
- Matplotlib'in renk haritası adları — plt.cm nesnesi veya 'gray_r' gibi bir dizi aracılığıyla erişilebilir.

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

```
figure, axes = plt.subplots(nrows=4, ncols=6, figsize=(6, 4))

for item in zip(axes.ravel(), digits.images, digits.target):
    axes, image, target = item
    axes.imshow(image, cmap=plt.cm.gray_r)
    axes.set_xticks([])  # x ekseni onay işaretlerini kaldır
    axes.set_yticks([])  # y ekseni onay işaretlerini kaldır
    axes.set_title(target)
plt.tight_layout()
```



Verileri Eğitim ve Test için Bölme (Splitting the Data for Training and Testing)

- Tipik olarak bir veri kümesinin alt kümesiyle bir model eğitin.
- Test için bir kısmını kaydedin, böylece görünmeyen verileri kullanarak bir modelin performansını değerlendirebilirsiniz.
- train_test_split işlevi, verileri rastgele hale getirmek için karıştırır, ardından veri dizisindeki örnekleri ve hedef dizideki hedef değerleri eğitim(train) ve test kümelerine böler.
  - Karıştırma, eğitim(train) ve test setlerinin benzer özelliklere sahip olmasını sağlamaya yardımcı olur.
  - İlk ikisinin eğitim(train) ve test kümelerine bölünmüş örnekler olduğu ve son ikisinin eğitim(train) ve test kümelerine bölünmüş karşılık gelen hedef değerler olduğu dört ögeden oluşan bir tuple döndürür.
- Ortak düşünce:
  - Büyük X örnekleri temsil eder.
  - Küçük y, hedef değerleri temsil eder.

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(
    digits.data, digits.target, random_state=11)  # tekrar üretilebilirlik için random_state
```

- Scikit-learn paketli sınıflandırma veri kümeleri dengeli sınıflara sahiptir.
  - Örnekler sınıflar arasında eşit olarak bölünür.
  - Dengesiz sınıflar yanlış sonuçlara yol açabilir.

Eğitim(Train) ve Test Seti Boyutları

- Train_test_split varsayılan olarak verilerin %75'ini eğitim(train) için ve %25'ini test için ayırır.

```
X_train.shape
```

```
(1347, 64)
```

```
X_test.shape
```

```
(450, 64)
```

```
y_train.shape
```

```
(1347,)
```

```
1 y_test.shape
```

```
(450,)
```

Modeli Oluşturma

- Scikit-learn'de modellere tahmin edici denir.
- KNeighborsClassifier tahmincisi k-en yakın komşu (k-nearest neighbors) algoritmasını uygular.

```
1 from sklearn.neighbors import KNeighborsClassifier
```

```
1 knn = KNeighborsClassifier()
```

Modeli KNeighborsClassifier Object'in fit yöntemiyle eğitme

- Tahmin ediciye örnek eğitim seti (X_train) ve hedef eğitim(train) seti (y_train) yükleyin.

```
1 knn.fit(X=X_train, y=y_train)
```

```
KNeighborsClassifier()
```

- n_neighbors, k-en yakın komşular algoritmasında k'ye karşılık gelir.
- `KNeighborsClassifier` default settings

- fit normalde verileri bir tahmin ediciye yükler, ardından bir modeli eğitmek için verilerden öğrenen sahne arkasında karmaşık hesaplamalar gerçekleştirir.
- KNeighborsClassifier'ın sığdırma yöntemi yalnızca verileri yükler.
  - İlk öğrenme süreci yok.
  - Tahmin edici ağırdır, iş yalnızca onu tahmin yapmak için kullandığınızda gerçekleştirilir.
- Pek çok modelin dakikalar, saatler, günler veya daha uzun sürebilen önemli eğitim aşamaları vardır.
  - Yüksek performanslı GPU'lar ve TPU'lar, model eğitim süresini önemli ölçüde azaltabilir.

KNeighborsClassifier'ın tahmin yöntemiyle Digits Sınıflarını Tahmin Etme

- Her test görüntüsünün tahmin edilen sınıfını içeren bir dizi döndürür.

```
1 predicted = knn.predict(X=X_test)
```

Show hidden output

```
1 expected = y_test
```

- İlk 20 test numunesi için tahmin edilen rakamlara karşı beklenen rakamlar. 18.dizine bakınız:

```
1 predicted[:20]
```

```
array([0, 4, 9, 9, 3, 1, 4, 1, 5, 0, 4, 9, 4, 1, 5, 3, 3, 8, 5, 6])
```

```
1 expected[:20]
```

```
array([0, 4, 9, 9, 3, 1, 4, 1, 5, 0, 4, 9, 4, 1, 5, 3, 3, 8, 3, 6])
```

- Test setinin tamamı için tüm yanlış tahminleri bulun:

```
1 wrong = [(p, e) for (p, e) in zip(predicted, expected) if p ≠ e]
```

```
1 wrong
```

```
[(5, 3),
 (8, 9),
 (4, 9),
 (7, 3),
 (7, 4),
 (2, 8),
 (9, 8),
 (3, 8),
 (3, 8),
 (1, 8)]
```

- Yani, 450 test örneğinden yalnızca 10 tanesi yanlış tahmin edildi.

Model Doğruluğunu Ölçmek için Metrikler

Tahminci Yöntem puanı

- Tahmin edicinin test verilerinde ne kadar iyi performans gösterdiğinin bir göstergesini döndürür.
- Sınıflandırma tahmin edicileri için, test verilerinin tahmin doğruluğunu döndürür.

```
print(f'{knn.score(X_test, y_test):.2%}')
```

Show hidden output

- Varsayılan k değeri 5 olan kNeighborsClassifier, yalnızca tahmin edicinin varsayılan parametrelerini kullanarak %97,78 tahmin doğruluğu elde etti.
- k için en uygun değeri belirlemeye çalışmak için hiperparametre ayarını kullanabilir.

Karışıklık Matrisi (Confusion Matrix)

- Belirli bir sınıf için doğru ve yanlış tahmin edilen değerleri (isabetler ve ıskalamalar) gösterir.

```
from sklearn.metrics import confusion_matrix
```

```
confusion = confusion_matrix(y_true=expected, y_pred=predicted)
```

```
confusion
```

```
array([[45,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 45,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0, 54,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 42,  0,  1,  0,  1,  0,  0],
       [ 0,  0,  0,  0, 49,  0,  0,  1,  0,  0],
       [ 0,  0,  0,  0,  0, 38,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0, 42,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 45,  0,  0],
       [ 0,  1,  1,  2,  0,  0,  0,  0, 39,  1],
       [ 0,  0,  0,  0,  1,  0,  0,  0,  1, 41]], dtype=int64)
```

- Sol üstten sağ alta doğru ana köşegende gösterilen doğru tahminleri gösterir.
- Asal köşegen üzerinde olmayan sıfır olmayan değerler, yanlış tahminleri gösterir.
- Her satır bir ayrı sınıfı temsil eder. (0-9)
- Sütunlar, kaç tane test örneğinin 0-9 sınıflarında sınıflandırıldığını belirtir.
- Satır 0, basamak sınıfı 0'ı gösterir; tüm 0'lar doğru tahmin edilmiştir.

  > `[45, 0, 0, 0, 0, 0, 0, 0, 0, 0]`

- Satır 8 rakam sınıfı 8'i gösterir—beş 8 yanlış tahmin edildi.

  > `[ 0, 1, 1, 2, 0, 0, 0, 0, 39, 1]`
  >
  > - 8 saniyenin %88,63'ü (44'ün 39'u) doğru tahmin edildi.
  > - 8s tanımak daha zor.

Karışıklık Matrisini Görselleştirme

- Bir ısı haritası(heat map), değerleri renkler olarak görüntüler.
- Karışıklık matrisini bir DataFrame'e dönüştürün, ardından grafiğini çizin.
- Ana diyagonal ve yanlış tahminler, ısı haritasında(heat map'te) kolayca göze çarpıyor.

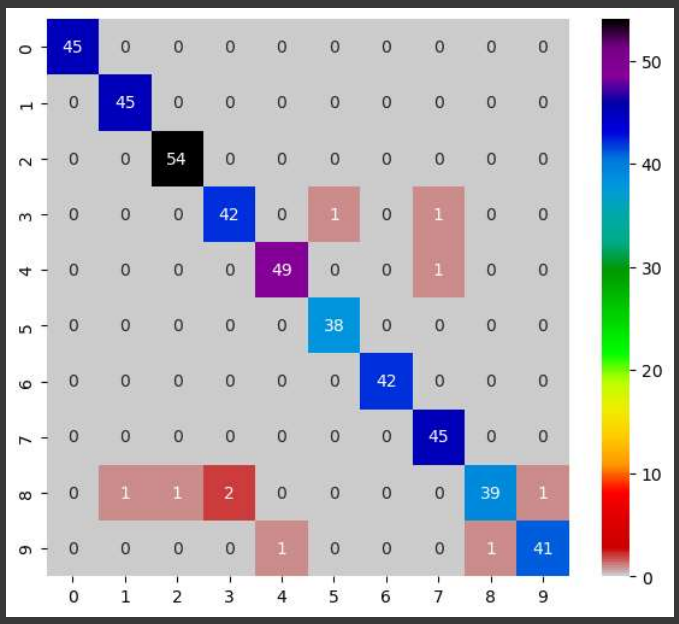
```
import pandas as pd
```

```
confusion_df = pd.DataFrame(confusion, index=range(10), columns=range(10))
```

```
confusion_df
```

```
import seaborn as sns
```

```
figure = plt.figure(figsize=(7, 6))
axes = sns.heatmap(confusion_df, annot=True,
                   cmap=plt.cm.nipy_spectral_r)
```



K-Katlama Çapraz Doğrulama (K-Fold Cross-Validation)

- Tüm verilerinizi eğitim ve test için kullanır.
- Modelinizin ne kadar iyi tahminler yapacağına dair daha iyi bir fikir verir.
- Veri kümesini k eşit boyutlu kıvrımlara böler (k-en yakın komşular algoritmasındaki k ile ilgili değildir.)
- Modelinizi k – 1 kat ile tekrar tekrar eğitir ve modeli kalan kat ile test eder.
- 1'den 10'a kadar numaralandırılmış katlarla k = 10 kullanmayı düşünün;
    - 1-9 katlamalarla çalışın, ardından 10 katlamayla test edin
    - 1–8 ve 10. katlarla antrenman yapın, ardından 9. katla test edin
    - 1–7 ve 9–10 katlamalarla çalışın, ardından 8 katlama ile test edin
    - ...

KFold Sınıfı

- KFold sınıfı ve cross_val_score işlevi k-katlı çapraz doğrulama gerçekleştirir.
- n_splits=10 kat sayısını belirtir.
- shuffle=True, verileri katlara ayırmadan önce rastgele sıralar.
    - Örnekler sıralanabilir veya gruplandırılabilirse özellikle önemlidir (daha sonra göreceğimiz Iris veri kümesinde olduğu gibi)

```
from sklearn.model_selection import KFold
```

```
kfold = KFold(n_splits=10, random_state=11, shuffle=True)
```

Modelinizi Eğitmek ve Test Etmek İçin cross_val_score İşlevini Çağırma

- estimator=knn — doğrulamak için tahmin edici
- X=digits.data — eğitim ve test için kullanılacak örnekler
- y=digits.target — örnekler için hedef tahminleri
- cv=kfold — eğitim ve test için örneklerin ve hedeflerin nasıl bölüneceğini tanımlayan çapraz doğrulama oluşturucu

```
from sklearn.model_selection import cross_val_score
```

```
scores = cross_val_score(estimator=knn, X=digits.data, y=digits.target, cv=kfold)
```

Show hidden output

- En düşük doğruluk %97,78 idi; biri %100'dü.

```
scores # her kat için doğruluk puanları dizisi
```

```
array([0.97777778, 0.99444444, 0.98888889, 0.97777778, 0.98888889,
       0.99444444, 0.97777778, 0.98882682, 1.        , 0.98882682])
```

```
print(f'Mean accuracy: {scores.mean():.2%}')
```

```
Mean accuracy: 98.78%
```

- Verilerin %75'i ile modeli eğittiğimizde ve daha önce %25 ile modeli test ettiğimizde elde ettiğimiz %97,78'den bile daha iyi ortalama doğruluktur.

En İyisini Bulmak İçin Birden Çok Modeli Çalıştırma

- Belirli bir veri kümesi için hangi makine öğrenimi modellerinin en iyi performansı göstereceğini önceden bilmek zordur.
  - Özellikle nasıl çalıştıklarının ayrıntılarını gizlediklerinde daha da zordur.
- KNeighborsClassifier, rakam görüntülerini yüksek bir doğruluk derecesi ile tahmin etse de, diğer tahmin edicilerin daha da doğru olması mümkündür.
- KNeighborsClassifier, SVC ve GaussianNB'yi karşılaştıralım.

```
from sklearn.svm import SVC
```

```
from sklearn.naive_bayes import GaussianNB
```

- Tahmin edicileri oluşturun:
  - Scikit-learn uyarısından kaçınmak için, SVC tahmin aracını oluştururken bir anahtar sözcük argümanı sağladık.

```
estimators = {
    'KNeighborsClassifier': knn,
    'SVC': SVC(gamma='scale'),
    'GaussianNB': GaussianNB()}
```

- Modelleri yürütün: *

```
for estimator_name, estimator_object in estimators.items():
    kfold = KFold(n_splits=10, random_state=11, shuffle=True)
    scores = cross_val_score(estimator=estimator_object,
        X=digits.data, y=digits.target, cv=kfold)
    print(f'{estimator_name:>20}: ' +
          f'mean accuracy={scores.mean():.2%}; ' +
          f'standard deviation={scores.std():.2%}')
```

Show hidden output

- KNeighborsClassifier ve SVC tahmincilerinin doğrulukları aynıdır, bu nedenle en iyisini belirlemek için her birinde hiperparametre ayarı yapmak isteyebiliriz.

Hiperparametre Ayarı

- Gerçek dünyadaki makine öğrenimi çalışmalarında, mümkün olan en iyi tahminleri üreten değerleri seçmek için hiperparametreleri ayarlamak isteyeceksiniz.
- kNN algoritmasında k için en iyi değeri belirlemek için farklı değerler deneyin ve performansı karşılaştırın.
- Scikit-learn ayrıca otomatikleştirilmiş hiperparametre ayarlama yeteneklerine sahiptir.

- 1'den 19'a kadar tek k değerleri olan KNeighboursClassifiers oluşturun.
- Her birinde k-katlı çapraz doğrulama gerçekleştirin.

```
for k in range(1, 20, 2):  # k is an odd value 1-19; odds prevent ties
    kfold = KFold(n_splits=10, random_state=11, shuffle=True)
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(estimator=knn,
        X=digits.data, y=digits.target, cv=kfold)
    print(f'k={k:<2}; mean accuracy={scores.mean():.2%}; ' +
          f'standard deviation={scores.std():.2%}')
```

Show hidden output

- Makine öğrenimi, özellikle büyük veri(big data) ve derin öğrenmede(deep learning) maliyetsiz değildir.
- Hesaplama süresi k ile hızla büyür, çünkü k-NN'nin en yakın komşuları bulmak için daha fazla hesaplama yapması gerekir.
- Çapraz doğrulama gerçekleştirmek ve sonuçları zamanlamak için cross_validate işlevini kullanabilir.