DATA MİNİNG MİDTERM PROJECT

| **Name** | **Surname** |  |  | **Number** |
| --- | --- | --- | --- | --- |
| Görkem Yahya | Bakan |  |  | 210201079 |
| Beyzanur | Kurt |  |  | 210201066 |
| Muhiddin | Fırat |  |  | 210201303 |
| Orhan Efe | Bayrak |  |  | 210201013 |
| Harun | Korkmaz |  |  | 210201006 |
| Yasin Furkan | Abasız |  |  | 210201005 |
| Muhammet Salih | Hasılcıo |  |  | 210201088 |
| Ömer Faruk | Semerci |  |  | 210201010 |
| Berkay | Genceroğlu |  |  | 210201054 |

İçindekiler

İÇERİK SAYFA

1 – Giriş : Projenin Amacı 3

2 - Veri Setimizin Özellikleri 4

3 - Çıktılar : Kod Çıktıları 5

4 – Çıktı ve Tartışma 6

5 – Kod Açıklamaları: 7

5.1-KNN Kodu Açıklaması 7-16

5.2-SVM Kodu Açıklaması 17-27

5.3-Veriye Nan Değerleri Ekleme Kodu Açıklaması 28-30

5.4-Veride Nan yerine Ortalama Değerini Ekleme Kodu Açıklaması 31-34

Giriş

Projenin Amacı:

**Bu proje, bir sayısal veri seti üzerinde veri işleme ve sınıflandırma tekniklerini kullanarak aşağıdaki adımları gerçekleştirmeyi hedeflemektedir:**

**1. Başlangıç Sınıflandırması:**

- Orijinal veri seti üzerinde SVM ve KNN algoritmalarını kullanarak sınıflandırma işlemi gerçekleştirilmiştir. Confusion matrix (karmaşıklık matrisi) ile sonuçlar analiz edilmiştir.

**2. NaN Değerlerle Değiştirme**:

- Veri setindeki belirli değerler rastgele seçilerek `NaN` değerler ile değiştirilmiştir. Bu adımın amacı, eksik verilerle başa çıkma yöntemlerini incelemektir.

**3. NaN Değerlerin Ortalama ile Doldurulması:**

- İkinci adımda oluşturulan `NaN` değerlerin, ilgili sütunun ortalaması ile doldurulması sağlanmıştır. Bu adımda elde edilen sonuçlar incelenmiştir.

**4. Sınıflandırma - Manipüle Edilmiş Veri:**

- Üçüncü adımda işlenmiş veri seti üzerinde tekrar SVM ve KNN algoritmaları ile sınıflandırma işlemleri gerçekleştirilmiştir. Karmaşıklık matrisi ile sonuçlar analiz edilmiştir.

VERİ SETİ

**Veri Kümesi Açıklaması**

**Bu veri kümesi, kullanıcıların demografik bilgilerini, tahmini gelirlerini ve İngilizce bilme seviyelerine ilişkin bilgileri içermektedir. Veri kümesindeki her sütunun açıklaması şu şekildedir:**

* **KullaniciID: Her kullanıcıya atanmış benzersiz bir kimlik numarasıdır.**
* **Cinsiyet: Kullanıcının cinsiyetini ifade eden ikili bir değer (1 erkek, 0 kadın).**
* **Yas: Kullanıcının yaşı, tam sayı olarak belirtilmiştir.**
* **TahminiMaas: Kullanıcının tahmini maaşı, para birimi cinsinden verilmiştir.**
* **IngilizceBilme: Kullanıcının İngilizce bilme seviyesi:**
  + **0-20 yaş: İyi seviyede İngilizce bilmektedir (değer: 0).**
  + **20-40 yaş: Orta seviyede İngilizce bilmektedir (değer: 1).**
  + **40 yaş ve üstü: Başlangıç seviyede İngilizce bilmektedir (değer: 2).**

**Bu veri kümesi, yaş gruplarına göre İngilizce bilme seviyesinin gelirle ilişkisini incelemek, ayrıca demografik bilgilerin analizini yapmak için uygundur. Her satır bir kullanıcıyı temsil eder ve bu kullanıcıya ait özellikler arasında İngilizce bilme seviyesi yaşa bağlı olarak belirlenmiştir.**

1. **İlk veri setimizin ismi “DataMiningDataSet”**
2. **İkinci random şekilde Nan yazdığımız veri setimizin adı “DataMiningDataSetContainsNan”**
3. **Üçüncü veri setimiz Nan değerler yerine Ortalam(mean) yazılmış halidir adı “DataMiningDataSetContainsNan”**

Çıktılar

Kod Çıktıları

**KNN-BEFORE**

Confusion Matrix:  
[[11 3 0]  
 [ 2 72 1]  
 [ 0 5 66]]

Accuracy: 0.93125

**KNN-AFTER**

Confusion Matrix:  
[[11 2 0]  
 [ 2 62 7]  
 [ 0 2 74]]

Accuracy: 0.91875

**SVM-BEFORE**

Confusion Matrix:  
[[14 0 0]  
 [ 1 74 0]  
 [ 0 0 71]]

Accuracy: 0.99375

**SVM-AFTER**

Confusion Matrix:  
[[14 0 0]  
 [ 1 74 0]  
 [ 0 0 71]]

Accuracy: 0.99375

Çıktı ve Tartışma

**Sonuçların Açıklamaları:**

1. **KNN Modeli:**
   * KNN modelinin "BEFORE" (öncesi) ve "AFTER" (sonrası) sonuçlarında görülen en önemli farklar, doğru sınıflandırılan örneklerin sayısındaki değişikliktir.
   * **"BEFORE"** aşamasında, özellikle ikinci sınıf için (orta seviyede İngilizce bilme) 72 doğru sınıflama yapılmışken, **"AFTER"** aşamasında bu sayı 62'ye düşmüş, aynı zamanda yanlış sınıflandırmaların sayısı artmıştır (7 yanlış sınıflama). Bu, manipüle edilmiş verilerin model üzerinde daha fazla hata yapmasına yol açtığını gösteriyor.
   * **Accuracy**: KNN modelinin doğruluk oranı "BEFORE" aşamasında %93.125 iken, "AFTER" aşamasında %91.875'e düşmüştür. Bu fark, manipülasyonun modelin performansını biraz olumsuz etkilediğini gösteriyor.
2. **SVM Modeli:**
   * SVM modelinde ise **"BEFORE"** ve **"AFTER"** sonuçları arasında herhangi bir fark yoktur. Her iki durumda da tüm sınıflar doğru şekilde sınıflandırılmıştır ve **Accuracy** değeri %99.375'tir.
   * Manipülasyonun SVM modelinin performansını etkilemediği, modelin veri kümesine karşı güçlü bir genel doğruluğa sahip olduğu gözlemleniyor.

**Yorumlar:**

* **KNN Modelindeki Farklar**: KNN modelinde, "AFTER" verisi ile "BEFORE" verisi arasında doğruluk oranı düşerken, yanlış sınıflandırmaların sayısındaki artış da dikkat çekici. Bu, manipülasyonun modelin performansını biraz kötüleştirdiğini gösteriyor. KNN gibi mesafeye dayalı algoritmalar, verilerdeki değişimlere oldukça duyarlıdır, özellikle veri üzerinde yapılan manipülasyonlar modelin karar süreçlerini etkileyebilir.
* **SVM Modelindeki Fark Yok**: SVM modeli, manipülasyona karşı daha dayanıklı görünüyor. Sonuçlar arasındaki fark yok denecek kadar küçük ve doğruluk oranı yüksek. SVM, daha güçlü bir genelleme yeteneğine sahip olabilir ve manipülasyondan daha az etkileniyor gibi görünüyor. Bu, SVM'in sınıflandırma sorunlarında daha stabil ve etkili bir model olabileceğini gösteriyor.

Sonuç olarak, **KNN** modelindeki sonuçlar manipülasyona daha duyarlı ve performans kaybı yaşanırken, **SVM** modelinin manipulasyona karşı daha dayanıklı olduğu ve çok az etkilenerek aynı doğruluk oranını koruduğu görülüyor. Bu, veri kümesindeki değişikliklerin hangi modele daha fazla etki ettiğini anlamak açısından önemli bir gözlemdir.

KNN KODU AÇIKLAMASI

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**1. pandas**

Pandas, Python'da veriyle çalışırken en çok kullanılan kütüphanelerden biridir. Bu kütüphane, veriyi tablolar halinde düzenler. Mesela, bir Excel dosyasındaki veriler gibi. Pandas, bu veriyi çok rahat bir şekilde okuyup, üzerinde işlem yapmamızı sağlar. Örneğin, bir dosyadan veriyi almak, eksik verileri bulmak, verileri sıralamak ya da bazı verileri çıkarıp yeni bir dosya oluşturmak için kullanılır. Pandas sayesinde veriyi çok kolay bir şekilde analiz edebiliriz.

**2. train\_test\_split**

Bu fonksiyon, verimizi ikiye ayırmamıza yardımcı olur: Bir kısmını modelimizi eğitmek için, diğer kısmını ise modelimizin doğruluğunu test etmek için kullanırız. Yani, modelin öğrenmesi için bazı veriler kullanılır, ancak modelin doğru çalışıp çalışmadığını görmek için bu verilerle hiç karşılaşmamış yeni verilerle test yapılır. Bu sayede modelin genelleme yeteneğini test etmiş oluruz. train\_test\_split, veriyi rasgele ve belirli bir oranda ayırmamıza olanak sağlar (örneğin, %80 eğitim, %20 test gibi). Bu, modelimizin doğru öğrenip öğrenmediğini anlamamıza yardımcı olur.

**3. StandardScaler**

Bazen verilerimiz çok farklı ölçekte olur. Mesela, bir sütunlarda uzunluklar (metre cinsinden) ve diğerlerinde yaş gibi küçük sayılar olabilir. Bu durumda, büyük sayılar modelin üzerinde daha fazla etkili olabilir, bu da modelin dengesiz çalışmasına yol açar. İşte burada StandardScaler devreye girer. Bu fonksiyon, verilerin her birini benzer bir ölçekte yapar, yani verilerin ortalamasını sıfır ve standart sapmasını bir yapar. Bu sayede, tüm veriler eşit şekilde modellenir ve model daha doğru çalışır.

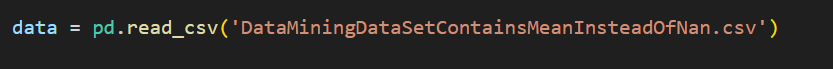
**4. KNeighborsClassifier**

KNN (K-En Yakın Komşu), basit ama etkili bir sınıflandırma algoritmasıdır. Bu algoritma, yeni bir veri noktasını sınıflandırmak için, o noktaya en yakın olan diğer verileri inceler. Yani, bu noktayı en yakın 3 ya da 5 veri ile karşılaştırarak hangi sınıfa ait olduğunu tahmin eder. KNN'nin en büyük avantajı, öğrenme sırasında fazla bir şey yapmamasıdır. Sadece tahmin yaparken en yakın komşuları arar. Ama çok fazla veriye sahip olduğumuzda, bu işlem biraz yavaşlayabilir. KNN'nin iyi çalışabilmesi için verilerin düzgün bir şekilde ölçeklendirilmesi gerekir çünkü mesafeyi ölçerken her özelliğin etkisi eşit olmalıdır.

**5. confusion\_matrix ve accuracy\_score**

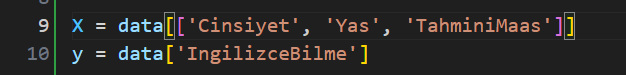
Bu iki metrik, modelimizin ne kadar doğru çalıştığını ölçmek için kullanılır. Confusion Matrix (karmaşıklık matrisi), modelimizin doğru ve yanlış tahminlerini bir tabloya döker. Bu tablonun bize söylediği şudur: Modelimiz hangi sınıflarda doğru tahmin yapmış, hangi sınıflarda yanlış yapmış? Örneğin, bir model bir resmi "kedili" olarak doğru sınıflandırmışsa, bu doğru bir tahmindir ve bunu gösteririz. Ama eğer bu resmi "köpek" olarak yanlış sınıflandırmışsa, bu yanlış bir tahmindir ve bunu da gösteririz.

Accuracy Score ise basitçe, modelin doğru yaptığı tahminlerin oranıdır. Yani, doğru tahminlerin toplam tahminlere oranını verir. Mesela modelimiz 100 tahminden 85’ini doğru yaptıysa, accuracy %85 olur. Ancak, eğer veri dengesizse, yani bir sınıf çok daha fazla sayıda varsa, yüksek accuracy her zaman iyi bir sonuç olmayabilir. Bu yüzden confusion matrix, modelin başarısını daha ayrıntılı şekilde gösterir.



Bu kod satırı, Python'da pandas kütüphanesini kullanarak bir CSV dosyasını okuma işlemidir :

1. **pd.read\_csv()**:
   * **read\_csv()**: Bu, pandas kütüphanesinin, bir CSV (Comma Separated Values - Virgülle Ayrılmış Değerler) dosyasını okumasını sağlayan bir fonksiyondur. CSV dosyaları genellikle tabular (tablolar halinde) verileri depolamak için kullanılır, yani veriler satır ve sütunlar halinde düzenlenir. Bu fonksiyon sayesinde, bir CSV dosyasındaki verileri Python'da kolayca kullanabileceğimiz bir formata (pandas DataFrame) dönüştürmüş oluruz.
   * **pd**: Burada pd, pandas kütüphanesinin kısaltmasıdır. Pandas’ı genellikle import pandas as pd şeklinde import (içe aktarma) ederiz ve böylece pd ile pandas fonksiyonlarına erişebiliriz.
2. **Dosya Yolu ('DataMiningDataSetContainsMeanInsteadOfNan.csv')**:
   * 'DataMiningDataSetContainsMeanInsteadOfNan.csv' kısmı, okuma işlemi yapılacak CSV dosyasının adını ve yolunu belirtir. Bu dosya, verileri içeren bir dosya olmalı ve formatı CSV (virgülle ayrılmış değerler) olmalıdır. Dosyanın adı burada "DataMiningDataSetContainsMeanInsteadOfNan.csv" olarak belirtilmiş.
   * **Önemli:** Dosya, kodu çalıştırdığınız ortamda mevcut olmalı ve doğru dosya yoluyla erişilebilir olmalıdır. Eğer dosya farklı bir klasördeyse, yolun tam olarak belirtilmesi gerekir (örneğin, 'C:/Users/Harun/Desktop/DataMiningDataSetContainsMeanInsteadOfNan.csv' gibi).
3. **Sonuç (DataFrame)**:
   * Kod çalıştırıldığında, CSV dosyasındaki tüm veriler pandas DataFrame formatında bir veri yapısına dönüştürülür. Bu veri yapısı, veriler üzerinde işlem yapmamızı, filtrelememizi, düzenlememizi ve analiz etmemizi sağlar.
   * Bu DataFrame, sütun adları ve satırlardan oluşur, her bir satır verinin bir örneğini (örneğin bir ürün, bir kullanıcı, bir ölçüm) ve her sütun ise o veriye ait bir özelliği temsil eder.



**1. X = data[['Cinsiyet', 'Yas', 'TahminiMaas']]**

Bu satır, veri setindeki belirli sütunları seçip, bu sütunları bir değişkende saklamak için kullanılır

* **data**: Daha önce okuduğumuz CSV dosyasındaki tüm veriyi temsil eden pandas DataFrame’idir.
* **[['Cinsiyet', 'Yas', 'TahminiMaas']]**: Burada, veri setindeki Cinsiyet, Yas ve TahminiMaas adlı üç sütunu seçiyoruz. Bu sütunlar, bağımsız değişkenler (features) olarak kullanılır. Yani, bu sütunlar modelin öğrenmesi gereken bilgiler olacaktır.
  + **Cinsiyet**: Bu sütun, kişilerin cinsiyetini (muhtemelen "Kadın = 1” veya "Erkek =1" gibi) içerir.
  + **Yas**: Bu sütun, kişilerin yaşlarını içeren sayısal veridir.
  + **TahminiMaas**: Bu sütun, kişilerin tahmini maaşlarını gösteren bir sayısal değerdir.
* **X**: X genellikle, makine öğreniminde modelin öğrenmesi için kullanılan bağımsız değişkenleri (özellikleri) temsil eder. Bu nedenle, X, bu üç sütunun yer aldığı yeni bir veri setidir.

Bu satırın amacı, data veri setinden sadece bu üç sütunu seçmek ve onları X isimli yeni bir değişkende saklamaktır. Bu sayede, modelin üzerinde çalışacağı veriler sadece Cinsiyet, Yas, ve TahminiMaas olacaktır.

**2. y = data['IngilizceBilme']**

Bu satır, hedef (bağımlı) değişkeni (label) seçmek için kullanılır. :

**data['IngilizceBilme']**: Burada, veri setindeki IngilizceBilme sütunu seçilmiştir. Bu sütun, kişilerin İngilizce bilip bilmediklerini belirten bir etiket olabilir (örneğin, "İleri =0" , “orta Orta=1", “Başlangıç=2” gibi). Bu sütun, modelin tahmin etmeye çalışacağı çıktıyı temsil eder.

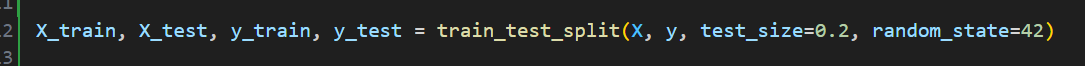
* **y**: y, makine öğreniminde genellikle modelin tahmin etmeye çalıştığı çıktıyı (bağımlı değişkeni) temsil eder. Yani, modelin öğrenmeye çalıştığı şey IngilizceBilme sütunundaki değerlere dayanarak, kişinin İngilizce bilip bilmediğini tahmin etmektir.

Bu satırın amacı, data veri setinden yalnızca IngilizceBilme sütununu seçmek ve onu y isimli bir değişkende saklamaktır. Bu sayede, modelin doğruluğunu ölçerken ve tahmin yaparken, y doğru sonuçları (etiketleri) içerir.

**Özetle:**

* **X**: Modelin eğitileceği bağımsız değişkenlerdir (bu örnekte, Cinsiyet, Yas, TahminiMaas).
* **y**: Modelin tahmin etmeye çalışacağı hedef değişkendir (bu örnekte, IngilizceBilme).

Bu iki değişken (X ve y), modelin eğitilmesi ve test edilmesi için kullanılan temel verilerdir. X, modelin öğrendiği giriş verilerini; y ise bu verilere karşılık gelen doğru çıktıları içerir.



**1. train\_test\_split(X, y, test\_size=0.2, random\_state=42)**

Bu satır, veri setini eğitim ve test verisi olarak ayırmamıza yardımcı olan bir fonksiyonu çağırıyor.

* **train\_test\_split()**: Bu fonksiyon, veriyi ikiye böler: Eğitim verisi ve test verisi. Eğitim verisi, modelin öğrenmesi için kullanılırken; test verisi, modelin doğruluğunu ölçmek için kullanılır. Bu fonksiyon, verileri rastgele ayırarak eğitim ve test setlerini oluşturur.
* **X**: Daha önce açıklamıştık, bu bağımsız değişkenleri (özellikleri) içeren veri setidir. Modelin öğrenmesi gereken bilgiler buradadır.
* **y**: Bu da hedef (bağımlı) değişkeni temsil eder. Modelin tahmin etmeye çalıştığı sonuçlar y veri setinde bulunur.
* **test\_size=0.2**: Bu parametre, verinin ne kadarının test setine ayrılacağını belirtir. Burada 0.2 ifadesi, verinin %20'inin test verisi olarak kullanılacağı anlamına gelir. Kalan %80 ise eğitim için ayrılacaktır. Bu oran genellikle 0.2 (yani %20) ya da 0.3 (yani %30) olarak belirlenir, ancak bu ihtiyaca göre değişebilir.
* **random\_state=42**: Bu parametre, veriyi rastgele ayırırken her seferinde aynı sonucu elde etmemizi sağlar. Yani, her seferinde aynı şekilde eğitim ve test verilerini ayırmak istiyorsak, random\_state parametresi kullanılabilir. 42 burada sadece bir sabit sayı, başka bir sayı da kullanılabilir. Ama bu sayıyı sabit tutarak, her çalıştırmada aynı verilerle eğitim ve test işlemi yapabiliriz.

**2. Sonuçlar (X\_train, X\_test, y\_train, y\_test)**

Bu fonksiyon çalıştırıldığında, aşağıdaki dört değişken oluşturulur:

* **X\_train**: Eğitim verisinin bağımsız değişkenlerini (özellikleri) içeren veri setidir. Bu veriler, modelin eğitilmesinde kullanılacaktır.
* **X\_test**: Test verisinin bağımsız değişkenlerini içeren veri setidir. Bu veriler, modelin doğruluğunu test etmek için kullanılacaktır.
* **y\_train**: Eğitim verisinin hedef değişkenlerini (etiketleri) içeren veri setidir. Model, bu verileri kullanarak öğrenir.
* **y\_test**: Test verisinin hedef değişkenlerini içeren veri setidir. Modelin tahminlerini bu etiketlerle karşılaştırarak doğruluğunu ölçeriz.

**Özetle:**

Bu satır, veri setini %80 eğitim verisi ve %20 test verisi olarak ikiye ayırır. Eğitim verisiyle model eğitilirken, test verisiyle modelin doğruluğu test edilir. random\_state parametresi sayesinde her seferinde aynı şekilde veri ayrımı yapılır. Bu, modelin tutarlı bir şekilde değerlendirilmesini sağlar.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**1. scaler = StandardScaler()**

Bu satır, veriyi ölçeklendirmek için kullanılacak bir StandardScaler nesnesi oluşturur.

* **StandardScaler()**: StandardScaler, sklearn kütüphanesinin bir sınıfıdır ve veriyi **standartlaştırmak** için kullanılır. Standartlaştırma, verilerin ortalamasını 0 ve standart sapmasını 1 yapmak anlamına gelir. Bu işlem, özellikle mesafeye dayalı algoritmalar (örneğin KNN ve SVM ) için önemlidir, çünkü bu algoritmalar veriler arasındaki mesafeleri hesaplar ve farklı ölçeklerdeki veriler modelin doğru çalışmasını zorlaştırabilir.
* **scaler**: Burada scaler, StandardScaler() sınıfından türetilmiş bir nesnedir. Bu nesne, veriyi standartlaştırmak için kullanılacak metodları içerir.

**2. X\_train = scaler.fit\_transform(X\_train)**

Bu satır, eğitim verisini (X\_train) standartlaştırma işlemini gerçekleştirir.

* **scaler.fit\_transform()**: Buradaki fit\_transform metodu, iki işlemi birleştirir:
  + **fit()**: Eğitim verisinin (X\_train) ortalama ve standart sapmasını hesaplar. Bu bilgiler, verilerin nasıl dönüştürüleceğini belirler.
  + **transform()**: Bu aşamada, eğitim verisi (X\_train) standartlaştırılır. Yani, her bir özellik (sütun) için, ortalama 0 ve standart sapma 1 olacak şekilde dönüştürülür.

Kısacası, eğitim verisinin her özelliği, fit() ile elde edilen istatistiklere göre dönüştürülür.

* **Sonuç**: Bu işlemden sonra, X\_train veri seti, her özelliğin ortalaması 0 ve standart sapması 1 olacak şekilde yeniden şekillendirilir.

**3. X\_test = scaler.transform(X\_test)**

Bu satır, test verisini (X\_test) aynı şekilde dönüştürmek için kullanılır. Buradaki işlem şu şekilde gerçekleşir:

* **scaler.transform()**: Test verisini dönüştürmek için kullanılan bir metottur. Buradaki fark, test verisinin dönüşümünde eğitim verisindeki (X\_train) hesaplanan ortalama ve standart sapmanın kullanılmasıdır. Yani, test verisini dönüştürürken, eğitim verisinin özelliklerine göre standartlaştırma yapılır. Bu, test verisinin eğitim verisinden bağımsız olmasını sağlar.
* **Sonuç**: Test verisi, eğitim verisinin istatistiklerine göre aynı şekilde dönüştürülür, yani her özelliğin ortalaması 0 ve standart sapması 1 olacak şekilde yeniden ölçeklendirilir.

**Özetle:**

Bu kod parçası, veri setindeki sayısal özellikleri standartlaştırmak için kullanılır. İlk olarak, eğitim verisi (X\_train) üzerinde standartlaştırma yapılır ve test verisi (X\_test) aynı şekilde dönüştürülür. Bu sayede, her iki veri seti de aynı ölçeğe getirilir. Bu işlem, KNN ve SVM gibi mesafeye dayalı algoritmaların doğru çalışabilmesi için önemlidir, çünkü bu algoritmalar veriler arasındaki mesafeyi hesaplar ve ölçek farklılıkları modelin performansını olumsuz etkileyebilir.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**1. model = KNeighborsClassifier(n\_neighbors=7)**

Bu satır, KNN (K-Nearest Neighbors) sınıflandırma modelini oluşturur.

* **KNeighborsClassifier()**: Bu, scikit-learn kütüphanesinde bulunan KNN algoritmasının sınıfıdır. KNN, sınıflandırma problemlerinde kullanılan bir algoritmadır ve temel olarak, bir veri noktasının sınıfını, en yakın komşularının sınıfına göre tahmin eder. Bu algoritmanın en önemli parametresi, "k" değeri (yakın komşu sayısıdır).
* **n\_neighbors=7**: Bu parametre, KNN algoritmasında kaç komşunun göz önünde bulundurulacağını belirler. Burada n\_neighbors=7, modelin her tahmin için en yakın 7 komşusunun sınıfına bakarak hangi sınıfa ait olduğuna karar vereceği anlamına gelir.
  + Yani, model, test verisi üzerinde her tahmin yaparken, en yakın 7 veriyi (komşuyu) bulur ve bu komşuların çoğunluğuna göre tahmin yapar. Örneğin, 7 komşudan 4’ü "Evet" ve 3’ü "Hayır" diyorsa, model "Evet" sonucunu verir.
* **model**: Burada model, oluşturduğumuz KNN sınıflandırıcısının nesnesidir. Bu nesne, KNN algoritmasının tüm işlevlerini (eğitim, tahmin, vb.) içerir.

**2. model.fit(X\_train, y\_train)**

Bu satır, modelin eğitim verisi ile eğitilmesini sağlar.

* **model.fit()**: fit() metodu, KNN modelini eğitim verisi (X\_train) ve etiketleri (y\_train) ile eğitir. Eğitim sürecinde, model veri üzerinde öğrenme yapmaz (KNN doğrudan bir öğrenme algoritması değildir), ancak eğitim verisini kullanarak, her veri noktasının hangi sınıfa ait olduğunu öğrenir ve bu sınıfları gelecekteki tahminlerde kullanmak üzere saklar.
  + **X\_train**: Eğitim verisi, bağımsız değişkenleri (özellikleri) içerir. Bu veriler, modelin öğrenmesi gereken bilgileri taşır.
  + **y\_train**: Eğitim verisinin etiketleri (sınıflar), modelin tahmin etmeye çalışacağı hedef değişkeni içerir.

**Özetle:**

* **KNeighborsClassifier(n\_neighbors=7)**: Bu satır, KNN algoritmasını oluşturur ve en yakın 7 komşuya dayalı tahminler yapılacağını belirtir.
* **model.fit(X\_train, y\_train)**: Bu satır, modelin eğitim verileriyle eğitilmesini sağlar. KNN algoritması, her veri noktasının hangi sınıfa ait olduğunu öğrenmez, ancak eğitim verisini "hatırlayarak" gelecekteki tahminleri yapmak için kullanır.

KNN, tahmin yaparken eğitim verisi üzerinde herhangi bir modelleme yapmaz; sadece veriyi "hatırlayıp" en yakın komşularına bakarak sınıf tahminini yapar. Bu nedenle, bu modelin eğitilmesi, veri üzerinde eğitim yapmaktan çok, veriyi uygun şekilde "depolamak" anlamına gelir.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**1. y\_pred = model.predict(X\_test)**

Bu satır, eğitilen modelin test verisi üzerinde tahmin yapmasını sağlar.

**model.predict()**: predict() metodu, modelin öğrendiği bilgileri kullanarak yeni verilere (bu durumda test verisine) tahminlerde bulunmasını sağlar. Bu metod, modelin daha önce eğitim sırasında öğrendiği desenlere dayalı olarak test verisindeki her bir örnek için bir sınıf tahmini yapar.

* **X\_test**: Bu, test verisini temsil eder. Eğitim sırasında model sadece eğitim verisini kullanarak öğrenir, ancak test verisini kullanarak modelin ne kadar iyi tahminler yapabildiğini ölçeriz. Burada test verisi (X\_test), modelin daha önce hiç görmediği verilerden oluşur.
* **y\_pred**: Bu, modelin test verisi için yaptığı tahminleri tutar. Yani, model.predict(X\_test) çalıştırıldığında, model her bir test örneği için hangi sınıfa ait olduğunu tahmin eder ve bu tahminler y\_pred değişkenine kaydedilir.

**Özetle:**

* **model.predict(X\_test)**: Bu satır, modelin test verisi üzerinde tahmin yapmasını sağlar. Eğitim sırasında öğrendiği bilgileri kullanarak, test verisindeki her bir örneğin sınıfını tahmin eder.
* **y\_pred**: Modelin test verisi üzerindeki tahminlerinin sonucudur. Bu, modelin doğru sınıfı tahmin edip etmediğini değerlendirebilmek için karşılaştırılacak olan tahmin değerleridir.

Bu işlem, modelin doğruluğunu ölçmek için önemlidir, çünkü test verisi üzerinde yapılan tahminlerle gerçek etiketler karşılaştırılır ve modelin ne kadar doğru tahmin yaptığına bakılır.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**1. conf\_matrix = confusion\_matrix(y\_test, y\_pred)**

Bu satır, **confusion matrix (karışıklık matrisi)** oluşturur.

* **confusion\_matrix()**: Bu fonksiyon, modelin yaptığı tahminlerin doğruluğunu değerlendirmek için kullanılan bir metriktir. Gerçek etiketlerle (test verisindeki doğru etiketler) modelin tahmin ettiği etiketleri karşılaştırır ve bunları bir matris şeklinde sunar. Bu matris, modelin sınıflandırma performansını anlamamıza yardımcı olur.
* **y\_test**: Bu, gerçek etiketleri (test verisinin doğru sonuçlarını) içerir. Modelin tahminlerini bu gerçek etiketlerle karşılaştıracağız.
* **y\_pred**: Bu, modelin test verisi için yaptığı tahminleri içerir. confusion\_matrix() fonksiyonu, bu tahminleri, gerçek etiketlerle karşılaştırır.
* **conf\_matrix**: Bu değişken, karışıklık matrisini tutar. Karışıklık matrisi, her sınıf için doğru ve yanlış tahminlerin sayısını gösteren 2x2 veya daha büyük bir matris olabilir. Bu matrisin her bir elemanı, modelin yaptığı sınıflandırmaların nasıl olduğuna dair bilgi verir.Bizim için 3x3 bir matrix vericektir çünkü tahmin etmeye çalıştığı sutunda 3 farklı değer vardır.

Örneğin, ikili sınıflandırma (binary classification) için şöyle bir karışıklık matrisi olabilir:

|  | **Tahmin: No** | **Tahmin: Yes** |
| --- | --- | --- |
| Gerçek: No | True Negative (TN) | False Positive (FP) |
| Gerçek: Yes | False Negative (FN) | True Positive (TP) |

Bu matrisin elemanları şunları ifade eder:

* + **True Positive (TP)**: Model doğru bir şekilde pozitif sınıfı tahmin etti.
  + **True Negative (TN)**: Model doğru bir şekilde negatif sınıfı tahmin etti.
  + **False Positive (FP)**: Model yanlış bir şekilde pozitif sınıfı tahmin etti (Type I Error).
  + **False Negative (FN)**: Model yanlış bir şekilde negatif sınıfı tahmin etti (Type II Error).

**2. accuracy = accuracy\_score(y\_test, y\_pred)**

Bu satır, **accuracy (doğruluk)** skorunu hesaplar.

**accuracy\_score()**: Bu fonksiyon, modelin doğruluğunu ölçmek için kullanılır. Modelin doğru tahmin ettiği örneklerin oranını hesaplar. Bu, genel olarak modelin ne kadar doğru çalıştığını ölçen en basit ve en yaygın metriktir.

* **y\_test**: Gerçek etiketler (doğru sonuçlar) içerir.
* **y\_pred**: Modelin tahmin ettiği etiketler içerir.
* **accuracy**: Bu değişken, modelin doğruluğunu tutar. Hesaplama şu şekilde yapılır:

Accuracy=Doğru Tahminler Toplam/ Örnek Sayısı

Yani, modelin doğru tahmin ettiği etiketlerin sayısı, toplam test örneği sayısına bölünür.

**Özetle:**

* **confusion\_matrix(y\_test, y\_pred)**: Bu, modelin test verisi üzerinde yaptığı tahminlerle gerçek etiketleri karşılaştıran bir matris oluşturur. Bu matris, modelin doğru ve yanlış tahminlerini gösterir ve her sınıfın ne kadar doğru tahmin edildiğini anlamamıza yardımcı olur.
* **accuracy\_score(y\_test, y\_pred)**: Bu, modelin doğruluk oranını hesaplar. Yani, modelin doğru sınıflandırma yaptığı örneklerin toplam örnek sayısına oranını verir. Bu, modelin genel performansını ölçmek için kullanılan temel bir metriktir.

metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**print("Confusion Matrix:\n", conf\_matrix)**

Bu satır, **karışıklık matrisini** ekrana yazdırır.

* **print()**: Python'da ekrana bir şey yazdırmak için kullanılan bir fonksiyondur. Bu fonksiyon, belirttiğimiz metni ya da değişkenin değerini ekrana çıkartır.
* **"Confusion Matrix:\n"**: Bu, yazdırılacak olan metni içerir. "Confusion Matrix:" kısmı ekrana yazdırılacak olan metni belirler ve \n ifadesi yeni bir satır başlatır. Bu, karışıklık matrisinin alt satırda görünmesini sağlar. Yani, önce "Confusion Matrix:" yazısı, sonra bir satır boşluk bırakarak matrisin kendisi yazdırılacak.
* **conf\_matrix**: Bu, önceki satırlarda hesaplanan karışıklık matrisidir. conf\_matrix değişkeni, modelin test verisi üzerinde yaptığı tahminlerin doğruluğunu gösteren 3x3 bir matrix göstericektir. Bu matrisin içeriği, modelin doğru ve yanlış tahminleri hakkında bilgi verir (True Positive, False Positive, True Negative, False Negative gibi).

Bu satır çalıştırıldığında, **karışıklık matrisi** ekrana yazdırılır, böylece modelin performansını daha detaylı bir şekilde gözlemleyebiliriz.

**2. print("Accuracy:", accuracy)**

Bu satır, **doğruluk (accuracy)** skorunu ekrana yazdırır. Şimdi bunu daha ayrıntılı açıklayalım:

* **print()**: Bu fonksiyon, Python'da bir şeyleri ekrana yazdırmak için kullanılır. Bu satırda, doğruluk (accuracy) bilgisini ekrana yazdırıyoruz.
* **"Accuracy:"**: Bu, yazdırılacak metnin bir kısmıdır. Burada "Accuracy:" ifadesi, ekranda doğruluk oranının yazdığı kısmı belirtir.
* **accuracy**: Bu, modelin doğruluğunu temsil eder ve daha önce hesapladık. Modelin doğru sınıflandırdığı örneklerin oranını verir. accuracy değişkeni, doğru tahminlerin toplam tahminlere oranını içerir.

Bu satır çalıştırıldığında, modelin doğruluk oranı ekrana yazdırılır. Bu sayede, modelin test verisi üzerindeki genel başarısını sayısal olarak görmüş oluruz.

**Özetle:**

* **print("Confusion Matrix:\n", conf\_matrix)**: Bu satır, karışıklık matrisini ekrana yazdırır. Karışıklık matrisi, modelin test verisi üzerinde yaptığı tahminlerin doğruluğunu ve yanlışlığını gösteren bir tablodur.
* **print("Accuracy:", accuracy)**: Bu satır, modelin doğruluk oranını ekrana yazdırır. Doğruluk, modelin doğru tahmin ettiği örneklerin oranını gösterir ve modelin genel başarısını ölçmek için kullanılır.

Bu iki satır, modelin test verisi üzerindeki performansını daha anlaşılır ve görsel olarak sunar.

SVM KODU AÇIKLAMASI

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**1. pandas**

Pandas, Python'da veriyle çalışırken en çok kullanılan kütüphanelerden biridir. Bu kütüphane, veriyi tablolar halinde düzenler. Mesela, bir Excel dosyasındaki veriler gibi. Pandas, bu veriyi çok rahat bir şekilde okuyup, üzerinde işlem yapmamızı sağlar. Örneğin, bir dosyadan veriyi almak, eksik verileri bulmak, verileri sıralamak ya da bazı verileri çıkarıp yeni bir dosya oluşturmak için kullanılır. Pandas sayesinde veriyi çok kolay bir şekilde analiz edebiliriz.

**2. train\_test\_split**

Bu fonksiyon, verimizi ikiye ayırmamıza yardımcı olur: Bir kısmını modelimizi eğitmek için, diğer kısmını ise modelimizin doğruluğunu test etmek için kullanırız. Yani, modelin öğrenmesi için bazı veriler kullanılır, ancak modelin doğru çalışıp çalışmadığını görmek için bu verilerle hiç karşılaşmamış yeni verilerle test yapılır. Bu sayede modelin genelleme yeteneğini test etmiş oluruz. train\_test\_split, veriyi rasgele ve belirli bir oranda ayırmamıza olanak sağlar (örneğin, %80 eğitim, %20 test gibi). Bu, modelimizin doğru öğrenip öğrenmediğini anlamamıza yardımcı olur.

**3. StandardScaler**

Bazen verilerimiz çok farklı ölçekte olur. Mesela, bir sütunlarda uzunluklar (metre cinsinden) ve diğerlerinde yaş gibi küçük sayılar olabilir. Bu durumda, büyük sayılar modelin üzerinde daha fazla etkili olabilir, bu da modelin dengesiz çalışmasına yol açar. İşte burada StandardScaler devreye girer. Bu fonksiyon, verilerin her birini benzer bir ölçekte yapar, yani verilerin ortalamasını sıfır ve standart sapmasını bir yapar. Bu sayede, tüm veriler eşit şekilde modellenir ve model daha doğru çalışır.

**4. from sklearn.svm import SVC**

* from sklearn.svm: Bu kısım, sklearn kütüphanesinin svm (Support Vector Machine) modülünden bir fonksiyon çağırır.
* import SVC: SVC, Support Vector Classification (Destek Vektör Sınıflandırıcı) algoritmasını temsil eder. Bu algoritma, veriyi sınıflandırmak için kullanılan güçlü bir yöntemdir. Özellikle doğrusal olmayan sınıflandırmalarda da oldukça başarılıdır. SVC, sınıflandırma modelini oluşturmak için kullanılan sınıfı temsil eder.

Bu satır, sınıflandırma için kullanılacak olan SVC algoritmasını programımıza dahil eder.

**5. confusion\_matrix ve accuracy\_score**

Bu iki metrik, modelimizin ne kadar doğru çalıştığını ölçmek için kullanılır. Confusion Matrix (karmaşıklık matrisi), modelimizin doğru ve yanlış tahminlerini bir tabloya döker. Bu tablonun bize söylediği şudur: Modelimiz hangi sınıflarda doğru tahmin yapmış, hangi sınıflarda yanlış yapmış? Örneğin, bir model bir resmi "kedili" olarak doğru sınıflandırmışsa, bu doğru bir tahmindir ve bunu gösteririz. Ama eğer bu resmi "köpek" olarak yanlış sınıflandırmışsa, bu yanlış bir tahmindir ve bunu da gösteririz.

Accuracy Score ise basitçe, modelin doğru yaptığı tahminlerin oranıdır. Yani, doğru tahminlerin toplam tahminlere oranını verir. Mesela modelimiz 100 tahminden 85’ini doğru yaptıysa, accuracy %85 olur. Ancak, eğer veri dengesizse, yani bir sınıf çok daha fazla sayıda varsa, yüksek accuracy her zaman iyi bir sonuç olmayabilir. Bu yüzden confusion matrix, modelin başarısını daha ayrıntılı şekilde gösterir.

**Özetle:**

Bu satırlarla, projemize aşağıdaki kütüphaneleri dahil etmiş olduk:

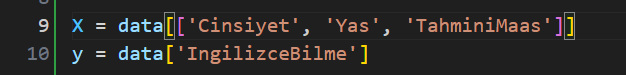
* **pandas**: Veri okuma ve işleme işlemleri için.
* **train\_test\_split**: Veriyi eğitim ve test setlerine ayırmak için.
* **StandardScaler**: Veriyi standartlaştırmak için.
* **SVC**: Sınıflandırma algoritması olan destek vektör makinelerini (SVM) kullanmak için.
* **confusion\_matrix, accuracy\_score**: Modelin doğruluğunu ve performansını değerlendirmek için.

Bu kütüphaneler, makine öğrenimi projelerinde sıkça kullanılır ve her biri belirli bir işlevi yerine getirir.



Bu kod satırı, Python'da pandas kütüphanesini kullanarak bir CSV dosyasını okuma işlemidir :

1. **pd.read\_csv()**:
   * **read\_csv()**: Bu, pandas kütüphanesinin, bir CSV (Comma Separated Values - Virgülle Ayrılmış Değerler) dosyasını okumasını sağlayan bir fonksiyondur. CSV dosyaları genellikle tabular (tablolar halinde) verileri depolamak için kullanılır, yani veriler satır ve sütunlar halinde düzenlenir. Bu fonksiyon sayesinde, bir CSV dosyasındaki verileri Python'da kolayca kullanabileceğimiz bir formata (pandas DataFrame) dönüştürmüş oluruz.
   * **pd**: Burada pd, pandas kütüphanesinin kısaltmasıdır. Pandas’ı genellikle import pandas as pd şeklinde import (içe aktarma) ederiz ve böylece pd ile pandas fonksiyonlarına erişebiliriz.
2. **Dosya Yolu ('DataMiningDataSetContainsMeanInsteadOfNan.csv')**:
   * 'DataMiningDataSetContainsMeanInsteadOfNan.csv' kısmı, okuma işlemi yapılacak CSV dosyasının adını ve yolunu belirtir. Bu dosya, verileri içeren bir dosya olmalı ve formatı CSV (virgülle ayrılmış değerler) olmalıdır. Dosyanın adı burada "DataMiningDataSetContainsMeanInsteadOfNan.csv" olarak belirtilmiş.
3. **Önemli:** Dosya, kodu çalıştırdığınız ortamda mevcut olmalı ve doğru dosya yoluyla erişilebilir olmalıdır. Eğer dosya farklı bir klasördeyse, yolun tam olarak belirtilmesi gerekir (örneğin, 'C:/Users/Harun/Desktop/DataMiningDataSetContainsMeanInsteadOfNan.csv' gibi). **Sonuç (DataFrame)**:
   * Kod çalıştırıldığında, CSV dosyasındaki tüm veriler pandas DataFrame formatında bir veri yapısına dönüştürülür. Bu veri yapısı, veriler üzerinde işlem yapmamızı, filtrelememizi, düzenlememizi ve analiz etmemizi sağlar.
   * Bu DataFrame, sütun adları ve satırlardan oluşur, her bir satır verinin bir örneğini (örneğin bir ürün, bir kullanıcı, bir ölçüm) ve her sütun ise o veriye ait bir özelliği temsil eder.



**1. X = data[['Cinsiyet', 'Yas', 'TahminiMaas']]**

Bu satır, veri setindeki belirli sütunları seçip, bu sütunları bir değişkende saklamak için kullanılır

* **data**: Daha önce okuduğumuz CSV dosyasındaki tüm veriyi temsil eden pandas DataFrame’idir.
* **[['Cinsiyet', 'Yas', 'TahminiMaas']]**: Burada, veri setindeki Cinsiyet, Yas ve TahminiMaas adlı üç sütunu seçiyoruz. Bu sütunlar, bağımsız değişkenler (features) olarak kullanılır. Yani, bu sütunlar modelin öğrenmesi gereken bilgiler olacaktır.
  + **Cinsiyet**: Bu sütun, kişilerin cinsiyetini (muhtemelen "Kadın = 1” veya "Erkek =1" gibi) içerir.
  + **Yas**: Bu sütun, kişilerin yaşlarını içeren sayısal veridir.
  + **TahminiMaas**: Bu sütun, kişilerin tahmini maaşlarını gösteren bir sayısal değerdir.
* **X**: X genellikle, makine öğreniminde modelin öğrenmesi için kullanılan bağımsız değişkenleri (özellikleri) temsil eder. Bu nedenle, X, bu üç sütunun yer aldığı yeni bir veri setidir.

Bu satırın amacı, data veri setinden sadece bu üç sütunu seçmek ve onları X isimli yeni bir değişkende saklamaktır. Bu sayede, modelin üzerinde çalışacağı veriler sadece Cinsiyet, Yas, ve TahminiMaas olacaktır.

**2. y = data['IngilizceBilme']**

Bu satır, hedef (bağımlı) değişkeni (label) seçmek için kullanılır. :

**data['IngilizceBilme']**: Burada, veri setindeki IngilizceBilme sütunu seçilmiştir. Bu sütun, kişilerin İngilizce bilip bilmediklerini belirten bir etiket olabilir (örneğin, "İleri =0" , “orta Orta=1", “Başlangıç=2” gibi). Bu sütun, modelin tahmin etmeye çalışacağı çıktıyı temsil eder.

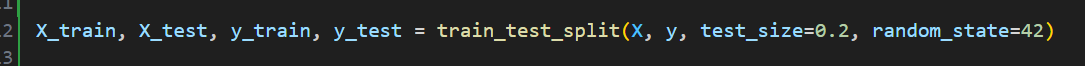
* **y**: y, makine öğreniminde genellikle modelin tahmin etmeye çalıştığı çıktıyı (bağımlı değişkeni) temsil eder. Yani, modelin öğrenmeye çalıştığı şey IngilizceBilme sütunundaki değerlere dayanarak, kişinin İngilizce bilip bilmediğini tahmin etmektir.

Bu satırın amacı, data veri setinden yalnızca IngilizceBilme sütununu seçmek ve onu y isimli bir değişkende saklamaktır. Bu sayede, modelin doğruluğunu ölçerken ve tahmin yaparken, y doğru sonuçları (etiketleri) içerir.

**Özetle:**

* **X**: Modelin eğitileceği bağımsız değişkenlerdir (bu örnekte, Cinsiyet, Yas, TahminiMaas).
* **y**: Modelin tahmin etmeye çalışacağı hedef değişkendir (bu örnekte, IngilizceBilme).

Bu iki değişken (X ve y), modelin eğitilmesi ve test edilmesi için kullanılan temel verilerdir. X, modelin öğrendiği giriş verilerini; y ise bu verilere karşılık gelen doğru çıktıları içerir.



**1. train\_test\_split(X, y, test\_size=0.2, random\_state=42)**

Bu satır, veri setini eğitim ve test verisi olarak ayırmamıza yardımcı olan bir fonksiyonu çağırıyor.

* **train\_test\_split()**: Bu fonksiyon, veriyi ikiye böler: Eğitim verisi ve test verisi. Eğitim verisi, modelin öğrenmesi için kullanılırken; test verisi, modelin doğruluğunu ölçmek için kullanılır. Bu fonksiyon, verileri rastgele ayırarak eğitim ve test setlerini oluşturur.
* **X**: Daha önce açıklamıştık, bu bağımsız değişkenleri (özellikleri) içeren veri setidir. Modelin öğrenmesi gereken bilgiler buradadır.
* **y**: Bu da hedef (bağımlı) değişkeni temsil eder. Modelin tahmin etmeye çalıştığı sonuçlar y veri setinde bulunur.
* **test\_size=0.2**: Bu parametre, verinin ne kadarının test setine ayrılacağını belirtir. Burada 0.2 ifadesi, verinin %20'inin test verisi olarak kullanılacağı anlamına gelir. Kalan %80 ise eğitim için ayrılacaktır. Bu oran genellikle 0.2 (yani %20) ya da 0.3 (yani %30) olarak belirlenir, ancak bu ihtiyaca göre değişebilir.
* **random\_state=42**: Bu parametre, veriyi rastgele ayırırken her seferinde aynı sonucu elde etmemizi sağlar. Yani, her seferinde aynı şekilde eğitim ve test verilerini ayırmak istiyorsak, random\_state parametresi kullanılabilir. 42 burada sadece bir sabit sayı, başka bir sayı da kullanılabilir. Ama bu sayıyı sabit tutarak, her çalıştırmada aynı verilerle eğitim ve test işlemi yapabiliriz.

**2. Sonuçlar (X\_train, X\_test, y\_train, y\_test)**

Bu fonksiyon çalıştırıldığında, aşağıdaki dört değişken oluşturulur:

* **X\_train**: Eğitim verisinin bağımsız değişkenlerini (özellikleri) içeren veri setidir. Bu veriler, modelin eğitilmesinde kullanılacaktır.
* **X\_test**: Test verisinin bağımsız değişkenlerini içeren veri setidir. Bu veriler, modelin doğruluğunu test etmek için kullanılacaktır.
* **y\_train**: Eğitim verisinin hedef değişkenlerini (etiketleri) içeren veri setidir. Model, bu verileri kullanarak öğrenir.
* **y\_test**: Test verisinin hedef değişkenlerini içeren veri setidir. Modelin tahminlerini bu etiketlerle karşılaştırarak doğruluğunu ölçeriz.

**Özetle:**

Bu satır, veri setini %80 eğitim verisi ve %20 test verisi olarak ikiye ayırır. Eğitim verisiyle model eğitilirken, test verisiyle modelin doğruluğu test edilir. random\_state parametresi sayesinde her seferinde aynı şekilde veri ayrımı yapılır. Bu, modelin tutarlı bir şekilde değerlendirilmesini sağlar.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**1. scaler = StandardScaler()**

Bu satır, veriyi ölçeklendirmek için kullanılacak bir StandardScaler nesnesi oluşturur.

* **StandardScaler()**: StandardScaler, sklearn kütüphanesinin bir sınıfıdır ve veriyi **standartlaştırmak** için kullanılır. Standartlaştırma, verilerin ortalamasını 0 ve standart sapmasını 1 yapmak anlamına gelir. Bu işlem, özellikle mesafeye dayalı algoritmalar (örneğin KNN ve SVM ) için önemlidir, çünkü bu algoritmalar veriler arasındaki mesafeleri hesaplar ve farklı ölçeklerdeki veriler modelin doğru çalışmasını zorlaştırabilir.
* **scaler**: Burada scaler, StandardScaler() sınıfından türetilmiş bir nesnedir. Bu nesne, veriyi standartlaştırmak için kullanılacak metodları içerir.

**2. X\_train = scaler.fit\_transform(X\_train)**

Bu satır, eğitim verisini (X\_train) standartlaştırma işlemini gerçekleştirir.

* **scaler.fit\_transform()**: Buradaki fit\_transform metodu, iki işlemi birleştirir:
  + **fit()**: Eğitim verisinin (X\_train) ortalama ve standart sapmasını hesaplar. Bu bilgiler, verilerin nasıl dönüştürüleceğini belirler.
  + **transform()**: Bu aşamada, eğitim verisi (X\_train) standartlaştırılır. Yani, her bir özellik (sütun) için, ortalama 0 ve standart sapma 1 olacak şekilde dönüştürülür.

Kısacası, eğitim verisinin her özelliği, fit() ile elde edilen istatistiklere göre dönüştürülür.

* **Sonuç**: Bu işlemden sonra, X\_train veri seti, her özelliğin ortalaması 0 ve standart sapması 1 olacak şekilde yeniden şekillendirilir.

**3. X\_test = scaler.transform(X\_test)**

Bu satır, test verisini (X\_test) aynı şekilde dönüştürmek için kullanılır. Buradaki işlem şu şekilde gerçekleşir:

* **scaler.transform()**: Test verisini dönüştürmek için kullanılan bir metottur. Buradaki fark, test verisinin dönüşümünde eğitim verisindeki (X\_train) hesaplanan ortalama ve standart sapmanın kullanılmasıdır. Yani, test verisini dönüştürürken, eğitim verisinin özelliklerine göre standartlaştırma yapılır. Bu, test verisinin eğitim verisinden bağımsız olmasını sağlar.
* **Sonuç**: Test verisi, eğitim verisinin istatistiklerine göre aynı şekilde dönüştürülür, yani her özelliğin ortalaması 0 ve standart sapması 1 olacak şekilde yeniden ölçeklendirilir.

**Özetle:**

Bu kod parçası, veri setindeki sayısal özellikleri standartlaştırmak için kullanılır. İlk olarak, eğitim verisi (X\_train) üzerinde standartlaştırma yapılır ve test verisi (X\_test) aynı şekilde dönüştürülür. Bu sayede, her iki veri seti de aynı ölçeğe getirilir. Bu işlem, KNN ve SVM gibi mesafeye dayalı algoritmaların doğru çalışabilmesi için önemlidir, çünkü bu algoritmalar veriler arasındaki mesafeyi hesaplar ve ölçek farklılıkları modelin performansını olumsuz etkileyebilir.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**model = SVC(kernel='linear', random\_state=42)**

* **model =**: Burada model adında bir değişken tanımlıyoruz ve bu değişken, oluşturduğumuz sınıflandırıcı modelini tutacak.
* **SVC**: Bu, daha önce de bahsettiğimiz gibi, **Support Vector Classification (Destek Vektör Sınıflandırıcı)** algoritmasını temsil eder. SVC sınıfı, **destek vektör makineleri (SVM)** algoritmasını kullanarak veriyi sınıflandırmak için bir model oluşturur. SVM, doğrusal ve doğrusal olmayan verilerle çalışabilen güçlü bir sınıflandırma algoritmasıdır.
* **kernel='linear'**: Bu parametre, **çekirdek fonksiyonu** olarak kullanılan türü belirler. SVM, veriyi sınıflandırırken bir "çekirdek fonksiyonu" kullanır ve bu fonksiyon, veriyi daha yüksek bir boyutta, yani daha karmaşık bir alanda sınıflandırmayı mümkün kılar.
  + **'linear'**: Burada, çekirdek fonksiyonu olarak **doğrusal (linear)** bir fonksiyon seçiyoruz. Bu, verinin doğrusal bir şekilde (düz bir çizgiyle) sınıflandırılacağı anlamına gelir. Yani, veriyi iki sınıfa ayırmak için düz bir çizgi (hiper-düzlem) kullanılır. Eğer veriler doğrusal olarak birbirinden ayrılabiliyorsa, bu seçenek etkili olur.
* **random\_state=42**: Bu parametre, modelin rastgelelik içeren işlemlerini kontrol eder. SVM modelini eğitim verisiyle eğitirken, rastgelelik içeren bazı işlemler yapılabilir. Bu parametre, bu rastgelelik işlemlerinin her seferinde aynı sonucu vermesini sağlar. Yani, modeli her çalıştırdığınızda aynı sonuçları almak için random\_state parametresi belirlenir. 42 sayısı, burada sadece örnek bir değer olup, farklı sayılar kullanarak farklı sonuçlar elde edebilirsiniz, ancak 42 genellikle bir sabit olarak tercih edilir.

Sonuç olarak, bu satır, **doğrusal bir çekirdek fonksiyonu** kullanarak, **rastgelelik kontrolü** ile **SVM** sınıflandırıcı modelini oluşturur.

**2. model.fit(X\_train, y\_train)**

* **model.fit()**: Bu, oluşturduğumuz modelin eğitilmesi için kullanılan fonksiyondur. fit() fonksiyonu, modelin eğitim verisi üzerinde öğrenmesini sağlar. Yani, model, eğitim verisini kullanarak verilerin sınıflarını öğrenir ve veriyi doğru şekilde sınıflandırmak için uygun parametreleri bulur.
* **X\_train**: Bu, eğitim setindeki **özellikler (features)** yani verilerimizin giriş kısmıdır. Bu veri, modelin öğrenmesi için kullanılan bilgileri içerir. Örneğin, X\_train'deki her bir satır, bir örneği (örneğin bir kişinin yaşı, cinsiyeti, maaşı vb.) temsil eder.
* **y\_train**: Bu, eğitim setindeki **etiketler (labels)** yani sınıflandırılacak sonuçları temsil eder. Yani, y\_train, modelin doğru cevabını (sınıf etiketini) içerir. Örneğin, y\_train'deki her bir değer, o örneğin sınıfını (örneğin, İngilizce bilip bilmediği) gösterir.

Bu satırda, model.fit(X\_train, y\_train) kodu, **SVM sınıflandırıcı modelini** eğitim verisi üzerinde eğitir. Eğitim süreci sırasında model, verilerin özellikleri ile etiketler arasındaki ilişkiyi öğrenir ve gelecekteki tahminlerde kullanılacak parametreleri oluşturur.

**Özetle:**

* **model = SVC(kernel='linear', random\_state=42)**: Burada, doğrusal bir çekirdek fonksiyonu kullanarak SVM sınıflandırıcısı (SVC) oluşturuyoruz. random\_state=42, modelin her çalıştırıldığında aynı sonuçları vermesini sağlamak için kullanılır.
* **model.fit(X\_train, y\_train)**: Bu satırda, model eğitim verisi (X\_train) ve etiketleri (y\_train) kullanarak eğitilir. Bu eğitim süreci sırasında, model verilerin sınıflarını öğrenir ve doğrusal bir sınıflandırıcı oluşturur.

Bu iki satır, modelin oluşturulması ve eğitilmesi sürecini başlatır. Eğitildikten sonra model, test verisi üzerinde tahminlerde bulunabilir.



**y\_pred = model.predict(X\_test)**

* **model.predict()**: Bu fonksiyon, eğitilmiş olan modelin test verisi üzerinde tahminlerde bulunmasını sağlar. Yani, model daha önce eğitim setinde öğrendiği bilgilere dayanarak, test setindeki verilere hangi sınıf etiketlerini vereceğini tahmin eder.
  + **Eğitim Süreci**: Model, eğitim verisiyle (X\_train) eğitildikten sonra, test verisini kullanarak bu öğrendiği bilgileri doğrular. predict() fonksiyonu, test verisini (X\_test) alır ve bu veriye ait tahmin edilen sınıf etiketlerini döndürür.
* **X\_test**: Bu, modelin test edilmesi için kullanılan veridir. Eğitim sırasında model bu verilerle "henüz karşılaşmamış" olur. Bu yüzden test verisi, modelin ne kadar iyi genelleme yapabildiğini görmek için kullanılır. X\_test'teki her bir örnek, modelin doğru sınıf etiketini tahmin etmeye çalıştığı yeni verilerdir.
* **y\_pred**: Bu, modelin X\_test verisi üzerinde yaptığı tahminlerin sonucudur. Model, her bir test örneği için bir sınıf etiketine karar verir ve bu etiketleri y\_pred değişkeninde saklar. Bu tahminler, gerçek etiketlerle karşılaştırılarak modelin doğruluğu değerlendirilir.

**Özetle:**

Bu satır, **modelin test verisi (X\_test) üzerinde tahmin yapmasını sağlar**. Eğitilmiş olan model, test verisini analiz eder ve her bir test örneği için bir sınıf etiketi tahmin eder. Tahminler, y\_pred değişkeninde saklanır.

Bu tahminlerin ardından, modelin ne kadar doğru sınıflandırma yaptığına bakmak için **karışıklık matrisi** ve **doğruluk skoru** gibi değerlendirme metrikleri kullanılabilir.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**conf\_matrix = confusion\_matrix(y\_test, y\_pred)**

* **confusion\_matrix()**: Bu fonksiyon, modelin tahminleri ile gerçek etiketleri karşılaştırarak bir **karışıklık matrisi (confusion matrix)** oluşturur. Karışıklık matrisi, modelin doğruluğunu değerlendirmek için oldukça yararlı bir araçtır.
  + **Gerçek Etiketler (True Labels)**: y\_test, test setindeki gerçek sınıf etiketlerini temsil eder. Yani, bu, verilerin aslında hangi sınıfa ait olduğunu gösterir.
  + **Tahmin Edilen Etiketler (Predicted Labels)**: y\_pred, modelin tahmin ettiği sınıf etiketlerini temsil eder. Yani, bu, modelin test setindeki veriler için ne tahmin ettiğini gösterir.

Karışıklık matrisi, her iki etiket kümesi arasındaki karşılaştırmayı yaparak aşağıdaki dört temel değeri içerir:

* + **True Positive (TP)**: Modelin doğru şekilde sınıflandırdığı pozitif örnekler (gerçek pozitif ve tahmin edilen pozitif).
  + **False Positive (FP)**: Modelin yanlış şekilde pozitif olarak sınıflandırdığı örnekler (gerçek negatif, fakat model bunu pozitif olarak tahmin etti).
  + **True Negative (TN)**: Modelin doğru şekilde sınıflandırdığı negatif örnekler (gerçek negatif ve tahmin edilen negatif).
  + **False Negative (FN)**: Modelin yanlış şekilde negatif olarak sınıflandırdığı örnekler (gerçek pozitif, fakat model bunu negatif olarak tahmin etti).

Bu dört değer, karışıklık matrisinde şöyle düzenlenir:

|  | **Predicted Positive** | **Predicted Negative** |
| --- | --- | --- |
| **Actual Positive** | TP | FN |
| **Actual Negative** | FP | TN |

Bu matris, modelin hangi sınıfları doğru tahmin ettiğini ve hangi sınıflarda hata yaptığını görselleştirir.

* **conf\_matrix**: Bu değişken, karışıklık matrisini saklar. Bu matris, modelin performansını analiz etmek için kullanılabilir. Matris, doğru ve yanlış tahminlerin sayısını gösterir.

**accuracy = accuracy\_score(y\_test, y\_pred)**

* **accuracy\_score()**: Bu fonksiyon, modelin **doğruluğunu** hesaplamak için kullanılır. Doğruluk, modelin doğru tahmin ettiği örneklerin toplam örnek sayısına oranıdır.
  + **Doğruluk Hesaplama**: Doğruluk, aşağıdaki formülle hesaplanır:

Doğruluk= Doğru tahminler (TP + TN) / Toplam örnek sayısı (TP + TN + FP + FN)

Bu formülde, doğru tahminler (TP + TN) ve toplam örnek sayısı (TP + TN + FP + FN) kullanılarak doğruluk hesaplanır.

* **accuracy**: Bu değişken, modelin doğruluğunu (accuracy) saklar. Yani, modelin test setindeki doğru sınıflandırmalarının oranını gösterir.

**Özetle:**

1. **conf\_matrix = confusion\_matrix(y\_test, y\_pred)**: Bu satır, **karışıklık matrisini** oluşturur. Gerçek etiketler (y\_test) ile modelin tahmin ettiği etiketler (y\_pred) karşılaştırılarak modelin doğruluğu ve hataları hakkında bilgi edinilir.
2. **accuracy = accuracy\_score(y\_test, y\_pred)**: Bu satır, **doğruluğu (accuracy)** hesaplar. Modelin doğru tahmin ettiği sınıf etiketlerinin oranını verir. Yüksek doğruluk, modelin iyi performans gösterdiği anlamına gelir.

Bu iki metrik, modelin performansını anlamak ve değerlendirmek için yaygın olarak kullanılır.

ekran görüntüsü, metin, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**print("Confusion Matrix:\n", conf\_matrix)**

Bu satır, **karışıklık matrisini** ekrana yazdırır.

* **print()**: Python'da ekrana bir şey yazdırmak için kullanılan bir fonksiyondur. Bu fonksiyon, belirttiğimiz metni ya da değişkenin değerini ekrana çıkartır.
* **"Confusion Matrix:\n"**: Bu, yazdırılacak olan metni içerir. "Confusion Matrix:" kısmı ekrana yazdırılacak olan metni belirler ve \n ifadesi yeni bir satır başlatır. Bu, karışıklık matrisinin alt satırda görünmesini sağlar. Yani, önce "Confusion Matrix:" yazısı, sonra bir satır boşluk bırakarak matrisin kendisi yazdırılacak.
* **conf\_matrix**: Bu, önceki satırlarda hesaplanan karışıklık matrisidir. conf\_matrix değişkeni, modelin test verisi üzerinde yaptığı tahminlerin doğruluğunu gösteren 3x3 bir matrix göstericektir. Bu matrisin içeriği, modelin doğru ve yanlış tahminleri hakkında bilgi verir (True Positive, False Positive, True Negative, False Negative gibi).

Bu satır çalıştırıldığında, **karışıklık matrisi** ekrana yazdırılır, böylece modelin performansını daha detaylı bir şekilde gözlemleyebiliriz.

**2. print("Accuracy:", accuracy)**

Bu satır, **doğruluk (accuracy)** skorunu ekrana yazdırır. Şimdi bunu daha ayrıntılı açıklayalım:

* **print()**: Bu fonksiyon, Python'da bir şeyleri ekrana yazdırmak için kullanılır. Bu satırda, doğruluk (accuracy) bilgisini ekrana yazdırıyoruz.
* **"Accuracy:"**: Bu, yazdırılacak metnin bir kısmıdır. Burada "Accuracy:" ifadesi, ekranda doğruluk oranının yazdığı kısmı belirtir.
* **accuracy**: Bu, modelin doğruluğunu temsil eder ve daha önce hesapladık. Modelin doğru sınıflandırdığı örneklerin oranını verir. accuracy değişkeni, doğru tahminlerin toplam tahminlere oranını içerir.

Bu satır çalıştırıldığında, modelin doğruluk oranı ekrana yazdırılır. Bu sayede, modelin test verisi üzerindeki genel başarısını sayısal olarak görmüş oluruz.

**Özetle:**

* **print("Confusion Matrix:\n", conf\_matrix)**: Bu satır, karışıklık matrisini ekrana yazdırır. Karışıklık matrisi, modelin test verisi üzerinde yaptığı tahminlerin doğruluğunu ve yanlışlığını gösteren bir tablodur.
* **print("Accuracy:", accuracy)**: Bu satır, modelin doğruluk oranını ekrana yazdırır. Doğruluk, modelin doğru tahmin ettiği örneklerin oranını gösterir ve modelin genel başarısını ölçmek için kullanılır.

Bu iki satır, modelin test verisi üzerindeki performansını daha anlaşılır ve görsel olarak sunar.

VERİYE NAN DEĞERLERİ EKLEME

KODU AÇIKLAMASI

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**1. pandas**

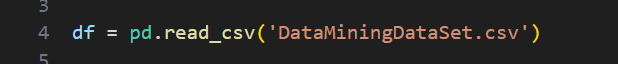
Pandas, Python'da veriyle çalışırken en çok kullanılan kütüphanelerden biridir. Bu kütüphane, veriyi tablolar halinde düzenler. Mesela, bir Excel dosyasındaki veriler gibi. Pandas, bu veriyi çok rahat bir şekilde okuyup, üzerinde işlem yapmamızı sağlar. Örneğin, bir dosyadan veriyi almak, eksik verileri bulmak, verileri sıralamak ya da bazı verileri çıkarıp yeni bir dosya oluşturmak için kullanılır. Pandas sayesinde veriyi çok kolay bir şekilde analiz edebiliriz.

**2.Numpy**

NumPy, Python programlama dilinde sayısal ve bilimsel hesaplamalar yapmak için kullanılan en popüler kütüphanelerden biridir. NumPy, veriyi diziler (arrays) şeklinde organize eder, yani sayılardan oluşan listeleri veya çok boyutlu matrisleri kolayca yönetebilmemizi sağlar. Python’un standart listelerine göre NumPy dizileri çok daha verimlidir; daha az bellek kullanır ve daha hızlı işlemler yapmamıza olanak tanır.

NumPy, sayılarla çalışırken özellikle matematiksel ve istatistiksel hesaplamalar yapmak için mükemmel bir araçtır. Örneğin, diziler üzerinde toplama, çıkarma, çarpma gibi işlemleri hızlı bir şekilde gerçekleştirebiliriz. Ayrıca, matris işlemleri, ortalama, standart sapma gibi istatistiksel hesaplamalar ve lineer cebir gibi daha karmaşık hesaplamalar da NumPy ile yapılabilir.

NumPy, büyük veri setleri ile çalışırken yüksek performans sağlar. Özellikle veri bilimi, makine öğrenmesi ve bilimsel araştırmalar gibi alanlarda, veri üzerinde hızlı hesaplamalar yapabilmek için yaygın olarak kullanılır. NumPy dizileri, matematiksel işlem ve analizleri çok hızlı bir şekilde gerçekleştirmemizi sağlar, bu yüzden veri analizinde ve modelleme süreçlerinde vazgeçilmezdir.



* **pd.read\_csv()**: Bu, Pandas kütüphanesinin bir fonksiyonudur ve bir CSV (Comma Separated Values - Virgülle Ayrılmış Değerler) dosyasını okumamıza yarar. CSV dosyası, verileri satırlar ve sütunlar şeklinde düzenleyen basit bir dosya formatıdır. Genellikle tabular (tablo) veri içerir. Bu fonksiyon, CSV dosyasındaki verileri alıp, Python programımızda **DataFrame** adı verilen veri yapısına dönüştürür.
* **'DataMiningDataSet.csv'**: Bu, okunan CSV dosyasının dosya adıdır. Yani, bu dosya "DataMiningDataSet.csv" adlı bir dosyadır ve dosyanın bulunduğu konumda okunur. Eğer dosya, Python script’inizin bulunduğu klasörde değilse, tam dosya yolunu vermeniz gerekebilir.
* **df**: Bu, CSV dosyasından okunan verilerin **DataFrame** şeklinde saklandığı değişkendir. DataFrame, Pandas kütüphanesinin sunduğu bir veri yapısıdır ve genellikle tablo şeklindeki verilerle çalışırken kullanılır. Yani, CSV dosyasındaki her bir satır ve sütun, DataFrame yapısında satırlar ve sütunlar olarak temsil edilir.

Özetle, bu satırda 'DataMiningDataSet.csv' adlı CSV dosyasındaki veriler, Pandas kütüphanesinin **read\_csv** fonksiyonu aracılığıyla okunur ve **df** adlı bir **DataFrame**'e dönüştürülür. Artık **df** değişkeni ile veriler üzerinde analizler yapabiliriz.



* **np.random.choice()**: NumPy kütüphanesinin bir fonksiyonudur ve rastgele seçim yapmamıza olanak tanır. Bu fonksiyon, belirttiğiniz veri kümesinden (diziden, listeden vb.) rastgele öğeler seçer. Burada, fonksiyon kullanılarak belirli bir sayıda **rastgele veri** seçmek istiyoruz.
* **df.index**: Pandas DataFrame'lerinde her satırın bir **indeksi** vardır. **df.index**, df adlı DataFrame'inin satırlarının indekslerini yani satır numaralarını döndürür. Yani, bu kısımda, CSV dosyasındaki her satırın benzersiz bir numaralandırılması gibi düşünülebilir. Bu indeksler, hangi satırın hangi veriye ait olduğunu tanımlar.
* **size=50**: Bu parametre, rastgele seçilecek **öğe sayısını** belirtir. Yani, burada **50** tane rastgele indeks seçilmesi isteniyor. Bu 50 indeks, DataFrame’in satırlarının numaralarından (indekslerinden) seçilecek.
* **replace=False**: Bu parametre, seçilen öğelerin **tekrarlanıp tekrarlanmayacağını** belirtir. **replace=False** demek, her bir öğe yalnızca bir kez seçilebilir ve tekrar seçilemez anlamına gelir. Yani, 50 farklı indeks seçilecek ve aynı indeks bir daha seçilmeyecek.

**Özetle:**

Bu satır, **df.index** içindeki 50 farklı satır indeksini rastgele seçer ve **nan\_indices** adlı değişkene atar. Seçilen bu indeksler, daha sonra veri kümesinde **eksik (NaN) değerler** eklemek için kullanıcağız.

metin, yazı tipi, ekran görüntüsü, grafik içeren bir resim

Açıklama otomatik olarak oluşturuldu

**for row in nan\_indices:**

* **for row in nan\_indices:**: Bu, bir **for döngüsü**dür ve **nan\_indices** adlı liste veya dizideki her bir öğeyi (yani her bir satır indeksini) sırayla alır ve döngü içerisinde işlem yapar.
* **nan\_indices** daha önceki adımda, DataFrame'deki rastgele seçilen 50 satırın indekslerini içeriyor. Bu döngü, her bir seçilen satır indeksini **row** adıyla alır ve her bir satırda belirtilen işlemi tekrarlar.

**df.loc[row, 'Yas'] = np.nan**

* **df.loc[]**: Pandas'ta **loc[]** fonksiyonu, belirli bir satır ve sütun üzerinde işlem yapmak için kullanılır. **loc[]**, satır ve sütun etiketlerine göre veri seçmek, düzenlemek veya değiştirmek için kullanılır. Burada, **row** satır numarasını ve **Yas** sütun adını belirtiyoruz.
* **df.loc[row, 'Yas']**: Bu ifade, DataFrame **df**'de **row** indeksine sahip satırdaki **Yas** sütununu seçer.
* **= np.nan**: Bu kısmı, **Yas** sütunundaki seçilen hücreye **NaN** (Not a Number) değerini atar. **np.nan**, NumPy kütüphanesinde eksik veya geçerli olmayan veriyi temsil eden bir değerdir. Yani, bu işlemle **Yas** sütunundaki belirtilen satırlara **eksik veri (NaN)** yerleştirilmiş olur.

**Özetle:**

Bu döngü, **nan\_indices** içindeki her bir satır indeksini alarak, o satırdaki **Yas** sütununa **NaN** değeri atar. Bu işlem, verilerde rastgele **eksik veri** (NaN) oluşturarak, veri kümesindeki bazı hücreleri bilinçli olarak boş bırakmak için kullanılır.

ekran görüntüsü, metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**df.to\_csv('DataMiningDataSetContainsNancsv', index=False)**

* **df.to\_csv()**: Bu, Pandas DataFrame'ini bir CSV dosyasına kaydetmek için kullanılan bir fonksiyondur. **to\_csv()**, DataFrame'deki verileri, istediğimiz dosya formatına (burada CSV) dönüştürerek belirtilen bir dosyaya kaydeder. Bu sayede, işlenmiş veya değiştirilmiş veriyi dışa aktarabiliriz.
* **'DataMiningDataSetContainsNancsv'**: Bu, verilerin kaydedileceği dosya adıdır. Burada **DataMiningDataSetContainsNancsv** adlı bir CSV dosyasına veriler kaydedilecektir. Bu ismin bir dosya uzantısı (örneğin .csv) eksik görünüyor. Genellikle dosya uzantısı .csv olarak yazılır, yani doğru kullanım **'DataMiningDataSetContainsNancsv.csv'** olmalıdır. Bu isim, CSV dosyasının adını belirtir ve kaydedildiğinde bu dosya adıyla kaydedilecektir.
* **index=False**: Bu parametre, CSV dosyasına kaydedilen verilerin satır indekslerini (yani, her bir satırın numarasını) **dahil etmemek** için kullanılır. Pandas, DataFrame'leri kaydederken her satırın başına bir indeks numarası ekler. Eğer **index=False** parametresi kullanılırsa, bu indeks numaraları kaydedilmez ve sadece verilerin kendisi dosyaya yazılır. Yani, bu seçenekle yalnızca sütun adları ve veri değerleri CSV dosyasına kaydedilir, satır numaraları kaydedilmez.

**Özetle:**

Bu satır, **df** adlı DataFrame'deki verileri **DataMiningDataSetContainsNancsv** adlı bir dosyaya CSV formatında kaydeder. **index=False** parametresi sayesinde, her bir satırın indeks numarası CSV dosyasına dahil edilmez, sadece veri ve sütun isimleri kaydedilir. Bu işlem, verileri dışa aktarıp, daha sonra başka bir yerde kullanmak veya paylaşmak için faydalıdır.

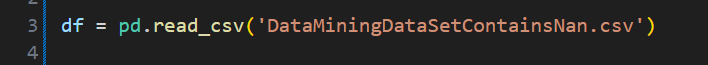
NAN DEĞERLERİ YERİNE ORTALAMA DEĞERİNİ EKLEME KODU AÇIKLAMASI

metin, yazı tipi, ekran görüntüsü, grafik içeren bir resim

Açıklama otomatik olarak oluşturuldu

**1. pandas**

Pandas, Python'da veriyle çalışırken en çok kullanılan kütüphanelerden biridir. Bu kütüphane, veriyi tablolar halinde düzenler. Mesela, bir Excel dosyasındaki veriler gibi. Pandas, bu veriyi çok rahat bir şekilde okuyup, üzerinde işlem yapmamızı sağlar. Örneğin, bir dosyadan veriyi almak, eksik verileri bulmak, verileri sıralamak ya da bazı verileri çıkarıp yeni bir dosya oluşturmak için kullanılır. Pandas sayesinde veriyi çok kolay bir şekilde analiz edebiliriz.



**df = pd.read\_csv('DataMiningDataSetContainsNan.csv')**

* **pd.read\_csv()**: Bu, Pandas kütüphanesinin en yaygın kullanılan fonksiyonlarından biridir ve bir **CSV (Comma Separated Values)** dosyasını okur. CSV dosyası, verileri virgülle ayırarak düzenleyen bir dosya formatıdır ve genellikle tabular (tablo şeklinde) veriler içerir. Pandas, bu CSV dosyasını alır ve veriyi bir **DataFrame** formatında döndürür. **read\_csv()** fonksiyonu sayesinde veriye erişebilir ve üzerinde işlem yapabiliriz.
* **'DataMiningDataSetContainsNan.csv'**: Bu, okunan dosyanın adı ve uzantısıdır. Burada **DataMiningDataSetContainsNan.csv** adlı bir CSV dosyasından veri alınacaktır. Bu dosya, **NaN (Not a Number)** içeren verilerle çalışmak için ekledik çünkü Nan yerine ortalam değerleri yazıcaz. Bu dosya verilerini okumak için Pandas **read\_csv()** fonksiyonu kullanılıyor.
* **df**: Bu, okunan verilerin saklandığı bir **DataFrame**'dir. Pandas DataFrame, tabular veri yapılarını yönetmek için kullanılan bir veri yapısıdır ve her sütunu bir **seriye**, her satırı ise bir **gözleme** karşılık gelir. Yani, **df** değişkeni, CSV dosyasındaki her satır ve sütunun **DataFrame** formatında bir temsiline sahip olacaktır. Bu sayede verilerle daha kolay işlem yapabiliriz.

**Özetle:**

Bu satır, **DataMiningDataSetContainsNan.csv** adlı CSV dosyasını okur ve veriyi **df** adlı bir **DataFrame** olarak Python ortamına alır. Artık **df** değişkeni ile CSV dosyasındaki verilere erişebilir ve veriler üzerinde analizler, temizlemeler veya dönüşümler yapabilirsiniz.



* **df['Yas']**: Bu kısım, **df** adlı DataFrame'deki **Yas** adlı sütuna erişir. Pandas DataFrame'lerinde, sütunlara **adlarıyla** ulaşabiliriz. Burada **'Yas'** sütununda kişilerin yaşları bulunduğunu varsayıyoruz. Yani, **df['Yas']**, **df** DataFrame'inin **Yas** sütunundaki tüm verileri alır.
* **df['Yas'].mean()**: **mean()** fonksiyonu, Pandas'ta bir sütunun **ortalamasını** hesaplamak için kullanılır. Bu fonksiyon, **Yas** sütunundaki tüm sayısal değerlerin ortalamasını bulur. Yani, **df['Yas'].mean()** ifadesi, **Yas** sütunundaki tüm yaş değerlerinin aritmetik ortalamasını döndürür.
* **round()**: **round()** fonksiyonu, verilen sayıyı **yuvarlar**. Burada, **df['Yas'].mean()** tarafından döndürülen ortalama değeri, en yakın tam sayıya yuvarlar. Bu sayede, kesirli bir sayı yerine, daha anlamlı bir tam sayı değeri elde edilmiş olur. Örneğin, ortalama 25.7 ise, **round()** fonksiyonu bunu 26'ya yuvarlar.
* **mean\_value**: Bu, hesaplanan yuvarlanmış ortalamanın **değerini** saklayacak olan değişkendir. Sonuç olarak, **mean\_value** değişkeni **Yas** sütunundaki tüm yaşların ortalamasının yuvarlanmış halini tutacaktır.

**Özetle:**

Bu satır, **Yas** sütunundaki tüm değerlerin ortalamasını hesaplar ve bu ortalamayı en yakın tam sayıya yuvarlar. Sonuç, **mean\_value** adlı değişkende saklanır. Bu işlem, eksik verileri (NaN) doldururken kullanılacak bir referans değeri olarak düşünülebilir, çünkü NaN değerlerini yerine koyarken genellikle sütunun ortalamasını kullanmak yaygın bir yöntemdir.



* **df['Yas']**: Bu, **df** adlı DataFrame'deki **Yas** adlı sütuna erişir. **df**, daha önce okunan veriyi saklayan DataFrame'imizdir ve **'Yas'** sütununda kişilerin yaşları yer almaktadır.
* **fillna(mean\_value)**: Pandas'ta **fillna()** fonksiyonu, bir sütundaki **eksik (NaN) değerleri** belirli bir değerle doldurmak için kullanılır. Burada **mean\_value**, **Yas** sütunundaki eksik değerleri yerine koymak için kullanılan değeri temsil eder. **mean\_value**, daha önce **df['Yas'].mean()** ile hesaplanan ve yuvarlanan ortalama değerdir. Yani, **fillna(mean\_value)** ifadesi, **Yas** sütunundaki tüm **NaN** (eksik) değerleri **mean\_value** ile doldurur.
* **df['Yas'] = ...**: Bu kısım, **fillna(mean\_value)** fonksiyonu ile elde edilen sonucu, **df['Yas']** sütununa geri atar. Yani, **Yas** sütunundaki eksik değerler, ortalama yaş değeri ile doldurulduktan sonra, bu yeni veri **df['Yas']** sütununa kaydedilir.

**Özetle:**

Bu satır, **Yas** sütunundaki eksik (NaN) değerleri **mean\_value** ile doldurur. **mean\_value**, **Yas** sütununun ortalama değeri olduğu için, eksik yaş verileri bu ortalama değerle tamamlanmış olur. Bu işlem, veriyi temizlemek ve eksik verilerle çalışırken analizlerinizi doğru şekilde yapmak için yaygın olarak kullanırız.

ekran görüntüsü, yazı tipi, metin, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**df.loc[df['Yas'] <= 20, 'IngilizceBilme'] = 0**

* **df.loc[...]**: Pandas'ta **loc[]** fonksiyonu, belirli bir koşula göre DataFrame üzerinde veri seçmek ve bu veriyi değiştirmek için kullanılır. Burada, koşul belirli bir yaş aralığına göre **IngilizceBilme** sütununu güncellemeyi amaçlamaktadır.
* **df['Yas'] <= 20**: Bu, **Yas** sütunundaki değerlerin 20 veya daha küçük olduğu satırları seçmek için kullanılan koşuldur. Yani, yaşları 20 veya daha küçük olan kişilerin verileri seçilecektir.
* **'IngilizceBilme' = 0**: Bu, seçilen satırlardaki **IngilizceBilme** sütunundaki değeri **0** olarak günceller. Bu işlem, yaşları 20 veya daha küçük olan kişiler için **IngilizceBilme** değerini 0 yapar, yani bu kişiler İngilizce bilmediği varsayılır.

**df.loc[(df['Yas'] > 20) & (df['Yas'] <= 40), 'IngilizceBilme'] = 1**

* **(df['Yas'] > 20) & (df['Yas'] <= 40)**: Burada iki koşul birleştirilmiştir. **Yas** sütunundaki değerlerin 20'den büyük ve 40'tan küçük veya eşit olduğu satırlar seçilir. Yani, yaşları 21 ile 40 arasında olan kişiler hedef alınır.
* **'IngilizceBilme' = 1**: Bu koşulu sağlayan satırlardaki **IngilizceBilme** sütunundaki değeri **1** olarak günceller. Bu işlem, yaşları 21 ile 40 arasındaki kişiler için **IngilizceBilme** değerini 1 yapar, yani bu kişilerin İngilizce bilmesi varsayılır.

**df.loc[df['Yas'] > 40, 'IngilizceBilme'] = 2**

* **df['Yas'] > 40**: Burada, **Yas** sütunundaki değeri 40'tan büyük olan satırlar seçilir. Yani, yaşları 41 ve daha büyük olan kişiler hedef alınır.
* **'IngilizceBilme' = 2**: Bu satırlardaki **IngilizceBilme** sütunundaki değeri **2** olarak günceller. Bu işlem, yaşları 41 ve daha büyük olan kişiler için **IngilizceBilme** değerini 2 yapar, yani bu kişilerin İngilizce bilgisi daha gelişmiş veya belirli bir seviyeye ulaşmış kabul edilir.

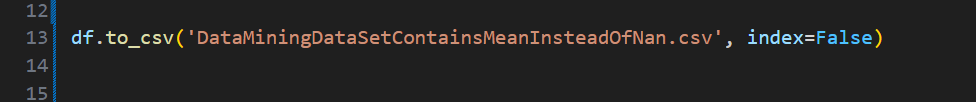
**Açıklama:**

Veri setinde **Yas** (yaş) sütunu üzerinde yapılan değişiklik, eksik verileri **ortalama değer** ile doldurduktan sonra verilerin anlamını bozmamak için gereklidir. Çünkü, yaş verisinin tamamlanması sırasında eksik değerler **ortalama yaş** ile doldurulmuştu ve bu, yaş dağılımını gerçek veriden farklılaştırabilir.

Bu nedenle, **IngilizceBilme** sütunundaki değerleri, **yaş gruplarına göre belirlemek** amacıyla bu işlemler yapılmıştır.

* Yaşları 20'den küçük olanlar, İngilizce bilmeyen kişiler (0 olarak işaretlenmiş),
* Yaşları 21 ile 40 arasında olanlar, İngilizceyi bilen kişiler (1 olarak işaretlenmiş),
* Yaşları 40'tan büyük olanlar ise, daha ileri seviyede İngilizce bilen kişiler (2 olarak işaretlenmiş) olarak kabul edilmiştir.

Bu işlem, **verinin anlamını korumak** ve doldurulan eksik değerlerin doğru şekilde analiz edilmesini sağlamak için yapılır. Yani, yaş verisinin değişmesiyle birlikte, **IngilizceBilme** etiketlerinin doğru bir şekilde güncellenmesi sağlanmış olur.



* **df.to\_csv()**: Bu, Pandas kütüphanesinin bir fonksiyonudur ve **DataFrame** olan **df** adlı veri setini bir CSV dosyasına kaydetmek için kullanılır. **df**, üzerinde çalıştığınız ve çeşitli işlemler yaptığınız veri kümesini temsil eder.
* **'DataMiningDataSetContainsMeanInsteadOfNan.csv'**: Bu, veri setinin kaydedileceği dosya adıdır. Burada **DataMiningDataSetContainsMeanInsteadOfNan.csv** adıyla, CSV dosyasının içeriğini dışarıya aktaracağız. Bu isim, dosyanın nasıl adlandırıldığını belirtir ve genellikle dosyanın içeriği hakkında bilgi verir (örneğin, eksik verilerin yerine ortalama değerlerin konduğuna dair bir izlenim verir dışardan okuyan biri için ).
* **index=False**: Pandas'ta, **to\_csv()** fonksiyonu, varsayılan olarak **DataFrame**'in satır numaralarını (index değerlerini) da CSV dosyasına kaydeder. Ancak burada **index=False** parametresi kullanılarak satır numaralarının kaydedilmesi engellenmiştir. Bu, sadece verilerin (sütun adları ve değerler) dosyaya yazılmasını sağlar ve satır numaralarının dışarıya aktarılmasını önler.

**Özetle:**

Bu satır, **df** veri setindeki verileri, **DataMiningDataSetContainsMeanInsteadOfNan.csv** adıyla bir CSV dosyasına kaydeder. Ayrıca, satır numaralarının dosyaya dahil edilmemesini sağlamak için **index=False** parametresi kullanılır. Bu işlem, verilerin dışa aktarılması ve ileride başka bir analiz için kaydedilmesi amacıyla yapılır.