A.Ü. ZİRAAT FAKÜLTESİ

TARIM MAKİNALARI VE TEKNOLOJİLERİ MÜHENDİSLİĞİ BÖLÜMÜ

TARIMDA VERİ BİLİMİNE GİRİŞ

Doç. Dr. Necati Çetin

1

19.04.2024

KAYNAKLAR

Python ile Makine Öğrenmesi
(Prof. Dr. Engin Sorhun)

Makine Öğrenmesi Teorik Yönleri ve Python Uygulamaları
(Dr. Öğr. Üyesi Sinan Uğuz)

Uygulamalarla Veri Bilimi
(Doç. Dr. Deniz Kılınç – Nezahat Başeğmez)

Python ile Makine Öğrenmesi
(Prof. Dr. Sadi Evren Şeker)

İnternet Kaynakları

necati.cetin@ankara.edu.tr
necaticetin1990@gmail.com

MAKİNE ÖĞRENMESİ ALGORİTMALARI

K-Nearest Neigbor (K-En Yakın Komşu)

K-En Yakın Komşu (K-NN) algoritması denetimli öğrenme tekniğini temel alan en basit makine öğrenmesi algoritmalarından birisidir.

K-NN algoritması, yeni veri ile mevcut veriler arasındaki benzerliği varsayar ve yeni veriyi mevcut verilere en çok benzeyen sınıfa/kategoriye yerleştirir.

K-NN algoritması mevcut tüm verileri saklar ve benzerliğe göre yeni veri noktasını sınıflandırır. Bu, yeni veriler ortaya çıktığında K-NN algoritması kullanılarak kolayca uygun bir kategoriye sınıflandırılabileceği anlamına gelir.

K-NN algoritması **Regresyon** için kullanılabileceği gibi **Sınıflandırma** için de kullanılabilir ancak çoğunlukla Sınıflandırma problemlerinde kullanılmaktadır.

K-NN parametrik olmayan bir algoritmadır, yani temel veriler üzerinde herhangi bir varsayımda bulunmaz. Tembel öğrenen algoritma olarak da adlandırılır çünkü eğitim setinden hemen öğrenmez, bunun yerine veri setini saklar ve sınıflandırma sırasında veri seti üzerinde bir eylem gerçekleştirir.

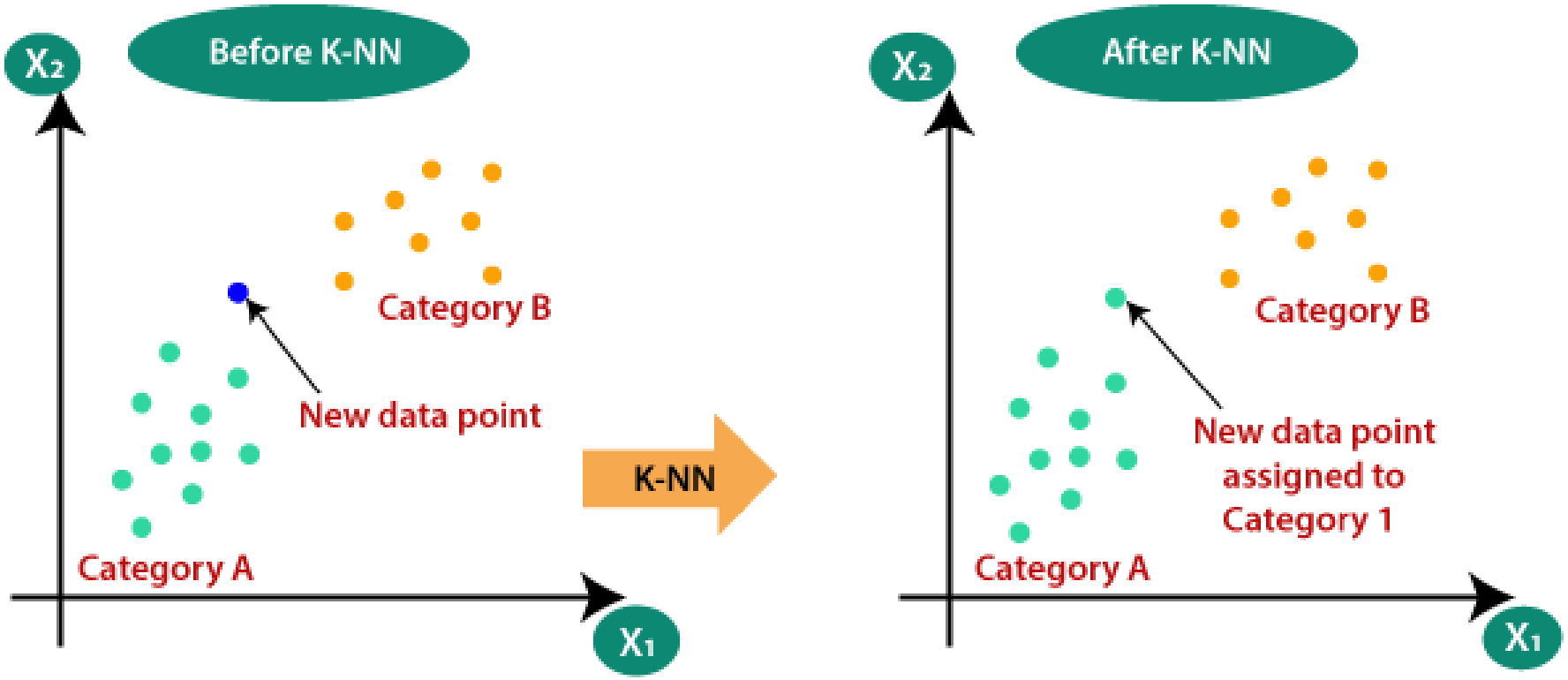
KNN algoritması eğitim aşamasında sadece veri setini saklar ve yeni veri aldığında bu yeni veriyi veri setine çok benzeyen bir kategoriye sınıflandırır.

Örnek: Diyelim ki elimizde kedi ve köpeğe benzeyen bir canlı resmi var. Onun kedi mi köpek mi olduğunu bilmek istiyoruz. Bu tanımlama için benzerlik ölçüsü üzerinde çalışan KNN algoritmasını kullanabiliriz. KNN modelimiz, yeni veri setinin kedi ve köpek görsellerine benzer özelliklerini bulacak ve en benzer özelliklere göre onu kedi veya köpek kategorisine koyacaktır.

Neden bir K-NN Algoritmasına ihtiyacımız var?

Diyelim ki Kategori A ve Kategori B olmak üzere iki kategori var ve yeni bir x_1 veri noktamız var. Bu veri noktası bu kategorilerden hangisinde yer alacak?

Bu tür problemleri çözmek için bir K-NN algoritmasına ihtiyacımız var. K-NN'nin yardımıyla belirli bir veri kümesinin kategorisini veya sınıfını kolayca tanımlayabiliriz.


Before K-NN
X2
Category B
New data point
Category A
X1
K-NN
After K-NN
X2
Category B
New data point
assigned to
Category 1
Category A
X1

K-NN nasıl çalışır?

K-NN'nin çalışması aşağıdaki algoritmaya dayanarak açıklanabilir:

Adım-1: Komşuların **K** sayısını seçin

Adım-2: K sayıda komşunun **Öklid mesafesini** hesaplayın

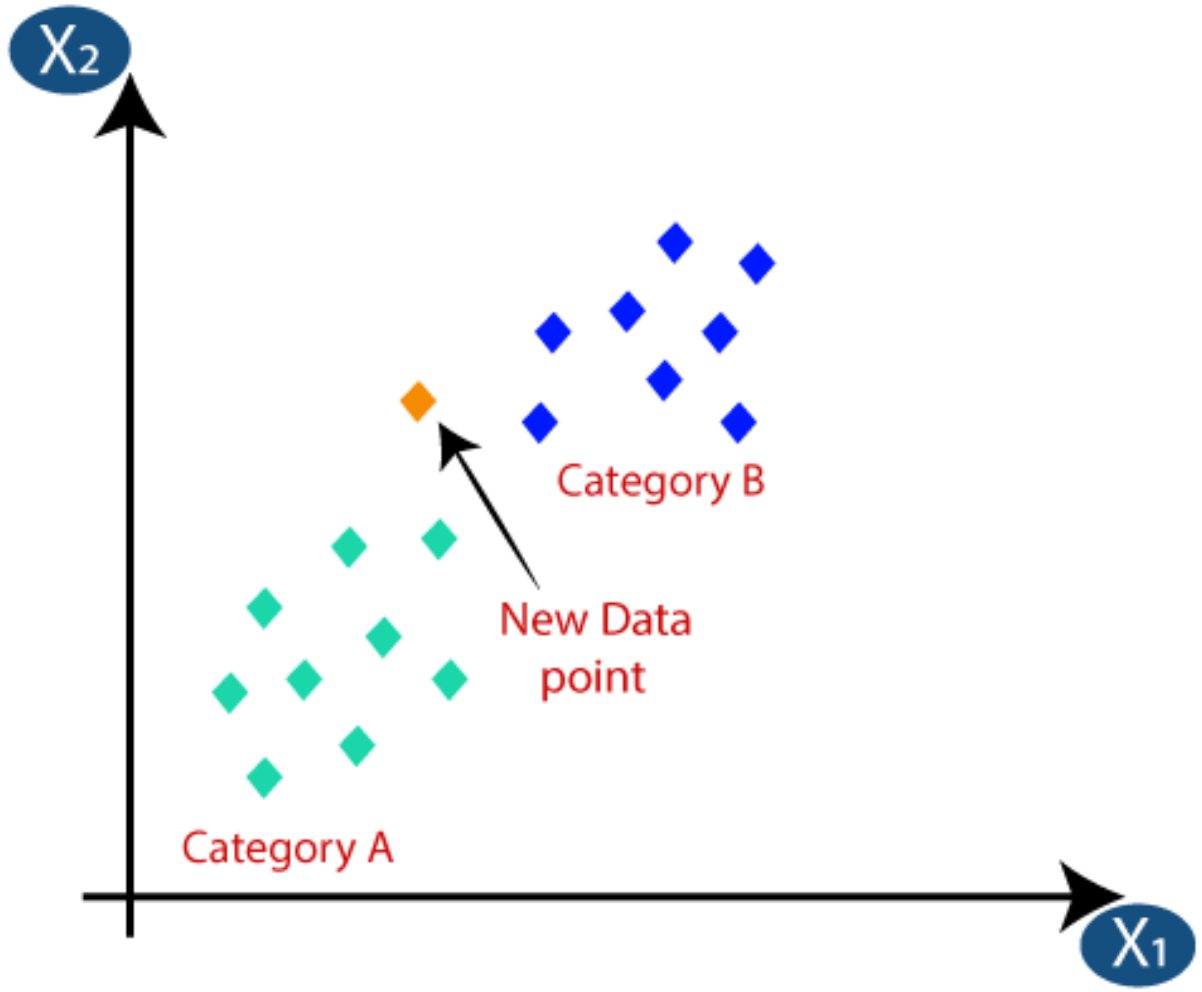
Adım-3: Hesaplanan Öklid mesafesine göre **en yakın K komşuyu** alın.

Adım-4: Bu K komşu arasında, her kategorideki **veri noktalarının sayısını sayın.**

Adım-5: Yeni veri noktalarını komşu sayısının **maksimum (en çok) olduğu kategoriye atayın.**

Adım-6: Modelimiz hazır.

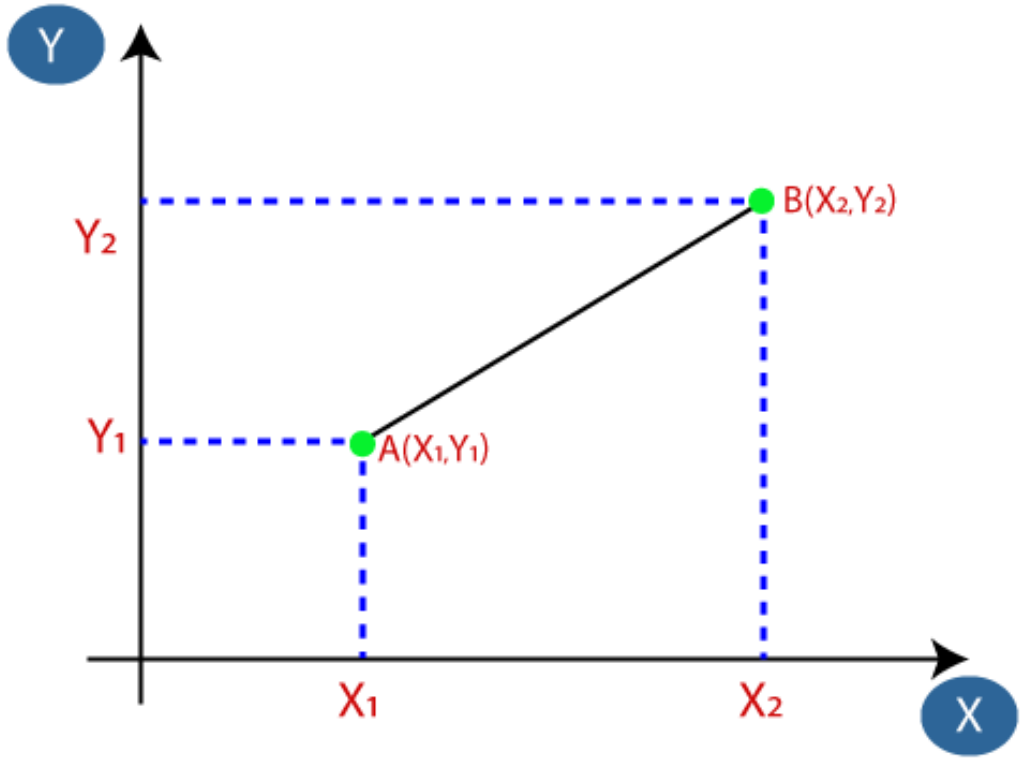
Diyelim ki yeni bir veri noktamız var ve onu gerekli kategoriye koymamız gerekiyor. Aşağıdaki görüntüyü göz önünde bulundurun:



Öncelikle K komşu sayısını seçeceğiz. Burada K=5 olarak seçelim.

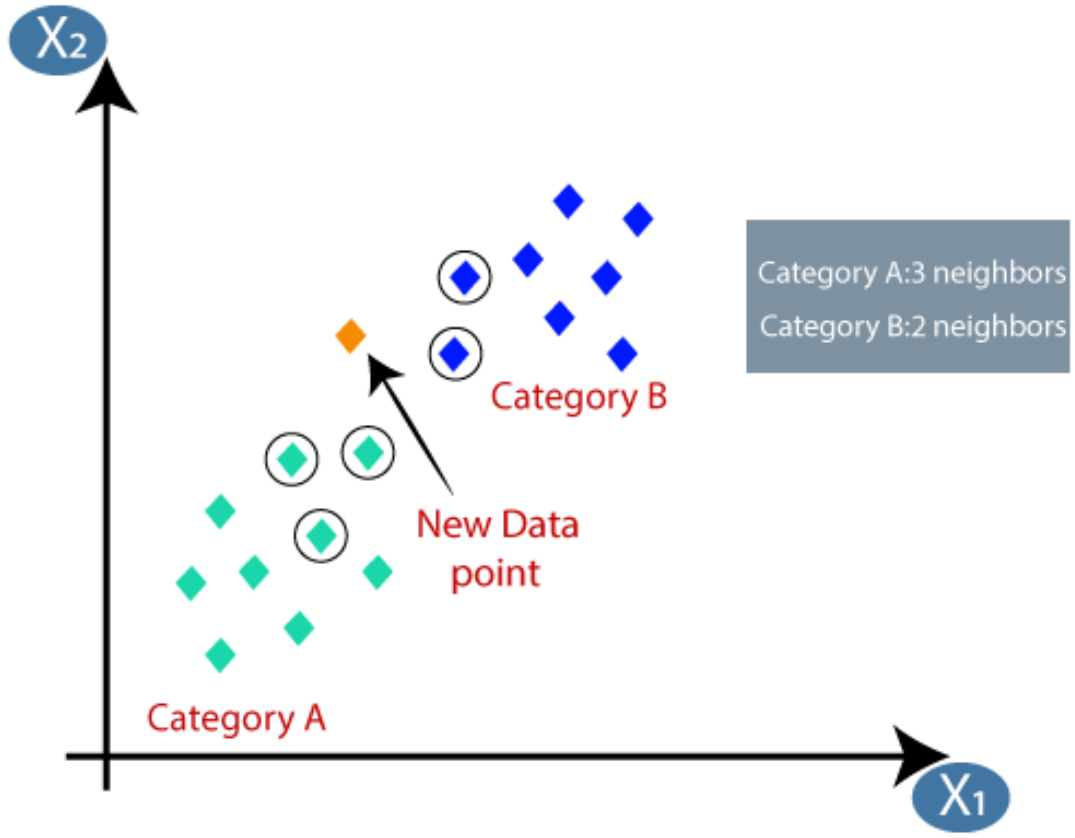
Daha sonra veri noktaları arasındaki Öklid mesafesini hesaplayacağız.

Öklid mesafesi, geometride iki nokta arasındaki mesafedir ve şu şekilde hesaplanabilir:

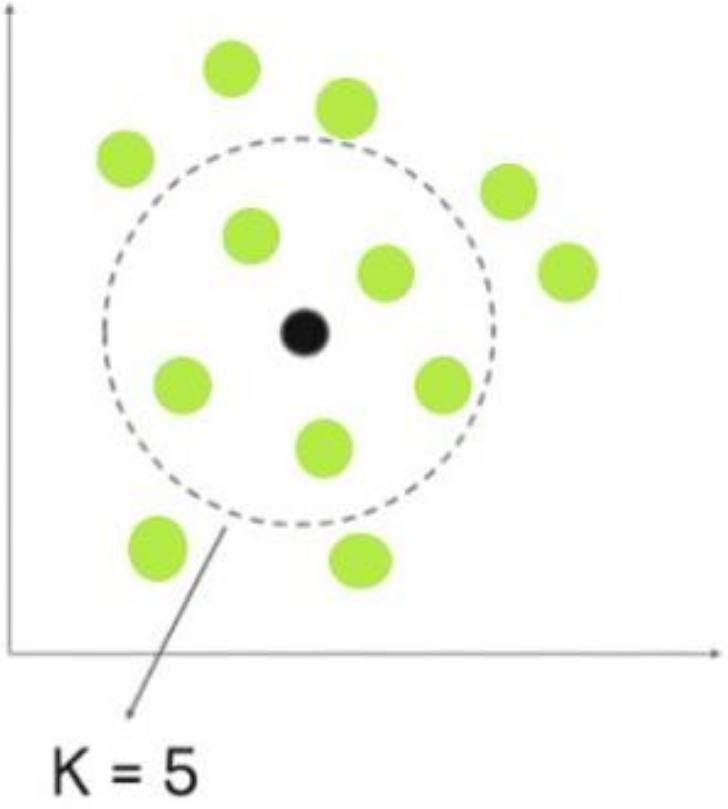


Euclidean Distance between A_1 and $B_2 = \sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

Öklid mesafesini hesaplayarak, A kategorisinde en yakın 3 komşu ve B kategorisinde en yakın 2 komşu olmak üzere en yakın komşuları elde ettik. Aşağıdaki görseli inceleyin:



Gördüğünüz gibi en yakın 3 komşu A kategorisindedir, dolayısıyla bu yeni veri noktası **A kategorisine** ait olmalıdır.



Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

Algoritmaya yönelik uzaklık hesaplaması tipleri (sadece bir kısmı);

- ***Euclidean*** (***Öklidyen***) Uzaklık hesaplaması,
- ***Manhattan*** Uzaklık Hesaplaması,
- ***Chebyshev*** Uzaklık Hesaplaması,
- ***Hamming*** Uzaklık Hesaplaması,
- ***Minkowski*** Uzaklık Hesaplaması,
- ***Mahalanobis*** Uzaklık Hesaplaması,
- ***Haversiene*** Uzaklık Hesaplaması,
- ***Levenshtein*** Uzaklık Hesaplaması,
- ***Sørensen-Dice*** Uzaklık Hesaplaması,
- ***Jacckard*** Uzaklık Hesaplaması.

K-NN Algoritmasında K değeri nasıl seçilir?

"K" için en iyi değeri belirlemenin özel bir yolu yoktur, dolayısıyla bunlardan en iyisini bulmak için bazı değerleri **denememiz** gerekir.

K için en çok tercih edilen değer 5'tir. K=1 veya K=2 gibi çok düşük bir K değeri gürültülü olabilir ve modelde aykırı değerlerin etkilerine yol açabilir. K için büyük değerler iyidir ancak bazı zorluklarla karşılaşılabilir.

KNN Algoritmasının Avantajları

- Uygulanması basittir.
- Gürültülü eğitim verilerine karşı dayanıklıdır
- Eğitim verilerinin büyük olması daha etkili olabilir.

KNN Algoritmasının Dezavantajları

- Bazen karmaşık olabilecek K değerini, her zaman deneyerek belirlememiz gerekir.
- Tüm eğitim örnekleri için veri noktaları arasındaki mesafenin hesaplanması nedeniyle hesaplama süreci fazladır.

KNN algoritmasının Python uygulaması

K-NN Algoritmasının Problemi:

Yeni bir SUV araba üreten bir araba üreticisi şirket var. Şirket, reklamları, söz konusu aracı satın almakla ilgilenen kullanıcılara göstermek istiyor. Bu problem için, sosyal ağ üzerinden birden fazla kullanıcının bilgisini içeren bir veri seti oluşturuluyor. Veri seti pek çok bilgi içeriyor ancak **bağımsız değişken** için dikkate alacağımız **Yaş** ve **Tahmini Maaş** değişkenlerini, **bağımlı değişken** için ise **Satın Alınan** değişkenini dikkate alacağız.

User ID	Gender	Age	EstimatedSalary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	0
15728773	Male	27	58000	0
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	0
15727311	Female	35	65000	0
15570769	Female	26	80000	0
15606274	Female	26	52000	0
15746139	Male	20	86000	0
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1
15617482	Male	45	26000	1
15704583	Male	46	28000	1
15621083	Female	48	29000	1
15649487	Male	45	22000	1
15736760	Female	47	49000	1

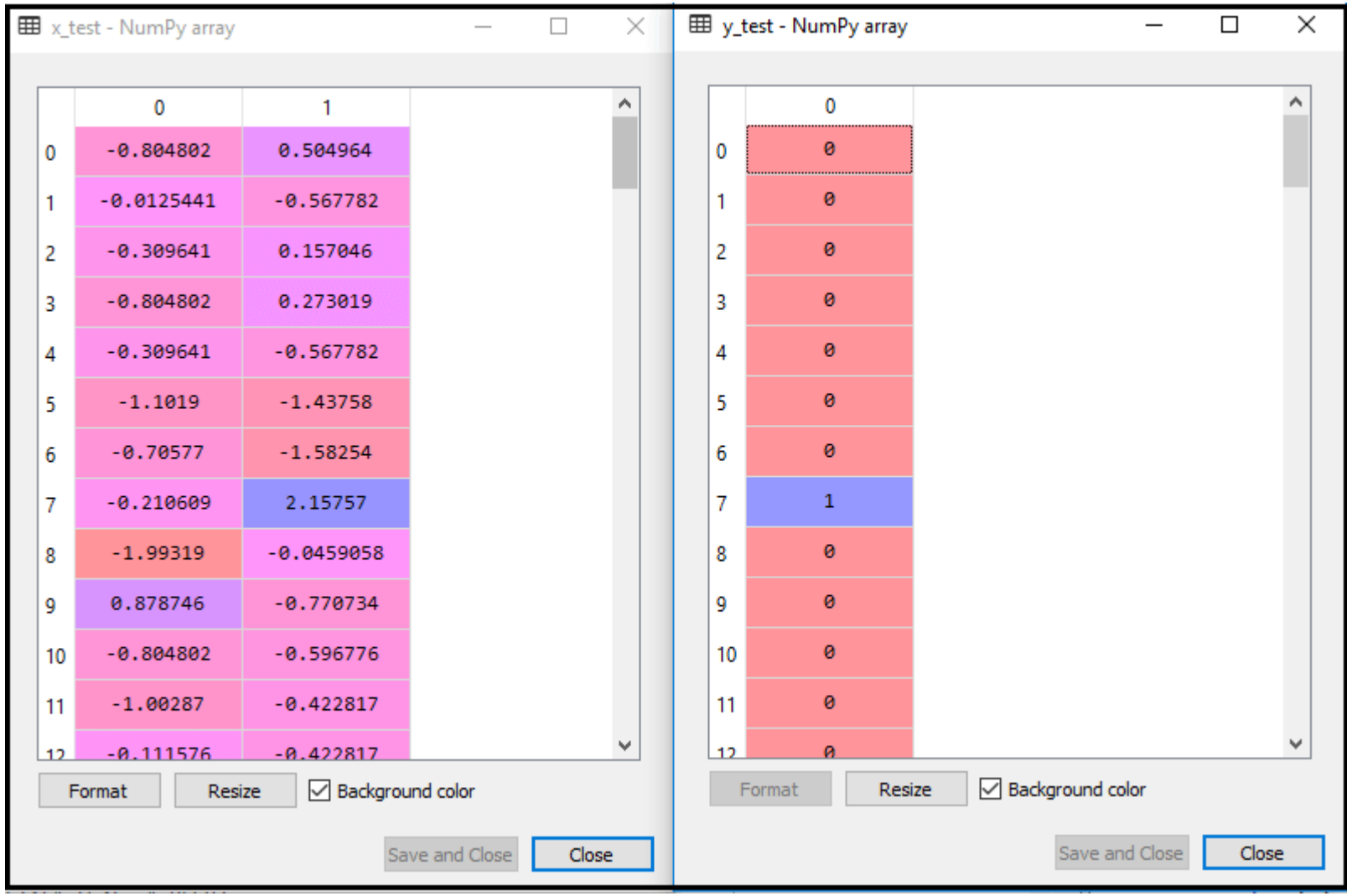
K-NN algoritmasını uygulama adımları:

- Veri Ön İşleme adımı
- K-NN algoritmasını Eğitim setine uydurma (fit etme, eğitme)
- Test sonucunu tahmin etmek
- Sonucun doğruluğunu test etmek (Karışıklık matrisinin oluşturulması)

Veri Ön İşleme Adımı

```
# importing libraries
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd

#importing datasets
data_set= pd.read_csv('user_data.csv')

#Extracting Independent and dependent Variable
x= data_set.iloc[:, [2,3]].values
y= data_set.iloc[:, 4].values

# Splitting the dataset into training and test set.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_state=0)

#feature Scaling
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

Yukarıdaki kodu çalıştırarak veri setimiz programımıza aktarılır ve önceden işlenir. Özellik ölçeklendirmeden sonra test veri kümemiz şöyle görünecektir:



	0	1
0	-0.804802	0.504964
1	-0.0125441	-0.567782
2	-0.309641	0.157046
3	-0.804802	0.273019
4	-0.309641	-0.567782
5	-1.1019	-1.43758
6	-0.70577	-1.58254
7	-0.210609	2.15757
8	-1.99319	-0.0459058
9	0.878746	-0.770734
10	-0.804802	-0.596776
11	-1.00287	-0.422817
12	-0.111576	-0.422817

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0
9	0
10	0
11	0
12	0

Eğitim verilerine uydurmak (fit etmek, eğitmek)

Şimdi K-NN sınıflandırıcısını eğitim verilerine fit edeceğiz.

Bunu yapmak için **Sklearn Neighbors** kütüphanesinin **KNeighborsClassifier** sınıfını içe aktaracağız.

Sınıfı içe aktardıktan sonra sınıfın **Classifier** nesnesini oluşturacağız.

Bu sınıfın model parametreleri şu şekilde olacaktır:

n_neighbors: Algoritmanın gerekli komşularını tanımlamak için. Genellikle 5 alır.

metric='minkowski': Bu varsayılan parametredir ve noktalar arasındaki mesafeye karar verir.

p=2: Standart Öklid metriğine eşdeğerdir.

Daha sonra sınıflandırıcıyı eğitim verilerine uyarlayacağız.

```
#Fitting K-NN classifier to the training set
from sklearn.neighbors import KNeighborsClassifier
classifier= KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2 )
classifier.fit(x_train, y_train)
```

Output: Yukarıdaki kodu çalıştırdığımızda çıktıyı şu şekilde elde edeceğiz:

```
Out[10]:
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

Test Sonucunu Tahmin Etme

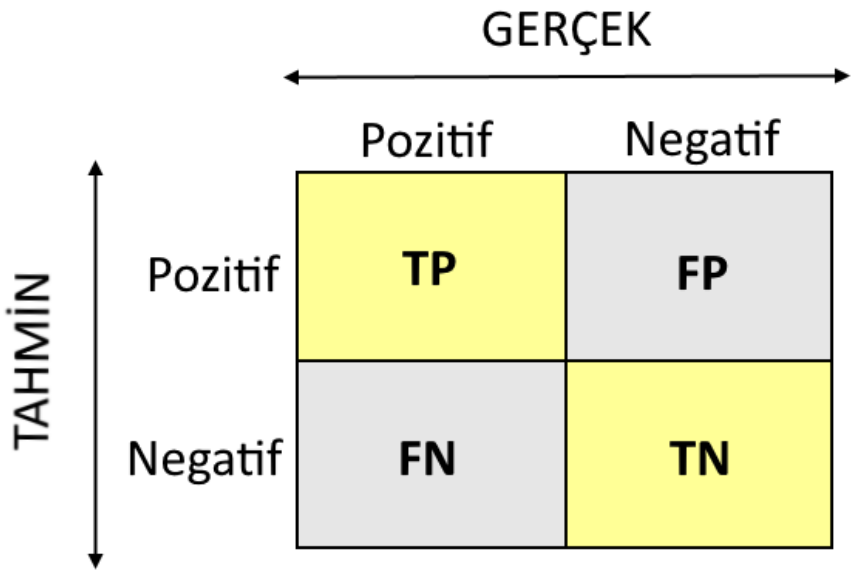
Test seti sonucunu tahmin etmek için bir **y_pred** vektörü oluşturacağız.

Aşağıda bunun kodu verilmiştir:

```
#Predicting the test set result
y_pred= classifier.predict(x_test)
```

Output:

Yukarıdaki kodun çıktısı şöyle olacaktır:



	Kuş	Kedi	Köpek
Kuş	15	3	4
Kedi	6	14	1
Köpek	5	2	20

Sarı renk doğru sınıflandırılanları, gri renk ise yanlış sınıflandırılanları temsil ediyor.

Karışıklık Matrisi

Yukarıda görülen matris sayesinde tahmin edilen değerlerin gerçek değerlerle kıyası yapılabilmektedir. Matris içinde verilenler ise, 1: TRUE ve 0: FALSE olacak şekilde, aşağıdaki gibidir:

TP (True Positive): 1 olarak sınıflandırdığımız ve gerçekten de 1 olan değerlerin sayısıdır.

FP (False Positive): 1 olarak sınıflandırdığımız fakat gerçekte 0 olan değerlerin sayısıdır.

FN (False Negative): 0 olarak sınıflandırdığımız fakat gerçekte 1 olan değerlerin sayısıdır.

TN (True Negative): 0 olarak sınıflandırdığımız ve gerçekten de 0 olan değerlerin sayısıdır.

Karışıklık Matrisinin Oluşturulması

Sınıflandırıcının doğruluğunu görmek için K-NN modelimizin **Karışıklık Matrisini** oluşturacağız.

```
#Creating the Confusion matrix
    from sklearn.metrics import confusion_matrix
    cm= confusion_matrix(y_test, y_pred)
```

Yukarıdaki kodda, merge_matrix fonksiyonunu içe aktardık ve cm değişkenini kullanarak çağırdık.

Output: Yukarıdaki kodu çalıştırdığımızda aşağıdaki matrisi elde edeceğiz:

Görselde 64+29= 93 doğru tahmin, 3+4= 7 yanlış tahmin olduğunu görüyoruz.

Accuracy (Doğruluk) değerini hesaplayınız?

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

Precision (Kesinlik) değerini hesaplayınız?

$$\text{Precision} = \frac{TP}{TP + FP}$$

Precision: Doğru sınıflandırılan verilerin oranını verir.

Recall değerini hesaplayınız?

$$\text{Recall} = \frac{TP}{TP + FN}$$

Recall: Sadece pozitif değerlerden doğru sınıflandırılanların oranını verir.

F-measure (F-score) değerini hesaplayınız?

$$\text{F-Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

F-Score (F-measure): Precision ve Recall değerlerinin harmonik ortalamasını verir.

Sensitivity değerini hesaplayınız?

$$\text{Sensitivity} = \frac{TP}{TP + FP}$$

Sensitivity: Recall ile aynıdır. Pozitif olarak sınıflandırılan verilerin gerçekteki pozitif verilere oranını verir.

Specificity değerini hesaplayınız?

$$\text{Specificity} = \frac{TN}{TN + FP}$$

Specificity: Sadece negatif olarak sınıflandırılan verilerin gerçekteki negatif verilere oranını verir.