KNN ÖRNEĞİ

```
1.# Standart kütüphaneleri projeye dahil ederiz
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd


2.# verisetimizi okuruz
dataset = pd.read_csv("../input/iris/Iris.csv")


3.dataset.info()
```

çıktı

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

```
4.# Sınıfların gözlemlenmesi
dataset.Species.unique()
```

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

```
5.dataset.head()
```

çıktı

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

```
6.# Öznitelik ve Etiketleri ayırdık
X = dataset[["SepalLengthCm", "SepalWidthCm","PetalLengthCm","PetalWidthCm"]]
y = dataset[["Species"]]
```

```
X
```

çıktı

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

Y
çıktı

	Species
0	Iris-setosa
1	Iris-setosa
2	Iris-setosa
3	Iris-setosa
4	Iris-setosa
...	...
145	Iris-virginica
146	Iris-virginica
147	Iris-virginica
148	Iris-virginica
149	Iris-virginica

```
7.# Eğitim ve Test verisetini ayırdık
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, rand
om_state=5)

8. from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=8)
classifier.fit(X_train, y_train)
```

```
    KNeighborsClassifier(n_neighbors=8)
```

```
9.y_pred = classifier.predict(X_test)
  y_pred
```

çıktı

```
array([1, 2, 2, 0, 2, 1, 0, 1, 0, 1, 1, 2, 2, 2, 0, 0, 2, 2, 0, 0, 1, 2,
       0, 1, 1, 2, 1, 1, 1, 2])
```

```
10.# Etiket kodlamasını tersine çevirip tahmin ve gerçek değerlerimizi gözleml
eyelim
pd.DataFrame(data={'Tahmin Değeri': encoder.inverse_transform(y_pred), 'Gerçek
Değer': encoder.inverse_transform(y_test), "Durum": y_pred == y_test})
```

Çıktı

	Tahmin Değeri	Gerçek Değer	Durum
0	Iris-versicolor	Iris-versicolor	True
1	Iris-virginica	Iris-virginica	True
2	Iris-virginica	Iris-virginica	True
3	Iris-setosa	Iris-setosa	True
4	Iris-virginica	Iris-virginica	True
5	Iris-versicolor	Iris-versicolor	True
6	Iris-setosa	Iris-setosa	True
7	Iris-versicolor	Iris-versicolor	True
8	Iris-setosa	Iris-setosa	True
9	Iris-versicolor	Iris-versicolor	True
10	Iris-versicolor	Iris-versicolor	True
11	Iris-virginica	Iris-virginica	True
12	Iris-virginica	Iris-virginica	True
13	Iris-virginica	Iris-virginica	True
14	Iris-setosa	Iris-setosa	True
15	Iris-setosa	Iris-setosa	True
16	Iris-virginica	Iris-virginica	True
17	Iris-virginica	Iris-virginica	True
18	Iris-setosa	Iris-setosa	True
19	Iris-setosa	Iris-setosa	True
20	Iris-versicolor	Iris-versicolor	True
21	Iris-virginica	Iris-virginica	True
22	Iris-setosa	Iris-setosa	True

Test verisinde aldığımız skor

```
from sklearn.metrics import confusion_matrix, accuracy_score, classification_r
eport, plot_confusion_matrix
print("Test accuracy: ", classifier.score(X_test, y_test))
print("Train accuracy: ",classifier.score(X_train, y_train))
print(confusion_matrix(y_test, y_pred)) # Setosa - 0, Versicolor - 1 , Virgini
ca - 2.
print(classification_report(y_test, y_pred))
```

çıktı

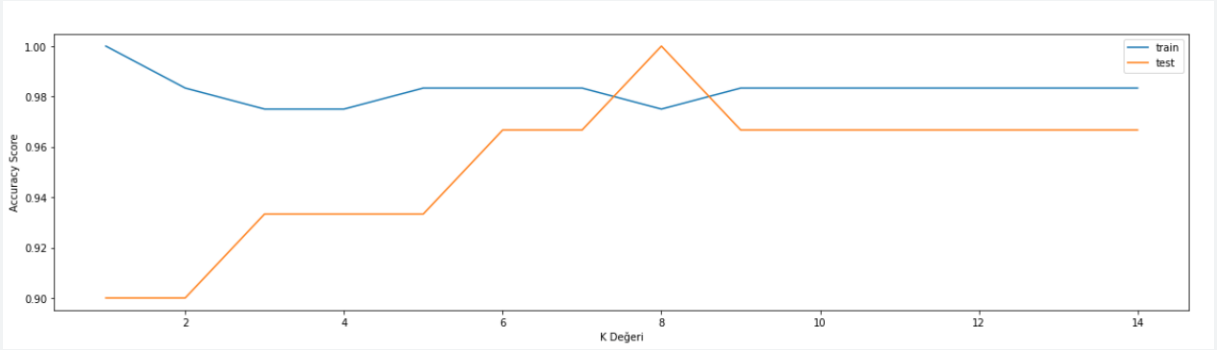
```
Test accuracy:  1.0
Train accuracy:  0.975
[[ 8  0  0]
 [ 0 11  0]
 [ 0  0 11]]
```

Karışıklık matrisi

```
uzunluk = range(1,15)
error1= []
error2= []
for k in uzunluk:
    classifier= KNeighborsClassifier(n_neighbors=k)
    classifier.fit(X_train,y_train)
    error1.append(classifier.score(X_train, y_train))
    error2.append(classifier.score(X_test, y_test))

plt.figure(figsize=(20,5))
```

```
plt.plot(uzunluk,error1,label="train")
plt.plot(uzunluk,error2,label="test")
plt.xlabel('K Değeri')
plt.ylabel('Accuracy Score')
plt.legend()
```

çıktı



KARAR AĞACI ÖRNEĞİ

```
# Verisetinin okunması
dataset = pd.read_csv("../input/iris/Iris.csv")

# Öznitelik ve Etiketlerin ayrılması
X = dataset[["SepalLengthCm", "SepalWidthCm","PetalLengthCm","PetalWidthCm"]]
y = dataset[["Species"]]

# Etiket Kodlaması
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
y = encoder.fit_transform(y) # Setosa - 0, Versicolor - 1 , Virginica - 2

# Eğitim ve Test verisetinin ayrılması
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, rand
om_state=5)

# Karar ağacı ve ağaç modülü
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(max_depth=6)
classifier.fit(X_train, y_train)

# Tahminlerin yapılması - Daha önce açıklandı
y_pred = classifier.predict(X_test)

# Etiket kodlamasını tersine çevirip tahmin ve gerçek değerlerimizi gözlemleye
lim
pd.DataFrame(data={'Tahmin Değeri': encoder.inverse_transform(y_pred), 'Gerçek
Değer': encoder.inverse_transform(y_test), "Durum": y_pred == y_test})
```

çıktı

	Tahmin Değeri	Gerçek Değer	Durum
0	Iris-versicolor	Iris-versicolor	True
1	Iris-virginica	Iris-virginica	True
2	Iris-virginica	Iris-virginica	True
3	Iris-setosa	Iris-setosa	True
4	Iris-virginica	Iris-virginica	True
5	Iris-virginica	Iris-versicolor	False
6	Iris-setosa	Iris-setosa	True
7	Iris-virginica	Iris-versicolor	False
8	Iris-setosa	Iris-setosa	True
9	Iris-versicolor	Iris-versicolor	True
10	Iris-versicolor	Iris-versicolor	True
11	Iris-virginica	Iris-virginica	True
12	Iris-virginica	Iris-virginica	True
13	Iris-virginica	Iris-virginica	True
14	Iris-setosa	Iris-setosa	True
15	Iris-setosa	Iris-setosa	True
16	Iris-virginica	Iris-virginica	True
17	Iris-virginica	Iris-virginica	True
18	Iris-setosa	Iris-setosa	True
19	Iris-setosa	Iris-setosa	True
20	Iris-versicolor	Iris-versicolor	True
21	Iris-virginica	Iris-virginica	True
22	Iris-setosa	Iris-setosa	True

```
# Skorların gözlemlenmesi
from sklearn.metrics import confusion_matrix, accuracy_score, classification_r
eport, plot_confusion_matrix
print("Test accuracy: ", classifier.score(X_test, y_test))
print("Train accuracy: ",classifier.score(X_train, y_train))
print(confusion_matrix(y_test, y_pred)) # Setosa - 0, Versicolor - 1 , Virgini
ca - 2.
print(classification_report(y_test, y_pred))
```

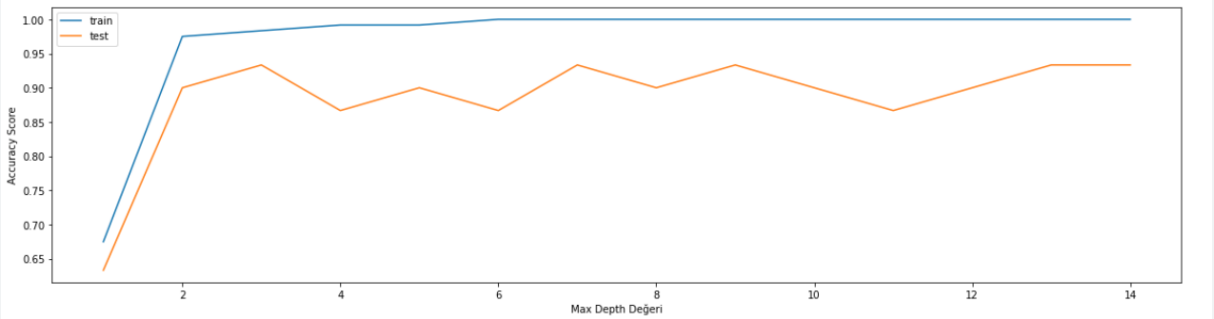
çıktı

```
Test accuracy:  0.9333333333333333
Train accuracy:  1.0
[[ 8  0  0]
 [ 0  9  2]
 [ 0  0 11]]
```

```
uzunluk = range(1,15)
error1= []
error2= []
for d in uzunluk:
    classifier= DecisionTreeClassifier(max_depth=d)
    classifier.fit(X_train,y_train)
    error1.append(classifier.score(X_train, y_train))
    error2.append(classifier.score(X_test, y_test))
plt.figure(figsize=(20,5))
plt.plot(uzunluk,error1,label="train")
plt.plot(uzunluk,error2,label="test")
plt.xlabel('Max Depth Değeri')
plt.ylabel('Accuracy Score')
plt.legend()
```

çıktı

```
<matplotlib.legend.Legend at 0x7f2cc9003550>
```



REGRESYON ÖRNEĞİ

2 değişken arasındaki doğrusal ilişkinin bir doğru denklemi olarak tanımlanıp, değişkenin değerlerinden biri bilindiğinde diğeri hakkında tahmin yapılmasını sağlar. Veriler arasında doğru tahmini yapabilmek için veriler için en iyi doğruyu oluşturmak gerekir. En iyi doğruyu oluştururken tüm noktalara en yakın bölge tercih edilmelidir. Lineer Regresyon'da bir doğru oluşturacağımız için bir bağımlı ve bir bağımsız değişken olmak üzere toplam 2 değişken üzerinde çalışılır.

Lineer Regresyon Algoritması

2 değişken arasındaki doğrusal ilişkinin bir doğru denklemi olarak tanımlanıp, değişkenin değerlerinden biri bilindiğinde diğeri hakkında tahmin yapılmasını sağlar. Veriler arasında doğru tahmini yapabilmek için veriler için en iyi doğruyu oluşturmak gerekir. En iyi doğruyu oluştururken tüm noktalara en yakın bölge tercih edilmelidir.

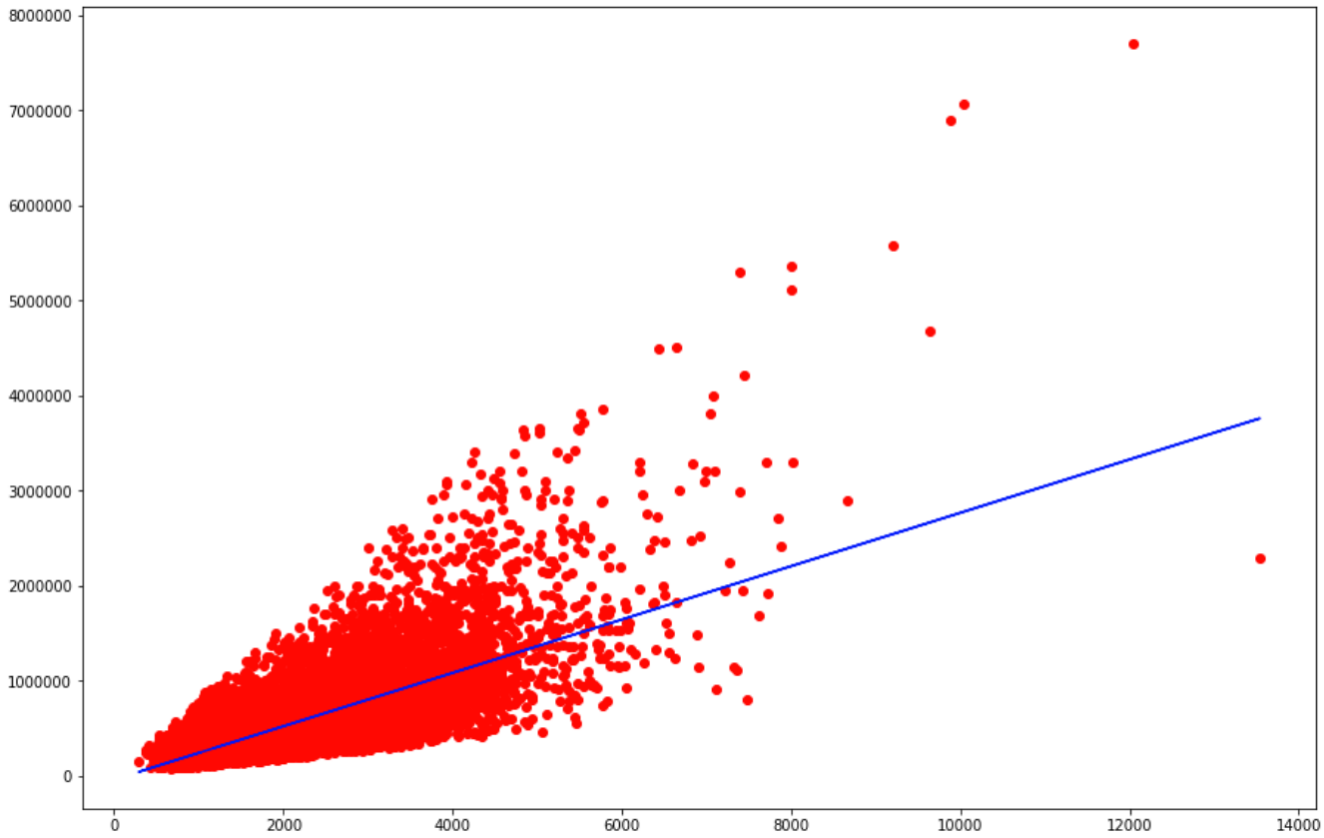
Lineer Regresyon'da bir doğru oluşturacağımız için bir bağımlı ve bir bağımsız değişken olmak üzere toplam 2 değişken üzerinde çalışacağız. Bu algoritma için evin büyüklüğü ile fiyatı arasındaki ilişkiyi ele alınmıştır. Bu ilişkiye göre evin fiyatını tahmin edilmiştir.

```
#kütüphane tanımlanması
from sklearn.linear_model import LinearRegression
linear_reg = LinearRegression()

x = data['sqft_living'].values.reshape(-1,1)
y = data['price'].values.reshape(-1,1)
```

```
#Doğrunun oluşturulması
linear_reg.fit(x,y)

b0 = linear_reg.intercept_
b1 = linear_reg.coef_
#Tahminler
y_head = linear_reg.predict(x)

print("B0 :",b0)
print("B1 :",b1)

#Tahmin Skoru
from sklearn.metrics import r2_score
print("R Square Values :",r2_score(y,y_head))

plt.figure(figsize=(15,10))
plt.scatter(x,y,color="red")
plt.plot(x,y_head,color="blue")
plt.show()
```

ÇIKTI

```
B0 : [-43580.74309447]
B1 : [[280.6235679]]
R Square Values : 0.4928532179037931
```



Yukarıda yaptığımız çalışmayla evin büyüklüğünün fiyat konusunu neredeyse yarı yarıya etkilediğini görüyoruz. Tabiki bir evi alırken sadece büyüklük değil bir çok parametre önemli. Bu sebepten ötürü lineer regresyon bizim iyi bir sonuç çıkarmamız için yetmiyor. Eğer elimizde tek bir bağımsız değişken varsa bu algoritmadan sağlıklı sonuç alabiliriz. Eğer birden çok bağımsız değişken varsa aşağıdaki algoritmalardan verimiz için en iyi olanı seçip onunla çalışmalıyız.

Multiple Linear Regresyon

Bir bağımlı değişken ile birden fazla bağımsız değişken arasındaki ilişkiyi inceleyen algoritmadır. Linear regresyon ile aynı mantığı kullanır. Sadece birden fazla bağımsız değişken içerir.

```
list_score = []
list_name = []
#kütüphane tanımlanması
from sklearn.linear_model import LinearRegression
multi_linear_reg = LinearRegression()

x = data.drop(['id','price','zipcode','lat','long','date'],axis=1).values
y = data['price'].values.reshape(-1,1)
multi_linear_reg.fit(x,y)

b0 = multi_linear_reg.intercept_
b1 = multi_linear_reg.coef_
#Tahminler
y_head = multi_linear_reg.predict(x)

print("B0 :",b0)
print("B1 :",b1)

#Tahmin Skoru
from sklearn.metrics import r2_score
print("R Square Values :",r2_score(y,y_head))
list_score.append(r2_score(y,y_head))
list_name.append("Multiple Linear Regresyon")
```

ÇIKTI

```
B0 : [6195319.95187286]
B1 : [[-3.93066524e+04  4.57450017e+04  1.09268480e+02 -1.60061929e-03
   2.68787845e+04  5.79071616e+05  4.32353616e+04  1.95103595e+04
   1.19721824e+05  5.15164697e+01  5.77520103e+01 -3.57015920e+03
   1.01595783e+01  2.48732761e+01 -5.50504053e-01]]
R Square Values : 0.6537318108687113
```

Multi Lineer Regresyon ile oldukça iyi bir oranla tahmin yapabildik. %65 oranında doğru fiyat tahmin edebiliyoruz.

Polinominal Lineer Regresyon

Bir bağımlı birden fazla bağımsız değişken arasında polinomal bir artış söz konusu ise bu algoritma bize en doğru sonucu verecektir.

```
#kütüphane tanımlaması
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
polynomial_reg = PolynomialFeatures(degree=4)
```

```
x = data.drop(['id','price','zipcode','lat','long','date'],axis=1).values
y = data['price'].values.reshape(-1,1)

polynomial_reg = PolynomialFeatures(degree =4)
x_polynomial = polynomial_reg.fit_transform(x)

reg = LinearRegression()
reg.fit(x_polynomial,y)
#Tahminler
y_head = reg.predict(x_polynomial)

#Tahmin Skoru
from sklearn.metrics import r2_score
print("R Square Values :",r2_score(y,y_head))
list_score.append(r2_score(y,y_head))
list_name.append("Polinominal Lineer Regresyon")
```

ÇIKTI

```
R Square Values : 0.8019069402461997
```

Polinominal Lineer Regresyon ile baya yüksek bir oranla tahmin yapabildik. Ev fiyatlarını bu algoritma ile %80 oranında doğru tahmin edebiliyoruz.