```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

import sklearn.model_selection
import sklearn.linear_model
import sklearn.metrics
```

## 2) Veri Toplamak

"Study Hours" Dataset: https://www.kaggle.com/datasets/himanshunakrani/student-study-hours

```
df = pd.read_csv("score.csv")
```

## 3) Veri İncelemek ve Görselleştirmek

```
df.head()
```

```
   Hours  Scores
0    2.5      21
1    5.1      47
2    3.2      27
3    8.5      75
4    3.5      30
```

```
df.shape
```

```
(25, 2)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Hours   25 non-null     float64
 1   Scores  25 non-null     int64
dtypes: float64(1), int64(1)
memory usage: 528.0 bytes
```

```
df.describe().T
```

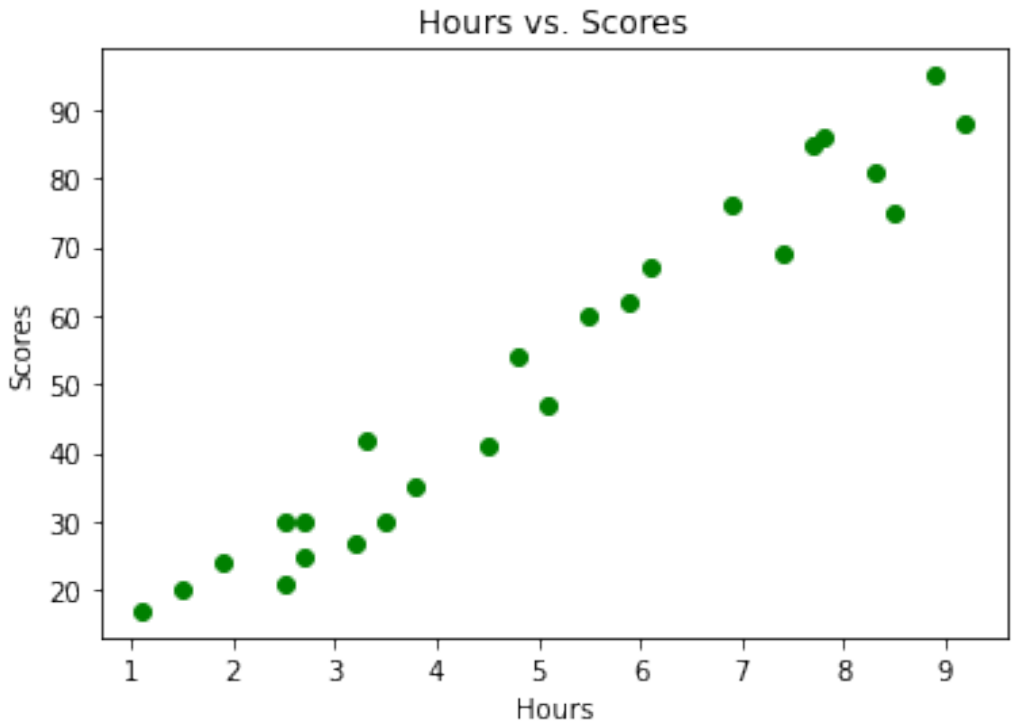
```
        count    mean        std   min   25%   50%   75%   max
Hours    25.0   5.012   2.525094   1.1   2.7   4.8   7.4   9.2
Scores   25.0  51.480  25.286887  17.0  30.0  47.0  75.0  95.0
```

```
plt.scatter(x=df["Hours"], y=df["Scores"], color="green")
```

```
plt.xlabel("Hours")
plt.ylabel("Scores")

plt.title("Hours vs. Scores")

plt.show()
```




## 4) Veriyi ML Modellerine Uygun Hale Getirmek

```
X = df["Hours"]
y = df["Scores"]

X_train, X_test, y_train, y_test =
sklearn.model_selection.train_test_split(X, y, train_size=0.8)

X_train = np.array(X_train)
y_train = np.array(y_train)
X_test = np.array(X_test)
y_test = np.array(y_test)

print((len(X_train)))
print((len(y_train)))

print((len(X_test)))
print((len(y_test)))
```

```
20
20
```

```
5
5
```

```
X_train = X_train.reshape(-1, 1)
y_train = y_train.reshape(-1, 1)
X_test = X_test.reshape(-1, 1)
y_test = y_test.reshape(-1, 1)
```

```
X_train.shape
```

```
(20, 1)
```

```
print(X_train.shape)
print(y_train.shape)
```

```
(20, 1)
(20, 1)
```

## 5) Model Seçimi ve Modelin Eğitilmesi

```
lin_model = sklearn.linear_model.LinearRegression()
```

```
lin_model.fit(X_train, y_train)
```

```
LinearRegression()
```

## 6) Modelin Optimize Edilmesi

```
predictions = lin_model.predict(X_test)
```

```
print(predictions)
```

```
[[78.29736504]
 [26.86917905]
 [49.18707109]
 [26.86917905]
 [91.88216889]]
```

```
for i in range(len(X_test)):
  print(f"{i} : Actual Value: {y_test[i]} - Predicted Value:
{predictions[i]}")
```

```
0 : Actual Value: [86] - Predicted Value: [78.29736504]
1 : Actual Value: [30] - Predicted Value: [26.86917905]
2 : Actual Value: [54] - Predicted Value: [49.18707109]
3 : Actual Value: [21] - Predicted Value: [26.86917905]
4 : Actual Value: [88] - Predicted Value: [91.88216889]
```

```
r2 = sklearn.metrics.r2_score(y_test, predictions)
mae = sklearn.metrics.mean_absolute_error(y_test, predictions)
mse = sklearn.metrics.mean_squared_error(y_test, predictions)
```

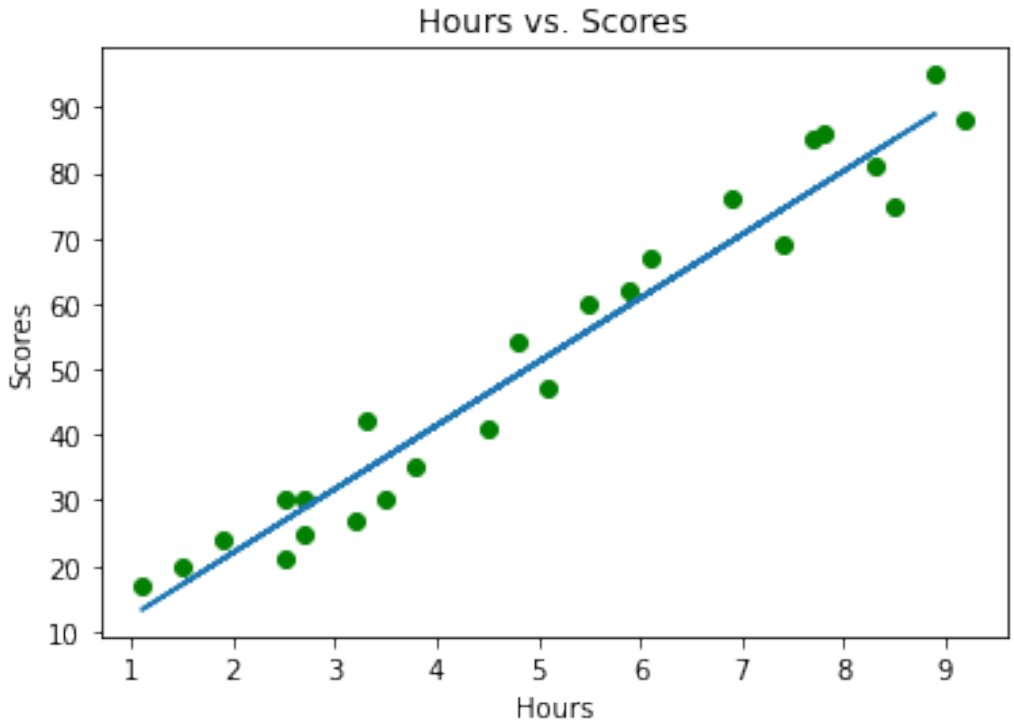
```
print(f"R2: {r2}")
print(f"MAE: {mae}")
print(f"MSE: {mse}")

X_predictions = lin_model.predict(X_train)

plt.scatter(x=df["Hours"], y=df["Scores"], color="green")
plt.plot(X_train, X_predictions)
plt.xlabel("Hours")
plt.ylabel("Scores")

plt.title("Hours vs. Scores")

plt.show()
```




## 7) Modelin Canlıya Alınması

```
lin_model.predict([[5]])

array([[51.12775735]])

X_predictions = lin_model.predict(X_train)

plt.plot(X_train, X_predictions)
plt.scatter(x=df["Hours"], y=df["Scores"], color="green")

plt.scatter(x=5, y=lin_model.predict([[5]]), s=100, color="red")
```

```
plt.xlabel("Hours")
plt.ylabel("Scores")

plt.title("Hours vs. Scores")

plt.show()
```