

```
In [2]: import pandas as pd
import numpy as np
```

```
In [3]: df=pd.read_csv("spark data.txt")
df
```

Out[3]:

|    | Hours | Scores |
|----|-------|--------|
| 0  | 2.5   | 21     |
| 1  | 5.1   | 47     |
| 2  | 3.2   | 27     |
| 3  | 8.5   | 75     |
| 4  | 3.5   | 30     |
| 5  | 1.5   | 20     |
| 6  | 9.2   | 88     |
| 7  | 5.5   | 60     |
| 8  | 8.3   | 81     |
| 9  | 2.7   | 25     |
| 10 | 7.7   | 85     |
| 11 | 5.9   | 62     |
| 12 | 4.5   | 41     |
| 13 | 3.3   | 42     |
| 14 | 1.1   | 17     |
| 15 | 8.9   | 95     |
| 16 | 2.5   | 30     |
| 17 | 1.9   | 24     |
| 18 | 6.1   | 67     |
| 19 | 7.4   | 69     |
| 20 | 2.7   | 30     |
| 21 | 4.8   | 54     |
| 22 | 3.8   | 35     |
| 23 | 6.9   | 76     |
| 24 | 7.8   | 86     |

```
In [4]: df.head()
```

Out[4]:

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

```
In [5]: 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
```

```
In [6]: df.isnull()
```

Out[6]:

|    | Hours | Scores |
|----|-------|--------|
| 0  | False | False  |
| 1  | False | False  |
| 2  | False | False  |
| 3  | False | False  |
| 4  | False | False  |
| 5  | False | False  |
| 6  | False | False  |
| 7  | False | False  |
| 8  | False | False  |
| 9  | False | False  |
| 10 | False | False  |
| 11 | False | False  |
| 12 | False | False  |
| 13 | False | False  |
| 14 | False | False  |
| 15 | False | False  |
| 16 | False | False  |
| 17 | False | False  |
| 18 | False | False  |
| 19 | False | False  |
| 20 | False | False  |
| 21 | False | False  |
| 22 | False | False  |
| 23 | False | False  |
| 24 | False | False  |

```
In [7]: df.describe()
```

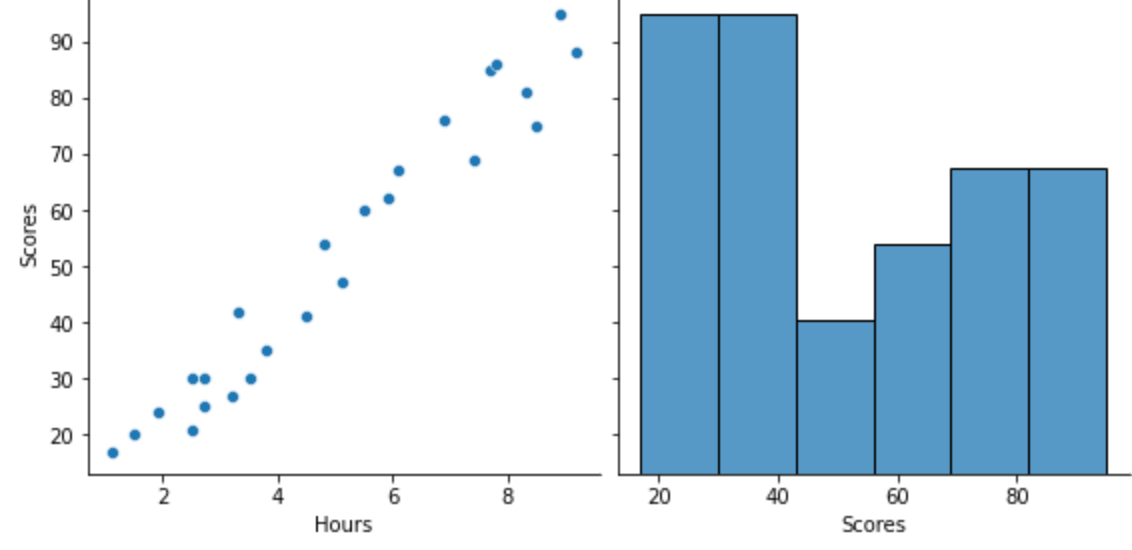
Out[7]:

|       | Hours     | Scores    |
|-------|-----------|-----------|
| count | 25.000000 | 25.000000 |
| mean  | 5.012000  | 51.480000 |
| std   | 2.525094  | 25.286887 |
| min   | 1.100000  | 17.000000 |
| 25%   | 2.700000  | 30.000000 |
| 50%   | 4.800000  | 47.000000 |
| 75%   | 7.400000  | 75.000000 |
| max   | 9.200000  | 95.000000 |

```
In [8]: # Import matplotlib and seaborn libraries to visualize the data
import matplotlib.pyplot as plt
import seaborn as sns

# Using pairplot we'll visualize the data for correlation
sns.pairplot(df, x_vars='Hours', 'Scores'],
              y_vars='Scores', size=4, aspect=1, kind='scatter')
plt.show()
```

C:\Anaconda\lib\site-packages\seaborn\axisgrid.py:1969: UserWarning: The `size` parameter has been renamed to `height`; please update your code.



```
In [9]: df.corr()
```

Out[9]:

|        | Hours    | Scores   |
|--------|----------|----------|
| Hours  | 1.000000 | 0.976191 |
| Scores | 0.976191 | 1.000000 |

```
In [10]: x = df.iloc[:, :-1].values #get a copy of dataset exclude last column
y = df.iloc[:, 1].values
```

```
In [11]: print(x)
```

```
[[2.5]
 [5.1]
 [3.2]
 [8.5]
 [3.5]
 [1.5]
 [9.2]
 [5.5]
 [8.3]
 [2.7]
 [7.7]
 [5.9]
 [4.5]
 [3.3]
 [1.1]
 [8.9]
 [2.5]
 [1.9]
 [6.1]
 [7.4]
 [2.7]
 [4.8]
 [3.8]
 [6.9]
 [7.8]]
```

```
In [12]: print(y)
```

```
[21 47 27 75 30 20 88 60 81 25 85 62 41 42 17 95 30 24 67 69 30 54 35 76
 86]
```

```
In [13]: # Splitting the dataset into the Training set 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=1/3, random_state=0)
```

```
In [14]: # Fitting Simple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[14]: LinearRegression()

```
In [15]: #retrieve the intercept:
print(regressor.intercept_)
```

```
2.5069547569547623
```

```
In [16]: #retrieving the slope (coefficient of x):
print(regressor.coef_)
```

```
[9.69062469]
```

```
In [17]: #making predictions
y_pred = regressor.predict(X_test)
```

```
In [18]: print(y_pred)
```

```
[17.04289179 33.51695377 74.21757747 26.73351648 59.68164043 39.33132858
 20.91914167 78.09382734 69.37226512]
```

```
In [19]: data = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
data
```

Out[19]:

|   | Actual | Predicted |
|---|--------|-----------|
| 0 | 20     | 17.042892 |
| 1 | 27     | 33.516954 |
| 2 | 69     | 74.217577 |
| 3 | 30     | 26.733516 |
| 4 | 62     | 59.681640 |
| 5 | 35     | 39.331329 |
| 6 | 24     | 20.919142 |
| 7 | 86     | 78.093827 |
| 8 | 76     | 69.372265 |

```
In [20]: from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

Mean Absolute Error: 4.691397441397438
Mean Squared Error: 25.463280738222547
Root Mean Squared Error: 5.046115410711743
```

```
In [ ]: '''#You can see that the value of root mean squared error is 4.64,
which is less than 10% of the mean value of the percentages of all the students i.e. 51.48.
This means that our algorithm did a decent job
```