Prediction Using Linear Regression with Python Scikit Learn

In this task we will predict the scores of a student based on the number of hours they studied using simple linear regression technique.

Author: Anagha Sabu

In [1]:

```
#importing necessary libraries and packages
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [2]:

```
#Reading the data
df = pd.read_excel('SparksTSK1.xlsx')
```

In [3]:

```
#Viewing first 5 rows
df.head()
```

Out[3]:

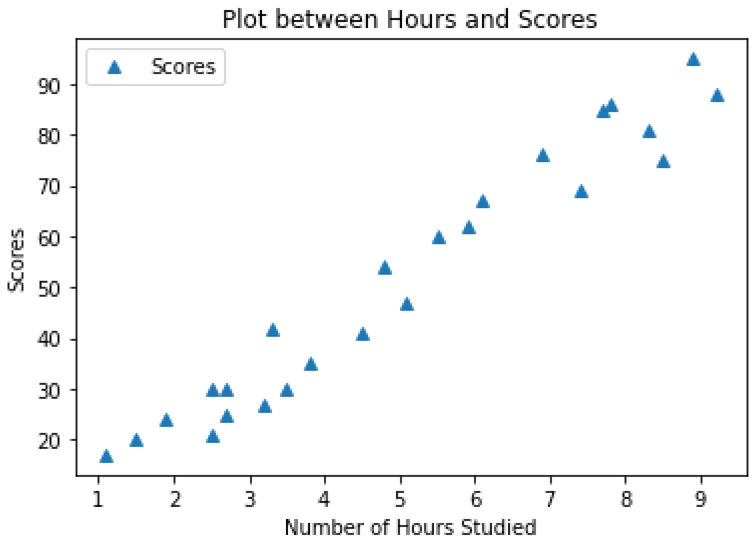
	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30

In [4]:

```
#Plotting the score distribution
df.plot(x='Hours', y='Scores', style ='^')
plt.title('Plot between Hours and Scores')
plt.xlabel('Number of Hours Studied')
plt.ylabel('Scores')
```

Out[4]:

```
Text(0, 0.5, 'Scores')
```



Here, we can see a positive linear relationship between number of hours studied and scores secured

Data Preparation

In [5]:

```
# defining X as 'Hours' column and y as 'Scores'
X = df.iloc[:,:-1].values
y = df.iloc[:,1].values
```

In [6]:

```
#splitting the dataset into training and testing
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=1)
```

Training a linear regression model in scikit-learn

In [7]:

```
#Algorithm training
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[7]:

```
LinearRegression()
```

In [8]:

```
regressor = LinearRegression().fit(X, y)
regressor
```

Out[8]:

```
LinearRegression()
```

In [9]:

```
regressor_accuracy = regressor.score(X_test, y_test)
regressor_accuracy
```

Out[9]:

```
0.9044315672829022
```

In [10]:

```
print('Linear Model Coefficient (m): ', regressor.coef_)
print('Linear Model Coefficient (c): ', regressor.intercept_)
```

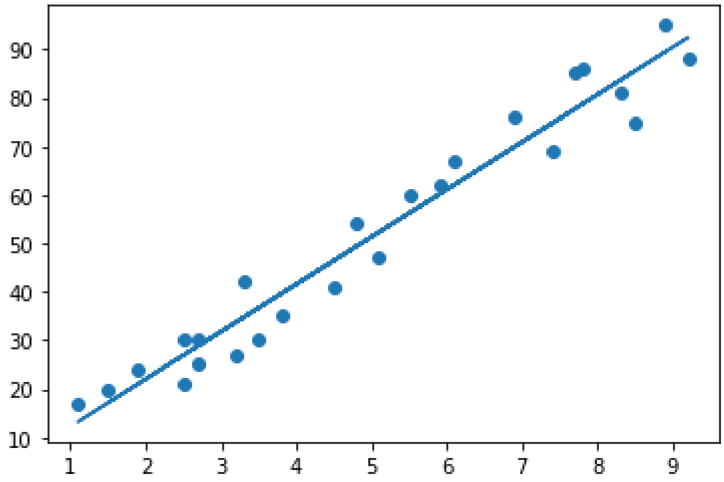
```
Linear Model Coefficient (m):  [9.77580339]
Linear Model Coefficient (c):  2.48367340537321
```

Evaluating trained model performance

In [11]:

```
# Plotting the regression line
line = regressor.coef_*X+regressor.intercept_
print(line[:5])
#test data
plt.scatter(X,y)
plt.plot(X, line);
#plt.show()
```

```
[[26.92318188]
 [52.3402707 ]
 [33.76624426]
 [85.57800223]
 [36.69898527]]
```



Prediction on test values

In [12]:

```
print(X_test)
y_cap = regressor.predict(X_test)
```

```
[[1.1]
 [3.3]
 [1.9]
 [8.5]
 [4.8]]
```

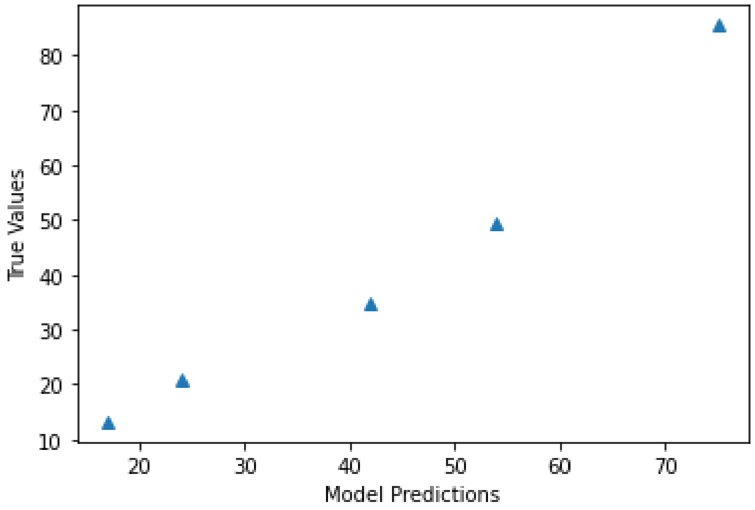
In [13]:

```
# Comparing Actual vs Predicted
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_cap})
print(df)
plt.plot(y_test, y_cap, "^")
plt.xlabel('Model Predictions')
plt.ylabel('True Values')
```

```
   Actual  Predicted
0      17  13.237057
1      42  34.743825
2      24  21.057700
3      75  85.578002
4      54  49.407530
```

Out[13]:

```
Text(0, 0.5, 'True Values')
```



In [14]:

```
X_new = [[4.5]]
Y_new = regressor.predict(X_new).reshape(1,-1)
Y_new
print("No of Hours = {}".format(X_new[0]))
print("Predicted Score = {}".format(Y_new[0]))
```

```
No of Hours = [4.5]
Predicted Score = [46.47478866]
```

In [ ]:

In [ ]: