

SURBHI DESAI**Data Science & Business Analytics Tasks****GRIPJUNE2021****Prediction using Supervised ML**

In this task, I have to predict the percentage of an student based on the number of study hours

In [1]:

```
#Importing all Libraries required in this notebook
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
```

Step 1 : Reading Data from the source

In [53]:

```
# This is Linear Regression using two Variables

s_data = pd.read_csv("C:\\Users\\surbhi\\Desktop\\surbhi_imp\\price.csv")
print("Data imported successfully")
s_data.head(10)
```

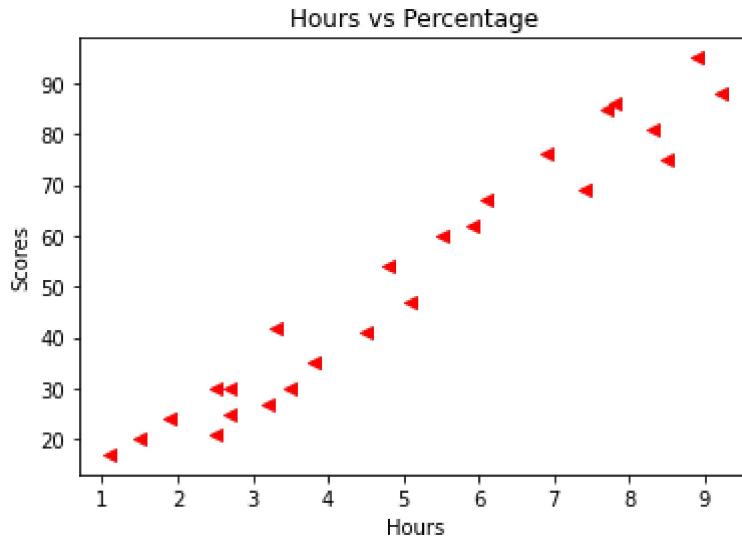
Data imported successfully

Out[53]:

| | 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 |

Step 2: Input Data Visualization

```
In [54]: plt.xlabel("Hours")
plt.ylabel("Scores")
plt.scatter(s_data.Hours, s_data.Scores, color= "red", marker = "<")
plt.title('Hours vs Percentage')
plt.show()
```



Step 3 : Data PreProcessing

In this step , we divide the data into "attributes"(Inputs) and "Labels" (Outputs)

```
In [36]: A = s_data.iloc[:, :-1].values
B = s_data.iloc[:, 1].values
```

Step 4 : Training the Model by splitting into training and testing sets and training the Algorithm

```
In [6]: from sklearn.model_selection import train_test_split
A_train, A_test, B_train, B_test = train_test_split(A, B,
                                                    test_size=0.2, random_state=0)
```

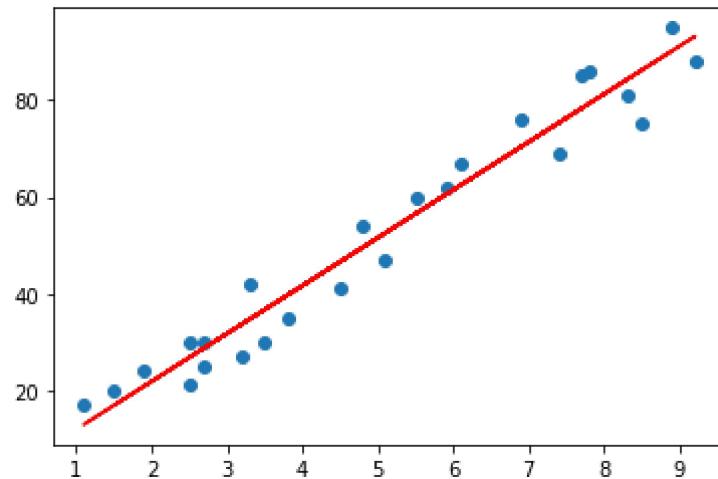
```
In [55]: from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(A_train.reshape(-1,1) , B_train)

print("Training complete.")
```

Training complete.

Step 5 : Plotting the line of Regression

```
In [56]: # Plotting the regression line  
line = regressor.coef_*A+regressor.intercept_  
  
# Plotting for the test data  
plt.scatter(A, B)  
plt.plot(A, line, color='red');  
plt.show()
```



Step 6 : Making Predictions

Now let's test our test-set data using our trained algorithm

```
In [57]: print(A_test)  
B_pred = regressor.predict(A_test)
```

```
[[1.5]  
[3.2]  
[7.4]  
[2.5]  
[5.9]]
```

Step 7 : Comparing Actual result to the Predicted Model Result

In [58]: # Comparing Actual vs Predicted
`df = pd.DataFrame({'Actual': B_test, 'Predicted': B_pred})
df`

Out[58]:

| | Actual | Predicted |
|---|--------|-----------|
| 0 | 20 | 16.884145 |
| 1 | 27 | 33.732261 |
| 2 | 69 | 75.357018 |
| 3 | 30 | 26.794801 |
| 4 | 62 | 60.491033 |

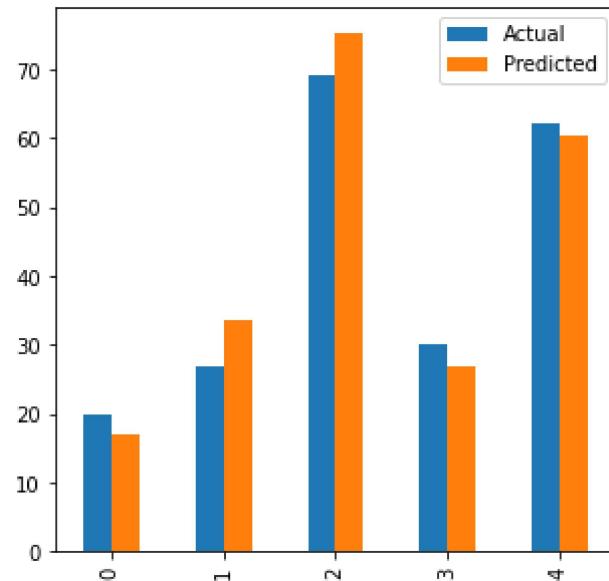
In [59]: # Estimating training and test score

```
print("Training Score:" , regressor.score(A_train,B_train))
print("Test Score:" , regressor.score(A_test,B_test))
```

Training Score: 0.9515510725211552
Test Score: 0.9454906892105356

In [60]: #Plotting Bar Graph between the actual and predicted values

```
df.plot(kind='bar', figsize=(5,5))
plt.show()
```



In [61]: # You can also test with your own data

```
hours = 9.25
own_pred = regressor.predict([[hours]])
print("No of Hours = {}".format(hours))
print("Predicted Score = {}".format(own_pred[0]))
```

No of Hours = 9.25
Predicted Score = 93.69173248737538

Thank you!

In []: