GRIP @ THE SPARKS FOUNDATION

TASK 1 : PREDICTION USING SUPERVISED MACHINE LEARNING

NAME : GADHA M KURUP

Aim : Performing Explanatory Data Analysis on the dataset under consideration to predict the percentage of the marks of the students based on the number of hours they studied.

Importing all required libraries

In [7]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

Reading data (from remote link)

In [8]:
```
url = "http://bit.ly/w-data"
s_data = pd.read_csv(url)
print("Data imported successfully")

s_data.head(10)
```

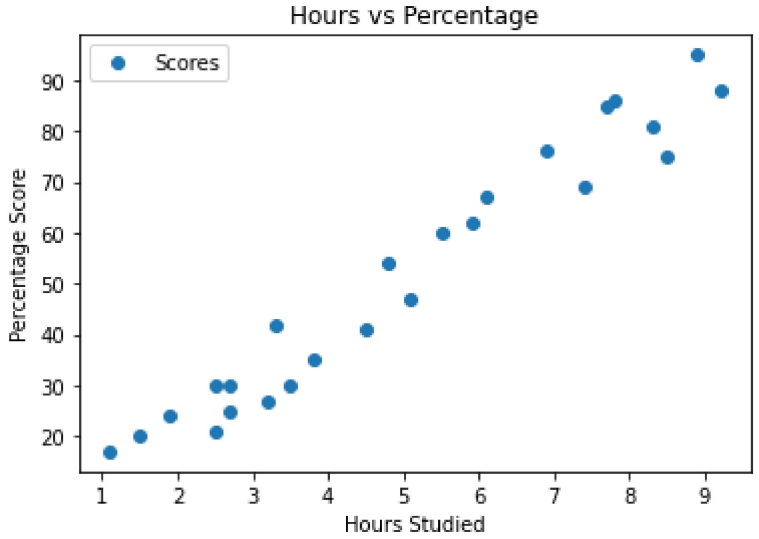
```
Data imported successfully
```

Out[8]:

	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

Plotting the distribution of scores

In [9]:
```
s_data.plot(x='Hours', y='Scores', style='o')
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```



Interpretation: We can see that there is a positive linear relation between the number of hours studied and percentage of score from the plot.

Preparing the data

The next step is to divide the data into "attributes" (inputs) and "labels" (outputs).

In [10]:
```
X = s_data.iloc[:, :-1].values
y = s_data.iloc[:, 1].values
```

The next step is to split this data into training and test sets. We'll do this by using Scikit-Learn's built-in train_test_split() method:

In [11]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                            test_size=0.2, random_state=0)
```

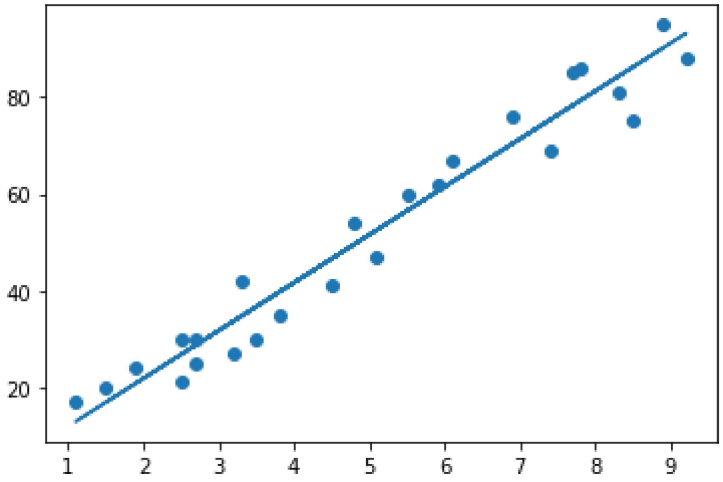
Training the Algorithm

In [13]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

print("Training complete.")
```

```
Training complete.
```

Plotting the regression line

In [15]:
```
line = regressor.coef_*X+regressor.intercept_

# Plotting for the test data
plt.scatter(X, y)
plt.plot(X, line);
plt.show()
```



Making Predictions

Now that we have trained our algorithm, it's time to make some predictions.

In [0]:
```
print(X_test) # Testing data - In Hours

y_pred = regressor.predict(X_test) # Predicting the scores
```

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

Comparing Actual vs Predicted Models

In [0]:
```
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df
```

Out[9]:

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

In [17]:
```
#Estimating training and testing the score

print("Training Score: ",regressor.score(X_train,y_train))
print("Test Score: ",regressor.score(X_test,y_test))
```

```
Training Score:  0.9515510725211552
Test Score:  0.9454906892105356
```

Evaluating the model

The final step is to evaluate the performance of algorithm. This step is particularly important to compare how well different algorithms perform on a particular dataset. For simplicity here, we have chosen the mean square error. There are many such metrics.

In [0]:
```
from sklearn import metrics
print('Mean Absolute Error:',
      metrics.mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Error: 4.183859899002982
```

Conclusion

Successfully completed prediction using Supervised Machine Learning and evaluated models performance.