Grip :- The Spark Foundation

Simple Linear Regression

In this regression task we will predict the percentage of marks that a student is expected to score based upon the number of hours they studied. This is a simple linear regression task as it involves just two variables.

Author:-

Gaurav Balavant Suryavanshi

In [1]:

```
# import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
#data=pd.read_clipboard()
#data
data = pd.read_csv('percentage1.csv')
data
```

Out[2]:

	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	5
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 [3]:

```
d_1=data.head(10)
```

In [4]:

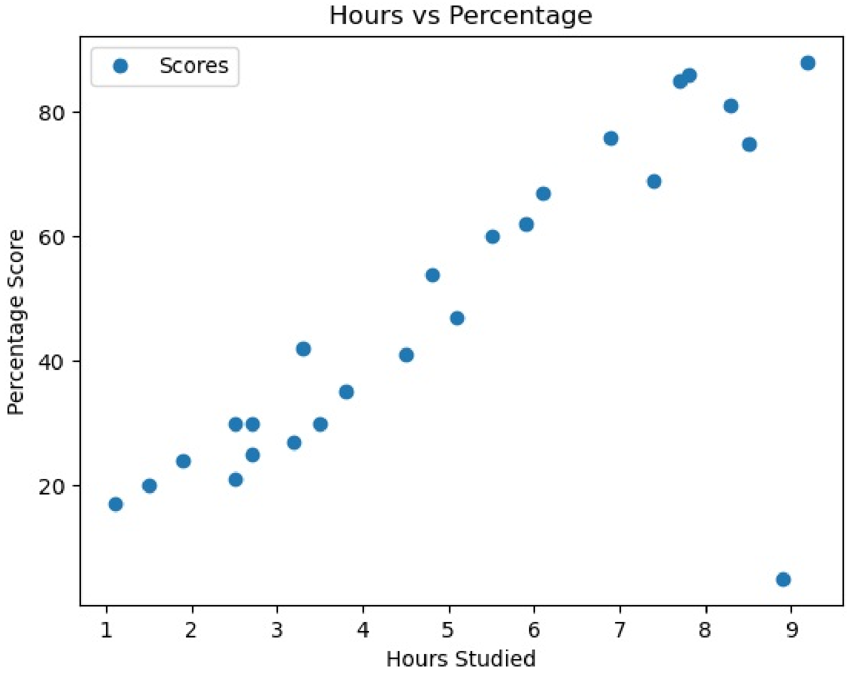
```
d_1
```

Out[4]:

	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

In [5]:

```
# plotting the distribution of score
data.plot(x='Hours' , y='Scores', style='o')
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```



From the graph above, we can clearly see that there is a positive linear relation between the number of hours studied and percentage of score.

Preparing the Data

In [6]:

```
x= data.iloc[:,:-1].values
```

In [7]:

```
y=data.iloc[:,1].values
```

In [25]:

```
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 [26]:

```
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(x_train,y_train)

print("Training complete")
```

```
Training complete
```

In [27]:

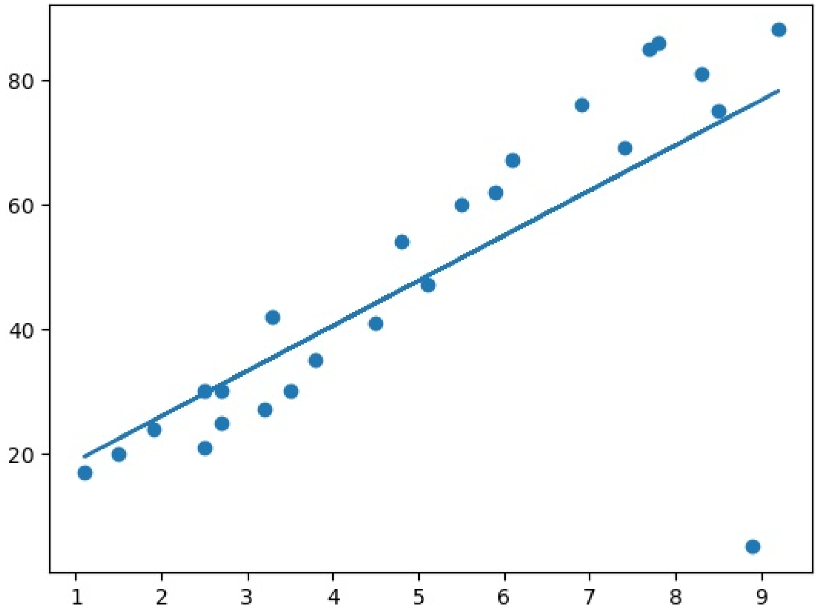
```
# plotting the regression line
line=regressor.coef_*x + regressor.intercept_
```

In [28]:

```
plt.scatter(x,y)
```

In [28]:

```
plt.scatter(x,y)
plt.plot(x,line);
plt.show()
```



Making Predictions

In [29]:

```
print(x_test)
y_pred=regressor.predict(x_test)
y_pred
```

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

Out[29]:

```
array([22.35400751, 34.67036773, 65.0990224 , 29.59892529, 54.23164573])
```

In [30]:

```
y_test
```

Out[30]:

```
array([20, 27, 69, 30, 62], dtype=int64)
```

In [31]:

```
#df =pd.DataFrame ({'Actual': Y_test, 'Predicted': Y_pred})
#df

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

Out[31]:

	Actual	Predicted
0	20	22.354008
1	27	34.670368
2	69	65.099022
3	30	29.598925
4	62	54.231646

In [43]:

```
regressor.intercept_
```

Out[43]:

```
11.486630842936677
```

In [44]:

```
regressor.get_params()
```

Out[44]:

```
{'copy_X': True, 'fit_intercept': True, 'n_jobs': None, 'positive': False}
```

In [52]:

```
# You can also test with your own data
Hr=np.array ([9.25]).reshape(1,1)
print("No. of hours : ",Hr)
print("Predicted Score: ",regressor.predict(Hr))
```

```
No. of hours :  [[9.25]]
Predicted Score:  [78.50212029]
```

In [54]:

```
from sklearn.metrics import  r2_score,  mean_absolute_error
b=r2_score(y_test,y_pred)
b
```

Out[54]:

```
0.9292844197063785
```

Evaluating the model

In [51]:

```
# mean absolute error
print('Mean Absolute Error:',
     mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Error: 4.418956364107212
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js