Author : Ganesh Dalal

GRIP @ The Sparks Foundation

Task 1: Prediction Using Supervised Machine Learning

This is a simple linear regression task involving two variables (x:independent variable,y:dependent variable).Linear Regression is used to study the statistical relation between the dependent and independent variable, in other words ittells us how a dependent variable variates with respect to the independent variable.

In [1]:
```
#Import Required Libraries
```

In [3]:
```
import pandas as pd
```

In [4]:
```
import numpy as np
```

In [6]:
```
import sklearn
```

In [7]:
```
import matplotlib.pyplot as plt
```

In [8]:
```
#Read The Dataset
```

In [11]:
```
df = pd.read_csv(r'C:\Users\Lenovo\Downloads\DATASET\Student_Scores.csv')
```

In [12]:
```
df #we can also use df.head() to view the top 5 datasets and similarily df.tai
l() to view the bottom 5 datasets is big
```

Out[12]:

	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 [13]:
```
#Inspecting the Dataset
```

In [14]:
```
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
```

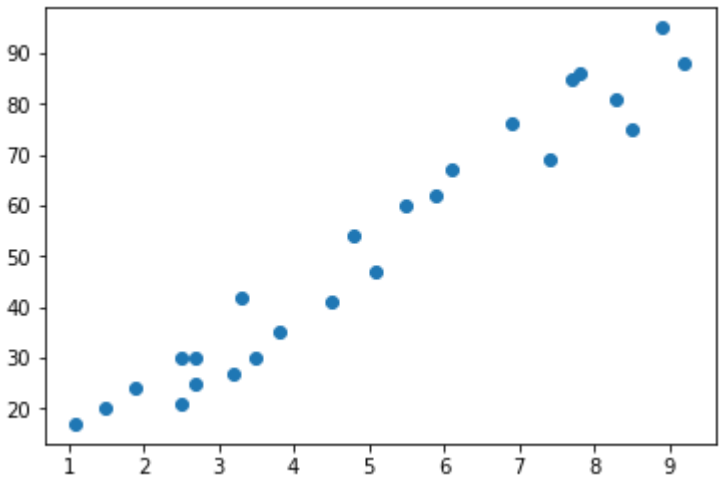
In [15]:
```
df.describe()   #shows the statistical description of the numerical dataset
```

Out[15]:

	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

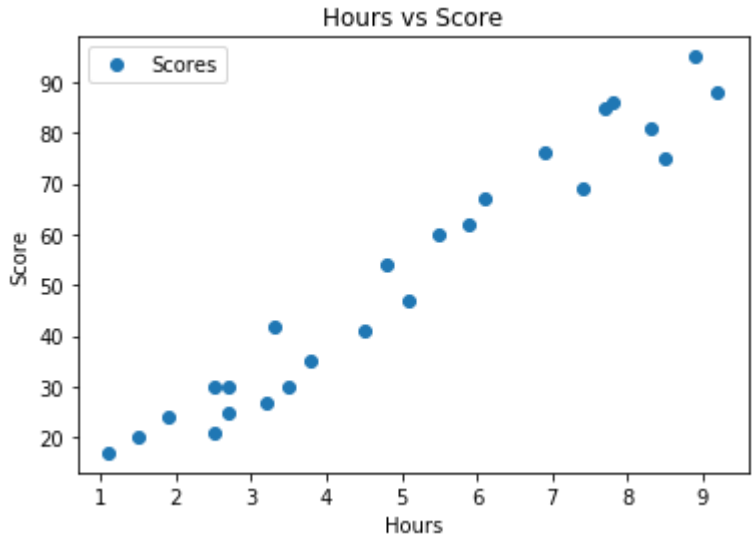
In [22]:
```
#Check Linearity
plt.scatter(df['Hours'], df['Scores'])
```

Out[22]:
```
<matplotlib.collections.PathCollection at 0x13c1e0f0>
```



In [28]:
```
#plotting the distribution of scores
df.plot(x='Hours',y='Scores',style='o')
plt.title('Hours vs Score')
plt.xlabel('Hours')
plt.ylabel('Score')
```

Out[28]: Text(0, 0.5, 'Score')



In [30]:
```
#Define the X(Independent) and Y(Dependent) variable
#now we have to divide the data into x and y variable 'x' is independent varia
bles and 'y' is a dependent variable
```

In [31]:
```
y=df['Scores']
```

In [32]:
```
x=df[['Hours']]
```

In [33]:
```
#Train_Test_Spliting the Dataset
```

In [34]:
```
#Here we split the whole dataset into training and test datasets by using the
train_test_split() method which is available in sklearn library
```

In [35]:
```
from sklearn.model_selection import train_test_split
```

In [41]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=
2569)
```

In [42]:
```
#Fitting The Model
#Here we train the model with the training dataset
```

In [43]:
```
from sklearn.linear_model import LinearRegression
```

In [44]:
```
model= LinearRegression()
```

In [45]:
```
model.fit(x_train,y_train)
```

Out[45]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [46]:
```
#Equation of Line of Linear Regression( Bestfit Line)
```

In [47]:
```
model.coef_
```

Out[47]:
```
array([9.82659749])
```

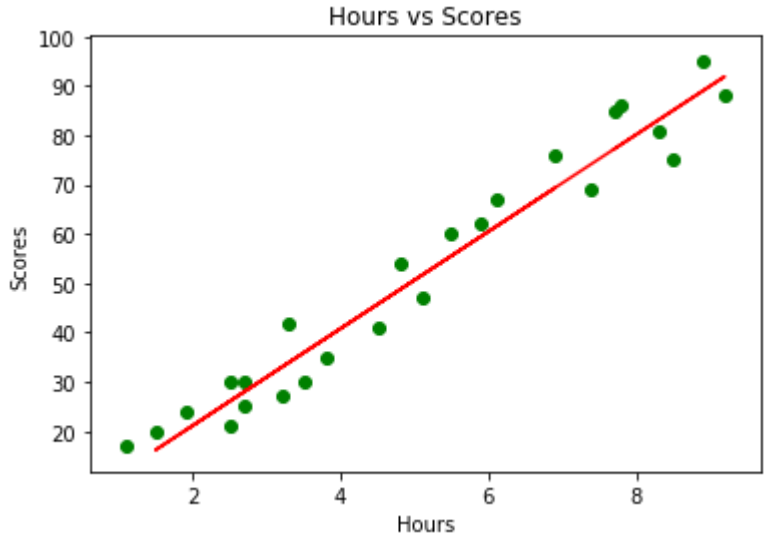
In [48]:
```
model.intercept_
```

Out[48]:
```
1.5005392900893142
```

In [49]:
```
#The Equation of the regression line
y_pred=model.coef_*(x_test)+model.intercept_
```

In [50]:
```
#Plotting the Data points and the line of regression
```

In [51]:
```
plt.plot(x_test,y_pred,color='red')
plt.scatter(x,y,color='green')
plt.xlabel("Hours")
plt.ylabel('Scores')
plt.title('Hours vs Scores')
plt.show()
```



In [52]:
```
#Making Predictions
```

In [53]:
```
print([x_test])
y_pred=model.predict(x_test)  #Prediction for the test dataset
```

```
[    Hours
17    1.9
23    6.9
5     1.5
6     9.2
10    7.7]
```

In [54]:
```
y_pred
```

Out[54]:
```
array([20.17107453, 69.304062  , 16.24043553, 91.90523624, 77.16534   ])
```

In [55]:
```
y_test #actual test dataset
```

Out[55]:
```
17    24
23    76
5     20
6     88
10    85
Name: Scores, dtype: int64
```

In [56]:
```
Actual_vs_Predicted=pd.DataFrame({'Actual':y_test,'Predicted':y_pred})
```

In [57]:
```
Actual_vs_Predicted
```

Out[57]:

	Actual	Predicted
17	24	20.171075
23	76	69.304062
5	20	16.240436
6	88	91.905236
10	85	77.165340

In [58]:
```
#predicted score if a student studies for 9.25 hours/day
score=model.predict([[9.25]])
print("The Predicted Score for the student if he studies 9.25 hrs/day is", flo
at(score))
```

```
The Predicted Score for the student if he studies 9.25 hrs/day is 92.39656611
306054
```

Prediction = 92.3965661

In [59]:
```
#Model Evaluation
```

In [62]:
```
from sklearn.metrics import  mean_absolute_error, mean_squared_error, r2_score
```

In [64]:
```
Mean_Absolute_Error=mean_absolute_error(y_test,y_pred)
print('Mean Absolute Error:',Mean_Absolute_Error)
```

```
Mean Absolute Error: 5.20486483584594
```

In [65]:
```
Mean_Squared_Error=mean_squared_error(y_test,y_pred)
print('Mean Squared Error:',Mean_Squared_Error)
```

```
Mean Squared Error: 30.052669675421168
```

In [66]:
```
R2_Score=r2_score(y_pred,y_test)
print('The R-2 Score of the model is:',R2_Score)
```

```
The R-2 Score of the model is: 0.9685197229910099
```

R-2 gives the score of model fit and in this case we have R-2=0.9685197 which is actually a great score for this model

Conclusion

Hence we successfully have trained the linear regression model to predict the score of the student who studies for 9.25 hrs/day. The predicted score by the model is 92.3965661%.