GRIP - The Spark Foundation

Data Science & Business Analytics Intern

Author: CHANDHINII.V

Task 1: Prediction using Supervised ML

Predict the percentage of an student based on the no. of study hours. This is a simple linear regression task as it involves just 2 variables.
What will be predicted score if a student studies for 9.25 hrs/ day?

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
from sklearn import model_selection
from sklearn import linear_model
```

In [5]:
```
df=pd.read_csv(r"C:\Users\USER\Documents\Book1.csv")
print("Load the data")
df
```

```
Load the data
```

Out[5]:

	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 [6]:
```
df.shape
```

Out[6]:
```
(25, 2)
```

In [8]:
```
df.columns
```

Out[8]:
```
Index(['Hours', 'Scores'], dtype='object')
```

In [9]:
```
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 [10]:
```
df.describe()
```

Out[10]:

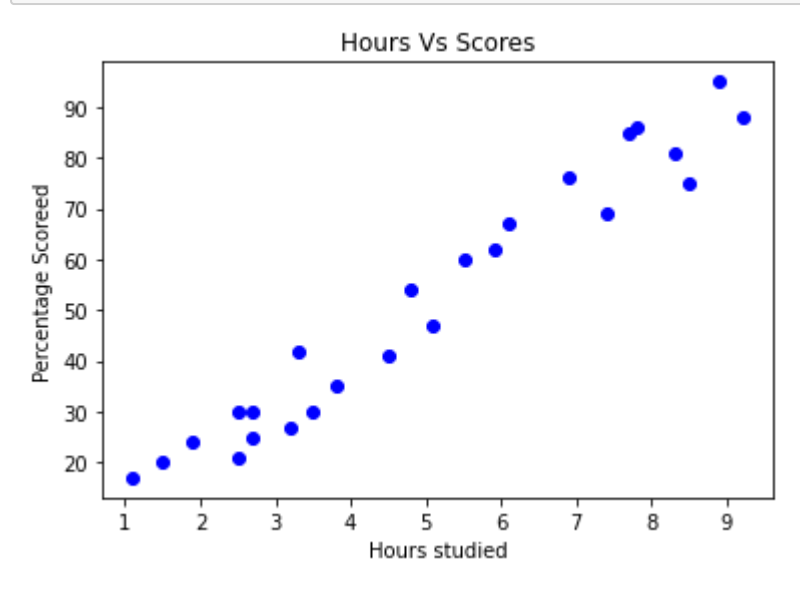
	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000
25%	2.700000	30.000000
50%	4.800000	47.000000
75%	7.400000	75.000000
max	9.200000	95.000000

In [11]:
```
df.groupby(['Hours'])['Scores'].mean()
```

Out[11]:
```
Hours
1.1    17.0
1.5    20.0
1.9    24.0
2.5    25.5
2.7    27.5
3.2    27.0
3.3    42.0
3.5    30.0
3.8    35.0
4.5    41.0
4.8    54.0
5.1    47.0
5.5    60.0
5.9    62.0
6.1    67.0
6.9    76.0
7.4    69.0
7.7    85.0
7.8    86.0
8.3    81.0
8.5    75.0
8.9    95.0
9.2    88.0
Name: Scores, dtype: float64
```

Exploring the dataset

In [12]:
```
plt.scatter(df['Hours'], df['Scores'], color='Blue',marker='o')
plt.title("Hours Vs Scores")
plt.xlabel("Hours studied")
plt.ylabel("Percentage Scoreed")
plt.show()
```



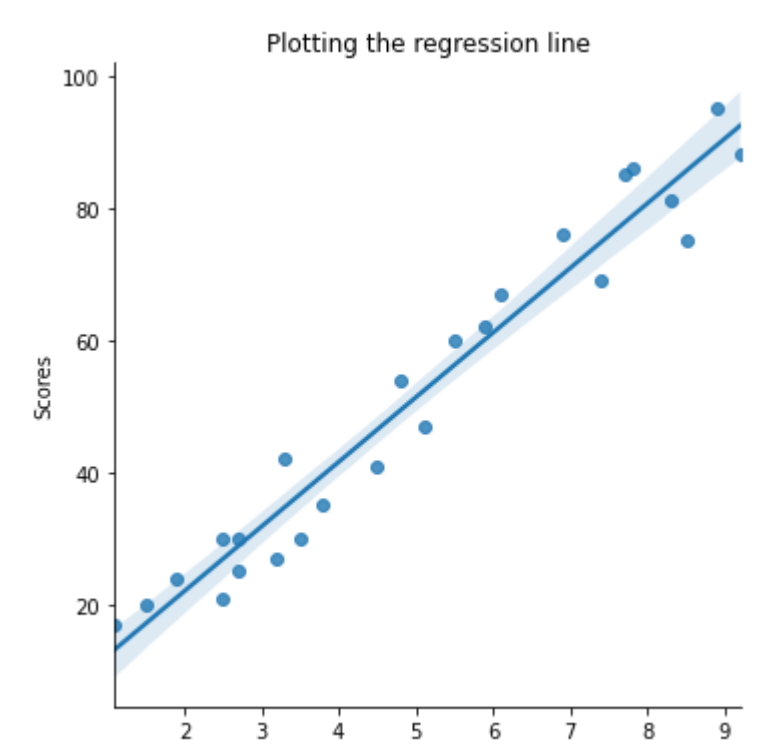
In [13]:
```
df.corr()
```

Out[13]:

	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

In [14]:
```
sns.lmplot(x="Hours",y="Scores", data=df)
plt.title("Plotting the regression line")
#sns.regplot(x="Hours", y="Scores", data=df)
```

Out[14]:
```
Text(0.5, 1.0, 'Plotting the regression line')
```



Dividing the data into attributes(inputs) and labels (outputs)

In [15]:
```
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values
```

In [16]:
```
X
```

Out[16]:
```
array([[2.5],
       [5.1],
       [3.2],
       [8.5],
       [3.5],
       [1.5],
       [9.2],
       [5.5],
       [8.3],
       [2.7],
       [7.7],
       [5.9],
       [4.5],
       [3.3],
       [1.1],
       [8.9],
       [2.5],
       [1.9],
       [6.1],
       [7.4],
       [2.7],
       [4.8],
       [3.8],
       [6.9],
       [7.8]])
```

In [17]:
```
y
```

Out[17]:
```
array([21, 47, 27, 75, 30, 20, 88, 60, 81, 25, 85, 62, 41, 42, 17, 95, 30,
       24, 67, 69, 30, 54, 35, 76, 86], dtype=int64)
```

Splitting the dataset into the Training set and Test set

In [18]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state = 0)
```

Training the Simple Linear Regression model on the Training set

In [19]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[19]:
```
LinearRegression()
```

Predicting the Test set results¶

In [20]:
```
y_pred = regressor.predict(X_test)
```

In [21]:
```
y_pred
```

Out[21]:
```
array([17.04289179, 33.51695377, 74.21757747, 26.73351648, 59.68164043,
       39.33132858, 20.91914167, 78.09382734, 69.37226512])
```

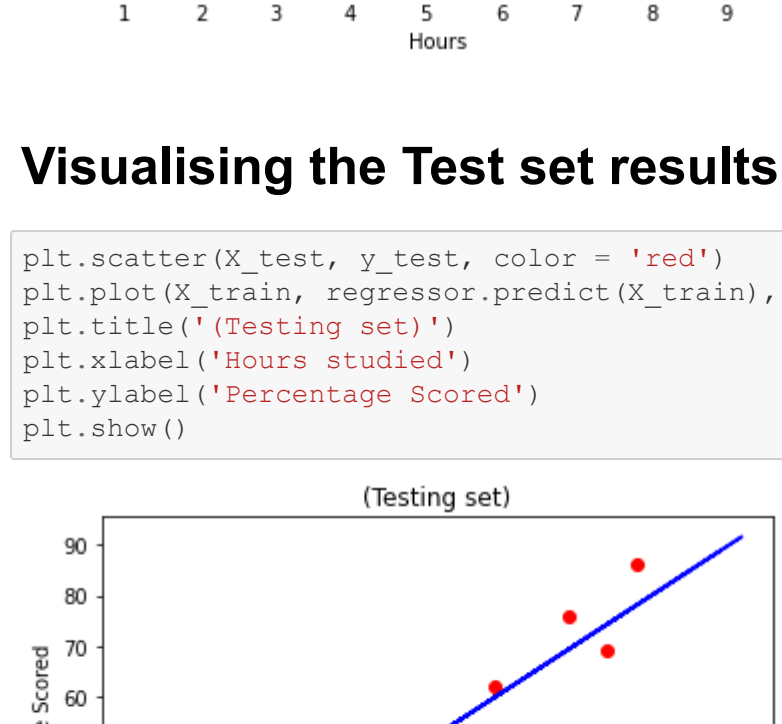
In [22]:
```
# Comparing Actual vs Predicted
df1 = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df1
```

Out[22]:

	Actual	Predicted
0	20	17.042892
1	27	33.516954
2	69	74.217577
3	30	26.733516
4	62	59.681640
5	35	39.331329
6	24	20.919142
7	86	78.093827
8	76	69.372265

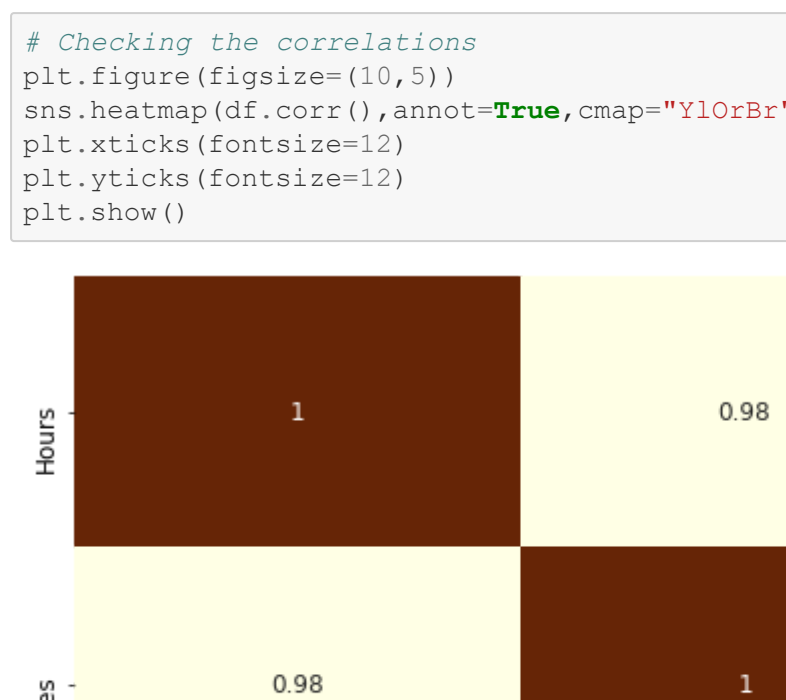
Visualising the Training set results

In [23]:
```
# PLotting the training set
plt.scatter(X_train,y_train, color='red')
plt.plot(X_train,regressor.predict(X_train),color='blue')
plt.title('(Trainig set)')
plt.xlabel('Hours')
plt.ylabel('Scores')
plt.show()
```

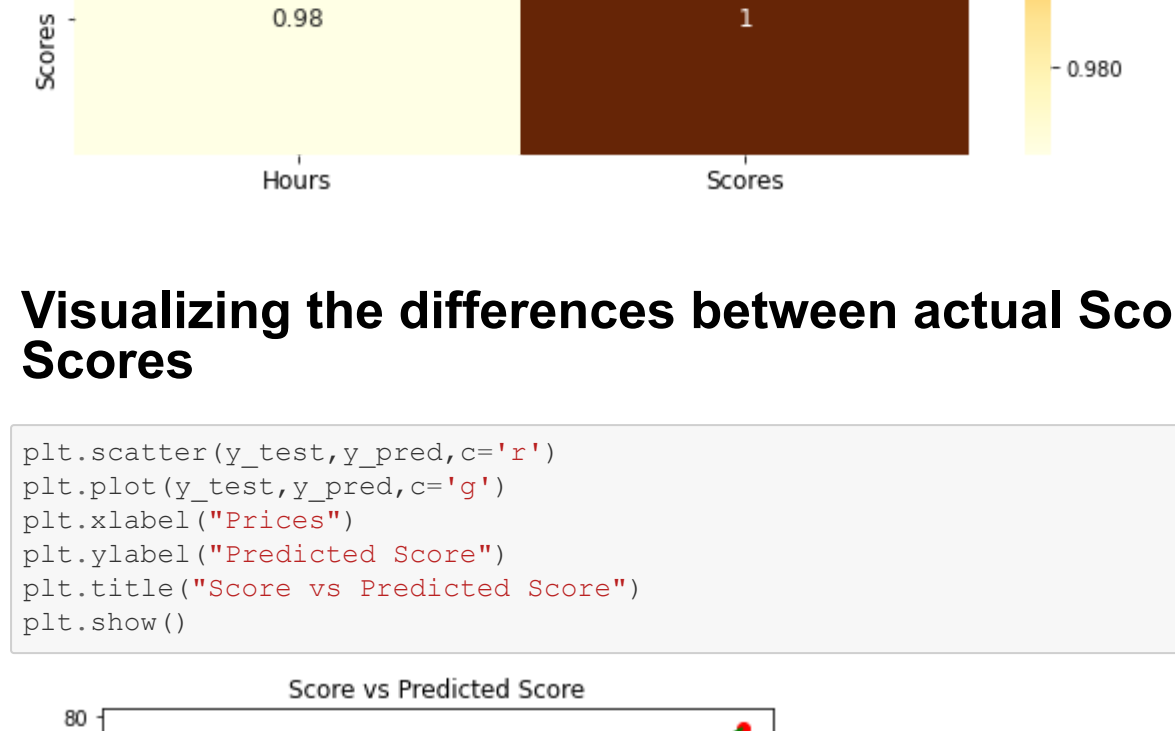


Visualising the Test set results

In [24]:
```
plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('(Testing set)')
plt.xlabel('Hours studied')
plt.ylabel('Percentage Scored')
plt.show()
```



In [25]:
```
# Checking the correlations
plt.figure(figsize=(10,5))
sns.heatmap(df.corr(),annot=True,cmap="YlOrBr",annot_kws={'fontsize':12})
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.show()
```



Visualizing the differences between actual Scores and predicted Scores

In [26]:
```
plt.scatter(y_test,y_pred,c='r')
plt.plot(y_test,y_pred,c='g')
plt.xlabel("Prices")
plt.ylabel("Predicted Score")
plt.title("Score vs Predicted Score")
plt.show()
```

What will be predicted score if a student studies for 9.25 hrs/ day?

Prediction through our model

In [27]:
```
Hours = np.array([[9.25]])
predict=regressor.predict(Hours)
print("No of Hours = {}".format(Hours))
print("Predicted Score = {}".format(predict[0]))
```

```
No of Hours = [[9.25]]
Predicted Score = 92.14523314523314
```

Checking accuracy of our model

In [28]:
```
print("Train : ",regressor.score(X_train,y_train)*100)
print("Test : ",regressor.score(X_test,y_test)*100)
```

```
Train :  95.01107277744313
Test :  95.5570080138813
```

Finding mean absolute error, r^2 score error and Mean Squared Error

In [29]:
```
from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
print('Mean absolute error:', metrics.mean_absolute_error(y_test, regressor.predict(X_test)))
print('r^2 score error:',r2_score(y_test, regressor.predict(X_test)))
print('Mean squared error: ',mean_squared_error(y_test, regressor.predict(X_test)))
```

```
Mean absolute error: 4.691397441397438
r^2 score error: 0.955570080138813
Mean squared error:  25.463280738222547
```

Mean absolute error: 4.691397441397446 which is quite accurate model for predicting the result

In [ ]:

In [ ]: