Task 1: Prediction Using Supervised ML

Predict the percentage of students based on number of study hours

Author: Afshan Saba

Importing libraries

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

Importing dataset

In [3]:
```
df=pd.read_csv("https://raw.githubusercontent.com/AdiPersonalWorks/Random/master/student_scores%20-%20student_
```

Checking Data

In [4]:
```
df.head()
```

Out[4]:

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30

Checking Null Values

In [5]:
```
df.isnull()
```

Out[5]:

	Hours	Scores
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False
5	False	False
6	False	False
7	False	False
8	False	False
9	False	False
10	False	False
11	False	False
12	False	False
13	False	False
14	False	False
15	False	False
16	False	False
17	False	False
18	False	False
19	False	False
20	False	False
21	False	False
22	False	False
23	False	False
24	False	False

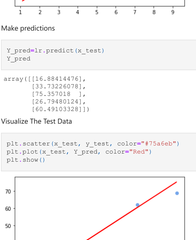
Make a list of columns

In [6]:
```
columns=list(df.columns)
```

In [7]:
```
X=df["Hours"].values.reshape(-1,1)
Y=df["Scores"].values.reshape(-1,1)
```

Visualize the Data

In [8]:
```
plt.scatter(X,Y, color="#1f77b4")
plt.xlabel("Hours")
plt.ylabel("Scores")
plt.title("Hours Vs Scores")
plt.grid()
plt.show()
```



The graph is showing a Linear relationship between hours and scores

Split the Data

In [9]:
```
from sklearn.model_selection import train_test_split
```

In [11]:
```
x_train, x_test, y_train, y_test=train_test_split(X,Y,test_size=0.20, random_state=0)
```

Loading modules for linear regression

In [12]:
```
from sklearn.linear_model import LinearRegression
```

In [13]:
```
lr=LinearRegression()
lr
```

Out[13]:
```
LinearRegression()
```

Training the model

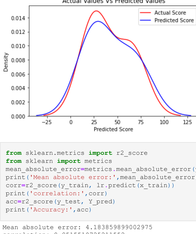
In [14]:
```
lr.fit(x_train, y_train)
```

Out[14]:
```
LinearRegression()
```

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

Visualize The Train Data

In [17]:
```
plt.scatter(x_train, y_train, color="#329ba8")
plt.plot(X, line, color="r")
plt.show()
```



Make predictions

In [18]:
```
Y_pred=lr.predict(x_test)
Y_pred
```

Out[18]:
```
array([[16.88414476],
       [33.73226078],
       [75.357018  ],
       [26.79480124],
       [60.49103328]])
```

Visualize The Test Data

In [21]:
```
plt.scatter(x_test, y_test, color="#75a6eb")
plt.plot(x_test, Y_pred, color="Red")
plt.show()
```


Make a Data Frame

In [22]:
```
df_predict=pd.DataFrame({"Hours":x_test.reshape(1,-1)[0],"Actual Score":y_test.reshape(1,-1)[0],"Predicted Scor
```

In [23]:
```
df_predict
```

Out[23]:

	Hours	Actual Score	Predicted Score
0	1.5	20	16.884145
1	3.2	27	33.732261
2	7.4	69	75.357018
3	2.5	30	26.794801
4	5.9	62	60.491033

Visualize the accuracy of the model

In [24]:
```
df_sorted=df_predict.sort_values(by="Hours")
df_sorted
```

Out[24]:

	Hours	Actual Score	Predicted Score
0	1.5	20	16.884145
3	2.5	30	26.794801
1	3.2	27	33.732261
4	5.9	62	60.491033
2	7.4	69	75.357018

In [27]:
```
title="Actual Values Vs Predicted Values"
ax1=sns.distplot(df_sorted["Actual Score"], hist=False, color="red", label="Actual Score")
sns.distplot(df_sorted["Predicted Score"], hist=False, color="blue", label="Predicted Score", ax=ax1)
plt.legend()
plt.title(title)
plt.show()
```

```
C:\Users\lenovo\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a depre
cated function and will be removed in a future version. Please adapt your code to use either `displot` (a figur
e-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\lenovo\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a depre
cated function and will be removed in a future version. Please adapt your code to use either `displot` (a figur
e-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```


In [28]:
```
from sklearn.metrics import r2_score
from sklearn import metrics
mean_absolute_error=metrics.mean_absolute_error(y_test, Y_pred)
print('Mean absolute error:',mean_absolute_error)
corre2_score(y_train, lr.predict(x_train))
print('Correlation:',corr)
acc=r2_score(y_test, Y_pred)
print('Accuracy:',acc)
```

```
Mean absolute error: 4.183859899002975
Correlation: 0.9515510725211552
Accuracy: 0.9454906892105356
```

Make the Predictions

In [30]:
```
hrs=9.25
pred=lr.predict([[9.25]])
print("The predicted score if a student studies for 9.25 hrs/day is", pred[0])
```

```
The predicted score if a student studies for 9.25 hrs/day is [93.69173249]
```

In [ ]:
```

```