THE SPARKS FOUNDATION

DATA SCIENCE AND BUSINESS ANALYTICS

Task-1

Predict the percentage of a Student based on the number of Study Hours

INTERN: Sonali Bhadra

In [58]:
```
#load all the necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
import statsmodels.formula.api as smf
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
```

In [59]:
```
#to avoid unwanted warnings
import warnings
warnings.filterwarnings('ignore')
```

In [60]:
```
#load the DataFrame
df=pd.read_csv("C:\\Users\\ASUS\\Desktop\\DATA\\study.csv")
df
```

Out[60]:

	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 [61]:
```
df.head()       #to show first 5 rows
```

Out[61]:

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

In [62]:
```
df.tail()     #to show last 5 rows
```

Out[62]:

	Hours	Scores
20	2.7	30
21	4.8	54
22	3.8	35
23	6.9	76
24	7.8	86

In [63]:
```
df.shape     #to show total no. of rows,columns
```

Out[63]:
```
(25, 2)
```

In [64]:
```
df.info
```

Out[64]:
```
<bound method DataFrame.info of     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 [65]:
```
df.describe()
```

Out[65]:

	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 [66]:
```
df.isnull().sum()     #to count missing values
```

Out[66]:
```
Hours     0
Scores    0
dtype: int64
```

In [67]:
```
df.columns
```

Out[67]:
```
Index(['Hours', 'Scores'], dtype='object')
```

In [68]:
```
df.dtypes
```

Out[68]:
```
Hours     float64
Scores      int64
dtype: object
```

In [69]:
```
df.corr()  #Compute pairwise correlation of columns, excluding NA/null values.
```

Out[69]:

	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

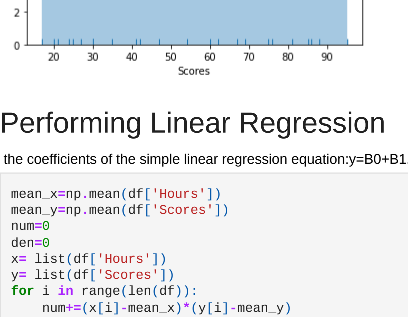
Outlier Removal

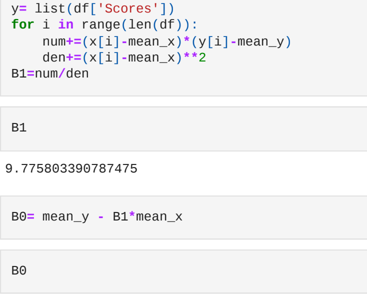
In [70]:
```
def null_detection(df):
    num_cols=[]

    count=0
    t=[]
    for i in num_cols:
        z=np.abs(stats.zscore(df[i]))
        for j in range(len(z)):
            if z[j]>3 or z[j]<-3:
                t.append(j)
                count+=1
    df=df.drop(list(set(t)))
    df=df.reset_index()
    df=df.drop('index', axis=1)
    print(count)
    return df
```

In [71]:
```
df=null_detection(df)
```
```
0
```

Distribution

In [72]:
```
sns.distplot(df["Scores"])
plt.show()

sns.distplot(df["Scores"],kde=False, rug=True)
plt.show()
```





Performing Linear Regression

Calculating the coefficients of the simple linear regression equation:y=B0+B1.x(B1:Slope,B0:Intercept)

In [73]:
```
mean_x=np.mean(df['Hours'])
mean_y=np.mean(df['Scores'])
num=0
den=0
x= list(df['Hours'])
y= list(df['Scores'])
for i in range(len(df)):
    num+=(x[i]-mean_x)*(y[i]-mean_y)
    den+=(x[i]-mean_x)**2
B1=num/den
```

In [74]:
```
B1
```

Out[74]:
```
9.775803390787475
```

In [75]:
```
B0= mean_y - B1*mean_x
```

In [76]:
```
B0
```

Out[76]:
```
2.4836734053731746
```

Marking Predictions

In [77]:
```
df['predicted_Scores'] = B0+B1*df['Hours']
```

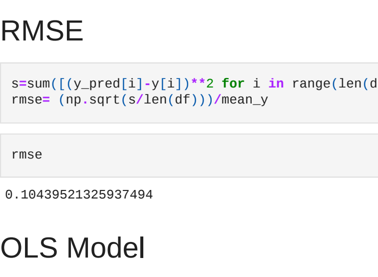
In [78]:
```
df.head()
```

Out[78]:

	Hours	Scores	predicted_Scores
0	2.5	21	26.923182
1	5.1	47	52.340271
2	3.2	27	33.766244
3	8.5	75	85.578002
4	3.5	30	36.698985

In [79]:
```
plt.scatter(df['Hours'], df['Scores'])
plt.scatter(df['Hours'], df['predicted_Scores'])
plt.plot()
```

Out[79]:
```
[]
```



Prediction of given value: 9.25

In [80]:
```
B0+B1*9.25
```

Out[80]:
```
92.90985477015732
```

In [81]:
```
y=list(df['Scores'].values)
y_pred = list(df['predicted_Scores'].values)
```

RMSE

In [82]:
```
s=sum([(y_pred[i]-y[i])**2 for i in range(len(df))])
rmse= (np.sqrt(s/len(df)))/mean_y
```

In [83]:
```
rmse
```

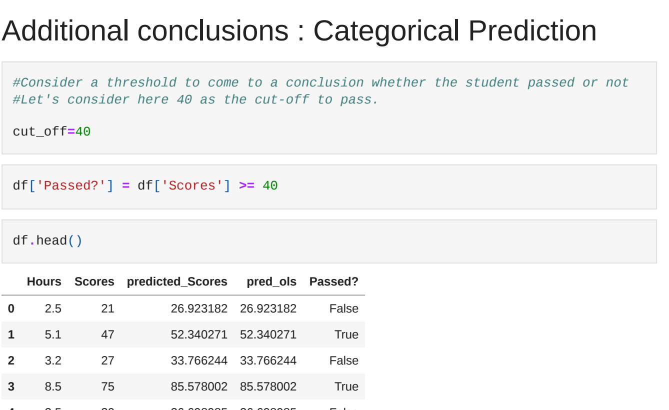
Out[83]:
```
0.10439521325937494
```

OLS Model

In [84]:
```
model=smf.ols('Scores ~ Hours' , data=df)
model=model.fit()
```

In [85]:
```
df['pred_ols']= model.predict(df['Hours'])
```

In [86]:
```
plt.figure(figsize=(12,6))
plt.plot(df['Hours'],df['pred_ols'])       #regression line
plt.plot(df['Hours'],df['Scores'], 'ro')   #scatter plot showing actual data
plt.title('Actual vs Predicted')
plt.xlabel('Hours')
plt.ylabel('Scores')

plt.show()
```



We can observe the predicted value for 9.25 hours is around 92 as above.

Additional conclusions : Categorical Prediction

In [87]:
```
#Consider a threshold to come to a conclusion whether the student passed or not
#Let's consider here 40 as the cut-off to pass.

cut_off=40
```

In [88]:
```
df['Passed?'] = df['Scores'] >= 40
```

In [89]:
```
df.head()
```

Out[89]:

	Hours	Scores	predicted_Scores	pred_ols	Passed?
0	2.5	21	26.923182	26.923182	False
1	5.1	47	52.340271	52.340271	True
2	3.2	27	33.766244	33.766244	False
3	8.5	75	85.578002	85.578002	True
4	3.5	30	36.698985	36.698985	False

Plotting the given data's Results

In [90]:
```
sns.countplot(df['Passed?'])
```

Out[90]:
```
<AxesSubplot:xlabel='Passed?', ylabel='count'>
```

Feature Engineering

In [91]:
```
feature =df['Hours'].values.reshape(-1,1)
target=df['Passed?'].values
```

Splitting the Data

In [92]:
```
x_train, x_test , y_train , y_test= train_test_split(feature,target,random_state=0)
```

Training the KNN Model

In [93]:
```
knn=KNeighborsClassifier(n_neighbors=5)
knn.fit(x_train,y_train)
```

Out[93]:
```
KNeighborsClassifier()
```

Accuracy

In [94]:
```
knn.score(x_train,y_train)
```

Out[94]:
```
0.9444444444444444
```

In [99]:
```
knn.score(x_test,y_test)
```

Out[99]:
```
0.8571428571428571
```

Predicting the outcomes

In [96]:
```
get_results =[[9.25]]
```

In [97]:
```
knn.predict(get_results)
```

Out[97]:
```
array([ True])
```

In [98]:
```
knn.predict([[24]])
```

Out[98]:
```
array([ True])
```

In [55]:
```
knn.predict([[3]])
```

Out[55]:
```
array([False])
```

----THANK YOU----

In [ ]:
```
```