THE SPARKS FOUNDATION

RAZA DESHPANDE

DATA SCIENCE INTERN

TASK 1

Predict the percentage of an student based on the no. of study hours.

In [3]:
```
# Importing all libraries required in this notebook
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
%matplotlib inline
from statsmodels.tools.sm_exog import sm
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
```

In [5]:
```
# Reading data from remote link
df = pd.read_csv("http://bit.ly/w-data")
```

In [6]:
```
df.head()
```

Out[6]:

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

In [7]:
```
df.columns
```

Out[7]:
```
Index(['Hours', 'Scores'], dtype='object')
```

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

Out[8]:
```
Hours     float64
Scores      int64
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 [17]:
```
df.describe
```

Out[17]:
```
<bound method NDFrame.describe 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 [9]:
```
df.corr()
```

Out[83]:

	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

In [19]:
```
def null_detection(df):
    num_cols = []

    count = 0
    i = []
    for i in num_cols:
        z = abs(stats.zscore(df[i]))
        for j in range(len(z)):
            if z[j]>3 or z[j]<-3:
                i.append(j)
                count+=1
    df = df.drop(labels=i)
    df = df.reset_index()
    df = df.drop('index',axis=1)
    print(count)
    return df
```

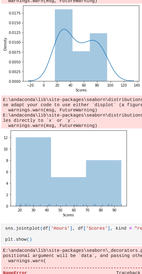
In [10]:
```
df = null_detection(df)
```

```
0
```

In [11]:
```
sns.distplot(df["Scores"])
plt.show()

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

```
E:\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



```
E:\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
E:\anaconda3\lib\site-packages\seaborn\distributions.py:2055: FutureWarning: The `axis` variable is no longer used and will be removed. Instead, assign variables directly to `x` or `y`.
  warnings.warn(msg, FutureWarning)
```


In [12]:
```
sns.jointplot(df['Hours'], df['Scores'], kind = "reg",annotate(stats.pearsonr)

plt.show()
```

```
E:\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

```
NameError                                 Traceback (most recent call last)
<ipython-input-12-e3f9383177d4> in <module>
----> 1 sns.jointplot(df['Hours'], df['Scores'], kind = "reg",annotate(stats.pearsonr)
      2
      3 plt.show()

NameError: name 'annotate' is not defined
```


In [13]:
```
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 [18]:
```
B1
```

Out[18]:
```
9.775803390787475
```

In [19]:
```
B0 = mean_y - B1*mean_x
```

In [20]:
```
B0
```

Out[20]:
```
2.4836734053731746
```

In [21]:
```
df['predicted_Scores'] = B0 +B1*df['Hours']
```

In [22]:
```
df.head()
```

Out[22]:

	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 [23]:
```
plt.scatter(df['Hours'], df['Scores'])
plt.scatter(df['Hours'], df['predicted_Scores'])
plt.plot()
```

Out[23]:
```
[]
```

Prediction of the given value is 9.25

In [24]:
```
B0 + B1*9.25
```

Out[24]:
```
92.90985477015732
```

In [25]:
```
y = list(df['Scores'].values)
y_pred = list(df['predicted_Scores'].values)
```

Root mean square error

In [26]:
```
s = sum([(y_pred[i] - y[i])**2 for i in range(len(df))])
rmse = (s/len(df))**0.5
```

In [27]:
```
rmse
```

Out[27]:
```
6.104395213259374594
```

OLS

In [28]:
```
model = smf.ols('Scores ~ Hours' , data = df)
model = model.fit()
```

In [29]:
```
df['pred_ols'] = model.predict(df['Hours'])
```

In [30]:
```
plt.figure(figsize=(12,6))
plt.plot(df['Hours'], df['pred_ols'])
plt.scatter(df['Hours'], df['Scores'], 'rm')
plt.title('Actual vs Predicted')
plt.xlabel('Hours')
plt.ylabel('Scores')

plt.show()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-30-4cc32703c3d7> in <module>
      3 plt.scatter(df['Hours'], df['Scores'], 'rm')
      4 plt.title('Actual vs Predicted')
----> 5 plt.xlabel('Hours')
      6 plt.ylabel('Scores')

AttributeError: module 'matplotlib.pyplot' has no attribute 'xlabel'
```


In [31]:
```
cut_off = 40
```

In [32]:
```
df['Passed?'] = df['Scores']>=40
```

In [33]:
```
df.head()
```

Out[33]:

	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

In [34]:
```
sns.countplot(df['Passed?'])
```

```
E:\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[34]:
```
<AxesSubplot:xlabel='Passed?', ylabel='count'>
```


In [35]:
```
feature = df['Hours'].values.reshape(-1, 1)
target = df['Passed?'].values
```

In [36]:
```
X_train, X_test, y_train, y_test = train_test_split(feature, target , random_state=0)
```

In [37]:
```
knn =KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
```

Out[37]:
```
KNeighborsClassifier()
```

In [38]:
```
knn.score(X_train, y_train)
```

Out[38]:
```
0.9444444444444444
```

In [39]:
```
knn.score(X_test, y_test)
```

Out[39]:
```
0.8571428571428571
```

In [40]:
```
get_results = 8 *[9.25]
```

In [41]:
```
knn.predict(get_results)
```

Out[41]:
```
array([ True])
```

In [42]:
```
knn.predict([[14]])
```

Out[42]:
```
array([ True])
```

In [43]:
```
knn.predict([[3]])
```

Out[43]:
```
array([False])
```

THANKYOU