In [ ]:
```
# GRIP : The Sparks Foundation

# Data Science and Buiseness Analytics Intern

# Author: ASHU RANA

# Task 1: Prediction using Supervised ML
***
In this task, We are going to predict the percentage score of a student based on the number of hours studied. The task has two variables where the feature is

Data :

Hours,Scores :
2.5,21 ;
5.1,47 ;
3.2,27 ;
8.5,75 ;
3.5,30 ;
1.5,20 ;
9.2,88 ;
5.5,60 ;
8.3,81 ;
2.7,25 ;
7.7,85 ;
5.9,62 ;
4.5,41 ;
3.3,42 ;
1.1,17 ;
8.9,95 ;
2.5,30 ;
1.9,24 ;
6.1,67 ;
7.4,69 ;
2.7,30 ;
4.8,54 ;
3.8,35 ;
6.9,76 ;
7.8,86 .
```

In [43]:
```
# Importing the required libraries

import pandas as pd                           # Pandas stands for panel data, library for data manipulation and data analysis
import numpy as np                            # Numpy stands for numerical pythn, library for numeric and scientific computing
import matplotlib.pyplot as plt               # library for Data Visualisation and draw various plots and charts
import seaborn as sns                         # library for visualisation and built over matplotlib
```

In [44]:
```
# Reading data from remote URL and printing that data

url = 'http://bit.ly/w-data'
data = pd.read_csv(url)
print(data.shape)
```

```
(25, 2)
```

In [45]:
```
# showing first 5 Rows of dataset
data.head()
```

Out[45]:

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

In [46]:
```
# describing about data like count, mean, min etc.
data.describe()
```

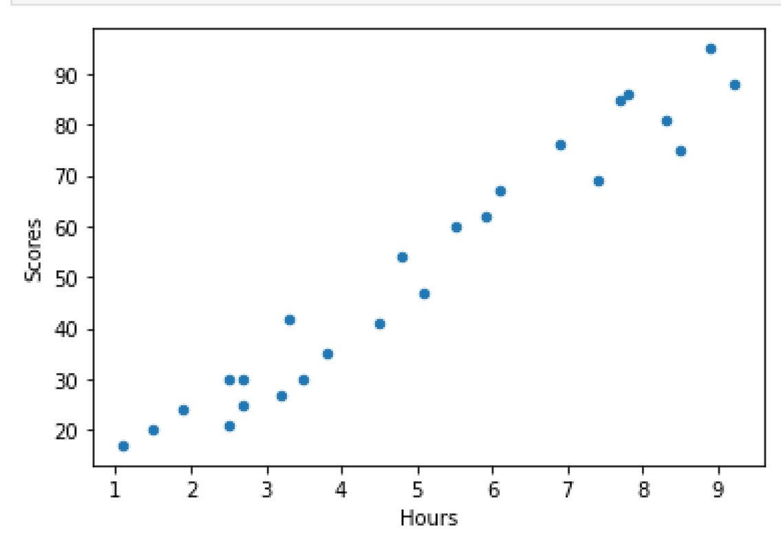
Out[46]:

	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 [47]:
```
# showing info about data like how many null values, no. of columns, memory, type of variable etc.
data.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 [48]:
```
# plotting the scatterplot of 2 variables where the no. of hours studied on x-axis & scores on y-axis
data.plot(kind='scatter', x='Hours', y='Scores')
plt.show()
```



In [49]:
```
# from above,we can see there is a linear relationship between two variables which can be validated from correlation coefficient
data.corr(method='pearson')
```

Out[49]:

	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

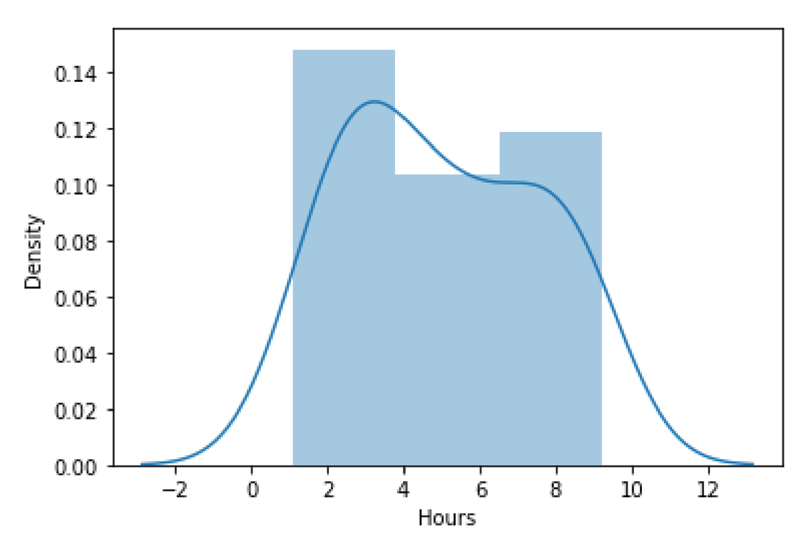
In [50]:
```
# coefficient is 0.976 approximately equal to 1 and is positive which means there is a positive linear relationship
# implies Hours is directly proportional to scores which also makes sense
```

In [51]:
```
# Assigning hours and scores columns of dataset in hours and scores as lists type So that we can use it directly
hours = data['Hours']
scores = data['Scores']
```

In [52]:
```
# plotting the distribution plot of the two variable, getting variables are in particular range & there are no outliers
sns.distplot(hours)
```

```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```

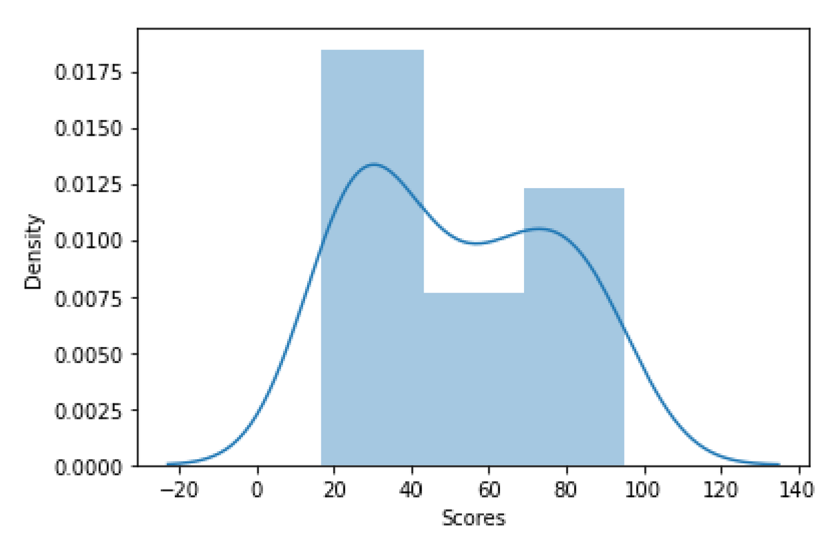
Out[52]:
```
<AxesSubplot:xlabel='Hours', ylabel='Density'>
```



In [53]:
```
sns.distplot(scores)
```

```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```

Out[53]:
```
<AxesSubplot:xlabel='Scores', ylabel='Density'>
```



Linear Regression

Here, we are using Linear Regression model. What linear regression does is that, it finds the slope and intercept(denoted by m) of the line where all the points fall. But, according to scatter plot, there can't be any such line in which all the points fall so, we find a line in which difference of the predicted values of the line and the actual values of the point is minimum i.e. sum of the difference of the predicted values and the actual values of the score has to be minimum.

In [54]:
```
# allocating data into x and y
x = data.iloc[:, :1].values
y = data.iloc[:, 1:].values
```

In [55]:
```
x
```

Out[55]:
```
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 [56]:
```
y
```

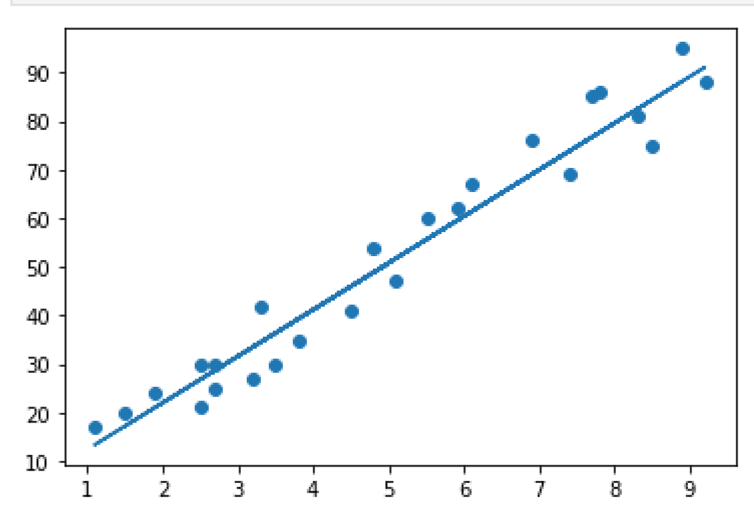
Out[56]:
```
array([[21],
       [47],
       [27],
       [75],
       [30],
       [20],
       [88],
       [60],
       [81],
       [25],
       [85],
       [62],
       [41],
       [42],
       [17],
       [95],
       [30],
       [24],
       [67],
       [69],
       [30],
       [54],
       [35],
       [76],
       [86]], dtype=int64)
```

In [57]:
```
# In this, we are first dividing our data into Train dataset(80% data) and Test dataset(20% data).
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=50)
```

In [58]:
```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(x_train, y_train)
```

Out[58]:
```
LinearRegression()
```

In [59]:
```
# Two outputs of the line: Slope of the line & Intercepts
m = reg.coef_
c = reg.intercept_
line = m*x + c
plt.scatter(x,y)
plt.plot(x,line)
plt.show()
```



In [61]:
```
# Target values which we got in the data and the predicted values
y_pred=reg.predict(x_test)
```

In [71]:
```
# This is the Predicted score
y_pred
```

Out[71]:
```
array([[88.21139357],
       [28.71845267],
       [69.02012231],
       [39.27365186],
       [13.36543566]])
```

In [72]:
```
y_test
```

Out[72]:
```
array([[95],
       [30],
       [76],
       [35],
       [17]], dtype=int64)
```

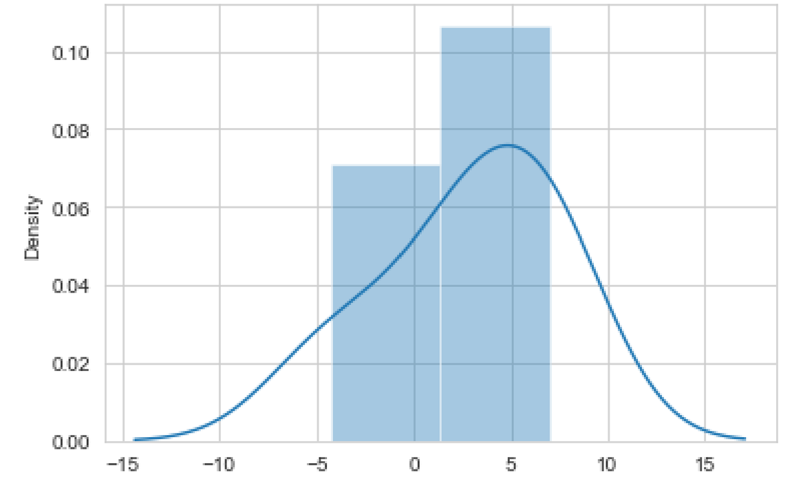
In [78]:
```
# Comparison Actual vs Predicted
actual_predicted = pd.DataFrame({'Target':[y_test], 'Predicted':[y_pred]})
actual_predicted
```

Out[78]:

	Target	Predicted
0	[[95], [30], [76], [35], [17]]	[[88.21139357388516], [28.718452665057836], [6...

In [82]:
```
# plotting a distribution plot for the difference between the targeted value and predicted value
# This difference is very close to zero and range is -5 to 5 which tells our model is fitting the data well
sns.set_style('whitegrid')
sns.distplot(np.array(y_test-y_pred))
plt.show()
```

```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```



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

In [83]:
```
h=9.25
s=reg.predict([[h]])
print("If a student studies for {} hours per day he/she will score {} % in exam.".format(h,s))
```

```
If a student studies for 9.25 hours per day he/she will score [[91.56986604]] % in exam.
```

Model Evaluation

In [86]:
```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Error: 4.5916495300630285
```

In [ ]: