The Spark Foundation

Data Science and Business Analytics Intern

Task 1

Prediction Using Supervised Machine Learning

- objective: To Predict the percentage of a student score on the basis of no. of study hours.
  - By : MAYURI ARUN PATHAK

In [1]:
```
#Setting Working Directory
import os
os.chdir("H:\Data Science\Internship\Spark")
```

Importing the Libraries

In [2]:
```
#Importing required libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

#To ignore the warnings
import warnings
warnings.filterwarnings("ignore")
```

Understanding Data

In [31]:
```
# Reading the data from remote link

data = pd.read_csv("student score.csv")
print("Data load successfully.")
```

```
Data load successfully.
```

In [41]:
```
#Let's observe the dataset
data.head()
```

Out[41]:

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

In [5]:
```
data.tail()
```

Out[5]:

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

In [6]:
```
# To find the number of columns and rows
data.shape
```

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

- There are 25 rows and 2 columns in a data.

In [7]:
```
#To find more information about our dataset,null values in data
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 [8]:
```
#Statistical Summary of data
data.describe()
```

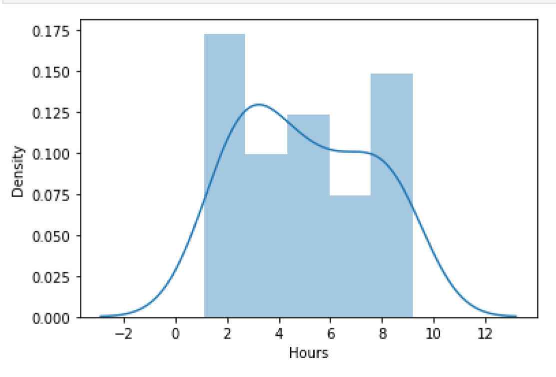
Out[8]:

	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

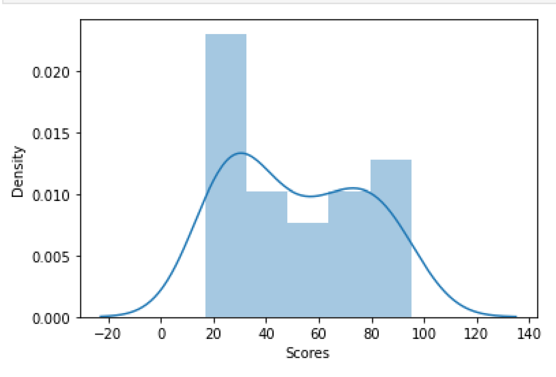
Visualizing Data

Univariate Analysis

In [9]:
```
#Distribution of variable Hours
sns.distplot(data["Hours"],bins = 5)
plt.show()
```



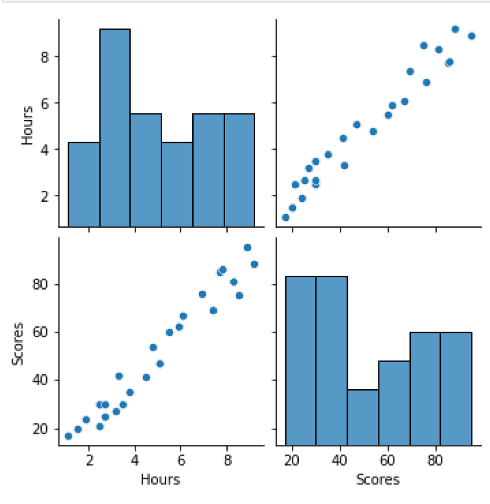
In [10]:
```
# Distribution of variable Scores
sns.distplot(data["Scores"],bins = 5)
plt.show()
```



- Variables in a particular region .There is no outlier present .It is good for Model.

Bivariate Analysis

In [11]:
```
# Scatter plot between hours and scores
sns.pairplot(data)
plt.show()
```



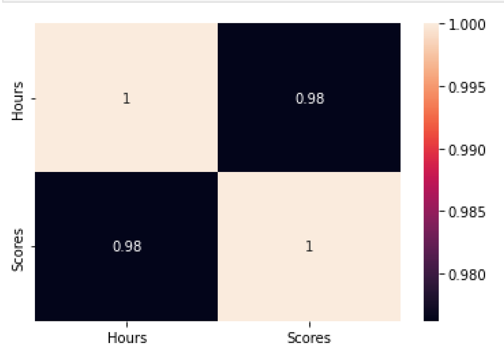
- Fairly linear relationship between the two variables.

In [33]:
```
# Correlation between Hours and Scores
data.corr()
```

Out[33]:

	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

In [13]:
```
#Visualizing correlation between Hours and Scores
sns.heatmap(data.corr(),annot = True)
plt.show()
```



- Using Heatmap , we clearly see that there is positive correlation between hours and scores.

Model Building

Simple Linear Regression

- Equation of linear regression

y = c + m1x1 + m2x2 + . . . +mnxn

y = c + m1 *Hours

In [45]:
```
#Defining the feature variable and the response variable
x = data['Hours']
y = data['Scores']
```

Train Test Split

In [35]:
```
#Importing library for train test split
from sklearn.model_selection import train_test_split
```

In [36]:
```
#Split data into train and test data
x_train,x_test,y_train,y_test = train_test_split(x,y,train_size = 0.7 , test_size = 0.3 ,random_state = 50)
```

In [17]:
```
# Feature variable head
x_train.head()
```

Out[17]:
```
3     8.5
19    7.4
7     5.5
10    7.7
2     3.2
Name: Hours, dtype: float64
```

Building Linear Model

In [18]:
```
# Importing Library for building linear model
import statsmodels.api as sm
```

In [19]:
```
# Add a costant
x_train_sm = sm.add_constant(x_train)

# Fit the regression line using 'OLS'
lr = sm.OLS(y_train , x_train_sm).fit()
```

In [20]:
```
#Print the Parameters
print(lr.params)
```

```
const    3.784308
Hours    9.521606
dtype: float64
```

In [21]:
```
print(lr.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                 Scores   R-squared:                       0.939
Model:                            OLS   Adj. R-squared:                  0.935
Method:                 Least Squares   F-statistic:                     232.5
Date:                Sat, 13 May 2023   Prob (F-statistic):           1.54e-10
Time:                        22:26:48   Log-Likelihood:                -53.827
No. Observations:                  17   AIC:                             111.7
Df Residuals:                      15   BIC:                             113.3
Df Model:                           1                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          3.7843      3.503      1.080      0.297      -3.682      11.250
Hours          9.5216      0.624     15.247      0.000       8.191      10.853
==============================================================================
Omnibus:                        4.887   Durbin-Watson:                   1.505
Prob(Omnibus):                  0.087   Jarque-Bera (JB):                1.531
Skew:                          -0.202   Prob(JB):                        0.465
Kurtosis:                       1.586   Cond. No.                         13.6
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

1.R - squared is 0.939

- Meaning that 93.9% of the variance in Scores is explained by 'Hours'

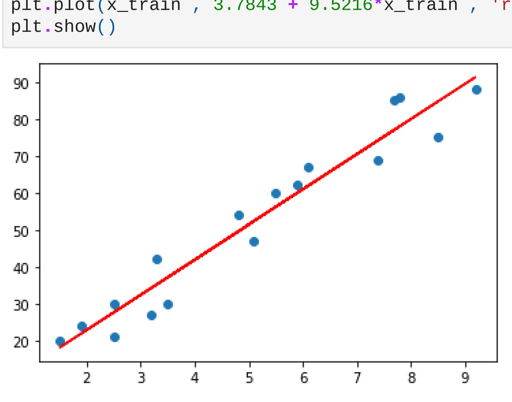
2. The coefficient for Hours is 9.5216, with a very low p value

- The coefficient is statistically significant.So the association is not purely by chance.

3. F statistic has a very low p value (practically low)

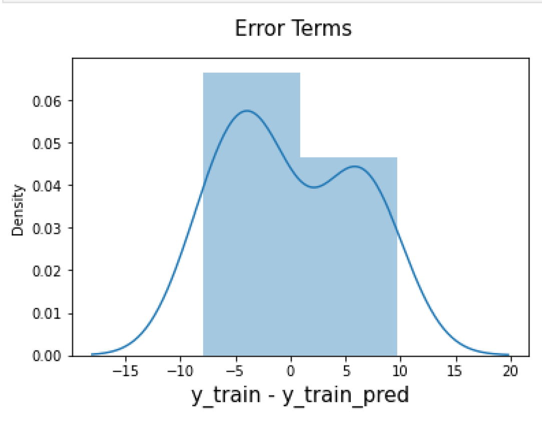
- The model fit is statistically significant and the explained variance is not purely by chance.
- Scores = 3.7843 + 9.5216 * Hours

In [38]:
```
# Visualize the model
plt.scatter(x_train , y_train)
plt.plot(x_train , 3.7843 + 9.5216*x_train , 'r')
plt.show()
```



In [23]:
```
# Predicted value of scores
y_train_pred = lr.predict(x_train_sm)
```

In [24]:
```
#Checking Linear regression asssumption
res = y_train_pred - y_train
fig = plt.figure()
sns.distplot(res, bins = 2)
fig.suptitle ('Error Terms' , fontsize = 15)
plt.xlabel('y_train - y_train_pred' , fontsize = 15)
plt.show()
```



- The residuals are followin the Bi model distribution and not normally distributed with mean 0. Therefore, we need more number of data points to train the models.

Prediction On The Test Data

In [25]:
```
# add constant
x_test_sm = sm.add_constant(x_test)

#Predicting the y values corresponding to X_test
y_test_pred = lr.predict(x_test_sm)
```

In [26]:
```
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
```

In [27]:
```
#RMSE
np.sqrt(mean_squared_error(y_test , y_test_pred))
```

Out[27]:
```
4.636798722184965
```

In [28]:
```
# r-squared
print("r-squared_train:" , r2_score(y_train , y_train_pred))
print("r-squared_test:" , r2_score(y_test , y_test_pred))
```

```
r-squared_train: 0.9393866419807256
r-squared_test: 0.9717199573955198
```

In [29]:
```
# What will be predicted score if a student studies for 9.2 hrs/day?
Score = 3.7843 + 9.5216*9.25
print("When a student studying for 9.25 hours then he/she will score.",Score)
```

```
When a student studying for 9.25 hours then he/she will score. 91.8591
```

Suggestions are always Welcome!

Thank You!