Task 1- Prediction using Supervise Machine Learning
In this task it is required to predict the percentage of a student on the basis of number of hours studies using Linear Regression Supervise Machine Learning Algorithm.

Steps:

Step 1:-Importing the dataset

Step2:-Visualizing dataset

Step3:-Data preparation

Step4:-Training the Algorithm

Step5:-Visualizing the Model

Step6:-Making predictions

Step7:-Evaluating the Model

Step 1 - Importing the dataset
In this step , we will import the dataset through the link with the help of pandas library and then we will observe the data

In [1]:
```
#Imposting all the required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
#Reading data from remote link
url="C:\\Users\\DELL\\OneDrive\\Desktop\Student.csv"
df=pd.read_csv(url)
```

In [3]:
```
#now lets observe the dataset
df.head()
```

Out[3]:

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

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

Out[4]:

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

In [5]:
```
#to find the numbers of coulumn and rows
```

```
df.shape
```

Out[5]:
```
(25, 2)
```

In [6]:
```
#to find out more information about our data set
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 [7]:
```
#for summary Statistics
df.describe()
```

Out[7]:

	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

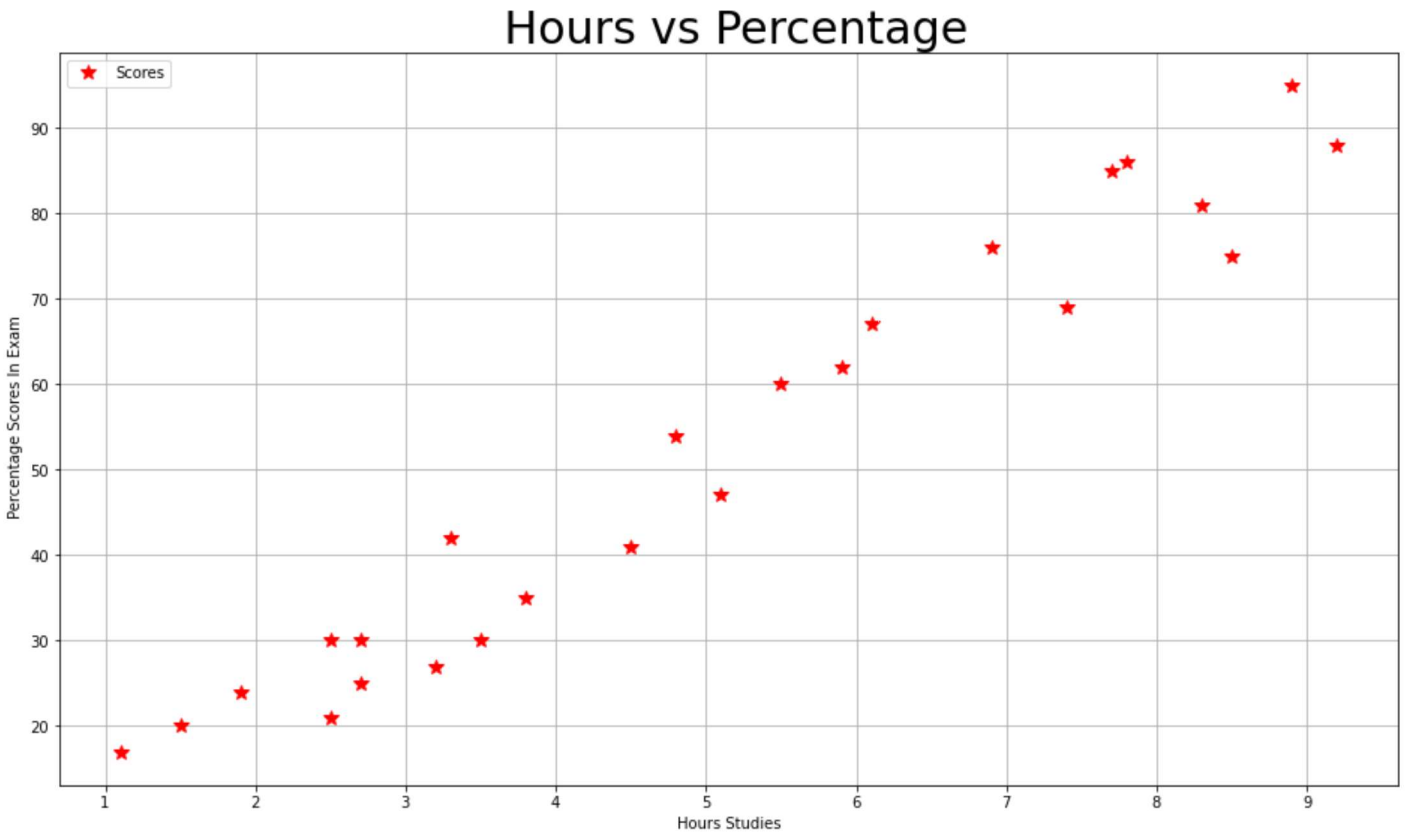
In [8]:
```
#Now we will check if our datasert contain null or missing values
df.isnull().sum()
```

Out[8]:
```
Hours     0
Scores    0
dtype: int64
```

as we can see that no null value in our data set so we can now move on to our next step

Step 2 -visualizing the dataset

In [15]:
```
#Plotting the dataset
plt.rcParams["figure.figsize"]=[16,9]
df.plot(x='Hours',y='Scores', style='*',color='red',markersize=10)
plt.title(" Hours vs Percentage",size=30)
plt.xlabel("Hours Studies",)
plt.ylabel(" Percentage Scores In Exam")
plt.grid()
plt.show()
```



From the graph above we can obseve that there is a linear Relationship between " Hours Studied and " Percentage Marks" . So we use Linear Regredssion Supervise Machine Learning Model to predict the further values.

In [16]:
```
#we can also use .corr  to determine correlation between the variables
df.corr()
```

Out[16]:

	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

Step 3 - Data Preparation

In this step we will divide the data into "features"(inputs) and "labels"(outputs) .After that we will split the whole dataset into 2 parts 1] testing data 2]training the data

In [35]:
```
#Using iloc function we will devide the data
x= df.iloc[:, :1].values
y=df.iloc[:, 1:].values
```

In [36]:
```
x
```

Out[36]:
```
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 [37]:
```
y
```

Out[37]:
```
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 [41]:
```
#Spliting data into training and testing 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=0)
#0.2 because we gonna divide our data into 20%:80% ratio
#20% :for sample model (test samples)
#80% :for best fit model(trian samples)
```

Step 4 - Training the Algorithm

We have to slit our data into trainnig and testing sets and now we will train our Model

In [42]:
```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(x_train,y_train)
#Regretion model fitted for train data
```

Out[42]:
```
LinearRegression()
```

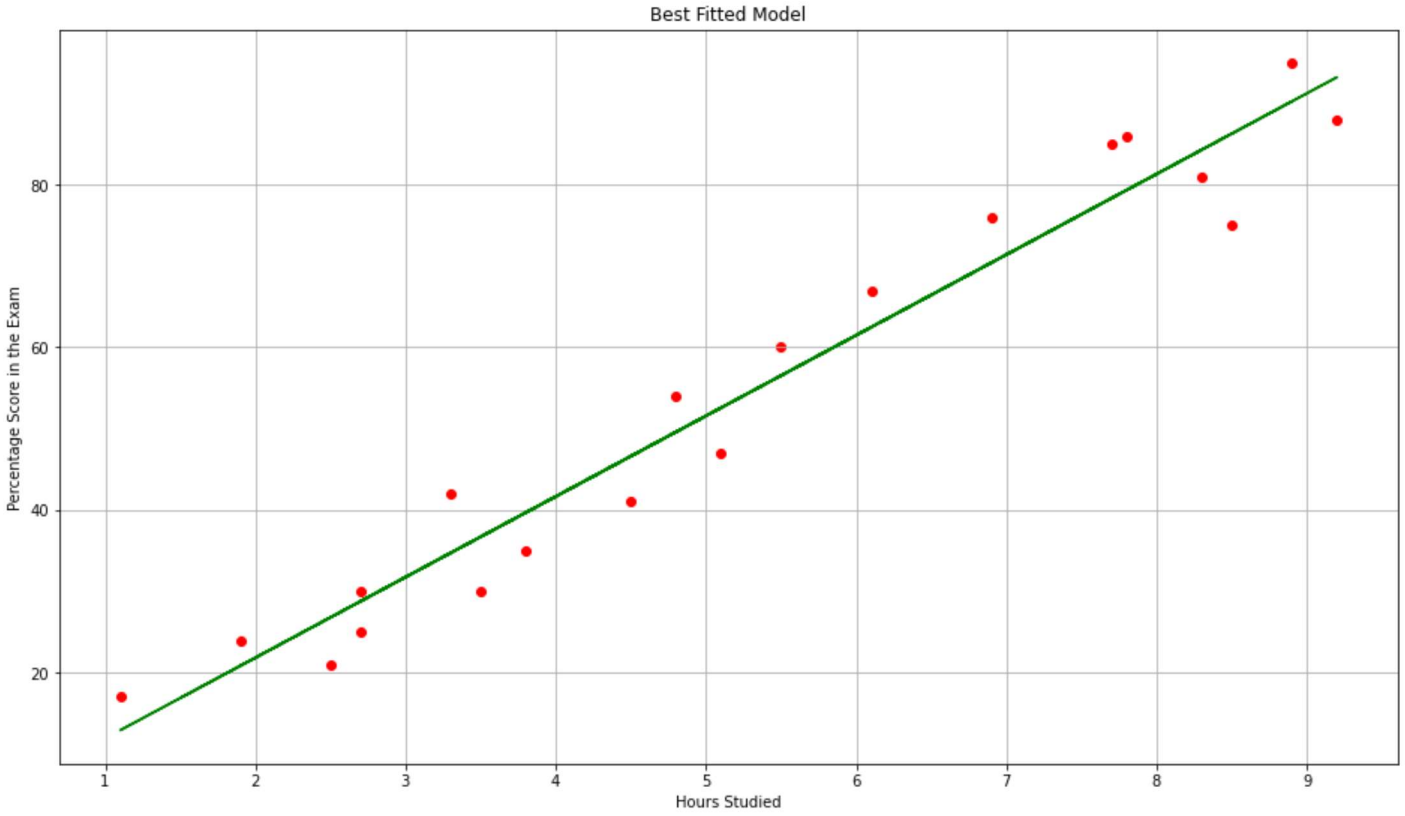
Step 5 - Visualizing the Model

After training the model now its time to Visualizing the Model

In [...

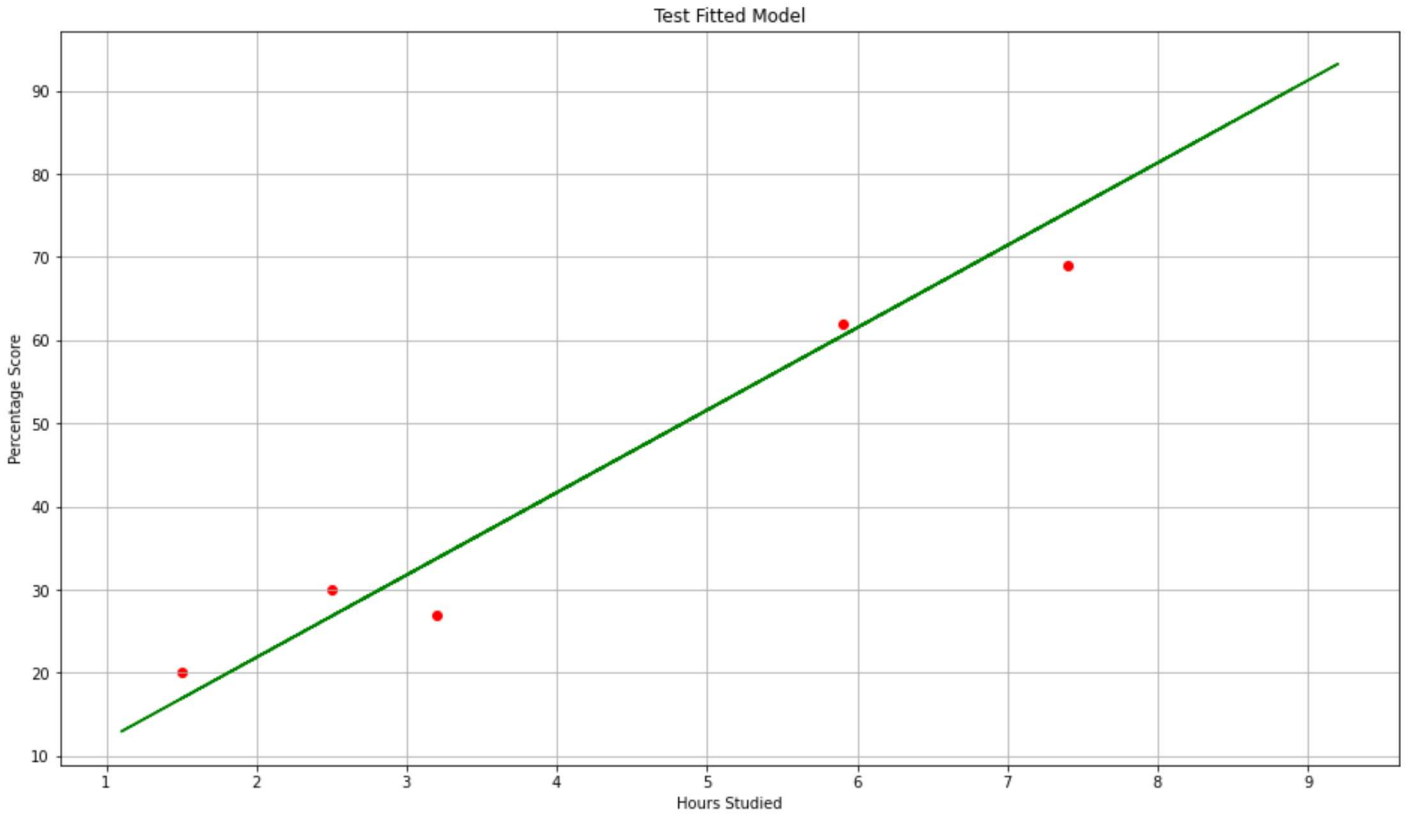
```
line=model.intercept_+model.coef_*x      #where
                                            #model.intercent_=alpha intercept parameter
                                            #model.coef_=beta slope parameter
                                            #line=E(y) Expected values of Percentage Mar
                                            #x=Hours Studied

#Plotting for the  training data
plt.rcParams["figure.figsize"]=[16,9]
plt.scatter(x_train,y_train,color='red')
plt.plot(x,line,color="green");
plt.xlabel("Hours Studied")
plt.ylabel("Percentage Score in the Exam")
plt.title("Best Fitted Model") #for 80% sample Model
plt.grid() #for backgound square lines
plt.show()
```



In [57]:

```
#plotting for the testing data
plt.rcParams["figure.figsize"]=[16,9]
plt.scatter(x_test,y_test,color="red")
plt.plot(x,line,color="green")
plt.xlabel("Hours Studied")
plt.ylabel("Percentage Score")
plt.title("Test Fitted Model") #20% sample model
plt.grid()
plt.show()
```



Step 6 - Making Predictions

Now that we have trained our algorithm , its time to make some prediction

In [58]:
```
#predicting values for test data using regression model fitted in train data.
print(x_test) #testing data - In hours
y_pred=model.predict(x_test) #predicting the Score
```

```
[[1.5]
 [3.2]
 [7.4]
 [2.5]
 [5.9]]
```

In [59]:
```
y_test #Actual data
```

Out[59]:
```
array([[20],
       [27],
       [69],
       [30],
       [62]], dtype=int64)
```

In [60]:
```
y_pred  #Predicted data
```

Out[60]:
```
array([[16.88414476],
       [33.73226078],
       [75.357018  ],
       [26.79480124],
       [60.49103328]])
```

In [66]:
```
#comparing Actual and Predicted data
comp=pd.DataFrame({"Actual":[y_test],"Predicted":[y_pred]})
comp
```

Out[66]:

	Actual	Predicted
0	[[20], [27], [69], [30], [62]]	[[16.884144762398037], [33.73226077948984], [7...

In [77]:
```
#Testing with our own data
for i in range(3):
    hours=float(input("Enter the hours: "))
    own_pred=model.predict([[hours]])
    print("The Predicted score if a person studies for",hours,own_pred[0])
```

```
Enter the hours: 6.5
The Predicted score if a person studies for 6.5 [66.43742717]
Enter the hours: 8.63
The Predicted score if a person studies for 8.63 [87.54712547]
Enter the hours: 9.25
The Predicted score if a person studies for 9.25 [93.69173249]
```

Step 7 -Evaluating the Model

In the last step , we gonna evaluate our trained model by calculating mean absolute error

In [79]:
```
from sklearn import metrics
print("Mean Absolute Error :", metrics.mean_absolute_error(y_test,y_pred))
```

```
Mean Absolute Error : 4.183859899002975
```

In [ ]: