

GRADUATE ROTATIONAL INTERNSHIP PROGRAM (GRIP)

THE SPARK FOUNDATION

NAME:AKASH GAWAS

TASK1:PREDICTION USING SUPERVISED ML

Predict the percentage of student on basis of how many hour in a day they study.

```
In [24]: #Import libraries
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn
%matplotlib inline

In [25]: #import dataset
data=pd.read_csv("My_data.csv")
data.head()
```

```
Out[25]:
```

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

```
In [26]: #check the information about our dataset
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: 464.0 bytes

In [27]: data.describe()

Out[27]:
```

	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

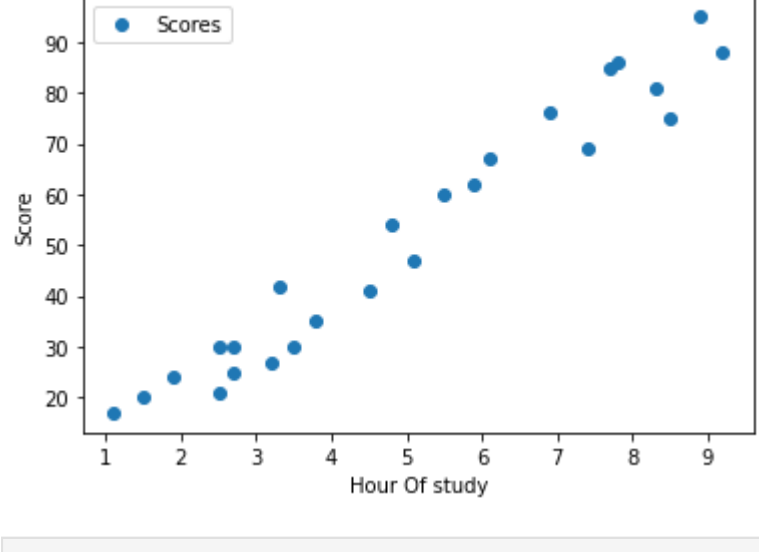
```
In [28]: #checking for missing or null value are present or not
data.isna().sum()

Out[28]:
Hours      0
Scores     0
dtype: int64

In [29]: data.shape

Out[29]: (25, 2)

In [30]: #ploting our dataset to get clear understanding about our dataset
data.plot(x="Hours",y="Scores",style="o")
plt.title("Hours vs Scores")
plt.xlabel("Hour Of study")
plt.ylabel("Score")
plt.legend()
plt.show()
```



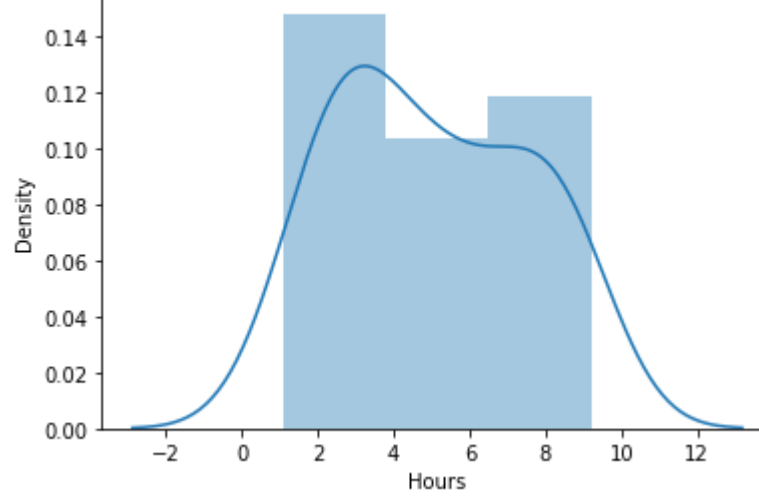
```
In [31]: # correlation is useful to find out relation among them.
data.corr()

Out[31]:
```

	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

```
In [32]: sns.distplot(data["Hours"])

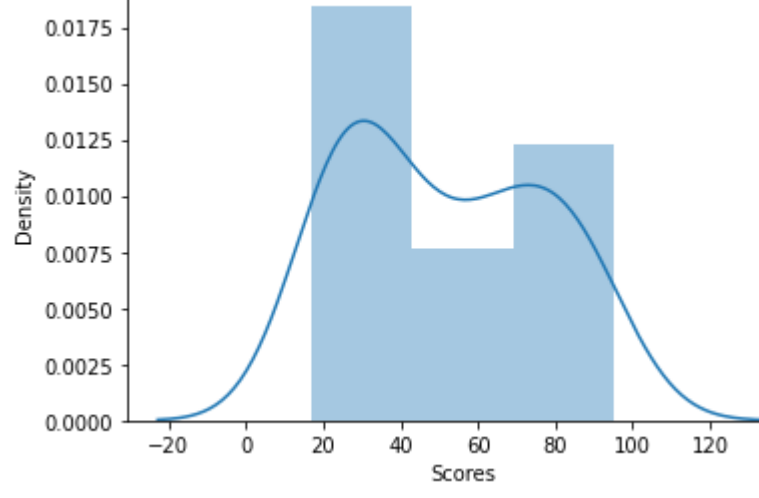
Out[32]: <AxesSubplot: xlabel='Hours', ylabel='Density'>
```



From the above graph we conclude that hour of study and score are strongly correlated with each other.

```
In [33]: sns.distplot(data["Scores"]);

Out[33]:
```



Now we are building a linear regression model.

```
In [34]: x=data.iloc[:, :-1]
y=data.iloc[:, -1]

In [35]: #splitting the dataset into train and test set
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)

In [36]: y_test

Out[36]:
```

5	20
2	27
19	69
16	30
11	62

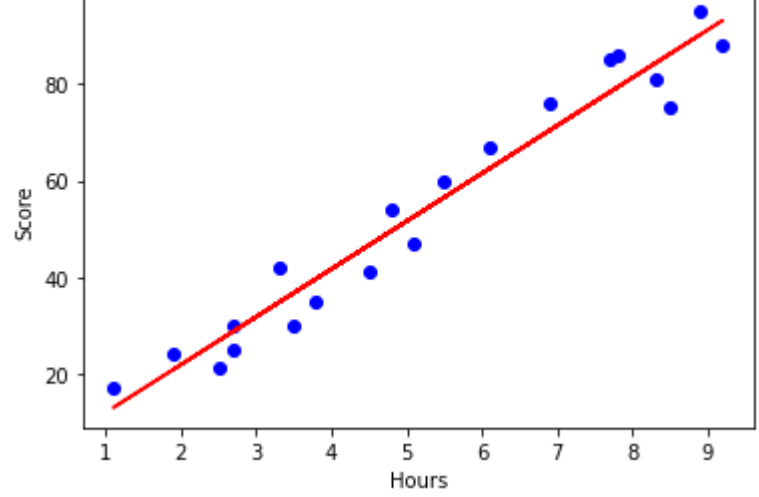
Name: Scores, dtype: int64

```
In [37]: # training our model
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(x_train,y_train)

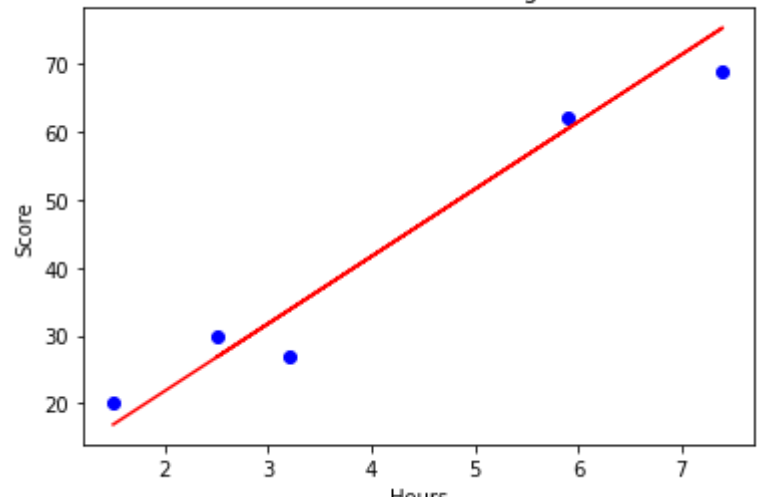
Out[37]: LinearRegression()

In [38]: y_pred=model.predict(x_test)

In [39]: #visualize the training test result
plt.scatter(x_train,y_train,color="blue")
plt.plot(x_train,model.predict(x_train),color="red")
plt.title("Hours vs Scores (Training set)")
plt.xlabel("Hours")
plt.ylabel("Score")
plt.show()
```



```
In [40]: plt.scatter(x_test,y_test,color="blue")
plt.plot(x_test,y_pred,color="red")
plt.title("Hours vs Scores (Testing set)")
plt.xlabel("Hours")
plt.ylabel("Score")
plt.show()
```



```
In [41]: #pred=algo.predict(x_test)
df=pd.DataFrame({"Actual":y_test,"Predict":y_pred})

In [42]: df

Out[42]:
```

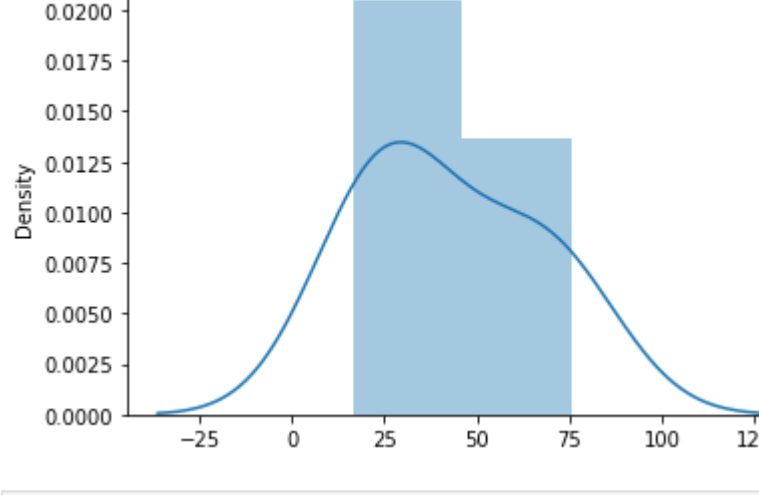
	Actual	Predict
5	20	16.884145
2	27	33.732261
19	69	75.357018
16	30	26.794801
11	62	60.491033

```
In [43]: prediction=model.predict([[9.5]])
prediction

Out[43]: array([96.16939661])

In [44]: sns.distplot(y_pred)

Out[44]: <AxesSubplot: ylabel='Density'>
```



```
In [45]: from sklearn import metrics
print("Mean Absolute Error:", metrics.mean_absolute_error(y_test, y_pred))

Mean Absolute Error: 4.18385989900298

In [46]: h=9.25
s=model.predict([[h]])
print(f"If students study for {h} hour per/day then he/she will score {s}% marks")

If student studies for 9.25 hour per/day then he/she will score [93.69173249]% marks
```