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 [1]:
```
#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 [2]:
```
#import dataset
data=pd.read_csv("My_data.csv")
data.head()
```

Out[2]:

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

In [3]:
```
#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 [4]:
```
data.describe()
```

Out[4]:

	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 [5]:
```
#checking for missing or null value are present or not
data.isna().sum()
```

Out[5]:
```
Hours     0
Scores    0
dtype: int64
```

In [6]:
```
data.shape
```

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

In [7]:
```
#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()
```

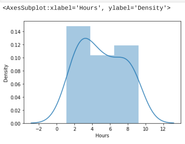
In [8]:
```
# correlation is useful to find out relation among them.
data.corr()
```

Out[8]:

	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

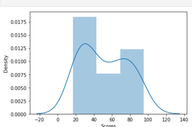
In [9]:
```
sns.distplot(data["Hours"])
```

Out[9]:
```
<AxesSubplot:xlabel='Hours', ylabel='Density'>
```



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

In [10]:
```
sns.distplot(data["Scores"]);
```



Now we building linear regression model

In [11]:
```
x=data.iloc[:,:-1]
y=data.iloc[:,-1]
```

In [12]:
```
#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 [13]:
```
y_test
```

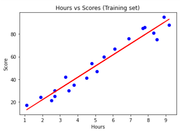
Out[13]:
```
5     20
2     27
19    69
16    30
11    62
Name: Scores, dtype: int64
```

In [14]:
```
# training our model
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(x_train,y_train)
```

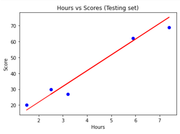
Out[14]:
```
LinearRegression()
```

In [15]:
```
y_pred=model.predict(x_test)
```

In [16]:
```
#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 [17]:
```
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 [18]:
```
#pred=algo.predict(x_test)
df=pd.DataFrame({"Actual":y_test,"Predict":y_pred})
```

In [19]:
```
df
```

Out[19]:

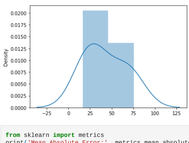
	Actual	Predict
5	20	16.884145
2	27	33.732261
19	69	75.357018
16	30	26.794801
11	62	60.491033

In [20]:
```
prediction=model.predict([[9.5]])
prediction
```

Out[20]:
```
array([96.16939661])
```

In [21]:
```
sns.distplot(y_pred)
```

Out[21]:
```
<AxesSubplot:ylabel='Density'>
```



In [22]:
```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Error: 4.183859899002982
```

In [23]:
```
h=9.25
s=model.predict([[h]])
print(f"If students studies for {h} hour per/day then he/she will score {s}% marks")
```

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