Author : Harish Nakireddy

GRIP :The SPark Foundation

Data Science and Business Analytics Intern

TASK 1: Prediction using Supervised ML

importing all required libraries

In [47]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

from sklearn.linear_model import LinearRegression
```

In [48]:
```
url = "http://bit.ly/w-data"
data = pd.read_csv(url)
data.head()
```

Out[48]:

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

checking data

In [49]:
```
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
```

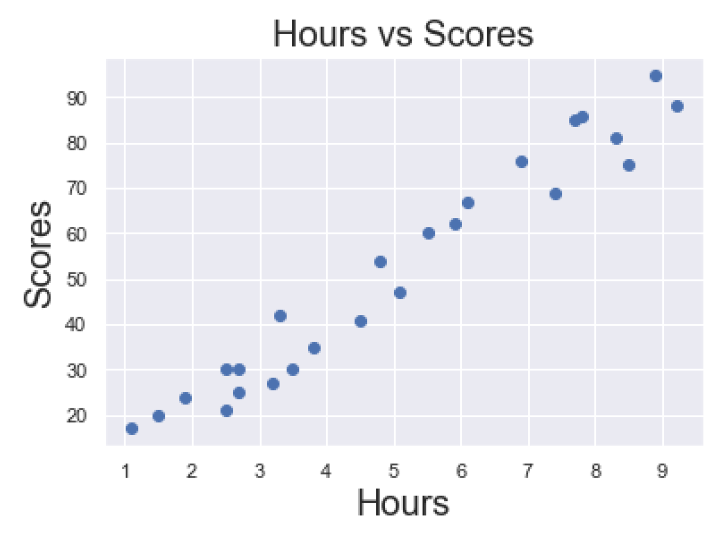
In [50]:
```
data.describe()
```

Out[50]:

	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 [51]:
```
#declare variables
x=data["Hours"]    #independent feature
y=data["Scores"]   #dependent feature
```

In [53]:
```
# plot scatter plot between x,y features
plt.scatter(x,y)
plt.title("Hours vs Scores",size=20)
plt.xlabel("Hours",size=20)
plt.ylabel("Scores",size=20)
plt.show()
```



In [54]:
```
#we can see that there is a linearity between the features from above gr
aph.we can use linear regresion
```

In [55]:
```
#reshape x and y with reshape method
X = x.values.reshape(-1,1)
Y = y.values.reshape(-1,1)
```

Spliting dataset into training and testing data

In [56]:
```
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,rando
m_state=1)
```

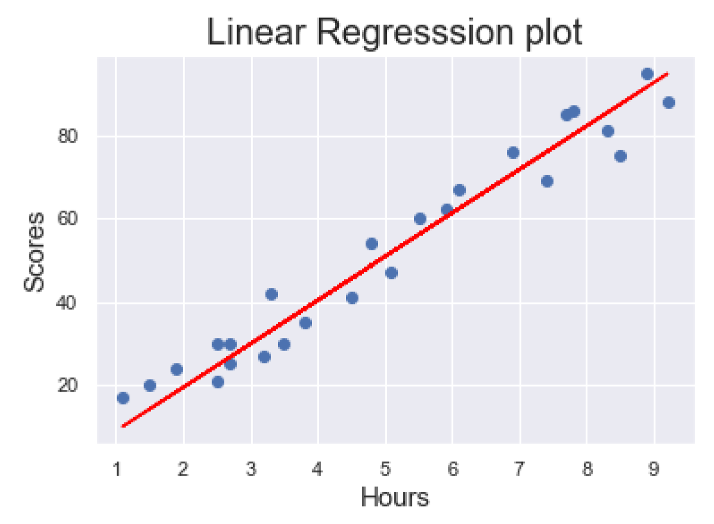
performing Regression

In [57]:
```
reg = LinearRegression()
reg.fit(x_train,y_train)
print("Model Training completed")
```

```
Model Training completed
```

In [59]:
```
#regression model
y_hat=reg.coef_*X + reg.intercept_

plt.scatter(X,Y)
plt.plot(X,y_hat,c="red")
plt.title("Linear Regresssion plot",size=20)
plt.xlabel("Hours",size=15)
plt.ylabel("Scores",size=15)
plt.show()
```



In [60]:
```
#visuallly we can see that the regression line fitting the data quite we
ll
```

Predicting values with model and compare with actual values

In [63]:
```
y_predict = reg.predict(x_test)

df = pd.DataFrame(y_test,columns=["Actual scores"])
df
```

Out[63]:

	Actual scores
0	17
1	42
2	24
3	75
4	54

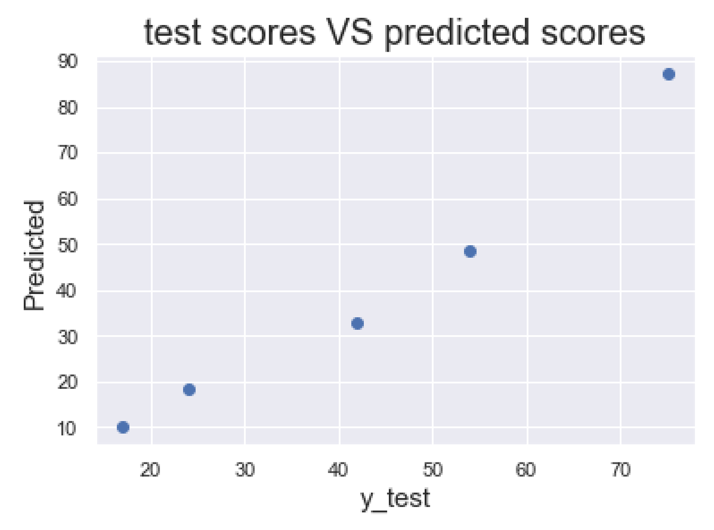
In [64]:
```
df["Predicted scores"] = y_predict
df
```

Out[64]:

	Actual scores	Predicted scores
0	17	9.970262
1	42	32.984700
2	24	18.339148
3	75	87.382463
4	54	48.676362

In [68]:
```
#plot test scores against predicted scores

plt.scatter(y_test,y_predict)
plt.title("test scores VS predicted scores",size=20)
plt.xlabel("y_test",size =15)
plt.ylabel("Predicted",size=15)
plt.show()
```



In [69]:
```
#predict value of given 9.25 hrs study time
hours=[[9.25]]
own_prediction = reg.predict(hours)
print("no. of hours = {}".format(hours))
print("Predicted score = {}".format(own_prediction))
```

```
no. of hours = [[9.25]]
Predicted score = [[95.22829438]]
```

Evaluating the model

In [70]:
```
#calculating R - squared value
r2 = reg.score(x_train,y_train)
print("R-square = ",r2)
```

```
R-square =  0.9637848283990599
```

In [71]:
```
#here the relation for data set is 96%.
#which means good corelation coefficient.
```

In [72]:
```
#calculating mean absolute error

from sklearn.metrics import mean_absolute_error
mae=mean_absolute_error(y_test,y_predict)
print("Mean absilute error = ",mae)
```

```
Mean absilute error =  7.882398086270432
```

In [74]:
```
#calclating root mean square error
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test,y_predict)
rmse =np.sqrt(mse)
print("Mean square error = ",mse)
print("Root mean square error = ",rmse)
```

```
Mean square error =  68.88092074277635
Root mean square error =  8.299453038771674
```

Thank You