In [ ]:

```python
# Author: Ram vriksh
```

GRIP @The Sparskfoundation

TASK 1:Prediction using supervised Machine learning

prediction of precentage of a student based on the number of study hours

this is simple linear regression task as it involve only two variables

In [1]:

```python
#import required libbraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

In [2]:

```python
#reading  data

data=pd.read_csv("student_scores - student_scores.csv")
print("read data successfully")
```

```
read data successfully
```

In [3]:

```python
data.head(8)
```

Out[3]:

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30
5	1.5	20
6	9.2	88
7	5.5	60

In [4]:

```python
data.shape
```

Out[4]:

```
(25, 2)
```

In [5]:

```python
data.dtypes
```

Out[5]:

```
Hours     float64
Scores      int64
dtype: object
```

In [6]:

```
data.describe()
```

Out[6]:

	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 [7]:

```
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]:

```
data.isnull().sum()
```

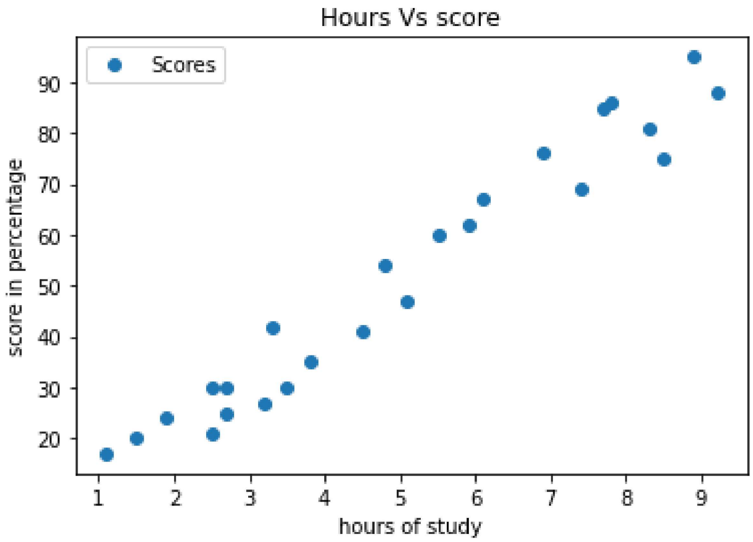
Out[8]:

```
Hours     0
Scores    0
dtype: int64
```

In [9]:

```
# plot the graph using data

data.plot(x='Hours',
          y='Scores',
          style= 'o')
plt.title('Hours Vs score')
plt.xlabel('hours of study')
plt.ylabel('score in percentage')
plt.show()
```



checking manually if the data have kind of relation using graph

from the above graph we can see clearly that the is linear relation between Hours and study

Split the data into training and test set

In [10]:

```
X=data.iloc[:,:-1].values
y=data.iloc[:,1].values
```

In [11]:

```
#split the data into train test set
X_train, X_test, y_train, y_test = train_test_split(
     X, y,
     test_size=0.3,
     random_state=100)
```

In [12]:

```
#check the co-relation
data.corr()
```

Out[12]:

	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

Train the model

In [13]:

```
#import linear model
#define a regression model and fit the data into it
from sklearn import linear_model
LR_model=linear_model.LinearRegression()

LR_model.fit(X_train,y_train)
```

Out[13]:

```
LinearRegression()
```

In [14]:

```
LR_model.intercept_
```

Out[14]:
```
1.495142109236383
```

In [15]:
```
LR_model.coef_
```

Out[15]:
```
array([9.87171443])
```

Predicting the marks using the model

In [21]:
```
y_pred=LR_model.predict(X_test)
y_pred
```

Out[21]:
```
array([28.14877107, 39.00765694, 34.07179972, 59.73825724, 16.30271375,
       74.54582888, 69.60997167, 48.87937137])
```

In [24]:
```
prediction=pd.DataFrame({'Hours':[i[0] for i in X_test],'Predictions':[k for k in y_tes
prediction
```

Out[24]:

	Hours	Predictions
0	2.7	25
1	3.8	35
2	3.3	42
3	5.9	62
4	1.5	20
5	7.4	69
6	6.9	76
7	4.8	54

In [22]:
```
y_test
```

Out[22]:
```
array([25, 35, 42, 62, 20, 69, 76, 54], dtype=int64)
```

In [26]:
```
#comparsion between actual and predicted marks
comparsion=pd.DataFrame({'predicted marks ':y_pred,'Actual marks':y_test})
comparsion
```

Out[26]:

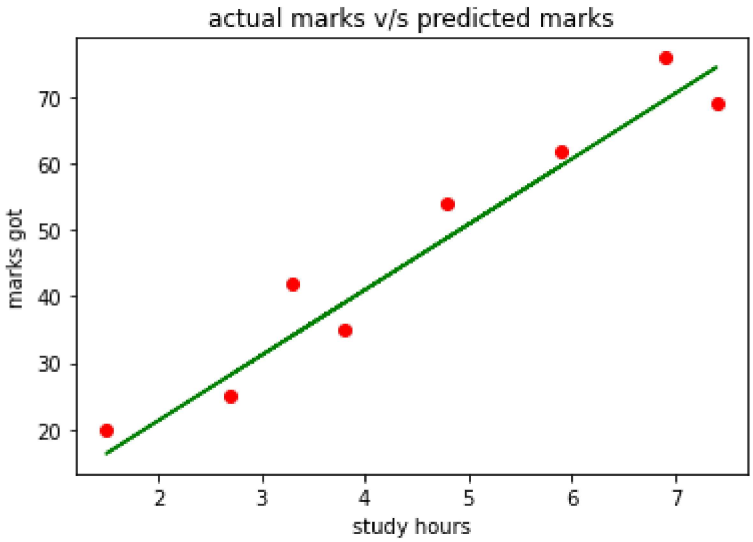
	predicted marks	Actual marks
0	28.148771	25
1	39.007657	35
2	34.071800	42
3	59.738257	62
4	16.302714	20
5	74.545829	69
6	69.609972	76

	predicted marks	Actual marks
7	48.879371	54

Visuallizing the comparsion between predicted and actual marks

In [27]:

```
plt.scatter(x=X_test,y=y_test,color='red')
plt.plot(X_test,y_pred,color='green')
plt.title('actual marks v/s predicted marks')
plt.xlabel('study hours')
plt.ylabel('marks got')
plt.show()
```



Evaluate the model

In [29]:

```
from sklearn.metrics import mean_absolute_error
print('mean absolute error in the model ',mean_absolute_error(y_pred,y_test))
```

```
mean absolute error in the model  4.762517892332275
```

find out the score if a student study for 9.25hrs/day

In [36]:

```
hours=[9.25]
marks_predicted=LR_model.predict([hours])
marks_predicted
```

Out[36]:

```
array([92.80850057])
```

by this model, we can say that if students study 9.25hrs/day he/she can get 92.80

precent marks

In [ ]: