Anisha Dhadge¶

Task 1: Prediction using Supervised ML¶

Predict the percentage of an student based on the no. of study hours.¶

This is a simple linear regression task as it involves just 2 variables.¶

Importing libraries¶

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import linear_model
import sklearn.metrics as sm
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

Loading data¶

In [2]:

```
df=pd.read_csv("A:\\msc1\\task1.csv")
```

In [3]:

```
df.head()
```

Out[3]:

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27

	Hours	Scores
3	8.5	75
4	3.5	30

In [4]:

```
df.columns
```

Out[4]:

```
Index(['Hours', 'Scores'], dtype='object')
```

In [5]:

```
df.shape
```

Out[5]:

```
(25, 2)
```

In [6]:

```
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]:

```
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

	Hours	Scores
75%	7.400000	75.000000
max	9.200000	95.000000

Checking for null values¶

In [8]:

```
df.isnull().sum()
```

Out[8]:

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

As we can see there are no null values

Data Visualization¶

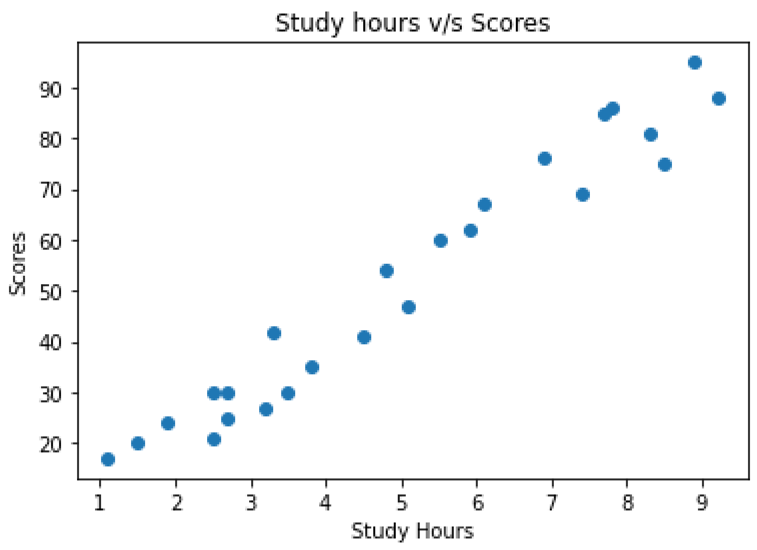
Scatter Plot¶

In [9]:

```
plt.scatter(df['Hours'],df['Scores'])
plt.title('Study hours v/s Scores')
plt.xlabel('Study Hours')
plt.ylabel('Scores')
```

Out[9]:

```
Text(0, 0.5, 'Scores')
```



The two variables Hours and Scores show positive linear relationship.

Preparing the data¶

Here we divide the data into feature and target variables

In [10]:

```
x=df.iloc[:, 0].values
y=df.iloc[:, 1].values
x=x.reshape(-1,1)
y=y.reshape(-1,1)
```

Splitting the data¶

Here we split data into training and testing sets

In [11]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

In [12]:

```
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(20, 1)
(20, 1)
(5, 1)
(5, 1)
```

Training of Model¶

We use Linear Regression Model

In [13]:

```
model=linear_model.LinearRegression()
```

In [14]:

```
model.fit(x_train,y_train)
print("Intercept: ",model.intercept_[0])
print("Coefficient: ",model.coef_[0][0])
```

```
Intercept:  2.2761104096145814
Coefficient:  9.669231813302295
```

Above are regression coefficients

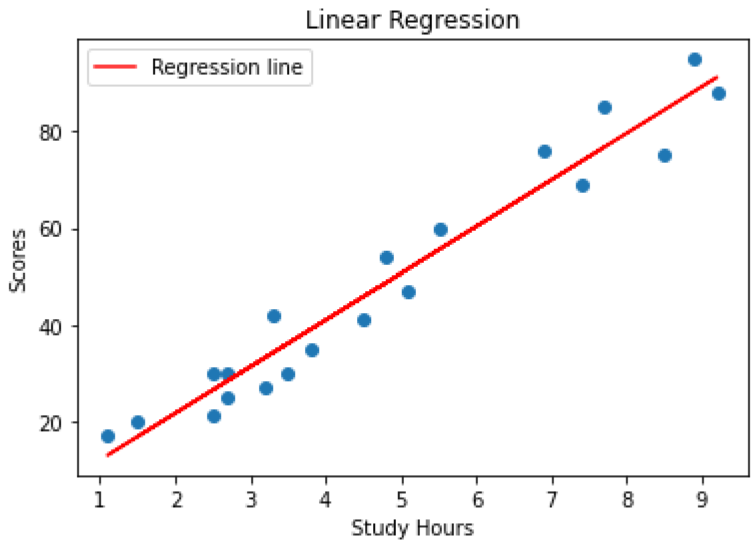
Regression line¶

In [15]:

```
plt.scatter(x_train,y_train)
plt.plot(x_train,model.intercept_+(x_train*model.coef_),color='red',label="Regression
line")
plt.legend()
plt.title('Linear Regression')
plt.xlabel('Study Hours')
plt.ylabel('Scores')
```

Out[15]:

```
Text(0, 0.5, 'Scores')
```



As we know the relationship between the two variables is positive linear relationship that is as the number of study hours increases the score increases

Predictions using testing set¶

In [16]:

```
y_pred=model.predict(x_test)
```

In [17]:

```
d1=pd.DataFrame(y_test)
d2=pd.DataFrame(y_pred)
```

Comparing actual and predicted values¶

In [18]:

```
d=pd.concat([d1,d2],axis=1)
d.columns=['Actual','Prediction']
d
```

Out[18]:

	Actual	Prediction
0	81	82.530734
1	62	59.324578
2	24	20.647651
3	86	77.696119

	Actual	Prediction
4	67	61.258424

What will be predicted score if a student studies for 9.25 hrs/ day?¶

In [21]:

```
pred=model.predict([[9.25]])
print("Predicted score if a student studies for 9.25 hrs/ day is: ",pred[0][0])
```

```
Predicted score if a student studies for 9.25 hrs/ day is:  91.7165046826608
```

Accuracy of model¶

In [20]:

```
print("Mean Absolute Error: ",metrics.mean_absolute_error(y_test,y_pred))
print("Mean Squared Error: ",metrics.mean_squared_error(y_test,y_pred))
print("R-squared: ",metrics.r2_score(y_test,y_pred))
```

```
Mean Absolute Error:  4.320792494581102
Mean Squared Error:  24.531882343025423
R-squared:  0.9485920319718663
```

R-squared reveals that 98% of the data fit the regression model.

Thank you¶