

**AIM: IMPLEMENTATION OF LINEAR REGRESSION & LOGISTIC REGRESSION****A) LINEAR REGRESSION:****THEORY:**

Linear regression is a basic and commonly used type of predictive analysis. The overall idea of regression is to examine two things:

1. does a set of predictor variables do a good job in predicting an outcome (dependent) variable?
2. Which variables in particular are significant predictors of the outcome variable, and in what way do they—indicated by the magnitude and sign of the beta estimates—impact the outcome variable?

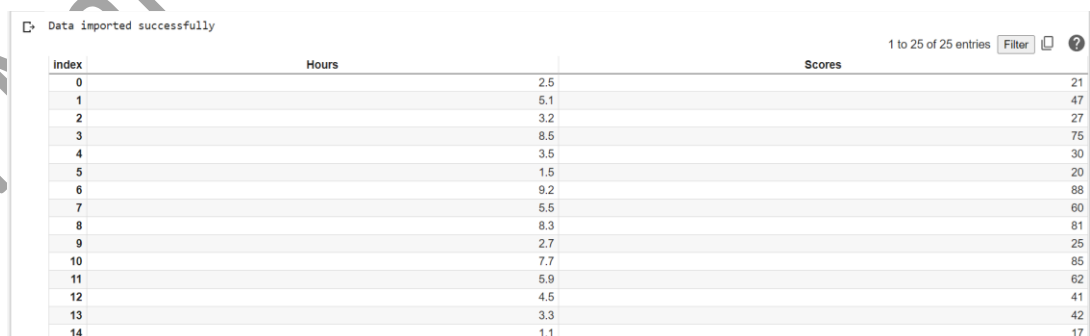
These regression estimates are used to explain the relationship between one dependent variable and one or more independent variables. The simplest form of the regression equation with one dependent and one independent variable is defined by the formula  $y = c + b \cdot x$ , where  $y$  = estimated dependent variable score,  $c$  = constant,  $b$  = regression coefficient, and  $x$  = score on the independent variable

**SOURCE CODE & OUTPUT:****1) IMPORTING LIBRARIES:**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

**2) COLLECTING DATA:**

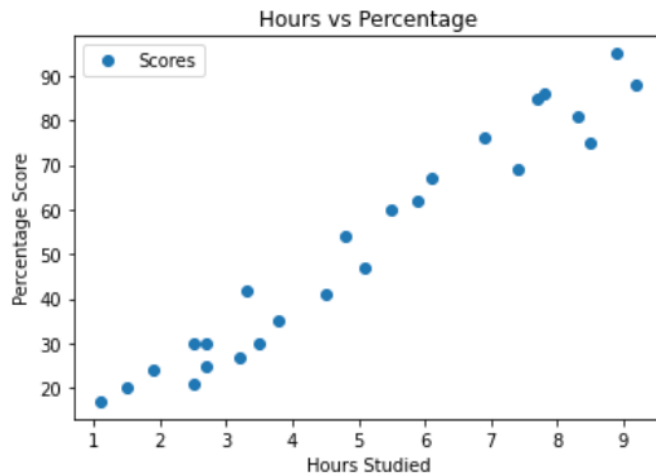
```
url = "http://bit.ly/w-data"
data = pd.read_csv(url)
print("Data imported successfully")
data
```



index	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
8	8.3	81
9	2.7	25
10	7.7	85
11	5.9	62
12	4.5	41
13	3.3	42
14	1.1	17

**3) Plotting the distribution of scores:**

```
data.plot(x='Hours', y='Scores', style='o')
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```



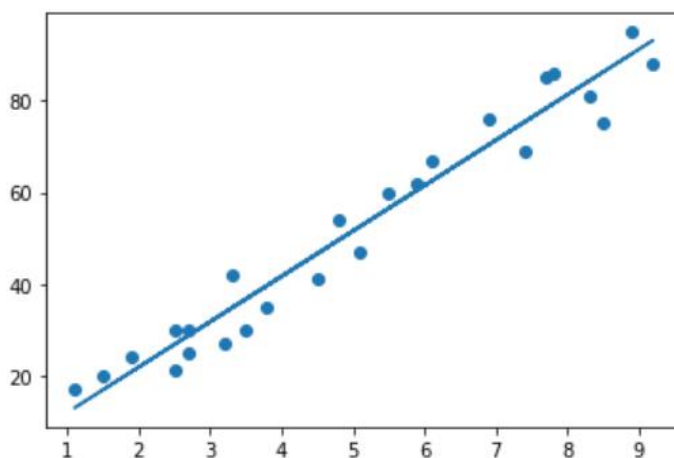
#### 4) TRAINING DATA:

```
#Training Data
X = data.iloc[:, :-1].values
y = data.iloc[:, 1].values
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)
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
print("Training complete.")
```

Training complete.

#### 5) PLOTTING THE REGRESSION LINE:

```
line = regressor.coef_*X+regressor.intercept_
# Plotting for the test data
plt.scatter(X, y)
plt.plot(X, line);
plt.show()
```



#### 6) PREDICTING THE SCORES:

```
y_pred = regressor.predict(X_test)
y_pred
```

```
array([16.88414476, 33.73226078, 75.357018 , 26.79480124, 60.49103328])
```

**7) COMPARING ACTUAL VS PREDICTED:**

```
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
```

```
df
```

Index	Actual	Predicted
0	20	16.884144762398023
1	27	33.732260779489835
2	69	75.35701799818725
3	30	26.79480124304026
4	62	60.491033277223885

Show 25 per page  
Like what you see? Visit the [data table notebook](#) to learn more about interactive tables.

**8) EVALUATING THE ALGORITHM:**

```
from sklearn import metrics
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('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred))
)
```

Mean Absolute Error: 4.183859899002982

Mean Squared Error: 21.598769307217456

Root Mean Squared Error: 4.647447612100373

**B) LOGISTIC REGRESSION:****THEORY:**

In statistics, the logistic model (or logit model) is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object being detected in the image would be assigned a probability between 0 and 1, with a sum of one.

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression[1] (or logit regression) is estimating the parameters of a logistic model (a form of binary regression). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labeled "0" and "1".

**SOURCE CODE & OUTPUT:****1) COLLECTING DATA:**

```
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd

data_set= pd.read_csv('suv_data.csv')
data_set
```

1 to 25 of 400 entries						
index	User ID	Gender	Age	EstimatedSalary	Purchased	
0	15624510	Male	19	19000	0	
1	15810944	Male	35	20000	0	
2	15668575	Female	26	43000	0	
3	15603246	Female	27	57000	0	
4	15804002	Male	19	76000	0	
5	15728773	Male	27	58000	0	
6	15598044	Female	27	84000	0	
7	15694829	Female	32	150000	1	
8	15600575	Male	25	33000	0	
9	15727311	Female	35	65000	0	
10	15570769	Female	26	80000	0	
11	15606274	Female	26	52000	0	
12	15746139	Male	20	86000	0	
13	15704987	Male	32	18000	0	
14	15628972	Male	18	82000	0	

**2) EXTRACTING INDEPENDENT AND DEPENDENT VARIABLE:**

```
x = data_set.iloc[:, [2,3]].values  
y = data_set.iloc[:, 4].values
```

**3) SPLITTING THE DATASET INTO TRAINING 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= 0.25, random_state=0)
```

**4) FEATURE SCALING:**

```
from sklearn.preprocessing import StandardScaler  
st_x= StandardScaler()  
x_train= st_x.fit_transform(x_train)  
x_test= st_x.transform(x_test)
```

**5) FITTING LOGISTIC REGRESSION TO THE TRAINING SET:**

```
from sklearn.linear_model import LogisticRegression  
classifier= LogisticRegression(random_state=0)  
classifier.fit(x_train, y_train)
```

**6) PREDICTING THE TEST SET RESULT:**

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

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,  
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,  
       1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,  
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1,  
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1])
```

**7) CREATING THE CONFUSION MATRIX:**

```
from sklearn.metrics import confusion_matrix  
cm = confusion_matrix(y_test, y_pred)  
cm
```

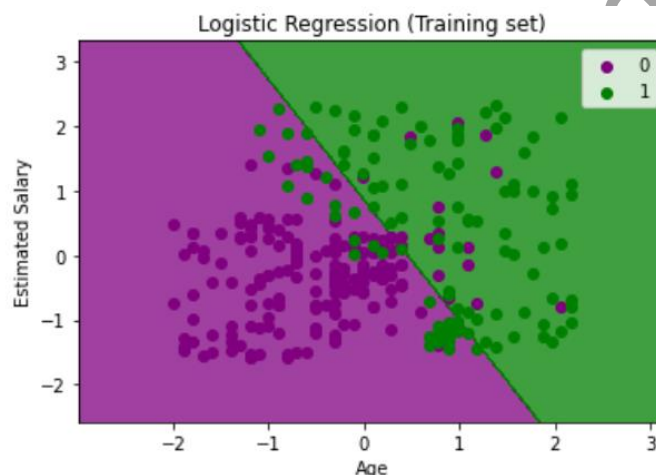
```
array([[65,  3],  
       [ 8, 24]])
```

**8) VISUALIZING THE TRAINING SET RESULT:**

```
from matplotlib.colors import ListedColormap
```

```
x_set, y_set = x_train, y_train
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step=0.01),
                    nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01)
)

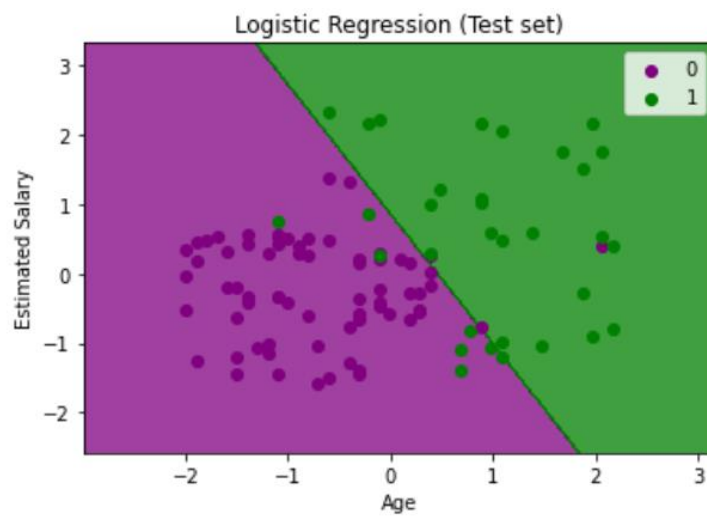
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
             alpha = 0.75, cmap = ListedColormap(('purple','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
               c = ListedColormap(('purple', 'green'))(i), label = j)
mtp.title('Logistic Regression (Training set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```



#### 9) VISUALIZING THE TEST SET RESULT:

```
from matplotlib.colors import ListedColormap
x_set, y_set = x_test, y_test
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step=0.01),
                    nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01)
)
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape), alpha = 0.75, cmap = ListedColormap(('purple','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
               c = ListedColormap(('purple', 'green'))(i), label = j)
mtp.title('Logistic Regression (Test set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
```

mtp.show()



**CONCLUSION:**

From this practical, I have learned about Implementation of Linear and logistic regression in python.