

## EXPERIMENT 3

ADITHIYA V

241501010

### A PYTHON PROGRAM TO IMPLEMENT LOGISTIC MODEL

#### AIM:

*TO IMPLEMENT A PYTHON PROGRAM WITH LOGISTIC MODEL*

#### CODE:

```
import pandas as pd
import numpy as np
from numpy import log,dot,exp,shape
from sklearn.metrics import confusion_matrix
data = pd.read_csv('/content/suv_data.csv')
print(data.head())

x = data.iloc[:, [2, 3]].values
y = data.iloc[:, 4].values
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test=train_test_split(x,y,test_size=0.10,
random_state=0)
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
print (x_train[0:10,:])

from sklearn.linear_model import LogisticRegression
classifier=LogisticRegression(random_state=0)
classifier.fit(x_train,y_train)
LogisticRegression (random_state=0)
y_pred = classifier.predict(x_test)
print(y_pred)

from sklearn.metrics import confusion_matrix
```

```
cm = confusion_matrix(y_test, y_pred)
print ("Confusion Matrix : \n", cm)
```

```
from sklearn.metrics import accuracy_score
print ("Accuracy : ", accuracy_score(y_test, y_pred))
```

*# User Defined function*

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test=train_test_split(x,y,test_size=0.10,
random_state=0)
```

```
def Std(input_data):
    mean0 = np.mean(input_data[:, 0])
    sd0 = np.std(input_data[:, 0])
    mean1 = np.mean(input_data[:, 1])
    sd1 = np.std(input_data[:, 1])
    return lambda x: ((x[0]-mean0)/sd0, (x[1]-mean1)/sd1)
```

```
my_std = Std(x)
my_std(x_train[0])
```

```
def standardize(X_tr):
    for i in range(shape(X_tr)[1]):
        X_tr[:,i] = (X_tr[:,i] - np.mean(X_tr[:,i]))/np.std(X_tr[:,i])
```

```
def F1_score(y,y_hat):
    tp,tn,fp,fn = 0,0,0,0
    for i in range(len(y)):
        if y[i] == 1 and y_hat[i] == 1:
            tp += 1
        elif y[i] == 1 and y_hat[i] == 0:
            fn += 1
        elif y[i] == 0 and y_hat[i] == 1:
            fp += 1
        elif y[i] == 0 and y_hat[i] == 0:
            tn += 1
    precision = tp/(tp+fp)
```

```

recall = tp/(tp+fn)
f1_score = 2*precision*recall/(precision+recall)
return f1_score
class LogisticRegression:
    def sigmoid(self,z):
        sig = 1/(1+exp(-z))
        return sig
    def initialize(self,X):
        weights = np.zeros((shape(X)[1]+1,1))
        X = np.c_[np.ones((shape(X)[0],1)),X]
        return weights,X
    def fit(self,X,y,alpha=0.001,iter=400):
        weights,X = self.initialize(X)

        def cost(theta):
            z = dot(X,theta)
            cost0 = y.T.dot(log(self.sigmoid(z)))
            cost1 = (1-y).T.dot(log(1-self.sigmoid(z)))
            cost = -((cost1 + cost0))/len(y)
            return cost
        cost_list = np.zeros(iter,)
        for i in range(iter):
            weights = weights - alpha*dot(X.T,self.sigmoid(dot(X,weights))-
np.reshape(y,(len(y),1)))
            cost_list[i] = cost(weights)
            self.weights = weights
        return cost_list
    def predict(self,X):
        z = dot(self.initialize(X)[1],self.weights)
        lis = []
        for i in self.sigmoid(z):
            if i>0.5:
                lis.append(1)
            else:
                lis.append(0)
        return lis
standardize(x_train)

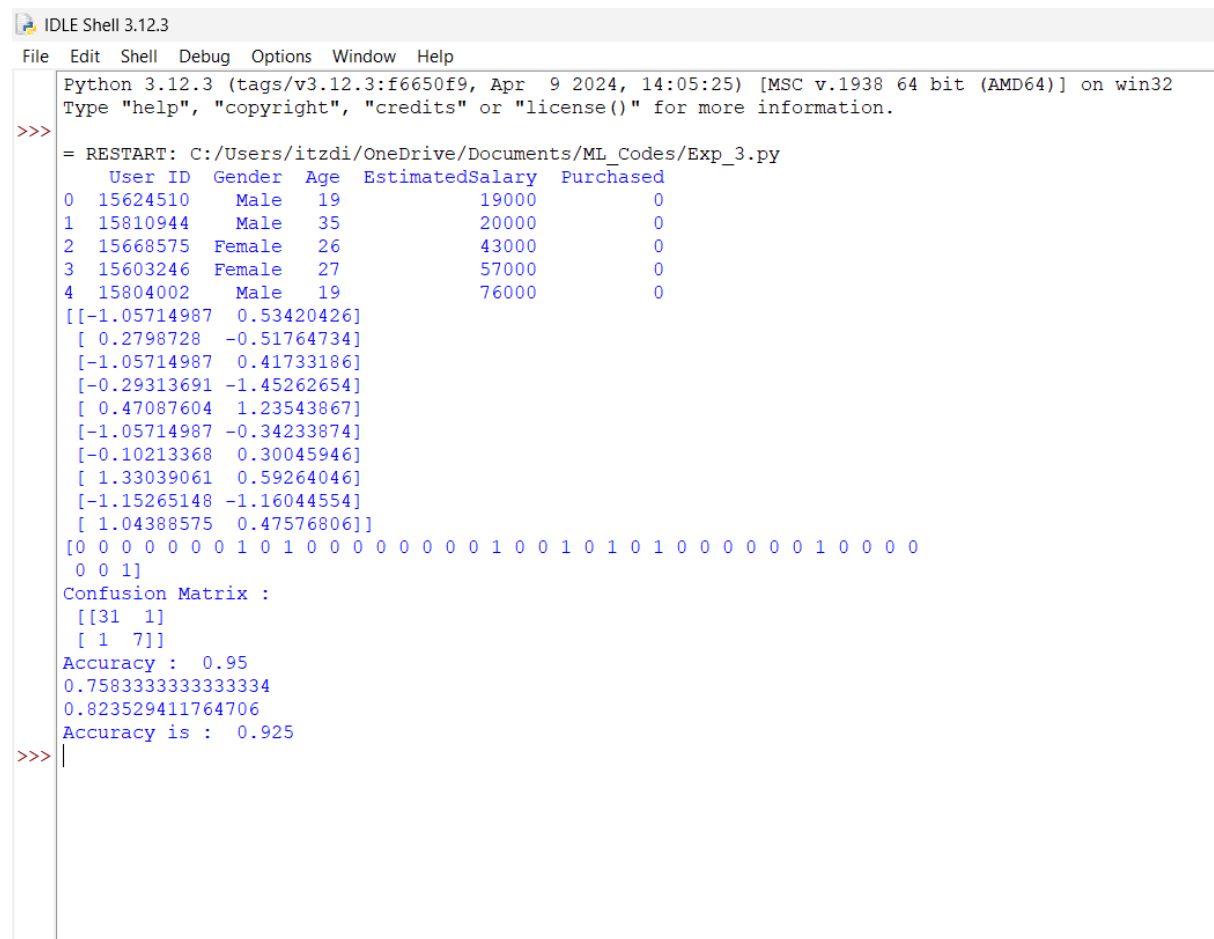
```

```

standardize(x_test)
obj1 = LogisticRegression()
model= obj1.fit(x_train,y_train)
y_pred = obj1.predict(x_test)
y_trainn = obj1.predict(x_train)
f1_score_tr = F1_score(y_train,y_trainn)
f1_score_te = F1_score(y_test,y_pred)
print(f1_score_tr)
print(f1_score_te)
conf_mat = confusion_matrix(y_test, y_pred)
accuracy = (conf_mat[0, 0] + conf_mat[1, 1]) / sum(sum(conf_mat))
print("Accuracy is : ",accuracy)

```

## OUTPUT:



```

IDLE Shell 3.12.3
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/itzdi/OneDrive/Documents/ML_Codes/Exp_3.py
  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
[[-1.05714987  0.53420426]
 [ 0.2798728  -0.51764734]
 [-1.05714987  0.41733186]
 [-0.29313691 -1.45262654]
 [ 0.47087604  1.23543867]
 [-1.05714987 -0.34233874]
 [-0.10213368  0.30045946]
 [ 1.33039061  0.59264046]
 [-1.15265148 -1.16044554]
 [ 1.04388575  0.47576806]]
[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 1 0 0 0 0
 0 0 1]
Confusion Matrix :
[[31  1]
 [ 1  7]]
Accuracy : 0.95
0.7583333333333334
0.823529411764706
Accuracy is : 0.925
>>>

```

**RESULT:**

*A PYTHON PROGRAM TO IMPLEMENT SIMPLE LINEAR REGRESSION USING  
LEAST SQUARE METHOD AS BEEN ANALYSED AND VERIFIED*