EXPERIMENT -3

A python program to implement logistic model

AIM:

To code a python program to implement logistic model.

CODE 1:

```
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())
```

OUTPUT 1:

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

CODE 2:

```
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, :])
```

OUTPUT 2:

```
[[-0.843 -0.820]

 [ 1.012  1.547]

 [-0.472 -0.579]

 [ 0.478  0.321]

 [-1.022 -1.215]
```

[-0.142 -0.117]

[1.254 1.843]

[-0.766 -0.703]

[0.339 0.199]

[-1.094 -0.940]]

CODE 3:

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

OUTPUT 3:

[0 0 0 1 0 0 1 0 0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 0 0]

CODE 4:

```
from sklearn.metrics import confusion_matrix,
accuracy_score
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix : \n", cm)
print("Accuracy : ", accuracy_score(y_test, y_pred))
```

OUTPUT 4:

Confusion Matrix :

[[23 2]

[3 12]]

Accuracy : 0.875

CODE 5:

```
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)
print(my_std(x_train[0]))
```

OUTPUT 5:

(-0.47, -0.58)

CODE 6:

```
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):
        return 1 / (1 + exp(-z))
    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)))
            return -((cost1 + cost0)) / len(y)
        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):
            lis.append(1 if i > 0.5 else 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 6:

0.888888888888889

0.8571428571428571

Accuracy is : 0.875

RESULT:

Thus a python program to implement logistic model is coded and the output is verified successfully.