Ex no: 3

A PYTHON PROGRAM TO IMPLEMENT THE LOGISTIC MODEL

Aim:

To implement a Python program for the logistic model using the.

Algorithm:

Step 1: Import Necessary Libraries:

● pandas for data manipulation

● sklearn.model_selection for train-test split

● sklearn. preprocessing for data preprocessing

● sklearn.linear_model for logistic regression

● matplotlib.pyplot for

plotting Step 2: Read the Dataset:

● Use pandas to read the dataset.csv dataset into a DataFrame.

Step 3: Preprocess the Data:

● Select the relevant columns for the analysis (e.g., 'Age', 'EstimatedSalary', 'Purchased').

● Encode categorical variables if necessary (e.g., using LabelEncoder or OneHotEncoder).

● Split the data into features (X) and the target variable (y).

Step 4: Split the Data:

● Split the dataset into training and testing sets using

train_test_split. Step 5: Feature Scaling:

● Standardize the features using StandardScaler to ensure they have the same

scale. Step 6: Create and Train the Model:

● Create a logistic regression model using LogisticRegression from sklearn.linear_model.

● Train the model on the training data using the fit method.

o Create a function named “Sigmoid ()” which will define the sigmoid values using the

o formula (1/1+e-z) and return the computed value.

o Create a function named “initialize()” which will initialize the values with zeroes and assign the value to the “weights” variable, initialize with ones and assign the value to the variable “x”, and return both “x” and “weights”.

o Create a function named “fit” which will be used to plot the graph according to the training data.

o Create a predict function that will predict values according to the training model created using the fit function.

o Invoke the standardize() function for “x-train” and “x-

test” Step 7: Make Predictions:

● Use the trained model to make predictions on the test data using the predict method.

o Use the “predict()” function to predict the values of the testing data and assign the value to the “y_pred” variable.

o Use the “predict()” function to predict the values of the training data and assign the value to “y_train” variable.

o Compute f1_score for both the training and testing data and assign the values to “f1_score_tr” and “f1_score_te” respectively

Step 8: Evaluate the Model:

● Calculate the accuracy of the model on the test data using the score method. (Accuracy = (tp+tn)/(tp+tn+fp+fn)).

● Generate a confusion matrix and classification report to further evaluate the model's performance.

Step 9: Visualize the Results:

● Plot the decision boundary of the logistic regression model (optional).

PROGRAM :

```
import pandas as pd
import numpy as np
from numpy import log, dot, exp, shape
from sklearn.metrics import confusion_matrix
# After you upload suv_data.csv manually in Colab, just use filename directly
data = pd.read_csv('suv_data - 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)
y_pred = classifier.predict(x_test)
print(y_pred)
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

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 LogisticRegressionCustom:
    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)
        return [1 if i > 0.5 else 0 for i in self.sigmoid(z)]
standardize(x_train)
standardize(x_test)
obj1 = LogisticRegressionCustom()
model = obj1.fit(x_train, y_train)
```

y_pred_custom = obj1.predict(x_test)

y_train_pred_custom = obj1.predict(x_train)

f1_score_tr = F1_score(y_train, y_train_pred_custom)

f1_score_te = F1_score(y_test, y_pred_custom)

print(f1_score_tr)

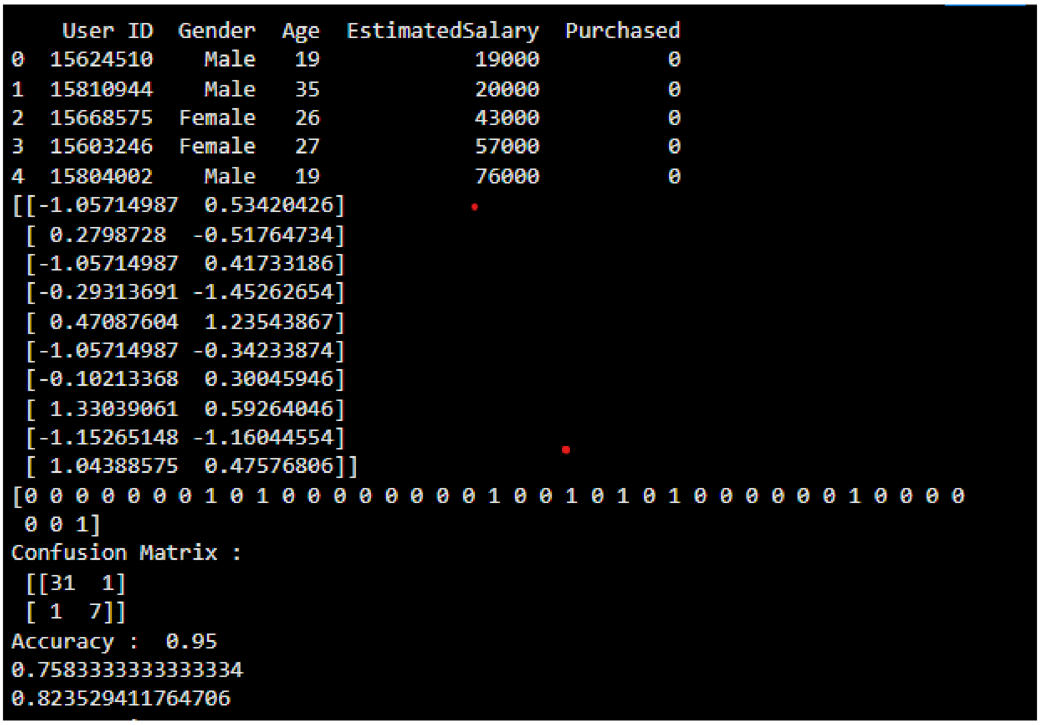
print(f1_score_te)

conf_mat = confusion_matrix(y_test, y_pred_custom)

accuracy = (conf_mat[0, 0] + conf_mat[1, 1]) / np.sum(conf_mat)

print("Accuracy is : ", accuracy)

OUTPUT :



```
    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 0 1 0 1 0 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 0 1]
Confusion Matrix :
 [[31  1]
 [ 1  7]]
Accuracy :  0.95
0.7583333333333334
0.823529411764706
```

RESULT:-

Thus, the Python program to implement logistic regression for the given dataset is analyzed, and thelogistic regression model is classified successfully. The performance of the developed model is measured using the F1-score and Accuracy.