A Python program to implement logistic model

Aim:

To implement Python program for the logistic using suv car dataset.

Algorithm:

Step 1: Import Necessary libraries.

- Pandas for data manipulation.
- Sklearn model selection for train-test split.
- Sklearn.preprocessing for data preprocessing.
- & matplotlib.pyplot for plotting.

Step 2: Read the Dataset:

- Use pandas to read the SOV-Cars.csv dataset into a DataFrame.

Step 3: Preprocess the Data:

- Select the relevant columns for the analysis (eg: 'Age', 'EstimatedSalary', 'Purchased').
- Encode categorical variables if necessary (eg: using LabelEncoder or one HotEncoder).
- Split the data into features (x) and target variable (y).

Step 4: Split the Data:

* Split the dataset into training and testing sets using train_test_split.

Step5: Feature Scaling:

* Standardize the features using Standard Scale to ensure they have the same scale.

Step6: Create and train the model:

* Create a logistic regression model using LogisticRegression from Sklearn.linear_model.

Step7: Make predictions:

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

=> use the 'predict() function to predict method. values of the toting data and assign the values to "y_pred" variable.

	User ID	Gender	Age	Salary	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 -1.45265652]
 [ 0.47087604  1.23543867]
 [-1.05714987 -0.34233784]
 [ 1.33039061  0.59264046]
 [ 1.04388515  0.47576806]]
```

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
data = pd.read_csv('../input/suvcsvs/suv_data.csv')
Print(data.head())
x = data.iloc[:, [2,3]].values
y = data.iloc[:, 4].values
```

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

Confusion matrix:

[[31 1]
 [1 7]]

Accuracy: 0.95

(-1.0176923932473028, 0.5361288590822568)

```
from sklearn.model_selection import train_test_split

Random:
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))
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)
```

```
mean 0 = np. mean (input - data[:, 0])
Sol 0 = np. std (input - data[:, 0])
mean 1 = np. mean (input - data[:, 1])
Sol 1 = np. mean (input - data[:, 1])
return lambda x: ([x[0] - mean 0) / sol 0, (x[1] - mean 1) /
                                                std 1)

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:
            tn += 1
        elif y[i] == 0 and y_hat[i] == 1:
            fp += 1
    precision = tp / (tp + fp)
    recall = tp / (tp + fn)
```

```
f1_score = 2 * precision * recall / (precision + recall)

return f1_score

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 = 0.0001, 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.zero(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)[:,], 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_trainna = obj1.predict(x_train)
f1_score_tr = F1_Score(y_train, y_train)
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)
```

Result:

Thus, the python program to implement logistic regression for the given suv_coors dataset is analyzed and the logistic regression model is classifies successfully. The performance of the developed model is measured using F_1-score and accuracy.