EXP.NO: 3. A PYTHON PROGRAM TO IMPLEMENT
Date : LOGISTIC MODEL

Aim:

To implement python program for the logistic model using suv car dataset.

Algorithm:

Step 1 : Import necessary library:

* Pandas for data manipulation.
* Sklearn.model_selection for train_test split
* Sklearn.preprocessing for data preprocessing
* Sklearn.linear_model for logistic regression.

Step 2: Read the dataset :

use pandas to read the suv_cars.csv dataset into a Dataframe.

Step 3: Split The Data :

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

Step 4: Feature scaling:

Standardize the features using StandardScaler to ensure they have the same scale.

Step 5: create and train the model :

create a logistic regression model using LogisticRegressi from sklearn_model.

Step 7: Make Prediction:

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

* Use the "predict()" function to predict the value of the testing data and assign the value to "y-pred" variable.

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))

Step 9: Visualize the Results:

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

[[-1.05714987 0.53420426]
...
[1.04388575 0.47576806]].

Confusion Matrix:

[[31 1]
[1 7]].

Accuracy: 0.95

0.7583333333333334

0.8235294117647706

Accuracy is: 0.925.

Result:

Thus the Python program for the given sun-cars dataset is executed and the logistic regression model is classified successfully. The performance of the developed model is measured using F1-score and Accuracy.

19/2

```
import numpy as np
from numpy import log, dot, exp, shape.
from sklearn.metrics import confusion_matrix.
data = pd.read_csv('../input/suvcars/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_
sc = StandardScaler();
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
Print(x_train[0:10, :]).
Classifier = LogisticRegression(random_state=0)
Classifier.fit(x_train, y_train)
LogisticRegression(random_state=0)
y_pred = classifier.predict(x_test)
Print(y_pred)

cm = confusion_matrix(y_test, y_pred)
Print("Confusion Matrix:\n", cm)
Print("Accuracy:", accuracy_score(y_test, y_pred))
```

User Defined function

```
x_train, x_test, y_train, y_test = train_split
(x, y, test_size = 0.10)
```

```
mean 0 = np.mean (input_data [:, 0])
sd 0 = np.std (input_data [:, 0])
mean 1 = np.mean (input_data [:, 1])
sd 1 = np.std (input_data [:, 1])
my_std = Std (x)
my_std (x_train [0])
```

```
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)
obj 1 = LogisticRegression()
model = obj 1.fit (x_train, y_train)
y_pred = obj 1.Predict (x_test)
y_trainn = obj1.predict (x_train).
f1_score_tr = F1_score (y_train, y_train)
f1_score_te = F1_score (y_test, y_pred)
Print ("Accuracy is :", accuracy).
```

	User ID	Gender	Age	Salary	Purchased
0					
1	15624510	Male	19	19000	0
2	...	...	...	...	...
3	...	...	...	...	...
4	...	...	...	...	...