DSS LAB

EXPERIMENT NO. 8

Aim: Implementation of Logistic Regression and its Performance Evaluation on dataset

Software Required: Python 3

Theory:

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.

Logistic regression predicts the output of a categorical dependent variable. Therefore, the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

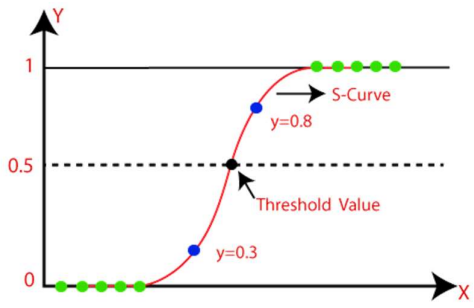
Logistic Regression is much similar to Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.

In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).

The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.

Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.

Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:



Implementation:

Python Implementation of Logistic Regression (Binomial):

To understand the implementation of Logistic Regression in Python, we will use the below example:

Example: There is a dataset given which contains the information of various users obtained from social networking sites. There is a car-making company that has recently launched a new SUV car. So, the company wanted to check how many users from the dataset,wants to purchase the car.

For this problem, we will build a Machine Learning model using the Logistic regression algorithm. The dataset is shown in the below image. In this problem, we will predict the purchased variable (Dependent Variable) by using 'age' and 'salary' (Independent variables).

Steps in Logistic Regression: To implement the Logistic Regression using Python, we will use the same steps as we have done in previous topics of Regression.

Below are the steps:

- Data Pre-processing step
- Fitting Logistic Regression to the Training set
- Predicting the test result
- Test accuracy of the result (Creation of Confusion matrix)
- Visualizing the test set result.

1. Data Pre-processing step: In this step, we will pre-process/prepare the data so that we can use it in our code efficiently. It will be the same as we have done in Data pre-processing topic. The code for this is given below:

```
# importing libraries
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd

#importing datasets
data_set= pd.read_csv('user_data.csv')
data_set
```

By executing the above lines of code, we will get the dataset as the 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
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

400 rows × 5 columns

Now, we will extract the dependent and independent variables from the given dataset. Below is the code for it:

#Extracting Independent and dependent Variable

```
x= data_set.iloc[:, [2,3]].values
y= data_set.iloc[:, 4].values
```

Now we will split the dataset into a training set and test set. Below is the code for it:

 # Splitting the dataset into training and test set.

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_state=0)
```

In logistic regression, we will do feature scaling because we want the accurate result of predictions. Here we will only scale the independent variable because the dependent variable has only 0 and 1 values. Below is the code for it:

#feature Scaling

```
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

2. Fitting Logistic Regression to the Training set:

We have well prepared our dataset, and now we will train the dataset using the training set. For providing training or fitting the model to the training set, we will import the **LogisticRegression** class of the **sklearn** library.

After importing the class, we will create a classifier object and use it to fit the model to the logistic regression. Below is the code for it:

```
#Fitting Logistic Regression to the training set
from sklearn.linear_model import LogisticRegression
classifier= LogisticRegression(random_state=0)
classifier.fit(x_train, y_train)
```

3. Predicting the Test & Train set Results

The model is well-trained on the training set, so we will now predict the result by using test & train set data.
Below is the code for it:

```
#Predicting the test set result
y_pred = classifier.predict(x_test)

#Predicting the training set result

y_pred_train = classifier.predict(x_train)
```

4. Testing and Training Data Accuracy of the result

Testing and Training Data Accuracy of the result is measured on the basis of the following parameters;

- Accuracy
- Precision
- Recall
- F1- Score
- Confusion Matrix

```
#Testing Data Evaluation
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, roc_curve

#accuracy score
acc_score=accuracy_score(y_test,y_pred)
print('Accuracy Score:',acc_score)
print('*'*65)
```

#Creating the Confusion matrix

```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test,y_pred)
print('Confusion Matrix\n',cm)
print('*'*65)
```

#Classification Report

```
clf_report=classification_report(y_test,y_pred)
print('CR',clf_report)
```

Output:

```
Accuracy Score: 0.89
*****************************************************************
Confusion Matrix
 [[65  3]
 [ 8 24]]
*****************************************************************
CR              precision    recall  f1-score   support

           0       0.89      0.96      0.92        68
           1       0.89      0.75      0.81        32

    accuracy                           0.89       100
   macro avg       0.89      0.85      0.87       100
weighted avg       0.89      0.89      0.89       100
```

#Training Data Evaluation

```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, roc_curve
```

#accuracy score

```
acc_score=accuracy_score(y_train,y_pred_train)
print('Accuracy Score:',acc_score)
print('*'*65)
```

#Creating the Confusion matrix

```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_train,y_pred_train)
print('Confusion Matrix\n',cm)
print('*'*65)
```

#Classification Report

```
clf_report=classification_report(y_train,y_pred_train)
print('CR', clf_report)
```

Output:

```
Accuracy Score: 0.8233333333333334
**********************************************************************
Confusion Matrix
 [[172  17]
 [ 36  75]]
**********************************************************************
CR               precision    recall  f1-score   support

           0       0.83      0.91      0.87       189
           1       0.82      0.68      0.74       111

    accuracy                           0.82       300
   macro avg       0.82      0.79      0.80       300
weighted avg       0.82      0.82      0.82       300
```

Conclusion:

In this way, we have studied and implemented, and evaluated the Logistic Regression model.