Branch: - Computer Science and Engineering **Class: - III Year**

Subject: - Big Data Analytics Lab **Sem: - VI**

Teacher Manual

PRACTICAL NO. 5

Aim: Write a program to implement Logistic regression.

Software Requirement: Jupyter

Theory:

Logistic Regression in Machine Learning:

- 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 valueswhich lie between 0 and 1**.**
- Logistic Regression is much similar to the 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:

Logistic Function (Sigmoid Function):

- The sigmoid function is a mathematical function used to map the predicted values to

probabilities.

- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Assumptions for Logistic Regression:

- The dependent variable must be categorical in nature.
- The independent variable should not have multi-collinearity.

Logistic Regression Equation:

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

- We know the equation of the straight line can be written as:
- In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):
- But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

The above equation is the final equation for Logistic Regression.

Type of Logistic Regression:

On the basis of the categories, Logistic Regression can be classified into three types:

- **Binomial**: In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.
- **Multinomial**: In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"
- **Ordinal**: In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

Program:

- Here, we will call some basic and important libraries to work.

```
import pandas as pd
```

```
import numpy as np
import matplotlib.pyplot as plt
```

- load the file into DataFrame object

```
df=pd.read_csv('Filename.csv')
```

- Head method shows us only the first 5 Rows

```
df.head()
```

- To get column names

```
df.columns
```

```
X = df[['column_name1', 'column_name2', 'column_name3',…., 'column_nameN']]

y = df['column_nameN+1']
```

- To draw a scatter plot we used scatter() function, scatter() function plots one dot for each observation. It needs two arrays of the same length, one for the values of the x-axis, and one for values on the y-axis

```
plt.scatter(df.column_name,df.column_nameN+1,marker='+',color='red')
```

- **Split arrays or matrices into random train and test subsets**, so we need to import **train_test_split from sklearn package**.

```
from sklearn.model_selection import train_test_split
```

- Next, we split 90% of the data to the training set while 10% of the data to test set using below code.
- The test_size variable is where we actually specify the proportion of the test set.

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.1)
```

- After splitting the data into training and testing sets, finally, the time is to train our algorithm. For that, we need to import Logistic Regression class. To use **Logistic Regression class we need to call it from sklearn package**.

```
from sklearn.linear_model import LogisticRegression
```

- Create object of Logistic Regression and call the fit() method along with our training data.

```
Model_LR = LogisticRegression()
Model_LR.fit(X_train,y_train)
```

- Now that we have trained our algorithm, it's time to make some predictions. To do so, we will use our test data and see how accurately our algorithm predicts the percentage score. To make predictions on the test data, execute the following script

```
Model_LR.predict(X_test)
```

```
Model_LR.score(X_test,y_test)
```

Result: