Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)

Subject: Machine Learning – I (DJ19DSC402)

AY: 2022-23

Experiment 6

(Logistic Regression)

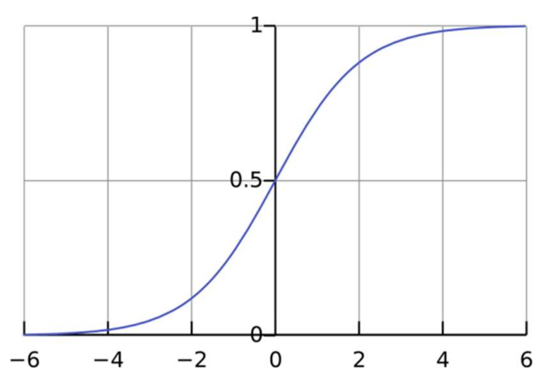
Name: Ayush Jain **SAP ID**: 60004200132

Aim: Implement Logistic Regression on a given Dataset with binary and multiclass labels.

Theory:

Logistic Regression is a statistical approach and a Machine Learning algorithm that is used for classification problems and is based on the concept of probability. It is used when the dependent variable (target) is categorical. It is widely used when the classification problem at hand is binary; true or false, yes or no, etc. For example, it can be used to predict whether an email is spam (1) or not (0). Logistics regression uses the sigmoid function to return the probability of a label.

Sigmoid Function is a mathematical function used to map the predicted values to probabilities. The function has the ability to map any real value into another value within a range of 0 and 1.



The rule is that the value of the logistic regression must be between 0 and 1. Due to the limitations of it not being able to go beyond the value 1, on a graph it forms a curve in the form of an "S". This is an easy way to identify the Sigmoid function or the logistic function. In regards to Logistic Regression, the concept used is the threshold value. The threshold values help to define the probability of either 0 or 1. For example, values above the threshold value tend to 1, and a value below the threshold value tends to 0.

Type of Logistic Regression

1. Binomial: This means that there can be only two possible types of the dependent variables, such as 0 or 1, Yes or No, etc.
2. Multinomial: This means that there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"
3. Ordinal: This means that there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

Binary Logistic Regression Major Assumptions

1. The dependent variable should be dichotomous in nature (e.g., presence vs. absent).
2. There should be no outliers in the data, which can be assessed by converting the continuous predictors to standardized scores, and removing values below -3.29 or greater than 3.29.
3. There should be no high correlations (multicollinearity) among the predictors. This can be assessed by a correlation matrix among the predictors. Tabachnick and Fidell (2013) suggest that as long correlation coefficients among independent variables are less than 0.90 the assumption is met. He aim of training the logistic regression model is to figure out the best weights for our linear model within the logistic regression. In machine learning, we compute the optimal weights by optimizing the cost function. **Cost function:** The cost function J(Θ) is a formal representation of an objective that the algorithm is trying to achieve. In the case of logistic regression, the cost function is called LogLoss (or Cross-Entropy) and the goal is to minimize the following cost function equation:
4. $$J(\theta) =- \frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} log(h_\theta(x^{(i)})) + (1 - y^{(i)}) log(1 - h_\theta(x^{(i)}))\right]$$

Gradient descent is a method of changing weights based on the loss function for each data point. We calculate the LogLoss cost function at each input-output data point. We take a partial derivative of the weight and bias to get the slope of the cost function at each point. (No need to brush up on linear algebra and calculus right now. There are several matrix optimizations built into the Python library and Scikit-learn, which allow data science enthusiasts to unlock the power of advanced artificial intelligence without coding the answers themselves). Based on the slope, gradient descent updates the values for the bias and the set of weights, then reiterates the training loop over new values (moving a step closer to the desired goal). This iterative approach is repeated until a minimum error is reached, and gradient descent cannot minimize the cost function any further. We can change the speed at which we reach the optimal minimum by adjusting the learning rate. A high learning rate changes the weights more drastically, while a low learning rate changes them more slowly.

Lab Assignments to complete in this session:

Use the given dataset and perform the following tasks:
Dataset 1: IRIS.csv
Dataset 2: Airlines_Passanger.csv

1. Perform the logistic regression to classify Dataset 1 and Dataset 2 by using python library.
2. Compare the results of Logistic Regression model for a given dataset by using F1 measure score.

DATASET 1: IRIS.CSV

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_ma
trix


# Load the dataset using pandas
df = pd.read_csv('/content/Iris.csv')
df
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

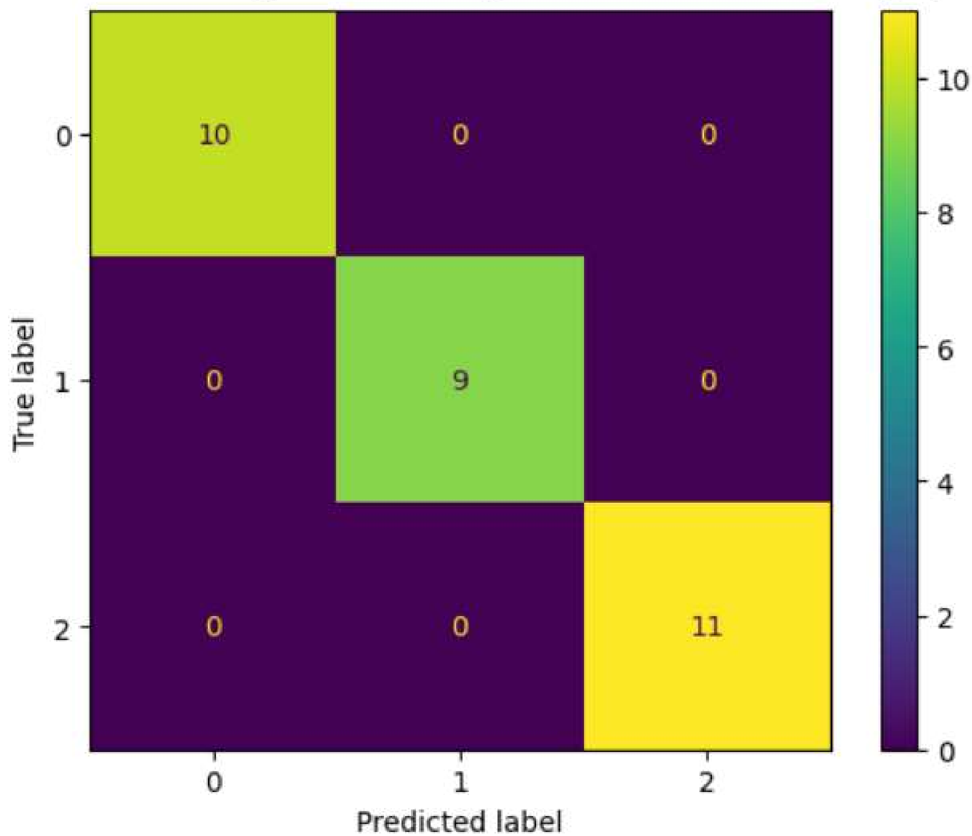
```
# Inspect the first few rows of the dataset
print(df.head())

# Check the shape of the dataset
print(df.shape)

# Check the data types of the columns
```

```python
print(df.dtypes)

# Check for missing values
print(df.isnull().sum())

# Check the class distribution
print(df['Species'].value_counts())
```

```
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa
(150, 6)
Id                 int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
```

```
cm=confusion_matrix(y_test,y_pred)
disp=ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fc804ec9cd0>
```




```
# Separate the features and target variable
X = df.iloc[:, :-1] # Features
y = df.iloc[:, -1] # Target variable

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random
_state=42)

# Initialize the logistic regression model
model = LogisticRegression()

# Fit the model to the training data
```

```
model.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))

# Print classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))

# Print confusion matrix
confusion_mat = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(confusion_mat)
```

```
Accuracy: 100.00%
Classification Report:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      1.00      1.00         9
 Iris-virginica       1.00      1.00      1.00        11

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30

Confusion Matrix:
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

DATASET 2: AIRLINES_PASSENGERS.CSV

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay,classificat
ion_report
from sklearn.preprocessing import MinMaxScaler

# Load the dataset using pandas
X_train=pd.read_csv("/content/train.csv")
X_test=pd.read_csv("/content/test.csv")


X_train
```

	Unnamed: 0	id	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	...	Inflight entertainment	On-board service	Leg room service
0	0	70172	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	3	4	...	5	4	3
1	1	5047	Male	disloyal Customer	25	Business travel	Business	235	3	2	...	1	1	5
2	2	110028	Female	Loyal Customer	26	Business travel	Business	1142	2	2	...	5	4	3
3	3	24026	Female	Loyal Customer	25	Business travel	Business	562	2	5	...	2	2	5
4	4	119299	Male	Loyal Customer	61	Business travel	Business	214	3	3	...	3	3	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
103899	103899	94171	Female	disloyal Customer	23	Business travel	Eco	192	2	1	...	2	3	1
103900	103900	73097	Male	Loyal Customer	49	Business travel	Business	2347	4	4	...	5	5	5
103901	103901	68825	Male	disloyal Customer	30	Business travel	Business	1995	1	1	...	4	3	2
103902	103902	54173	Female	disloyal Customer	22	Business travel	Eco	1000	1	1	...	1	4	5
103903	103903	62567	Male	Loyal Customer	27	Business travel	Business	1723	1	3	...	1	1	1

103904 rows × 25 columns

Baggage handling	Checkin service	Inflight service	Cleanliness	Departure Delay in Minutes	Delay in Minutes	satisfaction
4	4	5	5	25	18.0	neutral or dissatisfied
3	1	4	1	1	6.0	neutral or dissatisfied
4	4	4	5	0	0.0	satisfied
3	1	4	2	11	9.0	neutral or dissatisfied
4	3	3	3	0	0.0	satisfied
...	...	...	...	...	...	...
4	2	3	2	3	0.0	neutral or dissatisfied
5	5	5	4	0	0.0	satisfied
4	5	5	4	7	14.0	neutral or dissatisfied
1	5	4	1	0	0.0	neutral or dissatisfied
4	4	3	1	0	0.0	neutral or dissatisfied

```
X_train.shape
```

```
(103904, 25)
```

```
X_train=X_train.drop(columns=["id","Unnamed: 0"])
X_test=X_test.drop(columns=["id","Unnamed: 0",])
X_train.isnull().sum()
```

```
Gender                                 0
Customer Type                          0
Age                                    0
Type of Travel                         0
Class                                  0
Flight Distance                        0
Inflight wifi service                  0
Departure/Arrival time convenient      0
Ease of Online booking                 0
Gate location                          0
Food and drink                         0
Online boarding                        0
Seat comfort                           0
Inflight entertainment                 0
On-board service                       0
Leg room service                       0
Baggage handling                       0
Checkin service                        0
Inflight service                       0
Cleanliness                            0
Departure Delay in Minutes             0
Arrival Delay in Minutes             310
satisfaction                           0
dtype: int64
```

```
X_train=X_train.drop(columns="Arrival Delay in Minutes")
X_test=X_test.drop(columns="Arrival Delay in Minutes")
le=LabelEncoder()
ohe=OneHotEncoder()
X_train_encoded=pd.get_dummies(X_train)
X_test_encoded=pd.get_dummies(X_test)
X_test_encoded=X_test_encoded.dropna();
X_train_encoded.head()
```

	Age	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	Food and drink	Online boarding	Seat comfort	Inflight entertainment	...	Gender_Male	Customer Type_Loyal Customer	T
0	13	460	3	4	3	1	5	3	5	5	...	1	1	
1	25	235	3	2	3	3	1	3	1	1	...	1	0	
2	26	1142	2	2	2	2	5	5	5	5	...	0	1	
3	25	562	2	5	5	5	2	2	2	2	...	0	1	
4	61	214	3	3	3	3	4	5	5	3	...	1	1	

5 rows × 28 columns

Customer Type_disloyal Customer	Type of Travel_Business travel	Type of Travel_Personal Travel	Class_Business	Class_Eco	Class_Eco Plus	satisfaction_neutral or dissatisfied	satisfaction_satisfied
0	0	1	0	0	1	1	0
1	1	0	1	0	0	1	0
0	1	0	1	0	0	0	1
0	1	0	1	0	0	1	0
0	1	0	1	0	0	0	1

```
X_test_encoded.head()
```

	Age	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	Food and drink	Online boarding	Seat comfort	Inflight entertainment	...	Gender_Male	Customer Type_Loyal Customer
0	13	460	3	4	3	1	5	3	5	5	...	1	1
1	25	235	3	2	3	3	1	3	1	1	...	1	0
2	26	1142	2	2	2	2	5	5	5	5	...	0	1
3	25	562	2	5	5	5	2	2	2	2	...	0	1
4	61	214	3	3	3	3	4	5	5	3	...	1	1

5 rows × 28 columns

Customer Type_disloyal Customer	Type of Travel_Business travel	Type of Travel_Personal Travel	Class_Business	Class_Eco	Class_Eco Plus	satisfaction_neutral or dissatisfied	satisfaction_satisfied
0	0	1	0	0	1	1	0
1	1	0	1	0	0	1	0
0	1	0	1	0	0	0	1
0	1	0	1	0	0	1	0
0	1	0	1	0	0	0	1

```
y_train=X_train["satisfaction"]
y_test=X_test["satisfaction"]
y_train=le.fit_transform(y_train)
y_test=le.fit_transform(y_test)
y_train.shape
```

```
(103904,)
```

```
y_test
```

```
array([0, 0, 1, ..., 0, 0, 0])
```

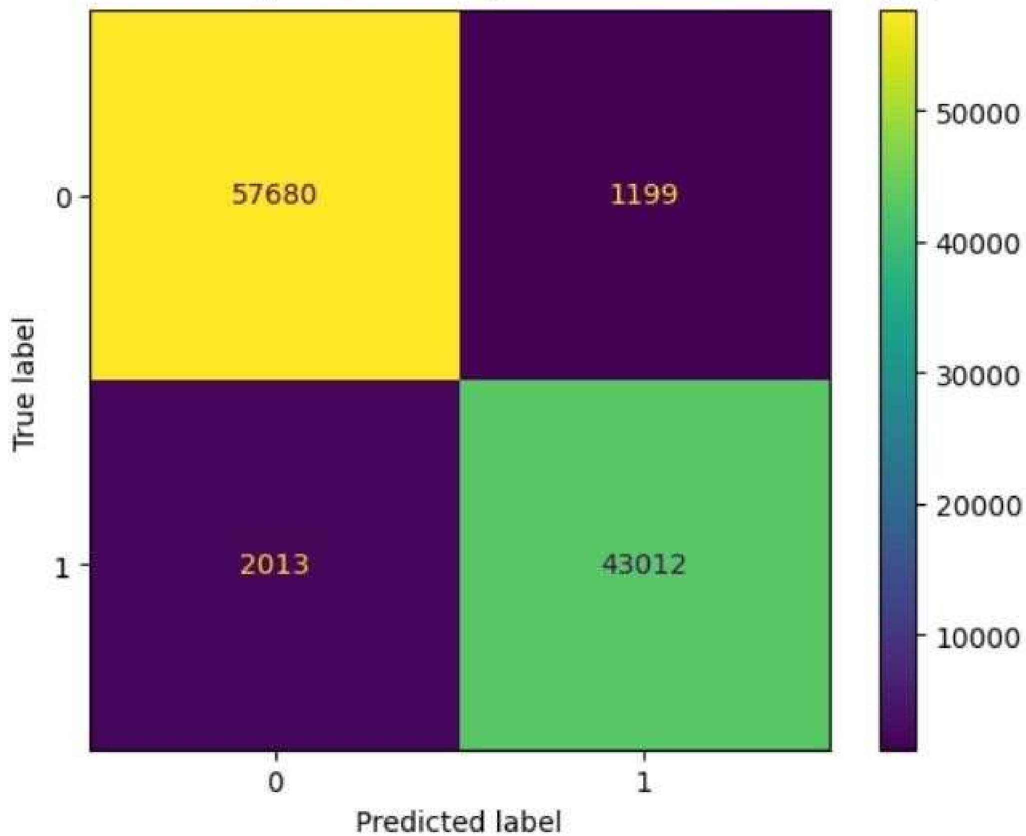
```
model=LogisticRegression()

model_lr=model.fit(X_train_encoded,y_train)
```

```
y_pred=model_lr.predict(X_test_encoded)
accuracy_score(y_test,y_pred)
```

```
0.9690868493994457
```

```
cm=confusion_matrix(y_test,y_pred)
disp=ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fd155a0a280>
```



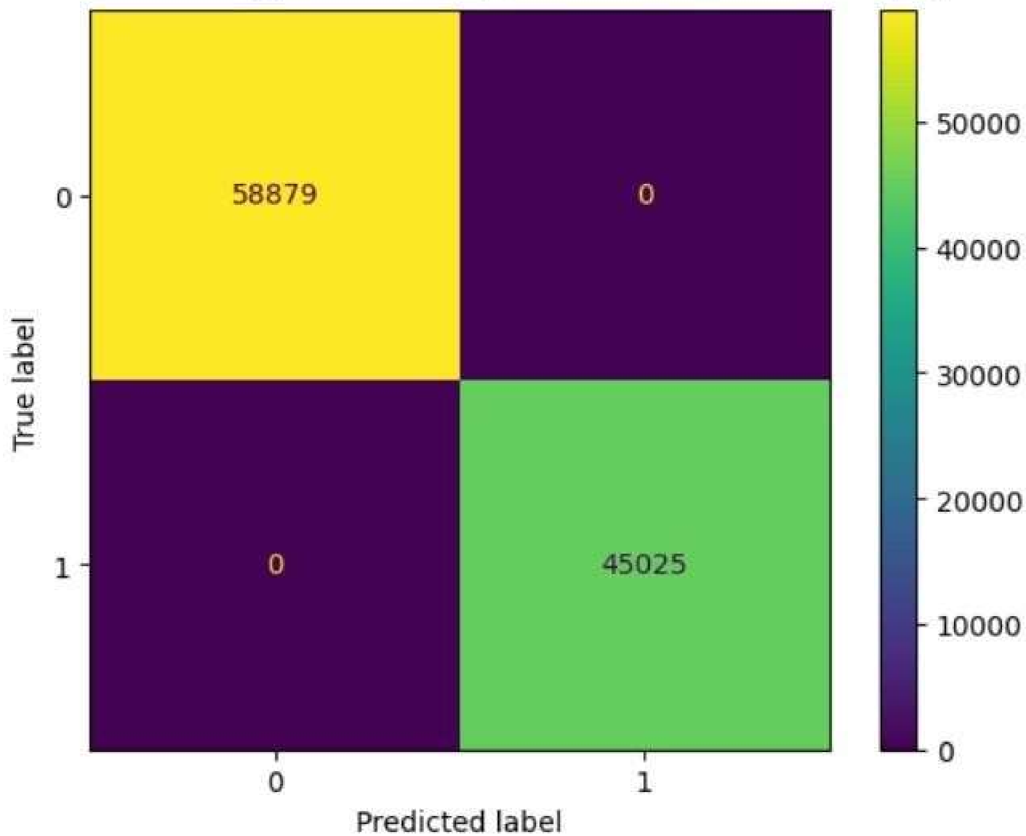

```
# Print classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))
## Normalizing Data
scaler=MinMaxScaler()
X_train_scaled=scaler.fit_transform(X_train_encoded)
X_test_scaled=scaler.fit_transform(X_test_encoded)
model_lr_normalized=model.fit(X_train_scaled,y_train)
y_pred_norm=model_lr_normalized.predict(X_test_scaled)
```

```
accuracy_score(y_test,y_pred_norm)
```

```
1.0
```

```
cm=confusion_matrix(y_test,y_pred_norm)
disp=ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fd152febf10>
```




```
# Print classification report
print("Classification Report:")
print(classification_report(y_test, y_pred_norm))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     58879
           1       1.00      1.00      1.00     45025

    accuracy                           1.00    103904
   macro avg       1.00      1.00      1.00    103904
weighted avg       1.00      1.00      1.00    103904
```