St. Francis Institute of Technology, Mumbai-400 103
Department of Information Technology

A.Y. 2024-2025
Class: TE-ITA/B, Semester: VI

Subject: **Data Science Lab**

Experiment – 5: To implement Regression.

1. **Aim:** To implement Linear and Logistic Regression to find out relation between variables.
2. **Objectives:** After study of this experiment, the student will be able to
   - Understand linear regression
   - Understand logistic regression
3. **Outcomes:** After study of this experiment, the student will be able to
   - Understand concepts of regression
4. **Prerequisite:** Fundamentals of Python Programming and Database Management System.
5. **Requirements:** Python Installation, Personal Computer, Windows operating system, Internet Connection, Microsoft Word.
6. **Pre-Experiment Exercise:**

Brief Theory:

- Concept of regression in machine learning**.**

Laboratory Exercise

A. **Procedure:** (the sheet for commands in attached with the file)

B. Paste Screenshots of above commands.

8. Post-Experiments Exercise

A. Extended Theory: (Soft Copy)

- Logistic regression

B. Questions:

- Use MNIST Dataset and apply logistic regression.

C. Conclusion:

Write the significance of the topic studied in the experiment.

D. References:

1. Logistic Regression using Python (scikit-learn) | by Michael Galarnyk | Towards Data Science

2. Logistic Regression in Python | Techniques for Logistic Regression (educba.com)

Name: Vishal Rajesh Mahajan AIDS EXP 05

Class: TE IT A Roll No: 56

A. Extended Theory:

Logistic regression

Logistic regression is a statistical method used for binary classification, predicting the probability that an outcome belongs to one of two classes (e.g., yes/no, spam/not spam). It uses the logistic (sigmoid) function, which transforms the output of a linear equation into a probability between 0 and 1.

The formula is:

$$P(y=1|X) = \frac{1}{1+e^{-(b_0+b_1x_1+b_2x_2+\ldots+b_nx_n)}}$$

Where x1,x2,...x1, x2, ...x1,x2,... are the features, and b0,b1,...b0, b1, ...b0,b1,... are the model's coefficients. The model is trained to find the best coefficients by minimizing log loss (or binary cross-entropy). Based on the predicted probability, the model classifies the outcome as either class 1 (above 0.5) or class 0 (below 0.5).

In [ ]:

```python
import sklearn
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB

# Load the dataset
loan_data = pd.read_csv('/content/loan-train.csv')

# Encode categorical columns using LabelEncoder
loan_data['Gender'] = LabelEncoder().fit_transform(loan_data['Gender'].astype(str))
loan_data['Married'] = LabelEncoder().fit_transform(loan_data['Married'].astype(str))
loan_data['Education'] = LabelEncoder().fit_transform(loan_data['Education'].astype(str))
loan_data['Self_Employed'] = LabelEncoder().fit_transform(loan_data['Self_Employed'].astype(str))
loan_data['Property_Area'] = LabelEncoder().fit_transform(loan_data['Property_Area'].astype(str))

# Handle 'Dependents' column: Replace '3+' with 3 and convert to numeric
loan_data['Dependents'] = loan_data['Dependents'].str.replace('+', '') # Removing the '+' sign fro
loan_data['Dependents'] = pd.to_numeric(loan_data['Dependents'], errors='coerce').fillna(0).astype

# Drop 'Loan_ID' column as it's not needed for the model
loan_data = loan_data.drop('Loan_ID', axis=1)

# Fill missing values with the mean of the respective columns
loan_data = loan_data.fillna(0)


# Define the feature variables (X) and target variable (Y)
X = loan_data.drop('Loan_Status', axis=1)
Y = loan_data['Loan_Status']

# Encode the target variable
Y = LabelEncoder().fit_transform(loan_data['Loan_Status'].astype(str))

# Split the data into training and testing sets (70% train, 30% test)
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=1)

# Create and train the Naive Bayes classifier
Naivebayes = GaussianNB()
b = Naivebayes.fit(X_train, y_train)

# Make predictions on the test data
c = Naivebayes.predict(X_test)

# Print the predictions
print(c)

# Calculate and print the accuracy score
acc = sklearn.metrics.accuracy_score(y_test, c)
print(acc)
```

```
[0 0 1 1 1 0 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0
 0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1
 1 0 0 0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 1 0 1 0 0 0 1 1 1 0 0 1 1 0 0 1 0 1]
0.7027027027027027
```

In [9]:

```python
import sklearn
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Load the dataset
loan_data = pd.read_csv('/content/loan-train.csv')

# Encode categorical columns using LabelEncoder
loan_data['Gender'] = LabelEncoder().fit_transform(loan_data['Gender'].astype(str))
loan_data['Married'] = LabelEncoder().fit_transform(loan_data['Married'].astype(str))
loan_data['Education'] = LabelEncoder().fit_transform(loan_data['Education'].astype(str))
loan_data['Self_Employed'] = LabelEncoder().fit_transform(loan_data['Self_Employed'].astype(str))
loan_data['Property_Area'] = LabelEncoder().fit_transform(loan_data['Property_Area'].astype(str))

# Handle 'Dependents' column: Replace '3+' with 3 and convert to numeric
loan_data['Dependents'] = loan_data['Dependents'].str.replace('+', '') # Removing the '+' sign fr
loan_data['Dependents'] = pd.to_numeric(loan_data['Dependents'], errors='coerce').fillna(0).astype

# Drop 'Loan_ID' column as it's not needed for the model
loan_data = loan_data.drop('Loan_ID', axis=1)

# Fill missing values with the mean of the respective columns
loan_data = loan_data.fillna(0)

# Define the feature variables (X) and target variable (Y)
X = loan_data.drop('Loan_Status', axis=1)
Y = loan_data['Loan_Status']

# Encode the target variable
Y = LabelEncoder().fit_transform(loan_data['Loan_Status'].astype(str))

# Split the data into training and testing sets (70% train, 30% test)
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=1)

# Create and train the Naive Bayes classifier
Naivebayes = GaussianNB()
Naivebayes.fit(X_train, y_train)
y_pred_nb = Naivebayes.predict(X_test)
acc_nb = accuracy_score(y_test, y_pred_nb)
print("Naive Bayes Accuracy: ", acc_nb)

# Create and train the Support Vector Machine (SVM) classifier
svm_model = SVC(random_state=1)
svm_model.fit(X_train, y_train)
y_pred_svm = svm_model.predict(X_test)
acc_svm = accuracy_score(y_test, y_pred_svm)
print("SVM Accuracy: ", acc_svm)

# Create and train the K-Nearest Neighbors (KNN) classifier
knn_model = KNeighborsClassifier()
knn_model.fit(X_train, y_train)
y_pred_knn = knn_model.predict(X_test)
acc_knn = accuracy_score(y_test, y_pred_knn)
print("KNN Accuracy: ", acc_knn)
```

```
Naive Bayes Accuracy:  0.7027027027027027
SVM Accuracy:  0.6702702702702703
KNN Accuracy:  0.6324324324324324
```