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 – 6: To implement Classification.

1. **Aim:** To implement classification modeling and evaluate performance of classifiers.
2. **Objectives:** After study of this experiment, the student will be able to
   - Understand classification.
3. **Outcomes:** After study of this experiment, the student will be able to
   - Understand concepts of classification in data science.
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 classification in machine learning**.** (Naive Byes, ID3, KNN, Random Forest)

**Laboratory Exercise**

A. **Procedure:** (Home _Loan Dataset)

```
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

loan_data = pd.read_csv('home_loan_train.csv')

#print(loan_data.columns.values)

#print(loan_data['Gender'])
loan_data['Gender'] =
LabelEncoder().fit_transform(loan_data['Gender'].astype(str))
#print(loan_data['Gender'])

loan_data['Married'] =
LabelEncoder().fit_transform(loan_data['Married'].astype(str))
#print(loan_data['Married'])

loan_data['Education'] =
LabelEncoder().fit_transform(loan_data['Education'].astype(str))
#print(loan_data['Education'])
```

```
loan_data['Self_Employed'] =
LabelEncoder().fit_transform(loan_data['Self_Employed'].astype(str))
#print(loan_data['Self_Employed'])

loan_data['Property_Area'] =
LabelEncoder().fit_transform(loan_data['Property_Area'].astype(str))
#print(loan_data['Property_Area'])


loan_data = loan_data.drop('Loan_ID',axis=1)
loan_data = loan_data.fillna(loan_data.mean())

X = loan_data.drop('Loan_Status',axis=1)
Y = loan_data['Loan_Status']
Y = LabelEncoder().fit_transform(loan_data['Loan_Status'].astype(str))

X_train, X_test, y_train, y_test = train_test_split(X,Y,test_size=0.3,
random_state=1)

Naivebayes = GaussianNB()

b = Naivebayes.fit(X_train,y_train)
c = Naivebayes.predict(X_test)

print(c)

acc = sklearn.metrics.accuracy_score(y_test, c)
print(acc)
```

B. Paste Screenshots of above commands.

## 8. Post-Experiments Exercise

### A. Extended Theory: (Soft Copy)

- Types of classification.

### B. Questions:

- Compare S&P500 dataset with other classifiers SVM and KNN.

### C. Conclusion:

Write the significance of the topic studied in the experiment.

### D. References:


1. Machine Learning Classification Strategy In Python (quantinsti.com)

2. https://blog.quantinsti.com/machine-learning-k-nearest-neighbors-knn-algorithm-python/

3. Discrete vs. Continuous Data: All You Need to Know (yummysoftware.com)

4. How Naive Bayes Algorithm Works? (with example and full code) | ML+ (machinelearningplus.com)

Name: Vishal Rajesh Mahajan AIDS EXP 06

Class: TE IT A Roll No: 56

A. Extended Theory:

# Types of classification.

**Classification** is a type of supervised learning technique in machine learning where the goal is to predict the category or class of given data points. It is used when the output variable is categorical in nature. Based on the number of output classes and the structure of the problem, classification is divided into several types:

**Binary Classification**

In binary classification, the model predicts one of two possible outcomes. It is used when there are only two distinct classes. For example, classifying emails as spam or not spam, or predicting whether a tumor is malignant or benign.

**Multiclass Classification**

This involves predicting one class out of more than two possible categories. Each input is assigned to exactly one class. Examples include handwritten digit recognition (0 to 9) or classifying types of animals.

**Multilabel Classification**

In multilabel classification, each input can be associated with more than one label or class at the same time. For instance, a movie may belong to multiple genres like action, thriller, and drama.

**Imbalanced Classification**

This occurs when the number of instances in each class is not evenly distributed. One class may have significantly more data than others, which can lead to biased models. A common example is fraud detection, where fraudulent transactions are much rarer than normal ones.

These types of classification problems are handled using various algorithms and strategies depending on the dataset and problem statement.

In [ ]:

```
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 0 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 1 0 1 1 1 1 1 1 0 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]:

```
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 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()
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
```