Date: 23-07-2025 **Name:** Pranavah Varun
Experiment: 1 **Roll No:** 3122237001039

Experiment 1: Working with Python packages-Numpy, Scipy, Scikit-Learn, Matplotlib

Aim: To explore Python packages such as NumPy, SciPy, Pandas, Scikit-learn, and Matplotlib, and apply machine learning workflows on datasets from UCI and Kaggle repositories.

Libraries used:

- Pandas - for data handling
- numpy - for numerical operations
- matplotlib.pyplot - for visualization
- sklearn - for model building and evaluation
- Scipy - provides a large collection of functions for advanced mathematical, scientific, and engineering computations

Exploring Python Libraries

NumPy

- **Key Functions:** array, reshape, arange, linspace, zeros, ones, sum, mean, std, dot, random
- **Operations:** Vectorized computation, matrix algebra, broadcasting

Pandas

- **Common Functions:** DataFrame, read csv, head(), info(), describe(), groupby(), merge(), pivot_table()
- **Data Cleaning:** dropna(), fillna(), astype(), map(), replace()

SciPy

- **Purpose:** Scientific computation and mathematical operations
- **Key Modules:** scipy.stats (statistical tests), scipy.optimize (optimization), scipy.integrate, scipy.spatial, scipy.signal

Scikit-Learn

- **Tasks:** Classification, regression, clustering, model evaluation
- **Key Modules:** datasets, model_selection, preprocessing, metrics, linear model, tree, svm

Matplotlib

- **Purpose:** Data visualization and graphical representation
- **Common Functions:** plot(), scatter(), bar(), hist(), imshow(), subplot(), title(), xlabel(), ylabel()

Objective: The objective of this experiment is to explore and understand the functionality of essential Python libraries used in data analysis and machine learning, namely **NumPy**, **Pandas**, **SciPy**, **Scikit-learn**, and **Matplotlib**.

The experiment also aims to:

- Understand the core operations such as array manipulations, data preprocessing, scientific computing, model building, and data visualization.
- Apply machine learning workflows to real-world datasets obtained from public repositories like the UCI Machine Learning Repository and Kaggle.
- Identify suitable machine learning tasks and algorithms for various datasets such as loan prediction, handwritten digit recognition, email spam classification, diabetes prediction, and the Iris dataset.
- Perform data loading, exploratory data analysis, preprocessing, feature selection, model training, and performance evaluation.

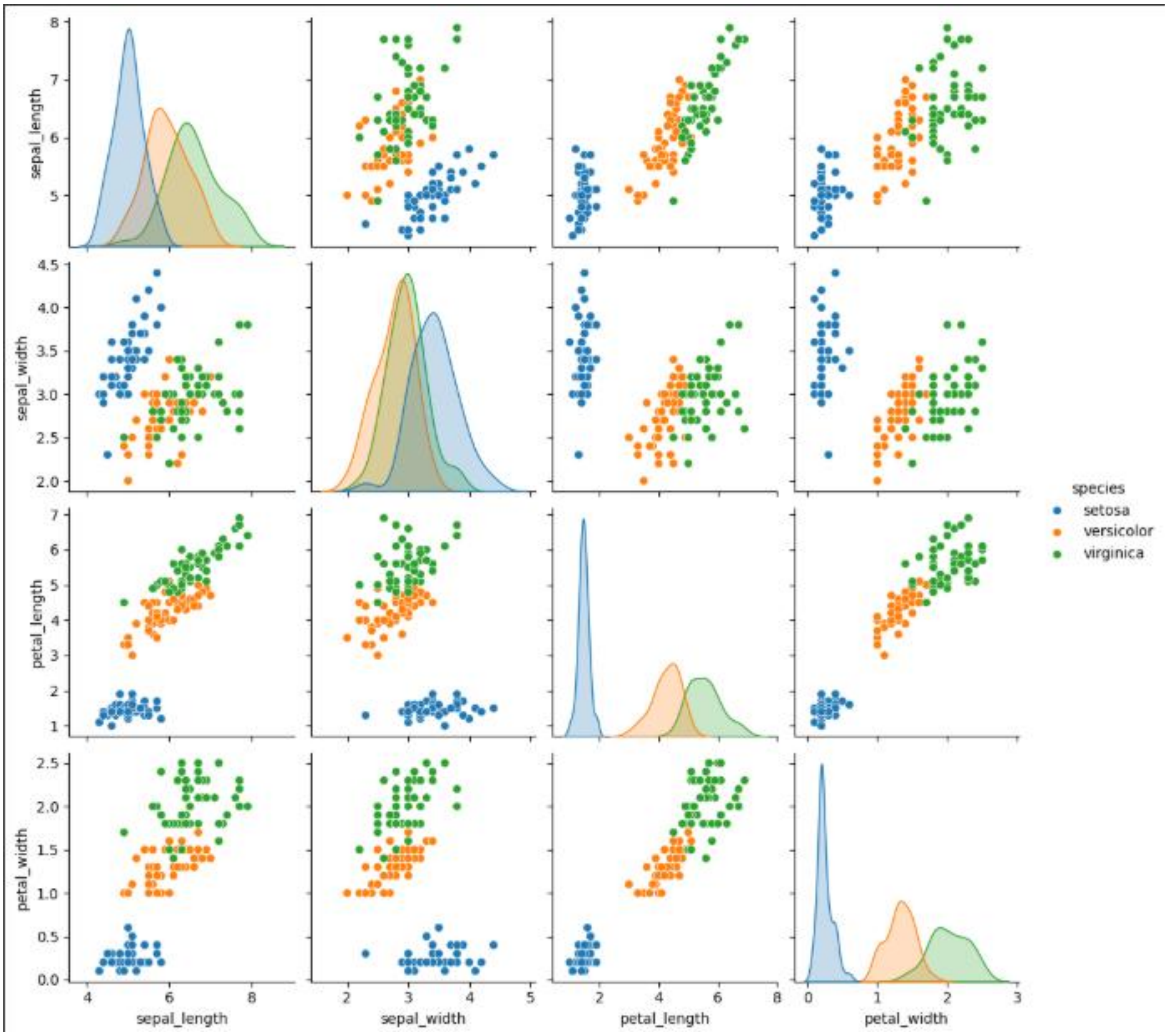
ML WORKFLOW STEPS

- Load dataset using Pandas (read_csv)
- EDA: describe(), info(), value_counts(), plotting
- Preprocessing: LabelEncoder, StandardScaler, handling NaN
- Feature Selection: SelectKBest, f_classif, chi2
- Split Data: train_test_split()
- Model Selection & Training
- Performance Evaluation: Accuracy, confusion matrix, classification report

CODE:

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

----------------------IRIS DATASET--------------------------------------------
# Load dataset
iris = sns.load_dataset('iris')

# EDA
print(iris.head())
sns.pairplot(iris, hue='species')
plt.show()

# Preprocessing
X = iris.drop('species', axis=1)
y = iris['species']

# Feature selection
selector = SelectKBest(score_func=f_classif, k='all')
X_new = selector.fit_transform(X, y)

# Split dataset
X_train, X_test, y_train, y_test = train_test_split
(X_new, y, test_size=0.2, random_state=42)

# Model
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)

# Evaluation
y_pred = model.predict(X_test)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

OUTPUT



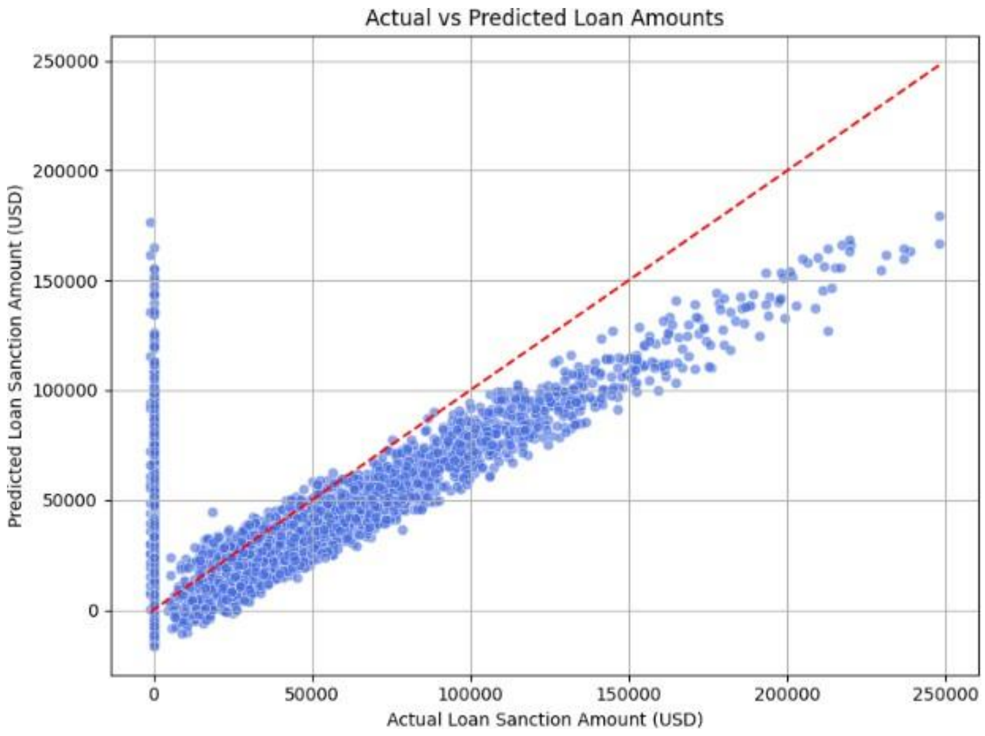
---------------------------LOAN AMOUNT DATASET-----------------------

```
# 1. Import libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from google.colab import drive

# 2. Load dataset from Google Drive
```

```
drive.mount('/content/drive')
df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/ML LAB SEM 5/train.csv')

# 3. Drop identifier columns and handle missing values
df.drop(columns=["Customer ID", "Name", "Property ID"], inplace=True)
df.dropna(inplace=True)

# 4. Define features and target
X = df.drop(columns=["Loan Sanction Amount (USD)"])
y = df["Loan Sanction Amount (USD)"]

# 5. Encode categoricals and scale numeric features
X = pd.get_dummies(X, drop_first=True)
X = StandardScaler().fit_transform(X)

# 6. Train-test split and model training
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)

# 7. Model evaluation
y_pred = model.predict(X_test)
print("RMSE:", mean_squared_error(y_test, y_pred))
print("R² Score:", r2_score(y_test, y_pred))

# 8. Plotting Actual vs Predicted values
plt.figure(figsize=(8,6))
sns.scatterplot(x=y_test, y=y_pred, alpha=0.6, color='royalblue')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')
plt.xlabel("Actual Loan Sanction Amount (USD)")
plt.ylabel("Predicted Loan Sanction Amount (USD)")
plt.title("Actual vs Predicted Loan Amounts")
plt.grid(True)
plt.tight_layout()
plt.show()
```

OUTPUT

```
RMSE: 1195267145.5071688
R² Score: 0.47512320259332885
```



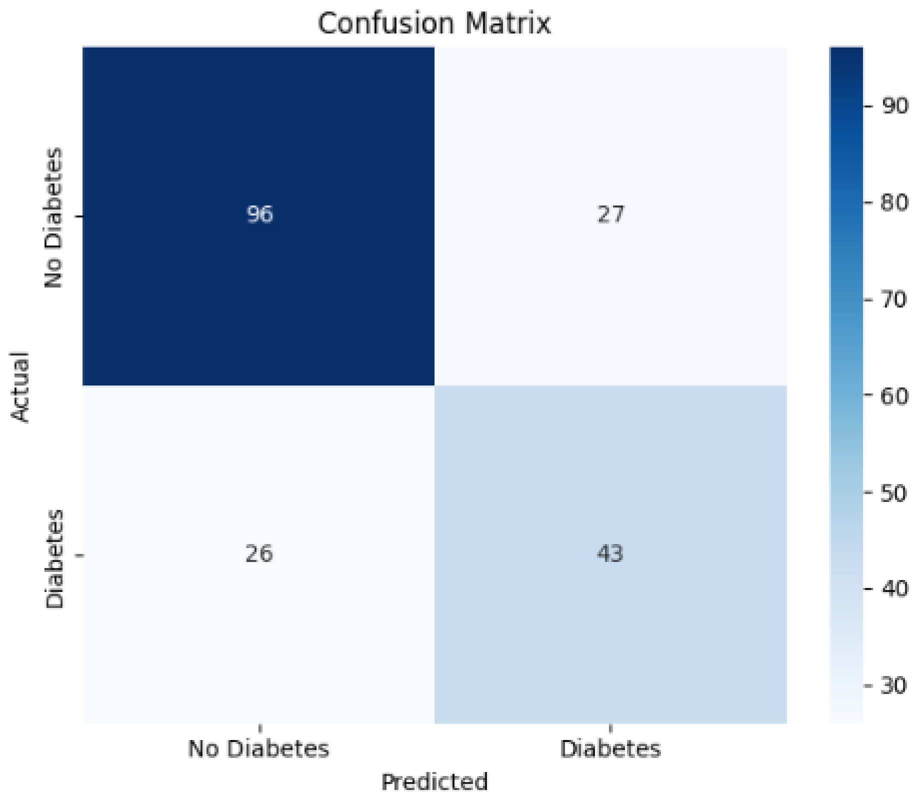
```
----------------------DIABETES DATASET----------------------------------------------
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score

# Load dataset
df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/
ML LAB SEM 5/diabetes.csv')    # PIMA Indian dataset

# EDA
print(df.info())
print(df.describe())

# Preprocessing
X = df.drop('Outcome', axis=1)
y = df['Outcome']
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split
X_train, X_test, y_train, y_test = train_test_split
(X_scaled, y, test_size=0.25, random_state=42)

# Model
```

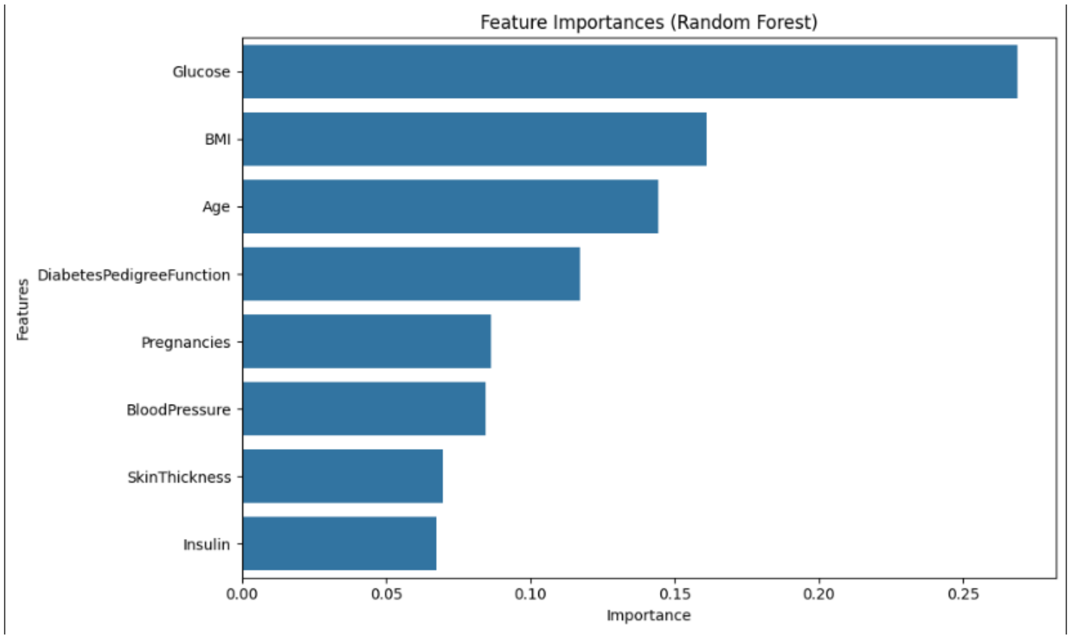
```
clf = RandomForestClassifier()
clf.fit(X_train, y_train)

# Evaluation
y_pred = clf.predict(X_test)
print(accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))

import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6,5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=["No Diabetes", "Diabetes"],
yticklabels=["No Diabetes", "Diabetes"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.tight_layout()
plt.show()

# Feature Importance
import numpy as np

feature_names = df.columns[:-1]  # All columns except 'Outcome'
importances = clf.feature_importances_
indices = np.argsort(importances)[::-1]

plt.figure(figsize=(10,6))
sns.barplot(x=importances[indices], y=feature_names[indices])
plt.title("Feature Importances (Random Forest)")
plt.xlabel("Importance")
plt.ylabel("Features")
plt.tight_layout()
plt.show()

from sklearn.metrics import roc_curve, auc

y_proba = clf.predict_proba(X_test)[:,1]
fpr, tpr, _ = roc_curve(y_test, y_proba)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(6,5))
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (AUC = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', linestyle='--')
plt.xlim([-0.01, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver Operating Characteristic")
plt.legend(loc="lower right")
plt.grid(True)
plt.tight_layout()
plt.show()
```

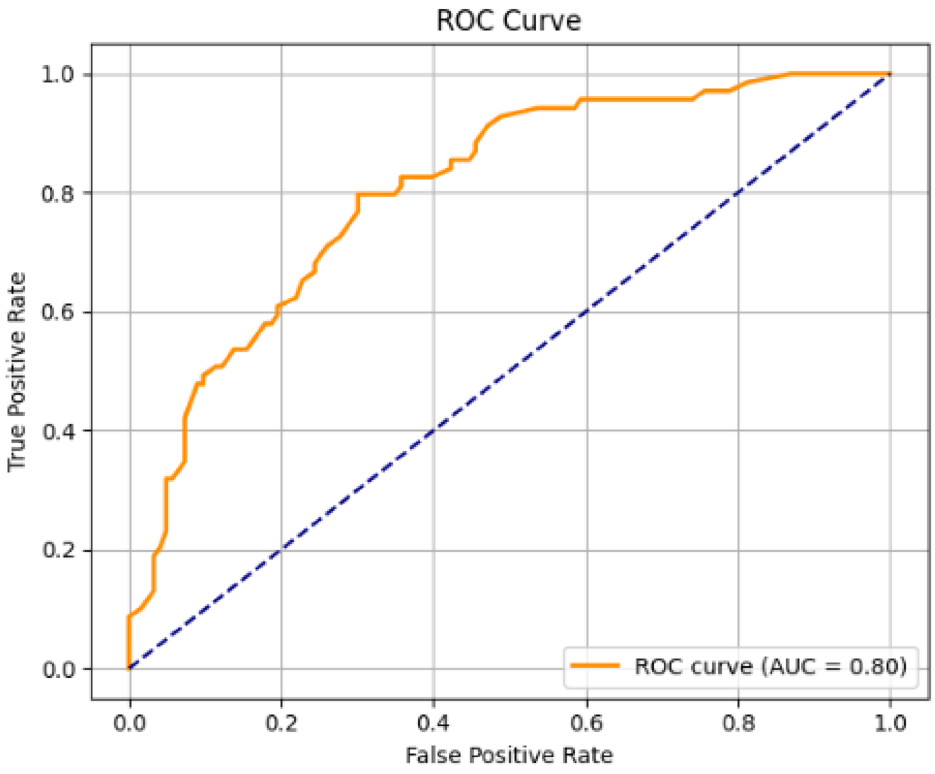
OUTPUT

```
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  -------                   --------------  ------
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
None
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479
std       3.369578   31.972618      19.355807      15.952218  115.244002
min       0.000000    0.000000       0.000000       0.000000    0.000000
25%       1.000000   99.000000      62.000000       0.000000    0.000000
50%       3.000000  117.000000      72.000000      23.000000   30.500000
75%       6.000000  140.250000      80.000000      32.000000  127.250000
max      17.000000  199.000000     122.000000      99.000000  846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000
0.7395833333333334
```

```
              precision    recall  f1-score   support

           0       0.80      0.79      0.80       123
           1       0.63      0.65      0.64        69

    accuracy                           0.74       192
   macro avg       0.72      0.72      0.72       192
weighted avg       0.74      0.74      0.74       192
```





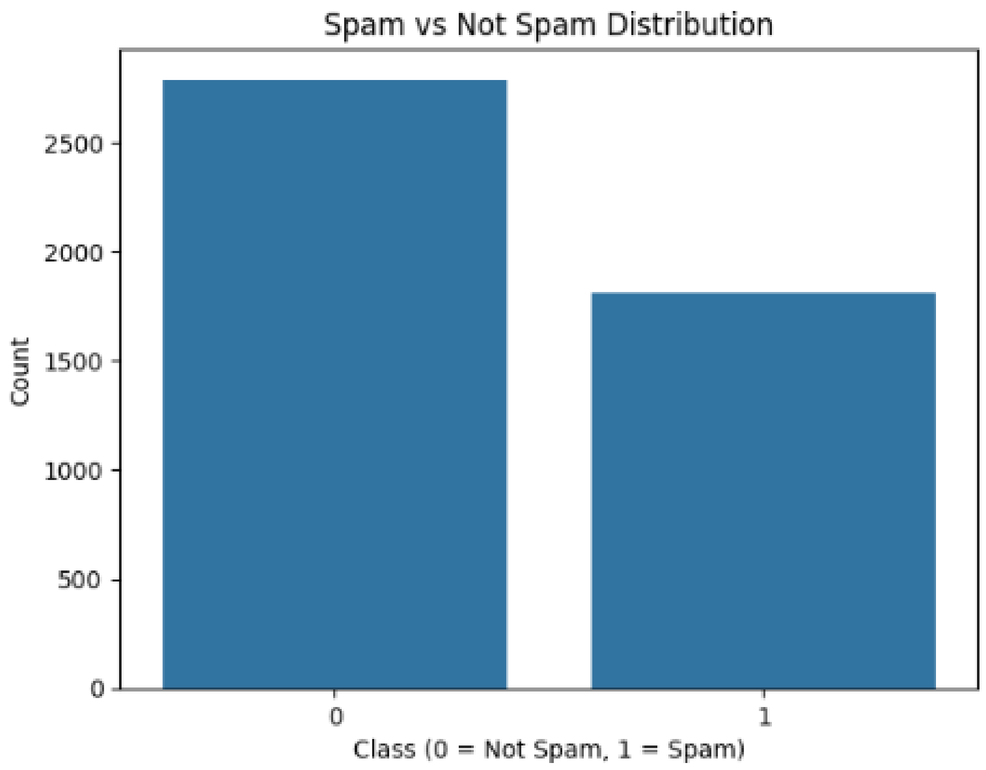


```
-------------------SPAM DATASET----------------------------
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report
```

```
# Load dataset
df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/ML LAB SEM 5/
spambase.data", header=None)   # UCI SpamBase dataset

# Preprocessing
X = df.iloc[:, :-1]
y = df.iloc[:, -1]

# Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Model
model = GaussianNB()
model.fit(X_train, y_train)

# Evaluation
y_pred = model.predict(X_test)
print(accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

OUTPUT

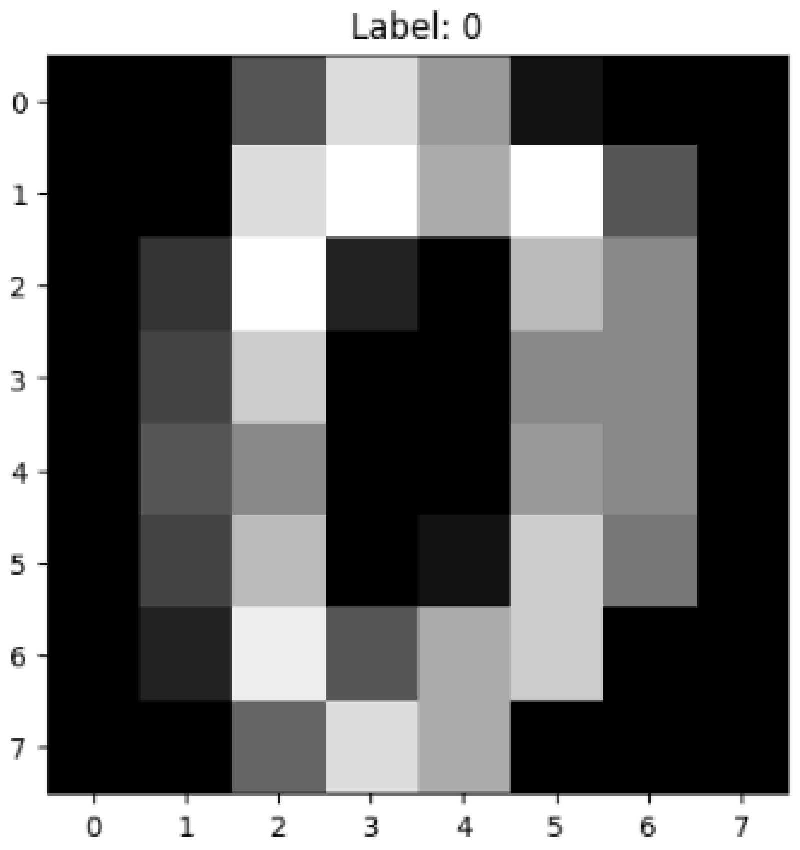
```
0.8067318132464713
              precision    recall  f1-score   support

           0       0.94      0.72      0.81       538
           1       0.70      0.93      0.80       383

    accuracy                           0.81       921
   macro avg       0.82      0.82      0.81       921
weighted avg       0.84      0.81      0.81       921
```



```
-------------------------MNIST DATASET----------
# Load dataset
digits = load_digits()
X = digits.data
y = digits.target

# Show a sample digit
plt.imshow(digits.images[0], cmap='gray')
plt.title(f'Label: {digits.target[0]}')
plt.show()

# Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Model
model = LogisticRegression(max_iter=3000)
model.fit(X_train, y_train)

# Evaluation
y_pred = model.predict(X_test)
print(accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

OUPUT

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        53
           1       0.96      0.94      0.95        50
           2       0.96      1.00      0.98        47
           3       0.98      0.96      0.97        54
           4       1.00      0.97      0.98        60
           5       0.94      0.95      0.95        66
           6       0.96      0.98      0.97        53
           7       1.00      0.96      0.98        55
           8       0.91      0.98      0.94        43
           9       0.97      0.95      0.96        59

    accuracy                           0.97       540
   macro avg       0.97      0.97      0.97       540
weighted avg       0.97      0.97      0.97       540
```



Results and Discussions:

Dataset	**ML Task**	**ML Type**	**Dataset Source**
Loan Amount Prediction	Regression	Supervised	Kaggle / UCI Repository
Handwritten Character Recognition	Classification	Supervised	MNIST Dataset
Email Spam Classification	Classification	Supervised	UCI SpamBase
MNIST (Digits Classification)	Classification	Supervised	MNIST Dataset
Predicting Diabetes	Classification	Supervised	PIMA Indian Dataset
Iris Dataset	Classification	Supervised	UCI Iris Dataset

Dataset Summary with Feature Selection and Algorithms

Dataset	**Type of ML Task**	**Feature Selection Technique**	**Suitable ML Algorithm**
Iris Dataset	Classification	ANOVA (f_classif), SelectKBest	Logistic Regression, KNN
Loan Amount Prediction	Regression	SelectKBest (f_regression)	Linear Regression, Random Forest
Predicting Diabetes	Classification	Chi2, SelectKBest	SVM, Random Forest, XGBoost
Email Spam Classification	Classification	Chi2, Mutual Info	Naive Bayes, Decision Tree
MNIST Handwritten Recognition	Classification	PCA or CNN-based feature selection	KNN, SVM, CNN (Deep Learning)

Learning Outcomes

- Gained practical experience in working with essential Python libraries such as **NumPy**, **Pandas**, **SciPy**, **Scikit-learn**, and **Matplotlib**.
- Understood the process of loading and preparing real-world datasets from public repositories like the UCI Machine Learning Repository and Kaggle.
- Learned to perform Exploratory Data Analysis (EDA) using summary statistics and visual tools such as histograms, heatmaps, and scatter plots.
- Applied data preprocessing techniques including handling missing values, encoding categorical variables, and feature scaling.
- Explored and applied feature selection techniques such as SelectKBest, chi2, and f_classif to improve model performance.
- Implemented end-to-end machine learning workflows for both classification and regression tasks.

- Evaluated models using metrics like accuracy, confusion matrix, classification report, mean squared error, and R^2 score.
- Developed confidence in identifying suitable machine learning models based on the dataset and task type.