ICS1512 – Machine Learning Algorithms Laboratory

Experiment 1: Working with Python Packages

Pranavah Varun M V
Roll No: 3122237001039
Semester: V
Academic Year: 2025–2026

Aim

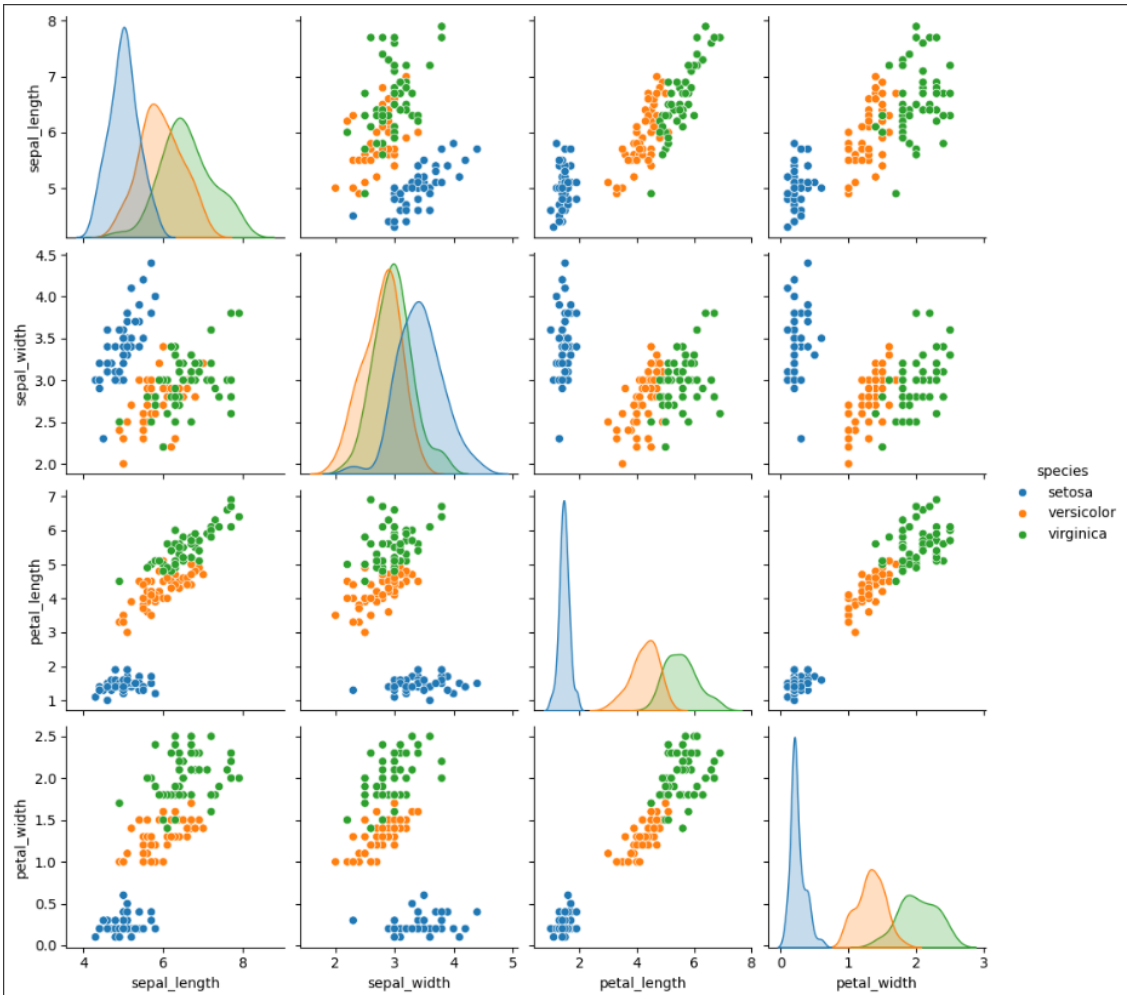
To explore Python libraries such as NumPy, Pandas, SciPy, Scikit-learn, and Matplotlib, and apply machine learning workflows on five datasets (Iris, Loan, Diabetes, Spam, MNIST) to understand data handling, preprocessing, model training, and evaluation.

1. Iris Dataset - Classification

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

iris = sns.load_dataset('iris')
sns.pairplot(iris, hue='species')
plt.show()

X = iris.drop('species', axis=1)
y = iris['species']
X_new = SelectKBest(score_func=f_classif, k='all').fit_transform(X,
    ↪ y)

X_train, X_test, y_train, y_test = train_test_split(X_new, y,
    ↪ test_size=0.2, random_state=42)
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)

```

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

Screenshot:



Result: 97% accuracy, strong precision and recall across all classes.

2. Loan Amount Prediction - Regression

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score

df = pd.read_csv('/content/drive/MyDrive/train.csv')
df.drop(columns=["Customer ID", "Name", "Property ID"], inplace=True
    ↪ )
```

```
df.dropna(inplace=True)

X = pd.get_dummies(df.drop(columns=["Loan Sanction Amount (USD)"]),
    ↪ drop_first=True)
y = df["Loan Sanction Amount (USD)"]
X = StandardScaler().fit_transform(X)

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)

y_pred = model.predict(X_test)
print("RMSE:", mean_squared_error(y_test, y_pred))
print("R  Score:", r2_score(y_test, y_pred))
```

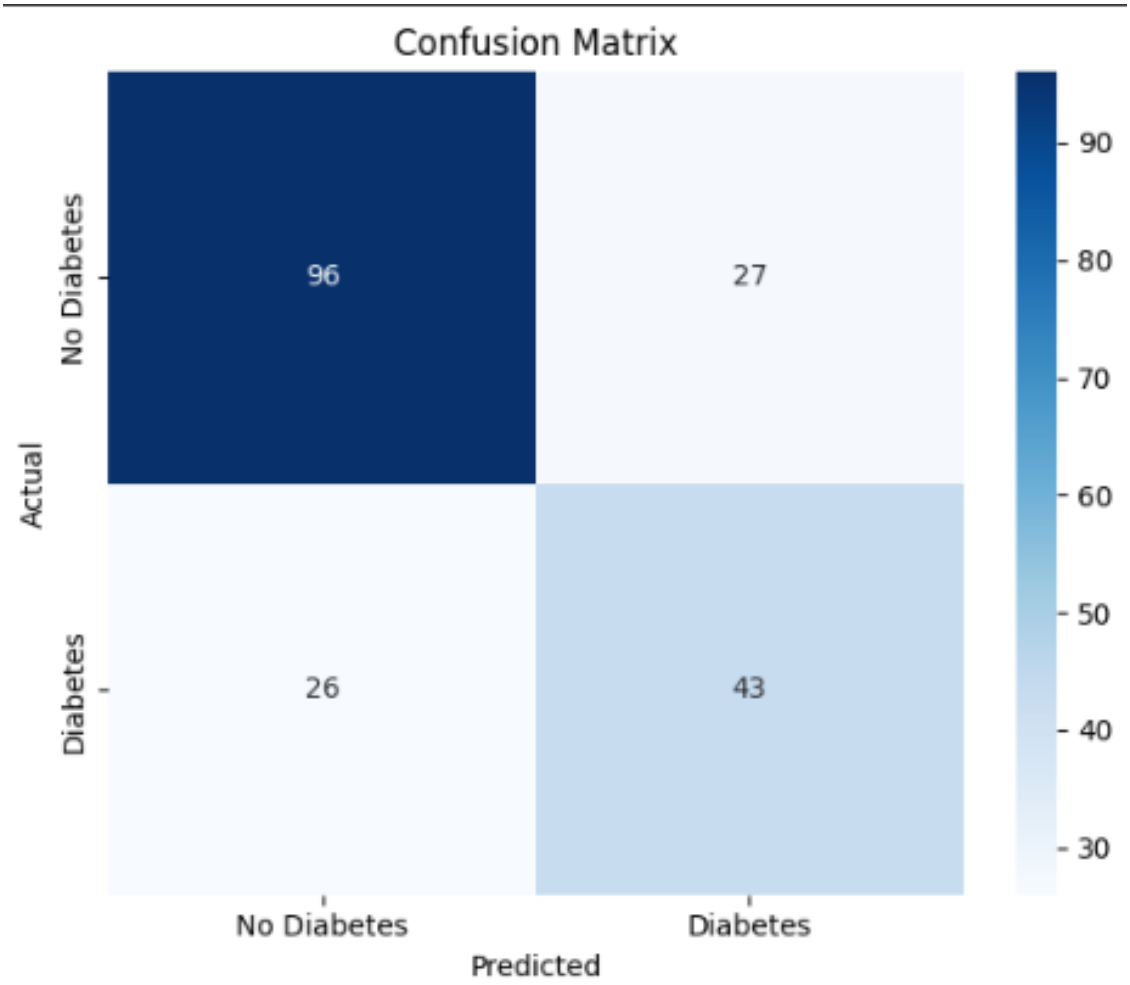
Screenshot:

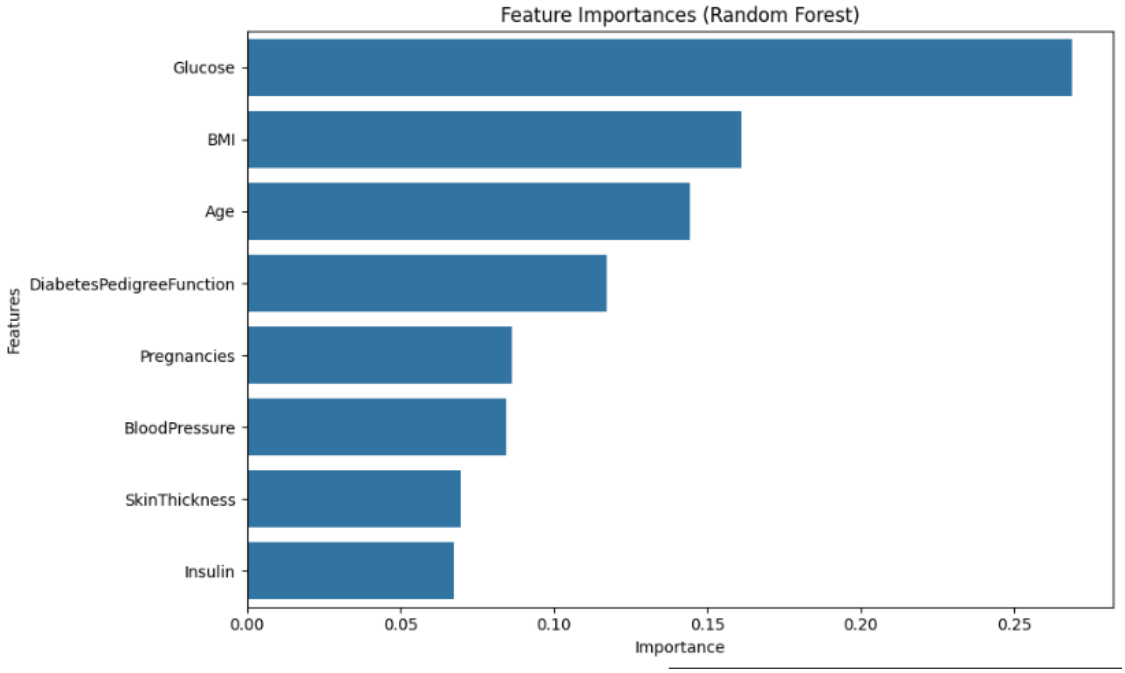
Result: RMSE = 1.19B, R^2 0.47 – moderate regression performance.

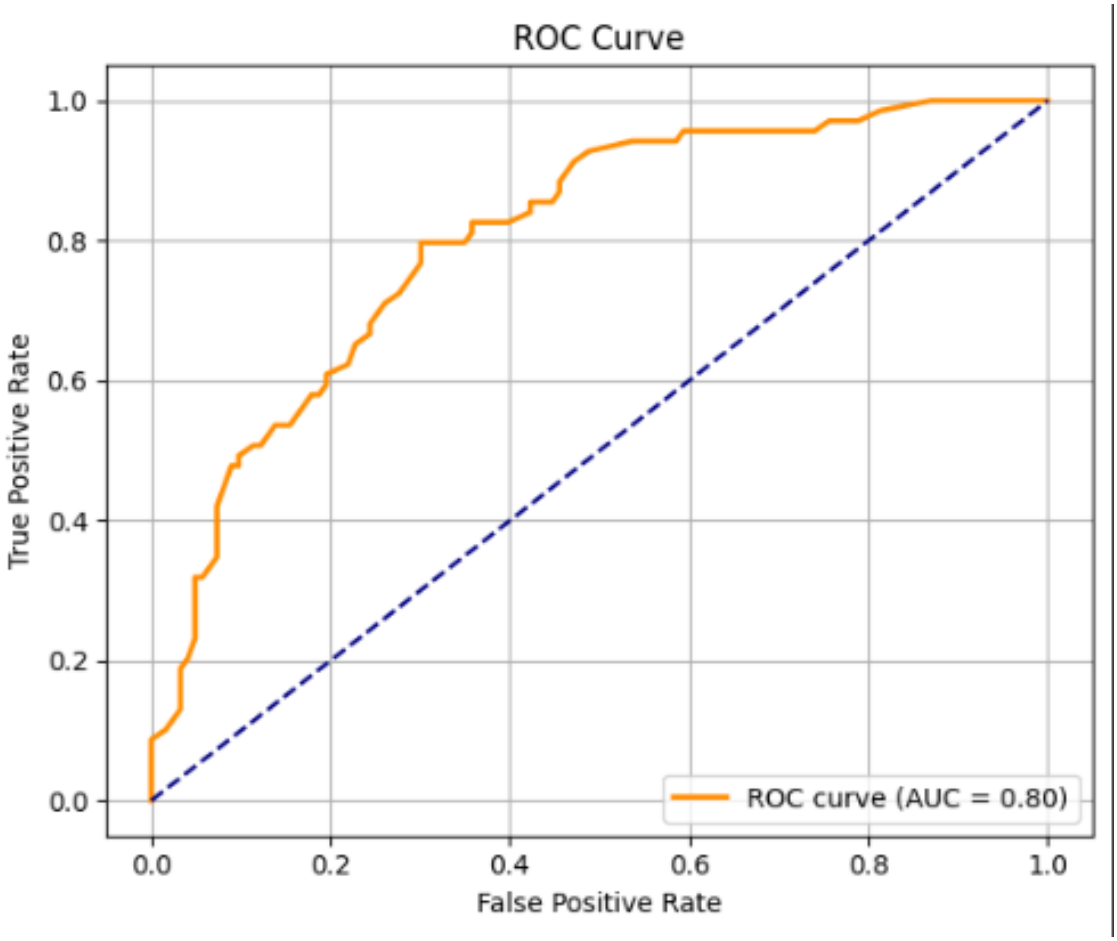
3. Diabetes Prediction - Classification

```python
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,
    ↪ confusion_matrix

df = pd.read_csv('/content/drive/MyDrive/diabetes.csv')
X = df.drop('Outcome', axis=1)
y = df['Outcome']

X_scaled = StandardScaler().fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
    ↪ test_size=0.25, random_state=42)

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

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

Screenshots:


Confusion Matrix
No Diabetes
Diabetes
Actual
96
27
26
43
90
80
70
60
50
40
30
No Diabetes
Diabetes
Predicted


Feature Importances (Random Forest)
Glucose
BMI
Age
DiabetesPedigreeFunction
Pregnancies
BloodPressure
SkinThickness
Insulin
Features
0.00
0.05
0.10
0.15
0.20
0.25
Importance



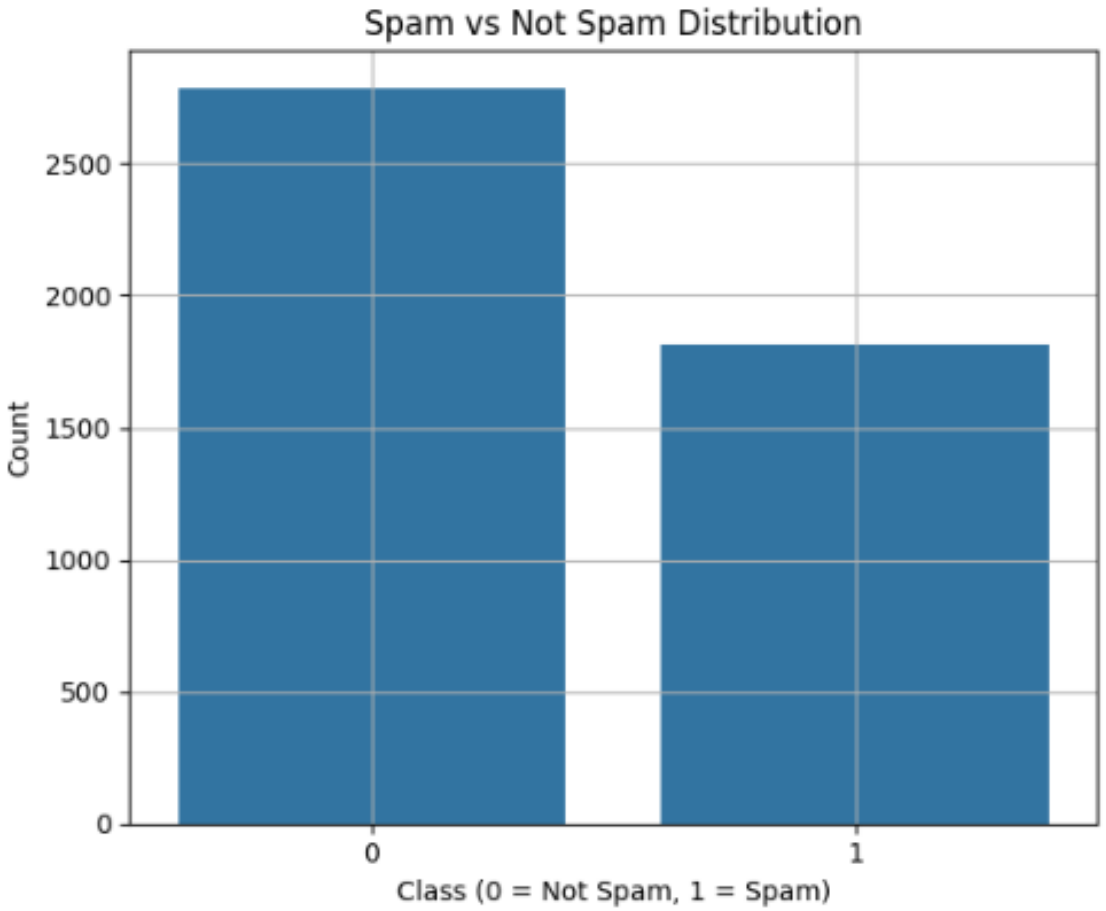
Result: 74% accuracy. Class 0 better predicted than class 1.

4. Spam Email Detection - Classification

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

df = pd.read_csv("/content/drive/MyDrive/spambase.data", header=None)
X = df.iloc[:, :-1]
y = df.iloc[:, -1]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
model = GaussianNB()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
```

```
print(accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Screenshot:



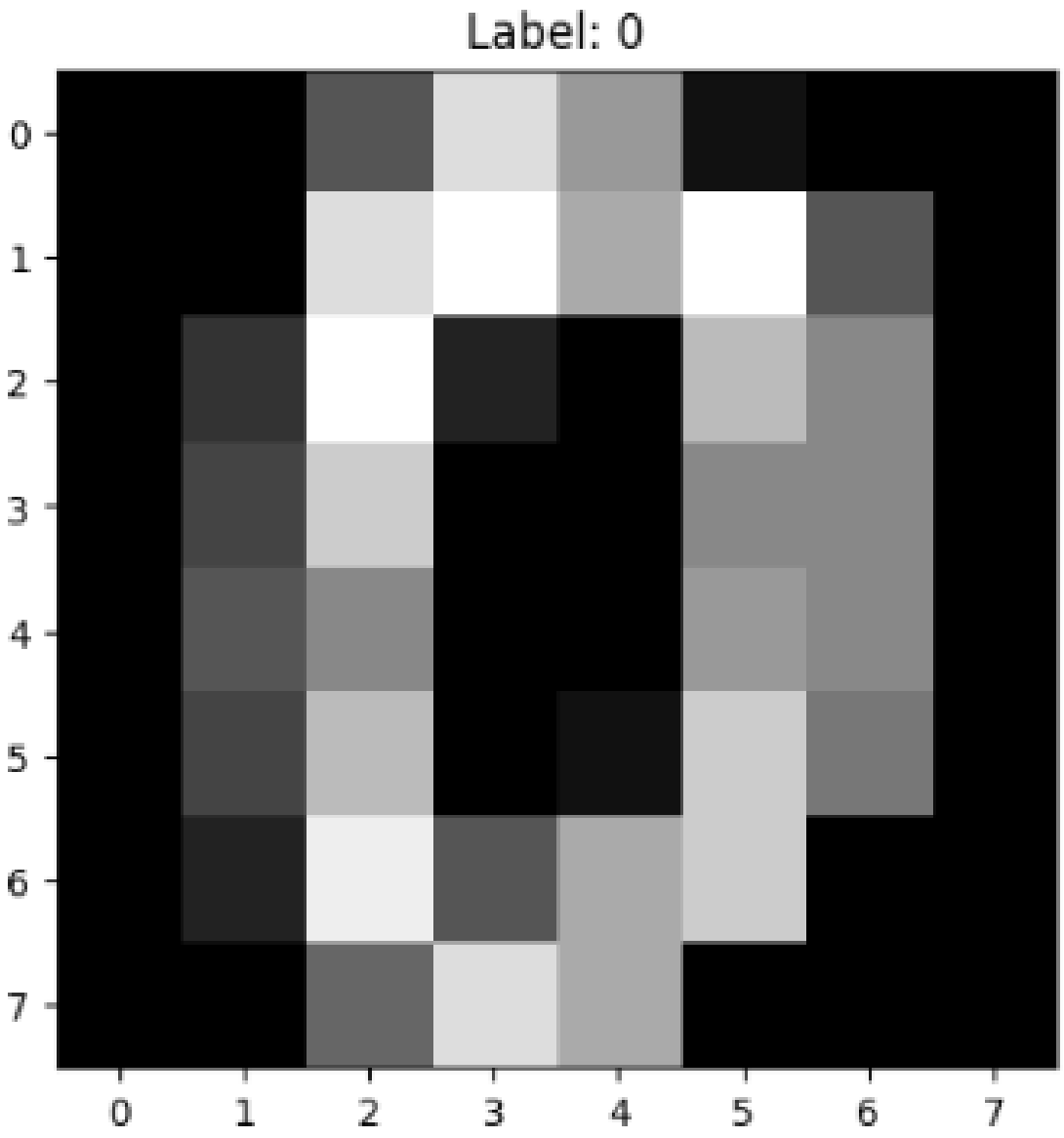
Result: Accuracy = 81%. Good recall for spam messages.

5. MNIST Digits - Classification

```
from sklearn.datasets import load_digits
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score

digits = load_digits()
X, y = digits.data, digits.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    ↪ =0.3, random_state=42)
model = LogisticRegression(max_iter=3000)
```

```
model.fit(X_train, y_train)

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

Screenshot:



Result: Accuracy = 97%. Robust digit classification with logistic regression.

Results and Discussions

Dataset Overview

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

Feature Selection and Algorithms

Dataset	Type of ML Task	Suitable ML Algorithm
Iris Dataset	Classification	Logistic Regression, KNN
Loan Amount Prediction	Regression	Linear Regression, Random Forest
Predicting Diabetes	Classification	SVM, Random Forest, XGBoost
Email Spam Classification	Classification	Naive Bayes, Decision Tree
MNIST Handwritten Recognition	Classification	KNN, SVM, CNN (Deep Learning)

Learning Outcomes

- Gained experience using Python libraries for ML (Pandas, NumPy, Scikit-learn, etc.).
- Learned data preprocessing, feature selection, and model evaluation techniques.
- Successfully applied regression and classification to real-world datasets.
- Understood when and how to use different ML algorithms like Logistic Regression, Naive Bayes, Random Forest, etc.
- Learned to interpret key performance metrics: accuracy, R^2, precision, recall, F1-score.