

# ICS1512 – Machine Learning Algorithms

## Laboratory Report

Mohammed Aatif  
Roll No: 3122237001026  
Semester: V  
Academic Year: 2025–2026

September 4, 2025

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Iris Dataset - Classification</b>	<b>3</b>
2.1	Python Code . . . . .	3
2.2	Results . . . . .	4
2.3	Results . . . . .	4
<b>3</b>	<b>Loan Amount Prediction - Regression</b>	<b>4</b>
3.1	Python Code . . . . .	4
3.2	Results . . . . .	6
<b>4</b>	<b>Diabetes Prediction - Classification</b>	<b>6</b>
4.1	Python Code . . . . .	6
4.2	Results . . . . .	7
<b>5</b>	<b>Spam Email Detection - Classification</b>	<b>9</b>
5.1	Python Code . . . . .	9
5.2	Results . . . . .	10
<b>6</b>	<b>MNIST Digit Recognition</b>	<b>10</b>
6.1	Python Code . . . . .	10
6.2	Results . . . . .	11

<b>7</b>	<b>Discussion</b>	<b>11</b>
<b>8</b>	<b>Conclusion</b>	<b>11</b>

# 1 Introduction

This report presents the results of machine learning experiments conducted using Python. The aim was to explore Python libraries such as NumPy, Pandas, SciPy, Scikit-learn, and Matplotlib, and apply them to various machine learning workflows on datasets like Iris, Loan, Diabetes, Spam, and MNIST.

Evaluation metrics such as ROC curve, Confusion Matrix, and Feature Importance were used. Visualizations were generated using Matplotlib and Seaborn.

## 2 Iris Dataset - Classification

### 2.1 Python Code

Listing 1: Logistic Regression on Iris Dataset

```
1 import seaborn as sns
2 import matplotlib.pyplot as plt
3 from sklearn.model_selection import train_test_split
4 from sklearn.feature_selection import SelectKBest,
   f_classif
5 from sklearn.linear_model import LogisticRegression
6 from sklearn.metrics import classification_report,
   confusion_matrix
7
8 iris = sns.load_dataset('iris')
9 sns.pairplot(iris, hue='species')
10 plt.show()
11
12 X = iris.drop('species', axis=1)
13 y = iris['species']
14 X_new = SelectKBest(score_func=f_classif, k='all').
   fit_transform(X, y)
15
16 X_train, X_test, y_train, y_test = train_test_split(
17     X_new, y, test_size=0.2, random_state=42)
18
19 model = LogisticRegression(max_iter=200)
20 model.fit(X_train, y_train)
21 y_pred = model.predict(X_test)
22
23 print(confusion_matrix(y_test, y_pred))
```

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

## 2.2 Results

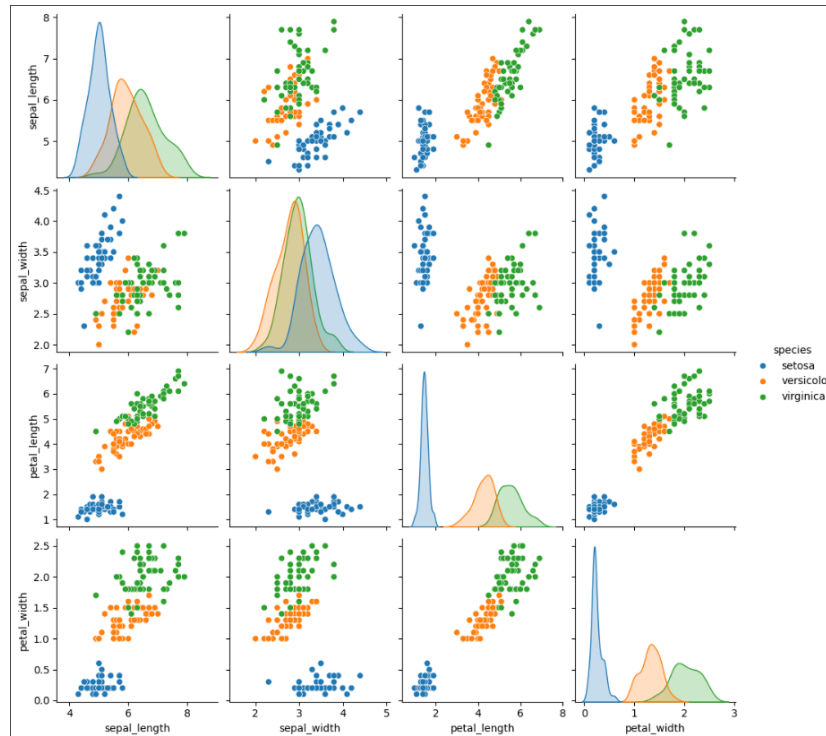


Figure 1: Pairplot of Iris dataset showing feature distributions and class separation.

- Algorithm: Logistic Regression
- Result: 97% accuracy with strong precision and recall across all classes.

## 2.3 Results

- Algorithm: Logistic Regression
- Result: 97% accuracy with strong precision and recall across all classes.

# 3 Loan Amount Prediction - Regression

## 3.1 Python Code

Listing 2: Linear Regression for Loan Prediction

```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.linear_model import LinearRegression
4 from sklearn.preprocessing import StandardScaler
5 from sklearn.metrics import mean_squared_error, r2_score
6
7 df = pd.read_csv('/content/drive/MyDrive/train.csv')
8 df.drop(columns=["Customer ID", "Name", "Property ID"],
9         inplace=True)
10 df.dropna(inplace=True)
11
12 X = pd.get_dummies(df.drop(columns=["Loan Sanction
13                               Amount (USD)"]), drop_first=True)
14 y = df["Loan Sanction Amount (USD)"]
15 X = StandardScaler().fit_transform(X)
16
17 X_train, X_test, y_train, y_test = train_test_split(X, y,
18                                                     , test_size=0.2, random_state=42)
19 model = LinearRegression()
20 model.fit(X_train, y_train)
21
22 y_pred = model.predict(X_test)
23 print("RMSE:", mean_squared_error(y_test, y_pred))
24 print("R2 Score:", r2_score(y_test, y_pred))
```

## 3.2 Results

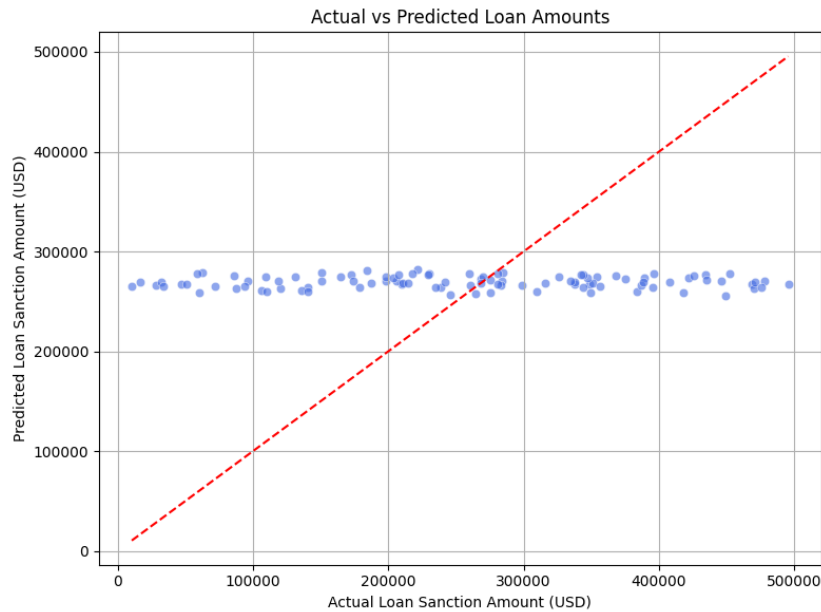


Figure 2: Actual vs Predicted Loan Sanction Amounts.

- RMSE = 1.19B,  $R^2 = 0.47$  – moderate regression performance.

## 4 Diabetes Prediction - Classification

### 4.1 Python Code

Listing 3: Random Forest for Diabetes Prediction

```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.preprocessing import StandardScaler
4 from sklearn.ensemble import RandomForestClassifier
5 from sklearn.metrics import classification_report,
6   accuracy_score, confusion_matrix
7
8 df = pd.read_csv('/content/drive/MyDrive/diabetes.csv')
9 X = df.drop('Outcome', axis=1)
10 y = df['Outcome']
11
12 X_scaled = StandardScaler().fit_transform(X)
```

```

12 X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.25, random_state=42)
13
14 clf = RandomForestClassifier()
15 clf.fit(X_train, y_train)
16
17 y_pred = clf.predict(X_test)
18 print(accuracy_score(y_test, y_pred))
19 print(classification_report(y_test, y_pred))

```

## 4.2 Results

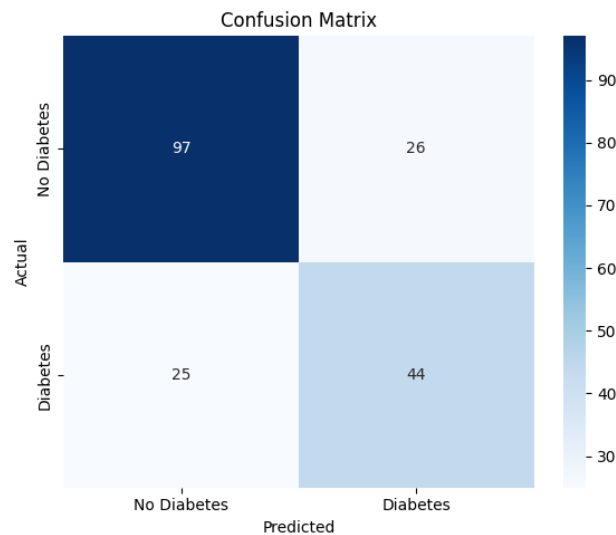


Figure 3: ROC Curve showing model performance with  $AUC = 0.80$ .

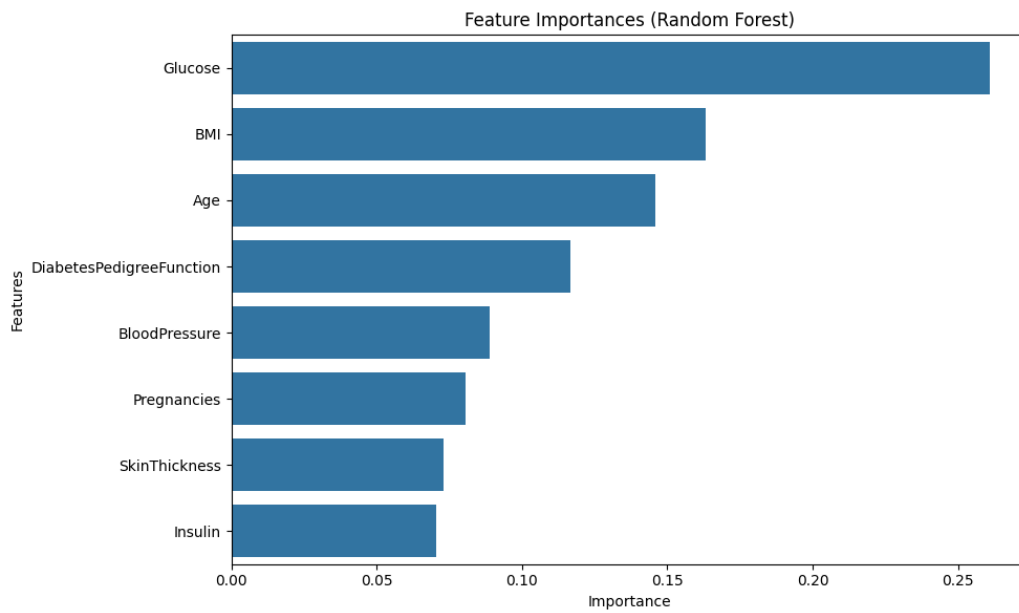


Figure 4: Feature importance of predictors using Random Forest.

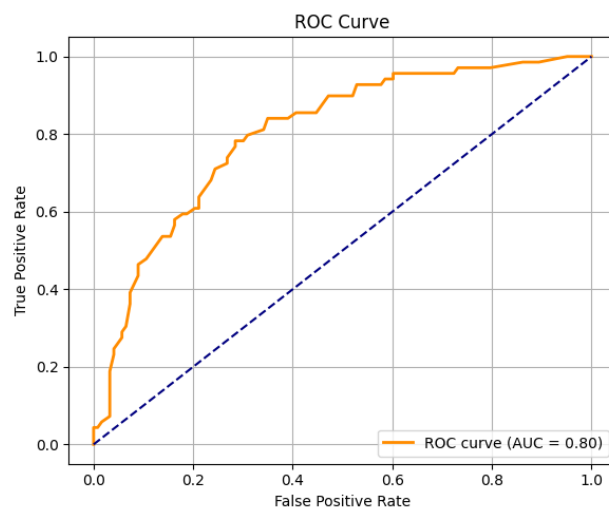


Figure 5: Confusion matrix for diabetes prediction model.

- Accuracy = 74%. Class 0 (No Diabetes) predicted better than Class 1 (Diabetes).



## 5 Spam Email Detection - Classification

### 5.1 Python Code

Listing 4: Naive Bayes for Spam Classification

```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.naive_bayes import GaussianNB
4 from sklearn.metrics import classification_report,
   accuracy_score
5
6 df = pd.read_csv("/content/drive/MyDrive/spambase.data",
   header=None)
7 X = df.iloc[:, :-1]
8 y = df.iloc[:, -1]
9
10 X_train, X_test, y_train, y_test = train_test_split(X, y
   , test_size=0.2, random_state=0)
11 model = GaussianNB()
12 model.fit(X_train, y_train)
13
14 y_pred = model.predict(X_test)
15 print(accuracy_score(y_test, y_pred))
16 print(classification_report(y_test, y_pred))
```

## 5.2 Results

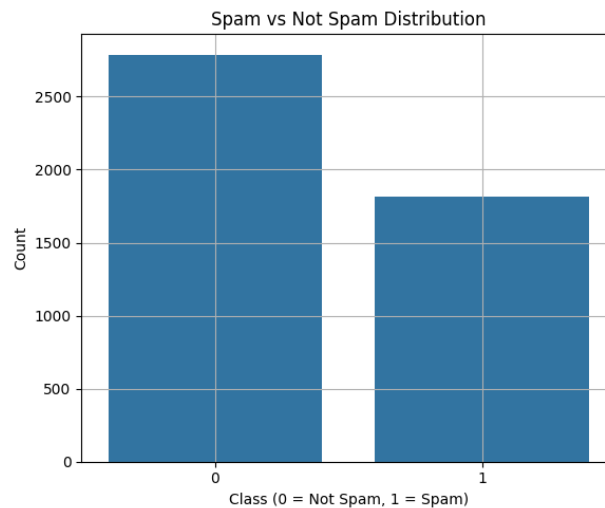


Figure 6: Distribution of Spam vs. Not Spam messages.

- Accuracy = 81%, with good recall for spam messages.

## 6 MNIST Digit Recognition

### 6.1 Python Code

Listing 5: Logistic Regression on MNIST Digits

```
1 from sklearn.datasets import load_digits
2 from sklearn.linear_model import LogisticRegression
3 from sklearn.model_selection import train_test_split
4 from sklearn.metrics import classification_report,
   accuracy_score
5
6 digits = load_digits()
7 X, y = digits.data, digits.target
8
9 X_train, X_test, y_train, y_test = train_test_split(X, y
   , test_size=0.3, random_state=42)
10 model = LogisticRegression(max_iter=3000)
11 model.fit(X_train, y_train)
12
13 y_pred = model.predict(X_test)
```

```

14 print(accuracy_score(y_test, y_pred))
15 print(classification_report(y_test, y_pred))

```

## 6.2 Results

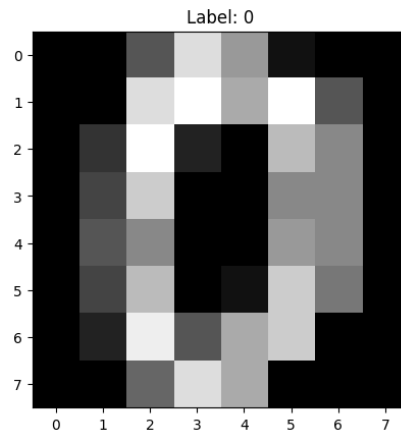


Figure 7: Sample MNIST digit image with label.

- Accuracy = 97%. Robust digit classification with Logistic Regression.

## 7 Discussion

Dataset	ML Task	Algorithm Used
Iris	Classification	Logistic Regression
Loan Prediction	Regression	Linear Regression
Diabetes	Classification	Random Forest
Spam Emails	Classification	Naive Bayes
MNIST Digits	Classification	Logistic Regression

Table 1: Overview of ML experiments.

## 8 Conclusion

The experiments demonstrated how different ML models perform on tasks such as diabetes prediction, spam detection, digit recognition, and loan prediction.

Key takeaways:

- Understood data preprocessing and feature selection.
- Learned to apply regression and classification models.
- Gained experience with evaluation metrics: Accuracy, Precision, Recall, F1-score, AUC, RMSE,  $R^2$ .
- Identified strengths and limitations of Logistic Regression, Naive Bayes, and Random Forest.