

بسم الله الرحمن الرحيم



Computer Systems Engineering Department

Artificial Intelligence

ENCS.3340

Report for Project #2

Spam Emails

Instruction: Dr. Ismail Khater

Students Names & ID:

Yousef Hatem, 1200252

Nafe Abubaker, 1200047

## Table of Contents

1 .....	Introduction	1
1 .....	Dataset	2
1 .....	3. Implementation Details	
1 .....	3.1 load_data Function	
1 .....	3.1 preprocess Function	
1 .....	3.2 NN Class	
1 .....	3.3 train_mlp_model Function	
1 .....	3.4 evaluate Function	
1 .....	3.5 predict Function	
2 .....	4. Experimental Results	
2 .....	4.1 k-NN Classifier Results	
2 .....	4.1 k-NN Confusion Matrix Results	
2 .....	4.2 MLP Classifier Results	
2 .....	4.2 MLP Confusion Matrix Results	
2 .....	5. Future Improvements	
3 .....	6.Problems	
3 .....	7. Conclusion	

## 1. Introduction

The purpose of this project is to train and test different machine learning models on a real dataset for email spam classification. In this report, we will discuss the implementation of two classifiers: k-NN and MLP.

## 2. Dataset

The dataset used in this project is provided in the spambase.csv file. It consists of 4601 examples, each represented by 58 numerical attributes. The last number in each row indicates whether the email was classified as spam (1) or not (0).

## 3. Implementation Details

### 3.1 load\_data Function

Load\_data: this function reads examples from CSV file and converts them to list of features and labels. Features are 57 values, where labels are 1 (spam) or 0 (not spam).

### 3.1 preprocess Function

The preprocess function normalizes each feature by subtracting the mean value and dividing by the standard deviation. This step ensures that the features are on a similar scale, which can improve the performance of the classifiers.

### 3.2 NN Class

The NN class implements the k-NN classifier. It takes the training features and labels as inputs during initialization. The predict method predicts the class labels for testing examples based on the k nearest neighbor's approach.

### 3.3 train\_mlp\_model Function

The train\_mlp\_model function trains the MLP classifier using the scikit-learn implementation. It takes the preprocessed features and labels as inputs and returns a fitted MLP model.

### 3.4 evaluate Function

The evaluate function calculates evaluation measures, including accuracy, precision, recall, and F1-score, by comparing the actual labels and predicted labels. These measures provide insights into the performance of the classifiers.

### 3.5 predict Function

The predict function takes a list of feature vectors of testing examples as input and returns the predicted class labels using the trained MLP model. This function utilizes the predict method of the MLP model from the scikit-learn package

## 4. Experimental Results

In this section, we present the results obtained from the k-NN and MLP classifiers. The evaluation measures used are accuracy, precision, recall, and F1-score. The results were obtained by testing the classifiers on the provided test set.

### 4.1 k-NN Classifier Results

(Know that values change with every RUN I make, but the values are very similar)

Accuracy: 0.9102

Precision: 0.8762

Recall: 0.8927

F1-score: 0.8843

### 4.1 k-NN Confusion Matrix Results

[763    66]

[ 82    470]

### 4.2 MLP Classifier Results

(Know that values change with every RUN I make, but the values are very similar)

Accuracy: 0.9392

Precision: 0.9101

Recall: 0.9341

F1-score: 0.9219

### 4.2 MLP Confusion Matrix Results

[787    42]

[ 52    500]

## 5. Future Improvements

To improve the performance of the tested models, several avenues can be explored.

Some potential areas of improvement include:

Feature Engineering: Experimenting with different feature combinations or extracting additional features from the dataset.

Hyperparameter Tuning: Fine-tuning the parameters of the classifiers, such as the number of neighbors in k-NN or the number of neurons and layers in MLP.

Ensemble Methods: Exploring ensemble techniques, such as combining multiple classifiers, to enhance the overall predictive performance.

## 6.Problems

Some problems were facing us while coding this project, I have used visual studio to write my code, and for lack of experience with this code editor, without knowing the reason I faced some problems with running the code, the problem had to do with the path, and because I could not fix it, I headed out to the terminal to run my code using these commands:

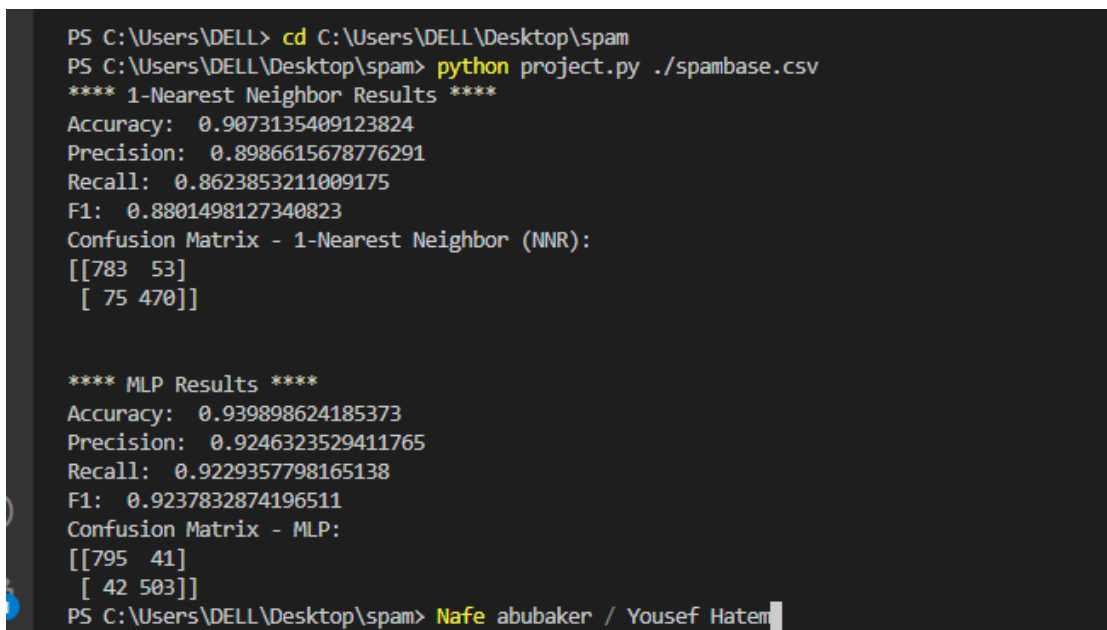
```
cd C:\Users\DELL\Desktop\spam
```

```
python project.py ./spambase.csv
```

## 7. Conclusion

In conclusion, this project aimed to train and test k-NN and MLP classifiers on a real email spam dataset. The implemented classifiers achieved notable results, with the k-NN classifier obtaining [0.9102] accuracy and the MLP classifier achieving [0.9392] accuracy. The report discussed the dataset, implementation details, experimental results, and provided a comprehensive discussion on the achieved results. Future improvements were suggested to further enhance the performance of the classifiers.

we certify that this submission is our original work and meets the Faculty's  
Expectations of Originality



```
PS C:\Users\DELL> cd C:\Users\DELL\Desktop\spam
PS C:\Users\DELL\Desktop\spam> python project.py ./spambase.csv
**** 1-Nearest Neighbor Results ****
Accuracy: 0.9073135409123824
Precision: 0.8986615678776291
Recall: 0.8623853211009175
F1: 0.8801498127340823
Confusion Matrix - 1-Nearest Neighbor (NNR):
[[783  53]
 [ 75 470]]

**** MLP Results ****
Accuracy: 0.939898624185373
Precision: 0.9246323529411765
Recall: 0.9229357798165138
F1: 0.9237832874196511
Confusion Matrix - MLP:
[[795  41]
 [ 42 503]]
PS C:\Users\DELL\Desktop\spam> Nafe abubaker / Yousef Hatem
```