Report

Università degli Studi di Cagliari

Email spam detection

Project for Artificial intelligence course

Students: Marco Ledda, Khajavali Shaik

Contents

1. Introduction

Email spam has been a persistent problem for individuals and businesses for many years. Spam emails can lead to wasted time and resources, increased risk of cyberattacks, and decreased productivity. As the volume of email traffic continues to grow, the need for effective spam detection techniques becomes even more pressing.

Machine learning has emerged as a promising solution for email spam detection, as it can automatically learn from large volumes of email data and identify patterns that distinguish between spam and legitimate emails. In this project, we aimed to develop an email spam detector based on machine learning techniques.

In this report, we present our approach for building a spam detector using two machine learning algorithms, including decision trees and artificial neural network and we have also the opportunity to compare the results. We trained and evaluated our models on a large dataset of email messages, which includes both spam and legitimate emails.

Our results demonstrate that our machine learning-based spam detector outperforms traditional rule-based approaches and achieves high accuracy and precision in detecting spam emails. We also discuss the limitations of our approach.

1.1 Structure of the project

The first step is to explore the dataset. This involves analyzing the data to understand its characteristics, such as the columns and the number of samples. Exploring the dataset also help us to identify any potential issues or challenges that may need to be addressed during the pre-processing stage.

The second step is to pre-process the dataset. Pre-processing involves cleaning the data and transforming it into a format that can be used for analysis. Specifically we have written a function which removes URLs, punctuation, numbers and the stopwords that are irrelevant for the classificationt taks. Furthermore we have performed a stemming which consists of reporting verbs to their basic form.

The third step is to perform a k-fold cross-validation. Cross-validation involves partitioning the dataset into k subsets and using k-1 subsets for training and the remaining subset for testing. This process is repeated k times, with each subset being used as the test set once. We have randomly subdivided into 5 disjoint subsets of identical size and we have used the CountVectorizer class proposed by Scikit-Learn library for converting text into a matrix of token counts. The fit method of this class allows us to learn a vocabulary dictionary of all tokens in the raw documents. After that we have computed the information gain for each word of the training set, using this formula:

$$\text{Information Gain} = \sum_{c_i \in c_i, \overline{c_i}} \sum_{t_k \in t_k, \overline{t_k}} P(t,c) \log \frac{P(t,c)}{P(t)P(c)} \tag{1}$$

The formula 1 is implemented through the function ”mutual info classif” proposed in Scikit-Learn library. This function takes in input the training fold and the labels vector and returns the score for each word present in the dataset. In this way we can sort in descending order the scores related to each word and select the most N influent words of the dataset. To realize the final feature vector of size N, (where N is the number of features) to be passed to the classifier we have to count how many times a word of the score vector is present in the training and testing fold.

Thenceforth, we have build and trained a Decision Tree and an Artificial Neural Network using respectively the classes ”DecisionTree” and ”MLPClassifier” proposed in Scikit-Learn library. Finally we have tested the two classifiers passing data that they haven’t seen during the training. We must specify that we have tested the two classifiers for each test fold by varying the number of features (N).

2. Experimental analysis

Our project is focused on developing an email spam detector using two different approaches - decision tree and artificial neural network. While the primary objective is to implement a functional spam detection system, we also aim to compare the performance of the two models in terms of misclassification rate. This comparison will provide us with valuable insights into the strengths and weaknesses of each approach and help us understand which model is more effective in identifying spam emails.

2.1 Experimental setup

Dataset. The dataset we have used is the ”Complete Spam Assassin” composed of 5456 samples both of train and test. During the k-fold cross validation, the training fold is composed of 4365 samples and the testing fold is composed of the remaining 1091 samples. Furthermore, the dataset is composed of two columns: e-mail bodies and label (class 1 means that the email is categorized as ”Spam”, class 0 means that the email is categorized as ”Ham”).

Models. We have used two models proposed in Scikit-Learn library:

- Decision Tree;
- MLP Classifier.

Hyperparameters. The hyperparameters of the Decision Tree are mainly two:

- The criterion of splitting, that has been set to ’log loss’. This parameter measures the quality of a split;
- The max depth, that has been set to 70. The motivation under this tuning is that we have seen better performances in terms of accuracy of classification.

Regard to the Artificial Neural Network, we have created an architecture of three layers:

- An input layer composed of N units;
- An hidden layer composed of $\sqrt{N}$ units;

- An output layer composed of only one unit.

Thus, varying the number of features, we are varying also the number of units of the input and hidden layer. Furthermore we have set the activation function to 'logistic' and the number of epochs performed in the backward propagation to 1000. The motivation under the tuning of the epochs in the backward propagation is as well we have seen better performances in terms of accuracy of classification.

Evaluation criteria. For evaluating these two models we have taken into account the misclassification rates. We have computed the misclassification rates as:

$$\text{Misclassification rate} = \frac{\text{Number of misclassified samples}}{\text{Total number of samples}} \tag{2}$$

The number of misclassified samples is computed as the sum of the false positive and false negative extracted from the confusion matrix computed by the apposite function proposed by Scikit-Learn library.

To evalute the models we have also plotted the ROC curve: a ROC (Receiver Operating Characteristic) curve is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The TPR is defined as:

$$\text{TPR} = \frac{\text{TP}}{\text{TP+FN}} \tag{3}$$

where TP is the number of true positive cases and FN is the number of false negative cases.

The FPR is defined as:

$$\text{FPR} = \frac{\text{FP}}{\text{TN+FP}} \tag{4}$$

where FP is the number of false positive cases and TN is the number of true negative cases.

2.2 Experimental results

We used roughly 1091 test samples to see the classification performance of the models used. As far as the decision tree is concerned, we can see that the misclassification rate does not show a notable decrease as the number of features increases. Instead, seeing at the neural network performances we can see that is more accurate than the Decision Tree at classifying for every values of features taken into consideration (N=100, 500, 1000). We can also see a significant decrease of the misclassification rate as the number of features increases.

The figures below shown the expressed misclassification rate as a percentage for both the decision tree (1) and the neural network (8):

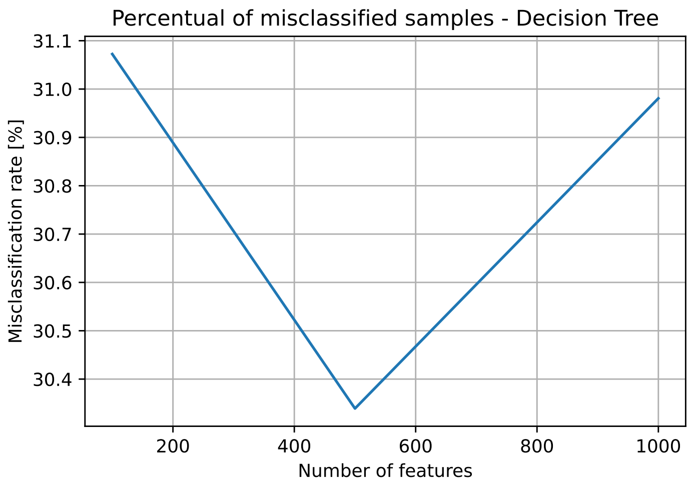

Figure 1: Misclassification rate as a percentage for the Decision Tree

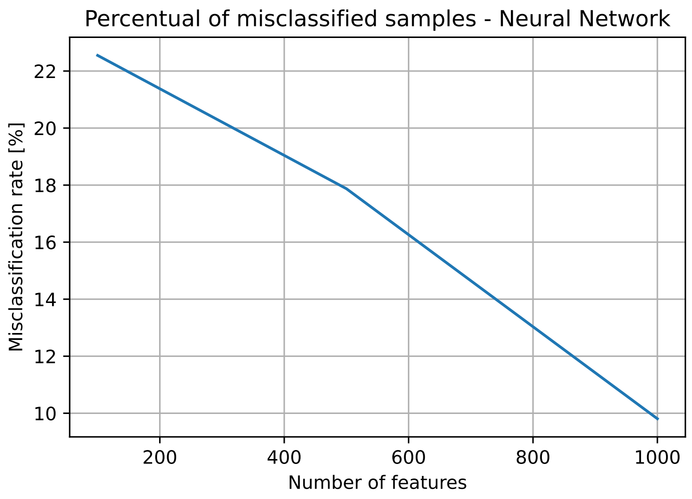

Figure 2: Misclassification rate as a percentage for the Artificial Neural Network

In addition to the misclassification rate curves as a function of the number of features, we also plotted the ROC curves for each value of features considered for both models. We decided to plot the ROC curves to get a more complete overview of the performance of the models taken into consideration terms of sensitivity and specificity.

The figures below are the ROC Curves for each number of features taken into account and for each model:

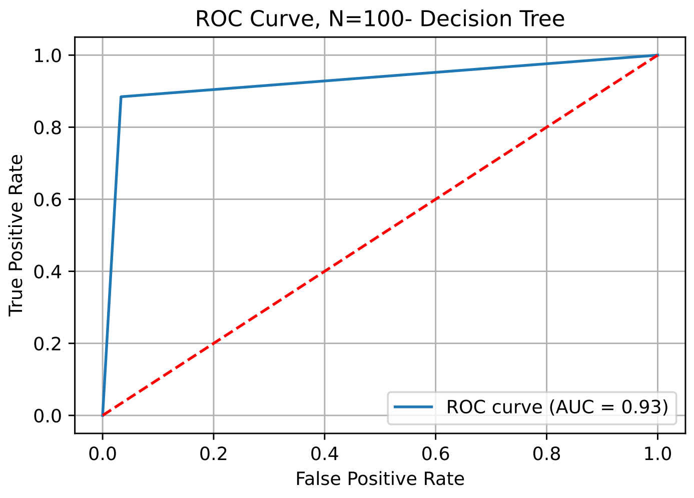

Figure 3: ROC Curve for N = 100 - Decision Tree

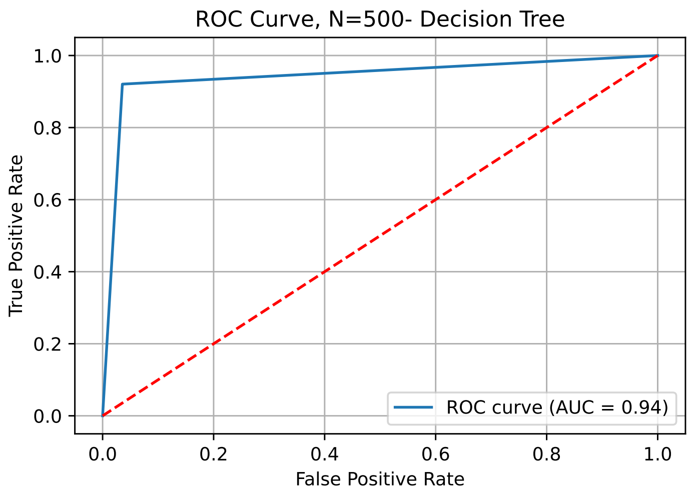

Figure 4: ROC Curve for N = 500 - Decision Tree

2.3 Limitations

Our experimental analysis has several limitations that can affect the accuracy of its results. One of the most significant limitations is the number of samples available for analysis. Increasing the number of samples can improve the accuracy of the project's results as it would provide more data for analysis, allowing us to make more accurate predictions.

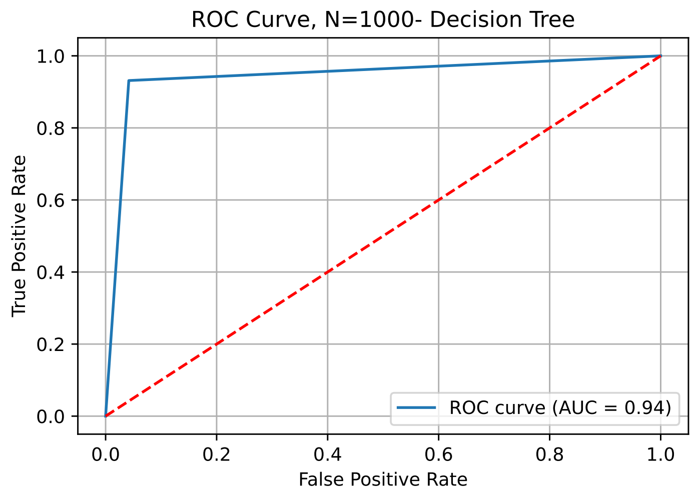

Figure 5: ROC Curve for N = 1000 - Decision Tree

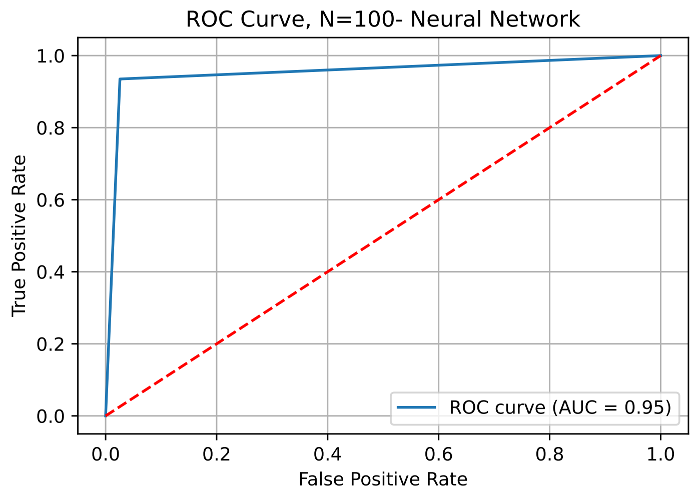

Figure 6: ROC Curve for N = 100 - Neural Network

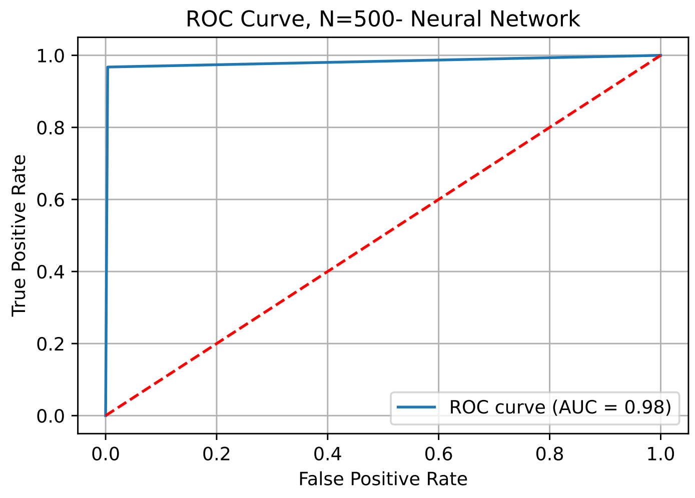

Figure 7: ROC Curve for N = 500 - Neural Network

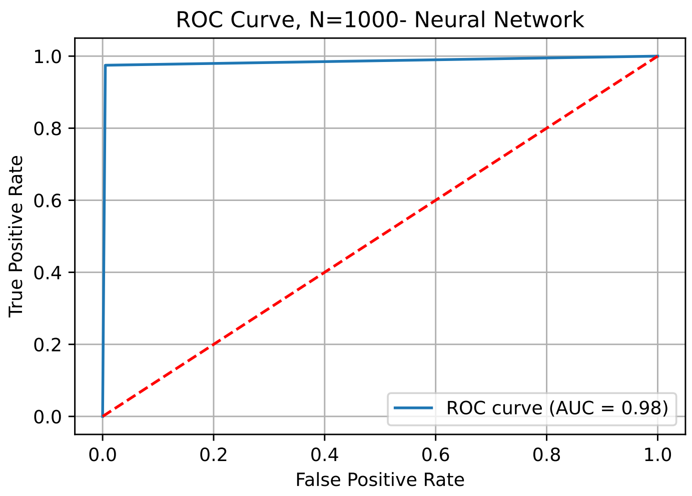

Figure 8: ROC Curve for N = 1000 - Neural Network

Another limitation of the project is that the features used for analysis may not be sufficient to accurately predict the outcome. To improve the accuracy of the project, we can extract better features that capture more relevant information.

Finally, the project's accuracy can be improved by using deeper artificial neural networks. Deeper neural networks can capture more complex relationships between the input data and the output, improving the accuracy of the model. However, using deeper neural networks requires more computational power, and the training process can be more time-consuming. For this motivation we thought it appropriate to use a very simple structure of the net with a single hidden layer.

In conclusion, while our project has limitations, we can increase the accuracy of its results by increasing the number of samples, extracting better features, and using deeper artificial neural networks. By addressing these limitations, we can create a more robust and accurate model for our analysis.