Assignment 1
Comparison of Machine Learning algorithms for spam classification

T-710-MLCS, Machine Learning in Cyber Security

Reykjavik University - School of Computer Science

Federico Maria Cruciani

federico24@ru.is

3. September 2024

1 Introduction

This report covers the work done for Assignment 1 of the Machine Learning in Cyber Security course. The goal of the assignment was to compare three supervised learning algorithms for the classification of spam messages and analyse how hyper-parameters influence their effectiveness. The algorithms are:

- Naive Bayes
- K-Nearest Neighbours (KNN)
- Logistic Regression

The following chapters delve into what experiments were conducted, how to run the Python code and some implementation details.

2 Experiments

The first three experiments conducted were focused on analysing the influence that hyper-parameters have on machine learning algorithms. Each model was tested on different values of one of its hyper-parameters and the results were compared by using the following metrics:

- Accuracy
- Precision
- Recall
- F1-Score
- Area Under the ROC Curve (AUC)

Additionally, the ROC Curve was plotted for each hyper-parameter value.

2.1 Influence of the dictionary size

This experiment compares the impact of different dictionary sizes on the Naive Bayes classifier. The values were taken from the set $\{100, 500, 1000, 2000, 3000\}$.

The dictionary size which yielded the best results was then used for the subsequent experiments.

2.2 Influence of the k parameter on KNN

This experiment aims at comparing the influence of the value k on the performance of the K-Nearest Neighbours classifier. The compared values were taken from the set $\{4, 6, 8, 10, 15, 20\}$.

2.3 Influence of hyper-parameters on Logistic Regression

This experiment, similarly to the previous, compares different values of one of the hyper-parameters of the Logistic Regression classifier. The parameter is the regularization strength (called 'C' in the scikit-learn library) and was chosen from the set $\{0.001, 0.4, 4, 20, 100\}$.

In addition, the metrics for the training set were also analysed to check for overfitting.

2.4 Comparison

This final experiment compares the three analysed models in order to find the best performing one. Each algorithm was trained using the hyper-parameter value which performed the best based on the previous experiments.

3 Running the experiments

Inside the project folder, each experiment is separated into its own Python file:

- *bayes.py* for the Naive Bayes experiment
- *knn.py* for the K-Nearest Neighbours experiment
- *logistic.py* for the Logistic Regression experiment
- *comparison.py* for the final comparison

It is possible to run each experiment by executing its file directly, for example for the Naive Bayes experiment:

```
python3 bayes.py
```

Alternatively the project contains a script called *runner.py* that can run all the experiments at once, or only one of them by specifying its name. To launch all the experiments simply execute the script with

```
python3 runner.py
```

The dataset folders must be placed inside the directory of the project.

4 Results

4.1 Dictionary size

The metrics obtained from the first experiment, shown in table 1, show that the Naive Bayes classifier has almost the exact same performance for dictionary sizes of 1000, 2000 and 3000, the only discriminant being the ROC AUC value that differs by only 0.001 for the three values.

Given this, the best value for the dictionary size was chosen to be 3000.

Size	Accuracy	Precision	Recall	F1 score
100	0.9423	0.9675	0.9154	0.9407
500	0.9462	0.9915	0.9000	0.9435
1000	0.9615	0.9918	0.9308	0.9603
2000	0.9615	0.9918	0.9308	0.9603
3000	0.9615	0.9918	0.9308	0.9603

Table 1: Naive Bayes metrics based on dictionary size

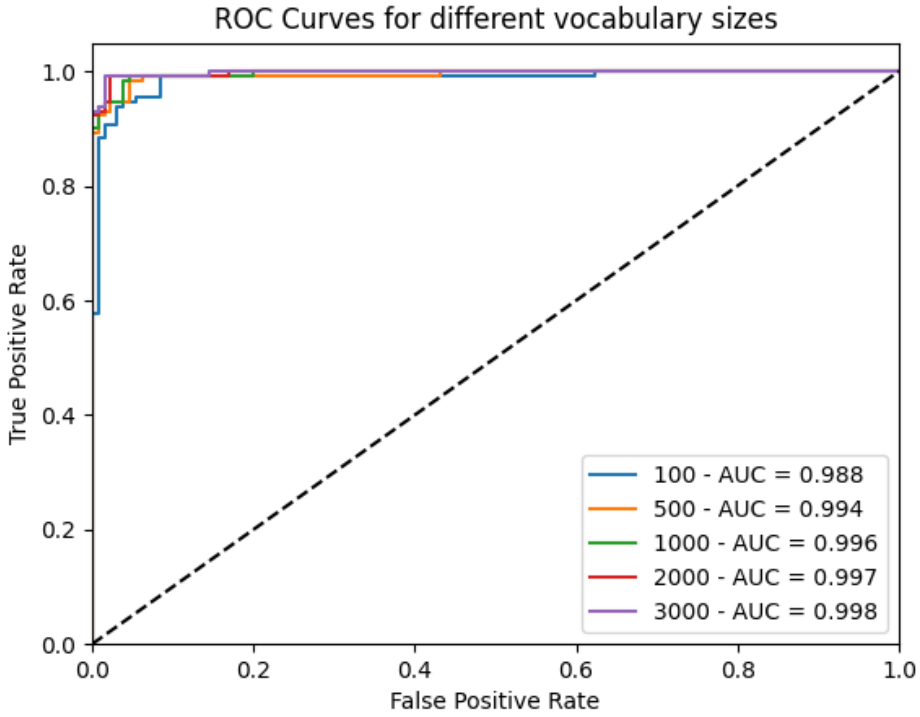

Figure 1: Naive Bayes ROC Curves

4.2 k parameter in KNN

The results of the K-Nearest Neighbours, shown in table 2, are much more clear than those of the Naive Bayes, in fact we can see that the model with $k = 10$ outperforms all the others, having better metrics almost all across the board. The only other comparable result is that of $k = 15$, which has the same Recall and a slightly better AUC value, but considering the other metrics in the end the chosen value was $k = 10$.

k	Accuracy	Precision	Recall	F1 score
4	0.9308	0.9308	0.9308	0.9308
6	0.9346	0.9248	0.9462	0.9354
8	0.9077	0.8681	0.9615	0.9124
10	0.9346	0.8951	0.9846	0.9377
15	0.8846	0.8205	0.9846	0.8951
20	0.9038	0.8523	0.9769	0.9104

Table 2: KNN metrics based on k parameter

4.3 Logistic Regression

As shown in table 3, the comparison of the values of the regularization strength once again gives ambiguous results: for C = 0.4 and C = 4 all the metrics, AUC included, have exactly the same value. This can be better explained by looking at the metrics for the training set in table 4, which show that the Logistic Regression model overfits for almost all C values and still has bad generalization for C = 0.001.

Considering this, the chosen value for the final comparison was C = 0.001.

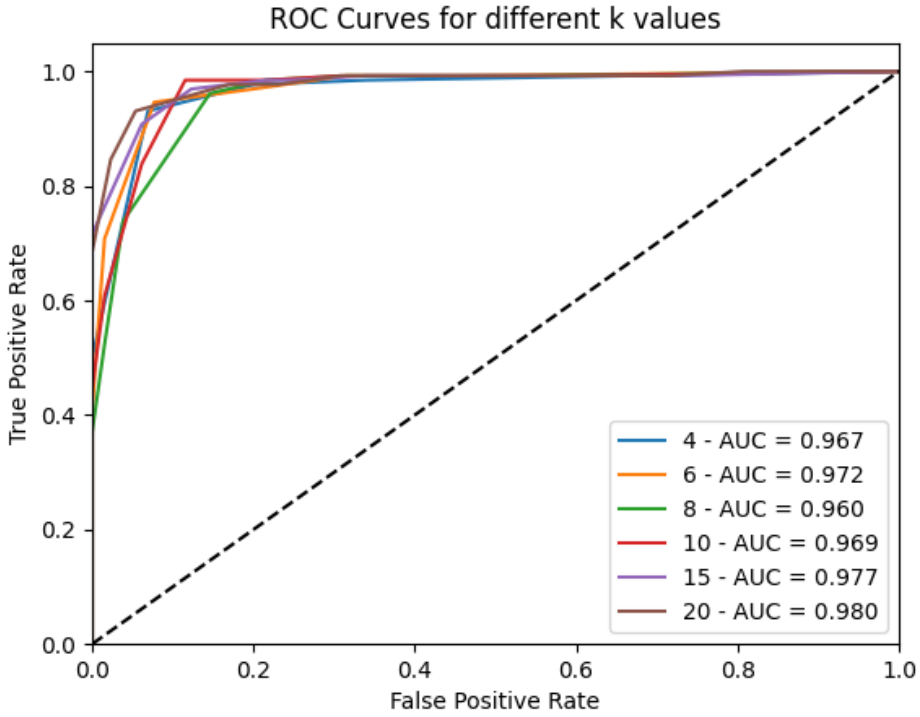

Figure 2: K-Nearest Neighbours ROC Curves

C	Accuracy	Precision	Recall	F1 score
0.001	0.9077	0.9818	0.8308	0.9000
0.4	0.9731	0.9424	0.9846	0.9734
4	0.9731	0.9624	0.9846	0.9734
40	0.9731	0.9695	0.9769	0.9732
100	0.9692	0.9621	0.9769	0.9695

Table 3: Logistic Regression metrics based on regularization strength

	Test set		Training set	
C	Recall	Precision	Recall	Precision
0.001	0.8308	0.9818	0.8803	0.9364
0.4	0.9846	0.9424	1.0000	1.0000
4	0.9846	0.9624	1.0000	1.0000
40	0.9769	0.9695	1.0000	1.0000
100	0.9769	0.9621	1.0000	1.0000

Table 4: Recall and precision metrics of test and training sets

4.4 Comparison

The comparison of the best results from the other experiments shows mixed results. The KNN classifier has the highest Recall score, but has much lower precision and AUC values compared to the others. The Naive Bayes algorithm instead has the best AUC value, which is very close to 1, and the best precision, accuracy and F1, but lower recall.

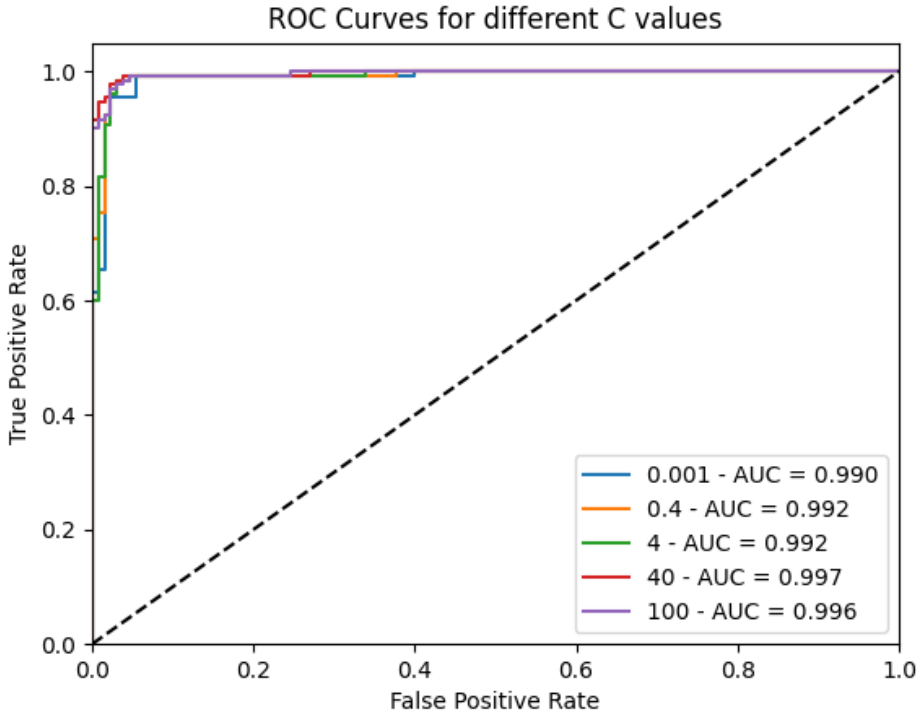

Figure 3: Logistic Regression ROC Curves

Model	Accuracy	Precision	Recall	F1 score
Bayes	0.9615	0.9918	0.9308	0.9603
KNN	0.9346	0.8951	0.9846	0.9377
Logistic	0.9077	0.9818	0.8308	0.9000

Table 5: Metrics of all the considered models

5 Conclusions

In conclusion, the experiments showed that hyper-parameters can have a big impact on the performance of a machine learning model, and also that there is not a definitive best algorithm for spam classification in this case. In fact, results show that the considered models can be good in one area but worse in another.

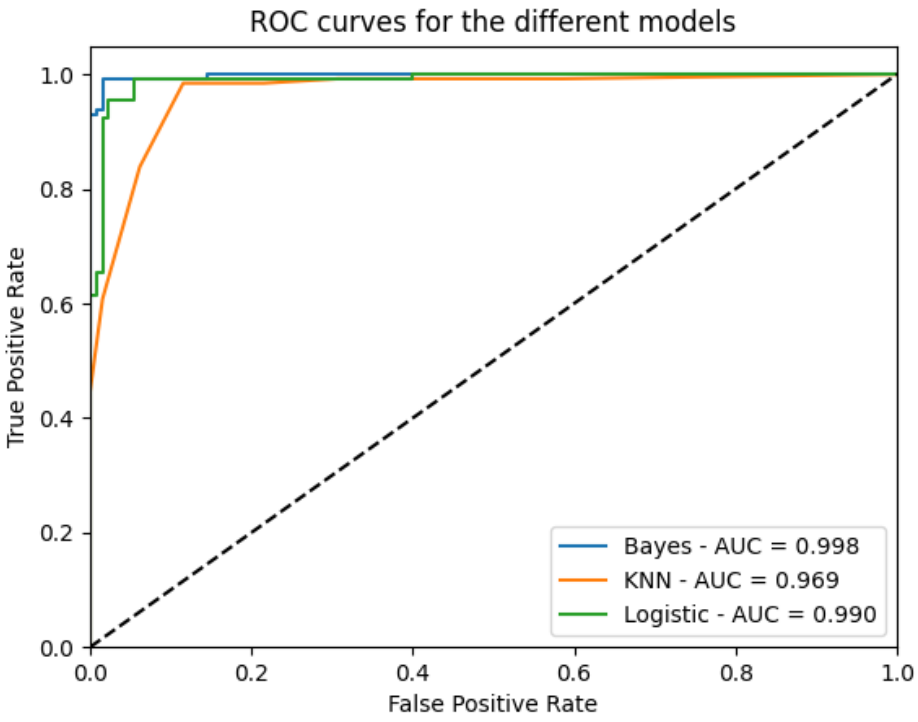

Figure 4: ROC curves of all the models