HÁSKÓLINN Í REYKJAVÍK
REYKJAVIK UNIVERSITY

Computer Science department

Machine Learning in Cybersecurity

T-710-MLCS

Report 1 - Spam Classification Experiments

Saldana Giorgio
Email: giorgio24@ru.is

3rd September 2024

Contents

1 Introduction

Spam classification is a crucial task in modern communication systems, where distinguishing between legitimate and spam messages can significantly enhance user experience and security. This report details a series of experiments conducted to evaluate the performance of three different supervised learning algorithms—Naive Bayes, K-Nearest Neighbors (KNN), Support Vector Machine (SVM) , and Logistic Regression—on a spam classification problem. The study aims to compare these models under varying conditions and hyperparameters, providing insights into their strengths and weaknesses.

The report is structured as follows:

- **Reproducibility**: Describe how to run the code to reproduce the metrics and evaluate the models for spam detection.
- **Materials and Methods**: Describes the tools and methodologies used in the experiments.
- **Results**: Presents the outcomes of the experiments, including key metrics and visualizations.
- **Discussion**: Analyzes the results and provides insights into the performance of each classifier.
- **Conclusion and Future Work**: Summarizes the findings and suggests potential directions for further research.

2 Reproducibility

To ensure that others can replicate the experiments described in this report, detailed instructions for setting up the environment, obtaining the data, and running the code are provided below. The full source code for all experiments is available in the following GitHub repository: `https://github.com/giorgiosld/Machine-Learning-in-Cybersecurity`.

2.1 Environment Setup

- **Python Version**: Ensure that you have Python 3.x installed. The code was developed and tested on Python 3.11.
- **Required Libraries**: Install the necessary Python libraries using pip:

```
pip install scikit-learn matplotlib numpy pandas
```

- **Version Control**: It is recommended to clone the GitHub repository containing the code and avoiding problem with the path of datasets:

```
git clone https://github.com/giorgiosld/Machine-Learning-in-Cybersecurity.gi
```

2.2 Data Acquisition

- Download the preprocessed dataset from the following link: `https://www.dropbox.com/s/yjiplngoa430rid/ling-spam.zip`
- Extract the dataset into the '../dataset/' directory so one level outside the zip file or follow next instructions.

2.3 Running the Experiments

- Navigate to the directory where the repository was cloned:

```
cd Machine-Learning-in-Cybersecurity/spam_detection/assignment
```

- Or extract the file and navigate to the directory:

```
cd assignment
```

- Activate the virtual environment if you cloned my repository:

```
pip3 install -r ../../requirements.txt
source ../../venv/bin/activate
```

- To execute each model separately, run the corresponding Python script:
  - **Naive Bayes Experiment**:

    ```
    PYTHONPATH=./ python3 models/naive_bayes_classifier.py
    ```

  - **KNN Experiment**:

    ```
    PYTHONPATH=./ python3 models/knn_classifier.py
    ```

  - **Logistic Regression Experiment**:

    ```
    PYTHONPATH=./ python3 models/logistic_regression_classifier.py
    ```

  - **Support Vector Machine**

    ```
    PYTHONPATH=./ python3 models/svm_classifier.py
    ```

- To execute all models together and print the metrics in the terminal, use:

```
python evaluator.py
```

  This script will run all three models, calculate the metrics (Accuracy, Precision, Recall, F1-score, AUC-ROC), and display the results directly in the terminal.
- After running each experiment, the results (including plots) will be saved in the 'resources/' directory.

2.4 Reproducing the Results

- The plots and metrics obtained from each experiment can be found in the ‘results/‘ directory. Use these to verify the reproducibility of the results discussed in this report.
- Ensure that your environment matches the versions of Python and the required libraries to avoid discrepancies in the results.
- Is also possible to modify the **PATH** of dataset directory through the variable PATH that is present in each files.

3 Materials and Methods

This section outlines the tools, libraries, and methodologies employed in conducting the experiments.

3.1 Materials

- **Python**: The primary programming language used for implementing the models and conducting the experiments.
- **Scikit-Learn**: A machine learning library in Python that provides tools for data preprocessing, model training, and evaluation.
- **Matplotlib**: Libraries used for plotting the ROC curves and other visualizations.
- **Git and GitHub**: Version control systems used for tracking changes and collaborating on the codebase.

3.2 Methodology

The methodology section is divided into several experiments, each focusing on a specific aspect of model performance. The steps taken for data preprocessing, model training, and evaluation are detailed below.

3.2.1 Data Preprocessing

- **Text Cleaning**: Removed the "Subject:" prefix, non-alphabetic characters, and single-character words from the emails.
- **Vectorization**: Converted the cleaned text data into numerical vectors using a bag-of-words model.

3.2.2 Experiment 1: Influence of Dictionary Size on Naive Bayes Classifier

- **Tested dictionary sizes**: 100, 500, 1000, 2000, 3000.
- **Evaluated metrics**: Accuracy, Precision, Recall, F1-score, AUC-ROC.
- **ROC curves plotted** for each dictionary size.

3.2.3 Experiment 2: Influence of K Parameter on KNN Classifier

- **Tested K values**: 4, 6, 8, 10, 15, 20.
- **Evaluated metrics**: Accuracy, Precision, Recall, F1-score, AUC-ROC.
- **ROC curves plotted** for each K value.

3.2.4 Experiment 3: Influence of Regularization Parameter (C) on Logistic Regression

- **Tested C values**: Five significantly different values.
- **Evaluated metrics**: Accuracy, Precision, Recall, F1-score, AUC-ROC.
- **ROC curves plotted** for each C value.

3.2.5 Experiment 4: Comparison of Training vs. Test Performance in Logistic Regression

- **Comparison**: Precision and recall on training and test datasets for the selected C values.
- **Analysis**: Identified signs of overfitting.

3.2.6 Experiment 5: Influence of kernel on Support Vector Machine

- **Tested kernels**: 'linear', 'rbf', 'poly', 'sigmoid'.
- **Evaluated metrics**: Accuracy, Precision, Recall, F1-score, AUC-ROC for each kernel.
- **ROC curves plotted** for the selected kernels.

3.2.7 Experiment 6: Comparison of All Classifiers

- **Comparison of Naive Bayes, KNN (K=10), Logistic Regression (optimal C value) and SVM (optimal kernel)**: Using all metrics.
- **Combined ROC curve plotted** for direct comparison.

4 Results

This section presents the results of the experiments, including the key metrics and ROC curve plots for each model.

4.1 Experiment 1: Influence of Dictionary Size on Naive Bayes Classifier

- **Best accuracy** at dictionary size 2000.
- **Highest AUC-ROC** at dictionary size 2000.

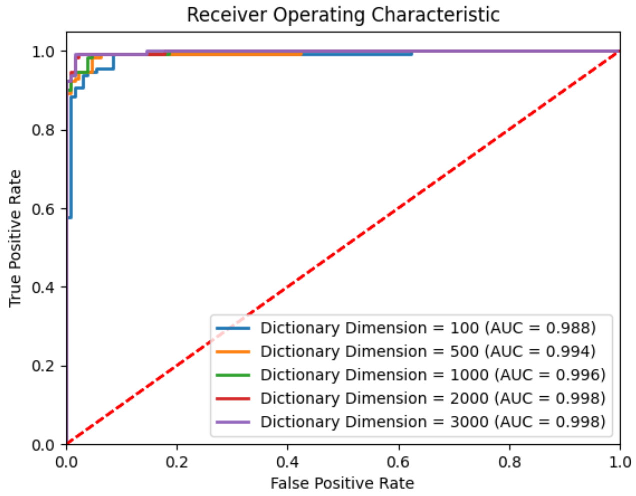


Figure 1: ROC Curves for Naive Bayes Classifier with Different Dictionary Sizes.

4.2 Experiment 2: Influence of K Parameter on KNN Classifier

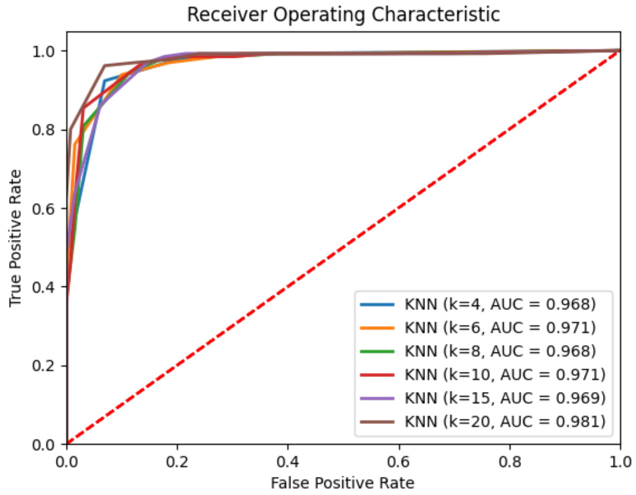


Figure 2: ROC Curves for KNN Classifier with Different K Values.

- **Optimal performance** at K=10, balancing precision and recall.

4.3 Experiment 3: Influence of Regularization Parameter (C) on Logistic Regression

- **Optimal C value** provided a good balance between model complexity and performance.

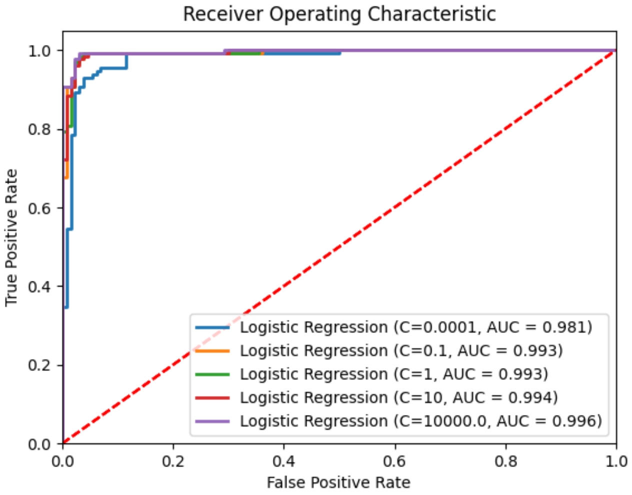
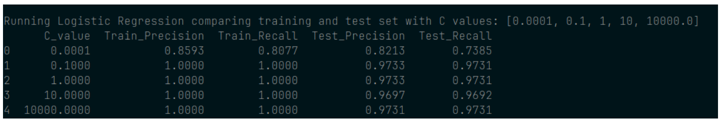


Figure 3: ROC Curves for Logistic Regression with Different Regularization (C) Values.

4.4 Experiment 4: Comparison of Training vs. Test Performance in Logistic Regression



```
Running Logistic Regression comparing training and test set with C values: [0.0001, 0.1, 1, 10, 10000.0]
      C_value  Train_Precision  Train_Recall  Test_Precision  Test_Recall
0      0.0001           0.8593        0.8077          0.8213       0.7385
1      0.1000           1.0000        1.0000          0.9733       0.9731
2      1.0000           1.0000        1.0000          0.9733       0.9731
3     10.0000           1.0000        1.0000          0.9697       0.9692
4  10000.0000           1.0000        1.0000          0.9731       0.9731
```

Figure 4: Comparison of Training vs. Test Performance in Logistic Regression.

- **Signs of overfitting** at high C values.

4.5 Experiment 5: Influence of Kernel on Support Vector Machine

- **Optimal kernel (Linear)** provided the best performance, balancing complexity and generalization.
- **Highest AUC-ROC** achieved with the Linear and rbf kernel.

4.6 Experiment 6: Comparison of All Classifiers

- **Naive Bayes outperformed** SVM, LR and KNN in most metrics.

5 Discussion

In this section, the results obtained from the experiments are analyzed and discussed in the context of model performance and generalization.

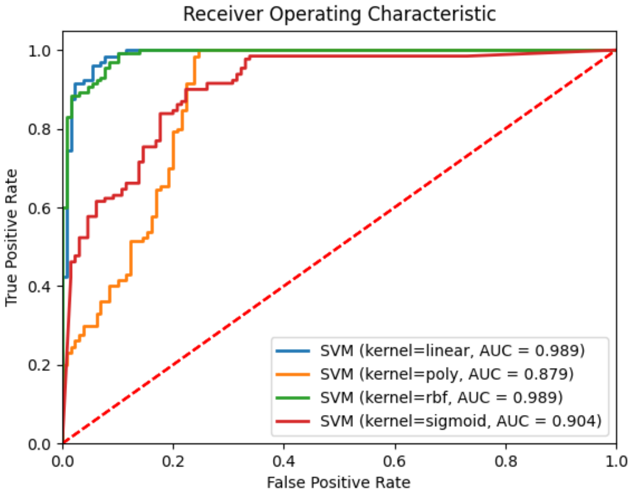


Figure 5: ROC Curves for SVM Classifier with Different Kernels.

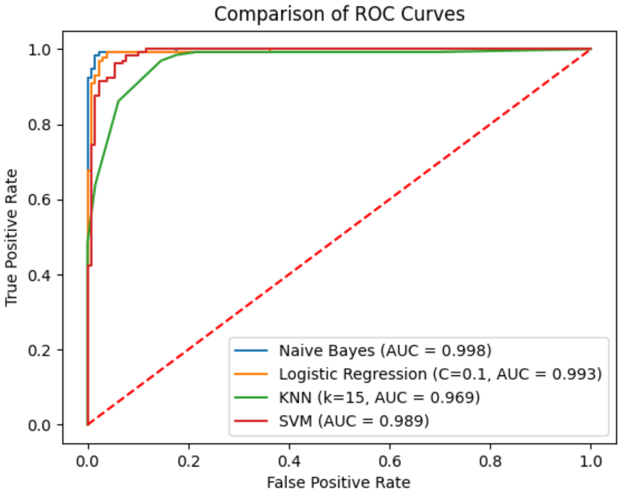


Figure 6: Combined ROC Curves.

5.1 Performance of Naive Bayes

Naive Bayes demonstrated consistent improvement in performance with increasing dictionary size. However, its assumption of feature independence, while simplifying computation, limited its ability to fully capture the complexity of spam messages. The best results were achieved at a dictionary size of 2000, beyond which the gains plateaued, suggesting that additional features contributed minimally to classification accuracy.

5.2 Performance of KNN

KNN's performance was highly dependent on the choice of K. Smaller K values led to overfitting, while larger values resulted in underfitting. The optimal K value (K=10) provided a good trade-off, ensuring that the model was neither too sensitive to noise nor too generalized. However, KNN's reliance on distance metrics and the curse of dimensionality limited its effectiveness compared to Logistic Regression.

5.3 Performance of Logistic Regression

Logistic Regression, with an optimal regularization parameter (C), emerged as the most effective classifier in this study. The regularization helped balance model complexity and prevented overfitting, resulting in superior performance across all metrics. The AUC-ROC scores consistently indicated that Logistic Regression had the best ability to distinguish between spam and non-spam emails.

5.4 Overfitting Analysis in Logistic Regression

The comparison between training and test metrics for Logistic Regression highlighted the risk of overfitting at low regularization (high C values). The gap between training and test performance became pronounced, confirming that the model was fitting the noise in the training data rather than learning generalizable patterns.

5.5 Performance of Support Vector Machine (SVM)

The SVM model demonstrated varying performance depending on the choice of kernel. Among the kernels tested, the 'linear' and 'rbf' (Radial Basis Function) kernels delivered the best results, both achieving an AUC-ROC score of 0.989. This suggests that these kernels were highly effective in separating the classes in the spam classification task.

- **Linear Kernel**: the linear kernel performed exceptionally well, likely due to the relatively high linear separability of the dataset. It achieved high precision, recall, and F1-scores across both classes.
- **RBF Kernel**: the RBF kernel also provided strong performance, closely matching the linear kernel in terms of AUC-ROC. It is particularly useful in capturing non-linear relationships within the data, which may explain its high performance despite the higher computational cost compared to the linear kernel.
- **Polynomial Kernel**: the polynomial kernel, on the other hand, significantly underperformed. It achieved an AUC-ROC of 0.879, with poor recall for class 1 (spam), indicating that it struggled to generalize well from the training data. This suggests that the dataset's features do not align well with the assumptions made by the polynomial transformation.
- **Sigmoid Kernel**: the sigmoid kernel showed moderate performance with an AUC-ROC of 0.904. While it performed better than the polynomial kernel, it was less effective than the linear and RBF kernels. This performance suggests that while

the sigmoid kernel can capture non-linearities, it might not be as well-suited for this particular dataset as the RBF kernel

5.6 Overall Comparison

The comparative analysis of the models, based on their best parameters, reveals that Naive Bayes outperformed the other algorithms in this study, achieving an AUC-ROC of 0.9977. This result underscores the model's effectiveness in distinguishing between spam and non-spam emails, even with its relatively simple approach. Although Naive Bayes had a slightly lower recall of 0.9308 compared to other models, its ability to produce a nearly perfect AUC-ROC highlights its robustness for this task.

Logistic Regression also demonstrated strong performance, with an AUC-ROC of 0.9934 and a recall of 0.9846. The optimal setting for the regularization parameter (C=0.1) allowed the model to balance simplicity and accuracy effectively, making it a reliable choice for spam classification, particularly when interpretability and consistency are critical.

The Support Vector Machine (SVM), using a linear kernel, achieved an AUC-ROC of 0.9889 and a recall of 0.9615. While SVM slightly trailed behind Naive Bayes and Logistic Regression in AUC-ROC, it still provided robust performance, especially in handling datasets where linear separability is a key factor. The linear kernel proved to be the best option for this dataset, reinforcing SVM's versatility in scenarios involving non-linear relationships.

K-Nearest Neighbors (KNN), configured with K=15, showed competent performance with an AUC-ROC of 0.9686 and a recall matching that of Logistic Regression at 0.9846. However, KNN was less effective compared to Naive Bayes, Logistic Regression, and SVM, particularly in handling the complexities of higher-dimensional data and more intricate decision boundaries.

In summary, Naive Bayes emerged as the most effective model for this specific spam classification task, particularly when considering the AUC-ROC metric. Nevertheless, both Logistic Regression and SVM (with a linear kernel) were strong contenders, offering high recall and competitive AUC-ROC scores, making them viable alternatives depending on the specific requirements of the classification task. KNN, while demonstrating good recall, was slightly less effective in this context, particularly in terms of AUC-ROC, suggesting it may not be as well-suited to tasks with more complex data structures. This analysis highlights the importance of selecting and tuning models based on the specific characteristics of the dataset and the goals of the classification task.

6 Conclusion

This experiment explored the effectiveness of four supervised learning algorithms—Naive Bayes, K-Nearest Neighbors (KNN), Logistic Regression, and Support Vector Machine (SVM)—in the task of spam classification. The experiments conducted provided valuable insights into the strengths and limitations of each approach, highlighting the importance of model selection and tuning in achieving optimal performance.

In conclusion, this analysis highlights that the best-performing model for spam classification can vary depending on the evaluation metric and the specific requirements of the task. Naive Bayes, with its simplicity, achieved the highest AUC-ROC in this study, demonstrating its strong ability to distinguish between spam and non-spam emails. This suggests that, for tasks where discriminative power is the primary concern, Naive Bayes can be an excellent choice. However, when considering other metrics such as precision, recall, and F1-score, Logistic Regression and SVM also proved to be highly effective. Logistic Regression, with its balance of complexity and interpretability, provided consistent results across all metrics, making it a reliable and versatile model, particularly when model simplicity and explainability are crucial. SVM, especially with the linear kernel, also performed strongly, making it a viable alternative in scenarios where the dataset might involve more complex decision boundaries or when a different type of regularization is required.

These findings underscore the importance of not relying solely on a single metric, such as AUC-ROC, to determine the best model. Instead, a comprehensive evaluation using multiple metrics is essential to fully understand each model's strengths and weaknesses. The choice of the best model should be guided by the specific context of the task, including the nature of the data, the importance of different types of errors (false positives vs. false negatives), and the need for model interpretability. By carefully considering these factors, practitioners can select the most appropriate model that aligns with their specific objectives.