Development of a Spam Filter

May 26, 2023

Contents

List of Figures

1 Introduction

The rise of short message communication in recent years has provided individuals and businesses with a fast and interactive way to communicate with their peers and customers. However, this has also led to an increase in the number of spam messages, particularly on publicly available channels. In response to this, our company has tasked me with building a spam filter for their open channel to enable effective communication with their customers while maintaining message quality.

As the data scientist for this project, I have been provided with a dataset compiled from different open-source collections of short messages. The task is to build a classification model that can identify if a message is spam and allows for an adjustable risk level, ensuring that low-risk messages are immediately displayed while others are moved to a folder for further analysis. There is also the need to provide a strategy for future adjustments to the solution to improve the spam classification and to not lose potential customers.

To achieve these objectives, this project will follow the CRISP-DM framework, a widely used methodology that provides a structured approach to planning a data science project. The report will cover all the necessary steps, including business understanding, data understanding, data preparation, modeling, evaluation, and a suggestion for deployment. (Shearer, 2000)

In this report, I will begin by providing an overview of the project and it's objectives. I will then detail the proposed project structure which will be stored on GitHub and explain how this repository will be organized. Then I will assess the quality of the dataset and visualize the findings for better understanding. Following I will create, train, and test a classification model using the dataset, and perform an error analysis to highlight any weaknesses in the chosen approach.

Finally, I will propose how the model can be integrated into the daily work of the service team, including suggestions on how a graphical user interface (GUI) could be implemented. By the end of this report, I aim to deliver a spam filter that meets the business requirements and provides the service team with a tool to maintain high message quality while ensuring effective communication with customers.

2 Organizing the Project with CRISP-DM

The first step in the CRISP-DM methodology is the Business Understanding step, which involves understanding the business objectives and requirements of the given task. In this step, we need to define the goals and objectives of the project, and identify the stakeholders. This will help us to define the scope of the project and ensure that we deliver a solution that meets the needs of the stakeholders. (Schröer, Kruse, & Gómez, 2021)

2.1 Business Understanding

In this project, the business objective is to build a spam filter for a new open communication channel that the company wants to install for its products. The aim is to enable fast and interactive feedback for their customers or possible future customers. The spam filter should be able to identify spam messages from short messages, and allow for adjustments with respect to the risk of allowing spam to pass. This means that the spam filter should be adjustable via different spam-risk levels, e.g., ”low-risk” (very restrictive) and ”high-risk” (not restrictive). The messages passing the ”low-risk”-level are immediately displayed, while the other ones are moved to a folder for further analysis.

The stakeholders in this project are the company, the service team who will be responsible for using the spam filter to filter out spam messages, and the customers, who will be using the open channel to provide feedback or ask questions on the company's products. It is important to keep in mind the expectations of these stakeholders while developing the spam filter, so that we can deliver a solution that is effective and meets the needs all groups. For the company the emphasis lies on not losing customers from misclassified messages, the service team want a system that simplifies their daily work and the customers do not want to read to much spam messages, although a lost customers will probably hurt the company more than a spam message entering the channel.

In addition to the primary objectives of the project, it is also important to consider any constraints and assumptions that may impact the project. For example, we could have limitations in terms of the resources available to us, such as the amount of data that we have access to or the computing power required to train and test our models. We may also have to make assumptions about the types of messages that are likely to be classified as spam, and adjust our approach accordingly. By considering these constraints and assumptions, we can ensure that we deliver a solution that is realistic and feasible given the resources available to us.

2.2 Constraints and Assumptions

Based on the dataset and the resources available, we can identify the following constraints:

- The dataset only includes English messages, so the model will only be able to classify English messages accurately.
- The dataset has a limited number of messages, which may not be representative of all types of messages that the company may receive.
- The analysis and model training will be done on a home desktop PC, which may have limitations in terms of computational power and memory.

and assumptions:

- The messages in the dataset are correctly labeled as ham or spam.

- The dataset is a representative sample of the types of messages that the company may receive.
- The spam filter will be used in a similar context as the dataset, so the model will generalize well to new messages.

2.2.1 GitHub Repository Structure

For organizing the GitHub repository, we will use the following folder structure:

- **data**: This folder will contain the raw and processed data files.
- **notebooks**: This folder will contain Jupyter notebooks used for data analysis, processing, modelling, and evaluation.
- **models**: This folder will contain trained models.
- **reports**: This folder will contain the project report.

The repository can be accessed with the following link:
https://github.com/marco507/SpamClassfication

2.3 Data Understanding

To start the data understanding step, we will first load the dataset and check for any missing values and duplicates. We will use the Pandas library to load the CSV file containing the messages and perform basic exploratory data analysis (EDA).

```
## Output ##
Number of missing values:
label     0
message   0
Number of duplicates: 403
Unique records: 5169
```

From the above output, we can see that there are no missing values in the dataset, but there are 403 duplicate records. We will remove these duplicate records from the dataset and look at the number of resulting unique records.

```
## Output ##
Unique records: 5169
```

Next, we will explore the distribution of the target variable 'label' with the help of Matplotlib and Seaborn. Additionally we look at the distribution of the character count of the messages.

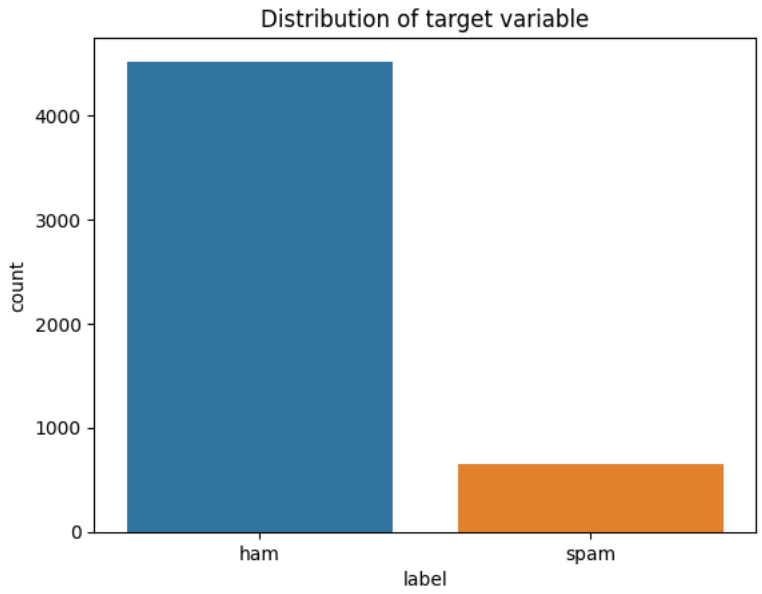

Figure 1: Distribution of the target variable

From the first visualization, we can see that the dataset is imbalanced as there are more 'ham' messages than 'spam' messages. This must be accounted for before we train our classifier.

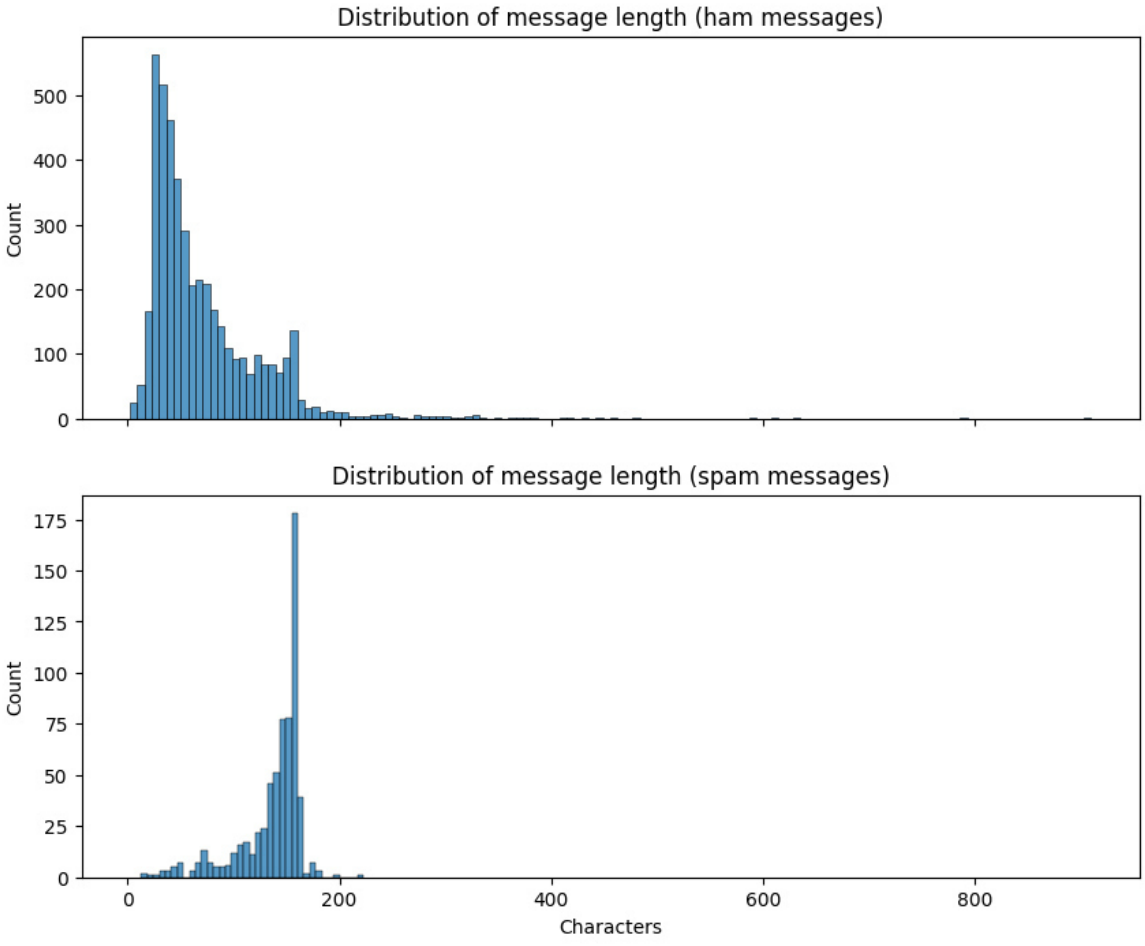

Figure 2: Distribution of the message lengths

Plotting the distribution of the message lengths, we can observe that spam messages tend to have longer message length compared to ham messages. Further looking into the number of digits and special characters reveals that spam messages tend to have more digits.

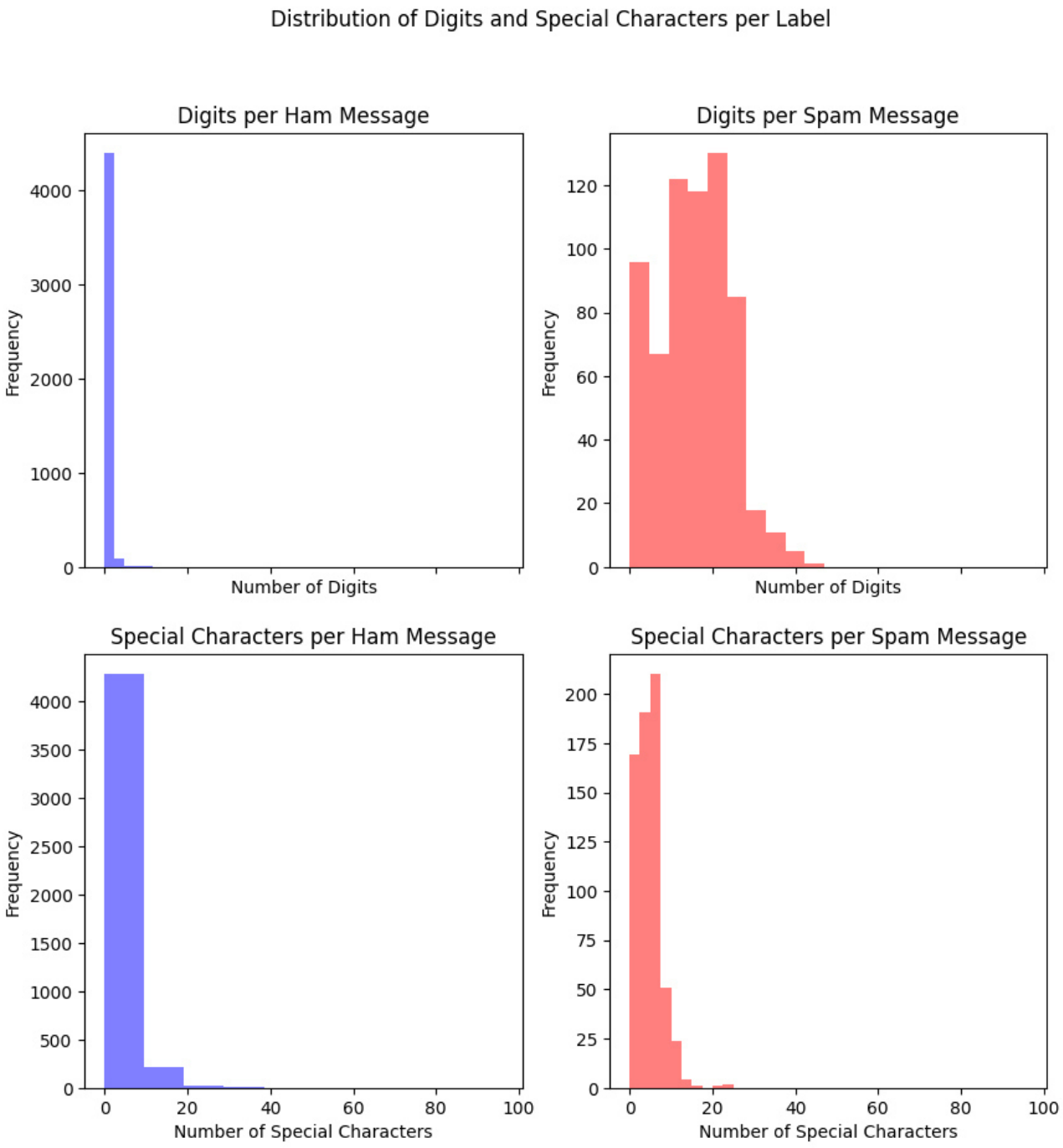

Figure 3: Distribution of the number of digits and special characters

Another interesting feature that we will explore is the number of unique words per message type and lastly we plot wordclouds to spot possible differences in vocabulary.

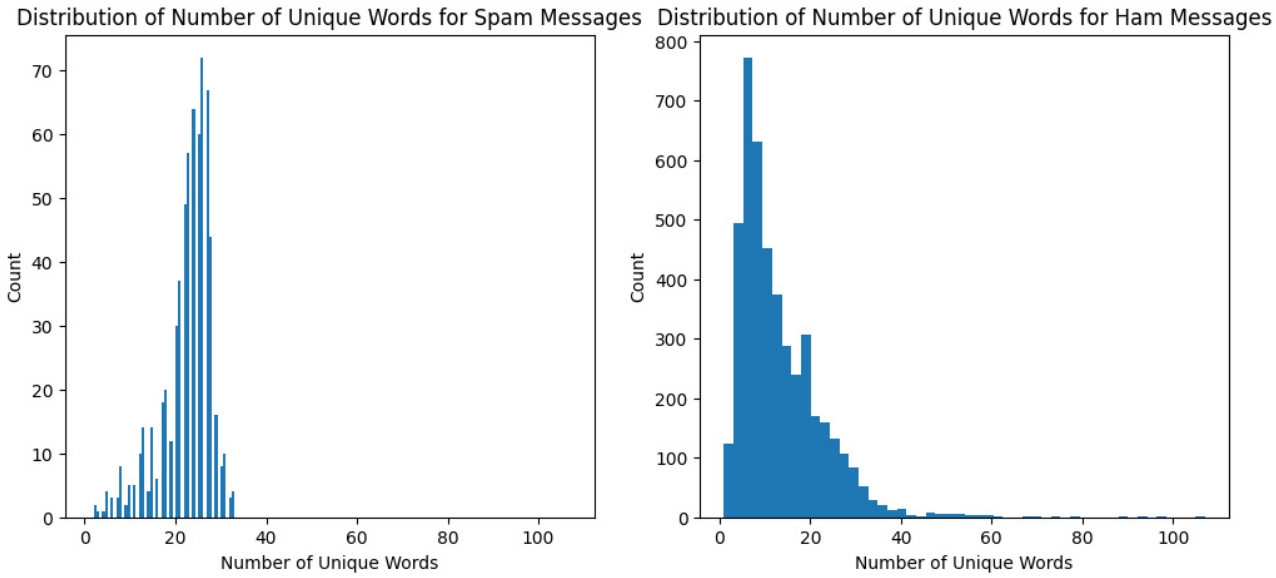

Figure 4: Distribution of the number of unique words

Figure 5: Wordclouds for ham and spam messages

With this we conclude the EDA and move further on to the data preparation step where we will balance the dataset and choose the most interesting features for training the model. The engineered features for each message will be message lengths, the number of unique words and a TF-IDF representation of the vocabulary. We will omit the special character distribution as ham and spam messages are very similar in this regard.

2.4 Data Preparation

For the next step we perform data preprocessing and feature engineering on the SMS message dataset. We load the data from the source CSV file, remove duplicates, and create new columns such as message length, number of digits, number of unique words, and a lemmatized version of the messages. The text data is then transformed into a matrix of TF-IDF values using the TfidfVectorizer class of the scikit-learn library. The resulting features are saved along with the trained vectorizer. Additionally the dataset is balanced by sampling an equal number of ham and spam messages. The

target variable is encoded as 0 for ham and 1 for spam and finally, the prepared dataset is saved as a new CSV file.

2.5 Model Training

For training, we first load our prepared dataset of messages and split it into training and testing sets. Then we perform a grid search with cross-validation to find the best hyperparameters for our classifier which will be a decision tree. Additionally, the feature importances are retrieved from the trained classifier, and a barplot is created to visualize the top 10 most important features.

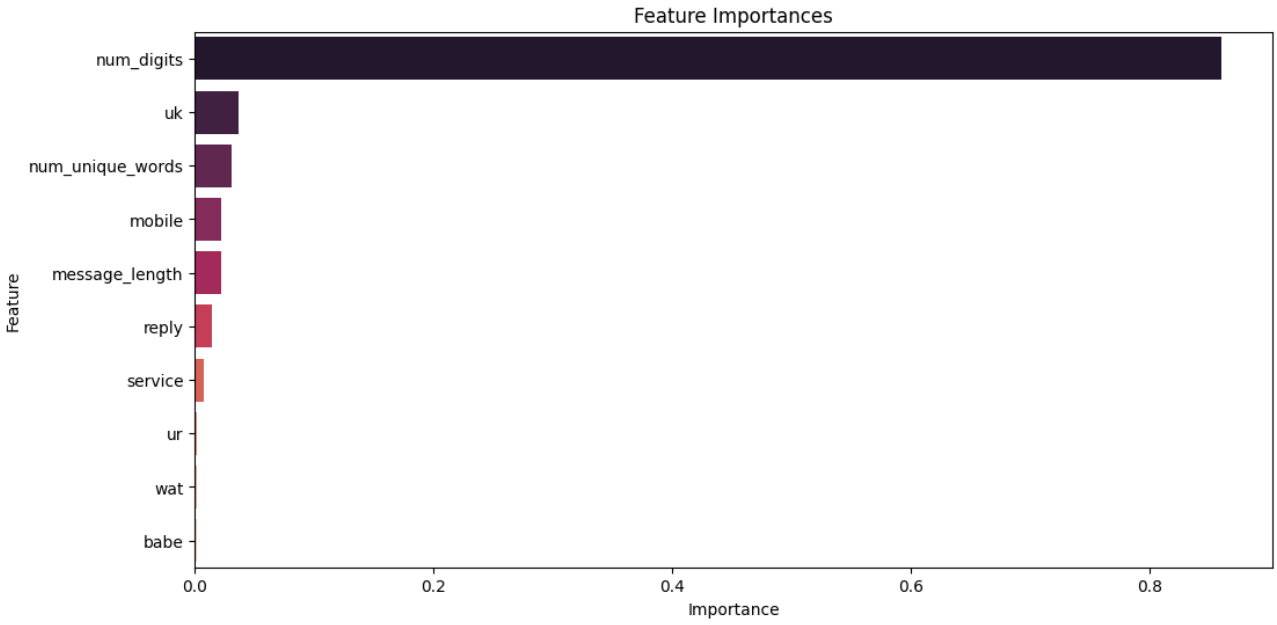

Figure 6: Feature importances

Examining the plot, we can see that the model clearly favours the number of digits in a message as decision factor. Following we have specific words and our other engineered features.

2.6 Error Analysis

After training our model, we will now further analyse it's performance. To do this we compute a confusion matrix on our training and test datasets. Additionally we also retrieve accuracy, precision and recall for each dataset respectively.

```
## Output ##
Train Accuracy: 97.4
Train Precision: 99.2
Train Recall: 95.6
Test Accuracy: 94.7
Test Precision: 95.3
Test Recall: 93.8
```

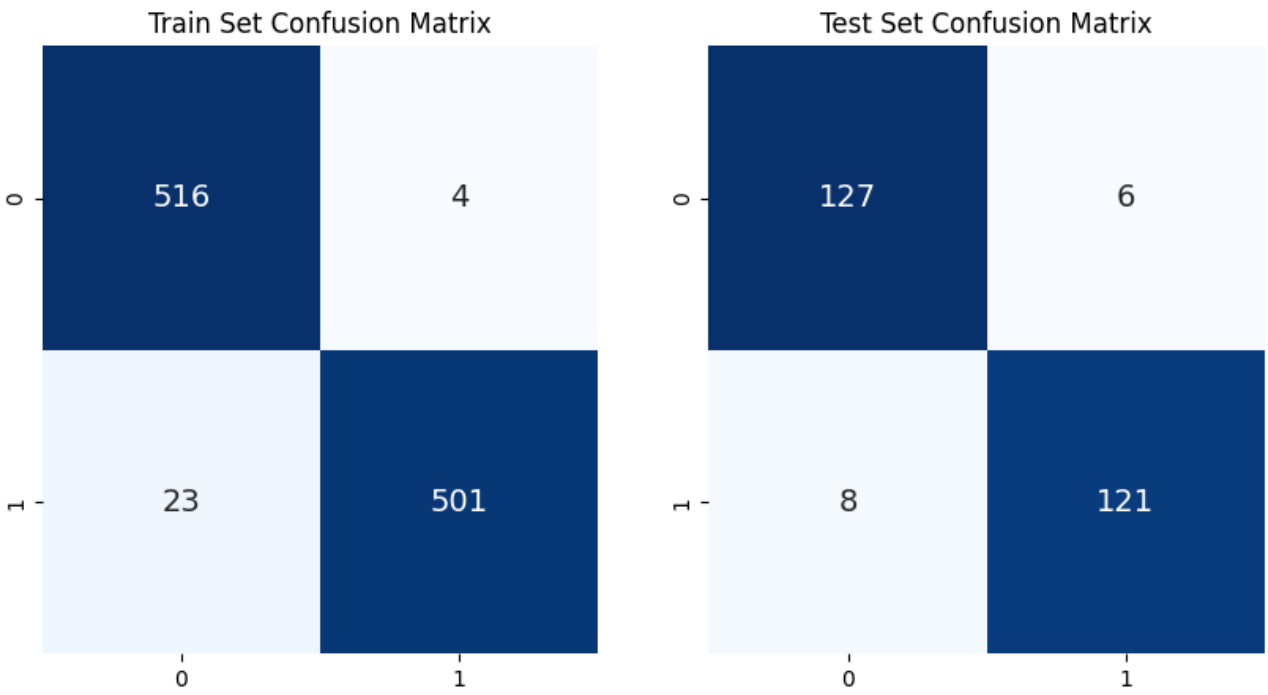

Figure 7: Confusion matrices

From the metrics we can conclude that the model performs sufficiently well on our initial dataset. Additionally we will look into the distribution of misclassified messages.

```
## Output ##
6 ham messages classified as spam
8 spam messages classified as ham
```

With this we can assume that the model is not particularly biased towards one of the message types. Finally we will test the model on 10 new messages that were generated with a language model and consist of 5 ham and 5 spam messages. The test results can be examined in the analysis Jupyter notebook and revealed a reasonable out-of-sample performance and generalisation. Although the emphasis of the model on digit counts makes spam messages with no digits and ham messages with more digits hard to classify correctly.

3 Deployment

To integrate the spam classification model into the service team's workflow, we could develop a web application using Django. The steps involved in creating and deploying this system would be as following:

1. Set up the Django project structure and configure the necessary settings.
2. Design the database schema to store messages and their classifications.
3. Set up an API endpoint that receives incoming messages, assuming we have the possibility to

control message flow from our communication channels backend.

4. Implement the web interface using Django's template system and HTML/CSS.
5. Integrate the spam classification model into the project and develop the function to classify messages.
6. Implement an adjustable spam level control, allowing the service team to set the level of stringency for classifying messages as spam. The level would be a percentage threshold that can be used for comparison with the probabilities of class membership that the model returns in percentages.
7. Deploy the Django application on a cloud platform.
8. Conduct thorough testing and quality assurance to ensure functionality and performance.
9. Provide documentation and training to guide the service team on using the web application effectively.
10. Maintain and improve the system based on feedback and ongoing monitoring.

4 Conclusion

The objective of this project was to build a classification model capable of identifying spam in short messages. The CRISP-DM methodology was followed, covering business understanding, data understanding, data preparation, modelling, evaluation, and a suggestion for deployment. A project structure was proposed which was then used to organize a GitHub repository containing the projects code, data and report. The dataset's quality was assessed, and findings were visualized for better comprehension. A decision tree classifier was created, trained, and tested using the provided dataset and additional out-of-sample data. An error analysis was conducted to identify weaknesses in the approach, which would be the reliance of digits for classifying spam. Finally, a proposal was made to integrate the model into the daily work of the service team. The solution would be in the form of a graphical user interface (GUI) based on a Django web application.

In summary, this project successfully developed a classification model for spam identification in short messages. The model returns class membership certainties in percentages which can be further utilized to build logic for adjustable risk levels. Further improvements to the model could be made by gathering additional data or data that is more representative of the use case, i.e. short messages of product feedback or questions. Additionally there is the possibility of engineering alternative features to reduce reliance on the digit count. The project demonstrates the efficacy of data science methodologies in addressing real-world challenges, offering a valuable solution for the company.

References

Schröer, C., Kruse, F., & Gómez, J. M. (2021). A systematic literature review on applying crisp-dm process model. *Procedia Computer Science*, *181*, 526–534.

Shearer, C. (2000). The crisp-dm model: the new blueprint for data mining. *Journal of data warehousing*, *5*(4), 13–22.