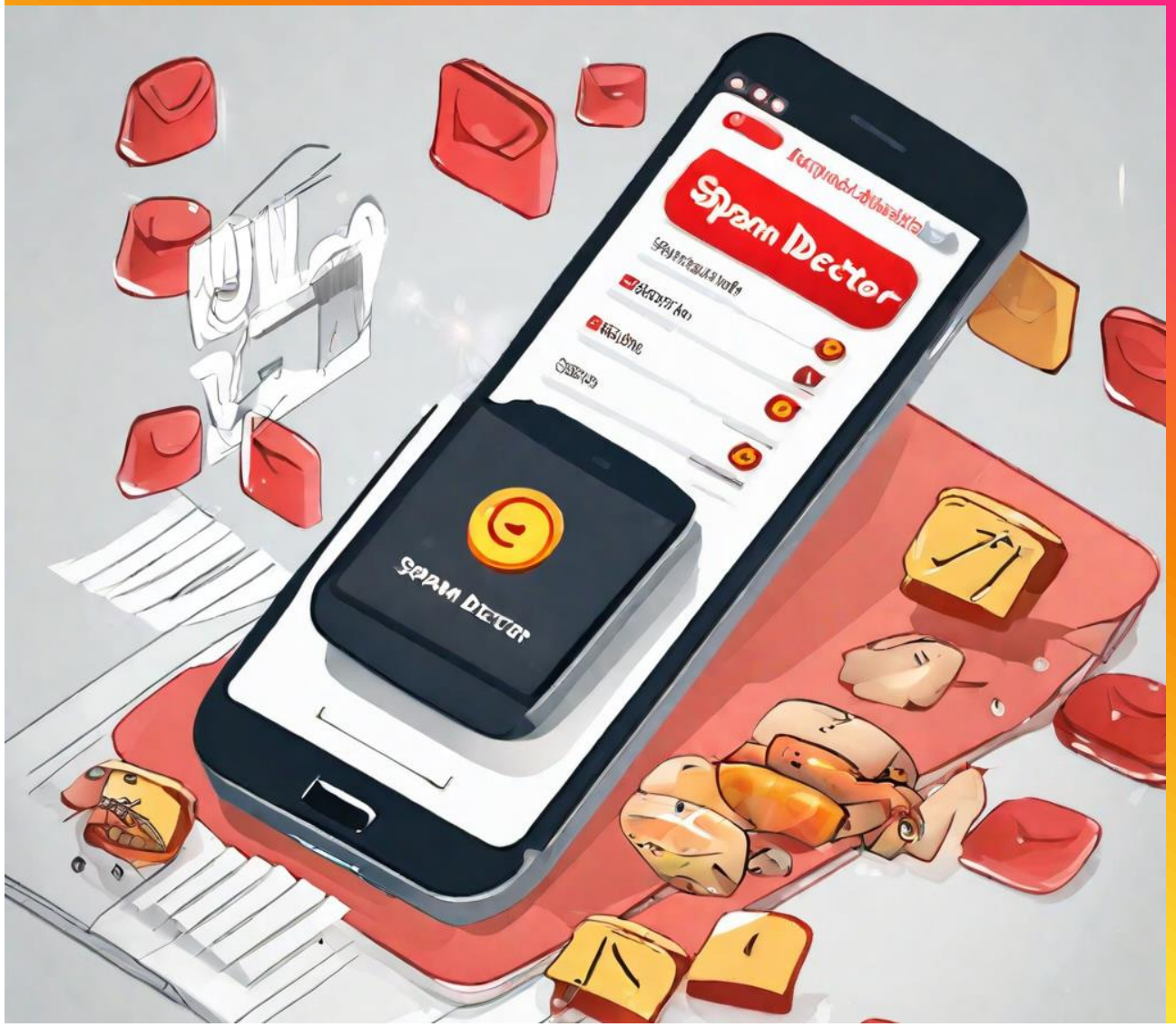
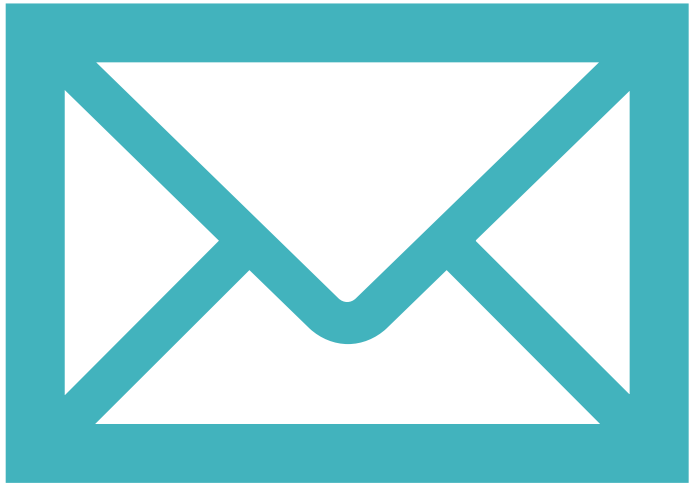
Spam Message Detector

Kshitij Mahajan



Problem Statement



- Spam messages frequently carry malicious links or phishing attempts posing significant threats to both organizations and their users. My system effectively detects and filters out spam messages, adding an extra layer of security that safeguards organizations against potential financial losses, legal consequences, and reputational harm.

Key Sections

- Explored Approaches
- Dataset
- Data Pre-processing and EDA
- Traditional Machine Learning Algorithms
- Deep Learning
- Large Language Models
- Conclusion and Findings

Explored Approaches

When tackling the spam detection challenge, I followed a path from simpler approaches to more advanced techniques.

This approach allows for a gradual understanding of the problem and helps us build a strong foundation before venturing into complex models.

While time constraints can influence our exploration, it is essential to adopt an efficient and effective strategy.

Explored Approaches

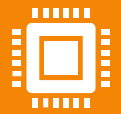
1.Simpler Approaches:

Implemented Logistic Regression, Random Forest, XGBoost, and Stacked Classifier algorithms.

These models provide a solid baseline for comparison and establish a benchmark for performance evaluation.

2. Intermediate Approach:

Implemented an LSTM (Long Short-Term Memory) model, a type of Recurrent Neural Network (RNN).



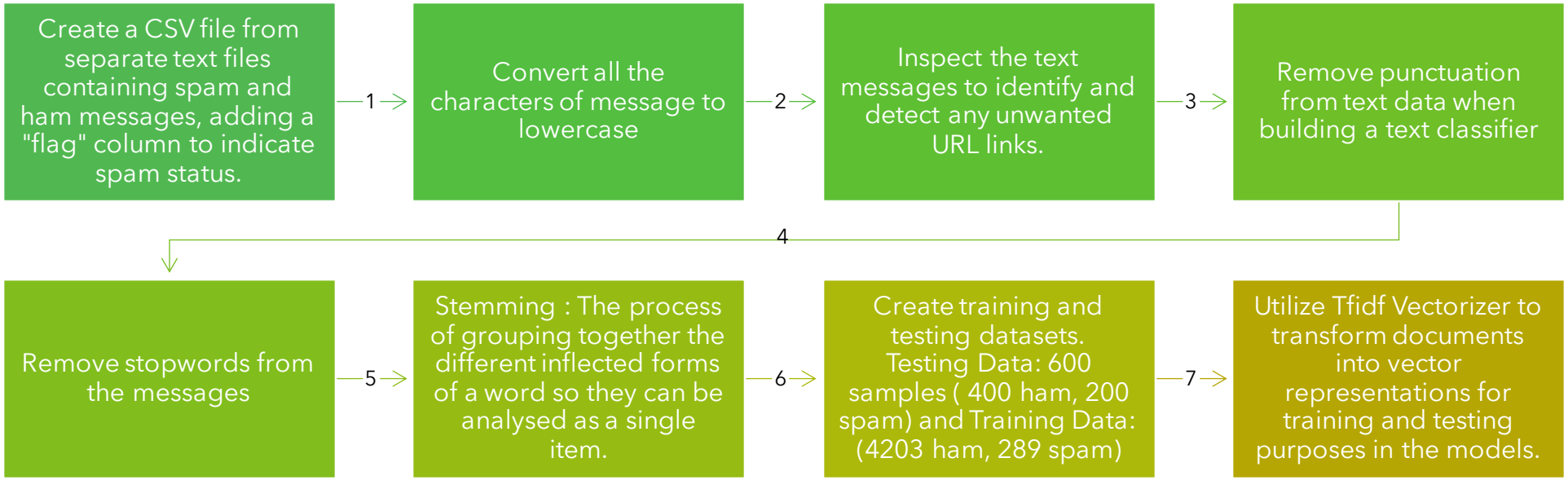
3. Advanced Approach:

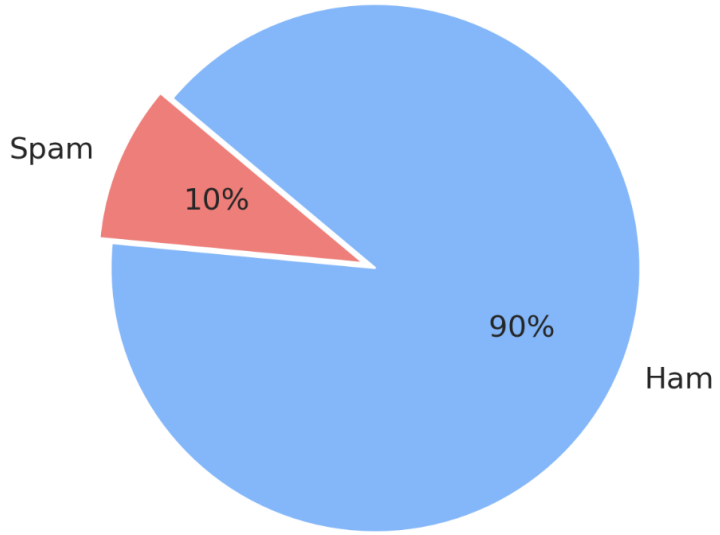
Leveraged the power of large language models (LLMs) for spam detection.

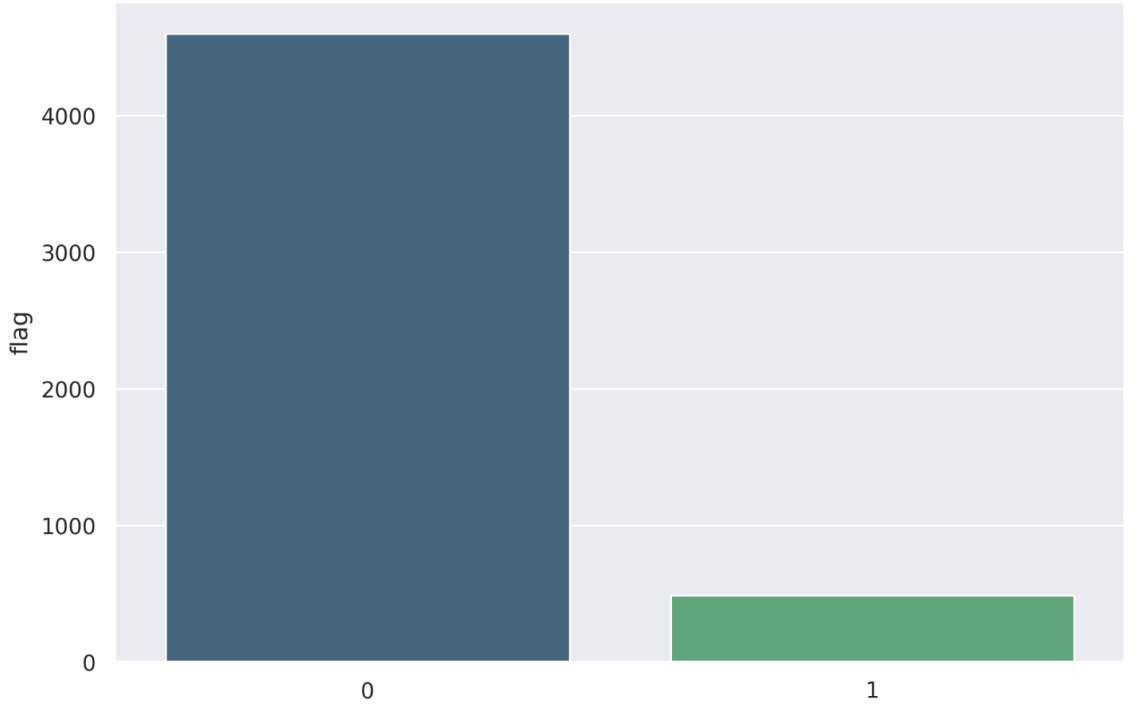
Dataset

- I was provided with two datasets: human.txt and spam.txt .
- The human.txt file contains SMS text messages submitted by humans (4603 messages) .
- The spam.txt includes messages flagged as spam (489 messages) .

Data Pre-processing

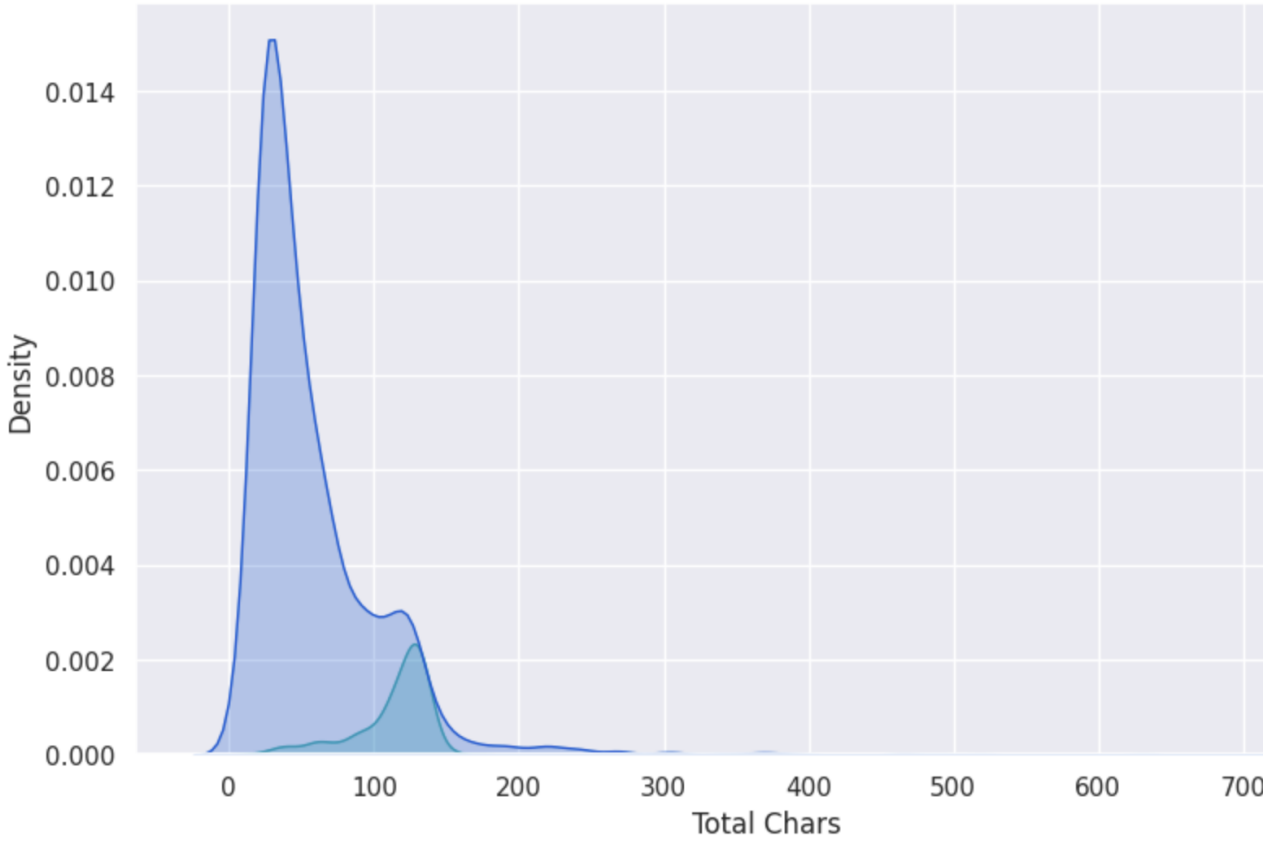


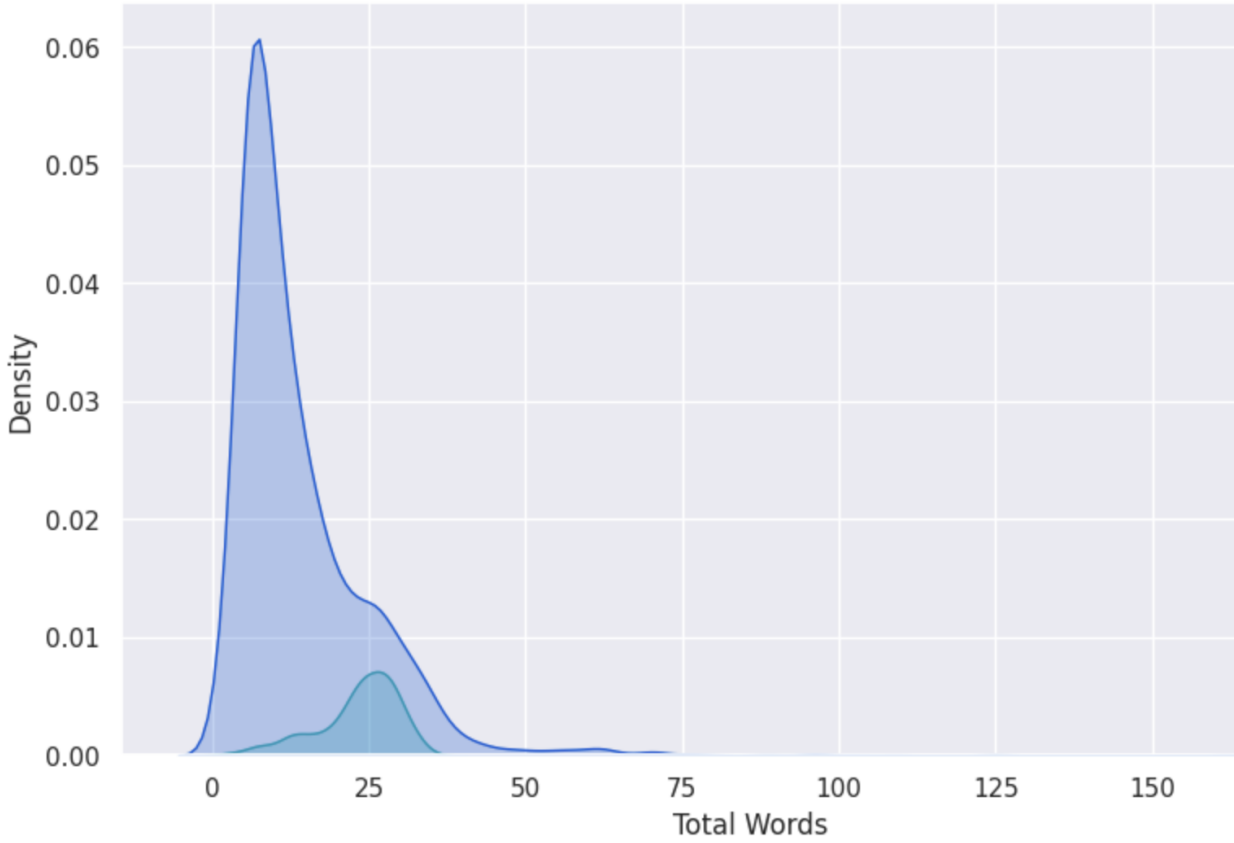




Exploratory Data Analysis

Only 10% of messages are spam. Hence, our dataset is imbalanced.

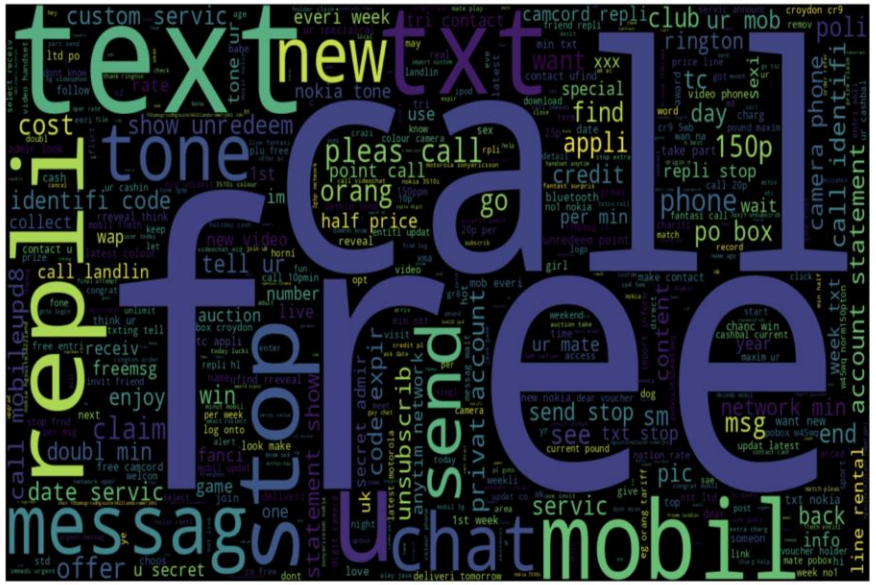




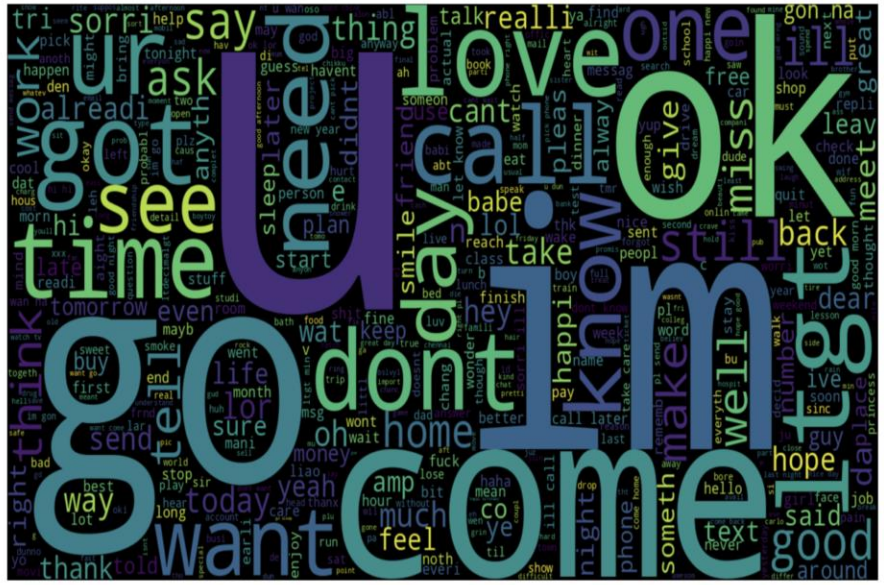
Exploratory Data Analysis

Clearly, one thing we can conclude here is that more the number of words in a text, there are more chances of it being spam

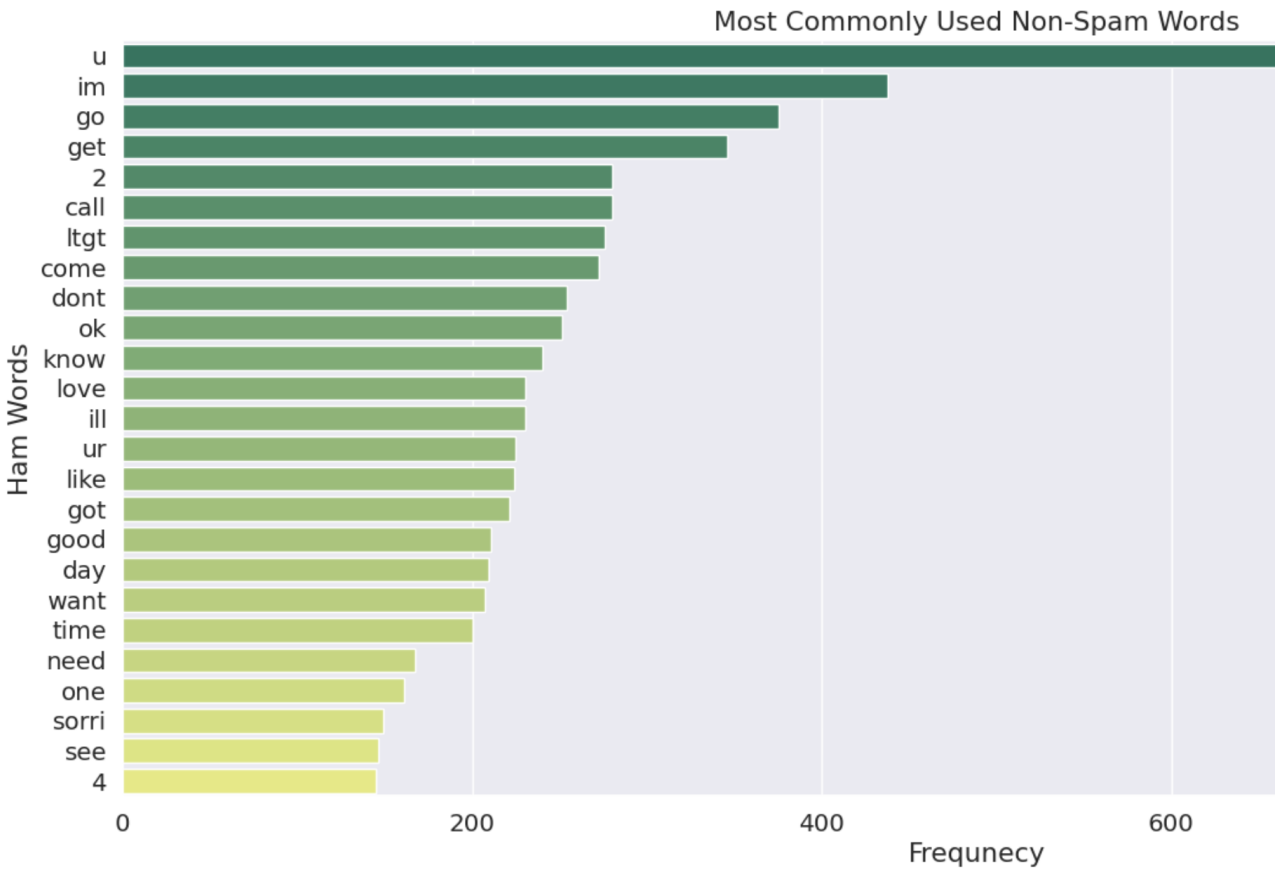
Exploratory Data Analysis

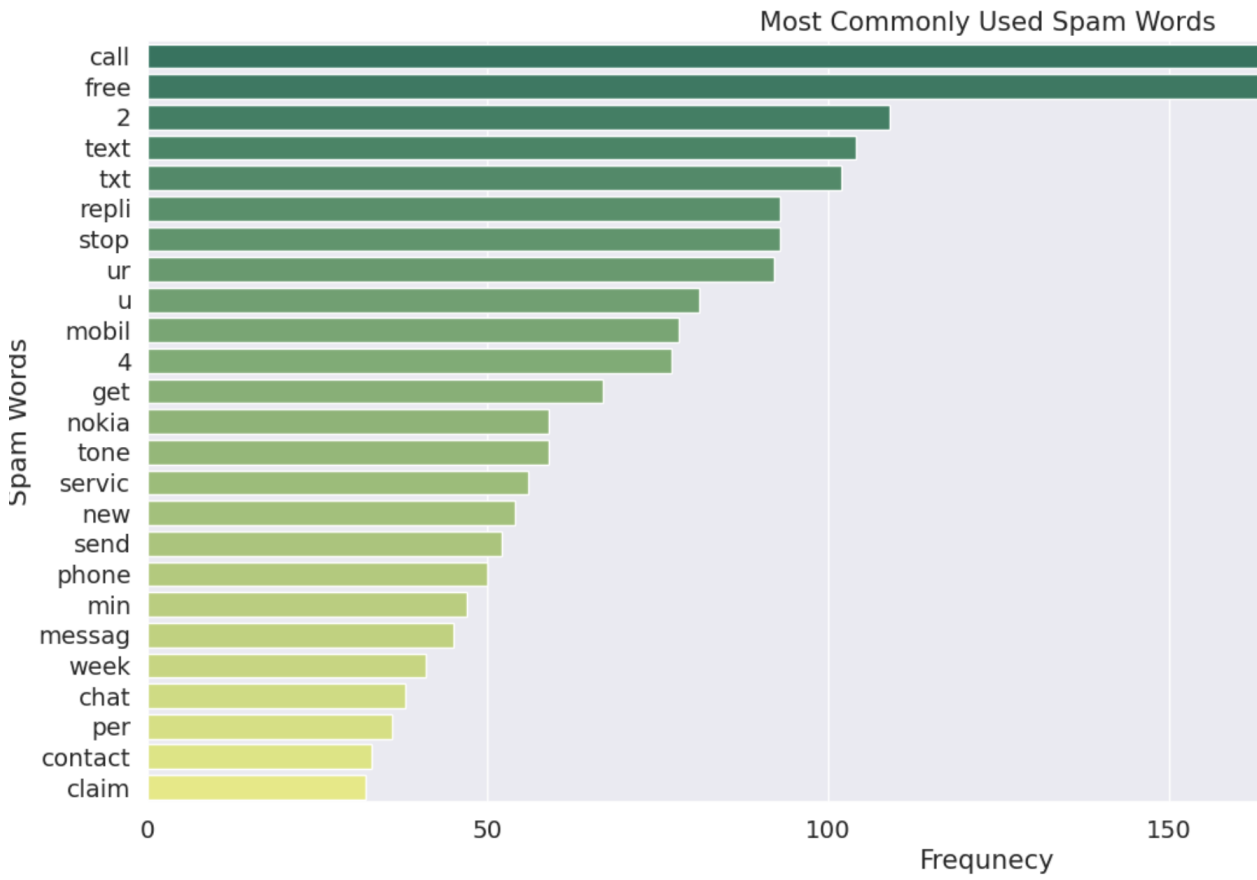


We can see that spam messages contain words such as free, call, and repli.



Word cloud of ham messages contain normal words that we use in day-to-day life conversation.



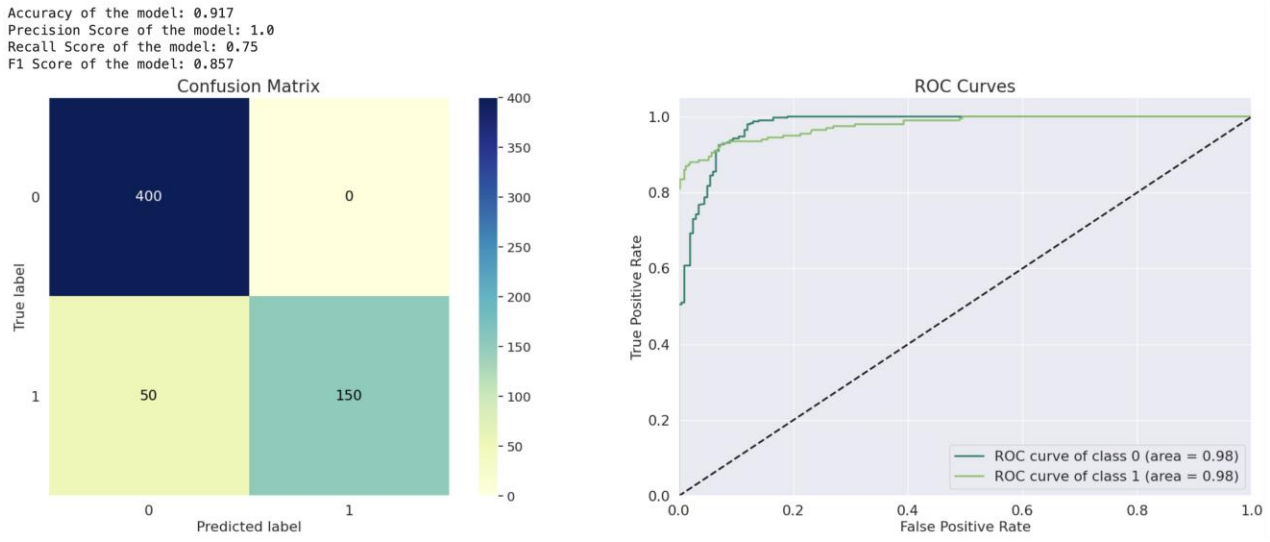


Exploratory Data Analysis

It can be seen that words common in spam messages are not commonly observed in ham messages. The percentage of occurrence of spam words are very high too in spam messages.

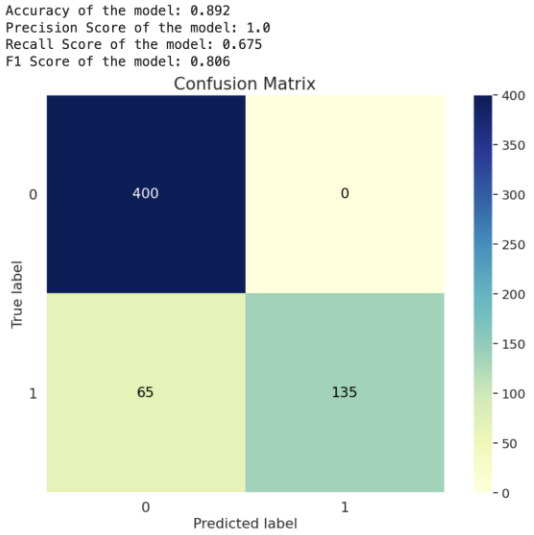
Logistic Regression Model

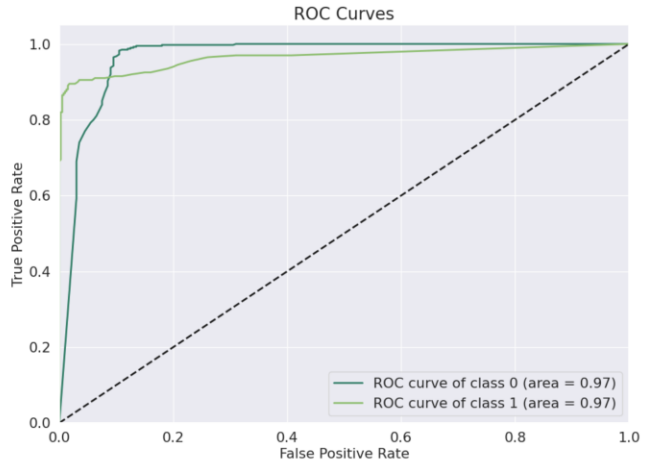
- Used GridSearchCV for finding the ideal set of hyper parameters for our training job
- The logistic regression model achieved perfect classification for ham messages, with no errors. However, it correctly identified 150 spam messages while incorrectly classifying 50 spam messages as ham.



Random Forest Classifier

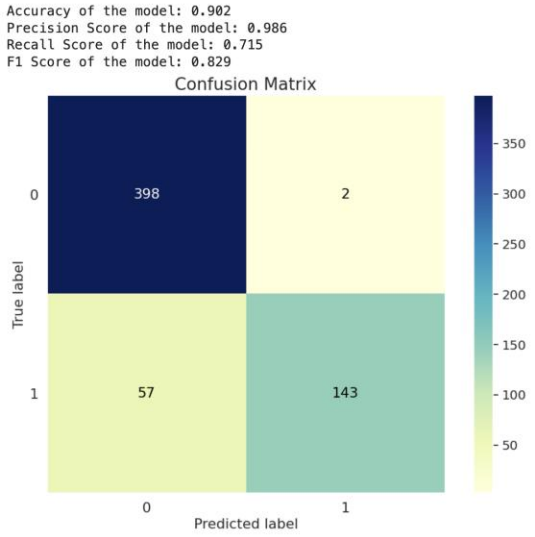
- Used GridSearchCV for finding the ideal set of hyper parameters for our training job.
- The random forest classifier model achieved error-free classification for ham messages. However, it correctly identified 135 spam messages while incorrectly classifying 65 spam messages as ham.

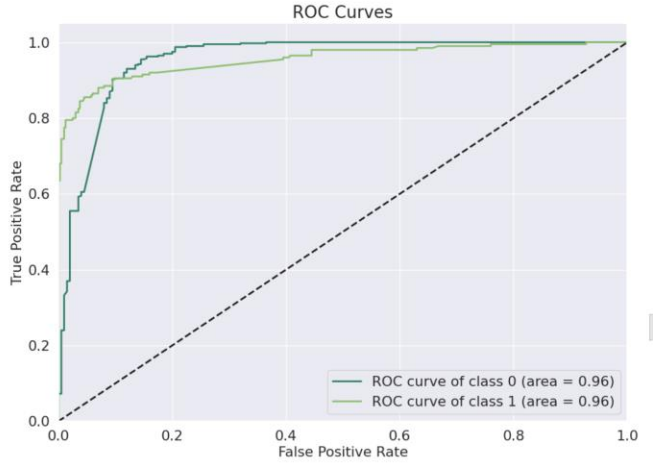




XGBoost Classifier

- Used GridSearchCV for finding the ideal set of hyper parameters for our training job
- Logistic Regression performs slightly better than Random Forest and XGBoost, potentially due to its robustness to feature scaling and better handling of the minority class.
- Logistic Regression's linear decision boundaries are advantageous for data with separable classes and a linear relationship between features and the target.
- To further improve performance, a stacked classifier combining all three models can be considered.

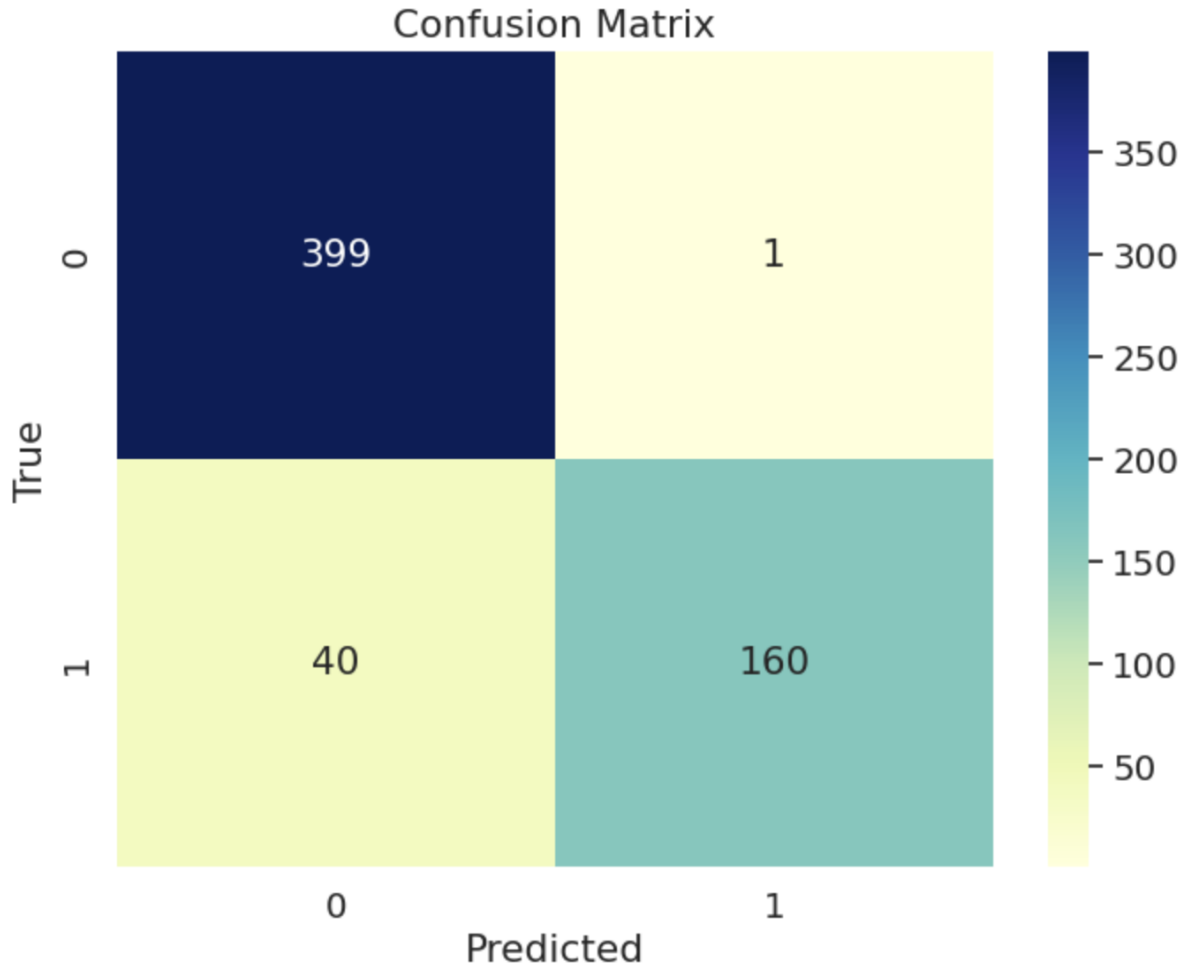




Stacked Classifier

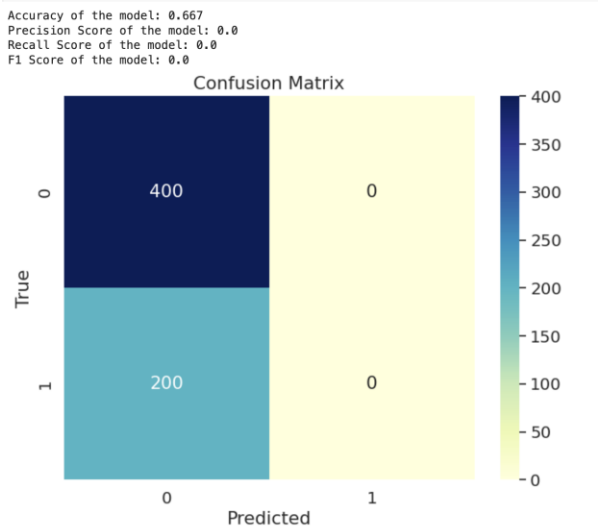
- Used Logistic Regression, Random Forest, and XGBoost as base models. Logistic Regression was selected as the final estimator.
- As expected Stacked classifier performs better than all individual models.

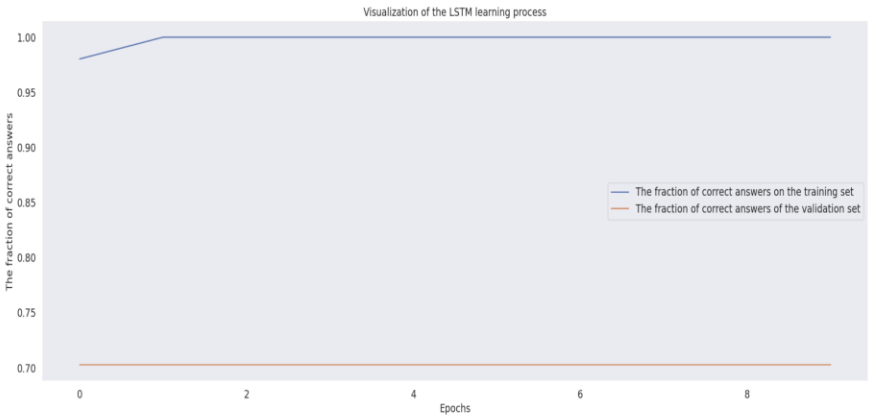
```
Accuracy of the model: 0.932
Precision Score of the model: 0.994
Recall Score of the model: 0.8
F1 Score of the model: 0.886
```



Deep Learning : LSTM (RNN)

+ LSTM is underperforming and consistently classifying the majority class. Imbalanced data handling techniques such as class weighting, oversampling, or specialized loss functions can be employed to address this issue and improve LSTM's focus on the minority class.



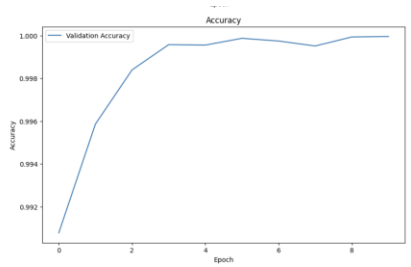


Large Language Models

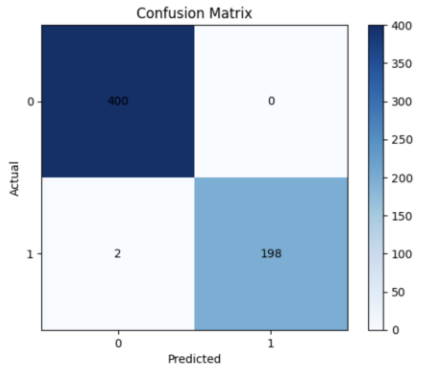
+ I explored two different LLMs:
  1. RoBERTa : BERT like arechitecture
  2. MistralAI : Text Generator LLM (7B parameters)
+ The approaches I tried:
  1. Finetuning
  2. Few shot learning + RAG

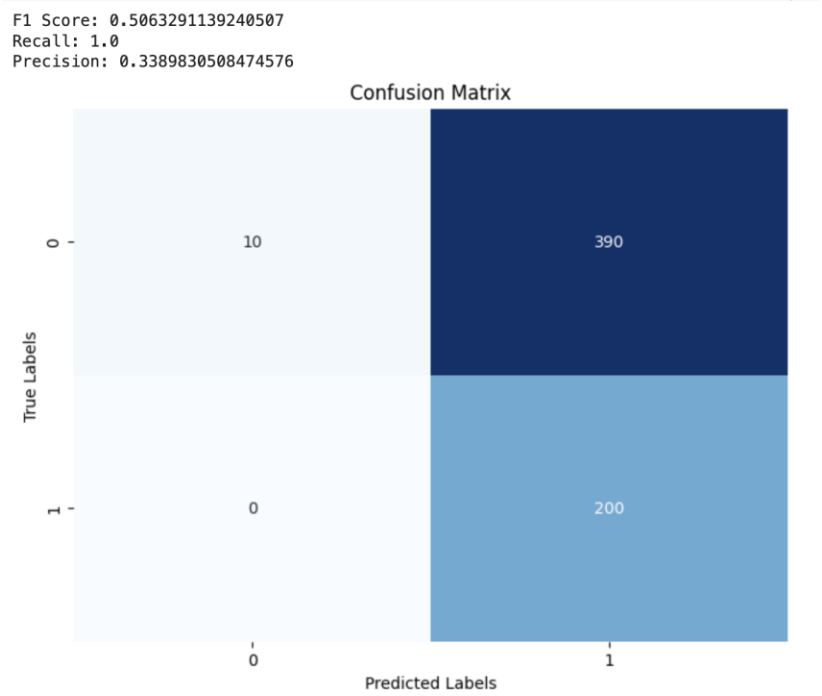
RoBERTa : Fine-tuned

- Finetuned RoBERTa is performing very well on the test dataset. RoBERTa LLM outperformed all traditional machine learning models.
- I have hosted the model at huggingface : [Fine-tuned RoBERTa]



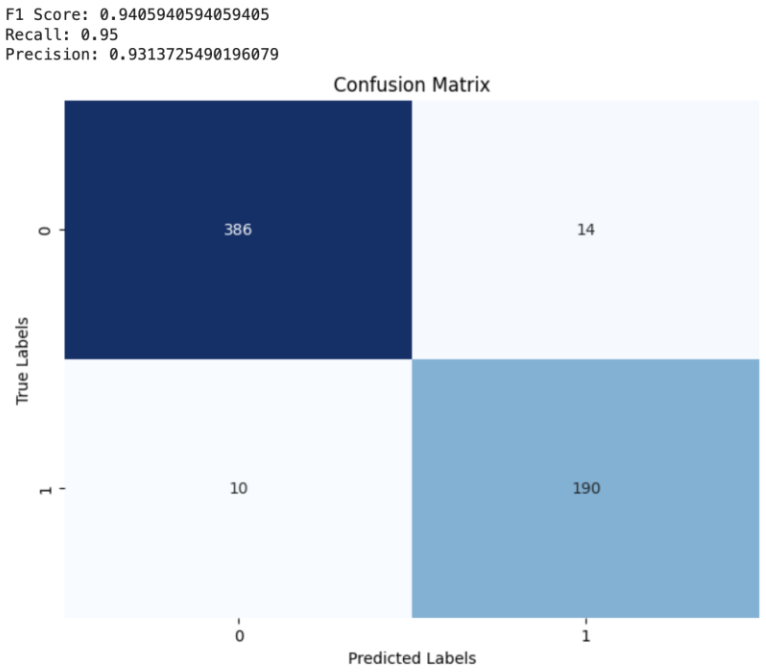

```
Accuracy: 0.9967
Precision: 1.0000
Recall: 1.0000
F1 Score: 1.0000
```





MistralAI: Plain and fine-tuned

- Plain MistralAI without any training performs poorly with the basic prompt template
- Further, I fine-tuned the model using PEFT and QLoRA
- Inference time of fine-tuned MistralAI is too long to evaluate it on the test data
- Fine tuning adapts the style, tone, and vocabulary of LLMs. Hence, it won't be as effective as RAG or finetuning anyway.

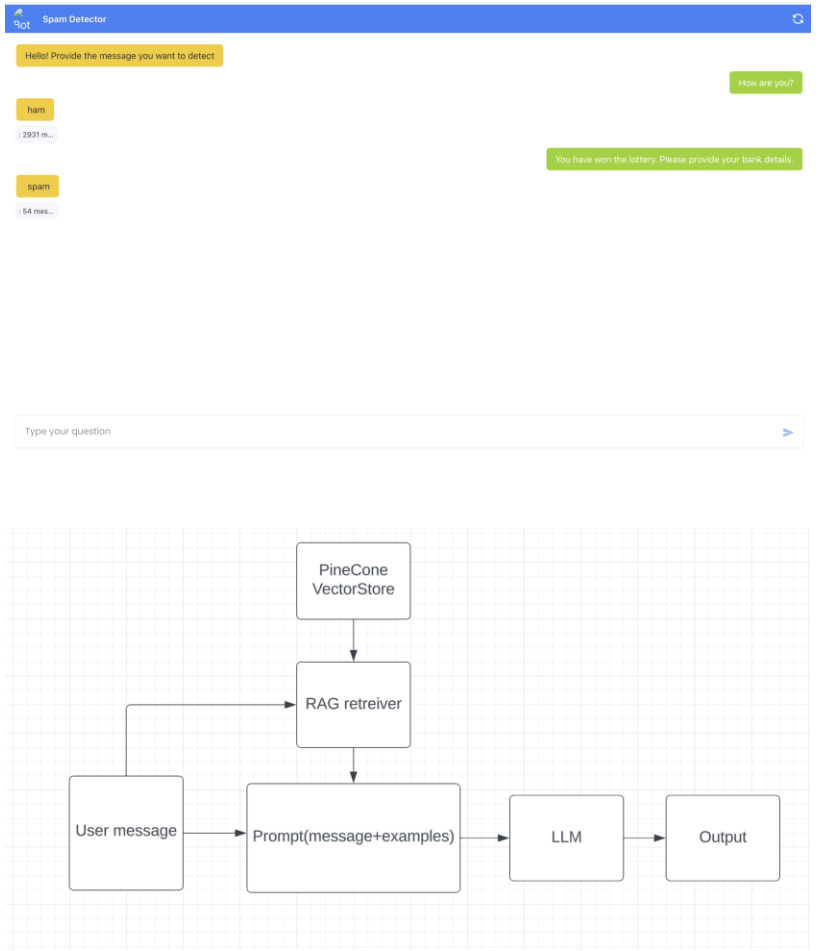


MistralAI : Few shot Learning + RAG

+ In few shot learning we provide examples of correctly identified messages along with the prompt
+ By implementing RAG, we can find the ideal set of examples from vector store to enhance the response further.

Chatbot Application UI

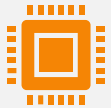
- MistralAI: few shot learning + RAG gave us the best results and we will be developing a chatbot which leverages similar approach at the backend.
- Chatbot app is hosted on huggingface space and, also returns the source documents for better understanding.
- Chatbot:Space link



Findings & Conclusion

RoBERTa performs well in general but are outperformed by MistralAI in the very-few-shot setting.

While LLMs are more robust and perform better in most cases, they require long training and inference times.



The practical application of LLMs is hindered by their substantial computational requirements for training and deployment,

Thanks