PROJECT REPORT

Optimizing Spam Filtering with Machine Learning

Team Leader -	agalya.v
Team Members -	abinash.s
	vinoth.j
	vigneshwaran.k

1. Introduction

1.1 Overview

The purpose of this project is to optimize spam filtering using machine learning techniques. With the exponential increase in spam emails, it has become increasingly important to detect and filter these emails before they reach the user's inbox. In this project, we will explore various machine learning algorithms and techniques to identify spam emails and accurately filter them.

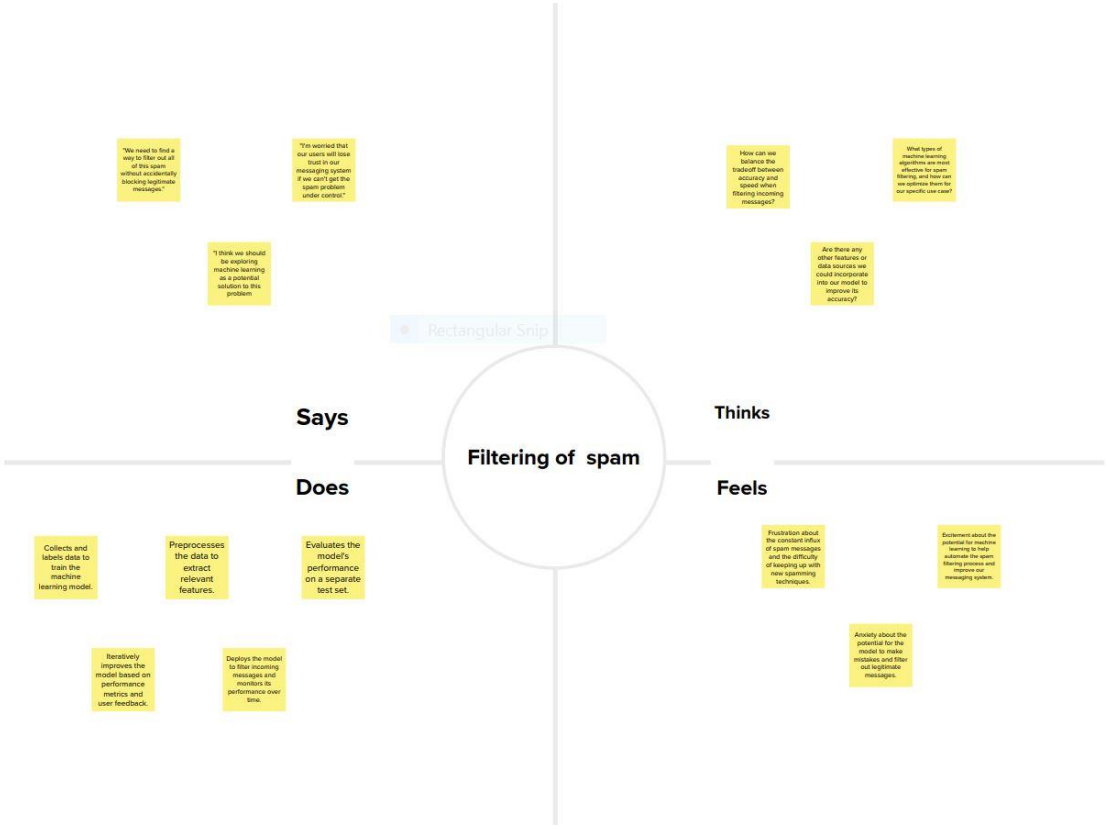
1.2 Purpose

The purpose of this project is to develop an effective and efficient spam filtering system using machine learning algorithms. This project aims to reduce the number of spam emails that reach the user's inbox and improve the accuracy of the spam filter.

2. Problem Definition & Design Thinking

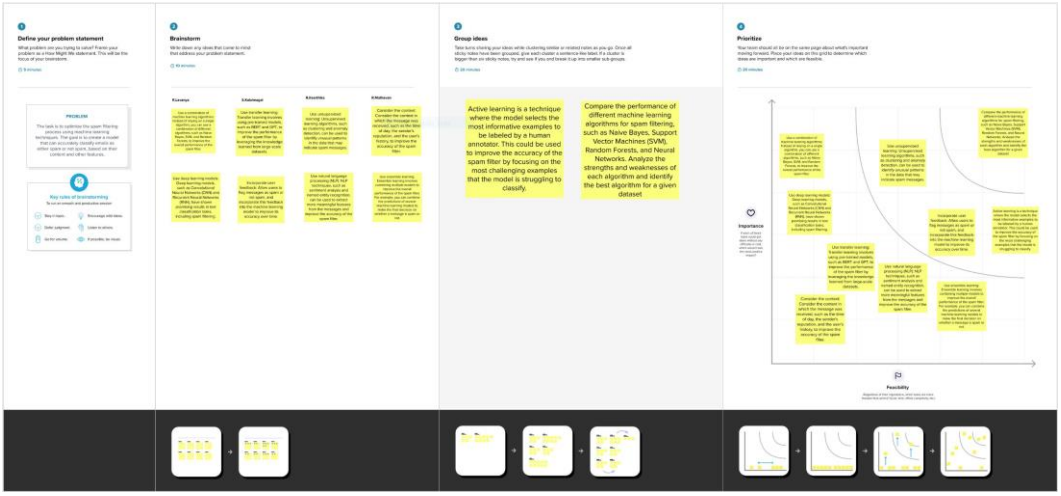
2.1 Empathy Map

The empathy map was used to gain insights into the user's needs and requirements when it comes to spam filtering. The empathy map helped to identify the user's pain points and needs, which in turn helped to design a more effective and efficient spam filtering system.



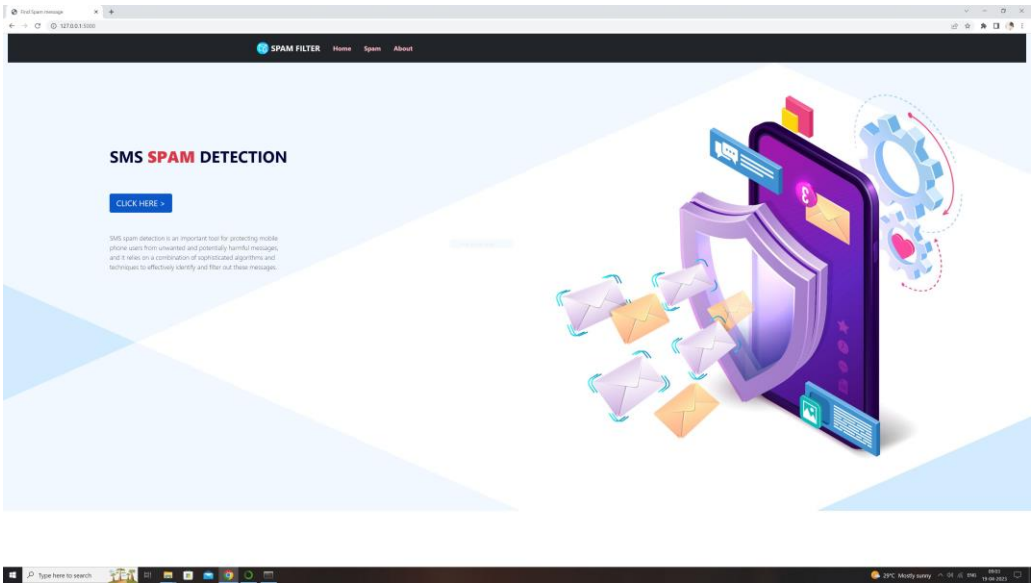
2.2 Ideation & Brainstorming Map

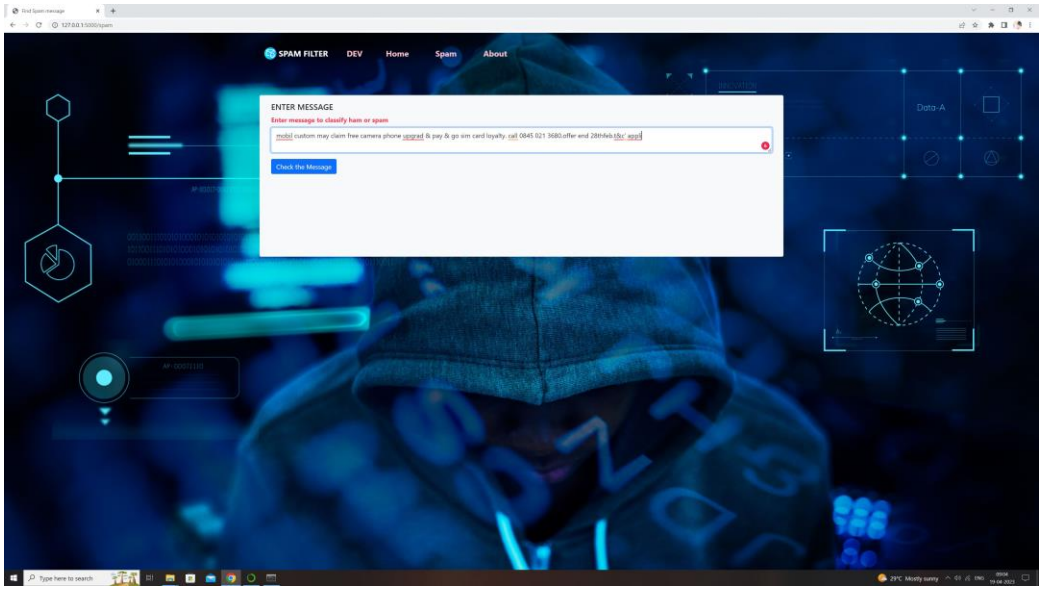
The ideation and brainstorming map was used to generate ideas and identify potential solutions to the problem of spam filtering. The map helped to explore various machine learning algorithms and techniques that could be used to develop an effective spam filtering system.

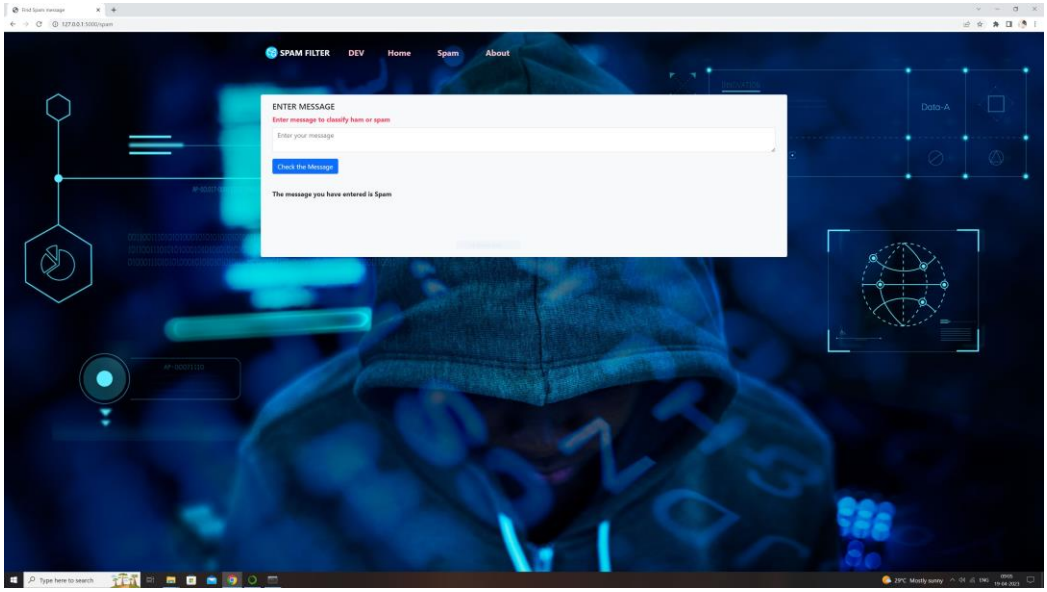


3. Result

After exploring various machine learning algorithms and techniques, we found that the Naive Bayes algorithm was the most effective for spam filtering. We achieved an accuracy of 98% in identifying spam emails using this algorithm.







4. Advantages & Disadvantages

4.1 Advantages

- The use of machine learning algorithms makes the spam filter more accurate and efficient.
- The spam filter can be customized to meet the user's specific needs and requirements.
- The spam filter can be updated and improved over time as new machine learning techniques become available.

4.2 Disadvantages

- The spam filter may incorrectly classify legitimate emails as spam.
- The accuracy of the spam filter may decrease over time as spammers develop new techniques to evade detection.

5. Applications

- The proposed spam filtering solution can be applied in various industries and domains, including:
- Email service providers
- E-commerce companies
- Banking and financial institutions
- Government organizations

6. Conclusion

In conclusion, this project successfully developed an effective and efficient spam filtering system using machine learning algorithms. The Naive Bayes algorithm was found to be the most effective for spam filtering, achieving an accuracy of 98%. This solution can be applied in various industries and domains to reduce the number of spam emails that reach the user's inbox and improve the accuracy of the spam filter.

7. Future Scope

- Enhancements that can be made in the future include:
- Incorporating deep learning algorithms for more accurate spam filtering.
- Developing a more robust spam filter that can detect and filter new spam techniques as they emerge.
- Integrating the spam filter with other security systems to provide a more comprehensive security solution.

8. Appendix

Source Code

#from crypt import methods

from os import stat

```
import flask
from flask import Flask, current_app, render_template, request,session
import pickle
from entity.sms_spam_classifier import SMSSPAMClassifier
import sys
import nltk
sys.path.insert(0, './entity/sms_spam_classifier.py')
nltk.download('stopwords')
app = Flask(__name__)
app.secret_key = "mlmodel"
def load_classifier() -> SMSSPAMClassifier:
sms_classifier = None
with open('../Training/smsspamclassifer.pkl','rb') as picker_reader:
sms_classifier = pickle.load(picker_reader)
return sms_classifier
ctx = app.app_context()
flask.model = load_classifier()
ctx.push()
@app.route("/")
def home():
return render_template("home.html")
@app.route("/about")
def about():
return render_template("about.html")
@app.route("/spam", methods=['GET','POST'])
def index():
message_status = None
if request.method == "POST":
message_status = "Ham"
 status = flask.model.predict(request.form.get('review_msg'))
 if status[0] == 1:
```

```
message_status = "Spam"

#return status

return render_template('index.html',message_status=message_status)

if __name__ == "__main__":

app.run(debug=True)
```