

Optimizing Spam Filtering with Machine Learning

Introduction

Overview 1.1

Over recent years, as the popularity of mobile phone devices has increased, Short Message Service (SMS) has grown into a multi-billion dollar industry.

At the same time, reduction in the cost of messaging services has resulted in growth in unsolicited commercial advertisements (spams) being sent to mobile phones.

Due to Spam SMS, Mobile service providers suffer from some sort of financial problems as well as it reduces calling time for users.

Unfortunately, if the user accesses such Spam SMS they may face the problem of virus or malware.

When SMS arrives at mobile it will disturb mobile user privacy and concentration. It may lead to frustration for the user.

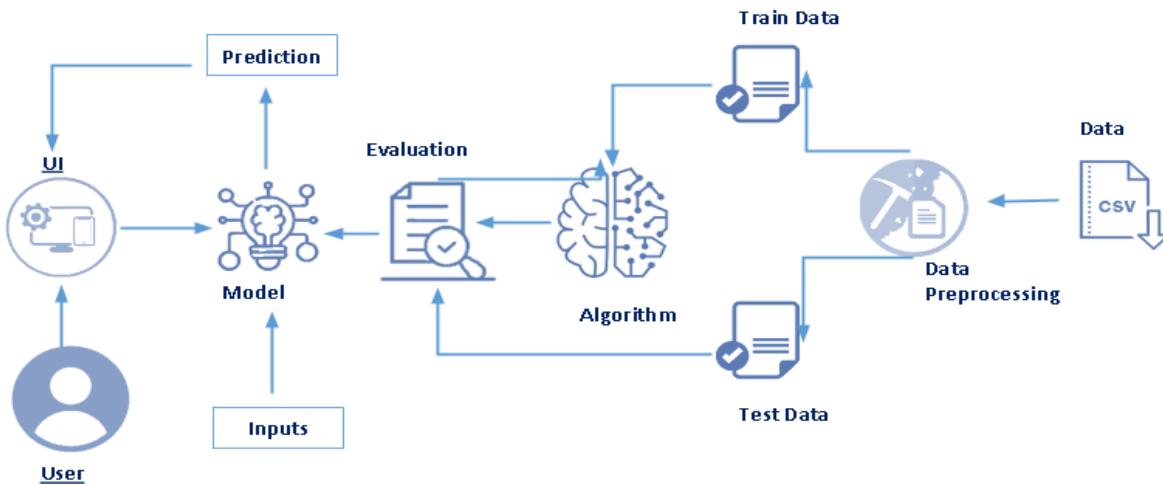
So Spam SMS is one of the major issues in the wireless communication world and it grows day by day.

To avoid such Spam SMS people use white and black list of numbers.

But this technique is not adequate to completely avoid Spam SMS. To tackle this problem it is needful to use a smarter technique which correctly identifies Spam SMS.

Natural language processing technique is useful for Spam SMS identification. It analyses text content and finds patterns which are used to identify Spam and Non-Spam SMS.

Technical Architecture:



Technical about the project:

Before any feature extraction or modeling can be done, the email messages must be preprocessed. This may involve removing HTML tags, converting all text to lowercase, removing stop words (common words like "the" and "and"), stemming or lemmatizing words (reducing them to their base form), and other text cleaning techniques.

The next step is to extract features from the preprocessed email messages. The most common approach is the bag-of-words model, where each email is represented as a vector of word frequencies. Another common approach is the term frequency-inverse document

frequency (TF-IDF) model, which assigns weights to each word based on its frequency in the email and across the entire dataset.

Project Description:

The goal of this project is to build a spam filter that can accurately classify incoming messages as either spam or legitimate. Spam, also known as junk mail, refers to unsolicited or unwanted messages sent in bulk to a large number of recipients

The spam filter will be built using machine learning algorithms that can learn to distinguish between spam and legitimate messages based on patterns and characteristics in the data. The project will involve the following steps:

Data Collection: A large dataset of emails will be collected and preprocessed. The dataset will contain both spam and legitimate emails to train the machine learning model.

Feature Extraction: The emails will be transformed into numerical feature vectors that can be used as input to the machine learning model. This will involve techniques such as bag-of-words and TF-IDF

The project will require knowledge of machine learning, natural language processing, and programming languages such as Python. The final product will be a highly effective and reliable spam filter that can save users time and reduce the risk of falling victim to phishing and other scams.

Project Flow:

- User interacts with the UI to enter the input.

- Entered input is analysed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

- Define Problem / Problem Understanding
 - Specify the business problem
 - Business requirements
 - Literature Survey
 - Social or Business Impact.
- Data Collection & Preparation
 - Collect the dataset
 - Data Preparation
- Exploratory Data Analysis
 - Descriptive statistical
 - Visual Analysis
- Model Building
 - Training the model in multiple algorithms
 - Testing the model
- Performance Testing & Hyperparameter Tuning
 - Testing model with multiple evaluation metrics

- Comparing model accuracy before & after applying hyperparameter tuning

- Model Deployment

- Save the best model

- Integrate with Web Framework

- Project Demonstration & Documentation

- Record explanation Video for project end to end solution

- Project Documentation-Step by step project development procedure.

Project Structure:

Create the Project folder which contains files as shown be

Name	Date Modified
> Folder Flask	24-01-2023 19:09
> Folder Images	24-01-2023 19:09
> Folder static	24-01-2023 19:09
✓ Folder templates	25-01-2023 11:09
</> index.html	24-01-2023 19:09
</> result.html	25-01-2023 11:09
</> spam.html	24-01-2023 19:09
Python app.py	25-01-2023 10:50
Python app1.py	25-01-2023 11:09
01 cv1.pkl	25-01-2023 10:48
Procfile	24-01-2023 19:09
requirements.txt	24-01-2023 19:09
Python Spam SMS Classifier - Deployment.py	24-01-2023 19:09
01 spam.h5	25-01-2023 09:21

- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- Spam.h5 is our saved model. Further we will use this model for flask integration.

purpose

The purpose of spam filtering is to automatically identify and remove unwanted or unsolicited messages, commonly referred to as "spam,"

from an individual's email inbox or a company's email server. Spam filtering serves several important purposes:

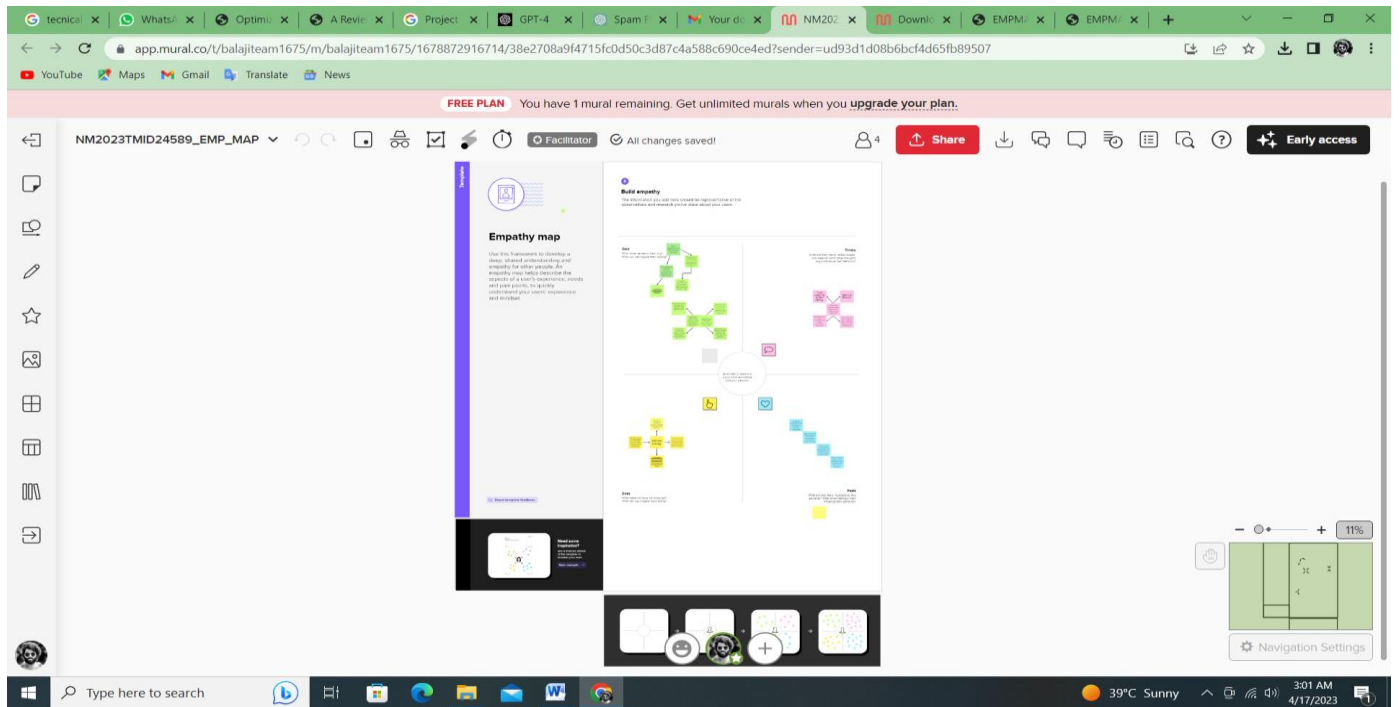
Spam filtering removes unwanted messages from an individual's email inbox, reducing clutter and making it easier to find important messages. This can save time and increase productivity.

Spam messages often contain scams and fraudulent offers, such as fake lottery winnings or investment opportunities. Spam filtering helps prevent individuals from falling victim to these scams and losing money.

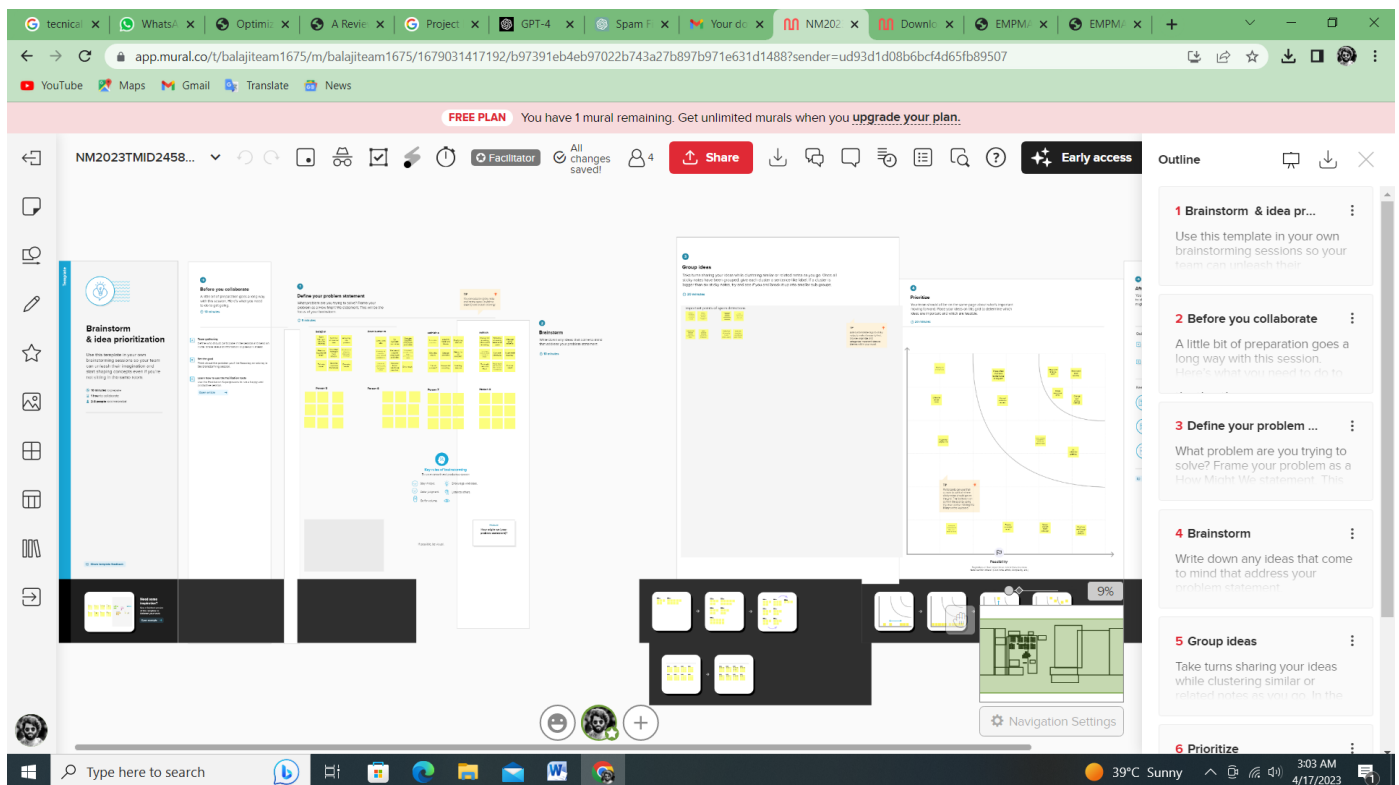
2.Problem Definition and Thinking:

Machine learning has become an increasingly popular tool in recent years ,given its ability to automatically detect patterns in data and make predictions about future events .this can be extremely useful for asking decisions in a wide range of domains,from financial trading to medical diagnoses Empathy Map

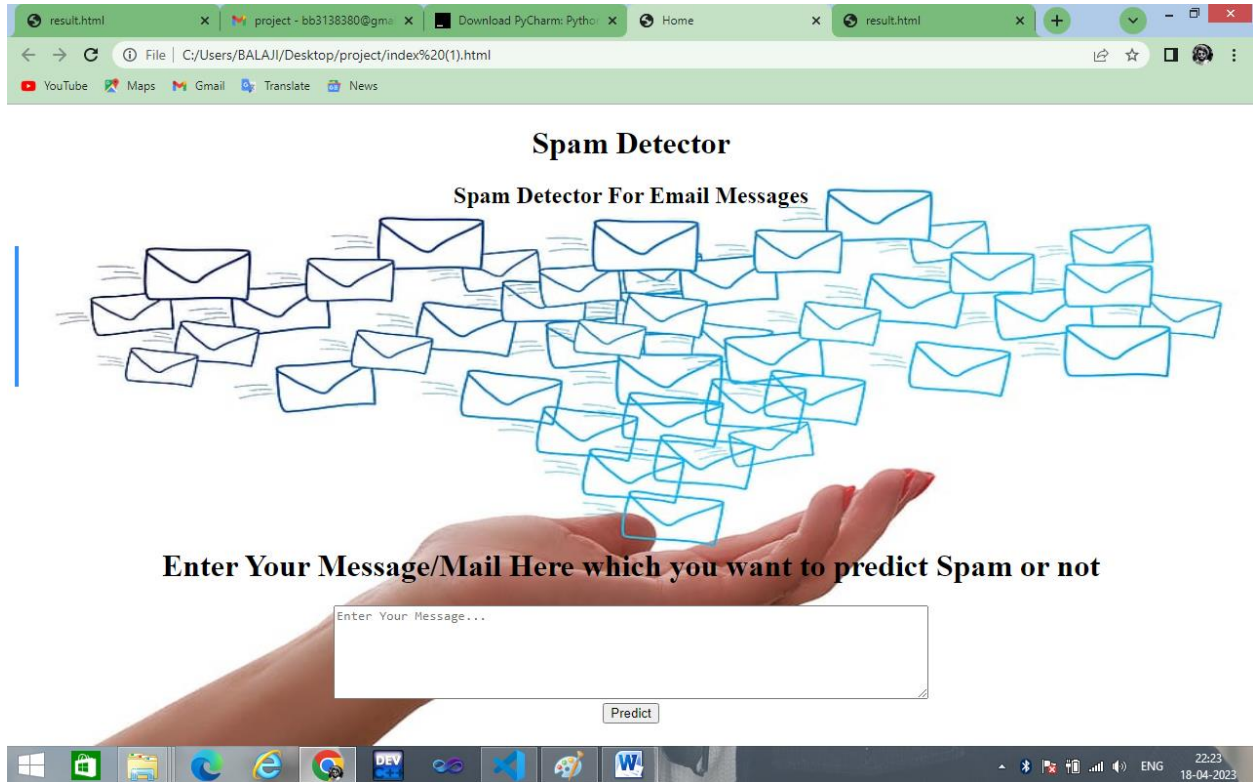
2.1 Emphy map

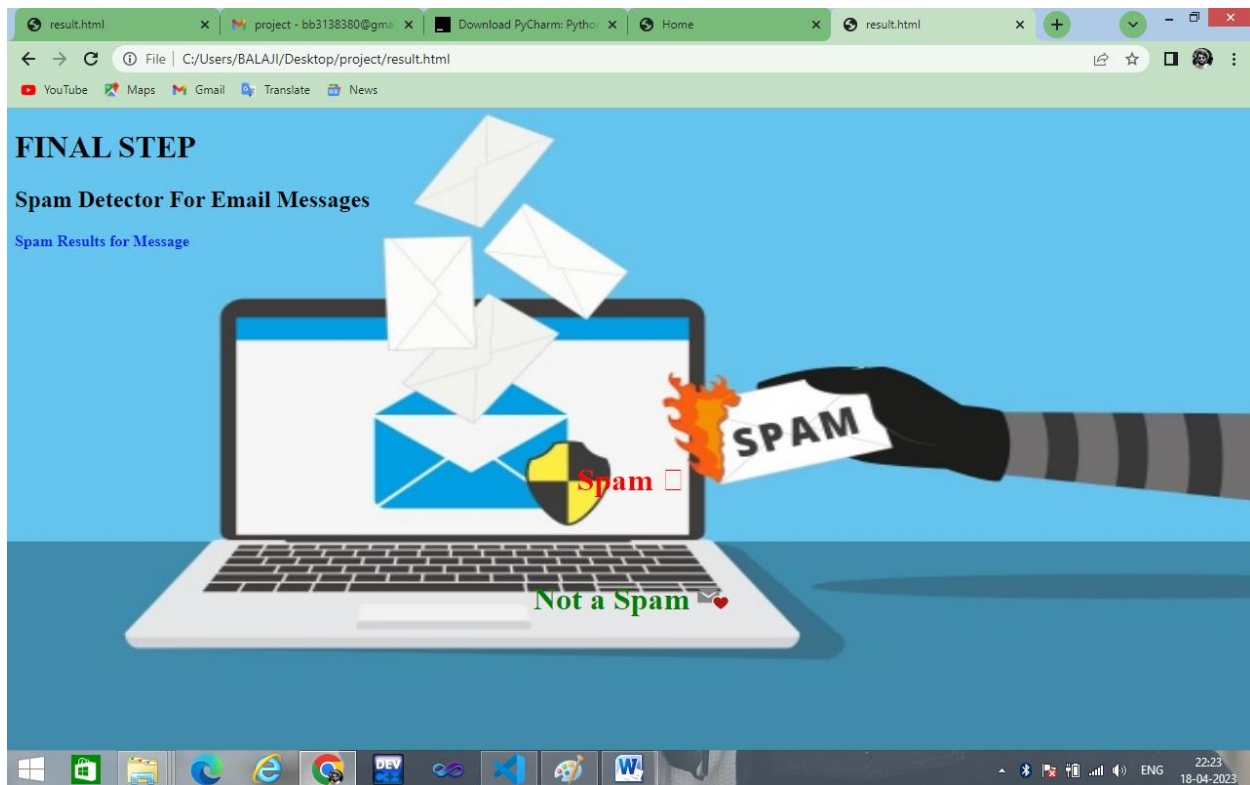
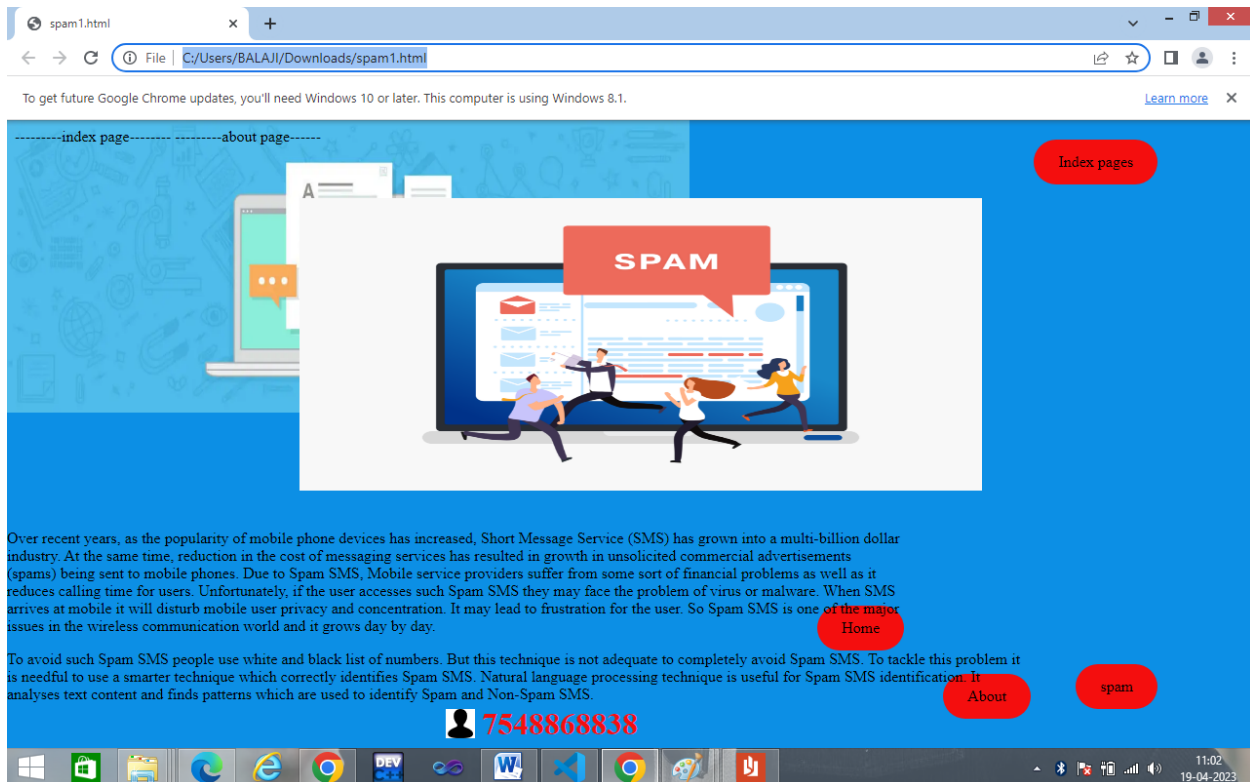


2.2 Ideation & Brainstorming Map



3.RESULT





4.ADVANTAGES AND DISADVANTAGES

Advantages of Spam Filtering:

1. **Saves Time and Increases Productivity:** Spam filtering helps to reduce the number of unwanted and irrelevant messages that users have to go through. This helps to save time and increase productivity.
2. **Protects Against Malware and Phishing Attacks:** Many spam messages contain malware or phishing attempts, which can be used to steal sensitive information or infect a computer or network. Spam filtering helps to protect against these threats by identifying and blocking suspicious messages.
3. **Prevents Fraud and Scams:** Spam messages often contain fraudulent offers, such as fake lottery winnings or investment opportunities. Spam filtering helps to prevent individuals from falling victim to these scams and losing money.
4. **Reduces Server Load:** Spam filtering reduces the amount of traffic on email servers, which can help to reduce server load, improve system performance, and save costs on bandwidth and storage.

Disadvantages of Spam Filtering:

1. **False Positives:** Spam filters can occasionally identify legitimate messages as spam, which can result in important emails being missed or delayed.
2. **False Negatives:** Conversely, spam filters can occasionally fail to identify spam messages, which can result in unwanted messages reaching the inbox.

3. **Overreliance on Technology:** Some users may become over-reliant on spam filtering technology and fail to properly evaluate the content of their messages, which can result in important messages being missed.
4. **Complexity:** Spam filtering technology can be complex to implement and maintain, which can result in higher costs and potential issues with system integration.

Overall, while spam filtering offers many advantages, there are also potential disadvantages that need to be considered. Proper configuration and ongoing monitoring can help to mitigate these issues and ensure that spam filtering remains an effective tool for protecting against unwanted and harmful messages.

5. APPLICATIONS

Spam filtering has many applications across various industries and contexts. Here are some examples:

1. **Email Providers:** Spam filtering is a crucial tool for email providers like Gmail, Yahoo, and Outlook. By filtering out unwanted messages, email providers can provide a better user experience for their customers and reduce the risk of malware or phishing attacks.
2. **E-commerce:** Online retailers use spam filtering to reduce the number of unsolicited marketing messages that they send to customers. This helps to improve the effectiveness of email

marketing campaigns and reduce the risk of customers unsubscribing from mailing lists.

3. **Social Media:** Social media platforms like Facebook and Twitter use spam filtering to identify and remove fake accounts and spammy content, such as comments or direct messages with links to phishing sites.
4. **Mobile Networks:** Mobile network operators use spam filtering to identify and block unwanted SMS messages, which can reduce the risk of customers being charged for unsolicited premium SMS services.
5. **Government Agencies:** Government agencies use spam filtering to identify and block malicious emails that may contain malware, phishing attempts, or fraudulent messages. This helps to protect sensitive government data and prevent cyberattacks.
6. **Educational Institutions:** Educational institutions use spam filtering to protect their networks from spam and phishing attempts, which can be especially important for protecting student data and preventing identity theft.

Overall, spam filtering has many practical applications across a wide range of industries and contexts, and is an important tool for protecting against unwanted and potentially harmful messages.

CONCLUSION

In conclusion, spam filtering is an important tool for protecting against unwanted and potentially harmful messages across various industries

and contexts. It serves to reduce clutter in inboxes, protect against malware and phishing attacks, prevent fraud and scams, save bandwidth and storage, and more. While there are potential disadvantages such as false positives and over-reliance on technology, proper configuration and ongoing monitoring can help to mitigate these issues and ensure that spam filtering remains an effective tool for protecting against unwanted and harmful messages. Overall, spam filtering plays a critical role in ensuring a more efficient, secure, and streamlined email experience.

spam filtering is an essential aspect of managing online communication, especially for individuals and businesses who receive a large volume of emails. By using advanced machine learning algorithms and techniques, spam filters can effectively block unwanted and potentially harmful messages, while allowing legitimate emails to pass through. Effective spam filtering not only improves email management and productivity but also enhances cybersecurity by reducing the risk of phishing attacks, malware infections, and other online threats. Therefore, it is crucial to regularly update and optimize spam filters to keep up with evolving spamming techniques and stay protected against new threats.

7.FUTURE SCOPE

The future scope for spam filtering is promising, as advancements in machine learning and artificial intelligence (AI) technologies continue to revolutionize the way we manage online communication. Here are some potential areas of development in spam filtering:

1. Deep learning algorithms: Deep learning algorithms have shown promising results in detecting and filtering out spam messages. As these algorithms continue to improve, we can expect them to

become even more accurate in identifying new and sophisticated spamming techniques.

2. Natural Language Processing (NLP): NLP can be used to analyze the content of emails and identify language patterns that are commonly associated with spam messages. With further development, NLP can help to identify and filter out more subtle and context-dependent spam messages.
3. Collaborative filtering: Collaborative filtering techniques can be used to analyze email communication patterns and identify spam messages based on the behavior of other users in the network. This approach could be particularly useful in identifying and filtering out spam messages that are tailored to specific individuals or organizations.
4. User feedback and reporting: By enabling users to report and provide feedback on spam messages, spam filters can improve their accuracy and learn to identify new and emerging spamming techniques.

Overall, the future of spam filtering looks promising, with continued advancements in machine learning, AI, and NLP technologies. As these technologies continue to evolve, we can expect spam filters to become even more effective in protecting users against spam messages and other online threats

8.APPENDIX

SOURCE CODE

WhatsApp GPT-4 Spam Filter Solution Inbox (153) - bb3138380@gmail.com spam.ipynb - Colaboratory

colab.research.google.com/drive/1SxD60lu908_Y8v31mw_MSS9SULEsvvG

spam.ipynb

File Edit View Insert Runtime Tools Help Last edited on April 10

+ Code + Text

optimizing spam filtering with machine learning

```
[ ] import tensorflow as tf
print(tf.__version__)

2.12.0

[ ] # import libraries for reading data, exploring and plotting
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
%matplotlib inline
# library for train test split
from sklearn.model_selection import train_test_split
# deep learning libraries for text pre-processing
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
# Modeling
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, GlobalAveragePooling1D, Dense, Dropout, LSTM, Bidirectional

[ ] url = 'https://raw.githubusercontent.com/ShresthaSudip/SMS_Spam_Detection_DNN_LSTM_BiLSTM/master/SMSSpamCollection'
```

Type here to search 38°C Sunny 3:56 AM 4/17/2023

WhatsApp GPT-4 Spam Filter Solution Inbox (153) - bb3138380@gmail.com spam.ipynb - Colaboratory

colab.research.google.com/drive/1SxD60lu908_Y8v31mw_MSS9SULEsvvG#scrollTo=1a_dTLUc2UY

spam.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text

```
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, GlobalAveragePooling1D, Dense, Dropout, LSTM, Bidirectional

url = 'https://raw.githubusercontent.com/ShresthaSudip/SMS_Spam_Detection_DNN_LSTM_BiLSTM/master/SMSSpamCollection'
messages = pd.read_csv(url, sep='\t', names=["label", "message"])
messages[:3]
```

label	message
0 ham	Go until Jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there g...
1 ham	Ok lar... Joking wif u oni...
2 spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive ...

```
[ ] messages.describe()
```

	label	message
count	5572	5572
unique	2	5169
top	ham	Sorry, I'll call later
freq	4825	30

```
[ ] duplicatedRow = messages[messages.duplicated()]
print(duplicatedRow[:5])
```

Executing (4s) <cell line: 1> > <module> > <module> > <module> > <module> > <module> > <module> > <module> > <module> > <module>

Type here to search 38°C Sunny 3:57 AM 4/17/2023

WhatsApp GPT-4 Spam Filter Solution. Inbox (153) - bb3138380@gmail.com spam.ipynb - Colaboratory

colab.research.google.com/drive/1SxD60lu908_Y8v31mw_MSS9SULEsvG#scrollTo=1a_dTLUc2UY

spam.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text

```
[ ] duplicatedRow = messages[messages.duplicated()]
print(duplicatedRow[:5])
```

```
label \
103 ham
154 ham
207 ham
223 ham
326 ham
```

```
message
103 As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your call...
154 As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your call...
207 As I entered my cabin my PA said, '' Happy B'day Boss !!!''. I felt special. She asked me 4 lunch....
223 Sorry, I'll call later
326 No calls..messages..missed calls
```

new section

```
#Give concise summary of a DataFrame.info()
messages.groupby('label').describe().T
```

	label	ham	spam
message	count	4825	747
	unique	4516	653

1s completed at 3:57 AM

(1) WhatsApp GPT-4 Spam Filter Solution. Inbox (153) - bb3138380@gmail.com spam.ipynb - Colaboratory

colab.research.google.com/drive/1SxD60lu908_Y8v31mw_MSS9SULEsvG#scrollTo=1a_dTLUc2UY

spam.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[ ] top Sorry, I'll call later Please call our customer service representative on FREEPHONE 0800 145 4742 between 9am-11pm as y...
```

	freq	
	30	4

```
[ ] messages.groupby('label').describe().T
```

	label	ham	spam
message	count	4825	747
	unique	4516	653
	top	Sorry, I'll call later Please call our customer service representative on FREEPHONE 0800 145 4742 between 9am-11pm as y...	
	freq	30	4

```
[ ] # Get all the ham and spam emails
ham_msg = messages[messages.label == 'ham']
spam_msg = messages[messages.label == 'spam']
# Create numpy list to visualize using wordcloud
ham_msg_text = " ".join(ham_msg.message.to_numpy().tolist())
spam_msg_text = " ".join(spam_msg.message.to_numpy().tolist())
```

```
[ ] # wordcloud of ham messages
ham_msg_cloud = WordCloud(width=520, height=260, stopwords=STOPWORDS, max_font_size=50, background_color="black", colormap='Blues').generate(ham_msg_text)
plt.figure(figsize=(16,10))
plt.imshow(ham_msg_cloud, interpolation='bilinear')
plt.axis('off') # turn off axis
plt.show()
```

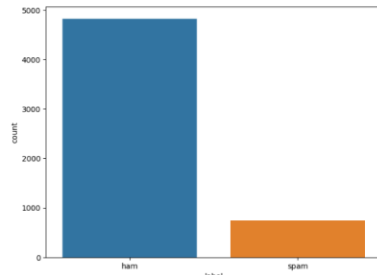
1s completed at 3:57 AM

[illegible]

Colaboratory interface showing a Jupyter Notebook with code for analyzing spam data.

```
[ ] # we can observe imbalance data here
plt.figure(figsize=(8,6))
sns.countplot(x='label', data=messages)
# Percentage of spam messages
print((len(spam_msg)/len(ham_msg))*100) # 15.48%
```

15.488528874895



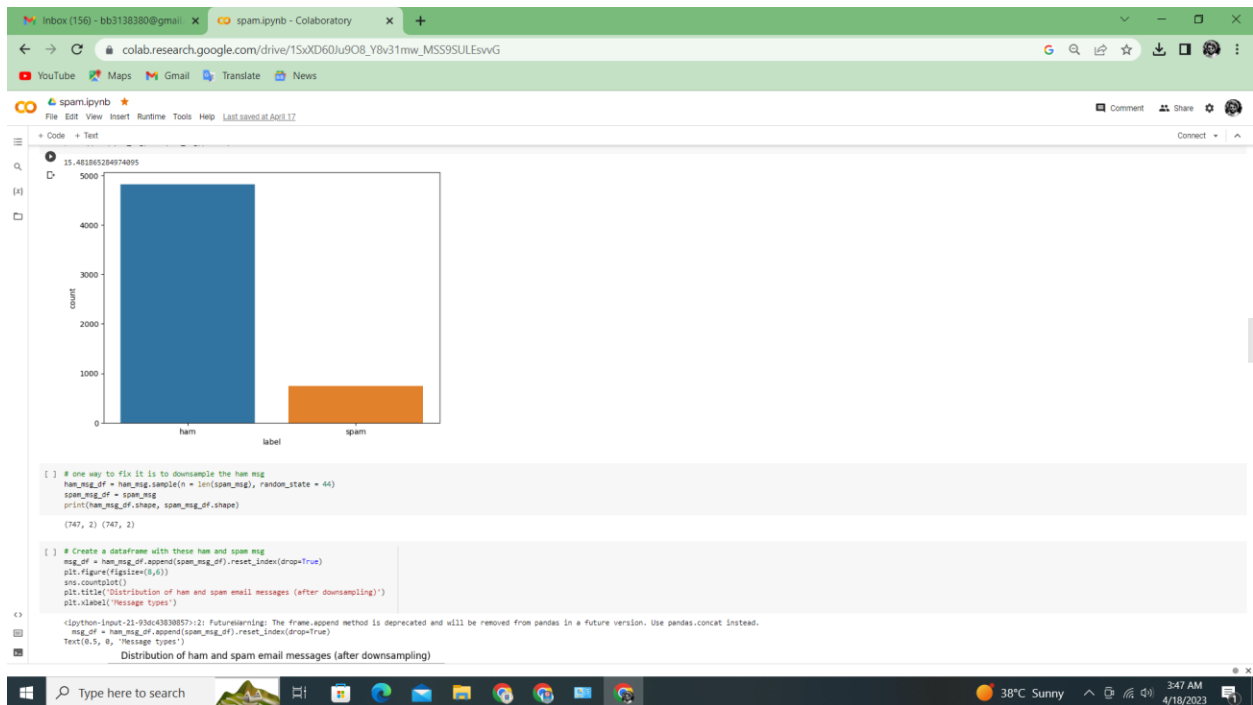
label	count
ham	4800
spam	750

```
[ ] # one way to fix it is to downsample the ham msg
ham_msg_of = ham_msg.sample(n = len(spam_msg), random_state = 44)
spam_msg_of = spam_msg
print(ham_msg_of.shape, spam_msg_of.shape)

(747, 2) (747, 2)
```

```
[ ] # Create a dataframe with these ham and spam msg
msg_of = ham_msg_of.append(spam_msg_of).reset_index(drop=True)
plt.figure(figsize=(8,6))
sns.countplot()
plt.show()
```

completed at 3:57 AM



Inbox (156) - bb3138380@gmail.com spam.ipynb - Colaboratory

colab.research.google.com/drive/1SxD60lu9O8_Y8v31mw_MSS9SULEsvvG

spam.ipynb

File Edit View Insert Runtime Tools Help Last saved at April 17

Code + Text

```
[ ] # Get length column for each text
msg_of['text_length'] = msg_of['message'].apply(len)
# Calculate average length by label types
labels = msg_of.groupby('label').mean()
labels

text_length
label
ham 73.238286
spam 138.670683

[ ] # Map ham label as 0 and spam as 1
msg_of['msg_type'] = msg_of['label'].map({'ham': 0, 'spam': 1})
msg_label = msg_of['msg_type'].values
# Split data into train and test
train_msg, test_msg, train_labels, test_labels = train_test_split(msg_of['message'], msg_label, test_size=0.2, random_state=34)

# Defining pre-processing hyperparameters
max_len = 50
trunc_type = "post"
padding_type = "post"
oov_tok = "<OOV>"
vocab_size = 500

[ ] tokenizer = Tokenizer(num_words = vocab_size, char_level=False, oov_token = oov_tok)
tokenizer.fit_on_texts(train_msg)

[ ] # Get the word_index
word_index = tokenizer.word_index
word_index

{'<OOV>': 1,
 'to': 2,
 'you': 3,
 'a': 4,
 'i': 5,
 'call': 6,
 'the': 7,
 'at': 8,
 'your': 9,
 'for': 10,
 'is': 11,
 '2': 12,
 'and': 13,
 'now': 14,
 'free': 15,
```

Type here to search

38°C Sunny 3:48 AM 4/18/2023

Inbox (156) · bb3138380@gmail.com

spam.ipynb - Colaboratory

colab.research.google.com/drive/1SxXD60luO8_Y8v31mw_MSS9SULesvwG

YouTube Maps Gmail Translate News

spam.ipynb

File Edit View Insert Runtime Tools Help Last saved at April 17

+ Code + Text

Comment Share

```
[ ] # fitting a dense spam detector model
num_epochs = 30
early_stop = EarlyStopping(monitor='val_loss', patience=3)
History = model.fit(training_padded, train_labels, epochs=num_epochs, validation_data=(testing_padded, test_labels), callbacks=[early_stop], verbose=2)

[ ] # Model performance on test data
model.evaluate(testing_padded, test_labels)

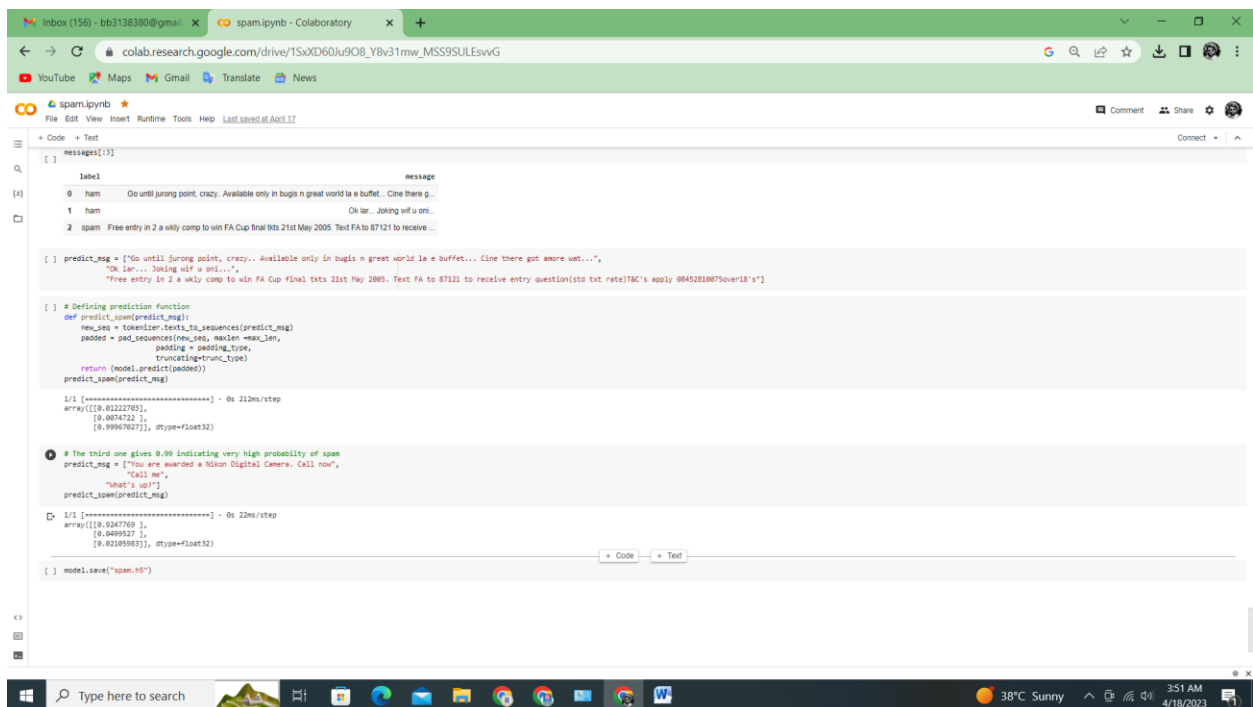
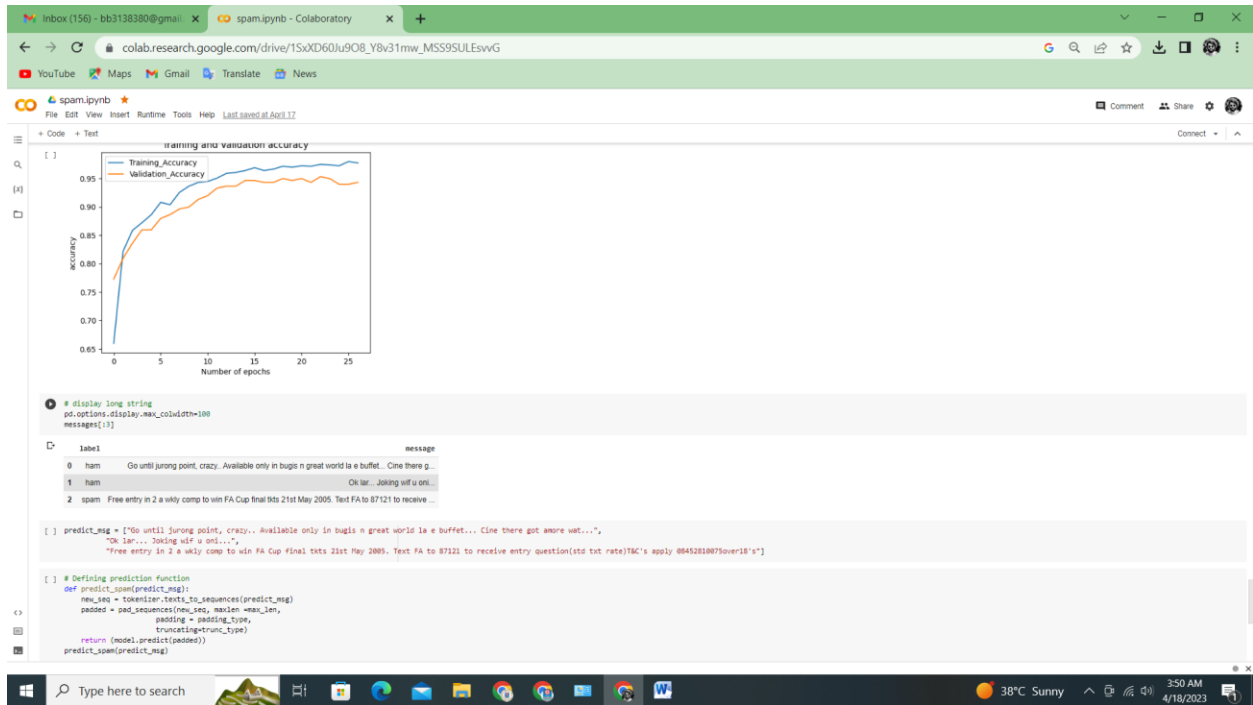
18/10 [=====] - 0s 2ms/step - loss: 0.1234 - accuracy: 0.9431
[0.123402208392479, 0.943143784999874]

[ ] # Read as a dataframe
metrics = pd.DataFrame(History.history)
# Rename column
metrics.rename(columns={'loss': 'Training_Loss', 'accuracy': 'Training_Accuracy', 'val_loss': 'Validation_Loss', 'val_accuracy': 'Validation_Accuracy'}, inplace=True)
def plot_graphs(ver1, ver2, string):
    metrics[[ver1, ver2]].plot()
    plt.title('Training and Validation' + string)
    plt.xlabel('Number of epochs')
    plt.ylabel(string)
    plt.legend([ver1, ver2])

plot_graphs(['Training_Loss', 'Validation_Loss', 'loss'])
```

Training and Validation loss

Epoch	Training Loss	Validation Loss
0	0.68	0.68
5	0.35	0.35
10	0.18	0.18
15	0.12	0.13
20	0.08	0.12
25	0.05	0.12



```
File Edit View Navigate Code VCS Help pythonProject - app.py
app.py
No Python interpreter configured for the project Use C:\Users\ELCOT\anaconda3\python.exe Configure Python interpreter
2 import pickle
3 import numpy as np
4 import re
5 import nltk
6 from nltk.corpus import stopwords
7 from nltk.stem import porterstemmer
8 from tensorflow.keras.models import load_model
9 # load the multinomial naive Bayes model and CountVector
10
11 # load the multinomial naive Bayes model
12 load_model = load_model('spam.h5')
13 cv = pickle.load(open('cv1.pkl','rb'))
14 app = Flask(__name__)
15
16 @app.route('/') # rendering the html template
17 def home():
18     return render_template('home.html')
19
20 @app.route('/spam',methods=['POST','GET'])
21 def prediction(): # route which will take you to the prediction page
22     return render_template('spam.html')
23
24 @app.route('/predict',methods=['POST'])
25 def predict():
26     if request.method == 'POST':
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
File Edit View Navigate Code VCS Help pythonProject - app.py
app.py
No Python interpreter configured for the project Use C:\Users\ELCOT\anaconda3\python.exe Configure Python interpreter
19
20 @app.route('/spam',methods=['POST','GET'])
21 def prediction(): # route which will take you to the prediction page
22     return render_template('spam.html')
23
24 @app.route('/predict',methods=['POST'])
25 def predict():
26     if request.method == 'POST':
27         message = request.form['message']
28         data = message
29
30         new_review = str(data)
31         print(new_review)
32         new_review = re.sub('[^a-zA-Z]', '', new_review)
33         new_review = new_review.lower()
34         new_review = new_review.split()
35         ps = porterstemmer()
36         all_stopwords = stopwords.words('english')
37         all_stopwords.remove('not')
38         new_review = [ps.stem(word) for word in new_review if not word in set(all_stopwords)]
39         new_review = ' '.join(new_review)
40         new_corpus = [new_review]
41         new_X_test = cv.transform(new_corpus).toarray()
42         print(new_X_test)
43         new_Y_pred = loaded_model.predict(new_X_test)
44
45 predict()
```

