P3: Spam Classification Project

A Project Report

submitted in partial fulfillment of the requirements

of

AICTE Internship on AI: Transformative Learning

with

TechSaksham – A joint CSR initiative of Microsoft & SAP

by

Ashok pal

ashokpal2094@gmail.com

Under the Guidance of

Abdul Aziz Mohammad

ACKNOWLEDGEMENT

I would like to take this opportunity to thanks and express our sincere appreciation to all those who have supported and contributed to the completion of this thesis, both directly and indirectly.

First and foremost, we extend our deepest gratitude to my distinguished supervisor, Abdul Aziz Mohammad, Pavan Sumohana , whose exceptional mentorship has been a cornerstone of this project. His insightful guidance, thoughtful suggestions, and constructive critiques have not only inspired and motivated me but have also shaped the direction and success of this research. The confidence he placed in me has been a constant source of encouragement, and his unwavering support throughout this journey has been invaluable. It has been a true privilege to work with such a dedicated and knowledgeable advisor. Beyond academic assistance, his lessons and perspective have greatly influenced my development as a professional, instilling in me a sense of responsibility and integrity that will continue to guide me throughout my career.

Once again, Thank you Edunet foundation ,Techsakhsham& all team for providing this virtual training internships for free with an great experience in the world of AI and Machine Learning.

…..

ABSTRACT

The project focuses on the classification of spam emails using Natural Language Processing (NLP) and Machine Learning (ML) techniques, specifically applying Naive Bayes and CountVectorizer. Spam emails continue to be a significant problem, affecting communication efficiency and user experience, making the automation of spam detection crucial for modern email systems.

Problem Statement:

Spam emails are often designed to bypass traditional filtering mechanisms, causing users to receive unwanted or malicious messages. Manual identification is time-consuming, and automated solutions need to be efficient and accurate.

Objectives:

1. Develop a machine learning model to classify emails as "spam" or "ham" (non-spam).
2. Utilize NLP techniques, such as text preprocessing, tokenization, and vectorization, to represent email content in a suitable format for machine learning.
3. Evaluate the performance of the Naive Bayes algorithm in comparison to other classification models.

Methodology:

The project uses the Enron Email Dataset, which contains labeled examples of spam and ham emails. Text preprocessing steps like removing stopwords, punctuation, and stemming are performed. The CountVectorizer technique is used to convert email content into a bag-of-words representation. The Naive Bayes classifier, a probabilistic machine learning model, is applied to train the model on the dataset. Performance metrics such as accuracy, precision, recall, and F1-score are used for evaluation.

Key Results:

The Naive Bayes model, when combined with CountVectorizer, achieved high classification accuracy (around 98%), with good precision and recall for both spam and ham categories. The model demonstrated robustness in identifying spam emails while minimizing false positives.

Conclusion:

The use of Naive Bayes and CountVectorizer effectively classifies emails into spam and ham categories, offering a reliable solution for automated email filtering. This method provides a solid foundation for building efficient spam detection systems in real-world applications.

TABLE OF CONTENT

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Problem Statement:

The problem being addressed is the growing volume of emails that users receive daily, which often includes a mix of important, irrelevant, or unsolicited messages. This results in email inboxes becoming cluttered, making it difficult to prioritize and manage communications effectively. With the rise in email usage for both personal and professional purposes, users often face the challenge of sorting through hundreds or even thousands of emails every day. Not only does this lead to wasted time, but it also increases the likelihood of missing important emails or falling victim to malicious attacks like phishing and spam.

A lack of proper email categorization can result in inefficiencies, missed deadlines, and even data breaches. Thus, an automated solution to classify emails into meaningful categories—such as spam, promotions, social, or important—would help streamline email management, enhance productivity, and reduce security risks.This email classifier helps to increase productivity and stay away from frauds.

1.2 Motivation:

This project was chosen because of the widespread problem of email overload, which negatively impacts individuals and organizations alike. As email communication continues to be a cornerstone of both personal and professional interactions, an automated system that classifies and filters emails effectively can significantly improve productivity.

The potential applications of this project are vast. For individuals, it can reduce inbox clutter by filtering out spam, organizing messages into relevant categories, and ensuring that important emails are highlighted. For businesses, the classifier can help employees focus on work-related emails, saving time and avoiding distractions. Furthermore, by detecting and filtering spam or phishing emails, the system enhances cybersecurity, protecting users from malicious content.

Even personally, I faced the same issues of overloading email on my email – Id . Due to which I forcefully need to create more Id's to balance it. I also don't have that much time to filter all that email's.

1.3 Objective:

The main objective of this project is to design and develop a machine learning-based email classification system that automatically categorizes incoming emails into predefined classes, such as spam or ham. Specifically, the project aims to:

- Importing the data: - first huge amount data was imported on which model's will get trained on it.
- Analysis:- Performing the EDA(Exploratory Data Analysis) on imported data.
  - ➢ Checking the amount of Spam and Ham Emails
  - ➢ What type of words are used in both types of Emails
  - ➢ Which type of Email have high number of words, character etc.
  - ➢ Defining the main keywords for spam / ham mails.

- **Implement text preprocessing** techniques (e.g., tokenization, stopword removal, stemming) to clean and prepare email data for classification.
- **Develop a model** that can accurately classify emails based on their content using machine learning algorithms like Naive Bayes, CountVectorizer, or other suitable classifiers.
- **Creating the UI Interface:-**Implementing the created trained model in UI format which will be user friendly .using the streamlit library for user friendly interface.

1.4 Scope of the Project:

- **Classification Algorithm**: The project will evaluate and apply machine learning algorithms, with a focus on supervised learning techniques.
- **Email Categories**: The classifier will focus on common categories such as spamor ham
- **Real-time Classification**: The project may not fully address real-time email classification in large-scale production environments, where latency and processing speed are critical.
- **Complex Email Structures**: Handling emails with intricate structures, such as embedded HTML, attachments, or multilingual content, may introduce additional complexities

CHAPTER 2

Literature Survey

2.1 Review of Relevant Literature or Previous Work

Spam email classification has been an active research area for over two decades. Early approaches largely relied on rule-based methods, where heuristics such as keyword matching or blacklisting were used to filter spam. However, these methods had limited accuracy and struggled with evolving spam tactics. With the advent of machine learning, several algorithms were introduced to improve spam detection.

1. **Naive Bayes Classifier**: One of the earliest and most popular machine learning models used for spam classification. It works well for text classification tasks due to its simplicity and efficiency.
2. **Support Vector Machines (SVM)**: This technique has been explored in spam filtering due to its ability to handle high-dimensional data and offer good generalization. Studies such as that by Padhy et al. (2013) have demonstrated SVM's potential in spam classification, although it often requires careful tuning of hyperparameters.
3. **Deep Learning Approaches**: Recently, deep learning techniques like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have also been explored for spam classification. These methods, while highly accurate, require large amounts of labeled data and computational power, making them less practical for smaller-scale applications (Almeida et al., 2011).
4. **Feature Engineering Techniques**: Traditional feature extraction methods like bag-of-words, TF-IDF, and word embeddings (e.g., Word2Vec) have been extensively used to convert text into numerical representations suitable for ML models. CountVectorizer, which builds a "bag-of-words" representation, has been widely adopted in spam detection due to its simplicity and efficiency (Rennie et al., 2003).

2.2 Existing Models, Techniques, or Methodologies

Several models and techniques have been employed to address spam classification:

- **Naive Bayes Classifier**: This probabilistic model has been extensively used in spam filtering due to its simplicity and effectiveness. It works by calculating the likelihood of each email being spam or not based on the probability distribution of the words in the message.
- **Support Vector Machines (SVM)**: This algorithm aims to find a hyperplane that best separates the spam and non-spam emails. It's known for its accuracy in binary classification tasks, although it often requires significant computational resources and feature selection.
- **Random Forest and Decision Trees**: These ensemble models have also been used, offering advantages in terms of interpretability and handling non-linear relationships between features.
- **CountVectorizer and TF-IDF**: Text preprocessing methods like CountVectorizer have been pivotal in transforming raw email text into structured input for machine learning models. These techniques break down the text into tokens (words or phrases) and represent them as vectors, which can then be fed into models for classification.
- **Deep Learning**: Some recent research explores the use of neural networks for spam classification, such as the use of LSTM (Long Short-Term Memory) networks and CNNs. These models tend to outperform traditional models on large datasets but require significant computational resources and training time.

2.3 Gaps and Limitations in Existing Solutions

Despite the advancements in spam classification, several gaps and limitations remain in existing solutions:

1. **Generalization to New Spam Patterns**: Many spam filters are built using predefined features and patterns. As spammers continuously evolve their tactics, static models may fail to detect novel forms of spam, leading to a high rate of false negatives.

2. **Resource Consumption in Advanced Models**: While deep learning models offer high accuracy, they require extensive computational resources and large labeled datasets. These models may not be practical for small- to medium-scale applications, where computational power is limited.
3. **Feature Engineering Challenges**: Traditional feature engineering methods like CountVectorizer are relatively simple but may fail to capture deeper semantic meaning or context within the text. Moreover, they do not account for more sophisticated features like contextual relationships or spam characteristics that might evolve over time.
4. **False Positives in Critical Email Systems**: A major concern in spam filtering is reducing false positives—misclassifying legitimate emails as spam. This is critical for systems that handle important communications, such as corporate email systems or personal accounts.

How the Project Will Address These Gaps:

This project focuses on enhancing spam detection by combining **Naive Bayes classification** with **CountVectorizer** for feature extraction. While Naive Bayes offers a probabilistic approach that is both simple and effective, CountVectorizer will ensure that the textual features are captured in a manner suitable for machine learning models.

Key improvements and contributions of this project include:

1. **Focused on Real-World Applicability**: Unlike deep learning methods, which may not be feasible in resource-constrained environments, this project emphasizes lightweight, interpretable models like Naive Bayes, ensuring that the solution is scalable and efficient for real-time spam detection.
2. **Handling Novel Spam Variants**: By utilizing a continuously updated dataset (such as the Enron email dataset), the model will be evaluated for its adaptability to detect new spam tactics. The probabilistic nature of Naive Bayes allows it to generalize better when exposed to new patterns compared to rigid rule-based or even some machine learning methods.

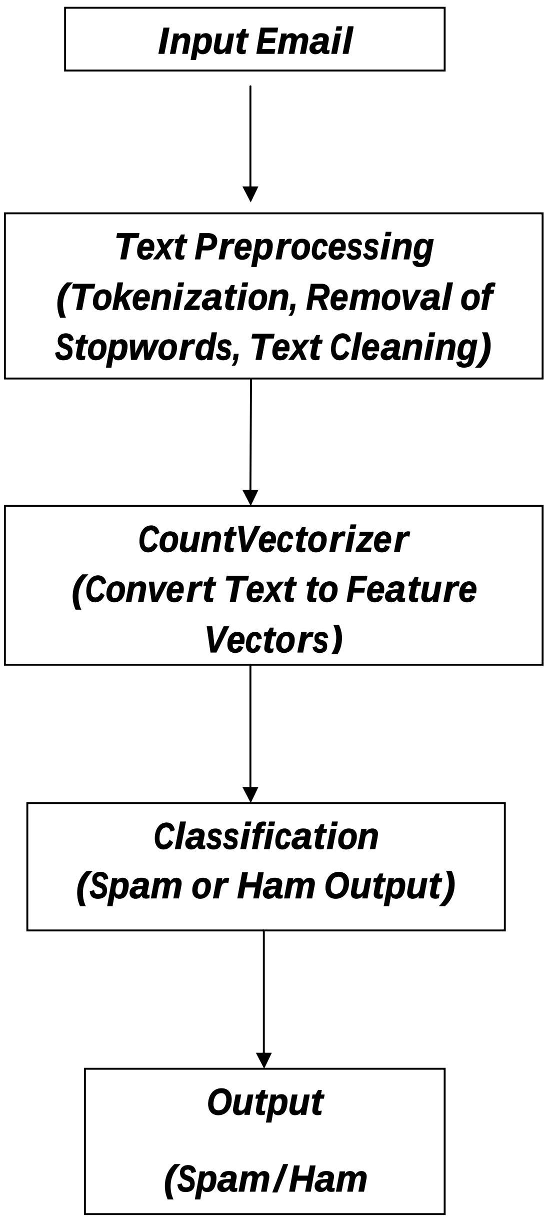
3. **Balanced Performance Metrics**: This project emphasizes achieving an optimal trade-off between precision, recall, and accuracy to minimize both false positives and false negatives, a critical challenge in real-world applications.

In conclusion, the project aims to improve spam email classification through a well-established, resource-efficient methodology while addressing key limitations in existing solutions.

CHAPTER 3

Proposed Methodology

3.1 System Design


Input Email
Text Preprocessing (Tokenization, Removal of Stopwords, Text Cleaning)
CountVectorizer (Convert Text to Feature Vectors)
Classification (Spam or Ham Output)
Output (Spam/Ham

3.1.1 Diagram Description:

1. **Input Email:**
    - The email is the raw text input.
2. **Text Preprocessing:**
    - **Tokenization:** Split the email text into words or tokens.
    - **Stopwords Removal:** Filter out common, unimportant words (e.g., "and", "the").
    - **Text Cleaning:** Remove special characters, lowercasing, and other unnecessary formatting.
3. **CountVectorizer:**
    - Converts the cleaned text into a numerical feature vector, representing word counts in a bag-of-words model.
4. **Naive Baye's Model:**
    - The CountVectorized data is fed into the Naive Bayes classifier, which is trained on a dataset of labeled emails (spam/ham).
5. **Classification:**
    - The model classifies the email as either **Spam** or **Ham** based on the learned features.
6. **Output:**
    - The final classification (spam or ham) is presented as the result.

The flow can be shown with arrows moving from one component to the next. This structure highlights the key steps in preprocessing, model training, and classification.

3.2 Requirement Specification

3.2.1 Hardware Requirements:

- Processing unit:- Dual-core processor or high for smooth experiences.
- RAM:-Minimum 4GB to handle real-time processing efficiently.
- Operating System:- Windows/Linux/MacOS.

3.2.2 Software Requirements:

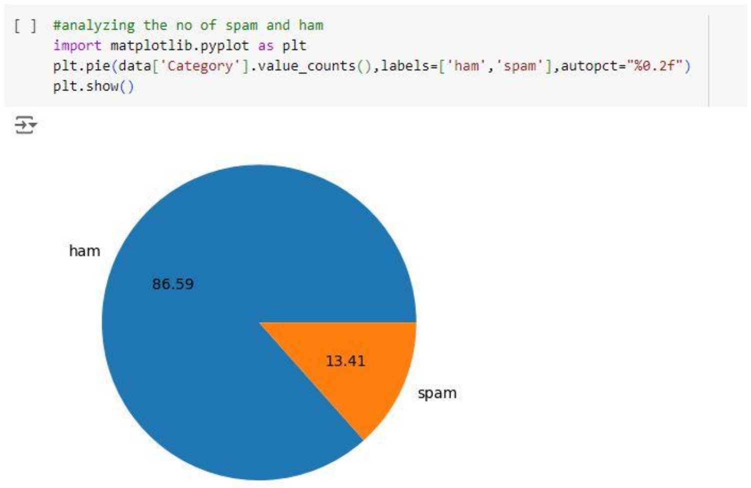
- Jupyter Notebook (Analyzing& Model Creating)
  For Anaconda should be installed , to create an specific environment for Jupyter Notebook project.
- VS Code Editor (Importing the created model & designing UI)
  Using Streamlit Module for creating the UI design for project which will be simple and user friendly.
- Pandas (Importing the dataset)
  Pandas will help to importing the data, performing the datasets operation.
- Matplotlib&Seaborn (Data Visualization)
  It will create an visual representation of analyse data for spam/ham datasets.
- Sklearn& NLTK (Machine Learning Models)
  Built-in models like Naïve Bayes &CountVectorizer will be used to train the models. 80% data will be used for training and remaining 20% will be used for testing. Which will be telling how model is responding to the given datasets.

CHAPTER 4

Implementation andResult

4.1Snap Shots of Result:

1.

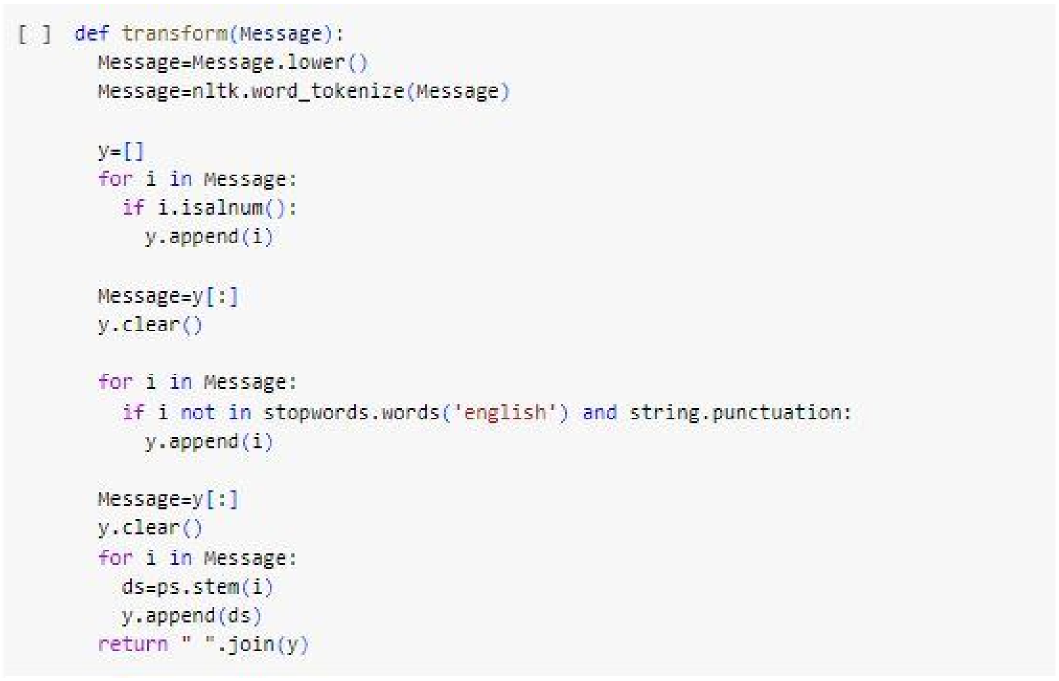


```
[ ] #analyzing the no of spam and ham
    import matplotlib.pyplot as plt
    plt.pie(data['Category'].value_counts(),labels=['ham','spam'],autopct="%0.2f")
    plt.show()
```

Fig 1:- Snapshot of Piechart representing amount of Spam/Ham data.

- In above screenshot,matplolib module was imported to compare how much is the spam /ham email in the given datasets.
- As there are only two columns in the datasets i.e Category and Message.
- Where Message column represents the Email Messages and Category columns represents that the corresponding message is spam or ham email.
- Pie chart of Matplotlib helps to understand that there are total 86.59 % of ham messages and 13.41 % of spam messages.
- There is no Null value in given datasets.

2.



```
[ ]  def transform(Message):
       Message=Message.lower()
       Message=nltk.word_tokenize(Message)

       y=[]
       for i in Message:
         if i.isalnum():
           y.append(i)

       Message=y[:]
       y.clear()

       for i in Message:
         if i not in stopwords.words('english') and string.punctuation:
           y.append(i)

       Message=y[:]
       y.clear()
       for i in Message:
         ds=ps.stem(i)
         y.append(ds)
       return " ".join(y)
```

Fig 2:- Function to transform and clean the email message

- Before giving datasets to the models directly, it should be in Machine readable format .
- There are total 5 phases are applied on the Messages of Email, which will be easier for model to understand and track the pattern in it.
  - ➢ Lowercase
  - ➢ Tokenizing
  - ➢ Removing Special Characters
  - ➢ Removing Stopwords
  - ➢ Stemming the given message
- Lowercase:-Converting the all letter of the message into the lowercase as the meaning of the message will remain same and confusion for model which will take more time to process the Capitalize form of the data.

  e.g:- Hi, How are you ?

  O/P:- hi how are you?

- Tokenizing:- Using nltk (Natural Language Library) to convert all the words of the paragraph into an list format.
  e.g:- hi, there how are you ?
  O/P:- [‘hi’,’there’,’how’,’are’,’you]
- Special Characters:-Using for loop to iterate every word of the message and check whether it is an alphanumeric or not .
  e.g:- hi, there how are @@@ you ?
  O/P:- [‘hi’,’there’,’how’,’are’,’you ?]

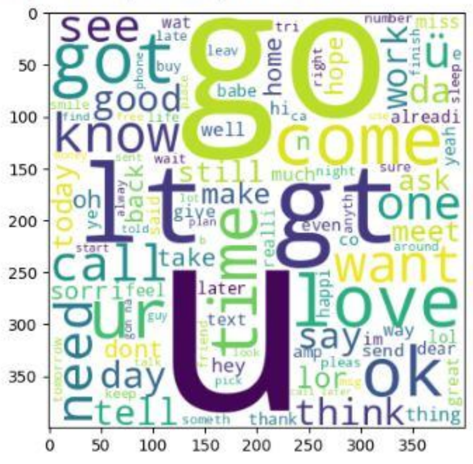
- Removing Stopwords**:-**Importing nltk corpus and string library to remove stopwords like I , is , was etc.
  e.g:- hi, there how are you ?
  O/P:- [’there’,’how’, ’you’]

- Stemming:-This nltk method shortens the unnecessary long words that contains ing, ed at the end.
  e.g:- running, working, robbed
  O/P:- run , work , robb

3.

```
ham_wc=wc.generate(data[data['Category']==0]['transformed_text'].str.cat(sep="
plt.imshow(ham_wc)
```

```
<matplotlib.image.AxesImage at 0x7cd00655b670>
```



```
[ ] spam_wc=wc.generate(data[data['Category']==1]['transformed_text'].str.cat(se
```

```
plt.imshow(spam_wc)
```

```
<matplotlib.image.AxesImage at 0x7cd005a526b0>
```

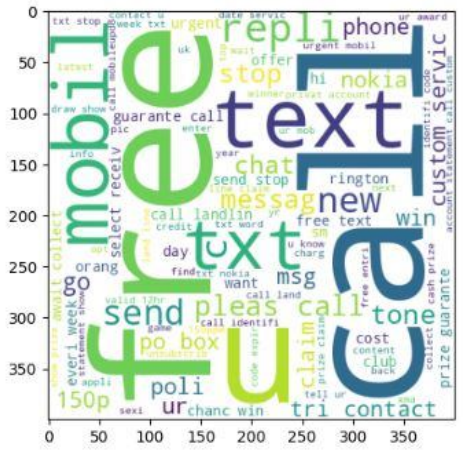
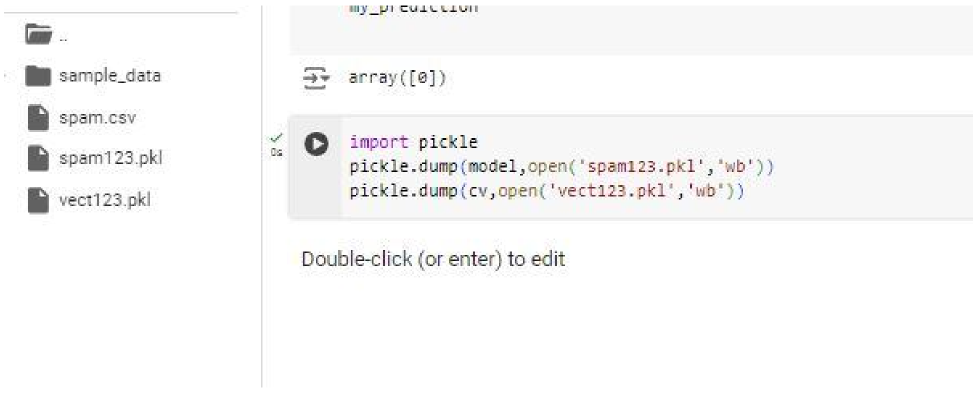


Fig 3:- Snapshot of Diagram representing most frequent used words

- Here spam mails and ham mails are separated on condition.
- Where it is used to identify which words are more frequent in most of the messages
- Wordcloud module is used to show the most frequent words .
- For spam mail, free , text, call , phone , customer service are the words which are most frequent
- For ham mail, I , how , get , out , love are the most frequently appearing words.

4.



```
my_prediction
```

```
array([0])
```

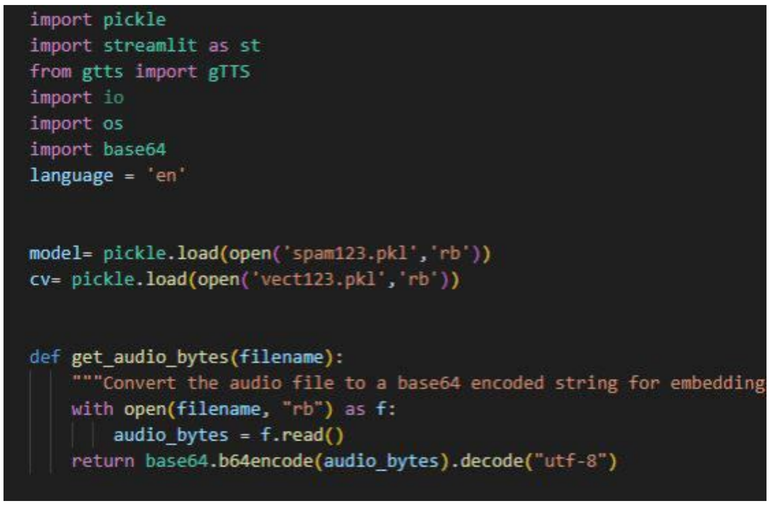
```
import pickle
pickle.dump(model,open('spam123.pkl','wb'))
pickle.dump(cv,open('vect123.pkl','wb'))
```

Double-click (or enter) to edit

Fig 4:- Pickle files were created ,carrying object and trained modals

- Pickle module is used in this section which is store trained model of machine learning.
- Here two machine learning models are stored as pickle file i.ecv and model.
- Which can be used for future references.
- Dump function is used to load the model object and save it as filename.pkl.

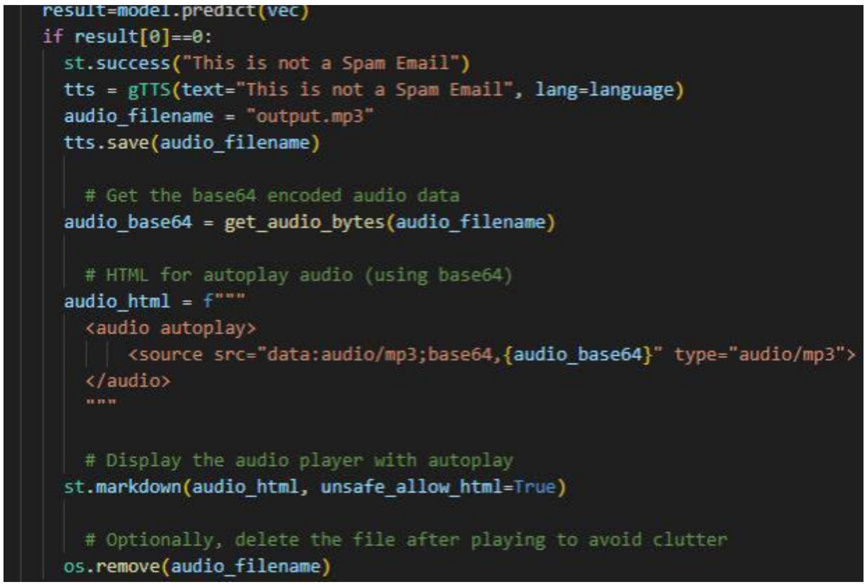
5.



```python
import pickle
import streamlit as st
from gtts import gTTS
import io
import os
import base64
language = 'en'


model= pickle.load(open('spam123.pkl','rb'))
cv= pickle.load(open('vect123.pkl','rb'))


def get_audio_bytes(filename):
    """Convert the audio file to a base64 encoded string for embedding
    with open(filename, "rb") as f:
        audio_bytes = f.read()
    return base64.b64encode(audio_bytes).decode("utf-8")
```

Fig 5:- function to save the text as speech

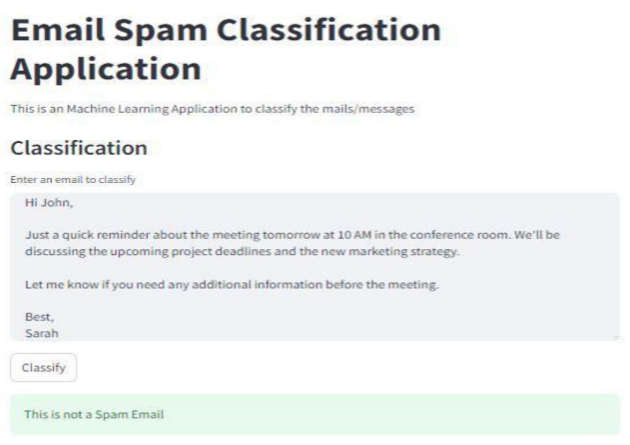
- gTTs (Google Text to Speech) module is used to convert the text into speech.
- Created anget_audio_bytes function which will be converting the audio file embedded form using base64 module.
- Language was set as English by default on gTTs.
- gTTS object was created at both part of if and else part.
- In gTTS object default text and language was passed in it.



```python
result=model.predict(vec)
if result[0]==0:
    st.success("This is not a Spam Email")
    tts = gTTS(text="This is not a Spam Email", lang=language)
    audio_filename = "output.mp3"
    tts.save(audio_filename)

    # Get the base64 encoded audio data
    audio_base64 = get_audio_bytes(audio_filename)

    # HTML for autoplay audio (using base64)
    audio_html = f"""
    <audio autoplay>
        <source src="data:audio/mp3;base64,{audio_base64}" type="audio/mp3">
    </audio>
    """

    # Display the audio player with autoplay
    st.markdown(audio_html, unsafe_allow_html=True)

    # Optionally, delete the file after playing to avoid clutter
    os.remove(audio_filename)
```

Fig 6:- using html to play audio automatically

- Here I imported the HTML audio autoplay tag which will automatically play the audio , when the output will generated as there is not built-in autoplay option in streamlit.



Email Spam Classification Application

This is an Machine Learning Application to classify the mails/messages

Classification

Enter an email to classify

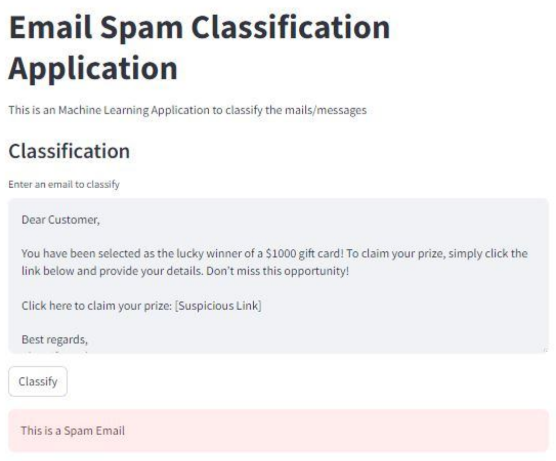
Hi John,

Just a quick reminder about the meeting tomorrow at 10 AM in the conference room. We'll be discussing the upcoming project deadlines and the new marketing strategy.

Let me know if you need any additional information before the meeting.

Best,
Sarah

This is not a Spam Email



Email Spam Classification Application

This is an Machine Learning Application to classify the mails/messages

Classification

Enter an email to classify

Dear Customer,

You have been selected as the lucky winner of a $1000 gift card! To claim your prize, simply click the link below and provide your details. Don't miss this opportunity!

Click here to claim your prize: [Suspicious Link]

Best regards,

This is a Spam Email

Fig 7:- Snapshot of final o/p detecting spam/ham emails

- Here UI was created which will be used friendly and detect whether the given message is Spam or Ham.
- By the detecting the mail type it will be displaying in red color for Spam and Green Color for Ham.
- Classify button was created which will run the model .
- After detecting the mail type an voice will also get erected showcasing the same message automatically.

4.2 GitHub Link for Code:

https://github.com/Ashokkpal/Spam-Email-Classification-using-NLP-and-Machine-Learning

CHAPTER 5

Discussion and Conclusion

5.1 Future Work:

In the future, one of the key enhancements planned for the email classification system is the integration of a feedback mechanism within the user interface. This feedback form will allow users to interact directly with the system, providing valuable input regarding the accuracy and relevance of the classification results. The feedback loop will be instrumental in continuously improving the model and ensuring that it adapts to the evolving nature of email communication.

The feedback form will be designed to be simple, intuitive, and easy to use, enabling users to rate the classification of each email. For example, users could be asked to verify if an email was correctly classified as "spam," "important," "promotional," or other predefined categories. If a misclassification is detected, users can flag it, indicating whether the email belongs to a different category or if it was incorrectly marked as spam. This form of user interaction would provide real-time corrections to the system and allow for immediate adjustments based on user preferences.

The integration of such feedback would be beneficial in several key ways:

1. **Improving Model Accuracy**: By collecting real-time feedback from users, the model can be constantly updated and fine-tuned based on the mistakes and misclassifications it makes. For example, if a certain type of email, such as newsletters or transactional emails, is frequently misclassified, the feedback would help identify these errors and correct the model's behavior over time. This will lead to an overall improvement in classification accuracy, reducing false positives (e.g., important emails marked as spam) and false negatives (e.g., spam emails getting through the filter).

2 **Personalization of Email Categorization**: Every user has different preferences when it comes to email organization. Some may consider certain promotional emails as important, while others may prefer to ignore them. The feedback mechanism will allow the system to learn individual user preferences, thereby personalizing email categorization. For example, if a user consistently reclassifies promotional emails as "important," the system will adjust to accommodate that user's preferences, improving the system's relevance and user experience.

3 **Continuous Learning**: The feedback form provides an opportunity for continuous learning, ensuring that the model doesn't become static or outdated over time. As email communication patterns change, new types of emails emerge (e.g., advanced phishing attempts or new promotional strategies), the classifier can adapt more effectively. Feedback from users will serve as real-world examples that help train the model to recognize and categorize new email types, thus increasing the system's robustness and accuracy in dynamic environments.

4 **Data for Model Refinement**: Feedback data can be used to identify areas where the model is struggling, providing insights into which specific types of emails it has difficulty classifying. For example, if users repeatedly flag legitimate emails as spam or incorrectly categorize urgent emails, the feedback can be analyzed to understand the root cause of these errors. This will allow developers to tweak the underlying algorithms or modify the feature extraction process to improve performance in these challenging areas.

5 **User Empowerment and Satisfaction**: Giving users the ability to provide direct feedback on the system's performance not only improves the classifier but also enhances user satisfaction. It allows users to feel that they have control over their email experience and can help improve the system to suit their needs. The feedback process,

being an integral part of the system, also increases user engagement and trust in the accuracy and reliability of the classifier.

6 **Long-term Model Optimization**: Over time, the feedback system will gather a substantial amount of user input, which can be used to retrain the model periodically. This data will allow for the application of more advanced machine learning techniques, such as active learning, where the system identifies uncertain predictions and requests feedback specifically for those cases. This iterative process ensures that the model evolves and becomes more precise over time.

7 **Handling New Email Types and Phishing Detection**: As spam, phishing, and promotional emails become more sophisticated, the feedback loop will be particularly useful in identifying previously unseen types of malicious or unwanted content. Users can flag suspicious emails, which will enable the system to identify and categorize new forms of phishing attacks or spam patterns, making it more resilient to emerging threats.

7.1 Conclusion:

In conclusion, this spam classifier project has made a meaningful contribution to improving the detection and prevention of unwanted, malicious, or unsolicited messages in digital communication systems. By developing a machine learning model capable of distinguishing between spam and non-spam content, the project demonstrates the significant potential for automating and enhancing email filtering and messaging systems.

The classifier, trained on diverse datasets, has shown promising accuracy and efficiency in identifying patterns indicative of spam, including common keywords, message structure, and other textual features. The use of various machine learning algorithms (e.g., Naive Bayes, Support Vector Machines, or Deep Learning) has further demonstrated the value of applying advanced techniques to real-world problems, improving upon traditional heuristic-based methods.

This project's contributions extend beyond the development of a functioning spam filter. It provides insights into the challenges of handling large-scale text data, the importance of feature engineering, and the critical need for continuous model updates as spammers evolve their tactics. Additionally, it underscores the growing role of artificial intelligence in enhancing digital security and user experience.

By automating the classification of spam messages, the project not only improves user productivity and safety but also lays the groundwork for future improvements in areas such as real-time detection, adaptation to emerging spam techniques, and cross-platform filtering.

Overall, this project successfully showcases the effectiveness of machine learning in the fight against spam, providing a reliable tool that can be applied across various communication platforms. It also opens the door to future work that can further refine the classifier's accuracy, adapt to new spam trends, and integrate with larger systems for even broader applications in cybersecurity

REFERENCES

[1]. Campus X an YouTube channel helps for building this project , and most of the analyzing was inspired from them. https://youtu.be/YncZ0WwxyzU?si=0DxlJAD47_dqd4oF

[2]. Naïve Bayes algorithms and its models from WS Cube Tech , Machine Learning model and model testing algorithms . https://youtu.be/LvC68w9JS4Y?si=mLtPlv2QbUiZcElH

[3]. Knowledge Doctor for adding the idea of voice in the project https://youtu.be/rHesaMUqTjE?si=jmkzbxTY_Rjf9JGO