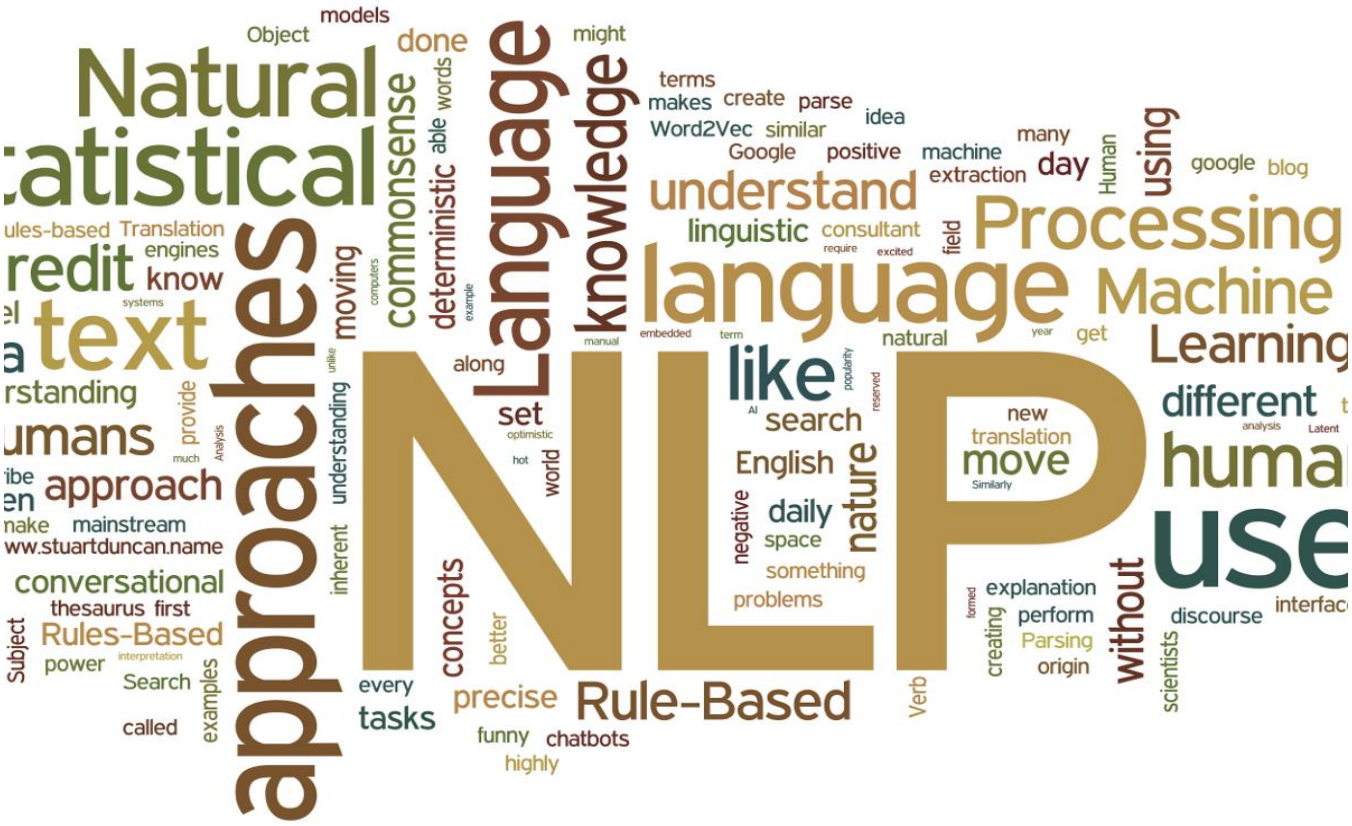
Helwan University

Faculty of Engineering

Communication and Information Department

Course Title: Machine Learning



Spam Mail Prediction Using Natural Language Processing

Submitted To: Asst. Prof. Ahmed E. youssef

Submitted By:

1- Esraa Matarawy Abdelmoniem ID: 222150084

2- Roqaya Hassan Hamed ID: 222150060

3-Tasnim Khaled Mohamed ID: 222150099

TABLE OF CONTENTS

1.INTRODUCTION

1.1 Overview of Spam Email Detection and Its Importance

In today's digital era, email communication remains one of the most essential tools for both personal and professional purposes. However, the widespread use of email has also given rise to significant cybersecurity threats, with spam emails being one of the most persistent and dangerous issues. Spam emails often contain malicious attachments, phishing links, or fraudulent content designed to steal sensitive information, deploy malware, or disrupt organizational workflows. The consequences of falling victim to such threats can range from financial losses to severe breaches of privacy and security.

Spam email detection plays a critical role in the field of cybersecurity by safeguarding individuals and organizations against these malicious threats. With billions of emails exchanged daily, automated spam detection systems must be both efficient and accurate to handle the vast amount of data in real-time. This is where Natural Language Processing (NLP) comes into play. NLP enables machines to analyze, understand, and extract meaningful insights from the textual content of emails. By leveraging NLP techniques and Machine Learning (ML) models, spam email detection systems can accurately classify emails based on their content, presence of links, and attachments. This project aims to contribute to this critical area by building a robust spam detection system using multiple ML models, with a focus on Logistic Regression for optimal results.

1.2 Problem Statement and Project Objectives

The primary problem addressed in this project is the accurate detection and classification of spam emails to prevent malicious activities, phishing attacks, and data breaches. Traditional spam detection techniques often rely on rule-based systems that are limited in handling dynamic and evolving spam patterns. These limitations necessitate advanced machine learning models capable of learning complex patterns from email data.

This project utilizes multiple machine learning models including Logistic Regression (LR), Random Forest (RF), and Naive Bayes (NB) to classify emails as spam or ham (legitimate). Among these, Logistic Regression paired with NLP techniques demonstrated the highest effectiveness in accurately detecting spam emails based on:

- The textual content of the email body.
- The presence of URLs and their potential risks.
- The existence of attachments (e.g., .pdf, .zip, .exe files).

The key objectives of this project are:

- To improve the accuracy of spam mail classification.
- To reduce false positives (legitimate emails misclassified as spam).
- To analyze various email features such as body content, links, and attachments for improved classification performance.
- To provide a scalable and adaptable spam detection framework suitable for real-world applications.

1.3 Research Questions and Hypothesis

This project seeks to address the following key research questions:

- “Can machine learning models effectively classify spam emails with high accuracy using NLP techniques?”

The hypothesis of this research is:

- Among the tested models (Logistic Regression, Random Forest, and Naive Bayes), Logistic Regression combined with NLP techniques will yield the highest accuracy and most reliable results due to its ability to handle textual data efficiently.

The experimental results confirmed that Logistic Regression outperformed the other models, achieving the best classification accuracy and performance metrics.

2.LITERATURE REVIEW

2.1 Definition and Background of Spam Email Detection

Spam email detection refers to the process of identifying and filtering unsolicited and often malicious emails from legitimate messages. Spam emails, also known as junk emails, typically include advertisements, phishing links, malware attachments, or fraudulent schemes. Over the years, spam emails have become one of the most significant cybersecurity challenges for individuals and organizations worldwide.

Historically, spam emails emerged in the early days of email communication, with the first recorded spam message sent in 1978 over ARPANET. As email usage grew, so did the volume and sophistication of spam. By the early 2000s, spam emails accounted for over 60% of global email traffic, necessitating robust detection mechanisms.

Spam emails often exhibit certain patterns, such as suspicious links, keyword triggers (e.g., "free money," "urgent action required"), and malicious attachments (e.g., .exe, .zip, .pdf files). However, spammers continuously evolve their tactics to bypass detection systems, including using obfuscated URLs, image-based spam, and random text patterns.

The challenges in spam detection include:

- **Volume and Scale:** Millions of spam emails are sent daily.
- **Evasion Techniques:** Use of random text, obfuscated URLs, or legitimate-looking sender addresses.
- **False Positives:** Legitimate emails occasionally flagged as spam.
- **Adaptability:** Spammers quickly adapt to new detection techniques.

Addressing these challenges requires sophisticated detection systems, often leveraging machine learning (ML) and Natural Language Processing (NLP) to analyze email content dynamically and accurately.

2.2 Traditional Approaches in Email Filtering

Before the advent of advanced machine learning models, spam detection relied on rule-based filtering techniques and heuristic-based filtering.

- **Rule-Based Filtering:** These systems operate based on pre-defined rules, such as identifying specific keywords in email subject lines or bodies (e.g., "win money," "limited offer"). Blacklists and whitelists were also commonly used, where emails from flagged domains or addresses were automatically classified as spam. Tools like **SpamAssassin** were early pioneers in this space.
    - **Limitations:** Rule-based systems struggled with evolving spam patterns and often resulted in high false positives when legitimate emails matched spam-like keywords.
- **Heuristic-Based Filtering:** These methods assessed emails based on a scoring mechanism. For example, an email containing multiple links or certain attachment types would receive a higher spam score.
    - **Limitations:** Heuristic systems lacked adaptability and failed against cleverly crafted spam messages that appeared legitimate.

Limitations of Traditional Approaches:

- Lack of scalability in handling high email volumes.
- Poor adaptability to changing spam tactics.
- High false-positive and false-negative rates.
- Inability to analyze nuanced text patterns effectively.

The shortcomings of these traditional approaches paved the way for machine learning-based spam detection models, which could learn and adapt to new spam patterns dynamically.

2.3 Evolution of Machine Learning Models in Spam Detection

Machine learning introduced a paradigm shift in spam detection, offering models that could learn from historical data and improve accuracy over time.

Early ML models for spam detection included:

- **Naive Bayes (NB):** Known for its simplicity and efficiency, Naive Bayes classifiers analyzed word frequency to predict whether an email was spam.
- **Support Vector Machines (SVM):** These were effective in finding decision boundaries for separating spam from non-spam emails.

In recent years, more advanced machine learning techniques have gained prominence:

- **Logistic Regression (LR):** A statistical model that predicts probabilities based on weighted features. It performs exceptionally well in binary classification problems like spam detection.
- **Random Forest (RF):** An ensemble model that uses multiple decision trees to improve accuracy and reduce overfitting.
- **Natural Language Processing (NLP):** NLP techniques enable models to understand the semantic meaning of email content, analyzing sentence structures, keywords, and patterns.

In this project, Logistic Regression, Random Forest, and Naive Bayes models were utilized for spam detection. While all three models performed adequately, Logistic Regression demonstrated the highest accuracy and reliability, making it the most effective choice for this task.

NLP's Role in Spam Detection:

- The context and meaning behind email content.
- Patterns in language use (e.g., repetitive spam-like phrases).

Combining NLP with machine learning significantly enhances the ability to detect subtle spam signals that rule-based filters often miss.

2.4 Review of Key Studies and Findings in Spam Email Prediction

Research in spam detection has highlighted the effectiveness of various machine learning models and techniques. Below are some key findings from relevant studies:

- **Naive Bayes Classifier:** Known for its simplicity and effectiveness in text classification. Studies show accuracy rates of around **85%–90%**, but the model struggles with complex text structures.
- **Random Forest Classifier:** Provides robust performance with lower false-positive rates but can sometimes be computationally expensive.
- **Logistic Regression Classifier:** Demonstrated higher accuracy and faster computation compared to Naive Bayes and Random Forest in several benchmark datasets.

In most studies, Logistic Regression emerged as the superior choice for text-based spam email detection, particularly when combined with TF-IDF (Term Frequency-Inverse Document Frequency) for text vectorization.

Evaluation Metrics in Literature:
Common metrics used in spam detection studies include:

- **Accuracy:** Proportion of correct predictions.
- **Precision:** Proportion of predicted spam emails that were actually spam.
- **Recall:** Proportion of actual spam emails correctly detected.
- **F1-Score:** Harmonic mean of precision and recall.

These metrics provide a comprehensive view of a model's performance in handling spam detection tasks.

3.DATASET OVERVIEW

3.1 DESCRIPTION OF THE DATASET USED

The dataset used in this project was sourced from **Kaggle**, a widely recognized platform for sharing datasets and conducting data-driven projects.

- **Dataset Size:** The dataset initially contained **5,728 emails**, but after identifying and removing **33 duplicate entries**, the final dataset consisted of **5,695 unique records**.
- **Class Distribution:** The dataset includes two classes:
  - **0 (Ham Emails):** 4,360 emails (non-spam)

    - **1 (Spam Emails):** 1,368 emails (Spam)
      This distribution highlights a significant **class imbalance**, with non-spam emails making up the majority of the dataset.
- **Shuffling and Stratification:** The original dataset was ordered, with spam and ham emails grouped sequentially. To ensure a fair representation of both classes during the training and testing phases, the data was **shuffled**. Additionally, during the **train-test split**, stratification was applied to maintain the original class distribution across both sets.
- **Time Frame:** The dataset is relatively recent, collected approximately **one year ago**, ensuring relevance to current email trends and phishing techniques.
- **Structure of the Dataset:** Each email entry contains:
    - **Body:** The main content of the email.
    - **Label:** A binary indicator where **1** represents spam and **0** represents ham.

This data set provides a solid foundation for building a robust machine-learning model capable of detecting spam emails effectively.

3.2 DATA PREPROCESSING TECHNIQUES APPLIED

Effective preprocessing is critical in preparing the dataset for machine learning models. Several techniques were applied to clean and prepare the text data and extract meaningful features:

1. **Handling Missing Data:**
    - Missing email bodies were replaced with an **empty string** ('') to ensure consistency in processing.
2. **Text Preprocessing:**
    - **Counting URLs:** Emails containing URLs have a higher probability of being spam. Therefore, URLs were **counted** and used as a numeric feature.
    - **Attachment Detection:** Keywords like "attachment", "attached", or "file" were searched to identify emails containing attachments. This binary feature (1 for attachment, 0 for no attachment) was added to the dataset.
    - **Lowercasing:** All text was converted to **lowercase** to maintain uniformity.
    - **Removing URLs:** URLs were removed from the email body after being counted.
    - **Removing Non-Alphanumeric Characters:** Special characters and symbols were removed to simplify the text.
3. **Feature Engineering:**
    - Numeric features were added to the dataset:
        - **Number of URLs** in the email body.
        - **Attachment Indicator:** Binary flag indicating the presence of attachments.

These preprocessing steps ensured that the dataset was clean, standardized, and enriched with informative features for model training.

3.3 FEATURE EXTRACTION AND ENGINEERING FOR EMAIL CLASSIFICATION

Feature extraction is a vital step in transforming the raw email text into a format that machine learning algorithms can interpret.

1. **TF-IDF Vectorization:**
    - The **Term Frequency-Inverse Document Frequency (TF-IDF)** technique was applied to convert email text into numerical features.
    - Key parameters used:
        - **Stop Words:** Removed English stopwords to reduce noise.
        - **Max Document Frequency (max_df):** Set to **0.7** to exclude overly common terms.
        - **Min Document Frequency (min_df):** Set to **5** to include only meaningful terms.
        - **Max Features:** Limited to **3,000** to control dimensionality.
    - TF-IDF ensures that common words contribute less weight, while rare but important words have a higher influence on the classification model.
2. **Combining Numeric Features:**
    - In addition to text features extracted via TF-IDF, the following numeric features were included:
        - Number of URLs in the email body.
        - Attachment Indicator (1/0) based on attachment presence.
    - These numeric features were horizontally stacked with the TF-IDF matrix using scipy.sparse.hstack to form the final feature matrix.
3. **Train-Test Split:**
    - The dataset was split into 80% training and 20% testing sets.
    - Stratified sampling ensured balanced representation of spam and ham emails in both subsets.
4. **Final Feature Matrix:**
    - The final matrix combined text-based and numeric features, providing a comprehensive representation of each email for the machine-learning models.

By carefully preprocessing, extracting, and engineering features, the dataset was transformed into a rich, structured format suitable for effective spam email detection using machine learning models.

4.METHODOLOGY AND MODEL EVALUATION

4.1 Explanation of the Machine Learning Models Used

We employed three commonly used machine learning models for text classification: **Logistic Regression (LR)**, **Random Forest (RF)**, and **Naive Bayes (NB)**. These models were chosen based on their efficiency and effectiveness in handling text data, particularly for tasks like spam email detection. Text-based classification tasks typically involve identifying relevant patterns in the features, and these models are well-known for their performance in such contexts.

- **Logistic Regression (LR)**: Logistic Regression is a linear classifier commonly used for binary classification tasks. It calculates probabilities using a logistic function and is particularly efficient when the dataset is linearly separable or when the feature set is large. For this project, we used Logistic Regression with class_weight='balanced' to address class imbalance (as spam emails are fewer in number compared to ham emails), and max_iter=1000 to ensure convergence during training.
- **Random Forest (RF)**: Random Forest is an ensemble method that builds multiple decision trees and combines their outputs to improve performance and reduce overfitting. It works well with high-dimensional datasets like text data and can automatically handle interactions between features. Like LR, we also applied class_weight='balanced' in the Random Forest model to ensure fair treatment of both spam and ham classes.
- **Naive Bayes (NB)**: Naive Bayes is a probabilistic classifier based on Bayes' theorem, assuming feature independence given the class. This model is particularly effective in text classification due to its simplicity and ability to handle high-dimensional data efficiently. In this project, we used the Multinomial Naive Bayes variant, which is ideal for text classification problems involving word counts.

To evaluate the performance of these models, we used the classification report and accuracy score, which provide insights into the precision, recall, and F1-score for each class (spam and ham).

4.2 Preprocessing and Feature Extraction Details

The preprocessing pipeline plays a crucial role in preparing the data for machine learning models, especially for text classification. Below are the key preprocessing techniques and feature extraction methods applied to the dataset:

- **Text Preprocessing**:
  - **Handling Missing Data**: Any missing email content was filled with an empty string to maintain dataset consistency.

    - **URL Count**: A key feature for spam detection is the presence of URLs in emails, which are often indicative of spam. A custom function was used to count the number of URLs in each email body.
    - **Attachment Detection**: Keywords like 'attachment', 'attached', and 'file' were searched for to detect whether an email contained an attachment.
    - **Lowercasing and Special Characters Removal**: To standardize the text, all emails were converted to lowercase, and non-alphanumeric characters were removed.
    - **Feature Extraction**: We used TF-IDF (Term Frequency-Inverse Document Frequency) to convert the text body into numerical features. This method reflects the importance of words in the context of the entire dataset, giving higher weight to words that are frequent in a particular email but rare across the dataset.
- **Handling Class Imbalance**: The dataset contained a significant class imbalance with 4360 ham emails and 1368 spam emails. To address this, the models were trained with class_weight='balanced' to give higher importance to the minority class (spam emails).

4.3 Description of the Logistic Regression Model

- The Logistic Regression model was implemented using sklearn's LogisticRegression function
- class_weight='balanced': This parameter ensures that the model takes into account the class imbalance during training by assigning higher weights to the minority class (spam).
- max_iter=1000: This ensures that the model has sufficient iterations to converge during training, particularly when the dataset is large or complex.

After training the model, we evaluated its performance on the test set using accuracy and the classification report, which showed that Logistic Regression achieved an accuracy of 98.95%. The model's precision, recall, and F1-score for both classes were also high, with an overall F1-score of 0.99 for ham emails and 0.98 for spam emails.

4.4 Random Forest Model for Spam Classification

The Random Forest model was implemented using sklearn's RandomForestClassifier The class_weight='balanced' parameter was used here as well to address the class imbalance, ensuring that the spam emails were given more importance during training. The model's performance was evaluated in a similar manner as Logistic Regression, achieving an accuracy of 98.95%, with a precision of 0.99 for ham emails and 0.98 for spam emails.

4.5 Naive Bayes Model and Its Application to Spam Detection

For the Naive Bayes model, we used Multinomial Naive Bayes, which is suitable for text classification tasks involving count-based features (like word frequency). The model was implemented using sklearn's MultinomialNB

The evaluation of this model yielded an accuracy of 98.43%, which, while slightly lower than Logistic Regression and Random Forest, still provided strong performance. The precision and recall values for both spam and ham emails were high, with precision = 0.98 and recall = 0.95 for spam emails, demonstrating that the Naive Bayes model is still a viable option for spam detection.

4.6 Summary of Model Results:

- **Logistic Regression**: Accuracy = 98.95%, High precision and recall for both classes.
- **Random Forest**: Accuracy = 98.95%, Similar performance to Logistic Regression, with slight differences in precision.
- **Naive Bayes**: Accuracy = 98.43%, Slightly lower performance but still competitive.

5.REAL-WORLD IMPACT AND APPLICATIONS

5.1 Importance of Spam Detection in Cybersecurity

Spam emails are a major vector for various types of cyberattacks, including phishing, malware distribution, and social engineering. In recent years, the volume of spam emails has surged, overwhelming both individual users and organizational email systems. Effective spam detection is therefore critical in safeguarding users from malicious threats that can compromise sensitive data, financial assets, and system integrity.

Spam detection serves as the first line of defense against several types of cyberattacks. By identifying and filtering out harmful emails, a spam detection system helps to minimize the chances of malicious content reaching an end-user. Spam detection by using machine learning models such as Logistic Regression, Random Forest, and Naive Bayes, significantly enhances this protective layer. With the ability to detect not only spam emails.

Additionally, with the rise of business email compromise (BEC), where attackers impersonate trusted entities to trick individuals into revealing confidential information, spam detection becomes even more essential. By using advanced text analysis techniques like NLP (Natural Language Processing) to evaluate email content, the proposed system can identify suspicious patterns or inconsistencies often associated with phishing attempts.

5.2 Potential Applications in Personal and Enterprise Email Systems

Spam detection is crucial in both personal and enterprise email systems, with significant benefits in both contexts. For personal email users, the primary focus is to protect against unsolicited and harmful messages, which can clutter inboxes and pose privacy risks. With spam emails, users are often subjected to unnecessary promotional messages, scams, and phishing attempts that can lead to financial losses, identity theft, and malware infections. By employing the spam detection model proposed in this project, individuals can safeguard their personal email accounts, ensuring a cleaner and more secure inbox with reduced risks from unwanted messages.

In the case of enterprise email systems, spam detection becomes even more critical due to the larger scale of operations and the increased likelihood of targeted attacks. For enterprises, email systems are often the primary communication channel, making them highly susceptible to sophisticated cyberattacks such as spear phishing and business email compromise (BEC). The proposed spam detection system, capable of analyzing email content in real-time, offers enterprises the ability to preemptively block or quarantine suspicious emails before they reach employees. This is vital for protecting sensitive corporate data, intellectual property, and client information.

Furthermore, the system can integrate with larger enterprise email systems to automatically categorize emails, apply advanced filtering, and alert the security operations center (SOC) teams when malicious activity is detected. By doing so, it not only enhances cybersecurity but also improves operational efficiency, as security teams can focus on more critical tasks instead of dealing with numerous spam-related issues.

5.3 Impact on Reducing Phishing and Malware Risks

Phishing attacks have become one of the most prevalent and harmful types of cyberattacks in recent years, often leading to data breaches, financial losses, and

reputational damage. Spam emails, particularly those containing malicious links or attachments, serve as the primary entry point for phishing and malware distribution. By detecting and blocking these types of emails, spam detection systems help prevent users from falling victim to these attacks.

In the context of phishing attacks, attackers often use deceptive tactics, such as mimicking legitimate email sources, to trick users into revealing sensitive information like login credentials, credit card numbers, or other personal details. The Spam Mail prediction developed in this project leverages advanced machine learning models, including Logistic Regression, Random Forest, and Naive Bayes, to classify emails based on text content, presence of URLs, and attachments. These features are essential for identifying phishing emails, which commonly include fake login pages or misleading links.

6.CONCLUSION

6.1 SUMMARY OF FINDINGS

In this project, we developed a spam email detection system using multiple machine learning models to classify emails as spam or not. We explored three widely used models—Logistic Regression, Random Forest, and Naive Bayes—on a dataset of 5728 emails obtained from Kaggle. After applying various preprocessing techniques like handling missing data, URL counting, and feature extraction through TF-IDF, we were able to train and evaluate the models effectively.

The models were evaluated using metrics such as accuracy, precision, recall, and F1-score, with Logistic Regression and Random Forest models achieving an accuracy of 98.95%, while Naive Bayes had a slightly lower performance at 98.43%. Both Logistic Regression and Random Forest performed similarly well, indicating their effectiveness for spam detection tasks, while Naive Bayes performed reasonably well as well, especially with regards to precision for non-spam emails.

6.2 CONTRIBUTIONS OF THE PROJECT

This project makes several significant contributions to the field of spam email detection. By leveraging well-established machine learning models, we provided a reliable solution for classifying emails based on their content, URLs, and attachments, ensuring that both personal and enterprise users can effectively filter out spam messages. The use of TF-IDF

for text-based feature extraction and the consideration of additional features like URL count and attachment presence have made the model more robust in accurately identifying spam emails.
Moreover, the project showcases the practical application of Logistic Regression, Random Forest, and Naive Bayes in solving real-world problems. It provides a foundation for further work on spam classification that can be adapted for more complex datasets and environments.

6.3 FUTURE WORK AND IMPROVEMENTS

While the model performs well, there are several areas for future improvements and enhancements. One possible direction for future work is the inclusion of deep learning models, such as Recurrent Neural Networks (RNNs) or Transformer-based models, which could potentially improve accuracy further, especially when dealing with more complex email structures or larger datasets.
Additionally, the system could be enhanced by incorporating other forms of email metadata, such as sender reputation, email domain analysis, and time of sending, which could provide further insights into whether an email is spam. Another promising improvement could be the application of ensemble methods or hybrid models, combining the strengths of multiple models for better accuracy and reduced false positives.
Another area to explore is the handling of imbalanced datasets, where the number of non-spam emails significantly exceed spam emails. Although we applied class balancing techniques, further work can be done to explore other techniques such as Synthetic Minority Over-sampling Technique (SMOTE) or cost-sensitive learning.
Lastly, this model can be deployed into real-time email filtering systems for both personal and enterprise use. Integration with existing email systems could help in filtering incoming emails and preventing the spread of unwanted or malicious emails.

7.REFERENCES

Kaggle Dataset. (Year). *Spam Email Dataset.* Kaggle. URL: https://www.kaggle.com/datasets/jackksoncsie/spam-email-dataset/data

8.APPENDICES

8.1CODE SNIPPETS

```
[2] import numpy as np
    import pandas as pd
    from sklearn.model_selection import train_test_split
    from sklearn.feature_extraction.text import TfidfVectorizer
    from sklearn.linear_model import LogisticRegression
    from sklearn.naive_bayes import MultinomialNB
    from sklearn.ensemble import RandomForestClassifier
    import plotly.express as px
    import re
    from sklearn.metrics import classification_report, accuracy_score
    import scipy.sparse as sp
```

```
[6] mail_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5728 entries, 0 to 5727
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Body    5728 non-null   object
 1   Label   5728 non-null   int64 
dtypes: int64(1), object(1)
memory usage: 89.6+ KB
```

```
[7] # checking the number of rows and columns in the dataframe
    mail_data.shape
```

```
(5728, 2)
```

```
[8] mail_data.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Label	5728.0	0.238827	0.426404	0.0	0.0	0.0	0.0	1.0

```
[9] len(mail_data['Body'].unique())
```

```
5695
```

```
[10] mail_data.isnull().sum()
```

	0
Body	0
Label	0

dtype: int64

```
[11] mail_data.duplicated().sum()
```

```
33
```

```
[12] duplicate=mail_data[mail_data.duplicated(keep='last')]
     duplicate
```

	Body	Label
1417	Subject: day off tuesday stinson , i would l...	0
1508	Subject: re : enron weather research good aft...	0
1532	Subject: schedule and more . . dr . kaminski ...	0
1667	Subject: re : summer work . . jinbaek , this...	0
1749	Subject: term paper dr . kaminski , attached...	0
1791	Subject: re : contact info glenn , please , ...	0
1801	Subject: departure of grant masson the resear...	0
1828	Subject: re : term project : brian , no prob...	0
1881	Subject: research allocations to egm hi becky...	0
1963	Subject: re : schedule and more . . jinbaek ,...	0
2188	Subject: tiger evals - attachment tiger hosts...	0
2297	Subject: telephone interview with the enron re...	0
2308	Subject: re : get together this coming tuesday...	0
2449	Subject: re : your mail zhendong , dr . kami...	0
2497	Subject: re : frank , yes . vince from : f...	0

```
[13] counts = mail_data['Label'].value_counts().reset_index()
     counts.columns = ['Label', 'Count']
     # Create a bar plot using Plotly Express
     fig = px.bar(counts, x='Label', y='Count', color='Label')
     fig.update_layout(title='Number of Spam and Ham Emails', xaxis_title='Label', yaxis_title='Count')
     fig.update_xaxes(tickvals=[0, 1], ticktext=['Ham', 'Spam'])
     fig.show()
```

```
[14] # separating the data as texts and label

     X = mail_data['Body']

     Y = mail_data['Label']
```

```
[ ]  print(X)
```

Show hidden output

```
     print(Y)
```

Show hidden output

```
[16] # Check class distribution
     print(mail_data['Label'].value_counts())
```

```
Label
0    4360
1    1368
Name: count, dtype: int64
```

```
     # Shuffle the dataset
     mail_data = mail_data.sample(frac=1, random_state=42).reset_index(drop=True)

     # Display the first few rows
     print(mail_data.head())
```

```
                                                Body  Label  \
0  Subject: wti - new eol product  ted ,  enclose...      0
1  Subject: claim your free $ 1000 home depot gif...      1
2  Subject: you don _ t know how to attract custo...      1
3  Subject: you want to submit your website to se...      1
4  Subject: impress your girl with a huge cumshot...      1

                                          Clean_Body  Num_URLs  Has_Attachment
0  subject  wti   new eol product  ted    enclose...         0           False
1  subject  claim your free   1000 home depot gif...         0           False
2  subject  you don _ t know how to attract custo...         0           False
3  subject  you want to submit your website to se...         0           False
4  subject  impress your girl with a huge cumshot...         0           False
```

```
[18] print(mail_data['Label'].head(10))
```

Show hidden output

```
[19] # Check class distribution
     print(mail_data['Label'].value_counts())
```

```
Label
0    4360
1    1368
Name: count, dtype: int64
```

```
[20] # Preprocessing Function
     def preprocess_email(text):
         # Handle missing values
         if pd.isnull(text):
             text = ""

         # Count URLs
         num_urls = len(re.findall(r'(https?://\S+)', text))

         # Check for attachments (simple keyword-based detection)
         has_attachment = bool(re.search(r'attachment|attached|file', text, re.IGNORECASE))

         # Remove URLs
         text = re.sub(r'(https?://\S+)', '', text)

         # Remove non-alphanumeric characters
         text = re.sub(r'\W', ' ', text)

         # Lowercase all text
         text = text.lower()

         return text, num_urls, has_attachment

     # Apply Preprocessing to 'body'
     mail_data['Clean_Body'], mail_data['Num_URLs'], mail_data['Has_Attachment'] = zip(*mail_data['Body'].map(preprocess_email))

     # Display Processed Data
     print(mail_data.head())
```

```
[21] #Feature Extraction
     # TF-IDF Vectorization for Text Body
     vectorizer = TfidfVectorizer(
         stop_words='english',
         max_df=0.7,
         min_df=5,
         max_features=3000,
         lowercase=True
     )

     X_text = vectorizer.fit_transform(mail_data['Clean_Body'])

     # Combine Text and Numeric Features
     X_combined = sp.hstack([
         X_text,
         np.array(mail_data['Num_URLs']).reshape(-1,1),
         np.array(mail_data['Has_Attachment']).reshape(-1,1)
     ])

     # Define Labels
     y = mail_data['Label']

     #Train-Test Split
     X_train, X_test, y_train, y_test = train_test_split(X_combined, y, test_size=0.2, random_state=42, stratify=y)
```

```
[22] # Train Logistic Regression Model
     log_reg_model = LogisticRegression(class_weight='balanced', max_iter=1000)
     log_reg_model.fit(X_train, y_train)
```

```
LogisticRegression(class_weight='balanced', max_iter=1000)
```

```
[23] # prediction on training data

     prediction_on_training_data = log_reg_model.predict(X_train)
     accuracy_on_training_data = accuracy_score(y_train, prediction_on_training_data)
     print('Accuracy on training data : ', accuracy_on_training_data)
```

```
Accuracy on training data :  0.9914884329986905
```

```
[25] # Evaluate Model
     y_pred = log_reg_model.predict(X_test)
     print("Model Accuracy:", accuracy_score(y_test, y_pred))
     print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Model Accuracy: 0.9834205933682374

Classification Report:
                precision    recall  f1-score   support

           0       1.00      0.98      0.99       872
           1       0.95      0.99      0.97       274

    accuracy                           0.98      1146
   macro avg       0.97      0.98      0.98      1146
weighted avg       0.98      0.98      0.98      1146
```

```
[26] rf_model = RandomForestClassifier(class_weight='balanced', random_state=42)
     rf_model.fit(X_train, y_train)
     # Random Forest Evaluation
     y_pred_rf = rf_model.predict(X_test)
     print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
     print("\nRandom Forest Classification Report:\n", classification_report(y_test, y_pred_rf))
```

```
Random Forest Accuracy: 0.9834205933682374

Random Forest Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       872
           1       0.96      0.97      0.97       274

    accuracy                           0.98      1146
   macro avg       0.98      0.98      0.98      1146
weighted avg       0.98      0.98      0.98      1146
```

```
[27] # Naive Bayes
     nb_model = MultinomialNB()
     nb_model.fit(X_train, y_train)
     # Naive Bayes Evaluation
     y_pred_nb = nb_model.predict(X_test)
     print("Naive Bayes Accuracy:", accuracy_score(y_test, y_pred_nb))
     print("\nNaive Bayes Classification Report:\n", classification_report(y_test, y_pred_nb))
```

```
Naive Bayes Accuracy: 0.9790575916230366

Naive Bayes Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       872
           1       0.97      0.94      0.96       274

    accuracy                           0.98      1146
   macro avg       0.98      0.97      0.97      1146
weighted avg       0.98      0.98      0.98      1146
```

```
[28] #Predict on a Sample Email (for each model)
     def predict_email(email_text, model):
         text, num_urls, has_attachment = preprocess_email(email_text)
         text_vector = vectorizer.transform([text])
         features = sp.hstack([
             text_vector,
             np.array([num_urls]).reshape(1, -1),
             np.array([has_attachment]).reshape(1, -1)
         ])
         prediction = model.predict(features)
         return "Spam" if prediction[0] == 1 else "Ham"
```

8.2TESTING WITH EXAMPLES

```
[29] # Example Prediction for each model
     sample_email = """
     Subject: Free Gift Card Offer!
     Click here: http://spamlink.com to claim your reward.
     """
```

```
# Test on Logistic Regression Model
print("\nLogistic Regression Sample Email Prediction:", predict_email(sample_email, log_reg_model))

# Test on Random Forest Model
print("\nRandom Forest Sample Email Prediction:", predict_email(sample_email, rf_model))

# Test on Naive Bayes Model
print("\nNaive Bayes Sample Email Prediction:", predict_email(sample_email, nb_model))
```

```
Logistic Regression Sample Email Prediction: Spam

Random Forest Sample Email Prediction: Spam

Naive Bayes Sample Email Prediction: Spam
```

```
[32] # Example Prediction for each model
     sample_email = """
     Subject: re : new color printer  sorry ,  don ' t we need to know the cost , as well .  - - - - - - - - - - - - - - - - - - - - - - forwarded by kevin g moore / hou / ect
     """
```

```
[33] # Test on Logistic Regression Model
     print("\nLogistic Regression Sample Email Prediction:", predict_email(sample_email, log_reg_model))

     # Test on Random Forest Model
     print("\nRandom Forest Sample Email Prediction:", predict_email(sample_email, rf_model))

     # Test on Naive Bayes Model
     print("\nNaive Bayes Sample Email Prediction:", predict_email(sample_email, nb_model))
```

```
Logistic Regression Sample Email Prediction: Ham

Random Forest Sample Email Prediction: Ham

Naive Bayes Sample Email Prediction: Ham
```