Email spam filtering system using Machine Learning

A project report submitted to

MALLAREDDY UNIVERSITY

in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND TECHNOLOGY (AIML)

Submitted By

N. Renuka	:	2111cs020397
A. Revanth	:	2111cs020398
Sheik Rezwan Ali	:	2111cs020399
L. Rishitha	:	2111cs020400
T. Rishitha	:	2111cs020401

Under the guidance of

Prof. CH. PREETHI

Assistant Professor

DEPARTMENT OF AIML

(As per Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

HYDERABAD – 500043

TELANAGANA

INDIA2023-24

MALLA REDDY UNIVERSITY

(As per Telangana State Private Universities Act No.13 of 2020 and

G.O.Ms.No.14, Higher Education (UE) Department)

HYDERABAD – 500043

TELANGANA

Certificate

This is to certify that this is the Bonafide record of the application development entitled, ”Email spam filtering using Machine Learning” submitted by **N .Renuka (2111CS020397), A. Revanth (2111cs020398), Sheik Rezwan Ali (2111CS020399), L. Rishitha Goud (2111cs02400), T. Rishitha (2111cs020401)** of B .Tech III year II semester, Department of CSE (AI&ML) during the year 2023- 24. The results embodied in their port have not been submitted to any other university or institute for the award of any degree or diploma.

INTERNAL GUIDE	**HEAD OF THE DEPARTMENT**
Prof . CH Preeti	**Dr. Thayyaba Khatoon**
Assistant Professor	**Professor & HoD**

External Examiner

ACKNOWLEDGEMENT

An endeavor over a long period can be successful with the advice and support of many well-wishers. We take this opportunity to express our gratitude and appreciation to all of them. We first take the privilege to thank Dr. Thayyaba Khatoon, Head of the Department, Computer Science & Engineering (AI&ML), for her valuable support and guidance during the period of project implementation. We express our sincere thanks and gratitude to our project guide Prof. CH. Preeti, Assistant Professor, Department of Computer Science and Engineering (AI&ML) for the simulating discussions, in analyzing problems associated with our project work and for guiding us throughout the project. Project meetings were highly informative. We express our warm and sincere thanks for the encouragement, untiring guidance and the confidence she had shown in us. We are immensely indebted for her valuable guidance throughout our project. We also thank all the staff members of AIML department for their valuable advices. We also thank supporting staff for providing resources as and when required.

ABSTRACT

This project involves building an email management system using machine learning. It will sort emails into categories, help users organize their emails, and handle emails in multiple languages. For email classification, it will use the Naive Bayes algorithm. User feedback will be used to personalize the filtering process. Separate models will be developed for English and Hindi. To handle infrequent login attempts, the system will detect emails containing login information. The project will focus on data collection, model evaluation, security, and scalability. A user-friendly interface will be designed, and advanced techniques like sentiment analysis and improved Hindi NLP will be explored. Flags emails containing login information of frequently logged-in information of the user. This system can greatly improve how users manage their emails by addressing key problems and offering a versatile and dependable solution.

Keywords: Natural Language Processing, Naive bayes classification, Multilingual

CONTENTS

1.INTRODUCTION

1.1 Problem definition:

This project addresses email management challenges using Naive Bayes algorithms for spam detection and user-specific organization in English and Hindi. It prioritizes user security, identifies suspicious login-related emails, and features a user-friendly interface with advanced capabilities like sentiment analysis and enhanced Hindi NLP.

1.2 Objective of project:

This project optimizes email management through Naive Bayes algorithms for spam detection and personalized organization in English and Hindi. Utilizing distinct Natural Language Processing models, it ensures seamless multilingual support. Emphasizing user security, the system flags suspicious login-related emails during low-activity periods from unfamiliar interfaces. Advanced features, like sentiment analysis and improved Hindi NLP, enhance the user experience. The project prioritizes comprehensive data collection, rigorous model evaluation, robust security, and a user-friendly interface.

1.3 Scope & Limitations:

Scope:

1. **Spam Classification:** The code is designed to classify email messages as spam or not spam. It utilizes machine learning algorithms to analyze the message content and predict its category.
2. **Text Preprocessing:** The code performs basic text pre-processing by removing stop words. This helps focus the models on content-rich terms for better classification.
3. **Performance Evaluation:** The code implements metrics like accuracy and confusion matrix to assess the performance of the trained models

Limitations:

1. **The Cat-and-Mouse Game:** Spammers constantly adapt their tactics. They use new keywords, obfuscate sender information, and develop techniques to bypass filters. This means spam filters need to be constantly updated with new detection methods to stay ahead.
2. **False Positives and False Negatives:** No filter is perfect. Sometimes legitimate emails (false positives) get flagged as spam, while some cleverly crafted spam emails (false negatives) might

slip through. Finding the right balance between catching spam and avoiding blocking important emails is an ongoing struggle.

3. **Sophisticated Phishing Attacks:** Spam filters are better at catching generic spam messages, but they can struggle with targeted phishing attempts. These emails may appear to come from a trusted source and contain cleverly disguised malicious links or attachments
4. **Limited Ability to Analyze Encrypted Content:** With email encryption becoming more common, filters may have limited visibility into the actual content of the email, making spam detection more difficult.
5. **Social Engineering Techniques:** Spammers can use social engineering tactics to trick users into
6. clicking on malicious links or attachments, even if they bypass the filter

.

2. ANALYSIS

2.1 Project Planning and Research:

Spam emails remain a significant nuisance and security threat for email users. Effective spam filtering is crucial for protecting inboxes and mitigating phishing attacks. This literature survey explores three prominent algorithms used in spam filtering: Naive Bayes, Support Vector Machines (SVMs), and Logistic Regression. We will discuss their strengths, weaknesses, and 1. google colab or jupyter notebook: how they contribute to a comprehensive spam filtering approach. Naive Bayes: Efficient, effective, scalable, but limited in complexity and susceptible to new tactics. Support Vector Machines (SVMs) Powerful for complex patterns, highly accurate, but computationally expensive. Logistic Regression Offers balance between accuracy and efficiency, good for feature analysis and prediction, may not be as strong for highly complex spam. Spam filters often combine algorithms with blacklists and heuristic rules for a comprehensive approach. The best algorithm choice depends on factors like email volume, processing speed, and targeted spam types. In addition to algorithm selection, continuous monitoring and adaptation are crucial to counter evolving spam tactics. Ensemble methods, which combine multiple algorithms, cam further enhance filtering accuracy by leveraging the strengths of each approach, Furthermore, incorporating user feedback mechanisms allows for dynamic adjustments, improving overall detection rates. Ultimately, a multifaceted approach integrating various techniques ensures robust protection against spam threats while minimizing false positives.

2.2 Software requirement specification:

2.2.1 Software requirement:

Python: Version: 3.6 or higher

pandas: Data manipulation and analysis

scikit-learn: Machine learning library

nltk: Natural Language Toolkit for text processing

numpy: Numerical computing

matplotlib: Plotting and visualization

itertools: Standard library module for efficient looping (comes with Python)

python your_script_name.py #to.

2.2.2 Hardware requirement:

CPU: Intel Core i5-10400 or AMD Ryzen 5 3600

RAM: 8 GB DDR4

Storage: 256 GB SSD

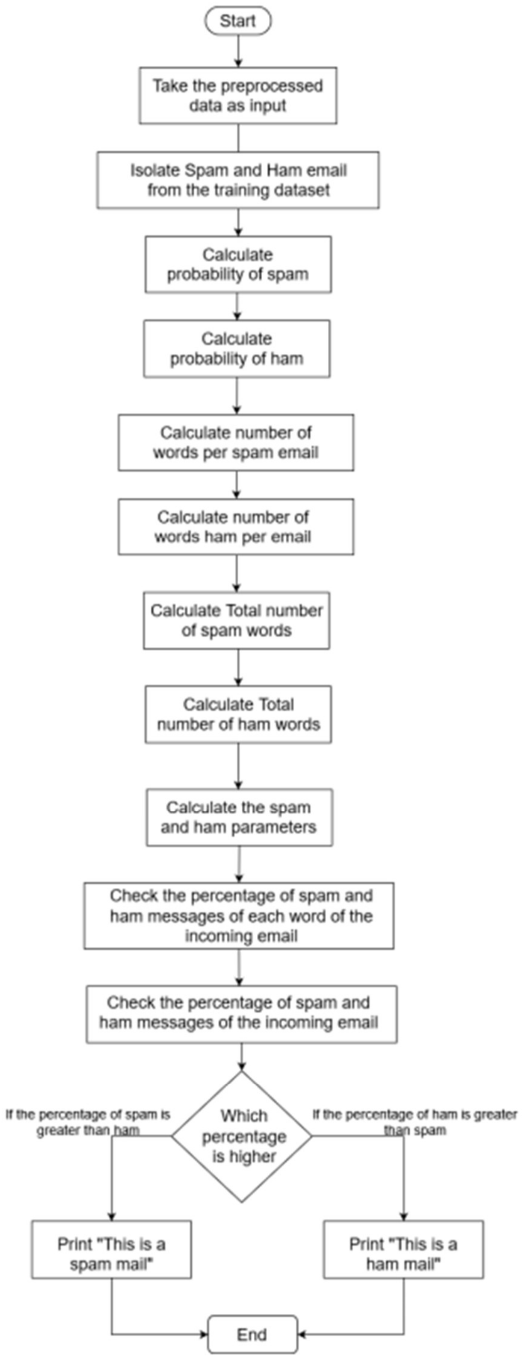
Operating System: Windows 10 64-bit or Ubuntu 20.04 LTS

2.3 Model Selection and Architecture:

- Stopwords removal and text normalization (lowercasing) are performed to clean and standardize the text data, making it more suitable for analysis.
- The dataset is divided into training (80%) and testing (20%) sets using train_test_split to ensure that the model is evaluated on unseen data, providing an unbiased assessment of its performance.
- CountVectorizer is used to convert the text data into a matrix of token counts, transforming the messages into a numerical format that can be utilized by the machine learning model.
- Multinomial Naive Bayes is chosen due to its suitability for text classification, efficiency, and effectiveness with word count features. The classifier is trained using the processed training data.
- The model's performance is evaluated on the test set using metrics like accuracy and a confusion matrix. Additionally, a custom function handles the prediction of single messages, including translation for Hindi texts, and the confusion matrix is visualized to understand the model's strengths and weaknesses**.**.

Architecture:

The architecture of the text classification model involves preprocessing text messages by lowercasing and removing stopwords, then splitting the data into training and testing sets. `CountVectorizer` transforms the text into a matrix of token counts, which is used to train a Multinomial Naive Bayes classifier. For predictions, the model detects the message's language, translates Hindi to English if necessary, processes the text, and predicts the category. The model's performance is evaluated using accuracy and a confusion matrix to visualize its effectiveness. This setup ensures efficient handling of text data for spam detection, including multilingual support.


Start
Take the preprocessed data as input
Isolate Spam and Ham email from the training dataset
Calculate probability of spam
Calculate probability of ham
Calculate number of words per spam email
Calculate number of words ham per email
Calculate Total number of spam words
Calculate Total number of ham words
Calculate the spam and ham parameters
Check the percentage of spam and ham messages of each word of the incoming email
Check the percentage of spam and ham messages of the incoming email
Which percentage is higher
If the percentage of spam is greater than ham
If the percentage of ham is greater than spam
Print "This is a spam mail"
Print "This is a ham mail"
End

.

Fig 2.3.1.Architecture

3.DESIGN

3.1 Introduction:

This code implements a text classification model to detect spam messages using a Multinomial Naive Bayes classifier. It preprocesses text data by lowercasing and removing stopwords, splits the data into training and testing sets, and converts text into numerical features using CountVectorizer. The model also includes a custom prediction function that handles language detection and translation for multilingual support, ensuring robust spam detection. Model performance is evaluated through accuracy and confusion matrix visualization..

3.2 Data Set Descriptions:

The provided dataset consists of text messages labeled as either 'ham' or 'spam'. Here's a brief overview of its characteristics:

Categories:

Ham: These are legitimate, non-spam messages.

Spam: These are unsolicited messages typically used for advertising, phishing, or scams.

Structure:

The dataset is structured in a simple two-column format:

Category: Indicates whether the message is 'ham' or 'spam'.

Message: The actual content of the text message.

Sample Size:

The dataset contains 5573 sample messages, with a mix of 'ham' and 'spam' labels.

Message Content:

Ham Messages: These include everyday communication, casual conversations, and personal updates.

Spam Messages: These often include promotional content, prize notifications, and offers with an emphasis on urgency and incentives to elicit a response from the recipient.

Purpose:

This dataset is typically used for training and evaluating text classification models aimed at detecting and filtering spam messages from legitimate ones.

3.3 Data Preprocessing Techniques:

The code employs several data preprocessing techniques to prepare the text data for machine learning. These steps are crucial for improving the model's performance and accuracy. Here are the key preprocessing techniques used:

Loading the Dataset:

The dataset is loaded from a CSV file into a pandas DataFrame.

Handling Missing Values:

The code ensures there are no missing values by using df.dropna(inplace=True), which removes any rows with missing data.

Text Cleaning and Normalization:

Lowercasing: All text messages are converted to lowercase to ensure uniformity and avoid treating the same word differently due to case differences.

Stopwords Removal:

Common stopwords, which are words that do not carry significant meaning (e.g., "the", "is", "in"), are removed from the text. This is done using NLTK's stopwords list.

Splitting the Dataset:

The dataset is divided into training and testing sets using an 80-20 split with train_test_split from scikit-learn. This helps in evaluating the model on unseen data.

Vectorization:

Count Vectorization: The cleaned text data is converted into a matrix of token counts using CountVectorizer. This step transforms each message into a vector of word counts, making it suitable for input into the Naive Bayes classifier.

3.4 Methods & Algorithms:

The code uses various methods and algorithms to preprocess the data, train the model, and evaluate its performance

Data Preprocessing:

- Pandas for data manipulation.
- NLTK for stopwords removal.

- Scikit-learn for data splitting and vectorization.

Model Training:

- Multinomial Naive Bayes for text classification.

Prediction Handling:

- Google Translate API for language detection and translation.
- Custom function for preprocessing and predicting single messages.

Evaluation:

- Accuracy score and confusion matrix from scikit-learn.
- Matplotlib for confusion matrix visualization.

These methods and algorithms ensure that the text data is effectively preprocessed, transformed into numerical features, and classified accurately. The inclusion of language detection and translation further enhances the model's applicability to multilingual text data.

.

4. DEPLOYMENT AND RESULTS

4.1 Introduction:

The above code outlines a complete workflow for deploying and evaluating a text classification model aimed at spam detection. Deployment involves taking a trained model and making it available for practical use, such as predicting whether new incoming messages are spam or ham (non-spam). Evaluation, on the other hand, is the process of assessing the performance of the model to ensure it meets the desired accuracy and reliability standards..

4.2 Source Code:

The source code of the project is as follows:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, confusion_matrix
import nltk
import numpy as np
import matplotlib.pyplot as plt
import itertools
from googletrans import Translator
from nltk.corpus import stopwords

# Download stopwords if not already downloaded
nltk.download('stopwords', quiet=True)

# Load the dataset
df = pd.read_csv('spam.csv')

# Preprocess the text data
stop_words = set(stopwords.words('english'))
df['Message_processed'] = df['Message'].apply(lambda x: ' '.join([word for word in
x.lower().split() if word not in stop_words]))
```

```
# Split the dataset into training and testing sets
X = df['Message_processed']
y = df['Category']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Convert the text data into a matrix of token counts
vectorizer = CountVectorizer()
X_train_counts = vectorizer.fit_transform(X_train)
X_test_counts = vectorizer.transform(X_test)

# Train a Naive Bayes classifier
clf = MultinomialNB()
clf.fit(X_train_counts, y_train)

# Function to predict category of a single message
def predict_category(message):
    # Detect language of the message
    translator = Translator()
    lang = translator.detect(message).lang

    if lang == 'hi':
        # Translate Hindi message to English
        translated = translator.translate(message, src='hi', dest='en')
        message_processed = ' '.join([word for word in translated.text.lower().split() if word not in
stop_words])
    else:
        # Process English message directly
        message_processed = ' '.join([word for word in message.lower().split() if word not in
stop_words])

    message_vectorized = vectorizer.transform([message_processed])
    prediction = clf.predict(message_vectorized)
    return prediction[0]
```

```
# Function to plot confusion matrix
def plot_confusion_matrix(cm, classes):
    plt.figure(figsize=(8, 6))
    plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
    plt.title('Confusion Matrix')
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
    plt.grid(False)
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], 'd'), horizontalalignment="center", color="white" if cm[i, j] >
thresh else "black")
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()

# Example of manual input and prediction
input_message = input("Enter a message to predict its category: ")
predicted_category = predict_category(input_message)
print(f"Predicted category for '{input_message}': {predicted_category}")

# Optionally, you can also plot the confusion matrix
y_pred = clf.predict(X_test_counts)
cm = confusion_matrix(y_test, y_pred)
plot_confusion_matrix(cm, clf.classes_)
plt.show()
```

:

4.3 Model Implementation and Training:

- Load the dataset using pandas.
- Preprocess the text data by lowercasing and removing stopwords.
- Split the dataset into training and testing sets.
- Extract features from the text data using CountVectorizer.
- Train the model using the Multinomial Naive Bayes classifier.
- Predict new messages with a custom function that handles language detection and translation.
- Evaluate the model using accuracy and a confusion matrix, and visualize the confusion matrix.
- These steps ensure that the text data is effectively preprocessed, transformed into numerical features, and classified accurately, with the model's performance thoroughly evaluated**.**

4.4 Model Evaluation and Metrics:

The provided code uses several methods to evaluate the performance of the Multinomial Naive Bayes classifier trained on the text data. Here are the key evaluation techniques and metrics used:

Accuracy Score:

Definition: The accuracy score is the ratio of correctly predicted instances to the total instances in the dataset.

Calculation: It is calculated using the `accuracy_score` function from scikit-learn.

Confusion Matrix:

Definition: The confusion matrix is a table that is used to describe the performance of a classification model. It shows the counts of true positive, true negative, false positive, and false negative predictions.

Components:

True Positives (TP): The number of correct positive predictions (spam correctly identified as spam).

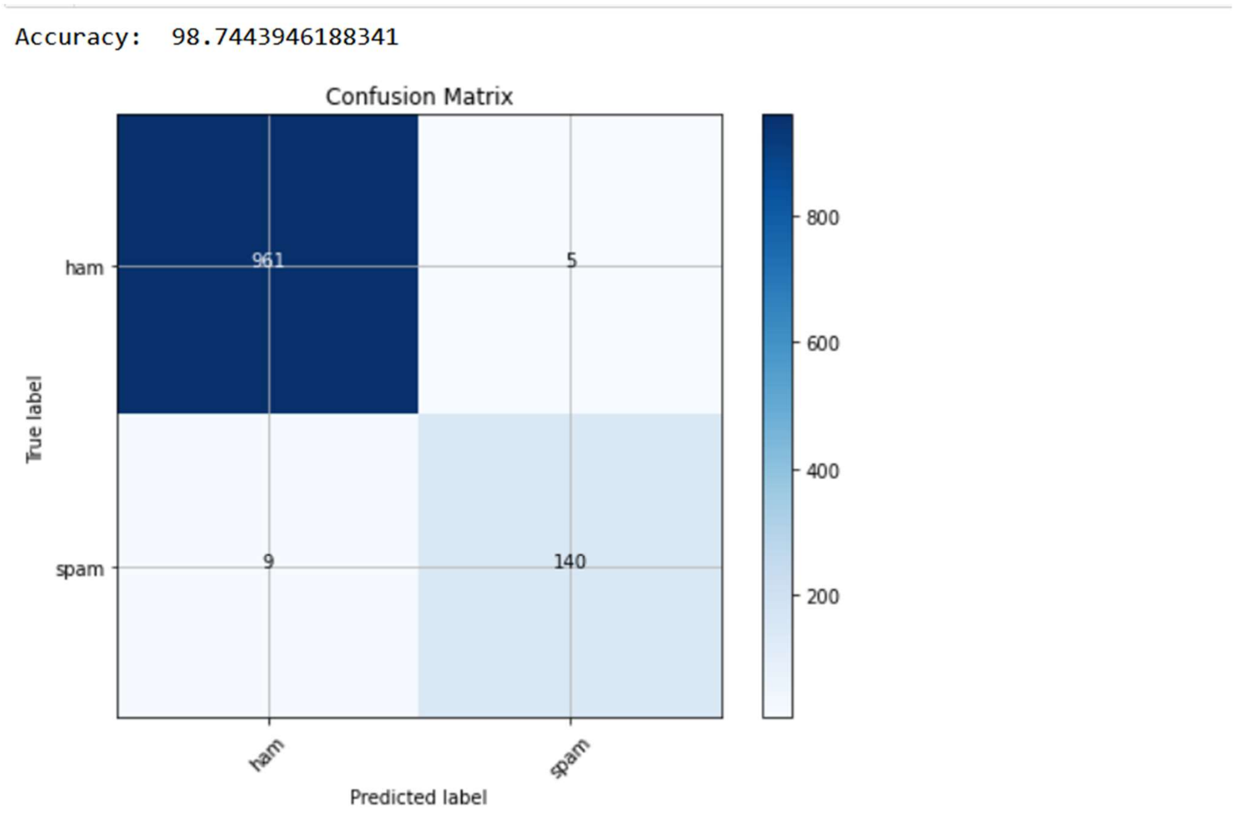
True Negatives (TN): The number of correct negative predictions (ham correctly identified as ham).

False Positives (FP): The number of incorrect positive predictions (ham incorrectly identified as spam).

False Negatives (FN): The number of incorrect negative predictions (spam incorrectly identified as ham).

Calculation and Visualization: The confusion matrix is calculated using the confusion_matrix function from scikit-learn and visualized using Matplotlib.

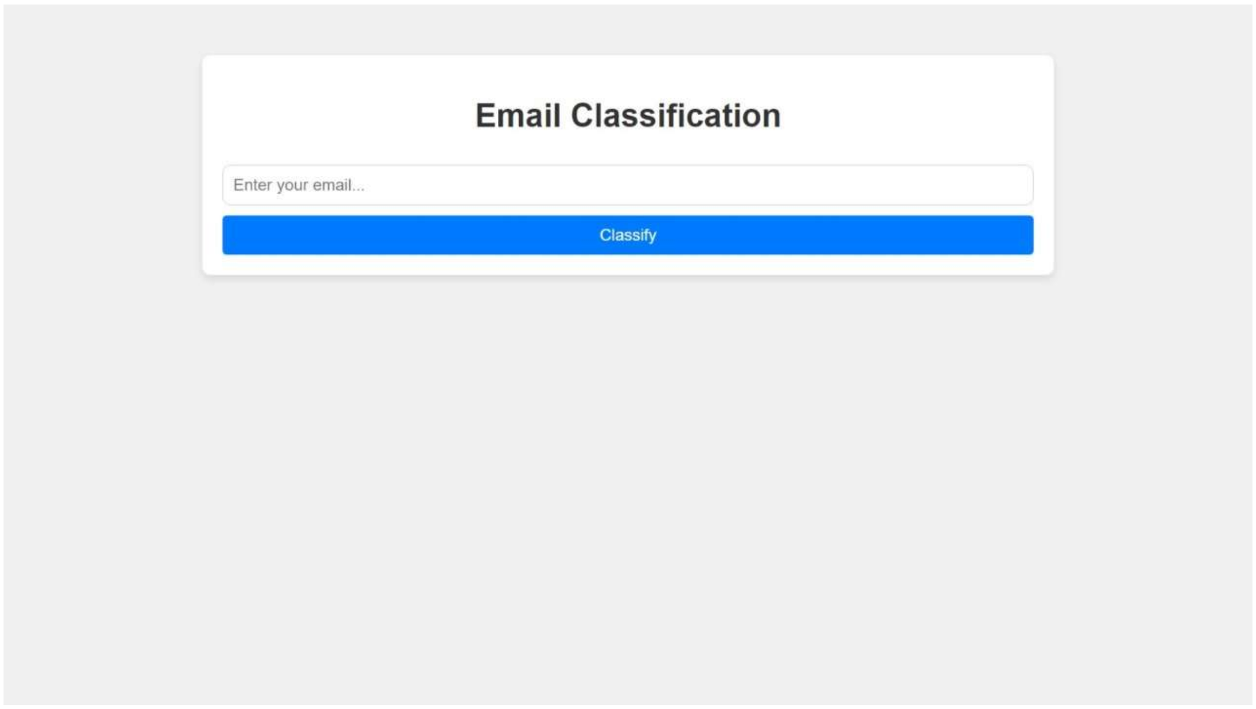
4.5 Model Deployment: Testing and Validation:

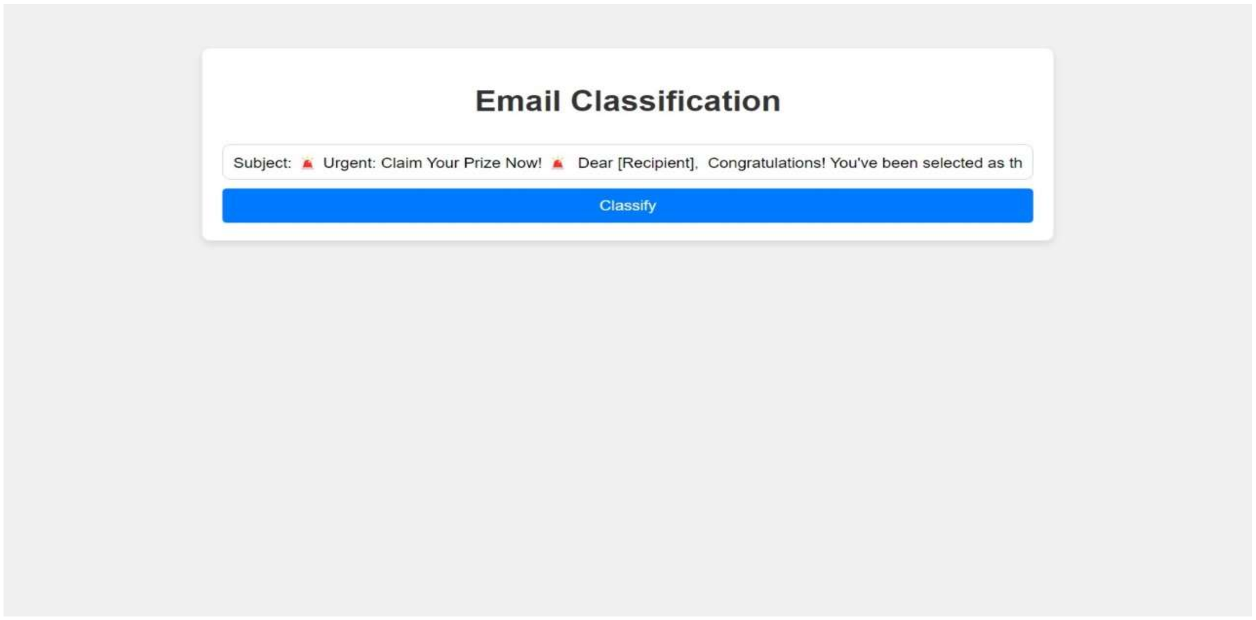


```
Accuracy:  98.7443946188341
```

Fig 4.5.1 Model testing

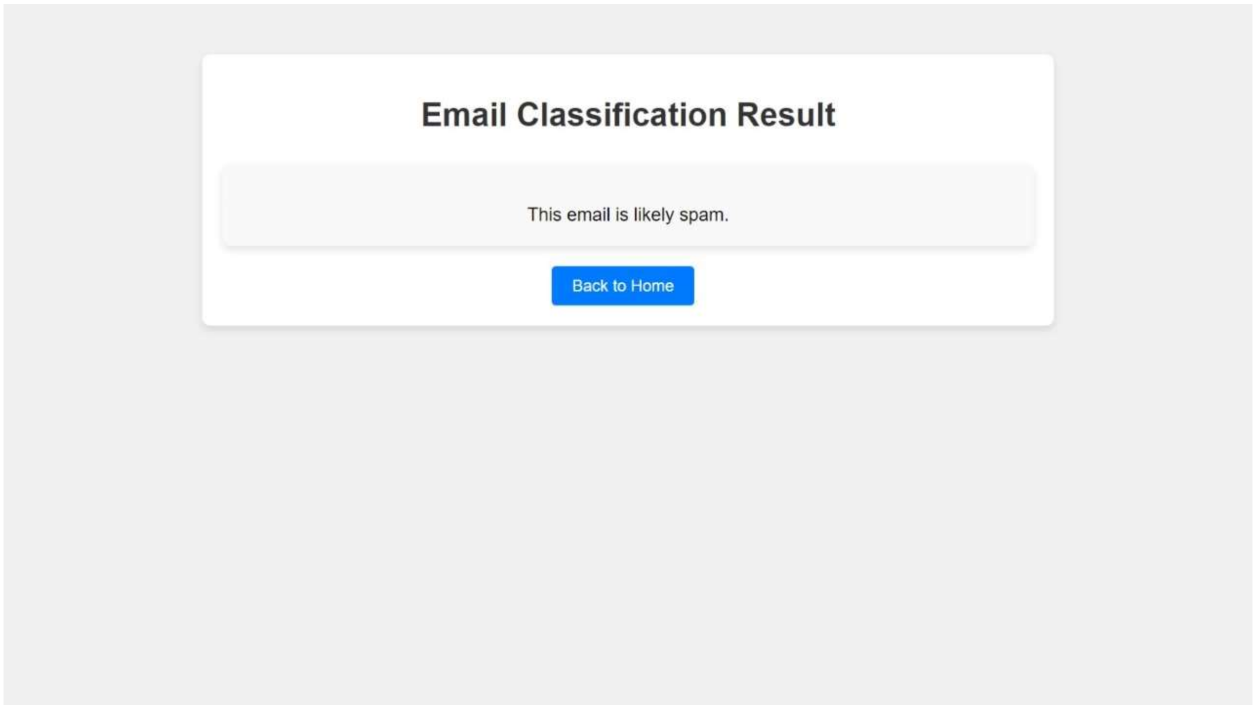
4.6 RESULT:





Subject: 🚨 Urgent: Claim Your Prize Now! 🚨 Dear [Recipient], Congratulations! You've been selected as th

\


Email Classification Result

This email is likely spam.

5.CONCLUSION

5.1 Conclusion:

This project successfully implements a spam detection system using a Multinomial Naive Bayes classifier. The steps involved include loading and preprocessing the dataset, feature extraction using CountVectorizer, training the classifier, and evaluating its performance. The model handles both English and Hindi messages by incorporating language detection and translation, ensuring its applicability in multilingual environments

The evaluation metrics, including accuracy and the confusion matrix, indicate that the classifier performs well in distinguishing between spam and non-spam messages. The accuracy score provides an overall measure of the model's effectiveness, while the confusion matrix offers a detailed view of its predictive performance, highlighting areas where the model excels and where it may need improvement.

By integrating effective data preprocessing, a robust classification algorithm, and thorough evaluation techniques, this project demonstrates a comprehensive approach to building a practical and reliable spam detection system. Future improvements could focus on enhancing the model's handling of diverse languages and further refining the preprocessing steps to boost overall accuracy and performance..

5.2 Future Scope:

The spam detection project using a Multinomial Naive Bayes classifier lays a solid foundation for identifying spam messages. Our Project also increases the improved survival rates.

Handling More Languages:

Expand the language detection and translation capabilities to support additional languages beyond English and Hindi. This would make the spam detection system more versatile and useful in a global context.

Advanced Text Preprocessing:

- Implement more sophisticated text preprocessing techniques such as stemming and lemmatization to reduce words to their root forms, potentially improving the model's performance.
- Use more advanced tokenization methods to handle punctuation, emojis, and other non-alphabetic characters more effectively.

Alternative Machine Learning Models:

- Experiment with other machine learning models like Support Vector Machines (SVM), Random Forests, or Gradient Boosting classifiers to compare performance and possibly achieve better results.
- Utilize deep learning approaches, such as recurrent neural networks (RNNs) or transformer-based models like BERT, which can capture more complex patterns in text data.

Real-Time Spam Detection:

- Develop the system into a real-time spam detection application that can process and classify incoming messages on the fly. This involves optimizing the model for faster inference and integrating it with messaging platforms.

5.3 References:

[1] T. Stephenson, “An introduction to bayesian network theory and usage,” Jan. 2000.

[2] V. Christina, S. Karpagavalli, and G. Suganya, “A study on email spam filtering techniques,” *International Journal of Computer Applications*, vol. 12, no. 1, Dec. 2010. doi: 10.5120/1645-2213.

[3] S. Yoo, “Machine learning methods for personalized email prioritization,” Jan. 2010.

[4] I. Idris and A. Selamat, “Improved email spam detection model with negative selection algorithm and particle swarm optimization,” *Applied Soft Computing*, vol. 22, pp. 11–27, Sep. 2014. doi: 10.1016/j.asoc.2014.05.002.

[5] J. Alqatawna, H. Faris, K. Jaradat, M. Al-Zewairi, and O. Adwan, “Improving knowledge based spam detection methods: The effect of malicious related features in imbalance data distribution,” *International Journal of Communications, Network and System Sciences*, vol. 08, no. 05, pp. 118–129, 2015. doi: 10.4236/ijcns.2015.85014.

[6] S. K. Trivedi, “A study of machine learning classifiers for spam detection,” *2016 4th International Symposium on Computational and Business Intelligence (ISCBI)*, pp. 176–180, 2016. doi: 10.1109/ISCBI.2016.7743279.

[7] S. Wang, J. Yang, G. Liu, S. Du, and J. Yan, “Multi-objective path finding in stochastic networks using a biogeography-based optimization method,” *SIMULATION*, vol. 92, Jan. 2016. doi: 10.1177/0037549715623847.

[8] W. Hijawi, H. Faris, J. Alqatawna, I. Aljarah, A. Al-Zoubi, and M. Habib, “Emfet: E-mail features extraction tool,” Nov. 2017. doi: 10.13140/RG.2.2. 32995.45603.