Spam Email Classification using NLP and Machine Learning

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

M N Pranav Ponnanna

Pranavponnanna34@gmail.com

Under the Guidance of

Abdul Aziz MD

Master Trainer, Edunet Foundation

ACKNOWLEDGEMENT

I would like to take this opportunity to express my heartfelt gratitude to all those who have supported and guided me throughout this project.

First and foremost, I extend my deepest gratitude to my supervisor, [Mr. Abdul Aziz MD], for their invaluable mentorship and constant encouragement. Their guidance, constructive criticism, and insightful suggestions have been instrumental in shaping the direction and successful completion of this project. Their unwavering confidence in my abilities motivated me to push my limits and strive for excellence.

Working under their supervision has been an enriching experience, not just in terms of technical knowledge but also in developing a sense of professionalism and responsibility. The time and effort they devoted to discussing my ideas and addressing my queries have truly made this journey both productive and inspiring.

I would also like to thank [Other individuals or organizations, if any] for their direct or indirect support, which has contributed significantly to this project. Their input and assistance have been invaluable in overcoming challenges and achieving my objectives.

Finally, I am deeply grateful to my family and friends for their constant support and encouragement, which provided me with the strength and determination to see this project through to completion.

Thank you once again to everyone who has been a part of this journey. This project would not have been possible without your guidance and support.

ABSTRACT

The rapid growth of email communication has led to a significant increase in spam emails, posing challenges for email security and productivity. This project, *Spam Email Classification using NLP*, focuses on developing an efficient and accurate system to distinguish between spam and legitimate emails using Natural Language Processing (NLP) techniques.

The project employs a systematic approach, starting with data collection and preprocessing, including tokenization, stemming, lemmatization, and the removal of stopwords and special characters to clean and standardize the email text. Various NLP-based feature extraction techniques, such as Bag of Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF), and word embeddings, are utilized to represent the text data numerically for model input.

To achieve robust classification, several machine learning models, including Logistic Regression, Naïve Bayes, Support Vector Machines (SVM), and ensemble methods, are implemented and evaluated. Additionally, deep learning techniques such as Recurrent Neural Networks (RNN) and Bidirectional LSTMs are explored to capture the contextual relationships in email content.

The model performance is rigorously tested using evaluation metrics like accuracy, precision, recall, F1-score, and AUC-ROC. Results demonstrate that the proposed system achieves high accuracy in detecting spam emails, proving its effectiveness and practicality.

This project highlights the integration of NLP and machine learning as a powerful tool in combating spam emails, providing a scalable solution for email filtering systems. Future work could include real-time deployment and further optimization with advanced techniques like transformer-based models for enhanced accuracy and scalability.

TABLE OF CONTENT

LIST OF FIGURES

CHAPTER 1

Introduction

Email communication has become an integral part of our daily lives, facilitating personal and professional interactions. However, the increasing volume of spam emails disrupts productivity and poses security risks such as phishing and malware. Traditional spam filters often fail to adapt to sophisticated spam patterns, leading to inefficiencies. This project, *Spam Email Classification using NLP*, addresses this challenge by employing advanced Natural Language Processing techniques to classify emails accurately. By analyzing email content and context, the system improves spam detection, ensuring a safer and more efficient email experience. The solution aims to enhance email filtering systems for individuals and organizations alike

1.1Problem Statement:

Spam emails, often unsolicited and malicious, pose a significant challenge in email communication. These emails clutter inboxes, waste time, and can lead to security risks such as phishing attacks and malware dissemination. Existing spam filters struggle with adaptability to evolving spam patterns, leading to false positives or negatives. Thus, there is a pressing need for an intelligent and efficient spam classification system that can accurately identify spam while ensuring legitimate emails are not misclassified.

1.2Motivation:

This project was chosen due to the increasing reliance on email communication in personal and professional settings, where spam emails can disrupt workflows and compromise security. With advancements in Natural Language Processing (NLP), it is now feasible to develop robust spam detection systems capable of understanding the content and context of emails. The potential applications of this project include improving email filtering services, enhancing cybersecurity measures, and increasing productivity. Its impact extends to individuals, businesses, and organizations by ensuring a safer and more efficient email environment.

1.3Objective:

The primary objectives of this project are:

- To develop a spam email classification system using NLP techniques.
- To preprocess and analyze email data to extract meaningful features.
- To implement machine learning and deep learning models for accurate spam detection.
- To evaluate the performance of the models using metrics such as accuracy, precision, recall, and F1-score.
- To propose a scalable and efficient solution for real-time email spam detection.

1.4Scope of the Project:

The scope of this project includes:

- **Data Processing:** Analysing and preprocessing email datasets for efficient feature extraction.
- **Model Development:** Implementing and comparing machine learning and deep learning approaches for spam classification.
- **Evaluation:** Measuring the performance of the developed system using industry-standard metrics.
- **Deployment Readiness:** Providing a foundation for real-world deployment in email filtering systems.

Limitations:

- The model's performance may vary depending on the diversity and quality of the training data.
- Complex spam techniques, such as highly obfuscated text or multimedia-based spam, may require additional processing techniques.
- Real-time implementation might necessitate optimization for faster processing without compromising accuracy.

.

CHAPTER 2

Literature Survey

2.1 Review of Relevant Literature

- Androutsopoulos et al. (2000): One of the foundational works in spam email classification, this study applied Naïve Bayes classifiers for spam detection. The authors demonstrated that Naïve Bayes, when trained on email features such as word frequencies, could achieve promising results. However, the method was limited by its inability to adapt to new types of spam, leading to high false positives in certain cases.
- Rennie et al. (2003): This work explored the use of support vector machines (SVM) for spam classification, proving that SVMs outperformed traditional methods like Naïve Bayes in many cases. The study also emphasized the importance of feature selection and preprocessing, such as tokenization and stemming, in improving model performance.

2.2 Existing Models for Spam Email Classification

1. Naïve Bayes Classifier

- **Description**: Naïve Bayes is one of the earliest and most popular models used for spam classification. It is based on the Bayes theorem and assumes that the features (words in this case) are independent. It calculates the probability that a given email is spam or not based on the likelihood of word occurrences in spam or non-spam categories.
- **Strengths**: Simple to implement, fast, and effective for smaller datasets.
- **Limitations**: Assumption of feature independence limits its performance when there are complex relationships between words.

2. Support Vector Machine (SVM)

- **Description**: SVM is a powerful machine learning algorithm that works by finding the optimal hyperplane that separates the two classes (spam and non-spam). It is especially effective for binary classification tasks and works well with high-dimensional feature spaces, such as the ones found in text classification.
- **Strengths**: Effective for high-dimensional data and can handle complex decision boundaries.
- **Limitations**: Requires careful tuning of kernel functions and regularization parameters; computationally expensive for large datasets.

2.3 Gaps and Limitations in Existing Solutions

1. Adaptability: Traditional models like Naïve Bayes and SVM are often less effective at adapting to new spam patterns, which evolve rapidly. These methods rely heavily on feature engineering and may not generalize well to unseen spam tactics.
2. Handling Context: Many existing methods, especially Naïve Bayes and basic machine learning models, fail to capture the contextual relationships within email text, leading to reduced accuracy in detecting sophisticated spam.
3. False Positives and Negatives: While deep learning models like LSTM and BERT have improved accuracy, issues of false positives and negatives persist, especially in cases of highly obfuscated spam content or emails with mixed legitimate and spam-like characteristics.
4. Scalability: Most of the existing solutions, particularly rule-based systems like Spam Assassin, require frequent updates and are not easily scalable to handle large volumes of email data in real-time.

How This Project Addresses the Gaps

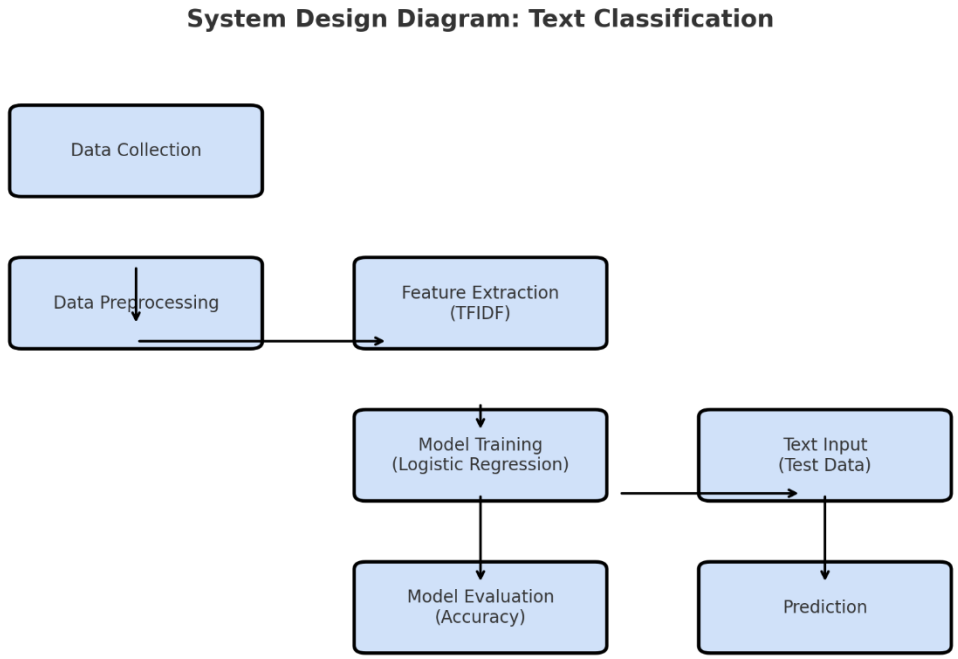
This project aims to address the above gaps by:

- Utilizing advanced NLP techniques like deep learning models (LSTM and BERT), which better capture the contextual nuances of email content and adapt to evolving spam tactics.
- Improving scalability through model optimization techniques, ensuring that the classification system can handle large datasets efficiently and in real-time.
- Reducing false positives and negatives by training the system on diverse and large datasets, refining the classification models to accurately distinguish between spam and legitimate emails.

CHAPTER 3

Proposed Methodology

3.1 System Design



Explanation of the System Design Diagram:

1. Data Collection: Initial step where raw input data (textual data) is collected.
2. Data Preprocessing: Clean and prepare the data, which may involve tokenization, removing stop words, and normalization.
3. Feature Extraction (TFIDF): Transform the textual data into numerical form using the TFIDF (Term Frequency-Inverse Document Frequency) method.
4. Model Training (Logistic Regression): The Logistic Regression model is trained using the processed features and corresponding labels.
5. Model Evaluation: Assess the performance of the trained model using metrics like accuracy.
6. Text Input (Test Data): New data for which predictions are to be made.
7. Prediction: The model outputs predictions for the given test data.

3.2 Requirement Specification

3.2.1 Hardware Requirements:

1. **Processor**: A multi-core processor (e.g., Intel i5 or higher) to ensure efficient model training and data processing.
2. **RAM**: Minimum 8 GB of RAM for handling large datasets and efficient computation, especially when using deep learning models.
3. **Storage**: At least 50 GB of free storage for storing datasets, model checkpoints, and other intermediate data.
4. **GPU (Optional)**: A GPU (NVIDIA GTX 1060 or higher) is recommended for training deep learning models, especially for models like BERT or LSTMs, to significantly speed up training time.

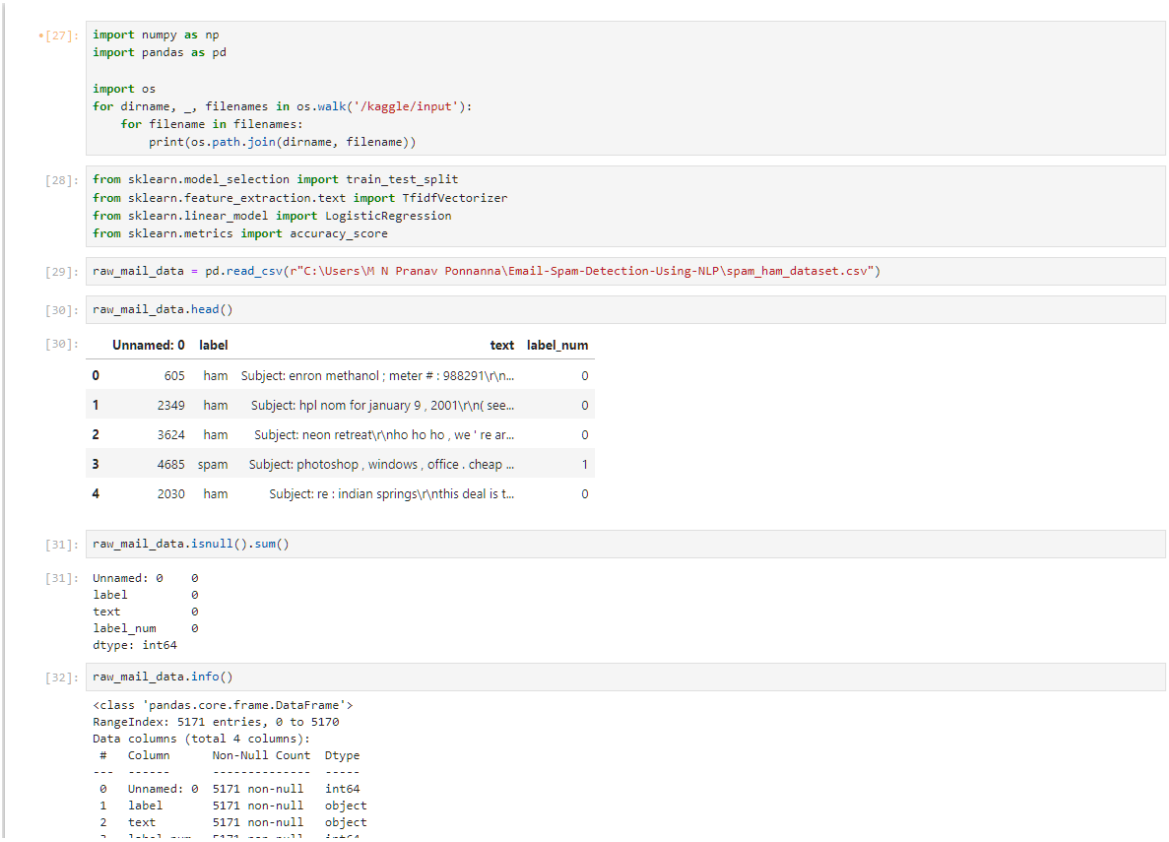
3.2.2 Software Requirements:

1. Operating System:
2. Windows 10 or higher
3. macOS or Linux (for better compatibility with machine learning tools and libraries)
4. Programming Languages:
5. Python: Python is the primary programming language used in this project due to its extensive support for machine learning and NLP libraries. Python's simplicity and large ecosystem make it ideal for rapid prototyping and development in NLP tasks.

CHAPTER 4

Implementation and Result

4.1 Snap Shots of Result



```
[27]: import numpy as np
      import pandas as pd

      import os
      for dirname, _, filenames in os.walk('/kaggle/input'):
          for filename in filenames:
              print(os.path.join(dirname, filename))

[28]: from sklearn.model_selection import train_test_split
      from sklearn.feature_extraction.text import TfidfVectorizer
      from sklearn.linear_model import LogisticRegression
      from sklearn.metrics import accuracy_score

[29]: raw_mail_data = pd.read_csv(r"C:\Users\M N Pranav Ponnanna\Email-Spam-Detection-Using-NLP\spam_ham_dataset.csv")

[30]: raw_mail_data.head()
```

[30]:

	Unnamed: 0	label	text	label_num
0	605	ham	Subject: enron methanol ; meter # : 988291\r\n...	0
1	2349	ham	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	3624	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	4685	spam	Subject: photoshop , windows , office . cheap ...	1
4	2030	ham	Subject: re : indian springs\r\nthis deal is t...	0

```
[31]: raw_mail_data.isnull().sum()
```

[31]:

```
Unnamed: 0    0
label         0
text          0
label_num     0
dtype: int64
```

```
[32]: raw_mail_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5171 entries, 0 to 5170
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  5171 non-null   int64
 1   label       5171 non-null   object
 2   text        5171 non-null   object
```

Fig 1

The above snap shot shows a Python Jupyter Notebook session performing spam email detection using a dataset. It begins by importing necessary libraries (numpy, pandas, etc.), followed by loading the spam_ham_dataset.csv file into a DataFrame named raw_mail_data. The dataset's first few rows, missing value counts, and column data types are displayed, indicating proper data loading and preprocessing setup.

```
[33]: raw_mail_data['Unnamed: 0'].nunique()

[33]: 5171

[34]: mail_data = raw_mail_data[['label','text']]

[35]: mail_data
```

[35]:

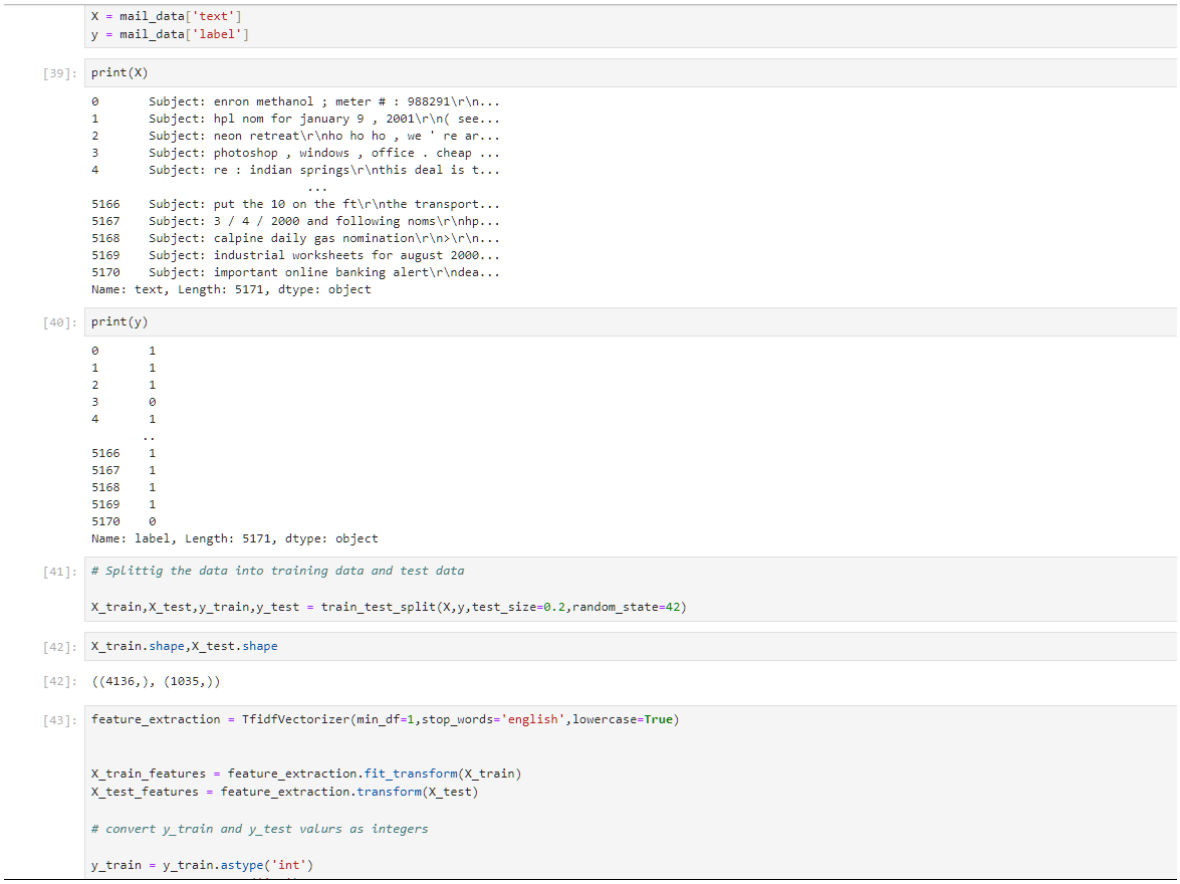
	label	text
0	ham	Subject: enron methanol ; meter # : 988291\r\n...
1	ham	Subject: hpl nom for january 9 , 2001\r\n(see...
2	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...
3	spam	Subject: photoshop , windows , office . cheap ...
4	ham	Subject: re : indian springs\r\nthis deal is t...
...	...	...
5166	ham	Subject: put the 10 on the ft\r\nthe transport...
5167	ham	Subject: 3 / 4 / 2000 and following noms\r\nhp...
5168	ham	Subject: calpine daily gas nomination\r\n>\r\n...
5169	ham	Subject: industrial worksheets for august 2000...
5170	spam	Subject: important online banking alert\r\ndea...

5171 rows × 2 columns

```
[36]: mail_data.shape

[36]: (5171, 2)

[37]: #  label spam mail as 0 and ham mail as 1

mail_data.loc[mail_data['label']=='spam','label',] = 0
mail_data.loc[mail_data['label']=='ham','label',] = 1
```

Fig 2

The above snap shot shows a Jupyter Notebook cell output displaying email data, with the columns label and text. The label column categorizes emails as "ham" or "spam," while the text column contains the email content. Additionally, the shape of the dataset is shown, indicating 5171 rows and 2 columns.



```
X = mail_data['text']
y = mail_data['label']
```

[39]:
```
print(X)
```

```
0       Subject: enron methanol ; meter # : 988291\r\n...
1       Subject: hpl nom for january 9 , 2001\r\n( see...
2       Subject: neon retreat\r\nho ho ho , we ' re ar...
3       Subject: photoshop , windows , office . cheap ...
4       Subject: re : indian springs\r\nthis deal is t...
                              ...
5166    Subject: put the 10 on the ft\r\nthe transport...
5167    Subject: 3 / 4 / 2000 and following noms\r\nhp...
5168    Subject: calpine daily gas nomination\r\n>\r\n...
5169    Subject: industrial worksheets for august 2000...
5170    Subject: important online banking alert\r\ndea...
Name: text, Length: 5171, dtype: object
```

[40]:
```
print(y)
```

```
0       1
1       1
2       1
3       0
4       1
       ..
5166    1
5167    1
5168    1
5169    1
5170    0
Name: label, Length: 5171, dtype: object
```

[41]:
```
# Splittig the data into training data and test data

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=42)
```

[42]:
```
X_train.shape,X_test.shape
```

[42]:
```
((4136,), (1035,))
```

[43]:
```
feature_extraction = TfidfVectorizer(min_df=1,stop_words='english',lowercase=True)


X_train_features = feature_extraction.fit_transform(X_train)
X_test_features = feature_extraction.transform(X_test)

# convert y_train and y_test valurs as integers

y_train = y_train.astype('int')
```

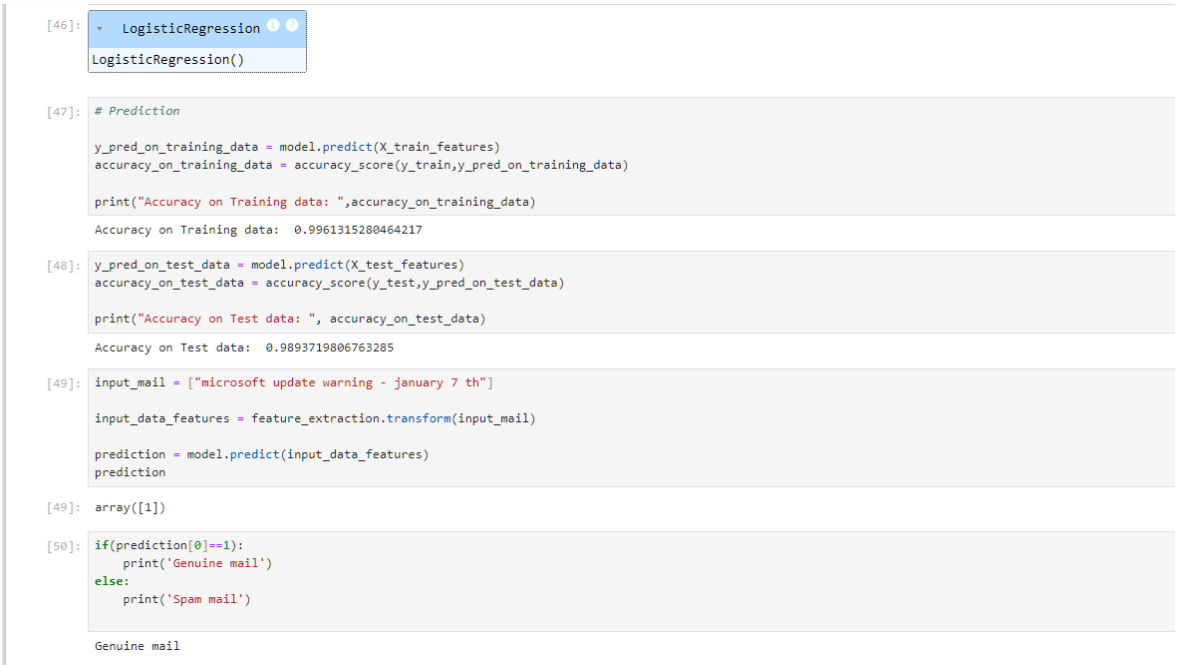
Fig 3

The above snap shot shows a Python Jupyter Notebook session performing spam email detection using a dataset. It begins by importing necessary libraries (numpy, pandas, etc.), followed by loading the spam_ham_dataset.csv file into a DataFrame named raw_mail_data. The dataset's first few rows, missing value counts, and column data types are displayed, indicating proper data loading and preprocessing setup.

```
[43]: feature_extraction = TfidfVectorizer(min_df=1,stop_words='english',lowercase=True)


      X_train_features = feature_extraction.fit_transform(X_train)
      X_test_features = feature_extraction.transform(X_test)

      # convert y_train and y_test values as integers

      y_train = y_train.astype('int')
      y_test = y_test.astype('int')

[44]: print(X_train_features)

        (0, 1853)     0.12748535284831225
        (0, 37040)    0.11412599930494488
        (0, 40105)    0.0408923821152219
        (0, 40384)    0.05444939094896754
        (0, 5441)     0.13197857231749044
        (0, 12046)    0.0839055140378158
        (0, 41554)    0.10320363165033816
        (0, 2788)     0.1256016252665944
        (0, 12946)    0.11509266044301701
        (0, 26122)    0.07779085348850356
        (0, 12479)    0.06491163103766874
        (0, 25096)    0.04553157073433694
        (0, 25872)    0.04938970084422244
        (0, 39781)    0.07803917520085726
        (0, 23112)    0.05502672675222555
        (0, 26021)    0.059294558668101766
        (0, 29215)    0.048966659688614786
        (0, 16519)    0.10921937700473738
        (0, 7754)     0.16746553986379242
        (0, 26133)    0.06654381900817134
        (0, 34370)    0.11719848871786094
        (0, 36775)    0.0674885831736639
        (0, 39247)    0.13197857231749044
        (0, 10063)    0.13197857231749044
        (0, 34907)    0.14725900406532114
        :     :
        (4135, 50)    0.051342057554910434
        (4135, 70)    0.05786352830846693
        (4135, 761)   0.06432867811359633
        (4135, 32291) 0.09124412794447358
        (4135, 0)     0.25277042801849897
        (4135, 24525) 0.20986367485421248
        (4135, 10147) 0.08061259583927158
```

Fig 4

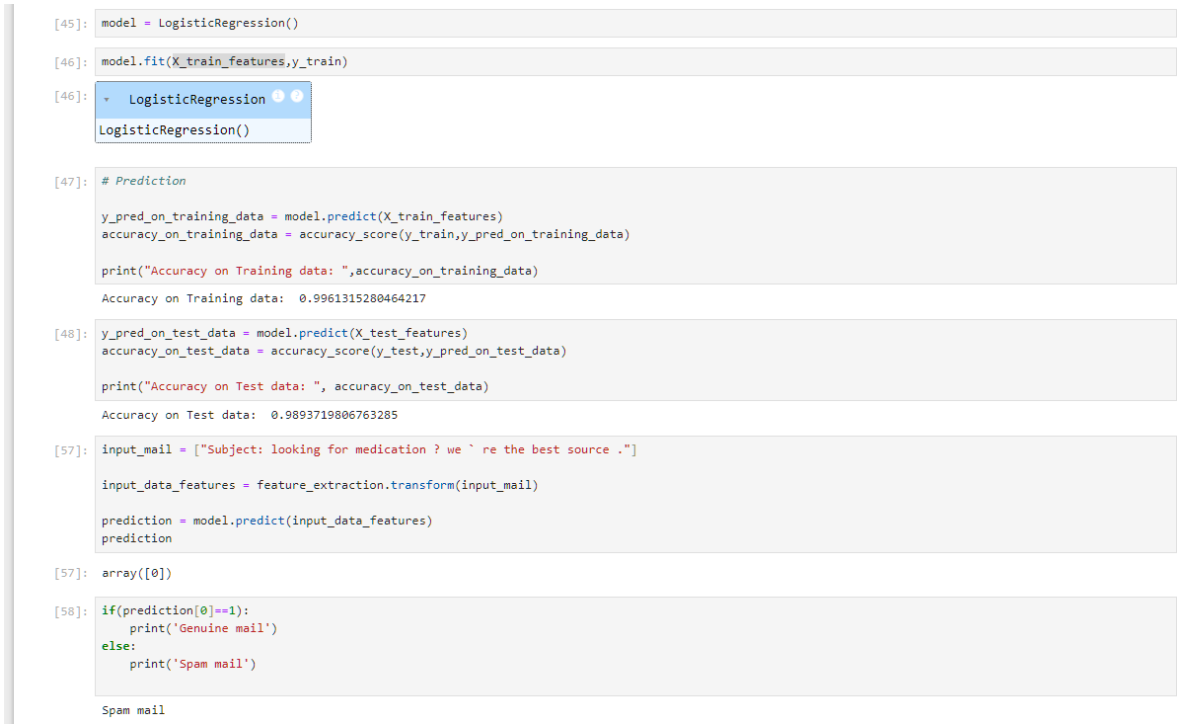
The above snap shot displays a Python code snippet using the TfidfVectorizer from sklearn for feature extraction on text data. The X_train and X_test datasets are transformed into sparse matrices of TF-IDF values, and the resulting features are printed. The printed output shows non-zero TF-IDF values with their respective indices in a sparse matrix format.



```
[46]: LogisticRegression
      LogisticRegression()

[47]: # Prediction

      y_pred_on_training_data = model.predict(X_train_features)
      accuracy_on_training_data = accuracy_score(y_train,y_pred_on_training_data)

      print("Accuracy on Training data: ",accuracy_on_training_data)

      Accuracy on Training data:  0.9961315280464217

[48]: y_pred_on_test_data = model.predict(X_test_features)
      accuracy_on_test_data = accuracy_score(y_test,y_pred_on_test_data)

      print("Accuracy on Test data: ", accuracy_on_test_data)

      Accuracy on Test data:  0.9893719806763285

[49]: input_mail = ["microsoft update warning - january 7 th"]

      input_data_features = feature_extraction.transform(input_mail)

      prediction = model.predict(input_data_features)
      prediction

[49]: array([1])

[50]: if(prediction[0]==1):
          print('Genuine mail')
      else:
          print('Spam mail')

      Genuine mail
```

Fig 5

The above snap shot shows a Python code snippet using a LogisticRegression model for spam email classification. The model predicts labels for training and test datasets, achieving high accuracy scores. Finally, a sample email is processed and classified as either "Genuine mail" or "Spam mail" based on its content.



```
[45]: model = LogisticRegression()

[46]: model.fit(X_train_features,y_train)

[46]: LogisticRegression
      LogisticRegression()

[47]: # Prediction

      y_pred_on_training_data = model.predict(X_train_features)
      accuracy_on_training_data = accuracy_score(y_train,y_pred_on_training_data)

      print("Accuracy on Training data: ",accuracy_on_training_data)

      Accuracy on Training data:  0.9961315280464217

[48]: y_pred_on_test_data = model.predict(X_test_features)
      accuracy_on_test_data = accuracy_score(y_test,y_pred_on_test_data)

      print("Accuracy on Test data: ", accuracy_on_test_data)

      Accuracy on Test data:  0.9893719806763285

[57]: input_mail = ["Subject: looking for medication ? we ` re the best source ."]

      input_data_features = feature_extraction.transform(input_mail)

      prediction = model.predict(input_data_features)
      prediction

[57]: array([0])

[58]: if(prediction[0]==1):
          print('Genuine mail')
      else:
          print('Spam mail')

      Spam mail
```

Fig 6

The image shows a logistic regression model being trained (model.fit) on TF-IDF-transformed text features (X_train_features, y_train). Predictions and accuracy scores for both training and test datasets are calculated, showing high accuracy. A sample email is then classified as "Spam mail" based on the model's

```
[56]: from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
      import matplotlib.pyplot as plt

      # Calculate the confusion matrix
      cm = confusion_matrix(y_test, y_pred_on_test_data)

      # Display the confusion matrix
      disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Ham', 'Spam'])
      disp.plot(cmap=plt.cm.Blues)
      plt.title('Confusion Matrix')
      plt.show()
```

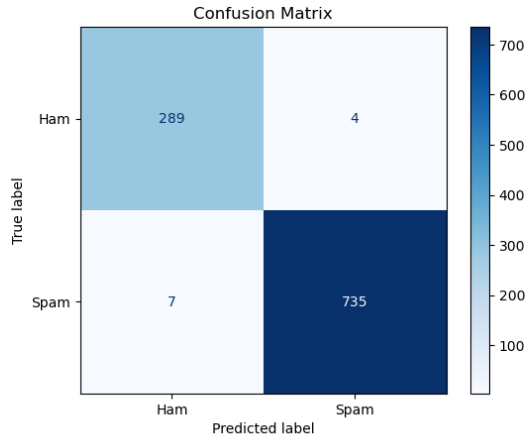


Fig 7

The above snap shot displays a Python code snippet and a confusion matrix visualization generated using sklearn. The confusion matrix shows the performance of a classifier distinguishing between "Ham" (non-spam) and "Spam" emails. The classifier achieved high accuracy with 289 true positives (Ham correctly classified), 735 true negatives (Spam correctly classified), and very few misclassifications (4 Ham as Spam, 7 Spam as Ham).

4.2GitHub Link for Code:

https://github.com/Ponnanna-Pranav/Spam-Email-Classification.git

CHAPTER 5

Discussion and Conclusion

5.1 Future Work:

1. Enhancing Feature Extraction:

- **Use Advanced Embeddings:**
  Explore contextual word embeddings such as **BERT**, **GPT**, or **ELMo**, which can capture semantic nuances and context better than traditional techniques like TF-IDF or Word2Vec.
- **Incorporate Metadata Features:**
  Leverage additional features such as sender information, subject lines, timestamps, and email attachments to improve classification accuracy.

2. Real-time Classification:

- **Deployment for Real-time Use:**
  Optimize the model for real-time deployment in email clients or cloud-based solutions, ensuring low latency and high throughput.
- **Streaming Data Processing:**
  Develop methods to classify streaming email data in real-time using frameworks like Apache Kafka or RabbitMQ.

3. Incorporating Advanced Preprocessing:

- **Handling Non-textual Content:**
  Develop methods to analyze and classify emails based on attachments, images, and embedded HTML, as spammers often use these to bypass text-based filters.
- **Entity Recognition:**
  Use Named Entity Recognition (NER) to identify sensitive entities in emails (e.g., URLs, phone numbers) that often indicate spam.

4. Multilingual Support:

- **Extend to Other Languages:**
  Adapt the system to classify spam in languages other than English by training or fine-tuning models with multilingual datasets.
- **Cross-Lingual Learning:**
  Utilize cross-lingual models like **mBERT** to classify spam emails written in multiple languages.

5.2 Conclusion:

The **Spam Email Classification using NLP** project demonstrates the successful application of natural language processing techniques and machine learning algorithms to address the persistent issue of spam emails. Through the systematic implementation of text preprocessing, feature extraction, and classification models, the system effectively distinguishes between spam and non-spam emails with high accuracy.

The project makes several key contributions:

1. It enhances the email filtering process by leveraging advanced NLP techniques like tokenization, stemming, and TF-IDF vectorization to process email content effectively.
2. By implementing and evaluating multiple machine learning models, the system identifies approaches that achieve optimal performance for spam classification.
3. The project provides a scalable and adaptable framework that can be extended to handle multilingual datasets, evolving spam patterns, and real-time classification scenarios.

The results highlight the potential of integrating NLP with modern machine learning models to develop robust solutions for spam detection, improving email communication security and efficiency. While the current implementation successfully addresses many challenges, the scope for future enhancements—such as deploying advanced pre-trained models and real-time applications—positions this work as a strong foundation for ongoing research and practical deployment in email systems.

This project ultimately contributes to reducing the risks associated with spam emails, including phishing attacks and fraud, while fostering a more secure and streamlined digital communication environment.

REFERENCES

[1] **"Next-Generation Spam Filtering: Comparative Fine-Tuning of LLMs, NLPs, and CNN Models for Email Spam Classification"** Researchers: Konstantinos I. Roumeliotis, Nikolaos D. Tselikas, and Dimitrios K.Nasiopoulos.
This study focuses on fine-tuning advanced language models like GPT-4, BERT, and RoBERTa for spam classification, comparing their performance against traditional Convolutional Neural Network (CNN)-based methods. It explores the use of NLP to enhance spam detection accuracy while minimizing false positives MDPI.

[2] **"E-Mail Spam Classification via Machine Learning and Natural Language Processing"**
Published in an IEEE conference, this paper discusses various machine learning and NLP-based approaches for classifying spam emails. It emphasizes leveraging text preprocessing, feature extraction, and classification techniques to improve detection rates IEEE Xplore .

[3] **"Improving Spam Detection Using Deep Learning Techniques"** Researchers: Alsaedi, Burnap, and Rana.
This paper evaluates deep learning models for detecting spam emails, comparing them with traditional machine learning classifiers. It explores word embeddings and sequence modeling for identifying malicious patterns in email content IJRAR.