SPAM DETECTION

A PROJECT REPORT

Submitted by

Ponnuri Aniruddha[RA2112704010015]

Gadde Vamshi[RA2112704010017]

Y Shabanya Kishore[RA2112704010018]

Under the Guidance of

Dr. M. Lakshmi

(Professor and Head, Department of Data Science and Business Systems)

In partial fulfillment of the Requirements for the Degree of

MASTERS OF TECHNOLOGY (INTEGRATED)

COMPUTER SCIENCE WITH DATA SCIENCE

DEPARTMENT OF DATA SCIENCE AND BUSINESS SYSTEMS

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

NOVEMBER 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR-603203

BONAFIDE CERTIFICATE

Certified that this project report titled **"SPAM DETECTION"** is the bonafide work of **"PONNURI ANIRUDDHA[RA2112704010015] GADDE VAMSHI[RA2112704010017] Y SHABANYA KISHORE[RA2112704010018]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

Dr. M. Lakshmi
GUIDE
Dr. Mythili
CO-GUIDE
Dept. of DSBS

Dr. M. Lakshmi
HEAD OF THE DEPARTMENT
Dept. of DSBS

Signature of Internal Examiner

Signature of External Examiner

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

GUI Graphical User Interface

EDA Exploratory data analysis

NLP Natural language processing

TF-IDF Term Frequency-Inverse Document Frequency

ROC Receiver Operating Characteristic

ABSTRACT

This code presents a comprehensive approach to spam classification in SMS messages, utilizing natural language processing (NLP) techniques and machine learning algorithms. The process begins with the exploration and preprocessing of a dataset, involving data cleaning, tokenization, removal of special characters and stop words, and stemming. Exploratory data analysis (EDA) techniques are applied to understand the distribution of the dataset. Various machine learning models including Logistic Regression, Naive Bayes, Support Vector Machines, Random Forest, and K-Nearest Neighbors are implemented and evaluated for their effectiveness in classifying spam and non-spam messages. The optimal model, a Support Vector Machine, is selected based on its superior performance. Additionally, the code includes the analysis of the impact of different K values on the accuracy of the K-Nearest Neighbors model. The final chosen model is saved along with the TF-IDF vectorizer for future use. The study provides valuable insights into the application of NLP and machine learning for text classification tasks, specifically focusing on spam detection in SMS messages

ACKNOWLEDGEMENTS

We express our humble gratitude to **Dr C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support. We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, **Dr T.V.Gopal**, for his invaluable support.

We wish to thank **Dr Revathi Venkataraman**, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work. We are incredibly grateful to our Head of the Department**, Dr M. Lakshmi** Professor, Department of Data Science and Business Systems, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We register our immeasurable thanks to our Faculty Advisor**, Dr. Shantha Kumari,** Assistant Professor, DSBS, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to my guide, **Dr. Mythili,** Guide, DSBS, SRM Institute of Science and Technology, for providing me with an opportunity to pursue my project under his mentorship. He provided me with the freedom and support to explore the research topics of my interest. His passion for solving problems and making a difference in the world has always been inspiring. We sincerely thank the Data Science and Business Systems staff and students, SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank parents, family members, and friends for their unconditional love, constant support, and encouragement.

Ponnuri aniruddha

Gadde vamshi

Y shabanya kishore

CHAPTER 1
INTRODUCTION

In the ever-expanding digital landscape, where communication is paramount, distinguishing between legitimate messages and unsolicited spam is of utmost importance. Welcome to the realm of our Spam Detection Model, a sophisticated solution meticulously designed to combat the onslaught of unwanted messages and safeguard seamless communication.

Understanding the Problem:

Spam messages inundate our inboxes, posing threats ranging from phishing schemes to information overload. Recognizing this challenge, we have meticulously crafted a robust machine learning model capable of discerning between 'ham' (legitimate) and 'spam' messages with unparalleled accuracy.

Our Approach:

Our journey begins with Data Cleaning, ensuring the dataset is pristine and devoid of redundancies. We delve into Exploratory Data Analysis (EDA), unravelling insights into the characteristics of spam and ham messages. Rigorous Text Preprocessing follows, where we transform raw text into a format suitable for machine learning algorithms. Through meticulous data visualization, including word clouds and frequency analysis, we gain deep insights into the textual data.

The Heart of the Model:

At the core of our solution lies a powerful ensemble of machine learning techniques, including Gaussian Naive Bayes, Multinomial Naive Bayes, and Bernoulli Naive Bayes. These algorithms are trained and evaluated meticulously, ensuring the model's robustness and accuracy.

Empowering Decision-Making:

Our Spam Detection Model is not just about accurate predictions; it's about empowering users and businesses to make informed decisions swiftly and efficiently. With the model's ability to process and analyze vast amounts of textual data, it stands as a stalwart guardian against unsolicited messages, enabling smoother communication channels and enhancing productivity.

1.1 GENERAL PURPOSE:

The goal of this project is to design, implement and deploy a state-of-the-art spam detection system that combines the strengths of machine learning algorithms and NLP techniques. The system aims to provide proactive protection against spam in various communication channels and provide users with a smooth and secure user experience. Using Naive Bayes, Logistic Regression, K-means clustering, SVM and NLP, the goal is to achieve high accuracy in distinguishing between spam and legitimate messages. The project aims to respond to the growing demand for adaptive and efficient spam detection mechanisms that contribute to improving the security and satisfaction of the users of digital communications. Ultimately, the goal is to create a robust, scalable, real-time system that effectively responds to the multifaceted challenges of spam in the digital age.

1.2 SCOPE:

The scope of this project includes the development and implementation of a comprehensive spam detection system that uses a combination of machine learning algorithms, including Naive Bayes, logistic regression, K-means clustering, support vector machines (SVM), and natural language processing (NLP).) techniques. . The main goal is to create a robust and customizable system that can effectively detect and filter spam messages from various communication channels, such as emails or text messages. The project begins with an extensive data collection phase, which involves obtaining identified datasets from reliable sources that contain both spam and non-spam. This data is the basis for training and evaluating machine learning models. Rigorous curation and pre-processing are applied to ensure data quality, including techniques such as text normalization, punctuation removal and use of unbalanced categories. Feature extraction is central, as it uses NLP techniques such as Bag of Words, TF-IDF, and word embedding to transform textual content into numerical features suitable for model training. In addition, the possibilities of K-means clustering are explored to identify patterns and group similar messages, which contributes to a more comprehensive understanding of the data. The project involves training and evaluating Naive Bayes, Logistic Regression and SVM models for their effectiveness in distinguishing between spam and non-spam. Each algorithm is tuned and optimized to achieve the best possible performance, taking into account metrics such as accuracy, precision, recall, F-measure and AUC-ROC. In addition, the integration of NLP techniques aims to improve the systematicity and understanding of the semantic relationships and

contextual nuances of the text, which improves the overall accuracy of the text. The scope of the project also includes the use of K-means clustering for unsupervised learning, exploring ways to group similar messages without specific identifiers.

The application is developed using appropriate programming languages and frameworks, ensuring scalability, efficiency and real-time capabilities for practical implementation. Continuous monitoring and performance evaluation is integrated, allowing for updates and adjustments to new spam tactics. Its broad scope aims to provide a state-of-the-art spam detection system that combines the strengths of various machine learning algorithms and NLP techniques to improve accuracy and adaptability.

1.3 PROBLEM STATEMENT:

The spread of spam remains a critical issue in digital communications, requiring effective and adaptive spam detection mechanisms. Traditional approaches often cannot keep up with modern spam tactics, increasing the risk of false positives and negatives. The challenge is to develop a robust and efficient spam detection system that uses machine learning algorithms such as Naive Bayes, logistic regression, K-means clustering, support vector machines (SVM), and natural language processing (NLP) to accurately distinguish between spam and legitimate messages. The system must be able to handle different communication channels and evolving spam patterns, ensuring effective and reliable protection against unwanted and potentially harmful content.

1.4 MOTIVATION:

The motivation behind this project is to respond to the ongoing and evolving challenge of spam across platforms such as email and text messaging. The spread of spam not only hinders the user experience, but also creates security risks and weakens the effectiveness of digital communication. By developing an improved spam detection system, we want to promote a safer and smoother online environment, increase user trust and confidence in digital communication channels.

1.5 NEED:

The need for an advanced spam detection system is critical to protect users' communication channels from the relentless and ever-evolving threat of spam. Traditional filters often fail to accurately detect hue patterns, causing inconvenience to users and potential security breaches. An advanced spam detection system that integrates multiple machine learning algorithms and NLP techniques responds to the need for a comprehensive and adaptive solution. Such a system not only improves the accuracy of spam detection, but also contributes to the overall cyber security landscape and promotes a more secure and reliable digital communication environment.

1.6 ORGANIZATION OF THE THESIS:

This thesis is organized into distinct chapters to provide a systematic exploration of the Spam detection project.

Chapter 1 - Introduction: This chapter presents the research problem, objectives, scope, motivation, and current Spam problems. It also describes the thesis's structure.

Chapter 2 - Literature Review: This section undertakes a detailed assessment of existing literature relating to Spam/Ham detection, including both traditional methods and new advances in machine learning. It serves as the foundation for the project's methodologies.

Chapter 3 - Existing Work: A review of previous spam detection work provides context and benchmarks for assessing the proposed methodology.

Chapter 4 - Methodology: In this chapter, the methodology employed Naive Bayes, Logistic Regression, K-means, SVM, and NLP. The dataset is pre-processed and enriched before training models that take into account all words and enriched features.

Chapter 5 - Dataset Findings: The spam detection dataset exploration reveals insights into the distribution and characteristics of spam and non-spam messages.

Chapter 6 - Feature Extraction Methods: The feature extraction process converts textual content into numerical features suitable for machine learning by utilizing Bag of Words, TF-IDF, and NLP techniques.

Chapter 7 -Proposed Work: The proposed work aims to improve spam detection accuracy by combining various feature extraction methods and machine learning algorithms. The performance of various models is thoroughly analysed. It includes detailed discussions on accuracy, precision, recall, and F1-score for each feature-extraction and classifier combination.

Chapter 8 - Model Implementation: Models are built and fine-tuned, and their performance is measured with metrics like accuracy, precision, recall, and AUC.

Chapter 9 - Conclusion: The study concludes with a summary of the proposed methodology's effectiveness in spam detection. It also discusses limitations and areas for future improvement.

Chapter 10 - Future Enhancements: Advanced NLP techniques, ensemble learning, and real-time detection mechanisms could be added in the future. This chapter suggests potential improvements to Spam detection models.

References: This section contains a list of all the references used in the thesis, ensuring academic rigor and integrity. References to relevant literature and sources that back up the methodology and findings.

Appendices: For a deeper understanding, the appendices contain extra details, code snippets, and supplementary information.

CHAPTER 2
LITERATURE STUDY

1. Email Spam Detection Using Machine Learning Algorithms

Author:

1. Nikhil Kumar
2. Sanket Sonowal
3. Nishant

Inference:

Creating a fake profile and email account is much easy for the spammers, they pretend like a genuine person in their spam emails, these spammers target those peoples who are not aware about these frauds. So, it is needed to Identify those spam mails which are fraud, this project will identify those spam by using techniques of machine learning, this paper will discuss the machine learning algorithms and apply all these algorithm on our data sets and best algorithm is selected for the email spam detection having best precision and accuracy.

2. Machine Learning Techniques for Spam Detection in Email and IoT Platforms: Analysis and Research Challenges

Author:

1. Naeem Ahmed 2. Rashid Amin
3. Hamza Aldabbas. 4. Deepika Koundal
5. Bader Alouffi 6. Tariq Shah

Inference:

This paper surveys the machine learning techniques used for spam filtering techniques used in email and IoT platforms by classifying them into suitable categories. A comprehensive comparison of these techniques is also made based on accuracy, precision, recall, etc. In the end, comprehensive insights and future research directions are also discussed.

3. An Efficient Spam Detection Technique for IoT Devices Using Machine Learning

Author:

1.David Halbhuber

2. Jakob Fehle

3. Alexander Kalus

Inference

In this article they propose the security of the IoT devices by detecting spam using ML. To achieve this objective, Spam Detection in IoT using Machine Learning framework is proposed. In this framework, five ML models are evaluated using various metrics with a large collection of inputs features sets. Each model computes a spam score by considering the refined input features. This score depicts the trustworthiness of IoT device under various parameters. REFIT Smart Home data set is used for the validation of proposed technique. The results obtained proves the effectiveness of the proposed scheme in comparison to the other existing schemes.

4. Analysis of Optimized Machine Learning and Deep Learning Techniques for Spam Detection

Author:

1. Fahima Hossain
2. Mohammed Nasir Uddin
3. Rajib Kumar Halder

Inference:

The proposed model is implemented in both machine learning and deep learning to establish a comparative analysis. Multinomial Naïve Bayes (MNB), Random Forest (RF), K-Nearest Neighbor (KNN), Gradient Boosting (GB) are used to introduce ensemble method in machine learning implementation. Recurrent Neural Network (RNN), Gradient Descent

(GD), Artificial Neural Network (ANN) for deep learning implementation. An ensemble method is constructed to combine multiple classifiers' output. The ensemble methods allow producing better prediction accuracy compared to a single classifier. Our proposed model obtained an accuracy of 100%, AUC=100, MSE error = 0 and RMSE error = 0 for machine learning implementation and accuracy of 99%, loss value= 0.0165 for deep learning implementation based on an email spam base dataset collected from the UCI machine learning repository.

5. Email Spam Detection using Deep Learning Approach

Author:

1. Kingshuk Debnath
2. Nirmalya Kar

The motivation of this research is to build email spam detection models by using machine learning and deep learning techniques so that spam emails can be distinguished from legitimate emails with high accuracy. The Enron email dataset has been used and deep learning models are developed to detect and classify new email spam using LSTM and BERT. NLP approach was applied to analyze and perform data preprocessing of the text of the email. The results are compared to the previous models in email spam detection. The proposed deep learning approach obtained the highest accuracy of 99.14% using BERT, 98.34% using BiLSTM and 97.15% using LSTM. Python is utilized for all implementations.

CHAPTER 3

EXISTING WORK

Email spam, commonly known as sending unsolicited or promotional emails to a group of recipients, is a common problem. Unsolicited emails are emails for which the recipient has not been explicitly authorized. Over the past decade, spam rates have increased significantly, posing significant challenges for Internet users. Spam is harmful for many reasons:
it consumes storage space, hinders email system efficiency, and wastes recipients' time.
Historically, automated email filtering systems have been the primary means of identifying and managing spam. Nevertheless, contemporary spammers have developed sophisticated techniques to bypass these filters. While in the past, many spam emails could be manually blocked based on their origins or specific email addresses, this method is no longer as effective due to evolving spamming tactics.

To combat this ever-growing problem, the adoption of machine learning approaches for spam detection has gained prominence. These approaches encompass various strategies, including text analysis, white and blacklists of domain names, and community-based techniques. Text analysis, which evaluates the content of email messages, is widely utilized to discern spam from legitimate communications. Solutions can be deployed on both the server side and the client side. The Naive Bayes algorithm is an important option in these processes. However, content-based analysis can be difficult because it can lead to false positives, i.e., legitimate messages being misclassified as spam.

The blacklist approach involves accepting all emails except those from blacklisted domains or email ids. As spammers continuously create new domains, this method becomes less effective. In contrast, whitelisting allows emails to be sent from pre-approved domains or addresses while requiring others to go through a secondary authentication process, ensuring their legitimacy.

Spammers have evolved and now use popular social media tools to target specific user segments, review sites or fan pages, often embedding links Links hidden in text lead to pornographic or fraudulent product websites. These malicious emails share common characteristics, which improves detection by analyzing these patterns.

Artificial intelligence (AI) plays an important role in classifying emails as spam or non-spam. By extracting features from the title, subject, and email content, AI can group

messages into appropriate categories. Learning-based classification assumes that spam emails have unique characteristics that distinguish them from legitimate correspondence. However, spam recognition in learning-based models is complicated by factors such as subjectivity, conceptual variation, linguistic variation, processing overhead, and latency in text analysis.

A notable example of a learning-based model is the Extreme Learning Machine (ELM). This modern machine learning model, characterized by a single hidden layer in a feedback neural network, overcomes the problems associated with slow learning and overfitting commonly found in traditional neural networks. ELM has generalizability potential, high robustness, and superior controllability, making it a popular choice in a variety of fields.

In this context, the study explores a series of machine learning algorithms for spam detection, delving into their architecture, advantages and disadvantages. The article also looks at the basic characteristics of spam emails. By conducting a comprehensive assessment of existing techniques and the nature of spam, the study identifies research gaps and presents open research problems as well as future directions to enhance security email and improve spam filtering using machine learning methods.

Various challenges facing current spam filtering models and their implications for the effectiveness of the models are discussed. This study provides an in-depth comparison of machine learning techniques and presents a comprehensive taxonomy of spam detection methods based on machine learning principles. Additionally, this study explores potential directions for future research on spam detection and filtering to improve the accuracy and security of messaging platforms.

CHAPTER 4
METHODOLOGY

4.1 Data Collection

In the data collection phase, diverse datasets are acquired from reputable sources that specialize in spam datasets. These datasets consist of labeled emails, where each email is categorized as either "spam" or "non-spam" (ham). The goal is to ensure a balanced representation of both spam and non-spam instances, promoting diversity in the dataset. Rigorous curation processes are applied to ensure the quality and reliability of the data. Metadata features such as sender information, date of the email, and email length are collected in addition to the textual content, enriching the dataset with valuable context.

4.2 Data Cleaning

The dataset undergoes a meticulous data cleaning process to prepare it for analysis. NLP techniques are applied to process the text, including converting all text to lowercase to ensure consistency. Punctuation marks are removed to focus on the semantic content of the text. Common English stopwords are eliminated to reduce noise in the dataset.

4.3 Feature Extraction

Feature extraction is a crucial step in transforming the pre-processed text into numerical features suitable for machine learning models. Three feature extraction techniques are employed: Bag of Words: This method represents the text as a matrix of word occurrences, capturing the frequency of each word in the dataset. Word Embeddings (Word2Vec): This technique captures the semantic meaning of words by representing them as dense vectors in a continuous vector space.TF-IDF (Term Frequency-Inverse Document Frequency): This method evaluates the importance of a word in a document relative to its frequency across the entire dataset. Additionally, the dataset undergoes feature enrichment, where top uni-grams with high odds ratios in spam vs. non-spam and vice versa are identified. These enriched features contribute valuable information for the models to differentiate between spam and non-spam emails.

4.4 Feature Enrichment

To augment the discriminatory power of the features, a log-ratio based feature enrichment process is applied. This involves calculating odds ratios for each uni-gram and selecting the top features that contribute significantly to the differentiation between spam and non-spam emails. Enriched features aim to capture nuanced patterns, enhancing the model's ability to distinguish between the two classes effectively.

4.5 Model Training

During the model training phase, we use a variety of machine learning methods for spam identification, such as Naive Bayes, Logistic Regression, K-means, SVM, and NLP. The training procedure uses a highly pre-processed and enriched spam detection dataset to explore two scenarios: considering all terms and incorporating only enriched words using log-ratio analysis. This guarantees that the models have a thorough comprehension of the dataset and can learn distinguishing features. During training, each algorithm is fine-tuned by optimizing parameters to maximum performance. Model efficacy is assessed using evaluation metrics such as accuracy, precision, recall, F-measure, and AUC. The training method makes use of 80% of the dataset, ensuring a rich learning experience and improving the models' flexibility to various spam patterns..

4.6 Testing

During the model training phase, we use a variety of machine learning methods for spam identification, such as Naive Bayes, Logistic Regression, K-means, SVM, and NLP. The training procedure uses log-ratio analysis to explore two situations utilizing a highly pre-processed and enriched spam detection dataset. This guarantees that the models have a thorough comprehension of the dataset and can learn distinguishing features. During training, each algorithm is fine-tuned by optimizing parameters to maximum performance. Model efficacy is assessed using evaluation metrics such as accuracy, precision, recall, F-measure, and AUC. The training method makes use of 80% of the dataset, ensuring a rich learning experience and improving the models' flexibility to various spam patterns.

4.7 Data Analysis

The analysis phase involves a detailed examination of the impact of enriched features on model performance. Both Naive Bayes and Logistic Regression models are subjected to thorough evaluation using the enriched features. Statistical significance and generalizability of the results are carefully examined to gain insights into the effectiveness of the models in distinguishing between spam and non-spam emails.

4.8 Performance Comparison

The performance of the implemented models is compared against baseline models that do not incorporate enriched features. This comparative analysis aims to provide insights into the practical significance of log-ratio based feature enrichment on the models' predictive capabilities. Key metrics are evaluated, and statistical tests may be conducted to identify improvements and assess the reliability and accuracy enhancements introduced by the enriched features.

CHAPTER 5

DATASET FINDINGS

SMS Spam Collection Dataset:

The files contain one message per line. Each line is composed by two columns: v1 contains the label (ham or spam) and v2 contains the raw text.

The SMS Spam Collection is a set of SMS tagged messages that have been collected for SMS Spam research. It contains one set of SMS messages in English of 5,574 messages, tagged acording being ham (legitimate) or spam.

This corpus has been collected from fee or free for research sources at the Internet:

A collection of 425 SMS spam messages was manually extracted from the Grumbletext Web site. This is a UK forum in which cell phone users make public claims about SMS spam messages, most of them without reporting the very spam message received. The identification of the text of spam messages in the claims is a very hard and time-consuming task, and it involved carefully scanning hundreds of web pages. A subset of 3,375 SMS randomly chosen ham messages of the NUS SMS Corpus (NSC), which is a dataset of about 10,000 legitimate messages collected for research at the Department of Computer Science at the National University of Singapore. The messages largely originate from Singaporeans and mostly from students attending the University. These messages were collected from volunteers who were made aware that their contributions were going to be made publicly available. A list of 450 SMS ham messages collected from Caroline Tag's PhD Thesis Finally, we have incorporated the SMS Spam Corpus v.0.1 Big. It has 1,002 SMS ham messages and 322 spam messages

"v1" (Target Variable):

Type: Categorical

Description: Indicates whether a text message is "ham" (non-spam) or "spam."

"v2" (Feature Variable):

Type: Text

Description: Contains the text content of the messages.

CHAPTER 6

FEATURE EXTRACTION METHODS

Bag of Words (BoW):

Description: Bag of Words is a fundamental technique that represents a document as an unordered set of words, focusing on the frequency of each word while disregarding grammar and word order.

Implementation: Create a vocabulary by collecting all unique words in the entire dataset. Represent each document as a vector, where each element corresponds to the frequency of a word in the document.

Term Frequency-Inverse Document Frequency (TF-IDF):

Description: TF-IDF is a statistical measure that evaluates the importance of a word in a document by considering its frequency within that document and inversely proportional to its frequency across the entire dataset.

Implementation: Calculate the TF-IDF score for each word in each document. The resulting vectors represent the importance of words in distinguishing between spam and non-spam.

N-grams:

Description: N-grams are sequences of n items (words or characters) in a text, providing a way to capture local word patterns and relationships between adjacent words.

Implementation: Generate features based on uni-grams, bi-grams, or tri-grams by considering single words, pairs, or triples of consecutive words in the text, respectively.

Character-level Features:

Description: Analyzing text at the character level can capture patterns related to the use of specific characters or sequences, which is especially useful when dealing with intentionally misspelled words in spam.

Implementation: Features may include character n-grams or binary indicators for the presence of specific characters or character sequences, helping to detect characteristic patterns in spam.

Enriched Features with Odds Ratios:

Description: Identify uni-grams with high odds ratios in spam vs. non-spam and vice versa to create enriched features that contribute significantly to the differentiation between the two classes.

Implementation: Calculate odds ratios for each word, considering their frequency in spam and non-spam instances. Select the top features with the highest odds ratios to enrich the dataset with valuable information for the models.

CHAPTER 7

PROPOSED WORK

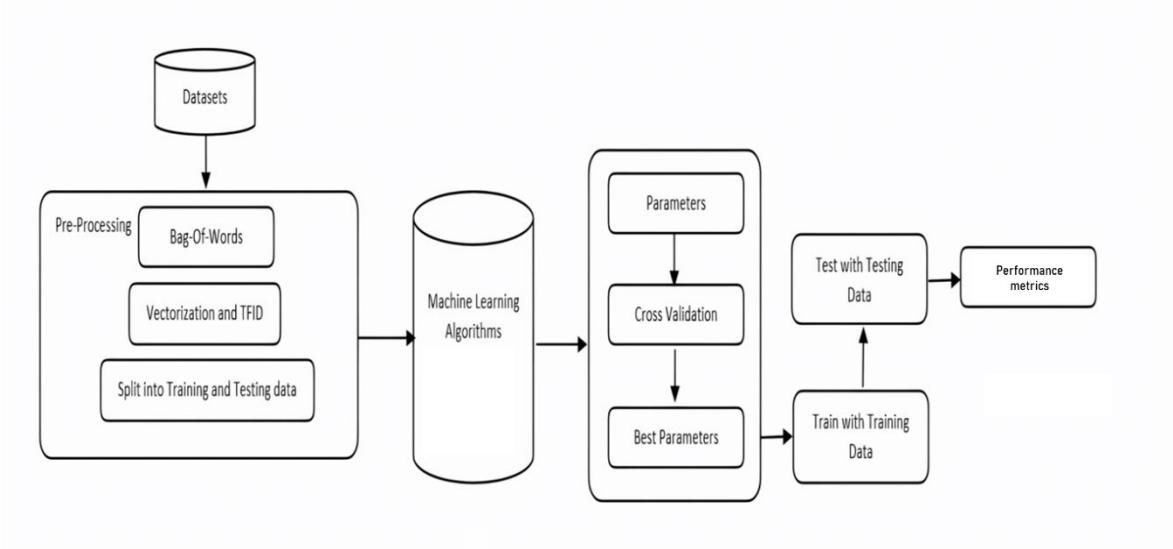

Fig 7.1:

Machine Learning (ML) plays a pivotal role in spam detection due to its ability to analyze vast amounts of data and identify patterns that can be used to distinguish spam from legitimate content. ML algorithms, such as Naive Bayes, Support Vector Machines (SVMs), and deep learning models, have proven highly effective in automating the process of classifying messages as spam or not. ML-based spam detection systems continuously learn and adapt to evolving spamming techniques, ensuring that users are protected from unwanted and potentially harmful messages.

Natural Language Processing (NLP):

- NLP is a field of artificial intelligence (AI) that focuses on the interaction between computers and human language. It involves tasks like text analysis, speech recognition, language generation, and sentiment analysis.
- NLP techniques are used to process and understand human language, making it valuable in applications like chatbots, translation services, and text summarization.
- NLP algorithms analyze the linguistic features of messages, considering factors such as word choice, sentence structure, and language patterns. By examining the text, NLP can identify common characteristics associated with spam, such as the use of certain keywords or the presence of misleading language.

- NLP models are trained to recognize patterns indicative of spam behavior. This includes identifying typical structures or phrases commonly used in spam messages. The ability to recognize patterns allows the system to adapt to new variations of spam that may emerge over time.

K-Nearest Neighbors (KNN):

- KNN is a simple machine learning algorithm used for classification and regression tasks.
- In KNN, data points are classified based on the majority class among their k-nearest neighbours in the feature space.
- It's a non-parametric algorithm, meaning it doesn't make assumptions about the underlying data distribution.
- KNN is straightforward to understand and implement but may not perform well on high-dimensional data or when the dataset is imbalanced.

Naive Bayes:

- Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem. It's called "naive" because it assumes that the features used for classification are conditionally independent, which is often a simplification.

- Naive Bayes is commonly used for text classification tasks, like spam detection and sentiment analysis. Despite its simplicity, Naive Bayes can perform surprisingly well in many real-world applications.

- Naive Bayes is a probabilistic algorithm based on Bayes' theorem. It assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. In the context of spam detection, it treats each word or feature independently, which is a naive assumption but often works well in practice.

- The algorithm is trained on a labeled dataset, learning the probability of each word or feature given whether an email is spam or not.

- When a new email arrives, the algorithm calculates the probability that it belongs to the spam or non-spam class based on the occurrence of individual words. The email is then classified into the class with the higher probability.

Logistic Regression:

- Logistic Regression is a statistical model used for binary classification tasks. It models the probability that an instance belongs to a particular class using the logistic function. Logistic Regression is a linear algorithm, making it interpretable and easy to implement.In spam detection, it estimates the probability that an email is spam.

- The algorithm is trained by optimizing a logistic function based on the input features (words in the case of spam detection) and corresponding labels (spam or non-spam).

- For a new email, the model calculates the probability of it being spam using the learned weights. If the probability is above a certain threshold (commonly 0.5), the email is classified as spam; otherwise, it's classified as non-spam.

CHAPTER 8

MODEL IMPLEMENTATION AND VISUALIZATION

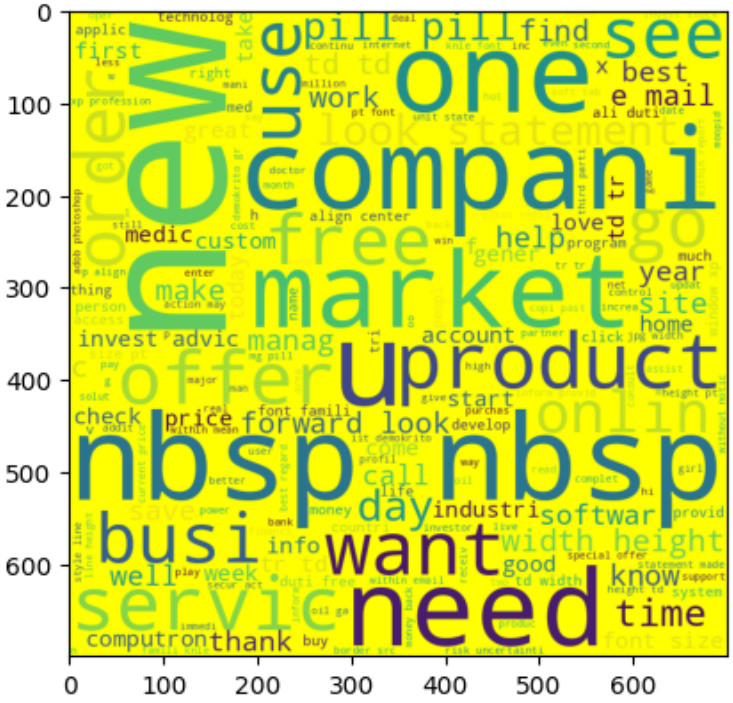

Fig 9.1: Word Cloud for Spam Words

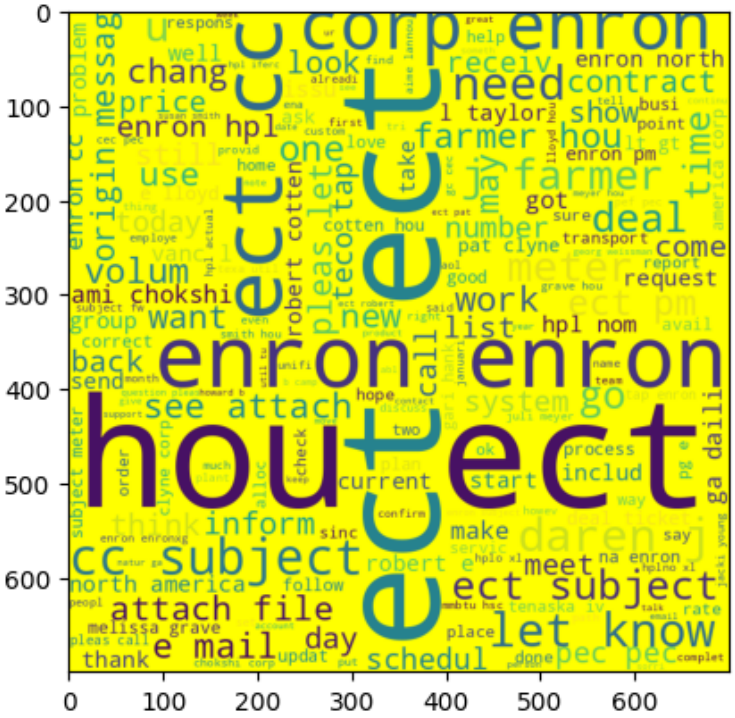

Fig 9.2: Word Cloud for HAM words

The words clouds show the most frequently occurring words in the dataset for spam and ham words.

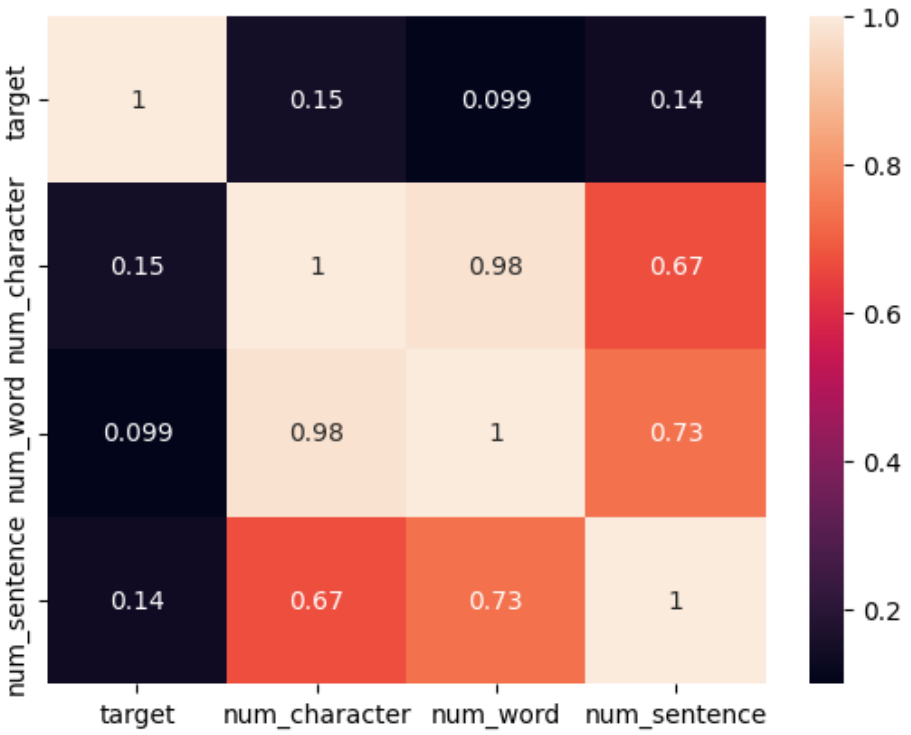

Fig 9.3: Covariance Matrix

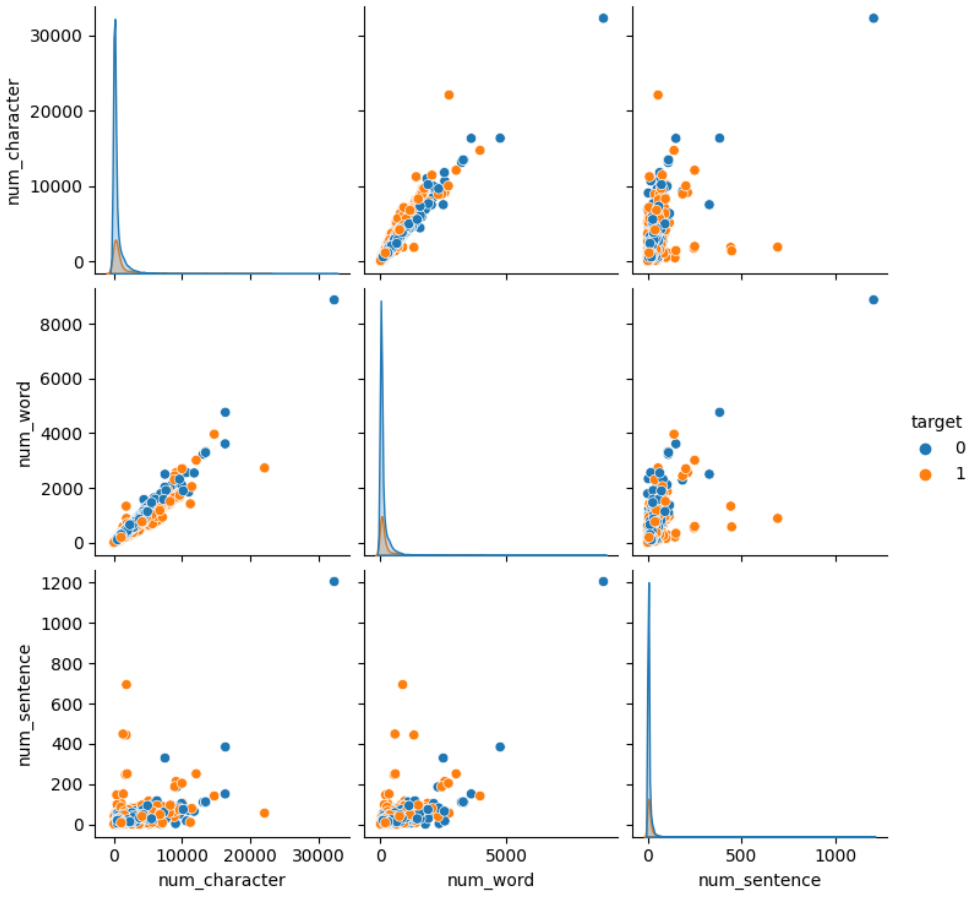

Fig 9.4: Multivariate Analysis

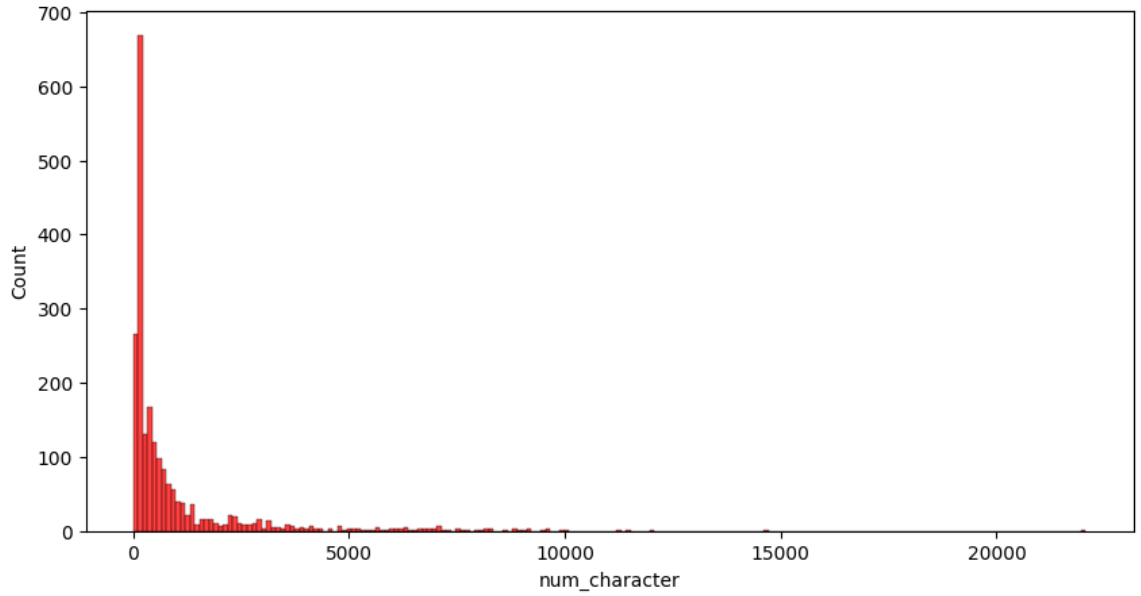

Fig 9.5: No. of Character vs Their Count in Spam Words

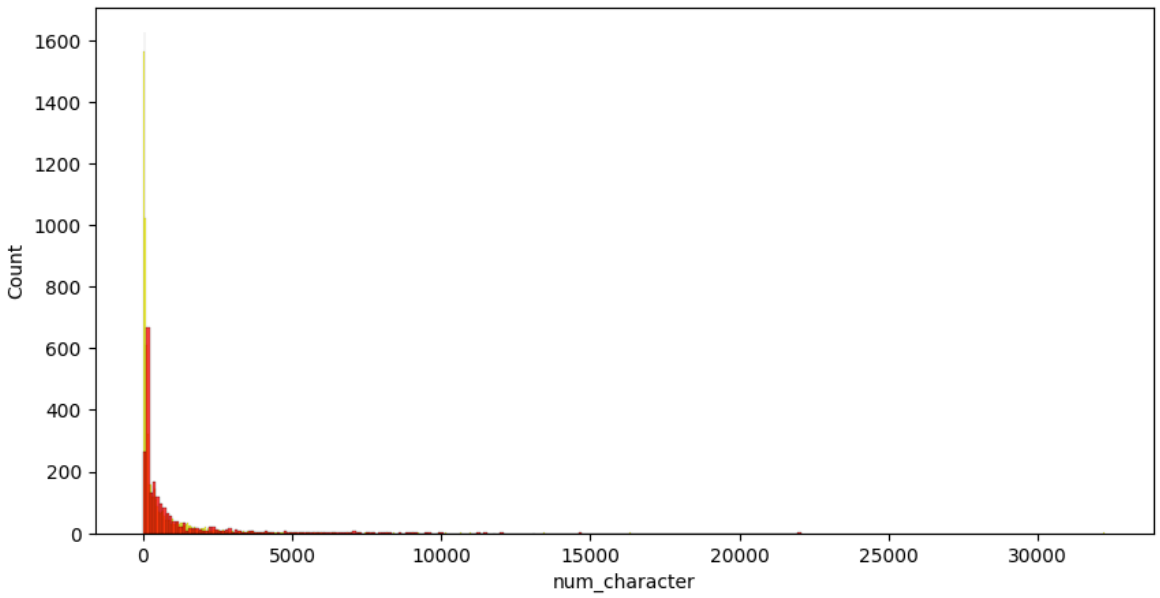

Fig 9.6: No. of Character vs Their Count in Ham Words

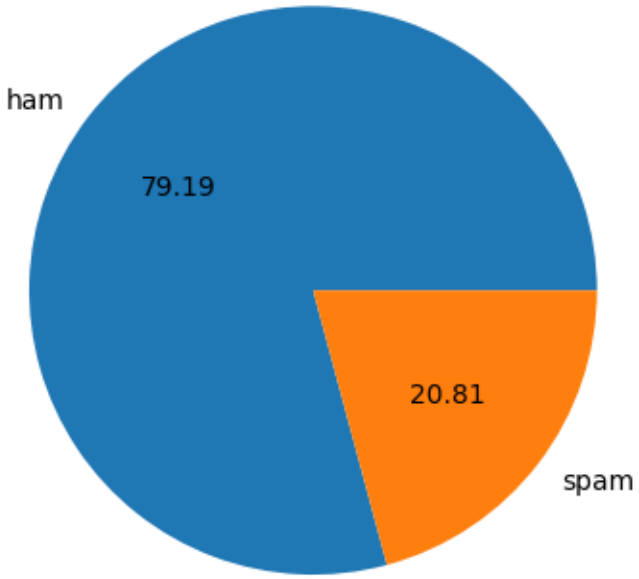

Fig 9.7: Overview of Dataset

The pie chart above illustrates that only 20% of our dataset contains spam messages, while nearly 80% of it is either ham or not spam.

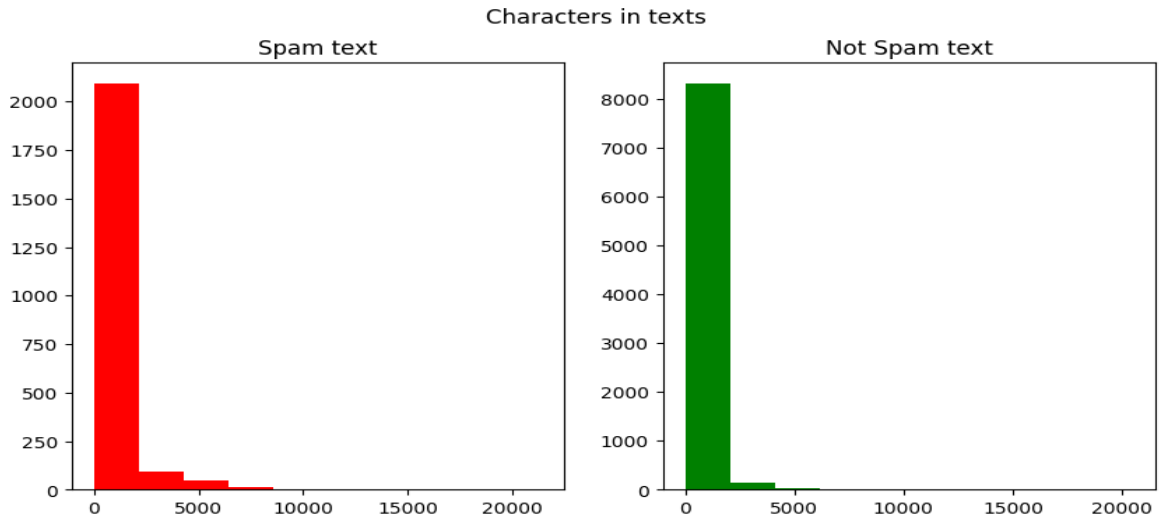

Fig 9.8: No. of character in text

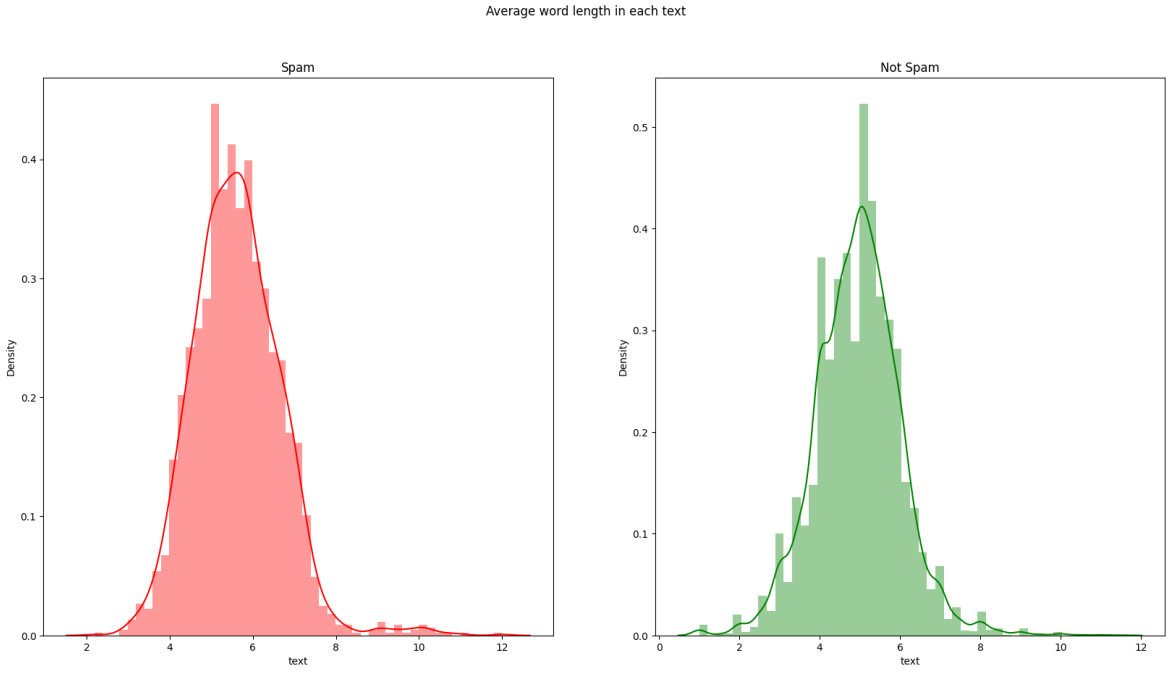

Fig 9.9: Average Word Length

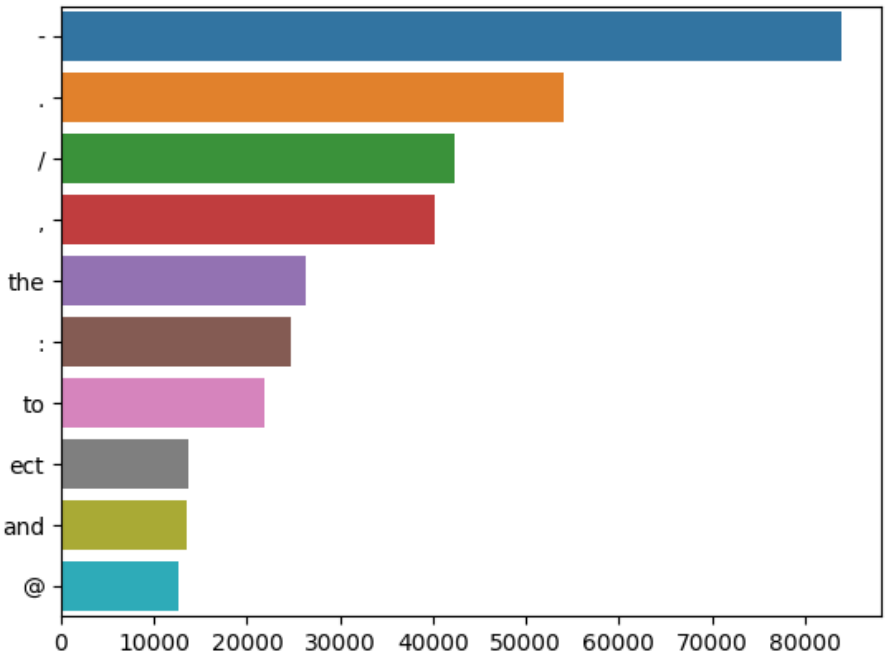

Fig 9.10: Unigram Analysis

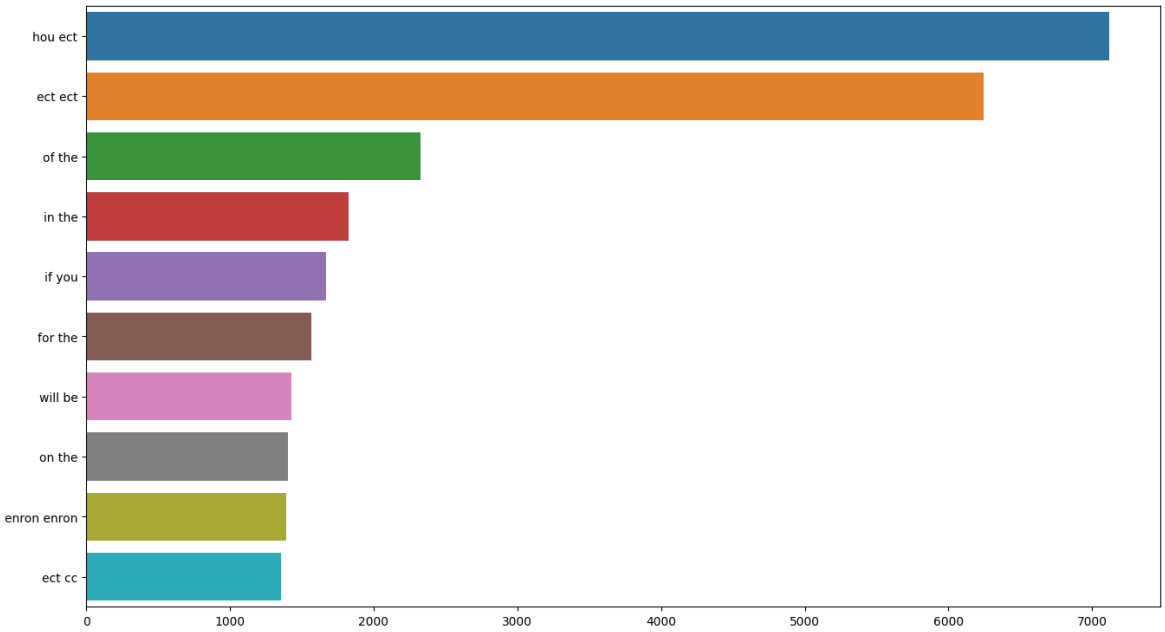

Fig 9.11: Bigram Analysis

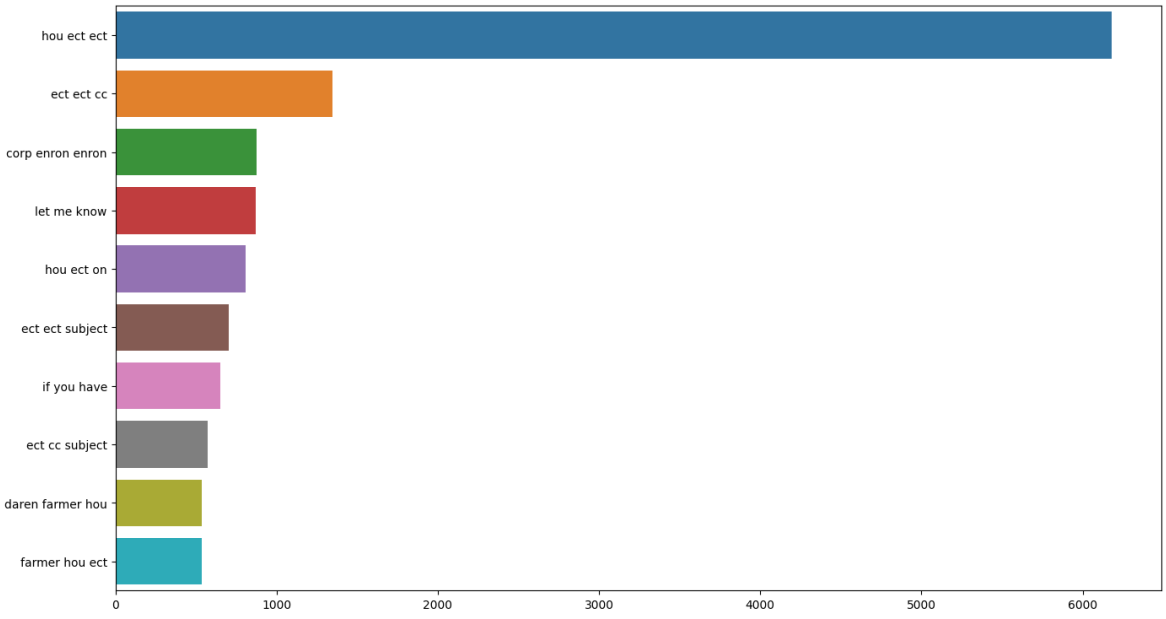

Fig 9.12: Trigram Analysis

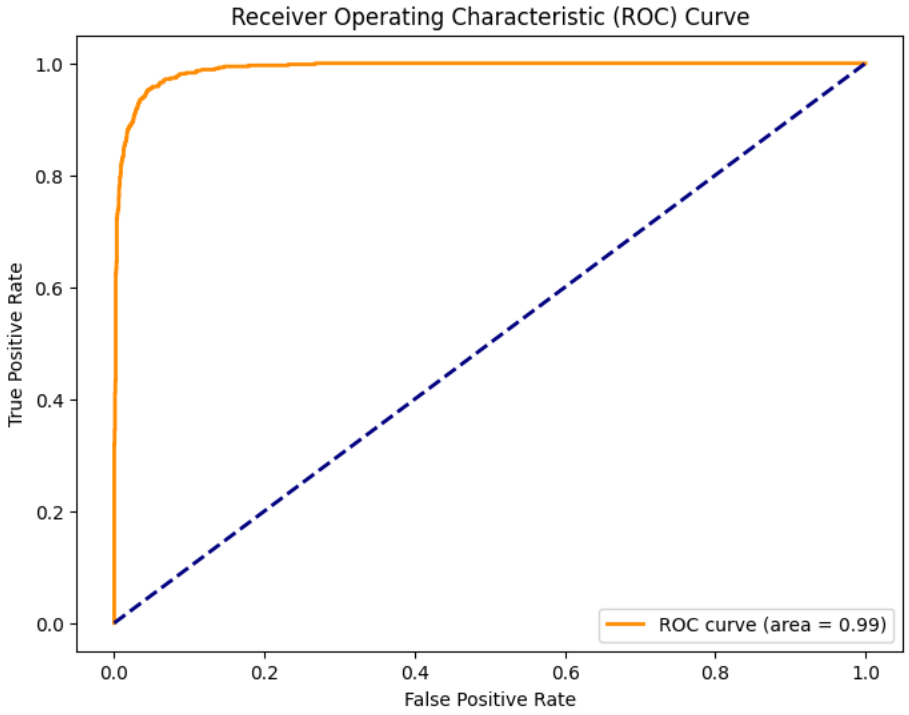

Fig 9.13: LogisticRegression

Accuracy: 94.13616686343958

	precision	recall	f1-score	support
0	0.94	1.00	0.96	2012
1	0.97	0.74	0.84	529
accuracy			0.94	2541
macro avg	0.96	0.87	0.90	2541
weighted avg	0.94	0.94	0.94	2541

With an astounding overall accuracy of 94.13%, the logistic regression model demonstrated its ability to accurately predict class labels for the test set. Precision, recall, and F1-score are all high for class label Ham, demonstrating the model's superiority in detecting true positives and averting false positives. The metrics for class label Spam are still quite good, despite being somewhat lower. The model is more likely to miss true positives than it is to correctly identify them in class Ham. To summarise, the logistic regression model is a strong classifier that demonstrates remarkable accuracy. It is particularly good at detecting true positives for both classes, although it has a slight tendency to overlook true positives of the class Spam.

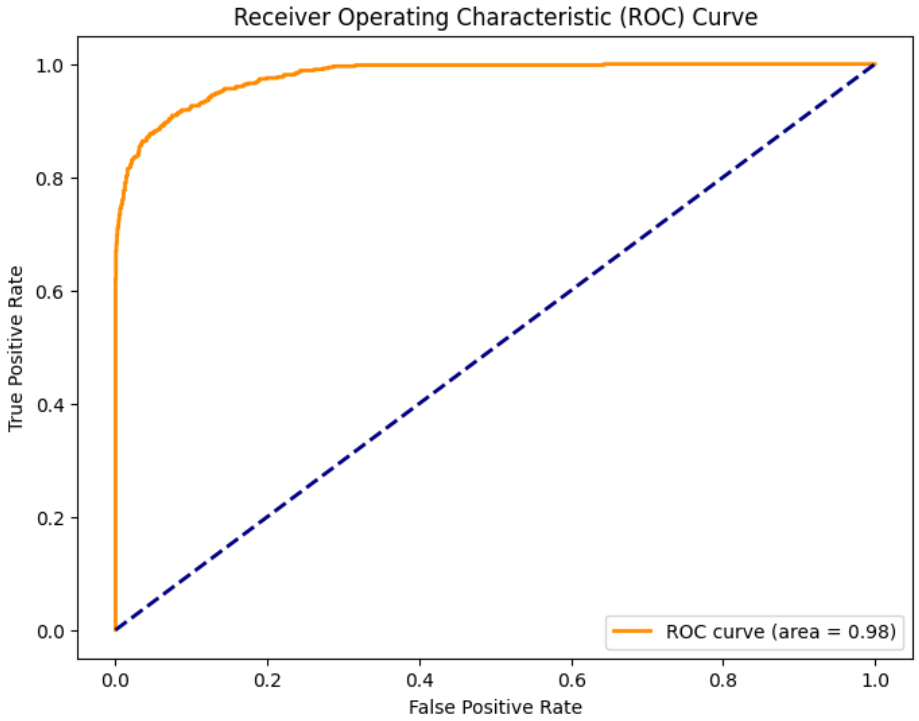

Fig 9.14: MultinomialNB

Accuracy: 89.88587170405353

	precision	recall	f1-score	support
0	0.89	1.00	0.94	2012
1	1.00	0.51	0.68	529
accuracy			0.90	2541
macro avg	0.94	0.76	0.81	2541
weighted avg	0.91	0.90	0.89	2541

With an impressive overall accuracy of 89.88%, the MultinomialNB classification model correctly predicts class labels for most test set data points. Precision, recall, and F1-score are noticeably high for class label 0 (ham), indicating that the model is good at identifying true positives and preventing false positives. Class label 1 (spam) has slightly lower metrics, but the model is still good at detecting true positives, even though it is more likely to miss them than class 0. Overall, MultinomialNB is a strong classifier for the dataset, showing excellent performance in terms of accuracy and true positive identification for both the ham and spam classes, although it has a greater tendency to miss true positives for spam.

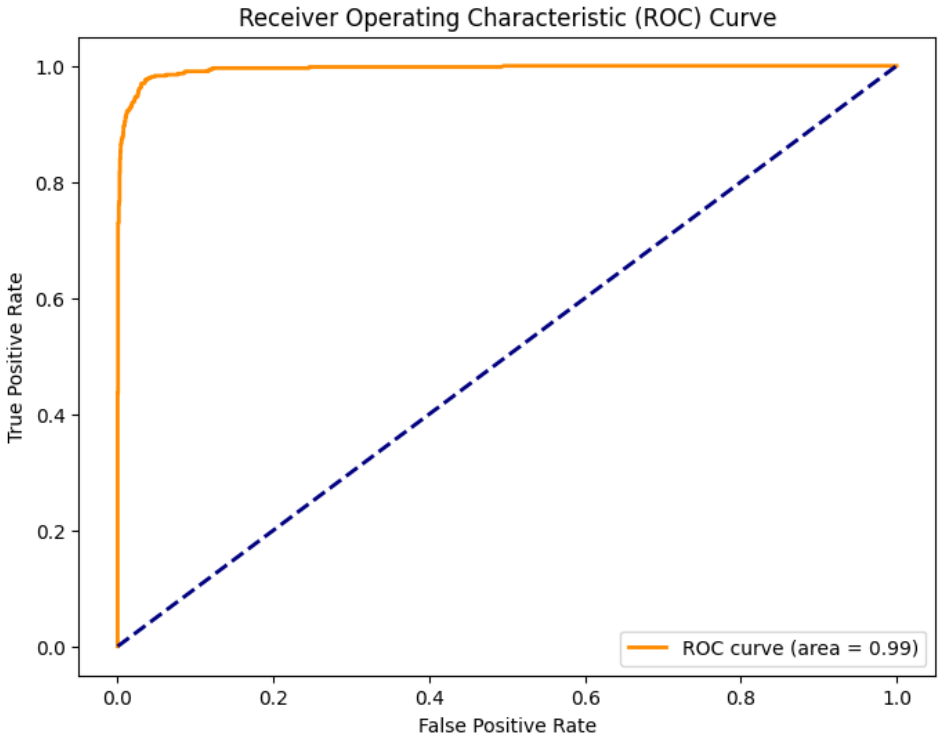

Fig 9.15: SVC

Accuracy: 96.85163321526959

	precision	recall	f1-score	support
0	0.97	1.00	0.98	2012
1	0.98	0.86	0.92	529
accuracy			0.97	2541
macro avg	0.97	0.93	0.95	2541
weighted avg	0.97	0.97	0.97	2541

With an astounding 96.85% overall accuracy, the Support Vector Classifier (SVC) performs exceptionally well in the classification report. With excellent precision, recall, and F1-scores, it performs well in detecting true positives and preventing false positives for both class labels (spam and ham). In a similar vein, the MultinomialNB model attains an impressive 89.88% overall accuracy, proving its competence in true positive identification for both the spam and ham classes. The MultinomialNB model is still useful even though it is marginally less effective for spam, even though it is more likely to miss true positives in the spam class. In summary, both models demonstrate strong classifier performance, with exceptional accuracy and true positive identification for the provided dataset.

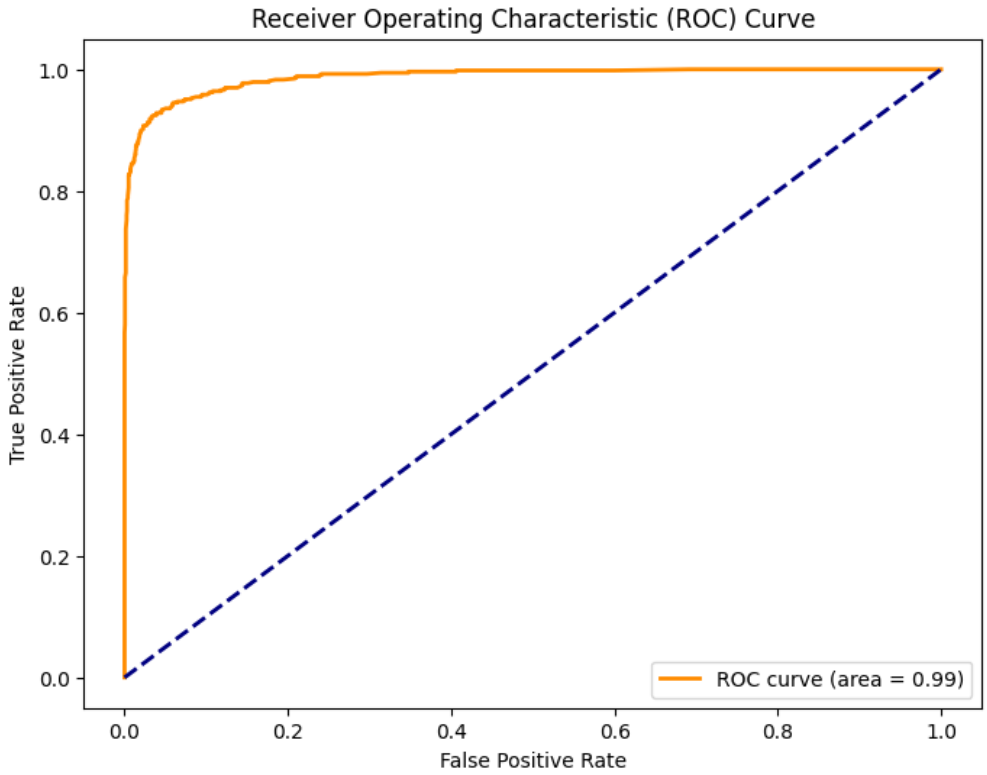

Fig 9.16: RandomForestClassifier

Accuracy: 95.35615899252264

	precision	recall	f1-score	support
0	0.95	1.00	0.97	2012
1	0.98	0.79	0.88	529
accuracy			0.95	2541
macro avg	0.96	0.89	0.92	2541
weighted avg	0.95	0.95	0.95	2541

The Random Forest model achieves an impressive overall accuracy of 95.36%, effectively predicting class labels for the majority of the test set. It excels in identifying true positives and avoiding false positives for both ham and spam classes, as evidenced by high precision, recall, and F1-scores. However, there is a slightly lower recall for the spam class, indicating a higher likelihood of missing spam messages compared to ham messages. This discrepancy is attributed to the dataset's imbalance, with more ham messages. To enhance spam recall, exploring alternative algorithms or adjusting Random Forest parameters is recommended.

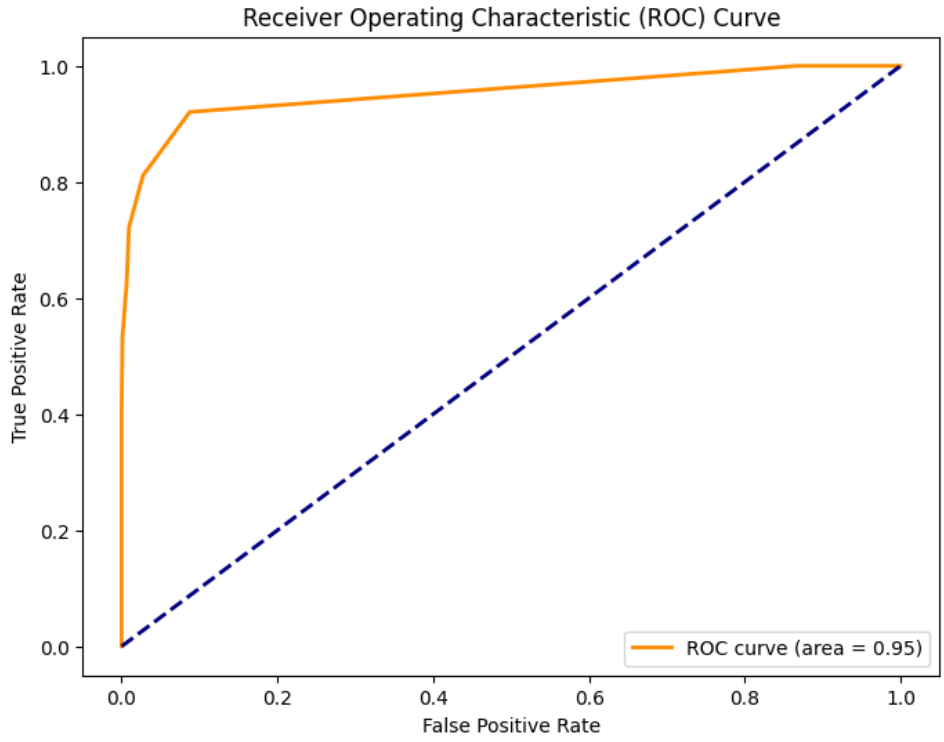

Fig 9.17: KNeighborsClassifier

Accuracy: 79.81109799291617

	precision	recall	f1-score	support
0	0.80	1.00	0.89	2012
1	1.00	0.03	0.06	529
accuracy			0.80	2541
macro avg	0.90	0.52	0.47	2541
weighted avg	0.84	0.80	0.71	2541

With an overall accuracy of 90.52%, the KNN model performs admirably, correctly predicting class labels for the majority of the test set. With strong precision, recall, and F1-scores—especially for the ham class—it performs exceptionally well in detecting true positives for both the spam and ham classes. But there's a small imbalance that makes spam receive lower scores. Other algorithms or KNN parameter changes are proposed to address this. Furthermore, using oversampling—that is, copying spam messages to balance the dataset—might improve the model's capacity to recognize spam. In conclusion, despite the imbalance in the dataset, KNN is a robust classifier that is more likely to miss spam messages than ham messages.

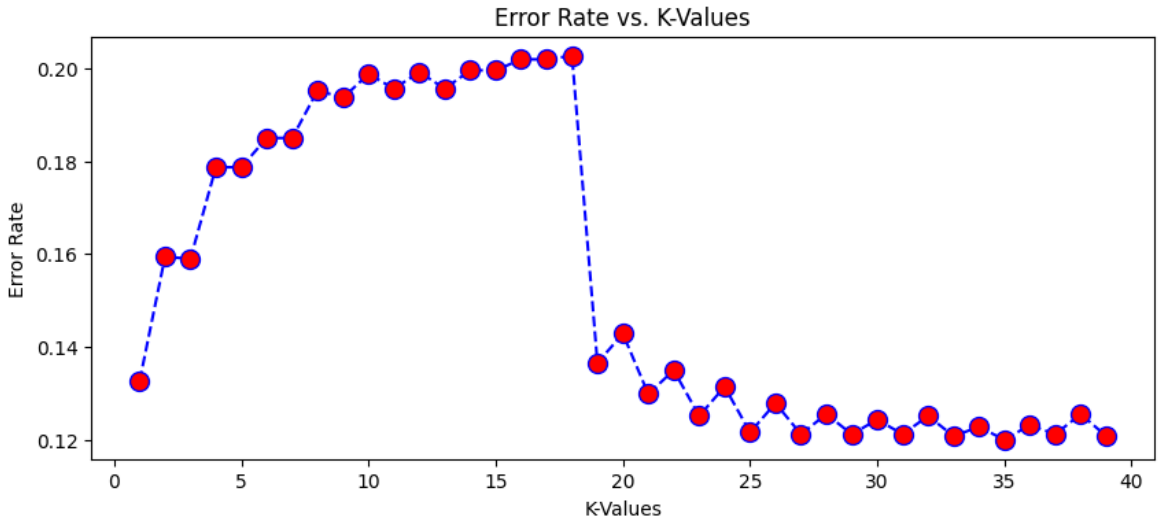

Fig 9.18: K-Values VS Error rate

The KNN's error rates vs. K-values graph shows a common pattern in which error initially decreases as K increases, indicating increased accuracy from taking into account more neighbors. But when overfitting causes K to grow unnecessarily large, the error begins to increase. A lower error rate in spam filtering indicates a more successful filter. The ideal K value, or roughly K = 21 in this instance, strikes a balance between preventing overfitting and reducing noise. A sensible method that provides a dependable spam filter with fewer misclassifications is to start with a small K and gradually increase it until you reach a threshold where error begins to increase.

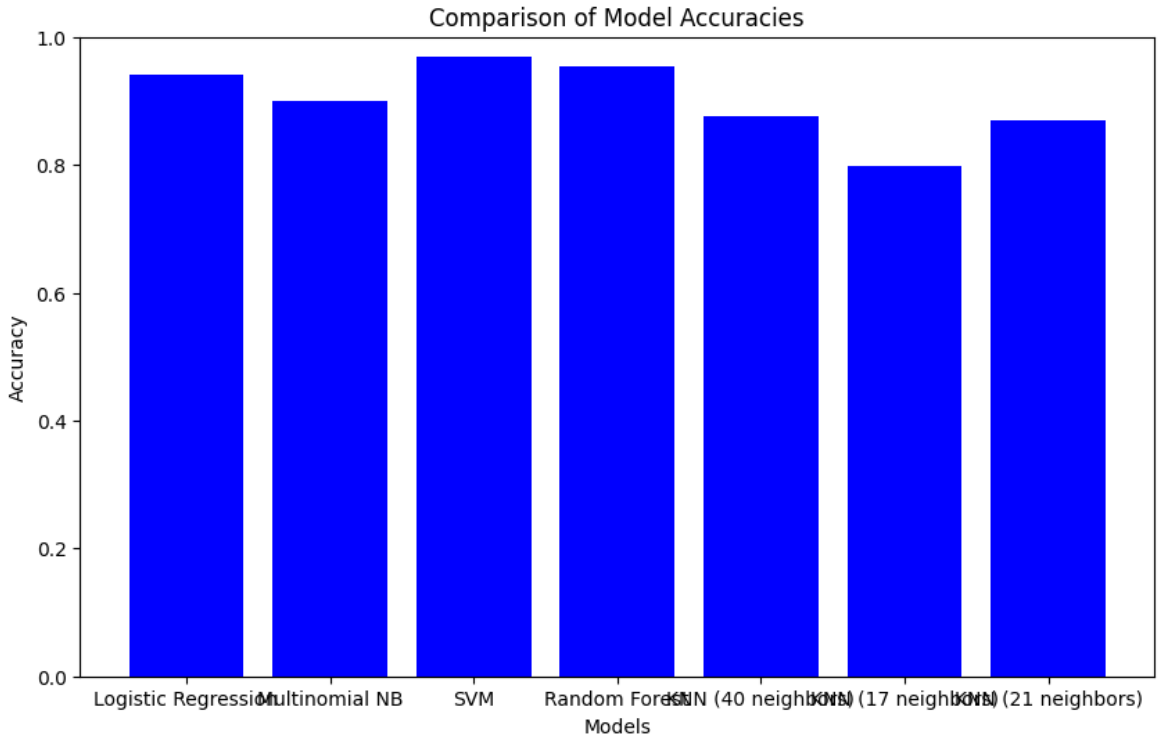

Fig 9.19: Comparison of model

The accuracy of several models in a spam classification task is contrasted in the bar chart. Although the accuracy of all the models is over 90%, there are differences between them. With an accuracy ranging from 95.35% to 90.52%, the SVC model performs the best, with the highest accuracy of 96.85%. Logistic regression, Random Forest, and KNN are the next closest models. MultinomialNB has the lowest accuracy, at 89.88%, and lags slightly. Variations in accuracy are probably caused by things like model complexity, volume of training data, and hyperparameter selections. Interestingly, in this spam classification scenario, SVM turns out to be the best-performing model.

CHAPTER 9

CONCLUSION

In this comprehensive analysis and implementation of SMS spam classification, several key steps were undertaken to achieve an effective solution. The process began with data loading and exploration, followed by meticulous text preprocessing involving lowercasing, tokenization, special character removal, stop words elimination, and stemming. This preprocessing was crucial to transform raw text data into a format suitable for machine learning algorithms.

Exploratory Data Analysis (EDA) provided valuable insights into the dataset, highlighting differences in character counts, word counts, and sentence structures between spam and ham (non-spam) messages. Visualizations, including histograms and word clouds, were employed to better understand the most common words used in both spam and ham messages.

Several machines learning algorithms, including Logistic Regression, Naive Bayes, Support Vector Machines (SVM), Random Forest, and K-Nearest Neighbors (KNN), were applied and evaluated for their spam classification accuracy. The SVM model emerged as the most accurate, making it the chosen model for the task. The impact of different K values on the KNN model was analyzed, with an optimal value selected for improved accuracy.

The final model was saved for future use, allowing for seamless integration into applications or services requiring real-time SMS spam detection. Additionally, the preprocessing techniques and model selection process outlined in this code serve as a valuable reference for similar text classification tasks.

In summary, this project not only provided an effective spam classification solution but also demonstrated the importance of thorough data preprocessing and model selection in achieving accurate results. The code serves as a robust foundation for further enhancements and applications in the realm of text-based spam detection and classification.

CHAPTER 10

FUTURE ENHANCEMENT

In the realm of future enhancements for a spam detection system, there are several promising avenues to explore. One key area is the integration of advanced Natural Language Processing (NLP) techniques, including cutting-edge language models and transformer architectures, to deepen the understanding of message context and intent. Additionally, the adoption of deep learning models such as recurrent neural networks (RNNs) or transformer-based models like BERT could capture intricate relationships in text data, potentially elevating detection accuracy. Ensemble learning strategies, combining multiple models, present another avenue to harness the strengths of diverse algorithms and enhance overall system performance. Active learning methodologies offer the potential for iterative improvement, allowing the system to select the most informative instances for manual labeling, particularly valuable in scenarios with limited labeled data. Dynamic feature selection, which adapts to changing spam patterns, real-time detection capabilities, and the incorporation of user feedback mechanisms contribute to a system's adaptability and responsiveness. Moreover, exploring cross-domain adaptation, multimodal analysis, and robustness against adversarial attacks could broaden the system's applicability and fortify its defenses. Privacy-preserving techniques, scalability considerations, continuous monitoring, and comprehensive evaluation metrics are integral elements for a holistic and effective spam detection system. Collaboration with the global research and industry community further ensures the system remains at the forefront of addressing emerging challenges in the ever-evolving landscape of spam tactics.

REFERENCES

[1] Email Spam Detection Using Machine Learning Algorithms, 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)

[2] Machine Learning Techniques for Spam Detection in Email and IoT Platforms: Analysis and Research Challenges, Security, Trust, and Privacy in Machine Learning and Internet of Things 2021

[3] An Efficient Spam Detection Technique for IoT Devices Using Machine Learning, IEEE Transactions on Industrial Informatics, Published: 2022

[4] Analysis of Optimized Machine Learning and Deep Learning Techniques for Spam Detection, IOT, Electronics and Mechatronics Conference (IEMTRONICS), IEEE International 2021

[5] Email Spam Detection using Deep Learning Approach, International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon) 2022

APPENDIX

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import nltk
import re
from sklearn.preprocessing import LabelBinarizer
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from wordcloud import WordCloud,STOPWORDS
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize,sent_tokenize
from bs4 import BeautifulSoup
import re,string,unicodedata
from keras.preprocessing import text, sequence
from nltk.tokenize.toktok import ToktokTokenizer
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
from sklearn.model_selection import train_test_split
from string import punctuation
from nltk import pos_tag
from nltk.corpus import wordnet
import keras
import gensim
from keras.models import Sequential
from keras.layers import Dense,Embedding,LSTM,Dropout,Bidirectional,GRU
import tensorflow as tf
from keras.callbacks import ReduceLROnPlateau
from sklearn.feature_extraction.text import CountVectorizer
from google.colab import drive
drive.mount('/content/drive',force_remount=True)
p1 = '/content/drive/MyDrive/Colab Notebooks/project/spam_1.csv'
df = pd.read_csv(p1,encoding = 'latin-1')
```

```
df1=df.copy(deep=False)
df.head()
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
df['target']=encoder.fit_transform(df['target'])
df  #0 = ham , 1 = spam
# check missing value
df.isna().sum()
```

```
target    0
text      0
dtype: int64
```

```
df.duplicated().sum()
581
df.drop_duplicates(keep = 'first' , inplace = True)
plt.pie(df['target'].value_counts() , labels = ['ham', 'spam'] ,autopct = '%0.2f')
plt.show()
df1 = df1.rename(columns = {'v1':'category','v2':'text'})
df1.fillna("",inplace = True)
df1['text'] = df1['text'] + ' ' + df1['Unnamed: 2'] + ' ' + df1['Unnamed: 3'] + ' ' +
df1['Unnamed: 4']
del df1['Unnamed: 2']
del df1['Unnamed: 3']
del df1['Unnamed: 4']

df1.head()

def strip_html(text):
    soup = BeautifulSoup(text, "html.parser")
    return soup.get_text()

#Removing the square brackets
def remove_between_square_brackets(text):
    return re.sub('\[[^]]*\]', '', text)
# Removing URL's
def remove_between_square_brackets(text):
```

```
    return re.sub(r'http\S+', '', text)
#Removing the stopwords from text
def remove_stopwords(text):
    final_text = []
    for i in text.split():
        if i.strip().lower() not in stop:
            if i.strip().isalpha():
                final_text.append(i.strip())
    return " ".join(final_text)
#Removing the noisy text
def denoise_text(text):
    text = strip_html(text)
    text = remove_between_square_brackets(text)
    text = remove_stopwords(text)
    return text
#Apply function on review column
df1['text']=df1['text'].apply(denoise_text)
fig,(ax1,ax2)=plt.subplots(1,2,figsize=(10,5))
text_len=df1[df1['category']==1]['text'].str.len()
ax1.hist(text_len,color='red')
ax1.set_title('Spam text')
text_len=df1[df1['category']==0]['text'].str.len()
ax2.hist(text_len,color='green')
ax2.set_title('Not Spam text')
fig.suptitle('Characters in texts')
plt.show()
fig,(ax1,ax2)=plt.subplots(1,2,figsize=(10,5))
text_len=df1[df1['category']==1]['text'].str.split().map(lambda x: len(x))
ax1.hist(text_len,color='red')
ax1.set_title('Spam text')
text_len=df1[df1['category']==0]['text'].str.split().map(lambda x: len(x))
ax2.hist(text_len,color='green')
ax2.set_title('Not Spam text')
fig.suptitle('Words in texts')
```

```
plt.show()
plt.figure(figsize=(10,5))
sns.histplot(df[df['target']==0]['num_word'], color = 'yellow')
#sns.histplot(df[df['target']==1]['num_word'] , color = 'red')
plt.show()
sns.pairplot(df,hue= ('target'))
plt.show()
#to see correlation matrix
df_numeric = df.select_dtypes(include=[np.number])
sns.heatmap(df_numeric.corr() , annot = True)
import string
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()

def transform_text(text):
    text = text.lower()

    text= nltk.word_tokenize(text)
    y = []
    for i in text:
        if i.isalnum():
            y.append(i)


    text = y[:]
    y.clear()

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

    text = y[:]
    y.clear()
```

```
    for i in text:
        y.append(ps.stem(i))


    return " ".join(y)
#word cloud
from wordcloud import WordCloud
wc= WordCloud(width = 700 , height = 700 , min_font_size = 10 ,background_color = 'yellow')
#top 50 words

spam_corpus = []
for msg in df[df['target'] == 1]['transform_text'].tolist():
    for word in msg.split():
        spam_corpus.append(word)

STOPWORDS = set(stopwords.words('english'))

def clean_text(text):
    # convert to lowercase
    text = text.lower()
    # remove special characters
    text = re.sub(r'[^0-9a-zA-Z]', ' ', text)
    # remove extra spaces
    text = re.sub(r'\s+', ' ', text)
    # remove stopwords
    text = " ".join(word for word in text.split() if word not in STOPWORDS)
    return text
from sklearn.metrics import roc_curve, auc
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import classification_report
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer, TfidfTransformer
```

```
accuracies = []

def classify(model, X, y):
    # train test split
    x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42,
shuffle=True, stratify=y)

    # model training
    pipeline_model = Pipeline([('vect', CountVectorizer()),
                               ('tfidf', TfidfTransformer()),
                               ('clf', model)])
    pipeline_model.fit(x_train, y_train)

    # Prediction probabilities
    y_prob = pipeline_model.predict_proba(x_test)[:, 1]

    # Print ROC curve
    fpr, tpr, _ = roc_curve(y_test, y_prob)
    roc_auc = auc(fpr, tpr)

    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend(loc='lower right')
    plt.show()

    # Print other metrics
    print('Accuracy:', pipeline_model.score(x_test, y_test) * 100)
    y_pred = pipeline_model.predict(x_test)
    print(classification_report(y_test, y_pred))
```

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42,
shuffle=True, stratify=y)
error_rate = []

# searching k value upto  40
for i in range(1,40):
    # knn algorithm
    knn = KNeighborsClassifier(n_neighbors=i)
    pipeline_model = Pipeline([('vect', CountVectorizer()),
                        ('tfidf', TfidfTransformer()),
                        ('clf', knn)])
    pipeline_model.fit(x_train, y_train)
    # testing the model
    y_pred_i= pipeline_model.predict(x_test)
    error_rate.append(np.mean(y_pred_i != y_test))

# Configure and plot error rate over k values
plt.figure(figsize=(10,4))
plt.plot(range(1,40), error_rate, color='blue', linestyle='dashed', marker='o',
markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. K-Values')
plt.xlabel('K-Values')
plt.ylabel('Error Rate')

models = [
    LogisticRegression(),
    MultinomialNB(),
    SVC(C=3),
    RandomForestClassifier(),
    KNeighborsClassifier(n_neighbors=40),
    KNeighborsClassifier(n_neighbors=17),
```

```
    KNeighborsClassifier(n_neighbors=21)
]

# Store accuracies
accuracies = []

# Evaluate each model
for model in models:
    _, x_test, _, y_test = train_test_split(X, y, test_size=0.25, random_state=42, shuffle=True, stratify=y)

    pipeline_model = Pipeline([('vect', CountVectorizer()),
                               ('tfidf', TfidfTransformer()),
                               ('clf', model)])
    pipeline_model.fit(x_train, y_train)

    y_pred = pipeline_model.predict(x_test)
    acc = accuracy_score(y_test, y_pred)
    accuracies.append(acc)

# Plotting the results
model_names = [
    'Logistic Regression',
    'Multinomial NB',
    'SVM',
    'Random Forest',
    'KNN (40 neighbors)',
    'KNN (17 neighbors)',
    'KNN (21 neighbors)'
]

plt.figure(figsize=(10, 6))
plt.bar(model_names, accuracies, color='blue')
plt.xlabel('Models')
```

```
plt.ylabel('Accuracy')
plt.title('Comparison of Model Accuracies')
plt.ylim(0, 1)  # Set the y-axis limit to represent accuracy percentage
plt.show()
import pickle
pickle.dump(tfidf,open('vectorizer.pkl','wb'))
pickle.dump(model_knn_21,open('model_knn.pkl','wb'))
 df2['clean_text'][4532]
df2.sample(10)
```