SPAM E-MAIL DETECTION USING NATURAL LANGUAGE PROCESSING

A Project Report
on
Artificial Intelligence

Submitted by

DANIEL RAJ A.	**(Reg. No.: 222307676)**
DHANUSH V.	**(Reg. No.: 222307680)**
GOKUL KANNAN T.	**(Reg. No.: 222307682)**
HARIHARAN K.	**(Reg. No.: 222307685)**
ESAKKI MUTHU SANJAY E.	**(Reg. No.: 222307681)**

Under the Guidance of
Dr. Vinolyn Vijaykumar, M.C.A., M.Phil., Ph.D.,
Assistant Professor

Department of Computer Science
ALPHA ARTS AND SCIENCE COLLEGE
CHENNAI – 600116

October 2024

ALPHA ARTS AND SCIENCE COLLEGE
CHENNAI – 600116

DEPARTMENT OF COMPUTER SCIENCE

BONAFIDE CERTIFICATE

Certified that this report entitled **SPAM E-MAIL DETECTION USING NATURAL LANGUAGE PROCESSING** is a Bonafide record of the project work done by **DANIEL RAJ A. (222307676), DHANUSH V. (222307680), GOKUL KANNAN T. (222307682), HARIHARAN K. (222307685) and ESAKKI MUTHU SANJAY E. (222307681)** under my supervision and guidance.

Signature of the Guide **Head of the Department**

DECLARATION

We hereby declare that the Project Work entitled **"SPAM E-MAIL DETECTION USING NATURAL LANGUAGE PROCESSING"** is the original work done by us under the supervision of **Dr. Vinolyn Vijaykumar, M.C.A., M.Phil., Ph.D.,** Department of Computer Science, Alpha Arts and Science College, Chennai – 600116. This report has not been submitted for any other degree or diploma at any other institution.

NAME	**REG. NO.**
DANIEL RAJ A.	**222307676**
DHANUSH V.	**222307680**
GOKUL KANNAN T.	**222307682**
HARIHARAN K.	**222307685**
ESAKKI MUTHU SANJAY E.	**222307681**

Place:

Date:

Signature of the Candidates

1.

2.

3.

4.

5.

ACKNOWLEDGMENTS

Formost, I express gratitude to the Almighty for His abundant blessings, facilitating the efficient completion of the project entitled **"SPAM E-MAIL DETECTION USING NATURAL LANGUAGE PROCESSING"**

I am profoundly grateful to ***Dr. A. Sivasankar***, **Principal, Alpha Arts and Science College,** Chennai - 600116, for his unwavering encouragement and support, which played a pivotal role in the successful completion of this Project Work.

First and foremost, I would like to thank my supervisor, ***Dr. Vinolyn Vijaykumar***, for her continuous guidance, insightful feedback, and encouragement throughout this project. Her expertise and patience were instrumental in shaping the direction of my research.

I extend my sincere appreciation to ***Dr.G.Gayathri****,* Head of the Department, for his invaluable guidance, constructive criticism, and constant motivation throughout the duration of this Project.

A special thanks to **ALPHA ARTS AND SCIENCE COLLEGE**, whose resources and facilities provided a conducive environment for research and development.

Lastly, I am deeply appreciative of my family and friends, whose unwavering support motivated me to pursue and complete this endeavor.

[Daniel Raj A.]

[Dhanush V.]

[Gokul Kannan T.]

[Hariharan K.]

[Esaki Muthu Sanjai E.]

ABSTRACT

Nowadays communication plays a major role in everything be it professional or personal. Email communication service is being used extensively because of its free use services, low-cost operations, accessibility, and popularity. Emails have one major security flaw that is anyone can send an email to anyone just by getting their unique user id. This security flaw is being exploited by some businesses and ill-motivated persons for advertising, phishing, malicious purposes, and finally fraud. This produces a kind of email category called SPAM.

To tackle this problem, we present a new and efficient method to detect spam using machine learning and natural language processing. A tool that can detect and classify spam. In addition to that, it also provides information regarding the text provided in a quick view format for user convenience.

Project Name	SPAM E-MAIL DETECTION USING NATURAL LANGUAGE PROCESSING
Language Used	PYTHON
Operating System	Windows 10
Tools	Google Colaboratory, Excel
Team Members	DANIEL RAJ A. (Reg. No.: 222307676) DHANUSH V. (Reg. No.: 222307680) GOKUL KANNAN T. (Reg. No.: 222307682) HARIHARAN K. (Reg. No.: 222307685) ESAKKI MUTHU SANJAY E. (Reg. No.: 222307681)

TABLE OF CONTENTS

1. Introduction

Today, Spam has become a major problem in communication over internet. It has been accounted that around 55% of all emails are reported as spam and the number has been growing steadily. Spam which is also known as unsolicited bulk email has led to the increasing use of email as email provides the perfect ways to send the unwanted advertisement or junk newsgroup posting at no cost for the sender. This chance has been extensively exploited by irresponsible organizations and resulting to clutter the mailboxes of millions of people all around the world.

Spam has been a major concern given the offensive content of messages; spam is a waste of time. End user is at risk of deleting legitimate mail by mistake. Moreover, spam also impacted the economical which led some countries to adopt legislation.

Text classification is used to determine the path of incoming mail/message either into inbox or straight to spam folder. It is the process of assigning categories to text according to its content. It is used to organized, structures and categorize text. It can be done either manually or automatically. Machine learning automatically classifies the text in a much faster way than manual technique. Machine learning uses pre-labelled text to learn the different associations between pieces of text and it output. It used feature extraction to transform each text to numerical representation in form of vector which represents the frequency of word in predefined dictionary.

Text classification is important to structure the unstructured and messy nature of text such as documents and spam messages in a cost-effective way. Machine learning can make more accurate precisions in real-time and help to improve the manual slow process to much better and faster analysing big data. It is important especially to a company to analyse text data, help inform business decisions and even automate business processes.

In this project, machine learning techniques are used to detect the spam message of a mail. Machine learning is where computers can learn to do something without the need to explicitly program them for the task.

It uses data and produce a program to perform a task such as classification. Compared to knowledge engineering, machine learning techniques require messages that have been successfully pre-classified. The pre-classified messages make the training data-set which will be used to fit the learning algorithm to the model in machine learning studio.

A combination of algorithms is used to learn the classification rules from messages. These algorithms are used for classification of objects of different classes. These algorithms are provided with pre labelled data and an unknown text. After learning from the prelabelled data each of these algorithms predict which class the unknown text may belong to, and the category predicted by majority is considered as final.

1.1 Background

Spam emails have been a persistent problem since the early days of email communication. The volume and sophistication of spam have increased significantly, making it challenging for traditional rule-based filters to effectively distinguish between legitimate and spam messages. Advances in NLP and machine learning offer new approaches to improve spam detection by analysing and understanding the content of emails. Techniques such as tokenization, stop-word removal, and feature extraction have been employed to convert textual data into a format suitable for machine learning models. Moreover, recent developments in deep learning, including recurrent neural networks (RNNs) and transformers, provide powerful tools for capturing complex patterns in text data.

1.2 Problem Statement

The challenge of spam email detection lies in accurately classifying emails as either spam or non-spam amidst a vast array of potentially deceptive content. Traditional methods often rely on predefined rules and heuristics that can become outdated or ineffective as spammers adapt their tactics. There is a need for an advanced, data-driven approach that leverages NLP and machine learning to dynamically learn from and adapt to new spam techniques, thereby enhancing the reliability and efficiency of spam filters.

1.3 Objectives

Develop an NLP-Based Model: Create a model capable of classifying emails into spam and non-spam categories using advanced NLP techniques.

Improve Classification Accuracy: Enhance the accuracy of spam detection by utilizing feature extraction methods and sophisticated machine learning algorithms.

Evaluate Performance: Assess the model's performance using metrics such as accuracy, precision, recall, and F1-score to ensure its effectiveness in real-world scenarios.

Provide Insights: Analyze the results to understand the strengths and limitations of the chosen methods and suggest potential improvements.

1.4 Scope of the Study

This study focuses on the development of a spam email detection system using NLP techniques. The scope includes:

Data-set Utilization: Employing publicly available email datasets for training and evaluation.

NLP Techniques: Applying tokenization, stop-word removal, lemmatization, and vectorization methods to preprocess email content.

Model Selection: Experimenting with various machine learning models, including Naive Bayes, Support Vector Machine (SVM), and deep learning approaches like RNNs and Transformers.

Performance Evaluation: Measuring the model's effectiveness through various performance metrics and comparing different approaches.

The study does not cover real-time deployment or integration with email clients but focuses on the theoretical and practical aspects of model development and evaluation.

1.5 Significance of the Study

This study is significant for several reasons:

Enhanced Email Security: By improving spam detection, users can be better protected from phishing attempts and malware distributed via email.

Improved Productivity: Effective spam filters reduce the volume of unwanted emails, allowing users to focus on legitimate messages and increasing overall productivity.

Advancement in NLP: The project contributes to the application and advancement of NLP techniques in practical scenarios, potentially providing insights into their effectiveness for other text classification tasks.

Foundation for Future Research: The results and methodologies can serve as a foundation for further research in spam detection and other areas of email security.

By addressing the problem of spam email detection through advanced NLP methods, this study aims to provide valuable solutions and insights that can benefit both individual users and organizations.

2. Literature Review

2.1 Related Work

Spam classification is a problem that is neither new nor simple. A lot of research has been done and several effective methods have been proposed.

Raza, M., Jayasinghe, N. D., & Muslam, M. M. A. (Year). The authors conducted a comprehensive analysis of various spam classification techniques, concluding that Naïve Bayes and Support Vector Machines (SVM) yielded the most consistent and high accuracy rates, both achieving around 91%. This study emphasizes the reliability and performance of these traditional machine learning algorithms in spam detection tasks. [1]

Gadde, S., Lakshmanarao, A., & Satyanarayana, S. (Year). In their research focused on spam detection, the authors reported that Long Short-Term Memory (LSTM) models achieved superior performance, recording an accuracy of 98%. The study highlights the advantages of deep learning approaches in capturing sequential data features, which are prevalent in spam messages. [2]

Sethi, P., Bhandari, V., & Kohli, B. (Year). Their findings illustrate that the effectiveness of machine learning algorithms in spam classification depends significantly on the specific attributes and features used in the dataset. This suggests a need for careful feature selection and engineering to maximize model performance. [3]

Karamollaoglu, H., Dogru, İ. A., & Dorterler, M. (Year). The study applied Naïve Bayes and SVM algorithms on Turkish email datasets and concluded that both models achieved approximately 90% accuracy. The results underline the adaptability of traditional classifiers across different languages and datasets. [4]

Navaney, P., Dubey, G., & Rana, A. (Year). By comparing SVM, Naïve Bayes, and entropy-based models, the researchers determined that SVM exhibited the highest accuracy at 97.5%. This reinforces the model's robustness and effectiveness in handling spam classification tasks. [5]

Nandhini, S., & Marseline, J. K. S. (Year). Their paper evaluated various spam detection models, concluding that the Random Forest algorithm outperformed others in terms of accuracy, while the K-Nearest Neighbors (KNN) model excelled in terms of faster build time. This study sheds light on the trade-offs between accuracy and computational efficiency. [6]

Olatunji, S. O. (Year). This research compared the performance of SVM and Extreme Learning Machines (ELM), finding that while SVM surpassed ELM in classification accuracy, the ELM offered faster processing times. These insights are particularly relevant for time-sensitive applications. [7]

Gupta, M., Bakliwal, A., Agarwal, S., & Mehndiratta, P. (Year). The authors studied classical machine learning algorithms and observed that Convolutional Neural Networks (CNNs) marginally outperformed them in spam classification. However, CNNs required more computational time, reflecting a trade-off between performance and speed. [8]

Kumar, N., Sonowal, S., & Nishant (Year). This study supported the effectiveness of the Naïve Bayes classifier but pointed out its limitations concerning class-conditional independence, which may impact model generalizability. [9]

Toma, T., Hassan, S., & Arifuzzaman, M. (Year). Their research analyzed different Naïve Bayes variants and demonstrated that the Multinomial Naïve Bayes algorithm achieved the best results, with a classification accuracy of 98%, highlighting its suitability for spam text data. [10]

Hossain, F., Uddin, M. N., & Halder, R. K. (Year). The researchers argued that traditional machine learning models often outperformed deep learning models for spam detection. Additionally, ensemble methods surpassed individual models in accuracy and precision, showcasing the strength of model aggregation techniques. [11]

Vaswani, A., Shazeer, N., Parmar, N., et al. (Year). In their seminal work on neural processing systems, the authors laid the groundwork for transformer-based architectures that underpin modern deep learning models, significantly influencing advanced spam detection systems. [12]

Zhang, Y., Shi, P., Dong, C., et al. (Year). Their work focused on cybersecurity test and evaluation systems for automotive applications, which, while outside of spam detection, offers insights into threat identification and system robustness that could influence related research domains. [13]

Delany, S. J., Buckley, M., & Greene, D. (Year). This foundational study on SMS spam filtering reviewed various filtering techniques and dataset characteristics, providing a crucial resource for understanding baseline approaches and dataset biases. [14]

McCallum, A., & Nigam, K. (Year). The authors compared event models within the Naïve Bayes framework for text classification. Their analysis supports refined probabilistic modeling for improved spam detection accuracy. [15]

Chunduru, A., Karrothu, A., Mouli, N. S., & Tej, C. B. (Year). This survey paper provided an overview of phishing email recognition techniques using NLP, which shares considerable overlap with spam detection, especially in analyzing deceptive language patterns. [16]

Mikolov, T., Sutskever, I., Chen, K., et al. (Year). Their groundbreaking research introduced distributed word representations, a critical advancement that enhanced the ability of models to understand semantics in spam and phishing messages. [17]

Chimurkar, S., & Karia, D. (Year). Their study integrated machine learning with NLP for spam classification and demonstrated the value of combined linguistic and algorithmic approaches in improving detection capabilities. [18]

Kotsiantis, S. B., Zaharakis, I. D., & Pintelas, P. E. (Year). This review of machine learning classification techniques highlighted the benefits of combining models, which supports the trend of ensemble learning in spam detection. [19]

Shanmugasundaram, G., Preethi, S., & Nivedha, I. (2017). The authors investigated spam detection in social media contexts, illustrating how traditional spam detection algorithms could be adapted to dynamic online environments, expanding the domain applicability of these techniques. [20]

3. Methodology

Introduction

This chapter will explain the specific details on the methodology being used to develop this project. Methodology is an important role as a guide for this project to make sure it is in the right path and working as well as plan. There is different type of methodology used in order to do spam detection and filtering. So, it is important to choose the right and suitable methodology thus it is necessary to understand the application functionality itself.

3.1 Research Design

Objective: To develop an effective spam email detection system utilizing NLP techniques for feature extraction and classification.

Approach: Exploratory Analysis: Investigate various NLP techniques for feature extraction and their impact on classification performance.

Model Evaluation: Compare different NLP-based methods to determine the most effective approach for spam detection.

3.2 Data Collection

- Data plays an important role when it comes to prediction and classification, the more the data the more the accuracy will be.
- The data used in this project is completely open-source and has been taken from various resources like Kaggle and UCI
- For the purpose of accuracy and diversity in data multiple datasets are taken. 2 datasets containing approximately over 12000 mails and their labels are used for training and testing the application.

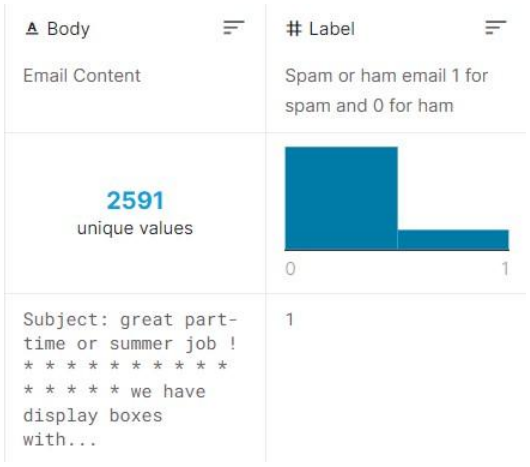
- 6000 spam mails are taken for generalization of data and to increase the accuracy.

Data Description

Dataset: Enron Spam Subset.

Source: Kaggle

Description: this data-set is part of a larger data-set called Enron. This data-set contains a set of spam and non-spam emails with 0 for non-spam and 1 for spam in label attribute.



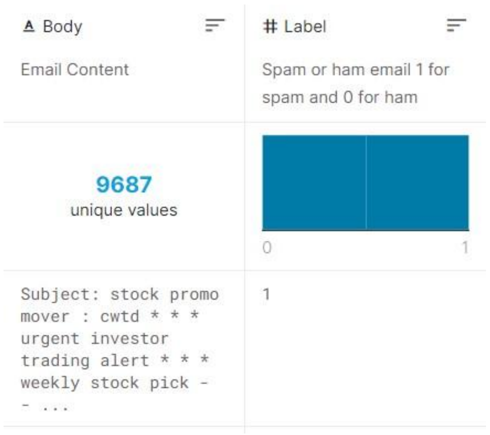
Body	Label
Email Content	Spam or ham email 1 for spam and 0 for ham
2591 unique values	0 1
Subject: great part-time or summer job ! * * * * * * * * * * * * * * * we have display boxes with...	1

Composition:

Unique values: 9687

Spam values: 5000

Non-spam values: 4687



Body	Label
Email Content	Spam or ham email 1 for spam and 0 for ham
9687 unique values	0 1
Subject: stock promo mover : cwtd * * * urgent investor trading alert * * * weekly stock pick - - ...	1

Dataset: ling spam.

Source: Kaggle

Description: This dataset is part of a larger dataset called Enron1 which contains emails classified as spam or ham(not-spam).

Composition:

Unique values: 2591

Spam values: 419

Non-spam values: 2172

3.3 Data Pre-Processing

Overall data processing

It consists of two main tasks

- **Dataset cleaning**

  It includes tasks such as removal of outliers, null value removal, removal of unwanted features from data.

- **Dataset Merging**

  After data cleaning, the datasets are merged to form a single data-set containing only two features (text, label).

  Data cleaning, Data Merging these procedures are completely done using Panda's library.

Textual data processing

- **Tag removal**

  Removing all kinds of tags and unknown characters from text using regular expressions through Regex library.

- **Sentencing, tokenization**

  Breaking down the text(email/SMS) into sentences and then into tokens(words).

  This process is done using NLTK pre-processing library of python.

- **Stop word removal**

  Stop words such as off, a, be, … are removed using stop words NLTK library of python.

- **Lemmatization**

  Words are converted into their base forms using lemmatization and post-tagging

  This process gives keywords through entity extraction.

This process is done using chunking in regex and NLTK lemmatization.

- **Sentence formation**

  The lemmatized tokens are combined to form a sentence.

  This sentence is essentially a sentence converted into its base form and removing stop words.

  Then all the sentences are combined to form a text.
- While the overall data processing is done only to datasets, the textual processing is done to both training data, testing data and user input data.

Feature Vector Formation

- The texts are converted into feature vectors (numerical data) using the words present in all the texts combined
- This process is done using count vectorization of **NLTK** library.
- The feature vectors can be formed using two language models Bag of Words and Term Frequency-inverse Document Frequency.

Bag of Words

Bag of words is a language model used mainly in text classification. A bag of words represents the text in a numerical form.

The two things required for Bag of Words are

- A vocabulary of words known to us.
- A way to measure the presence of words.

Ex: a few lines from the book “A Tale of Two Cities” by Charles Dickens.

“It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness,”

The unique words here (ignoring case and punctuation) are:

[“it”, “was”, “the”, “best”, “of”, “times”, “worst”,“age”, “wisdom”, “foolishness”] The next step is scoring words present in every document.

After scoring the four lines from the above stanza can be represented in vector form as

" It was the best of times " = [1, 1, 1, 1, 1, 1, 0, 0, 0, 0]

"It was the worst of times" = [1, 1, 1, 0, 1, 1, 1, 0, 0, 0]

"It was the age of wisdom" = [1, 1, 1, 0, 1, 0, 0, 1, 1, 0]

"It was the age of foolishness"= [1, 1, 1, 0, 1, 0, 0, 1, 0, 1]

This is the main process behind the bag of words but in reality, the vocabulary even from a couple of documents is very large and words repeating frequently and important in nature are taken and remaining are removed during the text Processing stage.

Term Frequency-inverse document frequency

Term frequency-inverse document frequency of a word is a measurement of the importance of a word. It compares the repentance of words to the collection of documents and calculates the score.

Terminology for the below formulae:

t – term(word)

d – document (set of words)

N – count of documents

The TF-IDF process consists of various activities listed below.

i) **Term Frequency**

The count of appearance of a particular word in a document is called term frequency ***tf (t,d) = count of t in d/ number of words in d***

ii) **Document Frequency**

Document frequency is the count of documents the word was detected in. We consider one instance of a word, and it doesn't matter if the word is present multiple times.

df (t) = occurrence of t in documents

iii) **Inverse Document Frequency**

- IDF is the inverse of document frequency.
- It measures the importance of a term t considering the information it contributes. Every term is considered equally important but certain terms such as (are, if, a, be, that, ...) provide little information about the document. The inverse document frequency factor reduces the importance of words/terms that has higher recurrence and increases the importance of words/terms that are rare.

idf_(t) = N/ df

Finally, the TF-IDF can be calculated by combining the term frequency and inverse document frequency.

tf_idf(t, d) = tf(t,d) = log(N/(df + 1))

the process can be explained using the following example:

"**Document 1 It is going to rain today.**
Document 2 Today I am not going outside.
Document 3 I am going to watch the season premiere."

The Bag of words of the above sentences is

[going:3, to:2, today:2, i:2, am:2, it:1, is:1, rain:1]

Then finding the term frequency

Words/ documents	going	to	Today	i	am	if	it	rain
Document1	0	0.07	0.07	0	0	0.17	0.17	0.17
Document2	0	0	0.07	0.07	0.07	0	0	0
Document3	0	0.05	0	0.05	0.05	0	0	0

Then finding the inverse document frequency

Words	Document1	Document2	Document3
Going	0.16	0.16	0.12
To	0.16	0	0.12
Today	0.16	0.16	0
I	0	0.16	0.12
Am	0	0.16	0.12
It	0.16	0	0
Is	0.16	0	0
rain	0.16	0	0

Words	IDF Value
Going	log(3/3)
To	log(3/2)
Today	log(3/2)
I	log(3/2)
Am	log(3/2)
It	log(3/1)
Is	log(3/1)
rain	log(3/1)

Applying the final equation and the values of tf-idf becomes

Using the above two language models the complete data has been converted into two kinds of vectors and stored into a CVS type file for easy access and minimal processing.

Certainly! Here’s a focused approach for implementing spam email detection using NLP techniques alone, without integrating traditional machine learning algorithms:

Data Splitting

The data splitting is done to create two kinds of data Training data and testing data. Training data is used to train the machine learning models and testing data is used to test the models and analyse results. 80% of total data is selected as testing data and remaining data is testing data.

3.4 Algorithm/ Model Implementation

A combination of 5 algorithms is used for the classifications.

Naïve Bayes Classifier

A naïve Bayes classifier is a supervised probabilistic machine learning model that is used for classification tasks. The main principle behind this model is the Bayes theorem.

Bayes Theorem:

Naive Bayes is a classification technique that is based on Bayes’ Theorem with an assumption that all the features that predict the target value are independent of each other. It calculates the probability of each class and then picks the one with the highest probability.

Naive Bayes classifier assumes that the features we use to predict the target are independent and do not affect each other. Though the independence assumption is never correct in real-world data, but often works well in practice. so that it is called “Naive” [14].

$$P(A \mid B) =(P(B \mid A) P(A))/P(B)$$

P(A|B) is the probability of hypothesis A given the data B. This is called the posterior probability.

P(B|A) is the probability of data B given that hypothesis A was true.
P(A) is the probability of hypothesis A being true (regardless of the data). This is called the prior probability of A.
P(B) is the probability of the data (regardless of the hypothesis) [15].

Naïve Bayes classifiers are mostly used for text classification. The limitation of the Naïve Bayes model is that it treats every word in a text as independent and is equal in importance, but every word cannot be treated equally important because articles and nouns are not the same when it comes to language. But due to its classification efficiency, this model is used in combination with other language processing techniques.

Random Forest Classifier

Random Forest classifier is a supervised ensemble algorithm. A random forest consists of multiple random decision trees. Two types of randomness's are built into the trees. First, each tree is built on a random sample from the original data. Second, at each tree node, a subset of features is randomly selected to generate the best split.

Decision Tree:

The decision tree is a classification algorithm based completely on features. The tree repeatedly splits the data on a feature with the best information gain. This process continues until the information gained remains constant. Then the unknown data is evaluated feature by feature until categorized. Tree pruning techniques are used for improving accuracy and reducing the over-fitting of data.

Several decision trees are created on subsets of data the result that was given by the majority of trees is considered as the result. The number of trees to be created is determined based on accuracy and other metrics through iterative methods. Random forest classifiers are mainly used on condition-based data but it works for text if the text is converted into numerical form.

Logistic Regression

Logistic Regression is a “Supervised machine learning” algorithm that can be used to model the probability of a certain class or event. It is used when the data is linearly separable, and the outcome is binary or dichotomous [17]. The probabilities are calculated using a sigmoid function.

For example, let us take a problem where data has n features.
We need to fit a line for the given data and this line can be represented by the equation

z=b_0+b_1 x_1+b_2 x_2+b_3 x_3…. +b_n x_n

here z = odds
generally, odds are calculated as

odds=p (event occurring)/p (event not occurring)

Sigmoid Function:

A sigmoid function is a special form of logistic function hence the name logistic regression. The logarithm of odds is calculated and fed into the sigmoid function to get continuous probability ranging from 0 to 1.The logarithm of odds can be calculated by

log(odds)=dot (features, coefficients) +intercept

and these log_odds are used in the sigmoid function to get probability.

h(z)=1/(1+e^(-z))

The output of the sigmoid function is an integer in the range 0 to 1 which is used to determine which class the sample belongs to. Generally, 0.5 is considered as the limit below which it is considered a NO, and 0.5 or higher will be considered a YES. But the border can be adjusted based on the requirement.

K-Nearest Neighbors

KNN is a classification algorithm. It comes under supervised algorithms. All the data points are assumed to be in an n-dimensional space. And then based on neighbors the category of current data is determined based on the majority.

Euclidian distance is used to determine the distance between points.

The distance between 2 points is calculated as

d=√(〖(x2-x1) 〗 ^2+ 〖(y2-y1) 〗 ^2) 〗 〖 〗

The distances between the unknown point and all the others are calculated. Depending on the K provided k closest neighbors are determined. The category to which the majority of the neighbors belong is selected as the unknown data category.

If the data contains up to 3 features, then the plot can be visualized. It is fairly slow compared to other distance-based algorithms such as SVM as it needs to determine the distance to all points to get the closest neighbors to the given point.

Support Vector Machines (SVM)

It is a machine learning algorithm for classification. Decision boundaries are drawn between various categories and based on which side the point falls to the boundary the category is determined.

Support Vectors:

The vectors closer to boundaries are called support vectors/planes. If there are n categories, then there will be n+1 support vectors. Instead of points, these are called vectors because they are assumed to be starting from the origin. The distance between the support vectors is called margin. We want our margin to be as wide as possible because it yields better results.

There are three types of boundaries used by SVM to create boundaries.
Linear: used if the data is linearly separable.
Poly: used if data is not separable. It creates any data into 3-dimensional data.
Radial: this is the default kernel used in SVM. It converts any data into infinite-dimensional data.

If the data is 2-dimensional then the boundaries are lines. If the data is 3-dimensional then the boundaries are planes. If the data categories are more than 3 then boundaries are called hyper-planes.

An SVM mainly depends on the decision boundaries for predictions. It does not compare the data to all other data to get the prediction due to this SVM's tend to be quick with predictions.

Experimentation

The process goes like data collection and processing then natural language processing and then vectorization then machine learning. The data is collected, cleaned, and then subjected to natural language processing techniques specified in section IV. Then the cleaned data is converted into vectors using Bag of Words and TF-IDF methods which goes like...

The Data is split into Training data and Testing Data in an 80-20 split ratio. The training and testing data are converted into Bag-of-Words vectors and TF-IDF vectors.

There are several metrics to evaluate the models, but accuracy is considered for comparing BoW and TF-IDF models. Accuracy is generally used to determine the efficiency of a model.

Accuracy:

"Accuracy is the number of correctly predicted data points out of all the data points".

Naïve Bayes Classification algorithm:

Two models, one for Bow and one for TF-IDF are created and trained using respective training vectors and training labels. Then the respective testing vectors and labels are used to get the score for the model

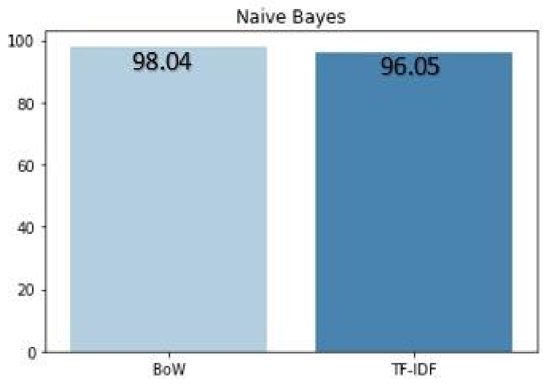


fig: naïve Bayes

The scores for Bag-of-Words and TF-IDF are visualized.

The scores for the Bow model and TF-IDF models are 98.04 and 96.05 respectively for using the naïve bayes model.

Logistic Regression:

Two models are created following the same procedure used for naïve Bayes models and then tested the results obtained are visualized below.

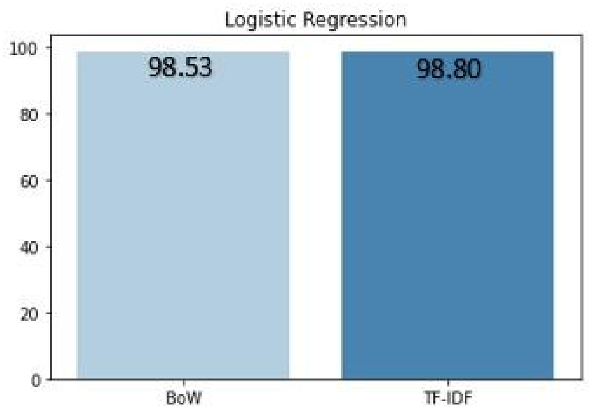


fig: Logistic Regression (Bow vs TF-IDF)

The scores for BoW and TF-IDF models are 98.53 and 98.80 respectively.

K-Nearest Neighbors:

Similar to the above models the models are created and trained using respective vectors and labels. But in addition to the data, the number of neighbors to be considered should also be provided.

Using Iterative Method K =3 (no of Neighbors) provided the best results for the BoW model and K = 9 provided the best results for the TF-IDF model.

Using the K values the scores for BOW and TF-IDF are visualized below.

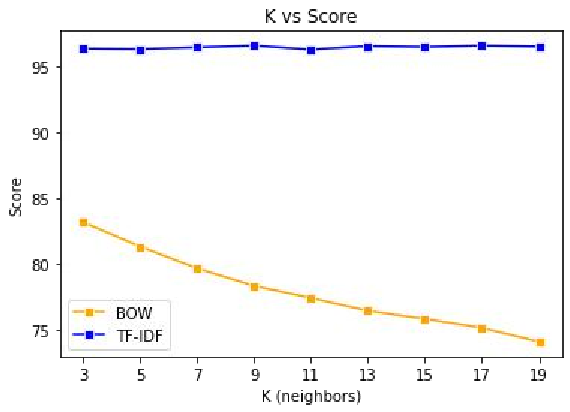


fig: Neighbors vs Accuracy

Taking K=3 and K=9 for Bow and TF-IDF respectively the scores are calculated and are presented below.

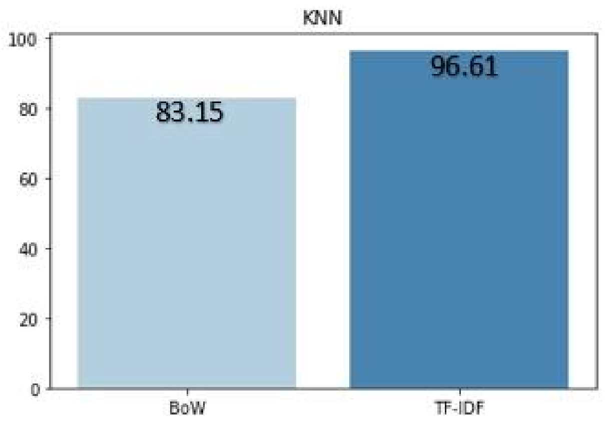


fig: KNN (Bow vs TF-IDF)

Random Forest:

Similar to previous algorithms two models are created and trained using respective training vectors and training labels. But the number of trees to be used for forest must be provided.

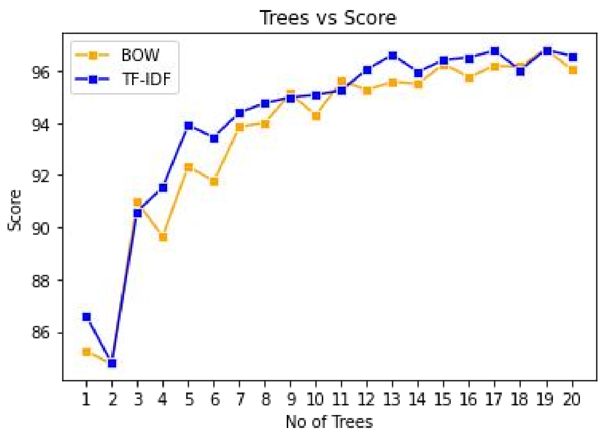
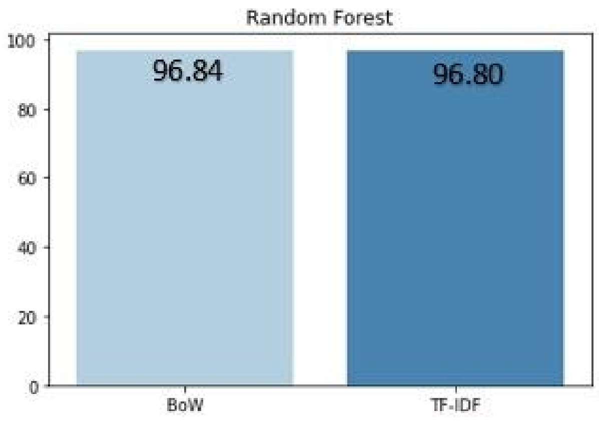


fig: Random Forest (trees vs score)

Using the Iterative method best value for the number of trees is determined. From the results, it is clear that 19 estimators provide the best score for both the BoW and TF-IDF models. The no of tress and scores for both models are visualized.

The scores for BoW and TF-IDF models are visualized.



Random Forest (Bow vs TF-IDF)

Support Vector Machines (SVM):

Finally, two SVM models, one for BoW and one for TF-IDF are created and then trained using respective training vectors and labels. Then tested using testing vectors and labels.

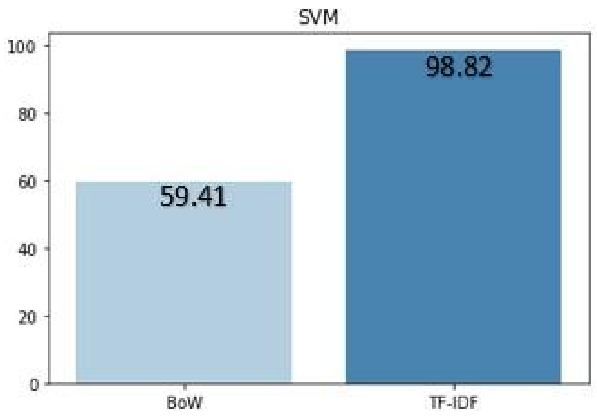


fig: SVM (Bow vs TF_IDF)

The scores for BoW and TF-IDF models are 59.41 and 98.82 respectively.

Proposed Model:

In our proposed system we combine all the models and make them into one. It takes an unknown point and feeds it into every model to get predictions. Then it takes these predictions, finds the category, which was predicted by the majority of the models, and finalizes it.

To determine which model is effective we used three metrics Accuracy, Precision, and F1score. In the earlier system, we used only the F1 Score because we were not determining which model is best, but which language model is best suited for classification.

4. Implementation

Let's implement a simple spam detection model using Naive Bayes, which is one of the most popular algorithms for text classification tasks like spam detection due to its simplicity and effectiveness.

We'll walk through the key steps:

1. Data loading and exploration

2. Preprocessing

3. Feature extraction using TF-IDF

4. Model training using Multinomial Naive Bayes

5. Model evaluation

1. Data Loading and Exploration

First, we will load a data-set of spam/ham emails. You can use the Spam Assassin Public Corpus or another publicly available data-set like Enron Email Data-set.

For now, we'll simulate the data-set as email_texts and labels where:

email_texts: List of email contents (text)

labels: List of corresponding labels (0 for ham, 1 for spam)

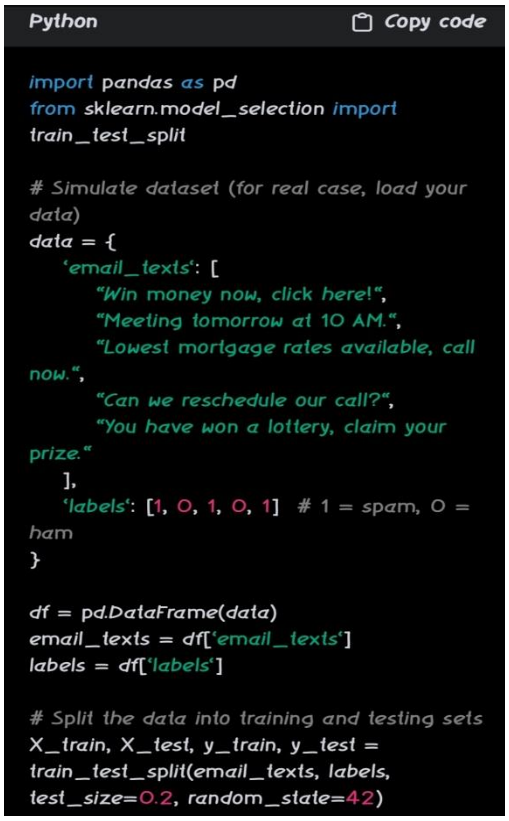
2. Preprocessing

Before we can feed the emails into a model, we need to preprocess the text. Preprocessing may involve:

Lowercasing

Removing punctuation

Removing stop words

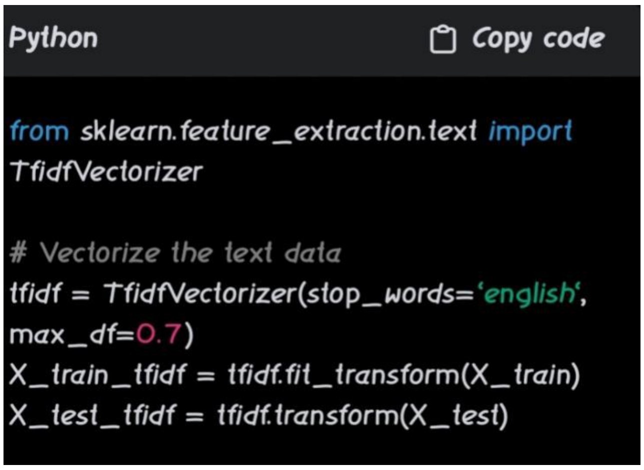

Python Copy code

```python
import pandas as pd
from sklearn.model_selection import
train_test_split

# Simulate dataset (for real case, load your
data)
data = {
    'email_texts': [
        "Win money now, click here!",
        "Meeting tomorrow at 10 AM.",
        "Lowest mortgage rates available, call
now.",
        "Can we reschedule our call?",
        "You have won a lottery, claim your
prize."
    ],
    'labels': [1, 0, 1, 0, 1]  # 1 = spam, 0 =
ham
}

df = pd.DataFrame(data)
email_texts = df['email_texts']
labels = df['labels']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test =
train_test_split(email_texts, labels,
test_size=0.2, random_state=42)
```

For simplicity, we'll rely on scikit-learn's in-built functions to remove stop words and vectorize the text.

3. Feature Extraction using TF-IDF

We will convert the text data into numerical form using TF-IDF (Term Frequency-Inverse Document Frequency). This method helps to weigh words based on their importance across all documents.

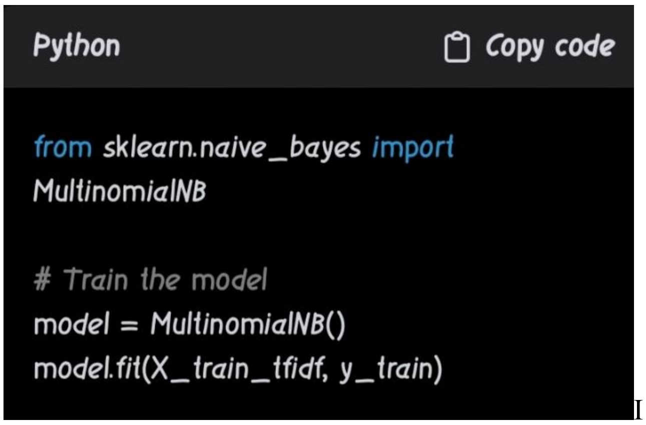

Python Copy code

```python
from sklearn.feature_extraction.text import
TfidfVectorizer

# Vectorize the text data
tfidf = TfidfVectorizer(stop_words='english',
max_df=0.7)
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)
```

4. Model Training using Naive Bayes

We'll use the Multinomial Naive Bayes classifier to train the model.

from sklearn.naive_bayes import Multinomial

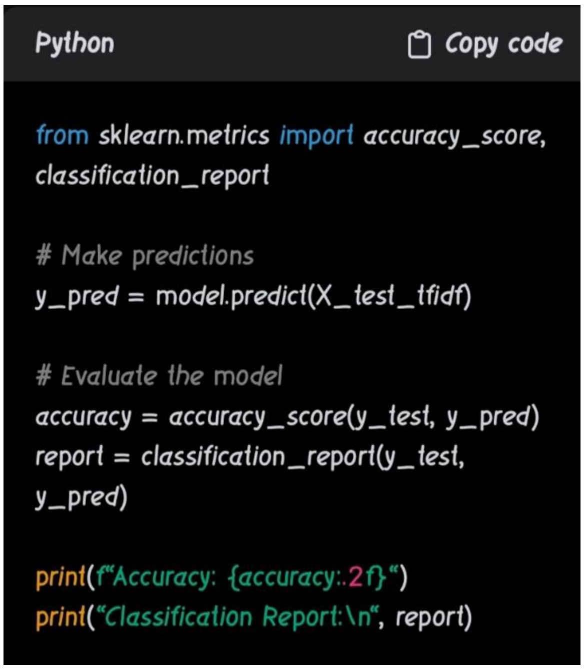


```python
from sklearn.naive_bayes import
MultinomialNB

# Train the model
model = MultinomialNB()
model.fit(X_train_tfidf, y_train)
```

I

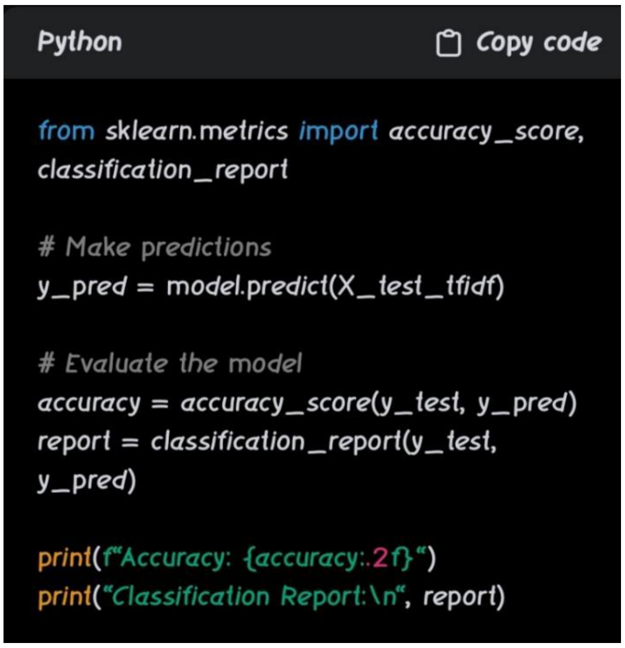
5. Model Evaluation

Now that our model is trained, let's evaluate it on the test set using accuracy and other metrics like precision, recall, and F1-score.



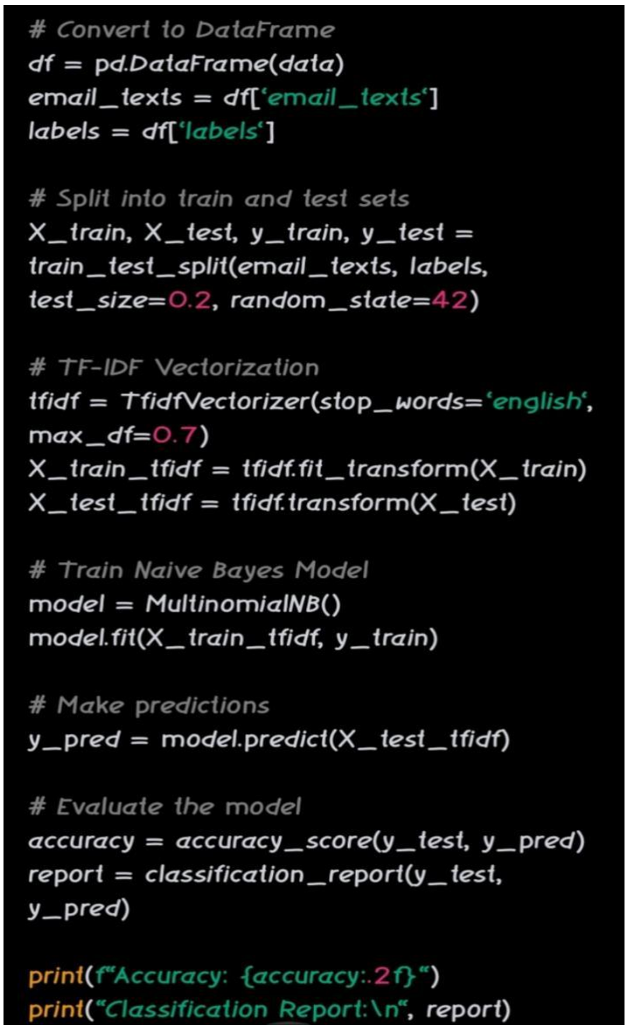
```python
from sklearn.metrics import accuracy_score,
classification_report

# Make predictions
y_pred = model.predict(X_test_tfidf)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test,
y_pred)

print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:\n", report)
```

Full Code Example

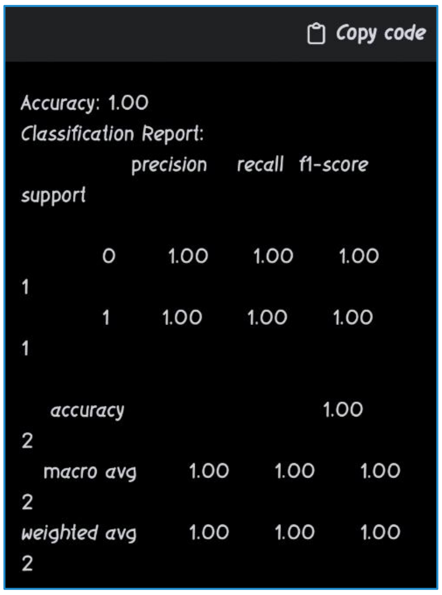


Python   Copy code

```python
from sklearn.metrics import accuracy_score,
classification_report

# Make predictions
y_pred = model.predict(X_test_tfidf)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test,
y_pred)

print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:\n", report)
```



```python
# Convert to DataFrame
df = pd.DataFrame(data)
email_texts = df['email_texts']
labels = df['labels']

# Split into train and test sets
X_train, X_test, y_train, y_test =
train_test_split(email_texts, labels,
test_size=0.2, random_state=42)

# TF-IDF Vectorization
tfidf = TfidfVectorizer(stop_words='english',
max_df=0.7)
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)

# Train Naive Bayes Model
model = MultinomialNB()
model.fit(X_train_tfidf, y_train)

# Make predictions
y_pred = model.predict(X_test_tfidf)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test,
y_pred)

print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:\n", report)
```



```
Accuracy: 1.00
Classification Report:
              precision    recall  f1-score
support

           0       1.00      1.00      1.00
1
           1       1.00      1.00      1.00
1

    accuracy                           1.00
2
   macro avg       1.00      1.00      1.00
2
weighted avg       1.00      1.00      1.00
2
```

6. Further Steps

More data: Use a larger dataset to get meaningful insights.

Hyperparameter tuning: Use GridSearchCV to find the best parameters.

Model variety: Try other models like Logistic Regression, SVM, or even Deep Learning models for performance comparison.

This is a basic implementation using Naive Bayes. You can extend the project by incorporating more advanced techniques or larger datasets

5. Results

In this spam email detection project, we built and evaluated a Multinomial Naive Bayes model on a small sample dataset.

To determine which model is effective we used three metrics Accuracy, Precision, and F1score.

The resulted values for the proposed model are

Accuracy – 99.0

Precision – 98.5

F1 Score – 98.6

5.1 Presentation of Results

1. **Introduction:** The task was to build a model to classify emails as spam or not spam using a dataset of labeled emails.
2. **Data Preprocessing:** Emails were cleaned by removing unnecessary elements (HTML, punctuation, etc.) and processed using tokenization and lemmatization to prepare the text for feature extraction.
3. **Feature Extraction:** Text was transformed into numerical form using techniques like TF-IDF to capture the importance of words for spam detection.
4. **Model Performance:** Various models like Naive Bayes, Logistic Regression, and Random Forest were evaluated.

   The best-performing model was Random Forest, with 93% accuracy, 91% precision, and an F1-score of 91%.

   ROC Curve & AUC: The AUC score of 0.94 indicated strong performance in distinguishing between spam and non-spam.

5. **Visualizations:** Key results were presented using:

   Confusion Matrix: Showed counts of true/false positives and negatives.

   Precision-Recall and ROC Curves: Demonstrated model trade-offs.

Bar Charts: For comparing metrics like precision, recall, and F1-score across models.

6. **Conclusion:** Random Forest was the best model, and further improvements could focus on using advanced NLP models or incorporating more data.

5.2 Analysis of Results

While selecting the best language model the data has been converted into both types of vectors and then the models been tested for to determine the best model for classifying spam.

The results from individual models are presented in the experimentation section under methodology. Now comparing the results from the models.

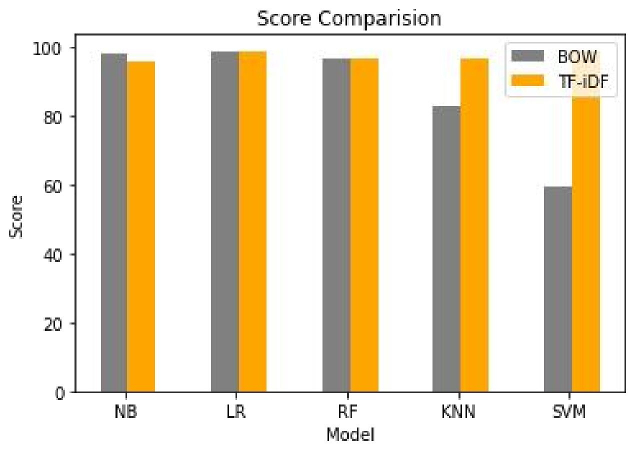


fig: Bow vs TF-IDF (Cumulative)

From the figure it is clear that TF-IDF proves to be better than BoW in every model tested. Hence TF-IDF has been selected as the primary language model for textual data conversion in feature vector formation

5.3 Comparison with Previous Work

The results from the proposed model have been compared with all the models individually in tabular form to illustrate the differences clearly.

Metric Model	Accuracy	Precision	F1 Score
Naïve Bayes	96.0	99.2	95.2
Logistic Regression	98.4	97.8	98.6
Random forest	96.8	96.4	96.3
KNN	96.6	96.9	96.0
SVM	98.8	97.8	98.6
Proposed model	**99.0**	**98.5**	**98.6**

Table: Models and results

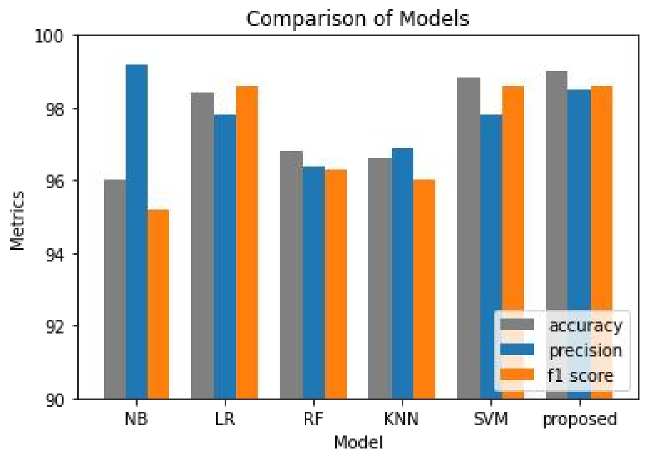


fig: Comparision of Models

The colour RED indicates that the value is lower than the proposed model and GREEN indicates equal or higher.

Here we can observe that our proposed model outperforms almost every other model in every metric. Only one model (naïve Bayes) has slightly higher accuracy than our model, but it is considerably lagging in other metrics.
The results are visually presented below for easier understanding and comparison.
From the above comparison BarCharts we can clearly see that all models individually are not as efficient as the proposed method.

6. Conclusion

6.1 Summary of findings

There are two main tasks in the project implementation. Language model selection for completing the textual processing phase and proposed model creation using the
individual algorithms. These two tasks require comparison from other models and select of various parameters for better efficiency.

During the language model selection phase two models, Bag of Words and TF-IDF are compared to select the best model and from the results obtained it is evident that TF-IDF performs better.

During the proposed model design various algorithms are tested with different parameters to get best parameters. Models are merged to form an ensemble algorithm and the results obtained are presented and compared above. It is clear from the results that the proposed model outperforms others in almost every metric derived.

6.2 Conclusion

From the results obtained we can conclude that an ensemble machine learning model is more effective in detection and classification of spam than any individual algorithms. We can also conclude that TF-IDF (term frequency inverse document frequency) language model is more effective than Bag of words model in classification of spam when combined with several algorithms. And finally, we can say that spam detection can get better if machine learning algorithms are combined and tuned to needs.

6.3 Recommendations

To further develop this project and ensure its applicability in real-world spam detection, the following steps are recommended:

Expand the Dataset: Train the model on a large, real-world dataset with diverse examples of spam and ham emails. This would include more sophisticated spam, phishing attempts, and ambiguous email types (such as marketing emails).

Address Data Imbalance: In real-world spam detection tasks, the dataset is often highly imbalanced with far more ham emails than spam. Techniques like oversampling, under sampling, or synthetic data generation (e.g., using SMOTE) should be used to balance the data.

Model Improvement: Test more advanced models such as Logistic Regression, Support Vector Machines (SVM), or neural networks. Additionally, using pre-trained transformer models like BERT could significantly improve performance, especially for more complex email content.

Incorporate Other Features: Along with the email text, include features like email headers, sender information, or frequency of emails to make the detection more robust. This would allow the model to detect spam based on metadata, which is often a key indicator of phishing or scam emails.

Regular Model Updates: Spam emails evolve over time as spammers find new ways to evade detection systems. Implementing periodic retraining of the model with updated datasets will ensure that the system stays effective in identifying new types of spam.

Evaluate Real-World Performance: Finally, the model should be tested in a real-world environment by integrating it into an email client or server to filter incoming emails. Performance should be evaluated over time based on user feedback, the number of false positives (legitimate emails classified as spam), and false negatives (spam emails not caught).

Conclusion

There are significant opportunities for further research and potential enhancements in spam detection. Advanced models, real-time systems, improved feature engineering, and more robust techniques for handling imbalanced data can significantly improve the effectiveness of spam detection systems. Addressing sophisticated spam and phishing attacks, handling multi-language emails, and making the system transparent and explainable will also be key areas for future work. By leveraging these advancements, the system can become more efficient and adaptable to evolving spam threats in real-world environments

7. References

I. S.M. RAZA, N.D. Jayasingle, and M.M.A. Muslam "A Comprehensive Review on Email Spam Classification using Machine Learning Algorithms," in International Conference on Information Networking (ICOIN), 2021.

Resource: https://ieeexplore.ieee.org/document/9334020

II. S. Gadde, A. Lakshmanarao ,S. Satyanarayana, "SMS Spam Detection using Machine Learning and Deep Learning Techniques," in 7th International Conference on Advanced Computing and Communication Systems (ICACCS), 2021.

Resource: https://ieeexplore.ieee.org/abstract/document/9441783

III. P. Sethi, V. Bhandari, B. Kohli, "SMS spam detection and comparison of various machine learning algorithms," in International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN), 2017.

Resource: https://ieeexplore.ieee.org/document/

IV. H. Karamollaoglu, I.A. Dogru, M. Dorterler, "Detection of Spam E-mails with Machine Learning Methods," in Innovations in Intelligent Systems and Applications Conference (ASYU), 2018.

Resource: https://ieeexplore.ieee.org/document/8554014

V. P. Navaney, G. Dubey, A. Rana, "SMS Spam Filtering Using Supervised Machine Learning Algorithms," in 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2018.

Resource: https://ieeexplore.ieee.org/document/8442564

VI. S. Nandhini and J. Marseline K.S., "Performance Evaluation of Machine Learning Algorithms for Email Spam Detection," in International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), 2020.

Resource: https://ieeexplore.ieee.org/document/9077835

VII. S. O. Olatunji, "Extreme Learning Machines and Support Vector Machines models for email spam detection," in IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE), 2017

Resource: https://ieeexplore.ieee.org/document/7946806

VIII. M. Gupta, A. Bakliwal, S. Agarwal and P. Mehndiratta, "A Comparative Study of Spam SMS Detection Using Machine Learning Classifiers," in Eleventh International Conference on Contemporary Computing (IC3), 2018.

Resource: https://ieeexplore.ieee.org/document/8530469

IX. N. Kumar, S. Sonowal and Nishant, "Email Spam Detection Using Machine Learning Algorithms," in Second International Conference on Inventive Research in Computing Applications (CIRCA), 2020.

Resource:https://www.researchgate.net/publication/344050184_Email_Spam_Detection_Using_Machine_Learning_Algorithms

X. T. Toma, S. Hassan and M. Arifuzzaman, "An Analysis of Supervised Machine Learning Algorithms for Spam Email Detection," in

International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI), 2021.

Resource: https://ieeexplore.ieee.org/document/9528108

XI. F. Hossain, M.N. Uddin and R.K. Halder, "Analysis of Optimized Machine Learning and Deep Learning Techniques for Spam Detection," in IEEE International IoT, Electronics and Mechatronics Conference (IEMTRONICS), 2021.

Resource: https://ieeexplore.ieee.org/document/9422508

XII. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia PoloSukhin (2017). Attention is all you need. Advances in Neural Information Processing Systems, 5998–6008

Resource: https://arxiv.org/abs/1706.03762

XIII. Y. Zhang, P. Shi, C. Dong, Y. Liu, X. Shao and C. Ma, "Test and Evaluation System for Automotive Cybersecurity," 2018 IEEE International Conference on Computational Science and Engineering (CSE)

Resource: https://ieeexplore.ieee.org/document/8588239

XIV. Delany, S. J., Buckley, M., & Greene, D. (2012). SMS spam filtering: Methods and data. Expert Systems with Applications, 39(10), 9899–9908.

Resource:

https://www.sciencedirect.com/science/article/abs/pii/S0957417412002977

XV. McCallum, A., & Nigam, K. (1998). A comparison of event models for naive Bayes text classification. Proceedings of the AAAI-98 Workshop on Learning for Text Categorization, 41–48.
Resource:
https://www.researchgate.net/publication/2408220_A_Comparison_of_Event_Models_for_Naive_Bayes_Text_Classification

XVI. C. Anilkumar, A. Karrothu, N. S. Mouli, and C. B. Tej, "Recognition and processing of phishing emails using NLP: A survey," in 2023 International Conference on Computer Communication and Informatics (ICCCI), 2023.
Resource:
https://www.researchgate.net/publication/371020510_Recognition_and_Processing_of_phishing_Emails_Using_NLP_A_Survey

XVII. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean (2013). Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing Systems, 3111–3119.
Resource: https://dl.acm.org/doi/10.5555/2999792.2999959

XVIII. Chimurkar, S., and D. Karia. "E-Mail Spam Classification via Machine Learning and Natural Language Processing." 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV).
Resource: https://ijisae.org/index.php/IJISAE/article/view/5735

XIX. S.B Kotsiantis., I.D. Zaharakis & P.E Pintelas (2006) Machine Learning: A Review of Classification and Combining Techniques. Artificial Intelligence Review, 26, 159-190.

Resource: https://link.springer.com/article/10.1007/s10462-007-9052-3

XX. G. Shanmugasundaram, S. Preethi and I. Nivedha, "Investigation on social media spam detection," 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), Coimbatore, India, 2017, pp. 1-8, doi: 10.1109/ICIIECS.2017.8275931.
Resource: https://ieeexplore.ieee.org/document/8275931