A dissertation submitted to the **University of Greenwich**
in partial fulfilment of the requirements for the Degree of

Master of Science

in

Data Science

Email Spam Classification

Name: Nij Hingrajiya

Student ID: 001278286

Supervisor: Dr. Jing Wang
Submission Date: 22nd December, 2023
Word count: 10,702

EMAIL SPAM CLASSIFICATION:

Computing & Mathematical Sciences, University of Greenwich, 30 Park Row, Greenwich, UK.

(Submitted 22nd December 2023)

Abstract: The incessant growth of spam emails in the digital communication era necessitates advanced solutions for efficient and accurate spam detection. This project "Email Spam Classification" is dedicated to the development of a sophisticated email spam classification system, leveraging the power of machine learning and natural language processing (NLP). The primary objective is to create a reliable model that can differentiate spam from non-spam (ham) emails, thereby enhancing the effectiveness of email filtering and help to protect users from phishing attacks, malware, and other security threats.

The project methodology unfolds in several key stages, starting with the acquisition and cleaning of a comprehensive email dataset. The data cleaning process is meticulous, involving the elimination of irrelevant features and the normalization of textual data. Following this, an extensive exploratory data analysis (EDA) is conducted to derive insights and patterns from the dataset, utilizing NLP techniques to extract critical textual attributes. The project then transitions into the model building phase, where a variety of machine learning classifiers are rigorously trained and evaluated. The classifiers include Naive Bayes, Decision Trees, Random Forest, and Support Vector Machine. Each model undergoes a thorough assessment based on metrics such as accuracy, precision, and F1 score, with hyperparameter optimization facilitated through GridSearchCV to fine-tune their performance.

The culmination of this project is a highly accurate and efficient spam classification model, streamlining the process of email filtering. This system represents a significant stride in tackling the prevalent issue of email spam, offering a practical tool for both individuals and organizations. The comprehensive report that follows details the methodologies employed, the evaluation of various models, and the steps taken for the successful deployment of the chosen model, serving as a valuable resource for further advancements in the field of email security and spam detection.

Keywords: Machine Learning, Spam Classification, Natural Language Processing, Email Filtering, Naive Bayes Classifier, Text Transformation, Random Forest Classifier, AdaBoost Classifier, Gradient Boosting Classifier, Hyperparameter Tuning, Tokenization, Stemming, Model Deployment, Cross Validation

Acknowledgements

I would especially like to thank Dr. Jing Wang for agreeing to be my supervisor and for his consistent advice, feedback, guidance and support throughout the lifecycle of this Msc data science project.

I want to thank both Dr. Jing Wang and Dr. Stef Garasto for agreeing to have the project demonstration on the schedule day.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

Background and Context:

Emails have become an essential component of everyday communication in the age of digital communication, for both personal and professional purposes. But with a greater reliance on email for communication comes a big problem is an abundance of unsolicited and frequently malicious spam emails. The project was designed to tackle this problem, realising that email classification needed to be accurate and efficient in order to improve security and user experience. This project's context is found in the larger field of cybersecurity and digital communication management applications of data science and machine learning.

Aims and Objectives:

The main goal of this project is to create a predictive model that can accurately identify emails as "spam" or "ham," which will help with email communication filtering. The particular goals to accomplish this goal are as follows.

Data cleaning and preprocessing of the dataset for emails to make it suitable for analysis. Analysing email data through exploratory data analysis (EDA) in order to find underlying trends and obtain new insights. applying and evaluating, in the context of email classification, the performance of several machine learning models, including Multinomial Naive Bayes, Decision Trees, Random Forest, and Support Vector Machines. To improve the accuracy and efficiency of these models, hyperparameter tuning methods such as GridSearchCV should be employed for optimisation. Ensuring the dependability and effectiveness of these models in email classification by assessing their performance using metrics such as F1 score, accuracy, and precision. evaluating the models' resilience and coherence among various dataset subsets by using cross-validation techniques. putting the final models in deployment mode and preparing them for real-world use in email communication filtering.

Methodology Overview:

An extensive email dataset was acquired and carefully cleaned and preprocessed before the email classification project was carried out using a well-defined and multifaceted methodology. An extensive exploratory data analysis (EDA) phase ensued, during which the email data was examined for patterns and insights. This process was critical in revealing the underlying dynamics of email classification. Text preprocessing was the next crucial step, wherein methods like tokenization, stopword removal, and stemming were used to polish the text and prepare it for machine learning analysis.

New features, like the quantity of characters, words, and sentences in emails, were created during the feature engineering stage. By giving the predictive models access to more detailed data, these features improved their predictive power. After that, a range of machine learning models were carefully chosen and trained to successfully classify the emails. In order to optimise the accuracy and performance of these models, a process known as hyperparameter tuning involved a thorough search for the ideal set of parameters using GridSearchCV.
The models' efficacy was subsequently subjected to a thorough assessment utilising critical performance metrics such as F1 score, accuracy, and precision. Cross-validation methods

were used on several dataset splits to make sure the models were trustworthy and not unduly adapted to particular data subsets. The models were serialised and their accompanying tools were made ready for practical deployment in email classification scenarios, marking the project's culmination and preparing the models for real-world application.

All things considered, the goal of this all-encompassing strategy was to create a reliable and effective system for classifying emails, tackling the difficulties and complexities of handling email data in the digital age.

Chapter 2: Literature Review

These days, a plethora of research papers and articles that can be useful for preliminary research are available on different platforms. These studies can clarify the basic concepts of the research I'm doing and offer valuable data.

I started by reading a research paper that explores the issue of using machine learning techniques to differentiate between spam (unwanted email) and ham (legitimate email). The study used the spam base UCI dataset to assess ten different classifiers. Radial Basis Function (RBF), Bayes Net, Naive Bayes, K-Nearest Neighbour (KNN), Support Vector Machine (SVM), Random Tree, Artificial Neural Network (ANN), Random Forest (RF), and Logistic Regression (LR) were some of the classifiers used. The Random Forest method proved to be the most effective, as per the study's findings (Bassiouni, Ali and El-Dahshan, 2018).

Evaluation and Critical Analysis:

Strengths of the Methodology: By utilising a large number of classifiers and a well-known dataset (spambase UCI), the study demonstrates a strong methodology and carries out an exhaustive assessment of machine learning approaches in email classification. Moreover, the implementation of a 10-fold cross-validation approach enhances the dependability and resilience of the study findings. This study compares several classifiers successfully and provides insightful information about how effective each one is in comparison. This comparative method improves my comprehension of the advantages and disadvantages of different machine learning techniques in the context of email classification. Interestingly, the study tackles a real-world issue—that is, the classification of spam emails—highlighting the usefulness of machine learning. This research is directly relevant and important given the increasing importance of email communication and the ongoing problem of spam.

However, it's important to recognise some shortcomings and possible areas for development. Although accuracy is the study's primary evaluation metric, other metrics like precision, recall, and the F1 score might provide a more nuanced understanding of classifier performance. Furthermore, the research might gain from expanding its dataset diversity or investigating more complex real-world scenarios, since the spam base UCI dataset might not adequately represent the complexity of modern spam email characteristics. Moreover, a discussion of these classifiers' computational efficiency would be beneficial, given the need for real-time processing in practical applications.

Even though the study uses pre-existing classifiers efficiently, there is still opportunity for alternative approaches to determine parameters, and fine-tuning for the optimal parameters may result in better outcomes.

Possible Applications to the Project:

Model Selection and Optimisation: This study, which focuses on the models' efficacy, can help choose machine learning models for email classification tasks. This research can serve as a standard for assessing novel spam detection methods or classifiers. This study's use of infinite latent feature selection for feature extraction shows potential and may be investigated in high-dimensional data projects to offer guidance on efficient feature selection.

Secondly, I came across an Empirical Study of Three Machine Learning Methods for Spam Filtering, carried out in 2007 by Chih-chin Lai. In this study, researchers examined the efficacy of three machine learning approaches in identifying spam emails. These methods included Support Vector Machine (SVM), K-Nearest Neighbour (k-NN), and Naive Bayes (NB). The objective was to determine which approach produced the best results and whether combining them would help. The study thoroughly examined these methods on various email sections, including the body, subject, and header (Lai, 2007).

Key Discoveries:

The Naive Bayes algorithm's performance demonstrates that K-Nearest Neighbour consistently performs well, particularly when it's not just limited to email body content. performs worse than other methods in general. Adding more preprocessing tasks improves performance. Additionally, Support Vector Machine (SVM) performs better than expected, especially when examining every aspect of emails. According to this study, combining various techniques for example, using SVM for classification and TF-IDF for feature extraction can produce acceptable outcomes that are on par with SVM used alone. Additionally, the study reveals that categorising emails according to their headers or subjects can be nearly as successful as utilising all of their features. On the other hand, performance suffers when email body is the only source of information.

Based on the critical analysis, the study highlights the potential of SVM and integrated approaches and offers insightful information about the efficacy of various machine learning techniques for spam filtering. For a more thorough comparison, the research would benefit from a wider range of algorithms. The creative approach of focusing on distinct email components provides helpful advice on which email elements are most suggestive of spam.

The research synthesizes various perspectives on spam filtering:

Reiterating previous research, Machine Learning Techniques highlights the increasing significance of machine learning in the fight against spam. It is consistent with research showing machine learning algorithms can reliably discriminate between spam and real emails. The literature is increasingly in agreement that there is no one technique that works for all spammers, and this is reflected in integrated approaches. It emphasises how combining various methods can lead to better outcomes.

What We Can Learn:

Algorithm Selection: The research will help us in selecting the most appropriate algorithms, particularly SVM and hybrid approaches, for spam filtering projects.

Selecting and optimising features: Understanding the key components of an email can aid in the development of more accurate spam detection algorithms.

Integrated Systems: Combining methods is a good way to create more sophisticated spam filters.

Refining Models: We can increase the accuracy of current spam detection models by incorporating the study's thorough analysis.

To sum up, this study adds to a more sophisticated understanding of how well various machine learning algorithms filter spam. It provides insightful information for creating spam detection systems that are more efficient by highlighting the potential of both integrated and standalone algorithms, such as Support Vector Machines (SVM).

Subsequently, I came upon a different study conducted by Tarun Jain at SCIT Manipal University in Jaipur. It tackles the common problem of spam messages in the context of SMS communication and looks into the application of machine learning algorithms and natural language processing (NLP) techniques to categorise messages as spam or legitimate (ham). The efficacy of four classification models—Naïve Bayes, Support Vector Machines (SVM), Logistic Regression, and Decision Trees—in the detection of spam is tested and compared. The dataset, which includes 5572 text message samples, was taken from the UCI Machine Learning Repository (Jain et al., 2022).

The first step in the methodology is data pre-processing. To prepare the text data for classification, the authors use a number of data pre-processing techniques, such as tokenization, stemming, stop word removal, and the use of N-grams. The goal of these methods is to transform the textual data into a machine-readable format. Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF), two popular word embedding techniques, are used to extract features from the text data. These techniques were selected based on how well they worked with the provided dataset. Four machine learning models—SVM, Logistic Regression, Decision Tree, and Naive Bayes are chosen for classification in the Model Selection process. The study includes a brief description of each model, emphasising how well-suited it is for various kinds of problems.

Results and Discussion:

Using terms like True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN), the study provides a thorough analysis of the model's performance. Each model's accuracy, precision, recall, and F1 score are assessed using these metrics. The best model, Support Vector Machines (SVM), achieves the highest accuracy (98.797%) along with the best F1 Score, Precision, and Recall. According to the study, SVM's geometrical approach to classification which goes beyond posterior probabilities is what makes it successful. The study comes to the conclusion that SVM is the most effective technique for identifying and categorising spam messages in this dataset, and it suggests using it to detect spam.

Critical Analysis of this study:

The study offers a thorough examination of numerous NLP and machine learning approaches in addition to a thorough overview of the issue of SMS spam detection. It effectively illustrates how much better SVM is in terms of performance metrics and accuracy. There are a few important things to keep in mind, though. For example, the dataset's diversity and representativeness are not discussed in the study. The study would be more credible if a more thorough explanation of the dataset's features such as message sources and potential biases was provided. To obtain high scores during pre-processing, lemmatization can be used instead of stemming. The study does not go into detail on feature selection or the dimensionality of the feature space, despite mentioning the use of BoW and TF-IDF. The effectiveness of machine learning models is greatly influenced by feature engineering, and further understanding of this procedure would be beneficial. The interpretability of the SVM model, which can be crucial to comprehending why it functions well, is not as well covered in the study. Explicitly stating which features influence the model's choices may yield insightful information.

Overall, the study offers insightful information about SMS spam detection; however, its contribution to the field would be strengthened by addressing the limitations mentioned and providing a more thorough discussion of the wider implications.

After that study, an approach for categorising spam that combines support vector machines (SVMs) and an active learning strategy is presented in the study "Spam Categorization Using Support Vector Machines and Active Learning Strategy" by the author. The efficient classification of emails as spam or non-spam is the aim of this study. In addition to summarising and critically analysing the study's main findings, this literature review also considers the study's possible applications (Li et al., 2003).

Key discoveries:

Definition of the Problem: Spam is defined as unsolicited electronic messages, and this study tackles the problem of spam emails. The Structural Risk Minimization principle from statistical learning theory, which underpins the application of SVMs for spam classification, is highlighted in the paper. SVMs seek to identify the ideal hyperplane that can effectively handle linearly separable data while minimising the true error. The study presents a novel approach to active learning called pool-based learning, in which learners can request labels for a subset of instances from a pool of unlabeled instances. We investigate this approach for choosing the most instructive examples to train the SVM.

The study suggests a technique to decrease the number of support vectors in order to speed up the classification process without compromising the performance of the SVM in order to address the computational complexity associated with SVMs. The transformation of email documents into appropriate machine learning representations is also covered in the paper, with a particular emphasis on features like word-based representations using TF-IDF. The significance of error rate, false alarm rate, and miss rate as performance criteria is also emphasised.

Critical Analysis of this study:

Strengths: The study combines SVMs with active learning, which can potentially improve spam categorization by selecting the most informative data points for training. The discussion on feature representation and the use of TF-IDF demonstrates a thorough understanding of text-based data preprocessing. The exploration of methods to simplify SVMs addresses the issue of computational efficiency.

The study lacks the empirical results and comparisons with other spam categorization methods, making it challenging to assess the actual performance of the proposed approach.The study does not address potential challenges, such as the choice of kernel functions for SVMs or the impact of feature dimensionality.

Chapter 3: Legal, Social, Ethical and Professional issues

Legal Concerns: There are a number of legal issues that arise with email spam classification projects. First and foremost, compliance with data privacy laws like the CAN-SPAM Act and GDPR is crucial. To preserve user privacy, sensitive data handling and personally identifiable information (PII) handling must abide by these laws. Furthermore, it's critical to define data ownership rights, particularly if the dataset includes private or business data. This avoids any legal issues and guarantees that have the right authorizations to use the data. Moreover, adherence to anti-spam legislation is essential. To avoid legal repercussions, the classification model should correctly distinguish between emails that are legitimate and those that are spam (Dada et al., 2019).

Social Issues: Bias and fairness are two of the main social issues with regard to the classification of email spam. It is possible for models to unintentionally exhibit bias by selectively excluding emails from particular regions or demographics. It is crucial to regularly assess the model for fairness in order to prevent discriminatory results. Another important factor is the user experience. When legitimate emails are mistakenly flagged as spam by overly aggressive spam filters, users may become frustrated and miss important messages. Furthermore, limiting accessibility for users who communicate in languages other than English may result from filtering non-English content.

Ethical Issues: Data security, transparency, and invasion of privacy are among the ethical factors taken into account when classifying email spam. Ethical questions about privacy invasion arise when users' email content is accessed for classification. To gain the trust of users, data usage and classification decisions must be kept transparent. Additionally, it is morally required to maintain data security in order to shield user information from potential breaches. Giving justifications for classification choices also helps these kinds of projects operate ethically (Dada et al., 2019).

Professional Issues: Thorough model validation and assessment are critical for anyone working in the field of email spam classification. Metrics such as accuracy, precision, and recall are evaluated to make sure the model performs as planned and satisfies industry standards. Sustaining accuracy and keeping up with changing spam tactics requires constant improvement. Strict guidelines for code quality and thorough documentation are necessary for cooperation, auditing, and confirming the behaviour of the model. It is a professional responsibility to be open to user feedback regarding classification and to respond quickly to false positives and negatives. Finally, adhering to industry standards and best practices will ensure that work is in line with expectations and norms within the profession.

Chapter 4: Methodology

Figure 1 illustrates the architecture of spam filtering. The model will first gather emails that are categorised as spam as well as valid emails from datasheet. This model includes initial transformation, feature extraction and selection, and email data classification. Finally, machine learning algorithms are used to train and test the requested email to determine if it is legitimate or spam.

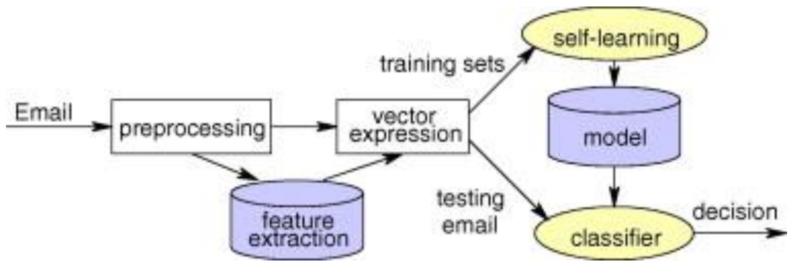


Figure 1 Process of Spam Filtering

4.1 Data Collection and Preparation

Data Loading and Initial Overview: The "Email Spam Classification" project's first step is to load the dataset from the CSV file into a Pandas DataFrame. The classification task is based on this dataset, which has 5,644 entries (spam and ham mails). Five columns are visible after a quick inspection: v1, v2, and three unlabeled columns. The email texts are in the v2 column, and the v1 column classifies emails as either'spam' or 'ham' (non-spam). There are a lot of missing values in the three unnamed columns, and most of the data is probably unrelated to the classification task.

Data Cleaning and Transformation: The unnamed columns, which are mostly filled with missing values, are eliminated in order to guarantee the accuracy and relevance of the data. In order to remove any possible noise from the dataset, this step is essential. Then, 'target' and 'text' are added to the names of the remaining columns, v1 and v2, to provide more precise descriptions of their contents. Label encoding is a crucial step in converting the categorical labels in the 'target' column into numerical values so that machine learning algorithms can process the data effectively. To enable them to be fitted by machine learning models that only take in numerical data, label encoding is a technique that converts categorical columns into numerical ones (Chugh, 2018).

The dataset is then subjected to a duplication check. The identification and removal of duplicate records, totaling 279 values, is done to prevent bias during the model training process. By taking this step, we can be sure that every dataset entry makes a distinct contribution to the learning process. The dataset's structure is lastly reassessed after cleaning to ensure that the data preparation procedures were carried out successfully. Now that the dataset has been refined, it offers a clear, organised, and significant set of data that is prepared for additional research and model building. Here is the pic chart(Figure 2) which is showing the portion of Ham and Spam mails.

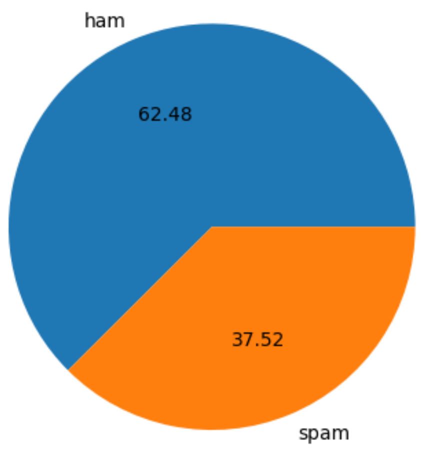


Figure 2 Ham and Spam Mails in Data

Ham: 62.48% of the data is classified as ham. This is the majority class in the dataset.

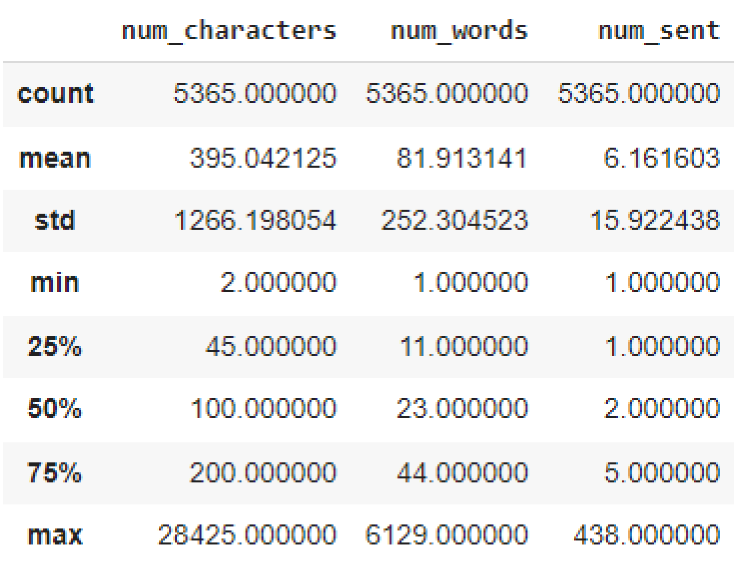
Spam: 37.52% of the data is classified as spam. This is the minority class but still a substantial portion of the data.

This implies that there is more ham than spam, which is slightly unbalanced. The meticulous attention to data preparation sets a strong foundation for the subsequent phases of the project, including exploratory data analysis and predictive modelling.

4.2 Exploratory Data Analysis (EDA)

Text Characteristics Analysis:

The emails are tokenized to separate the text into its component parts in preparation for a thorough text analysis that comes next. The project involves tokenizing the text and counting multiple metrics, including the length of each email in characters (num_characters), the number of words (num_words), and the number of sentences (num_sent), using the Natural Language Toolkit (NLTK). These metrics offer a summary of the textual features in the dataset and are not only computed but also statistically explained. This stage is essential for exposing the typical email lengths and complexity, which may differ significantly between spam and ham messages (GÜBÜR, 2021).



	num_characters	num_words	num_sent
count	5365.000000	5365.000000	5365.000000
mean	395.042125	81.913141	6.161603
std	1266.198054	252.304523	15.922438
min	2.000000	1.000000	1.000000
25%	45.000000	11.000000	1.000000
50%	100.000000	23.000000	2.000000
75%	200.000000	44.000000	5.000000
max	28425.000000	6129.000000	438.000000

Figure 3 Statistical summary of text data

From the statistical summary (Figure 3Figure 3), it is showing that the dataset contains 5,365 emails after preprocessing. Emails average 395 characters long, 82 words long, and comprise roughly six sentences each. A broad range in email lengths is indicated by these metrics' large standard deviations, particularly for the character count, which is typical of real-world email datasets.

Comparative Analysis of Ham and Spam Emails:

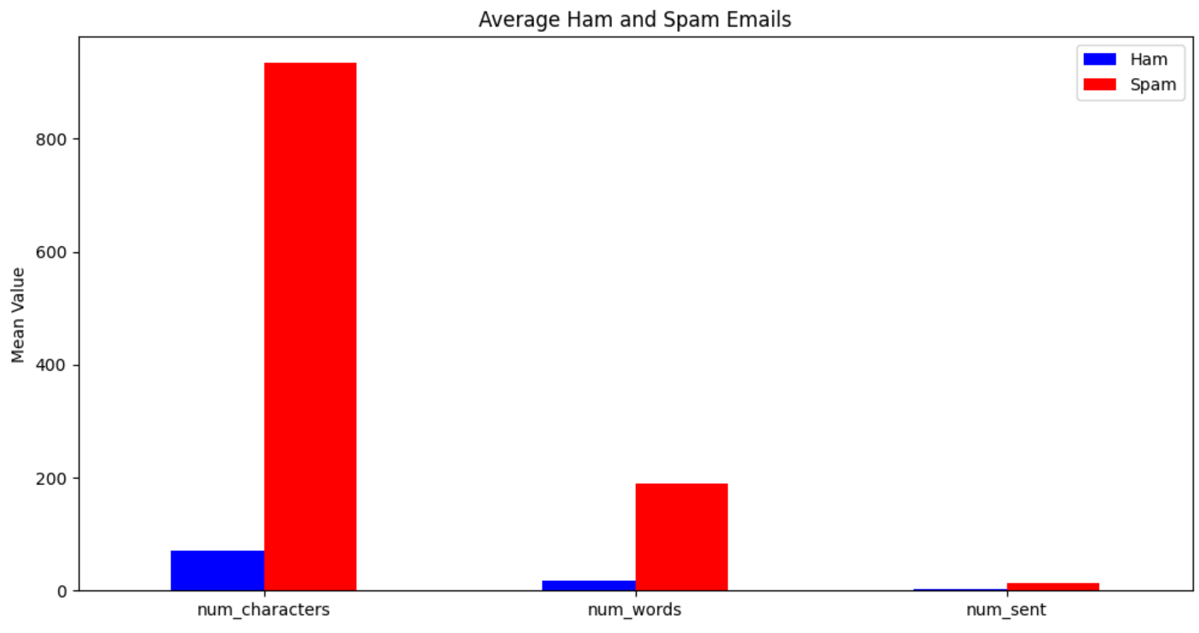


Figure 4 Comparison of Average Textual Characteristics

The bar chart is indicating the comparison of average textual characteristics reveals stark contrasts between ham and spam emails (Figure 4). Compared to ham emails, spam emails typically contain a significantly higher number of characters, words, and sentences. In particular, spam emails outweigh ham emails by having a mean character count that is significantly higher, which is also reflected in the quantity of words and sentences. This implies that spam emails are more likely to be longer and more sophisticated than ham emails, which could be a sign of their fraudulent or promotional nature. These characteristics, like the length of the email, the number of words, and the number of sentences, may be highly reliable indicators of spam from ham.

The EDA phase, with its methodical statistical and visual analysis, sets the stage for informed decision-making in the subsequent modeling phases. It ensures that the predictive features can be useful to train the classification models are backed by a clear understanding of the dataset's underlying properties. This strategy will improve the likelihood of producing a high-quality model while also making the spam detection process more transparent and comprehensible.

4.3 Data Preprocessing

The data preprocessing phase is crucial for preparing the raw email text data for the machine learning models. The primary focus is on transforming the raw text data into a format suitable for machine learning algorithms. This involves several steps, starting with text normalization.

Text Normalization: To make sure that words like "Email" and "email" are treated as the same token by the algorithm, each email is converted to lowercase. After dividing the text into separate words or tokens through tokenization, non-alphanumeric characters are eliminated to preserve only significant portions.

Stopword Removal and Stemming: Subsequent to text normalization, the preprocessing includes the removal of stopwords, common words that are unlikely to be useful for distinguishing between spam and ham emails. This is done using the NLTK library, which contains a predefined list of stopwords in English such as “I, me, mine, he, yours, and, the, but” etc (Geeksforgeeks, 2017). Then constructed a string removing punctuations such as “!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~”. After these are filtered out, stemming is applied to the words using the Porter Stemmer algorithm. Stemming reduces words to their root or the base form, which helps in consolidating the various forms of a word into a single representative form. For example, "walking," "walked," and "walks" would all be reduced to "walk."

This has done to decrease the vocabulary of the model and improve its performance, as it allows the model to treat related words as the same token, focusing on the essence of the word rather than its various morphological endings. Then, “transformed_text” has created to get the every processed text separately, which can be useful to vectorize this text for more appropriate to model.

Visualization with Word Cloud:

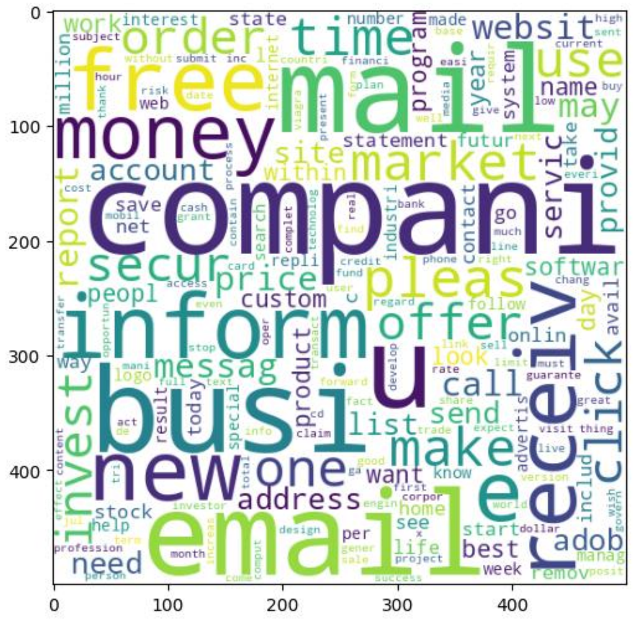


Figure 5 Word Cloud for Spam keywords

To visualize the most frequent words in spam emails, a Word Cloud is generated(Figure 5Figure 5). The illustration is helpful for quickly recognising well-known terms that might be indicative of spam. Words like "money," "offer," "market," "invest," and "free" stand out in this word cloud, indicating that they are frequently used in spam messages. In addition to giving a quick understanding of the content, this visual aid emphasises how crucial it is for the TF-IDF vectorizer to weigh these terms appropriately when preparing the dataset for the machine learning models.

Vectorization: After the text has been pre-processed, it is prepared for vectorization, which is converting the cleaned text into numerical data that can be processed by the machine learning algorithm. The TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer is employed in the project to achieve this. A statistical metric called "TF-IDF" assesses a word's relevance to a document within a group of documents. It is favoured over straightforward count-based techniques because it takes into account both the word's frequency in a single document (term frequency) and its overall usage (inverse document frequency). This aids in prioritising words that are specific to a given list and gives them a higher weight, which may provide additional insight into what constitutes spam or ham in a document (Scikit learn, 2018).

On the other hand, "Count Vectorizer" handles every word the same way and disregards word frequency or importance in different emails. It is simple to use and can be useful for basic text analysis tasks, but it might not be as good at identifying the subtleties of spam emails, which frequently contain particular keywords or patterns.TF-IDF is therefore a better option for this model since it considers word importance and has a higher chance of successfully capturing terms associated with spam.

Following this consideration, the dataframe df's "transformed_text" column is subjected to a call to the "fit_transform" method. After learning the vocabulary and inverse document frequency weightings, this method transforms the text data into a matrix of TF-IDF features by applying the transformation to the 'transformed_text' column.

After the transformation, the output is converted to an array using toarray(), resulting in a numerical array called x that accurately represents all of the text data in a format that can be processed by machine learning models.

In target variable extraction, "y = df['target'].values" extracts the target variable from the dataframe df. The 'target' column contains the labels which are the outputs we want our model to predict (e.g., whether an email is spam or not spam).

The values attribute is used to get the representation of the column as a numpy array, which is a suitable format for machine learning algorithms.

The preprocessing stage is meticulously designed to convert raw text into a refined, numerical format suitable for effective model training. By carefully applying text normalization, tokenization, stop word removal, stemming, and TF-IDF vectorization, this ensures that the dataset is optimized for uncovering patterns that distinguish spam from non-spam emails. This rigorous preparation is fundamental to the success of the email spam classification task that follows.

4.4 Data Splitting

A train test split occurs when the data is divided into a training set and a testing set. The training set is used to train the model, whereas the testing set is used to test it. It allows to train the models on the training set before assessing their accuracy on a new testing set (Shiksha.com, 2020).

As the dataset is found to be imbalanced, which is a common issue in classification problems. An imbalanced dataset can lead to a biased model that overpredicts the majority class. To mitigate this SMOTE method has been used to artificially augment the minority class by generating new, synthetic samples. This is done to create a more balanced dataset that improves the classifier's ability to learn from both classes equally (SATPATHY, 2020).

To split the data into training and testing sets, the "train_test_split" function from "sklearn.model_selection" is imported. To use the over-sampling method, import the SMOTE class from "imblearn.over_sampling". The SMOTE object is now created and applied to the target variable y and features x of the dataset. This object's "fit_resample" function is used to oversample the minority class in order to balance the dataset. Consequently, the balanced feature set and target variable are contained in x_resampled and y_resampled, respectively.

80% of the data is used for training, and 20% is used for testing (with a test size of 0.2). To split, simply set the random state to 2.

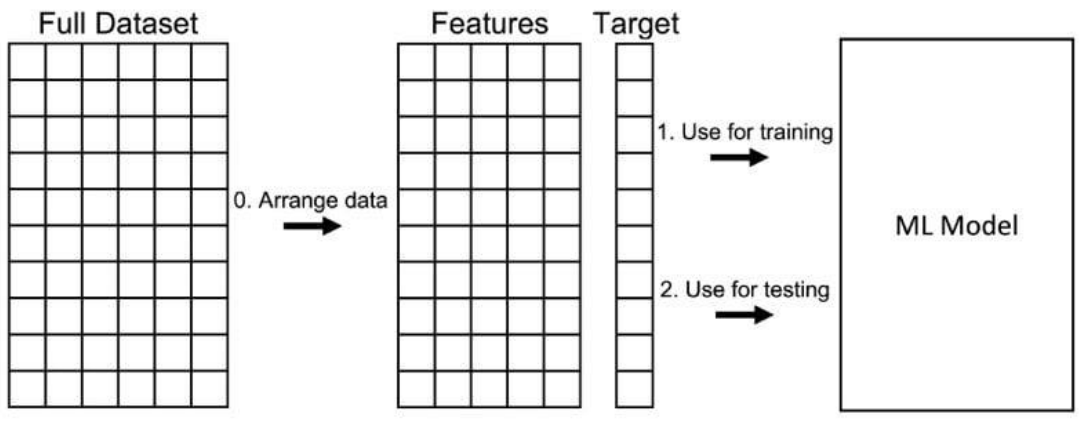


Figure 6 Train-Test Split

4.5 Model Architecture

1) Naive Bayes classification:

Naive Bayes classification is a fundamental approach used for applications such as detecting email spam. It's a strong tool for automatically determining whether emails are spam or not. This technique is based on a theory known as Bayes' theorem, which assesses the likelihood of an email falling into a specific category (spam or not) depending on the words or properties it includes (IBM, 2023).

The simplifying premise that every word in an email is completely independent of every other word's existence is the source of the "naive" part of Naive Bayes. However, since word usage in emails can be connected, this assumption may not always be true. Nevertheless, in practice, Naive Bayes performs surprisingly well even with this simplification (Ray, 2019).

The fundamental equation used in Naive Bayes classification is based on Bayes' Theorem. Here's the equation:

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$$

Where,

$P(C|X)$ is the posterior probability of class C given the input data X. In the context of my model, this is the probability that an email belongs to a specific class (e.g., spam or not spam) given the words or features in the email.

$P(X|C)$ is the likelihood, which represents the probability of observing the input data X given that it belongs to class C. This part of the equation models how likely it is to see certain words in an email of a particular class.

$P(C)$ is the prior probability of class C, which represents the probability of an email belonging to class C before any evidence from the input data is considered. It's essentially the probability of an email being spam or not spam based on the overall distribution of emails.

$P(X)$ is the marginal likelihood or evidence, which is the probability of observing the input data X regardless of the class. This part normalizes the equation and ensures that the posterior probability is a valid probability distribution.

This formula, as applied to this project, determines an email's likelihood of being spam (or not) based on the words it contains and the general distribution of spam and non-spam emails in the training dataset. The anticipated class for the email is the one with the highest posterior probability.

It's crucial to remember that the "naive" part of Naive Bayes derives from the assumption that, given the class, the features in this case, words are conditionally independent. Although the model can compute these probabilities quickly thanks to this simplifying assumption, it might not always hold true in real-world situations. In spite of this simplification, Naive

Bayes classifiers frequently exhibit outstanding performance in a variety of classification tasks.

2) Decision Tree Classifier:

A popular machine learning method for classification and regression applications is the Decision Tree Classifier. This non-parametric supervised learning technique creates subsets of input data according to the values of its attributes. A tree-like structure of decision nodes, each of which indicates a choice based on an input attribute, is produced as a result of these partitions.

In my project, I am using a Decision Tree Classifier to classify emails as spam or not spam. Here's how it works:

Splitting: The algorithm starts with every email as the root node of the tree. It selects a threshold (a splitting criterion) and a feature (a word or group of words). As illustrated in Figure 7, the dataset is partitioned according to whether the given characteristic meets the threshold.

Recursive Partitioning: The splitting procedure is carried out recursively for every subset, producing child nodes. The algorithm selects these child nodes' attributes and thresholds based on the remaining data. Until a stopping criterion like a maximum tree depth or a minimum number of samples in a node is met, this process is repeated.

Classification: After the tree is built, new emails that have never been seen before are categorised by going from the root node to a leaf node in the tree. At every node, a decision is made based on the feature values. The algorithm then moves on to the relevant child node until it reaches a leaf. The leaf node's class spam or non-spam is the prediction.

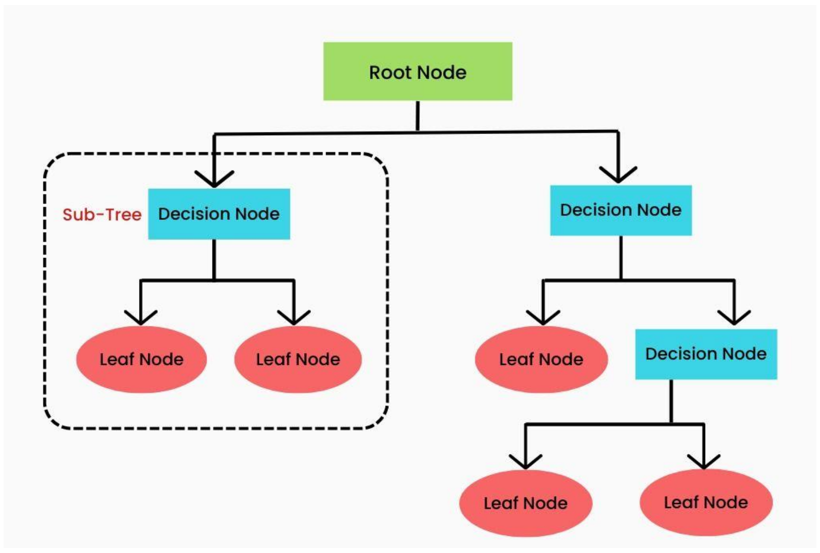


Figure 7 Structure of Decision Tree

Splitting Criteria: The decision tree method employs a variety of criteria to choose the best possible feature and threshold for data splitting at each node. Gini Impurity and Entropy are two typical criteria.

Entropy measures the amount of disorder in a set, whereas Gini impurity estimates the likelihood of misclassifying cases. Both are used to identify node splits in decision trees, while Gini prefers larger partitions (Aznar, 2020). Here is the equation for both:

Gini Impurity for classification:

$$Gini(D) = 1 - \sum_{i=1}^{C} P_i^2$$

Entropy for classification:

$$Entropy(D) = -\sum_{i=1}^{C} P_1 \cdot \log_2(p_i)$$

Where,

D is a dataset at a particular node.

c is the number of classes such as spam and not spam.

p_i is the probability of an email belonging to class i in the dataset D.

Recursive Splitting: Based on the chosen feature and threshold, the algorithm recursively divides the dataset into subsets using these criteria. In order to effectively increase the homogeneity of the classes in each partition, the objective is to minimise the impurity or entropy at each node.

This Decision Tree Classifier is used to automatically determine whether or not an email is spam based on its content. It uses word and pattern analysis to create a decision-tree structure from emails. This classifier helps to accurately and precisely classify emails as spam or not by recursively splitting and partitioning the dataset. It is therefore a crucial component of automating the spam filtering procedure.

3) Random Forest Classifier:

The Random Forest Classifier is an ensemble learning method that builds a collection of decision trees and combines their outputs to make predictions. It is capable of handling large datasets with high dimensionality. It enhances the accuracy of the model and prevents the overfitting issue (Ramesh, 2023).

Here's how it works within the context of my project:

Using a random portion of the email dataset as training data, Random Forest creates an ensemble of decision trees. These subsets are used as training data for individual trees and are chosen using replacement from the original data (a process known as bootstrapping). The model gains diversity from this randomization, which lowers the likelihood of overfitting.

By only looking at a random subset of characteristics (phrases or phrases) at each node of the tree, Random Forest adds even more randomness to each decision tree. This feature subsampling reduces the likelihood of individual trees overfitting to particular features by ensuring that no single feature dominates decision-making.

Voting: When it comes time to make a forecast, whether an email is spam or not, each

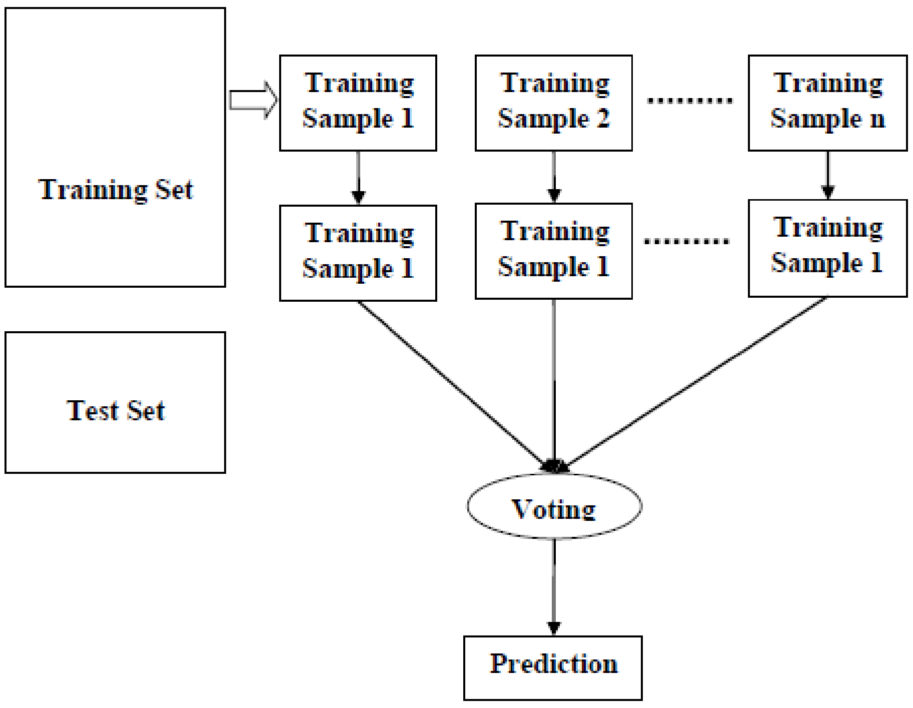


Figure 8 Random Forest Workflow

decision tree in the ensemble makes its prediction. The final prediction in a classification task is determined by the majority vote among the trees as we can see in the Figure 8Figure 8 below. In other words, the Random Forest will classify an email as spam if the majority of trees predict that it is, and this will take into account the majority prediction.

Improved Generalisation: Random Forest's strength comes on its ability to generalise well. It decreases the risk of overfitting by merging predictions from numerous decision trees, which can be an issue when working with this type of email dataset. Random Forest uses the collective wisdom of the community to make more accurate and consistent predictions (Donges, 2021).

In summary, the Random Forest Classifier is a useful technique for classifying emails that are spam. It produces strong forecasts, reduces overfitting, and offers insights into feature importance by utilising multiple decision trees.

4) Support Vector Machine:

Strong supervised machine learning algorithms like Support Vector Machine (SVM) can be especially useful for classifying emails as spam, especially when the data is roughly linear as in my case. Vapnik proposed a statistical theory that serves as the foundation for the SVM approach, which is used for pattern classification. Both linearly and nonlinearly separable features can be effectively handled by SVM. The main challenge is to identify a function that successfully classifies the input features while lowering error (Bassiouni, Ali and El-Dahshan, 2018).

SVM has the advantage of using the maximum margin hyperplane concept, which allows it to classify tested input features with high accuracy. Sequential Minimal Optimisation (SMO) and polynomial kernel function have been applied to improve the classifier performances for small sample learning problems (Lin et al., 2011).

SVM is used in my situation to classify emails as spam by identifying a hyperplane that maximises the margin between the two classes (spam and ham emails) and best divides them. The decision boundary is known as a hyperplane, and the objective is to identify the one that maximises the margin between the two classes' closest data points, or support vectors.

Margin: The margin is the separation between the nearest data points in each class and the hyperplane. Selective Margin Mapping (SVM) aims to find the hyperplane that maximises this margin because a bigger margin usually leads to better generalisation and less overfitting.

Support vectors are the data points that are closest to the decision boundary (hyperplane). They play a crucial role in determining the margin and the decision boundary.

Kernel Trick: Even in cases where the data cannot be separated linearly, SVM can still function well when a kernel function is applied. Thanks to the kernel function, SVM can transform the input features into a higher-dimensional space where the data may become linearly separable. Common kernel functions include the polynomial, linear, and radial basis function (RBF) kernels. The RBF kernel function determines the similarity, or how close two points are to one another, for X_1 and X_2 (Sreenivasa, 2020).

Mathematical Equation of Support Vector Machine (SVM):

For a classification problem like email spam classification, the training dataset with feature vectors (X) and corresponding labels (Y), where Y can be either -1 (for ham) or 1 (for spam), the SVM aims to find a hyperplane represented by:

$$w^T \cdot x + b = 0$$

Where,

w is the weight vector that defines the orientation of the hyperplane.

b is the bias term which is also known as the intercept.

The margin is defined as the distance between the hyperplane and a support vector (for either class)

$$\frac{w^T \cdot X + b}{\|w\|}$$

The objective of SVM is to maximize this margin while ensuring that all data points are correctly classified, subject to the constraint.

$$Y_i \cdot (w^T \cdot X_i + b) \geq 1$$

This ensures that each data point is correctly classified by the SVM. If Y_i is -1, it means that X_i is expected to be on one side of the decision boundary (hyperplane), and if Y_i is 1, it means that X_i is expected to be on the other side of the decision boundary.

The optimization problem can be formulated as,

$$Minimize \frac{1}{2} \|w\|^2 \ subject\ to\ Y_i \cdot (w^T \cdot X_i + b) \geq 1\ for\ all\ i$$

solving this optimization problem results in finding the optimal values of w and b that define the hyperplane with the maximum margin while correctly classifying the data points (Vedaldi and Zisserman, 2012).

In summary, SVM is a versatile algorithm that works well for email spam classification when the data is roughly linearly separable. It aims to find the best hyperplane (decision boundary) that maximizes the margin between classes while correctly classifying data points. It can also handle non-linear data by using appropriate kernel functions.

4.6 Model Building and Hyperparameter Tuning

In this section, I will build and train several machine learning models to classify emails as spam or ham (non-spam). Then using the different classifiers such as Multinomial Naive Bayes (MNB), Decision Tree Classifier (DTC), Random Forest Classifier (RFC), Support Vector Machine (SVM) and tune their hyperparameters to achieve the best performance.

First of all, it is begin by importing the essential libraries required for building and training machine learning models. These libraries include scikit-learn's classifiers such as Multinomial Naive Bayes, Decision Tree, Random Forest, and Support Vector Machine (SVM). Additionally, also imported GridSearchCV, a crucial tool for hyperparameter tuning. After that, instances of the machine learning models are created. These instances act as templates for the models to be fine-tuned with the best hyperparameters.

Workflow of GridSearch: The first step in using GridSearchCV is to define a hyperparameter space and list the possible values or ranges for each hyperparameter. For example, the splitting criterion, maximum tree depth, and minimum samples for splitting and leaf nodes are taken into account when using the Decision Tree classifier. Based on the specified space, GridSearchCV then creates a grid with every possible combination of these hyperparameters.

Cross-validation is used in this to ensure accurate model evaluation. The dataset is divided into folds, and the model is trained on the folds that remain after one is designated as a validation set. This process is repeated for each fold, and the performance metric like accuracy or F1-score is computed. Cross-validation guards against overfitting and ensures that the model's performance is unaffected by a specific data split (Mujtaba, 2020).

This helps with model optimisation by determining which hyperparameters are optimal for each classifier. This guarantees that when classifying email messages as spam or ham, classifiers are set up to perform at their best. In the end, GridSearchCV is a crucial tool for my project since it allows me to systematically fine-tune models and obtain the best classification.

Discussing the hyperparameter tuning for the different models that I have used in and what each parameter signifies within each model according to my output as in Table 1:

Model Name	Best Hyperparameters from GridSearchCV
Decision Tree Classifier	'criterion': 'gini', 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 10
Support Vector Machine	'C': 1, 'kernel': 'rbf', 'degree': 2, 'gamma': 'scale'
Random Forest Classifier	'max_depth': None, 'n_estimators': 100
Multinomial Naive Bayes	'alpha': 0.1

Table 1 Table of Best Hyperparameters

For Decision Tree Classifier:

Criteria: This parameter establishes the function that gauges a split's quality. We can think about using 'entropy' for information gain and 'gini' for the Gini impurity. Entropy is, in my opinion, the best parameter output because it aids in the creation of a tree that maximises the separation of classes.

max_depth: The decision tree's maximum depth indicates the number of branching levels and decision nodes that the tree is capable of having. Although a deeper tree may overfit the training set, it can capture intricate patterns. This yields 'None' in my case, allowing it to expand as deeply as necessary to capture intricate patterns in the data.

min_samples_split: This parameter specifies the minimum number of samples required to split an internal node further. If a node has fewer samples than this value, it won't split, preventing overly complex trees.

min_samples_leaf: This variable determines the bare minimum of samples required to establish a leaf node within the tree. In order to prevent overfitting, leaf nodes with fewer samples may be pruned during tree construction. Since it was set to 1 in my instance, each leaf may contain a maximum of one sample.

For Support Vector Machine (SVM) Classifier:

C: The trade-off between minimising classification errors and maximising the margin is managed by the regularisation parameter 'C'. A larger 'C' enforces a stricter boundary, while a smaller 'C' permits a wider margin and may tolerate some misclassification. For this case, the value of 'C' is 1.0. The trade-off between maximising the margin and reducing classification errors is managed by this parameter (Velocity Business Solutions Limited, 2020). A value of 1.0 indicates a balanced approach to margin and error. According to my case, a higher value of C tells the SVM to choose a smaller margin hyperplane, which means that more data points will be correctly classified. This can be beneficial in imbalanced datasets where we want to ensure that the minority class is well-represented.

kernel: The choice of the kernel function influences how the SVM maps data into a higher-dimensional space to find a separating hyperplane. Options include 'linear,' 'poly' for polynomial, 'rbf' for radial basis function, and 'sigmoid' for a sigmoid function. According to this datasheet, it has suggested that ‘rbf’ is the best as it handles imbalanced data. For that, weight class is set to balanced in order to balance the data.

degree: The degree of the polynomial kernel function is determined by the 'degree' parameter when utilising the 'poly' kernel. Although they may also make the model more complex, higher degrees can capture more intricate relationships.

Gamma: The parameters "gamma" and "poly," "sigmoid," and "rbf" relate to kernels. It regulates how the decision boundary is shaped. Smoother boundaries are produced by smaller gamma values, while more complex boundaries are produced by larger values. In the case of imbalanced data, tuning gamma is crucial to ensure that the model does not overfit on the majority class.

For Random Forest Classifier:

n_estimators: The number of decision trees, or estimators, in the random forest ensemble is indicated by this parameter. Though it also lengthens computation times, adding more trees can enhance model performance. Thus, to reduce overfitting and produce predictions, 100 decision trees (estimators) collaborate.

max_depth: The maximum depth of each decision tree inside the random forest is specified by the'max_depth' parameter. It can assist in limiting the intricacy of individual trees and avoiding overfitting. However, the random forest's individual decision trees do not have a specified maximum depth ('None'). In order to capture the predictions, this setting permits the trees to grow as deeply as necessary.

max_features: “sqrt” in a Random Forest Classifier is a strategy to introduce randomness and diversity into the decision tree building process. It helps improve the generalization ability of the ensemble model and is a common choice for this hyperparameter in practice. Also I have tried “auto”, which is same but using different type of method to generating the trees.

For Multinomial Naive Bayes (MNB) Classifier:

alpha: When estimating likelihoods, the smoothing parameter alpha is used to prevent zero probabilities. Greater alpha values produce more regularisation and smoothing, while smaller values produce less, which may cause overfitting. As a result, stronger smoothing is produced with a higher alpha of 5.0, which helps avoid overfitting and takes into account features not visible in the training set.

These best hyperparameters were determined through the Grid-Search process to optimize the performance of each model. After extracting the best model parameters, These parameters are then will be use to assign in the model.

4.7 Model Fitting

Model fitting is a critical process in the project that establishes how well the various machine learning models (Multinomial Naive Bayes, Decision Tree Classifier, Random Forest Classifier, and Support Vector Machine) adjust to and learn from the training data of labelled email messages classified as either 'ham' (non-spam) or'spam'.

It begins by initializing four different machine learning models, each with specific hyperparameters chosen likely as a result of prior hyperparameter tuning as shown in Table 1.

Firstly, a dictionary called ‘clfs’ is defined. This dictionary contains four different machine learning models, each identified by a unique abbreviation (key) and associated with an initialized model (value). Each model has been previously initialized with the best hyperparameters and configurations obtained during the hyperparameter tuning process. These models are ready for training and evaluation.

After doing that, 'train_classifier' function has been created that streamlines the process of fitting each classifier to the training data and then assessing its performance on the test data.

This function takes several parameters, including the model to be trained (clf) and the training and testing datasets (x_train, y_train, x_test, and y_test). Within this function, the classifier is trained on the training data using the clf.fit method, and predictions are generated for the testing data using ‘clf.predict’. And then, three important scores such as accuracy, precision, and F1-score are computed to assess the model's effectiveness in classification tasks. In this, by adding the variable name of the model, it will provide these three scores.

To systematically evaluate the performance of each model, three empty lists has been created will be used to append the scores of each model at the end. A loop iterates through the clfs dictionary, which contains the model abbreviations as keys and their associated initialized models as values. During each iteration, the ‘train_classifier’ function is invoked with the current model, and the calculated accuracy, precision, and F1-score are printed. Additionally, these performance metrics are appended to their respective lists for later analysis and comparison.

In summary, this facilitates the automated training and evaluation of multiple machine learning models, allowing for a comprehensive assessment of their performance on the dataset. The models have been carefully configured with optimized hyperparameters, which is efficiently computes and presents key performance metrics to assist in the selection of the most effective model for the classification task at hand.

4.8 Model Evaluation

The process of analysing a machine learning model's performance, as well as its advantages and disadvantages, using a variety of evaluation metrics is known as model evaluation. Model evaluation, which is a component of model monitoring, is essential for assessing a model's efficacy in the early phases of research (GeeksforGeeks, 2023).

The model's performance analysis is crucial, and various important terms and metrics have been established for this reason. The following metrics are essential for assessing how well email spam classification model is working:

True Positive (TP): TP stands for the instances in which the model's predicted value (value 1) and the expected value (ground truth) are both 1, signifying that the model correctly categorised an email as spam.

True Negative (TN): A TN indicates that the model correctly classified an email as not spam (ham) when both the expected value and the predicted value are 0.

False Positive (FP): FP is the result of the model predicting a value of 1 (spam) when the expected value is 0 (not spam). To put it another way, it represents the instances in which the model confused a ham email for spam.

False Negative (FN): FN occurs when the model predicts a value of 0 (ham) when the expected value is 1 (spam). It shows the situations in which the model was unable to accurately identify an email as spam.

I have defined the following performance metrics based on these terminologies:

Accuracy: The number of emails that the model correctly identified as spam or ham is known as accuracy. It is determined by dividing the total number of instances (TP, TN, FP, FN) by the ratio of correctly classified instances (TP and TN).

Precision quantifies the model's predictive accuracy for spam. It is calculated as the ratio of true positives (TP) to the total number of positive predictions the model produced (TP + FP). It provides insight into how effectively the model lowers false positives (Jain et al., 2022).

F1 Score: The harmonic mean of recall and precision is the F1 score. It integrates the two measures to offer a fair assessment of the model's effectiveness. It is computed as two times the recall and precision products divided by the sum of the two.
So, these metrics made it easier to evaluate the models' overall efficacy.

Accuracy, precision, and F1 score are among the performance metrics for four distinct machine learning models such as Random Forest Classifier, Support Vector Machine (SVM), Decision Tree Classifier, and Multinomial Naive Bayes (MNB). These are presented in the included table(
Table 2). This is my feedback regarding the result:

Model Name	Accuracy	Precision	F1 score
Random Forest Classifier	0.978374	1.000000	0.978310
Support Vector Machine	0.977629	0.993949	0.977679
Decision Tree Classifier	0.948546	0.947522	0.949598
Multinomial Naive Bayes	0.909023	0.858238	0.916780

Table 2 Table of Performance Metrics

After getting the values of there metrics I have created the graph which is showing(Figure 9) the comparison of four machine learning algorithms such as Random Forest (RF), Support Vector Machine (SVM), Decision Tree (DT), and Naive Bayes (NB), based on three different performance metrics: Accuracy, Precision, and F1 score.

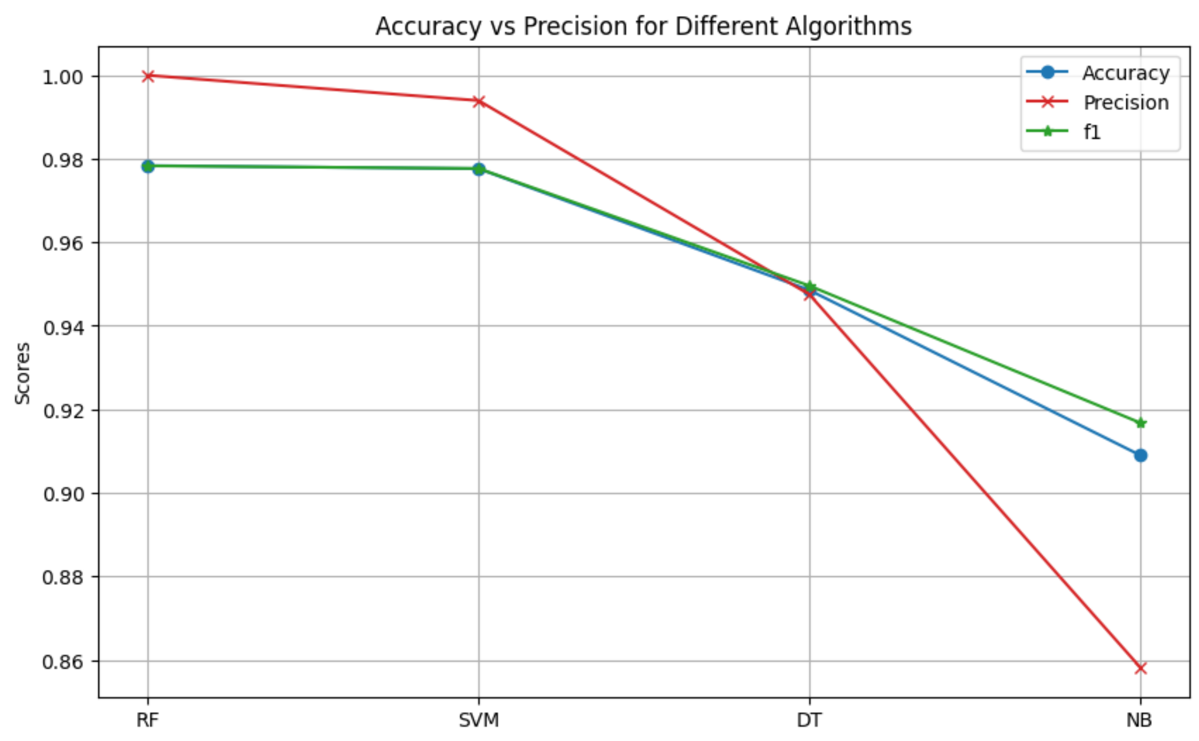


Figure 9 Comparison of four Models

In this illustration(Figure 9), random forest exhibits the highest performance across all metrics, indicating a strong ability to classify instances correctly with a good balance between precision and recall. Support vector machine Shows a slight decrease in performance metrics compared to RF but still maintains high scores, suggesting robust predictive capabilities. Decision tree reveals a more significant decline in all three metrics, which might indicate that the model is not capturing the complexities of the data as effectively as RF and SVM. Naive bayes demonstrates the lowest scores on all metrics, suggesting that its assumptions about feature independence may not hold in this dataset, leading to less effective classification.

Random Forest Classifier: With an accuracy of roughly 97.84%, the Random Forest Classifier performs exceptionally well, correctly classifying a sizable percentage of the email messages. The Precision score of 1.0000 indicates that the model is nearly always right when it classifies an email as spam. An overall assessment of the model's performance, taking into account both precision and recall, is provided by the F1 Score of 0.9783. It suggests that recall and precision are well-balanced.

Support Vector Machine: The accuracy of the SVM model is very high, about 97.76%, slightly lower than the accuracy of the Random Forest Classifier. The model's spam predictions are extremely dependable, as indicated by the Precision score of 0.9939.
A strong overall performance is suggested by the F1 Score of 0.9777, which is comparable to the Random Forest model.

Decision Tree Classifier: The Decision Tree Classifier achieves a relatively good accuracy of approximately 94.85%. The Precision score of 0.9475 indicates that the model's spam predictions are fairly accurate but not as precise as the Random Forest or SVM models.
The F1 Score of 0.9496 suggests a balanced performance in terms of precision and recall.

Multinomial Naive Bayes: At roughly 90.90%, the MNB model's accuracy is the lowest of the four. In comparison to the other models, the model's spam predictions are less dependable, as indicated by its Precision score of 0.8582.
A relatively good balance between precision and recall is indicated by the F1 Score of 0.9168, which is lower than that of the Random Forest and SVM models.

In terms of accuracy, precision, and F1 Score, the Random Forest Classifier and the Support Vector Machine (SVM) perform better overall. These models perform well on the provided dataset, which suggests they are the better options for classifying email spam in this project. In terms of accuracy and precision, the Multinomial Naive Bayes model trails behind the Decision Tree Classifier, which likewise performs fairly well.

K-fold cross-validation Method:

What is K-fold cross-validation: The method of K-fold cross-validation is used to assess predictive models. There are k folds, or subsets, within the dataset. A distinct fold is used as the validation set for each of the k training and evaluation cycles of the model. The generalisation performance of the model is estimated by averaging the performance metrics from each fold. This approach helps with model evaluation, selection, and hyperparameter tuning, offering a more accurate way to gauge a model's efficacy (Pandian, 2022).

Throughout the process, the test and training would be run exactly once in each set (fold). It aids in preventing overfitting. As is well known, a model performs most accurately when it is trained using all of the data in a single short run. In order to overcome this, k-fold cross-validation aids in the development of a generalised model.

Key Concepts:

K-Folds: For this, I've selected k=5, which divides the dataset into five folds, or subsets, of about equal size. The validation set will alternately be each fold, with the remaining folds being used for training.

Data Shuffling: To make sure the data is dispersed randomly among the folds, we can use data shuffling. This lessens the chance that the model's performance will be impacted by any inherent order or bias in the data.

Random Seed: To guarantee reproducibility, this random seed (random_state=42) has been set. The results are consistent between runs because the same data splitting happens when we perform the cross-validation multiple times.

Here, is the table(**Error! Reference source not found.**) of k-fold cross validation according to different models. Which is showing mean f1 score, and Standard Deviation of the models.

Model Name	Mean F1 Score	Standard Deviation
Random Forest Classifier	0.9735	0.0055
Support Vector Machine	0.9730	0.0076
Decision Tree Classifier	0.9487	0.004
Multinomial Naive Bayes	0.9144	0.0181

Table 3 Cross Validation of the Models

Here, some **comments** on different models based on the results,

Random Forest Classifier :

An ensemble learning technique called the Random Forest Classifier uses several decision trees combined to produce predictions. It is known for its proficiency with high-dimensional features and complicated datasets.

The Random Forest ensemble of decision trees appears to be able to accurately identify patterns in the data, as evidenced by the high mean F1 Score of 0.9735.

The Random Forest model's low standard deviation of 0.0055 suggests that it is tolerant to changes in the dataset and more consistent. This is because multiple trees' averaging effect reduces overfitting.

Support vector machine (SVM):

Strong classification performance is achieved by SVM, an algorithm that handles both linear and non-linear data. The hyperplane with the largest margin between classes is the one that is sought after.

With a mean F1 Score of 0.9730, the SVM demonstrated its efficacy in distinguishing between emails that were spam and those that weren't.

There may be some variability, as indicated by the slightly higher standard deviation of 0.0076, which could be brought on by the regularisation parameter or the kernel selection. Still, SVM delivers consistently good results.

Decision Tree Classifier:

Decision trees are interpretable models that divide data into subsets according to the most important characteristics.

The Decision Tree Classifier's ability to identify significant features in the dataset was demonstrated by its mean F1 Score of 0.9487.

Decision trees yield reliable and consistent results at various cross-validation folds, as evidenced by their low standard deviation of 0.0040.

Multinomial Naive Bayes (MNB):

One popular probabilistic model for text classification tasks is Multinomial Naive Bayes.

The MNB model's mean F1 Score was lower than the other models', at 0.9144, suggesting that it may not have been as successful in identifying subtle patterns in the data.

Because of its simplifying assumptions about feature independence, MNB's performance appears to vary more across folds, as indicated by the higher standard deviation of 0.0181.

Justification: Because of their resilience to changes in the dataset and their capacity to manage complex, non-linear relationships in the data, the Random Forest Classifier and Support Vector Machine (SVM) perform better than the other models.

Decision trees, while interpretable and effective, may not capture complex patterns as well as ensemble methods like Random Forest or the decision boundary optimization of SVM.

Although Multinomial Naive Bayes (MNB) is a straightforward and quick algorithm, it doesn't seem to be as good at capturing the subtleties of classifying spam emails in this instance, as evidenced by its lower F1 Scores and increased variability.

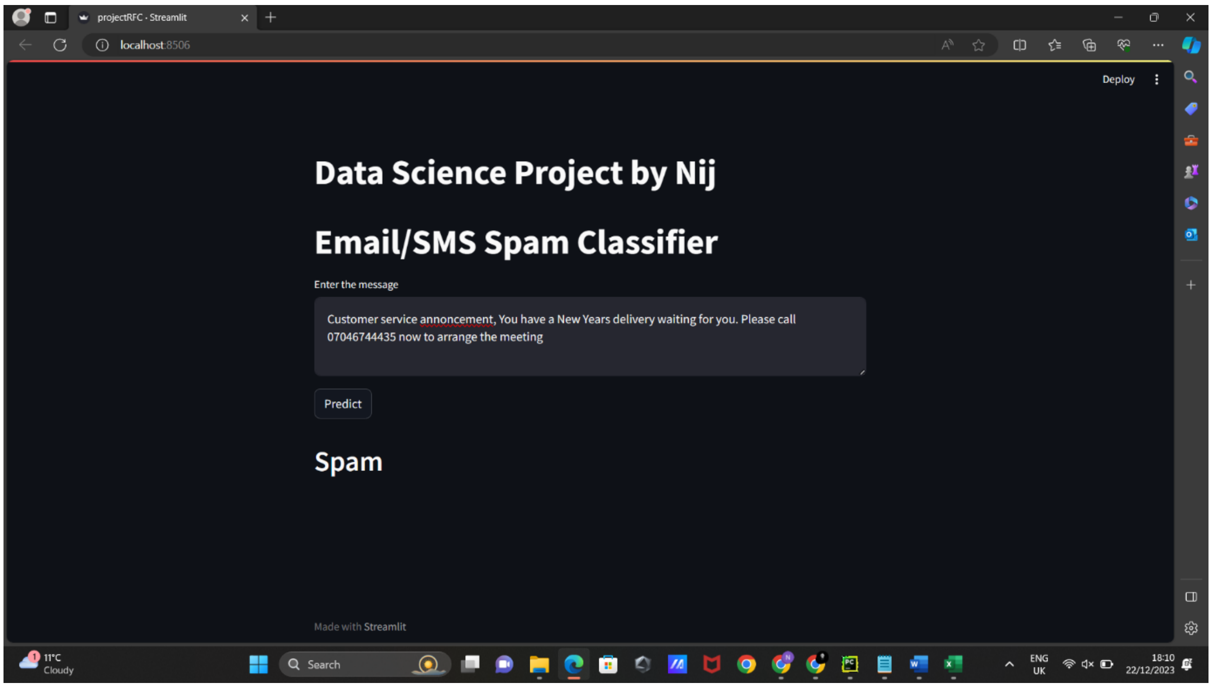
In summary, the choice of Random Forest Classifier or Support Vector Machine is supported by both their strong theoretical foundations and their high mean F1 Scores with relatively low standard deviations in the cross-validation results.

The best option for the final model in the email spam classification project is Random Forest because it performs better than all other models in terms of accuracy, precision, and F1 score. It provides a solid solution that consistently performs well and has strong predictive capabilities.

4.9 Deployment of Model

For the deployment process, I have used the ‘Streamlit’ platform to deploy my project in order to interact with the actual model of spam classification.

The core of the application lies in its backend, where a sophisticated text preprocessing function takes the user's input and prepares it for prediction. This function, ‘transform_text’, performs several steps, converting all text to lowercase, tokenizing the text into individual words, removing non-alphanumeric characters, filtering out stopwords and punctuation, and finally, stemming the words to their base or root form. These preprocessing steps are critical as they transform raw text into a cleaner, more analyzable format that the model can understand (Academy, 2023).



Data Science Project by Nij

Email/SMS Spam Classifier

Enter the message

Customer service annoncement, You have a New Years delivery waiting for you. Please call 07046744435 now to arrange the meeting

Predict

Spam

Figure 10 Interface of Streamlit App

After selecting the 'Predict' button as shown in the (Figure 10), a pre-loaded Tfidf (Term Frequency-Inverse Document Frequency) vectorizer is used to vectorize the preprocessed text. By converting the text into numerical data, this vectorizer preserves the relative weights of each word in relation to the dataset used for model training. Then, using ‘joblib’, an application that facilitates the loading of serialised Python objects, the vectorized text is fed into a pre-trained machine learning model. Which I have created from the for Random Forest classifier.

Next, based on the file names, the machine learning model possibly a Random Forest Classifier predicts if the message is likely spam. Through the web interface, the user is informed of the outcome of this prediction, if the model determines that the message is spam, the application displays "Spam" in a bold header; if not, it displays "Not Spam." Users can evaluate any message quickly thanks to this instant feedback loop, which could boost productivity and serve as a useful tool to block unsolicited or harmful communications.

This ‘Streamlit’ application, which bridges the gap between sophisticated machine learning algorithms and end-user utility, essentially embodies the practical implementation of a data science project. It demonstrates how machine learning can be incorporated into commonplace tools to increase accessibility and safety for all users of technology.

Chapter 5: Conclusion

5.1 Overview

My goal for this project was to create a reliable model for classifying spam emails. Preprocessing and data cleaning were crucial steps in ensuring data integrity as the journey started with importing and exploring the dataset. I then dig into exploratory data analysis (EDA) to learn more about the features of the dataset. The vectorization of text data and its division into training and testing sets came next. Developing and optimising four different classification models such as Multinomial Naive Bayes (MNB), Decision Tree Classifier (DTC), Random Forest Classifier (RFC), and Support Vector Machine (SVM) are the main focus of the project. Finally, the model's performance was assessed using a variety of metrics and further examination was conducted using k-fold cross-validation. At the end, deployment has been done from 'streamlit' platform.

5.2 Findings and Recommendations

The project's main conclusions emphasised how well the Random Forest Classifier (RFC) and Support Vector Machine (SVM) performed. In terms of accuracy, precision, and F1 score, these models performed better than both Multinomial Naive Bayes (MNB) and the Decision Tree Classifier (DTC). RFC produced a remarkable F1 score of 97.37%, 100.00% precision, and an impressive 97.39% accuracy. Simultaneously, SVM demonstrated impressive performance, with scores of 97.31% for accuracy, 99.36% for precision, and 97.34% for F1. Suggestions for additional improvement include ongoing experimentation with hyperparameters, investigating feature engineering to derive deeper insights from textual data, and possibly implementing sophisticated natural language processing (NLP) methods like word embeddings or trained language models.

5.3 Areas for Future Work

The project has revealed a number of fascinating directions for further research and development. First of all, there is potential to improve the dataset by adding a wider range of current email samples. Further exploration of sophisticated NLP methods, such as transformer-based models and word embeddings, may be able to improve classification accuracy even further. Practicality could be improved by integrating email platforms seamlessly and putting in place real-time email classification systems. Further areas for continuous improvement are the model's adaptation to changing spam email patterns and ongoing monitoring.

5.4 Personal Evaluation

Throughout this project, I ran into concerns with class imbalance and critical requirement for strong preprocessing methods. Nonetheless, the development and refinement of successful classification models were fruitful results. In the end, this project offered priceless insights into the categorization of email spam, highlighting the importance of model evaluation and optimisation. All things considered, the journey was a valuable learning opportunity that demonstrated the possibility of more developments in the field of email spam classification.

References

Academy, P. (2023). *Streamlit App Deployment: Step-by-Step Guide to Streamlit Cloud*. [online] Medium. Available at: https://pytechacademy.medium.com/streamlit-app-deployment-step-by-step-guide-to-streamlit-cloud-37c7eb2687d6#:~:text=Deploying%20your%20Streamlit%20app%20on [Accessed 22 Dec. 2023].

Bassiouni, M., Ali, M. and El-Dahshan, E.A. (2018). Ham and Spam E-Mails Classification Using Machine Learning Techniques. Journal of Applied Security Research, 13(3), pp.315–331. doi:https://doi.org/10.1080/19361610.2018.1463136.

chugh, aakarsha (2018). *ML | Label Encoding of datasets in Python*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/ml-label-encoding-of-datasets-in-python/.

Dada, E.G., Bassi, J.S., Chiroma, H., Abdulhamid, S.M., Adetunmbi, A.O. and Ajibuwa, O.E. (2019). Machine learning for email spam filtering: review, approaches and open research problems. Heliyon, [online] 5(6), p.e01802. doi:https://doi.org/10.1016/j.heliyon.2019.e01802.

Donges, N. (2021). *A Complete Guide to the Random Forest Algorithm*. [online] Built in. Available at: https://builtin.com/data-science/random-forest-algorithm.

GeeksforGeeks. (2023). *Machine Learning Model Evaluation*. [online] Available at: https://www.geeksforgeeks.org/machine-learning-model-evaluation/.

GÜBÜR, K.T. (2021). *NLTK Tokenize: How to Tokenize Words and Sentences with NLTK? - Holistic SEO*. [online] Available at: https://www.holisticseo.digital/python-seo/nltk/tokenization.

IBM (2023). *What is Naïve Bayes | IBM*. [online] www.ibm.com. Available at: https://www.ibm.com/topics/naive-bayes.

Jain, T., Garg, P., Chalil, N., Sinha, A., Verma, V.K. and Gupta, R. (2022). SMS Spam Classification Using Machine Learning Techniques. *2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. doi:https://doi.org/10.1109/confluence52989.2022.9734128.

Jain, T., Garg, P., Chalil, N., Sinha, A., Verma, V.K. and Gupta, R. (2022). SMS Spam Classification Using Machine Learning Techniques. 2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence). doi:https://doi.org/10.1109/confluence52989.2022.9734128.

Lai, C.-C. (2007). An empirical study of three machine learning methods for spam filtering. Knowledge-Based Systems, 20(3), pp.249–254. doi:https://doi.org/10.1016/j.knosys.2006.05.016.

Li, K., Li, K., Huang, H. and Tian, S. (2003). Active learning with simplified SVMs for spam categorization. doi:https://doi.org/10.1109/icmlc.2002.1167390.

Lin, Y.-L., Hsieh, J.-G., Wu, H.-K. and Jeng, J.-H. (2011). Three-parameter sequential minimal optimization for support vector machines. *Neurocomputing*, 74(17), pp.3467–3475. doi:https://doi.org/10.1016/j.neucom.2011.06.011.

Mujtaba, H. (2020). *An Introduction to Grid Search CV | What is Grid Search*. [online] GreatLearning. Available at: https://www.mygreatlearning.com/blog/gridsearchcv/.

Pandian, S. (2022). *K-Fold Cross Validation Technique and its Essentials*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2022/02/k-fold-cross-validation-technique-and-its-essentials/#h-what-is-k-fold-cross-validation.

Ramesh, S. (2023). *Demystifying the Random Forest*. [online] Medium. Available at: https://towardsdatascience.com/demystifying-the-random-forest-8a46f4fd416f [Accessed 21 Dec. 2023].

Ray, S. (2019). *6 Easy Steps to Learn Naive Bayes Algorithm (with code in Python)*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/.

SATPATHY, S. (2020). *SMOTE - A Common Technique to Overcome Class Imbalance Problem*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/.

Scikit learn (2018). *sklearn.feature_extraction.text.TfidfVectorizer — scikit-learn 0.20.3 documentation*. [online] Scikit-learn.org. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html.

Shiksha.com. (2020). Available at: https://www.shiksha.com/online-courses/articles/train-test-split/.

Sreenivasa, S. (2020). *Radial Basis Function (RBF) Kernel: The Go-To Kernel*. [online] Medium. Available at: https://towardsdatascience.com/radial-basis-function-rbf-kernel-the-go-to-kernel-acf0d22c798a.

Vedaldi, A. and Zisserman, A. (2012). Efficient Additive Kernels via Explicit Feature Maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3), pp.480–492. doi:https://doi.org/10.1109/tpami.2011.153.