SMS SPAM DETECTION USING NATURAL LANGUAGE PROCESSING AND ENSEMBLE LEARNING

A MAJOR PROJECT REPORT

Submitted by

ADITYA KUMAR

[Reg No: RA1911030010100]

ADARSH MAHAPATRA

[Reg No: RA1911030010071]

Under the guidance of

Dr. C. Fancy

Assistant Professor, Department of Networking and Communications

in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

with specialization in CYBER SECURITY

DEPARTMENT OF NETWORKING AND COMMUNICATIONS

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603 203

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that this B.Tech project titled "**SMS SPAM DETECTION USING NATURAL LANGUAGE PROCESSING AND ENSEMBLE LEARNING**" is the bonafide work of Mr. **Aditya Kumar [Reg No. RA1911030010100]** and Mr. **Adarsh Mahapatra [Reg No. RA1911030010071]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

15/5/23

Dr. C. FANCY
SUPERVISOR
Assistant Professor
Department of Networking and Communications

Dept. of Networking and Communications
Kattankulathur
603 203
SRM IST

16/5/23

Dr. ANNAPURANI. K
HEAD OF THE DEPARTMENT
Department of Networking and Communications

15/5/23

SIGNATURE OF INTERNAL EXAMINER

SIGNATURE OF EXTERNAL EXAMINER

Department of Networking and Communications

SRM Institute of Science and Technology

Own Work Declaration Form

Degree/ Course	: B.Tech in Computer Science and Engineering with Specialization in Cyber Security
Student Names	: Aditya Kumar, Adarsh Mahapatra
Registration Number	: RA1911030010100, RA1911030010071
Title of Work	: SMS Spam Detection using Natural Language Processing and Ensemble Learning

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is our own except where indicated, and that We have met the following conditions:

- Clearly referenced / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:

I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.

Aditya Kumar
RA1911030010100

Adarsh Mahapatra
RA1911030010071

If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

ACKNOWLEDGEMENT

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

We wish to thank **Dr Revathi Venkataraman**, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work. We are incredibly grateful to our Head of the Department, **Dr. Annapurani Panaiyappan K.** Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our program coordinators **Dr. M. Saravannan**, Associate Professor, Department of Networking and Communications, SRM Institute of Science and Technology, and Panel Head, **Dr. P Supraja**, Associate Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for their inputs during the project reviews and support. We register our immeasurable thanks to our Faculty Advisor, **Dr. CNS Vinoth Kumar**, Associate Professor, Networking and Communications, SRM Institute of Science and Technology, for leading and helping us to complete our course. Our inexpressible respect and thanks to my guide, **Dr. C Fancy**, Assistant Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under her mentorship.

We sincerely thank the Networking and Communications Department staff and students, SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank parents, family members, and friends for their unconditional love, constant support, and encouragement.

Aditya Kumar (RA1911030010100)

Adarsh Mahapatra (RA1911030010071)

ABSTRACT

Short Messaging Service, or SMS Spam is a common occurrence in everyday life and some people fall victim to phishing links and texts on a regular basis. Recent improvements in communication have made SMS into MMS (Multimedia Messaging Service) enabling users to send images and voice messages. Advancements in Artificial Intelligence and Cyber Security have made it possible to create AI models to detect if text is spam or is legitimate and in creating techniques such as Image Steganography in hiding data in an image. This research proposes an efficient tool by combining the power of Artificial Intelligence in detection spam messages by creating a custom model KNR from various algorithmic models using Ensemble Learning and using Image Steganography to hide text in an image while sending it the receiver. By choose Multinomial Naïve Bayes, K Nearest Neighbors and Random Forest algorithm were used to create our new model KNR which resulted in 97.96% accuracy and 94.57% precision in prediction if a text message is spam or legitimate. Further, the proposed work includes development of an Image Steganography tool by using 1 Bit Least Significant Bit method to integrate the ensemble model in the Steganography tool. When the sender wants to send confidential data to receiver, they will use the tool and hide their message in an image and sent this image to the receiver. The receiver, upon receiving the image will decrypt the image. This is where the detector model will parse through the message and predict wether the message is legitimate or spam. This process provides a layer of security as if an attacker were to know of this technique, even if they send spam text trying to phish, the model will detect the spam message and notify the receiver.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

ABBREVATION	EXPANSION
SMS	Short Messaging Service
AI	Artificial Intelligence
SVM	Support Vector Machine
NB	Naïve Bayes model
GNB	Gaussian Naïve Bayes
BNB	Bernoulli Naïve Bayes
MNB	Multinomial Naïve Bayes
KN	K Nearest Neighbors
DTC	Decision Trees Classifier
LR	Logistic Regression
RF	Random Forest
AdaBoost	AdaBoost Classifier
BgC	Bagging Classifier
ETC	Extra Trees Classifier
GBT	Gradient Boosting Classifier
XGB	Extreme Gradient Boosting Classifier
LSB	Least Significant Bit
TFIDF	Term Frequency-Inverse Document Frequency

CHAPTER 1

INTRODUCTION

This chapter introduces the background of SMS and how Artificial Intelligence is used to create tools that can detect if a text message is spam or legitimate. The development and improvement of new techniques such as feature extraction have made it possible to create lightweight modules that can be easily implemented with SMS applications with advanced models trained by using algorithms such as Naïve Bayes, Support Vector Machines, and so on.

1.1 OVERVIEW OF SMS

SMS messaging is a popular means of communication, particularly due to its widespread availability and ease of use. It is commonly used for personal conversations, such as keeping in touch with family and friends, as well as for professional purposes, such as sending and receiving work-related messages [1]. Additionally, businesses and organizations use SMS messaging for notifications, alerts, and reminders, such as appointment reminders, delivery notifications, and alerts for flight delays or cancellations.

Despite the growth of other messaging platforms like WhatsApp, Facebook Messenger, and WeChat, SMS messaging continues to be a widely used form of communication. According to a report by Mobilesquared, the number of SMS messages sent globally increased by 16.6% in 2020, and this growth is expected to continue in the coming years.

SMS messaging is often used as a means of two-factor authentication (2FA), where a code is sent to a user's phone to verify their identity. This can help to improve security, as it adds an extra layer of protection to accounts. However, SMS-based 2FA has been criticized for being less secure than other methods, such as using a dedicated authentication app, due to the risk of SIM-swapping attacks [2-4] , where a hacker takes control of a victim's phone number and intercepts SMS messages containing 2FA codes.

SMS is still being used and in an even bigger scale nowadays, it was used for sending simple and short messages and was charged by network couriers but now has evolved to MMS or Multimedia Messaging Service, enabling users to send images and voice messages along with text. MMS will be referred as SMS as it as a next evolution and for simplicity. Even though there are many private messaging services such as WhatsApp and Line, but they are used for sending text and voice messages and images to individuals and ground saved in contacts and are used for casual conversations. Whereas SMS is used for a variety of reasons, such as getting notification from banks on transactions and alerts, notifications from government for environmental warnings such as heavy rainfall and lightning (in some states), for One Time Passwords on payments and log-ins. They are also sent by network carriers in form of short code notifying about your SIM card balance and when you have exhausted your daily mobile data. These were just some of the use cases where SMS is an important mode of communication.

SMS can be armed by hackers to perform phishing scams, which are scams where a cybercriminal will send a text or a URL linked to fake and malicious websites in order to appear as a legitimate organization to trick people in giving usernames, passwords, bank account details and personal information [5]. They look like they are coming from a trusted source such as banks, online stores and promotions for social media services such as Facebook and LinkedIn. They are presented as urgent to the target such as stating that their bank account has been compromised and that they are from the bank customer support team and would like to help the victim. The familiarity of the message and lack of awareness on the victim's part can easily make them fall for these attacks.

Some organizations such as banks and websites might only send important information and provide services such as 2FA but it can be easily compromised due to vulnerabilities such as SS7 or Signaling System 7 protocols which is widely used in telecommunications to exchange information to route calls and texts [6]. These were designed in the 1980s and are still being used today but they are very vulnerable and a hacker can easily hack them to send messages even from one's own contacts and phish them for information, the encryptions placed by organizations to prevent these are easily broken and intercepted.

By using Artificial Intelligence, it is possible to detect if a message is spam text sent by a hacker as the AI models are trained on large datasets and trained to identify patterns and characteristics associated with spam. [7] The models work by preprocessing the text data

and finding features such as length of the message, occurrence of common spam words and type of content. Even though many models have been trained and deployed in some small SMS applications such as Signal, they are easily being circumvented by hackers as they come up with new techniques such as MITM attacks.

This research proposes to build a software that is divided into two parts, Spam Detection model using AI and building an Image Steganography tool to hide text data in images and send them instead of text so that it provides additional layer of security along with the encryption provided by the SMS application while in transit.

1.2 WEAPONIZATION OF SMS

Spam has been a problem since the early days of the internet, with unsolicited messages being sent via email and other messaging platforms. Over time, spammers have become increasingly sophisticated in their methods, using techniques such as spoofing and social engineering to trick recipients into opening and responding to their messages. Some of the attacks that use SMS as a means are:

Smishing: This is a type of phishing attack that uses SMS messages instead of email. The attacker sends a text message to the victim, usually with a sense of urgency or alarm, and asks them to click on a link or provide sensitive information such as login credentials or credit card details [8]. The link usually takes the victim to a fake website that looks legitimate, but is designed to steal their information.

Spoofing: SMS spoofing involves disguising the sender of an SMS message, so that it appears to come from a legitimate source such as a bank or government agency. The attacker may use a technique called 'number spoofing' to make it look like the message is coming from a trusted source, when in fact it is not.

Malware: Attackers can also use SMS messages to deliver malware to a victim's device. The message may contain a link to a malicious website or app, or may include an attachment that, when opened, infects the device with malware [9].

DoS attacks: Attackers can use SMS messages to launch denial of service (DoS) attacks against a target. This involves flooding the target's phone with a large number of messages, overwhelming the device and making it unable to function properly.

SIM swapping: SIM swapping involves the attacker convincing a victim's mobile phone carrier to transfer their phone number to a new SIM card that the attacker controls [10]. Once the attacker has control of the victim's phone number, they can use it to access the victim's accounts that require two-factor authentication via SMS.

1.3 ROLE OF AI IN SMS SPAM DETECTION

AI-powered systems use a combination of techniques such as machine learning algorithms, natural language processing (NLP), and behavioral analysis. Machine learning algorithms are trained on large datasets of labeled spam and non-spam messages to identify patterns and features that distinguish spam from legitimate messages [11]. This allows the system to automatically detect and block spam messages based on their content.

NLP techniques are used to analyze the content of messages, looking for common patterns or language that is often used in spam messages. For example, many spam messages use overly aggressive or urgent language to try to convince the recipient to take a particular action, such as buying a product or visiting a website. By analyzing the language used in messages, AI-powered systems can identify these common patterns and accurately classify messages as spam.

Behavioral analysis is another important technique used in SMS spam detection. This involves looking at factors such as message frequency, sender reputation, and other characteristics that may indicate spam. For example, if a sender is consistently sending large volumes of messages to multiple recipients, this may be an indication that the messages are spam. Similarly, if a sender has a history of sending messages that have been flagged as spam, this may also be an indication that future messages from that sender should be blocked.

Real-time data analysis is also an important component of SMS spam detection. This involves continuously monitoring incoming messages and identifying emerging spam trends. For example, if a large number of spam messages start using a particular keyword or phrase, this may indicate a new spam campaign that needs to be blocked.

1.4 EVOLUTION OF SPAM DETECTION METHODS

The evolution of spam detection methods has been driven by the increasing sophistication of spammers and the need for more effective and efficient ways of detecting and blocking spam messages [12]. Initially, spam detection relied on simple rule-based methods that looked for specific keywords or phrases in messages that were commonly associated with spam. However, as spammers became more sophisticated, these methods proved to be insufficient in detecting and blocking spam.

Rule-based systems: Rule-based systems were one of the earliest methods used for spam detection. These systems use predefined rules to identify spam messages based on certain criteria such as the presence of specific keywords or phrases. Rule-based systems were effective in the early days of spam when spam messages were more predictable and uniform in their structure and content.

Content-based filtering: Content-based filtering is a technique that analyzes the content of messages to identify spam. This method uses statistical analysis and machine learning algorithms to analyze the text, sender information, and other attributes of the message to identify patterns and characteristics of spam messages. Content-based filtering is more effective than rule-based systems as it can adapt to new types of spam messages.

Bayesian filtering: Bayesian filtering is a type of content-based filtering that uses statistical algorithms to calculate the probability that a message is spam. Bayesian filtering uses a probability model based on the characteristics of spam and non-spam messages to classify messages as either spam or non-spam. Bayesian filtering is effective in detecting new types of spam messages and can adapt to changes in the characteristics of spam messages.

Reputation-based filtering: Reputation-based filtering is a technique that uses the reputation of the sender to determine if a message is likely to be spam. Reputation-based filtering uses information about the sender's IP address, domain, and other factors to determine if a message is likely to be spam. This technique is effective in identifying messages sent from known spam sources and can be used in combination with other spam detection methods.

1.5 CHALLENGES IN SPAM DETECTION

SMS spam detection faces several challenges that can impact the accuracy and effectiveness of the detection model. These challenges include the diversity of messages, limited availability of data, imbalanced datasets, the dynamic nature of SMS spam, and privacy concerns.

SMS spam can take various forms and can contain different languages, formats, and content, making it challenging to develop a one-size-fits-all spam detection model. Additionally, limited availability of SMS spam datasets and the heavily imbalanced datasets can affect the accuracy and robustness of the detection model.

The dynamic nature of SMS spam and the ever-evolving techniques used by spammers, such as URL shorteners, image-based spam, and spoofing, can make it challenging for detection models to keep up. Furthermore, SMS spam detection models may require access to user data, including message content and contact lists, raising privacy concerns for users.

Addressing these challenges requires ongoing research and innovation to develop effective SMS spam detection models that accurately identify and filter out unwanted messages while ensuring that legitimate messages are not filtered out. The development and refinement of SMS spam detection models will continue to be an important area of research as the use of SMS continues to grow.

Feature engineering: Feature engineering is the process of selecting and extracting relevant features from the SMS message that can be used to distinguish between spam and legitimate messages. However, identifying the most informative features can be

challenging, as the relevance of features can vary depending on the context and content of the message.

Overfitting: Overfitting occurs when the model becomes too complex and begins to fit the noise in the training data, rather than the underlying pattern. This can lead to poor performance on new data and reduced generalizability of the model [13].

Limited labeled data: Supervised machine learning models require labeled data to train the model. However, labeled SMS spam datasets can be limited in size, quality, and diversity, which can impact the accuracy and robustness of the detection model.

Concept drift: Concept drift occurs when the distribution of the input data changes over time, making the model less effective in detecting new and emerging spam messages. This can be challenging to address, as SMS spam techniques are constantly evolving.

Model interpretability: While machine learning models can achieve high accuracy in detecting spam messages, they can be difficult to interpret. This can make it challenging to understand how the model is making decisions and to identify and correct any errors or biases.

Overall, addressing these challenges is crucial in developing effective SMS spam detection models that can accurately identify and filter out unwanted messages while ensuring that legitimate messages are not filtered out. This requires a combination of machine learning techniques, domain expertise, and ongoing evaluation and refinement of the model.

1.6 NEED FOR SMS SPAM DETECTION

The need for SMS spam detection has become increasingly important in today's digital age where mobile devices are ubiquitous and SMS messaging is a popular communication tool. SMS spam can be highly disruptive, annoying, and in some cases, can even be used to perpetrate fraud and other types of cybercrime.

One of the primary reasons for the need for SMS spam detection is the sheer volume of spam messages being sent. As the use of mobile devices has grown, so has the volume of spam messages. These messages can flood the user's inbox, making it difficult to distinguish legitimate messages from spam. This can lead to missed important messages and a poor user experience.

Another reason for the need for SMS spam detection is the potential harm that can result from malicious SMS messages. Some SMS spam messages may contain links to phishing websites or malware-infected downloads. Clicking on these links can compromise the user's device or personal information, leading to financial loss or identity theft.

Moreover, SMS spam messages can also contain fraudulent offers or scams, which may result in financial loss for the user. For example, a spam message may offer a prize or promotion in exchange for the user's personal information or payment. Falling for these scams can result in significant financial loss or harm to the user.

Finally, the need for SMS spam detection is also driven by regulatory requirements. Many countries have regulations in place that require mobile service providers to implement spam filtering and detection measures to protect consumers from unwanted SMS messages.

1.7 SIGNIFICANCE OF CYBER SECURITY IN SPAM DETECTION

Cyber Security techniques such as Encryption and Authentication form the basis of network security. The protocols used decide the robustness of the security provided to the SMS application.

While encryption does not directly detect spam messages, it can help to reduce the risk of spam messages being sent or received and can make it more difficult for spammers to carry out their attacks. Encryption can be used to protect confidential information that may be targeted by spammers, such as email addresses, phone numbers, or other sensitive data. By encrypting this information, spammers are less likely to be able to obtain it and use it to send spam messages, secure communication channels between users, making it more difficult for spammers to intercept messages and inject spam into the conversation, prevent spammers from tampering with messages by ensuring that messages cannot be

modified or altered in transit. This can help to reduce the risk of spam messages being injected into conversations or messages being modified to include spam links or other malicious content. It can also prevent spoofing attacks, which are commonly used by spammers to impersonate legitimate senders. By encrypting messages, spammers are less likely to be able to forge the sender's identity and send spam messages.

Sender authentication technologies, such as SPF, DKIM, and DMARC, are designed to verify that messages are coming from authorized senders. SPF allows domain owners to specify which mail servers are authorized to send messages on behalf of their domain, and checks that incoming messages are being sent from those authorized servers. DKIM allows senders to sign messages with a digital signature, which can be verified by the recipient's email server to ensure that the message has not been tampered with in transit. DMARC builds on these technologies by providing a policy framework for domain owners to specify how they want their messages to be handled if they fail SPF or DKIM checks. By implementing these authentication technologies, messaging systems can help to prevent spoofing attacks and reduce the overall volume of spam messages that are sent and received. They can also help to reduce the risk of phishing attacks by ensuring that messages are being sent to authorized recipients, and by verifying that messages are coming from legitimate sources.

1.8 NEED FOR ENSEMBLE LEARNING

Ensemble learning is a powerful technique for improving the performance of spam detection systems. Ensemble learning involves combining multiple classifiers, each of which is designed to identify different types of spam messages or spam characteristics, into a single system.

The basic idea behind ensemble learning is that by combining the predictions of multiple classifiers, the overall accuracy of the system can be improved. Ensemble learning algorithms can take many forms, such as bagging, boosting, or stacking.

Ensemble learning has been shown to be highly effective for spam detection, with many researchers reporting significant improvements in performance when using ensemble techniques. By combining multiple classifiers, ensemble learning can improve the overall

accuracy of the system, reduce false positives and false negatives, and provide a more robust defense against spam attacks.

Moreover, as spam detection systems become more complex and more difficult to train, ensemble learning can help to reduce the risk of overfitting and increase the generalization ability of the model. By combining multiple classifiers that are trained on different subsets of the data or using different algorithms, ensemble learning can help to overcome the limitations of individual classifiers and create a more accurate and robust spam detection system.

In conclusion, ensemble learning is a powerful technique that can be used to improve the performance of spam detection systems. By combining multiple classifiers, ensemble learning can increase the accuracy of the system, reduce false positives and false negatives, and provide a more robust defense against spam attacks.

1.9 MECHANISMS FOR THE ENSEMBLE MODEL

To develop a custom ensemble model, algorithms need to be chosen to build their models first and the models with high accuracy and precision compared to the other models were chosen for the custom model, along with their accuracy and precision, the diversity of the models, their complexity, the time and resource constraints and the data used for the project.

Naive Bayes is a family of probabilistic algorithms used for classification tasks in machine learning. It is based on Bayes' theorem to find the probability of an event occurring based on prior knowledge of conditions related to the event. I Bayes classifiers assume that the features are independent of each other given the class label, hence the name "naive".

GNB assumes that the features are continuous and follow a Gaussian (normal) distribution. It calculates the mean and standard deviation of each feature for each class and uses these values to calculate the probability of a given sample belonging to each class. MNB assumes that the features are discrete and countable, such as word occurrences in a text document.

Support Vector Machines find a hyperplane that separates different classes of data points in the feature space. They can be used for SMS spam detection by learning a boundary between the two classes, spam and non-spam. SVMs try to find the best separating hyperplane that maximizes the margin. In this case, the features extracted from SMS messages are used as input to the SVM algorithm, and the labels are assigned as either spam or non-spam. The SVM algorithm is trained on a set of labeled SMS messages, and then it is used to classify new, unlabeled messages as either spam or non-spam based on the learned boundary.

K Nearest Neighbor ‘s idea is that similar items are close to each other in a feature space. In other words, if a new SMS message is received, the KNN algorithm will look at the k closest SMS messages in the training data and classify the new message based on the majority class of those k neighbors. One advantage of using KNN for SMS spam detection is that it is a simple and intuitive algorithm that is easy to implement. Additionally, KNN can be effective when the data has a clear separation between the different classes and when the number of features is small. However, it can be computationally expensive for large datasets.

Decision Tree Classifier algorithm builds a decision tree based on the given training data, and the tree is used to classify new instances based on the features of the data In the context of SMS spam detection, DTC can be trained on a labeled dataset of SMS messages, with the spam/ham label being the target variable. The features can be extracted from the messages using techniques such as bag-of-words or TF-IDF, and the decision tree can be built based on the frequency and context of the words. The resulting model can then be used to classify new SMS messages as spam or ham based on their feature values.

Logistic Regression is used for predicting a continuous outcome variable based on one or more predictor variables. It works by finding the best linear relationship between the input variables and the output variable by minimizing the difference between predicted and actual values. In other words, it tries to find the straight line that best fits the given data points. In the case of spam detection, LR can be trained on a labeled dataset consisting of SMS messages labeled as either spam or not spam. LR learns to model the relationship between the input features of the SMS messages, such as the presence or

absence of certain words, and the output label of spam or not spam. Once the LR model is trained on the dataset, it can be used to predict the label of new SMS messages as either spam or not spam. LR produces a probability score between 0 and 1 for each new SMS message, which can be the threshold to make a final binary classification decision.

Random Forest is a supervised machine learning algorithm used for classification and regression tasks. It is based on decision trees, but unlike decision trees that can easily overfit to the training data, RFs use a technique called bootstrap aggregating or bagging to reduce overfitting. For SMS spam detection, RF can be a good choice because it can handle large datasets with high dimensionality (i.e., many features), and it is not sensitive to irrelevant or redundant features.

Adaboost works by iteratively training weak learners on different weighted versions of the dataset. After each iteration, the weights are changed to make weak learners as strong. In the context of SMS spam detection, Adaboost can be used as a classifier to accurately distinguish between spam and non-spam messages. The algorithm is particularly useful when there is a class imbalance in the data, meaning there are significantly more non-spam messages than spam messages. Adaboost can help to address this imbalance by focusing more on the minority class (spam) during the training process. It has also been shown to have a high classification accuracy and low false positive rate in SMS spam detection tasks.

Bagging classifier is used to reduce the variance in the predictions of a single model by averaging together the predictions of multiple models that have been trained on different subsets of the training data. This can improve the overall accuracy and stability of the model, especially when dealing with noisy or complex datasets.

This project focused on using Bagging classifier with the default classifiers defined in sklearn, i.e. Decision Trees. Bagging can be used for SMS spam detection because it can handle high-dimensional datasets, which is often the case with text data. Bagging can also handle noisy data and can reduce the overfitting that may occur when using a single classifier. Moreover, bagging can be easily parallelized, which makes it a scalable solution for large datasets.

Extra Trees Classifier (ETC) is a type of ensemble learning method that combines multiple decision trees to improve classification performance. ETC works similarly to Random Forest but differs in the way the trees are constructed. One reason to use ETC for SMS spam detection is that it is an effective method for handling high-dimensional data. In SMS spam detection, the feature space can be quite large, and ETC can help in identifying the most informative features for distinguishing between spam and non-spam messages. Additionally, ETC has a faster training time compared to other ensemble methods like Random Forest. This can be an advantage when working with large datasets, as it can help in reducing the training time.

Gradient Boosting classifier builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. In each stage, it fits a new weak learner to the residual errors that were not captured by the previous weak learners. The key difference between GBC and other boosting methods is that GBC doesn't place any restrictions on the type of weak learner used. In other words, it can be used with a variety of different weak learners such as decision trees, SVMs, and neural networks.

In SMS spam detection, GBC can be used to identify the important features that distinguish spam messages from legitimate messages. It can also handle imbalanced datasets and noisy data, which is common in SMS spam detection. Overall, GBC can provide high accuracy and generalization performance for SMS spam detection.

Extreme Gradient Boosting is known for its high performance and accuracy in various machine learning competitions.

In the context of SMS spam detection, XGBoost can be used as a classifier to distinguish between spam and non-spam messages. It can be trained on a dataset of labeled messages, where the features of the messages are the input variables and the labels indicate whether the message is spam or not. Once trained, XGBoost can be used to predict the label of new incoming messages and classify them as spam or not. Compared to other classification algorithms, XGBoost can handle missing values and outliers, and it has built-in regularization to prevent overfitting. It also has a wide range of hyperparameters that can be tuned to optimize its performance on a particular task. Therefore, XGBoost can be a useful tool for SMS spam detection when high accuracy is required.

1.10 IMAGE STEGANOGRAPHY AS A SECURITY FEATURE

Steganography is the art of hiding information within other information, such as an image, audio file, or video, in a way that makes it difficult to detect or intercept by unauthorized parties. Steganography is often used as a security feature in communication systems to protect sensitive information from interception or detection [14].

Steganography has been used throughout history as a means of hiding secret messages and information in plain sight. The practice of steganography can be traced back to ancient times, with early examples of steganography found in the use of invisible ink, coded messages, and hidden writing.

One of the earliest recorded examples of steganography dates back to ancient Greece, where the historian Herodotus described how the ancient Greeks used a wax tablet to conceal a message by covering it with a layer of wax and then writing the secret message on the surface of the wax. The message could then be revealed by melting away the wax.

During the Middle Ages, steganography became more sophisticated with the development of cipher systems, such as the Vigenère cipher, which involved using a key to encode and decode messages. In the Renaissance era, steganography was used by artists and poets to embed secret messages and symbols within their works of art and literature.

The use of steganography continued to evolve over time, with new techniques and technologies emerging. During World War II, steganography was used by spies and intelligence agencies to conceal information within photographs and other documents. In recent years, steganography has been used in digital communication systems to hide messages within digital images, audio files, and video.

Today, steganography is an important technique in cybersecurity, used to protect sensitive information from interception or detection. While steganography has a long history and has been used in many different contexts, its basic principles remain the same: the art of hiding secret messages and information in plain sight, often using clever encoding or masking techniques to make the hidden message difficult to detect.

1.11 LEAST SIGNIFICANT BIT TECHNIQUE

LSB (Least Significant Bit) is a widely used technique in image steganography, where the secret information is hidden by altering the least significant bits of the pixel values of an image. Since the least significant bits determine the pixel color's intensity, modifying them will result in minimal changes to the image's visual appearance, making the hidden information undetectable to the human eye.

The LSB technique is popular because it is relatively simple to implement and is compatible with a wide range of image file formats. Additionally, it allows for a relatively high capacity of hidden information, as multiple bits can be modified for each pixel without significantly impacting the image quality.

The capacity of hiding information in LSB is much higher than the other two as LSB can hide information in each bit of each pixel whereas spread spectrum and phase coding spread the hidden message across multiple bands resulting in lower capacity.

LSB is more secure as it hides by only shifting 1 bit (or more depending on configuration) which is not perceptible to the human eye whereas the other two may introduce noticeable change in the cover file.

It is more reliable as it is more robust than others because it is less affected by noise and errors in the file whereas the other two can cause significant distortion in the original message, and because the tool will take image of any dimension and not have a limit, this makes it the better choice.

Figure 1.1: Example of LSB

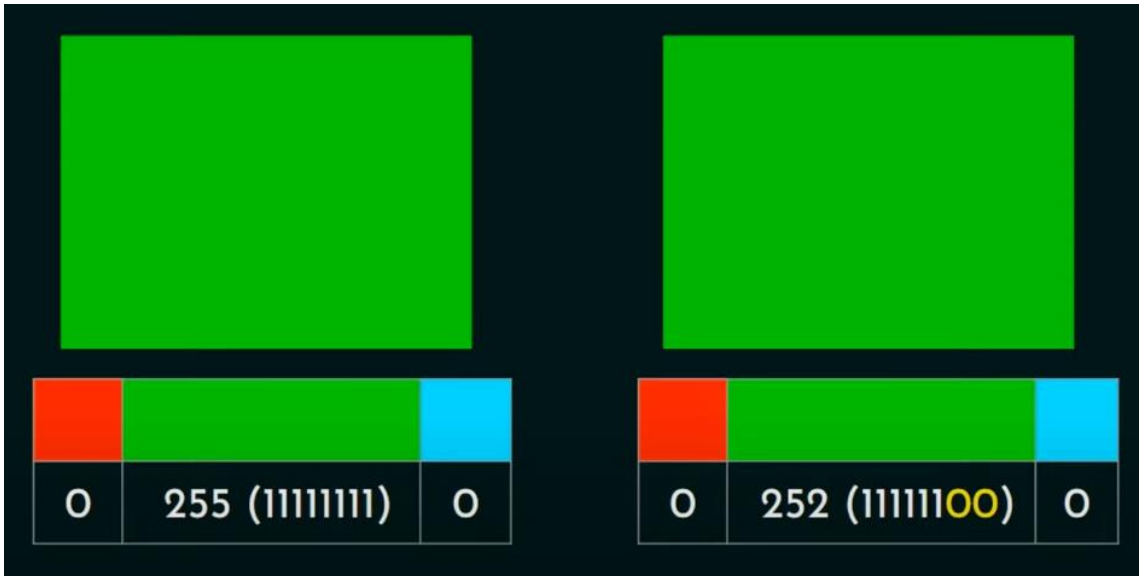


Figure 1.2: Change in colour after LSB

1.12 MOTIVATION

The motivation behind building an SMS spam detection model is to improve the accuracy of identifying and filtering unwanted spam messages, which can be a significant source of annoyance and security risks for mobile phone users. Spam messages can contain malicious links or attachments that can harm the users' devices, steal personal information, or even cause financial loss. By building a robust spam detection model, we

can help users filter out unwanted messages, protect their devices, and improve their overall user experience.

Additionally, spam detection models can also help businesses in identifying and filtering out unsolicited marketing messages, which can harm their reputation and reduce the effectiveness of their marketing campaigns. By using a spam detection model, businesses can improve their targeting and ensure that their messages reach the intended audience, resulting in higher engagement and conversion rates.

Overall, the development of an accurate and reliable SMS spam detection model has significant benefits for both users and businesses, making it an essential area of research in the field of machine learning and natural language processing.

1.13 SCOPE OF PROJECT

The scope of this project is to develop a SMS spam detection model using machine learning algorithms. The model will be trained on a labeled dataset of SMS messages, and will be able to classify incoming messages as either spam or not spam. The project aims to provide a practical solution for filtering unwanted spam messages, and can be potentially useful for various applications.

In addition to the SMS spam detection model, the project aims to integrate this model into an image steganography tool. The tool will allow users to hide text messages within images, and the SMS spam detection model will be used to ensure that any incoming text messages are not spam messages. This will provide an additional layer of security and privacy to the users of the steganography tool.

The LSB technique used in image steganography replaces the least significant bit of the pixel values in the image with the bits of the secret message, thus not changing the image's overall appearance. The project aims to use a 1-bit LSB technique, which replaces only one bit of each pixel with a secret message bit, ensuring that the changes made to the image are imperceptible to the human eye.

The integration of the SMS spam detection model with the image steganography tool will enable users to classify the SMS messages into spam or ham using the machine learning

model and then hide the messages inside an image using the image steganography tool. This will help users to securely transmit sensitive information through SMS messages without being detected by spam filters.

1.14 OBJECTIVE OF THE PROJECT

The primary objective of this project is to develop a machine learning model for SMS spam detection. The model should be able to classify incoming SMS messages as either spam or ham (legitimate) with high accuracy. Additionally, the project aims to implement the SMS spam detection model in an image steganography tool using 1-bit least significant bit (LSB) encoding.

The secondary objective of the project is to explore various machine learning algorithms and ensemble techniques to identify the best-performing model for SMS spam detection. The project will involve preprocessing the SMS dataset, including text cleaning, tokenization, and feature engineering. The selected machine learning algorithms will be trained, optimized, and evaluated using appropriate metrics.

Finally, the project aims to integrate the SMS spam detection model into an image steganography tool using Python. The tool will be able to encode secret messages within image files using 1-bit LSB encoding and decode them using the trained SMS spam detection model. The project's overall objective is to develop an effective and practical tool for secure communication that can be used in various settings.

CHAPTER 2

LITERATURE SURVEY

This survey describes the techniques used for building SMS Spam Detection models overs the years, listing out the historical changes in development and emergence of new techniques in the field.

2.1 RULE BASED FILTERING

Rule Based filtering was the earliest phase of spam detection. The rules were typically based on specific keywords, phrases, or patterns that were commonly found in spam messages. This approach was effective in the early days of spam, when most spam messages followed a predictable pattern and used similar language and formatting. However, it was limited by its inability to keep up with the rapidly changing nature of spam messages. Spammers quickly learned to adapt to rule-based filters by changing the wording, formatting, and delivery method of their messages, making it difficult for filters to keep up.

Jain et al [15] created 9 rules such as “If URL present in the image, THEN it is probably a smishing message”, rules such as these were then judged on their spam detecting performance and it was found that messages containing suspicious keywords like “free” and ”win” had 87% of being spam. Whereas Xia et al. [16] also added hashes of words in the ruleset and made rules to detect evasion methods such as typing “cash” as “c@sh” and “ca$h”. Foozy et al [17] used rules to focus more on advertising/promoting spam messages by developing rules such as detecting the word “winner” and look for keywords that might indicate advertisement and subscription messages.

Another limitation of rule-based filtering was its tendency to generate false positives, where legitimate messages were incorrectly identified as spam and filtered out. This was especially true for messages that contained words or phrases that were also commonly found in spam messages. Despite its limitations, rule-based filtering laid the foundation

for more sophisticated spam detection methods that followed, such as statistical filtering, heuristic filtering, and hybrid filtering. These newer approaches rely on machine learning algorithms to analyze the content, behavior, and characteristics of messages, allowing them to adapt to new forms of spam and learn from previously classified messages.

But this technique had huge drawbacks as the rules could be spoofed easily and it would take a human to keep updating the ruleset as the usage of words and slangs change with time. And also having too many datasets could increase the time for detection of spam messages.

2.2 CONTEXT BASED FILTERING

Context-based filtering takes into account the context in which a message is received and analyzed to determine its likelihood of being spam. This approach considers a range of factors beyond just the content of the message itself, such as the sender's reputation, the content of previous messages from the same sender, and the behavior of the recipient in responding to similar messages. For example, if a user regularly receives messages from a particular sender that they have previously marked as spam, a context-based filter might be programmed to give a higher spam score to any future messages from that same sender. Similarly, a filter might analyze the behavior of the recipient in responding to similar messages in the past, and use this information to determine the likelihood that a new message is spam.

Karami et al. [18] used LIMC (Linguistic Inquiry and Word Count) features for training their detection model, the features were categorized such as Capital words, Spam words, Unique words, URLs, SMS Frequency, rate of Spam words to Unique words, and similar rate of one feature to another feature with score of Punctuations, Pronouns, Exclamation marks and more. They observed that Random Forest model with LIWC gave the highest accuracy in their testing of 98.47%. Hidalgo et al. [19] would incorporate Bayesian Filters in their model, namely SpamAssassin, which was a very powerful tool at the time for labelling spam messages correctly, they trained NB and SVM models and came to the conclusion that NB was the most reliable algorithm to train a model for detecting spam messages. Roy et al. [20] used CNN and RNN models as their detector model and

extracted features and ran NB, RF, LR classifiers on them and received 99.44% accuracy with CNN model but this was highly computational and would take a lot of time.

Context-based filtering proved to be more effective than rule-based filtering, which relies on a set of predefined rules to identify and filter out unwanted messages. Rule-based filters are limited by their inability to adapt to new forms of spam, while context-based filters are able to learn from user behavior and adapt to new types of spam over time.

2.3 FEATURE EXTRACTION

Taking a page from Context based filtering, this technique would train the models based on the features of messages from the dataset such as length of words, the occurrence of specific words, and so on.

Sharaff et al. [21] proposed using Chi-square and Information gain to select attributes for training the model. Chi squared is test that is commonly used to determine whether an observed distribution of data significantly deviates from an expected distribution. The test is based on the difference between the observed and expected frequencies of a particular event, divided by the expected frequency. The resulting statistic follows a chi-squared distribution, which can be used to determine the probability that the observed deviation is due to chance. This involves calculating the chi-squared statistic for each feature, which measures the extent to which the frequency of that feature differs between spam and legitimate messages. Features with higher chi-squared values are considered more relevant for spam detection, as they are more likely to be associated with spam messages. Information gain is based on the idea of entropy, which is a measure of the uncertainty or randomness of a system. The information gain of a particular feature is calculated by comparing the entropy of the classification problem before and after splitting the dataset on that feature. If the feature is able to split the dataset into two or more groups that are more homogeneous with respect to the target variable (i.e. spam or legitimate), it is considered to have high information gain and is more likely to be relevant for spam detection. The models only gave a slight increase in accuracy but were a proof of concept that feature extraction can be used to increase accuracy of the models. Uysal et al [22] used Gini Index instead of Information Gain and found that capitalization of words

affected the model but it would be better to lowercase all text while preprocessing. Zainal et al [23] removed common occurring words which later on were referred as stopwords and removes unique symbols and urls which gave their model clearer parameters for training.

2.4 HYBRID TECHNIQUES

These were combinations of rule based filters and machine learning algorithms such as NB and SVM. The rule-based filter may be used to quickly and accurately identify messages that match certain predefined criteria, such as messages containing specific keywords or originating from known spam domains. The machine learning algorithm can then be used to classify the remaining messages based on their content and other features.

Murugan et al [24] used Decision trees and Genetic algorithm for training their model, the decision trees were splitting the data based on Gini index and the genetic algorithm would then create a new offspring in the decision tree which would refine the features such as specific word count and ratio of length to common characters. Ghourabi et al [25] used TF-IDF vectorizer with CNN and LSTM model and tested SVM, NB, Decision trees, RF and comparing the models resulted in CNN-LSTM model having 0.2% increase in accuracy compared to traditional algorithms. Zagrouba et al [26] would use supervised and unsupervised learning in their model building approach and using TF-IDF with K means algorithm and LR for hybrid model and obtained 98.8% accuracy but 95.2% precision.

2.5 ARTIFICIAL IMMUNE SYSTEMS

AIS is used along with Negative Selection Algorithm (NSA) to reduce the number of false positives to improve the precision of the models. This algorithm is based on the principle of self and non-self. The algorithm first generates a set of self-antigens, which represent the normal behavior of the system or data. Then, when a new message arrives, the algorithm checks if it contains non-self-antigens that are not part of the normal behavior

of the system. If the message contains non-self-antigens, it is classified as spam. Saleh et al [27] used NSA and get 98.57% accuracy after 6 cycles, but the model would not perform similarly if trained again because of the selection algorithm being random and it would also not help if the dataset was skewed.

Table 2. 1: Survey of Papers

Type	Approached used	Techniques discussed	Inference	Limitations
Rule Based	Rule-based Framework [15]	Created rules such as fake url, advertising messages.	9 Rules created by authors showed that stopwords are most efficient	Many more rules could be added to increase the efficiency
	Constant Time Complexity [16]	Used hashes of words and compared to rules.	Increase in number of rules increased with time consumed	Multiple rules on a huge dataset would take a lot of time.
	Splitting SMS phishing from SMS [17]	Rules for splitting messages in two groups	Used rules focused on messages if they contain the word winner or are ads	Only two rules is not enough for detection of spam and further processing will need to be done
Content Based	Static SMS Spam [18]	Linguistic inquiry and word count	Counts the number of common spam categorized words	Dataset skewedness highly impacts the accuracy of model
	SMS Spam Filtering [19]	Bayesian filters like SpamAssassin	Gives 97% accuracy	Accuracy can be increased with ensemble learning
	Deep Learning to Filer SMS Spam [20]	CNN models	Model gives 96% accuracy	High computation
Feature Extraction	Feature Selection	Applied Information Gain	Feature extraction gives a slight	Only Naïve Bayes and SVM

	Techniques [21]	technique instead of chi square	increase in accuracy	algorithms were used
	Feature Extraction and Selection on SMS [22]	Chi square and gini index	Message length, presence of url taken in account	Spelling and capitalization affects the model
	Feature Extraction Optimization [23]	Removing common occurring spam keywords, url links, monetary symbols	Gives model more parameters	Spelling and capitalization affects the model
Hybrid	Detecting Streaming of Twitter Spam [24]	Decision trees and genetic algorithm	Combined 2 models	New models accuracy depends on the performance of the other two
	A hybrid CNN-LSTM model [25]	TFIDF, CNN	Combining CNN and LSTM gives a model with 96% accuracy	Training model takes a lot of computational resources
	Hybrid SMS Spam Filtering System [26]	Naïve bayes, SVM, linear regression	Tuning SVM with TF-IDF have 100% precision	Naïve bayes will be a better model in most cases

Rule based filtering was the first method that was used in detecting spam in text, but because of the huge possibilities and increasing amount of varying text, this technique was not feasible. Content based technique was a huge step as it took the sender name, the content, the language and repetition of certain words that would indicate spam in text. Feature extraction is used nowadays as it gives multiple datapoints to be included in the model. Hybrid techniques have enabled combining Content based filtering and Feature extracting along with more complex models such as CNN-LSTM along with NB, SVM and so on.

CHAPTER 3

SYSTEM REQUIREMENTS

This chapter details the requirements needed for the program to execute. The hard ware and software requirements are listed below.

3.1 SOFTWARE REQUIREMENTS

Python3: Python3 was used as the language for both parts of this project. It was used for training the AI models using various libraries and was also used for creating the Image Steganography tool which was integrated with the KNR model. The specific version used for this project was python 3.9

Jupyter Notebook: Jupyter Notebook was used as the IDE for the SMS Spam Detector model

Python IDLE: IDLE was used as the IDE for the Image Steganography tool. The base IDE was enough to build the tool with python libraries.

Numpy: Numpy library was used for creating multi-dimensional arrays for storing the dataset.

Pandas: Pandas library helped in importing the csv data into a dataframe. It was used for data preprocessing such as storing the features extracted as words, creating new dataframe after vectorizing the text and using MinMaxScaler on and many more frames were created using this library while training the multiple models and storing the accuracy and precision of the said models.

Nltk: The Nltk library provided multiple functions that were used to preprocess the textual data. Using the library, Number of Characters, Words and Sentences were obtained and were used as features for training the models.

Seaborn: Seaborn library was to visualize data at the Exploratory Data Analysis phase of the project

Wordcloud: Wordcloud library was used to visualize most occurring words in the spam and ham dataset

Sklearn: Sklearn library was used to obtain algorithms and were used to train the models. The train_test_split function was used to split the dataset and then NB, SVM, RF. KNN and other algorithm functions were imported from the library and accuracy and precision was obtained which was later used in choosing the models for the custom ensemble model.

TfidfVectorizer: the vectorizer was used to convert the textual count of words into numerical values based on their value in the context of the message.

Tkinter: Tkinter library was used to create the GUI of the Image Steganography tool, its built in functions such as Labels and Buttons were used to build the whole GUI of the tool.

Stegano: stegano python library was used to import the LSB function which was configured to used 1-bit LSB method in hiding text data.

3.2 HARDWARE REQUIREMNETS

The SMS Spam Detection model and the Steganography tool was developed on a Laptop but it is recommended to use a Laptop or a Desktop with the following configurations:

4 GB of DDDR4 RAM or higher: Atleast 4 GB RAM is recommended for training the algorithmic models and optimizing them with varying max_features.

NVIDIA GTX 1650 or higher: The powerful the GPU used in the system will also affect the speed of training the models.

Windows 10 or higher: It is recommended to check if all the libraries required are compatible with the operating system and each other.

I5 Processor or higher: Higher quality processor will have more threads and cores which will help in faster model training.

CHAPTER 4

PROPOSED METHODOLOGY

This chapter describes the modules and the steps taken for creating the custom ensemble model, KNR. Three AI models, Naïve Bayes, K Nearest Neighbors and Random Forest were integrated for the proposed model.

4.1 ARCHITECTURE OF SPAM DETECTION MODEL

The dataset will be preprocessed and then the previously discussed models will be trained, from the trained models, the best performing models will be chosen for the custom model created using Ensemble Learning and they are picked based on their accuracy and precision. The flow of the process in shown in Figure 4.1.

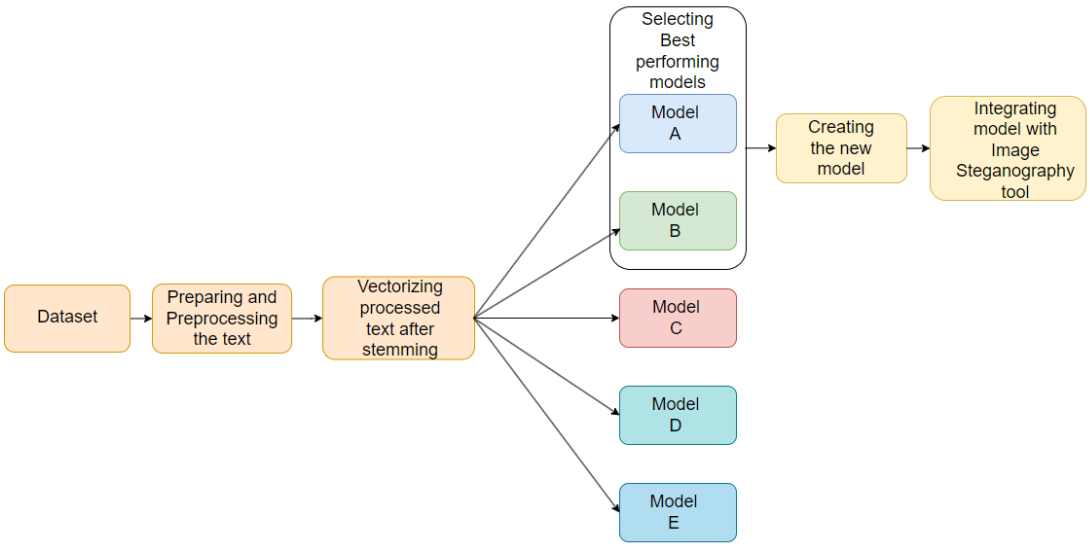


Figure 3.1: Architecture of training Spam Detector models

4.2 EXPLORATORY DATA ANALYSIS

The dataset was taken from UCI Machine Learning from Kaggle. It consists of 5574 text messages, out of which 418 were duplicate so while preparing the dataset, we remove the duplicates and are left with 5156 text messages which consists of 4503 ham messages and 653 spam messages as shown in Table 4.1.

Table 4.1: Dataset Composition

SPAM	HAM	TOTAL
653	4503	5574
12.66%	87.34%	100%

When visualizing the dataset after removing the duplicates, we see that there is 12.66% spam messages and 87.34 ham messages as in Figure 4.2. We extract features in the dataset which are number of characters, number of words and number of sentences and we see that ham messages are shorter in all three features whereas spam messages are longer. Using heatmap to see the correlation of these features we see that the correlation of number of characters is lowest compared to other features so we will select the number of characters as primary feature.

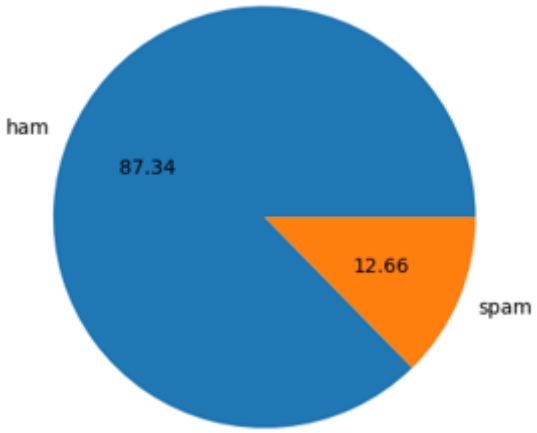


Figure 4.2: Spam to Ham ratio

Using nltk, features from the dataset were extracted such as Number of character, Number of words and Number of sentences and with seaborn library, the data was visualized using histplot function. The ham data is colored as blue and spam data is colored as red.

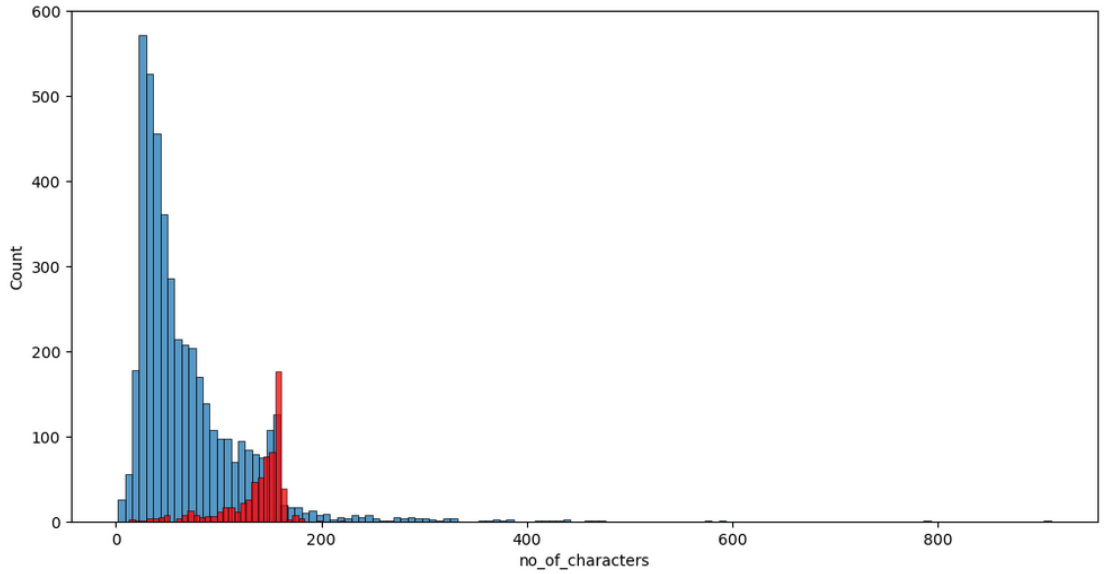


Figure 4.3: Count of Number of Characters

Figure 4.3 displays the count of Number of Characters in dataset. The legitimate/ham labelled words are highlighted in blue whereas red indicated characters in spam words.

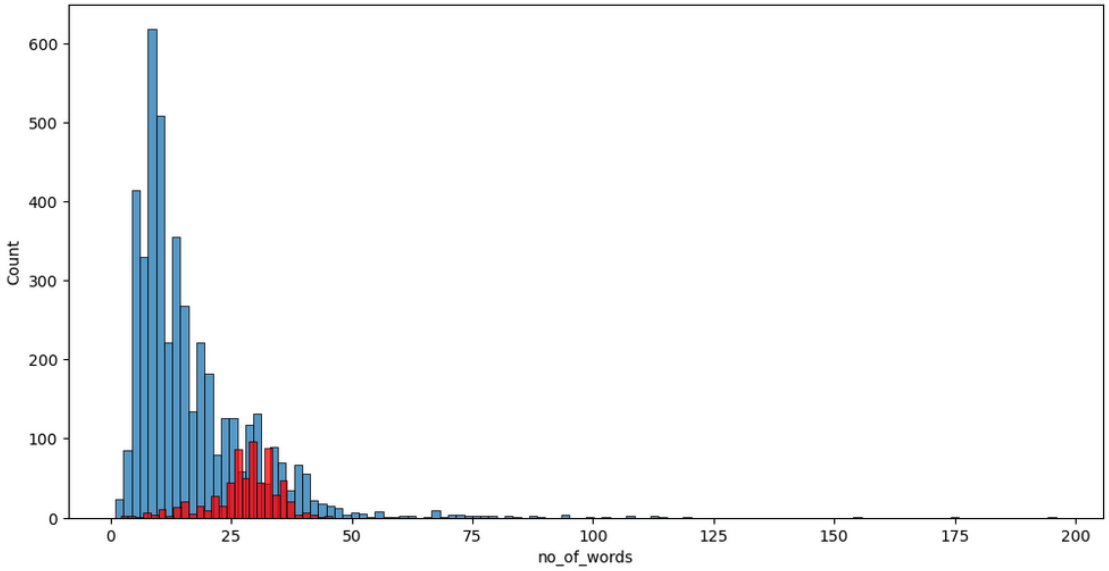


Figure 4.4: Count of Number of Words

Figure 4.4 displays the count of Number of Words in dataset. The legitimate/ham labelled words are highlighted in blue whereas red indicated characters in spam words.

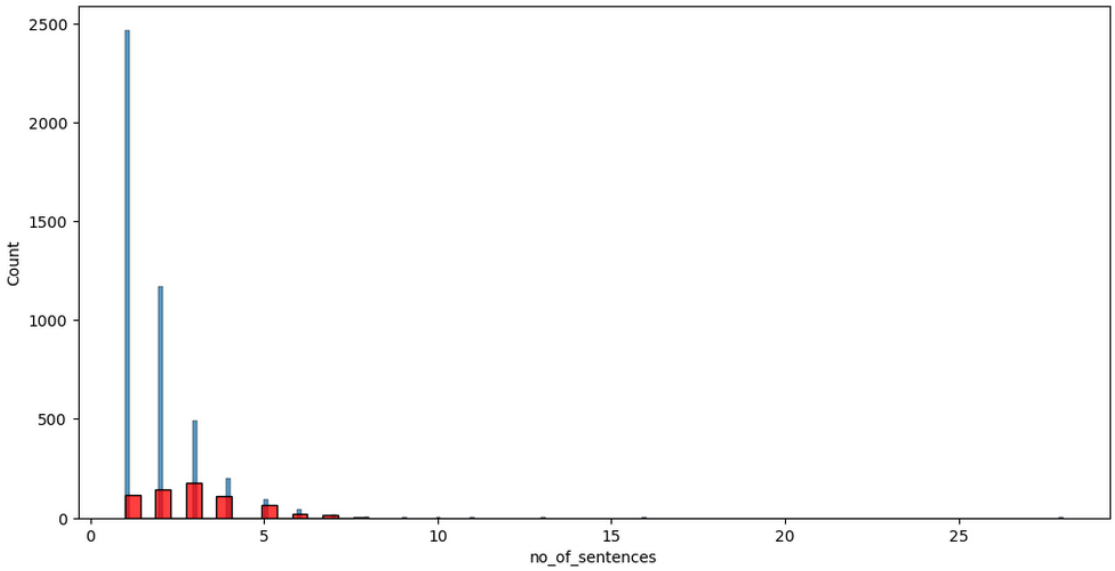


Figure 4.5: Count of Number of Sentences

Figure 4.5 displays the count of Number of Sentences in dataset. The legitimate/ham labelled words are highlighted in blue whereas red indicated characters in spam words.

From the Figure 4.3 Figure 4.4 and Figure 4.5, it was observed that the number of characters, words and sentences of ham messages are smaller in length as compared to spam messages respectively.

This shows that people in contacts send shorter messages most of the time whereas a scammer would send a lengthy message to their victim. This inference is also extracted as a feature and later implemented in the model.

For later processes such as feature extraction, we find the correlation between the different filters and we see number of words and number of sentences has the highest correlation so we will have to consider there will be high bias in the model.

Using heatmap function from seaborn library, a heatmap is generated representing correlation between Number of characters, Number of words and Number of Sentences in Figure 4.6.

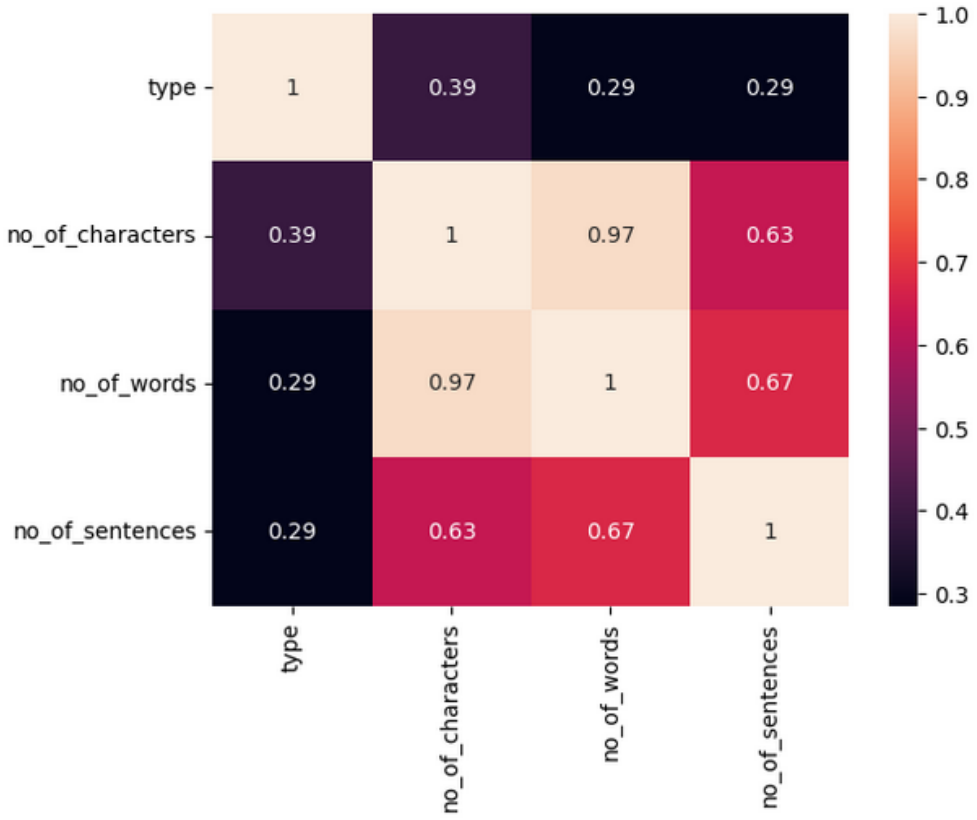


Figure 4.6: Corelation Heatmap

To visualize the words that are most common in spam and ham messages, wordcloud library is used to generate a word cloud for the respective labels. Spam wordcloud shows the most common words used such as “call” , “free”, “claim”, “reply”, “win” and so on tries to indicate a sense of urgency to its victim or lure them with a reward and tell them to contact the scammer again to obtain their so called prize as in Figure 4.7.

As for the Ham wordcloud, the most common words used are words such as “come”, “go”, “love”, “home”, “call”, and so on indicating familiarity with the message sender who might be a friend, spouse, family or work colleagues as in Figure 4.8.

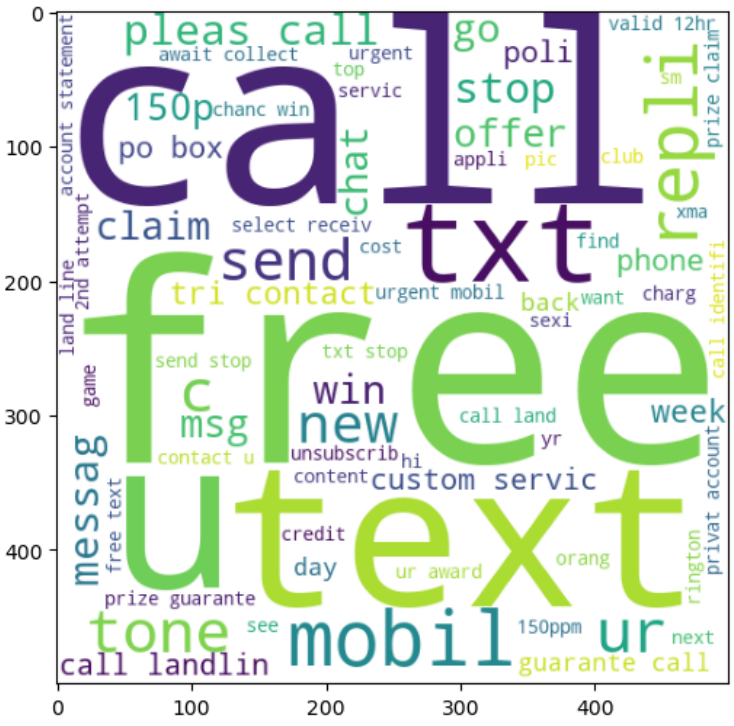


Figure 4.7: Spam Data Wordcloud

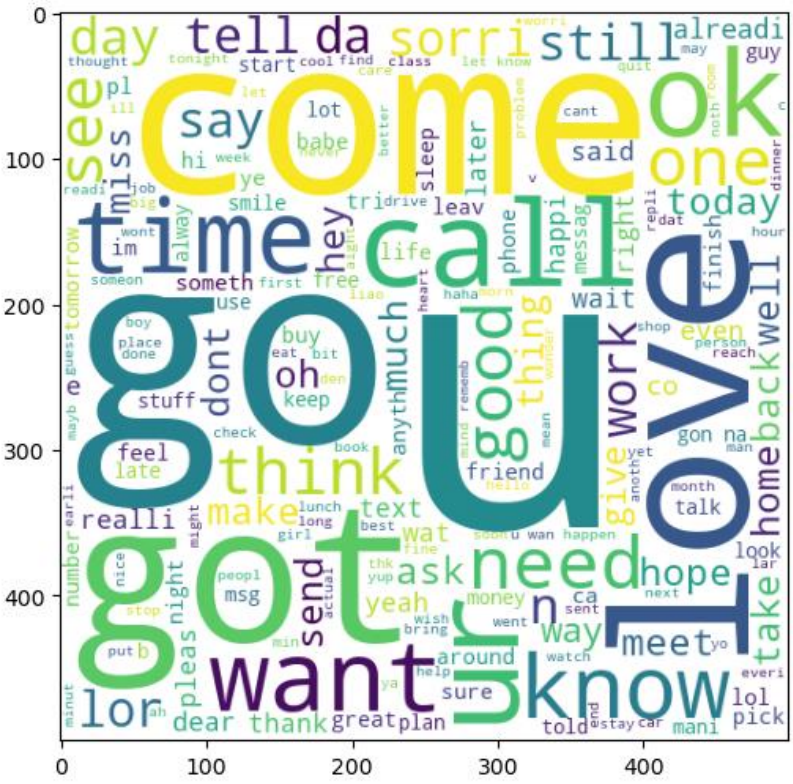


Figure 4.8: Ham Data Wordcloud

Visualizing the frequency of these words in the spam and ham wordcloud was done using the Counter function along with seaborn. The X axis in the graphs show the frequency while the Y axis show the words found in the respective wordclouds.

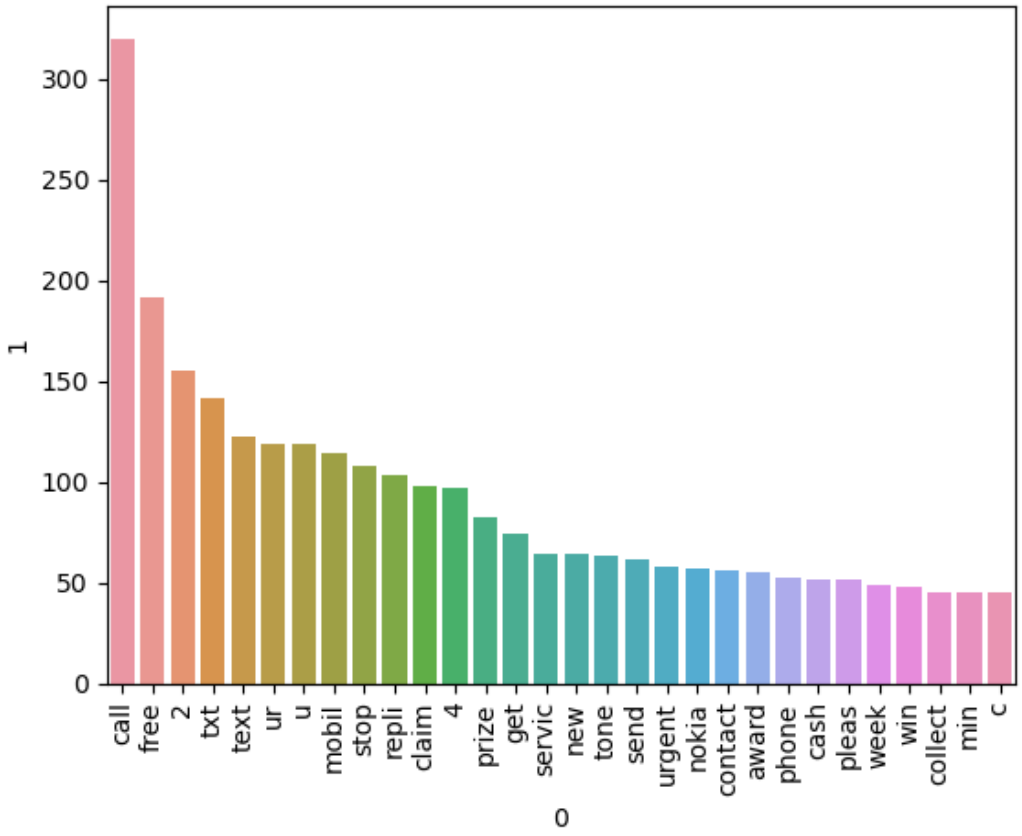


Figure 4.9: Frequency of words in spam dataset

The frequency of the words “call”, “free”, “text” are very high in the spam labelled dataset as in Figure 4.9. This documents the frequency of words which will be taken in calculation of their TF-IDF value while vectorizing them to convert them from text to numbers so that the model can take them as input.

Whereas informal and shorter language such as “u”, “ur” ,”go” as seen in the ham labelled dataset as in Figure 4.10.These will also be vectorized using TF-IDF and will be labelled as ham/legitimate for the model.

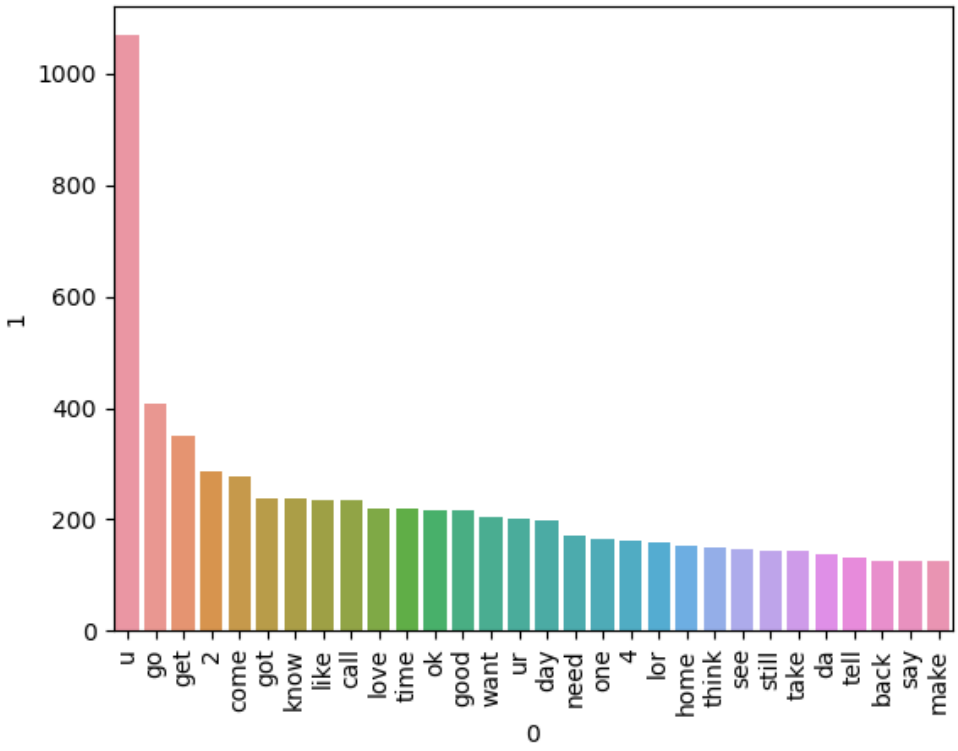


Figure 4.10: Frequency of words in ham dataset

4.3 DATA PREPROCESSING

Data preprocessing is an essential step in the data analysis pipeline, as it helps to ensure that the data is properly formatted and cleaned before it is used to build machine learning models or other data analysis techniques. The quality of the data used to build models has a direct impact on the accuracy of the results. Data that is missing values, contains errors or inconsistencies, or is biased can lead to inaccurate models. Data collected from multiple sources may be stored in different formats, making it difficult to integrate into a single dataset. Data preprocessing can help to standardize the format and ensure consistency across the data. Machine learning models rely on data to learn patterns and relationships. Preprocessing can help to ensure that the data used to build the models is in the correct format, contains relevant features, and is free from noise or errors, which can improve the accuracy of the models.

The primary steps taken for preprocessing are explained in the following paragraph:

Lowercasing: The reason why we lowercase the text data is that words with different cases are treated as different words by the computer. For instance, “Cat” and “cat” would be treated as different words even though they refer to the same object. This would lead to sparsity and redundancy in the dataset. By converting all the text data to lowercase, we can eliminate this redundancy and simplify the dataset, making it easier to work with.

Tokenization: Tokenization is a text preprocessing technique that involves breaking down a piece of text into smaller units called tokens. These tokens could be words, phrases, or other meaningful sequences of characters, such as punctuation marks or numerical values. The main purpose of tokenization is to convert a text document into a form that can be easily processed by a computer. By breaking down the text into smaller units, we can analyze and manipulate the text data more easily, making it suitable for further analysis, such as sentiment analysis or text classification. For our model, we use the NLTK tokenizer to divide the text into a list of words/tokens. This step is important for feature extraction, as it allows the model to capture the frequency of individual words in the text.

Removal of special characters: Special characters are symbols, punctuation marks, or other non-alphanumeric characters that are not part of the standard alphabet or numerical system. Examples of special characters include commas, periods, exclamation marks, question marks, parentheses, and hyphens, among others. The main reason for removing special characters is to simplify the text data and make it easier to process. By removing special characters, we can reduce the noise in the text data, which can improve the accuracy of subsequent analysis, such as text classification or sentiment analysis. Tokens obtained from the previous step may contain special characters, such as punctuation marks and symbols, which do not play any role in the classification of messages. Therefore, in this step, we remove special characters using regular expressions.

Removal of stopwords: Stopwords are the most common words which are used in grammar, like pronouns and prepositions and so on. These words do not add much meaning to the text and can be safely removed without affecting the classification accuracy. In this step, we remove stopwords using NLTK’s built-in list of stopwords.

Stemming: Stemming is a text preprocessing technique that involves reducing words to their root or base form, known as a stem, by removing suffixes or prefixes. The purpose of stemming is to reduce the dimensionality of the data and to group together variations of a word that have the same meaning. To avoid duplicate entries and capture the root form of a word, stemming is performed as a preprocessing phase on the tokens obtained from tokenization. For example, stemming the root word “like” includes likes, liked, liking, likely, and so on. There are several stemming algorithms available, such as Porter stemmer and Snowball stemmer, which can be used depending on the requirements of the project.

4.4 CHOOSING NB VARIANT

The dataset was divided into 80-20 ratio of train-test data for the model as shown in Table 4.2. Tf-Idf vectorizer from sklearn library was used based on [20]. After creating out testing and training data we first create three Naïve Bayes models for choosing our primary base model, they are Gaussian NB, Multinomial NB and Bernoulli NB.

Table 4.2: Train Test Split

Train data	Test data
80%	20%
4459 messages	1115 messages

GNB, MNB, and BNB are all variants of the I Bayes algorithm used for classification tasks. The key difference between these variants lies in the assumptions they make about the distribution of the features.

Gaussian Naive Bayes assumes that the continuous features follow a Gaussian distribution, i.e., a normal distribution. It is well-suited for continuous data and can be

used for both binary and multi-class classification problems. Multinomial I Bayes assumes that the features follow a multinomial distribution, i.e., the frequency of occurrence of each feature is modeled as a discrete probability distribution. It is particularly well-suited for text classification problems, where the data consists of frequency counts of words or other discrete features. Bernoulli I Bayes assumes that the features are binary, i.e., they can take on only two values, such as 0 or 1. It is well-suited for binary data and can be used for both binary and multi-class classification problems.

4.5 VECTORIZER FOR THE MODEL: TF-IDF

Vectorizing is a technique used to convert text data into a numerical representation that can be used by machine learning algorithms. A vectorizer maps each word or term in the text data to a unique number or index, which can then be used to create a vector representation of the text.

The use of vectorizers is important in NLP because machine learning algorithms require numerical input data, and text data is inherently non-numerical. By converting text data into a numerical representation using a vectorizer, the data can be used as input to machine learning algorithms for tasks such as classification, clustering, and sentiment analysis.

Vectorizers are also useful in reducing the dimensionality of text data. Text data often contains many words or terms, which can result in a large number of features or dimensions in the input data. By using a vectorizer, the text data can be represented in a lower-dimensional space, which can improve the performance of machine learning algorithms and reduce the computational cost of processing the data.

The three commonly used vectorizers are CountVectorizer, TF-IDF Vectorizer and Word2Vec. CountVectorizer is a simple technique that counts the frequency of each word or term in the text and creates a sparse matrix representation of the text. It assigns a numerical value to each word in a text based on its frequency of occurrence. CountVectorizer is useful for tasks such as document classification and topic modeling, but it does not take into account the importance of each word or term in the text. TF-IDF Vectorizer is similar to CountVectorizer, but it also takes into account the importance of each word or term in the text by weighting the frequency based on its inverse document

frequency. This means that words that occur frequently across all documents are given a lower weight, while words that occur less frequently are given a higher weight. TF-IDF Vectorizer is useful for tasks such as information retrieval and text classification, where the goal is to identify the most important words or terms in the text. Word2Vec is a neural network-based technique that creates a vector representation of words based on their context in the text. It is capable of capturing the semantic relationships between words and can be used to perform tasks such as word analogy and sentence similarity. Word2Vec is more complex and computationally intensive than CountVectorizer and TF-IDF Vectorizer, but it can be more effective in capturing the meaning of words and phrases in the text.

As this project focuses on text classification, TF-IDF Vectorizer was used for the model. TF-IDF, or term frequency-inverse document frequency, is a powerful tool used to determine the relevance of words within a given corpus. This measurement is widely considered to be one of the most effective ways to assess the significance of a term in a given text. The TF-IDF framework assigns a weight to each word in a document based on the frequency of its occurrence within the text, as well as the reciprocal of its frequency across the entire corpus.

When it comes to terminology, there are a few key terms that are worth noting. Firstly, the term frequency (TF) of a given word or term is the number of times that term occurs within a particular document. This is divided by the total number of words in the document to arrive at the TF value.

Next, we have the document frequency (DF) of a term, which refers to the number of documents within a given corpus that contain that term. This is a crucial factor in determining the relevance of a term, as it can help us to understand how frequently it appears across a wide range of texts.

Finally, there is the inverse document frequency (IDF) of a term. This is calculated by taking the logarithm of the total number of documents in the corpus, divided by the number of documents that contain the term in question. Essentially, IDF helps us to understand how unique or rare a particular term is within the overall corpus.

Putting these elements together, we arrive at the full TF-IDF formula. This is calculated by multiplying the term frequency by the inverse document frequency, as represented in the equation (1):

$$TF - IDF = TF * IDF \quad \text{.....(1)}$$

When a particular term has a high TF-IDF score, it indicates that the term appears frequently within a particular document, but is relatively rare across the entire corpus. Conversely, a low TF-IDF score suggests that the term appears frequently across the entire corpus, and may not be as relevant to the specific document in question.

Overall, TF-IDF is a highly effective tool for assessing the relevance and significance of specific terms within a given corpus. By taking into account the frequency of a term both within and outside of a specific document, this framework can help to identify the most important and distinctive features of a text.

4.6 PROCESSING THE EXTRACED FEATURES

After getting the features such as Number of Characters, Words and Sentences, along with the words that are vectorized using TF-IDF will range from 0 to more than 1. As this project focuses on binary classification, i.e spam and ham messages, the count of these features and vectors will be scaled from 0 to 1 using the MinMaxScaler function from sklearn library.

MinMaxScaler is used when we want to normalize the feature values of a dataset to a fixed range. This is particularly useful when the feature values have different scales or units of measurement, which can cause problems for machine learning algorithms that assume all features are on the same scale. It is also useful when working with algorithms that are sensitive to differences in feature scales, such as k-nearest neighbors or support vector machines. By scaling the features to a common range, we can ensure that these algorithms will not be biased towards features with larger values or ranges.

The MinMaxScaler works by first subtracting the minimum value of the feature and then dividing by the range of the feature, which is the difference between the maximum and minimum values. The resulting values are then scaled to the desired range, typically 0 to 1. The formula for MinMaxScaler can be represented in (2):

$$scaled\_value = (original\_value - min\_value) / (max\_value - min\_value) \ldots (2)$$

Where the scaled value is equal to the difference in original value and min value divided by max value and min value.

4.7 MODEL BUILDING: MULTIPLE ALGORITHMIC MODELS

After choosing as MNB as our NB model and completing the build test for one model, further models will be trained based on NB, KN, RF, ETC, SVM, AdaBoost, LR, XGB, GBDT, BgC and DT. Using TF-IDF as the vectorizer once again, the trained models will be optimized as well using max_features parameter in TF-IDF Vectorizer.

The max_features parameter in TF-IDF is used to specify the maximum number of features (unique words) to be considered during vectorization. It is an optional parameter, and its default value is None, which means that all features (unique words) in the corpus will be considered during vectorization. When the number of unique words in the corpus is large, it can lead to a high-dimensional vector representation, which can negatively impact the performance of machine learning models due to the curse of dimensionality. Therefore, setting a limit on the number of features (unique words) can help in reducing the dimensionality of the vector representation.

The value of max_features can be an integer or a float. If it is an integer, then it represents the maximum number of features to be considered. If it is a float, then it represents the maximum percentage of features to be considered, i.e., it specifies the fraction of features to be used. For this project, the models were trained with max_features set as 1000, 2000 and 3000 for all the models in case if performance of any model improved after changing

this parameter. After training the models three more times, we have 11 models at 4 different optimizations. It was observed that the models had ± 1% change in accuracy but upto ± 4% change in precision on different variant of the same algorithm as shown in Table 4.4.

4.8 CUSTOM ENSEMBLE MODEL: KNR

After optimizing the models with changing max_features 3 times, the top performing models were KNN, NB and RF each at max_features at 3000. Stacking classifier used for creating the ensemble model. In Stacking, multiple models are trained on the same dataset, and the predictions from these models are then combined to make a final prediction. Comparison of chosen models is shown in Table 4.5.

The stacking classifier consists of two stages. In the first stage, multiple base models are trained on the training data. In the second stage, a meta model is trained on the outputs of the base models, which are used as input features. The meta model learns to combine the outputs of the base models to make a final prediction.

The reason for using stacking is that it uses a more complex approach to combine the predictions of the base models. Bagging and boosting typically use simple methods such as averaging or weighted averaging to combine the predictions of the base models. This can be effective, but it doesn't take into account the individual strengths and weaknesses of each base model. Bagging was also used before in this project with default configurations to observe the accuracy and prediction of the model. In contrast, stacking uses a meta-model that learns how to combine the predictions of the base models. This meta-model can take into account the strengths and weaknesses of each base model and learn how to weight their predictions accordingly.

Table 4.5: Comparison on custom model with its base models

Model	Accuracy	Precision
KNN	90.60%	100%
NB	97.38%	98.23%
RF	96.89%	100%

KNR	97.48%	94.26%

4.9 IMAGE STEGANOGRAPHY TOOL IMPLEMENTATION

To develop an effective spam detection model within a steganography tool, it is necessary to merge two distinct functionalities: steganography and spam detection. Steganography, a technique used to hide secret messages within digital images or other media, requires embedding the hidden message in a cover image or media, while spam detection involves identifying and filtering unwanted or unsolicited messages in email, social media, or other online platforms.

To achieve this, we implemented a tool using the tkinter framework and a 1 bit LSB steganography technique, which entails replacing the least significant bit of each pixel in the cover image with a corresponding bit of the secret message. The first step in this process was designing a user-friendly graphical interface using tkinter. The interface allows users to input their message and select the image they want to use as the cover image for embedding the message.

Incorporating a spam detection algorithm in the tool ensures that the message being embedded is free of any malicious or unsolicited content. Before embedding the message in the image, the algorithm scans it to detect any potential spam or malicious content, flagging it for the user's attention. This feature offers an additional layer of security and prevents the tool's users from unknowingly sharing or distributing spam or malware.

Once the spam detection algorithm has been run, the steganography tool can then proceed with the embedding process. The tool uses the 1 bit lsb steganography technique to encode the message in the image, making it virtually undetectable to the naked eye. This technique is accomplished by replacing the least significant bit of each pixel value with a corresponding bit from the message.

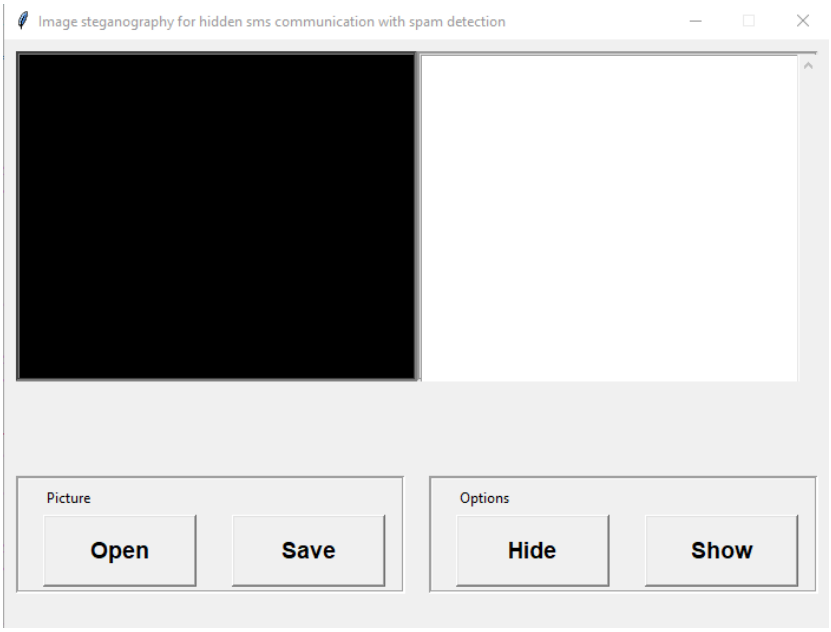


Figure 4.11: GUI Interface

The left side of interface will show the image which is picked by the sender or the image which is to be decoded by the receiver and the hidden text that is to be hidden or shown will show on the right side.

By clicking on Open will open file explorer where the person can pick an image to hide text in.

After choosing the image we can write our message on the right side and then click on hide to hide the message and then save to save the new image with the hidden text.

The sender will then send the image on sms to the receiver. This tool will also show the image size and can be customized to show over metadata as shown in Figure 4.12.

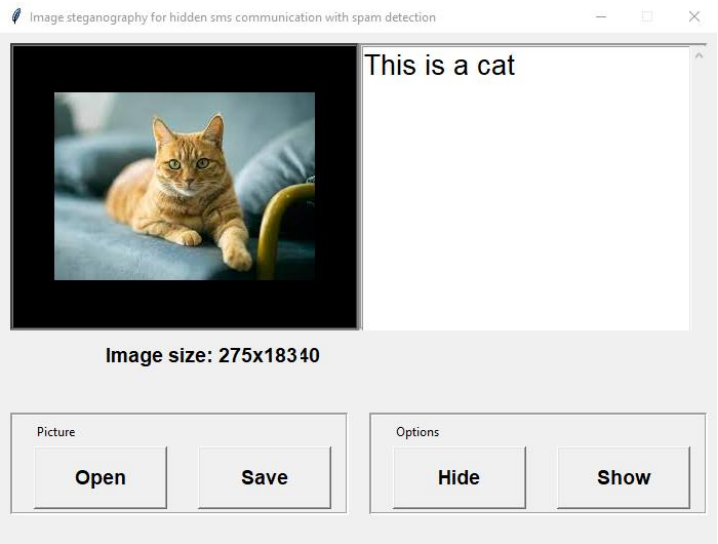


Figure 4.12: Selecting image and hiding text

When the receiver receives this image, they can click on Show, the application will run the ai model first and predict if the sent text was a spam message or legitimate message as shown in Figure 4.13 and Figure 4.14 respectively.

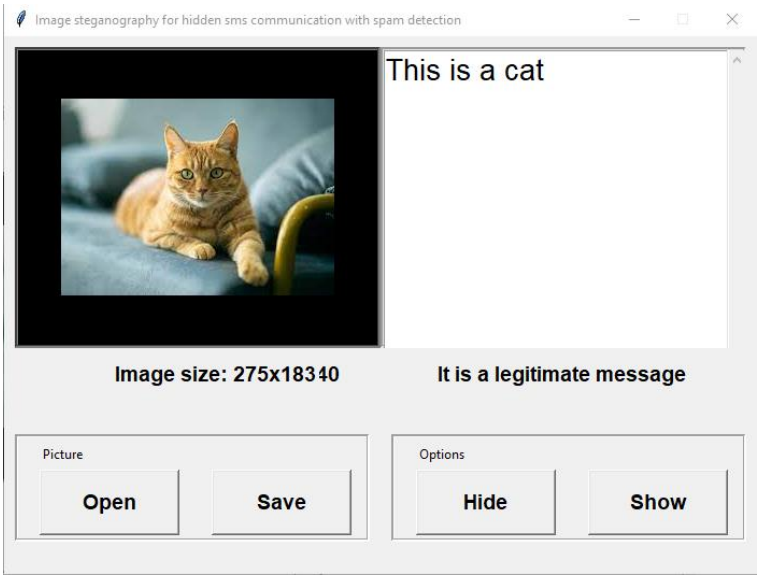


Figure 4.13: Example of Text detected as legitimate

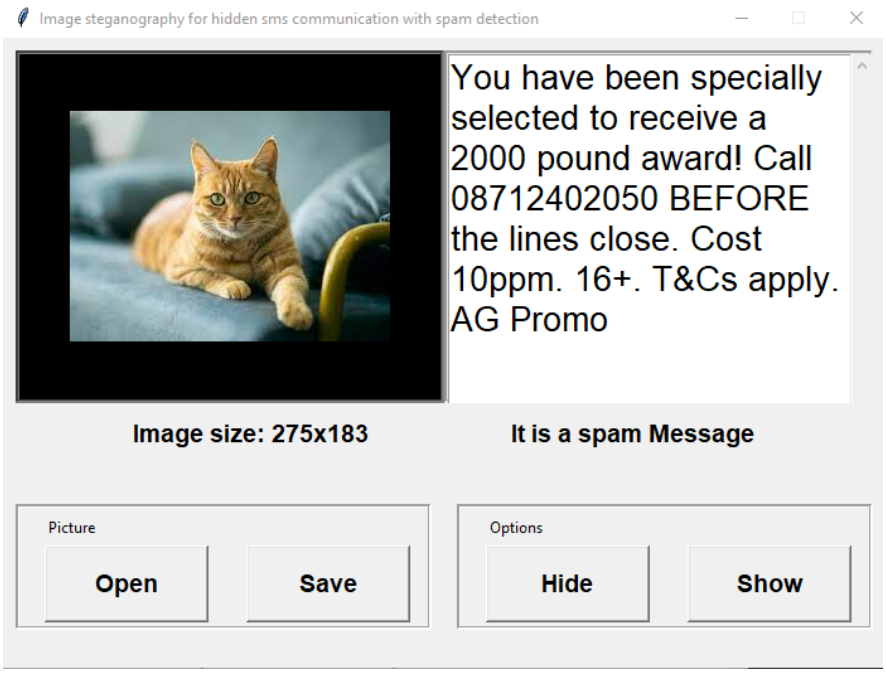


Figure 4.14: Example of Text detected as spam

All the previous five stages combined together created framework for the model. After creating the model, the variation of Naïve Bayes model was chosen and further models were trained and compared. KNN, NB and RF models were combined with Strapping technique and it was successfully implemented with the Image Steganography tool. The tool can hide the message in the image and when decrypting the message, the tool will detect if the message was spam or not.

CHAPTER 5

RESULTS AND DISCUSSIONS

This chapter explains the findings after the models were trained. The final results after building models were as follows:

The NB variant chosen was Multinomial Naïve Bayes as it performed better than Gaussian NB and Bernoulli NB as shown in Figure 5.1:

Table 5.1: Results of Naïve Bayes Variants

Algorithm	Accuracy	Precision
Gaussian NB	85.07%	46.4%
Multinomial NB	95.63%	100%
Bernoulli NB	96.70%	98.11%

The high performing models were K Nearest Neighbors with 97.95% accuracy with 100% precision, Naive Bayes with 96.63% accuracy and 98.24% precision and Random Forest with 97.19% accuracy and 100% precision, hence these three models were chosen for the custom ensemble model.

The custom model was created using Stacking classifier with Naive Bayes model as the base estimator. KNR model resulted with 97.48% accuracy and 94.26% precision.

The final output of this process is a steganographic image, which can be saved and shared like any other normal image as in Table 5.2.

Table 5.2: Accuracy and Precision of Algorithmic Models with optimization

Algorithm	Accuracy	Precision	Accuracy at max features set to 1000	Precision at max features set to 1000	Accuracy at max features set to 2000	Precision at max features set to 2000	Accuracy at max features set to 3000	Precision at max features set to 3000
KN	89.63	100	91.37	97.95	91.18	100	90.60	100
NB	95.63	100	97.57	96.63	97.48	98.24	97.38	98.23
RF	96.22	100	97.28	99.09	97.19	100	96.89	100
ETC	96.89	100	97.48	95.83	97.48	99.10	97.38	98.23
SVM	97.48	98.24	97.48	98.24	97.57	97.43	97.86	98.30
AdaBoost	96.80	96.39	95.93	92.72	96.31	93.75	96.31	92.98
LR	94.76	95.55	95,63	95.04	95.54	95.91	95.05	95.69
XGB	96.70	93.96	96.89	96.42	96.70	97.22	96.70	94.73
GBDT	94.67	90.90	95.25	96.77	95.25	98.87	95.44	98.90
BgC	95.34	86.66	94.67	81.39	95.54	86.88	95.25	85.95
DT	92.92	83.15	92.73	78.50	93.31	79.27	92.92	81.81

Table 5.3: Comparison on custom model with its base models

Model	Accuracy	Precision
KNN	90.60%	100%
NB	97.38%	98.23%
RF	96.89%	100%
KNR	97.48%	94.26%

The embedded message can be extracted using the same steganography tool. However, before displaying or sharing the extracted message, it is imperative to rerun the spam detection algorithm to ensure that it does not contain any malicious content or spam. The comparison of the KNR model along with its base models are displayed in Table 5.3. This model was implemented in the Image Steganography tool. The tool has 4 options, Open, Save, Hide and Show. After clicking on open, the user can choose an image as their cover image to hide their message. After choosing the image, the user can type their message and then click Hide to hide it in the cover image and click on Save to save it. This image when received by the receiver can be decrypted by opening the image by clicking Open and clicking on Show and then the KNR model processes the text data with the accuracy and precision as displayed by the model shown in Table 5.3.

CHAPTER 6

CONCLUSION AND FUTURE WORK

Researchers are still exploring the field of spam detection, our contribution has focused on improving the current models after doing an in depth analysis by comparing multiple algorithmic models and implementing additional security with Image Steganographic Techniques. This chapter details the conclusion of this research work as well as future enhancements that can be done with this research.

6.1 CONCLUSION

The custom ensemble model KNR was created using Multinomial Naïve Bayes, K Nearest Neighbors and Random Forest classifiers with using Stacking Ensemble technique has 97.9% accuracy and 95.57% precision dubbed as KNR model. The models individually had 100% precision but then it would imply that its completely detecting the test spam texts from the dataset but it is possible that the model can label even legitimate messages as spam so having some leeway is preferable.

As for the accuracy of the individual models with respect to the custom model, the custom model gives 7% higher accuracy than KN, 0.5% higher than NB and 1% higher accuracy than RF models when max features is set to 3000.

As for the Image Steganography tool, the tool accepts .jpg and .png format images and is able to hide text successfully without any loss of the hidden text, and when a steganographed image is decrypted, the original message is the same as the hidden message and the spam detector module of the tool is able to successfully label if the text is spam or legitimate and this tool is lightweight and easy to use and is compatible with proprietary and private messaging applications. This satisfies the overall objective of the research.

6.2 FUTURE ENHANCEMENTS

For the Spam Detector part, the dataset can by increased if the format of the dataset is compatible with the dataset used in this project, or can be converted to the format using pandas but might not work properly. Additionally, features such as reinforced learning like asking the user if the detector was able to label the prediction correctly or not could increase the model's accuracy.

As for the Image Steganography tool, 1 Bit LSB was used in the project but it is possible to implement other techniques of steganography such as Spread Spectrum and Phase coding but it also comes with drawbacks of higher chances of corrupting the hidden message, it can be mitigated using 2 bit LSB or 3 but LSB for shorter sized messages as larger messages can visibly change the cover image.

This work can be integrated with popular SMS/MMS applications such as Google Message and Apple Messages but the detection model would require multiple GB's worth of datasets if it were to be implemented with such a big application.

REFERENCES

[1] Ling, R. (2010). Texting as a life phase medium. Journal of Computer-mediated communication, 15(2), 277-292.

[2] Peeters, C., Patton, C., Munyaka, I. N., Olszewski, D., Shrimpton, T., & Traynor, P. (2022, May). SMS OTP Security (SOS) Hardening SMS-Based Two Factor Authentication. In Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security (pp. 2-16).

[3] Alharbi, E., & Alghazzawi, D. (2019). Two factor authentication framework using otp-sms based on blockchain. Transactions on Machine Learning and Artificial Intelligence, 7(3), 17-27.

[4] Awale, S. M., & Gupta, P. (2019). Awareness of sim swap attack. International Journal of Trend in Scientific Research and Development, 3(4), 995-997.

[5] Mishra, S., & Soni, D. (2019, August). SMS phishing and mitigation approaches. In 2019 twelfth international conference on contemporary computing (ic3) (pp. 1-5). IEEE.

[6] Ullah, K., Rashid, I., Afzal, H., Iqbal, M. M. W., Bangash, Y. A., & Abbas, H. (2020). SS7 vulnerabilities—a survey and implementation of machine learning vs rule based filtering for detection of SS7 network attacks. IEEE Communications Surveys & Tutorials, 22(2), 1337-1371.

[7] Chordiya, A. R., Majumder, S., & Javaid, A. Y. (2018, May). Man-in-the-middle (MITM) attack based hijacking of HTTP traffic using open source tools. In 2018 IEEE International Conference on Electro/Information Technology (EIT) (pp. 0438-0443). IEEE.

[8] Yeboah-Boateng, E. O., & Amanor, P. M. (2014). Phishing, SMiShing & Vishing: an assessment of threats against mobile devices. Journal of Emerging Trends in Computing and Information Sciences, 5(4), 297-307.

[9] Idika, N., & Mathur, A. P. (2007). A survey of malware detection techniques. Purdue University, 48(2), 32-46.

[10] Jhaveri, R. H., Patel, S. J., & Jinwala, D. C. (2012, January). DoS attacks in mobile ad hoc networks: A survey. In 2012 second international conference on advanced computing & communication technologies (pp. 535-541). IEEE.

[11] Abayomi-Alli, O., Misra, S., Abayomi-Alli, A., & Odusami, M. (2019). A review of soft techniques for SMS spam classification: Methods, approaches and applications. Engineering Applications of Artificial Intelligence, 86, 197-212.

[12] Lota, L. N., & Hossain, B. M. (2017). A systematic literature review on sms spam detection techniques. International Journal of Information Technology and Computer Science (IJITCS), 9(7), 42-50.

[13] Ying, X. (2019, February). An overview of overfitting and its solutions. In *Journal of physics: Conference series* (Vol. 1168, p. 022022). IOP Publishing.

[14] Nashat, D., & Mamdouh, L. (2019). An efficient steganographic technique for hiding data. Journal of the Egyptian Mathematical Society, 27, 1-14.

[15] Jain, A. K., & Gupta, B. B. (2018). Rule-based framework for detection of smishing messages in mobile environment. Procedia Computer Science, 125, 617-623.

[16] Xia, T. (2020). A constant time complexity spam detection algorithm for boosting throughput on rule-based filtering systems. IEEE Access, 8, 82653-82661.

[17] Foozy, M., Feresa, C., Ahmad, R., & Abdollah, M. F. (2014). A practical rule based technique by splitting SMS phishing from SMS spam for better accuracy in mobile device. International Review on Computers and Software, 9(10), 1776-1782

[18] Karami, A., & Zhou, L. (2014). Improving static SMS spam detection by using new content-based features.

[19] Gómez Hidalgo, J. M., Bringas, G. C., Sánz, E. P., & García, F. C. (2006, October). Content based SMS spam filtering. In Proceedings of the 2006 ACM symposium on Document engineering (pp. 107-114).

[20] Roy, P. K., Singh, J. P., & Banerjee, S. (2020). Deep learning to filter SMS Spam. Future Generation Computer Systems, 102, 524-533.

[21] Sharaff, A. (2019). Spam detection in SMS based on feature selection Techniques. In Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2018, Volume 2 (pp. 555-563). Springer Singapore.

[22] Uysal, A. K., Gunal, S., Ergin, S., & Gunal, E. S. (2013). The impact of feature extraction and selection on SMS spam filtering. Elektronika ir Elektrotechnika, 19(5), 67-72.

[23] Zainal, K., & Jali, M. Z. (2016). A review of feature extraction optimization in SMS spam messages classification. In Soft Computing in Data Science: Second International Conference, SCDS 2016, Kuala Lumpur, Malaysia, September 21-22, 2016, Proceedings 2 (pp. 158-170). Springer Singapore.

[24] Senthil Murugan, N., & Usha Devi, G. (2018). Detecting streaming of Twitter spam using hybrid method. Wireless Personal Communications, 103, 1353-1374.

[25] Ghourabi, A., Mahmood, M. A., & Alzubi, Q. M. (2020). A hybrid CNN-LSTM model for SMS spam detection in arabic and english messages. Future Internet, 12(9), 156.

[26] Baaqeel, H., & Zagrouba, R. (2020, November). Hybrid SMS spam filtering system using machine learning techniques. In 2020 21st International Arab Conference on Information Technology (ACIT) (pp. 1-8). IEEE.

APPENDIX A

SAMPLE CODE

1. CODE FOR SMS SPAM DETECTOR MODEL:

```
import numpy as np

import pandas as pd

#Importing Dataset

df = pd.read_csv('spam.csv')

df.shape

df.info()

df.sample(5)

#cleaning dataset

from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()

df['type'] = encoder.fit_transform(df['type'])

df.head()

df.isnull().sum()

df.duplicated().sum()

df = df.drop_duplicates(keep='first')

df.duplicated().sum()

df.shape

#Exploratory Data Analysis

df.head()
```

```
df['type'].value_counts()

import matplotlib.pyplot as plt

plt.pie(df['type'].value_counts(), labels=['ham','spam'],autopct="%0.2f")

plt.show()

import nltk

!pip install nltk

nltk.download('punkt')

df.head()

import seaborn as sb

plt.figure(figsize=(12,6))

sb.histplot(df[df['type'] == 0]['no_of_words'])

sb.histplot(df[df['type'] == 1]['no_of_words'],color='red')

plt.figure(figsize=(12,6))

sb.histplot(df[df['type'] == 0]['no_of_sentences'])

sb.histplot(df[df['type'] == 1]['no_of_sentences'],color='red')

sb.heatmap(df.corr(),annot=True)

#Data Preprocessing

from nltk.corpus import stopwords

import string

string.punctuation

from nltk.stem.porter import PorterStemmer

ps = PorterStemmer()

nltk.download('stopwords')
```

```
def transform_text(text):
    text = text.lower() #lowercase
    text = nltk.word_tokenize(text)#tokenize

    y = []
    for i in text:
        if i.isalnum():
            y.append(i)
    #remove special characters
    text = y[:]
    y.clear()

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

    text = y[:]
    y.clear()

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

```
    return " ".join(y)
df['transformed_text'] = df['text'].apply(transform_text)
df.head()
!pip install wordcloud
from wordcloud import WordCloud
wc = WordCloud(width=500,height=500,min_font_size=10,background_color='white')
spam_wc = wc.generate(df[df['type'] == 1]['transformed_text'].str.cat(sep=" "))
plt.figure(figsize=(15,6))
plt.imshow(spam_wc)
ham_wc = wc.generate(df[df['type'] == 0]['transformed_text'].str.cat(sep=" "))
plt.figure(figsize=(15,6))
plt.imshow(ham_wc)
spam_corpus = []
for msg in df[df['type'] == 1]['transformed_text'].tolist():
    for word in msg.split():
        spam_corpus.append(word)
len(spam_corpus)
ham_corpus = []
for msg in df[df['type'] == 0]['transformed_text'].tolist():
    for word in msg.split():
        ham_corpus.append(word)
len(ham_corpus)
#Model Building
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
```

```
tfidf = TfidfVectorizer()

X = tfidf.fit_transform(df['transformed_text']).toarray()

X.shape

y = df['type'].values

from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)

from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB

from sklearn.metrics import accuracy_score,confusion_matrix,precision_score

gnb = GaussianNB()

mnb = MultinomialNB()

bnb = BernoulliNB()

gnb.fit(X_train,y_train)

y_pred1 = gnb.predict(X_test)

print(accuracy_score(y_test,y_pred1))

print(confusion_matrix(y_test,y_pred1))

print(precision_score(y_test,y_pred1))

mnb.fit(X_train,y_train)

y_pred2 = mnb.predict(X_test)

print(accuracy_score(y_test,y_pred2))

print(confusion_matrix(y_test,y_pred2))

print(precision_score(y_test,y_pred2))

bnb.fit(X_train,y_train)

y_pred3 = bnb.predict(X_test)

print(accuracy_score(y_test,y_pred3))
```

```
print(confusion_matrix(y_test,y_pred3))
print(precision_score(y_test,y_pred3))
!pip install xgboost
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
svc = SVC(kernel='sigmoid', gamma=1.0)
knc = KNeighborsClassifier()
mnb = MultinomialNB()
dtc = DecisionTreeClassifier(max_depth=5)
lrc = LogisticRegression(solver='liblinear', penalty='l1')
rfc = RandomForestClassifier(n_estimators=50, random_state=2)
abc = AdaBoostClassifier(n_estimators=50, random_state=2)
bc = BaggingClassifier(n_estimators=50, random_state=2)
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)
gbdt = GradientBoostingClassifier(n_estimators=50,random_state=2)
```

```
xgb = XGBClassifier(n_estimators=50,random_state=2)

clfs = {
    'SVC' : svc,
    'KN' : knc,
    'NB': mnb,
    'DT': dtc,
    'LR': lrc,
    'RF': rfc,
    'AdaBoost': abc,
    'BgC': bc,
    'ETC': etc,
    'GBDT':gbdt,
    'xgb':xgb
}

def train_classifier(clf,X_train,y_train,X_test,y_test):
    clf.fit(X_train,y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test,y_pred)
    precision = precision_score(y_test,y_pred)


    return accuracy,precision

train_classifier(svc,X_train,y_train,X_test,y_test)

accuracy_scores = []

precision_scores = []
```

```
for name,clf in clfs.items():

    current_accuracy,current_precision = train_classifier(clf,
X_train,y_train,X_test,y_test)

    print("For ",name)
    print("Accuracy - ",current_accuracy)
    print("Precision - ",current_precision)

    accuracy_scores.append(current_accuracy)
    precision_scores.append(current_precision)
performance_df =
pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy':accuracy_scores,'Precision':precision
_scores}).sort_values('Precision',ascending=False)
performance_df
#Model Optimization
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
tfidf = TfidfVectorizer(max_features=1000)
X = tfidf.fit_transform(df['transformed_text']).toarray()
y = df['type'].values
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
svc = SVC(kernel='sigmoid', gamma=1.0)
knc = KNeighborsClassifier()
mnb = MultinomialNB()
```

```
dtc = DecisionTreeClassifier(max_depth=5)

lrc = LogisticRegression(solver='liblinear', penalty='l1')

rfc = RandomForestClassifier(n_estimators=50, random_state=2)

abc = AdaBoostClassifier(n_estimators=50, random_state=2)

bc = BaggingClassifier(n_estimators=50, random_state=2)

etc = ExtraTreesClassifier(n_estimators=50, random_state=2)

gbdt = GradientBoostingClassifier(n_estimators=50,random_state=2)

xgb = XGBClassifier(n_estimators=50,random_state=2)

clfs = {

    'SVC' : svc,

    'KN' : knc,

    'NB': mnb,

    'DT': dtc,

    'LR': lrc,

    'RF': rfc,

    'AdaBoost': abc,

    'BgC': bc,

    'ETC': etc,

    'GBDT':gbdt,

    'xgb':xgb

}

def train_classifier(clf,X_train,y_train,X_test,y_test):

    clf.fit(X_train,y_train)

    y_pred = clf.predict(X_test)
```

```
    accuracy = accuracy_score(y_test,y_pred)
    precision = precision_score(y_test,y_pred)

    return accuracy,precision
accuracy_scores = []
precision_scores = []

for name,clf in clfs.items():

    current_accuracy,current_precision = train_classifier(clf,
X_train,y_train,X_test,y_test)

    print("For ",name)
    print("Accuracy - ",current_accuracy)
    print("Precision - ",current_precision)

    accuracy_scores.append(current_accuracy)
    precision_scores.append(current_precision)
performance_1000_df =
pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_max_features_1000':accuracy_scores,
'Precision_max_features_1000':precision_scores}).sort_values('Precision_max_features
_1000',ascending=False)
performance_1000_df
final_df=performance_df.merge(performance_1000_df)
final_df
```

```
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
tfidf = TfidfVectorizer(max_features=2000)
X = tfidf.fit_transform(df['transformed_text']).toarray()
y = df['type'].values
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
svc = SVC(kernel='sigmoid', gamma=1.0)
knc = KNeighborsClassifier()
mnb = MultinomialNB()
dtc = DecisionTreeClassifier(max_depth=5)
lrc = LogisticRegression(solver='liblinear', penalty='l1')
rfc = RandomForestClassifier(n_estimators=50, random_state=2)
abc = AdaBoostClassifier(n_estimators=50, random_state=2)
bc = BaggingClassifier(n_estimators=50, random_state=2)
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)
gbdt = GradientBoostingClassifier(n_estimators=50,random_state=2)
xgb = XGBClassifier(n_estimators=50,random_state=2)
clfs = {
    'SVC' : svc,
    'KN' : knc,
    'NB': mnb,
    'DT': dtc,
    'LR': lrc,
    'RF': rfc,
    'AdaBoost': abc,
```

```
    'BgC': bc,
    'ETC': etc,
    'GBDT':gbdt,
    'xgb':xgb
}
def train_classifier(clf,X_train,y_train,X_test,y_test):
    clf.fit(X_train,y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test,y_pred)
    precision = precision_score(y_test,y_pred)

    return accuracy,precision
accuracy_scores = []
precision_scores = []

for name,clf in clfs.items():

    current_accuracy,current_precision = train_classifier(clf,
X_train,y_train,X_test,y_test)

    print("For ",name)
    print("Accuracy - ",current_accuracy)
    print("Precision - ",current_precision)
```

```
    accuracy_scores.append(current_accuracy)
    precision_scores.append(current_precision)
performance_2000_df =
pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_max_features_2000':accuracy_scores,
'Precision_max_features_2000':precision_scores}).sort_values('Precision_max_features
_2000',ascending=False)
performance_2000_df
final_df=final_df.merge(performance_2000_df)
final_df
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
tfidf = TfidfVectorizer(max_features=3000)
X = tfidf.fit_transform(df['transformed_text']).toarray()
y = df['type'].values
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
svc = SVC(kernel='sigmoid', gamma=1.0)
knc = KNeighborsClassifier()
mnb = MultinomialNB()
dtc = DecisionTreeClassifier(max_depth=5)
lrc = LogisticRegression(solver='liblinear', penalty='l1')
rfc = RandomForestClassifier(n_estimators=50, random_state=2)
abc = AdaBoostClassifier(n_estimators=50, random_state=2)
bc = BaggingClassifier(n_estimators=50, random_state=2)
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)
gbdt = GradientBoostingClassifier(n_estimators=50,random_state=2)
xgb = XGBClassifier(n_estimators=50,random_state=2)
```

```
clfs = {
    'SVC' : svc,
    'KN' : knc,
    'NB': mnb,
    'DT': dtc,
    'LR': lrc,
    'RF': rfc,
    'AdaBoost': abc,
    'BgC': bc,
    'ETC': etc,
    'GBDT':gbdt,
    'xgb':xgb
}
def train_classifier(clf,X_train,y_train,X_test,y_test):
    clf.fit(X_train,y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test,y_pred)
    precision = precision_score(y_test,y_pred)

    return accuracy,precision
accuracy_scores = []
precision_scores = []

for name,clf in clfs.items():
```

```
    current_accuracy,current_precision = train_classifier(clf,
X_train,y_train,X_test,y_test)


    print("For ",name)
    print("Accuracy - ",current_accuracy)
    print("Precision - ",current_precision)


    accuracy_scores.append(current_accuracy)
    precision_scores.append(current_precision)
performance_3000_df =
pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_max_features_3000':accuracy_scores,
'Precision_max_features_3000':precision_scores}).sort_values('Precision_max_features
_3000',ascending=False)
performance_3000_df
final_df=final_df.merge(performance_3000_df)
final_df
final_df.T
#Custom Ensemble Model
from sklearn.ensemble import StackingClassifier
kn = KNeighborsClassifier()
rfc = RandomForestClassifier(n_estimators=50, random_state=2,)


estimators=[('kn', kn), ('rfc', rfc)]
final_estimator=RandomForestClassifier()
```

```
clf = StackingClassifier(estimators=estimators, final_estimator=final_estimator)
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print("Accuracy",accuracy_score(y_test,y_pred))
print("Precision",precision_score(y_test,y_pred))
kn = KNeighborsClassifier()
mnb = MultinomialNB()
rfc = RandomForestClassifier(n_estimators=50, random_state=2,)


estimators=[('kn', kn), ('nb', mnb), ('rfc', rfc)]
final_estimator=RandomForestClassifier()
clf = StackingClassifier(estimators=estimators, final_estimator=final_estimator)
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print("Accuracy",accuracy_score(y_test,y_pred))
print("Precision",precision_score(y_test,y_pred))
import pickle
pickle.dump(tfidf,open('vectorizer.pkl','wb'))
pickle.dump(clf,open('model.pkl','wb'))
```

2. Code for Image Steganography tool:

```
from tkinter import *
from tkinter import filedialog
import tkinter as tk
```

```
from PIL import Image as Pil_Image, ImageTk as Pil_ImageTk
import os
from stegano import lsb
import detector

root=Tk()
root.title("Image steganography for hidden sms communication with spam detection")
root.geometry('700x500')
root.resizable(False,False)

def showimage():
    global filename
    filename=filedialog.askopenfilename(initialdir=os.getcwd(),title='Select Image
File',filetype=(("PNG file","*.png"),("JPG file","*.jpg"),("All file","*.txt")))
    img=Pil_Image.open(filename)
    img=Pil_ImageTk.PhotoImage(img)
    imgsize="Image size: %dx%d" % (img.width(), img.height())
    sizelbl=Label(root,text=imgsize,font="arial 14 bold")
    sizelbl.place(x=100,y=300)
    lbl.configure(image=img,width=250,height=250)
    lbl.image=img

def Hide():
    global secret
```

```
    message=text1.get(1.0,END)
    secret=lsb.hide(str(filename),message)


def Show():
    try:
        clear_message=lsb.reveal(filename)
        text1.delete(1.0,END)
        text1.insert(END,clear_message)
        answer=detector.predict(clear_message)
        ailabel=Label(root,text=answer,font="arial 14 bold")
        ailabel.place(x=400,y=300)
    except:
        ailabel=Label(root,text="Error: no message found",font="arial 14 bold")
        ailabel.place(x=400,y=300)


def saveimage():
    secret.save("hidden.png")


f1=Frame(root,bd=3,bg="black",width=340,height=280,relief=GROOVE)
f1.place(x=10,y=10)


lbl=Label(f1,bg="black")
lbl.place(x=40,y=10)
```

```
f2=Frame(root,bd=3,bg="white",width=340,height=280,relief=GROOVE)

f2.place(x=350,y=10)


text1=Text(f2,font="Robote
20",bg="white",fg="black",relief=GROOVE,wrap=WORD)

text1.place(x=0,y=0,width=320,height=295)

scroll=Scrollbar(f2)

scroll.place(x=320,y=0,height=300)

scroll.configure(command=text1.yview)

text1.configure(yscrollcommand=scroll.set)

f3=Frame(root,bd=3,width=330,height=100,relief=GROOVE)

f3.place(x=10,y=370)

Button(f3,text="Open",width=10,height=2,font="arial 14
bold",command=showimage).place(x=20,y=30)

Button(f3,text="Save",width=10,height=2,font="arial 14
bold",command=saveimage).place(x=180,y=30)

Label(f3,text="Picture").place(x=20,y=5)

f4=Frame(root,bd=3,width=330,height=100,relief=GROOVE)

f4.place(x=360,y=370)


Button(f4,text="Hide",width=10,height=2,font="arial 14
bold",command=Hide).place(x=20,y=30)

Button(f4,text="Show",width=10,height=2,font="arial 14
bold",command=Show).place(x=180,y=30)

Label(f4,text="Options").place(x=20,y=5)
```

APPENDIX B

SCREENSHOTS

Information about the dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5559 entries, 0 to 5558
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   type    5559 non-null   object
 1   text    5559 non-null   object
dtypes: object(2)
memory usage: 87.0+ KB
```

Head of dataset

	type	text
344	ham	Yo, call me when you get the chance, a friend ...
5057	ham	Just wondering, the others just took off
3970	ham	Gosh that , what a pain. Spose I better come t...
2571	ham	But if she.s drinkin i'm ok.
2584	ham	Whos this am in class:-)

Head of dataset after LabelEncoder

	type	text
0	0	Hope you are having a good week. Just checking in
1	0	K..give back my thanks.
2	0	Am also doing in cbe only. But have to pay.
3	1	complimentary 4 STAR Ibiza Holiday or £10,000 ...
4	1	okmail: Dear Dave this is your final notice to...

Result of GNB model

```
gnb.fit(X_train,y_train)
y_pred1 = gnb.predict(X_test)
print(accuracy_score(y_test,y_pred1))
print(confusion_matrix(y_test,y_pred1))
print(precision_score(y_test,y_pred1))
```

```
0.8507751937984496
[[762 134]
 [ 20 116]]
0.464
```

Result of MNB model

```
mnb.fit(X_train,y_train)
y_pred2 = mnb.predict(X_test)
print(accuracy_score(y_test,y_pred2))
print(confusion_matrix(y_test,y_pred2))
print(precision_score(y_test,y_pred2))
```

```
0.9563953488372093
[[896   0]
 [ 45  91]]
1.0
```

Result of BNB model

```
bnb.fit(X_train,y_train)
y_pred3 = bnb.predict(X_test)
print(accuracy_score(y_test,y_pred3))
print(confusion_matrix(y_test,y_pred3))
print(precision_score(y_test,y_pred3))
```

```
0.9670542635658915
[[894   2]
 [ 32 104]]
0.9811320754716981
```

Accuracy and Precision of the models before optimization

	Algorithm	Accuracy	Precision
1	KN	0.896318	1.000000
2	NB	0.956395	1.000000
5	RF	0.962209	1.000000
8	ETC	0.968992	1.000000
0	SVC	0.974806	0.982456
6	AdaBoost	0.968023	0.963964
4	LR	0.947674	0.955556
10	xgb	0.967054	0.939655
9	GBDT	0.946705	0.909091
7	BgC	0.953488	0.866667
3	DT	0.929264	0.831579

Accuracy and Precision with max_features set as 1000

	Algorithm	Accuracy_max_features_1000	Precision_max_features_1000
5	RF	0.972868	0.990909
0	SVC	0.974806	0.982456
1	KN	0.913760	0.979592
9	GBDT	0.952519	0.967742
2	NB	0.975775	0.966387
10	xgb	0.968992	0.964286
8	ETC	0.974806	0.958333
4	LR	0.956395	0.950495
6	AdaBoost	0.959302	0.927273
7	BgC	0.946705	0.813953
3	DT	0.927326	0.785047

Accuracy and Precision with max_features set at 2000

	Algorithm	Accuracy_max_features_2000	Precision_max_features_2000
1	KN	0.911822	1.000000
5	RF	0.971899	1.000000
8	ETC	0.974806	0.991071
9	GBDT	0.952519	0.988764
2	NB	0.974806	0.982456
0	SVC	0.975775	0.974359
10	xgb	0.967054	0.972222
4	LR	0.955426	0.959184
6	AdaBoost	0.963178	0.937500
7	BgC	0.955426	0.868852
3	DT	0.931202	0.792793

Accuracy and Precision with max_features set at 3000

	Algorithm	Accuracy_max_features_3000	Precision_max_features_3000
1	KN	0.906008	1.000000
5	RF	0.968992	1.000000
9	GBDT	0.954457	0.989011
0	SVC	0.978682	0.983051
2	NB	0.973837	0.982301
8	ETC	0.973837	0.982301
4	LR	0.950581	0.956989
10	xgb	0.967054	0.947368
6	AdaBoost	0.963178	0.929825
7	BgC	0.952519	0.859504
3	DT	0.929264	0.818182

Accuracy and Precision comparison after optimizations

	Algorithm	Accuracy	Precision	Accuracy_max_features_1000	Precision_max_features_1000	Ac
0	KN	0.896318	1.000000	0.913760	0.979592	
1	NB	0.956395	1.000000	0.975775	0.966387	
2	RF	0.962209	1.000000	0.972868	0.990909	
3	ETC	0.968992	1.000000	0.974806	0.958333	
4	SVC	0.974806	0.982456	0.974806	0.982456	
5	AdaBoost	0.968023	0.963964	0.959302	0.927273	
6	LR	0.947674	0.955556	0.956395	0.950495	
7	xgb	0.967054	0.939655	0.968992	0.964286	
8	GBDT	0.946705	0.909091	0.952519	0.967742	
9	BgC	0.953488	0.866667	0.946705	0.813953	
10	DT	0.929264	0.831579	0.927326	0.785047	

Accuracy and Precision comparison after optimizations contd.

Accuracy_max_features_2000	Precision_max_features_2000	Accuracy_max_features_3000	Precision
0.911822	1.000000	0.906008	
0.974806	0.982456	0.973837	
0.971899	1.000000	0.968992	
0.974806	0.991071	0.973837	
0.975775	0.974359	0.978682	
0.963178	0.937500	0.963178	
0.955426	0.959184	0.950581	
0.967054	0.972222	0.967054	
0.952519	0.988764	0.954457	
0.955426	0.868852	0.952519	
0.931202	0.792793	0.929264	

Accuracy and Precision comparison after optimizations contd.

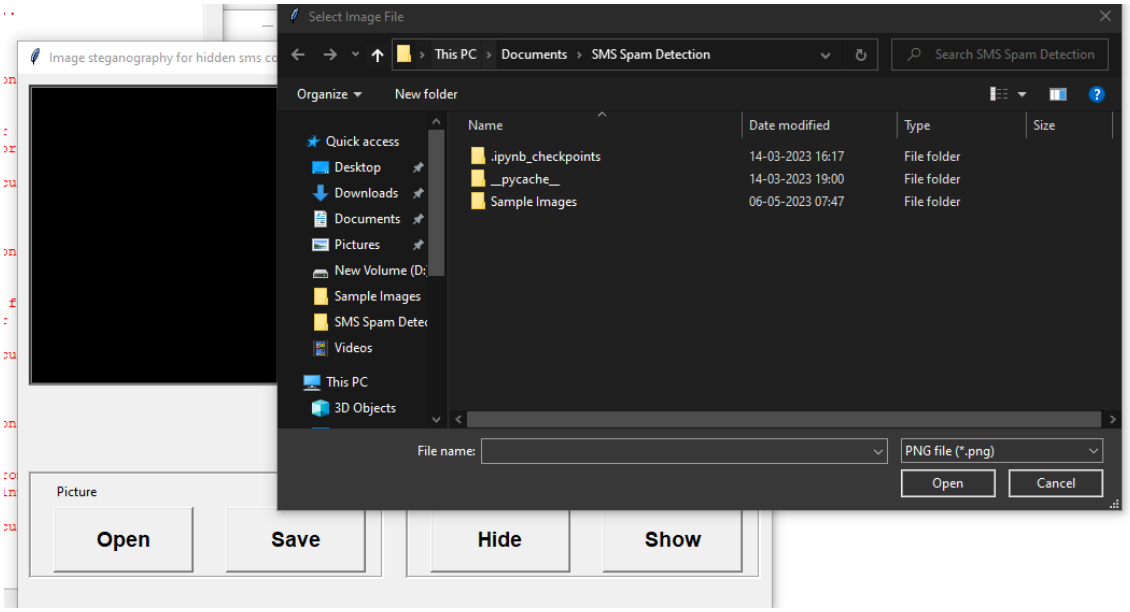
res_2000	Precision_max_features_2000	Accuracy_max_features_3000	Precision_max_features_3000
0.911822	1.000000	0.906008	1.000000
0.974806	0.982456	0.973837	0.982301
0.971899	1.000000	0.968992	1.000000
0.974806	0.991071	0.973837	0.982301
0.975775	0.974359	0.978682	0.983051
0.963178	0.937500	0.963178	0.929825
0.955426	0.959184	0.950581	0.956989
0.967054	0.972222	0.967054	0.947368
0.952519	0.988764	0.954457	0.989011
0.955426	0.868852	0.952519	0.859504
0.931202	0.792793	0.929264	0.818182

Accuracy and Precision of KNR model

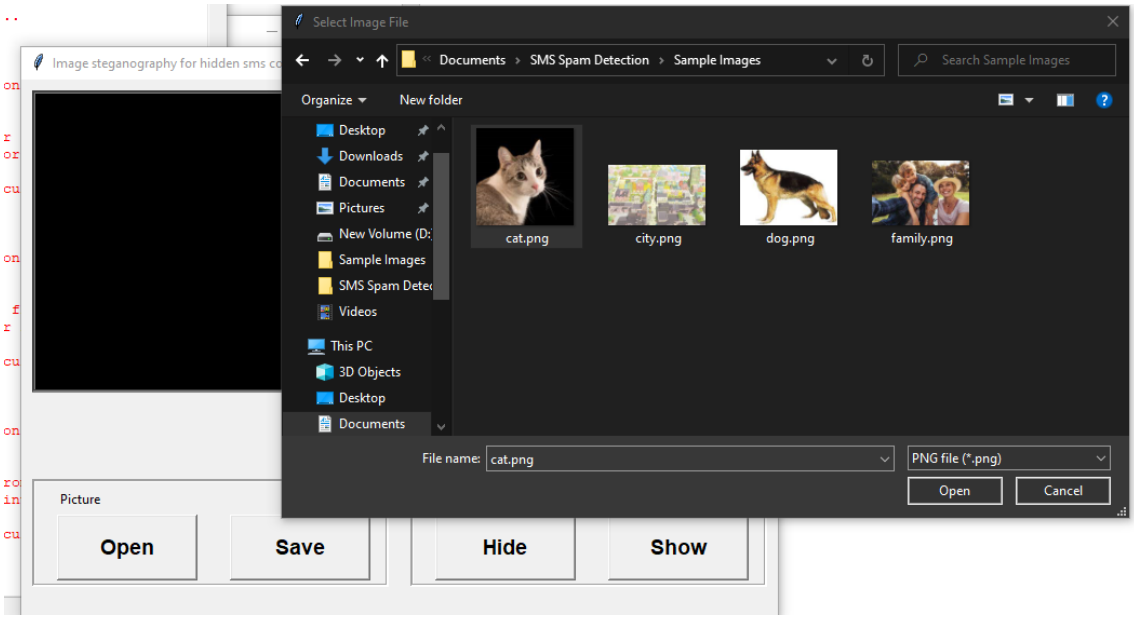
```
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print("Accuracy",accuracy_score(y_test,y_pred))
print("Precision",precision_score(y_test,y_pred))
```

```
Accuracy 0.9796511627906976
Precision 0.9457364341085271
```

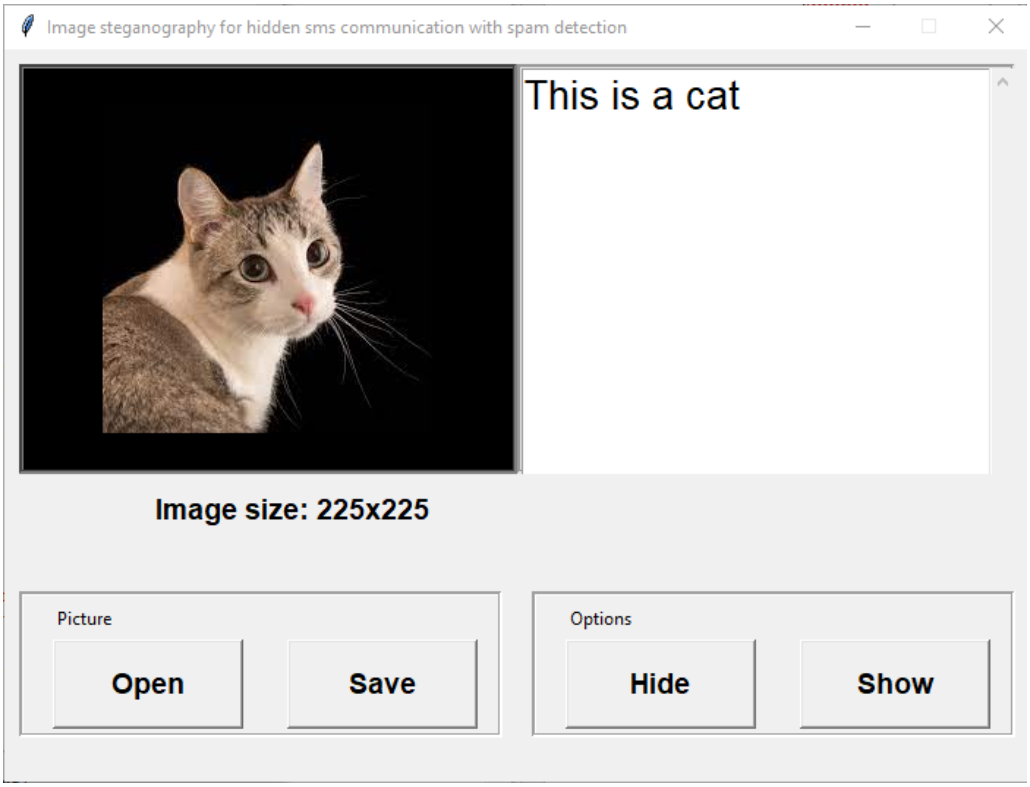
Opening and selecting an image in the tool



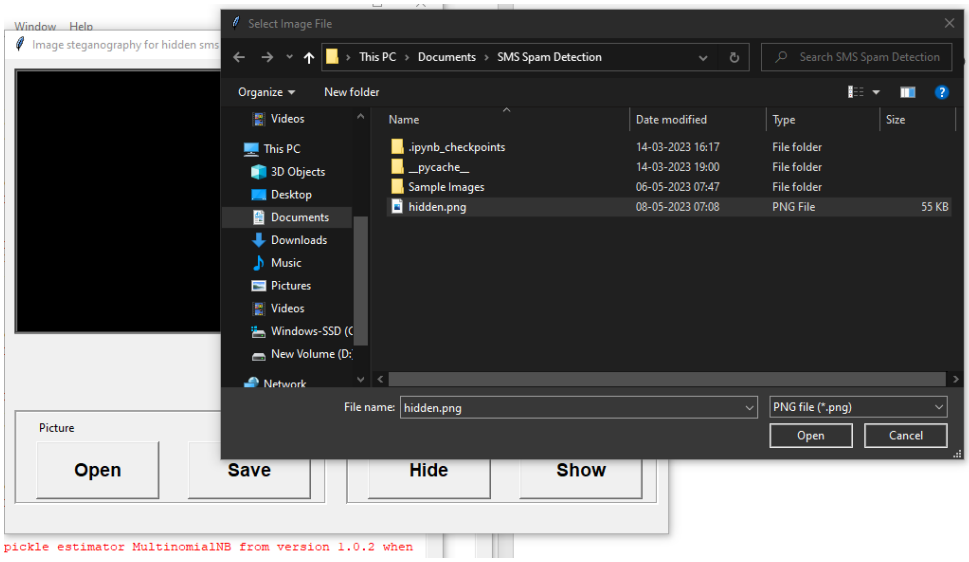
Opening and selecting an image in the tool contd.



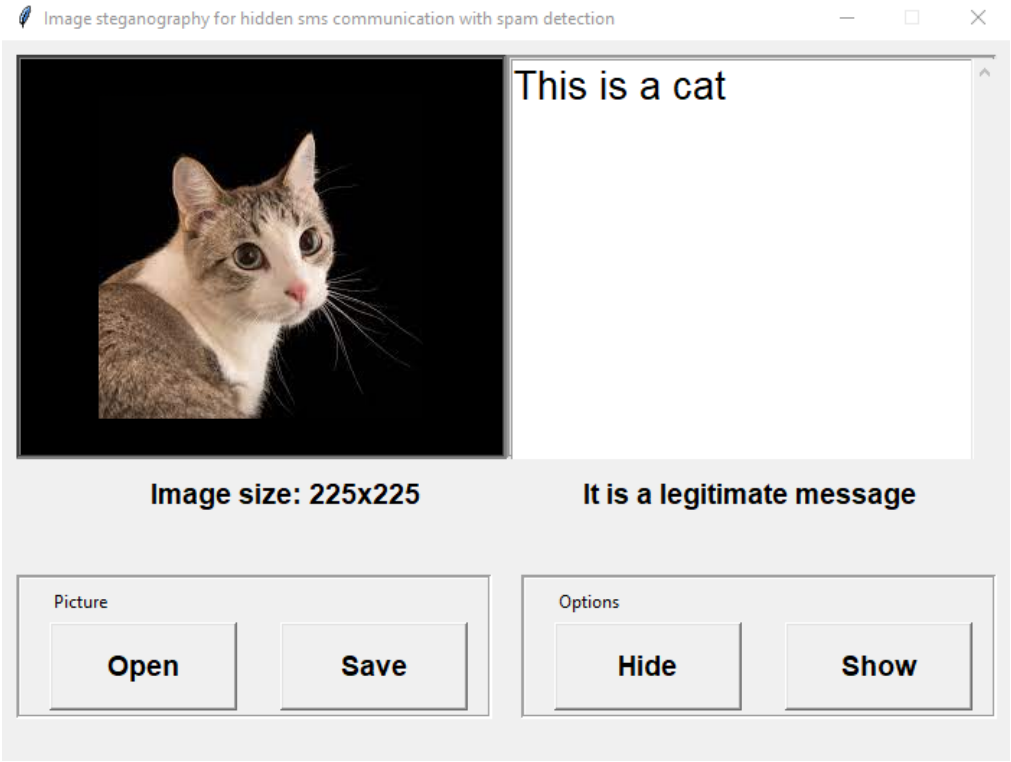
Opening and selecting an image in the tool contd.



Selecting the cover image to extract the hidden message



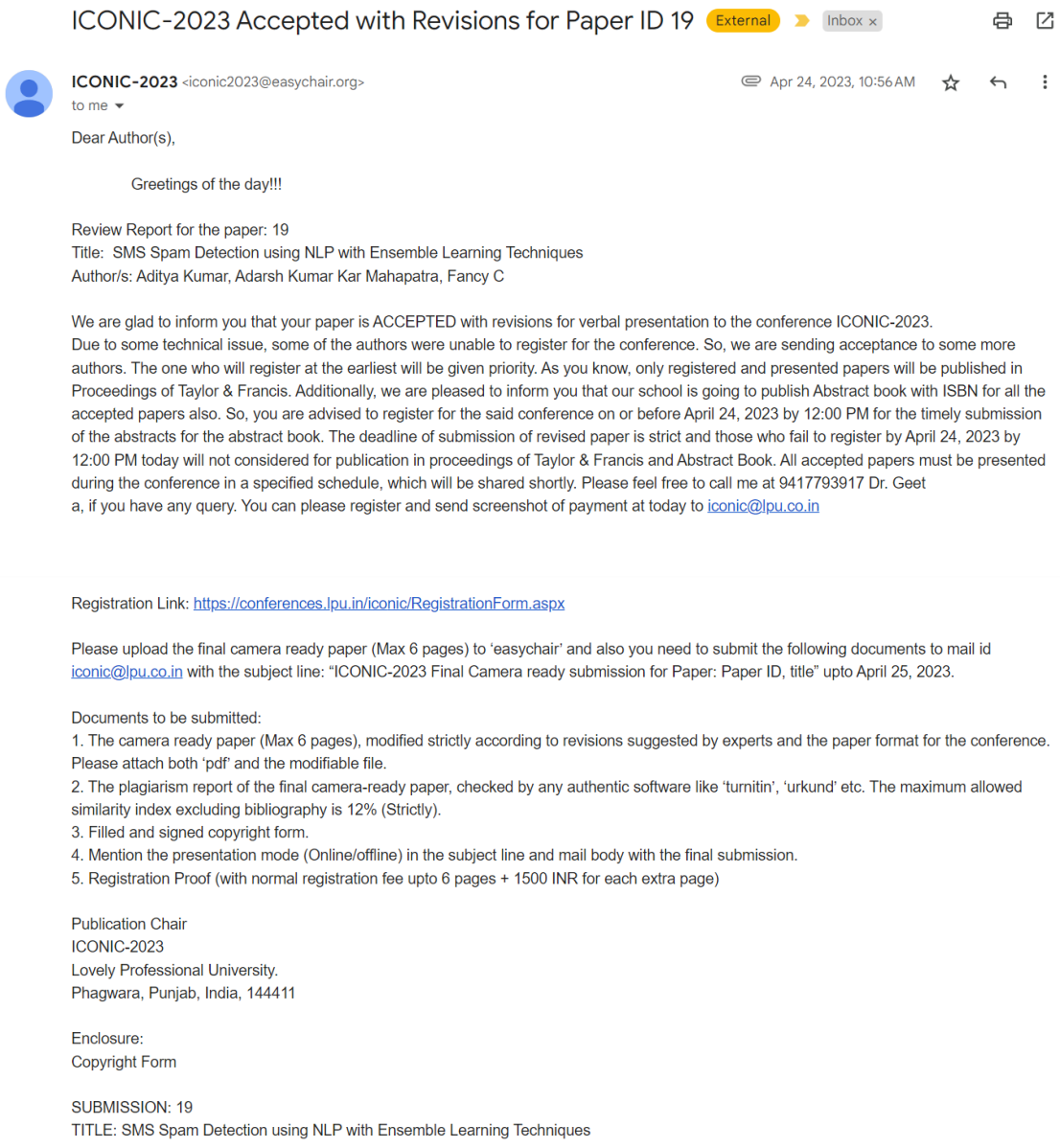
The prediction of the text by KNR model in the GUI



APPENDIX C

JOURNAL PUBLICATION

Our paper was accepted in ICONIC 2023: International Conference on Networks, Intelligence and Computing



ICONIC-2023 Accepted with Revisions for Paper ID 19 External Inbox ×

ICONIC-2023 <iconic2023@easychair.org> Apr 24, 2023, 10:56 AM
to me

Dear Author(s),

Greetings of the day!!!

Review Report for the paper: 19
Title: SMS Spam Detection using NLP with Ensemble Learning Techniques
Author/s: Aditya Kumar, Adarsh Kumar Kar Mahapatra, Fancy C

We are glad to inform you that your paper is ACCEPTED with revisions for verbal presentation to the conference ICONIC-2023.
Due to some technical issue, some of the authors were unable to register for the conference. So, we are sending acceptance to some more authors. The one who will register at the earliest will be given priority. As you know, only registered and presented papers will be published in Proceedings of Taylor & Francis. Additionally, we are pleased to inform you that our school is going to publish Abstract book with ISBN for all the accepted papers also. So, you are advised to register for the said conference on or before April 24, 2023 by 12:00 PM for the timely submission of the abstracts for the abstract book. The deadline of submission of revised paper is strict and those who fail to register by April 24, 2023 by 12:00 PM today will not considered for publication in proceedings of Taylor & Francis and Abstract Book. All accepted papers must be presented during the conference in a specified schedule, which will be shared shortly. Please feel free to call me at 9417793917 Dr. Geet
a, if you have any query. You can please register and send screenshot of payment at today to iconic@lpu.co.in

Registration Link: https://conferences.lpu.in/iconic/RegistrationForm.aspx

Please upload the final camera ready paper (Max 6 pages) to 'easychair' and also you need to submit the following documents to mail id iconic@lpu.co.in with the subject line: "ICONIC-2023 Final Camera ready submission for Paper: Paper ID, title" upto April 25, 2023.

Documents to be submitted:
1. The camera ready paper (Max 6 pages), modified strictly according to revisions suggested by experts and the paper format for the conference. Please attach both 'pdf' and the modifiable file.
2. The plagiarism report of the final camera-ready paper, checked by any authentic software like 'turnitin', 'urkund' etc. The maximum allowed similarity index excluding bibliography is 12% (Strictly).
3. Filled and signed copyright form.
4. Mention the presentation mode (Online/offline) in the subject line and mail body with the final submission.
5. Registration Proof (with normal registration fee upto 6 pages + 1500 INR for each extra page)

Publication Chair
ICONIC-2023
Lovely Professional University.
Phagwara, Punjab, India, 144411

Enclosure:
Copyright Form

SUBMISSION: 19
TITLE: SMS Spam Detection using NLP with Ensemble Learning Techniques

APPENDIX D

PLAGARISM REPORT

SRM

Office of Controller of Examinations

REPORT FOR PLAGARISM CHECK ON THE SYNOPSIS/THESIS/DISSERTATION/PROJECT REPORTS

1	Name of the Candidate (**IN BLOCK LETTERS**)	ADITYA KUMAR ADARSH MAHAPATRA
2	Address of the Candidate	EMERALD 601 ROYAL GEMS, CHALA, VAPI, GUJARAT (396191) **Mobile Number**: 9979425368 Flat 202, D-663-64, VISHWAKARMA COLONY, M.B. ROAD, NEW DELHI (110044) **Mobile Number**:9953734113
3	Registration Number	RA1911030010100 RA1911030010071
4	Date of Birth	24TH DECEMBER 2001 25th JULY 2001
5	Department	DEPARTMENT OF NEWTORKING AND COMMUNICATIONS
6	Faculty	ENGINEERING AND TECHNOLOGY
7	Title of the Synopsis/ Thesis/ Dissertation/ Project	SMS SPAM DETECTION USING NATURAL LANGUAGE PROCESING AND ENSEMBLE LEARNING
8	Name and address of the Supervisor/Guide	Dr. C. FANCY SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, KATTANKULATHUR – 603 203 **Mail ID:** fancyc@srmist.edu.in **Mobile Number**: 9487175375
9	Name and address of the Co-Supervisor/Co-Guide (in any)	- Mail ID: Mobile Number:
10	Software Used	TURNITIN

11	Date of Verification	12/5/23		
12	**Plagiarism Details: (to attach the final report)**			
Chapter	**Title of the Chapter**	**Percentage of similarity index (including self citation)**	**Percentage of similarity index (excluding self citation)**	**% of plagiarism after excluding Quotes, Bibliography, etc.,**
1	Introduction	3%	0%	0%
2	Literature Survey	3%	0%	0%
3	System Requirements	2%	0%	0%
4	Proposed Methodology	0%	0%	0%
5	Results and Discussion	0%	0%	0%
6	Conclusion and Future Enhancements	0%	0%	0%
Thesis abstract		0%	0%	0%
Appendices		0%	0%	0%

We declare that the above information have been verified and found true to the best of our knowledge

Signature of the Candidate	**Name and Signature of the Staff (Who uses the plagiarism check software)**
Aditya Kumar **Adarsh Mahapatra**	15/5/23 **Dr. C. Fancy**
Name and Signature of the Supervisor/ Guide	**Name and Signature of the Co-Supervisor/ Co-Guide**
15/5/23 **Dr. C. Fancy**	—

Dr Annapurani K.

Name and Signature of the HOD

SMS V4

ORIGINALITY REPORT

8%	5%	2%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	"Intelligent Technologies and Applications", Springer Science and Business Media LLC, 2019 Publication	1%
2	Submitted to Indian School of Business Student Paper	1%
3	assets.researchsquare.com Internet Source	<1%
4	Submitted to Universiti Kebangsaan Malaysia Student Paper	<1%
5	aircconline.com Internet Source	<1%
6	www.abacademies.org Internet Source	<1%
7	Submitted to University of Exeter Student Paper	<1%

8	dspace.adu.ac.ae Internet Source	<1 %
9	Submitted to Universiti Kebangsaan Malaysia Student Paper	<1 %
10	"Intelligent Technologies and Applications", Springer Science and Business Media LLC, 2019 Publication	<1 %
11	aircconline.com Internet Source	<1 %
12	www.abacademies.org Internet Source	<1 %
13	www.creativetech24.com Internet Source	<1 %
14	Submitted to Queen Mary and Westfield College Student Paper	<1 %
15	archive.org Internet Source	<1 %
16	www.medikom.iocspublisher.org Internet Source	<1 %

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		