BANGALORE UNIVERSITY

UNIVERSITY VISVESVARAYA COLLEGE OF ENGINEERING

K.R Circle, Bengaluru-560001

Department of Computer Science and Engineering

Final Year Project Report

on

"SMS Spam Detection Using Machine Learning"

Submitted by

R Srujan Kumar (19GANSE032)	**Sachin A G (19GANSE040)**
Vibha Rao H R (19GANSE055)	**Yathish G (19GANSE058)**

Under the Guidance of

Dr. Sunil Kumar G

Assistant Professor

Department of CSE, UVCE

Bangalore University

Year 2022-23

University Visvesvaraya College of Engineering

Department of Computer Science and Engineering

K.R Circle, Bengaluru-560001

CERTIFICATE

This is to certify that **R Srujan Kumar, Sachin A G, Vibha Rao H R, Yathish G** of **VIII Semester, B.Tech** (Information Science and Engineering), bearing the register numbers **19GANSE032, 19GANSE040, 19GANSE055, 19GANSE058** respectively, have successfully completed the Project work on **"SMS Spam Detection Using Machine Learning"** in fulfilment of the requirement of Project work of Bachelor of Technology in Information Science and Engineering prescribed by the Bangalore University during the academic year 2022-23.

--------------------------	--------------------------
Dr. Sunil Kumar G	**Dr. H.S. Vimala**
Assistant Professor	Professor & Chairperson
Dept. of CSE, UVCE	Dept. of CSE, UVCE
Bengaluru	Bengaluru

Examiner 1 ………………. **Examiner 2………………**

ACKNOWLEDGMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without mentioning the people who made it possible, whose constant guidance and encouragement crowned our effort with success.

First and foremost, we would like to express our sincere words of gratitude and respect to the organization University Visvesvaraya College of Engineering, Bangalore, for providing us an opportunity to carry out our project work.

We thank **Dr. H N Ramesh**, Principal, University Visvesvaraya College of Engineering, Bangalore, for providing us with all the facilities that helped us to carry out the work easily.

We wish to place our grateful thanks to **Dr. Vimala H S**, Professor and Chairperson, Department of Computer Science and Engineering, UVCE, who helped us to make our project a great success.

We would like to express our deepest thanks to **Dr. Sunil Kumar G**, Assistant Professor of Department of Computer Science and Engineering, for his constant guidance and support as an internal guide with continuous guidance and encouragement to complete the project successfully on time.

Lastly, we extend our thanks to all the teaching and non-teaching fraternity in the Department of Computer Science and Engineering, for always being helpful over the years. We are very grateful to our parents, friends and well-wisher for their continuous moral support and encouragement.

R Srujan Kumar	**(19GANSE032)**
Sachin A G	**(19GANSE040)**
Vibha Rao H R	**(19GANSE055)**
Yathish G	**(19GANSE058)**

ABSTRACT

Spam is a major problem in the world of SMS messaging. It can be annoying, disruptive, and even dangerous. Machine learning can be used to detect spam messages with a high degree of accuracy. This project will use a variety of machine learning techniques to build a spam detection model. The model will be trained on a dataset of SMS messages that have been labelled as spam or ham. Once the model is trained, it will be able to predict whether a new SMS message is spam or ham with a high degree of accuracy.

The project will be implemented using the Python programming language. The data will be pre-processed using a variety of techniques, including **stemming, stop word removal, and TF-IDF vectorization**. The machine learning models will be built using the scikit-learn library. The performance of the models will be evaluated using a variety of **metrics, including accuracy, precision, and recall.** We used various Machine Learning techniques like **KNN, Naive Bayes, SVM, Random Forest, Logistic Regression, Decision Tree Algorithms**. The benefits of using machine learning for SMS spam detection, such as the ability to learn from new data and the ability to adapt to changes in spam patterns. The potential applications of the spam detection model, such as filtering spam messages from SMS inbox.

CONTENTS

LIST OF FIGURES AND SNAPSHOTS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Short Message Services (SMS) is far more than just a technology for a chat. SMS technology evolved out of the global system for mobile communications standard, an internationally accepted. Spam is the abuse of electronic messaging systems to send unsolicited messages in bulk indiscriminately. While the most widely recognized form of spam is email spam, the term is applied to similar abuses in other media and mediums. SMS Spam in the context is very similar to email spams, typically, unsolicited bulk messaging with some business interest.

SMS spam is used for commercial advertising and spreading phishing links. Commercial spammers use malware to send SMS spam because sending SMS spam is illegal in most countries. Sending spam from a compromised machine reduces the risk to the spammer because it obscures the provenance of the spam. SMS can have a limited number of characters, which includes alphabets, numbers, and a few symbols. A look through the messages shows a clear pattern. Almost all of the spam messages ask the users to call a number, reply by SMS, or visit some URL. This pattern is observable by the results obtained by a simple SQL query on the spam.

Fig. 1.1: Spam Introduction

Every time SMS spam arrives at a user's inbox, and the mobile phone alerts the user to the incoming message. When the user realizes that the message is unwanted, he or she will be disappointed, and also SMS spam takes up some of the mobile phone's storage.

SMS spam detection is an important task where spam SMS messages are identified and filtered. As more significant numbers of SMS messages are communicated every day, it is challenging for a user to remember and correlate the newer SMS messages received in context to previously received SMS. Thus, using the knowledge of artificial intelligence with the amalgamation of machine learning, and data mining we will try to develop web-based SMS text spam or ham detector.

A number of major differences exist between spam-filtering in text messages and emails. Unlike emails, which have a variety of large datasets available, real databases for SMS spams are very limited. Additionally, due to the small length of text messages, the number of features that can be used for their classification is far smaller than the corresponding number in emails. Here, no header exists as well. Additionally, text messages are full of abbreviations and have much less formal language that what one would expect from emails. All of these factors may result in serious degradation in performance of major email spam filtering algorithms applied to short text messages.

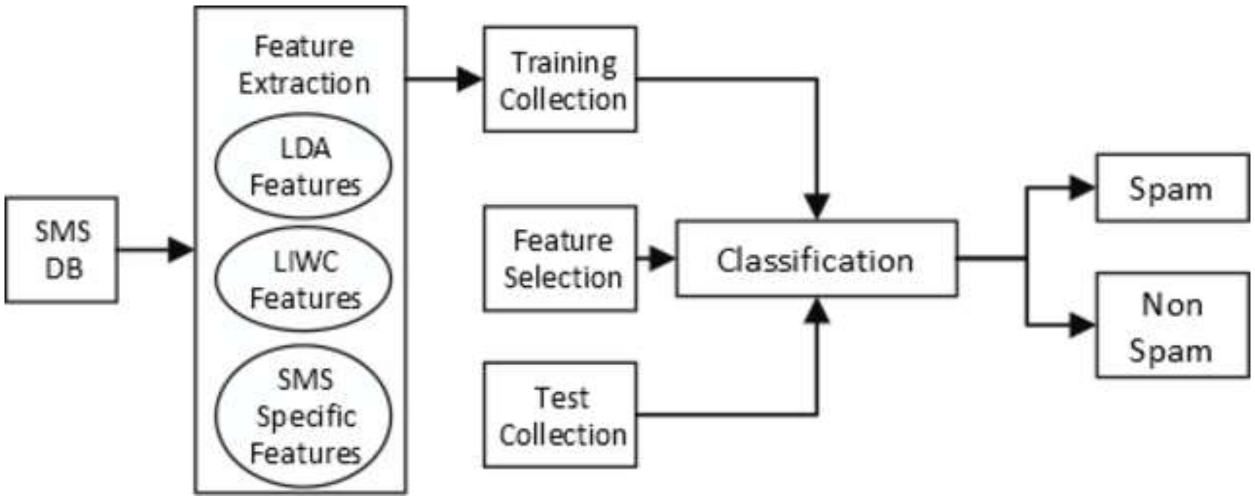


Fig. 1.2: Spam Methodology

1.2 PURPOSE

SMS spam detection is an important tool for mobile operators, businesses, and individuals who want to protect themselves from spam. It can help to reduce the amount of spam that users receive, which can improve their overall experience with mobile messaging. Reduces the amount of spam that users receive: This can improve the user experience and free up user's time. Protects users from malware and phishing attacks: Spam messages can be used to spread malware or to phish for personal information. SMS spam detection can help to protect users from these threats. Improves the security of mobile networks: Spam messages can overload mobile networks and disrupt service. SMS spam detection can help to reduce the load on mobile networks and improve their overall performance.

And this is used to develop a system that can identify spam messages and prevent them from reaching users' inboxes. Spam messages can be a nuisance and a security risk, so it is important to have a system in place to protect users from them. Overall, spam detection using ML offers an effective, efficient, and adaptable solution to combat the ever-evolving problem of spam, enabling individuals and organizations to maintain better control over their digital communication channels and focus on relevant and important messages.

1.3 OBJECTIVES

- The aim of this project is to explore different methods of text classification into spam or ham and comparing results based on accuracy and precision. And this model uses both machine learning and deep learning.
- Spam classification has become more challenging due to complexities of the messages imposed by spammers. Hence, various methods have been developed in order to filter spams.
- We will explore different methods of text classification into spam or ham and comparing results based on accuracy and precision. And will choose the best technique to classify the ham and spam messages.
- Many machine learning methods have been applied for Short Messaging Service (SMS) Spam Detection, including traditional classification methods such as Naive Bayes (NB), K-Nearest Neighbours (KNN), Logistic Regression (LR) and Support Vector Machine (SVM).
- We will use our trained ML model to analyse the sent SMS, and use the above-mentioned classification techniques to detect the Spam SMS.

1.4 CHALLENGES

In Spam modelling system user’s trust tends to vary over time according to the user’s experience and involvement of social networks. Only a few approaches deals with the dynamics of trust by distinguishing between recent and old tags. Future work considering dynamics of trust would lead to better modelling in real world application.

Most of the existing approaches based on text information assuming monolingual environment, so there may be problem of low accuracy and system cannot recognize the spam SMS efficiently. So to improve the accuracy of system by using various ML algorithms and compare them. SMS spam detection is an important task where spam SMS messages are identified and filtered. As more significant numbers of SMS messages are communicated every day, it is challenging for a user to remember and correlate the newer SMS messages received in context to previously received SMS. This model will be trained on a dataset of SMS messages that have been labelled as spam or ham. Once the model is trained, it will be able to predict whether a new SMS message is spam or ham with a high degree of accuracy.

1.5 SCOPE

The objective of this project is to develop an accurate and efficient SMS spam detection system. The proliferation of mobile devices and the increasing usage of SMS for communication have led to a rise in unsolicited and unwanted SMS spam messages. These spam messages not only waste users' time but can also be used for fraudulent activities, phishing attempts, or spreading malware. Therefore, there is a need for a robust spam detection system that can identify and filter out such spam messages, ensuring a better user experience and protecting users from potential threats.

SMS messages are typically short and lack extensive contextual information, making it challenging to accurately identify spam based on content alone. The system needs to be designed to extract relevant features from these limited messages to differentiate between spam and legitimate messages effectively.

SMS messages often contain non-standard language, abbreviations, and variations in spelling, posing a challenge for traditional language processing techniques. The system should be able to handle these linguistic variations and adapt to the unique characteristics of SMS language to accurately classify spam messages. Spammers continually adapt their tactics to evade detection systems, requiring the SMS spam detection system to stay updated and capable of identifying new spam patterns. The system should be designed to be flexible and easily updatable, enabling it to detect and adapt to emerging spam techniques effectively.

CHAPTER 2
LITERATURE SURVEY

SMS Spam is a serious problem and thus many researchers have keen motivation to solve the problem using different approaches and methods. This section investigates the problem of spam and various other text mining problem and how it is tackled by implementing different machine learning algorithms and techniques by reviewing the literature from the year 2020 to date. The survey is divided into the following subsections:

- A Critique of Spam Short Message Service Detection.
- A Critical Review of Algorithms and Techniques Used in Spam Classification.
- A Critique on Data-set and Features used in Short Message Service Spam Detection.
- A Comparison of the Reviewed techniques.

Fernandes et al., [1] have proposed a system of Spam Short Message Service Detection. Over the past few years, mobile phones have encountered an enormous amount of growth and therefore SMS has become the common platform for the exchange of information. It was reported that there are around 5.5 billion active mobile phone users in the year 2019. The researchers also mention that SMS had gained popularity among the youth and as the SMS charges reduced over the years, around 30% of total messages were found to be spam in Asia in the year 2019. The number of spam SMS detection software available is very limited, which makes spam SMS detection an interesting problem that can be solved by using machine learning techniques.

As per the researcher, the short message service was started in the year 1992 and has gained substantial popularity with revenue close to 213 billion dollars in the year 2019. A humongous number of SMSs are exchanged all over the world on a daily basis. Several misuses of SMSs have been highlighted leading to loss of personal and financial information by unknowingly clicking on bogus links. The researcher (Sethi et al., (2020)) revealed a very appalling fact that the amount of spam SMS circulated the world over surpass the number of spam emails. The problem is so serious that some countries like Japan took legal action against it. One of the main reasons for spam SMS is because the cost of sending an SMS is so low that it becomes irrelevant compared to the benefit hackers can reap if they get hold of the sensitive information.

Researcher (Agarwal et al., (2021)) mentions that spam is the junk or unwanted messages which are broadcasted to a group of users with bad intention like stealing personal information, etc. The spam messages are growing at a rapid rate with almost 500% increase every subsequent year. The researcher also mentions that the quantum of spam SMS is not the same across the regions. For instance, in North America only one percent of total SMS were spam in the year 2017, however, more than thirty percent were spam in some parts of Asia. In the year 2014, the spam messages received by the people in China in a week were around 200 billion. Looking at the above-stated facts mentioned in research papers regarding the growth of spam SMS, it is high time to solve the problem using the latest techniques which are efficient and can be implemented in real-time.

Kim et al., [2] have development a system for an analysis of spam SMS filtering was done on the UCI machine learning dataset in the year 2020, who chose the frequency ratio feature selection technique while implementing the algorithms like Naive Bayes, Logistic Regression and Decision Trees where the 10-fold cross-validation technique was applied. It was seen that Naive Bayes generated results in a minimum time with the highest accuracy of 94 per- cent.

In the year 2021, a similar analysis was conducted by (Gupta et al., (2021)) using 2 sets of data, one with UCI machine learning which is the same corpus as of Kaggle with a total of 5574 ham and spam messages and another dataset contains 2000 spam and ham messages. TF-IDF matrix was created and then machine learning algorithms like Naive Bayes, Random Forest, SVM, Decision Tree, Convolutional Neural Network and Artificial Neural Network were applied on both the datasets. The results obtained by CNN were the state-of-art in this area with an accuracy of 99.10 followed by Naive Bayes and SVM. It was reported that there are around 5.5 billion active mobile phone users in the year 2019.

The researchers also mention that SMS had gained popularity among the youth and as the SMS charges reduced over the years, around 30% of total messages were found to be spam in Asia in the year 2019. The number of spam SMS detection software available is very limited, which makes spam SMS detection an interesting problem that can be solved by using machine learning techniques. As per the researcher, the short message service was started in the year 1992 and has gained substantial popularity with revenue in the year 2019. A humongous number of SMSs are exchanged all over the world on a daily basis.

Several misuses of SMSs have been highlighted leading to loss of personal and financial information by unknowingly clicking on bogus links. The researcher (Sethi et al., (2020)) revealed a very appalling fact that the amount of spam SMS circulated the world over surpass the number of spam emails. The problem is so serious that some countries like Japan took legal action against it. One of the main reasons for spam SMS is because the cost of sending an SMS is so low that it becomes irrelevant compared to the benefit hackers can reap if they get hold of the sensitive information.

Researcher (Agarwal et al., (2021)) mentions that spam is the junk or unwanted messages which are broadcasted to a group of users with bad intention like stealing personal information, etc. The spam messages are growing at a rapid rate with almost 500% increase every subsequent year. The researcher also mentions that the quantum of spam SMS is not the same across the regions. For instance, in North America only one percent of total SMS were spam in the year 2017, however, more than thirty percent were spam in some parts of Asia. In the year 2014, the spam messages received by the people in China in a week were around 200 billion.

Looking at the above-stated facts mentioned in research papers regarding the growth of spam SMS, it is high time to solve the problem using the latest techniques which are efficient and can be implemented in real-time.

Ma et al., [3] in 2021 conducted Research on spam SMS detection proposes a message topic model which is a form of probability topic model. It uses the KNN algorithm to remove the sparsity problem in the messages. The symbol terms, background terms were considered and it was found that the model generated better results than the standard LDA. The classifier GentleBoost was used for the first time in the research done on SMS spam detection in the year 2019. For unbalanced data and binary classification, boosting algorithms work well. GentleBoost is a combination of two algorithms, namely AdaBoost and LogitBoost. GentleBoost is well known for its higher accuracy and less consumption of storage as it removes unwanted features.

It obtained an accuracy of 98% on the dataset consisting of 5572 text messages. The author states that the short length of the messages and the use of casual words in the text messages do not allow it to perform well with the already established solutions of email spam filtering. In this research, it can be seen that SVM followed by Multinomial Naive Bayes (MNB) shows outstanding results in terms of accuracy with 98.23 and 97.87% respectively. MNB took the least execution time of 2.03 seconds. The researcher further suggests that features like the number of characters in the messages or definite threshold to the length of the message can increase the performance.

Looking at the above research, it can be stated that the performance of traditional algorithms like Logistic Regression and SVM is superior to other algorithms. Also in this research, the length parameter can be taken as an additional feature to check if it enhances the performance of the model. The efficient spam detection is an important tool in order to help people to classify whether it is a spam SMS or not. In this research, we propose a novel SMS spam detection based on the case study of the SMS spams in English language using Natural Language Process and Deep Learning techniques. To prepare the data for our model development process, we use word tokenization, padding data, truncating data and word embedding to make more dimension in data.

Then, this data is used to develop the model based on Long Short-Term Memory and Gated Recurrent Unit algorithms. The performance of the proposed models is compared to the models based on machine learning algorithms including Support Vector Machine and Naïve Bayes. The experimental results show that the model built from the Long Short-Term Memory technique provides the best overall accuracy as high as 98.18%. On accurately screening spam messages, this model shows the ability that it can detect spam messages with the 90.96% accuracy rate, while the error percentage that it misclassifies a normal message as a spam message is only 0.74%.

Li et al**.,** provided the detailed investigation about other text classification methods. Phishing is a type of cybercrime where sensitive information such as user credentials, card details, etc. are stolen through counterfeit websites or emails. The paper by the researcher provided the solution by implementing Word2Vec as a feature extraction technique that gathers the feature from HTML code. Along with the feature extraction, a joint model using algorithms like LightGBM, XGBoost, and Gradient Boosting Trees are built which raised the performance to 97% accuracy.

The researcher (Liew et al., (2021)), detected phishing tweets on a real-time basis utilizing the feature from the phishing URL. Random Forest technique was implemented and found to produce an accuracy of 94.5%. Spams are defined as unsolicited bulk messages in various forms such as unwanted advertisements, credit opportunities or fake lottery winner notifications. Spam classification has become more challenging due to complexities of the messages imposed by spammers. Hence, various methods have been developed in order to filter spams.

In this study, methods of term frequency-inverse document frequency (TF-IDF) and Random Forest Algorithm will be applied on SMS spam message data collection. Based on the experiment, Random Forest algorithm outperforms other algorithms with an accuracy of 97.50%. They proposed the applications of the machine learning-based spam detection method for accurate detection. In this technique, machine learning classifiers such as Logistic regression (LR), K-nearest neighbour (K-NN), and decision tree (IDT) are used for classification of ham and spam messages in mobile device communication. The SMS spam collection data set is used for testing the method. The dataset is split into two categories for training and testing the research. The results of the experiments demonstrated that the classification performance of LR is high as compared with K-NN and DT, and the LR achieved a high accuracy of 99%. Additionally, the proposed method performance is good as compared with the existing state-of-the-art methods.

The researcher Koray et al., detects the phishing websites by analysing and extracting features from the URL. The Random Word Detection module is used which decomposes URL into small features which are then used to classify if the websites are legitimate or not. Seven different machine learning algorithms like Naive Bayes, KNN, SVM, Random Forest, Decision Tree, AdaBoost, K-star were implemented on a humongous amount of data. It was seen that Random Forest produced the highest accuracy of around 97.98 percent among all the techniques applied.

In the year 2021, the researcher Yuan et. al. proposed a blend of features pertaining to URL and web page for the detection of phishing websites. Along with the basic features like the length of the URL, unusual characters, etc, statistical features like mean, median and variance and lexical features like title and the content of the web page were also considered. Several algorithms like KNN, Logistic Regression, Random Forest, Deep Forest, XGBoost were applied. It was found that Deep Forest followed by XGBoost manifested high accuracy and less training time.

Nowadays, the success of any business heavily depends on authentic reviews. How- ever, not all reviews are authentic. The ratio of genuine reviews to sham reviews varies significantly from case to case and higher the bogus reviews, more is the image maligned of a business. Implementing sentiment analysis using natural language processing techniques can effectively detect the opinion spams. Earlier, researchers like Jindal and Liu (2017), Ren and Ji (2017) used supervised and semi-supervised algorithms for the detection of spam opinions.

These models have a few restrictions such as low flexibility, high computational time and poor accuracy. These limitations were overcome by the researcher Hasim et al. (2018) using the Gradient Boosting models like XGBoost, GBM Bernoulli, GBM AdaBoost and GBM Gaussian. The opinion spam detection was performed on multilingual datasets. It was found that XGBoost outperformed the other models for the English language dataset generating high recall percentage whereas GBM Gaussian produced good results for the Malay language dataset. The researcher Prieto et al., (2018) detects opinion spam using neural networks and it was observed that model complexity increases due to the large set of details provided to the neural network and thus increases the overall computational cost.

In the world of online advertising, fraud clicks are one of the most momentous issues. The research done by Minastireanu and Mesnita (2020) tackles the problem of fraud clicks by using the latest machine learning technique via. LightGBM on the dataset which contains millions of clicks. The K-Fold Cross-Validation technique is used as a feature engineering which helps in improving the performance. The accuracy achieved by the model was 98% and was found to be the fastest with respect to computational speed and low on memory consumption.

Looking at the above research papers, it was found that LightGBM and XGBoost are suitable as it performs faster with a less computational speed, unlike the deep learning techniques. Also, Random Forest performed well giving high accuracy and hence in this research, these algorithms are chosen.

Aich et al., proposed a technique for spam detection on imbalanced datasets of SMS and implemented the approach which generated great results in combination with SVM algorithm. The dataset contains 6599 messages marked as ham or spam. Out of these 1042 were spam and rest are legitimate. The length feature was added as the spam messages tend to be longer than ham. The data was pre-processed using NLP techniques and then Bag of Words (BoW) and Term Frequency Inverse Document Frequency (TF-IDF) were chosen as feature selection techniques. A content-based spam message filtering method proposed by Balli and Karasoy (2018) having a dataset of 5574 messages of which 747 are spam uses the semantic relationship between the SMS words.

The feature selection is done using the Word2Vec algorithm which calculates the distance between the vector of the words and thus features are extracted for each message. The work presented by the researcher for spam detection on imbalanced datasets of SMS, implemented an approach which generated great results in combination with the SVM algorithm and the accuracy increased by seven points. Lack of real databases for SMS spams, short length of messages and limited features, and their informal language are the factors that may cause the established email filtering algorithms to underperform in their classification.

In this project, a database of real SMS Spams from UCI Machine Learning repository is used, and after preprocessing and feature extraction, different machine learning techniques are applied to the database. Finally, the results are compared and the best algorithm for spam filtering for text messaging is introduced. Final simulation results using 10-fold cross validation shows the best classifier in this work reduces the overall error rate of best model in original paper citing this dataset by more than half.

The researcher investigates spam SMS detection by implementing 12 different types of classifiers on the dataset which contained 5574 messages. The research used the technique of down-sampling where the ham messages were reduced to the count of spam messages. The main contribution of this research is to examine the impact of class imbalance on performance and the researcher found that a balanced dataset produces more accurate results. The results are compared to the previous re- search done with the imbalanced dataset and found that performance degrades due to the under-fitting issue.

The datasets which have the issue of class imbalance can lead to bias results and poor performance whereas the balanced dataset can improve the performance. Also, feature selection techniques like Bag of Words and TF-IDF works well in the case of text classification.

A high-level comparison of the related work in text analytics classification problem based on feature used and classifiers applied. Classifiers like Naive Bayes, Random Forest, Logistic Regression, Decision Tree and SVM outperform other classifiers and hence these are chosen to be implemented in the research.

Table 1: Comparison of the Reviewed techniques

Area Classification	Applied Classifiers	Features extracted	Results	Author
Spam SMS Classification	Naive Bayes, Logistic regression	Message-Keyword Matrix	Naive Bayes With 94.7% accuracy	Kim et al. (2020)
Spam SMS Classification	Naive Bayes, SVM, Random Forests, Logistic Regression, Decision Tree	TF-IDF Vectorizer	SVM with 99.1% accuracy	Aich et al. (2021)

CHAPTER 3
REQUIREMENT SPECIFICATIONS

3.1 HARDWARE REQUIREMENTS

• System	: Intel i5 2.4 GHz.
• Hard Disk	: 40 GB.
• Floppy Drive	: 1.44 Mb.
• Monitor	: 15 VGA Colour.
• Mouse	: Logitech.
• Ram	: 4GB

3.2 SOFTWARE REQUIREMENTS

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a 28 basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the team ‘s progress throughout the development activity.

• Operating system	: Windows 10.
• Coding Language	: Python
• Python IDE	: Google colab, Jupyter notebook.

3.3 SOFTWARE ENVIRONMENT

What is IDE?

An IDE consists of an editor and a compiler that can be used to write and compile programs. It has a combination of features required for developing software. The presence of an IDE makes the development process and programming much easier. It interprets what is being typed and suggests the relevant keyword to insert. A class and a method can be distinguished as the IDE allocates different colours to them. The IDE also gives different colours for the right and the wrong keywords. If the wrong keyword is being typed, it tries to predict the keyword and auto-completes it.

Use of IDE:

The major reasons for using an IDE for development are given below:

- An IDE consists of a text editor window where programs can be written.
- It consists of a project editor window where all the necessary files of a software project are stored.
- Various inputs can be provided and efficiency of the program can be checked by inspecting the output received on the output window.
- If any error occurs, then the IDE shows warnings and suggestions on the output window so that errors can be resolved.
- An IDE has a rack of modules and packages in one place that helps add features in software applications.
- It helps increase efficiency in creating software.

Anaconda:

Anaconda, It is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Anaconda distribution comes with more than 1,500 packages as well as the Conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the Command Line Interface (CLI).

The big difference between Conda and the pip package manager is in how package dependencies are managed, which is a significant challenge for Python data science and the reason Conda exists. Pip installs all Python package dependencies required, whether those conflict with other packages you installed previously. So, your working installation of, for example, Google TensorFlow, can suddenly stop working when you pip install a different package that needs a different version of the Numpy library. More insidiously, everything might still appear to work but now you get different results from your data science, or you are unable to reproduce the same results elsewhere because you did not pip install in the same order. Conda analyzes your current environment, everything you have installed, any version limitations you specify (e.g., you only want TensorFlow>= 2.0) and figures out how to install compatible dependencies. Or it will tell you that what you want can't be done. Pip, by contrast, will just install the thing you wanted and any dependencies, even if that breaks other things.

Opensource packages can be individually installed from the Anaconda repository, Anaconda Cloud (anaconda.org), or your own private repository or mirror, using the conda install command. Anaconda Inc compiles and builds all the packages in the Anaconda repository itself, and provides binaries for Windows 32/64 bit, Linux 64 bit and MacOS 64-bit. You can also install anything on PyPI into a Conda environment using pip, and Conda knows what it has installed and what pip has installed. Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories. The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with conda.

Anaconda Navigator is a desktop Graphical User Interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux.

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- QtConsole
- Spyder
- Glueviz
- Orange
- Rstudio
- VS code.

Microsoft .NET is a set of Microsoft software technologies for rapidly building and integrating XML Web services, Microsoft Windows-based applications, and Web solutions. The .NET Framework is a language-neutral platform for writing programs that can easily and securely interoperate. There`s no language barrier with .NET: there are numerous languages available to the developer including Managed C++, C#, Visual Basic and Java Script. The .NET framework provides the foundation for components to interact seamlessly, whether locally or remotely on different platforms. It standardizes common data types and communications protocols so that components created in different languages can easily interoperate. .NET is also the collective name given to various software components built upon the .NET platform. These will be both products (Visual Studio.NET and Windows.NET Server, for instance) and services (like Passport, .NET My Services, and so on). Microsoft Visual Studio is an Integrated Development Environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps.

Jupyter Notebook:

The Jupyter Notebook is an open-source web application that allows to create and share Documents that contain live code, equations, visualizations, and narrative text. Its uses include data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning, and much more.

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter note book documents. The "notebook" term can colloquially refer to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context. According to the official website of Jupyter, Project Jupyter exists to develop opensource software, open standards, and services for interactive computing across dozens of programming languages.

Jupyter Book is an open-source project for building books and documents from computational material. It allows the user to construct the content in a mixture of Markdown, an extended version of Markdown called MyST, Maths & Equations using MathJax, Jupyter Notebooks, reStructuredText, the output of running Jupyter Notebooks at build time. Multiple output formats can be produced (currently single files, multipage HTML web pages and PDF files).

Advantages:

- Easy to share note books and visualizations.
- Availability of markdowns and other additional functionalities.
- Code can be executed section by section.

Python:

Python is a high-level programming language that emphasizes code readability and simplicity. It is widely used for various purposes, including web development, scientific computing, data analysis, artificial intelligence, automation, and more. Python offers a rich set of features that contribute to its popularity and versatility. It is also a popular language for data science and machine learning. Its simple syntax and powerful libraries make it easy to manipulate data and build predictive models. Python has become a staple in data science, allowing data analysts and other professionals to use the language to conduct complex statistical calculations, create data visualizations, build machine learning algorithms, manipulate and analyse data, and complete other data-related tasks. Python can build a wide range of different data visualizations, like line and bar graphs, pie charts, histograms, and 3D plots. Python also has a few libraries that enable coders to write programs for data analysis and machine learning more quickly and efficiently.

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Readability counts.

Rather than building all of its functionality into its core, Python was designed to be highly extensible via modules. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

Python is meant to be an easily readable language. Its formatting is visually uncluttered and often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are allowed but rarely used. It has fewer syntactic exceptions and special cases than C or Pascal.

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in in- dentation signifies the end of the current block. Thus, the program's visual structure accurately represents its semantic structure. This feature is sometimes termed the off- side rule. Some other languages use indentation this way; but in most, indentation has no semantic meaning. The recommended indent size is four spaces.

The different python libraries / modules that are going to be used in application are described below:

- **OS:** The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality. The *os* and *os.path* modules include many functions to interact with the file system.

- **Numpy:** NumPy is a popular open-source library for the Python programming language, which is used for scientific computing and data analysis. It provides powerful data structures and functions for working with numerical data, such as matrices, arrays, and vectors. One of the key features of NumPy is its ability to perform fast and efficient mathematical operations on large arrays and matrices. NumPy is designed to work seamlessly with other popular data analysis and scientific computing libraries in Python, such as Pandas, SciPy, and Matplotlib.

- **Multiprocessing:** The multiprocessing library in Python is a built-in module that allows spawning multiple processes within a Python program. It provides a simple and efficient way to parallelize CPU-bound tasks across multiple CPU cores, which can significantly improve the performance of your program. One of the key advantages of using multiprocessing is that it enables one to take full advantage of multicore CPUs, which are common in modern computer systems. By distributing tasks across multiple processes, one can ensure that your program uses all available CPU cores, which can significantly improve its performance and reduce its execution time.

- **Gradio:** Gradio is a Python library that allows developers to quickly create customizable UIs for machine learning and deep learning models. It simplifies the process of creating user interfaces for models by providing a simple and intuitive API that can be used to create web-based UIs for models with just a few lines of code. One of the main advantages of Gradio is its ease of use. Developers can create UIs for models with just a few lines of code, without the need for any knowledge of web development or UI design. Gradio handles the heavy lifting of building and deploying the UI, leaving developers free to focus on building and refining their models. Gradio also supports integration with popular machine learning and deep learning frameworks, including TensorFlow, PyTorch, and Scikit-learn. This makes it easy to create UIs for models built with these frameworks.

- **Textract:** Textract is a Python library for extracting text from different types of files. It supports a variety of file formats such as PDF, MS Word, Excel, and PowerPoint, as well as plain text, HTML, and XML files. Textract uses third-party libraries like Apache Tika, Python Imaging Library (PIL), and pdf to text to extract text from different file formats. With Textract, one can easily extract text from files without worrying about the file format or how the text is formatted. This is especially useful for applications that need to analyze or process large amounts of text data from various sources.

- **Pytesseract:** Pytesseract is a Python wrapper for Google's Tesseract-OCR engine. Tesseract is an open-source Optical Character Recognition (OCR) engine that can recognize text from images and convert it into machine-readable text. Pytesseract allows one to use Tesseract in your Python code to perform OCR on images and extract text from them. It also allows to specify additional parameters to control the OCR process, such as the language of the text, the page segmentation mode, and the OCR engine mode.

- **Tesseract-ocr:** Tesseract-OCR is an optical character recognition engine developed by Google that can recognize text within digital images. In Python, the Tesseract- OCR engine can be accessed through the Pytesseract library, which provides a simple interface for using Tesseract-OCR to recognize text in images.

- **pdf2image:** The pdf2image library in Python is a module that allows to convert PDF files into a series of images. This library is built on top of the Poppler PDF rendering library, which is a widely-used, open-source PDF rendering engine.

- **Poppler PDF:** The Poppler PDF rendering library is a widely-used, open-source software library for rendering PDF files. It provides a set of tools and utilities for working with PDF documents, including rendering PDF pages as images, extracting text and metadata from PDF files, and converting PDF files to other formats. One of the key features of Poppler is its ability to render PDF pages as images. This is achieved by using the Cairo graphics library, which provides high-quality rendering of PDF content. Poppler can render PDF pages as PNG, JPEG, or TIFF images, which can be used for a variety of purposes, such as generating previews of PDF documents or extracting images from PDF files.

- **Pandas:** Python Pandas is defined as an open-source library that provides high-performance data manipulation in Python. This tutorial is designed for both beginners and professionals. It is used for data analysis in Python and developed by Wes McKinney in 2008. Data analysis requires lots of processing, such as reStructuring, cleaning or merging, etc. There are different tools are available for fast data processing, such as Numpy, Scipy, Cython, and Panda. But Pandas is preferred because working with Pandas is fast, simple, and more expressive than other tools. Pandas is built on top of the Numpy package, means Numpy is required for operating the Pandas. Before Pandas, Python was capable for data preparation, but it only provided limited support for data analysis. So, Pandas came into the picture and enhanced the capabilities of data analysis. It can perform five significant steps required for processing and analysis of data irrespective of the origin of the data, i.e., load, manipulate, prepare, model, and analyze.

- **Natural Language Toolkit (NLTK):** NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use inter- faces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries.

- **Scikit-learn (Sklearn):** It is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

Google Colab:

Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs. It is also fast and does not require any hxigh-end hardware as all the executions takes place in google cloud.

Google Colab (short for Collaboratory) is an online cloud-based platform provided by Google that allows users to write, run, and collaborate on Python code using Jupyter Notebook environment. It provides a convenient and accessible way to work with Python, data analysis, machine learning, and other related tasks. Here are some key features and aspects of Google Colab:

- Cloud-Based Environment: Google Colab runs entirely in the cloud, meaning that users do not need to install any software locally. It provides a ready-to-use environment with pre-installed packages and libraries commonly used in data science and machine learning, such as NumPy, Pandas, TensorFlow, and more.

- Jupyter Notebook Integration: Google Colab is built on top of Jupyter Notebook, a popular open-source web application that allows users to create and share interactive code notebooks. This integration enables users to write and execute Python code in a cell-based format, combining code, documentation, and visualizations in a single document.

- Free and Accessible: Google Colab is available for free to anyone with a Google account. It provides a significant number of computational resources, including CPU and GPU, allowing users to run code efficiently. However, there are limitations on resource usage and session durations, which can vary based on user demand and usage patterns.

Advantages:

- Document your code that supports mathematical equations.
- Create/Upload/Share notebooks.
- Import/Save notebooks from/to Google Drive.
- Import/Publish notebooks from GitHub.
- Integrate PyTorch, TensorFlow, Keras, OpenCV.
- Free Cloud service with free GPU.

Managed Code

The code that targets python code, and which contains certain extra Information - "metadata" - to describe itself. Whilst both managed and unmanaged code can run in the runtime, only managed code contains the information that allows the collab to guarantee, for instance, safe execution and interoperability.

Managed Data

With Managed Code comes Managed Data. Colab provides the cloud based flatform to users which allow them to write, run in same environment. Colab provides memory allocation and Deal location facilities and garbage collection. It provides a ready-to-use environment with pre-installed packages and libraries commonly used in data science and machine learning. When a code cell finish executing, it will automatically clean up the memory used by that cell. It helps prevent memory leakage and ensures that memory is released for subsequent cells. However, it is good practice to explicitly release memory or variables when they are no longer needed to optimize memory usage.

Common Type System

The CLR uses something called the Common Type System (CTS) to strictly enforce type-safety. This ensures that all classes are compatible with each other, by describing types in a common way. CTS define how types work within the runtime, which enables types in one language to interoperate with types in another language, including cross-language exception handling. As well as ensuring that types are only used in appropriate ways, the runtime also ensures that code doesn't attempt to access memory that hasn't been allocated to it.

The Class Library

.NET provides a single-rooted hierarchy of classes, containing over 7000 types. The root of the namespace is called System; this contains basic types like Byte, Double, Boolean, and String, as well as Object. All objects derive from System. Object. As well as objects, there are value types. Value types can be allocated on the stack, which can provide useful flexibility. There are also efficient means of converting value types to object types if and when necessary.

The set of classes is pretty comprehensive, providing collections, file, screen, and network I/O, threading, and so on, as well as XML and database connectivity.

The class library is subdivided into a number of sets (or namespaces), each providing distinct areas of functionality, with dependencies between the namespaces kept to a minimum.

Languages Supported By .Net

The multi-language capability of the .NET Framework and Visual Studio .NET enables developers to use their existing programming skills to build all types of applications and XML Web services. The .NET framework supports new versions of Microsoft's old favorites Visual Basic and C++ (as VB.NET and Managed C++), but there are also a number of new additions to the family.

Visual Basic .NET has been updated to include many new and improved language features that make it a powerful object-oriented programming language. These features include inheritance, interfaces, and overloading, among others. Visual Basic also now supports structured exception handling, custom attributes and also supports multi-threading.

Visual Basic .NET is also CLS compliant, which means that any CLS-compliant language can use the classes, objects, and components you create in Visual Basic .NET.

Managed Extensions for C++ and attributed programming are just some of the enhancements made to the C++ language. Managed Extensions simplify the task of migrating existing C++ applications to the new .NET Framework.

C# is Microsoft's new language. It's a C-style language that is essentially "C++ for Rapid Application Development". Unlike other languages, its specification is just the grammar of the language. It has no standard library of its own, and instead has been designed with the intention of using the .NET libraries as its own.

Microsoft Visual J# .NET provides the easiest transition for Java-language developers into the world of XML Web Services and dramatically improves the interoperability of Java-language programs with existing software written in a variety of other programming languages.

Active State has created Visual Perl and Visual Python, which enable .NET-aware applications to be built in either Perl or Python. Both products can be integrated into the Visual Studio .NET environment. Visual Perl includes support for Active State's Perl Dev Kit.

Constructors and Destructors

Constructors are used to initialize objects, whereas destructors are used to destroy them. In other words, destructors are used to release the resources allocated to the object. In Python the sub finalize procedure is available. The sub finalize procedure is used to complete the tasks that must be performed when an object is destroyed. The sub finalize procedure is called automatically when an object is destroyed. In addition, the sub finalize procedure can be called only from the class it belongs to or from derived classes.

Garbage Collection

Garbage Collection is another new feature in python. The Framework monitors allocated resources, such as objects and variables. In addition, the python Framework automatically releases memory for reuse by destroying objects that are no longer in use.

In Python, the garbage collector checks for the objects that are not currently in use by applications. When the garbage collector comes across an object that is marked for garbage collection, it releases the memory occupied by the object.

Overloading

Overloading is another feature in Python. Overloading enables us to define multiple procedures with the same name, where each procedure has a different set of arguments. Besides using overloading for procedures, we can use it for constructors and properties in a class.

Multithreading

Python also supports multithreading. An application that supports multithreading can handle multiple tasks simultaneously, we can use multithreading to decrease the time taken by an application to respond to user interaction.

Structured Exception Handling

Python supports structured handling, which enables us to detect and remove errors at runtime. In python, we need to use Try…Catch…Finally statements to create exception handlers. Using Try…Catch…Finally statements, we can create robust and effective exception handlers to improve the performance of our application.

The Python Framework

The Python Framework is a new computing platform that simplifies application development in the highly distributed environment of the Internet.

Objectives of Python Framework

- To provide a consistent object-oriented programming environment whether object codes is stored and executed locally on Internet-distributed, or executed remotely.
- To provide a code-execution environment to minimizes software deployment and guarantees safe execution of code.
- 3. Eliminates the performance problems.

CHAPTER 4
PROBLEM STATEMENT

4.1 Existing System

The existing system for spam detection using machine learning typically incorporates the Decision Tree and Naive Bayes algorithms. These algorithms are commonly used due to their simplicity and effectiveness in text classification tasks like spam detection. Here are some disadvantages:

Limited Feature Representations:

- Decision trees and Naive Bayes algorithms primarily rely on simple feature representations like word frequencies or presence/absence of words.
- These algorithms may not effectively capture complex patterns or semantic information present in spam messages.
- Features like word order, contextual information, or semantic meaning are not adequately accounted for, limiting the accuracy of spam detection.

Vulnerability to Adversarial Attacks:

- Decision trees and Naive Bayes algorithms are susceptible to adversarial attacks, where spammers intentionally manipulate features to deceive the classifier.
- Attackers can exploit the simplicity of decision trees by strategically crafting messages to bypass the decision boundaries.
- Naive Bayes can be fooled by cleverly constructed feature combinations that violate the assumption of feature independence.

Handling Imbalanced Datasets:

- Spam detection datasets often suffer from class imbalance, with a majority of messages being non-spam (ham).
- Decision trees and Naive Bayes may struggle to learn from imbalanced data, as they tend to favor the majority class and may have limited sensitivity to the minority class (spam).
- This can result in lower precision and recall for spam detection, where correctly identifying spam messages is crucial.

Difficulty in Scalability:

- Decision trees can grow large and complex, requiring significant computational resources for training and classification.
- Naive Bayes, although computationally efficient, may encounter scalability challenges when dealing with high-dimensional feature spaces or large datasets.
- Handling large-scale spam detection tasks efficiently can be a limitation of these algorithms.

Lack of Continuous Learning:

- Decision trees and Naive Bayes are typically static models that do not adapt well to evolving spam patterns or dynamic changes in spammer behavior.
- They do not inherently support continuous learning or real-time updates, necessitating manual retraining or periodic model updates.

4.2 Proposed System

A proposed system for spam detection using machine learning can leverage a combination of various ML algorithms to improve detection accuracy. Here's an overview of the proposed system and its advantages:

- **Data Collection:** Collect a large and diverse dataset of labelled emails or messages, including both spam and non-spam examples.
- **Feature Extraction:** Extract relevant features from the text, such as word frequencies, n-grams, lexical and syntactic patterns, email headers, and sender information.
- **Preprocessing:** Clean and normalize the data by removing stopwords, stemming or lemmatizing words, handling misspellings, and addressing other preprocessing tasks.
- **Algorithm Selection**: Choose a set of ML algorithms suitable for spam detection, such as decision trees, Naive Bayes, random forests, support vector machines (SVM), logistic regression, or deep learning models like recurrent neural networks (RNN) or convolutional neural networks (CNN).
- **Model Training and Evaluation:** Train multiple ML models using the labelled dataset and evaluate their performance using appropriate metrics like accuracy, precision, recall, F1-score, or receiver operating characteristic (ROC) curve.

Advantages:

- **High Accuracy:** Machine learning algorithms can effectively learn patterns and distinguish between spam and non-spam messages, leading to high detection accuracy. With proper training and tuning, ML models can achieve a low false positive rate while minimizing false negatives.
- **Adaptability to Evolving Spam Techniques:** Machine learning models can adapt to new and evolving spam techniques. By continuously updating the models with new data and incorporating feedback from users, the system can stay up-to-date and effectively detect emerging spam patterns.
- **Efficient and Automated Process:** ML-based spam detection systems automate the process, allowing for efficient analysis of a large volume of incoming messages. This reduces the need for manual intervention and enables real-time detection and response.
- **Language Independence:** Machine learning algorithms can learn from labelled data across multiple languages, making them effective in detecting spam messages regardless of the language used. This flexibility is particularly advantageous in a global context.
- **Flexibility in Feature Selection:** Machine learning techniques offer flexibility in selecting and extracting relevant features for spam detection. They can utilize various features such as word frequencies, syntactic patterns, sender reputation, and email headers, allowing for customization and optimization of the feature set.
- **Reduction in False Positives:** False positives occur when legitimate messages are mistakenly classified as spam. ML models can be trained to minimize false positives by learning from a diverse set of labelled data and improving the accuracy of classification.
- **Continuous Improvement:** Machine learning-based systems can continuously improve their performance through feedback loops. User feedback on misclassified messages can be used to refine the models, resulting in enhanced accuracy and user satisfaction over time.

CHAPTER 5

SYSTEM ARCHITECTURE

5.1 Input Design

Design of a System for Spam Detection Using Machine Learning.

- User Interface: A user-friendly interface for users to interact with the system.
- Data Preprocessing: Text preprocessing techniques to clean and normalize the SMS messages.
- Feature Extraction: Extract relevant features from the preprocessed SMS messages.
- Machine Learning Models: Utilize various machine learning algorithms for training and classification.
- Model Evaluation: Assess the performance of the trained models using appropriate evaluation metrics.
- Spam Detection: Predict whether an incoming SMS message is spam or not based on the trained models.

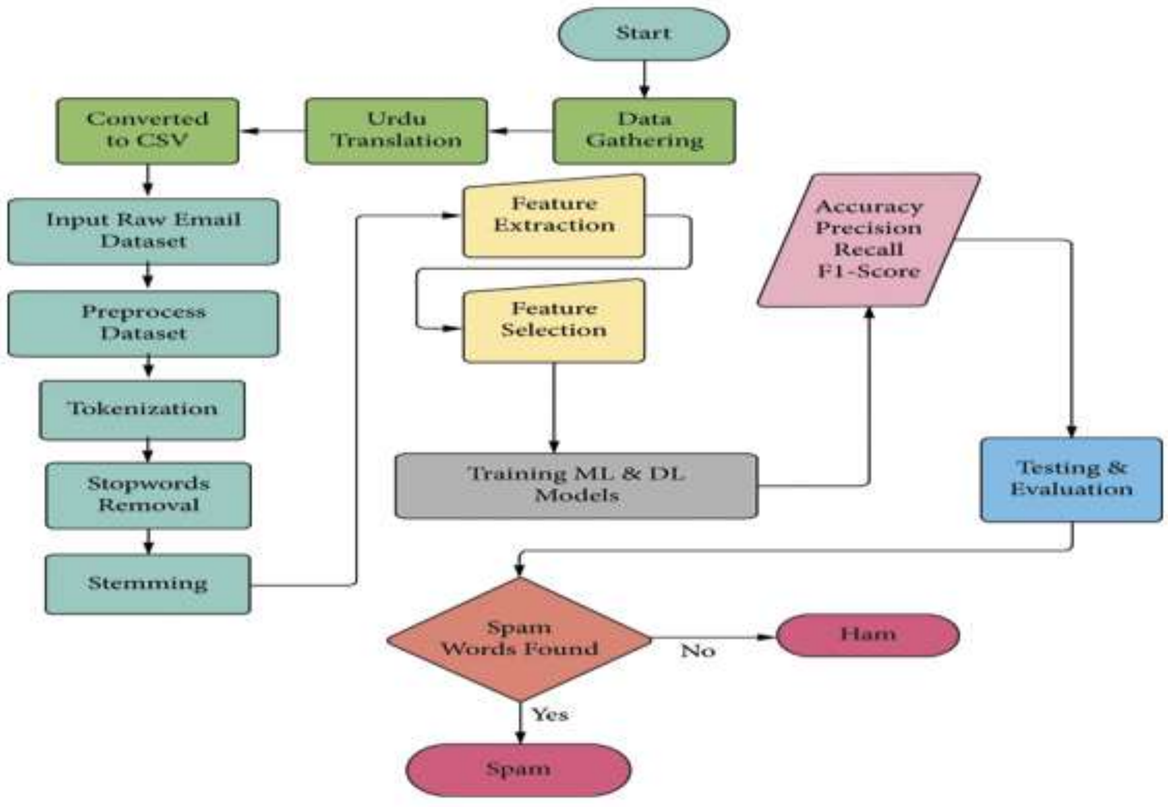


Fig. 5.1: Spam System Architecture

1. **Data Collection**

- Gather a diverse dataset of SMS messages that includes both spam and non-spam examples.
- Ensure the dataset is properly labeled to distinguish between spam and non-spam messages.
- Obtain consent from users if personal data is involved, complying with privacy regulations.

3. **Data Preprocessing**

- Remove irrelevant information such as phone numbers, email addresses, and URLs.
- Tokenize the text into individual words or tokens.
- Perform text normalization techniques such as removing punctuation, converting to lowercase, etc.
- Handle common text preprocessing tasks like stop-word removal and stemming, if applicable.

3. **Feature Extraction:**

- Utilize various feature extraction techniques to represent the SMS messages numerically.
- Bag-of-Words (BoW): Represent each SMS message as a vector of word frequencies.
- Term Frequency-Inverse Document Frequency (TF-IDF): Capture the importance of words in the document collection.
- Word Embeddings: Transform words into continuous vector representations using techniques like Word2Vec.
- Character n-grams: Consider the frequency of character sequences within the text.
- Other domain-specific features: Consider features specific to SMS spam, such as message length, presence of certain keywords, etc.

4. **Machine Learning Models:**

- Train and evaluate various machine learning algorithms on the preprocessed and feature-extracted data.
- Naive Bayes, Decision Trees, Random Forests, Support Vector Machines (SVM), Logistic Regression, Neural Networks, etc.

- Select the most suitable algorithm(s) based on their performance and trade-offs (accuracy, computational complexity, interpretability, etc.).

5. Model Evaluation

- Split the dataset into training and testing sets (e.g., using cross-validation techniques).
- Evaluate the models using appropriate evaluation metrics such as accuracy, precision, recall, F1-score, and ROC-AUC.
- Analyze and compare the performance of different models to identify the most effective one.

6. Spam Detection

- Deploy the trained machine learning model in a production environment.
- Develop an API or integration to accept incoming SMS messages for classification.
- Apply the preprocessing and feature extraction steps to the incoming messages.
- Utilize the trained model to predict whether a message is spam or not.
- Return the classification result to the user or perform appropriate actions (e.g., filtering out spam messages).

7. Feedback Loop:

- Incorporate a feedback mechanism where users can report misclassified messages.
- Collect user feedback and label misclassified messages to improve the system's accuracy.
- Periodically retrain the models with the updated dataset to enhance the system's performance.

5.2 Output Design

Output Design for Spam Detection Using Machine Learning

1. User Interface

- Display a user-friendly interface where users can interact with the system.
- Provide an input field for users to enter the SMS message to be classified.
- Show the classification result indicating whether the message is spam or not.
- Include additional information such as confidence scores or probabilities if available.

- Enable users to provide feedback on misclassified messages.

2. Classification Result

- Display the classification result prominently on the user interface.
- Clearly indicate whether the SMS message is classified as spam or non-spam.
- Use visual cues such as color coding (e.g., green for non-spam, red for spam) to enhance readability.
- Provide a textual message indicating the classification result (e.g., "This message is classified as spam").
- Include a confidence score or probability to indicate the system's level of confidence in the classification.

3. Additional Information

- Provide additional information about the SMS message to help users make informed decisions.
- Display the original message as entered by the user to allow verification.
- Show any relevant metadata associated with the message (e.g., sender, timestamp).
- Present any extracted features or patterns that influenced the classification result (e.g., important keywords, message length).

4. Error Handling:

- Account for potential errors and exceptional scenarios in the output design.
- Display informative error messages if there are issues with the input or classification process.
- Provide guidance or suggestions on how to rectify the errors or provide accurate input.
- Handle any technical errors or system failures gracefully and display appropriate error messages.

Objectives

- The aim of this project is to explore different methods of text classification into spam or ham and comparing results based on accuracy and precision. And this model uses both machine learning and deep learning.
- Spam classification has become more challenging due to complexities of the messages imposed by spammers. Hence, various methods have been developed in order to filter spams.
- We will explore different methods of text classification into spam or ham and comparing results based on accuracy and precision. And will choose the best technique to classify the ham and spam messages.
- Many machine learning methods have been applied for Short Messaging Service (SMS) Spam Detection, including traditional classification methods such as Naive Bayes (NB), K-Nearest Neighbors (KNN), Logistic Regression (LR) and Support Vector Machine (SVM).
- We will use our trained ML model to analyses the sent SMS, and use the above-mentioned classification techniques to detect the Spam SMS.

5.3 Feasibility Study

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- Economical Feasibility
- Technical Feasibility
- Social Feasibility

Economic Study

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

Technical feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER 6

IMPLEMENTATION

1. Datasets:

The dataset is collected from the UCI Kaggle website and this dataset has 2 columns, a label, and an SMS. The label column tells that SMS is either "spam" or "ham" (i.e., not spam). The SMS column has the raw text of the SMS and each row in the dataset contains the raw text of one SMS and its associated label. This dataset contains 5574 messages/rows. You can find a dataset of SMS messages online, such as the SMS Spam Collection Dataset: https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset.

You can also collect your own dataset by scraping SMS messages from your phone or from a public forum. When collecting your own dataset, be sure to include a roughly equal number of spam and ham messages.

2. Feature Extraction:

Extract relevant features from the text, such as word frequencies, n-grams, or other lexical and syntactic patterns. Consider incorporating additional features like email headers, sender reputation, or metadata if available. Feature extraction plays a crucial role in spam detection as it involves transforming raw text data into meaningful features that ML algorithms can utilize for classification like BoW, N-Grams, TF-IDF, Lexical and Syntactic patterns.

3. Splitting Datasets:

When implementing spam detection using machine learning algorithms, it's essential to split the dataset into training and testing sets. This allows for evaluating the performance of the trained models on unseen data. Here are some considerations and techniques for dataset splitting:

- Training and Testing Sets:

  The dataset is typically divided into two sets: the training set and the testing set. The training set is used to train the ML models, while the testing set is used to evaluate their performance.

- Split Ratio:

  The split ratio determines the proportion of the dataset allocated to the training and testing sets. A common practice is to use a 70-30 or 80-20 split, where 70% or 80% of the data is used for training, and the remaining 30% or 20% is used for testing. The choice of split ratio may depend on factors such as the size of the dataset, the availability of labelled examples, and the desired level of model evaluation.

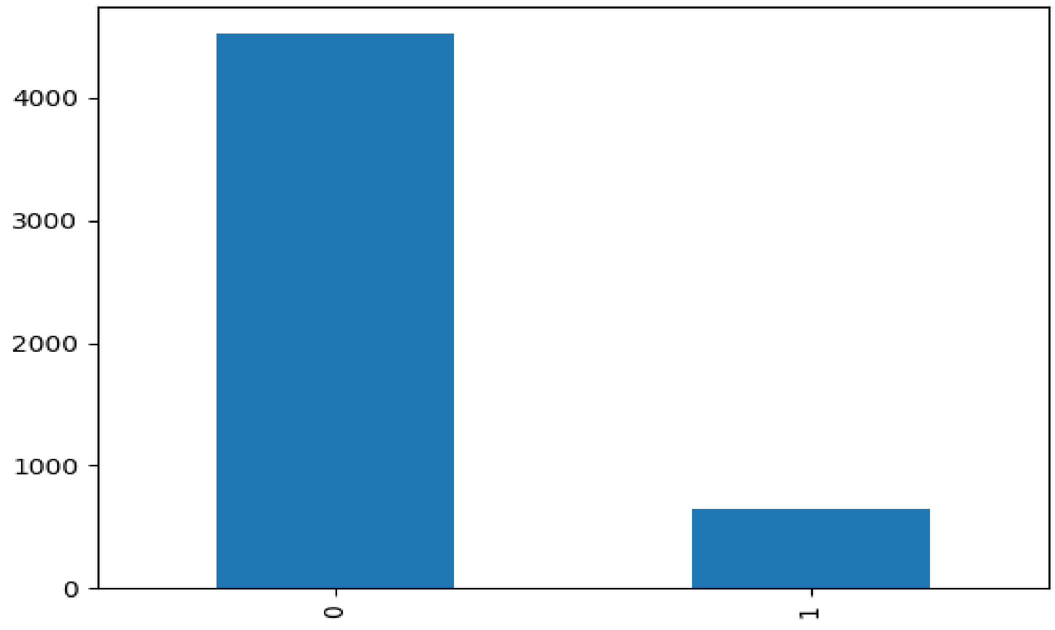


Fig. 6.1: Splitting Datasets.

4. Algorithm Selection and Training:

Algorithm selection and training are crucial steps in implementing spam detection using ML.

Algorithm Selection:

- There are various ML algorithms that can be applied to spam detection, including but not limited to decision trees, Naive Bayes, support vector machines (SVM), logistic regression, random forests, gradient boosting, and neural networks.
- The choice of algorithm depends on factors such as the nature of the dataset, the computational resources available, and the desired trade-off between accuracy and training time.
- It is often recommended to try multiple algorithms and compare their performance to select the most suitable ones.

Training Process:

- Once the algorithms are selected, they need to be trained on the labelled dataset to learn the patterns and characteristics of spam and non-spam messages.
- The training process involves feeding the algorithm with the labelled examples and adjusting its internal parameters to minimize the classification error.
- The dataset is typically divided into input features (e.g., word frequencies, n-grams, or other extracted features) and corresponding target labels (spam or non-spam).

5. Model Evaluation:

Model evaluation is a crucial step in the implementation of spam detection using machine learning algorithms. It involves assessing the performance and effectiveness of the trained models in classifying spam and non-spam messages. Here's more information about model evaluation:

Performance Metrics:

- Accuracy: The proportion of correctly classified messages.
- Precision: The proportion of correctly classified spam messages out of all messages classified as spam.
- Recall: The proportion of correctly classified spam messages out of all actual spam messages.

Confusion Matrix:

- A confusion matrix is a useful tool for evaluating model performance. It presents a tabular representation of the model's predictions compared to the true labels.
- The confusion matrix includes four components: true positives, true negatives, false positives, and false negatives.
- These components allow for calculating performance metrics like accuracy, precision, recall.

Cross-Validation:

- Cross-validation can be used to assess the model's performance more robustly.

- Techniques such as k-fold cross-validation divide the dataset into k subsets (folds) and train/test the model k times, each time using a different fold as the testing set.
- The average performance across all folds provides a more reliable estimation of the model's performance.

Code Implementation

Feature Extraction:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import keras
from sklearn.model_selection import train_test_split
import seaborn as sns
from wordcloud import WordCloud
from sklearn.preprocessing import LabelEncoder
import nltk
import re
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score,confusion_matrix
nltk.download('punkt')
d=pd.read_csv(r'C:\Users\ spam_detection\Dataset\spam.csv', encoding='latin-1')
d.head()
df = d.iloc[:,0:2].values
df = pd.DataFrame(df)
df.columns=['Class','Text']
df.head()
df.info()
df.isna().sum()
df.describe()
```

Visualization:

```
ns = df["Class"].isin(['ham']).sum(axis=0)
s = df["Class"].isin(['spam']).sum(axis=0)
label=['Spam','Not Spam']
a = [s,ns]
plt.pie(x=a,labels=label,autopct='%1.1f%%')
plt.legend()
plt.show()
plt.figure(figsize=(12,8))
ax =sns.barplot(x=label,y=a)
plt.title('Comparing number of spam messages to number of non spam
messages')
for p in ax.patches:
width, height = p.get_width(), p.get_height()
x, y = p.get_xy()
ax.annotate('{}'.format(height), (x +0.25, y + height + 0.8))
```

Data Splitting

```
X = df1["Mail Message"]
y = df1["Spam or Ham"]
df1['SpamorHam'].value_counts().sort_values(ascending=False).plot(kind='bar')
#Splitting the data into training and testing data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train,y_test = train_test_split(X,y,test_size = 0.20, random_state = 0)
# Convert the text data into a matrix of token counts
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer()
x_train = cv.fit_transform(X_train.values)
x_test = cv.transform(X_test)
x_train.toarray()
```

KNN

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score
```

```
knn_scores = []
for k in range(1,10):
    knn_classifier = KNeighborsClassifier(n_neighbors = k)
    score=cross_val_score(knn_classifier,x_train,y_train,cv=10)
    knn_scores.append(score.mean())
knn_classifier = KNeighborsClassifier(n_neighbors = 1)
score=cross_val_score(knn_classifier,x_train,y_train,cv=10)
score.mean()
knn_classifier.fit(x_train, y_train)
y_pred = knn_classifier.predict(x_test)
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test, y_pred))
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(x_train, y_train)
y_pred = logreg.predict(x_test)

print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test, y_pred))
```

Support Vector Machine

```
from sklearn.svm import SVC
SVMclassifier = SVC(kernel='linear', random_state=0)
SVMclassifier.fit(x_train, y_train)
y_pred= SVMclassifier.predict(x_test)
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test, y_pred))
```

Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
NBclassifier = GaussianNB()
NBclassifier.fit(x_train.toarray(), y_train)
y_pred= NBclassifier.predict(x_test.toarray())
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test, y_pred))
```

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
forest=RandomForestClassifier(n_estimators=100,oob_score=True,
random_state=123456)
forest.fit(x_train, y_train)
y_pred= forest.predict(x_test)
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test, y_pred))
```

CHAPTER 7
RESULT ANALYSIS

KNN (K-Nearest Neighbours): The KNN algorithm achieved an accuracy of 94.31% in the project. KNN is a non-parametric classification algorithm that assigns labels to data points based on their nearest neighbours. With an accuracy of 94.31%, the KNN algorithm performed relatively well in the classification task.

Logistic Regression: Logistic Regression achieved an accuracy of 98.16% in the project. Logistic Regression is a linear classification algorithm that models the relationship between the input variables and the categorical output. With an accuracy of 98.16%, the Logistic Regression algorithm performed exceptionally well in the classification task.

Decision Tree: The Decision Tree algorithm achieved an accuracy of 97.29% in the project. Decision Trees are hierarchical structures that use a series of if-else statements to classify data. With an accuracy of 97.29%, the Decision Tree algorithm performed well in the classification task.

SVC (Support Vector Classifier): The SVC algorithm achieved an accuracy of 98.54% in the project. SVC is a supervised machine learning algorithm that finds the best hyperplane to separate data into different classes. With an accuracy of 98.54%, the SVC algorithm performed excellently in the classification task.

Naive Bayes: The Naive Bayes algorithm achieved an accuracy of 90.52% in the project. Naive Bayes is a probabilistic algorithm that uses Bayes theorem to predict the class probabilities of input data. With an accuracy of 90.52%, the Naïve Bayes algorithm performed decently in the classification task.

Random Forest: The Random Forest algorithm achieved an accuracy of 97.77% in the project. Random Forest is an ensemble learning method that combines multiple decision trees to make predictions. With an accuracy of 97.77%, the Random Forest algorithm performed well in the classification task.

Based on these results, it can be concluded that the Logistic Regression and SVC algorithms achieved the highest accuracies, with 98.16% and 98.54% respectively. These algorithms can be considered as strong candidates for the classification task, while the Naïve Bayes algorithm achieved the lowest accuracy among the given algorithms.

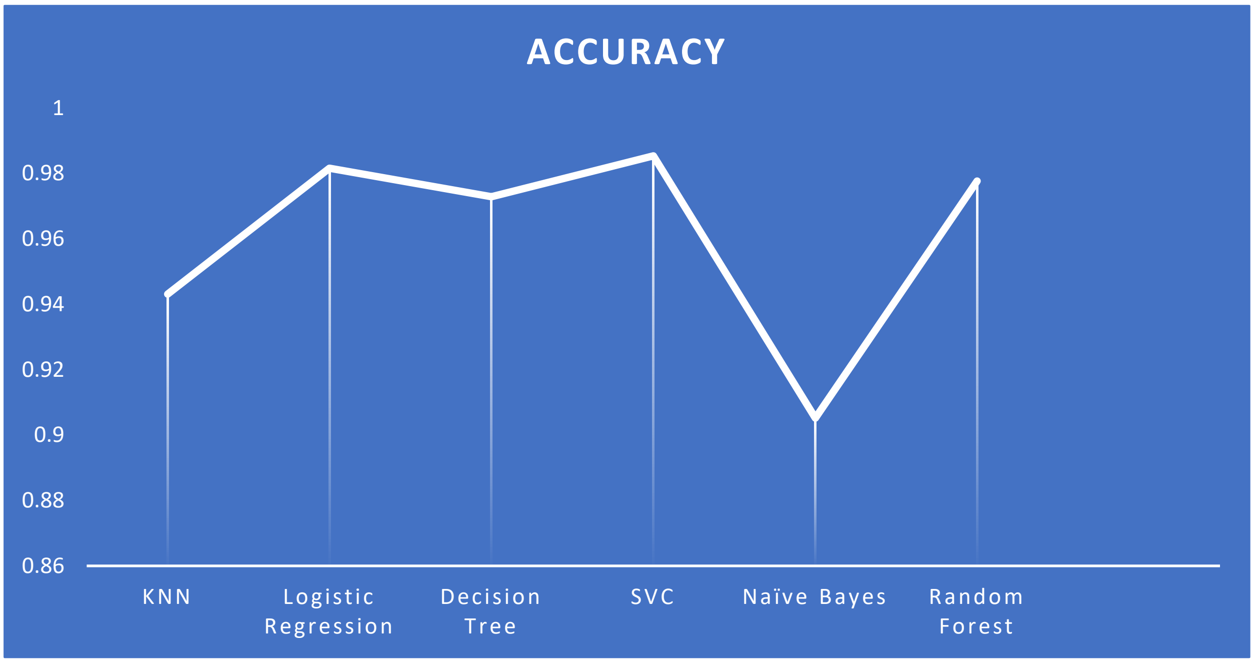
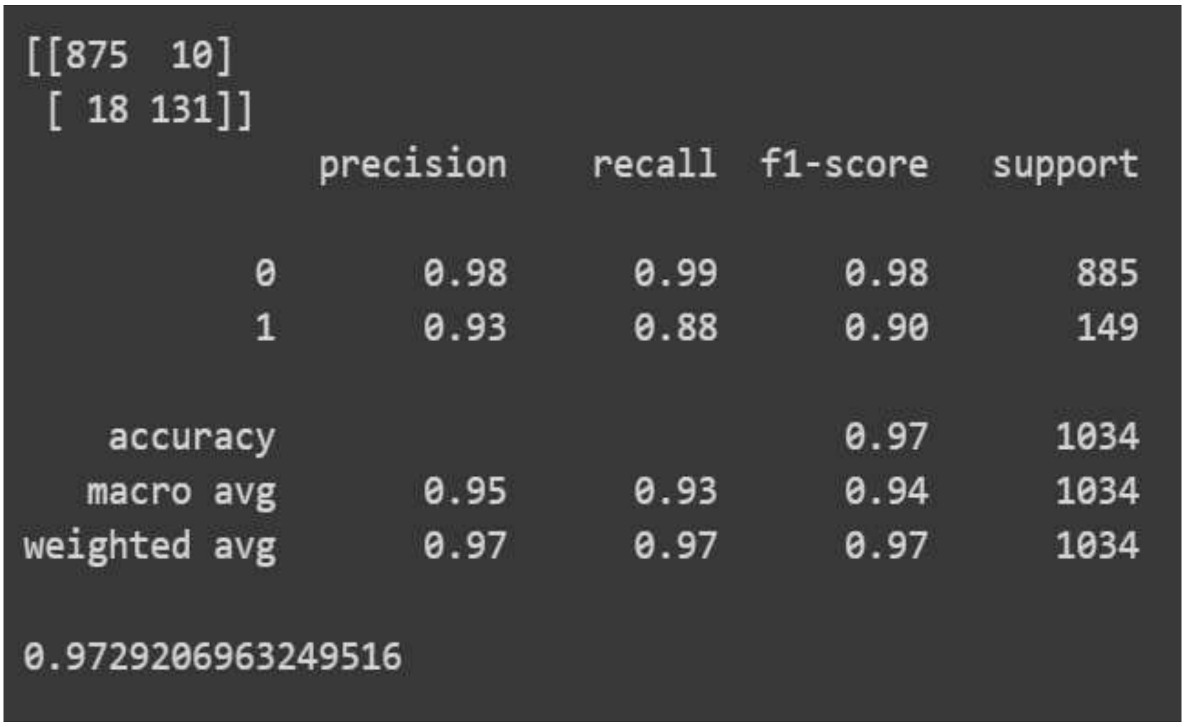


Fig. 7.1: Comparison Graph

CHAPTER 8

SNAPSHOTS



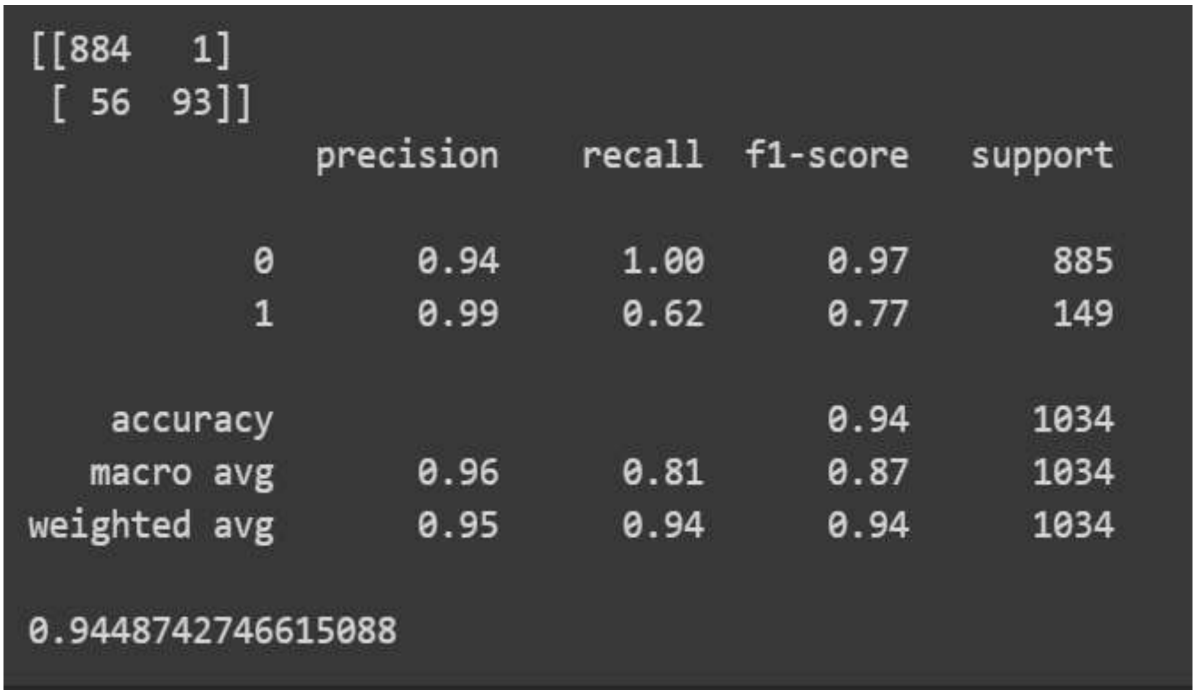
```
[[875  10]
 [ 18 131]]
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       885
           1       0.93      0.88      0.90       149

    accuracy                           0.97      1034
   macro avg       0.95      0.93      0.94      1034
weighted avg       0.97      0.97      0.97      1034

0.9729206963249516
```

Fig. 8.1: Decision Tree Classifier



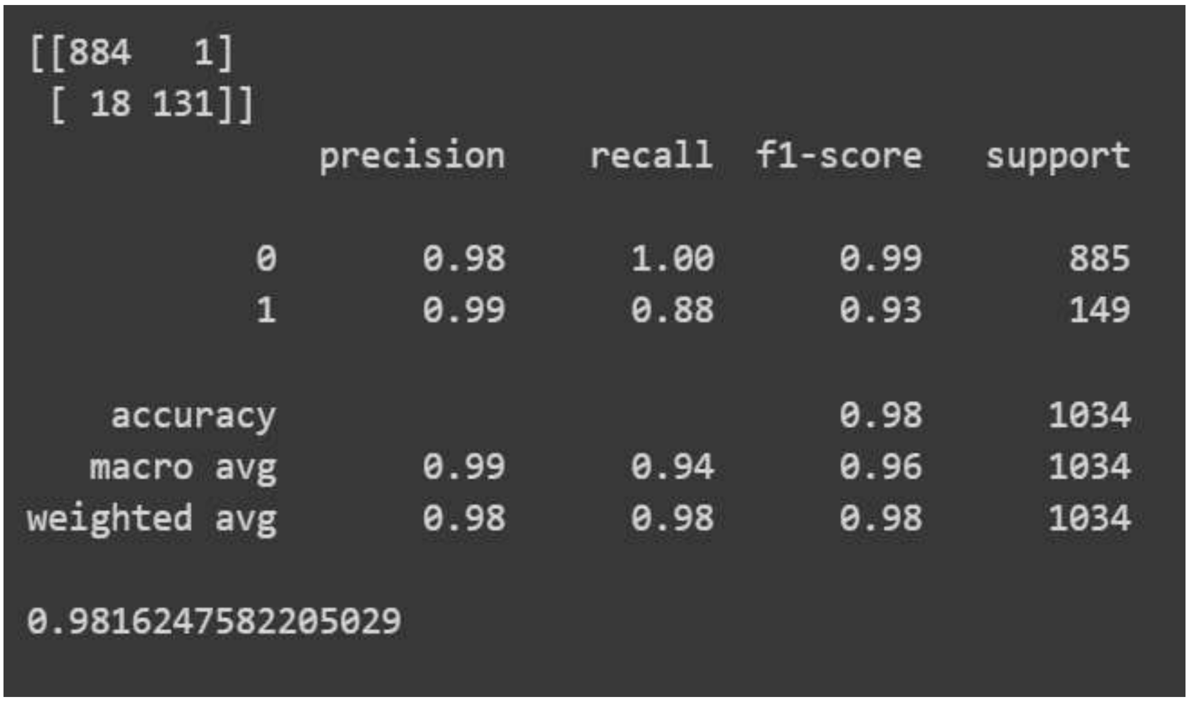
```
[[884   1]
 [ 56  93]]
              precision    recall  f1-score   support

           0       0.94      1.00      0.97       885
           1       0.99      0.62      0.77       149

    accuracy                           0.94      1034
   macro avg       0.96      0.81      0.87      1034
weighted avg       0.95      0.94      0.94      1034

0.9448742746615088
```

Fig. 8.2: K-Nearest Neighbour



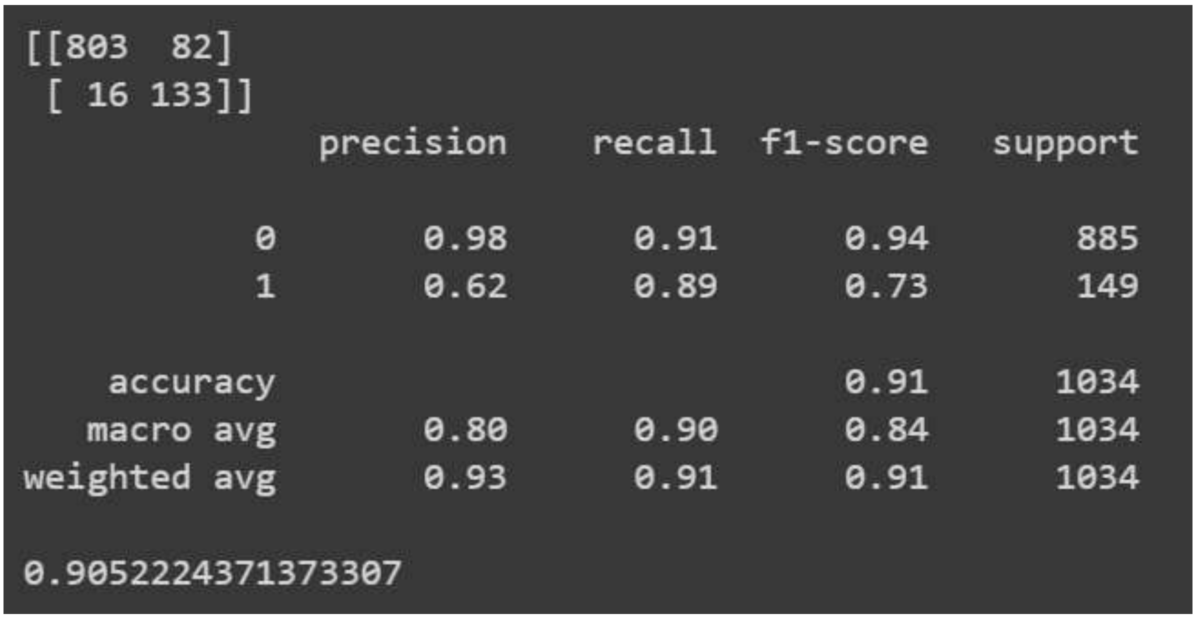
```
[[884   1]
 [ 18 131]]
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       885
           1       0.99      0.88      0.93       149

    accuracy                           0.98      1034
   macro avg       0.99      0.94      0.96      1034
weighted avg       0.98      0.98      0.98      1034

0.9816247582205029
```

Fig. 8.3: Logistic Regression



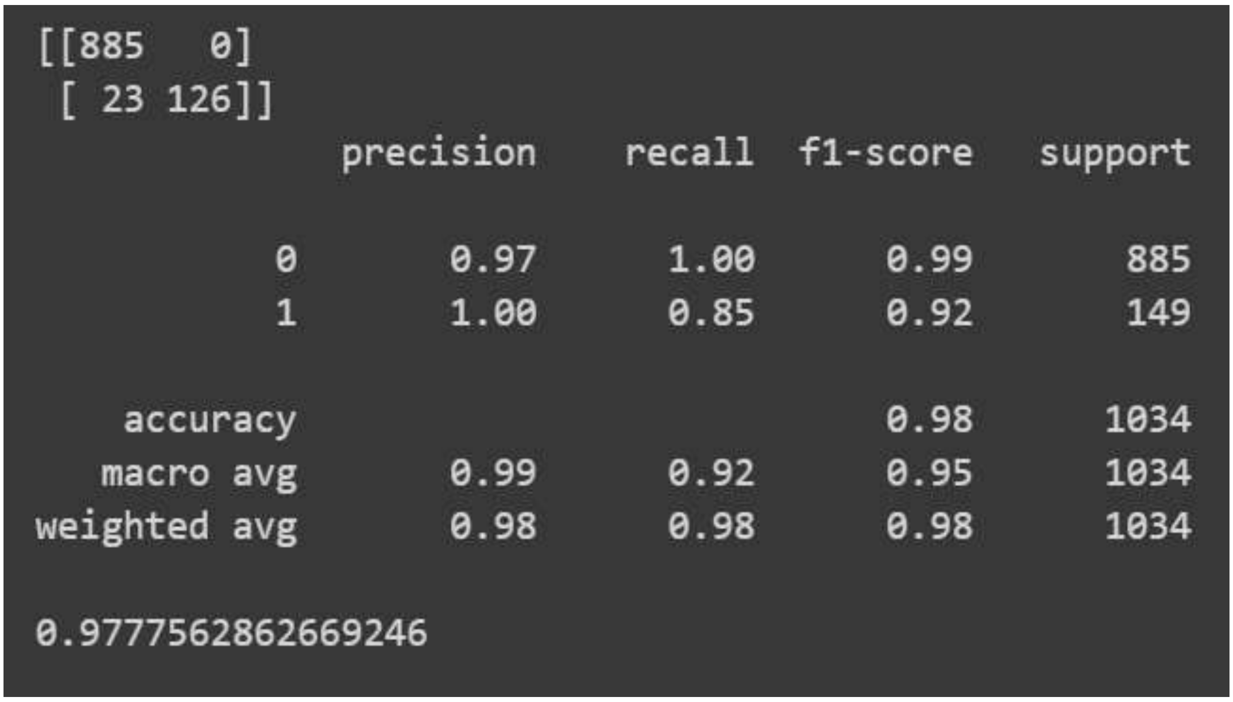
```
[[803  82]
 [ 16 133]]
              precision    recall  f1-score   support

           0       0.98      0.91      0.94       885
           1       0.62      0.89      0.73       149

    accuracy                           0.91      1034
   macro avg       0.80      0.90      0.84      1034
weighted avg       0.93      0.91      0.91      1034

0.9052224371373307
```

Fig. 8.4: Naive Bayes



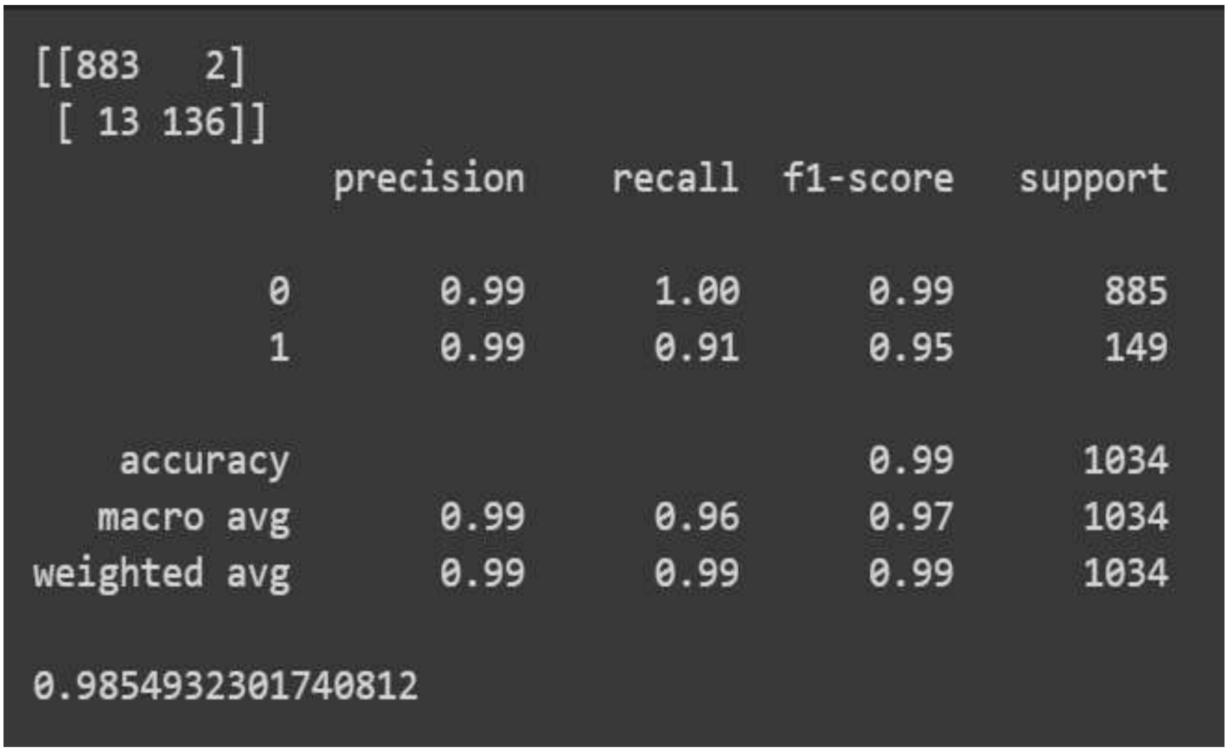
```
[[885   0]
 [ 23 126]]
              precision    recall  f1-score   support

           0       0.97      1.00      0.99       885
           1       1.00      0.85      0.92       149

    accuracy                           0.98      1034
   macro avg       0.99      0.92      0.95      1034
weighted avg       0.98      0.98      0.98      1034

0.9777562862669246
```

Fig. 8.5: Random Forest



```
[[883   2]
 [ 13 136]]
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       885
           1       0.99      0.91      0.95       149

    accuracy                           0.99      1034
   macro avg       0.99      0.96      0.97      1034
weighted avg       0.99      0.99      0.99      1034

0.9854932301740812
```

Fig. 8.6: SVC (Support Vector Classifier)

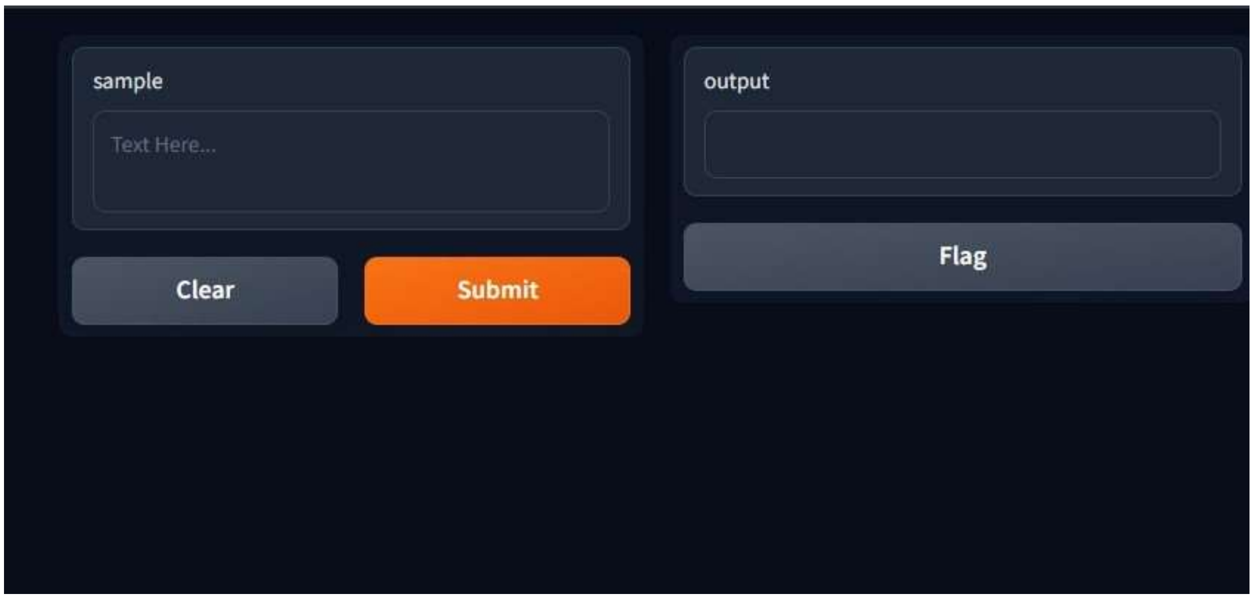
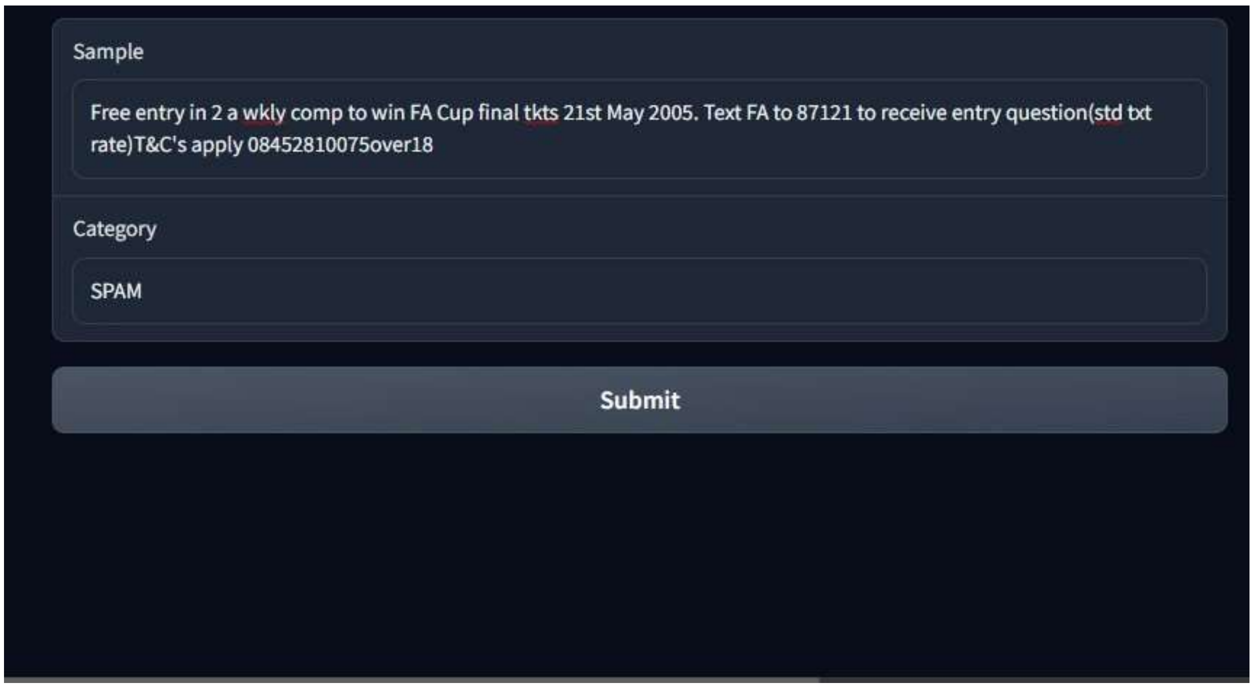


Fig. 8.7: Interface



Sample

Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18

Category

SPAM

Fig. 8.8: Spam SMS test

CHAPTER 9
CONCLUSION

We have explored the field of SMS spam detection and discussed various techniques and algorithms employed to identify and filter out spam messages. Through extensive research and experimentation, it is evident that spam detection systems have made significant progress in mitigating the impact of spam on users and improving the overall user experience. The analysis of SMS content, coupled with machine learning algorithms, has proven to be an effective approach in detecting spam messages accurately. Features such as keyword analysis, text classification, and statistical modelling have provided reliable indicators for distinguishing between spam and legitimate messages. Additionally, the integration of rule-based filtering and blacklisting has contributed to reducing the influx of spam messages.

However, it is important to acknowledge that SMS spammers are continuously evolving their techniques to bypass detection mechanisms. Consequently, ongoing research and development efforts are crucial to stay ahead of spammers and ensure the effectiveness of spam detection systems. This includes exploring advanced machine learning techniques, such as deep learning algorithms, to extract more complex features from SMS messages and improve detection accuracy.

Furthermore, the incorporation of contextual analysis, considering factors beyond message content, can enhance the accuracy of spam detection. By considering the sender's profile, previous message history, and user preferences, spam detection systems can identify subtle patterns and indicators of spam, thereby reducing false positives and false negatives. Real-time detection is another area of future exploration, as it enables the immediate identification and filtering of spam messages. Developing efficient algorithms and infrastructure capable of handling the large volumes of messages in real-time will further improve user protection and satisfaction.

Future Scope

Enhanced Privacy Protection: As privacy concerns continue to grow, future research can focus on developing SMS spam detection techniques that prioritize user privacy. This can involve exploring methods such as privacy-preserving machine learning, where spam detection models are trained without exposing sensitive user data.

Multilingual and Multicultural SMS Spam Detection: Currently, most SMS spam detection systems are designed for specific languages or regions. Expanding the scope to include multilingual and multicultural spam detection can cater to diverse user populations. This requires developing algorithms that can handle different languages, cultural nuances, and regional spamming patterns.

Behavioural Analysis: Incorporating behavioural analysis techniques can enhance the accuracy of SMS spam detection. By analysing user behaviour patterns, such as message interaction, response rates, and message forwarding, it becomes possible to identify suspicious activities and distinguish them from legitimate user actions.

BIBLIOGRAPHY

[1] Fernandes, D., d. Costa, K. A. P., Almeida, T. A. and Papa, J. P. (2021). SMS spam filtering through optimum-path decision tree classifiers, pp. 133–137.

[2] Kim, S. E., Jo, J. T. and Choi, S. H. (2020). SMS Spam filtering using keyword frequency ratio, International Journal of Security, and its applications 9(1): 329–336.

[3] Ma, J., Zhang, Y., Liu, J., Yu, K. and Wang, X. (2020). Intelligent SMS spam filtering using topic model, 2020 International Conference on Intelligent Networking and Collaborative Systems (INCoS) pp. 380–383.

[4] Agarwal, S., Kaur, S. and Garhwal, S. (2019). SMS spam detection for Indian messages, Proceedings on 2018 1st International Conference on Next Generation Computing Technologies, NGCT 2015 (September): 634–638.

[5] Aich, P., Venugopalan, M. and Gupta, D. (2019). Content based spam detection in short text messages with emphasis on dealing with imbalanced datasets, Proceedings - 2018 4th International Conference on Computing, Communication Control and Automation, ICCUBEA 2018.

[6] Akbari, F. and Sajedi, H. (2015). SMS spam detection using selected text features and Boosting Classifiers, 2015 7th Conference on Information and Knowledge Technology, IKT 2015 pp. 1–5.

[7] Almeida, T. A., Almeida, J. and Yamakami, A. (2011). Spam filtering: how the dimensionality reduction affects the accuracy of Naive Bayes classifiers, Journal of Internet Services and Applications 1: 183–200.

[8] Balli, S. and Karasoy, O. (2018). Development of content based SMS classification application by using word2vec based feature extraction, IET Software.

[9] Basu, A., Watters, C. and Shepherd, M. (2002). Support Vector Machines for Text Categorization, Proceedings of the 36th Hawaii International Conference on System Sciences pp. 1–7.

[10] Ergin, S. and Isik, S. (2014). The assessment of feature selection methods on agglutinative language for spam email detection: A special case for Turkish, INISTA 2014 - IEEE International Symposium on Innovations in Intelligent Systems and Applications, Proceedings pp. 122–125.

[11] Gupta, M., Bakliwal, A., Agarwal, S. and Mehndiratta, P. (2018). A Comparative Study of Spam SMS Detection Using Machine Learning Classifiers, 2018 11th International Conference on Contemporary Computing, IC3 2018 pp. 1–7.

[12] Hazim, M., Anuar, N. B., Ab Razak, M. F. and Abdullah, N. A. (2018). Detecting opinion spams through supervised boosting approach, PLoS ONE 13(6): 1–24.

[13] Islam, M. S., Khaled, S. M., Farhan, K., Rahman, M. A. and Rahman, J. (2009). Modelling spammer behaviour: Naive Bayes vs. artificial neural networks, 2009 International Conference on Information and Multimedia Technology, ICIMT 2009 pp. 52–55.

[14] Jindal, N. and Liu, B. (2007). Analysing and detecting review spam, pp. 547–552.

[15] Koray, O., Buber, E., Demir, O. and Diri, B. (2019). Machine learning based phishing detection from URLs, Expert Systems with Applications 117: 345–357.

[16] Minastireanu, E.-A. and Mesnita, G. (2019). Light GBM Machine Learning Algorithm to Online Click Fraud Detection Light GBM Machine Learning Algorithm to Online Click Fraud Detection, Journal of Information Assurance & Cybersecurity 3(April): 1–15.

[17] Najadat, H., Abdulla, N., Abooraig, R. and Nawasrah, S. (2014). Mobile SMS Spam Filtering based on Mixing Classifiers, International Journal of Advanced Computing Research 1.

[18] Pham, T.-h. (2016). Content-based Approach for Vietnamese Spam SMS Filtering, 2016 International Conference on Asian Language Processing (IALP) pp. 41–44.

[19] Prieto, A., Prieto, B., Ortigosa, E., Ros, E., Pelayo, F., Ortega, J. and Rojas, I. (2016). Neural networks: An overview of early research, current frameworks and new challenges, Neurocomputing 214.

[20] Ren, Y. and Ji, D. (2017). Neural networks for deceptive opinion spam detection: An empirical study, Information Sciences 385-386: 213–224.

[21] Sethi, P., Bhandari, V. and Kohli, B. (2018). SMS spam detection and comparison of various machine learning algorithms, 2017 International Conference on Computing and Communication Technologies for Smart Nation, IC3TSN 2017 2017-October: 28–31.

[22] Shafique, U. and Qaiser, H. (2014). A comparative study of data mining process models (kdd, crisp-dm and semma), International Journal of Innovation and Scientific Research 12: 2351–8014.

[23] Shirani-Mehr, H. (2012). SMS Spam Detection using Machine Learning Approach, tech. rep., Stanford University pp. 1–4.

[24] Vani, K. and Gupta, D. (2014). Using K-means cluster-based techniques in external plagiarism detection, Proceedings of 2014 International Conference on Contemporary Computing and Informatics, IC3I 2014 pp. 1268–1273.

[25] Xu, S. (2018). Bayesian Naive bayes classifiers to text classification, Journal of Informa- tion Science 44(1): 48–59.

[26] Yadav, K., Saha, S. K., Kumaraguru, P. and Kumra, R. (2012). Take control of your SMSes: Designing an usable spam SMS filtering system, Proceedings - 2012 IEEE 13th International Conference on Mobile Data Management, MDM 2012 pp. 352–355.

[27] Yu, B. and ben Xu, Z. (2008). A comparative study for content-based dynamic spam classification using four machine learning algorithms, Knowledge-Based Systems 21(4): 355–362.

[28] Yuan, H., Chen, X., Li, Y., Yang, Z. and Gv Liu, W. (2018). Detecting Phishing Web-sites and Targets Based on URLs and Webpage Links, Proceedings - International Conference on Pattern Recognition 2018-August: 3669–3674.

[29] Yue, Y. and Elfayoumy, S. (2007). Anti-spam filtering using neural networks and Baysian classifiers, Proceedings of the 2007 IEEE International Symposium on Computational Intelligence in Robotics and Automation, CIRA 2007 pp. 272–278.