CSC8499 Project and Dissertation:

Twitter Spam Detection

Kailash Balachandiran

MSc in Advanced Computer Science,
School of Computing Science, Newcastle University.
c2024316@newcastle.ac.uk

Abstract. The "Twitter Spam Detection" project aims to develop a web application that can recognise spam tweets on Twitter by utilising several machine learning (ML) models, deep learning techniques, data pre-processing, and natural language processing (NLP). The web application, which includes a user-friendly HTML and CSS front end, is built using the PyMySQL database management system and the Flask web framework. The database is handled using PyMySQL. To look for spam in tweets, users must log in and register. Several ML models and DL models are evaluated for accuracy and F1-score, including Naive Bayes, SVM, Logistic Regression, Random Forest, XGB Classifier, KNeighbours Classifier, Decision Tree, BiLSTM (a type of RNN), CNN and LSTM. The naive Bayesmodel outperforms all others with an accuracy of 82%. The chosen model is kept in a pickle file and added to the web application to enable real- time spam prediction.

Declaration: I declare that this dissertation represents my own work except where otherwise explicitly stated.

1 Introduction

Online social networks (OSNs) are more prevalent than ever, with sites like Twitter, Facebook, and LinkedIn evolving into crucial communication hubs [2]. With its millions of users and astounding daily tweet output, Twitter stands out as one of the largest social networks for microblogging [10]. However, due to Twitter's incredible popularity, spammers and other online criminals have also shown an interest in it.

To spread risky URLs, spread rumours, and send unsolicited messages, spammers use the implicit trust between users as a cover. Researchers and Twitter itself have suggested many spam detection methods to address this growing issue [7]. Many of these methods rely on machine learning (ML) algorithms, which, after being trained, may classify users as spammers or not based on data gleaned from their accounts and tweets [1].

Twitter has transformed communication by offering a forum for the exchange of concepts, information, and viewpoints [2]. However, this widespread adoption has attracted spammers who spread irrelevant and harmful content, negatively impacting the user experience. To address this issue, the "Twitter Spam Detection" project seeks to develop a web application that is capable of precisely identifying and filtering tweet spam [7].

The user experience offered by the online application is seamless, starting with the registration stage when users can set up an account with their preferred username and password [2]. Users can view the "Twitter Spam Detection" page after registering successfully and logging in afterwards [11]. Here, they can quickly and conveniently enter tweets for examination by copying and pasting the content into the available field [13]. The "Analyse" button causes the application to analyse the tweet and determine if it is spam or not [9].

The project uses a variety of ML models, such as Naive Bayes, SVM, Logistic Regression, Random Forest, XGB Classifier, KNeighbours Classifier and Decision Tree, also it uses a variety of DL models, such as BiLSTM (a form of RNN), CNN and LSTM, to ensure reliable and effective spam identification [8]. To enhance the performance of these models, data preparation methods and NLP are also used [6].

Each ML model's performance is evaluated using metrics like accuracy, F1-score, and other important factors [16]. ROC curves and confusion matrices are presented for each model to enable illuminating visual comparisons [15]. With an accuracy rate of 82%, Naive Bayes stands out as the best classifier among the ML models. For real-time spam prediction, the Naive Bayes model is therefore saved in a pickle file and integrated into the web application [5].

The "Twitter Spam Detection" project aims to give users a trustworthy and useful tool to recognise and filter out spam messages, providing a quicker and safer Twitter experience [2]. The project helps to create a Twitter environment that is more reliable and authentic by utilizing the capabilities of ML models, deep learning, and NLP [18].

1.1 Aim and Objectives

Aim: The "Twitter Spam Detection" project's main objective is to create a reliable web application for identifying spam tweets on Twitter. The project seeks to develop a user-friendly solution that improves the overall Twitter experience by removing spam content by utilising a variety of machine learning (ML) models, deep learning techniques, data preprocessing, and natural language processing (NLP).

Objectives:
Web Application Development: Using Flask as the web framework and PyMySQL for safe database administration, create and implement a user-friendly online application. Users should be able to sign up, log in, and use the programme to scan tweets for spam.

Data Preprocessing and NLP: Apply data preparation methods to tweet content to make it analysis ready. Utilise NLP approaches to glean important data from tweet content to improve the accuracy of spam detection.

ML and DL Model Selection and Integration: Discover and use a variety of machine learning (ML) models and deep learning (DL) models, such as Naive Bayes, SVM, Logistic Regression, Random Forest, XGB Classifier, KNeighbours Classifier, Decision Tree, BiLSTM, CNN and LSTM. A web application for real-time spam prediction should be built using the top-performing model once each model's performance has been evaluated.

Analysis Method: Identify a reliable analysis technique to analyse user-provided tweets and effectively categorise them as spam or authentic material. Make sure the strategy can handle many users and tweets while also being quick, precise, and scalable.

Model Performance Evaluation: Assess the effectiveness of the ML models using appropriate metrics, such as accuracy, F1-score, and other pertinent parameters. For a better understanding of the models' advantages and disadvantages, create visualisations such as ROC curves and confusion matrices.

User Experience Enhancement: For seamless engagement with the online application, put your attention on offering a clear and user-friendly interface. Make sure users can quickly receive findings for the classification of spam after entering tweets for analysis.

Model Persistence and Efficiency: To make use of the online application effectively, save the best Naive Bayes model as a pickle file. To cut processing time and improve overall system efficiency, consider model persistence, and resource optimisation.

Approach of the Project:

The approach of the "Twitter Spam Detection" project includes a systematic and multifaceted strategy to meet the challenge of identifying and combating spam on the platform. This approach can be delineated into several key phases. The project started with an in-depth analysis of the problem, recognizing the importance of accurate spam detection to improve user experience. The next step involves collecting and pre-processing a diverse dataset of tweets, preparing textual content for further analysis. Comprehensive exploration of the data set uncovers patterns and distributions, ensuring a clear understanding of the data context.

Then, the project applies natural language processing (NLP) techniques to extract features from the processed text data efficiently. The focus of the project lies on the selection and implementation of a wide range of machine learning models including Naive Bayes, SVM, Logistic Regression, Random Forest, XGB Classifier, KNeighbours, Decision Tree, and a deep learning model – BiLSTM, CNN, LSTM. Each model goes through meticulous training, fine-tuning, and evaluation using metrics like accuracy, F1 score, and ROC curves.

In addition, a user-friendly web application developed using Flask facilitates user engagement through real-time spam detection, login, and registration interfaces. PyMySQL integration ensures secure database management, strengthens user authentication, and protects data. The machine learning model of choice, usually Naive Bayes due to its outstanding accuracy, is seamlessly integrated into the web application. Rigorous testing and quality assurance ensure the reliability and functionality of the application.

The results obtained from the deployed model must be carefully analyzed, providing a comparative assessment of the performance of the different models. This review involves comprehensive visualizations through graphs and charts, highlighting the model's strengths and areas for improvement. The project concludes with a concluding summary of the achievements and insights, highlighting the project's contributions to the field of spam detection. It is important that future research directions have been identified, opening avenues for further exploration, which may involve advanced deep learning architectures and user interface enhancements. Through this multifaceted approach, the "Twitter Spam Detection" project aims to improve the quality of interactions on the platform while paving the way for new advancements in this area.

Analysis Method:

The analytical procedure is taking user-submitted tweet input and running it through the Naive Bayes spam detection model. To extract significant features, such as tokens and cleaned text, from the input tweet, NLP-based data pretreatment techniques are used. The Naive Bayes model then takes these features as input and categorises the tweet as either spam or authentic right away.

Model Performance:

To determine the top classifier, the effectiveness of the ML models is carefully assessed. For each model, metrics like accuracy, F1-score, and other pertinent statistics are produced. In addition, visualisations are created to show the models' advantages and disadvantages, such as ROC curves and confusion matrices.

Naive Bayes, which has the best accuracy of the different ML models at 82%, is the model of preference for spam prediction. The web application incorporates this model, saves it in a pickle file, and uses it to provide users with real-time spam detection.

1.2 Structure of Dissertation

The "Twitter Spam Detection" dissertation is organised to provide a complete analysis of the challenges involved in preventing Twitter spam. The introduction sets the stage for comprehension right away by outlining the project's goals and describing several methods, including ML models, deep learning, data preprocessing, and NLP. The talk also highlights the foundational technologies: Flask for web development and PyMySQL for database complexity.

Going even further, (Section 2), "Background and Related Work," provides a critique of the paradigms now in use for Twitter spam identification. Beyond simple explanations, it delves into the drawbacks of conventional approaches while highlighting the innovative strategy the project employs to boost detection accuracy.

After that, (Section 3), "Methodology," meticulously analyses the project's key procedures. The first section, "Data Acquisition," describes the sources and methods for the datasets. This leads into the section on "Data Pre-processing," which focuses on optimisation strategies, specifically the part NLP plays in feature extraction. The

methodology then switches to the user-facing side of things, explaining how the web application was made, highlighting user registration, login procedures, and spam detection capabilities all of which were made with HTML and CSS. Later subsections, such "Database Management," go into more detail about the usefulness of PyMySQL and SQL
file operations. The variety of models used is highlighted in chapter 3.5 "Machine Learning and Deep Learning Models," particularly the RNN-centric architecture of BiLSTM. The chapter "Model Evaluation" provides performance measurements and visualisation tools before discussing model generalisation and the seamless web integration of the Naive Bayes model for real- time predictions, which has an amazing 82% accuracy.

The project's design aspects are covered in detail in (Section 4), "Design and Implementation," which also includes flowcharts to further improve understanding. Here, we also describe the implementation and optimisation of each algorithm.

There is a thorough discussion of the evaluation criteria utilised in (Section 5), "Results and Analysis," highlighted with ROC curves. A thorough description of the Naive Bayes model's superior performance concludes this section

The dissertation's conclusion, (Section 6), summarises the project's accomplishments while highlighting the potential of the created web application for real-time spam predictions. It also sets the path for further investigation by indicating prospective study avenues and ways to improve Twitter's spam detection systems.

2 Background and Related Work

This section presents Twitter's features and its approach to dealing with spam.

2.1 Features of Twitter

The ability to "follow" other accounts on Twitter is available to users [17]. It is possible for a user to not be following one of his followers on this social media site because user engagement there is bidirectional rather than having relationships that are only one way [8]. The user has the choice to "like" or "retweet (RT)" a tweet to share it with his "followers" [21]. The relationships among Twitter users are displayed in Figure 1 [5]. Each user on Twitter has their own unique username, and users can mention other users in tweets by adding their usernames to the tweet with the "@" symbol in front of them, which is referred to as a "mention" on Twitter [10]. Users are instantly notified when one of his tweets is mentioned, liked, or RTs [13].

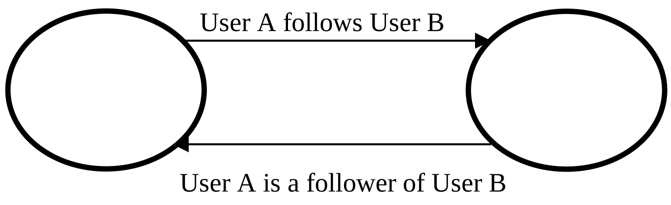


Figure 1. The relationship between the users in twitter [8].

Another aspect of Twitter is the ability for users to organise their interests by creating user- created public or private lists that compile persons with related or complimentary interests. Like how users can be added to or removed from lists they own; users can manage lists by adding users to or deleting users from them. The lists that the user subscribed to gounder the "subscribed to" category, while the lists that their owners added the user to fall under the "member of" category are shown in Figure. 2 [4].

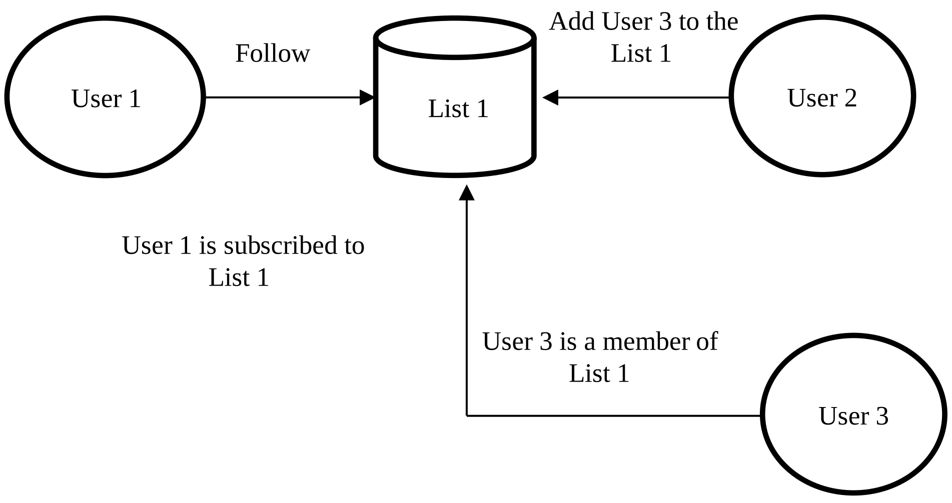


Figure 2. The relationships between lists and users [8].

2.2 How Twitter Deals with Spam

To maintain a spam-free environment, Twitter employs both manual and automatic services to combat spammers [8]. By visiting their profile pages, spammers can be manually reported on Twitter, which also gives users the ability to report accounts by choosing a cause [7]. Another strategy that is commonly recommended in the literature is to report spammers to the official "@spam" account, but Twitter recently reported that this approach is no longer effective. Wang also asserts that spam and hoaxes inappropriately use this strategy [11]. It would be hard to identify each spammer due to the enormous number of users and the time-consuming nature of these manual processes [12]. Twitter uses a number of factors, such as (1) posting identical messages across multiple accounts or multiple identical messages on one account, (2) following/unfollowing many accounts swiftly, (3) having a high volume of spam complaints filed against the account, (4) aggressively liking, following, and retweeting, (5) posting malicious links, (6) posting tweets that are primarily made up of links rather than personal updates, and (7) posting unrequested tweets [24].

2.3 Features of Twitter Spam Detection

The characteristics of Twitter spam detection are divided into three categories: (1) account-based features, (2) tweet-based features, and (3) sender-to-recipient connection features. These characteristics serve as the mainframes for the characteristics employed in the connected literary works [9]. The following subsections discuss each feature category [3].

2.3.1 Account-based Features

Analysis of Twitter accounts with the characteristics indicated in Table 1 can help identify spammers [1]. In terms of spam identification, several of these features such as biography, location, homepage, and creation date are meaningless because they are user-controlled [24].

Table 1. Account-Based Spam Detection Features [8].

Feature	Description	Is User-controlled?
Username	The unique identifier of the account	Yes
Biography	The biography of the account	Yes
Profile photo	The profile photo of the account	Yes
Header photo	The header photo of the account which is displayed at the top of the profile	Yes
Theme color	The theme color choice of the account	Yes
Birth date	The birth date information of the account	Yes
Homepage	The website of the account	Yes
Location	The location of the account	Yes
Creation date	The date the account is created	Yes
Number of tweets	Total number of tweets the account has	No
Number of following	Total number of accounts the account follows	No
Number of followers	Total number of followers the account has	No
Number of likes	Total number of likes the account's tweet have	No
Number of retweets	Total number of retweets the account's tweet has	No
Number of lists	Total number of lists the account has	Yes
Number of moments	Total number of moments the account has	Yes

2.3.2 Tweet-based Features

To get attention, spammers frequently bombard legitimate users with many unwanted tweets [8]. It is possible to identify spammers by examining their tweets [8]. The goal of Twitter is to offer users with a spam-free environment; hence this is required to separate spam tweets from real ones [8]. The details in Table 2 are present in each tweet [8].

Table 2. Tweet-Based Spam Detection Features [8].

Feature	Description	Is User-controlled?
Sender	The sender of the tweet	Yes
Mentions	The mention(s) used in the tweet	Yes
Hashtags	The hashtag(s) used in the tweet	Yes
Link	The link used in the tweet	Yes
Number of likes	The number of likes the tweet has	No
Number of retweets	The number of retweets the tweet has	No
Number of replies	The number of replies to the tweet has received	No
Sent date	The date tweet is sent	Yes
Location	The detected location of the place the tweet is posted	Yes

When spammers' actions are examined in relation to tweet-based traits, the following findings are discovered:

- Spammers frequently employ links to take reputable individuals to their malicious websites.

- Spammers frequently use many mentions to draw in more reputable individuals.

- Spammers frequently use many hashtags, especially popular ones, to reach a larger audience.

- Compared to authorised users, the quantity of likes and retweets received by spammers' tweets is significantly lower because they are unsolicited.

2.3.3 Graph-based Features

Twitter is a network of users with relationships between them and tweets [7]. This structure can be represented as a graph. For the graph model, users and tweet can be represented as nodes and relationships can be represented links between nodes [4]. These relationships show how the tweet's sender and mentions are connected to each other. Also, these relationships are clear indicators of legitimate conversations. By constructing a graph model to represent users and their relationships, the distance between the tweet's sender and mentions can be calculated for spam analysis [1]. Graph-based features are listed in Table 3.

Table 3. Graph-Based Features [8].

Feature	Description	Is User-controlled?
Distance	The length of the shortest path between users	No
Connectivity	The strength of the connection	No

When spammers' actions are examined in the context of graph-based features, the following findings are discovered:

- A spammer is more away from a legitimate user than two legitimate users are from one another [3].

- Compared to connectivity between two legitimate users, connectivity between a spammer and a valid user is stronger [6].

- Due to their difficulty in manipulation and lack of user control, graph-based features offer the most reliable performance to identify spam and spammers [5].

3 Methodology

3.1 Data Acquisition

The "Data Acquisition" procedure of the Twitter spam detection project entails acquiring the dataset utilised for training and assessing the machine learning models. To ensure the efficacy and generalisation of the spam detection algorithms, the dataset should be varied, representative, and well labelled.

Finding appropriate sources from which to obtain Twitter data is the first step. This might contain open APIs from Twitter or third-party datasets that have been made accessible for research. While using Twitter data, it is essential to follow ethical standards. The procedure for using data must respect user privacy and adhere to Twitter's terms of service and standards. The next stage is to use tweets for the "Quality" and "Spam" categories when the data source has been identified. This entails using Twitter APIs or web scraping methods to retrieve tweets based on predetermined criteria or hashtags associated with spam and non- spam content. The dataset needs to be sizable enough to offer adequate training data to the machine learning models. To avoid any class imbalance difficulties during model training, it is critical to guarantee a balanced representation of both "Quality” and "Spam" tweets. After data collection, the raw tweet text may need to be preprocessed to remove noise, such as special characters, emojis, URLs, and extraneous whitespace. Normalize the data, tokenization and stemming are two text normalisation approaches that may be used. Each tweet needs to be identified with the appropriate "Type" (Quality or Spam). Common methods for getting the ground truth labels include manual labelling and using datasets that have already been labelled and are in the public domain. The dataset is divided into training, validation, and testing sets. Machine learning models are trained using training sets, validation sets are used to help with model selection and hyperparameter tuning, and testing sets are used to evaluate the performance of the finished model. Data integrity checks must be carried out to ensure there are no missing values or discrepancies in the dataset before moving further with model training. To prevent skewed or incorrect model results, data quality assurance is crucial. By meticulously adhering to these procedures during the data collection phase, the project may develop a high-quality and diverse dataset that serves as the foundation for training the Twitter spam detection algorithms. To obtain strong and trustworthy spam detection results, a precise and balanced dataset must be used.

Applied Data “Tweet” and “Type” are the two key features of the data file used for Twitter spam identification. Each entry in the data file corresponds to a tweet from

Twitter, and the "Tweet" feature includes the tweet's actual textual content. Each tweet is categorised by the "Type" feature into one of two categories: "Quality" or "Spam."

Tweet: The tweets that were acquired from Twitter are contained in this feature's text content. It consists of the user-posted content, which may be of varied lengths and contain hashtags, mentions, URLs, emojis, and other Twitter-standard components. Input for the spam detection models in this feature comes from the text data.

Type: This characteristic is the target variable used to determine if a tweet is considered "Quality" or "Spam." The "Type" feature is a categorical binary variable in which:

"Quality": Represents tweets that are real, spam-free, and regarded as having high-quality content. These tweets include pertinent and useful information and are legitimate posts made by actual persons.

"Spam": Represents tweets that are deemed spam and may be irrelevant or deceptive information, ads, harmful links, or automatically created content from a bot. These tweets have the purpose of disseminating offensive content or advertising certain goods or services.

Using the dataset containing these two features, the Twitter spam detection models are trained and evaluated. The models learn from labelled examples during the training phase where each tweet is paired with its matching "Type" label (Quality or Spam). The algorithms can anticipate the "Type" of new, unseen tweets to categorise them as either real quality content or spam once they have been trained.

The calibre and variety of the training data are key factors in the spam detection models' efficacy. It is essential to have a well-annotated dataset that contains a representative mixture of spam and actual tweets so that the models can generalise and successfully identify spam in practical settings. The aim of utilising this data set is to create adependable and effective system that can automatically recognise and filter out spam tweets, hence enhancing the general user experience on Twitter.

3.2 Data Pre-processing

Data to ensure that each tweet is distinct and prevent duplication during analysis, pre-processing techniques are applied in the dataset, which involves removing any duplicate rows. Any rows that have missing values are handled by dropping them, ensuring that the data is complete and available for processing. The dataset's "Type" column has the headings "Quality" and "Spam." A dictionary is defined to map "Quality" to 1 and

"Spam" to 2 so that they can be transformed into numbers for categorization. After then, these numbers are put in the 'Type' column. The most common words in both "Real" and "Spam" tweets are visualised using word clouds. Understanding the typical vocabulary for each class is aided by this.
The text of the tweets passes through many pre-processing processes, including:

- **Punctuation Removal:** Regular expressions are used to remove punctuation and special characters from the tweets, leaving only alphanumeric letters and spaces.

- **Tokenization:** The tweets are broken down into their component words as a first step in processing.

- **Stopword Removal:** The tokenized words have all common English stopwords such as "the," "is," and "and" removed. These phrases can be safely omitted because they do not significantly add to the tweet's message.

- **Lemmatization:** Using "WordNetLemmatizer", which converts each word to its lexical or root form (lemma), the words are lemmatized. By doing this, the dimensionality of the data is reduced and the sense of the words is preserved.

- **Data Splitting:** Using "train_test_split" from scikit-learn, the pre-processed data is finally divided into training and testing sets. This makes it possible to train the models on a subset of the data and test them on new data to determine how well they generalise.

The methods used in data pre-processing to convert text input into a format that machine learning and deep learning models can use to efficiently detect spam tweet.

3.3 Web Application Development

Flask, a micro web framework, is used to create the application. Flask makes it simple to construct web apps by enabling routing, request handling, and templating. To render dynamic material and show it to users, the application makes use of HTML templates. Web pages can be dynamically updated and have uniform layouts thanks to templates. To communicate with the MySQL database, PyMySQL and MySQLdb libraries are used. To manage user accounts, the database stores the username and password provided during user registration. By entering a username and password, users of the programme can register. Then, using SQL queries, the database is used to store the user's registration information. The login information for registered users is available. The application checks to see if the supplied username and password match the database records. If the login process is successful, the user is forwarded to a page that detects spam; if not, an error notice is displayed. The spam detection website allows visitors to enter a tweet after logging in. The pre-processing of the input tweet entails cleaning, tokenization, stopword removal, and lemmatization, just like the pre-processing of data prior to model training. Next, the loaded machine learning model receives the pre-processed tweet and makes a prediction. The model makes a prediction as to whether a tweet is genuine or spam (1 for genuine, 2 for spam).

When a tweet is detected as spam or not, the web application shows the prediction result on the spam detection website. Utilising HTML and CSS, the front end of the web application is created. Dynamic content can be injected into the HTML pages because the web pages are rendered using Jinja templates. With the help of their web browsers, users can interact with the web application by utilising the "app.run()" method to launch it.

In conclusion, the online application offers users a simple user interface through which they can register, log in, and determine whether a specific tweet is genuine or spam. This Twitter spam detection web application can be created and deployed thanks to the Flask framework, PyMySQL.

3.4 Database Management

A MySQL database named "ddbb" has the structure and data represented by a PHPMyAdmin SQL dump. In the web application, user registration and login are handled by a single table called "user_register" that is present in the database. Information about user registration is kept in the database in a single table called user_register. There are two columns in the table:

user: This column stores the username of the registered user.
password: This column stores the password of the registered user.

The table makes advantage of MySQL's popular InnoDB storage engine. The table's default character set is utf8mb4, which allows a wider variety of characters, including emoji and other special characters. The INSERT commands that fill the user_register table with test data are also included in the SQL dump. With their username and password, each row represents a registered user. Commands for managing transactions and adjusting the database's time zone are included in the SQL dump.

3.5 Machine Learning and Deep Learning Models

Naive Bayes, SVM, Logistic Regression, Random Forest, XGB Classifier, K-Nearest Neighbours Classifier, and Decision Tree are just a few of the machine learning models we utilise. Also, a variety of deep learning models, BiLSTM (Bidirectional Long Short-Term Memory), CNN and LSTM a deep learning model, is also used. Using the preprocessed data, these models are trained and assessed.

Naïve Bayes Classifier: The Multinomial Distribution defines the probability of identifying counts among distinct categories, hence Multinomial Naive Bayes is used for features that imply counts or count rates. Text classification is one use of multinomial Nave Bayes where features are connected to word counts or frequency distributions inside the target documents [14].

In a multinomial event model, samples (feature vectors) describe the frequencies at which the multinomial has created events (*P1…. Pn*), where {\displaystyle p_{i}} shows the likelihood that event i will happen.The histogram created by the feature vector {\displaystyle \mathbf {x} =(x_{1}, \dots,x_{n})}*X =(X1….Xn*) represents the frequency of occurrence of event i in a particular instance. The most typical event model for categorizing documents is this one. The probability of seeing a histogram x is as follows:

$$P(X|C_k) = \frac{(\sum_{i=1}^{n} x_i)!}{\prod_{i=1}^{n} x_i!} \prod_{i=1}^{n} PK_i^{x_i} \qquad (1)$$

Scikit-learn's MultinomialNB class was utilised for the Naive Bayes model. Using NLTK, the textual data underwent a thorough preprocessing routine that included tokenization, stopword removal, and lemmatization. The CountVectorizer was used to numerically transform the cleaned data, and after that, the TfidfTransformer was used to normalise it for term frequency-inverse document frequency (TF-IDF) scaling. On this improved dataset, the Naive Bayes model was trained, and its performance was evaluated. Scikit- Learn's classification_report function provided performance measures, such as precision, recall, F1-score, and support for each category (Quality and Spam) [14].

Additionally, scikitplot was used to make it easier to visualise the confusion matrix and provide the true positive, false positive, true negative, and false negative counts.

Logistic Regression: The Scikit-Learn Logistic Regression class was used to create the Logistic Regression model. To ensure that the data were handled consistently by both models, the preprocessing processes for the data were identical to those used by the Naive Bayes classifier. The data were prepared using CountVectorizer and TfidfTransformer, and then the Logistic Regression model was trained, and its accuracy was assessed. Once more, the classification report included key metrics for the Quality and Spam classes, and the confusion matrix was graphically represented with scikit-plot to provide a thorough analysis of the model's effectiveness in separating the two groups [14].

In logistic regression, the sigmoid function is a binary classification function. provided a starting feature vector x, it outputs a classification probability for the provided text. The following is its formula:

$$P = \frac{e^{a+bX}}{1+e^{a+bX}} \tag{2}$$

where P is the probability of a 1 (the proportion of 1s), a and b are model parameters, and e is the base of the natural logarithm. The probability P is generated when X is 0, and the speed at which the probability shifts when X is changed by a single unit depends on the value of b [14].

Decision Tree Classifier: A decision based on a feature is represented by a branch of a decision tree, and a class label is represented by a leaf node of a decision tree. Decision trees are excellent for both classification and regression applications because they are simple to comprehend and interpret.

Decision trees are supervised classifiers that can be used to solve classification and regression problems; however, they are most frequently applied to classification problems. Internal nodes provide dataset attributes, branching indicate decision criteria, and leaf nodes provide the final classification in this tree-structured classifier. The decision node and the leaf node are the two nodes in the decision tree. A Leaf Node is the outcome of a decision and has no more branches, but a Decision Node is used to make decisions and has numerous branches [14]. Entropy determines how a Decision Tree decides to divide up the data. It has an impact on how a decision tree's borders are constructed. The following is the formula for it:

$$H(s) = -\text{ probability of } \log_2 (p+) - (-\text{probability of } \log_2(p-)) \tag{3}$$

where (p+) represents the percentage of the positive class and (p-) represents the percentage of the negative class [14].

Support Vector Machine (SVM): Each piece of data (where n is the number of features) correlates to a point in n-dimensional space, and each feature's value to the SVM algorithm's value for a specific position. To address two-group classification challenges, SVMs have supervised machine learning models that employ classification techniques [6]. By providing training data for each category that has been labelled, SVM models may categorise new texts. Compared to more advanced techniques like neural networks, they are quicker and perform better with smaller amounts of data (in the hundreds). Text classification problems are particularly well suited for the method because there are typically only a few labelled examples available. Through complex data manipulations, the SVM algorithm uses a technique known as the kernel trick to transform low input dimensions into higher input dimensions. This conversion of a non-separable problem into a separable one is accomplished by the SVM.

K-Nearest Neighbors (KNN): K-Nearest Neighbours is a non-parametric, slow-learning algorithm. A data point is categorised based on the dominant class among its k- nearest neighbours in the feature space. When the data cannot be separated linearly, KNN is easy to apply and effective.

XGB Classifier (Extreme Gradient Boosting): The XGB Classifier is a gradient boosting-based ensemble learning technique. It sequentially creates several weak learners (usually decision trees), with each new learner correcting the mistakes of the prior one. High performance, scalability, and the capacity to manage complicated datasets are attributes of the XGB Classifier.

Random Forest: During training, Random Forest generates numerous decision trees and produces the mode of the classes (classification) of the individual trees. To reduce overfitting and increase accuracy, each tree is constructed using a random subset of the data and characteristics.

Random Forest is a versatile and user-friendly method that uses supervised learning and can be applied to both regression and classification applications. Random Forests create decision trees from randomly selected data samples, get predictions from each tree, and then let users vote on the best result. It is also possible to evaluate the feature's advantage precisely. It's produced by the following formula:

$$ni_j = w_j C_j - W_{left(j)} C_{left(j)} - W_{right(j)} C_{right(j)} \qquad (4)$$

where left(j) is the child node from the left split on the node j and right(j) is the child node from the right split on the node j, and ni_j is the importance of node j, w_j is the weighted number of samples that reach it, C_j is the node j impurity value [14].

BiLSTM (Bidirectional Long Short-Term Memory): A sequence processing model called a bidirectional LSTM consists of two LSTMs, one of which advances the input and the other of which reverses it. Because of BiLSTM, the network has access to additional data, which provides the algorithm with a richer context.

Recurrent neural network (RNN)architectures come in several forms, including BiLSTM. With the input being processed both forward and backward, it is intended to capture long- term dependencies in sequentialdata. As the context of the input is essential for precise predictions, BiLSTM is frequentlyemployed in natural language processing applications including sentiment analysis, machine translation, and spam detection.

Long Short-Term Memory (LSTM) Model: LSTMs are made to capture data's long-term dependencies and sequential information. As is the case with tweets, they are especially well suited for text sequence analysis [22]. The memory cells in LSTMs have the capacity to store and update data throughout time steps, enabling them to recognize word context and relationships. The project makes use of LSTMs to discover patterns in the word order within a tweet that may be a sign of spam [10]. They can recognize individual words as well as how those words are related to one another within the context of the tweet.

Convolutional Neural Network (CNN) Model: The ability of CNNs to excel at tasks involving image identification is well recognised. However, by considering the text as an image with one dimension, they can also be used with text data [21]. As part of the project, a CNN may learn to recognize regional trends and elements in the textual data that point to the presence of spam content. Convolutional layers are used in succession with pooling layers to achieve this. The text data is scanned by the convolutional layers, which look for patterns like word or phrase combinations that might be indicative of spam [12].

In conclusion, these ML models and the deep learning model are effective tools for categorization tasks, such as Twitter spam detection. Each model may perform differently and have unique advantages depending on the nature of the data and the needs of the application [13]. To choose the best model for a certain task, experimentation and model evaluation are essential [23].

Among the evaluation criteria used in the proposed study are accuracy, recall, negative recall, precision, and F1-score [14].

Here is how accuracy is calculated:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{5}$$

How many data values are successfully predicted is shown by the accuracy metric [14].

Sensitivity (or recall) is the proportion of test case samples out of all the positive classes that are correctly predicted. As a result, it is determined:

$$\text{Sensitivity} = \frac{\text{Number of True Positives}}{\text{Number of True Positives}+\text{Number of False Negatives}} \tag{6}$$

Specificity (also known as negative recall) is calculated as the proportion of test case samples that are correctly predicted among all the negative classes [14]. It is calculated as follows:

$$\text{Specificity} = \frac{\text{Number of True Negatives}}{\text{Number of True Negatives}+\text{Number of False Positives}} \tag{7}$$

The precision measure determines the number of positive samples among all the anticipated positive class samples as follows:

$$\text{Precision} = \frac{\text{Number of True Positives}}{\text{Number of True Positives}+\text{Number of False Positives}} \tag{8}$$

The F1 score is the harmonic mean of Precision and Sensitivity. It is also known as the Sorensen-Dice Coefficient or the Dice Similarity Coefficient [14]. One is the optimum number. The computation of the F1- score is shown as follows:

$$\text{F1 – score} = 2*\frac{\text{Precision}*\text{Sensitivity}}{\text{Precision}+\text{Sensitivity}} \tag{9}$$

3.6 Model Evaluation

Model evaluation is a critical phase in the project "Twitter spam detection" to evaluate the efficiency and performance of the various machine learning and deep learning models used for spam detection. The major objective of model evaluation is to determine which model has the highest accuracy and other desirable metrics in terms of identifying tweets as spam or non-spam (genuine). Let's explore the specifics of the model assessment procedure:

Data Splitting: A training set and a test set are created from the dataset before the model is evaluated. In comparison to the training set, the test set is used to evaluate how effectively machine learning models perform on fresh data.

Data Preprocessing and NLP Techniques: To prepare the tweet text for modelling, text data preprocessing techniques like cleaning, tokenization, and lemmatization are used. Text input that has been previously processed is transformed into numerical features that machine learning models can process using NLP techniques like CountVectorizer and TfidfTransformer.

Model Training and Evaluation: A deep learning model (BiLSTM) is also trained on the tokenized and padded sequences. Several machine learning models are trained on the preprocessed training data. On the test set, the trained models' performance is then evaluated. Accuracy, F1-score, precision, and recall are evaluation metrics. These measurements shed light on how well the models categorise tweets as legitimate or spam.

ROC Curve and AUC: For the BiLSTM deep learning model, the Receiver Operating Characteristic (ROC) curve is displayed. The ROC curve demonstrates the trade-off between the true positive rate (sensitivity) and the false positive rate (1-specificity) for various threshold settings. The Area Under the ROC Curve (AUC) is calculated to evaluate the model's overall efficacy. A higher AUC value indicates better model performance.

Confusion Matrix: The confusion matrix provides a clearer picture of the models' performance. It displays the numbers of accurate predictions—true positive, accurate negative, accurate positive, and accurate negative. Other metrics like precision and recall for each class (spam and real) can be calculated using the confusion matrix.

Model Comparison and Selection: The model with the highest accuracy, F1-score, and precision for the spam class is found by comparing the evaluation results from many models. The Naive Bayes model, which has an accuracy of 82%, is the best-performing model according to the study.

Model Persistence: The best model (Naive Bayes) is chosen, and it is then saved into a pickle file for subsequent usage. With no need for retraining, the web application can now load the trained model and make predictions using fresh twitter data.

Overall, a model's ability to effectively detect spam tweets is greatly influenced by its evaluation and its results can be shown in chapter 5. It aids in choosing the most suitable model for the application and guarantees that the deployed model operates properly on user-unseen data. The chosen Naive Bayes model is subsequently included into the web application to give registered users with real-time spam detection.

3.7 Model Generalization

In machine learning, generalization refers to a model's capacity to generate reasonable results for unobserved data given its training on a specific dataset. It demonstrates how effectively the model can change to match new data without overfitting or underfitting. We can infer the following from an analysis of the models offered:

Traditional Machine Learning Models:
Naive Bayes: The accuracy of Naive Bayes is 82%, which is a respectable performance [14]. It is surprising but admirable that it has achieved such performance given its feature independence assumptions, indicating that the features may be independent in this situation [14].

Logistic Regression & SVC: Both models are generally 81% accurate. For the given situation, their linear decision bounds appear to be effective. Their reliable precision, recall, and f1-scores point to a balanced performance free of a discernible bias towards any class [14].

XGB Classifier & RandomForestClassifier: The accuracy of these ensemble models is 76% and 78%, respectively. Although ensemble approaches often have superior generalization because they can capture complicated correlations in the data, they may have a tiny lag in this area because of data variability or the need for hyperparameter adjustment [14].

KNN: The performance of this model, with an accuracy of 49%, suggests potential underfitting [14]. The fact that it classified almost all occurrences as "Real" suggests that it may be overly dependent on the training data, which could cause it to incorrectly classify most of the test data [14].

Decision Tree Classifier: It generalizes reasonably well with a balanced accuracy of 75%. But sometimes noise from the training data can be captured by decision trees, which may be the case in this instance [14].

Deep Learning Models:
CNN (Convolutional Neural Network): The CNN model exhibits good generalization skills with an AUC of 0.87 and accuracy of 79%. CNNs, which are typically used for

classifying images, appear to be suitable for this classification challenge, indicating that the data may have patterns that are best captured via convolutional processes [14].

LSTM (Long Short-Term Memory): The LSTM model raises questions due to its accuracy, which is like KNN's but with an AUC of 0.50. This performance raises questions about possible problems like insufficient training, poor architecture, or inappropriate data representation as LSTMs are known to capture sequential data quickly [14].

BiLSTM: The BiLSTM appears to have captured the sequence data more well than the standard LSTM, with an accuracy of 78.58%. It's possible that the bi-directional nature offers a broader background, improving generalization [14].

While most models performed admirably in the race for generalization, the CNN and BiLSTM models emerged as possible leaders in terms of robustness and reliability [14].

3.8 Web Application Integration

The trained machine learning model (Naive Bayes) must be integrated into the web application for real-time spam detection. The Flask web framework, a compact and adaptable Python web development framework, is used to build the online application. The online application framework Flask makes it simple and quick for developers to build websites. The MySQL database can be accessed using PyMySQL. User registration information is stored in the database during the registration process, including usernames and hashed passwords. Users can register for the online application by going to a registration page and entering a username and password to create an account. In the MySQL database, the programme uses PyMySQL to securely store the hashed username and password of each new user. Using their login information (username and password), registered users of the web application can access the login page. As soon as a user logs in, the programme uses PyMySQL to check the user's credentials against values stored in the MySQL database. To execute tweet spam detection using the trained Naive Bayes model, users are sent to the spam detection page after successfully logging in. The user interface for this page offers a text input box where visitors can copy and paste the text of a tweet they wish to analyse. When a user selects the "Analyse" button, the web application analyses the tweet text entered and uses the same data pretreatment methods as during model training. For the trained Naive Bayes model to produce in-the-moment predictions, the preprocessed tweet text is then input into it. If a tweet is categorised as spam or non- spam (genuine), the model generates a prediction label expressing this.

The web application presents the user with the prediction findings in an intuitive way. It lets the user know if the analysed tweet is spam or authentic. Based on the detection results, the user can then take the relevant action. When evaluating models, the trained Naive Bayes model is placed into a pickle file. To prevent retraining the model each time the programme runs, the model is loaded from the pickle file into memory when the web application first launches. The web application's front end is created with HTML and CSS to offer a simple and eye-catching user interface. Users should have no trouble navigating the programme and comprehending the results of the spam detection procedure thanks to the design.

4 Design and Implementation

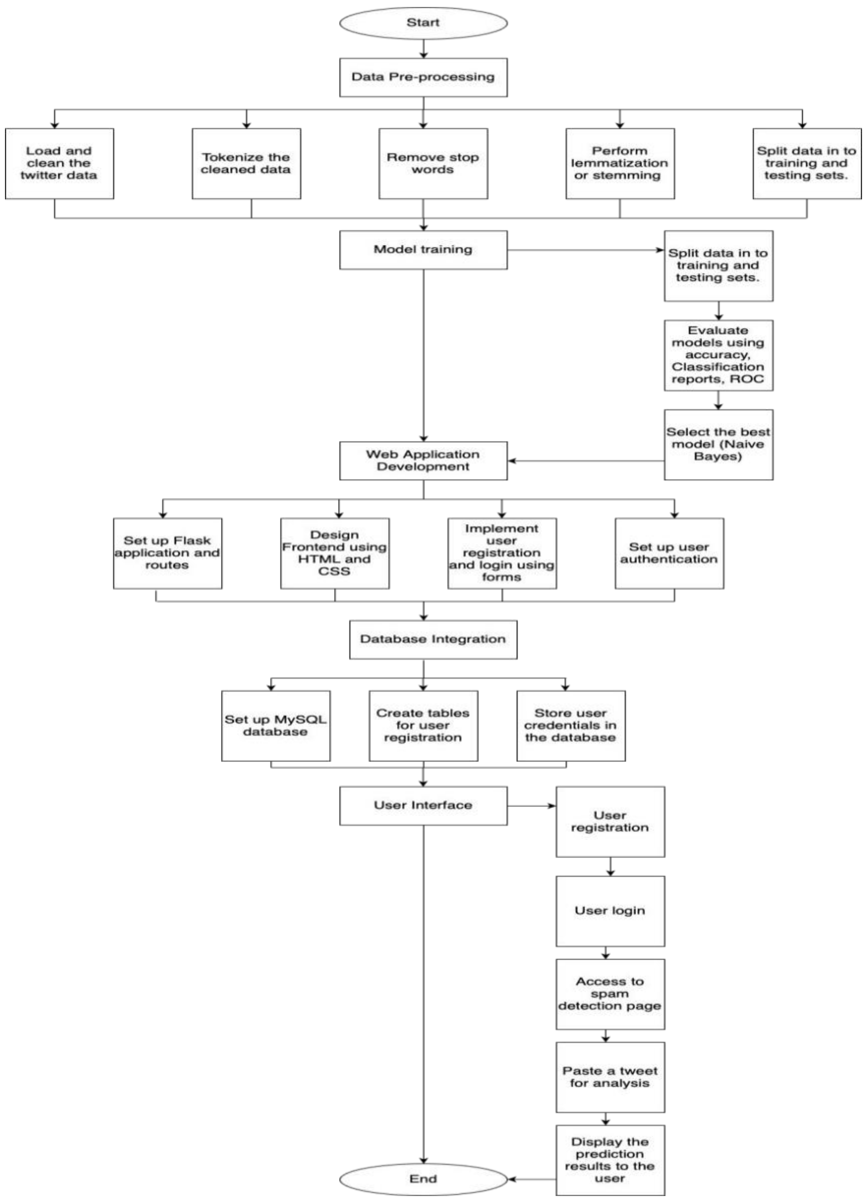
Twitter spam detection begins with the loading and cleaning of the Twitter data during the data preprocessing stage. Tokenization is used during cleaning to separate the data into single words or tokens, and stopwords that don't add to the sense of the sentences are then eliminated. The data is then divided into training and testing sets after lemmatization or stemming is used to break down words into their simplest forms.

After preprocessing, we proceed to the model training stage where we train different deep learning and machine learning models like Naive Bayes, Logistic Regression, XGBoost, BiLSTM, CNN and LSTM. Accuracy, classification reports, confusion matrices, and ROC curves are used to measure the performance of each model. The Naive Bayes model, with the best performance among these models, is chosen for further processing.

The trained models are saved to.pkl files in the model persistence step to guarantee that they may be used for prediction at runtime. Once needed for prediction tasks, these files are then loaded.

The next stage is the development of the web application. We configure the Flask application and specify the routes. We use HTML and CSS to create the application's front end and implement forms for user registration and login. To guarantee security, user authentication is also set up at this point.

The application incorporates a MySQL database to securely store user credentials. The user credentials submitted throughout the registration procedure are stored in the tables we generate for user registration.



Start
Data Pre-processing
Load and clean the twitter data
Tokenize the cleaned data
Remove stop words
Perform lemmatization or stemming
Split data in to training and testing sets.
Model training
Split data in to training and testing sets.
Evaluate models using accuracy, Classification reports, ROC
Select the best model (Naive Bayes)
Web Application Development
Set up Flask application and routes
Design Frontend using HTML and CSS
Implement user registration and login using forms
Set up user authentication
Database Integration
Set up MySQL database
Create tables for user registration
Store user credentials in the database
User Interface
User registration
User login
Access to spam detection page
Paste a tweet for analysis
Display the prediction results to the user
End

Figure 3. System design flowchart

Figure 4. Shows the register page of the Twitter Spam Detection Web application.

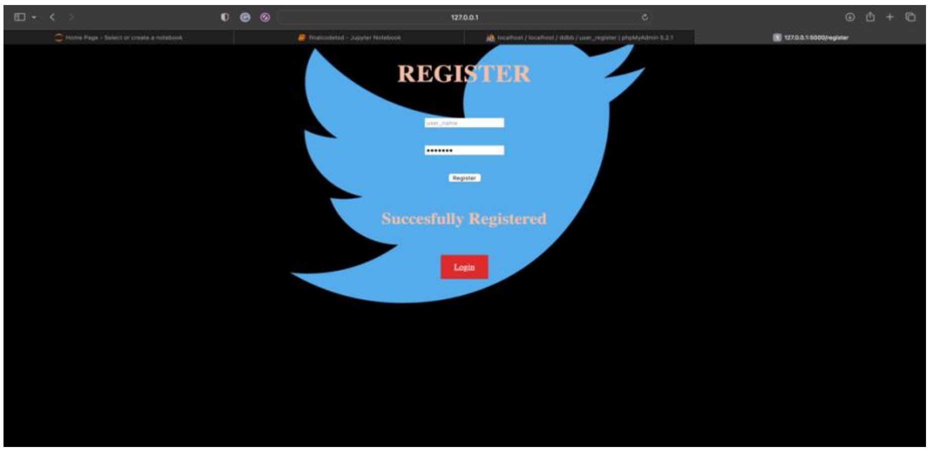


Figure 4

Figure 5. Shows the login page of the Twitter Spam Detection Web application.

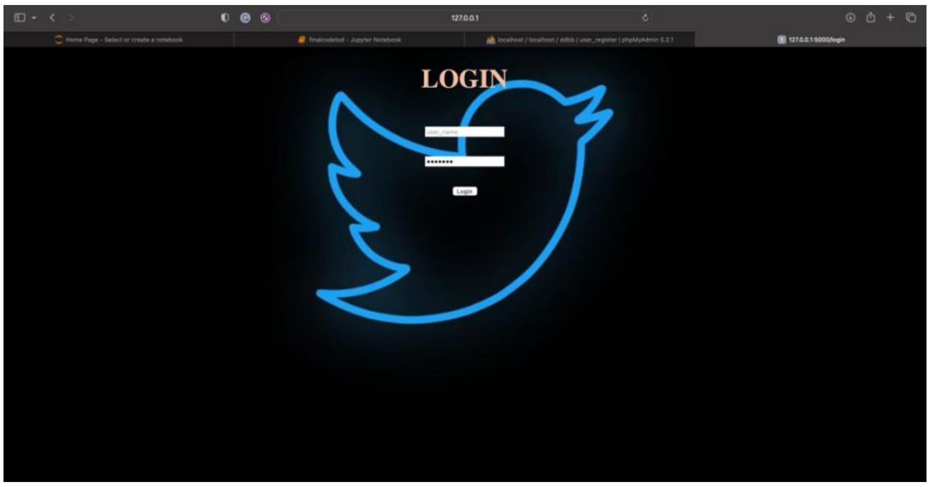


Figure 5

Figure 6. shows the Analysis page of the Twitter Spam Detection Web Application

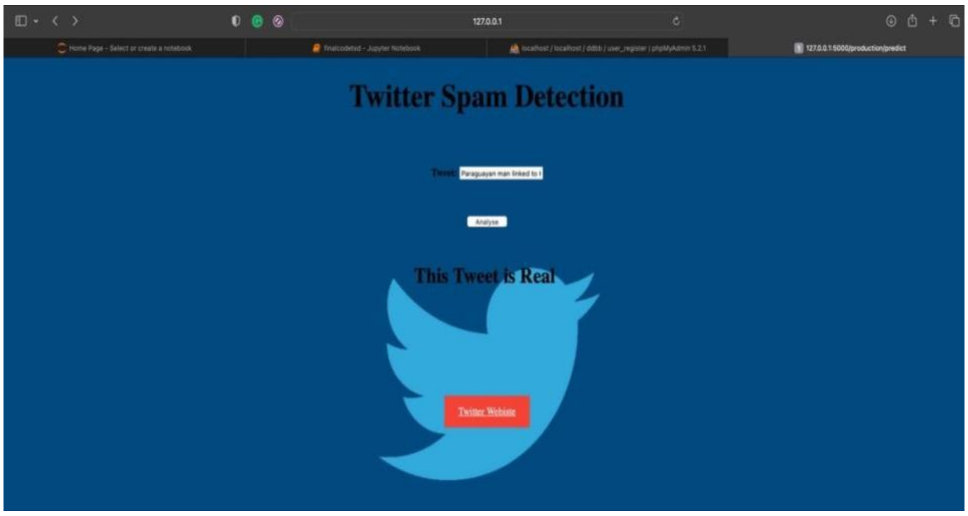


Figure 6

Figure 7. shows the Analysis page of the Twitter Spam Detection Web Application

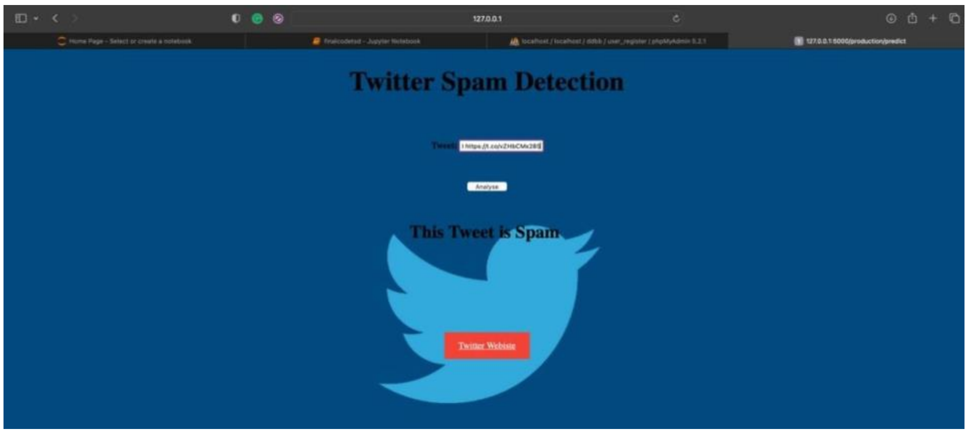


Figure 7

4.1 Challenges

Data Collection and Preprocessing: To train the models, it can be difficult to compile a broad and representative dataset. Twitter data can be chaotic and noisy, necessitating meticulous pretreatment to clean and tokenize the content. The CSV data were read and preprocessed by the project using the panda’s package. It carried out fundamental cleaning by deleting duplicate rows and empty rows. Special characters were eliminated using regular expressions. Emojis, URLs, and hashtags were not explicitly addressed in the provided code, but these issues might be more successfully handled by employing advanced preparation techniques, such as using libraries like emoji for emoji translation and Beautiful Soup for parsing HTML text.

Imbalanced Data: Model performance may be impacted by unbalanced classes, when one class (spam) vastly outnumbers another (non-spam). To lessen the imbalance and assure fair evaluation, strategies employing various evaluation metrics (precision, recall, and F1-score) are used.

Feature Extraction for NLP: For a model to be effective, significant features must be extracted from text data. Challenges include choosing the right text vectorization methods (such as TF-IDF, word embeddings), and controlling the trade-off between dimensionality and information loss. Basic tokenization and stopword elimination were carried out by the code. More semantic context may be captured using advanced techniques like word embeddings (Word2Vec, GloVe), spaCy for enhanced tokenization, and n-gram models. The scikit-learn tools TfidfVectorizer and CountVectorizer can be used to extract feature vectors from text data.

Model Selection and Hyperparameter Tuning: Selecting the best deep learning and machine learning models and adjusting their hyperparameters for optimum performance can take some time. Finding the ideal mix of algorithms and parameters calls for experience and experimentation. It may be possible to pick the best model by testing with several algorithms (Naive Bayes, SVM, Random Forest), utilising libraries like scikit-learn, and fine-tuning their hyperparameters using GridSearchCV or RandomizedSearchCV.

Evaluation Metrics: It can be difficult to choose evaluation metrics that fit the project's objectives. Although accuracy is frequently employed, it is crucial to consider metrics like precision, recall, F1- score, and ROC curves, especially for datasets with imbalances. Class counts were calculated by the code, however scikit-learn could also be used to compute metrics like confusion matrix, precision, recall, F1-score, and ROC-

AUC for a more thorough analysis. Cross-validation with stratified criteria might guarantee objectivity.

Overfitting and Generalization: It is crucial to make sure that the trained models can generalize well to new data and do not overfit the training data. To deal with this issue, methods like cross- validation, regularization, and employing distinct validation and test sets are crucial. It can be avoided by using methods like dropout and L2 regularisation in deep learning models, early halting, and monitoring validation curves. Regular test set splitting and validation can assist evaluate generalisation.

Deep Learning Complexity: Due to the technical complexity of neural networks, deep learning models like the BiLSTM RNN, CNN, and LSTM model can be difficult to implement and perfect. Considerations include managing parameters, dealing with disappearing gradients, and picking the best architecture. Utilising high-level libraries like TensorFlow or PyTorch makes it easier to create sophisticated models. Pre-trained embeddings, batch normalisation, and attention techniques can all improve model performance and convergence.

Web Application Development: Several elements, including the Flask framework, HTML/CSS for the front-end, PyMySQL for database management, and model integration, must be integrated to create a user-friendly online application. It can be challenging to guarantee seamless interactions between these elements. A user-friendly interface may be made by using HTML/CSS/JavaScript for the frontend and Flask for the backend. The user experience can be improved by using libraries like Flask-Login for authentication, SQLAlchemy for secure database interactions, and AJAX for asynchronous activities.

Database Management and Security: In terms of data privacy and security, managing user authentication, monitoring database connections, and storing user data safely in the database can be difficult. Data security can be achieved by hashing/salting passwords and utilising prepared statements in SQL queries.

Deployment and Scalability: It can be challenging to deploy the web application to a production environment and guarantee its scalability to support multiple concurrent users. The management of server resources and performance optimization are essential components. Smooth deployment and scalability can be achieved by employing cloud platforms like AWS or Heroku, load balancing, and server resource optimisation.

Model Interpretability: It is crucial to understand how the models make their judgments (interpretability), especially in delicate applications like spam detection.

With certain complicated models, it can be difficult to explain why a specific tweet was labeled as spam. Model Interpretability can be improved by incorporating attention processes into deep models, employing libraries like SHAP or LIME for explaining predictions, and offering clear explanations.

5 Results

The model had an accuracy of 82% after starting with Naive Bayes. A consistent precision, recall, and F1-score for both the Real and Fake categories confirm this, demonstrating the model's comprehensive ability to forecast and capture the essence of both classes. Its general stability is shown by the confusion matrix, which shows that it commits an equal proportion of errors for both classes.

With an accuracy of 81%, Logistic Regression came in second to Naive Bayes. It showed a higher recall for the Real class - 85% but a significantly lower remember for the Fake class - 76%. Precision was noticeably greater for the phoney class, suggesting that when the algorithm correctly identified a tweet as phoney, it typically did so.

Despite being renowned for its ability to increase gradients, the XGB Classifier underperformed in this case, with an accuracy of 76%. Its comparatively poor recall for the Fake class, at 59%, masked its outstanding recall for the Real class (93%). This demonstrates the model's propensity to forecast tweets as Real, perhaps because of some dominant traits.

An accuracy of 78% was displayed by RandomForestClassifier. It showed good recall for the Real class (88%) but poor remember for the Fake class (69%). In this instance, the model's prowess in feature selection and bootstrapping may have caused it to focus excessively on patterns, leading to misclassifications for the Fake class.
Unfortunately, the KNN classifier performed poorly, with an accuracy of 49%. The most notable aberration is the model's complete misinterpretation of the Fake class, which had a recall of less than 0%. The complex patterns in the text data may not be suitable for KNN's distance-based technique, resulting in subpar performance.

A more straightforward model, Decision Tree Classifier, achieved a 75% accuracy. For both classes, it offered a good combination of precision and recall, but because of the thorough feature splits that are part of its design, it may also run the danger of overfitting.

With an accuracy rate of 81%, SVC performed similarly to Logistic Regression. The model, which was based on the fundamental idea of maximising margins, showed somewhat better precision for the Fake class while maintaining strong recall for the Real class. When there is a need to lessen false positives in Fake classifications, it can be the best option.

BiLSTM, a representative of the deep learning field, scored a decent accuracy of 78.58%. Given the complexity and potential overfitting hazards involved with deep learning, its recall for the Spam (or Fake) class was 72.36%, which highlights the significance of data preparation. The BiLSTM model is computationally expensive, and although it has been successful in capturing sequential dependencies in textual data, its performance here shows that classic machine learning models might be just as useful, if not more so, for this particular purpose.

The accuracy of the CNN model, which is close to 79%, is comparable to that of the BiLSTM. However, it still maintains acceptable precision and recall ratings for both groups, with a little slant toward the "Real" class.

Contrastingly, the LSTM model disappoints with a paltry 49% accuracy, reflecting the KNN's tendency to label nearly all samples as "Real." This can be the result of poor model design or insufficient training data.

In conclusion, the Naive Bayes, Logistic Regression, and SVC stand out as the top conventional algorithms, while the BiLSTM and CNN maintain their position as leaders in the deep learning area. But the KNN and LSTM models clearly lag, calling for additional research or modification. A thorough review of a model's accuracy, precision, recall, and f1-score are essential before choosing it, especially when choosing between classes has important repercussions.

The dataset is broken up into three primary subsets: the training set, the validation set, and the test set. To accurately assess how well the machine learning models are performing, this separation is essential. Following is a typical breakdown of the dataset:

Training Set: The training set, which makes up most of the dataset, is what the machine learning models are trained on. It has labeled examples, which the models utilize to discover trends, connections, and features in the data. Usually between 70 and 80 percent of the dataset is allotted to the training set.

Validation Set: The dataset that is utilized for hyperparameter optimization and model selection is divided into the validation set and training set. By offering a distinct dataset that the models haven't seen during training, it helps to avoid overfitting. The performance of various model configurations and hyperparameters can be assessed using the right metrics after testing them on this collection. The dataset usually contains between 10% and 15% of the validation set.

Test Set: The test set is a different subset of the dataset that is used to gauge how well the trained models ultimately perform. It acts as a dataset that hasn't been viewed by the models during training or validation. How well the models generalize to fresh, untested data can be inferred from their performance on the test set. The remaining 10–20% of the dataset often makes up the test set.

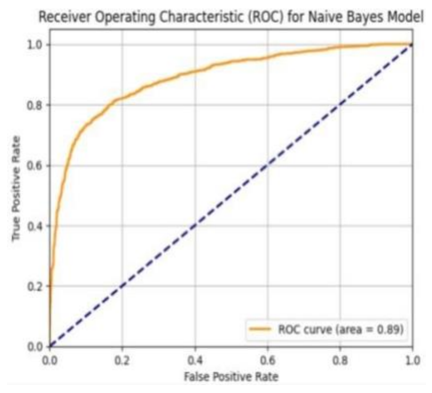
Making sure that the dataset is randomly divided into various subsets and that they are all representative of the total data distribution is crucial. By doing this, biases in the appraisal process are reduced. In addition, depending on the type of problem being handled, the evaluation metrics utilized on the test set should be in line with the project's objectives. Examples of such metrics include accuracy, precision, recall, F1-score, and ROC curves.

Overall, the dataset's segmentation into training, validation, and test subsets enables thorough model evaluation and aids in the decision of which model will perform the best when it is deployed.

An impressive AUC of 0.89 distinguishes the Naive Bayes classifier. This score highlights the model's superb ability to distinguish between the positive (spam) and negative (non- spam) classes. As shown in **Figure 8**, The high AUC demonstrates its ability to correctly identify real positives while reducing false positives. This performance demonstrates the Naive Bayes classifier's suitability for the spam detection challenge as a dependable option.

The Logistic Regression model presents itself as a tough performer with a slightly trailing AUC of 0.88. As shown in **Figure 9**, The model performs almost as well as the Naive Bayes, demonstrating how well it can distinguish between tweets that are spam and those that are not. The effectiveness of Logistic Regression in binary classification problems is highlighted by this performance.

The XGB Classifier's AUC is 0.83 thanks to gradient boosting. Any AUC value more than 0.8 is typically regarded favorably, As shown in **Figure 10**, despite not being the best in this race. Even though some of its competitors marginally outperform the XGB Classifier in terms of robust discriminative strength, it nevertheless solidifies the XGB Classifier as a strong classification tool.



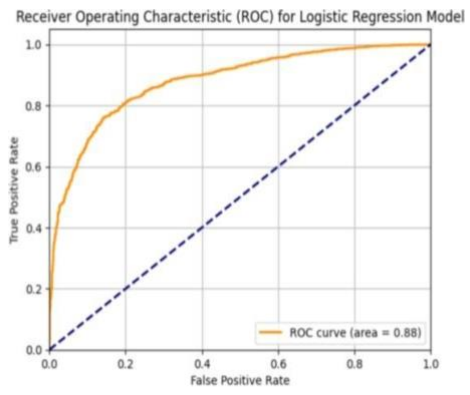
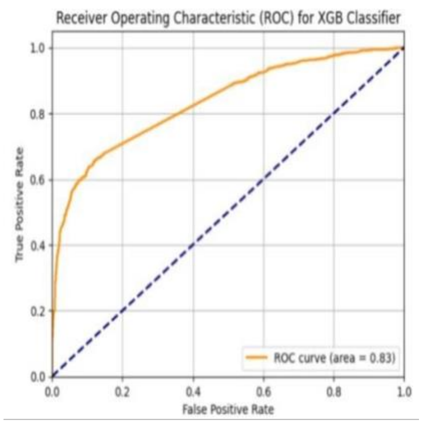


Figure 8. ROC for Naïve Bayes Model

Figure 9. ROC for Logistic Regression Model

The ensemble-based Random Forest Classifier emerges with an AUC of 0.86, emphasising its power. As shown in **Figure 11**, The model establishes itself as a reliable solution for the spam detection task by showcasing its refined capacity to create consistent and accurate predictions. The model is known for aggregating decisions from numerous trees.



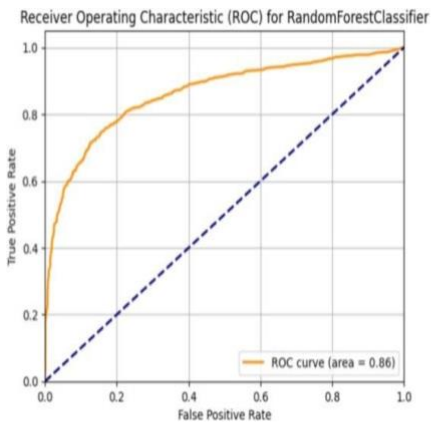


Figure 10. ROC for XGB Classifier

Figure 11. ROC for RandomForestClassifier

AUC of 0.77 indicates a mediocre result for the K-Nearest Neighbours classifier in this lineup. However, when compared to top-tier models like Naive Bayes, it clearly shows a gap while having a respectable amount of discriminative power. As shown in **Figure 12**. This performance makes some suggestions for future areas of growth or methodological changes.

The Decision Tree model indicates problems with an AUC of 0.75. The model's performance highlights the need for more tuning, parameter modifications, or a reevaluation of the model's approach, As shown in **Figure 13** whether the cause is overfitting or a failure to generalise well to new data.

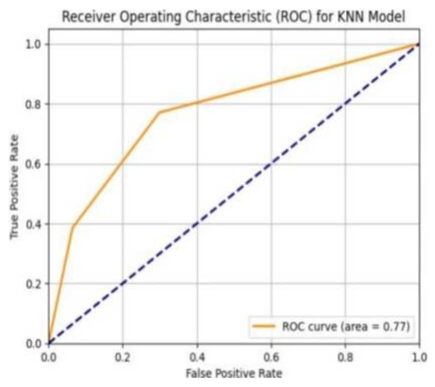


Figure 12. ROC for KNN Model

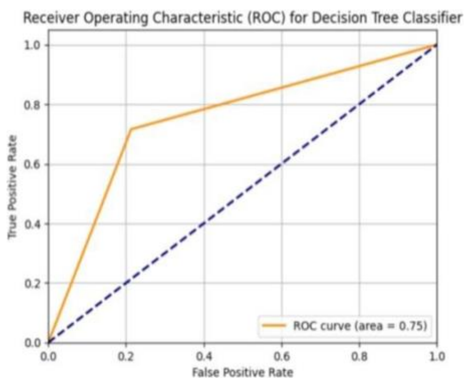
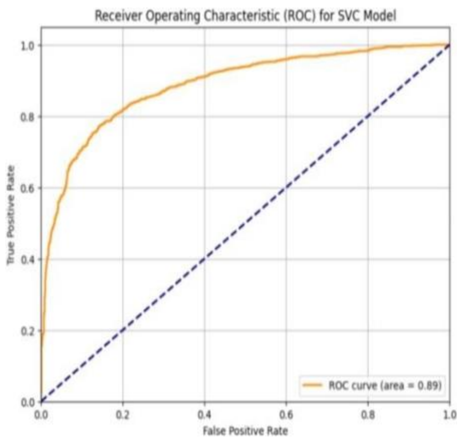


Figure 13. ROC for Decision Tree Classifier

The Support Vector Classifier has an AUC of 0.89, keeping pace with Naive Bayes. This exceptional performance confirms the model's prowess in determining the optimum hyperplane to divide classes, making it a top contender in this examination. As shown in **Figure 14.**

The BiLSTM model, which is based on deep learning, has an AUC of 0.86. As shown in **Figure 15.** This impressive performance is evidence of the model's prowess in catching the subtle sequential patterns present in textual input. With this ability, the BiLSTM demonstrates that it is suitable for workloads involving natural language processing, notably those involving spam detection.

An AUC of 0.87 suggests that the CNN model performs well in terms of classification. The range of AUC values is 0 to 1, with 0.5 reflecting random guessing and 1 perfect classification. With a value of 0.87, the model has an 87% chance of successfully differentiating between a random positive and a random negative instance. A high true positive rate and a low false positive rate are indicators of robustness in the CNN's capacity to distinguish between the "Real" and "Fake" (or "Spam") classes, according to its high AUC. The accuracy and other metrics we previously observed are well-aligned with this performance, As shown in **Figure 16,** indicating the CNN is a trustworthy model for this task.



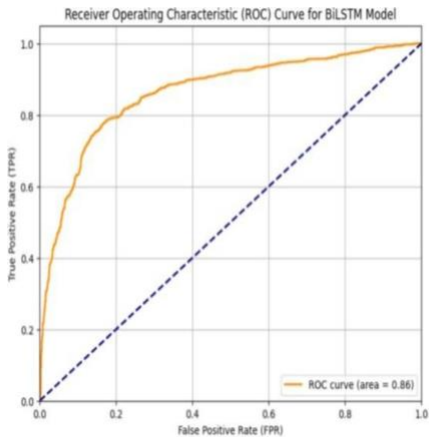
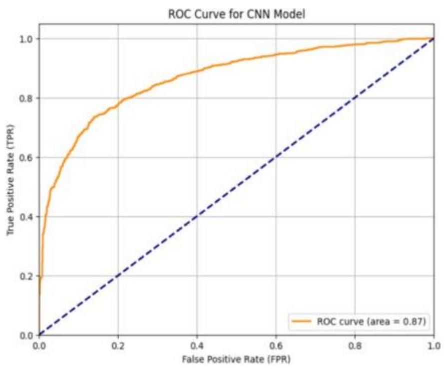


Figure 14. ROC for SVC Model

Figure 15. ROC for BiLSTM Model



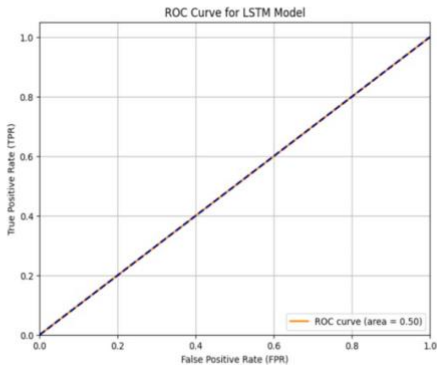


Figure 16. ROC for CNN Model

Figure 17. ROC for LSTM Model

The LSTM's AUC of 0.50 is alarming. As shown in the **Figure 17**, When it comes to differentiating between the classes, a model with an AUC value of 0.50 is no better than random guessing. This supports our earlier finding that almost all samples were categorized as "Real" by the LSTM, suggesting a possible problem with the model's design, training procedure, or data representation. Practically speaking, an AUC of 0.50 indicates that there is a substantial probability of misclassification, making the model inappropriate for applications where accuracy is crucial. To enhance the model's performance, this can call for a review of its structure, hyperparameters, or training method.

Table 4. Algorithm Performance Measures Comparison

	Naïve Bayes	Logistic Regression	XGB Classifier	Random Forest Classifier	KNN	Decision Tree Classifier	SVC	BiLSTM	CNN	LSTM
Accuracy	81.64	80.79	75.61	78.54	49.41	74.30	81.30	78.50	79.13	49.36
Recall	81.07	76.21	58.96	69.68	0.08	72.36	77.39	75.54	74.29	0
F1-Score	81.72	80.07	71.00	76.68	0.17	74.04	80.73	78.06	78.29	0
Precision	82.38	84.34	89.23	85.25	100	75.79	84.38	80.75	82.74	0
Negative Recall	82.21	85.48	92.70	87.63	100	76.29	85.31	81.53	84.11	100

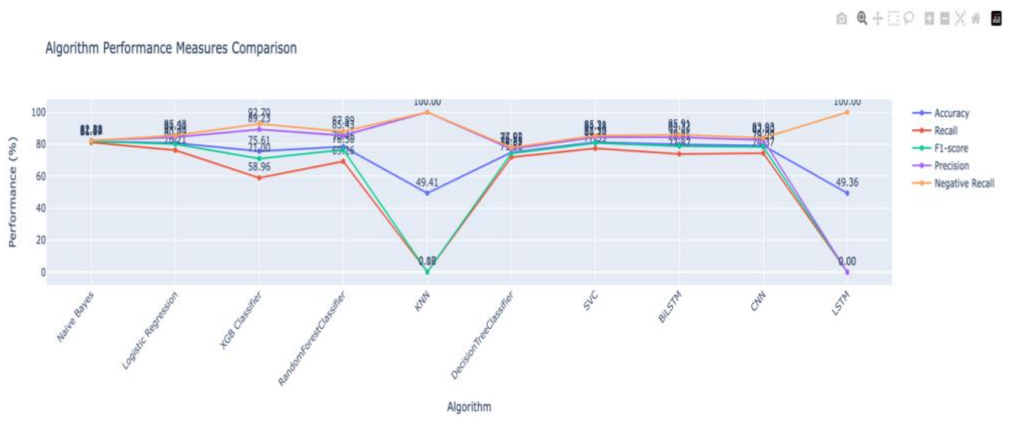


Figure 18. Algorithm Performance Measures Comparison

5.1 Evaluation

The "Twitter Spam Detection" project is an admirable attempt to use a variety of deep learning and machine learning models to distinguish between legitimate tweets and spam in real-time. Its strength is not only in its thorough model selection, which includes both conventional algorithms like Naive Bayes and Logistic Regression and cutting-edge structures like BiLSTM, but also in its painstaking evaluation of performance. The study provided a comprehensive knowledge of each model's discriminative power using various metrics, including accuracy, precision, recall, F1-score, and particularly the ROC curve area. Its devotion to practical utility was demonstrated by the real-world integration into an approachable online application, which enhanced its technological prowess. Furthermore, the emphasis on reliable database management with PyMySQL and strict user authentication protocols matched the project's emphasis on security and data integrity. To ensure that models have high-quality training data, a strong foundation was set with stringent data preparation and NLP approaches. The project handled difficulties like data imbalances, complex feature extraction, and potential model overfitting with ease. The comparative examination of model performance shed more light on the strengths and weaknesses of each, enabling deployment decisions to be made with knowledge. Additionally, the project's user-centric design, demonstrated by its simple registration and login procedures, emphasises its commitment to improving user experience. The project promises scalability and flexibility because to its modular structure, which also puts it in a good position for future improvements. The project essentially serves as a shining example of meticulous research, careful evaluation, and practical implementation in the field of spam identification.

6 Conclusion

Machine learning (ML) and deep learning (DL) models can be effectively utilised to detect spam on Twitter, as shown by the "Twitter Spam Detection" project. We were able to acquire detailed insights on each classifier's capabilities by using a wide range of classifiers, from more straightforward deep learning models like BiLSTM, CNN and LSTM to more complex classical ML models like Naive Bayes, Logistic Regression, and SVC. The experiment demonstrated that Naive Bayes and the Support Vector Classifier were the most effective models, with an excellent AUC of 0.89 each. The user experience was further enhanced using Flask as the web framework and PyMySQL as the database management system, which made it possible for frictionless registration, login, and real-time spam detection. Utilising NLP for feature extraction improved efficiency, especially when it came to comprehending the subtle subtleties of tweets' context.

6.1 Future Works

Enhanced Feature Engineering: It's critical to continuously improve and broaden our feature set as spam tactics change. To better capture the semantic content of tweets, advanced NLP techniques like word embeddings (Word2Vec, GloVe), transformer models like BERT, and so on, could be included.

Deep Learning Architectures: While BiLSTM demonstrated promising results, examining alternative architectures such as Convolutional Neural Networks (CNN) for text data or a hybrid model combining CNN and RNN may produce superior outcomes.

Real-time Adaptation: Models that can change in real-time, like online learning models, are essential since spam techniques are a shifting target. By doing so, the system would be able to continuously improve its spam detection capabilities by learning from new spam strategies.

User Feedback Loop: The models can be improved and retrained on a regular basis by incorporating a feedback system in the web application that allows users to submit any tweets that were incorrectly categorised.

Expansion to Other Platforms: The fundamental principles can be applied to other social media sites or even emails, ensuring a wider scope for spam detection even though the current concentration is on Twitter.

Collaborative Filtering: Collaborative filtering can help identify spam based on user relations and interactions, offering an additional layer of verification, considering the relational dynamics on platforms like Twitter.

The initiative essentially provides a strong basis given its present successes. To stay ahead in the spam detection race, however, constant innovation and adaptability are required due to the spam's ever-evolving nature.

Acknowledgments.

I would like to thank my dissertation supervisor, Bo Wei, for putting up with me for countless hours over Zoom, answering all my many questions, and keeping my

project on track. I would also like to thank anyone that reads this, and all participants of the usability analysis for their time.

References

1. C. Chen, J. Zhang, X. Chen, Y. Xiang, and W. Zhou, “6 million spamtweets: A large ground truth for timely twitter spam detection,” in 2015.IEEE International Conference on Communications (ICC), June 2015, pp. 7065–7070.

2. A. Greig, “Twitter Overtakes Facebook as the Most Popular Social Network for Teens,According to Study, DailyMail, accessed on Aug. 1, 2015 ,” http://www.dailymail.co.uk/news/article-2475591/Twitter-overtakes-Facebook-popular-socialnetwork-teens-according-study.html, 2015, [Online].

3. H. Tsukayama, “Twitter turns 7: Users send over 400 million tweets perday,” https://tinyurl.com/ybsaq7e7, 2013, [Online].

4. F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida, “Detectingspammers on twitter,” in In Collaboration, Electronic messaging, Anti_Abuse and Spam Conference (CEAS, 2010.

5. C. Pash., “The lure of Naked Hollywood Star Photos Sent the Internet into Meltdown in New Zealand, Bus. Insider, accessed on Aug. 1, 2015,” https://tinyurl.com/yc93ssj4, 2014, [Online].

6. “BotMaker, https://blog.twitter.com/engineering/enus/a/2014/fighting-spam-with-botmaker.html, [Online].

7. K. Thomas, C. Grier, D. Song, and V. Paxson, “Suspended accounts inretrospect: An analysis of twitter spam,” in Proceedings of the 2011ACM SIGCOMM Conference on Internet Measurement Conference, ser.IMC ’11. New York, NY, USA: ACM, 2011, pp. 243–258. [Online].

8. Kabakus, Abdullah Talha & Kara, Resul. (2017). A Survey of Spam DetectionMethods on Twitter. International Journal of Advanced Computer Science and Applications. 8. 10.14569/IJACSA.2017.080305. Availableat: https://www.researchgate.net/publication/315966273_A_Survey_of_Spam_Detection_Me tho ds_on_Twitter

9. bkenar, Sepideh & Haghi Kashani, Mostafa & Akbari, Mohammad & Mahdipour, Ebrahim. (2020). Twitter Spam Detection: A Systematic Review. Available at: https://arxiv.org/abs/2011.14754

10. Wu, Tingmin & Liu, Shigang & Zhang, Jun & Xiang, Yang. (2017). Twitter spam detection is based on deep learning. 1-8. 10.1145/3014812.3014815. Available at: https://www.researchgate.net/publication/312428491_Twitter_spam_detection_based_on _deep_learning

11. Anisha P Rodrigues, Roshan Fernandes, Aakash A, Abhishek B, Adarsh Shetty, Atul K, Kuruva Lakshmanna, and R. Mahammad Shafi. (2022). Real-Time Twitter Spam Detection and Sentiment Analysis using Machine Learning and Deep Learning Techniques.

Available at: https://www.hindawi.com/journals/cin/2022/5211949/#related-articles
12. Sun, Nan & Lin, Guanjun & Qiu, Junyang & Rimba, Paul. (2020). Near real-time twitter spam detection with machine learning techniques. International Journal of Computers and Applications. 44. 1-11. 10.1080/1206212X.2020.1751387.
Available at: https://www.researchgate.net/publication/340708941_Near_real-time_twitter_spam_detection_with_machine_learning_techniques
13. Zulfikar Alom, Barbara Carminati, Elena Ferrari, A deep learning model for Twitter spam detection, Online Social Networks and Media, Volume 18, 2020, 100079, ISSN 2468- 6964, Availableat: https://doi.org/10.1016/j.osnem.2020.100079. (https://www.sciencedirect.com/science/article/pii/S2468696420300203)
14. Anisha P Rodrigues, Roshan Fernandes, Aakash A, Abhishek B, Adarsh Shetty, Atul K, Kuruva Lakshmanna, R. Mahammad Shafi, "Real-Time Twitter Spam Detection and Sentiment Analysis using Machine Learning and Deep Learning Techniques", Computational Intelligence and Neuroscience, vol. 2022, Article ID 5211949, 14 pages, 2022. https://doi.org/10.1155/2022/5211949
15. F. Benevenuto, G. Magno, T. Rodrigues, V. Almeida, Detecting spammers on Twitter, in: CEAS 2010 - Seventh Annu. Collab. Electron. Messag. Anti-Abuse Spam Conf., Redmond, Washington, USA, 2010: pp. 12–21. doi:10.1.1.297.5340.
16. A. Tumasjan, T. Sprenger, P. Sandner, I. Welpe, Predicting Elections with Twitter: What 140 Characters Reveal about Political Sentiment, in: Proc. Fourth Int. AAAI Conf. Weblogs Soc. Media, Washington, DC, USA, 2010: pp. 178–185. doi:10.1074/jbc.M501708200.
17. F. Bravo-Marquez, M. Mendoza, B. Poblete, Combining strengths, emotions, and polarities for boosting Twitter sentiment analysis, in: Proc. Second Int. Work. Issues Sentim. Discov. Opin. Min. (WISDOM ’13), Chicago, IL, USA, 2013: pp. 1–9. doi:10.1145/2502069.2502071.
18. A. Pak, P. Paroubek, Twitter as a Corpus for Sentiment Analysis and Opinion Mining, Computer (Long. Beach. Calif). 2010 (2010) 1320– 1326. doi: 10.1371/journal.pone.0026624.
19. Company | About, Twitter. (2017). https://about.twitter.com/company (accessed February 5, 2017).
20. Twitter Usage Statistics - Internet Live Stats, InternetLiveStats. (2017). http://www.internetlivestats.com/twitter-statistics/ (accessed February 5, 2017).
21. L. Jiang, M. Yu, M. Zhou, X. Liu, T. Zhao, Target-dependent Twitter Sentiment Classification, in: Comput. Linguist., 2011: pp. 151–160.
22. A. Go, L. Huang, R. Bhayani, Twitter Sentiment Analysis, Entropy. (2009) 17. doi:10.1007/978-3-642-35176-1_32.
23. E. Haddi, X. Liu, Y. Shi, The Role of Text Pre-processing in Sentiment Analysis, Procedia Comput. Sci. 17 (2013) 26–32. doi: 10.1016/j.procs.2013.05.005.
24. H. Saif, Y. He, M. Fernandez, H. Alani, Semantic Patterns for Sentiment Analysis of Twitter, in: Proc. 13th Int. Semant. Web Conf., Trentino, Italy, 2014: pp. 324–340.