Ministry of Science and Higher Education of the Republic of Kazakhstan

Suleyman Demirel University

Abdulla Ydyrys

Identifying spam messages for Kazakh Language using Hybrid Machine Learning Model

THESIS

Presented in Partial Fulfilment for the

Master of Technical Sciences Degree in Computer Science

(degree code: 7M06102)

Department of Computer Science

Faculty of Engineering and Natural Sciences

Supervisor: **PhD Assist Prof. Nazerke Sultanova**

Kaskelen 2023

Suleyman Demirel University
Faculty of Engineering and Natural Sciences
Department of Computer Science

Dean of Faculty

PhD, Associate Professor

Zhamanov A.

"06" 06 2023

Topic of the thesis:

Identifying spam messages for Kazakh Language using Hybrid Machine Learning Model

Thesis submitted as part of the requirements for the award of the MSc in "7M06102 - Computer Science" SDU, 2021-2023

Head of Department _____ Assistant Professor, PhD Mukash Zh.

Academic Supervisor _____ Assistant Professor, PhD Sultanova N.

Master student _____ Ydyrys Abdulla

Kaskelen 2023

Ministry of Science and Higher Education of the Republic of Kazakhstan

Suleyman Demirel University

Abdulla Ydyrys

Identifying spam messages for Kazakh Language using Hybrid Machine Learning Model

THESIS

Presented in Partial Fulfilment for the

Master of Technical Sciences Degree in Computer Science

(degree code: 7M06102)

Department of Computer Science

Faculty of Engineering and Natural Sciences

Supervisor: **PhD Assist Prof. Nazerke Sultanova**

Kaskelen 2023

Suleyman Demirel University
Faculty of Engineering and Natural Sciences
Department of Computer Science

Dean of Faculty

PhD, Associate Professor

Zhamanov A.

_____

« _____ »_____ 2023

Topic of the thesis:

Thesis submitted as part of the requirements for the award of the MSc in "7M06102 - Computer Science" SDU, 2021-2023

Head of Department _____ Assistant Professor, PhD Mukash Zh.

Academic Supervisor _____ Assistant Professor, PhD Sultanova N.

Master student _____ Ydyrys Abdulla

Kaskelen 2023

Declaration

I confirm that this is my own work and the use of all material from other sources has been properly and fully acknowledged.

Abdulla Ydyrys

2023

Acknowledgements

I want to express my sincere gratitude to Nazerke Sultanova, who oversaw my thesis, for her constant support throughout my research.

The external reviewers Martin Lukac from Nazarbayev University and Almas Serikbayev from International University of Information Technologies deserve my sincere gratitude as well. Their constructive feedback and insightful suggestions have played a crucial role in enhancing the quality of my work.

I also appreciate Suleyman Demirel University for giving me the resources and facilities that I needed to conduct my research successfully.

Dedication

I want to dedicate this thesis to my parents, Ydyrys Kasymkhan and Toleubai Altyngul.

I also want to dedicate this thesis to my supervisor, Sultanova Nazeke. Her expertise has been essential to completing this work successfully.

I want to to dedicate this thesis to all the others who have supported me. Their contributions have been important in making this project a reality.

Abstract

The rapid growth of digital communication has led to an increase in unwanted spam messages, which pose a significant challenge for users. While numerous machine learning techniques have been employed to combat spam in widely spoken languages, there is a lack of research on spam detection in less common languages, such as Kazakh. In this study, we propose a hybrid machine learning model for identifying spam messages in the Kazakh language. Our approach combines the strengths of different machine learning techniques to enhance the accuracy and efficiency of spam detection.

To evaluate the performance of our hybrid model, we manually collected a dataset of Kazakh language messages, which were labeled as spam or non-spam. The results demonstrate that our hybrid approach surpasses traditional machine learning methods in terms of accuracy, precision, recall, and F1-score. This model not only improves the efficiency of spam filtering but also enhances the user experience for Kazakh-speaking individuals and promotes the development of language-specific spam detection techniques for other underrepresented languages.

Аңдатпа

Цифрлық байланыстың қарқынды өсуі спам хабарламалардың көбеюіне әкелді, бұл пайдаланушылар үшін айтарлықтай қиындық тудырады. Кең таралған тілдерде спаммен күресу үшін машиналық оқытудың көптеген әдістері қолданылғанымен, қазақ тілі сияқты аз таралған тілдерде спамды анықтау бойынша зерттеулер жеткіліксіз. Бұл зерттеуде біз қазақ тіліндегі спам хабарламаларды анықтауға арналған гибридті машиналық оқыту үлгісін ұсынамыз. Біздің әдіс спамды анықтаудың дәлдігі мен тиімділігін арттыру үшін әртүрлі машиналық оқыту әдістерінің күшті жақтарын біріктіреді.

Гибридті үлгіміздің өнімділігін бағалау үшін біз спам немесе спам емес деп белгіленген қазақ тіліндегі хабарламалардың деректер жинағын жинадық. Нәтижелер біздің гибридті тәсіліміздің дәлдік, толықтық және F1 ұпайлары бойынша дәстүрлі машиналық оқыту әдістерінен асып түсетінін көрсетеді. Бұл модель спамды фильтрлеудің тиімділігін арттырып қана қоймайды, сонымен қатар қазақ тілді адамдар үшін пайдаланушы тәжірибесін жақсартады және басқа да аз таралған тілдердегі спамды анықтау әдістерін дамытуға ықпал етеді.

Аннотация

Быстрый рост цифровых коммуникаций привел к увеличению количества нежелательных спам-сообщений, которые создают серьезную проблему для пользователей. Хотя для борьбы со спамом на широко распространенных языках используются многочисленные методы машинного обучения, исследования по обнаружению спама на менее распространенных языках, таких как казахский, отсутствуют. В этом исследовании мы предлагаем модель гибридного машинного обучения для выявления спам-сообщений на казахском языке. Наш подход сочетает в себе сильные стороны различных методов машинного обучения для повышения точности и эффективности обнаружения спама.

Чтобы оценить эффективность нашей гибридной модели, мы вручную собрали набор данных сообщений на казахском языке, которые были помечены как спам или не спам. Результаты показывают, что наш гибридный подход превосходит традиционные методы машинного обучения с точки зрения точности, полноты и оценки F1. Эта модель не только повышает эффективность фильтрации спама, но также улучшает взаимодействие с пользователем для казахскоязычных пользователей и способствует развитию языковых методов обнаружения спама для других недостаточно представленных языков.

Abbreviations

ML Machine Learning

URL Uniform Resource Locator

MNB Multinomial Naive Bayes

KNN K-Nearst Neighbor

SVM Support Vector Machine

DT Decision Tree

RFC Random Forest Classifier

LR Logistic Regression

NLP Natural Language Processing

Table of Contents

Chapter 1

Introduction

1.1 Background and Motivation

The Internet is progressively becoming an essential component of our daily routines. Its usage is predicted to keep expanding. However, parallel to the Internet's growth, there has been a significant surge in the volume of unwanted spam messages over recent years. These unsolicited messages can originate from anywhere in the world, as long as there is Internet access. It is noteworthy that despite the advancements in anti-spam solutions and technologies, the quantity of spam messages continues to escalate at an alarming rate [1].

The significant rise in spam volume over the past year has generated considerable interest among researchers to combat this issue. Numerous researchers are currently engaged in developing new filters that aim to prevent spam from reaching its intended recipients. These filters operate at either the server or client level, employing various techniques to block unwanted messages. Machine learning (ML) algorithms have demonstrated success in text categorization, leading researchers to explore their application in spam filtering. There are several categories of spam filtering methods, including Content Based, List Based, Rule Based, Case Base, and Adaptive approaches [2].

Spam messages in Kazakh language, like spam messages in any other language, can have various characteristics. Here are some common characteristics often

found in spam messages:

Unwanted Content: Spam messages typically contain unsolicited and irrelevant content. They might promote products, services, or websites that the recipient did not request or express interest in.

Mass Distribution: Spam messages are usually sent in bulk to a large number of recipients simultaneously. They are not personalized and often lack specific details that would indicate a targeted message.

Deceptive Subject Lines: Spam messages often use deceptive subject lines to grab the recipient's attention or trick them into opening the message. They might use all caps, excessive punctuation, or misleading information to entice the recipient.

Poor Grammar and Spelling: Spam messages often contain grammatical errors, misspellings, and awkward sentence structures. This is because many spam messages are generated automatically or by non-native speakers.

Phishing Attempts: Some spam messages may attempt to deceive recipients into disclosing sensitive data, like login credentials, financial information, or government identification numbers. These messages often mimic legitimate organizations or websites and urge the recipient to click on a link or provide sensitive information.

URL Shorteners: Spam messages often include shortened URLs or redirect links. These are used to conceal the actual destination of the link and may lead to malicious websites or phishing pages.

There is a significant research lack when it comes to the Kazakh language, despite the fact that there is plenty of literature on identifying spam messages in widely used languages like English. There are currently insufficient utilities and materials available to efficiently identify spam messages written in Kazakh. This gap offers a number of research opportunities, including investigating ways to recognize and filter out spam messages specifically in Kazakh and resolving the particular difficulties involved in working in this field. Several of these difficulties include:

Insufficiently labeled datasets: One of the main challenges is that there aren't enough collections of spam and non-spam messages in Kazakh that have been properly labeled. This makes it hard to teach and test machine learning models for identifying spam. The initial and crucial step to solve this problem is to create datasets that have accurate labels.

Distinctive language characteristics: The way Kazakh language works might be different from other languages, and the methods used to extract important features for detecting spam messages may not work well for Kazakh. Finding new ways to extract language-specific features for Kazakh could be an important research challenge to tackle.

Exploring alternative machine learning techniques to detect spam in the Kazakh language could present a promising research opportunity. This involves creating and testing different approaches to effectively identify and filter out spam messages in Kazakh.

Spam comes in many different forms and keeps changing all the time. That's why it's really important to keep coming up with new ways to detect and combat it. Evaluating the effectiveness of existing spam identification methods highlights the importance of developing hybrid approaches that offer enhanced accuracy. Furthermore, the absence of spam detection systems specific to the Kazakh language further emphasizes the necessity for a proposed system that can identify unsolicited messages.

1.2 Aims and Objectives

The primary aim of this project is to develop a hybrid machine learning model for identifying spam messages in the Kazakh language, while ensuring a low false positive rate and a low false negative rate. This means that non-spam messages should never be classified as spam - should have a low false positive rate. This is the ratio of the number of normal messages mistakenly recognized as spam to the amount of all normal messages. Spam messages should never be classified as non-spam - should have a low false negative rate. This is the ratio of the number of missed spam to the total spam volume. To achieve this, we will collect and

process a large dataset of Kazakh language messages to train and test the model. We will evaluate and compare the model's performance using various metrics. The end result will be a user-friendly interface that can be easily integrated into any applications and can handle large volumes of messages.

The overall objectives of this research project can be summarized as follows:

1. to build a corpus of Kazakh language spam messages
2. to thoroughly examine and experiment with various machine learning models, with the aim of finding the most appropriate ones for constructing a hybrid system that detects spam in the Kazakh language
3. to offer a practical and productive resolution to tackle the issue of unwanted spam messages specifically in the Kazakh language

By recommending a combination of machine learning models that can successfully identify spam messages in the Kazakh language, we aim to improve the knowledge of current literature.

1.3 Thesis Outline

The thesis was divided into the following chapters:

Chapter 2: Literature Review. This chapter offers a thorough analysis of the existing literature on spam detection, machine learning models, and methods used in the field.

Chapter 3: Methodology. Subsection: Dataset. The collection, preparation, and construction of the corpus of spam messages in Kazakh was covered in this subsection. Subsection: Data Preprocessing. This subsection describes the processes used to clean, remove stop words, special characters, digits, and links from the dataset. Subsection: Feature Extraction, Data Split: Test and Train Division, Training Phase. The methods used for feature extraction, spitting the data, and the training phase of the models were covered in this subsection.

Chapter 4: Results for Non-hybrid and Hybrid Models, User Interface. The findings and results from experiments utilizing non-hybrid and hybrid spam de-

tection models were presented in this chapter. It also describes the user interface created for the suggested solution.

Chapter 5: Discussion and Conclusion. The key findings were summarized in the last chapter, along with its major contributions, limitations, and potential directions for future research.

Chapter 2

Literature review

In the literature review section of this work, we thoroughly examine the current research on ML-based methods for filtering spam. The review covers a range of algorithms and techniques proposed in previous studies. We carefully assess the advantages and drawbacks of each approach. In our work, we will leverage the classifiers evaluated in this section to create a hybrid spam filtering algorithm. Our aim is to merge the positive aspects of multiple classifiers while mitigating their limitations, resulting in an innovative approach to spam detection.

2.1 Naive Bayes Filtering Method

The technique of Naive Bayes spam filtering involves the application of Bayes' theorem. In the training phase, the filter computes and remembers the importance of each word found in the text. Later, when a message is received, it is categorized as either "spam" or "non-spam" based on whether the importance of its words exceeds a predetermined limit. The formula 2.1.1 below illustrates how the likelihood of a message containing a particular spam word is calculated:

$$P(sp|w) = \frac{P(sp) * P(w|sp)}{P(sp) * P(w|sp) + P(nsp) * P(w|nsp)} \tag{2.1.1}$$

When a particular word is detected, the likelihood of a message being considered spam is expressed as P(sp/w). P(sp) quantifies the likelihood of a message

being classified as spam, whereas P(w/sp) denotes the probability of a specific word occurring in a spam message. P(nsp) signifies the general probability of a message being categorized as non-spam, while P(w/nsp) indicates the likelihood of encountering a particular word in a non-spam message [3].

When it comes to spam detection in the Kazakh language, the Multinomial Naive Bayes (MNB) algorithm can be employed. It is a Naive Bayes classifier variant that works especially well for text classification applications. The method determines the probability of occurrence for each feature in spam and non-spam messages during the training phase. It estimates the likelihood of a specific feature appearing in a spam message and the likelihood of the same feature occurring in a non-spam message [4]. The availability of a comprehensive and well annotated training dataset in Kazakh is crucial to MNB's performance. This dataset should include a diverse range of spam and non-spam messages representative of the language's characteristics. Once the MNB model is trained, it can be used to classify incoming messages as spam or non-spam with a high degree of accuracy. By assigning a probability score to each classification, it enables users to set an appropriate threshold for filtering out unwanted spam messages. In general, although Naive Bayes is a popular and efficient approach for filtering out spam messages, it does have its limitations. The "naive" in its name stems from the assumption that the features utilized for classification are mutually exclusive of one another. However, in real-world datasets, this assumption might not be applicable to particular features, which can result in less accurate classification outcomes.

2.2 Support vector machine

The Support Vector Machine (SVM) algorithm is commonly employed to categorize messages as either spam or non-spam (binary classes), although it can also be adjusted for handling multiple classes. It operates by identifying the most suitable boundary, known as a hyperplane, to distinguish between positive and negative instances. By evaluating this boundary, the algorithm determines whether a new message is categorized as spam or non-spam [5]. The classification

accuracy can be improved by maximizing the margin between hyperplanes. The margin refers to the space between the hyperplane and the closest data points belonging to each category. By maximizing this margin, SVMs aim to achieve better generalization and reduce the risk of overfitting. This property makes SVMs particularly useful when the training data is limited or when there is a class imbalance, such as in spam filtering where the number of spam messages is usually much smaller than non-spam messages. Once the SVM model is trained, it can be used to classify new, unseen messages as either spam or non-spam based on their feature representations. By assigning a label to each message, SVM enables effective spam filtering in the Kazakh language. Many research studies have demonstrated the efficiency of SVM in spam classification, achieving an accuracy rate of 96% [6]. Nevertheless, training SVM can be computationally demanding, dependent on the selection of the kernel function, susceptible to overfitting, and need the calibration of hyperparameters.

2.3 K-nearest neighbor

K-nearest neighbor (KNN) spam detection is an approach used to identify spam messages. The KNN algorithm analyzes the characteristics of a new message and compares them to the features of known spam and non-spam messages. By considering the classifications of the KNN, the algorithm assigns the new message to the class (spam or non-spam) that is most prevalent among its closest neighbors. In the context of spam filtering, this means that messages exhibiting similar characteristics to known spam messages are also likely to be spam. The primary factor used to identify similar samples is the distance measure. Typically, the Euclidean distance is employed to determine how close the samples are to each other. When a new message without a label is received, the KNN algorithm assesses its similarity to the labeled training examples by calculating the Euclidean distance. To compute the distance between two features, xi and xj, and find neighboring samples, a vector feature $<$ a1 (x), a2 (x), ..., an (x) $>$ is defined [7]. This distance calculation is carried out using Equation 2.3.1 below:

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2} \tag{2.3.1}$$

The figure 2.1 provides a visual representation of the relative positions of the neighboring data points, indicating their proximity to the new message. Among its nearby data points, there are three neighbors that belong to the "Not Spam" category, and four neighbors that fall into the "Spam" category. In this case, the algorithm would classify the new message as spam since the majority of its neighbors are in the "Spam" category. This decision-making process demonstrates how the KNN algorithm is simple and effective in classification tasks.

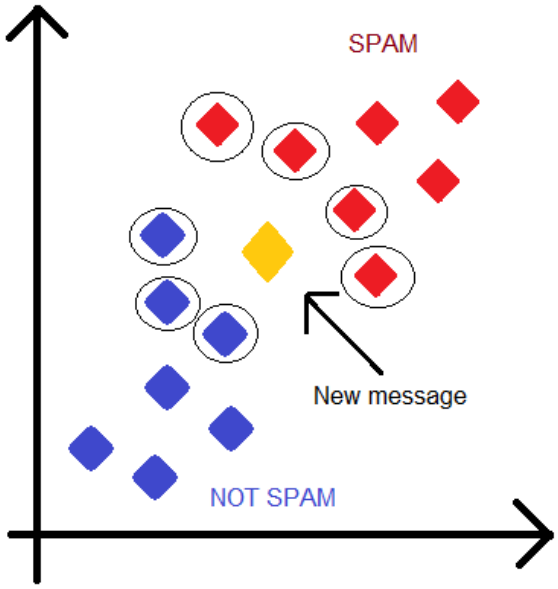

Figure 2.1: K-nearest neighbor

One significant difficulty with this algorithm is its higher computational burden when compared to other machine learning methods [8]. Finding the nearest neighbors involves evaluating the item being categorized against each item in the dataset, which demands a substantial amount of linear operations corresponding to the dataset's size. Furthermore, the performance of the KNN algorithm heavily depends on the choice of the parameter k. Selecting an optimal value for k is

crucial as choosing a small value may result in overfitting, causing to heightened sensitivity of the model to irrelevant data points and outliers, while choosing a large value may cause underfitting, leading to oversimplified classification boundaries by the model and fails to capture the intricacies of the data. One way to determine the optimal value of k is through cross-validation, where the dataset is divided into training and validation sets. The algorithm is trained with different values of k, and the performance is evaluated on the validation set using appropriate metrics such as accuracy, precision, recall, or F1 score. The value of k that yields the best performance is then selected as the optimal parameter [9]. Overall, while the KNN algorithm has its limitations, it remains a valuable tool in machine learning, particularly for tasks such as classification, where the interpretability and adaptability of the model are important factors.

2.4 Decision Tree

A Decision Tree (DT) is a type of organized tree structure employed in the identification of spam messages. This DT uses many features and patterns inside the message content to analyze them in a manner similar to humans thought process to make an informed judgment. It looks at things like the existence of ambiguous terms, the overuse of capital letters, the frequency with which exclamation points or questions are used, and the sheer volume of hyperlinks. By traversing the branches of this decision tre, it can accurately classify incoming messages as spam or non-spam, thereby helping us filter out unwanted and potentially harmful content. The visual representation depicted in Figure 2.2 showcases a DT model. In this model, each node represents a decision that relies on the value of a specific feature, while the branches illustrate the different potential results of that decision. To identify whether a text message is spam or not, the DT approach utilizes input words (F), their frequencies (V), and labels (C). The features taken into account throughout the decision-making process are represented by the input words (F). These words related frequencies (V) inform us how frequently they appear in the message. A message's classification as spam or not is indicated by the labels (C). By analyzing these inputs, the DT algorithm can accurately classify text messages based on their likelihood of being spam.

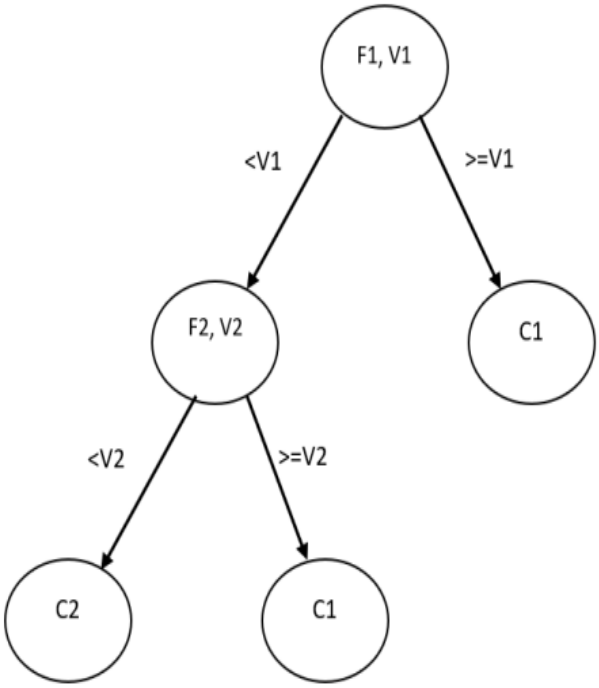

Figure 2.2: Decision Tree

When building a tree based on pre-determined attribute values, the process involves dividing the original set into multiple smaller subsets. This division is done repeatedly on each subset, following a recursive approach known as recursive partitioning. The recursion continues until the subset at a particular node shares a similar value with the target variable.

Determining the ideal size of the final tree presents a significant difficulties when employing DT. Alongside this challenge, the learning process also entails addressing other issues, such as selecting the most suitable attribute for dividing the data, determining when to halt the learning process, choosing an appropriate pruning technique to reduce the tree's size, and evaluating the accuracy of the resulting tree [10]. Therefore, it's really important to adjust the settings of the tree, as this could negatively impact on the model's overall performance.

2.5 Random Forest Classifier

Random Forest Classifier (RFC) is an ensemble learning technique utilized for classification and regression tasks. In the context of spam detection, Random

Forest operates by constructing an ensemble of decision trees based on randomly selected subsets of features and training data. Each decision tree independently classifies incoming messages as spam or non-spam based on various features and their thresholds. During the classification process, the Random Forest utilizes the predictions made by each decision tree, and the ultimate classification is determined by selecting the option that receives the most votes from all the trees [11]. One of the advantages of using the Random Forest technique is its ability to select features and take into account numerous subsets of features, rather than solely focusing on a small set of features that effectively distinguish the training data. This ensemble approach helps improve the accuracy and robustness of the spam detection system, as the combined decision of multiple trees can better handle variations and complexities in the data.

The study [12] successfully classified spam messages by employing various machine learning methods. The researchers implemented ten different classifiers on a benchmark dataset to determine the most effective classifier. They employed a 10-fold cross-validation technique to assess accuracy. The findings revealed that the RFC achieved an accuracy of approximately 95.45%, outperforming the other classifiers utilized in the study.

Random Forest's main drawbacks for spam detection include its lack of interpretability, making it challenging to understand the decision-making process, and its potential for overfitting if the model is too complex or the number of trees is excessive relative to the training data size.

2.6 Logistic Regression

Logistic regression (LR) is a popular algorithm for spam detection due to its simplicity and effectiveness in binary classification tasks. It models the relationship between the input features (such as content, sender information, and metadata) and the probability of a message being classified as spam [13].

In this particular study [14], the researchers investigate methods to minimize the logistic loss function in the context of spam filtering. They conduct a performance analysis of various techniques with the aim of determining whether a given

message is classified as spam or not. The paper emphasizes the effectiveness of LR as one of the top techniques for distinguishing between spam and non-spam messages.

One advantage of LR is its interpretability. The algorithm estimates the coefficients for each input feature, allowing us to understand the impact of each feature on the likelihood of an email being spam. This can provide insights into the important factors contributing to spam detection. LR also handles sparse data well, which is common in spam detection scenarios. It can handle a large number of features and works efficiently even when dealing with high-dimensional data. However, LR has some limitations. One major drawback is its assumption of a linear relationship between the input features and the output. This means that if the relationship is non-linear, LR may not capture complex patterns effectively. Another challenge with LR is handling imbalanced datasets, where the number of spam messages is significantly lower than non-spam messages. Since LR focuses on maximizing overall accuracy, which can sometimes lead to a preference for the larger class and make it challenging to accurately predict the smaller class.

Chapter 3

Methodology

In the upcoming sections, we introduce techniques and approaches that encompass tasks such as reading and preparing the data, creating a bag of words and labels (feature extraction), splitting the data, creating and fitting (training) the model, and making predictions, which is illustrated the workflow architecture in the figure 3.1 below.

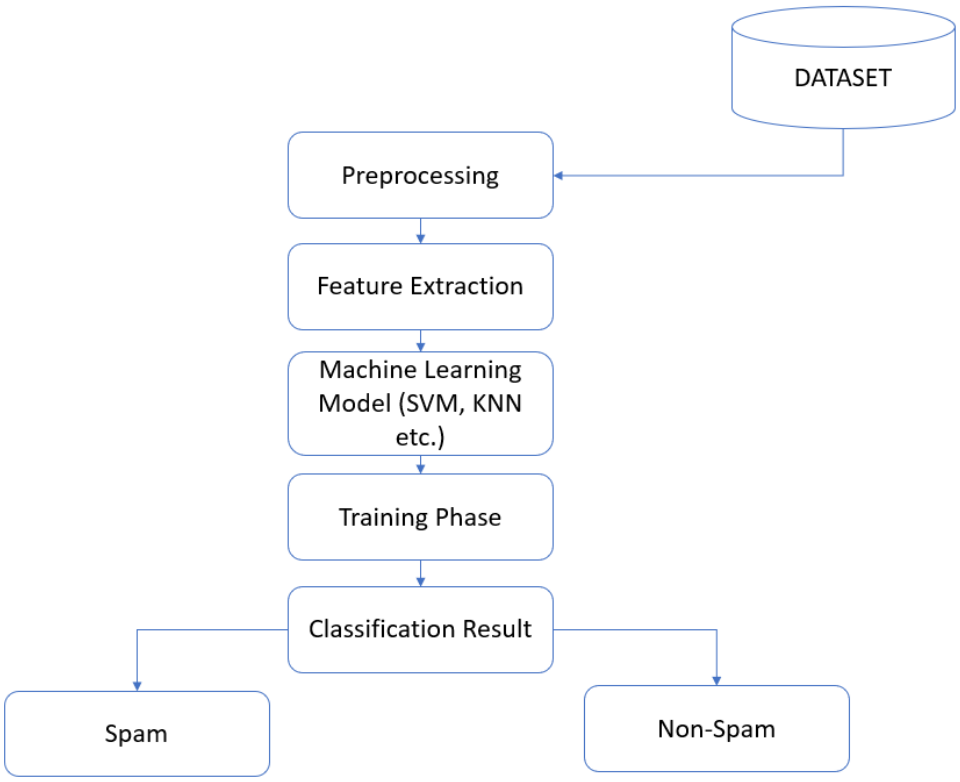

Figure 3.1: Workflow Architecture

3.1 Dataset

Our work stands out or is distinct in that we assembled the dataset through manual procedures, which adds a special element or originality to our research. To ensure a comprehensive and diverse dataset, a wide range of sources were considered, including emails, SMS messages, and websites. By manually gathering the dataset, we had complete authority over the quality and precision of the data, thereby enhancing the dependability and credibility of our findings. Although the process of manual collection was laborious and time-intensive, ensuring the accuracy and trustworthiness of the data utilized in our research was of utmost importance. The dataset consists of 2000 entries and 2 categories, where each message is categorized as either non-spam (0) or spam (1). Among the 2000 messages, 400 are identified as spam and 1600 as not spam. Considering the scarcity of spam messages in real-world scenarios, having 400 spam messages, which make up 20% of the dataset, is deemed satisfactory for the specific issue being addressed [15]. In the figure below 3.2, you can see an example of a dataset with 10 rows, which is used for illustrative purposes.

	text	spam
0	Плутон жоғарғы ғаламшарларының тобына кіреді, ...	0
1	Ресейде кофехана 1654 жылы пайда болады. Дәл о...	0
2	Дерекке сүйенсек, Бекет ата 1750 жылы туылып, ...	0
3	Сізді қолайлы қарым-қатынас шарттары әлі де кү...	1
4	Ол Ақ үйде 1837 жылы президенттің басқару уақы...	0
5	Күн сағаттары күн сәулесі арқылы уақыт анықтау...	0
6	Ақсай - Қарағанды облысы, Шалқар алабындағы өз...	0
7	Азаматтық - құқықтық жауапкершілік мынадай нег...	0
8	ЮНЕСКО Әлемдік мұра тізімінде Бельгияның 11 ны...	0
9	Үлкен қарқылдақ - біздегі қарқылдақтардың ішін...	0
10	Жаңа жылдық "Музыкалық жарыс" науқанына қатысы...	1

Figure 3.2: Dataset

3.2 Data Preprocessing

In machine learning, preparing the dataset is important as it involves converting it into a format that can be easily understood by computers. Our code includes

various techniques for preprocessing the data, which aim to enhance its quality and make it better suited for machine learning models.

Python's **string punctuation** is a predefined set of characters stored in the string module [16]. It includes all punctuation marks as a string. This feature is used to strip punctuation from a given sentence. The process of removing punctuation involves iterating over each character in the sentence and checking if it belongs to the string punctuation set. If it doesn't, the character is considered relevant and is retained in the modified text. Through the exclusion of punctuation, the model can concentrate on the meaningful words in the text instead of the distracting punctuation marks.

Stopwords are words that are very common and don't carry much important information. In Kazakh language, the word "Shylau" (Шылау) falls into this category, as depicted in Figure 3.3. These words don't have a significant meaning on their own and don't function as independent parts of a sentence. Removing stopwords has the potential to enhance performance by reducing the number of tokens to only those that are significant. Consequently, this could lead to an improvement in classification accuracy. Deciding when to eliminate stopwords does not follow a strict rule. However, if our goal involves tasks like Language Classification, Spam Filtering, or anything related to text classification, it would be advisable to remove stopwords.

```
['алайда', 'арқылы', 'бен', 'бенен', 'бері', 'бетер',
'бойы', 'болмаса', 'бірақ', 'бірге', 'бірде', 'біресе
', 'бірталай', 'бұрын', 'гөрі', 'дегенмен', 'дейін',
'жайлы', 'жайында', 'жуық', 'және', 'жөнінде', 'кейде
', 'кейін', 'мен', 'не болмаса', 'немесе', 'пен', 'пе
нен', 'сайын', 'салым', 'себебі', 'секілді', 'сияқты
', 'сол себепті', 'сонда да', 'сондайақ', 'сондықтан
', 'соң', 'сықылды', 'таман', 'тарта', 'таяу', 'тб',
'туралы', 'тәрізді', 'шамалы', 'шақты', 'шейін', 'я б
олмаса', 'яки', 'ғұрлы', 'қабат', 'қарай', 'қаралы',
'қатар', 'қоса', 'үшін', 'һәм', 'әйтеуір', 'әйткенмен
', 'әйтседе', 'әлде', 'әрі', 'өйткені']
```

Figure 3.3: Kazakh Stopword Lists

Lemmatization is a process of stripping words of their inflectional endings to reveal their base form. In our work, we utilized the kaz-nlp tool to perform morphological processing tasks, such as lemmatization. This set of tools offers a variety of capabilities that handle different Natural Language Processing (NLP) challenges. When it comes to lemmatization, the initial step involves splitting sentences into individual tokens, or words, which are then subjected to the "tag-sentence" method provided by the kaz-nlp tool [17]. This approach carries out the required analysis to recognize and separate the lemmas from the tokens, producing a standardized representation of the underlying meaning of the words. Figure 3.4 below illustrates lemmatization in Kazakh.

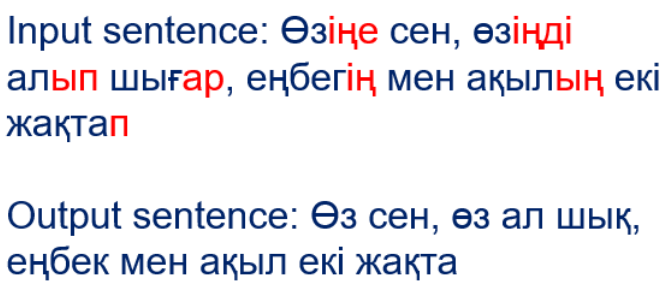

Figure 3.4: Lemmatization in Kazakh

We combined the aforementioned methods with various other preprocessing techniques, including converting to lowercase, tokenization, and removing links and numbers, into a single **preprocess-text** function. This function can be used as a pipeline to preprocess Kazakh datasets effectively.

The provided code performs preprocessing on a Kazakh dataset, as shown in the Appendix A. It starts by importing the essential modules and libraries, including string, pandas, and numpy. Additionally, it imports specific modules from the kaz-nlp library for tokenization, morphology analysis, and tagging. The code defines a function called "preprocess_text" that takes a message as input. Within this function, the message is converted to lowercase and several regular expressions are used to remove various elements from the text. HTTP links, bitly links, and www links are eliminated. Digits and punctuations are also removed. The text is then split into a list of words, and stop words are filtered out. Next, the code utilizes the loaded tagger to analyze each word in the message. It retrieves

the lemma (base form) of each word by extracting it from the tag, specifically targeting words with the "_R" tag. The lemmatized words are then concatenated to form the preprocessed text.

By applying the pipeline to every individual value within the dataset, we obtain preprocessed data as depicted in the Figure 3.5 below.

	text	spam	preprocessed_text
0	Қазан федералды университеті - көптеген факуль...	0	қазан федералды университет көптеген факультет...
1	Оның шәкірттік кезеңдегі өлеңдері (1907-11) жа...	0	о шәкірттік кезең өлеңдері жас ақын қоғамдық қ...
2	Көтеріліске қарапайым көшпелі қазақтардың едәу...	0	көтеріліс қарапайым көшпелі қазақ едәуір бөліг...
3	Құрылымдық - функцияналдық талдау құбылыстар м...	0	құрылымдық функцияналдық талда құбылыс процес ...
4	Техникалық ұсыныс - техникалық тапсырманы талд...	0	техникалық ұсыныс техникалық тапсырма талда нә...
5	Жансая Ғалымжанқызы Мусина 1985 жылы 12 наурыз...	0	жансая ғалымжанқызы мусина жылы наурыз батыс қ...
6	Біртұтас электр энергетикалық жүйе жоғары және...	0	біртұтас электр энергетикалық жүйе жоғары аса ...
7	Сіздің ғашығыңыз кіммен сөйлесетінін білгіңіз ...	1	сіз ғашығыңыз кім сөйлесетінін біл кел ме ол с...
8	Тілдер қатыстық және абсолюттік прогресс бағыт...	0	тіл қатыс абсолютті прогресс бағыт дам табиғи ...
9	Матисс тұс пен көлем арасындағы байланыс жәрде...	0	матисс тұс көлем ара байланыс жәрдем сызықтық...
10	ШҰҒЫЛ ЖАҢАЛЫҚ!!! Сіз біздің 1 000 000 теңге де...	1	шұғыл жаңалық сіз біз теңге дейінгі жүлделі дж...

Figure 3.5: Preprocessed Kazakh Dataset

3.3 Feature extraction

In our research, we employ the Term Frequency Inverse Document Frequency (TF-IDF) approach on preprocessed data to assess the importance of individual words in a text. TF-IDF consists of two components: the initial component is known as TF, and the subsequent component is referred to as IDF. TF determines how often a word appears in a document [18]. The value is obtained by dividing the number of times a specific word appears in a document by the total number of words in that document, as shown in Equation 3.3.1:

$$TF(t) = \frac{\text{Number of times a specific word t appears in a document}}{\text{Total number of words in the document}} \tag{3.3.1}$$

IDF measures the importance of a word. The IDF value is computed by taking

the logarithm of the ratio between the total number of documents in a collection and the number of documents that contain the specific word, as illustrated in Equation 3.3.2:

$$IDF(t) = log\_e(\frac{\text{Total number of documents}}{\text{Number of documents with word t in it}}) \tag{3.3.2}$$

In order for machine learning algorithms to handle and evaluate the textual data efficiently, we must first convert it into a numerical representation using TF-IDF. This allows us to find important patterns and insights within the text.

3.4 Data Split: Test and Train Division

In the subsequent step of the process, we divide the data into two parts: the training set and the testing set. The purpose behind this division is to train the model using the training set, and then evaluate its performance on the separate test set. The code snippet presented in Appendix B illustrates the dataset splitting process utilizing the train_test_split method [19].

The dataset was divided using the following parameters:

- x: Represents the features for the model.
- y: Corresponds to the labels indicating whether a message is spam or not.
- test_size: Specifies the proportion of the dataset to allocate for the test set. In this case, 30% (0.3) of the data was set aside for testing.
- random_state: Sets the random seed to ensure reproducibility. The value of 10 was used here.

After the dataset was split, the code prints the number of rows in the test set and the train set. The test set contains 600 rows, while the train set consists of 1400 rows. Additionally, each row in both sets has 400 columns, representing the features or attributes of the messages.

This division allows us to train the model on a majority of the data (70%) and assess its performance on the remaining unseen data (30%). It helps evaluate how

well the model generalizes to new, unseen messages and gives us insights into its effectiveness in detecting spam in the Kazakh language.

3.5 Training Phase

Once the data is prepared and the necessary packages are imported, we proceed to train various classifiers using the training set. During this phase, the model learns from the labeled data to identify patterns and characteristics that distinguish spam messages from non-spam messages. We explore a range of algorithms, including SVM, KNN, MNB, DT, RFC, and LR. These classifiers have different approaches and strengths, which enable us to thoroughly evaluate how well they perform for our particular task. In order to achieve our research aims, we intend to employ these classifiers and algorithms to extract valuable insights from our data and make accurate predictions. Moreover, these algorithms come with readily accessible Python libraries, making it easier to apply and integrate them into our project. In Python, we can utilize the scikit-learn library [20], which provides a wide range of tools and algorithms for various tasks, including spam classification. To train machine learning algorithms, we need to create an instance of the chosen algorithm by initializing its parameters. We can modify the algorithm's parameters and set up its behavior at this phase. Afterward, we need to use the training data to train the algorithm. This involves providing the algorithm with the input features (x_train) and the target labels (y_train). The algorithm will learn from the training data to find patterns and relationships in the data.

During the training phase, it is important to ensure that the dataset is diverse and representative of the different types of spam messages that may be encountered. By effectively training ML models for spam detection in the Kazakh language, we can create system that can accurately identify and filter out spam messages, improving the user experience and protecting against potential security risks.

Chapter 4

Results

4.1 Non-hybrid Machine Learning Models Results

The effectiveness of individual, non-hybrid algorithms is evaluated in this section using metrics including accuracy, precision, recall, and F1-score. These metrics give important information about the efficiency and dependability of the standalone algorithms. Accuracy 4.1.1 measures the overall correctness of the spam detection system by calculating the ratio of correctly classified spam and non-spam messages to the total number of messages. It is a general indicator of the system's effectiveness in correctly identifying spam and non-spam messages.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{4.1.1}$$

Precision 4.1.2 is a metric that quantifies the proportion of correctly classified spam messages among all the messages predicted as spam. It evaluates the system's ability to accurately label messages as spam, minimizing false positives.

$$Precision = \frac{TP}{TP + FP} \tag{4.1.2}$$

Recall 4.1.3 represents the proportion of actual spam messages correctly identified by the system. It assesses the system's ability to capture all spam messages

and avoid false negatives.

$$Recall = \frac{TP}{TP + FN} \tag{4.1.3}$$

The F1-score is a balanced measure that combines precision and recall into a single metric. The equation 4.1.4 for the F1-score is presented below:

$$Precision = \frac{TP}{TP + FP} \tag{4.1.4}$$

The implementation of Non-hybrid ML models is presented in Appendix C. The following table 4.1 presents the outcome of the evaluation measures for each classification technique applied to the test dataset:

	Accuracy	Precision	Recall	F1-Score
SVM	0.985000	0.973451	0.948276	0.960699
KNN	0.983333	0.956897	0.956897	0.956897
MNB	0.981667	0.972973	0.931034	0.951542
RFC	0.976667	0.947368	0.931034	0.939130
LR	0.948333	0.967033	0.758621	0.850242
DT	0.946667	0.956522	0.758621	0.846154

Table 4.1: Result metrics for non-hybrid models

Looking at the metrics we obtained, we can draw the following conclusions: SVM achieved the highest accuracy and a reasonably high f1-score, indicating a balanced performance in terms of precision and recall. LR and DT showed high precision but relatively low recall, indicating their ability to avoid false positives but potential difficulty in identifying all spam messages. DT exhibited lower recall and f1-score compared to other ML models, suggesting its lower performance. Overall, based on these metrics, SVM and KNN demonstrated the most effective performance in identifying spam messages. In the figure 4.1 below, we can observe the confusion matrix for SVM, which demonstrates the superior performance compared to other algorithms. It provides a comprehensive overview of how well the model performed in classifying messages as either spam or not spam.

In this particular case, the confusion matrix reveals the following results:

- True Positives (Spam): The model correctly identified 108 messages as

spam.

- False Negatives (Spam): However, the model incorrectly labeled 8 messages as not spam when they were actually spam.
- False Positives (Not Spam): Conversely, the model mistakenly classified 6 messages as spam when they were actually not spam.
- True Negatives (Not Spam): The model accurately recognized 478 messages as not spam.

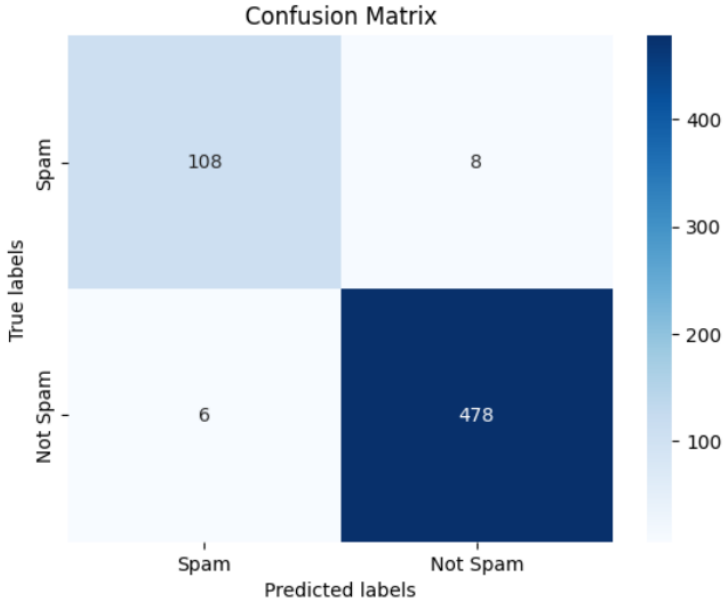

Figure 4.1: Confusion Matrix for SVM

By studying the confusion matrix, we gain valuable insights into the strengths and weaknesses of the SVM model in effectively classifying messages as spam or not spam in the Kazakh language.

4.2 Hybrid Machine Learning Models Results

Non-hybrid models like SVM and KNN are straightforward, quick, and convenient to understand and apply, rendering them appropriate for less complex tasks. However, these models rely solely on a single ML algorithm, which limits their performance to the capabilities of that specific algorithm. On the other hand, a

hybrid ML model for spam detection combines multiple ML algorithms to generate predictions. The benefit of a hybrid model lies in its ability to leverage the strengths of different algorithms, leading to enhanced overall performance. For instance, one algorithm might excel in precision but lack in recall, while another algorithm could demonstrate the opposite behavior. By integrating these algorithms, a hybrid model can strike a balance between precision and recall, which yields improved performance outcomes. The hybrid ML model we are working on combines the strengths of multiple algorithms, allowing us to tap into their collective power. This approach enhances our ability to make accurate predictions in spam detection. In our work, we combine different algorithms using the VotingClassifier technique. Specifically, we utilize SVM, MNB, RFC, KNN, DT, and LR to build a reliable spam detection model. In this method, predictions from various classifiers are combined using either hard or soft voting. In the case of hard voting (Figure 4.2), the final prediction is based on the majority decision of the classifiers. For instance, if SVM and KNN classify a message as spam while RFC classifies it as non-spam, the majority vote would classify the message as spam since two out of the three classifiers identified it as such. The simplicity of hard voting makes it straightforward to implement and interpret, making it a practical choice in scenarios where a clear decision is needed from the base classifiers.

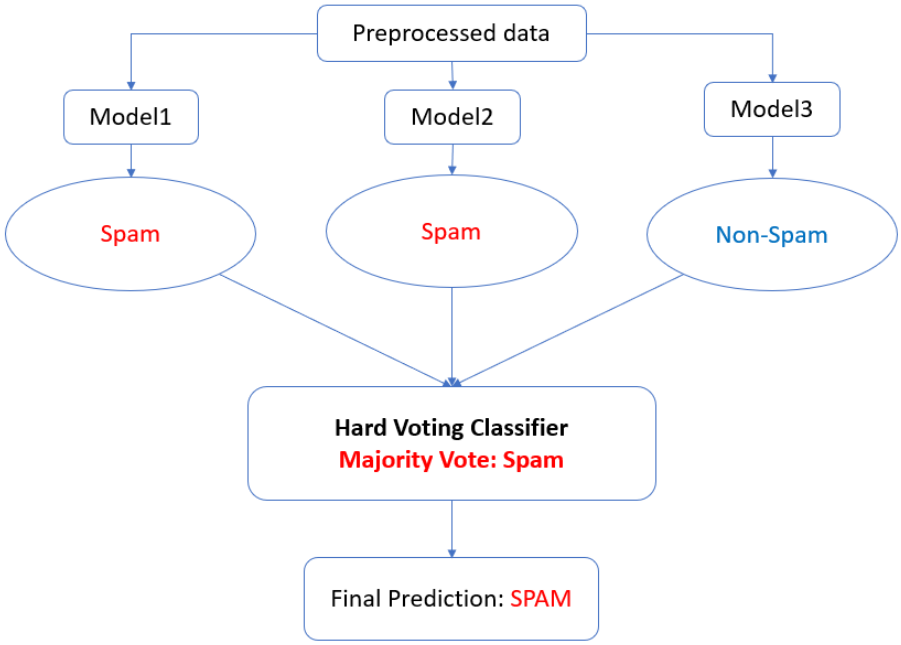

Figure 4.2: Hard Voting Classifier Architecture

In situations where the count of votes for each class is equivalent and there is an even number of classifiers in the ensemble, determining the majority vote can be challenging. To address this ambiguity, the VotingClassifier method employs a technique known as soft voting. Instead of making binary predictions (0 or 1), each classifier assigns probability scores to each class label (spam or non-spam). Let's consider an example where we have two classifiers, SVM and KNN. In this case, each classifier assigns probability scores to the two classes: spam and non-spam. In the figure 4.3 below, you can see a visual representation of their probability scores. The class label with the highest probability score is then selected as the final prediction. This approach avoids the need for tie-breaking strategies used in hard voting, such as random selection or weighted voting schemes [21].

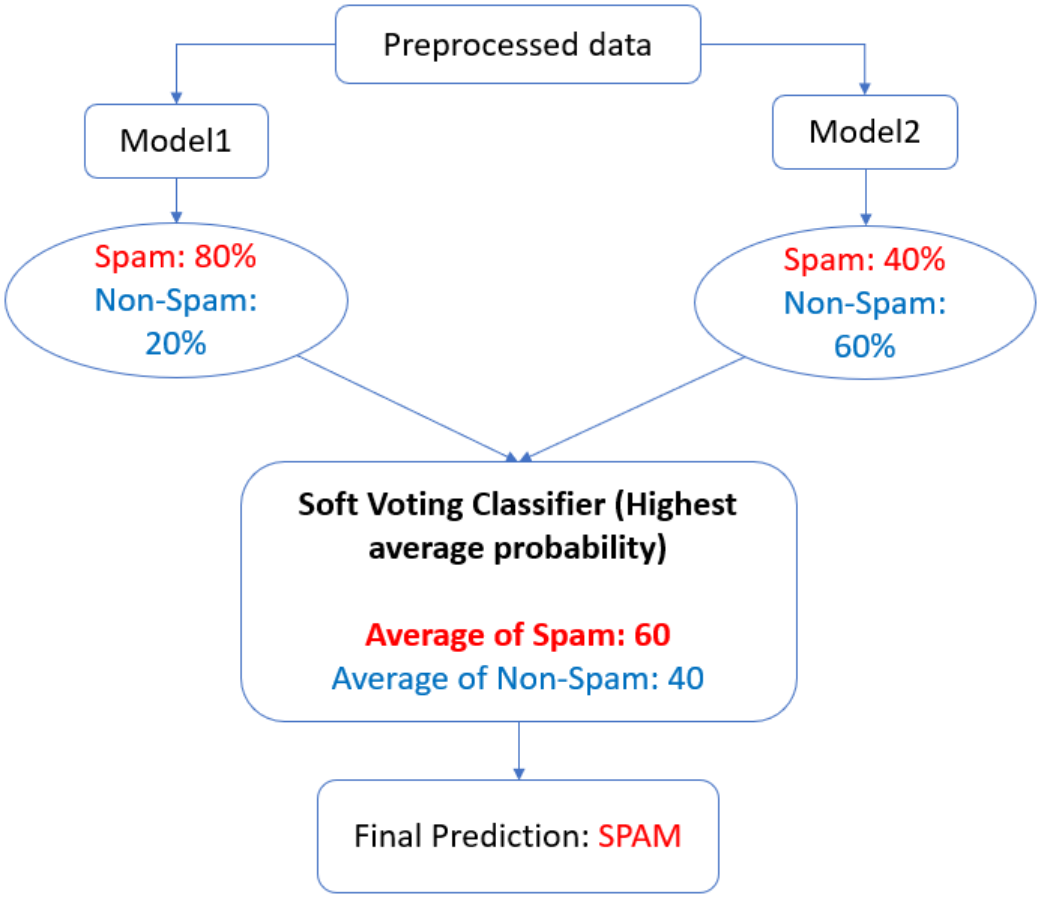

Figure 4.3: Soft Voting Classifier Architecture

The implementation of Hybrid ML models is presented in Appendix D. Outlined in the subsequent table 4.2 are the performance measures for the hybrid models:

	Accuracy	Precision	Recall	F1-Score
SVM + KNN	0.986667	0.982143	0.948276	0.964912
MNB + RFC	0.985000	0.965217	0.956897	0.961039
SVM + LR	0.983333	0.964912	0.948276	0.956522
RFC + KNN	0.983333	0.973214	0.939655	0.956140
MNB + KNN	0.981667	0.972973	0.931034	0.951542
SVM + MNB	0.980000	0.948276	0.948276	0.948276
SVM + RFC	0.980000	0.948276	0.948276	0.948276
MNB + LR	0.978333	0.981308	0.905172	0.941704
RFC + LR	0.965000	0.979798	0.836207	0.902326
SVM + DT	0.961667	0.951456	0.844828	0.894977
KNN + LR	0.958333	0.978947	0.801724	0.881517
MNB + DT	0.951667	0.957895	0.784483	0.862559
RFC + DT	0.950000	0.967391	0.767241	0.855769
KNN + DT	0.948333	0.977528	0.750000	0.848780
DT + LR	0.948333	0.956989	0.767241	0.851675

Table 4.2: Result metrics for hybrid models

From the obtained outcomes, it is evident that the hybrid model "SVM + KNN" displayed superior performance across various metrics. This particular hybrid model attained an accuracy level of 0.986667, indicating that it accurately classified 98.67% of the spam messages. With a precision of 0.982143, the model demonstrated a low rate of misclassifying non-spam messages as spam. Moreover, it achieved a recall of 0.948276, correctly identifying 94.83% of the actual spam messages. The F1-score, which takes into account both precision and recall, also yielded a value of 0.964912 for this hybrid model. This hybrid model "SVM + KNN" offers distinct advantages over using SVM alone, and we can understand these benefits by examining the confusion matrix, as depicted in Figure 4.4:

- True Positives (Spam): The hybrid model correctly classified 110 messages as spam.
- False Negatives (Spam): The hybrid model mistakenly labeled 6 messages as not spam when they were actually spam.
- False Positives (Not Spam): The hybrid model incorrectly classified 2 messages as spam when they were actually not spam.

- True Negatives (Not Spam): The hybrid model accurately recognized 482 messages as not spam.

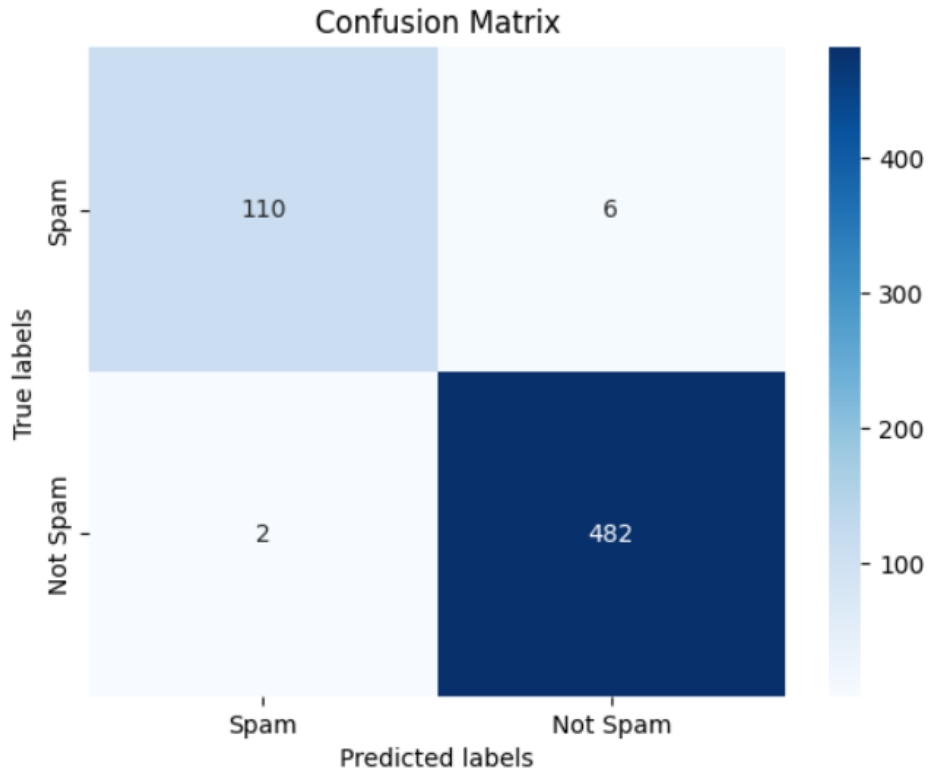

Figure 4.4: Confusion Matrix for Hybrid ML (SVM + KNN) model

The hybrid ML model, which combines SVM and KNN for spam detection in the Kazakh language, offers the following advantages over using SVM alone:

- Improved Accuracy: By combining the strengths of SVM and KNN, the hybrid model achieves higher accuracy in distinguishing between spam and not spam messages compared to stand-alone SVM. This is evident from the increased number of true positives and true negatives in the confusion matrix.
- Reduced False Negatives: With the hybrid model, there are fewer instances where spam messages are wrongly labeled as not spam. It does a better job at recognizing spam, so there are fewer mistakes in identifying them. This is essential in preventing spam messages from reaching the user and improving the overall satisfaction of the user.

- Effective Handling of False Positives: The hybrid model also demonstrates efficiency in minimizing false positives, which means fewer legitimate messages are incorrectly classified as spam. This reduces the chance of missing crucial information by making sure that critical non-spam messages are not inadvertently filtered out.

The findings indicate that hybrid model perform better at spam identification than traditional non-hybrid techniques, leading to improved spam detection capabilities and a more trustworthy filtering system.

4.3 User Interface

During this research, a user interface was developed for a spam detection system in the Kazakh language. By categorizing incoming messages as spam or not, this web application gives users a simple way to interact with the system. The user interface was developed using Python. This adaptability makes it simple to integrate with many systems, which enhances the interface's overall functionality. The user interface has a text area for users to enter their messages. To start the prediction process, users can click the "Болжау" button, which means "Predict" in Kazakh. To accomplish this, it goes through a sequence of processes. First, the input message is preprocessed to transform it into a suitable format for analysis. Then, the preprocessed text is vectorized using the TF-IDF technique. The vectorized input is then passed into a trained model for prediction. Behind the scenes, the algorithm responsible for making these predictions is a combination of SVM and KNN, which consistently produced the best accurate outcomes in our evaluations. Based on the prediction result, the code displays a corresponding header indicating whether the message is classified as spam or not. If the result is 1, indicating a spam message, the header is displayed in red with the label "Спам (Spam)", as presented in Figure 4.5.

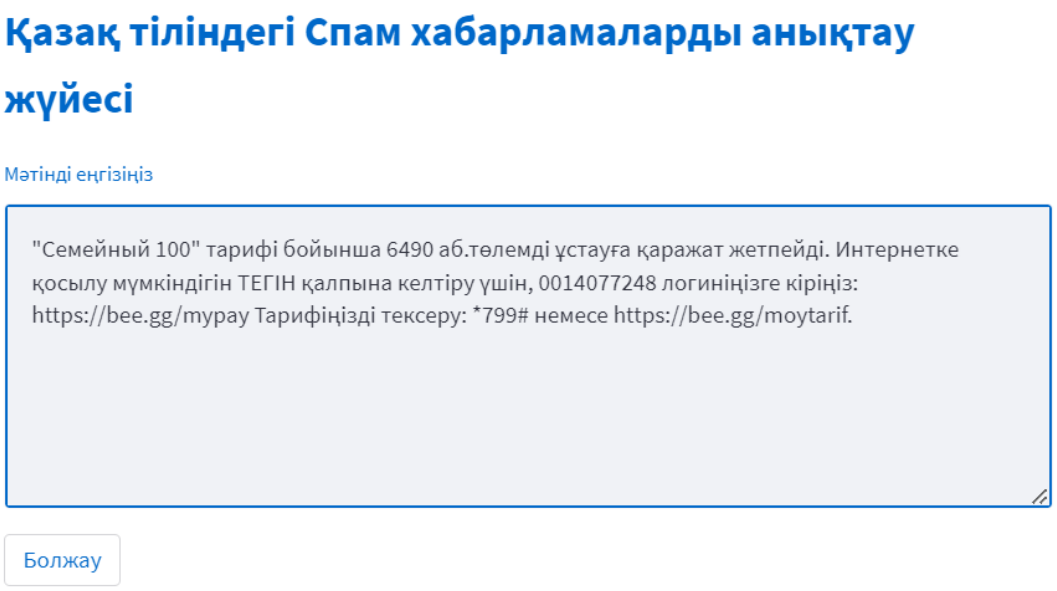

Figure 4.5: User Interface: Spam indication

Conversely, if the result is not 1, indicating a non-spam message, the header is displayed in blue with the label "Спам емес(Not spam)", as shown in Figure 4.6.

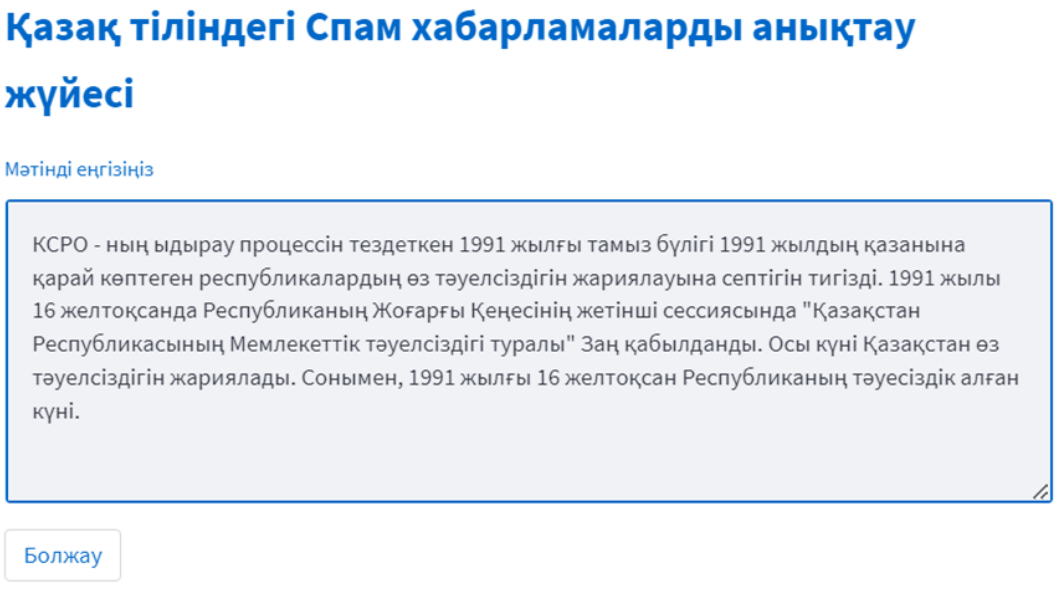

Figure 4.6: User Interface: Not Spam indication

With its intuitive design and effective functionality, this user interface empowers users to easily identify spam, thereby significantly improving the overall messaging experience in Kazakh.

Chapter 5

Discussion and Conclusion

The findings indicate that SVM and KNN exhibited strong performance in comparison to other models such as RFC, LR, MNB, and DT across all metrics. By combining SVM and KNN, their performance was further enhanced, highlighting the effectiveness of combining standalone well-performing models. It can be inferred that the hybrid model is more appropriate for this particular problem and dataset under examination, as it yields superior outcomes compared to employing a single model. Furthermore, an user interface has been developed, which is capable of classifying incoming messages as spam or non-spam. This interface allows users to input messages and obtain real-time classification results, providing a convenient solution for spam detection. Moving forward, there are plans to increase the size of the dataset, which is anticipated to have a positive impact on its performance. As part of our future work, we will focus on continual improvement and optimization of the hybrid ML model for spam detection. By doing so, we aim to create a spam detection solution that is not only highly effective and efficient but also in its ability to remain agile in responding to novel and emerging spamming techniques. Enhancing the processing of the Kazakh language has broader implications, not limited to its specific research context, but also extends to other languages facing similar difficulties. In summary, there exists considerable opportunity for future endeavors to advance the processing of the Kazakh language.

Bibliography

[1] KasperskyLab. Global spam volume as percentage of total e-mail traffic from 2011 to 2021. Feb. 2023. URL https://www.statista.com/statistics/420400/spam-email-traffic-share-annual.

[2] E.G. Dada et al. Machine learning for email spam filtering: review, approaches and open research problems. Heliyon, vol. 5, Jun. 2019. URL https://doi.org/10.1016/j.heliyon.2019.e01802.

[3] G.S. Karimovich, K.S. Jaloldin ugli, and O.I. Salimbayevich. Analysis of machine learning methods for filtering spam messages in email services. 2020 International Conference on Information Science and Communications Technologies (ICISCT), Nov. 2020. URL https://doi.org/10.1109/ICISCT50599.2020.9351442.

[4] G. Singh, B. Kumar, L. Gaur, and A. Tyagi. Comparison between multinomial and bernoulli naïve bayes for text classification. 2019 International Conference on Automation, Computational and Technology Management (ICACTM), pages 593–596, 2019. URL https://doi.org/10.1109/ICACTM.2019.8776800.

[5] L. Bansal and N. Tiwari. Feature selection based classification of spams using fuzzy support vector machine. 2020 International Conference on Smart Electronics and Communication (ICOSEC), pages 258–263, 2020. URL https://doi.org/10.1109/ICOSEC49089.2020.9215443.

[6] N.L. Octaviani, E. Hari Rachmawanto, C.A. Sari, and I.M.S. De Rosal. Comparison of multinomial naïve bayes classifier, support vector machine, and recurrent neural network to classify email spams. 2020 International

Seminar on Application for Technology of Information and Communication (iSemantic), pages 17–21, Sep. 2020. URL https://doi.org/10.1109/iSemantic50169.2020.9234296.

[7] Ş. Seral, P. Kemal, K. Halife, and G. Salih. A new hybrid method based on fuzzy-artificial immune system and k-nn algorithm for breast cancer diagnosis. Computers in Biology and Medicine, vol. 37(3):415–423, 2007. URL https://doi.org/10.1016/j.compbiomed.2006.05.003.

[8] M. RAZA, N.D. Jayasinghe, and M. M. A. Muslam. A comprehensive review on email spam classification using machine learning algorithms. 2021 International Conference on Information Networking (ICOIN), pages 327–332, Jan. 2021. URL https://doi.org/10.1109/ICOIN50884.2021.9334020.

[9] Z. Tembusai, H. Mawengkang, and M. Zarlis. K-nearest neighbor with k-fold cross validation and analytic hierarchy process on data classification. International Journal of Advances in Data and Information Systems, vol. 2, Jan. 2021. URL https://doi.org/10.25008/ijadis.v2i1.1204.

[10] A.I. Taloba and S.S.I. Ismail. An intelligent hybrid technique of decision tree and genetic algorithm for e-mail spam detection. 2019 Ninth International Conference on Intelligent Computing and Information Systems (ICICIS), pages 99–104, Apr. 2021. URL https://doi.org/10.1109/ICICIS46948.2019.9014756.

[11] C. Meda, E. Ragusa, C. Gianoglio, R. Zunino, A. Ottaviano, E. Scillia, and R Surlinelli. Spam detection of twitter traffic: A framework based on random forests and non-uniform feature sampling. 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pages 811–817, Aug. 2016. URL https://doi.org/10.1109/ASONAM.2016.7752331.

[12] M. Bassiouni, M. Shafaey, and E-S. El-Dahshan. Ham and spam e-mails classification using machine learning techniques. Journal of Applied Security Research, vol. 13:315–331, May 2018. URL https://doi.org/10.1080/19361610.2018.1463136.

[13] A. Wijaya and A Bisri. Hybrid decision tree and logistic regression classifier for email spam detection. 2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE), pages 1–4, Oct. 2016. URL https://doi.org/10.1109/ICITEED.2016.7863267.

[14] Sh. Khanday and S. Parveen. Logistic regression based classification of spam and non-spam emails. 2nd International Conference on ICT for Digital, Smart, and Sustainable Development (ICIDSSD), Mar. 2021. URL https://doi.org/10.4108/eai.27-2-2020.2303291.

[15] A. Ydyrys, Alua Bilakhanova, and N. Sultanova. Identifying spam messages for kazakh language using hybrid machine learning model. Suleyman Demirel University Bulletin: Natural and Technical Sciences, vol. 62 (1):142–150, 2023. ISSN 2709-2631. doi: 10.47344/sdubnts.v62i1.936. URL https://journals.sdu.edu.kz/index.php/nts/article/view/936.

[16] Python Software Foundation. Python documentation: string.punctuation. URL https://docs.python.org/3/library/string.html#string.punctuation.

[17] O. Makhambetov, A. Makazhanov, I. Sabyrgaliyev, and Zh. Yessenbayev. Kaznlp: Nlp tools for kazakh language. Nov. 2018. URL https://github.com/makazhan/kaznlp.

[18] G. Ubale and S. Gaikwad. Sms spam detection using tfidf and voting classifier. 2022 International Mobile and Embedded Technology Conference (MECON), pages 363–366, 2022. URL https://doi.org/10.1109/MECON53876.2022.9752078.

[19] F., G. Varoquaux, A. Gramfort, O. Grisel, L. Hermida, R. Martynov, et al. scikit-learn: train_test_split. URL https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, et al. scikit-learn: Machine learning in Python. Journal of Machine Learning Research, vol. 12:2825–2830, 2011. URL https://scikit-learn.org.

[21] C. Cornelio, M. Donini, A. Loreggia, et al. Voting with random classifiers (vorace): theoretical and experimental analysis. Autonomous Agents and Multi-Agent Systems, vol. 35, May 2021. URL https://doi.org/10.1007/s10458-021-09504-y.

Appendix A

Data Preprocessing Pipeline for Kazakh lanuguage

```
import numpy as np
import pandas as pd
import string
import os
import re
from kaznlp.tokenization.tokhmm import TokenizerHMM
from kaznlp.morphology.analyzers import AnalyzerDD
from kaznlp.morphology.taggers import TaggerHMM

analyzer = AnalyzerDD()
analyzer.load_model(os.path.join('kaznlp', 'morphology','mdl'))
mdl = os.path.join('kaznlp', 'tokenization', 'tokhmm.mdl')

tagger = TaggerHMM(lyzer=analyzer)
tagger.load_model(os.path.join('kaznlp', 'morphology','mdl'))

def preprocess_text(message):
    #to lowercase
    message = message.lower()
    # remove http links
    message = re.sub(r'http\S+', '', message)
    # remove bitly links
    message = re.sub(r'bit.ly/\S+', '', message)
    # remove www links
    message = re.sub(r'www\S+', '', message)
    # remove digits
    message = re.sub(r'\d+', '', message)
```

```
28      #remove punctuations
29      message = re.sub(r'[^\w\s]', '', message)
30      list_of_words = message.split()
31      # remove stop words
32      message = [t for t in list_of_words  if t not in stop_words]
33
34      texts = ''
35      for i, a in enumerate(tagger.tag_sentence(message)):
36          lemma_reg = re.search('(.+?)_R', a)
37          if lemma_reg:
38              lemmatized_words = lemma_reg.group(1)
39          texts += lemmatized_words + " "
40
41      return texts
```

Appendix B

The dataset splitting

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split

tfidf = TfidfVectorizer(max_features=400)
x = tfidf.fit_transform(df['preprocessed_text']).toarray()
y = df['spam'].values

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
    random_state=10)

print('rows in test set: ' + str(x_test.shape))
print('rows in train set: ' + str(x_train.shape))
```

Appendix C

Implementation of Non-Hybrid ML models

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix,
    precision_score, recall_score, f1_score

mnb = MultinomialNB()
svm = SVC(kernel='sigmoid', gamma=1.0, probability=True)
dt = DecisionTreeClassifier(max_depth=5)
rfc = RandomForestClassifier(n_estimators=50, random_state=10)
knn = KNeighborsClassifier(n_neighbors=20)
lr = LogisticRegression(solver='liblinear', penalty='l1')

def train_clsf(clsf, x_train, y_train, x_test, y_test):
    clsf.fit(x_train, y_train)
    y_pred = clsf.predict(x_test)
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    rec = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    return accuracy,precision,rec,f1

clsfs = {
    'MNB': mnb,
```

```
    'SVM' : svm,
    'DT': dt,
    'RFC': rfc,
    'KNN' : knn,
    'LR': lr,
}

acc_score = []
prec_score = []
rec_score = []
f1_score = []

for name,clsf in clsfs.items():
    acc,precision,rec,f1 = train_clsf(clsf, x_train, y_train, x_test, y_test
    )
    print(name)
    print("Accuracy:", acc)
    print("Precision:", precision)
    print("Recall:", rec)
    print("F1-Score:", f1)
    acc_score.append(acc)
    prec_score.append(precision)
    rec_score.append(rec)
    f1_score.append(f1)
```

Appendix D

Implementation of Hybrid ML models

```
from sklearn.ensemble import VotingClassifier

svm_knn=VotingClassifier(estimators=[('SVM',svm),('KNN',knn)],voting='soft')
svm_mnb=VotingClassifier(estimators=[('SVM',svm),('MNB',mnb)],voting='soft')
svm_dt=VotingClassifier(estimators=[('SVM',svm),('DT',dt)],voting='soft')
svm_rfc=VotingClassifier(estimators=[('SVM',svm),('RFC',rfc)],voting='soft')
svm_lr=VotingClassifier(estimators=[('SVM',svm),('LR',lr)],voting='soft')
mnb_knn=VotingClassifier(estimators=[('MNB',mnb),('KNN',knn)],voting='soft')
mnb_lr=VotingClassifier(estimators=[('MNB',mnb),('LR',lr)],voting='soft')
mnb_dt = VotingClassifier(estimators=[('MNB',mnb),('DT',dt)],voting='soft')
mnb_rfc=VotingClassifier(estimators=[('MNB',mnb),('RFC',rfc)],voting='soft')
rfc_lr=VotingClassifier(estimators=[('RFC',rfc),('LR',lr)],voting='soft')
rfc_dt=VotingClassifier(estimators=[('RFC',rfc),('DT',dt)],voting='soft')
rfc_knn=VotingClassifier(estimators=[('RFC',rfc),('KNN',knn)],voting='soft')
knn_lr=VotingClassifier(estimators=[('KNN',knn),('LR',lr)],voting='soft')
knn_dt=VotingClassifier(estimators=[('KNN',knn),('DT',dt)],voting='soft')
dt_lr = VotingClassifier(estimators=[('DT', dt),('LR', lr)],voting='soft')

def train_clsf(clsf, x_train, y_train, x_test, y_test):
    clsf.fit(x_train, y_train)
    y_pred = clsf.predict(x_test)
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    rec = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    return accuracy,precision,rec,f1

```

```
clsfs = {
    'SVM + KNN': svm_knn,
    'SVM + MNB': svm_mnb,
    'SVM + RFC': svm_rfc,
    'SVM + DT': svm_dt,
    'SVM + LR': svm_lr,
    'MNB + KNN': mnb_knn,
    'MNB + DT': mnb_dt,
    'MNB + RFC': mnb_rfc,
    'MNB + LR': mnb_lr,
    'RFC + DT': rfc_dt,
    'RFC + KNN': rfc_knn,
    'RFC + LR': rfc_lr,
    'KNN + DT': knn_dt,
    'KNN + LR': knn_lr,
    'DT + LR': dt_lr,
}

acc_score = []
prec_score = []
rec_score = []
f1_score = []

for name,clsf in clsfs.items():
    acc,precision,rec,f1 = train_clsf(clsf, x_train, y_train, x_test, y_test
    )
    print(name)
    print("Accuracy:", acc)
    print("Precision:", precision)
    print("Recall:", rec)
    print("F1-Score:", f1)
    acc_score.append(acc)
    prec_score.append(precision)
    rec_score.append(rec)
    f1_score.append(f1)
```