GRADUATE PROGRAMS INSTITUTE

RESEARCH PAPER CLASSIFICATION USING TEXT MINING APPROACHES

Yüksel KOÇ

MASTER'S THESIS

İstanbul, January 2024

RESEARCH PAPER CLASSIFICATION USING TEXT MINING APPROACHES

Yüksel KOÇ

MASTER'S THESIS
COMPUTER ENGINEERING
DEPARTMENT
COMPUTER ENGINEERING MASTER'S PROGRAM
WITH THESIS

MASTER'S THESIS ADVISOR
Asst. Prof. Dr. Burçin KÜLAHÇIOĞLU

MASTER'S THESIS JURY MEMBERS
Prof. Dr. Abdurazzag Ali A ABURAS

Asst. Prof. Dr. Zeynep Behrin Güven AYDIN

ACKNOWLEDGMENTS

I am grateful to my advisor, Burçin KÜLAHÇIOĞLU, for her advice, support, and encouragement during the thesis process. Her knowledge and constructive input helped shape the path of my study and improve the quality of this work.

I would like to thank my family and friends for their support and understanding during my studies. Their support gave me the strength and motivation to overcome challenges and pursue my academic goals.

January 2024

Yüksel KOÇ

CONTENTS

ABSTRACT

RESEARCH PAPER CLASSIFICATION USING TEXT MINING APPROACHES

With the exponential growth of digital text data, the need for effective ways to manage and organize them electronically has made natural language processing technique and methods more important than ever before. Classification of research articles by text mining methods is an important aspect of scholarly communication as it provides a systematic way to organize and classify research publications according to their topics, methodologies, and other relevant characteristics. Classification of research articles is an important tool for effective access and discovery of research publications, facilitating research synthesis and analysis, promoting interdisciplinary research, and quality assurance. In this thesis, different classification models were used and their performances were compared. A system is proposed to help researchers search for research articles more efficiently. The data set was studied using the R language and the results were shown with an application prepared with R Shiny.

Keywords: Research paper classification; machine learning; deep learning

Date: 29.01.2024

ÖZET

METİN MADENCİLİĞİ YAKLAŞIMLARI KULLANILARAK AKADEMİK BELGE TÜRLERİNİN SINIFLANDIRILMASI

Dijital metin verilerinin katlanarak büyümesiyle birlikte, bunları elektronik ortamda yönetmenin ve organize etmenin etkili yollarına olan ihtiyaç, doğal dil işleme teknik ve metotlarını her zamankinden daha önemli hale getirmiştir. Metin madenciliği metotları ile araştırma makalelerinin sınıflandırması, araştırma yayınlarını konularına, metodolojilerine ve diğer ilgili özelliklerine göre organize etmek ve sınıflandırmak için sistematik bir yol sağladığından bilimsel iletişimin önemli bir yönüdür. Araştırma makalelerinin sınıflandırılması; araştırma yayınlarına etkili bir şekilde erişim ve keşif, araştırma sentezi ve analizinin kolaylaştırılması, disiplinlerarası araştırmanın teşvik edilmesi ve kalite güvencesi için önemli bir araçtır. Bu tezde farklı sınıflandırma modelleri kullanılmıştır ve performansları karşılaştırılmıştır. Araştırmacıların araştırma makalelerini daha verimli aramalarına yardımcı olacak bir sistem önerilmiştir. Veri seti üzerinde R dilini kullanılarak çalışılmış ve sonuçlar R Shiny ile hazırlanan bir uygulama ile gösterilmiştir.

Anahtar Sözcükler: Araştırma belgelerinin sınıflandırılması; makine öğrenmesi; derin öğrenme

Tarih: 29.01.2024

ABBREVIATIONS

ANN	: Artificial Neural Network
BERT	: Bidirectional Encoder Representations from Transformers
BoW	: Bag Of Words
DF	: Document Frequency
GLMNET	: Lasso and Elastic-Net Regularized Generalized Linear
LSTM	: Long Short-Term Memory
NB	: Naïve Base
NLP	: Natural Language Processing
RNN	: Recurrent Neural Network
SVM	: Support Vector Machine
TF	: Term Frequency
TF-IDF	: Term Frequency-Inverse Document Frequency
XGBoost	: Extreme Gradient Boosting
XGBTree	: Extreme Gradient Boosting Tree

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

There is no doubt that the most valuable asset today is information. Everything that has been researched, discovered and produced since the beginning of human history has been transferred to our time and formed our current body of knowledge. For a significant part of this long period, information was not aggregated and easily accessible. Especially with the rapid development and widespread use of digital technologies, these masses of information have begun to be integrated into the digital environment.

Nowadays, information is easily accessible. We can access very comprehensive information with a single click. But how quickly and with how much accuracy can we access the information we are looking for in this sea of information? Regarding our subject, when academic publications in text format are transferred to the digital environment, situations such as distortions in the texts, the addition of foreign characters may occur, and the information is in an unstructured state. Therefore, extracting information and insight from these texts can be complex and time-consuming. At this stage, text mining, natural language processing and machine learning methods come into play. By using these technologies together, great progress has been made in classifying information and accessing accurate information.

Text classification issues include a wide range of challenges that are presented in the literature. These include author-work matching, email categorization, spam detection, text subject identification, and sentiment analysis. The key solving such challenges in the areas of classification and clustering is to extract meaningful representations from textual input. Information gain, hidden meaning indexing, and the display of word or phrase frequencies are examples of widely used text representation techniques in the literature (Baharudin et al., 2010).

Word representation techniques that contain words as vectors have become more popular as artificial neural networks (ANN) gain traction. Prominent instances of ANN-based word representation techniques include GloVe, Word2Vec, and FastText (Chiu and Baker, 2020). There exist models that go beyond word vectorization and turn entire sentences or paragraphs into vectors. The growth of representation strategies in modern research is a reflection of the complexity and dynamism involved in tackling the many-faceted problems of text categorization.

Using pre-processing techniques is essential when trying to extract the best features to accurately describe a dataset. This stage is important for the effectiveness of classification efforts in addition to helping to refine the dataset. Pre-processing entails a number of procedures, including careful removal of undesired parts from texts, morphological analysis, and spelling error repair. Term strings that are considered irrelevant for text representation, punctuation, digits, and foreign words are all removed methodically. The dataset receives a revolutionary purification through this careful curation, guaranteeing that the analysis and classification that follow are based on high-quality and relevant textual data. Thus, the deliberate use of pre-processing techniques serves as a catalyst to improve the precision and effectiveness of text analysis categorization procedures (Kowsari et al., 2019).

.

Our aim in this work is to classify academic publications using text mining and machine learning approaches. While making this classification, we preferred the R language, which is used in applications in the field of data science and statistics. We believe that it will be useful in providing users with the opportunity to use a different language and platform in academic publication research. In this report, we describe the project steps, technologies used and results in detail.

The remainder of this thesis as follows: In Chapter 2, we explore related studies on the subject, data mining, text mining, classification models and datasets. In Chapter 3, we prepare a combined dataset in CSV form. We then start implementing the R codes for text mining and classification process with different models. In Chapter 4, we develop an interface with R shiny for viewing all the information and results about our work. Finally, in Chapter 5, we discuss the results of models.

2 RELATED WORKS

In this section, we examined text mining methods and machine learning approaches used for text classification. We investigated previous studies on this subject, the approaches used in these studies, and the results obtained.

Upon examination of the data utilized in the data mining experiments conducted thus far, it seems that they are grounded in relational data structures. Most of this information is included in text databases. Text databases contains articles, newspapers, periodicals, digital libraries, documents from a range of sources, such as e-mails and web pages. The significance of its bases grew as a result of the internet's fast growth in linked text data publications. Currently, a lot of institutions, including administrative and industrial groups, keep their information electronically in text databases.

The need for text databases is growing, and conventional methods of information retrieval are no longer adequate. Without having a complete understanding of the texts' substance, it is challenging to formulate pertinent questions and obtain the data required to evaluate them. Because of this, people have attempted to create algorithms that compare several manuscripts and highlight the elements that make the relationships between them clear.

Statistical probability values were computed based on the frequency of occurrence of the terms that comprise the texts in certain research designed to classify them. When statistical techniques are insufficient, word meanings and the text's grammatical structure were scrutinized. Numerous research on information retrieval have been conducted in order to model the data gathered using various techniques and to develop appropriate algorithms.

Rule-based systems underpinned natural language processing until the 1980s (Guida and Mauri, 1986). Later on, when machine learning techniques were more widely used and processors became more advanced, research started to be done on actual issues that rule-based symbolic systems were unable to handle.

Similar to other machine learning fields, natural language processing approaches and issues started to yield better outcomes with the advent of deep learning. Word2Vec, one of the most popular techniques to learn word embedding, was created by Mikolov et al. in 2013 and is one of the most significant research presented following deep learning (Mikolov et al., 2013). This research made it possible to convey words in a vector format without losing their meaning. ANN was used to vectorize the words, and to prevent meaning loss, consideration was given to the words to the right and left of the words. More favorable outcomes were obtained by models trained with a very basic artificial neural network, comprising only one hidden layer, as opposed to more intricate feedforward and recurrent neural networks.

Another important study is using of recurrent neural networks in speech recognition in a study conducted by Graves et al. in 2013. GMM (Gaussian Mixture Model) and HMM (Hidden Markov Model) models have shown incredibly excellent outcomes when bi-directional LSTM networks are used, together with voice datasets from TIMIT and the Wall Street Journal (Graves et al., 2013).

In a study conducted in 2013 convolutional neural networks, which are widely used in the field of computer vision, were used in the sentence modeling studied by Kalchbrenner et al. (Kalchbrenner et al., 2014). For the purpose of semantic modeling phrases, a network with dynamic k-max pooling was developed.

In identifying interrogative phrases, dynamic neural networks which are easily applicable to any language have demonstrated remarkable performance. Another study using convolutional neural networks is sentence classification by Kim in 2014 (Kim, 2014). The single convolution layer network trained using Word2Vec word vectors has gave the most successful results for four of the seven different data sets used.

Sequence to sequence models, which generate output by making word predictions, are the subject of another study using LSTM networks. The study conducted by Sutskever et al. in 2014 is a model used to map a word sequence to another word sequence (Sutskever et al., 2014). The word sequence that is accepted as input in this machine translation model is vectorized using an LSTM model, and the word sequence generated from the vector is decoded using another LSTM model.

Conneau et al. They have presented a new architecture, VDCNN (Very Deep CNN), for text processing, which works directly at the character level and uses only small convolutions and pooling operations (Conneau et al., 2016). The model, which operates on characters the smallest representation units of texts was inspired by deep networks that have been successful in the field of computer vision. Three-layer convolution and maximum pooling layers are used in local operations. The model was applied to eight distinct data sets, and when the number of layers was raised to twenty-nine, it was found that the outcomes got better.

BERT (Bidirectional Encoder Representations from Transformers), created by Google workers Devlin et al. in 2018, is a model intended to pre-train bidirectional text representations, based on recent significant research in natural language processing (Devlin et al., 2018). The goal of this methodology is to give people searching on Google more precise results.

2.1 TEXT CLASSIFICATION

One of the core methods in natural language processing is text classification, which is the process of taking information from raw text input and classifying it according to that information. Many text classification models have been created throughout time to address various facets of this difficult problem. Among them, two key models stand out: single label and multi-label text classifications. Each is designed for a specific text classification job.

Every document or text passage in the single-label text categorization model is placed into one predetermined category. When a text entity falls into a single discrete class or category, this strategy works effectively. By giving each text occurrence a precise and unambiguous classification, it streamlines the classification process.

Conversely, the multi-label text categorization approach allows for scenarios in which a text passage or document may concurrently fall under more than one category. This adaptable strategy is especially helpful when handling complicated and varied information that may touch on a number of different subjects or themes. A single text passage can have many pertinent labels applied to it, providing for a more sophisticated comprehension of the material thanks to multi-label text categorization.

The single label and multi-label text classification models are two approaches that cover distinct classification requirements and provide flexibility in handling the diverse characteristics of textual data in various applications and domains. Figure 2.1 shows text classification pipeline overview.

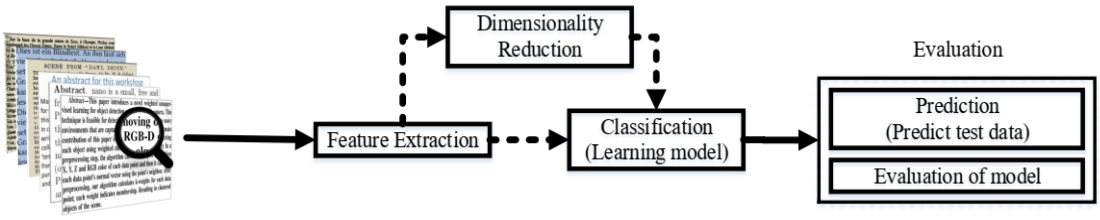


Figure 2.1 Text Classification Pipeline Overview (Kowsari et al., 2019)

Feature Extraction: In machine learning and data analysis, the process of selecting, transforming, or deriving pertinent information from raw data to produce a condensed, more informative representation known as features is called feature extraction. Machine learning algorithms use these qualities as input to recognize patterns, correlations, or other properties in the data. Feature extraction is very useful when working with unstructured or high-dimensional data. Term Frequency-Inverse Document Frequency (TF-IDF), Term Frequency (TF), Word2Vec, and Global Vectors for Word Representation (GloVe) are typical feature extraction approaches.

Dimensionality Reduction: Dimensionality reduction refers to the act of reducing the quantity of input variables or features inside a dataset. Simplifying the dataset while keeping as much of its crucial information as feasible is the main objective. Analyzing and analyzing high-dimensional data, where each data point includes several attributes, may be difficult. In order to overcome these difficulties, dimensionality reduction techniques convert the data into a lower-dimensional space. It could be more effective to use dimensionality reduction for pre-processing rather than creating cheap classifiers.

Classification Techniques: In machine learning, the term "classification techniques" refers to a group of formulas and procedures used to assign labels to objects and group them into predetermined classes or categories. Developing a model that can identify patterns in labeled training data and then forecast the class labels of novel, unknown data points is the main objective of classification. In order to train an algorithm to generate predictions on fresh, unseen data, classification is a supervised learning activity. The algorithm is trained on a dataset with known labels.

Evaluation: The last step in the text categorization process is critical, and a thorough understanding of a model's functioning is necessary for text categorization strategies to be effective. A variety of assessment techniques must be used in order to evaluate supervised learning processes. Even though accuracy calculation is the most straightforward evaluation method, it is useless for data sets that are unbalanced. In order to obtain a more nuanced view of a model's performance, practitioners frequently investigate additional measures including accuracy, recall, F1-score and precision, especially when working with unequal class distributions. This all-encompassing method of evaluation guarantees a more precise and insightful assessment of the text classification model's capabilities.

2.1.1 Single Label Text Classification

One type of natural language processing (NLP) work is single-label text classification, which is giving a document, phrase, or paragraph that contains text a single predetermined label or category. The main goal is to assign the text to the one class or category that most accurately describes its content. In order to enable a machine learning model to predict the correct label for instances that are not yet visible, this method usually entails training the model to identify patterns and characteristics in the text data.

Although single-label classification is theoretically easier than multi-label classification, the model must accurately distinguish between closely related groups, and problems may occur when dealing with ambiguous or overlapping categories. The labeled dataset's quality and feature selection have a big influence on how well the model performs in single-label text classification tasks.

2.1.2 Multi Label Text Classification

A procedure called "multi-label text classification" in NLP enables a given text segment, such as a document, phrase, or paragraph, to be concurrently connected with several predefined labels or categories. Unlike single-label text classification, which gives a single label to every instance, multi-label text classification allows the assignment of many labels based on the numerous components or themes present in the text. The goal is to capture the complexity and range of information that might cover several problems or themes in a single occurrence.

Some of the challenges associated with multi-label text categorization include handling label dependencies, managing imbalanced datasets, and ensuring the model can accurately anticipate a varied number of labels for different occurrences. Metrics like accuracy, recall, and F1-score for each label can be incorporated into the model evaluation in multi-label classification. Furthermore, multi-label classification issues are often solved using techniques like label powerset and binary relevance.

Comparative studies of text classification techniques Yang's 1999 study (Yang, 1999) is among the earliest. It uses a data set to test the performance of Reuters classifiers such KNN, LLSF WORD, and Naive Bayes.

In their study in 2002, Leopold and Kindermann examined the impact of model and computational complexity of the TF-IDF technique weights on text classification performance using support vector machines (Leopold and Kindermann, 2002). The attributes represented by term frequencies are more significant to Kernel when using support vector machines to categorize text than the selection of they shown.

In order to address the difficulties encountered in learning, in 2006 Zhang and Zhou introduced (Zhang and Zhou, 2006) an ANN-based technique dubbed BP-MLL (Backpropagation for Multi Label Learning). They used this method to tackle practical issues like text categorization and functional genomics, and they contrasted it with other well-known learning techniques.

Sun et al. They examined text classification techniques with imbalanced data (Sun et al., 2009). A classification model was put out in order to arrange the various text classification tactics, and the effectiveness of these strategies was then examined using the model as a basis for analysis.

Zhang et al. They examined the effectiveness of text representation techniques such TF-IDF, LSI, and multi-words in classification using datasets in both English and Chinese in 2011 (Zhang et al., 2011). SVM was used to classify texts, and the effectiveness of the representation techniques was contrasted.

Jin et al. They carried out one of the classification experiments where word embeddings, also known as word embeddings, were utilized as text representations in 2016 (Jin et al., 2016). This study employed Naive Bayes as a classifier, assuming that words will exhibit distinct vector distributions in different contexts. It obtained good performance on two sets of data, one of which was balanced and the other of which was not. Studies using fuzzy logic-based methodologies for text categorization have been carried out in addition to the conventional methods.

Text categorization has been the subject of several research projects in addition to those listed in the literature. Through their examination of the literature, Dhar et al. They classified text categorization systems in 2020 into many categories, including classical, fuzzy logic, deep learning, and graph-based (Dhar et al., 2021).

2.2 TEXT CLASSIFICATION TECHNIQUES

In the previous section, we said that text classification is generally done in two different types in terms of results: single-label and multi-label. When we look technically, we see that there are two approaches to text classification. These are the statistical approach and the machine learning approach. There are many techniques used in both types of approaches. Each of these techniques has advantages and disadvantages.

Statistical techniques are purely mathematical processes, and they act as the mathematical foundation for all other text classifiers. It works similar to a computer program, executing the given instructions without any ability of its own. To achieve a good classification, the amount of information to be handled by the application has to be concise and it is achieved by reducing the dimensionality in the data (Thangaraj and Sivakami, 2018).

The increasing amount, pace, and diversity of data necessitates the automation of text processing operations, especially text classification. Many tactics have been developed to meet this demand. In certain cases, automation is accomplished by creating a set of logical rules to categorize documents using knowledge engineering approaches, frequently guided by expert judgments. With this method, the categorization process may be automated more easily since it offers a structured framework that is led by human expertise.

The categorization problem is under the purview of the supervised learning principle in the field of machine learning. Before beginning the actual classification process, the system is trained and evaluated using preset class information in supervised learning. For the algorithm to discover patterns and connections between features and classes, labeled data is necessary, with each instance being linked to a predetermined class.

We used Naive Bayes (NB), Support Vector Machines (SVM), Long Short Term Memory (LSTM), Bidirectional Encoder Representations from Transformers (BERT), Lasso and Elastic-Net Regularized Generalized Linear (GLMNET) model, eXtreme Gradient Boosting (xgbTree) that we used in our study. We will share detailed information about classification techniques and the results we obtained in the next section.

3 PROPOSED PROJECT METHODOLOGY

In this section, we will explain the steps of our study in detail. Firstly, a suitable dataset for our study was searched and suitable datasets were obtained. A sample dataset was created in a way that was balanced. Then, the dataset was cleaned in detail from unnecessary characters and words for information extraction. This step is very important for the algorithms used to make faster and more accurate classification.

Since we will use our dataset to both train the model and test the model, 80% of the dataset is reserved for model training and 20% as test data. In the next stage, models were created with the selected statistical and machine learning algorithms and these models were trained with the training data. In the last stage, the models were tested with test data. The confusion matrix and result datasets of each model were created and saved in CSV (Comma-separated values) format to be displayed later in the R shiny interface.

The basic flow chart of our work shown In Figure 3.1.

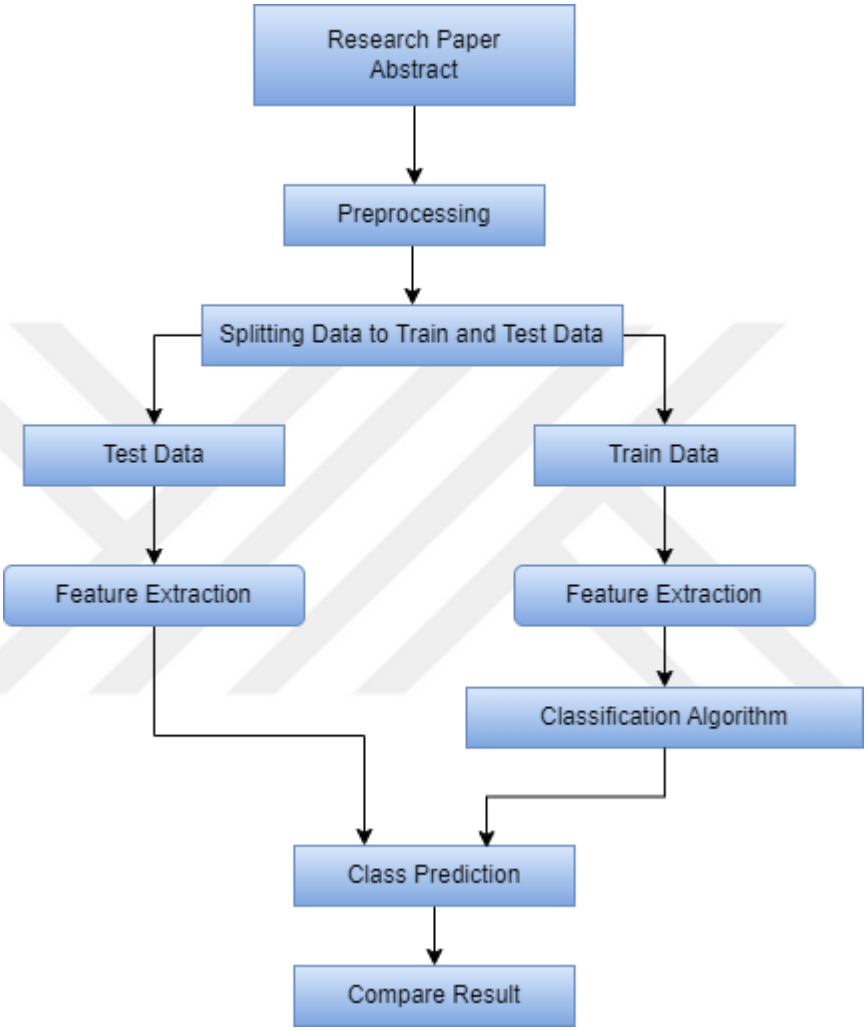
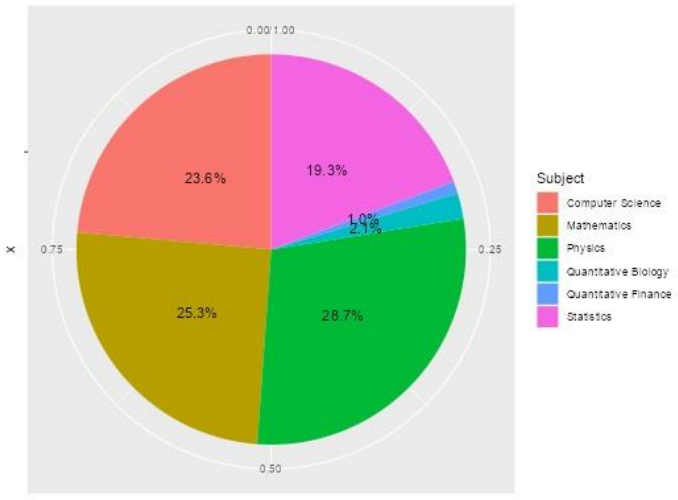


Figure 3.1 Flow Chart of Our Study

All these studies were prepared in the R Studio environment using the R language. tm, caret, dplyr, stringr etc. Keras and tensorflow Python libraries were used for R libraries, LSTM and BERT via the conda environment. In the final, an interface was created with R Shiny and the work done was shown more clearly.

3.1 DATASETS

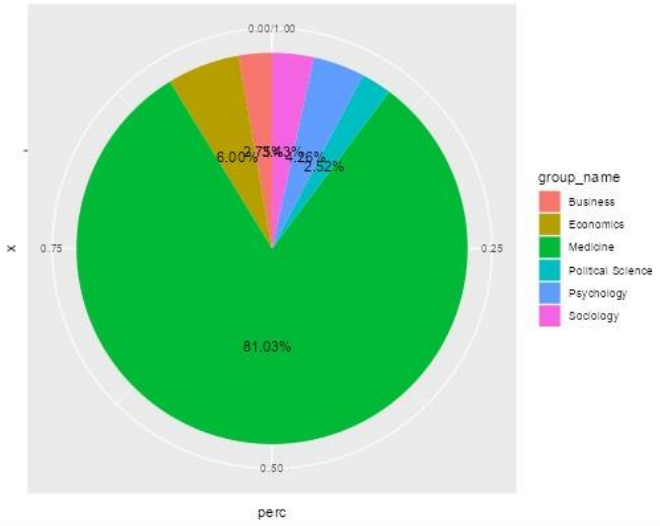
In this study, we created a new dataset using two different datasets. Our first dataset is in CSV file format and named train.csv (Janatahack: Independence Day 2020 ML Hackathon, 2020). It obtained with the multi-label technique. The dataset was created by coding the subject in binary form, including the title and abstract, into the relevant column. We added a topic column to the dataset and wrote the topics of all research papers as text in that column. The topic headings, number of topics and rates of our first dataset are given in Figure 3.2. In this dataset, it seems that nearly half of the subjects are computer science. Quantitative Finance and Quantitative Biology topics are available in very small numbers. For this reason, we thought it would negatively affect the results. Since we had similar subject headings in our other dataset, we did not include the records with this subject heading in the dataset we created.




```
[1] "Records Count.........:  20972"
Computer Science        Mathematics             Physics Quantitative Biology Quantitative Finance
            4949               5297                6013                 447                 209
                                Statistics
```

Figure 3.2 Dataset-1 Summary

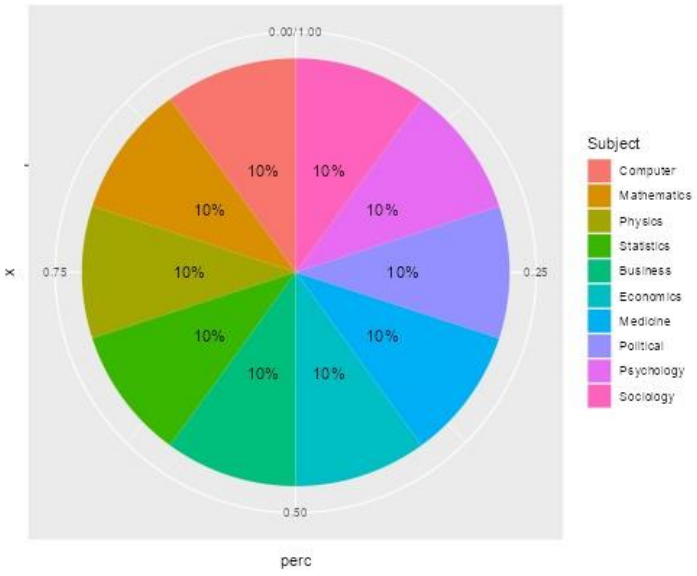
Our second dataset is in CSV file format and named data_use_article_database.csv (Brian Stacy, Lucas Kitzmüller, Xiaoyu Wang, Daniel Gerszon Mahler, and Umar Serajuddin, 2023). Dataset was created as a result of a very comprehensive study. In this dataset, doi number, country of publication, relevant countries, year of publication, etc. There are many columns. From this dataset, a dataset containing the title, abstract and group name columns necessary for our work was created. The dataset contains a very high number of abstracts of publications in the field of medicine. The subject headings, number of subjects and rates of this dataset are shown in Figure 3.3.




```
[1] "Records Count........:  1037748"
Business       Economics          Medicine Political Science       Psychology        Sociology
   28571           62241            840920             26185            44191            35640
```

Figure 3.3 Dataset-2 Summary

There are more than 1 million records in this dataset. It would be very difficult to use this amount of data in practice because the capacity constraint. For this reason we prepare a limited and balanced dataset for modeling. Therefore, a new dataset was created from the two datasets, containing 10 different subjects and 30000 records, approximately 3000 for each. There are approximately 2.7 million words in total in the abstract section of our records in the dataset. The summary of our final dataset is shown in Figure 3.4 and a section from dataset is shown in Figure 3.5.




```
[1] "Records Count..........:  30000"
[1] "Abstract Words Count...:  2697215"
   Computer Mathematics     Physics  Statistics    Business   Economics    Medicine   Political  Psychology
       3000        3000        3000        3000        3000        3000        3000        3000        3000
  Sociology
       3000
```

Figure 3.4 Final Dataset Summary

TITLE	ABSTRACT	Subject
[illegible]	[illegible]	Political
[illegible]	[illegible]	Economics
[illegible]	[illegible]	Political
[illegible]	[illegible]	Physics
[illegible]	[illegible]	Psychology
[illegible]	[illegible]	Statistics
[illegible]	[illegible]	Mathematics
[illegible]	[illegible]	Computer
[illegible]	[illegible]	Statistics
[illegible]	[illegible]	Statistics
[illegible]	[illegible]	Sociology
[illegible]	[illegible]	Psychology
[illegible]	[illegible]	Economics
[illegible]	[illegible]	Business
[illegible]	[illegible]	Medicine
[illegible]	[illegible]	Economics

Figure 3.5 An Example Section from Dataset

3.2 DATA PREPARATION AND PREPROCESSING

3.2.1 Text Preprocessing

An essential component of issues with natural language processing is text pre-processing. Because the words, phrases, and characters that are specified at this point serve as the fundamental building blocks for all other study phases, including morphological analysis and word type identification (Kannan et al., 2014). Text data typically contains numbers, dates, special characters, and commonly used words like pronouns, conjunctions, and prepositions. In text representations, these are the units that are either nonexistent or very minor. For this reason, in order to prevent issues in the subsequent phases, it is important to delete the texts from the texts at the pre-processing step when the data is produced.

The process steps for text cleaning can be as follows;

Lowercasing: Convert all text to lowercase to ensure consistency.

Removing Punctuation: Strip all punctuation marks from the text.

Tokenization: Break the text into individual words or tokens.

Removing Stop Words: Remove common stop words such as 'and', 'the', 'is', etc.

Stemming or Lemmatization: Reduce words to their root form.

Removing Non-alphabetic Characters: Remove any remaining non-alphabetic characters.

Whitespace Removal: Remove any extra whitespace that might have been introduced during the cleaning process.

Let T represent the input text, and T_{clean} represent the cleaned text.

1. **Lowercasing**:

   T_{clean} = lowercase (T)

2. **Removing Punctuation**:

   T_{clean} = remove_punctuation (T_{clean})

3. **Tokenization**:

   T_{clean} = tokenize (T_{clean})

4. **Removing Stop Words**:

   T_{clean} = remove_stopWords (T_{clean})

5. **Stemming or Lemmatization**:

   T_{clean} = stem_or_lemmatize (T_{clean})

6. **Removing Non-alphabetic Characters**:

   T_{clean} = remove_nonAlphabetic_chars (T_{clean})

7. **Whitespace Removal**:

   T_{clean} = remove_whitespace (T_{clean})

The sequential processes is used for cleaning the input text T and creating the cleaned text T_{clean}. The text is transformed or filtered differently at each phase, producing a cleaner version of the original text. An outcome after text cleaning process is shown in Figure 3.6.

```
This $\\ text  \\ was prepared  \\for {R}^n$ showing text f $ \\phi_{\\l_1(\\)} cleaning example. a\n let
$ D $ text includes (x+D) lots of []{\u0086\u0093} special characters like $ ¼ ^.Numbers, dates,\n
special characters $\\_1(\\ \\ (x+D)) $. , like pronouns, conjunctions, $ \\_1(\\ \\ (x+D)) $  and
prepositions are frequently  $ x+D $ found in text data. In text representations, $ x_0 $ these are the
{\\_1} $.\n, $ \\l_1(\\) units that\nare either nonexistent or very minor. $ \\_{\\_1} > 0 $ of $ \\ For
this reason, in order to prevent $.  issues in the subsequent phases,\n it is important\\{} \\mu_\\  to
delete the  := \_{x}\\ texts from (\\\n\\ (x+D)) \\{} is the texts at the pre-processing $ \\ step when
the data is produced(\\)\n$, $ x+D $ o $ \\\\$  $ x_0 $ o $ \\$,\n $.\n

this text prepar show text phi clean exampl let text includ lot special charact like number date special
charact like pronoun conjunct preposit frequent found text data in text represent unit either nonexist
veri minor for reason order prevent issu subsequ phase import delet text text pre process step data produc
```

Figure 3.6 Before & After Text Cleaning of an Example Text

3.2.1.1 Tokenization

We can simply define tokenization as the division of a text into pieces called tokens. these tokens usually represent a meaningful word. Depending on the structure of the text, they can also be meaningless strings of characters. Despite the fact that text fragmentation appears simple given the technologies at our disposal, there are a number of challenges that differ based on the language of the text being divided.

The basic algorithm for tokenization can be as follow:

1. Create a blank list at first to store tokens: Tokens=[]
2. Set up variables to record the token's initial index and current state: current_token=`` and start_index=0
3. For each character t_i in the input text T:
    - If t_i is a whitespace or punctuation:
        - Add current_token to the list of tokens if it isn't empty:

          Tokens.append(current_token)
        - Current_token should be reset to a blank string.
        - Put the following character's index into start_index.
    - Else:
        - Append t_i to current_token.
4. After processing every character, add current_token to the list of tokens if it is not empty:

   Tokens.append(current_token)
5. Return the list of tokens: Tokens

Let T represent the input text, Tokens represent the tokens obtained from tokenization, and n represent the number of tokens.

The tokenization can be represented as follow:

1. **Text Input**:

   $T=t_1t_2t_3\ldots t_m$

   Where ti represents the i_{th} character in the input text T, and m represents the length of the text.

2. **Tokenization**:

   $Tokens=\{token_1, token_2, token_3, \ldots, token_n\}$

   Where $token_i$ represents the i_{th} token obtained from the tokenization process.

The input text T is divided into n tokens by the tokenization process, each of which represents a significant textual unit. Depending on the particular tokenization approach being employed, the tokens may be words, sentences, or other units. The following is an example of tokenization at the below.

"We can simply define tokenization as the division of a text into pieces called tokens"

The tokens in this case are as follows:

"We" "can" "simply" "define" "tokenization" "as" "the" "division" "of" "a" "text" "into" "pieces" "called" "tokens"

3.2.1.2 Stop Words

Stop words are often used terms that are usually regarded as having minimal value in text analysis since they do not significantly add to the meaning or context of the text and exist frequently in a variety of publications. In the pre-processing stage, these terms are usually filtered out or removed from text input in order to improve the effectiveness of machine learning and NLP activities.

Common words like "the", "and", "is", "in", and "of" are examples of stop words. These words are necessary for creating grammatically proper sentences, but they are also common in practically all documents and may not convey any particular semantic meaning. Eliminating stop words can assist the analysis concentrate on more relevant phrases, decrease the complexity of the data, and increase computing efficiency.

3.2.1.3 Capitilization

Different capitalization conventions are used to create grammatically correct statements in the domain of text and document data. Handling differentiating capitalization may be difficult, particularly when classifying long texts that could include several sentences. One common method for dealing with inconsistent capitalization is to change all of the letters to lowercase. By ensuring consistency in the handling of text data, this standardization makes analysis and classification procedures more efficient. Lowercasing smoothes out differences caused by different capitalization, making text processing more uniform and efficient.

3.2.1.4 Noise Removal

Punctuation and special characters are among the many superfluous characters that are frequently found in text and document data collections. Although these punctuation marks and unique characters are necessary for humans to understand texts, classification algorithms may find them difficult to use. These kinds of characteristics have the potential to cause noise and reduce the efficacy of machine learning algorithms. Pre-processing techniques are utilized to address this issue by methodically removing or handling these unnecessary characters. This makes the data more streamlined and makes it easier for algorithms to accurately analyze and classify it. This pre-processing stage is essential to improving text-based machine learning tasks' dependability and efficiency.

3.2.1.5 Stemming and Lemmatization

Removing suffixes and prefixes from words to return them to their base or root form is known as stemming. In order to enable the system to regard several occurrences of a word as same, similar words are converted into a common base or stem. Since stemming is a heuristic process, linguistically sound stems may not always be produced.

Lemmatization is the process of taking a word's meaning and context into account and then reducing it to its base or root form, also known as a lemma. Lemmatization is distinct from stemming in that it requires a more in-depth examination of words and is dependent on a lexical knowledge base, such a dictionary or a morphological study of words.

3.2.2 Weighted Words

Terms or phrases in a text that have been given certain numerical values or weights according to their significance, importance, or relevance in a particular context are known as weighted words. In NLP and other machine learning applications, the practice of assigning weights to words is frequently utilized to ascertain the relative significance of phrases within a text or corpus. We employed the well-known bag-of-words feature extraction technique in this study.

3.2.2.1 Bag of Words

The bag-of-words (BoW) model stands out as among the methods most frequently employed for deriving weighted word characteristics in text analysis. This approach provides a straightforward and clear representation of a text document by focusing on specific sections and characteristics, particularly word frequency (Kowsari et al., 2019). In the BoW model, the frequency of each word in the corpus is assigned a numerical value, allowing for the identification of key elements within the texts.

Mathematical representation of BoW can be written as below:

- Let D be an assortment of N documents: $D = \{d_1, d_2, \ldots, d_N\}$.
- Let V be the vocabulary that contains every term that is unique in every document: $V = \{w_1, w_2, \ldots, w_M\}$, where M is the total number of unique words.
- A vector X_i of length M may be used to represent each document d_i, where X_{ij} is the frequency of word w_j in document d_i.
- The BoW representation of a document d_i can be expressed as:

  $X_i = (X_{i1}, X_{i2}, \ldots, X_{iM})$
- Where X_{ij} is the count of word w_j in document d_i.

The BoW method involves creating a list of words, and the resulting matrix of words is generated without considering grammar, word order, or the semantic connections among words. This is because, in the BoW model, words are treated independently and do not form sentences. The resulting matrix provides a concise yet informative representation of the text data.

We can write a simple algorithm for BoW as below;

1. **Create Vocabulary**:
    - Iterate through all documents in the collection D.
    - Extract unique words from each document and build a vocabulary V containing all unique words.
2. **Initialize Document Vectors**:
    - Initialize a matrix X of size NxM, where N is the number of documents and M is the size of the vocabulary.
    - Initialize all elements of X to zero.
3. **Count Word Frequencies**:
    - For each document d_i in the collection:
        - Tokenize the document into individual words.
        - For each word w_j in the vocabulary:
            - Count the frequency of w_j in d_i and store the count in X_{ij}.
4. **Return BoW Representation**:
    - The resulting matrix X represents the Bag of Words representation of the collection of documents.

An example of a BoW from the dataset we utilized that we objected to is as follows:

“recent discovery tensor invigorated interest exponent matrix multiplication determined rank symmetrized matrix multiplication better understanding multiplication present explicit rank symmetrized matrix waring decomposition sm langle rangle describe symmetry group”

Bag-of-Words (BoW) {“better”, “decomposition”, “describe”, “determined”, “discovery”, “explicit”, “exponent”, “group”, “interest”, “invigorated”, “langle”, “matrix”, “multiplication”, ”present”, “rangle”, ” rank”,” recent”, “sm”,”symmetrized”,” symmetry”,” tensor”,” understanding”,” waring”}

Bag-of-Feature (BoF) = {1,1,1,1,1,1,1,1,1,1,1,3,3,1,1,2,1,1,2,1,1,1,1}

3.2.2.2 Term Frequency-Inverse Document Frequency

TF-IDF is a statistical representation that shows how important a word is to a document in a data set. The frequency with which a word appears in the document is indicated by the term frequency (TF) component. Notably, while calculating term frequency, all terms are originally given the same weight. This frequency-based measure sheds light on the frequency of particular words in individual papers and is a useful starting point for comprehending the significance of keywords in a larger context.

TF = (Number of occurrences of a word in a document) / (Number of words in a document)

There are many different methods of calculating term frequency. Some of those:

- Term frequency itself (frequency of occurrence in the document): tf (t,d) = f,d,
- Boolean frequencies tf(t,d) = 1 if t is at d, 0 otherwise,
- Term frequency adjusted according to document length: tf(t,d) = f,d/(number of words in the document),
- Logarithmically scaled frequency: tf(t,d) = log (1 + f,d),

where:

- d stands for document
- t refers to term
- f stands for frequency

(Schütze et al., 2008).

IDF is a factor that raises the weight of words that are less common but more significant to the document and decreases the weight of high-frequency terms that appear often in all papers. For that collection of data, the term is deemed to be of low relevance as the IDF value gets closer to zero. The IDF value formula is:

IDF =Log[(Number of documents) / (Number of documents containing the word)]

TF-IDF aims to indicate the importance of words/terms in the text by multiplying these two statistics. If a word has a higher TF-IDF score, that word is more important for text representation.

3.3 CLASSIFICATION METHODS

The most important step in the text categorization process is selecting the optimal classifier. The conventional machine learning methods used to categorize the dataset are covered in this part, along with the most often used techniques: Naive Bayes (NB), Support Vector Machine (SVM), Long Short Time Memory (LSTM), Bidirectional Encoder Representations from Transformers (BERT), Lasso and Elastic-Net Regularized Generalized Linear (GLMNET), eXtreme Gradient Boosting (XGBTREE). We used Bag of Words feature extraction method in these techniques.

3.3.1 Naive Bayes (NB)

Naive Bayes is a probabilistic classifier which works based on the Bayes theorem. It determines the probability of each feature occurring in each class and returns the most likely class (Chowdhury and Schoen, 2020).

The Bayes theorem is used by the probabilistic Naive Bayes model to estimate the likelihood that a given data item will belong to a specific class. It is predicated on the idea that, given the class, the properties used to characterize the instances are independent of one another

$$P(y|x) = \frac{P(y)P(x|y)}{P(x)} \qquad (3.1)$$

Here $P(y|x)$ represents the probability of the label we want to classify according to its x features. $P(x|y)$ refers to the probability of observing features x under label y. $P(y)$ refers the probability of the label and $P(x)$ refers the probability of observing x features. Primary distinction between the theorem and naive bayes is that the former assumes that characteristics are unconnected to each other and that attributes belonging to the same class are not correlated. Naive Bayes offers a way to determine the probability function $P(x|y)$ of each class y given an item x by utilizing the specifics in sample data.

We can use these estimations to categorize data once we get them.

$$p(x|y) = \prod_{i=1}^{n} p(x_i|y)\,, \qquad (3.2)$$

where x_i is the value of the i^{th} attribute in x and n is the number of attributes.

$$p(x) = \prod_{i=1}^{k} p(C_i)\, p(x|C_i) \qquad (3.3)$$

For text categorization, naive Bayes is commonly applied in two separate rounds. The multi-variate Bernoulli model is based on naive Bayes and uses each text as a vector of binary variables that individually signal the presence or absence of a certain word. However, while determining a document's likelihood, only the words that are included in the text are considered.

3.3.2 Support Vector Machine (SVM)

A well-liked supervised machine learning method for regression and classification is support vector machine. This approach uses a hyperplane to divide multidimensional data. SVM makes use of kernel functions to find support vector classifiers in higher dimensions in a methodical manner. In reality, it does not convert the data into higher dimensions; instead, if the data are in a higher dimension, it computes their connection. The term "kernel trick" refers to the method for determining the relationships between the data. Among the information is called part trap (Chowdhury and Schoen, 2020).

The dataset elements are plotted using a Support Vector Machine (SVM) in an N-dimensional space, where N is the total number of characteristics across all corpora. By maximizing the margins between the data points (support points) in the training dataset, the optimal hyperplane to partition the data may be found. Better empirical performance is therefore an advantage of raising the margin. A further argument in favor of this is that the chance of misclassification is reduced even in the event that the border is placed erroneously.

There are two types of SVM

Linear SVM: When classes can be separated linearly, a linear hyperplane is found to divide them. This is known as a linear support vector machine (SVM).

Non-linear SVM: This type of SVM handles non-linear decision boundaries by converting data into higher-dimensional spaces through the use of kernel functions.

Figure 3.7 shows the separations of Linear and Non-Linear SVM.

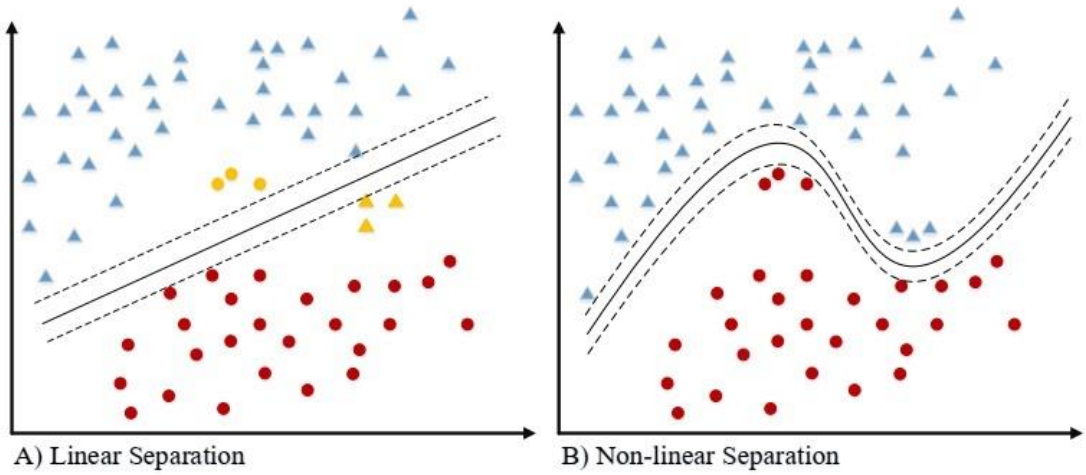


Figure 3.7 Separations of SVM for a 2D Dataset (Kowsari et al., 2019)

The decision function of a linear SVM classifier may be expressed mathematically as follows:

$$f(x) = w^T x + b \quad (3.4)$$

where:

- x stands for a document's feature vector.
- The weight vector acquired during training is denoted by the letter w
- b stands for the bias word.
- The decision function output, denoted by $f(x)$, shows the expected class label for x.

In real world problem it is not likely to get an exactly separate line dividing the data within the space. And we might have a curved decision boundary. We might have a hyperplane which might exactly separate the data but this may not be desirable if the data has noise in it. It is better for the smooth boundary to ignore few data points than be curved or go in loops, around the outliers (Jakkula, 2006).

In conclusion, Support Vector Machine is an excellent supervised learning technique that is extensively used. It works well for classification tasks by identifying the best hyperplane to maximize the margin between classes, which improves generalization to new data.

3.3.3 Long Short Term Memory (LSTM)

The Long Short-Term Memory architecture of artificial recurrent neural networks (RNNs) is used in deep learning. With feedback connections, LSTM differs from conventional feedforward neural networks. It can process streams of data, including audio and video, as well as immediate data, like pictures. Speech recognition, unsegmented and linked handwriting recognition, anomaly detection in network traffic, and intrusion detection systems (IDSs) are a few applications where LSTM may be used.

A typical LSTM unit consists of an input gate, an output gate, a forget gate, and a cell. The cell retains values for a variety of durations, and these three gates regulate the information that enters and exits the cell. Figure 3.8 shows the architecture of a LSTM unit.

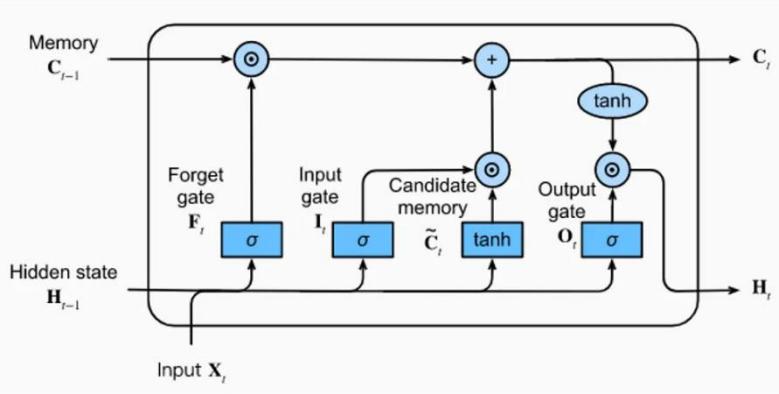


Figure 3.8 Architecture of a LSTM Unit (Medium, 2024)

We can represent an LSTM classifier mathematically:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (3.5)$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (3.6)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (3.7)$$

$$g_t = tanh(W_g x_t + U_g h_{t-1} + b_g) \quad (3.8)$$

$$c_t = f_t * c_{t-1} + i_t * g_t \quad (3.9)$$

At the end, the output of the LSTM can be determined using equation (3.10):

$$h_t = o_t * \tanh(c_t) \quad (3.10)$$

- σ represents the sigmoid activation function.
- W_i, W_f, W_o, W_g are the weight matrices for input gates, forget gates, output gates, and cell update gates, respectively.
- U_i, U_f, U_o, U_g are the weight matrices for input gates, forget gates, output gates, and cell update gates, respectively (for the recurrent connections).
- b_i, b_f, b_o, b_g are the bias vectors.
- i_t, f_t, o_t, g_t represent the input gate, forget gate, output gate, and cell gate activations at time step t.
- c_t is the cell state at time step t.
- h_t is the hidden state (output) at time step t.

(Steven and Stefan, 2020)

This is a mathematical depiction of a single LSTM cell's forward pass. In actuality, an LSTM layer is formed by stacking multiple LSTM cells, and an LSTM network is formed by stacking multiple LSTM layers.

Since there may be unexpected gaps between significant occurrences in a time series, LSTM networks are ideal for categorization, processing, and prediction based on time series data. Traditional RNN training can lead to disappearing and exploding gradient issues. To address these issues, LSTMs were created.

3.3.4 Bidirectional Encoder Representations from Transformers (BERT)

BERT is a transfer learning model that was created with the goal of offering more insightful and in-depth learning than earlier models in language comprehension problems. In contrast to other conventional language models, BERT is trained to comprehend sentence context by working both forward and backward.

The attention mechanism is part of the Transformer architecture, which is used by BERT. A key component of many effective models for language comprehension problems is this design. Pre-training and fine-tuning procedures for BERT are shown in Figure 3.9.

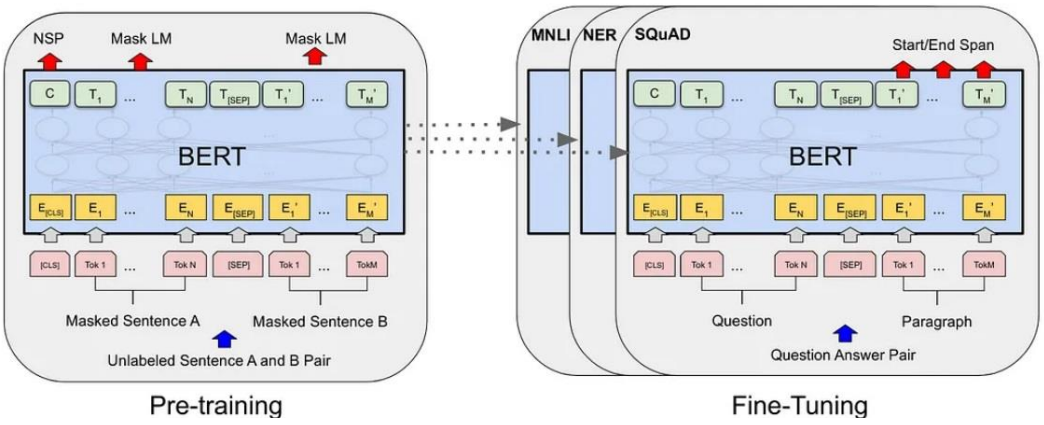


Figure 3.9 Pre-training and fine-tuning procedures for BERT (Devlin at al., 2018)

Unlike recent language representation models, BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be finetuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications (Devlin at al., 2018).

3.3.5 Generalized Linear Models (GLMNET)

A package in the R programming language called GLMNET (Generalized Linear Models with L1 and L2 Regularization) is available. There is an algorithm in this library that operates on generalized linear models (GLM). Regression analysis is the exclusive usage of the GLMNET package, which contains the L1 (Lasso) and L2 (Ridge) regularization techniques.

A linear form model equation is shown at the below:

$$g(\mu) = \alpha + \beta_1 x_1 + .. + \beta_n x_n \,, \qquad (3.11)$$

where a linear predictor is the linear combination of the explanatory variables. The coefficients of regression are denoted by α and β_j, where j = 1,... n in the case of n independent variables.

GLMNET is used specifically for regression analysis. It can go beyond linear regression models and create more complex and streamlined models. It helps control model complexity and prevent overfitting through editing techniques. GLMNET offers the ability to evaluate the performance of models using cross-validation.

3.3.6 Decision Trees (XGBTREE)

The decision tree is an older technique of text and data mining categorization. Decision tree classifiers, or DTCs, have shown effective in a variety of classification applications. This method's structure is a data space decomposition done hierarchically.

The XGBoost (Extreme Gradient Boosting) package contains the XGBTree algorithm, which is based on decision trees. A tree-based model inside the machine learning toolkit XGBoost is referred to as xgbTree. XGBoost is primarily suited for regression and classification tasks. A gradient boosting approach is used by XGBTree. This approach combines weak learners (typically decision trees) to produce a powerful model.

Mathematically, we can write the model in the form

$$\hat{y}_i = \sum_{k=1}^{K} f_k\,(x_i), f_k \ \in F \qquad (3.12)$$

where F is the set of all potential classification and regression trees (CARTs), K is the number of trees, and f_k is a function in the functional space F. The objective function to be optimized is given by at the below:

$$obj(\theta) = \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} w(f_k), \qquad (3.13)$$

where $w(f_k)$ is the complexity of the tree f_k.

XGBoost is quick and efficient, especially when handling big data sets. It uses parallel processing and a variety of optimization approaches to increase speed. With the use of XGBoost, one may assess the model's performance using auxiliary methods like cross-validation.

3.4 CLASSIFICATION & RESULTS

The classification process was done with 6 different models using the R language. For classification, we need to train the data and then validate the trained dataset. After verification, some metrics are calculated from confusion matrix and these metrics show how successful our trained models are.

3.4.1 Confusion Matrix

A confusion matrix is a tool for evaluating machine learning performance by displaying the accuracy of a classification model. The number of false positives, false negatives, true positives, and true negatives are displayed. This atrix aids in misclassification detection, prediction accuracy enhancement, and model performance analysis.

A confusion matrix, which is a N x N matrix with N being the total number of target classes, is used to evaluate how well a classification model performs. The matrix compares the anticipated values of the machine learning model to the actual target values. This gives us a complete picture of the types of errors and performance indicators in our classification model.

The basic confusion matrix template in Table 3.1 is for a simple binary classification task. The confusion matrix may also be used to show the outcomes of multiclass classification issues. The diagonal boxes in the confusion matrix, from left to right, show real expected positives. The other boxes show the numbers for false positives, false negatives, and genuine negatives, depending on which class is selected. For metric definitions, we used the values specified in the basic confusion matrix in Table 3.1.

Table 3.1 Basic Confusion Matrix

		PRETICTED CLASS	
		POSITIVE	NEGATIVE
ACTUAL CLASS	POSITIVE	True Positive (TP) = 10	False Negative (FN) = 2
	NEGATIVE	False Positive (FP) = 1	True Negative (TN) = 7

Here are the metrics we use:

Accuracy: It refers to the proportion of correct predictions in a model's total predictions.

Accuracy = (TP+TN) / (P+N), where P and N are total positive and negative classes.

In the given example, Accuracy is (10+7) / (12+8) = 0.85

Recall: It refers to the ratio of true positives to total positives. High sensitivity indicates how well the model recognizes positive examples.

Recall = TP / (TP + FN)

In the given example, Recall Value is: 10 / (10 + 2) = 0.833

Precision: Precision shows how many of the samples the model predicts as positive are actually positive.

Precision = TP / (TP + FP)

In the given example, Precision Value is: 10 / (10 + 1) = 0.909

F1 Score: F1 score is a metric that evaluates the accuracy of a model and is the harmonic average of precision and recall values.

F1 Score = 2 x ((Precision*Recall) / (Precision + Recall))

In the given example, F1 Score is: 2 x ((0.909 * 0.833) / (0.909+0.833)) = 0.869

Due to the size of our dataset, it generally took a long time to train and test all our models. In this section, we showed the parameters with which we ran the models and the confusion matrices and metrics of the results we obtained.

3.4.2 Results of SVM Model

We applied the SVM model with the parameters kernel="linear", cost=1. The complexity matrix is shown in Figure 3.10

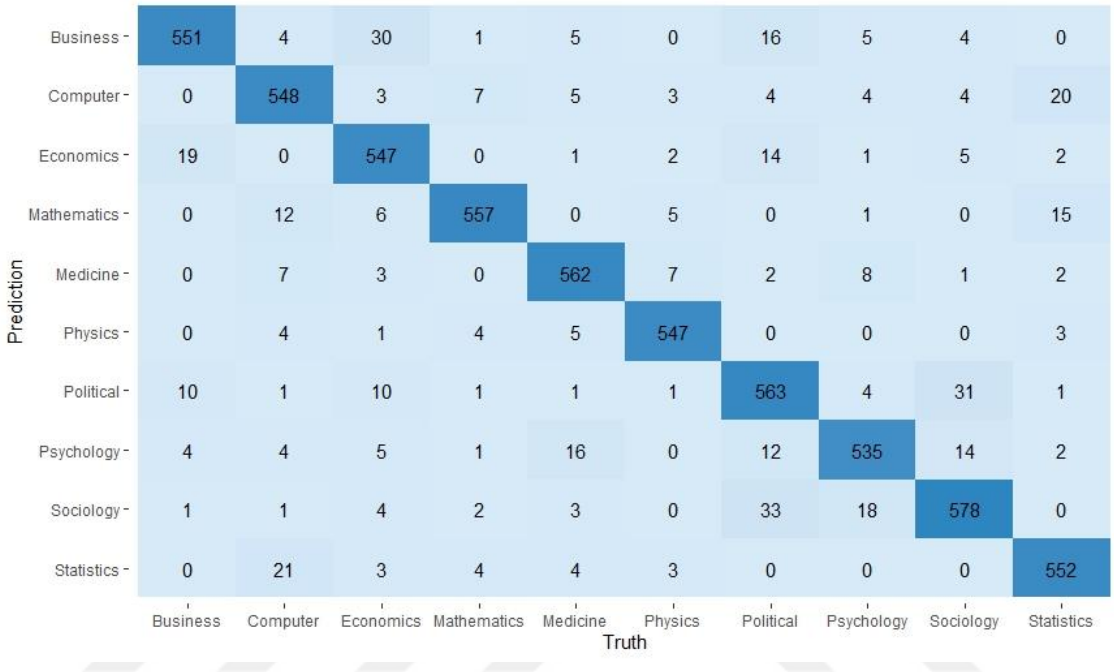


Figure 3.10 Confusion Matrix of SVM Model

The SVM model was run with 6000 test data points. As seen in the confusion matrix in Figure 3.10, the SVM model made very accurate predictions and predicted a total of 5540 of the paper topics correctly and 460 incorrectly. We saw that it made erroneous predictions mostly on political-sociology, computer-statistics and business-economics. The metrics for SVM classification are shown at the Table 3.2.

Table 3.2 Metrics of SVM Model

Accuracy	Recall	Precision	F1
0.9233	0.9242	0.9239	0.9239

3.4.3 Results of BERT Model

We applied the pretrained BERT model with the parameters below. Figure 3.11 shows confusion matrix of BERT model.

```
layer_dense(units = 100, activation = "relu", input_shape = ncol(features_train)) %>%
layer_dense(units = 40, activation = "relu", kernel_initializer = "he_normal", bias_initializer = "zeros",
kernel_regularizer = regularizer_l2(0.05)) %>% layer_dropout(rate = 0.2) %>%
layer_dense(units = 1, activation = "sigmoid")
model_BERT %>% compile(
  optimizer = "Adam",
  loss = "binary_crossentropy",
  metrics = c("acc", "AUC"))
```

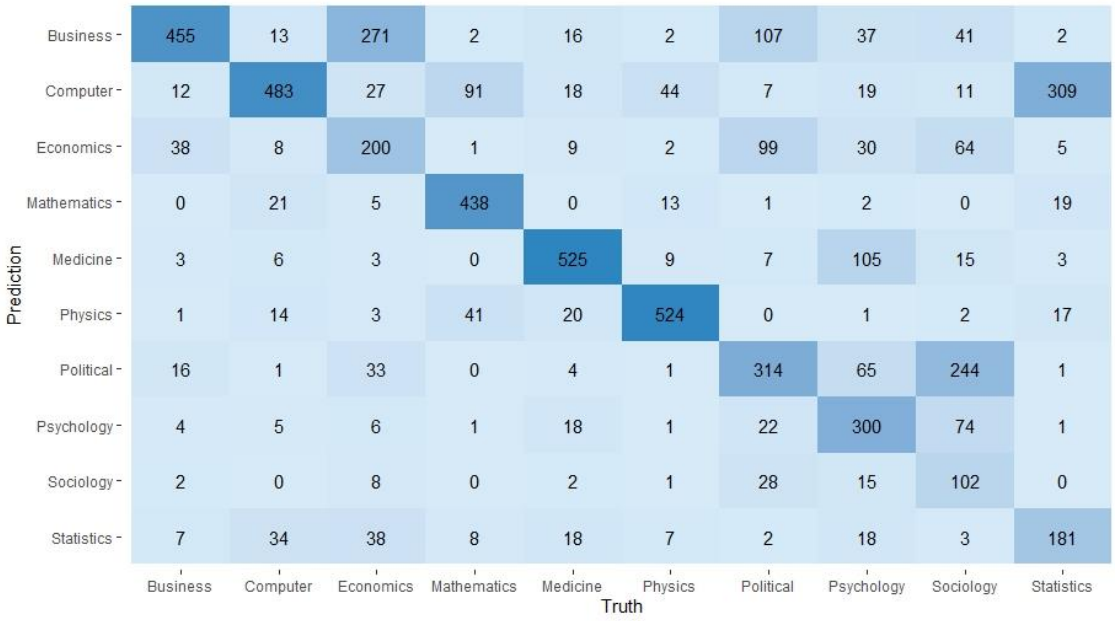


Figure 3.11 Confusion Matrix of BERT Model

The BERT model was run with 6000 test data points. We used multi label text classification in BERT model. Our model predicted a topic for 5806 abstracts. As seen in the confusion matrix in Figure 3.11, the BERT model predicted a total of 3522 of the paper topics correctly and 2284 incorrectly. We saw that it made a high amount of erroneous predictions mostly on computer-statistics, computer-mathematics, political-sociology, business-political, medicine-psychology and business-economics. The metrics for BERT classification are shown at the Table 3.3.

Table 3.3 Metrics of BERT Model

Accuracy	Recall	Precision	F1
0.6066	0.6023	0.6263	0.5855

3.4.4 Results of GLMNET Model

We applied the GLMNET model with the parameter maxitglm = 10^6. Figure 3.12 shows confusion matrix of GLMNET model.

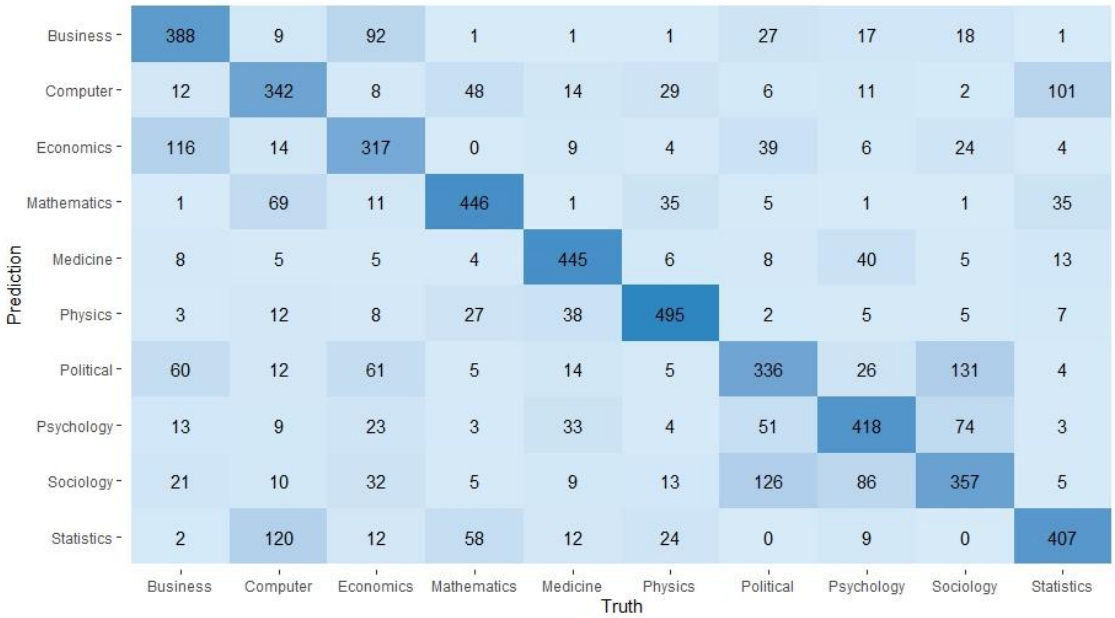


Figure 3.12 Confusion Matrix of GLMNET Model

The GLMNET model was run with 6000 test data points. As seen in the confusion matrix in Figure 3.12, the GLMNET model predicted a total of 3951 of the paper topics correctly and 2049 incorrectly. We saw that it made erroneous predictions mostly on computer-statistics, computer-mathematics, political-sociology, sociology-psychology and business-economics. The metrics for GLMNET classification are shown at the Table 3.4.

Table 3.4 Metrics of GLMNET Model

Accuracy	Recall	Precision	F1
0.6585	0.6585	0.6621	0.6595

3.4.5 Results of Naive Bayes Model

We applied the Naive Bayes model. Figure 3.13 shows confusion matrix of NB model.

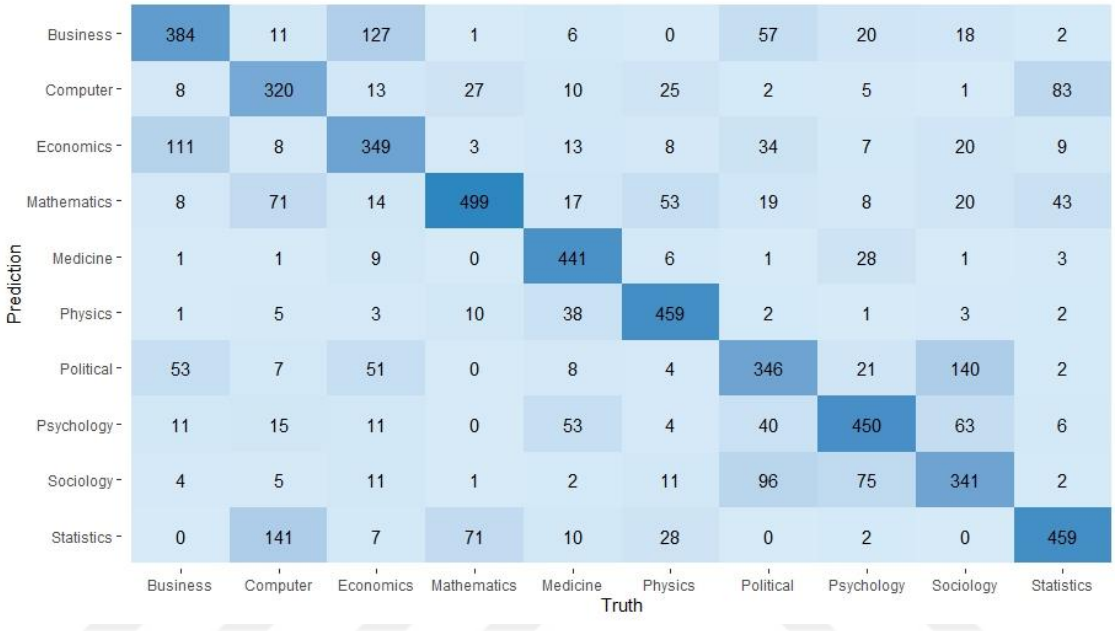


Figure 3.13 Confusion Matrix of Naive Bayes Model

The Naive Bayes model was run with 6000 test data points. As seen in the confusion matrix in Figure 3.13, the Naive Bayes model predicted a total of 4048 of the paper topics correctly and 1952 incorrectly. We saw that it made erroneous predictions mostly on computer-statistics, mathematics-statistics, political-sociology, political-business, sociology-psychology and business-economics. The metrics for Naive Bayes classification are shown at the Table 3.5.

Table 3.5 Metrics of Naive Bayes Model

Accuracy	Recall	Precision	F1
0.6746	0.6737	0.6818	0.6745

3.4.6 Results of XGBTREE Model

We applied the XGBTREE model with the parameters below: method="cv", number=3, search="random", verboseIter=T. Figure 3.14 shows confusion matrix of XGBTREE model.

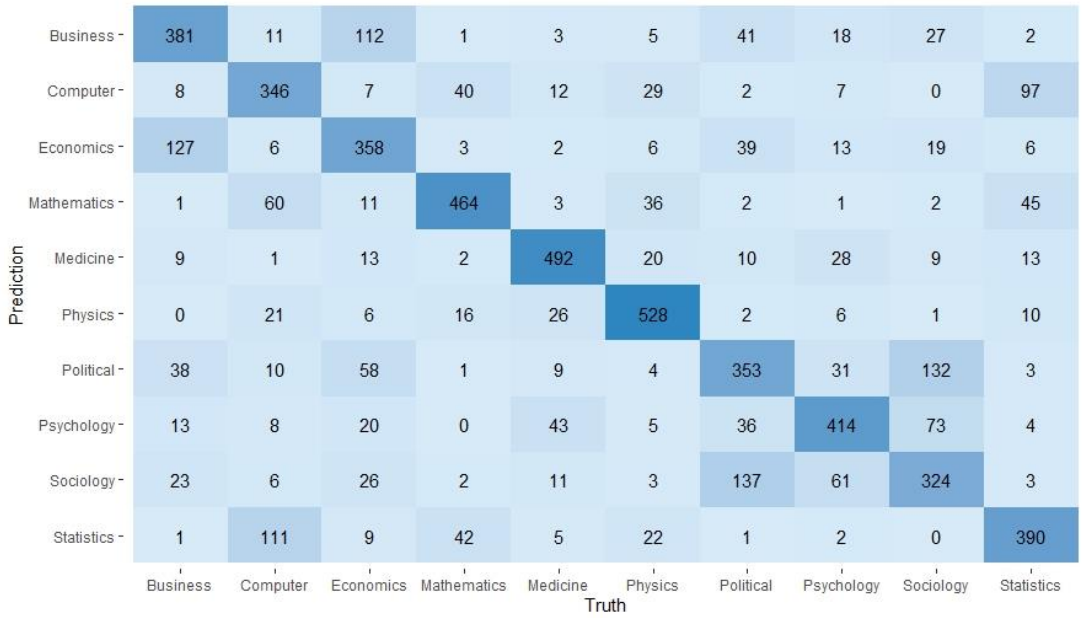


Figure 3.14 Confusion Matrix of XGBTREE

The XGBTREE model was run with 6000 test data points. As seen in the confusion matrix in Figure 3.14, the XGBTREE model predicted a total of 4050 of the paper topics correctly and 1950 incorrectly. We saw that it made erroneous predictions mostly on computer-statistics, mathematics-statistics, political-sociology, political-business, sociology-psychology and business-economics. The metrics for XGBTREE classification are shown at the Table 3.6.

Table 3.6 Metrics of XGBTREE Model

Accuracy	Recall	Precision	F1
0.6750	0.6746	0.6744	0.6741

3.4.7 Results of LSTM Model

We set the LSTM model with the parameters below:

layer_embedding(name = "input", input_dim = num_words, input_length = maxlen, output_dim = 8) %>% layer_lstm(name = "LSTM", units = 8, kernel_regularizer = regularizer_l1_l2(l1 = 0.05, l2 = 0.05), return_sequences = F) %>% layer_dense(name = "Output", units = 1, activation = "sigmoid")

We compiled the model with the parameters below:

compile(optimizer = optimizer_adam(learning_rate= 0.001), metrics = "accuracy", loss = "binary_crossentropy")

We trained the model with the parameters below:

batch_size = 64, epochs = 10, validation_split = 0.1, verbose = 1,

Figure 3.15 shows confusion matrix of LSTM model.

Figure 3.15 Confusion Matrix of LSTM Model

The LSTM model was run with 6000 test data points. We used multi label text classification in LSTM model. Our model predicted a topic for 3791 abstracts. As seen in the confusion matrix in Figure 3.15, the LSTM model predicted a total of 2129 of the paper topics correctly and 1662 incorrectly. We saw that it made a high amount of erroneous predictions mostly on political-economics, computer-statistics, computer-mathematics, political-sociology, business-political, medicine-psychology and business-economics. The metrics for LSTM classification are shown at the Table 3.7.

Table 3.7 Metrics of LSTM Model

Accuracy	Recall	Precision	F1
0.5615	0.5484	0.5557	0.5492

3.4.8 Overall of Models

Since we had a large number of models, we only used the bag-of-words method for feature extraction. We showed the results we obtained collectively in Table 3.8. The running times of our models' algorithms became longer as the amount of data increased. There was not much change in accuracy values. As can be seen from the table, the highest accuracy rate was obtained in the SVM classifier. For this reason, the SVM method was used in the prediction and recommendation section of the R shiny application.

Table 3.8 Experimental Results of Models

Model Type	Feature Extraction	Accuracy	Recall	Precision	F1
SVM	BoW	0.9233	0.9242	0.9239	0.9239
XGBTREE	BoW	0.6750	0.6746	0.6744	0.6741
NB	BoW	0.6746	0.6737	0.6818	0.6745
GLMNET	BoW	0.6585	0.6585	0.6621	0.6595
BERT	BoW	0.6066	0.6023	0.6263	0.5855
LSTM	BoW	0.5615	0.5484	0.5557	0.5492

Figure 3.16 shows accuracy rates graphic of the models.

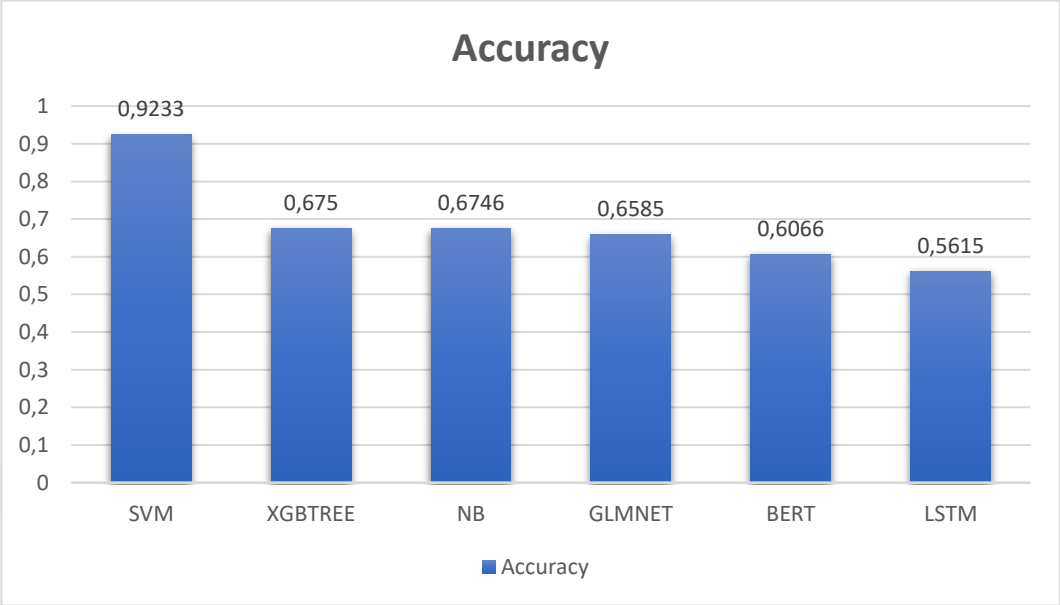


Figure 3.16 Accuracy Rates Graphic of the Models

Figure 3.17 shows recall rates graphic of the models.

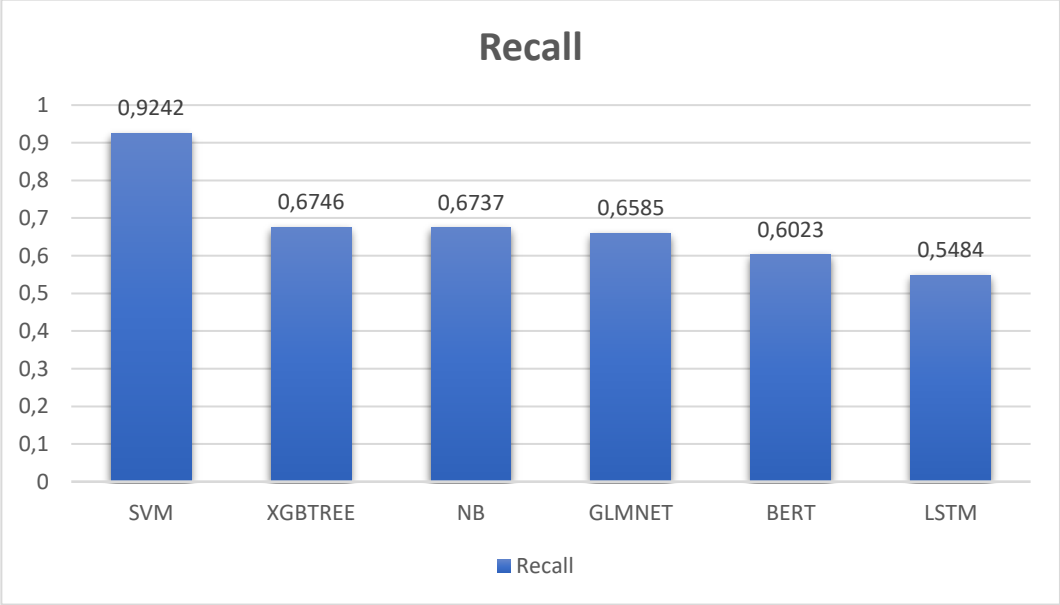


Figure 3.17 Recall Rates Graphic of the Models

Figure 3.16 shows accuracy rates graphic of the models.

Figure 3.18 shows precision rates graphic of the models.

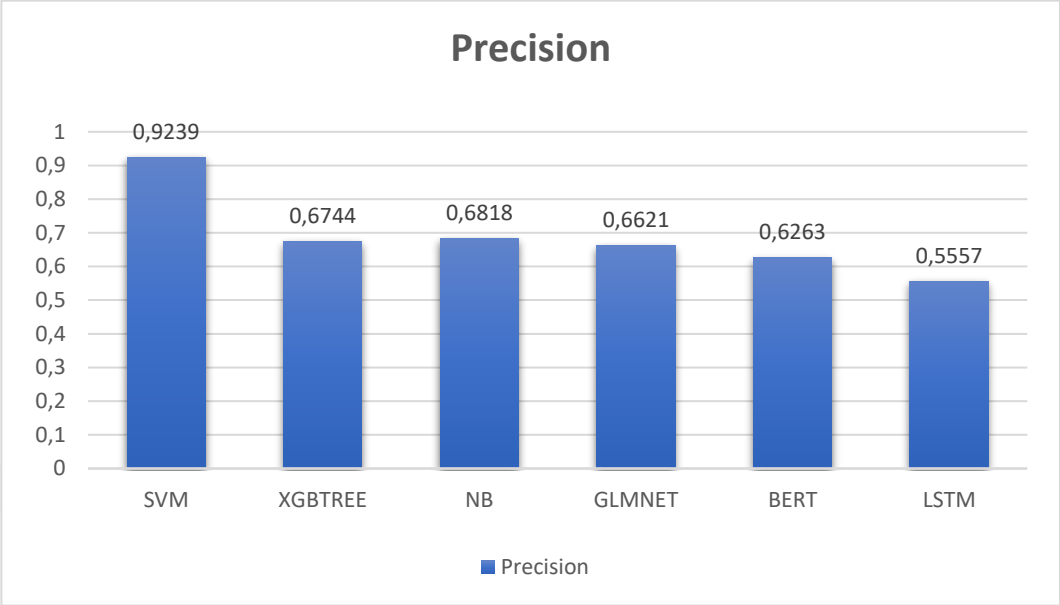


Figure 3.18 Precision Rates Graphic of the Models

Figure 3.19 shows F1 rates graphic of the models.

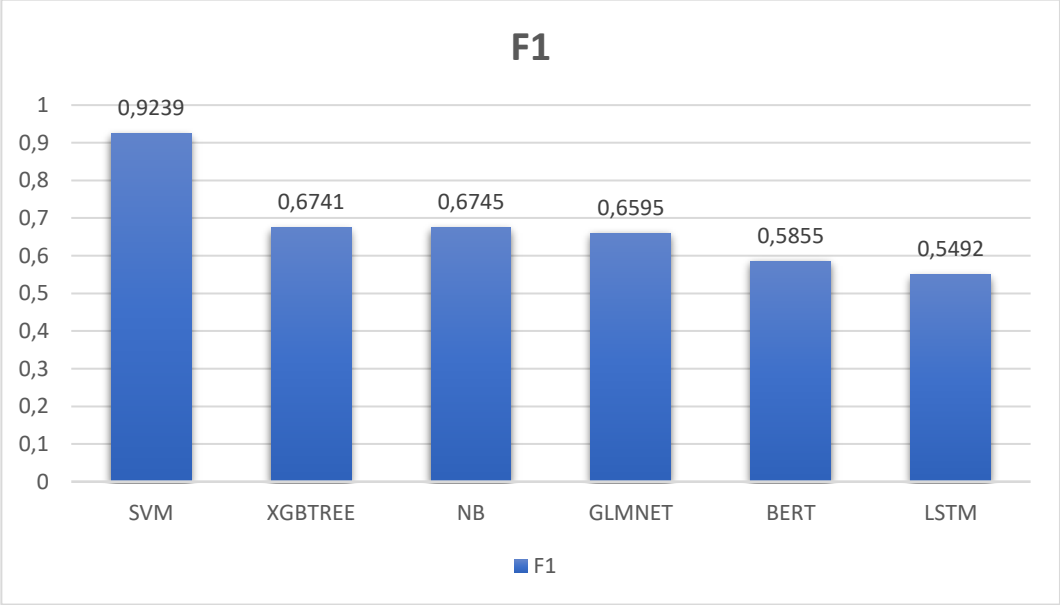


Figure 3.19 F1 Rates Graphic of the Models

4 IMPLEMENTATION

We developed web application to effectively display the results of our work. Since we run our models with the R language, we developed this application using the visual library Shiny. By using a tabbed structure, we showed a different processing result in each tab. The first screen, where we can also see the information of the datasets used, is shown in Figure 4.1.

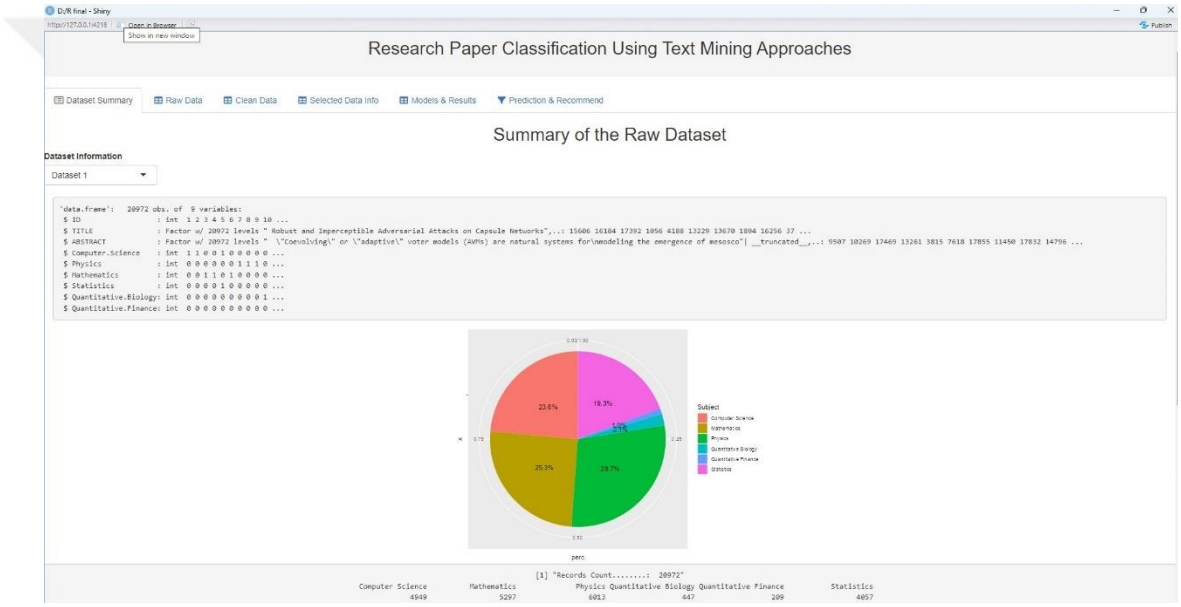


Figure 4.1 Dataset Summary Section of R Shiny Application

Using the second tab, we showed the raw, unprocessed data in the datasets, is shown in Figure 4.2.

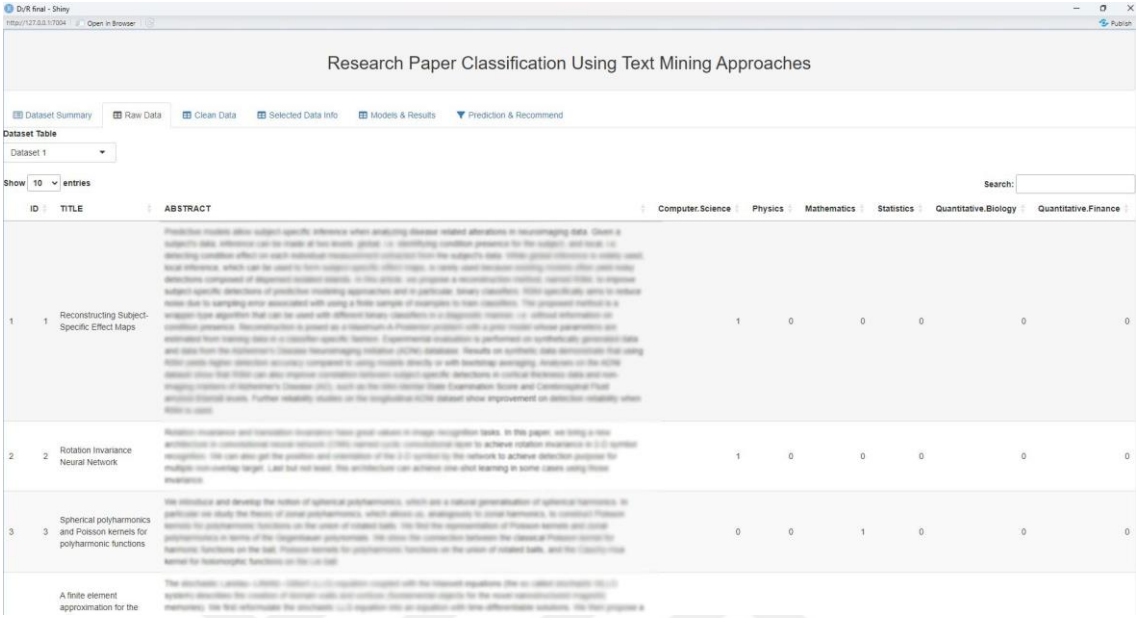


Figure 4.2 Raw Data View Section of R Shiny Application

In the Clean Data tab, the merged, cleaned and subject tagged version of the two datasets is shown, can be seen in Figure 4.3.

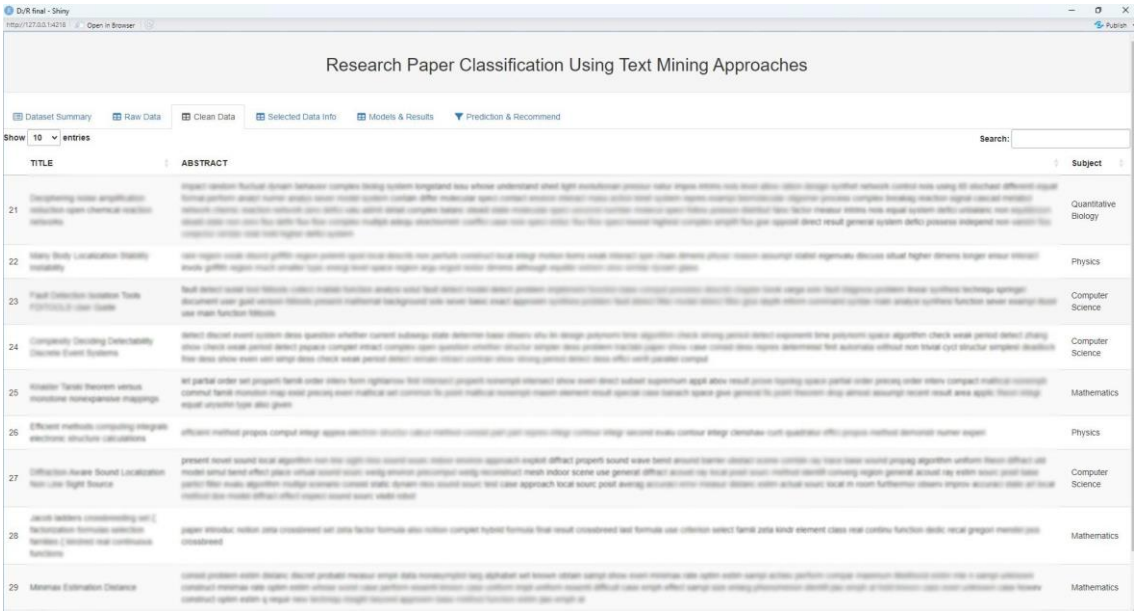


Figure 4.3 Clean Data View Section of R Shiny Application

In the fourth tab, information and data about our main dataset, which we created by selecting a balanced dataset from the cleaned dataset are shown. The view of section can be seen in Figure 4.4.

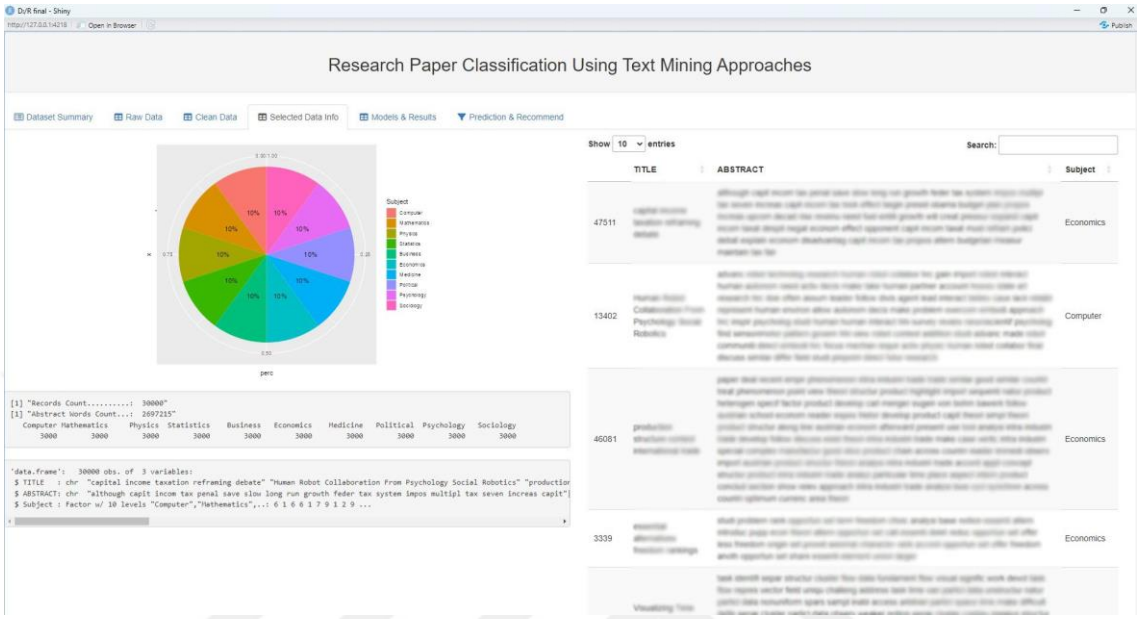


Figure 4.4 Selected Data Information Section of R Shiny Application

In the Models & Results tab, the confusion matrices and actual&prediction data tables obtained as a result of running each model are shown depending on the user's selection. Figure 4.5 shows the Models&Results section.

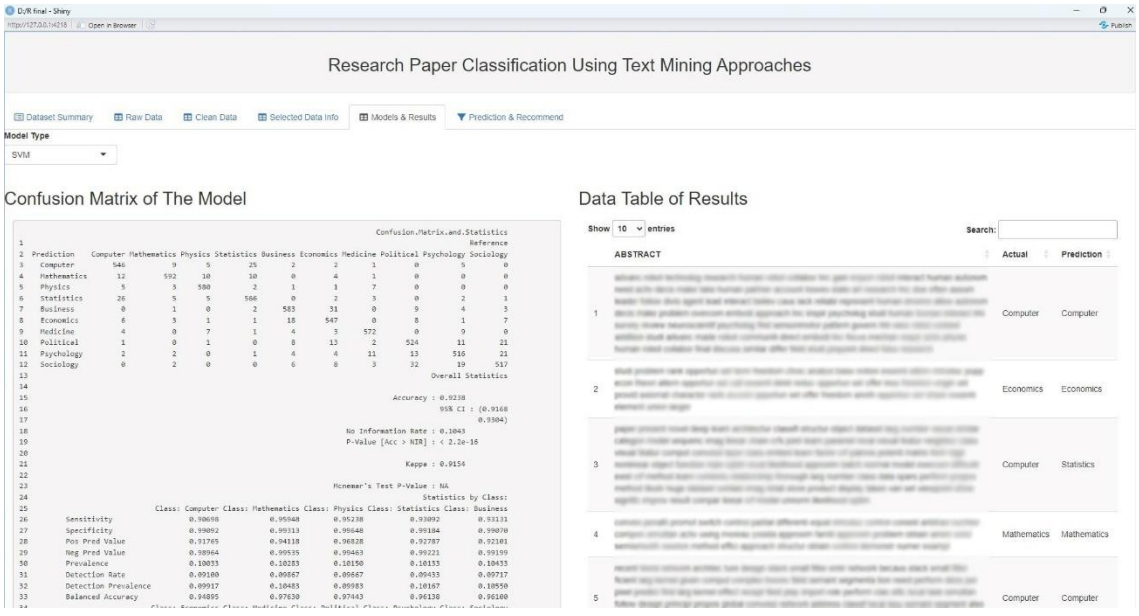


Figure 4.5 Models&Results Section of R Shiny Application

In our last tab, there is a recommendation system that predicts the topic of the abstract that the user enters in the text box with the SVM model and lists 50 of the closest topic abstract in our dataset. An example search can be seen in the Figure 4.6.

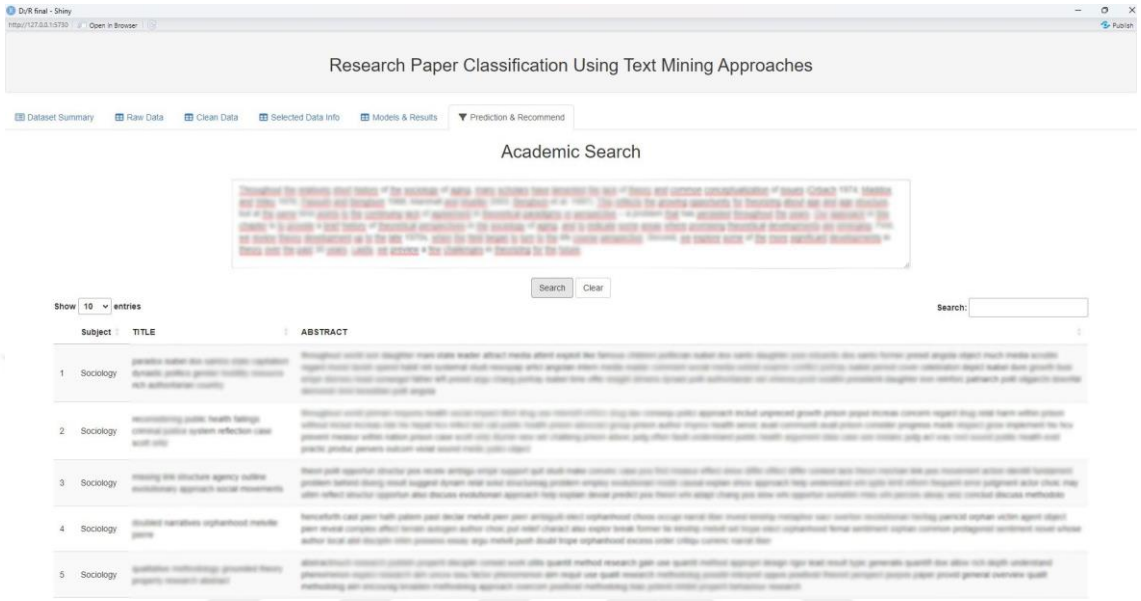


Figure 4.6 Prediction & Recommendation Section of R shiny Application

The advantages and disadvantages of the models use can be summarized as follows:

Naive Bayes, it is a quick classifier that can be used for both binary and multi-class issues. It also takes less training than discriminative models such as logistic regression and converges earlier. However, it is impossible to achieve interactions between the features. Relative probabilities rather than mathematically exact probabilities were used in the calculation.

XgbTree model is easily understood after some clarification. It has insights that are dynamic and founded in expert knowledge. Unsuitable for multilevel categorical variables, complicated for numerous valued characteristics that are unknown, and skewed information gain.

LSTM is particularly good at identifying long-range relationships in sequential data. It can handle variable-length input sequences and are capable of learning complicated patterns in data. But LSTM has limits. It may need a lot of resources for training because it is computationally demanding, especially when dealing with big datasets. LSTM training can take a lot of time, and when there isn't enough training data, overfitting may occur.

The BERT model is very versatile and effective for a range of natural language processing (NLP) applications, including named entity recognition, text classification, and question answering. It may be fine-tuned on specific tasks with very few labeled instances. BERT model do have certain drawbacks, though. It is less accessible to smaller research groups or organizations with low computational resources since it demands significant computational resources for both pre-training and fine-tuning.

GLMNET can easily handle huge datasets and has good processing efficiency. Furthermore, GLMNET has the ability to provide probabilistic predictions, which makes it possible to estimate uncertainty in classification tasks. It does, however, have certain restrictions. It makes the assumption that there is a linear relationship between the log-odds of the target variable and the input characteristics, which may not always hold true in intricate real-world datasets.

Our best model SVM works effectively in high-dimensional areas and is especially useful for issues involving intricate decision boundaries. SVM is capable of handling both linear and non-linear classification tasks by leveraging a variety of kernel functions. Nevertheless, SVM has its drawbacks. In particular, when working with huge datasets or non-linear kernel functions, it can be computationally demanding. Careful selection of hyperparameters, such the regularization parameter, may also be necessary for SVM.

5 CONCLUSION

There are several academic studies on text classification. These techniques help us in many applications in the digital environment such as search engines, recommendation systems, sentiment analysis and topic tagging.

We investigated related works that focused on text classification. We searched for appropriate datasets, and obtained a new one by combining two different datasets. After preprocessing step, we applied different classification methods such as Naive Bayes (NB), Support Vector Machine (SVM), Long Short Time Memory (LSTM), Bidirectional Encoder Representations from Transformers (BERT), Lasso and Elastic-Net Regularized Generalized Linear (GLMNET), eXtreme Gradient Boosting (XGBTREE). The accuracy rate may have decreased slightly if there were too many labels and a large amount of data. The accuracy rate may increase slightly with data reduction. We saw that the best accuracy rate of the models we investigated was in SVM. Using this model, we designed a system that classifies the given abstract according to its subject. In addition, it suggests to the user some other research papers' abstract in the same subject with similar topic. All of these works were implemented in R and the application was developed in R shiny.

Conventional methods mostly concentrate on text-based characteristics for categorization, ignoring important data found in other modalities like tables, figures, and citations. At the future works, rich semantic links found in scholarly articles can be used to improve categorization. It can be improved by using knowledge graph embeddings and multimodal learning. In order to provide a thorough representation of academic publications, the integration of text, figures, tables, and citation networks as input modalities can be used. Knowledge graph embeddings can also be used to record the

semantic links among subjects, venues, authors, and articles. The research findings will have consequences for improving scholarly information retrieval systems and will further academic article categorization techniques. We developed a system which will help researchers. We believe that it will be better with these kind of additions.

REFERENCES

Janatahack: Independence Day 2020 ML Hackathon (2020). Retrieved April 29, 2023, from https://datahack.analyticsvidhya.com/contest/janatahack-independence-day-2020-ml-hackathon/True/#ProblemStatement

Brian Stacy, Lucas Kitzmüller, Xiaoyu Wang, Daniel Gerszon Mahler, and Umar Serajuddin. 2023. Dataset from "Missing Evidence: Tracking Academic Data Use around theWorld". Retrieved December 5, 2023, from https://datacatalog.worldbank.org/search/dataset/0065200/Data-Use-in-Academia-Dataset

Architecture of a LSTM Unit. Retrieved January 26, 2024 from https://medium.com/@ottaviocalzone/an-intuitive-explanation-of-lstm-a035eb6ab42c

Baharudin, B., Lee, L. H., Khan, Aurangzeb & Khan, K. (2010). A review of machine learning algorithms for text-documents classification. *Journal of Advances in Information Technology, 1(1),* 4-20. https://doi.org/10.4304/jait.1.1.4-20

Chiu, B., & Baker, S. (2020). Word embeddings for biomedical natural language processing: A survey. *Language and Linguistics Compass*, *14*(12). https://doi.org/10.1111/lnc3.12402

Chowdhury, S., & Schoen, M. P. (2020). Research paper classification using supervised machine learning techniques. *2020 Intermountain Engineering, Technology and Computing (IETC)*, 7-8.

Conneau, A., Schwenk, H., Barrault, L., & Lecun, Y. (2016). Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dhar, A., Mukherjee, H., Dash, N. S., & Roy, K. (2021). Text categorization: past and present. *Artificial Intelligence Review*, *54*(4), 3007-3054.

Graves, A., Jaitly, N., & Mohamed, A. R. (2013, December). Hybrid speech recognition with deep bidirectional LSTM. In *2013 IEEE workshop on automatic speech recognition and understanding* (pp. 273-278). IEEE.

Guida, G., & Mauri, G. (1986). Evaluation of natural language processing systems: Issues and approaches. *Proceedings of the IEEE*, *74*(7), 1026-1035.

Jin, P., Zhang, Y., Chen, X., & Xia, Y. (2016, July). Bag-of-embeddings for text classification. In *IJCAI* (Vol. 16, pp. 2824-2830).

Jakkula, V. (2006). Tutorial on support vector machine (svm). *School of EECS, Washington State University*, *37*(2.5), 3.

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Kannan, S., Gurusamy, V., Vijayarani, S., Ilamathi, J., Nithya, M., Kannan, S., & Gurusamy, V. (2014). Preprocessing techniques for text mining. *International Journal of Computer Science & Communication Networks*, *5*(1), 7-16.

Kim, Y. (2014) Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, 1746-1751. arXiv: 1408.5882 https://doi.org/10.3115/v1/D14-1181

Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., & Brown, D. (2019). Text classification algorithms: A survey. *Information*, *10*(4), 150. https://doi.org/10.3390/info10040150

Leopold, E., & Kindermann, J. (2002). Text categorization with support vector machines. How to represent texts in input space?. *Machine Learning*, *46*(1), 423-444.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, *26*.

Schütze, H., Manning, C. D., & Raghavan, P. (2008). *Introduction to information retrieval* (Vol. 39, pp. 234-265). Cambridge: Cambridge University Press.

Steven, E., & Stefan, G. (2020). Time series forecasting using lstm networks: a symbolic approach. *arXiv preprint arXiv:2003.05672*.

Sun, Y., Wong, A. K., & Kamel, M. S. (2009). Classification of imbalanced data: A review. *International journal of pattern recognition and artificial intelligence*, *23*(04), 687-719.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, *27*.

Thangaraj, M., & Sivakami, M. (2018). Text classification techniques: A literature review. *Interdisciplinary journal of information, knowledge, and management*, *13*, 117. https://doi.org/10.28945/4066

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Information retrieval*, *1*(1), 69-90.

Zhang, M. L., & Zhou, Z. H. (2006). Multilabel neural networks with applications to functional genomics and text categorization. *IEEE transactions on Knowledge and Data Engineering*, *18*(10), 1338-1351.

Zhang, W., Yoshida, T., & Tang, X. (2011). A comparative study of TF* IDF, LSI and multi-words for text classification. *Expert systems with applications*, *38*(3), 2758-2765.