S	
E1	
E2	
For Office Use Only	

Masters Project Final Report

(MCS)

2019

Project Title	**Estimating Task Complexity of Text Analysis Tasks**
Student Name	**W.M.T. Chathurika**
Registration No. & Index No.	**Reg. No: 2017MCS014** **Index No:17440143**
Supervisor's Name	**Dr. Ruwan Weerasinghe**

For Office Use ONLY

Estimating Task Complexity of Text Analysis Tasks

A dissertation submitted for the Degree of Master of Computer Science

W.M.T. Chathurika
University of Colombo School of Computing
2020

Declaration

The thesis is my original work and has not been submitted previously for a degree at this or any other university/institute.

To the best of my knowledge it does not contain any material published or written by another person, except as acknowledged in the text.

Student Name: W.M.T. Chathurika

Registration Number: 2017MCS014

Index Number: 17440143

__________W.M.T.Chathurika___________

Signature: Date: 21-06-2020

This is to certify that this thesis is based on the work of

Mr./Ms.

under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by:

Supervisor Name:

____

Signature: Date:

Abstract

Text analysis is one of the most common approaches in machine learning applications. The process of analyzing raw data to make conclusions on those data and to find trends and answers for some questions which are known as data analytics captures a broad scope in the field of computing. “Text Analysis” is the term that is used to describe the process of computational analysis of text data. It involves numerous techniques and approaches to bring text data to an end where they can be mined for trends, patterns, or insights. The accuracy of machine learning algorithms depends on the size of the train data set. The quantity of data is affected by various factors. It depends on the complexity of the problem, training method, and diversity of inputs. Depending on the type of data, they can be expensive. Due to that, it is useful to know the amount of data needed before training a model. In this paper, the broad area of text analytics would be broadened down to text classification to reduce the complexity in the experimental approach.

In this research text classification algorithms will be trained using different datasets, that consist of different amounts of data and different features to observe the accuracies.

Acknowledgments

At the outset, I wish to express my sincere gratitude to my supervisor Dr. Ruwan Weerasinghe, senior lecturer of the University of Colombo School of Computing - UCSC , for his un- ceasing guidance, constant supervision, and providing valuable resources throughout my research.

Also, I would like to convey my utmost gratitude to all the other academic members of the University of Colombo School of Computing - UCSC for the knowledge they passed.

Finally, I would like to thank my colleagues who gave me ideas and my family for their valuable support and encouragement given endlessly, to make this project successful.

Table of Contents

Table of Figures

Chapter 1

1 Introduction

1.1 Introduction

The process of analyzing raw data to make conclusions on those data and to find trends and answers for some questions which are known as data analytics captures a broad scope in the field of computing. “Text Analysis” is the term which is used to describe the process of computational analysis of text data. It involves numerous techniques and approaches to bring text data to an end where they can be mined for trends, patterns, or insights.

The computer will generate linguistically valid interpretations of text analysis tasks such as content tagging, text extraction, entity recognition, and text classification. The use of machine learning for text analytics makes it feasible to process large amounts of text data in less amount of time. When adopting machine learning approaches customized models are created to learn through examples and improve over time.

The first step of a machine learning approach is gathering text data. Gathering data is an important task as these data are used as training samples to build the classification/extraction models in the machine learning approach. The goal is to train the models in order to make them able to analyze text and make predictions automatically. In these tasks, it is logical to want to know the amount of training data that will be needed in order to train a model. The quantity of data points needed is affected by a huge range of factors, all of which have a varying degree of influence on the eventual size of the dataset. The amount of data needed depends both on the complexity of the problem and on the complexity of the chosen algorithm. Multiple reasons demonstrate the importance of predicting the dataset size in advance of the training process. Acquiring training data for a machine learning task can be expensive. Because of that, it is crucial in machine learning projects to determine the amount of training data that is needed to achieve a specific performance goal, such as the classifier accuracy. In classification tasks, when the dataset is too small, the classifier has more degrees of freedom to construct the decision boundary which can cause overfitting [1]. As well as the train data set size, the number of its features can have a considerable effect on the outcome. For example, as more features are added, the classifiers have a high chance to find a hyperplane to split the data. If the dimensionality is increased without considering the number of training samples the feature space can become sparser and the classifier may overfit.

In this paper, the broad area of text analytics would be narrowed down to text classification in order to reduce the complexity of the experimental approach.

1.2 Problem Domain

This paper addresses an issue that arises in text data analytics. This study is correlated with heuristic methods for deciding the amount of text data required for data analytics tasks. The research will mainly focus on text classification over the other text analytics approaches.

1.2.1 Machine Learning

Machine learning is the process that powers many of the services people are using. It is a sub-branch of the tree of Artificial Intelligence. Machine learning algorithms use statistics to find patterns in large amounts of data which can be text, images, digits, etc. Search engines like Google and Baidu, social media feeds like Facebook and Twitter, voice assistants like Alexa and Siri are based on machine learning algorithms. All these applications are collecting as much data about their users as to provide the best services to them. Machine learning has three methods of learning which are supervised, unsupervised, and reinforcement. In this research, the domain of text classification will be addressed. Therefore, the research will address supervised learning. The data are labeled to tell the machine what pattern it should be looking for.

Machine learning algorithms have three phases known as implementing, training, and testing.

This research is mainly focused on the training phase, predicting the amount of data and number of features that are needed in a text classification task.

1.2.2 Text Analysis

As shown in Figure 1 text analytics is known as the automated process of deriving information that is relevant and important from unstructured text data. Computational linguistics, information retrieval, and statistical methods are being applied in text analytics approaches. Text analytics is an example of the application of machine learning.

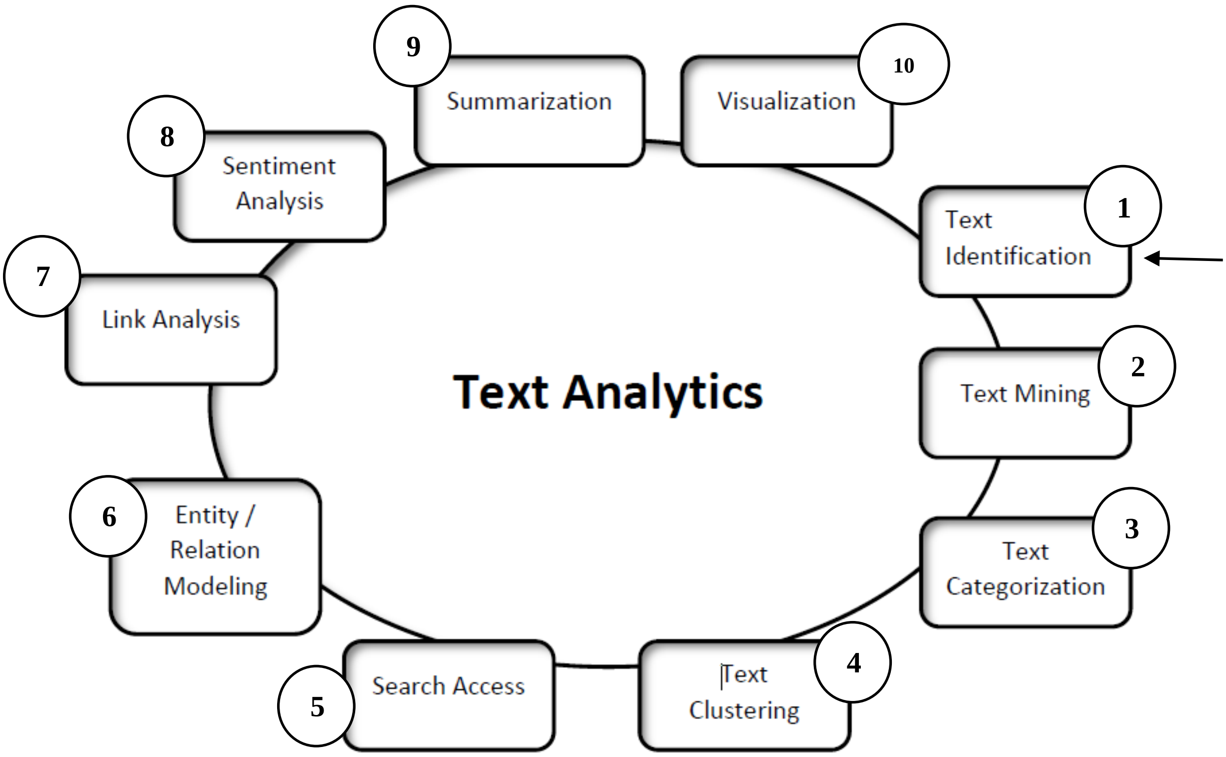


Figure 1: Text Analytics process flow:

Text analytics allows organizations to do the extraction and classification of text data obtained from emails, product reviews, survey responses, etc. These approaches let an organization extract specific information such as keywords, names, or contact details and let the organization to categorize reviews as positive and negative.

1.2.3 Text Classification

From the text analytics approaches, the research is going to mainly focus on text classification tasks. Text classification is considered one of the most important natural language processing (NLP) techniques. It is the process of assigning predetermined labels/tags/categories on unstructured data as shown in Figure 2.

In machine learning, there are two types of training methods that we can adopt in text classification using machine learning. Those are supervised learning and unsupervised learning. Supervised learning algorithms are trained using labeled data which helps to make predictions on outcomes for unforeseen data. Unsupervised learning does not require supervision over the model. The model is allowed to work on its own in order to discover outcomes. Unsupervised learning algorithms mainly use unlabeled data. Classification algorithms belong to supervised learning since training data are used to initialize the model.

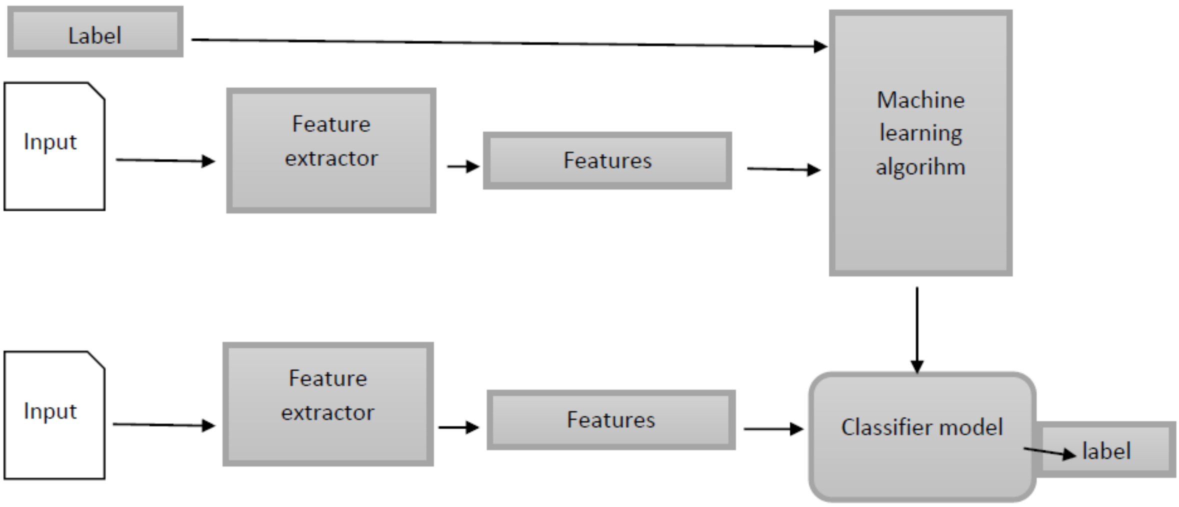


Figure 2: Text classification process

1.3 Problem

Deciding the amount of train data needed in data analytics tasks before the data gathering is a rarely touched area in the domains of modern computer science.

Machine learning algorithms that can be adopted in text analytics tasks often encounter problems with the high dimensionality of data.

The size of the training dataset has a huge impact on text analytics tasks from many perspectives. The training dataset size clearly affects the accuracy of the output of a machine learning task. In addition to the accuracy, acquiring training data in machine learning tasks is expensive when considering man-hours, equipment run time, license fees, etc. If the model is pre-trained, the amount of data needed to train the model is less than a non-pre-trained model. Because of these reasons, it is important to predetermine the amount of training data needed to achieve a particular text analytics task. But no limitations on the size of train data have been defined so far and it is considered that the larger the data set is more accurate the output would be. The question "How much data do you want exactly?" remains unanswered.

The amount of training data can vary depending on certain reasons. The training data size is determined by the complexity of the model. When the number of parameters considered by the model increases, the size of the train data set also increases. The more complex the model is, the more data are required by the model. The training method which is used to train the model causes variations in the amount of train data. Depending on the variations in the number of labels produced from data and the effort the model takes to produce those labels, input data size can vary. The size of the input dataset can be determined by the diversity of inputs.

1.4 Motivation

As mentioned in the "Problem" section the question "How much data is needed to do the classification task" remains unanswered. There is no specific method to predict the exact or approximate amount of train data which would be sufficient to get the most accurate output from a text classification model. It would be easy for researchers and those who are involved in text analysis tasks if they know the amount of data that should be collected or created before they step into the training phase.

1.5 Research Contribution

The contribution of the research is to study the effect of training data set size on the text classification models and to determine the amount of data needed by different algorithms for different features. The study targets sample size planning before the data collection step. The motivation behind the study is that the independent samples for classifier training and validation, being expensive and rare. Data collection may take a lot of effort and time. Having an insight into the data amount at the beginning can avoid the unnecessary cost for train data as the training data size depend on the nature of the task, nature of the model, and other facts such as the number of features, variables, etc.

1.5.1 Goal

To develop an approach that is capable of predicting the size of the train data set in text classification tasks.

1.5.2 Objectives

The objective of the research is to propose a method to do initial estimations of text classification tasks.

1. Estimating the amount of training data needed for a classification task before the data collection process by examining the number of features, variables, and the nature of the classification model.

2. Study the correlation between the output of a classification algorithm and the number of features in the training dataset.

3. By making the correct estimation, predict if a particular task is feasible depending on the cost and effort that has to be paid to collect train data.

1.6 Scope

This research is a sample size determination (SSD) approach that determines the size of the sample data needed for text classification tasks.

In this research, a solution to problems that arise in text analysis operations due to not knowing the train dataset size prior to the training process is developed. The scope of the research is narrowed down from text analysis tasks to text classification. In the experimental approach, the researcher is mainly focused on text classification algorithms. The research was conducted over six classification algorithms which are

- K-Nearest Neighbor Classifier
- Decision Tree Classifier
- Logistic Regression Classifier
- Stochastic Gradient Descent (SGD) Classifier
- Multinomial Naïve Bayes
- Support Vector Machines (SVM)

For all these classifiers default hyperparameters were used.

In the research, three annotated datasets were used each containing more than 20,000 labeled data.

Chapter 2

2 Literature Review

2.1 Introduction

In data analytics tasks such as text classification using deep learning, measures of the complexity in classification tasks can estimate the difficulty of dividing data into the expected classes.

One major problem with data analytics tasks is the difficulty of estimating the complexity of the task. At the beginning of the task, it is difficult to predict that, how much training data are needed to train a training model, how many classes are needed when doing classification tasks in text analytics, how many variables are needed to solve the problem. Because of that during the first stages of the task, researchers/developers do not have a good understanding of the problem domain.

There exist researches that have been conducted on estimating the complexity of software products, which provides a common estimation model for any software product. The research “A Neural Network Approach to Software Project Effort Estimation” [1] propose a model to estimate the cost, effort, and duration of a software product using artificial neural networks(ANN), prior to the implementation of the software. Though this research does not address the text analytics tasks particularly, it presents a method, using neural networks that could be adopted in other tasks. The research consists of a method of training two sets of artificial neural networks. Data to train the ANN are supplied by BT. In this research four main programs work together allowing the user to create train data sets for the ANN, normalizing the data, training the ANN, taking a trained neural network to answer questions based on estimations of cost for the particular software. Since the estimation of the duration, cost, and effort for a software project is done, the researchers have considered the cost drivers as Effectiveness of code, function points, MBI, and productive index. When applying or adopting the concepts to the proposed research, other factors that determine the complexity of text classification tasks must be considered.

“Reading Metrics for Estimating Task Efficiency with Machine Translation Output” [2] proposes that reading derived metrics are better proxies of task performance than the standard

automatic metric. The research is based on identifying better metrics that help in estimating the efficiency of a task, rather than the metrics which are already available. In this research logical puzzles have been taken into consideration. 80 different logical puzzles have been tested for participants to read metrics to estimate efficiency.

“Estimating Linguistic Complexity for Science Texts” [3] propose a method for estimating the complexity in various tasks involving Natural Language Processing, including text classification, with Recurrent neural networks. This differs from the proposed research, as the research [3] is based on estimating the “linguistic complexity for science texts” particularly.

“How Complex Is Your Classification Problem?” [5] discuss in deep on complexity measures for text classification tasks dividing them into categories as Feature-based measures, Linearity measures. Neighborhood measures, Network measures, Dimensionality, and class imbalance measures. It describes how a classification task can be succeeded or failed depending on these features. But it does not address how to make estimations on the classification task such as the amount of training data needed for a particular task, number of variables, etc.

The research “Complexity measures of supervised classification problems” [6] consider the geometric complexity of a class boundary in measuring the difficulty of a classification task. The research addresses a small set of classification problems to observe up to which extent a training dataset represents a test set.

[7] is an approach to approximate the relationship of input-output among many variables. It is an experimental approach which is conducted to show how generalized sampling theorem can be applied for approximation problems using neural network. In their study, they propose the least in size training data set can be found for any multi-dimensional function based on the knowledge of the frequency power spectrum. Though the research discusses the train data size approximation, it is not based on text analytics tasks and does not propose an overall idea on how to determine the dataset size for any model rather than two-layered neural networks.

“Sensitivity of hyperspectral classification algorithms to training sample size” [8] is research on determining the sensitivity of Multi-Classifier Decision Fusion (MCDF) and MCDF framework in the Discrete Wavelet Transform domain (DWT-MCDF) for a limited number of train data. It is an experimental approach where they apply feature extraction and classification methods over different numbers of train data to obtain the conclusion that even with sufficient training data classification tasks can fail due to poor performance due to the poor quality of train data. In the experiment, the dimensionality of data also has been changed with the data amount to observe the sensitivity of classifiers on the quality of data.

The research “Impact of training corpus size on the quality of different types of language models for Serbian” [9] has been conducted in the domain of Natural Language Processing (NLP) in the Siberian language. The paper describes the relationship between the quality of the language model and the size of the textual corpus which is used in the training process. The research has been conducted on three types of n-grams which are word-based which is trained on the textual corpus, lemma-based that is trained on a corpus consisting of lemmas, and class-based that is trained on a corpus of word classes. The three language models have been trained using the SRILM toolkit with similar values of datasets with different sizes and different vales of data with different sizes. The researchers have concluded the research with results stating that the lemma model and word-based model require “more” training data to deliver results with more accuracy. But the researcher does not present that amount numerically as an exact or approximate value. Therefore, even though the research has been conducted following a similar approach as this paper, the researcher of [9] fails to present a method to predict or approximate the train data size for a given data analytics task.

The research “Effect of Training Set Size on SVM and Naïve Bayes for Twitter Sentiment Analysis” [10] discusses the impact of training data size on the accuracy of classification of text. They address the issue of approximating the data set size for the classification, being within the frame of the two famous algorithms, SVM and Naïve Bayes.

Chapter 3

3 Methodology

3.1 Introduction

The methodology of the research is more experimental and addressed with the knowledge gained through the literature review. As mentioned in previous chapters, the scope of the research is limited to text classification. In the broad area of machine learning, the research touches on the supervised learning method. As mentioned in the previous chapters, the goal of the research is to develop a method to predict the accurate or approximate amount of data needed for a text classification task with high accuracy. This chapter provides a comprehensive overview of the steps which were followed in the implementation process.

In this chapter, the selection of datasets, machine learning tools, and techniques that were used in the classification process and reasons for those selections are discussed in deep.

3.1.1 Problem Analysis

In the text classification, the data can be categorized into a hundred or more categories depending on the requirement. The most efficient approach of classification is using machine learning approaches to categorize text data. In the machine learning approach, a model needs to be created for the classification task and that model should be trained with training data. Then the model learns itself to generate outputs for similar inputs after being well trained. The traditional and common point of view is that the size of this training input data for machine learning (supervised learning) should be high to obtain the most accurate outputs.

But collecting data is an expensive task. Because of that having an insight into the exact/approximate amount of data for a particular task is important rather than training the model blindly for a huge amount of training data.

3.1.2 Model / Design

According to the proposed concept, text classification algorithms are trained and tested in scikit learn with datasets with different sizes, different numbers of features, and variables. The relationship between the accuracy and the training dataset is examined to approximate the size of the train data needed for a classification task.

3.2 Dataset Selection

When selecting text data, they were specifically chosen depending on the dataset size, the number of attributes, and the number of classes. The research is focused on text classification of the English language.

In the research, three datasets were used which contain more than 20000 annotated data. These three datasets consist of two features, which are being encoded as 1 and 0.

Dataset 1

The initial dataset was obtained from the UCI Machine Learning Repository. All selected data are labeled data. The initial dataset contains over 5000 SMS messages labeled as spam and non-spam.

```
= RESTART: C:\Users\Thili\Python38\final\with_SMSSpamCollection_binaryclassifica
tion\with_bag of word\sample5_svm - V2.py
Python: 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916 64 bit (A
MD64)]
NLTK: 3.4.5
Scikit-learn: 0.22.1
Pandas: 1.0.1
Numpy: 1.18.1
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       5572 non-null   object
 1   1       5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
None
      0                                                  1
0   ham  Go until jurong point, crazy.. Available only ...
1   ham                      Ok lar... Joking wif u oni...
2  spam  Free entry in 2 a wkly comp to win FA Cup fina...
3   ham  U dun say so early hor... U c already then say...
4   ham  Nah I don't think he goes to usf, he lives aro...
ham     4825
spam     747
```

Figure 3: Details of the initial dataset

According to figure 3, the dataset consists of 4825 non-spam messages and 747 spam messages. The dataset consists of 5575 SMS messages. These messages were preprocessed by following the below steps.

Firstly, the class labels were converted into binary values using LabelEncoder from sklearn. The rest of the data were preprocessed by replacing email addresses, URLs, phone numbers, and other symbols using regular expressions.

```
from sklearn.preprocessing import LabelEncoder

# convert class labels to binary values, 0 = ham and 1 = spam
encoder = LabelEncoder()
Y = encoder.fit_transform(classes)

print(Y[:10])
```

Figure 4: Using Label Encoder

```
# use regular expressions to replace email addresses, URLs, phone numbers, other numbers
# Replace email addresses with 'email'
processed = text_messages.str.replace(r'^.+@[^\.].*\.[a-z]{2,}$', 'emailaddress')
# Replace URLs with 'webaddress'
processed = processed.str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$','webaddress')
# Replace money symbols with 'moneysymb' (£ can by typed with ALT key + 156)
processed = processed.str.replace(r'£|\$', 'moneysymb')

# Replace 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenumber'
processed = processed.str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$','phonenumbr')

# Replace numbers with 'numbr'
processed = processed.str.replace(r'\d+(\.\d+)?', 'numbr')

# Remove punctuation
processed = processed.str.replace(r'[^\w\d\s]', ' ')

# Replace whitespace between terms with a single space
processed = processed.str.replace(r'\s+', ' ')

# Remove leading and trailing whitespace
processed = processed.str.replace(r'^\s+|\s+?$', '')

# change words to lower case - Hello, HELLO, hello are all the same word
processed = processed.str.lower()
print(processed)

from nltk.corpus import stopwords

# remove stop words from text messages

stop_words = set(stopwords.words('english'))
```

Figure 5: Data preprocessing

Dataset 2

The second dataset contains 25000 movie reviews obtained from IMDb. These data are labeled as positive and negative which is suitable for binary sentiment classification. The overall distribution of the dataset was balanced.

Dataset 3

The third publicly available dataset contains 21000 tweets which are classified as aggressive and non-aggressive. The dataset consists of multiple features. One feature was selected in order to make it feasible to use for binary classification.

3.3 Feature Extraction

In the research, how the accuracy is affected by the feature selection is also examined. The process of feature selection is crucial in machine learning tasks as machines cannot process text data in raw form. Raw data need to be broken down into a numerical format to make them readable by the machine. In this task, features are created using the domain knowledge of the data for the classification algorithms. Three methods were followed to do the feature extraction task which are

- Bag of Words
- Count Vectorizer
- Tfidf Vectorizer (Term Frequency-Inverse Document Frequency)

3.3.1 Bag of Words

When using Bag of Words for feature creation, the words in each processed text message are considered as features. Because of that, the processed set of messages were tokenized into words. 1500 most common words were considered as features.

```
#---------------------generating features--------------------------
from nltk.tokenize import word_tokenize

# create bag-of-words
all_words = []

for message in processed:
    words = word_tokenize(message)
    for w in words:
        all_words.append(w)

all_words = nltk.FreqDist(all_words)


# print the total number of words and the 15 most common words
print('Number of words: {}'.format(len(all_words)))
print('Most common words: {}'.format(all_words.most_common(15)))


# use the 1500 most common words as features
word_features = list(all_words.keys())[:1500]


# The find_features function will determine which of the 1500 word features are contained in the review
def find_features(message):
    words = word_tokenize(message)
    features = {}
    for word in word_features:
        features[word] = (word in words)

    return features
```

Figure 6: Using Bag of Words for feature creation

3.3.2 Count Vectorizer

CountVectorizer is the simplest method of vectorizing text. Count Vectorizer can implement both tokenization and occurrence counting. In this research the dataset was trained, using Count Vectorizer in feature creation. Firstly the raw data were processed by removing unnecessary terms. Then the data set was split into train data and test data. Count vectorizer was applied to train data and test data to vectorize them.

```python
processed = processed.apply(lambda x: ' '.join(
    term for term in x.split() if term not in stop_words))
# Remove word stems using a Porter stemmer
ps = nltk.PorterStemmer()
processed = processed.apply(lambda x: ' '.join(
    ps.stem(term) for term in x.split()))


#Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(processed, Y, test_size=0.25)
print(len(X_train))
print(len(X_test))
print(len(y_train))
print(len(y_test))

#Transform the training data into count vectors


vectorizer = CountVectorizer(min_df=0.0, max_df=1.0,
                             ngram_range=(1,3))
X_train_cv = vectorizer.fit_transform(X_train).astype(float)



#Transform the test data into count vectors


vectorizer = CountVectorizer(min_df=0.0, max_df=1.0,
                             ngram_range=(1,3))
X_test_cv = vectorizer.fit_transform(X_test).astype(float)

docs_test = X_test
#
```

Figure 7: Applying CountVectorizer

```
['aa', 'aa exhaust', 'aa exhaust hang', 'aah', 'aah speak', 'aah speak tomo', 'a
aooooright', 'aaooooright work', 'aathi', 'aathi dear', 'aathi love', 'aathi lov
e lot', 'abbey', 'abbey happi', 'abbey happi new', 'abdomen', 'abdomen gyna', 'a
bdomen gyna infect', 'abel', 'abel havnumbrhear', 'abel havnumbrhear sn', 'aberd
een', 'aberdeen unit', 'aberdeen unit kingdom', 'abi', 'abi hw', 'abi hw keep',
'abi say', 'abi say hi', 'abiola', 'abj', 'abj serv', 'abj serv stay', 'abl', 'a
bl anyth', 'abl buy', 'abl buy liquor', 'abl come', 'abl deliv', 'abl deliv basi
c', 'abl dont', 'abl dont know', 'abl eat', 'abl get', 'abl get half', 'abl get
littl', 'abl give', 'abl give or', 'abl go', 'abl go shop', 'abl join', 'abl num
br', 'abl numbr friday', 'abl numbr met', 'abl pay', 'abl pay charg', 'abl rais'
, 'abl rais lt', 'abl show', 'abl show much', 'abl text', 'abl text readi', 'abn
orm', 'abnorm call', 'abouta', 'abouta much', 'abouta much chanc', 'absenc', 'ab
senc gud', 'absenc gud mrng', 'absolutli', 'absolutli fine', 'abstract', 'abstra
ct still', 'abstract still wake', 'abt', 'abt alreadi', 'abt alreadi concentr',
'abt alreadi rite', 'abt dvg', 'abt dvg cold', 'abt event', 'abt event esp', 'ab
t function', 'abt function thnk', 'abt leona', 'abt leona oop', 'abt make', 'abt
 make pic', 'abt movi', 'abt movi wan', 'abt muz', 'abt muz call', 'abt numbr',
'abt numbr row', 'abt syd', 'abt syd leh', 'abt tat', 'abt tel', 'abt tht', 'abt
```

Figure 8: features generated by CountVectorizer

One drawback of Bag of Words representation is the generation of a sparse matrix. It only focuses on word representation ignoring the relationship between neighboring words. But as shown in Figure 8, CountVectorizer considers N-gram features. By using different types of feature extraction techniques in this research, different domains of the same problem have been addressed.

3.3.3 Tfidf Vectorizer (Term Frequency-Inverse Document Frequency)

Tfidf (Term Frequency-Inverse Document Frequency) Vectorizer calculates how frequently a word appears in a context. Through the frequency of the word, it calculates a score for each word. Therefore, words with high frequencies get a higher score, while words with less frequency obtain a low score. Each word is been allocated a weight value proportional to the appearance in the context. The set of words with the highest scores is used to represent the document. Hence those words are being identified as features.

The *term frequency* is a ratio of the count of a word's occurrence in a document and the number of words in the document. Thus, it is a normalized measure that takes into consideration the document length. Let us show the count of the word *i* in document *j* by tf_{ij}. The *document frequency* of word *i* represents the number of documents in the corpus with the word *i* in them. Let us represent document frequency for word *i* by df_i. With *N* as the number of documents in the corpus, the tf-idf weight w_{ij} for word *i* in document *j* is computed by the following formula: [17]

$$w_{ij} = tf_{ij} \times (1 + \log\tfrac{1+N}{1+df_{ij}})$$

```
#Transform the training data into tfidf vectors

vectorizer = TfidfVectorizer(min_df=0.0, max_df=1.0,
                             ngram_range=(1,3))
X_train_tfidf = vectorizer.fit_transform(X_train).astype(float)


#Transform the test data into tfidf vectors

vectorizer = TfidfVectorizer(min_df=0.0, max_df=1.0,
                             ngram_range=(1,3))
X_test_tfidf = vectorizer.fit_transform(X_test).astype(float)

docs_test = X_test
#....................
```

Figure 9: Application of TfidfVectorizer

3.4 Handle data imbalance

In the classification process balancing data is a crucial step. Data imbalance problem occurs when the amount of data in one class outnumbers the other classes by a comparatively large portion. Having to deal with imbalanced problems in the dataset can be excessive work. But this is important as balancing data leads to inaccurate accuracy metrics and the inefficiency of production performance. [18]

In the research, the objective is to observe the maximum accuracy scores for different samples of data. The same set of classifiers is being tested against different samples of data increasing by 500 in each iteration.

Data will be selected randomly for a particular iteration. In case the selected data consists of imbalanced classes, the accuracy of that particular iteration will be reduced regardless of the sample size.

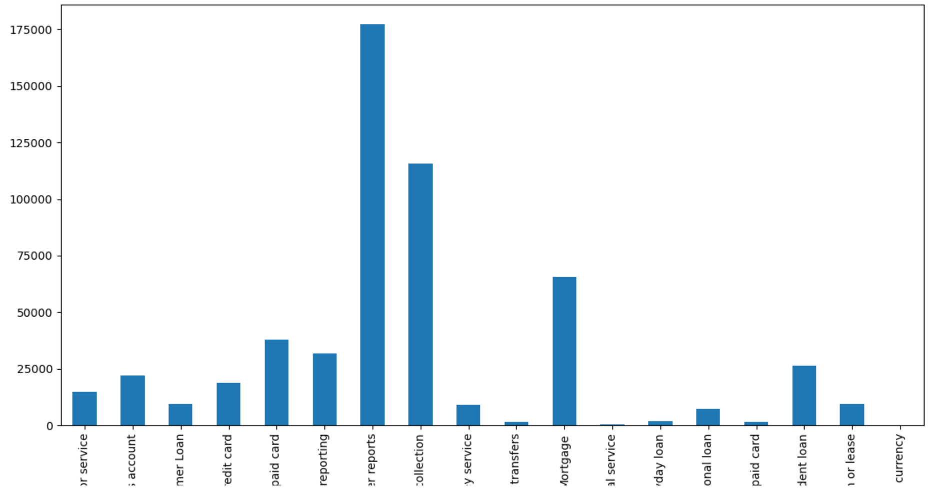


Figure 10:Data distribution for the first 500 data rows in Movie_reviews classification dataset

Figure 10 shows the distribution of the selected sample of 500 data rows from the second data set (Movie reviews). This distribution can produce accuracy scores which can also be misleading.

As the classifiers can turn out to be overfitted as a result of data imbalance, it can create a false sense of high accuracy. As the conclusions have to be made depending on these accuracy scores it is important to eliminate the data imbalance for each selected data sample.

To achieve this task Synthetic Minority Over Sampling Technique for imbalanced data (SMOTE) was used. SMOTE chooses a minority class input variable in its process. Then finds it's K-nearest neighbor. K-neighbor is required to be specified as an argument in the SMOTE() function. One of the neighbors is selected and a synthetic point is placed on the line that joins the point under consideration of the neighbor.

This process will be repeated until the data is balanced. [19] . The synthetic instances generated by the SMOTE function are being added to the original dataset. Then the modified data set is passed to the classifier.

This approach mitigates the problem of overfitting produced due to the random oversampling as the SMOTE function generates synthetic examples rather than a replication of instances.

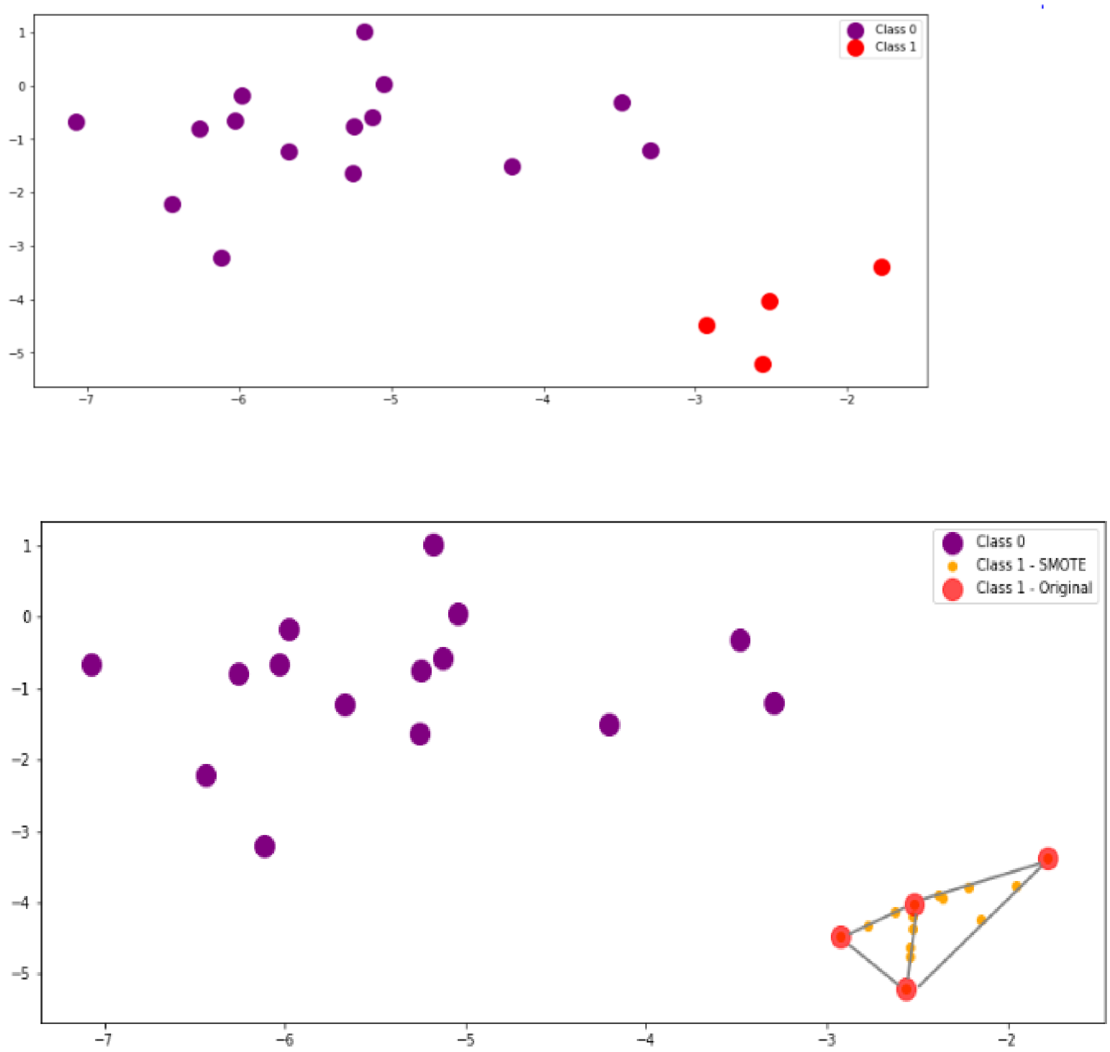


Figure 11:Generating synthetic instances

3.5 Text Classification

From the text analytics approaches, the research is going to mainly focus on text classification tasks. Text classification is considered one of the most important natural language processing (NLP) techniques. It is the process of assigning predetermined labels/tags/categories on unstructured data.

In machine learning, there are two types of training methods that we can adopt in text classification using machine learning. Those are supervised learning and unsupervised learning. Supervised learning algorithms are trained using labeled data which helps to make predictions on outcomes for unforeseen data.

Unsupervised learning does not require supervision over the model. The model is allowed to work on its own to discover outcomes. Unsupervised learning algorithms mainly use unlabeled data. Classification algorithms belong to supervised learning since training data are used to initialize the model.

In the research 6 classification algorithms were used to obtain the accuracy values.

- K Neighbors Classifier
- Decision Tree Classifier
- Logistic Regression
- SGD Classifier
- Multinomial NB
- SVC

A classifier pipeline was created to achieve this target. For each classification algorithm, default hyperparameter values were used.

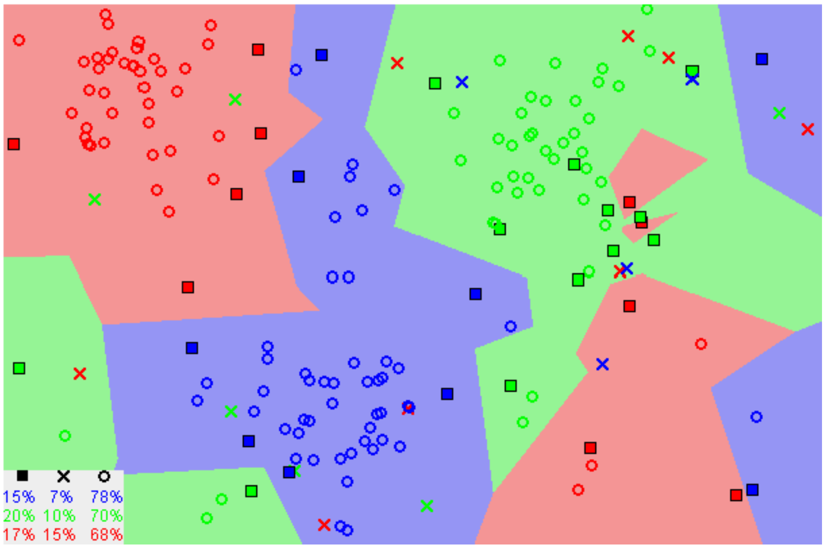
```
#Construct the classifier pipeline using a SGDClassifier algorithm
    print ('\nApplying the classifier...\n')
    text_clf = Pipeline([('vect', CountVectorizer(stop_words='english')),
                         ('tfidf', TfidfTransformer(use_idf=True)),
                         ('clf',clf)
    ])
    #Fit the model to the training data
    text_clf=text_clf.fit(X_train, y_train)

    predicted = text_clf.predict(docs_test)
    #Calculate mean accuracy of predictions
    accuracy = (np.mean(predicted == y_test))*100
    print(accuracy)
    worksheet.write(row, col, name)
    worksheet.write(row, col + 1, accuracy)
    row += 1
workbook.close()
```

Figure 12: clasifier pipeline

At the end of each classification task, accuracy was calculated for each classification algorithm.

- **K Neighbors ClassifierK**

K- Nearest Neighbor (KNN) is a supervised machine learning algorithm that relies on labeled data to learn. When doing classification, the KNN algorithm assumes that similar objects are exciting in close proximity.

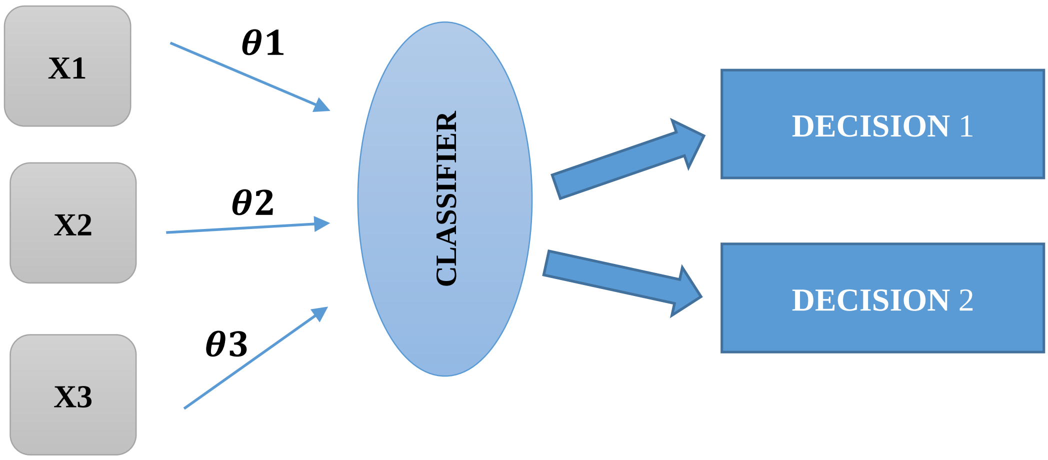


In the classification process, KNN should be initialized with the number of neighbors. In this research, the default values were taken. For each data, the distance between the query example and the current example from data should be calculated. The index of the example data and the distance is being put into an ordered collection. Then the collection is being sorted in ascending order by the distances. The first K entries from the sorted collection are then selected and the labels are checked. Finally, the mode of the K label is returned. The disadvantage of using this model in the research was slowing down the overall performance of the script as it is significantly slower when the number of examples is increased.

- **Decision Tree Classifier**

In indecision tree classification the set of training samples are being split into smaller subsets while the decision tree is developed incrementally. At the end of the learning process of the classifier, a decision tree that covers the provided training set is being returned. The decision tree partitions the provided data into clusters and empty regions. [20] . This is a widely used method in classification tasks which is very straightforward. The classifier is organized with a set of test questions and conditions. Starting from the root node these teat conditions are applied to the data sample until it reaches the leaf node. The label associated with the leaf node is getting assigned to the particular record of the provided data sample.

- **Logistic Regression**



In this research binary logistic regression was used in binary classification tasks. Using maximum – likelihood estimation, the classifier estimates the coefficients. Maximum likelihood estimation is a widely used learning algorithm that is used by many other machine learning algorithms. The best coefficients will result in a model that predicts a value that is closer to one for the default class and a value closer to 0 for the other class. When making predictions using the test data, the logistic regression model will generate a value and by looking at this value (if it is closer to one or zero) the decision would be made.

- **Stochastic Gradient Descent (SGD) Classifier**

In SGD classification some random samples from the given dataset are selected instead of the entire dataset for each iteration. The sample is randomly shuffled and selected to perform the iteration

$$for\ i\ in\ range\,(m):$$
$$\theta_j = \theta_j - \alpha\,(\hat{y}^i - y^i)\,x_j^i$$

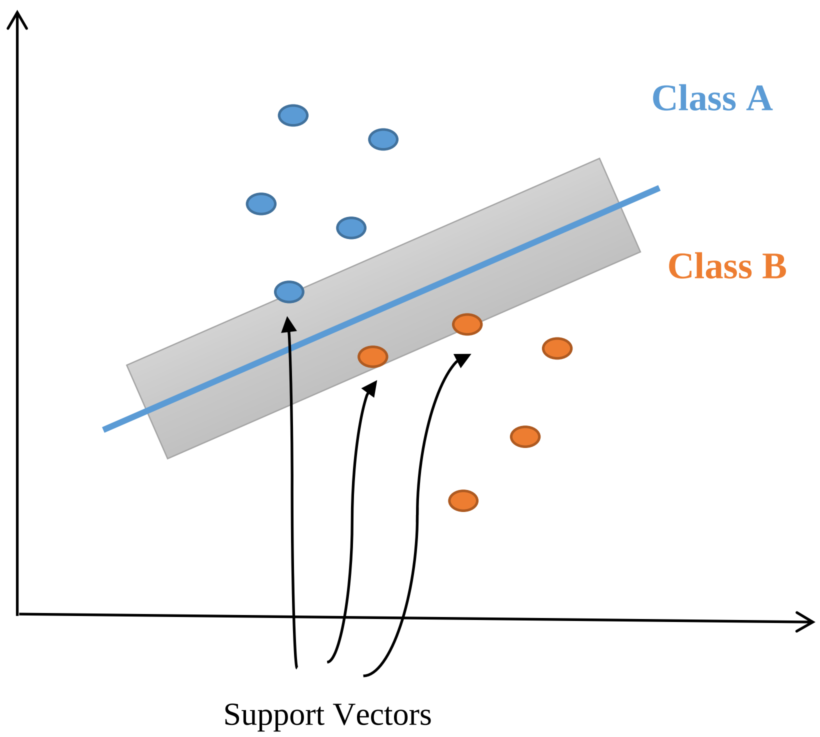
- **Multinomial Naïve Bayes Classifier**

This algorithm is more suitable for the classification of discrete features.

Multinomial Naïve Bayes accepts feature counts as integers but still, it accepts fractional counts such as TFIDF that is used in this research. In the research, the same set of samples are being vectorized using Count Vectorizer and TFIDF vectorizer. [21]

- **Support Vector Classifier (SVC)**

SVC returns the best fit hyperplane that divides the data provided to the model. This generates a high accuracy compared to other classification models such as logistic regression and decision tree.



A hyperplane separates the objects that belong to different classes. Support vectors are the data points that stay close to the hyperplane decided by the svc model. The separating line is defined by these points by calculating the margins. The margin is a gap placed in between the two lines on the closest class points. Margin is calculated the perpendicular distance between these two points.

In the classification process, SVC generates the hyperplane to separate the classes in the best way.

Chapter 4

4 Results and Evaluation

4.1 Overview

This paper addresses an issue that arises in text data analytics. This study is correlated with heuristic methods for deciding the amount of text data required for data analytics tasks. The research will mainly focus on text classification over the other text analytics approaches.

"Text Analysis" is the term which is used to describe the process of computational analysis of text data. It involves numerous techniques and approaches to bring text data into an end where they can be mined for trends, patterns, or insights. The computer will generate linguistically valid interpretations of text analysis tasks such as content tagging, text extraction, entity recognition, and text classification. The use of machine learning for text analytics makes it feasible to process large amounts of text data in less amount of time. When adopting machine learning approaches customized models are created to learn through examples and improve over time.

The first step of a machine learning approach is gathering text data. Gathering data is an important task as these data are used as training samples to build the classification/extraction models in the machine learning approach. The goal is to train the models to make them able to analyze text and make predictions automatically.

In these tasks, it is logical to want to know the amount of training data that will be needed to train a model. In this paper, the broad area of text analytics would be narrowed down to text classification to reduce the complexity in the experimental approach.

4.2 Evaluation Approach

Each independent execution is involved in an evaluation task. In this research, results are being generated for different sets of inputs. (train data and test data). And will be observed, the different outputs generated for the classification algorithm for different vectorization methods.

The evaluation approach is a combination of experimental and mathematically based approaches, at the same time it does outcome mapping which is an impact evaluation approach which unpacks an initiative's theory of change, provides a framework to collect data on immediate, basic changes that lead to longer, more transformative change, and allows for the plausible assessment of the initiative's contribution to results.

In the research different text classification methods will be executed with different sets of data, containing different amounts and different attributes. A confusion matrix will be generated in which the accuracy of the classification task is obtained. It will experiment on how the change of dataset size affects the accuracy of the classification task. And in the research, will experiment on how the changes of features result in different outcomes in text classification.

4.3 Data sets involved with the evaluation

Multiple sets of publicly available text data are used in the research. Data sets are obtained from the "UCI Machine Learning Repository " (https://archive.ics.uci.edu/ml/index.php). The UCI Machine Learning Repository is a collection of databases that are used by the machine learning community for the empirical analysis of machine learning algorithms.

Different datasets containing different numbers of attributes are used in the research. The research is mainly focused on text classification tasks. Therefore, datasets with different numbers of classes are used. Observing the outputs generated for different conditions (different vectorization mechanisms, different number of attributes), the combination between success rate and the size of the dataset will be identified.

In this

4.4 Results

As mentioned in previous chapters the objective of the research is to observe the accuracy of text classification algorithms for different sample sizes of data. In this approach, accuracy was calculated for the trained model, and then prediction was made for the test dataset to check the accuracy of the model for unseen data.

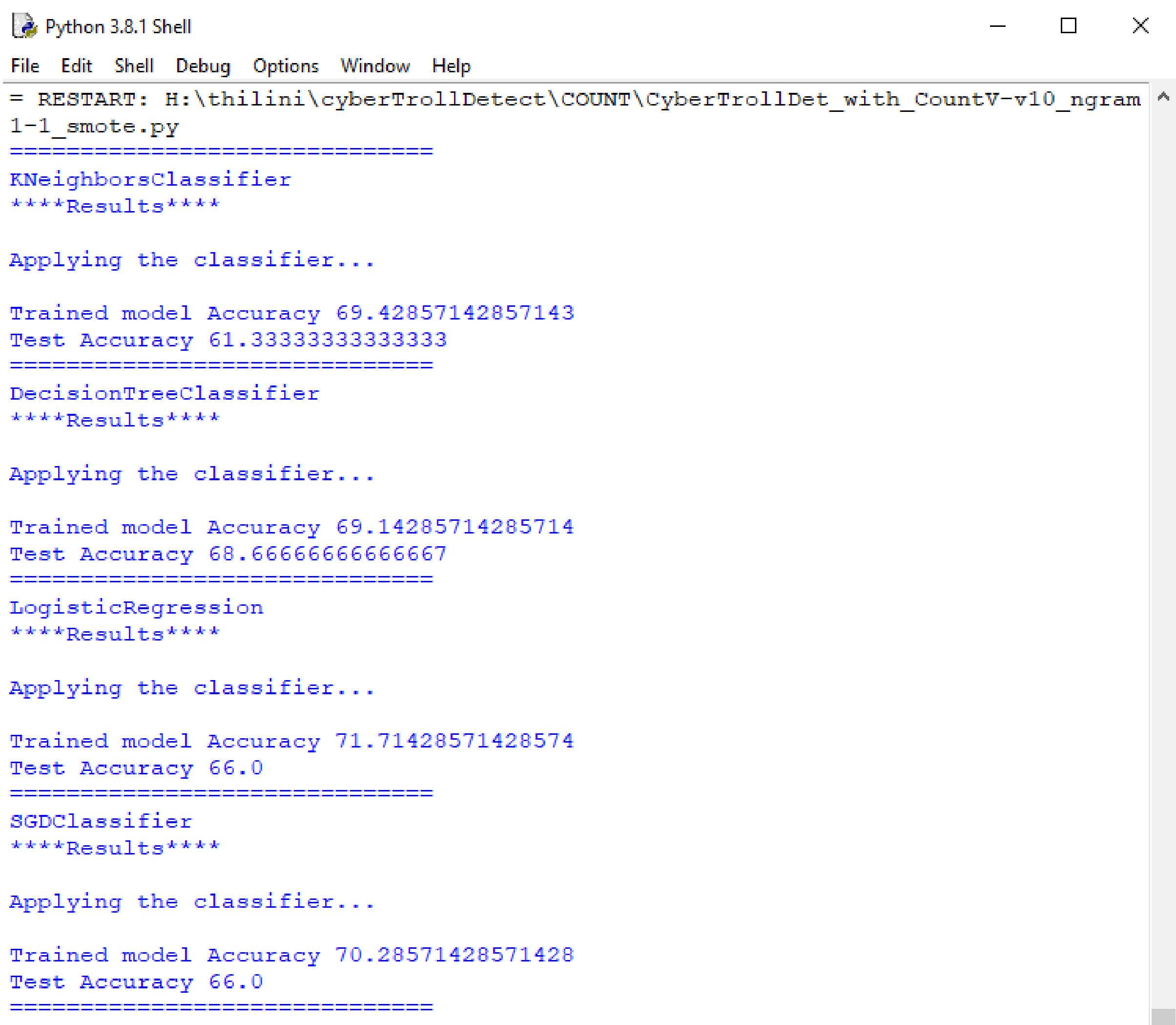


Figure 13: Results generated for CyberTroll Detection dataset with count vectorizer and 1,1 ngram range

Figure 14 summarizes the results generated from the six classifiers described in the previous chapter. results were generated for samples of data starting from 500 sample size. Then the sample size is increased by 500. The same task is performed for both count vectorizer and tfidf vectorizer for ngram ranges (1,1), (1,2), and (1,3).

Cyber Troll Detection dataset - Count vectorizer - Ngram range 1,1												
Sample Size	KNeighborsClassifier		DecisionTreeClassifier		LogisticRegression		SGDClassifier		MultinomialNB		SVC	
	train	test	TRAIN	TEST	TRAIN	TEST	TRAIN	TEST	TRAIN	TETS	TRAIN	TEST
500	50.45455	36	60.98309	58.66667	65.80867	59.33333	64.65645	62.66667	54.34461	64.66667	61.96617	52
1000	51.37017	38	65.01959	51.33333	69.0256	59.66667	66.85606	55.66667	61.55695	62.33333	64.93992	49.66667
1500	52.03748	40.22222	66.3118	55.11111	68.26669	62.88889	67.44874	56.88889	64.01971	65.77778	68.31169	53.77778
2000	51.56425	42.83333	64.35754	56.33333	69.21788	64.83333	65.36313	59	69.38547	67.16667	69.10615	61
2500	53.5514	38.53333	66.86916	57.46667	70.37383	69.06667	66.58879	58.53333	68.36449	65.2	71.21495	72
3000	53.1533	46	67.89967	60.88889	69.98445	68.11111	68.47001	60	67.82161	67.11111	72.83644	71.66667
3500	53.25141	42.57143	67.38169	59.61905	69.92827	68.47619	67.78553	61.71429	69.70221	65.90476	71.22836	70.95238
4000	55.09002	44.75	66.88007	61.91667	71.07589	71.75	68.37689	65.16667	68.89239	69.66667	72.60343	74.25
4500	55.19437	45.25926	67.32691	62.2963	70.70208	72.07407	69.68306	67.11111	68.29674	70	73.05856	75.33333
5000	55.72014	46.46667	69.00932	64.06667	71.72508	69.8	68.98351	67	69.79151	68.46667	73.91929	73.2
5500	57.28355	48	69.33132	62.90909	72.74712	70.48485	72.08527	69.63636	69.47953	69.63636	74.09368	75.15152
6000	56.99742	48.33333	69.55175	65.44444	72.52982	71.66667	70.3195	70.66667	68.99402	70.61111	74.85712	75.33333
6500	57.35242	53.28205	70.64401	61.89744	73.48837	71.33333	72.93381	70.76923	73.1127	70.41026	76.01073	73.74359
7000	57.30807	46.85714	70.9406	65.42857	72.32329	73	72.57334	72.47619	70.3382	71.80952	75.66121	76.19048
7500	58.72473	48.93333	72.06843	65.86667	74.23017	73.15556	74.13686	73.06667	71.77294	70.93333	76.85848	76.66667
8000	57.52186	48.08333	71.70714	64.875	74.84784	72.25	74.6007	72.83333	71.6475	71.33333	77.39443	75.79167
8500	58.53573	50.66667	71.26385	67.4902	74.47639	74.70588	74.18681	74.19608	73.56683	71.72549	77.52418	78.35294
9000	59.15475	50.14815	72.60078	67.44444	75.18856	72.25926	75.14954	74.22222	72.41873	70.48148	78.15345	77
9500	62.76842	58.80702	72.77533	68.35088	75.3154	74.63158	75.46311	74.59649	72.18289	71.4386	78.53596	78.38596
10000	59.91784	52.16667	72.96948	68.13333	75.69249	74.16667	75.83333	75	72.30047	71.86667	79.34272	78.5

Figure 14: Accuracy scores for Cyber Troll Detection dataset - Count vectorizer - Ngram range 1,1

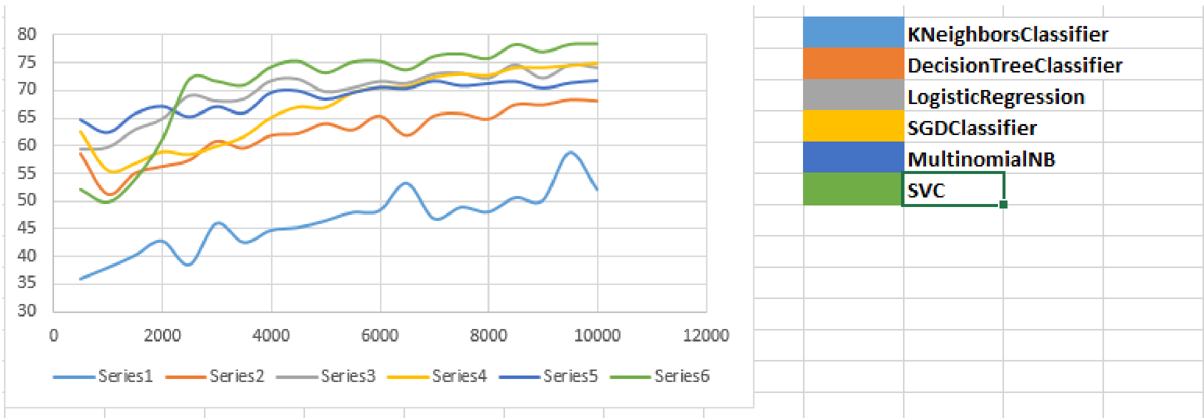


Figure 15:Figure 14:Graph for accuracy scores generated on Cyber Troll Detection dataset - Count vectorizer - Ngram range 1,1

Cyber Troll Detection dataset - Count vectorizer - Ngram range 1,2												
Sample size	KNeighborsClassifier		DecisionTreeClassifier		LogisticRegression		SGDClassifier		MultinomialNB		SVC	
	train	test	TRAIN	TEST	TRAIN	TEST	TRAIN	TEST	TRAIN	TEST	TRAIN	TEST
500	57.22222	40	59.33333	56.66667	60.88889	53.33333	65.55556	62	51.33333	66.66667	60.66667	46
1000	58.03318	36.66667	65.58516	57	63.66118	60.33333	62.74164	61.33333	63.99425	61.66667	61.95402	48.66667
1500	58.15742	38.22222	66.38819	56.44444	66.77818	62	67.24568	60.66667	71.10316	68.88889	63.71795	53.33333
2000	58.49691	36	65.08239	61	66.23912	61.66667	65.83416	62.66667	67.48057	69.33333	63.13866	54.83333
2500	59.02957	40.4	66.10386	59.33333	70.85616	63.46667	67.85928	58.66667	71.71062	68.8	69.15984	57.6
3000	60.07769	42.77778	67.93467	59.33333	73.21955	64.88889	69.15342	59.22222	75.27091	69.66667	67.97341	64.88889
3500	60.94575	41.2381	68.05004	60.19048	71.93033	65.42857	67.98706	61.04762	71.89689	70.28571	65.65067	62.57143
4000	60.96129	43.41667	67.97079	61.08333	70.17993	66.5	68.78899	60.5	79.92093	68.83333	71.49485	70.33333
4500	61.18182	44.07407	69.79221	64.74074	68.85714	72.96296	68.20779	63.62963	75.35065	71.62963	71.45455	73.55556
5000	61.2757	42.6	69.88318	63.4	68.66822	69.6	69.01869	63.86667	76.1215	70.66667	71.84579	72.13333
5500	59.99313	42.24242	70.77635	63.93939	70.32733	69.21212	70.90527	63.75758	76.84275	72.06061	72.49541	72.9697
6000	61.73655	44.88889	71.50745	64.33333	70.96335	69.27778	72.07172	62.83333	77.83744	71.83333	73.89482	73.88889
6500	63.47842	45.79487	71.54218	65.74359	68.96839	70.92308	71.07885	66.51282	76.68274	72.5641	70.8087	72.10256
7000	64.67442	46	72.84053	66.47619	73.00664	73.09524	73.70432	67.61905	79.25249	72.42857	75.83056	73.85714
7500	63.02997	47.11111	72.80659	68.17778	74.13454	75.24444	74.09104	70.08889	81.35074	73.91111	76.3307	76.44444
8000	63.82916	47.66667	72.84658	67.91667	72.51017	75.75	73.99829	71.25	78.86949	76	75.65591	76.66667
8500	63.9816	49.64706	73.16285	69.37255	74.46957	75.92157	72.98345	71.37255	80.21447	74.27451	76.52207	76.35294
9000	64.93014	48.33333	73.55757	69.18519	75.48512	74.88889	73.79043	71.62963	80.3881	74.81481	77.32212	75.7037
9500	64.87599	50.42105	73.96032	69.22807	75.91787	77.22807	73.83596	73.54386	81.73982	75.33333	77.53336	76.5614
10000	66.22161	51.8	73.62729	70.8	76.16374	77.23333	74.33955	76.53333	81.67025	75.53333	78.68043	77.63333

Figure 16: Figure 14:Accuracy scores for Cyber Troll Detection dataset - Count vectorizer - Ngram range 1,2

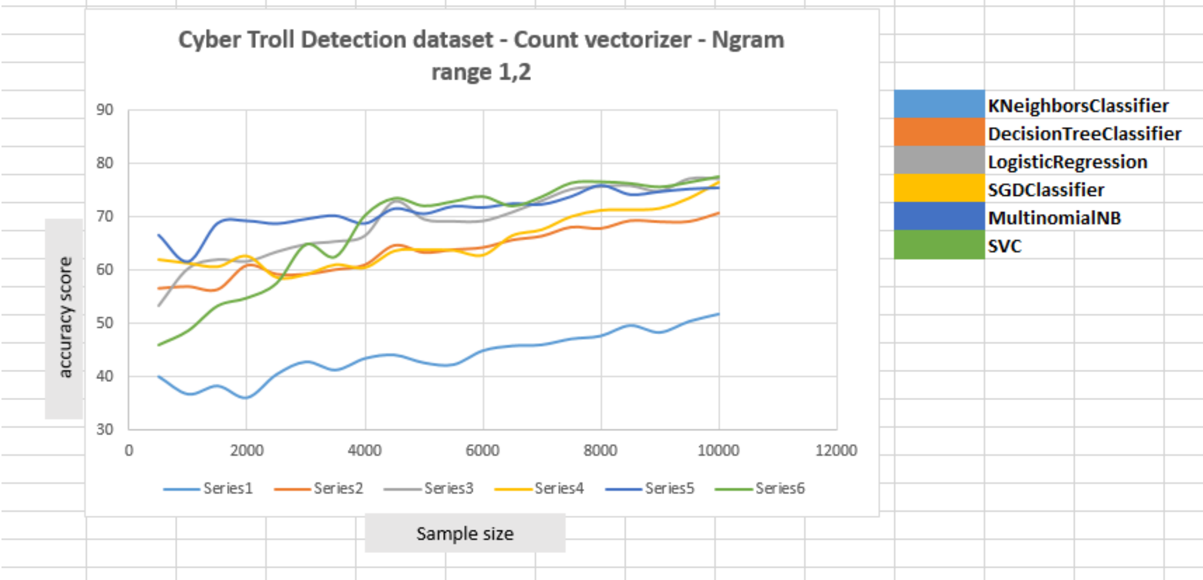


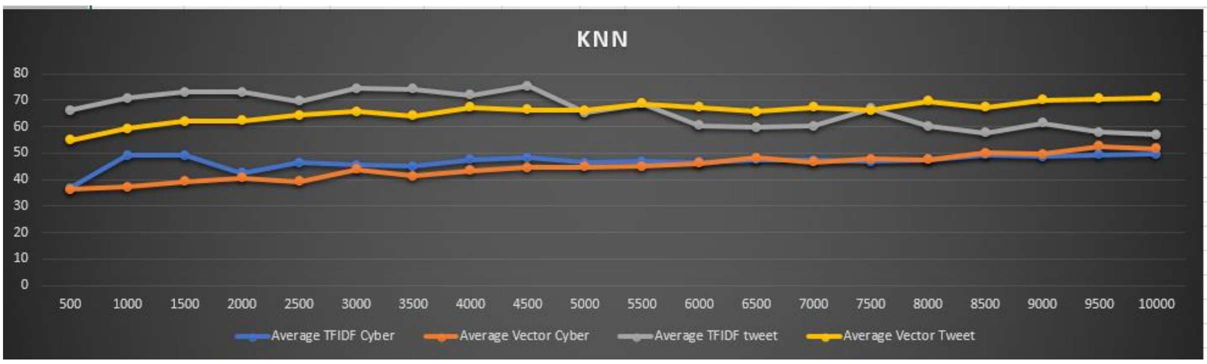
Figure 17:Figure 15:Figure 14:Graph for accuracy scores generated on Cyber Troll Detection dataset - Count vectorizer - Ngram range 1,2

4.5 Conclusions

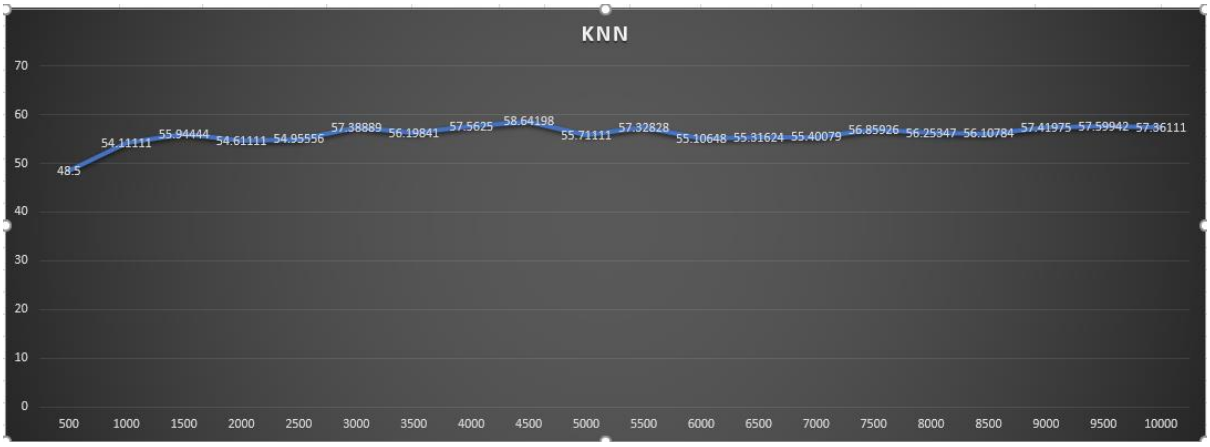
Depending on the results obtained following conclusions were made

4.5.1 Conclusion of K-Nearest Neighbor Algorithm

Sample Size	Average TFIDF Cyber	Average Vector Cyber	Average TFIDF tweet	Average Vector Tweet
500	36.67	36.22222222	66.22222222	54.8888889
1000	49.22	37.22222222	70.66666667	59.3333333
1500	49.26	39.33333333	73.11111111	62.0740741
2000	42.56	40.55555556	73.05555556	62.2777778
2500	46.49	39.33333333	69.64444444	64.3555556
3000	45.37	43.96296296	74.44444444	65.7777778
3500	45.11	41.26984127	74.31746032	64.0952381
4000	47.56	43.47222222	71.97222222	67.25
4500	48.17	44.56790123	75.30864198	66.5185185
5000	46.47	44.84444444	65.28888889	66.2444444
5500	46.91	44.92929293	68.78787879	68.6868687
6000	46.72	46.05555556	60.33333333	67.3148148
6500	47.56	48.30769231	59.79487179	65.6068376
7000	47.4	46.58730159	60.19047619	67.4285714
7500	46.83	47.67407407	66.75555556	66.1777778
8000	47.56	47.625	60.23611111	69.5972222
8500	49.41	49.9869281	57.68627451	67.3464052
9000	48.77	49.64197531	61.24691358	70.0246914
9500	49.42	52.59649123	57.83625731	70.5497076
10000	49.71	51.65555556	56.97777778	71.1



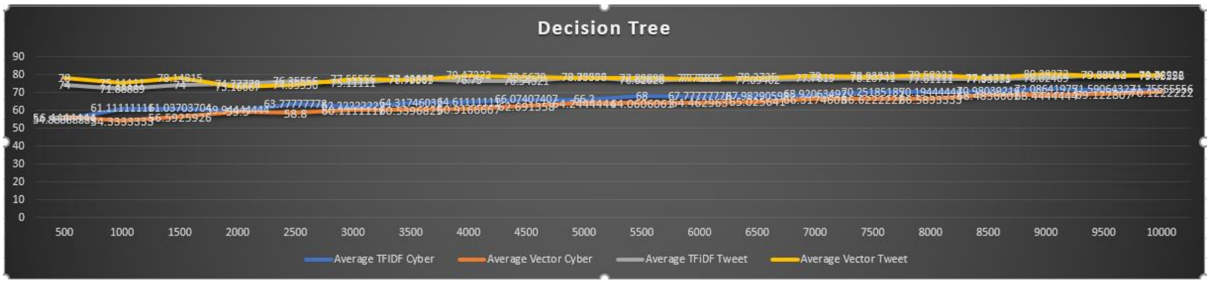
Sample Size	Average Accuracy
500	48.5
1000	54.11
1500	55.94
2000	54.61
2500	54.96
3000	57.39
3500	56.2
4000	57.56
4500	58.64
5000	55.71
5500	57.33
6000	55.11
6500	55.32
7000	55.4
7500	56.86
8000	56.25
8500	56.11
9000	57.42
9500	57.6
10000	57.36



The data set used for the experiment is the “tweeter data set” other use the “cyber troll data set”. The techniques that are applied to the data set are Vectorization and TFIDF. For each data set and technique there are three n-gram ranges [(1,1) ,(1,2), (1,3)]. The size of both data set is 10k. Training starts from 500 entries and gradually increases to 10K. Accuracy is recorded as data size increases. After which average of three n-grams is calculated for each dataset and technique which gives a total of 4 new-accuracies (frequencies) and plotted against the increasing data size on the x-axis and accuracy on the y-axis. From which it is observed that for each frequency accuracy increases rapidly until 1.5k data size after which increase becomes gradual until 4.5k. Outer-fitting is observed TFIDF in the case of both algorithms while for Count Vector overfitting is negligible. From 1.5k to 4.5k average increase in accuracy is 2%. It is also observed that after there is some overfitting from 1.5k to 4.5k. At 10k the average accuracy is 57% which is only 2% more than what it was at 1.5k. With possible underfitting. Finally, the average accuracy of a whole experiment is computed and plotted against the data set which also proved that 4.5k is the ideal data size in the case of KNN.

4.5.2 Conclusion for Decision Tree Algorithm

Sample Size	Average TFIDF Cyber	Average Vector Cyber	Average TFiDF Tweet	Average Vector Tweet
500	54.88888889	56.4444444	74	78
1000	61.11111111	54.3333333	71.88889	75.11111
1500	61.03703704	56.5925926	74	78.14815
2000	59.94444444	59.5	74.77778	73.16667
2500	63.77777778	58.8	76.35556	74.35556
3000	62.22222222	60.1111111	75.11111	77.55556
3500	64.31746032	60.5396825	77.42857	76.79365
4000	64.61111111	60.9166667	76.75	79.47222
4500	66.07407407	62.691358	76.54321	78.5679
5000	66.2	64.2444444	78.73333	78.35556
5500	68	64.0606061	76.62626	77.89899
6000	67.77777778	64.462963	77.7963	77.75926
6500	67.98290598	65.025641	77.09402	78.2735
7000	68.92063492	66.3174603	77.7619	79
7500	70.25185185	66.6222222	78.20741	78.93333
8000	70.19444444	66.5833333	77.61111	79.58333
8500	70.98039216	68.4836601	77.33333	77.94771
9000	72.08641975	68.4444444	78.02469	80.38272
9500	71.59064327	69.122807	79.63743	79.38012
10000	71.75555556	70.1222222	79.05556	79.72222



Sample Size	Total average
500	65.83333
1000	65.61111
1500	67.44444
2000	66.84722
2500	68.32222
3000	68.75
3500	69.76984
4000	70.4375
4500	70.96914
5000	71.88333
5500	71.64646
6000	71.94907
6500	72.09402
7000	73
7500	73.5037
8000	73.49306
8500	73.68627
9000	74.73457
9500	74.93275
10000	75.16389

The data set used for the experiment is the "tweeter data set" other use "cyber troll data set". The techniques that are applied to the data set are Vectorization and TFIDF. For each data set and technique there are three n-gram ranges [(1,1) ,(1,2), (1,3)]. The size of both data set is 10k. Training starts from 500 entries and gradually increases to 10K. Accuracy is recorded as data size increases. After which average of three n-gram is calculated for each dataset and technique which gives a total of 4 new-accuracies (frequencies) and plotted it against the increasing data size on the x-axis and accuracy on the y-axis. It is observed through this plot that accuracy for cyber data set to increase with an increase in data-size up-to 5.5k after which there is some outer-fitting with TFIDF and increase rate of frequency also decreases with the bare minimum rate after 7k for both TFIDF and Count vector. In the case of the tweeter dataset, the change of accuracy rate remains good up to 5.5k for TFIDF while for Vector Count there is a lot of outer-fitting. Which causes the accuracy rate to increase and decrease for Vector-Count. For the cyber dataset change rate also become a bare minimum after 7k. The total average plot also shows that change from 7k to 10 is only 2% so for the decision tree ideal dataset size could about 7k in this case.

4.5.3 Conclusion for Logistic Regression

Sample Size	Average TFIDF Cyber	Average TFiDF Tweet	Average Vector Cyber	Average Vector Tweet
500	60.8888889	71.33333	56.444444	76.44444
1000	58.5555556	78	57.555556	78.11111
1500	62.7407407	78.74074	61.62963	79.33333
2000	62	80.11111	62.5	78.72222
2500	66.9777778	80.04444	64.577778	79.46667
3000	64.5925926	82.11111	65.333333	81.14815
3500	68.2539683	82.88889	67.333333	81.36508
4000	69.2777778	82.02778	68.444444	83.16667
4500	69.1358025	83.16049	70.296296	82.96296
5000	69.8444444	84.02222	69.666667	84.08889
5500	70.2424242	83.29293	69.939394	83.63636
6000	70.3888889	83.64815	70.722222	83.72222
6500	71.6752137	84	71.059829	83.67521
7000	71.4126984	84.12698	73.142857	84.2381
7500	73.8666667	84.37037	74.311111	85.0963
8000	73.8194444	84.84722	74.833333	84.52778
8500	74.9673203	83.96078	75.594771	84.57516
9000	74.654321	85.49383	74.185185	85.7284
9500	75.3099415	85.25146	76.538012	85.08772

10000	75.8444444	85.46667	76.655556	85.63333

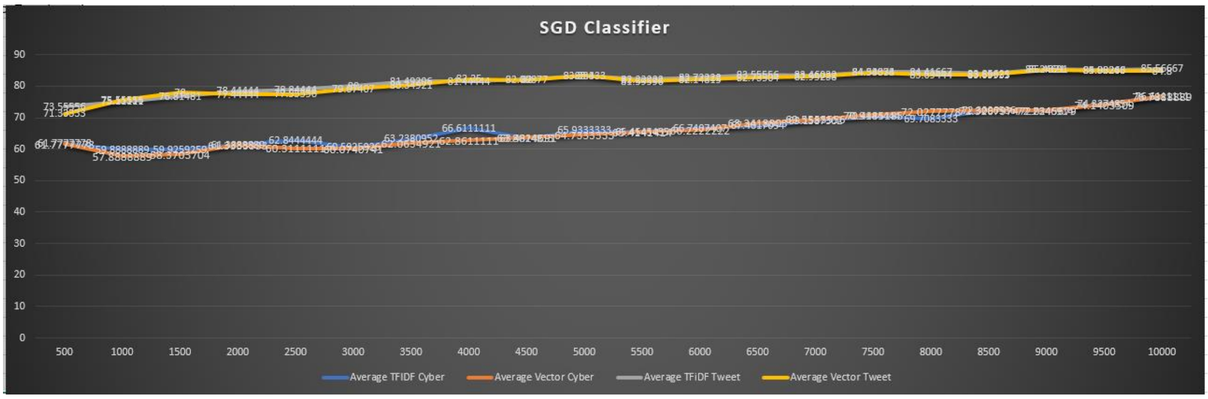
Sample Size	Total average
500	66.27778
1000	68.05556
1500	70.61111
2000	70.83333
2500	72.76667
3000	73.2963
3500	74.96032
4000	75.72917
4500	76.38889
5000	76.90556
5500	76.77778
6000	77.12037
6500	77.60256
7000	78.23016
7500	79.41111
8000	79.50694
8500	79.77451
9000	80.01543
9500	80.54678
10000	80.9

The data set used for the experiment is the "tweeter data set" other use the "cyber troll data set". The technique that is applied to the data set is Vectorization and TFIDF. For each data set and technique there are three n-gram ranges [(1,1) ,(1,2), (1,3)]. The size of both data set is 10k. Training starts from 500 entries and gradually increases to 10K. Accuracy is recorded as data size increases. After which average of three n-gram is calculated for each dataset and techniques which give a total of 4 new-accuracies (frequencies) and plotted it against the increasing data size on the x-axis and accuracy on the y-axis. It is noticed that the increase in accuracy is gradual throughout the whole dataset. In the case of the "tweeter dataset" the maximum accuracy is achieved at 6.5k that is about 84% while in the case of the cyber dataset the change in accuracy remains gradually constant with 71% at 6.5 and 75% at 10k. This is observed for both Vector-Count and TFIDF. From 6.5k to 10k average increase is 3%. So 6.5k is a desirable size. Although average frequency increases but the change rate remain extremely little.

4.5.4 Conclusion for SGD Classifier

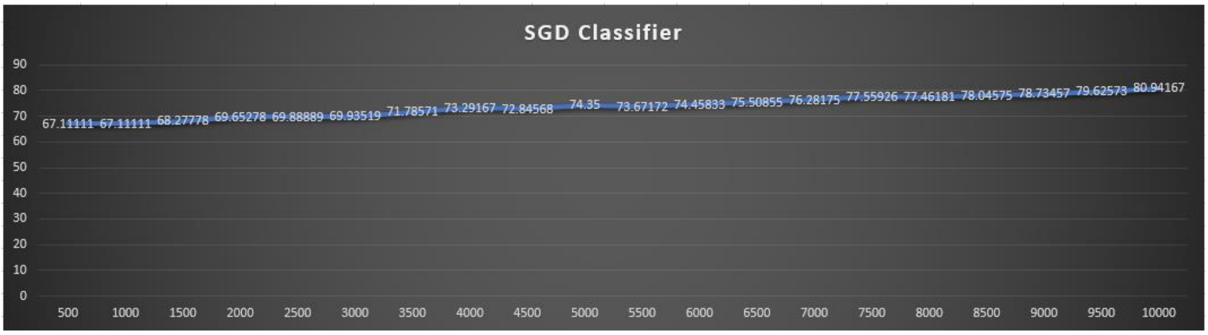
Sample Size	Average TFIDF Cyber	Average Vector Cyber	Average TFiDF Tweet	Average Vector Tweet
500	61.7777778	61.7777778	73.55556	71.33333
1000	59.8888889	57.8888889	75.11111	75.55556
1500	59.9259259	58.3703704	76.81481	78
2000	61.3888889	61.3333333	78.44444	77.44444
2500	62.8444444	60.3111111	78.84444	77.55556
3000	60.5925926	60.0740741	80	79.07407
3500	63.2380952	62.0634921	81.49206	80.34921
4000	66.6111111	62.8611111	81.44444	82.25
4500	63.4814815	63.8024691	82.09877	82

5000	65.9333333	64.7333333	83.4	83.33333
5500	65.4545455	65.4141414	82.22222	81.59596
6000	66.7407407	66.2222222	82.72222	82.14815
6500	67.4017094	68.3418803	83.55556	82.73504
7000	69.1587302	69.5555556	83.46032	82.95238
7500	70.4444444	70.9185185	84.53333	84.34074
8000	69.7083333	72.0277778	84.41667	83.69444
8500	72.3006536	72.3267974	83.85621	83.69935
9000	72.0246914	72.2345679	85.4321	85.24691
9500	74.3274854	74.1403509	85.05263	84.98246
10000	76.6111111	76.7888889	85.56667	84.8



Sample Size	**Total average**
500	67.11111
1000	67.11111
1500	68.27778
2000	69.65278
2500	69.88889
3000	69.93519
3500	71.78571
4000	73.29167
4500	72.84568
5000	74.35
5500	73.67172
6000	74.45833
6500	75.50855
7000	76.28175
7500	77.55926
8000	77.46181
8500	78.04575
9000	78.73457
9500	79.62573

10000	80.94167

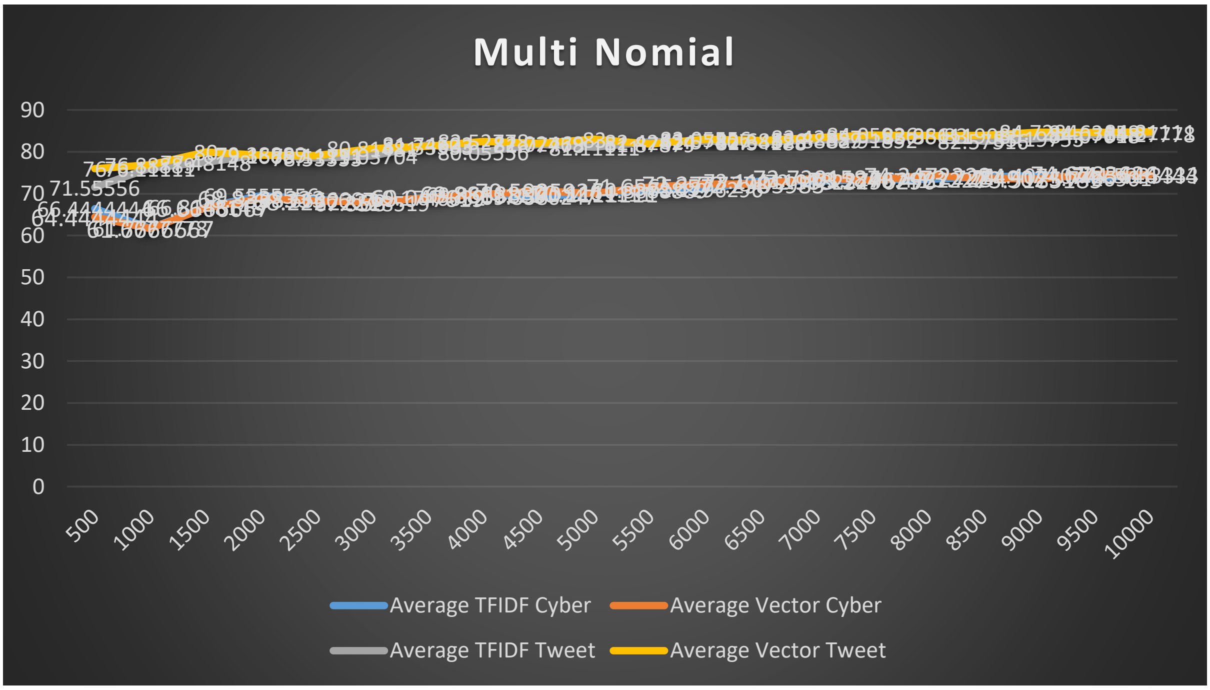


The data set used for the experiment is the "tweeter data set" other use "cyber troll data set". The techniques that are applied to the data set are Vectorization and TFIDF. For each data set and technique there are three n-gram ranges [(1,1) ,(1,2), (1,3)]. The size of both data set is 10k. Training starts from 500 entries and gradually increases to 10K. Accuracy is recorded as data size increases. After which average of three n-gram is calculated for each dataset and technique which gives a total of 4 new-accuracies (frequencies) and plotted it against the increasing data size on the x-axis and accuracy on the y-axis. From which it is observed that accuracy for tweeter database increase gradually until 6k where it achieves high accuracy of 82% from where to 10k the increase is barely minimum about 3%. For the Cyber dataset, the increase in frequency remains gradual throughout the training although it achieves the accuracy of about 65% percent at 6k. Its accuracy keeps on growing whereas it gives about accuracy of 75% at 10k. . In the case of a total average plot of 6.5k to 10k, there is a 4 % increase. So the optimum data size should be around 6k to 7k. Increasing further will only be desirable if computation time is little and the overall increase in accuracy is high.

4.5.5 Conclusion for Multinomial Naïve Bayes Algorithm

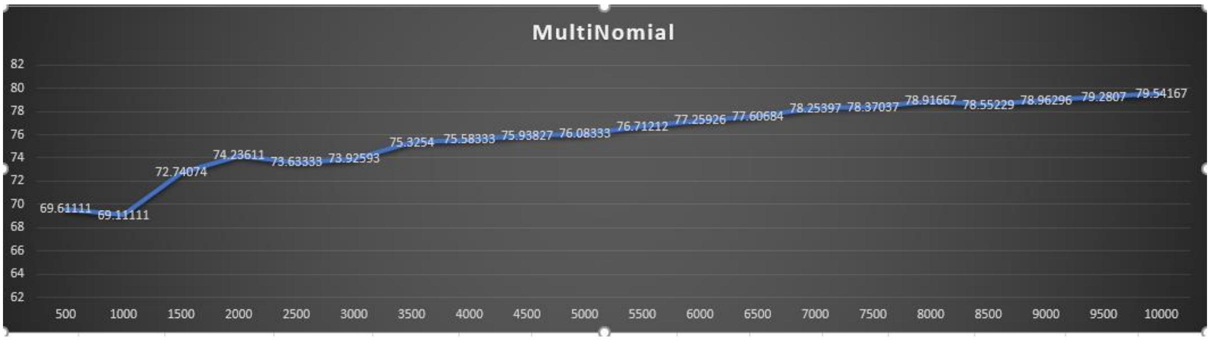
Sample Size	Average TFIDF Cyber	Average Vector Cyber	Average TFIDF Tweet	Average Vector Tweet
500	66.4444444	64.4444444	71.55556	76
1000	61.7777778	61.6666667	76.11111	76.88889
1500	66.8148148	66.6666667	77.48148	80
2000	69.5555556	68.8333333	79.38889	79.16667
2500	68.6666667	68.2222222	78.53333	79.11111
3000	67.8518519	68	79.03704	80.81481
3500	69.1746032	69.047619	81.74603	81.33333
4000	69.8611111	69.8888889	80.05556	82.52778

4500	69.3580247	70.5925926	81.77778	82.02469
5000	70.1111111	70.1111111	81.11111	83
5500	70.8888889	71.6565657	82.42424	81.87879
6000	71.1296296	72.2777778	82.57407	83.05556
6500	73.1111111	72.2905983	82.34188	82.68376
7000	72.968254	73.7301587	82.88889	83.42857
7500	73.2740741	73.2296296	82.91852	84.05926
8000	73.4722222	74.3472222	83.98611	83.86111
8500	74	73.7254902	82.57516	83.9085
9000	74.4074074	73.5185185	83.19753	84.7284
9500	73.7426901	74.6783626	84.07018	84.63158
10000	74.5333333	74.5444444	84.27778	84.81111



Sample Size	**Total average**
500	69.61111
1000	69.11111
1500	72.74074
2000	74.23611
2500	73.63333
3000	73.92593
3500	75.3254
4000	75.58333
4500	75.93827

5000	76.08333
5500	76.71212
6000	77.25926
6500	77.60684
7000	78.25397
7500	78.37037
8000	78.91667
8500	78.55229
9000	78.96296
9500	79.2807
10000	79.54167

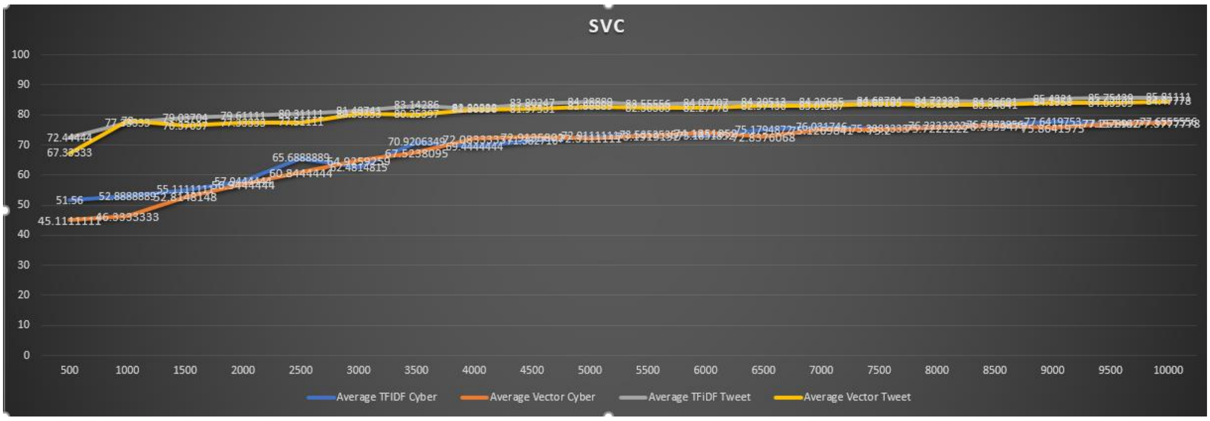


The data set used for the experiment is the "tweeter data set" other use "cyber troll data set". The techniques that are applied to the data set are Vectorization and TFIDF. For each data set and technique there are three n-gram ranges [(1,1) ,(1,2), (1,3)]. The size of both data set is 10k. Training starts from 500 entries and gradually increases to 10K. Accuracy is recorded as data size increases. After which average of three n-gram is calculated for each dataset and technique which gives a total of 4 new-accuracies (frequencies) and plotted it against the increasing data size on the x-axis and accuracy on the y-axis. In both case, TFIDF and VectorCount accuracy increase gradually for both data sets this remain about 6k after 6k change rate becomes bare minimum even with drastic data size change. The average accuracy only changes 3% from 6k to 10k. The average accuracy plot also concludes 6k should be the ideal data size.

4.5.6 Conclusion for SVC Algorithm

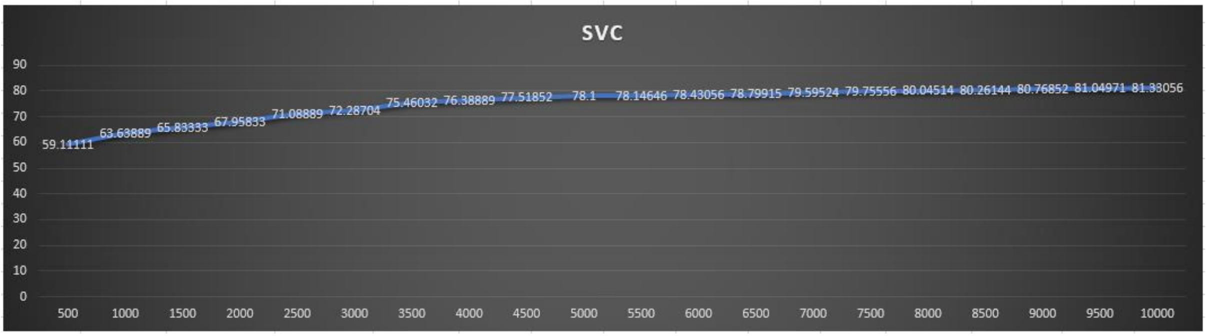
Sample Size	Average TFIDF Cyber	Average Vector Cyber	Average TFiDF Tweet	Average Vector Tweet
500	51.5555556	45.1111111	72.44444	67.33333
1000	52.8888889	46.3333333	77.33333	78

1500	55.1111111	52.8148148	79.03704	76.37037
2000	57.9444444	56.9444444	79.61111	77.33333
2500	65.6888889	60.8444444	80.31111	77.51111
3000	62.4814815	64.9259259	81.40741	80.33333
3500	70.9206349	67.5238095	83.14286	80.25397
4000	69.4444444	72.0833333	82.22222	81.80556
4500	71.382716	72.9135802	83.80247	81.97531
5000	72.9111111	72.3111111	84.28889	82.88889
5500	73.5353535	73.1919192	83.55556	82.30303
6000	73.1851852	74.1851852	84.07407	82.27778
6500	75.1794872	72.8376068	84.20513	82.97436
7000	76.031746	75.1269841	84.20635	83.01587
7500	75.3333333	75.2	84.63704	83.85185
8000	76.2222222	75.7222222	84.72222	83.51389
8500	76.7973856	76.5359477	84.36601	83.34641
9000	77.6419753	75.8641975	85.4321	84.1358
9500	77.1578947	77.251462	85.75439	84.03509
10000	77.6555556	77.3777778	85.81111	84.47778



Sample Size	Total average
500	59.11111
1000	63.63889
1500	65.83333
2000	67.95833
2500	71.08889
3000	72.28704
3500	75.46032
4000	76.38889
4500	77.51852
5000	78.1
5500	78.14646
6000	78.43056
6500	78.79915

7000	79.59524
7500	79.75556
8000	80.04514
8500	80.26144
9000	80.76852
9500	81.04971
10000	81.33056



The data set used for the experiment is the “tweeter data set” other use “cyber troll data set”. The techniques that are applied to the data set are Vectorization and TFIDF. For each data set and technique there are three n-gram ranges [(1,1) ,(1,2), (1,3)]. The size of both data set is 10k. Training starts from 500 entries and gradually increases to 10K. Accuracy is recorded as data size increases. After which average of three n-gram is calculated for each dataset and technique which gives a total of 4 new-accuracies (frequencies) and plotted it against the increasing data size on the x-axis and accuracy on the y-axis. From which it is observed that for each frequency accuracy increase gradually to 6.5k. In the case of TFIDF there an increase after 6.5k to 10k less than 2 percent in the case of both datasets. While for Count Vector this change is a bit high between 3 to 4 percent. When the plot of total average accuracy is plotted it is also seen there is only an increase of about 3 to 4 percent accuracy from 6.5k to 10k with high accuracy of 78.8% at 6.5k. So in this case ideal size will be about 6.5k.

References

[1]. A. Shenoy, "Text Classification with Extremely Small Datasets", *Medium*, 2019. [Online]. Available: https://towardsdatascience.com/text-classification-with-extremely-small-datasets-333d322caee2. [Accessed: 05- May- 2020].

[2]. F. Nadeem and M. Ostendorf, “Estimating Linguistic Complexity for Science Texts,” in Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications, New Orleans, Louisiana, 2018, pp. 45–55.

[3]. S. Klerke, S. Castilho, M. Barrett, and A. Søgaard, “Reading metrics for estimating task efficiency with MT output,” in *Proceedings of the Sixth Workshop on Cognitive Aspects of Computational Language Learning*, Lisbon, Portugal, 2015, pp. 6–13.

[4]. C. W. Dawson, “A Neural Network Approach to Software Project Effort Estimation,” vol. 16, p. 9, 1996.

[5]. O. Bisikalo and I. Bogach, “Complexity Class of Semantics-related Tasks of Text Processing,” p. 9.

[6]. A. Lorena, L. Garcia, J. Lehmann, M. Souto and T. Ho, "How Complex Is Your Classification
Problem?", *ACM Computing Surveys*, vol. 52, no. 5, pp. 1-34, 2019.

[7]. Tin Kam Ho and M. Basu, "Complexity measures of supervised classification problems", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 3, pp. 289-300, 2002.

[8]. A. Malinowski, J. Zurada, and P. Aronhime, "Minimal Training Set Size Estimation For Neural Network- based Function Approximation", University of Louisville.

[9]. M. A. Lee *et al.*, "Sensitivity of hyperspectral classification algorithms to training sample size," *2009 First Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing*, Grenoble, 2009, pp. 1-4.

[10]. S. Ostrogonac, M. Sečujski and D. Mišković, "Impact of training corpus size on the quality of different types of language models for Serbian," *2012 20th Telecommunications Forum (TELFOR),* Belgrade, 2012, pp. 720-723.

[11]. O. Abdelwahab, M. Bahgat, C. J. Lowrance and A. Elmaghraby, "Effect of training set size on SVM and Naive Bayes for Twitter sentiment analysis," *2015 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT),* Abu Dhabi, 2015, pp. 46-51. 14

[12]. J. Ding, X. Li and V. N. Gudivada, "Augmentation and evaluation of training data for deep learning," *2017 IEEE International Conference on Big Data (Big Data)*, Boston, MA, 2017, pp. 2603-2611.

[13]. Daniyal, W. Wang, M. Su, S. Lee, C. Hung and C. Chen, "A guideline to determine the training sample size when applying big data mining methods in clinical decision making," *2018 IEEE International Conference on Applied System Invention (ICASI)*, Chiba, 2018, pp. 678-681.

[14]. L. Zhang, "Improving the Efficacy of Artificial Neural Network Training by Optimizing Training Data for the Simulation and Prediction of Electroencephalogram Chaotic Patterns," *2018 IEEE 17th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC)*, Berkeley, CA, 2018, pp. 145-153.

[15]. M. Martin, B. Sciolla, M. Sdika, P. Quétin and P. Delachartre, "Segmentation of neonates cerebral ventricles with 2D CNN in 3D US data: suitable training-set size and data augmentation strategies," *2019 IEEE International Ultrasonics Symposium (IUS)*, Glasgow, United Kingdom, 2019, pp. 2122-2125.

[16]. Beleites, Claudia, Ute Neugebauer, Thomas W Bocklitz, Christoph Krafft and Juergen Popp. “Sample size planning for classification models.” *Analytica chimica acta* 760 (2013): 25-33

[17]. "TfidfVectorizer – From Data to Decisions", From Data to Decisions, 2020. [Online]. Available: https://iksinc.online/tag/tfidfvectorizer/. [Accessed: 04- Mar-2020]..

[18]. Rout, Neelam. (2018). Handling Imbalanced Data: A Survey.

[19]. Mahendru, K., 2019. How To Deal With Imbalanced Data Using SMOTE. [online] Medium. Available at: <https://medium.com/analytics-vidhya/balance-your-data-using-smote-98e4d79fcddb#:~:text=Find%20its%20k%20nearest%20neighbors,steps%20until%20data%20is%20balanced> [Accessed 11 April 2020].

[20]. A. Chakure, "Decision Tree Classification", Medium, 2019. [Online]. Available: https://towardsdatascience.com/decision-tree-classification-de64fc4d5aac. [Accessed: 17- Apr- 2020]

[21]. [3]"Naive Bayes text classification", *Nlp.stanford.edu*, 2020. [Online]. Available: https://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html. [Accessed: 23- March- 2020]..