Machine Learning for Twitter Sentiment Classification on Imbalanced Data

Marjana Islam, ID: 011 151 213
Mushtari Maknun, ID: 011 151 212
Nusyba Binta Jahan, ID: 011 151 216
Saba Anowar Moulee, ID: 011 151 238
Mohammad Mirazur Rahman, ID: 011 151 110

Department of Computer Science and Engineering
United International University

A Capstone Report submitted for the degree of
BSc in Computer Science & Engineering
May 2019

Abstract

Tweets are small texts that convey the instant reaction of the Twitter users. As Twitter is a very dynamic and public social media, tweets can be an enormous source of information. In this paper,we have compared a number of established classifiers and existing lexical analysis resources on a imbalanced supervised dataset for classifying the sentiment of tweets acquired from Twitter. We have tried to find the best algorithm which works for class imbalance problem without handling it through additional measures. We have applied a combination of algorithms and word embedding techniques to elevate the results of the classification which includes Ensemble Classifications, Logistic Regressions and Neural Networks. Experimental results show varied differences among the classifiers.

Acknowledgements

Firstly we would like to thank our parents and siblings. We would also like to express our gratitude to Dr. Dewan Md. Farid, Associate Professor, United International University and Abu Shafin Mohammad Mahdee Jameel, Assistant Professor, United International University for their continuous support on the report and the whole project. We would specially thank Md. Saddam Hossain Mukta, Assistant Professor of United International University for his support when we were stuck with Tweet pre-processing. Last but not the least, we would like to thank Sajid Ahmed, Lecturer of United International University for his contribution in the algorithm and coding part of our project.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Project Overview

1.1 Introduction

In this world of Digitization, social media is a part of everyone's daily life. Among them, one of the most popular public social networking platform based on active users is Twitter. Twitter is where users write micro-posts (known to as tweets, limited to 280 characters) to share their thoughts on anything and everything. Thus this particular social network has become one of the most interesting resources for a variety of applications, ranging from presidential elections. to the stock market to product marketing.

The constantly increasing interaction is generating massive datasets that document collective behavior. In this huge repository, we can find concerns, interests and intentions of the global community with respect to varied national, political, economic and cultural phenomena. When a number of tweets are acquired in a real life scenario, it is to be observed that the average sentiment of those tweets are not equally distributed. Which means real life twitter data consists of class imbalance.

Hence in this project, to keep it as authentic as possible to real life examples, we worked with dataset with class imbalance problem. We used traditional machine learning algorithms to build classification models which would have unknown tweets as input and predict is class label i.e. their underlying sentiment. In our project we are working with a dataset that consists of 13 different class labels, which translates that our classifiers

have to learn to classify 13 different emotions from an imbalanced dataset.

1.2 Purpose of the Project

Tweets convey the state of minds from a large population on earth. This collection of data provides a precious source of information. Using machine learning to extract public sentiment from public web-based micro-blogging sites like Twitter can work for various research and marketing purposes. Although real life twitter data can be hugely varied and disproportionately distributed, machine learning algorithms may work past it to achieve good results.

1.3 Problem Statement

In this project, we perform a comparative study on different established machine learning algorithms on their ability to classify 13 different human emotion from an imbalanced collection of tweets accumulated from Twitter. Our main target is to distinguish a model which can analyze a given tweet and classify the sentiment behind it from this imbalanced dataset without any synthetic manipulation to the data.

1.4 Organization of the Project

The materials presented in this report is divided into 10 chapters including Introduction. The Project Report is organized as below.

- **Chapter 2**: We discuss about the works, papers and projects related to our project.
- **Chapter 3**: In this chapter we give an overview of the general Algorithms and supervised learning.
- **Chapter 4**: We discuss about the general flowchart of how we will implement our project

- **Chapter 5**: It discusses the Methodology that we followed for our project. It includes discussions about the the algorithms that we implemented and their parameters.
- **Chapter 6**: We have discussed the in-depth results of our algorithms including the parameters used.
- **Chapter 7**: In this chapter, we have discussed about the specifications and engineering decisions made for our project.
- **Chapter 8**: The standards that we have followed throughout the project is mentioned in this chapter.
- **Chapter 9**: We discussed the different impacts and constraints that our project will have.
- **Chapter 10**: This chapter includes an in-depth discussion, summary and the limitations of our project.

Chapter 2

Related Work

2.1 Introduction

We have so far read a few conference papers, journals and project reports. From the papers, we have gathered many information relevant to our project. The papers collected will undergo preliminary elimination to remove backdated information. We have also found few published websites that matches to our project.

2.2 Research and publications

2.2.1 Basic Research

From our thorough scanning of the papers, we have seen how researchers scrutinized linguistic feature's efficacy for tweeter sentiment detection [1]. They took a supervised approach to the problem and use tweeter hashtag to build the dataset. We also found a paper where the authors built a model for classifying tweets into positive and negative with respect to a query term using Natural Language Processing [2][3]. However, we tend to conjointly found a paper where the opinion words in the tweets are determined as semantic orientation.

Another research worker instigated POS specific prior polarity features and explored the utilization of Tree Kernel for sentiment analysis [3]. The features for their dataset is based on prior polarity of words which they calculated using DAL and WordNet. Words

with polarity smaller than 0.5 are considered as negative, greater than 0.8 as positive and all in between as neutral. They have designed a tree representation of tweets and use PT for comparing all possible subtrees.

We found a paper where the researchers used decision trees to classify Twitter Sentiment[4]. They used subject related tweets for their dataset. For feature generation, to reduce the dimension of the dataset, they used sentiment words as features. It was conjointly noticed that the utilization of emoticons as features failed to improve performance rather had a small negative impact. We have been manually collecting Twitter corpus/data to train our model. However we stumbled upon a paper where the researchers conferred a technique of automatic assortment of a corpus that may be used for training a sentiment classifier[5]. Then they analyzed the collected corpus based on linguistic feature.

During the training phase of the classifier, they have experimented with unigrams, bigrams and trigrams. High-order n-grams should better capture patterns of sentiment but low-order n-grams give better data coverage. They tried both SVM and multinomial Nave Bayes Classifier. The Nave Bayes classifier yielded the best results. Also, the bigrams achieved the best results. We found a paper where the researchers have advocated to enhance the target-dependent sentiment classification of tweets by employing both target-dependent and context-aware approaches [6]. Specially the target-dependent approach refers to incorporating syntactical features generated using words syntactically connected with the given target within the tweet. They have additionally projected to take the associated tweets of the present tweet into deliberation. They have built a SVM classifier to perform subjectivity classification and polarity classification. For labeling tweets, they have used Graph-based optimization which can further boost up their performance.

In a paper published in Springer Vienna [7], they have evolved a technique to recognize homophilies by analyzing the Big 5 personality characteristic (Openness, Conscientiousness, Extraversion, Agreeableness & Neuroticism) of users from their interactions in an egocentric network. The strength of the models was differentiated by applying classification and regression prediction potential techniques. We additionally reviewed a paper[8]

where the researchers delineated disparate text mining approach to find numerous textual patterns wherever information is unstructured. They pre-processed data using two basic methods:(a) Feature Selection and (b) Feature Extraction. Classification can be categorized into two parts: (a) Machine Learning-based Text Classification (MLTC) and (b) Ontology-based Text Classification.

Another researcher constructed two state-of-the-art SVM classifiers that detects - 1) sentiment of tweets (message-level task) and 2) sentiment of a term enclosed by that tweet (term-level task)[9]. There are three possible outcomes: positive, negative, or neutral. They have enforced a range of features based on surface-form, semantic and sentiment features. Furthermore, two large wordsentiment association lexicons were generated by them - one from tweets with sentiment-word hashtags, and another from tweets with emoticons.

In a paper published in the AAAI conference[1], the researchers appraised the benefit of subsisting lexical resources including features that encapsulates information about the casual and artistic language used in micro-blogging. They have applied a supervised approach to solve the problem. Three distinct corpora of Twitter messages were implemented in their experiments. For development and training, they use a) Hash-tagged dataset(HASH) b) Emoticon data set(EMOT) and for evaluation c) (ISIEVE) dataset. They classified them into positive, negative, and neutral sentiments. Data preprocessing included three steps - 1) Tokenization 2) Normalization and 3) Part-Of-Speech (POS) tagging. They have also used a) n-gram features for the baseline b) Lexicon features c) Part-Of-Speech features and d) Micro-blogging features for sentiment analysis.

In another recent paper[10], the researchers proposed learning sentiment specific word embedding (SSWE) for the investigation of sentiment. They have encrypted the sentiment data into the continual illustration of words in order to separate good and bad to the different ends of the spectrum. They have trained 3 neural networks models (SSWEh, SSWEr, SSWEu) by using the derivative of the loss through back-propagation with regard to the full set of parameters and update the parameters using Ada-Grad. Sentiment-specific word embedding (SSWEh, SSWEr, SSWEu) outperform existing neu-

ral models (C& W, word2vec) by large margins. SSWEu performs best in three lexicons.

In a paper from Stanford [2], the researchers presented the results of machine learning algorithms by applying distant supervision. They have shown that machine learning algorithms like Naive Bayes, Maximum Entropy, and SVM have accuracy above 80% when trained with emoticon data. The machine learning classifiers that have been used are a) Naive Bayes, b) Maximum Entropy (MaxEnt), and c) Support Vector Machines (SVM). The methods used for feature extraction are a) unigrams, b) bigrams, c) unigrams and bigrams, and d) unigrams with part-of-speech tags.

Another researcher have proposed a context sensitive neural network model for sentiment inspection [11], amalgamating contextualized features from pertinent Tweets into the model within the form of word embedding vectors. It is troublesome to establish the sentiment polarity of some tweets because of the unspecified words. This persuades a context sensitive neural network model for classification of Twitter sentiment. By applying such context-based sub network, their model can boost sentiment classification performance from 80.68% to 91.33% in macro-F score.

In another paper published in the AAAI conference [12], to extract six mood states (tension, depression, anger, vigor, fatigue, confusion) from the agglomerated Twitter content, the researchers have used a psychometric instrument and computed a six-dimensional mood vector for individual days in the timeline. They have evolved and used an extended version of a well known psychometric instrument, the Profile of Mood States (POMS). POMS measures six individual sentiments - Tension, Depression, Anger, Vigour, Fatigue, and Confusion.

In a paper published in 2010[13], the researchers have merged boosting and active learning algorithm for classifying multi-class genomic data to improve the prediction accuracy of DNA variants of Brugada syndrome (BrS). Nave Bayes Classifier and clustering have been used to find the most informative instances for labeling. Boosting algorithm has also been used as a base classifier to train the model. We reviewed a paper where the researchers have proposed two bagging based methods: 1) ADASYNBagging, and 2)

RSYNBagging for dealing with imbalanced classification by employing bagging and sampling techniques[14]. They have compared the performance of proposed hybrid methods with UnderBagging and SMOTEBagging methods on 11 imbalanced benchmark datasets. Initial results suggest that their proposed methods are competent in the task of correctly classifying instances of both majority and minority classes of imbalanced datasets.

Our dataset is heavily imbalanced. So we took a look at one of the paper where the researchers have reviewed the state of the art on ensemble techniques to address a two-class imbalanced data-sets problem and to propose a taxonomy that defines a general framework within each algorithm can be placed[15]. They have carried out a thorough empirical comparison of the performance of ensemble approaches with a twofold objective. The first one is to analyze which one offers the best behavior among them. The second one is to observe the suitability of increasing classifiers complexity with the use of ensembles instead of the consideration of a unique stage of data preprocessing and training a single classifier.

In another paper[16], the researchers have summarized some of the LIWC dimensions that reflect language correlates of attentional focus, emotional state, social relationships, thinking styles, and individual differences. The researchers have also showed their effort to develop LIWC along with some of the basic psychometrics of words. By using these new text analysis methods, it can also be expanded to capture cultural differences mirrored in language use.It is about reviewing several computerized text analysis methods and describing how Linguistic Inquiry and Word Count (LIWC) was created and validated.

2.2.2 Significant Advances

Now a days, the research area in the domain of opinion mining from micro-blogging data is getting extended. We have listed few of the significant advances made in this field below -

Paper Name	**Findings**
Robust Sentiment Detection of Twitter from Biased and Noisy Data [17]	Proposed two step sentiment classification technique for Twitter, which first classifies the tweets as subjectives and then as objectives. It then further distinguishes the subjective tweets as positive or negative.
NRC-Canada: Building the State of the ART in Sentiment Analysis of Tweets [9]	Constructed two state-of-the-art SVM classifiers that detects - 1) sentiment of tweets (message-level task) and 2) sentiment of a term enclosed by that tweet (term-level task). Here, two large wordsentiment association lexicons were generated - one from tweets with sentiment-word hashtags, and another from tweets with emoticons.
Context-Sensitive Twitter Sentiment Classification Using Neural Network [11]	Proposed context sensitive neural network model for sentiment inspection [11], amalgamating contextualized features from pertinent Tweets into the model within the form of word embedding vectors. They also re-implemented SVM and SSWE algorithm in the formation of their model.

Continued on next page

Paper Name	Findings
Twitter as a Corpus for Sentiment Analysis and Opinion Mining [5]	During the training phase of the classifier, they have experimented with unigrams, bigrams and trigrams. High-order n-grams should better capture patterns of sentiment but low-order n-grams give better data coverage. They tried both SVM and multinomial Nave Bayes Classifier. The Nave Bayes classifier yielded the best results. Also, the bigrams achieved the best results.
Target-dependent Twitter Sentiment Classification [6]	They have built a SVM classifier to perform subjectivity classification and polarity classification. For labeling tweets, they have used Graph-based optimization which can further boost up their performance.
Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification [10]	They have trained 3 neural networks models (SSWEh, SSWEr, SSWEu) by using the derivative of the loss through back-propagation with regard to the full set of parameters and update the parameters using Ada-Grad. Sentiment-specific word embedding (SSWEh, SSWEr, SSWEu) outperform existing neural models (C&W, word2vec) by large margins. SSWEu performs best in three lexicons.

Table 2.1: Table showing the recent paper findings.

2.2.3 Current Status

We have found few programs and websites that is related to our project-

- **www.tweetanalyzer.net** - This is a website that takes the Twitter username and collects all the Tweets of that specific user and classifies them. It provides an overall 'mood percentage' of the user based on the Tweets. The disadvantage of this website is that it cannot classify any individual tweets. It simply gives us the 'Most Positive', 'Least Positive', 'Most Vulgar' and 'Favorite Emoji'. It also needs a minimum of 10 tweets to work. The algorithm or the methods it deploys is unknown as the developer have not made that public.

- **Sentigem** - This is also a website that simply analyses sentiments of a given text and then classifies them to positive, negative and neutral sentiments.

- **Sentiment Viz** - This is a visualization website that sends a query to the Twitter server and retrieves the tweets and then visualized in the graph. This website also provides more functions like timeline, heat maps, map and finally tweets that contain the queried word.

2.2.4 Dataset

Our mentor have provided us with a dataset that he had acquired before. The dataset was in an text file. It had a Tweet ID and Emotion in a line. Both the data was tab-separated. The dataset has over 2000 data.

TweetID	Emotion
549283862572843008	anxiety
549284051434369024	neutral
549283925278085120	sadness
549283854897664001	enthusiasm
549283942021746690	joy
549283874900287489	sadness
549283954587881472	neutral
549284026255945728	sadness
549283997122310144	calm
549284018114420736	enthusiasm

Figure 2.1: Snippets of the initial dataset that we have acquired.

We will have to modify the dataset. We will convert the text file to an excel file and add another column containing the actual Tweet. We will also collect more Tweets and add them to the datasets to enrich with updated words and emotions.

	A	B	C
1	TWEET ID	TWEET	EMOTION
2	549283862572843008		anxiety
3	549284051434369024	@nicktheguitar apparently gin or juniper berries were used to induce miscarriage way back when, but can't find any convincing evidence.	neutral
4	549283925278085120		sadness
5	549283854897664001	Well you're good and I'd come see you guys if you come round San Fr	enthusiasm
6	549283942021746690	The guys want to wish you a very happy, relaxed boxing day - full of food and fun!!	joy
7	549283874900287489	R.I.P. Geno Smith	sadness
8	549283954587881472	Need a TD to finish the half and get ball start of second #Chiefs	neutral
9	549284026255945728		sadness
10	549283997122310144		calm
11	549284018114420736	#Kids' 1st line of code? ScriptCraft is a #Minecraft Mod (server plugin) that lets you extend it using Javascript ~ http://scriptcraftjs.org	enthusiasm
12	549283879342059520		neutral
13	549283963181625344	I am proud to be a Muslimah, and you?	enthusiasm
14	549283971532472321	This Man's Story is Incredible, Inspiring, & Includes Running Away from a Wolverine http://bit.ly/16Ykktz	enthusiasm
15	549283891853672448		enthusiasm
16	549284055095971840	Alice Cooper, Keith Moon, Harry Nilsson, and Marc Bolan convene in 1972.	interestingness
17	549283851663446016	Check out this item in my Etsy shop https://www.etsy.com/listing/200082306/native-american-inspired-decorated ...	neutral
18	549283954722099200	#PatriotsTalk Halftime live, click on NOW! @CSNNE.com w @MikeGiardi & friends.	enthusiasm
19	549284018018324480	@NilaGrier happy birthday to your mommy!!! I hope you spend an incredible dayy	enthusiasm

Figure 2.2: Snippets of the dataset after retrieving Tweets.

Currently the data set have 13 class labels and around 1210 instances as many of the Twitter tweets were either deleted or the account have been suspended. The following figure shows the distribution of the class labels among the instances.

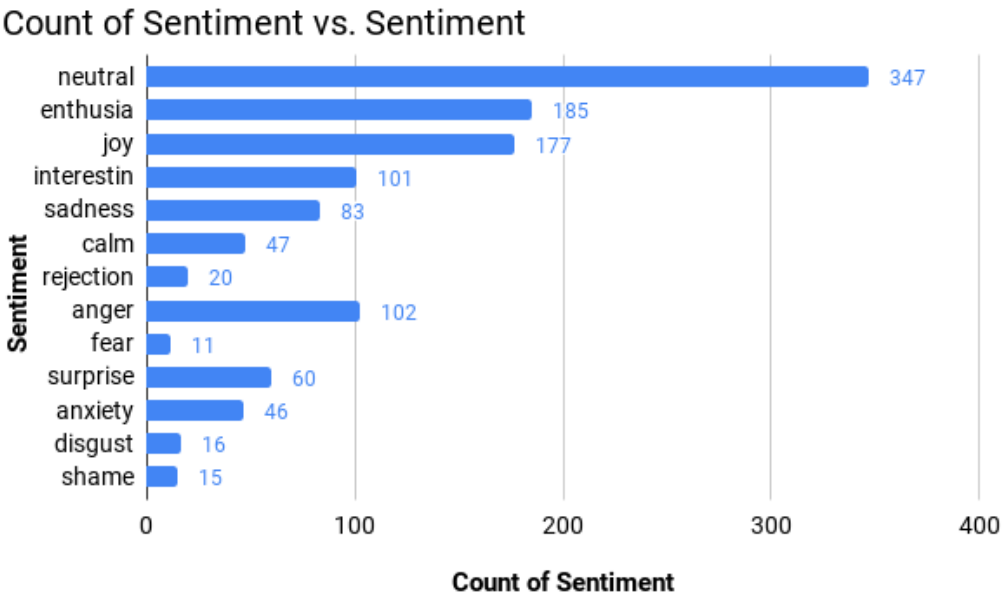


Figure 2.3: Dataset Class Label Distribution.

As you can see from the graph, our dataset is hugely imbalanced with a maximum of 350 in a class and with a minimum of 11 instances in another class.

2.3 Summary

To summarize the research papers, we have found out that the research seems to be grouped according to the following-

- Use of prior polarity of words.
- Use of neutral data for both training and testing data.
- Use of emoticons for both training and test data-sets.
- Use of n-grams.
- Count of word frequency.
- Use of neural network.
- Classifiers used till now -

Classifiers	Naive Bayes
	Decision Tree
	Maximum Entropy
	Support Vector Machine

Chapter 3

Supervised Learning

3.1 Introduction

Our main target is to build a model which can analyze a given tweet and classify the sentiment behind it. In this chapter we are going to discuss the models, terms and algorithms that we are going to come across while completing this project.

3.2 Text Mining

Text mining is the procedure of acquiring data from text. It is usually acquired through the formulation of trends and patterns by means of pattern learning. Text mining typically calls for the process of arranging the input text, extracting patterns from the arranged data, and eventually evaluating and decoding the output.

Usual text mining task include text classification, text clustering, concept/entity extraction, sentiment analysis, etc.

For our sentiment analysis, we will need the help of both text classification & text clustering.

3.3 Classification

Classification is a data mining function. It represents and differentiates data categories. The objective of classification is to precisely predict class labels / values of instances whose attribute values are known, but class labels are unknown. It is a kind of data investigation that extracts models / classifier representing important data classes. It is often referenced as supervised learning because the classes are predetermined before examining the data. It is a two-step process:

- **Learning stage (Training phase)** - A classification model is constructed. The training data that comprises of instances and their associated class labels are analyzed and any classification algorithm constructs the classifier.
- **Classification stage** - The constructed classification model is used to forecast class labels for the provided test data.

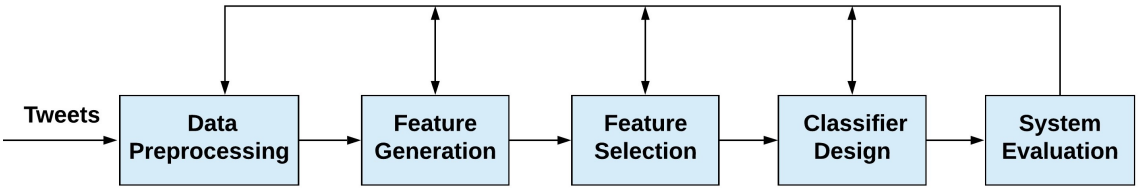


Figure 3.1: Overview of the Design Of a Classification System.

For the purpose of our project, we will be considering few famous machine learning algorithms like Random Forest, Bagging, Boosting, Logistic Regression and Neural Networks. We will look in details about these algorithms.

3.3.1 Random Forest

The Random Forest Classifier (RF) is able to classify large amounts of data with accuracy. It constructs a number of decision trees based on randomly selected subset of features at training time and outputting the class that is the ensemble of trees vote for the most popular class.

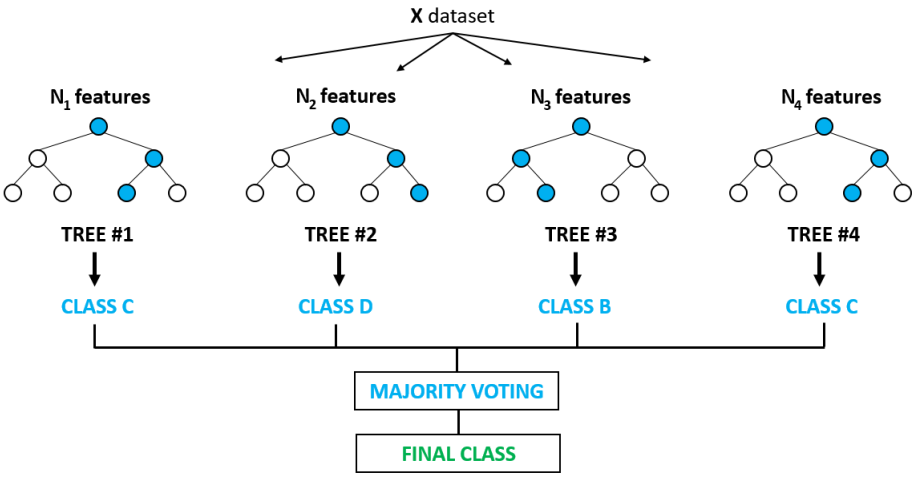


Figure 3.2: A representation of how the Random Forest Algorithm works.

3.3.2 Bagging

Bagging consists in training different classifiers with bootstrapped replicas of the original training data-set. Hence, diversity is obtained with the re-sampling procedure by the usage of different data subsets. A majority or weighted vote is used to detect the class. Bagging is usually known to work for class imbalance data as its samples the original data into smaller chunks. The algorithm below shows the pseudocode for Bagging.

Algorithm 1 Bagging Algorithm

1: **Input**: Training data, D, number of iterations, k, and a learning scheme.
2: **Output**: Ensemble model, $M*$
3: **procedure** BAGGING
4: **for** $i = 1$ to k **do**
5: create bootstrap sample D_i, by sampling D with replacement;
6: use D_i, and learning scheme to derive a model, M_i;
7: **end for**
8: **To use** $M*$ **to classify a new instance,** X_{New}:
9: Each $M_i \in M*$ classify X_{NEW} and return the majority vote;
10: **end procedure**

3.3.3 AdaBoost

AdaBoost gives more focus to examples that are harder to classify. The quantity of focus is measured by a weight, which initially is equal for all instances. After each iteration, the weights of misclassified instances are increased; the weights of correctly classified instances are decreased. Furthermore, another weight is assigned to each individual classifier depending on its overall accuracy. A majority or weighted vote is used to detect the class. Algorithm 2 shows the pseudocode for AdaBoost.

Algorithm 2 AdaBoost Algorithm

1: **Input**: Training data, D, number of iterations, k, and a learning scheme.
2: **Output**: Ensemble model, $M*$
3: **procedure** AdaBoost
4: Initialise weight, $x_i \in D$ to $\frac{1}{d}$
5: **for** $i = 1$ to k **do**
6: sample D with replacement according to instance weight to obtain D_i;
7: use D_i, and learning scheme to derive a model, M_i;
8: compute ERROR (M_i);
9: **if** ERROR $(M_i) \geq 0.5$ **then**
10: go back to step 6 and try again;
11: **end if**
12: **for** each correctly classified $x_i \in D$ **do**
13: multiply weight of x_i by $\frac{error(M_i)}{1-error(M_i)}$
14: **end for**
15: normalise weight of instances;
16: **end for**
17: **end procedure**
18: **To use $M*$ to classify a new instance, X_{New}:**
19: **procedure** Classify
20: **for** $i = 1$ to n **do**
21: $W_i = \log \frac{1-error(M_i)}{error(M_i)}$; // weight of the classifier's vote
22: $c = M_i(X_{New})$; //class prediction by M_i
23: add w_i to weight for class c;
24: **end for**class with largest weight;
25: **end procedure**

3.3.4 Logistic Regression

Logistic regression (LR) is a kind of predictive analysis. We have used Multinominal Logistic Regression (LR) as it is a multiclass problem. It is also known as Softmax Regression. Multinomial LR classifiers are commonly used as an alternative to naive Bayes classifiers. This is because they do not assume statistical independence of the random variables / features that serve as predictors.

3.3.5 Neural Network

Neural Network is composed of a large number of highly interconnected processing elements (neurons) working simultaneously to solve specific problems. Neural networks can be used for both regression and classification, and they can be seen as an extension of logistic regression. Neural Networks learn by examples and also generates features. Once properly trained, It has the ability derive meaning from complicated or imprecise data. It contains 3 layers - 1)Input layer, 2) Hidden Layer and 3) Output layer.

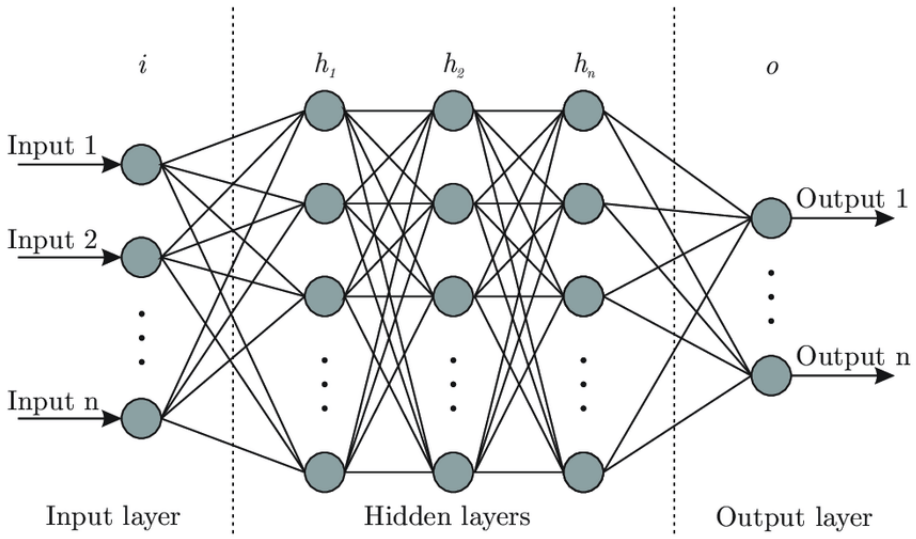


Figure 3.3: A representation of a Neural Network model.

3.3.5.1 Convolutional Neural Network

Convolutional Neural Networks (CNN) is one of the main categories of neural networks as it can automatically detect its important features without any human supervision. CNN also uses special convolution operations and can also execute parameter sharing. CNN can be thought of a combination of two components : feature extraction part and classification part. The convolution + pooling layer is the feature extraction part while the fully connected part can be referred as the classification part. Convolutional layer is the main block of CNN. It is a mathematical operation which merges two sets of information. For applying CNN parameters are chosen, filters with strides are applied and pooling is added if requires.

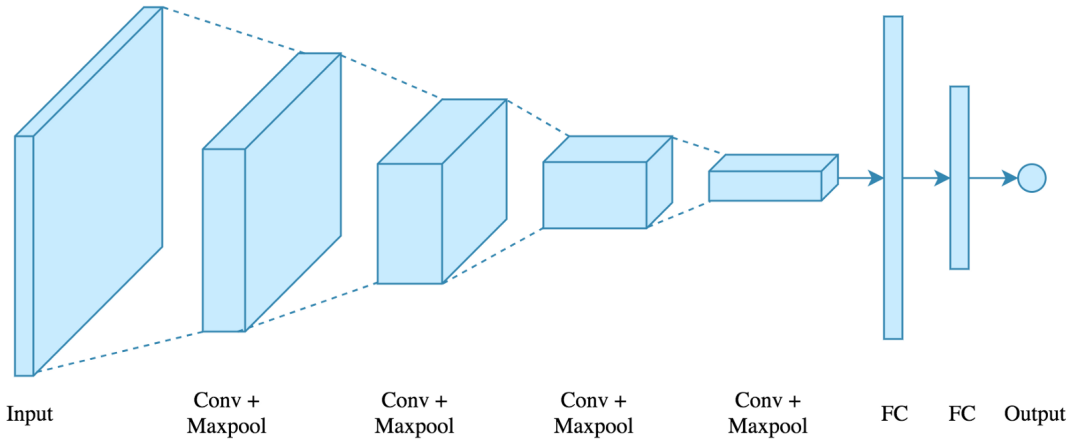


Figure 3.4: A representation of a Convolutional Neural Network model.

3.3.5.2 Recurrent Neural Network

Recurrent Neural Network (RNN) is a type of Neural Network that is used for classifying sequential data where the output from previous step are fed as input to the current step. RNN have a memory which remembers all information about what has been calculated. It uses the same parameters for each input as it performs the same task on all the inputs

or hidden layers to produce the output. This reduces the complexity of parameters, unlike other neural networks.

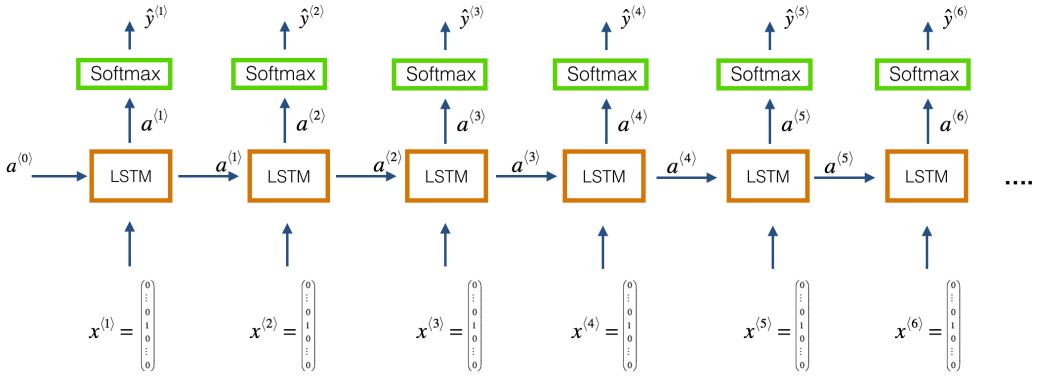


Figure 3.5: A representation of how Recurrent Neural Network works.

3.3.6 Evaluating Classifier Performance

The performance measures of a classifier t ells us how accurate the classifier was to predict the class label of the instances. The following table shows us the terms that needs to be known before evaluating performances.

Term	Positively Classified	Negatively Classified	Positive Instances	Negative Instances
True Positive (TP)	✓		✓	
True Negative (TN)		✓		✓
False Positive (FP)		✓	✓	
False Negative (FN)	✓			✓

Table 3.1: Table showing the terms for classifier performance evaluation.

The classification accuracy of a classifier on a given test set is the percentage of test set instances that are correctly classified by the classifier. This is also referred to as the overall recognition rate of the classifier, that is, it reflects how well the classifier recognises instances of the various classes. The classification accuracy is measured by

either of the following equations -

$$Accuracy = \frac{TP + TN}{P + N} \tag{3.1}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.2}$$

Where P = Total number of positive instance. N = Total number of negative instance. The "error rate" or "misclassification rate" of a classifier on a given test set is the percentage of test set instances that are misclassified by the classifier, which is also known as the "resubstitution error".

$$ErrorRate = \frac{FP + FN}{P + N} \tag{3.3}$$

The sensitivity is referred to as the true positive (recognition) rate (i.e., the proportion of positive instances that are correctly identified)

$$Sensitivity = \frac{TP}{P} \tag{3.4}$$

The specificity is referred to as the true negative (recognition) rate (i.e., the proportion of negative instances that are correctly identified)

$$Specificity = \frac{TN}{N} \tag{3.5}$$

Precision can be thought of as a measure of exactness (i.e., what percentage of instances labeled as positive are actually such)

$$Precision = \frac{TP}{TP + FP} \tag{3.6}$$

Recall is the fraction of relevant instances that has been retrieved over the total amount of relevant instances.

$$Recall = \frac{TP}{TP + FN} \tag{3.7}$$

3.4 Summary

The model, terms and algorithms used to analyze a given tweet and classify the sentiment behind it are given below:

- Text mining for acquiring data/pattern from dataset.
- Classification function to precisely predict class labels.
- Machine learning algorithms like Random forest, Bagging, Boosting, Logistic Regression for system evolution.
- Neural Networks such as RNN, CNN for system evolution.
- Accuracy, Error rate, Sensitivity, Specificity, Precision & Recall to measure the performance of a classifier.

Chapter 4

Structure of the System

4.1 Introduction

With the ever expanding escalation in online social networking, text mining and social analytics are trending issues in forecasting analytics. The typical approach to learning a text classifier is to transform unstructured text documents into a structure known as the bag-of-words representation. And finally applying any conventional propositional learning strategy to the result.

Imperatively, this means dividing tweets/texts into their constituent words, thus constructing a group of corpus. Then each tweet/text are transformed into definite features with counts of word frequency. Following this, a classifier can be learned. Assessment of the classifier is executed by interleaved testing and training. That is, a projection is assembled for each incoming instance before it is integrated into the model.

4.2 Workflow and Diagram

The machine learning based text classifiers are generally supervised learning technique. The model has to be trained on some labeled/characterized training data before it can classify any actual test data. The training data is the informative chunks of the full original data that have been labelled manually. After satisfactory training, the classifier can then be used to classify new unlabeled data.

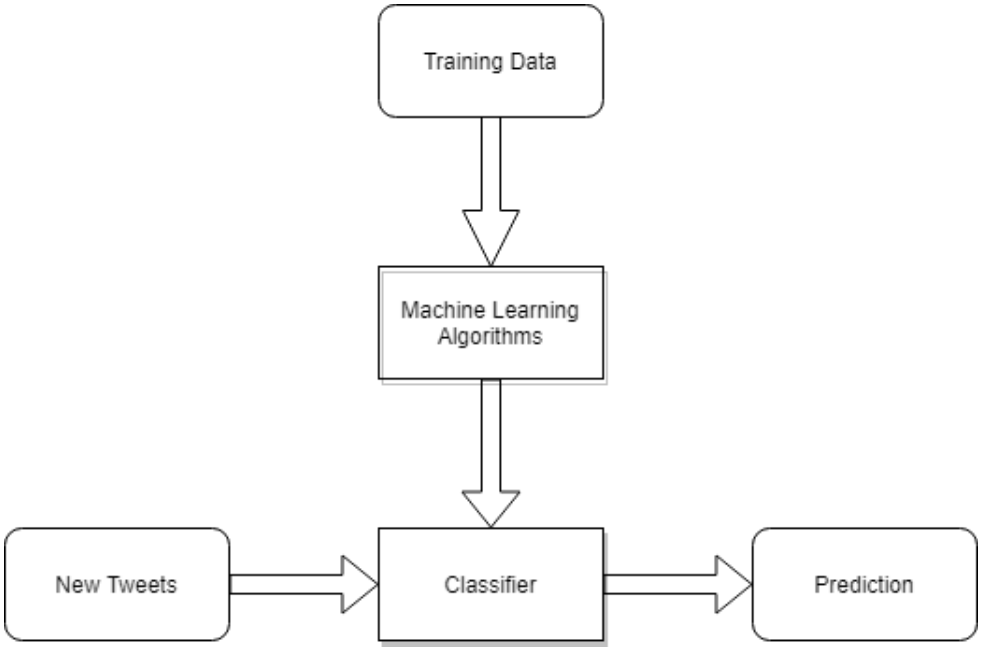


Figure 4.1: Diagram showing the flow of the general system.

4.3 Procedure

Data Preprocessing:

Data preprocessing is the integral part of our project. It is also the first and foremost task in hand. Doing data pre processing we will get the clear visual of our dataset. We needed to delete duplicate tweets to eliminate over-fitting problem. we also removed conflicting tweets i.e. same tweets having different emotion to have a more accurate model.

Feature generation:

We are working with tweets and those are basically short text posts. From those texts we have to train the model to classify the underlying emotions. Generally the classification problems are viewed as a dataset with features as columns and instances as rows. We are required to process the texts such that each text is broken in to a number of features which can be viewed as columns in a dataset. So removing subjective words and stop words was part of this section.

Feature selection:
Not every word in a sentence bear emotional value that would help classify its sentiment. Using POS tagging, Polarity Classification and Frequency of the words we need to select the ones which has a significant weight on the overall class label.

Building model:
This section of our project includes using different suitable classification and clustering algorithms on our processed dataset to build a model that can successfully detect sentiments for tweets.

Accuracy check:
The model has to work on the test dataset and yield a high accuracy rate for our project to be complete. Any model with error rate greater than 0.5 is unacceptable. Different model is to be created and compared with a various combination of features and algorithms to determine the most accurate result. And that would be our final model classifier.

4.4 Summary

The summary of this chapters are described in the list below:

- To classify the unlabeled data, at first the train data must be preprocessed by eliminating the duplicate and conflicting tweets to get a more accurate model.
- Each tweet must be separated into features to show the features as columns.
- Feature with a significant weight should be selected.
- Apply classification and clustering algorithms to build the model.
- Different model must be created to compare and check accuracy on each model.

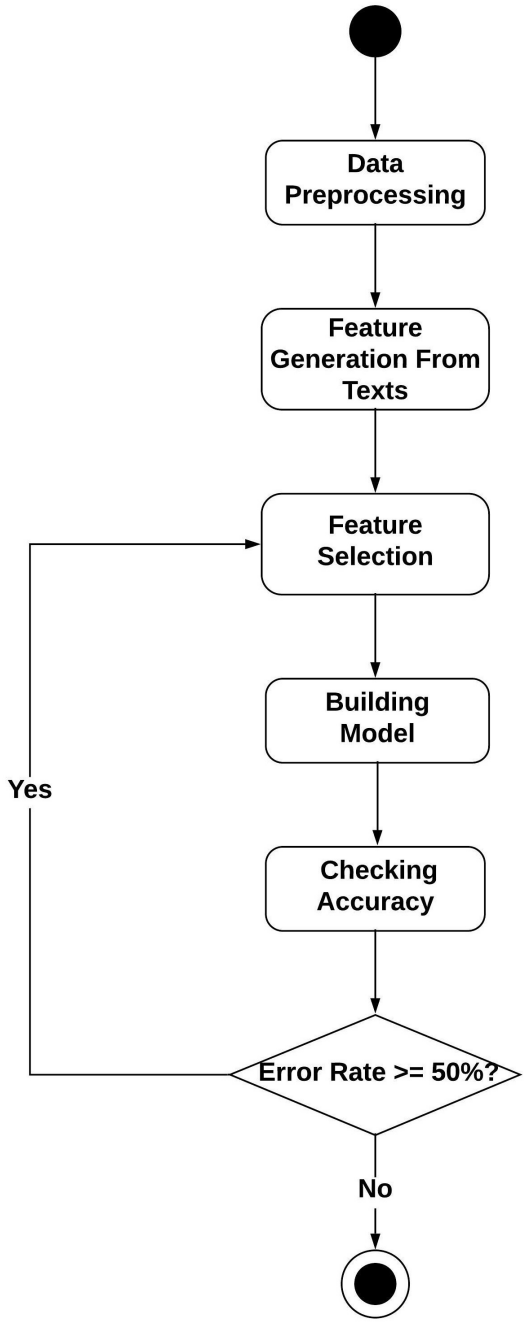


Figure 4.2: Flowchart showing the overview of the procedures.

Chapter 5

Methodology

5.1 Introduction

This chapter will provide the details of the methods that we have followed in our project. We have followed various techniques as discussed in our 2nd Chapter. All the previous works that we researched on Natural Language Processing and Twitter Sentiment Classification either had 2 classes i.e positive or negative. In rare cases it had 3 classes i.e. positive, negative and neutral. But according to what we have proposed before, we worked with **13 classes / emotions** (*'Anger'*, *'Anxiety'*, *'Calm'*, *'Disgust'*, *'Enthusiasm'*, *'Fear'*, *'Interestingness'*, *'Joy'*, *'Neutral'*, *'Rejection'*, *'Sadness'*, *'Shame'*, *'Surprise'*). A lot of data collection have been done for this reason. Rest of this section delineates the steps & algorithms followed in this work.

5.2 Preprocessing

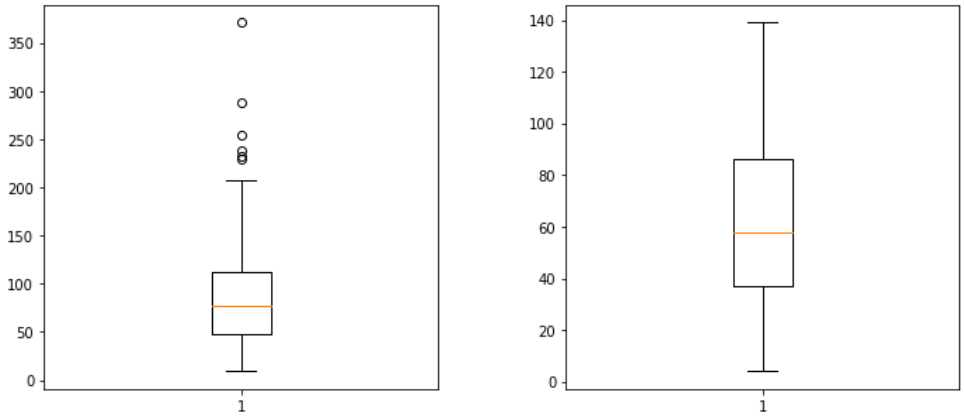
In this project, we have used the dataset given by our mentor Dr. Dewan Md. Farid. However, the dataset contained a lot of anomalies. Since this can hamper our model training, we have performed some preprocessing on our dataset before implementing the algorithms. General preprocessing includes removal of duplicate tweets and also applying active learning to validate the tweets and their corresponding emotion in the given dataset. There after we have done further preprocessing in our dataset in 2 approaches
-

5.2.1 Using LIWC

LIWC stands for Linguistic Inquiry and Word Count. Using the LIWC software we transformed our data, which was in string form into integer form. LIWC assigns numerical values to each 70 of its pre-defined psychological, social and grammatical linguistic categories, in regards to the affinity of the words in a sentence with those categories. After feeding our string format dataset into LIWC, we achieved a refined numerical dataset with 70 different attributes with 1210 instances distributed in 13 different class labels. This numeric to multinomial dataset was later used for building ensemble classifier and logistic regression models. The dataset was randomly divided into train and test set where 70% belonged to the training set.

5.2.2 Using Python

We checked our whole dataset for the character count since Twitter has a character restriction of only 280 per tweet. We found some tweets that had well over 280 characters. This was caused due to the change in character encoding while we were collecting the Tweet from Twitter. These garbage tweets were then deleted. Finally we ended up with tweets that had a max of 140 characters. The box plot shows the difference before and after cleaning of 1st step.



(a) Before Cleansing; **(b)** After Cleansing;

Figure 5.1: Box Plot of Character Count of the Tweets.

The following steps/actions were taken to cleanse the tweets.

- Few tweets contain HTML tags and URLs. These will make our model more inaccurate. So, removed the tags and URLs from the tweet.
- Few tweets also contained hash-tags(#), mentions (@) & emoticons in our tweets. We have ignored these as we won't be dealing with them right now.
- A lot of the tweets contains negations. We have substituted them with their full forms.
- We are keeping only the letters. Thus we are ignoring any digits that are in the tweets.
- All the punctuation marks will also be ignored.
- Finally all the tweets are converted to lowercase letters as in case of natural language processing, capital letters and small letters of the same alphabet has different meaning to the computer.

The cleaned dataset has 1210 samples. From these samples 70% of the samples were selected to construct the training dataset. The rest was further divided to equally to create the test and validation dataset.

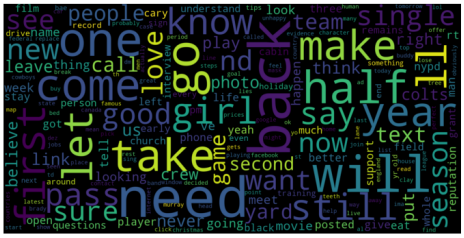
Formally, the dataset S can be shown as below:

$$S = S^1 \cup S^2 \cup S^3 \cup S^4 \cup S^5 \cup S^6 \cup S^7 \cup S^8 \cup S^9 \cup S^10 \cup S^11 \cup S^12 \cup S^13 \tag{5.1}$$

Here, S^1, S^2, S^3, S^4, S^5, S^6, S^7, S^8, S^9, S^10, S^11, S^12, S^13 are the set of *'Anger'*, *'Anxiety'*, *'Calm'*, *'Disgust'*, *'Enthusiasm'*, *'Fear'*, *'Interestingness'*, *'Joy'*, *'Neutral'*, *'Rejection'*, *'Sadness'*, *'Shame'*, *'Surprise'* instances respectively.

5.3 Implemented Algorithms

From our background research we have come to a conclusion that we will be implementing algorithms such as logistic regression, neural network, etc. But before we implement them, we first need to visualize the tweets that we have. We need to see their contents. Our dataset contains specifically two classes (S^9 and S^5) that have a significant number of samples than the rest. So we construct a WordCloud plot of these two classes.



(a) Class - Neutral (331);

(b) Class - Enthusiasm (186);

Figure 5.2: WordCloud Plot of 2 classes.

We then create a term frequency directly from sparse matrix for all the emotions. We see that stop words like to, the, and, in etc. have the most frequency.

Index	Anger	Anxiety	Calm	Disgust	Enthusiasm	Fear	Interestingness	Joy	Neutral	Rejection	Sadness	Shame	Surprise	Total
the	39	19	12	6	80	4	41	55	146	3	25	4	20	454
to	31	24	16	6	80	3	40	51	135	8	34	9	14	451
you	35	7	7	8	53	8	36	65	62	11	11	3	6	312
and	16	15	16	1	44	1	28	45	62	2	16	5	12	263
in	11	7	16	4	39	3	23	21	84	4	14	3	17	246
not	34	11	8	5	13	2	6	18	47	17	21	3	15	200
it	17	6	3	1	36	2	12	24	39	8	24	1	6	179
is	18	4	7	0	31	1	23	23	52	4	8	1	6	178
of	14	8	8	3	27	0	10	18	61	4	13	2	4	172
on	19	5	4	3	30	2	12	13	61	3	10	2	8	172

Figure 5.3: Term Frequency Matrix (Top 10 words).

A zipf plot is drawn. The zipf plot follows a law. The zipf law states that the probability of occurrence of words or other items starts high and tapers off. Thus, a few occur very often while many others occur rarely.

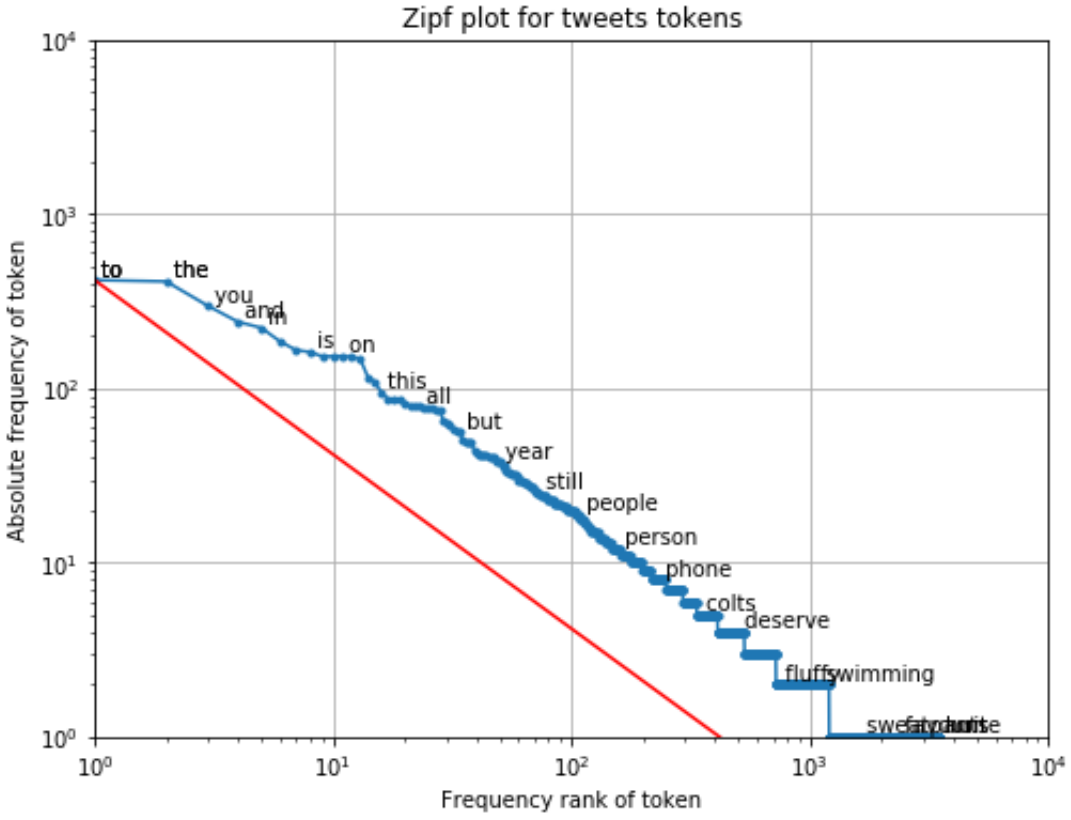


Figure 5.4: Zipf Plot for Tweet Tokens.

The words need to be encoded as integers or floating point values for use as input to a machine learning algorithm, called feature extraction (or vectorization). This will be done by implementing different vectorizer.

5.3.1 LDA

We have implemented ensemble classifiers with Linear Discriminant Analysis (LDA). LDA is a supervised dimensionality reduction technique. LDA calculates the best projection directions on which the within-class scatter matrix is minimized while the between-class scatter matrix is maximized under the Fisher criterion (Fukunaga, 1990).

5.3.2 Ensemble Classifiers with LDA

As stated earlier, our dataset had class imbalance problem. Ensemble-based algorithms are proven to work better than single classifier models for imbalanced data. The ultimate goal of ensemble methods is to combine the predictions of several base estimators built with a given learning algorithm in order to improve robustness over a single estimator. We applied a correlation similarity measure for dimensional reduction before building the classifier model.

5.3.2.1 Random Forest

With the dataset preprocessed using the LIWC software and LDA, we at first applied Random Forrest ensemble classifier. We changed the value of $n - estimators = 100$ and $max - features = sqrt(n - features)$ for classification tasks (where $n - features$ is the number of features in the data). This resulted on the model with very high accuracy in test set but comparatively low on train set.

Algorithm 3 Random Forest with LDA

1: **procedure** RF_LDA
2: $sc \leftarrow StandardScaler()$
3: $X\_train \leftarrow sc.fit\_transform(x\_train)$
4: $X\_test \leftarrow sc.fit\_transform(x\_test)$
5: $model \leftarrow RandomForest()$
6: $model.predict(x\_train, y\_train)$
7: **end procedure**

5.3.2.2 Bagging

With the dataset having 70 feature columns formed using LIWC, we applied Bagging classifier from python library scikit learn. We used Bagging as the base classifier with parameters max-samples and max-features which control the size of the subsets. Fitted on the training set, bagging showed great results. Changing the n-component parameter of LDA showed different outcomes.

Algorithm 4 Bagging with LDA

```
procedure BAGGING_LDA
    sc ← StandardScaler()
    X_train ← sc.fit_transform(x_train)
    X_test ← sc.fit_transform(x_test)
    model ← Bagging()
    model.predict(x_train, y_train)
end procedure
```

5.3.2.3 AdaBoost

We also used AdaBoost classifer on the preprocessed dataset using LIWC and LDA. Although the training accuracy of Adaboost was not as impressive as the former two, the train and test accuracy had low difference. Which means the model was not overfitted. We also tuned several parameters such as n-estimators, max-depth and min-samples-split to achieve the best outcome from it.

Algorithm 5 Boosting with LDA

```
procedure ADABOOST_LDA
    sc ← StandardScaler()
    X_train ← sc.fit_transform(x_train)
    X_test ← sc.fit_transform(x_test)
    model ← AdaBoost()
    model.predict(x_train, y_train)
end procedure
```

5.3.3 Logistic Regression

Logistic regression (LR) is a kind of predictive analysis.We have used Multinomial Logistic Regression (LR) as it is a multi-class problem. It is also known as Softmax Regression. Multinomial LR classifiers are commonly used as an alternative to naive Bayes classifiers. This is because they do not assume statistical independence of the random variables/ features that serve as predictors. Here Multinomial Logistic Regression is performed along with few other embedding techniques. These are listed below-

5.3.3.1 Using LDA

We have implemented multi-class logistic regression classifier on our numeric to nominal dataset along with linear discriminant analysis. We tuned different parameters with each iteration to observe the model's changes. With LDA parameter n-components being 3, we reached to an outcome that was the best among all the classifier models till now.

Algorithm 6 Logistic Regression with LDA

1: **procedure** LR_LDA
2: $sc \leftarrow StandardScaler()$
3: $X\_train \leftarrow sc.fit\_transform(x\_train)$
4: $X\_test \leftarrow sc.fit\_transform(x\_test)$
5: $model \leftarrow LogisticRegression()$
6: $model.predict(x\_train, y\_train)$
7: **end procedure**

5.3.3.2 Using CountVectorizer

We have used 'CountVectorizer (CV)' along with logistic regression to perform our classification. We have previously found the most used stopwords (the, to, and, it, etc)in our dataset. We will use those stopwords and remove them from our dataset to make the model more accurate. First we ran the model training for a range of features. The model with the best accuracy is chosen for our test case. We did this for using unigrams (1 Word). This process was repeated for different $n - grams$. We got the best test accuracy when the number of features was 150.

Algorithm 7 Logistic Regression using Count Vectorizer

1: **procedure** LR_CV
2: $model \leftarrow Pipeline(CountVectorizer, LogisticRegression(max\_features, ngram\_range, stop\_words))$
3: $model.fit(x\_train, y\_train)$
4: **end procedure**

5.3.3.3 Using Term Frequency-Inverse Document Frequency

Term Frequency-Inverse Document Frequency (TFIDF) is another way to convert textual data to numeric form. It increases co-relatively to the number of times that word

materialize in the document. Both the x_train and x_test were transformed using the vectorizer and then normal logistic regression is performed.

Algorithm 8 Logistic Regression using TFIDF Vectorizer

procedure LR_TFIDF
    $model \leftarrow TfidfVectorizer(max\_features \leftarrow 1000, ngram\_range \leftarrow (1,3))$
    $x\_train\_tfidf \leftarrow model.transform(x\_train)$
    $x\_test\_tfidf \leftarrow model.transform(x\_test)$
    $LogisticRegression.fit(x\_train\_tfidf, y\_train)$
end procedure

5.3.3.4 Using Doc2Vec

Every paragraph is mapped to a unique vector, represented by a column in matrix D and every word is also mapped to a unique vector, represented by a column in matrix W. The paragraph vector and word vectors are averaged or concatenated to predict the next word in a context. The paragraph token can be thought of as another word. It acts as a memory that remembers what is missing from the current context or the topic of the paragraph.

Algorithm 9 Logistic Regression using Doc2Vec

procedure LR_DOC2VEC_DBOW
    $model \leftarrow Doc2Vec$
    $model.build\_vocab(x\_test + x\_train)$
    for each i in $epoch$ **do**
        $model.train$
        $model.alpha- = .02$
    end for
    $x\_train \leftarrow get\_vectors(model, x\_train)$
    $x\_test \leftarrow get\_vectors(model, x\_test)$
    $LogisticRegression.fit(x\_train, y\_train)$
end procedure

- **Using Distributed Bag Of Words (DBOW)**
  For our Distributed Bag of Words, we will have to build a vocabulary for the model. The vocabulary is built on the whole dataset (Tweets). We have used $epoch = 100$ to build the vocabulary. After each epoch, the weight of the the words are being updated. The vectors are generated by predicting a probability

distribution of words in a paragraph. Then normal Logistic Regression is used to train and test the model. Both the test and train tweets was converted to vectors using the vocabulary that was built earlier.

- **Using Distributed Memory Concatenated (DMC)**
  The Distributed Memory Concatenated method is similar to DBOW except that there is a minor change in how the vocabulary is created. It uses the concatenation method. Other than that, everything is same. It takes the mean of the vectors. The vectors are generated by predicting a center word based on context words and a context paragraph.

- **Using Distributed Memory Mean (DMM)**
  The Distributed Memory Mean method is similar to DMC except that this method uses mean method. The vectors are generated by predicting a center word based on context words and a context paragraph.

- **Combination of DBOW, DMC, DMM**
  The models generated above was then concatenated to form a single model.The concatenated document vectors in different arrangements boosted the performance of the model.

5.3.4 Neural Network

We have constructed a neural network for the sentiment classification. Although our dataset is a highly imbalanced dataset, we managed to achieve some accuracy. But before implementing Neural Network, we had to perform more pre-processing of our Tweets and its classes.

- **Tweets**: We had to tokenize our Tweets. This causes each Tweet string to break into different tokens. However, the number of tokens in each samples were not same as the Tweets varied in length. So we had to use padding. We padded the sequence to the maximum length.

- **Sentiments**: We had the sentiments as strings. Thus machine learning algorithm will not be able to classify them. Also we couldn't simply replace them with integer values as the algorithms may use the integers to prioritize them. So we had to use

the built-in function of Python 'np_utils.to_categorical' which first encodes the classes using 'One-Hot Encoding' and then assigning an integer value to remove the possibilities of prioritizing the classes.

We have implemented Fully Connected Neural Network (FC_NN), Recurrent Neural Network (RNN) along with Long Short Term Memory (LSTM), Convolutional Neural Network (CNN) along with Fully connected Neural Network (FC_NN) and Dropout. We will discuss about their characteristic and their parameters in the section below-

5.3.4.1 FC_NN

We experimented the model with 200 epochs. We found that the model had a good balance between the test and train accuracy when we had 100 epochs. The model becomes too complex as the training accuracy goes $\geq 85\%$ while the testing accuracy decreases to lowest of about 15%. Therefore after experimenting with different epochs, we decided to settle with 100.

The model contains 2 hidden layer with 64 hidden nodes per layer. We have used 'relu' activation function in this 2 layers. The final layer uses 'softmax' activation and has 13 nodes. The model is compiled with the 'Adam' optimizer (a variant on Stochastic Gradient Descent) and trained using the 'categorical_crossentropy' loss.

Algorithm 10 Fully Connected Neural Network

1: **procedure** FC_NN
2: $model \leftarrow Sequential$
3: $model.add(Dense)$
4: $model.add(Dense)$
5: $model.add(Dense)$
6: $model.compile(loss, optimizer, metrics)$
7: $model.fit(x\_train, y\_train, epoch, batch\_size)$
8: $model.predict(x\_test)$
9: **end procedure**

5.3.4.2 RNN + LSTM

We ran the training phase for upto 50 epochs. We found that the model becomes more complex as we train for more than 20 epochs thus causing higher train accuracy ($> 93\%$)

but with lower test accuracy ($< 15\%$). Therefore we have decided to select epochs to be 20. We have used dropout to prevent over-fitting of the training data. The output layer is a 'Dense' fully-connected layer. This produces a probability using 'softmax' activation.

The model is compiled with the 'Adam' optimizer (a variant on Stochastic Gradient Descent) and trained using the 'categorical_crossentropy' loss.

Algorithm 11 Recurrent Neural Network with LSTM

1: **procedure** RNN_LSTM
2: $model \leftarrow Sequential$
3: $model.add(Embedding)$
4: $model.add(LSTM)$
5: $model.add(Dense)$
6: $model.add(Dropout)$
7: $model.add(Dense)$
8: $model.compile(loss, optimizer, metrics)$
9: $model.fit(x\_train, y\_train, epoch, batch\_size)$
10: $model.predict(x\_test)$
11: **end procedure**

5.3.4.3 CNN

We also ran the training phase for 100 epochs. The model becomes too complex as the training accuracy goes $> 97\%$ while the testing accuracy decreases to lowest of about 3%. Therefore after experimenting with different epochs, we decided to settle with 25.

For the convolutional layer, we have used 'valid padding' so our input will be of same length after the conv layer. The stride is fixed at 1. And we have used 'relu' activation for the first conv layer. A 'MaxPooling' is then performed on the previous output layer followed by 'Flatten'. There is a middle layer that is a 'Dense' fully-connected layer with nodes of 250. The output layer is also a 'Dense' fully-connected layer using 'sigmoid' activation.

The model is compiled with the 'Adam' optimizer (a variant on Stochastic Gradient Descent) and trained using the 'categorical_crossentropy' loss.

Algorithm 12 Convolutional Neural Network with Fully Connected Network

```
1: procedure CC_FC_NN
2:     model ← Sequential
3:     model.add(Embedding)
4:     model.add(Dropout)
5:     model.add(Conv1D)
6:     model.add(MaxPooling3d)
7:     model.add(Flatten)
8:     model.add(Dense)
9:     model.add(Dense)
10:    model.compile(loss, optimizer, metrics)
11:    model.fit(x_train, y_train, epoch, batch_size)
12:    model.predict(x_test)
13: end procedure
```

5.4 Performance Evaluation

By implementing the above algorithms we have seen that Bagging and Boosting performs too good with the train case. Logistic Regression with CountVectorizer and RNN with LSTM was also pretty good. But in case of the test case Logistic Regression with LDA was the best. A detailed view of all the performance evaluations are available on the next chapter.

5.5 Summary

To create our model the following models have been used :

- Preprocessing 1: Using LIWC software to transform string format dataset to a numeric form with 70 features.
- Preprocessing 2: As our dataset contains a lot of anomalies, we have deleted tweets contain HTML tags and URLs ,hashtags (), mentions (@) , negations to clean the dataset.
- Algorithms:
  Ensemble classifiers -
  - Random Forest with linear discrimination analysis (LDA)

- Bagging with linear discrimination analysis (LDA)
- Boosting with linear discrimination analysis (LDA)

Logistic Regression

- Logistic Regression using count vectorizer
- Logistic Regression Using Term Frequency-Inverse Document Frequency
- Logistic Regression Using Doc2Vec
- Logistic Regression Using Distributed Bag of Words (DBOW)
- Logistic Regression Using Distributed Memory Concatenated (DMC)
- Logistic Regression Using Distributed Memory Mean (DMM)
- Logistic Regression Using combination of DBOW, DMC, DMM

Neural Network

- Fully Connected Neural Network(FCNN)
- Recurrent Neural Network (RNN) with LSTM
- Convolutional Neural Network (CNN) along with Fully connected Neural Network

Chapter 6

Experimental Results and Discussions

6.1 Introduction

In this chapter, we present the experimental results and analysis of the methods we followed. All the experiments were run ten times using the same dataset to get rid of anomalities. The classifiers were implemented using Scikit-learn library, Keras library, Gensim library, NLTK and Python 3.

6.2 Result of Used Algorithms

6.2.1 Ensemble Classifier

We have implemented different ensemble classifiers with LDA with varying parameters. The results are tabulated in the following sections.

6.2.1.1 Random Forest

The table below shows us the train statistics with varying parameter.

	Accuracy	**Precision**	**Recall**	**F1-Score**	**Parameters**
1	92%	94%	90%	92%	n_comp = 1
2	98%	98%	98%	98%	n_comp = 3

Table 6.1: LR + RF Train Result.

The table below shows us the test statistics with varying parameter.

	Accuracy	Precision	Recall	F1-Score	Parameters
1	24%	16%	20%	17%	n_comp = 1
2	35%	30%	35%	32%	n_comp = 3

Table 6.2: LR + RF Test Result.

6.2.1.2 Bagging

The table below shows us the train statistics with varying parameter.

	Accuracy	Precision	Recall	F1-Score	Parameters
1	92%	93%	92%	93%	$n_comp = 1$
2	97%	97%	97%	97%	$n_comp = 3$

Table 6.3: LR + Bagging Train Result.

The table below shows us the test statistics with varying parameter.

	Accuracy	Precision	Recall	F1-Score	Parameters
1	25%	18%	21%	19%	n_comp = 1
2	36%	35%	36%	35%	n_comp = 3

Table 6.4: LR + Bagging Test Result.

6.2.1.3 AdaBoost

The table below shows us the train statistics with varying parameter.

	Accuracy	Precision	Recall	F1-Score	Parameters
1	33%	09%	15%	12%	n_comp = 1
2	38%	31%	38%	30%	n_comp = 3

Table 6.5: LR + Boosting Train Result.

The table below shows us the test statistics with varying parameter.

	Accuracy	Precision	Recall	F1-Score	Parameters
1	29%	11%	15%	12%	n_comp = 1
2	32%	22%	32%	26%	n_comp = 3

Table 6.6: LR + Boosting Test Result.

6.2.2 Logistic Regression

We have experimented the logistic regression models that we have made with different parameters. The sections below shows us the statistics along with the parameters.

6.2.2.1 Using Linear Discriminant Analysis

The table below shows us the train statistics with varying parameter.

	Accuracy	Precision	Recall	F1-Score	Parameters
1	33%	30%	33%	22%	n_comp = 1
2	57%	41%	57%	44%	n_comp = 3

Table 6.7: LR + LDA Train Result.

The table below shows us the test statistics with varying parameter.

	Accuracy	Precision	Recall	F1-Score	Parameters
1	30%	16%	30%	17%	n_comp = 1
2	44%	33%	44%	35%	n_comp = 3

Table 6.8: LR + LDA Test Result.

6.2.2.2 Using Count Vectorizer

The table below shows us the train statistics with varying parameter.

	Accuracy	Precision	Recall	F1-Score	Parameters
1	56.90%	Micro - 57% Macro - 76% Weighted - 63%	Micro - 57% Macro - 41% Weighted - 57%	Micro - 57% Macro - 48% Weighted - 55%	max_features = 500 stop_words = 10 n_gram - (1,1)
2	83.56%	Micro - 84% Macro - 93% Weighted - 86%	Micro - 84% Macro - 74% Weighted - 84%	Micro - 84% Macro - 81% Weighted - 84%	max_features = 500 stop_words = 15 n_gram - (1,2)

Table 6.9: LR + CV Train Result.

The table below shows us the test statistics with varying parameter.

	Accuracy	Precision	Recall	F1-Score	Parameters
1	34.90%	Micro - 35% Macro - 30% Weighted - 32%	Micro - 35% Macro - 22% Weighted - 35%	Micro - 35% Macro - 23% Weighted - 29%	max_features = 500 stop_words = 10 n_gram - (1,1)
2	32.84%	Micro - 33% Macro - 22% Weighted - 29%	Micro - 33% Macro - 20% Weighted - 33%	Micro - 33% Macro - 19% Weighted - 28%	max_features = 500 stop_words = 15 n_gram - (1,2)

Table 6.10: LR + CV Test Result.

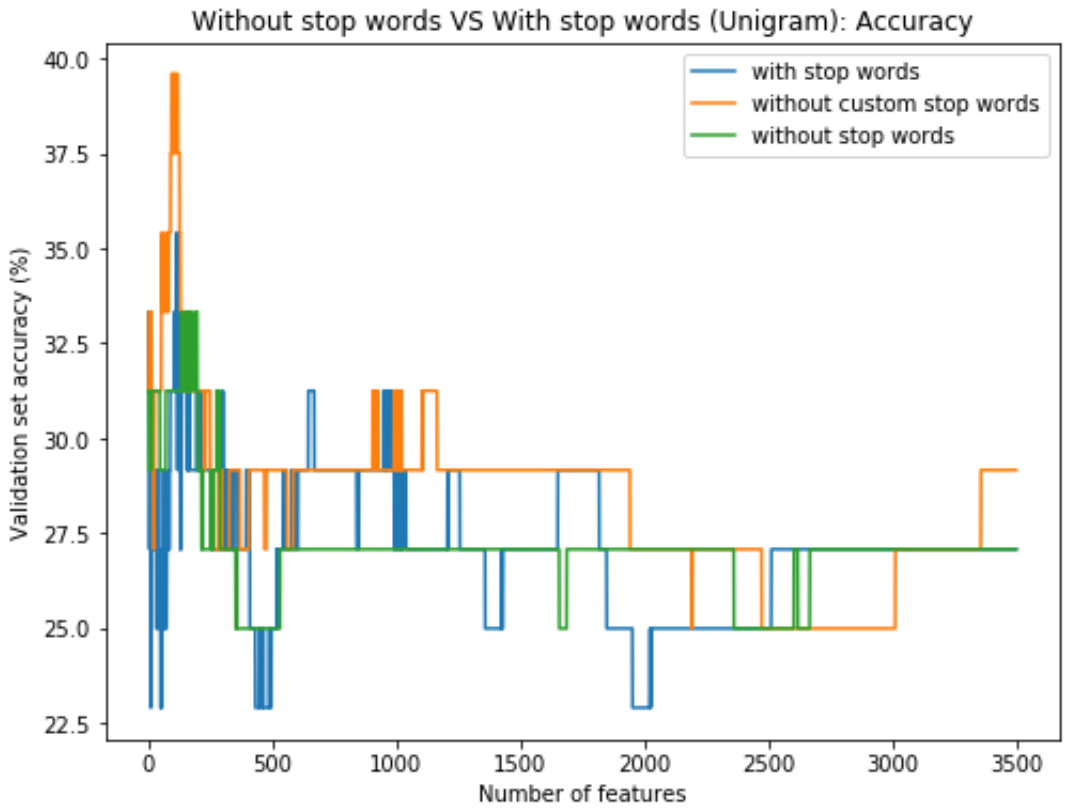


Figure 6.1: Accuracy of Unigram without stop words VS with stop words.

6.2.2.3 Using TFIDF Vectorizer

The table below shows us the train statistics with varying parameter.

	Accuracy	Precision	Recall	F1-Score	Parameters
1	54.18%	Micro - 54% Macro - 50% Weighted - 64%	Micro - 54% Macro - 25% Weighted - 54%	Micro - 54% Macro - 26% Weighted - 48%	max_features = 1000 n_gram - (1,3)

Table 6.11: LR + TFIDF Train Result.

The table below shows us the test statistics with varying parameter.

	Accuracy	Precision	Recall	F1-Score	Parameters
1	35.93%	Micro - 36% Macro - 22% Weighted - 29%	Micro - 36% Macro - 19% Weighted - 36%	Micro - 36% Macro - 17% Weighted - 27%	max_features = 1000 n_gram - (1,3)

Table 6.12: LR + TFIDF Test Result.

6.2.2.4 Using Doc2Vec

Using DBOW:

The table below shows us the train statistics with varying parameter.

	Accuracy	Precision	Recall	F1-Score	Parameters
1	31.54%	Micro - 32% Macro - 16% Weighted - 28%	Micro - 32% Macro - 11% Weighted - 08%	Micro - 32% Macro - 32% Weighted - 21%	alpha = 065 epoch = 200
2	31.63%	Micro - 32% Macro - 37% Weighted - 39%	Micro - 32% Macro - 12% Weighted - 11%	Micro - 32% Macro - 11% Weighted - 23%	alpha = 065 epoch = 100

Table 6.13: LR + Doc2Vec + DBOW Train Result.

The table below shows us the test statistics with varying parameter.

	Accuracy	Precision	Recall	F1-Score	Parameters
1	30.77%	Micro - 31% Macro - 10% Weighted - 11%	Micro - 10% Macro - 11% Weighted - 10%	Micro - 17% Macro - 31% Weighted - 19%	alpha = 065 epoch = 200
2	30.74%	Micro - 30% Macro - 10% Weighted - 17%	Micro - 30% Macro - 11% Weighted - 10%	Micro - 30% Macro - 30% Weighted - 18%	alpha = 065 epoch = 100

Table 6.14: LR + Doc2Vec + DBOW Test Result.

Using DMC:

The table below shows us the train statistics with varying parameter.

	Accuracy	Precision	Recall	F1-Score	Parameters
1	44.65%	Micro - 45% Macro - 44% Weighted - 46%	Micro - 45% Macro - 24% Weighted - 45%	Micro - 45% Macro - 27% Weighted - 40%	alpha = 0.65 epoch = 100 shuffle_epoch = 10
2	43.49%	Micro - 43% Macro - 51% Weighted - 46%	Micro - 43% Macro - 25% Weighted - 43%	Micro - 34% Macro - 43% Weighted - 39%	alpha = .095 epoch = 100 shuffle_epoch = 10

Table 6.15: LR + Doc2Vec + DMC Train Result.

The table below shows us the test statistics with varying parameter.

	Accuracy	Precision	Recall	F1-Score	Parameters
1	32.84%	Micro - 33% Macro - 22% Weighted - 25%	Micro - 33% Macro - 19% Weighted - 33%	Micro - 33% Macro - 18% Weighted - 26%	alpha = .065 epoch = 100 shuffle_epoch = 10
2	39.02%	Micro - 39% Macro - 29% Weighted - 32%	Micro - 39% Macro - 23% Weighted - 39%	Micro - 39% Macro - 22% Weighted - 30%	alpha = .095 epoch = 100 shuffle_epoch = 10

Table 6.16: LR + Doc2Vec + DMC Test Result.

Using DMM:

The table below shows us the train statistics with varying parameter.

	Accuracy	Precision	Recall	F1-Score	Parameters
1	33.69%	Micro - 34% Macro - 23% Weighted - 33%	Micro - 34% Macro - 12% Weighted - 34%	Micro - 34% Macro - 10% Weighted - 24%	alpha = .065 epoch = 30 shuffle_epoch = 1 window = 4
2	43.94%	Micro - 44% Macro - 47% Weighted - 46%	Micro - 44% Macro - 23% Weighted - 25%	Micro - 44% Macro - 25% Weighted - 39%	alpha = .095 epoch = 30 shuffle_epoch = 10 window = 4

Table 6.17: LR + Doc2Vec + DMM Train Result.

The table below shows us the test statistics with varying parameter.

	Accuracy	Precision	Recall	F1-Score	Parameters
1	36.96%	Micro - 37% Macro - 19% Weighted - 28%	Micro - 37% Macro - 19% Weighted - 37%	Micro - 37% Macro - 16% Weighted - 26%	alpha = .065 epoch = 30 shuffle_epoch = 1 window = 4
2	35.93%	Micro - 36% Macro - 16% Weighted - 25%	Micro - 36% Macro - 19% Weighted - 36%	Micro - 36% Macro - 17% Weighted - 27%	alpha = .095 epoch = 30 shuffle_epoch = 10 window = 4

Table 6.18: LR + Doc2Vec + DMM Test Result.

6.2.3 Neural network

The different neural networks that was constructed earlier was tested fee times by varying the parameters. In the sections below, we have included the evaluation table for both test and train cases including the parameters.

6.2.3.1 FC_NN

The table below shows us the train statistics with varying parameter.

	Accuracy	Precision	Recall	F1-Score	Parameters
1	73.41%	Micro - 73% Macro - 68% Weighted - 72%	Micro - 73% Macro - 50% Weighted - 73%	Micro - 73% Macro - 53% Weighted - 70%	kernel_size = 5 filters = 64 pool_size = 4 embedding_size = 5 batch_size = 32 epochs = 200 max_features = 1000

Continued on next page

	Accuracy	Precision	Recall	F1-Score	Parameters
2	41.33%	Micro - 41% Macro - 29% Weighted - 38%	Micro - 41% Macro - 18% Weighted - 41%	Micro - 41% Macro - 18% Weighted - 33%	kernel_size = 5 filters = 64 pool_size = 4 embedding_size = 5 batch_size = 64 epochs = 300 max_features = 2000
3	32.08%	Micro - 32% Macro - 27% Weighted - 30%	Micro - 32% Macro - 12% Weighted - 32%	Micro - 32% Macro - 11% Weighted - 20%	kernel_size = 5 filters = 64 pool_size = 4 embedding_size = 10 batch_size = 32 epochs = 300 max_features = 3000

Table 6.19: FC_NN Train Result.

The table below shows us the test statistics with varying parameter.

	Accuracy	Precision	Recall	F1-Score	Parameters
1	18.40%	Micro - 18% Macro - 10% Weighted - 6%	Micro - 18% Macro - 11% Weighted - 18%	Micro - 18% Macro - 10% Weighted - 16%	kernel_size = 5 filters = 64 pool_size = 4 embedding_size = 5 batch_size = 32 epochs = 200 max_features = 1000

Continued on next page

	Accuracy	Precision	Recall	F1-Score	Parameters
2	25.62%	Micro - 26% Macro - 18% Weighted - 20%	Micro - 26% Macro - 14% Weighted - 26%	Micro - 26% Macro - 13% Weighted - 19%	kernel_size = 5 filters = 64 pool_size = 4 embedding_size = 5 batch_size = 64 epochs = 300 max_features = 2000
3	30.77%	Micro - 31% Macro - 07% Weighted - 12%	Micro - 31% Macro - 13% Weighted - 31%	Micro - 31% Macro - 08% Weighted - 16%	kernel_size = 5 filters = 64 pool_size = 4 embedding_size = 10 batch_size = 32 epochs = 300 max_features = 3000

Table 6.20: FC_NN Test Result.

6.2.3.2 RNN + LSTM

The table below shows us the train statistics.

	Accuracy	Precision	Recall	F1-Score	Parameters
1	91.55%	Micro - 92% Macro - 88% Weighted - 92%	Micro - 92% Macro - 76% Weighted - 92%	Micro - 92% Macro - 76% Weighted - 91%	kernel_size = 5 filters = 64 pool_size = 4 lstm_output_size = 70 embedding_size = 5 batch_size = 32 epochs = 50 max_features = 1000

Continued on next page

	Accuracy	Precision	Recall	F1-Score	Parameters
2	74.39%	Micro - 74% Macro - 56% Weighted - 73%	Micro - 74% Macro - 47% Weighted - 74%	Micro - 74% Macro - 46% Weighted - 72%	kernel_size = 5 filters = 64 pool_size = 4 lstm_output_size = 70 embedding_size = 5 batch_size = 32 epochs = 50 max_features = 500

Table 6.21: RNN + LSTM Train Result.

The table below shows us the test statistics.

	Accuracy	Precision	Recall	F1-Score	Parameters
1	24.59%	Micro - 25% Macro - 14% Weighted - 23%	Micro - 25% Macro - 15% Weighted - 25%	Micro - 25% Macro - 15% Weighted - 24%	kernel_size = 5 filters = 64 pool_size = 4 lstm_output_size = 70 embedding_size = 5 batch_size = 32 epochs = 50 max_features = 1000
2	28.71%	Micro - 29% Macro - 21% Weighted - 28%	Micro - 29% Macro - 23% Weighted - 29%	Micro - 29% Macro - 21% Weighted - 27%	kernel_size = 5 filters = 64 pool_size = 4 lstm_output_size = 70 embedding_size = 5 batch_size = 32 epochs = 50 max_features = 500

Table 6.22: RNN + LSTM Test Result.

6.2.3.3 CNN + FC_NN

The table below shows us the train statistics.

	Accuracy	Precision	Recall	F1-Score	Parameters
1	80.23%	Micro - 80% Macro - 85% Weighted - 81%	Micro - 80% Macro - 64% Weighted - 80%	Micro - 80% Macro - 67% Weighted - 79%	kernel_size = 5 filters = 64 pool_size = 4 embedding_size = 5 batch_size = 32 epochs = 25 max_features = 1500
2	51.30%	Micro - 51% Macro - 35% Weighted - 50%	Micro - 51% Macro - 29% Weighted - 51%	Micro - 51% Macro - 27% Weighted - 46%	kernel_size = 5 filters = 64 pool_size = 8 embedding_size = 2 batch_size = 32 epochs = 25 max_features = 1000

Table 6.23: CN + FC_NN Train Result.

The table below shows us the test statistics.

	Accuracy	Precision	Recall	F1-Score	Parameters
1	18.4%	Micro - 18% Macro - 11% Weighted - 16%	Micro - 18% Macro - 11% Weighted - 18%	Micro - 18% Macro - 11% Weighted - 17%	kernel_size = 5 filters = 64 pool_size = 4 embedding_size = 5 batch_size = 32 epochs = 25 max_features = 1500

Continued on next page

	Accuracy	Precision	Recall	F1-Score	Parameters
2	30.77%	Micro - 31% Macro - 21% Weighted - 29%	Micro - 31% Macro - 19% Weighted - 31%	Micro - 31% Macro - 16% Weighted - 24%	kernel_size = 5 filters = 64 pool_size = 8 embedding_size = 2 batch_size = 32 epochs = 25 max_features = 1000

Table 6.24: CN + FC_NN Test Result.

6.3 Discussions of the Result

We used different preprocessing approaches and a number of machine learning algorithms to construct classification models. Multiple parameters were tuned in each classifier to get the best outcome possible from those models. Although some models outperformed in training sets, in can be concluded that the test results of the classifiers were pretty underwhelming. Which translates that the class imbalance problem in our dataset was too high for general machine learning algorithms to handle.

6.4 Summary

The table below summarizes the best accuracy we got and its corresponding recall, precision and f1-scores. *Weighted scores are taken as the our data is highly skewed.

Algorithm Name	**Best Training Accuracy**	**Best Testing Accuracy**	**Test Precision**	**Test Recall**	**Test F1-Score**
RF + LDA	98%	35%	30%	35%	32%
Bagging + LDA	97%	36%	35%	36%	35%
Adaboost + LDA	38%	32%	22%	32%	26%
LR + LDA	57%	44%	33%	44%	35%
LR + CV	83.56%	32.84%	29%	33%	28%
LR + TFIDF	54.18%	35.93%	29%	36%	27%
LR + DOC2VEC + DBOW	31.63%	30.74%	17%	10%	18%
LR + DOC2VEC + DMC	43.49%	39.02%	32%	39%	30%
LR + DOC2VEC + DMM	43.94%	35.93%	25%	36%	27%
FC_NN	41.33%	25.62%	20%	26%	19%
RNN + LSTM	74.39%	28.71%	28%	29%	27%
CNN + FC_NN	51.30%	30.77%	29%	23.31%	21%

Table 6.25: Table showing the best results.

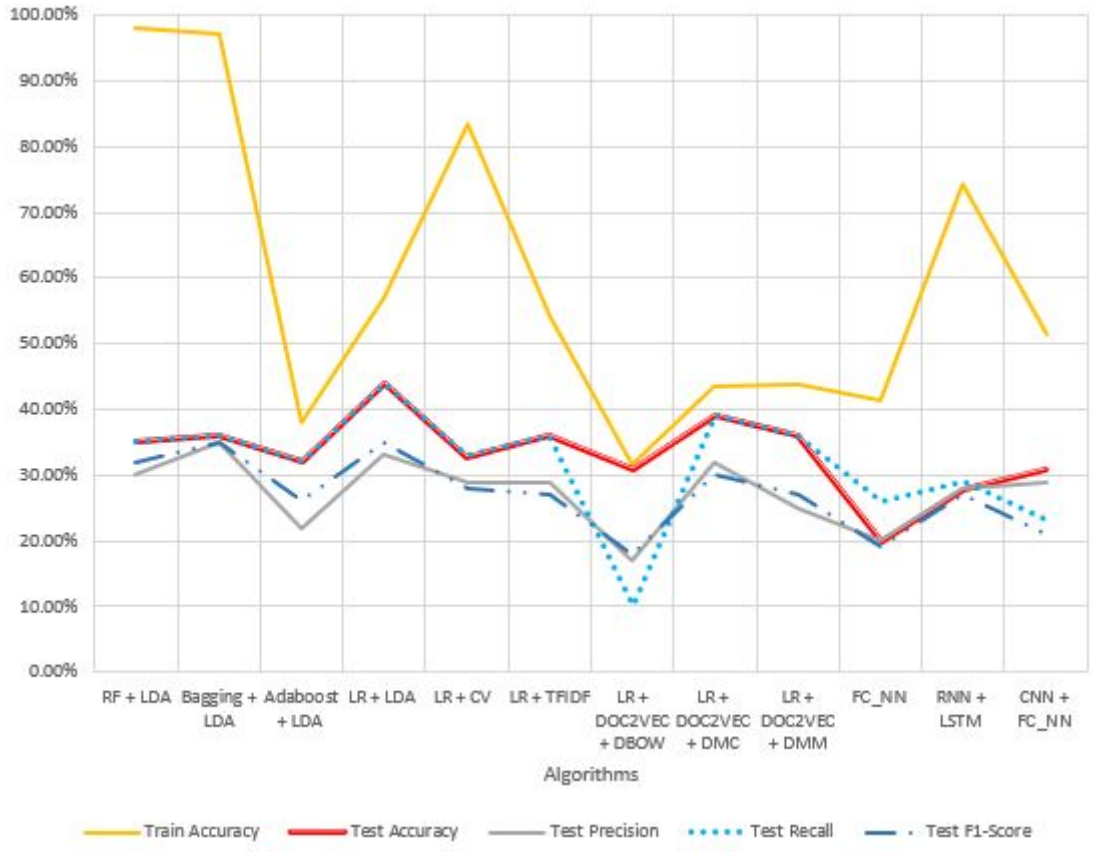


Figure 6.2: Comparison of the Experimental Results.

Chapter 7

Specification

7.1 Introduction

We will be developing a model for our Twitter Sentiment Classification. Before any implementation, a few background checks and analysis needs to be done. These are outlined in the sections below.

7.2 Design of a New System

7.2.1 Overview

We have decided to make system that will collect Tweet from the user and determine the emotion behind it.

7.2.2 Purpose

The purpose of this work is to determine the emotion/sentiment of a given Tweet with a much less hassle.

7.2.3 Scope

This system will provide more flexibility in determining the emotion of what a person Tweets. Since it has **13 emotion levels**, it can provide a wide range of outputs.

7.2.4 Existing Systems

There are only a very few number of web pages available for sentiment classification. Amongst the notable ones, 2 of them (TweetAnalyzer & Sentigem) are already offline now. These two were relatable to what we want to do. The remaining one (Sentiment Viz) is online but it has too many limitations-

- It doesn't take input of a single Tweet.
- It takes only keywords as input. And returns the top tweets with that keywords from Twitter.
- It doesn't provide a definitive emotion rather gives a emotion map.

7.2.5 Proposed System

- Will provide '13' definitive emotion.
- Will take a single tweet as an input

7.2.5.1 Diagrams

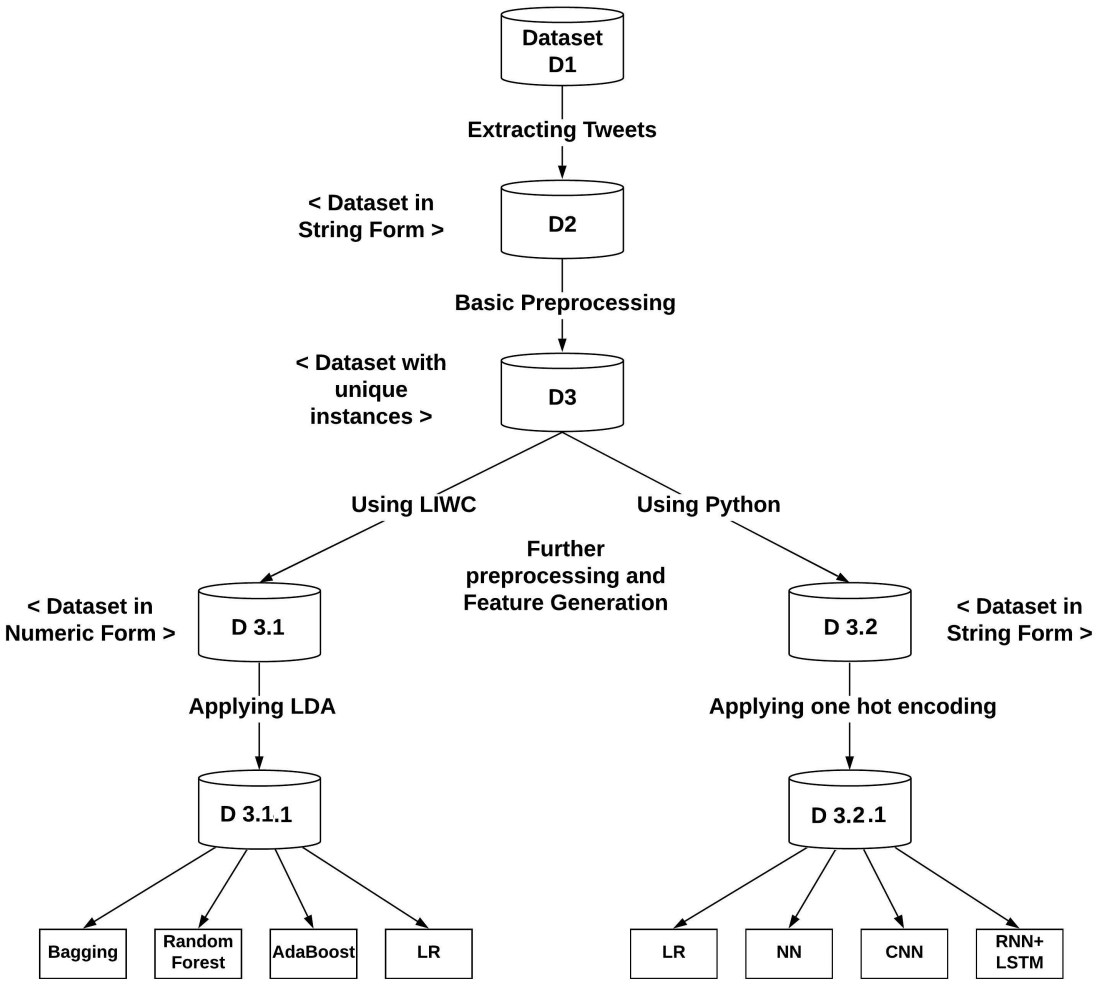


Figure 7.1: Overview of the processes we have applied.

7.3 Complex Engineering Decisions

We had many options for the software tools that we had for this project. We will be describing about them and the complex engineering decisions that we had to make.

7.3.1 Programming Language

For the machine learning model generation, We were presented with few good options such as Julia, R, Haskell and Python. We will discuss about the reasons why we decided to choose one and ignore the rest.

- **Python** - We have decided to use Python for the model creating and generating purpose. Although it has both pros and cons but the cons had more effect on us.
  - **Advantages**:
    - Extensive support libraries.
    - Integration feature.
    - Improved programmers productivity.
    - Extensive help on the Internet.
    - Portable.
  - **Disadvantages**:
    - Underdeveloped Database Access Layer.
    - Weak in mobile computing.
    - Speed is slow.
- **Julia** - We have decided not to use Julia due to the most major disadvantage it has.
  - **Advantages**:
    - Julia is faster.
    - It supports cross-platform package usage.
  - **Disadvantages**:
    - Less documentation and packages available.
- **R** - R has disadvantages similar to Julia. Thus we didn't use this one.
  - **Advantages**:
    - It supports extension.
    - Better for exploring dataset.

  - **Disadvantages**:
    * Less documentation and packages available.
    * Harder to maintain larger codes.

- **Haskell** - We have decided not to use Haskell as we are not used to the programming.
  - **Advantages**:
    * Fast Prototyping Performance
  - **Disadvantages**:
    * Less documentation and packages available.
    * Too much memory usage.

7.3.2 Development Tools - IDE

There are loads of development tools and environments available. From those, we shortlisted few and then finally chose the one that's best for our interest.

- **Spyder**- We will be using Spyder due to its superiority over the other available IDEs.
  - **Advantages**:
    * Better for data analysis.
    * Light weight IDE.
    * Faster than other available IDEs.
    * Can run a specific portion of a code without compiling everything.
  - **Disadvantages**: There are no disadvantages of spyder.

- **PyCharm**- We won't be using PyCharm as the disadvantages outweighs the advantages.
  - **Advantages**:
    * Supports many plugins.
  - **Disadvantages**:
    * Cannot run a specific part of the code.

7.3.3 Document Generation Tools

We were presented with 2 document generation tools. Below are their characteristics.

- **Latex**- We chose to use Latex for our document generation.
  - **Advantages**:
    - Better for tables and illustrations.
    - Consistent handling of references and bibliography.
  - **Disadvantages**:
    - Hard for entry level users.
- **Microsoft Word**- We won't use Word as it has too many disadvantages for this type of document.
  - **Advantages**:
    - Better at spell check.
    - Easier to use as it WYSIWYG.
  - **Disadvantages**:
    - Cannot edit simultaneously through internet.
    - Bad at keeping the contents and formatting same.

We chose to go with Latex as it has much more advantages than the other.

7.4 Deployment

Since this system will need a high computational power, it has to abide by few systems.

- **Operating System** - We don't have any specific requirements for the operating system. However, Windows 10 is preferred.
- **Hardware** - For model training, a GPU with a memory of 4GB is highly recommended for the use of 'TensorFlow' and 'Keras'.

Chapter 8

Standards

8.1 Compliance with standards

In general, we have to be conforming to certain types of rules such as specification, law, policy and regulations. In this era of information, data protection has never been more important. And keeping the data protected is crucial.

8.1.1 Types of standards

We will be following these different standards throughout the project -

8.1.1.1 Technological

Programming Languages	Python 3.7.0
	Java
Markup Languages	LaTeX
Data Formats	Comma Separated Value file (CSV)
	Microsoft Excel
Softwares	Weka
	Anaconda Navigator

Table 8.1: Technological Standards we have followed.

8.1.1.2 Ethics

Research about our methods and algorithms will involve a great deal of collaboration and systematization among different people in different branch of knowledge. Ethical

standard will promote the value that will be essential for collaborative work.
Thus we have to be fair, ethical, honest and have a proper sense judgment. We also have to have a moral responsibility to protect the research participants from harm.

So for the initial stage, we plan to follow -

- **ACM Code Of Ethics**
- **IEEE/ ACM software engineering code of ethics**
- **W3C Privacy Standard**

8.1.1.2.1 License

We have used few softwares throughout the project. Some of them are free to use while for some software like Weka, we have to add the citation of the developers. This has been followed to avoid any legal and ethical complications.

The dataset that we have acquired is collected from Twitter. This data is publicly available to everyone. However, Twitter states that we are not allowed to share our full data set with others: "If you provide Content to third parties, including downloadable datasets of Content or an API that returns Content, you will only distribute or allow download of Tweet IDs and/or User IDs."[**?**]

We will also be following these standards for ethics -

1. **GNU General Public License**. This software license is widely used for open-source software and freebies also. This license declares that any software under it is free to use, distribute and open for further development under the same license.

2. **ISO/IEC 12207** - International Organization for Standardization and International Electrotechnical Commission issued this standard combinedly for software engineers. This standard promotes not to develop any software which may harm mankind, society and environment economically, physically or in any other form.

8.1.1.3 Safety

We have to keep our website safe from hackers so that hackers don't use exploit our site to sniff our users' data or plant a virus in their computers.

8.2 Summary

To perform our project, we have followed certain types of rules and standards. The dataset that we used is available to everyone. The tools that we used, most of them are open-source software whereas the software that are not free, we have added citation of the developers.

Chapter 9

Impacts and Constraints

9.1 Introduction

Although we are doing a software based project, like any other projects, we also have both impacts and constraints. These are discussed in this chapter.

9.2 Impacts

The impact that our project may have is discussed below -

- **Economic Impact** - Our model can be utilized by the consumers to collect data concerning product or amenities before creating any purchase. It would allow companies to gain an overview of the huge public reactions behind any certain topics. They can use our project to fish out insights from the public and act accordingly. They can get an idea about how their products are perceived in the market without surveying people directly which is a plus point in the marketing research and growth.

- **Environmental Impact** - As our project is a software based project so our project is free from all sorts of materials and chemicals that may harm the wildlife or the environment. However, it would also not help the development of the environment.

- **Social Impact** - Our Model can also be used to determine the emotional condition of people in certain areas which could be a key factor in social revolution. Given an ID, people can see if the person is in a good mood or in a bad mood and then act accordingly to cheer him up or to motivate him. Political candidates could also use it to leverage public image in their campaign.

- **Political Impact** - Our project can be a leverage for governments as they can use our website to analyze and classify the public reactions for any decisions they take. Political parties can use this tool to see how people are reacting to their election manifests. They can use this result to do the needful for a more positive response. A Government can also use this tool to analyze people's reaction

- **Ethical Impact** - Although this project is built for the betterment of the society, it can also be used for purposes that have bad intentions. A Government can use this to identify the sentiment of a few groups of people and use that as a leverage to force them stuffs that they would normally avoid. People can also use this tool to take leverage of people's sentiment and make a profit.

- **Health and Safety Impact** - Our project doesn't have any health and safety impact directly. However, people can use this project to find any people who are suicidal and motivate them.

9.3 Constraints

We haven't faced any constraints till now, but since we are working with data collected from another specific source, we may encounter some more constraints like the following in the distant future

1. **Flexibility** -

   (a) We are using Twitter dataset for both test and training data. So the model built from that might not be able to classify sentiment if the given input is from a different social networking site.

(b) Our project will only work as long as Twitter remains public and open to all. If for any reasons Twitter changes their policy and restricts the privacy of their users tweets, then the model will not work.

2. **Sustainability** -

(a) Its sustainability is also to be questioned because it will only be effective if people are using Twitter frequently. If Twitter usage reduces exponentially, the software benefits will also disintegrate.

(b) The softwares that we used to develop both the model and application may impose fees and may restrict our use due to any change in their policy.

(c) The data is collected from Twitter. Twitter is distributing these datasets for free. If there is any fee imposed, we may have to limit our model.

3. **Availability** -

(a) We will currently be developing the model for tweets that are made in English. This system wont work for any languages other than English. So non-English speakers/tweets cannot avail the impact of this software.

(b) We are working freely as our government hasn't imposed any rules or regulations regarding the usage of public data. It may hamper our project if any sort of rules and regulations is passed in the future.

9.4 Summary

The chapter have been summarized and discussed in the items below:

- Impacts:
  - Has an economic impact as it helps the consumers to know about public reactions about their product.
  - Can be used to know the mental condition of certain area.
  - Political candidates can use this to leverage their public image and to analyze publics reaction.

  - As our project is a software based project, it does not have any environmental or healthy and safety impact.

- Constraints :

  - If Twitter/government change their terms and policies or if the user of twitter reduces then our project might not work efficiently.
  - Non-English speaker, users of another social media site may not be benefitted from our project.

Chapter 10

Conclusion

10.1 Introduction

In this paper, we have implemented several Ensemble Methods and Deep Learning techniques for classifying human emotions associated with tweets and have shown a comparison between their performances. Keeping in mind that the underlying class distribution is densely imbalanced, we have decided first to apply ensemble classifiers as ensemble methods have shown promising performance in the classification of imbalanced datasets. We also have applied Logistic regression, Neural Network, CNN and RNN along with LSTM for classifying the sequential text data. We have compared the results of each classifier to scrutinize which one performs better in case of massively imbalanced data.

At first we feed the data-set in LIWC software to quantify features in a text and assign integer value for each word in the text in seventy different psychological and grammatical categories. Then we applied LDA to reduce the dimentionality of features and create a feature sub-space. Then we adapted Random Forest, Bagging, AdaBoost and showed their outcomes. We also have applied Logistic Regression, Neural Network, CNN and RNN along with LSTM and compared the results of these Deep learning methods along with the ensemble methods. For each classifying model we randomly split the data-set into 70-30 ratio where train set is 70% and test set is 30%.

Most of our model showed significantly good results in the training sets but the test accuracy is not as impressive as the train one solely for the unhandled major class imbalance

problem. Among the ensemble classifiers and Neural Network models, the Random Forest and RNN with LSTM respectively, shows comparatively good results. But overall, the logistic regression model with LDA out performs the test accuracy of other classifiers.

10.2 Limitation

The research works we have studied in this context have only dealt with either positive and negative sentiments or with five basic human emotions. But here, we are dealing with 13 distinct human emotions where the underlying class distribution is densely imbalanced which makes the problem much more difficult to unravel. Therefore, initially we applied several machine learning methods to investigate which one of them performs comparatively better on this massively imbalanced data-set.

In future, we desire to handle class imbalance problem with robust traditional sampling techniques such as SMOTE, MSMOTE along with under sampling algorithms such as RUSBoost. We believe this will lead significant advances on the test accuracy of the models.

10.3 Summary

This paper is about comparing different machine learning algorithms like Random Forest, Bagging, Boosting, Logistic Regression and Neural Networks on highly imbalanced twitter data. LIWC software is used for feature quantification whereas LDA is applied to reduce the dimensionality of features and create a feature sub-space. The model shows significantly high accuracy in the train data but very poor accuracy in the test data. Among all the classifiers logistic regression model with LDA performs the best.

Bibliography

[1] E. Kouloumpis, TheresaWilson, and J. Moore, "Twitter sentiment analysis: The good the bad and the omg!" *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain*, January 2011. 4, 6

[2] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *CS224N Project Report, Stanford*, 2009. 4, 7

[3] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passonneau, "Sentiment analysis of twitter data," *Proceedings of the Workshop on Languages in Social Media, Portland, Oregon*, June 2011. 4

[4] S. Wakade, C. Shekar, K. J. Liszka, and C.-C. Chan, "Text mining for sentiment analysis of twitter data," *Proceedings of the International Conference on Information and Knowledge Engineering (IKE)*, 2012. 5

[5] A. Pak and P. Paroubek, "Twitter as a corpus for sentiment analysis and opinion mining," *Proceedings of the Seventh conference on International Language Resources and Evaluation*, May 2010. 5, 10

[6] L. Jiang, M. Yu, M. Zhou, X. Liu, and T. Zhao, "Target-dependent twitter sentiment classification," *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, June 2011. 5, 10

[7] M. S. H. Mukta, M. E. Ali, and J. Mahmud, "Identifying and validating personality traits-based homophilies for an egocentric network," *Springer Vienna*, September 2016. 5

[8] R. Irfan, C. K. King, D. Grages, S. Ewen, S. U. Khan, S. A. Madani, J. Kolodziej, L. Wang, D. Chen, A. Rayes, N. Tziritas, C.-Z. Xu, A. Y. Zomaya, A. S. Alzahrani,

and H. Li, "A survey on text mining in social networks," *The Knowledge Engineering Review*, March 2015. 5

[9] S. M. Mohammad, S. Kiritchenko, and X. Zhu, "Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets," *7th International workshop on Semantic Evaluation Exercises (SemEval-2013), Atlanta, USA*, June 2013. 6, 9

[10] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning sentiment-specific word embedding for twitter sentiment classification," *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, June 2014. 6, 10

[11] Y. Ren, Y. Zhang, M. Zhang, and D. Ji, "Context-sensitive twitter sentiment classification using neural network," *30th AAAI Conference on Artificial Intelligence (AAAI-16)*, February 2016. 7, 9

[12] J. Bollen, H. Mao, and A. Pepe, "Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena," *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain*, January 2011. 7

[13] D. M. Farid, A. Nowé, and B. Manderick, "Combining boosting and active learning for mining multi-class genomic data," pp. 1–2, 2016. 7

[14] S. Ahmed, A. Mahbub, F. Rayhan, R. Jani, S. Shatabda, and D. M. Farid, "Hybrid methods for class imbalance learning employing bagging with sampling techniques," in *2017 2nd International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS)*, Dec 2017, pp. 1–5. 8

[15] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 4, pp. 463–484, July 2012. 8

[16] Y. R. Tausczik and J. W. Pennebaker, "The psychological meaning of words: Liwc and computerized text analysis methods," *Journal of Language and Social Psychology*, vol. 29, no. 1, pp. 24–54, 2010. 8

[17] L. Barbosa and J. Feng, "Robust sentiment detection on twitter from biased and noisy data," *23rd International Conference on Computational Linguistics, Posters Volume, 23-27*, August 2010. 9

Appendix A

Code Snippets

A.1 Preprocessing

A.1.1 Using Python

```
from nltk.tokenize import WordPunctTokenizer
import re
tok = WordPunctTokenizer()
pat1 = r'@[A-Za-z0-9_]+'
pat2 = r'https?://[^ ]+'
combined_pat = r'|'.join((pat1, pat2))
www_pat = r'www.[^ ]+'
negations_dic = {"isn't":"is not", "aren't":"are not", "wasn't":
    "was not", "weren't":"were not", "haven't":"have not","hasn't":
    "has not","hadn't":"had not","won't":"will not",
     "wouldn't":"would not", "don't":"do not", "doesn't":
     "does not","didn't":"did not",
     "can't":"can not","couldn't":"could not","shouldn't":
     "should not","mightn't":"might not",      "mustn't":"must not"}
neg_pattern = re.compile(r'\b(' + '|'.join(negations_dic.keys()) + r')\b')
def tweet_cleaner(text):
  soup = BeautifulSoup(text, 'html.parser')
    souped = soup.get_text()
    try:
        bom_removed = souped.decode("utf-8-sig").replace(u"\ufffd", "?")
    except:
        bom_removed = souped
    stripped = re.sub(combined_pat, '', bom_removed)
    stripped = re.sub(www_pat, '', stripped)
```

```
    lower_case = stripped.lower()
    neg_handled = neg_pattern.sub(lambda x: negations_dic[x.group()], lower_case)
    letters_only = re.sub("[^a-zA-Z]", " ", neg_handled)
    words = [x for x  in tok.tokenize(letters_only) if len(x) > 1]
    return (" ".join(words)).strip()
df['text'] = df['text'].apply(tweet_cleaner)
```

A.2 Ensemble Classifier

A.2.1 Random Forest

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.ensemble import RandomForest

lda = LDA(n_components=1)
X_train = lda.fit_transform(X_train, y_train)
X_test = lda.transform(X_test)

classifier = RandomForest()
```

A.2.2 Bagging

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.ensemble import BaggingClassifier

lda = LDA(n_components=1)
X_train = lda.fit_transform(X_train, y_train)
X_test = lda.transform(X_test)

classifier = BaggingClassifier()
```

A.2.3 AdaBoost

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.ensemble import BoostingClassifier

lda = LDA(n_components=1)
X_train = lda.fit_transform(X_train, y_train)
X_test = lda.transform(X_test)

classifier = BoostingClassifier()
```

A.3 Logistic Regression

A.3.1 Using Linear discriminant analysis

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

lda = LDA(n_components=1)
X_train = lda.fit_transform(X_train, y_train)
X_test = lda.transform(X_test)

classifier = LogisticRegression()
classifier.fit(X_train, y_train)
```

A.3.2 Using Count Vectorizer

```
vectorizer.set_params(stop_words=my_stop_words, max_features=n,
ngram_range=ngram_range)
checker_pipeline = Pipeline([
    ('vectorizer', CountVectorizer()),
    ('classifier', LogisticRegression())])
sentiment_fit = checker_pipeline.fit(x_train, y_train)
y_pred = sentiment_fit.predict(x_test)
```

A.3.3 Using TDIF Vectorizer

```
model_ug_tfidf = TfidfVectorizer(max_features=1000,ngram_range=(1, 3))
x_train_tfidf = model_ug_tfidf.transform(x_train)
x_test_tfidf = model_ug_tfidf.transform(x_test).toarray()
clf = LogisticRegression()
clf.fit(x_train_tfidf, y_train)
```

A.3.4 Using Doc2Vec

Using DBOW:

```
cores = multiprocessing.cpu_count()
model_ug_dbow = Doc2Vec(dm=0, size=100, negative=5, min_count=2, workers=cores,
alpha=0.065, min_alpha=0.065)
model_ug_dbow.build_vocab([x for x in tqdm(all_x_w2v)])

for epoch in range(100):
    model_ug_dbow.train(utils.shuffle([x for x in tqdm(all_x_w2v)]),
```

```
    total_examples=len(all_x_w2v), epochs=1)
    model_ug_dbow.alpha -= 0.002
    model_ug_dbow.min_alpha = model_ug_dbow.alpha
train_vecs_dbow = get_vectors(model_ug_dbow, x_train, 100)
test_vecs_dbow = get_vectors(model_ug_dbow, x_test, 100)

clf = LogisticRegression()
clf.fit(train_vecs_dbow, y_train)
```

Using DMC:

```
model_ug_dmc = Doc2Vec(dm=1, dm_concat=1, size=100, window=5,
negative=5, min_count=2, workers=cores, alpha=0.085, min_alpha=0.065)
model_ug_dmc.build_vocab([x for x in tqdm(all_x_w2v)])

for epoch in range(100):
    model_ug_dmc.train(utils.shuffle([x for x in tqdm(all_x_w2v)]),
    total_examples=len(all_x_w2v), epochs=10)
    model_ug_dmc.alpha -= 0.002
    model_ug_dmc.min_alpha = model_ug_dmc.alpha

train_vecs_dmc = get_vectors(model_ug_dmc, x_train, 100)
test_vecs_dmc = get_vectors(model_ug_dmc, x_test, 100)

clf = LogisticRegression()
clf.fit(train_vecs_dbow, y_train)
```

Using DMM:

```
cores = multiprocessing.cpu_count()
model_ug_dmm = Doc2Vec(dm=1, dm_mean=1, size=100, window=2,
negative=5, min_count=2, workers=cores, alpha=0.095, min_alpha=0.065)
model_ug_dmm.build_vocab([x for x in tqdm(all_x_w2v)])

for epoch in range(100):
    model_ug_dmm.train(utils.shuffle([x for x in tqdm(all_x_w2v)]),
    total_examples=len(all_x_w2v), epochs=10)
    model_ug_dmm.alpha -= 0.002
    model_ug_dmm.min_alpha = model_ug_dmm.alpha

train_vecs_dmm = get_vectors(model_ug_dmm, x_train, 100)
test_vecs_dmm = get_vectors(model_ug_dmm, x_test, 100)

clf = LogisticRegression()
```

```
clf.fit(train_vecs_dbow, y_train)
```

A.4 Neural network

We won't be sharing any code snippets for the neural networks we have implemented.