A Project Report on

INVESTIGATE SENTIMENTAL DATA FROM TWITTER FOR BUSINESS STRATEGIES USING MACHINE LEARNING

Submitted in partial fulfillment of the requirements for the award of the degree in

BACHELOR OF TECHNOLOGY

IN

DATA SCIENCE

About to Submit By

B. YAMINI	**20KP1A4404**
T. VISHNU VARDHAN	**20KP1A4456**
P. SRI SOWMYA	**20KP1A4437**
SK. IRFAN	**20KP1A4447**

Under the Esteemed Guidance of

Ms. P. Bushra Anjum, M. Tech

ASSISTANT PROFESSOR

DEPARTMENT OF DATA SCIENCE

NRI INSTITUTE OF TECHNOLOGY

(APPROVED BY AICTE, AFFILICIATED TO JNTUK, KAKINADA)

VISADALA, PERECHARLA, GUNTUR -522438

2020-2024

NRI INSTITUTE OF TECHNOLOGY

(Approved by AICTE & Affiliated to JNTU- Kakinada)

DEPARTMENT OF DATA SCIENCE

CERTIFICATE

This certifies that **B. Yamini (20KP1A4404), T. Vishnu Vardhan (20KP1A4456), P. Sri Sowmya (20KP1A4437), and SK. Irfan (20KP1A4447)** are working on the project "**INVESTIGATE SENTIMENTAL DATA FROM TWITTER FOR BUSINESS STRATEGIES USING MACHINE LEARNING**" as a partial fulfilment of the requirements for the award of a Bachelor of Technology degree in the Department of Data Science , NRI Institute of Technology, which is associated with Jawaharlal Nehru Technology University, Kakinada. This is a record of genuine work done by them under my direction and supervision during the 2020–2024 academic year.

Guide
Ms. P. Bushra Anjum (M. Tech)
Assistant Professor
Department of Data Science

Head of the Department (HOD)
Mr. D. Koteswara Rao(M.Tech)
Assistant Professor
Department of Data Science

Internal Examiner

External Examiner

DECLARATION

This declaration certifies that the project report, " **INVESTIGATE SENTIMENTAL DATA FROM TWITTER FOR BUSINESS STRATEGIES USING MACHINE LEARNING**" completed under the direction of **MR. D. KOTESWARA RAO**, partially satisfies the requirements needed to be awarded a Bachelor of Technology degree in Data science. This is a record of the legitimate work we completed, and none of the outcomes included in this project have been duplicated or copied from other sources. No other university or institute has received the project's results in exchange for the awarding of any other degree or diploma.

NRI INSTITUTE OF TECHNOLOGY

(Approved by AICTE & Affiliated to JNTU- Kakinada)

DEPARTMENT OF DATA SCIENCE

Department Vision:

To create a first-class academic department that trains the next generations of students to become globally competant Data Scientists, Data Analysts and Researchers by providing total solution in social aspects.

Department Mission:

M1.To develop industry conducive environment by providing state-of-art infrastructure to compete in data-driven world.

M2.To empower students to provide innovative and cognitive solutions with the help of data analytical skillset and new advancements in high performance computing in the field of data science.

M3. To build data intensive system through socio-economic aspect by promoting cross-disciplinary thinking that expands expertise in cutting edge technologies and acquire professional ethics

NRI INSTITUTE OF TECHNOLOGY

(Approved by AICTE & Affiliated to JNTU- Kakinada)

DEPARTMENT OF DATA SCIENCE

PEOs for DATASCIENCE

PEO1 Graduates will have solid basics in Mathematics, Programming, Machine Learning, Artificial Intelligence and Data Science fundamentals and advancements to solve technical problems.

PEO2 Graduates will have the capability to apply their knowledge and skills acquired to solve the issues in real world Data Science sector and to develop feasible and viable systems.

PEO3 Graduates will have the potential to participate in life-long learning through the successful completion of advanced degrees, continuing education, certifications and/or other professional developments.

PEO4 Graduates will have the ability to apply the gained knowledge to improve the society ensuring ethical and moral values.

PSOs for DATASCIENCE

PSO1 Data Science graduates are able to become leaders in the Industry and Academia with the help of advanced knowledge and skill, which can empower them to analyse, design, develop and implement their learning to develop the society.

PSO2 The ability to develop skills to address and solve social and environmental problem with ethics and perform multidisciplinary projects with advance technologies and tools.

Abstract

Businesses are in high demand for Sentiment Analysis research since it allows them to quickly discover how consumers perceive their brand and make necessary adjustments to their business plans. Using modelling techniques and 70,000 tweets, this research attempts to assess the sentiment polarity of relevant tweets. Natural language processing and word embeddings are used in tandem to successfully train and classify tweets with positive outcomes. Changes in trainable data showed that eliminating stopwords may reduce the precision of the categorization.

A thorough pre-processing stage arrangement is intended to gradually make the tweets more comprehensible for standard language handling techniques. Given that every dataset example consists of two tweets and a sentiment. So, machine learning under supervision is applied.

Furthermore, Random Forest, logistic regression, and support vector machine-based sentiment analysis models are suggested. The primary goal is to more effectively break down emotions. Positive and negative sentiment are the two categories into which tweets are categorised in Twitter sentiment analysis. The random forest method attains a maximum accuracy of 95%.

ACKNOWLEDGEMENT

We would like to sincerely thank our prestigious institution, the **"NRI INSTITUTE OF TECHNOLOGY,"** for giving us the chance to our long-held dream.

We are deeply grateful to the principal, **Prof. Kota. Srinivasu,** for his insightful recommendations that helped the course be completed on time.

We are grateful to Professor **D. KOTESWARA RAO,** Head of the Department of CSD, for his unwavering support and for making time in his busy schedule for us.

We would like to express our sincere gratitude to our project guide, **Ms. P. Bushra Anjum,** for her invaluable advice and unwavering support, without which we could not have succeeded in this endeavour.

We are grateful to the entire teaching and non-teaching personnel of the Data Science department for their invaluable assistance in making this project a success.

Project Members

B. YAMINI (20KP1A4404)

T. VISHNU VARDHAN (20KP1A4456)

P. SRI SOWMYA (20KP1A4437)

SK. IRFAN (20KP1A4447)

INDEX

CONTENTS **PAGE NO**

CHAPTER-1

1. Introduction

This project will examine the entire development process, provide a thorough specification, talk about the background information, and evaluate how the project was carried out. The report will also examine the project's technological evolution, outlining important phases and provide analysis of choices made as well as results from testing any algorithmic models developed during the process.

The main goal of the Project Initiation Document (PID) was to categorise texts in order to use modelling approaches to comprehend the sentiment polarity. Since the project's inception, there haven't been many modifications to the PID, thus the analysis and creation of models to address the research issue have continued to take centre stage.

1.1 Background Context

With the advent of deep learning techniques more recently, natural language processing, or NLP, is a processing technique that aims to demonstrate how computers can comprehend human language. With the use of natural language processing (NLP), computers can now read text, listen to voice, and analyse it by identifying key phrases and sentiments.A well-liked mainstay in the field of deep learning is sentiment analysis. Before social media sources were introduced, the primary source of commentary was always movie reviews.

Natural language processing (NLP) can assist companies and organisations in comprehending the sentiment of their clientele through comments left on well-known social media platforms like Facebook and Twitter by utilising TensorFlow and Keras.

There is a 140 character limit on Twitter for users to write on at once. Because of this restriction, there is ambiguity regarding the display of sentences on this blogging platform. This affects the usage of emojis like vand the abbreviations and uncommon phrases like lol and aka.

Results can vary even though there are numerous studies that support and refute the removal of stopwords and collection words. This will be investigated in this essay to determine whether the impacts are significant enough to bolster the case for their removal. Stopwords are often removed using a pre-compiled stop list; in this instance, the NLTK package was utilised to accomplish this.

1.2 Initial Project Background

Since Twitter was founded in 2006, it has been increasingly popular, making it the primary application for gathering data for this study. Even though movie reviews were the primary corpus utilised in the early articles on sentiment analysis, more recent research has indicated that social media is a better source of data for opinion mining and would therefore be a better option to pursue.

The project's goal is to take machine learning methods and apply them to deep learning scenarios with large amounts of data. Using the Tweetpy tool to access Twitter's API, 65,000 tweets were taken from the social media platform for this study.

Sentiment analysis can benefit from the application of machine learning since it has several fundamental advantages, the primary one being the various learning foundations it provides. By using machine learning, data analysis may be done with considerably more efficiency and precision. For this, a computational framework is perfect since it allows sentiment analysis and opinion mining to be tailored to the specific topic.

Analysing unigrams' effects in relation to polarity was the goal of this study. Feelings A sentence's polarity refers to its orientation and the emotion it expresses; it might be positive, negative, or neutral. Understanding the sentiment orientation of text in written or spoken language is crucial, but this isn't always the case because slang and sarcasm can make it so that sometimes even we humans find it difficult to fully comprehend the feeling of what we are reading or hearing.

1.3 Ethical Issues

Actual data taken from the Twitter API is used in this project. Even though the data came from a social media platform, it is all anonymous. Although it was possible to track who wrote each tweet at the time of data scraping, all instances of who wrote what and where it came from were not extracted along with the data, making it impossible to link the data to a specific person. Prior ethical approval was obtained for data extraction, handling, and preparation.

1.4 Aim and Objectives

1.4.1 Objective 1: Classify Textual Data through Modelling Techniques

This project's primary goal was to categorise information from the social media site Twitter. Using a Python library, data was scraped from the Twitter API. After that, it was cleared of any URLs and hashtags to ensure that the data remained anonymous and contained no linkages to the tweet posters. Two iterations of the well-known Deep Learning Neural Network models were developed using the Keras toolkit with the goal of achieving high accuracy numbers for text classification.

In order to enable the model to comprehend and match words to distinct polarities—Positive, Negative, and Neutral sentiment—polarity scores are acquired using a Python library.Modelling techniques enable supervised learning to produce desired outcomes and offer a framework for assessing findings' sensitivity, accuracy, and recall.

1.4.2 Objective 2: Model Investigation

The research and assessment of models was the project's secondary goal. Nine trained models with three different versions of the same data would be retained while numerous hyper- parameter possibilities were experimented with and altered. This would allow comparison of elements that had contributed to previous works regarding the issue.

1.4.3 Objective 3: Model Evaluation

Common visualisation approaches are used for evaluation, such as confusion matrices to illustrate error metrics and graphs to demonstrate the model's training process.

It was crucial that all models be run with identical hyper-parameters and layers in order to compare how well each model learns from the modified training data. This makes it possible to observe how the removal—or lack thereof—affects each model's ability to generalise and how quickly it can learn from the texts it encounters.

1.5 Research question

"What are some ways to analyse the sentiment of Twitter data using modelling techniques?"The project's goal was to discover how machine learning was applied to the analysis of noisy social media data by using modelling techniques made possible by the Keras toolkit. These methods will be examined, models created, and evaluation performed to help comprehend how using these methods helps identify sentiment in noisy data.

When cleaning the social media data, there are several noises that must be taken into account. Numbers would remain in any text for the sake of this study, but other elements like hashtags and URLs would be eliminated. In addition to complying with GDPR and preserving anonymity within the data being utilised, this was done to ensure that the data could not be linked back to any specific person as that was not the aim of the study. External-facing URLs don't improve the data in any way either.

1.6 Report Structure

This report's remaining sections are organised as follows. The second section, the Literature Review, explores the history of the discipline by looking at the early examples of Computational Linguistics, the origins of AI, Text Classification, Sentiment Analysis, and the field's current state. In Section 3, the project's development is covered, along with details on how the

data was acquired, cleansed, and tokenized for use on the networks.

In Section 4, the project's implementation is covered, including the implementation approach and how it was helpful throughout, the results obtained, and how those recommended.

The outcomes were assessed in more detail. The project is critically evaluated in Section 5, along with a reflection on the results and output. This section also seeks to assess future directions for this field of study as well as ways in which the research may be strengthened.

1.7 Proposed System

The system plans to use tweets collected from the Twitter dataset for sentiment analysis. The best algorithm has been selected after a number of algorithms have been used and evaluated against the dataset that is available. An overview of the sentiment analysis process is provided in Figure 1. The dataset will be pre-processed using the methods listed below once it has been cleaned and separated (isolated) into training and testing datasets. In order to decrease the dataset's dimension, features will be extracted. The next step is to develop a model that the classifier will use to distinguish between positive and negative tweets.

The classifier will be fed real-time tweets once again in order to test the real-time data. Not every tweet from every other site is subjected to sentiment analysis by the suggested system. Sentiment analysis is used to categorise tweets about products on the market into positive or negative categories inside a highly domain-restricted system. A graphical user interface (GUI) will be supplied to the end-user, enabling them to input keywords or sentences associated with a certain product. The user will be able to view every tweet associated with that product. The quantity of comments posted by other people—both favourable and negative—will be visible to the user. This will assist them in updating their work and production plans.

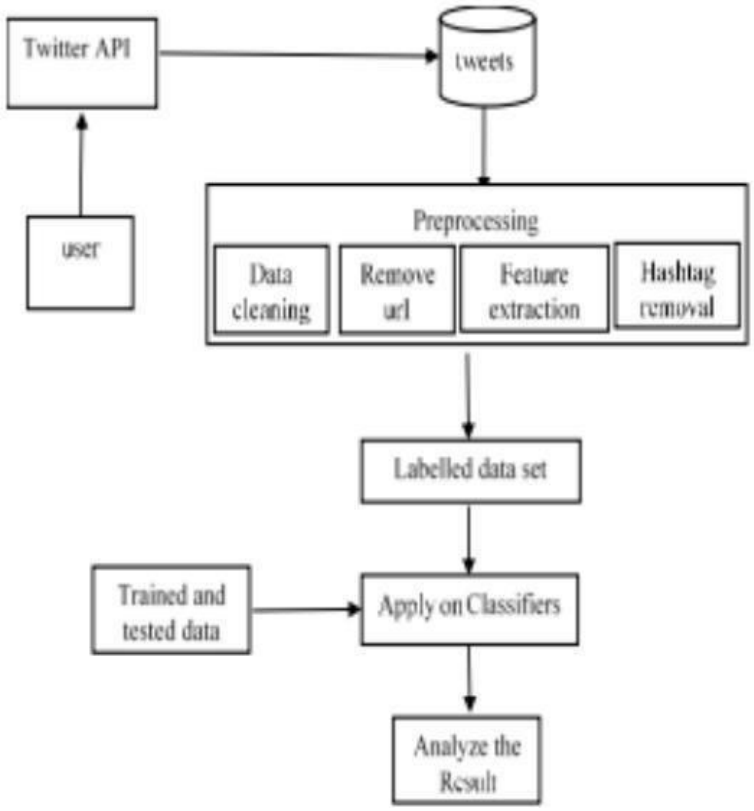


Fig:1 Architecture of Sentiment Analysis

CHAPTER-2

2. Literature Reviw

The goal of this section is to go over the technical background of the subject area, including everything from text classification to sentiment analysis to the early days of natural language processing (NLP). This section, which is divided into three sections, aims to provide background information for the study by discussing the research that has been done in the subject area thus far.

2.1Technical Background

The goal of artificial intelligence (AI) research is to show how machines may be intelligent in contrast to humans and other animals. The goal of artificial intelligence (AI) is to take what humans have been doing for thousands of years, comprehend how we think, and demonstrate how we can utilise this understanding to influence a world that is considerably bigger and more complex than we first realised. Though research in this field started after World War II, the phrase artificial intelligence was first used in 1956. (Norvig and Russell, 2016) In his work "Computing Machinery and Intelligence," published in October 1950, Alan Turing presented one of the earliest approaches to artificial intelligence (AI) aimed at creating robots capable of human-like behaviour.Turing suggested a test known as the "Imitation Game," which was eventually dubbed the Turing Test. It involved three players: an interrogator (C), a woman (B), and a male (A). This test was designed to provide an answer to the question, "Can machines think?" The goal of the game was to see if the interrogator could tell which person was a woman and which was a guy based just on how they answered questions. The interrogator gave them the names X and Y. The interrogator was unable to determine the identity of the respondent merely by listening to the tone of voice, for example, as the questions were typewritten.

In the first testing, one person acted as A and another as B, supporting the interrogator C. However, the concern arose as to what would happen if the person posing as A was later swapped out for A.

This brought up a fresh issue. It was crucial to modify the questions posed

to the computer in order to prevent the machine's answers from revealing that it was a machine rather than a person responding.

The test's objective was to see whether a computer could carry on a conversation using a voiceprinter. If the computer succeeded, it would have demonstrated its capacity to mimic human speech patterns effectively enough to be mistaken for a sentient creature. If the interrogator cannot determine from the respondent's statements whether or not it is a computer, then the machine passes the Turing Test. A computer needs to be able to do specific tasks in order to stand a chance of passing such an exam. These include machine learning, automated reasoning, knowledge representation, and natural language processing.

The Turing Test involved no physical interaction between the interrogator and the subject they wereasking questions to. This was because Turing deemed physical simulation as something unnecessary for intelligence, but the test was expanded upon, known as Total Turing Test, that involved objects being sent through a hatch to the participant, with the addition of a video signal. This was done to allow the interrogator to test the perceptual abilities of the participant.

The Total Turing Test would therefore require a computer to have the capabilities knownas Computer Vision and Robotics. These 6 necessary capabilities are the 6 disciplines that AI is composed of. All these capabilities are known as the area of AI that allow a computer to be able toAct Humanly and for this, we owe Turing a lot of credit.

2.2 Natural Language Processing

Computational Linguistics

The field of artificial intelligence (AI) and computational linguistics known as "natural language processing" (NLP) aims to analyse and analyse meaningful sentences and phrases that appear in natural language. Ferdinand de Saussure, a scientist, conducted the first research in linguistics in the early 1900s and developed the idea of language as a science.

Despite the fact that Saussure passed away before his research was

published, two of his colleagues did write a 1916 summary of his findings.

How does language connect to thought is the issue that the study of linguistics aimed to answer.Noam Chomsky outlined his thesis in 1957 in his little book "Syntactic Structures." Although Chomsky's theory was grounded in syntactic models that date back to the Indian linguist Panini, it was formal enough to enable the development of programmes that could be used to test the theory.In his work, Chomsky covered the concept of phrase structures and how humans communicate through natural language. Constituent analysis is used to formulate linguistic description at the syntactic level.A sentence represented physically as an inverted tree, with each node labelled with the phrasal ingredient it represents, is called a phrase structure tree.Phrase structure evolved into a kind of grammar that made it possible for computers to comprehend languages, although this kind of grammar structure had to be developed in order to accomplish this.The ultimate objective of this new language structure was to build a computer that could mimic the functions of the human brain, including thought and speech.However, there needs to be some thought given to the use of language in resolving generalisation problems inside models. Adjectives typically appear before nouns in English, but what about in a language like French, where this isn't the case?Phrase structure is helpful, but it has limitations in that a computer's ability to comprehend the significance of a structure may be hindered by variations in one language's structure when it comes to another language.

The first step in teaching a computer to speak or understand human language is to use these grammatical models of sentence structure to teach the machine the correct order in which to speak the language. While it is helpful to think about a sentence's structure, it is equally helpful to comprehend the syntactic categories that result from this structure.

A sentence's syntax is the way its words are arranged (What is Syntax? Definition, English Syntax Examples, n.d. This is significant because, in addition to making the phrase comprehensible to a human speaker of the language, the sentence's structure is necessary for a machine to be able to construct sentences with the same structure. Although syntax is essential,

diction is also helpful.

In essence, the Diction describes word choice; the pairing of the two can result in sentences with comparable meanings but distinct structures despite utilising the same words. For instance, "The Dog Barked Loudly" and "The Dog Barked Loudly." Even if the content of the two words is the same, each phrase's syntax differs from the other.

2.2.1 Machine Translation

The most common usage for machine translation, which is an automatic text translation process, is to convert one language into another. Warren Weaver first proposed it in his article in 1949. "More than just pointing out the obvious—that linguistic diversity seriously hinders cross-cultural communication and deters international understanding—needs to be said." Making sure we get a word's underlying meaning is a challenge in translating. Sometimes it's impossible to determine whether a word actually means what it says on the page.For instance, if we try to read words through an opaque mask that has just a little hole in it, we won't be able to see the word next to the one we are reading. It's difficult to determine which meaning a word like "fast" conveys—"rapid" or "motionless."It was at this point in the 1960s that the necessity of machine translation began to be discussed. In order to address the necessity of machine translation at the time, a group of scientists known as ALPAC (Automatic Language Processing Advisory Committee) published a paper titled "Languages and Machines" in the early 1960s. The study came to the conclusion that human translators were more efficient and less expensive than machine translation.It was evident by the middle of the 1960s that machine translation was not working, as stated succinctly in the report "There is no emergency in the field of translation." The issue is not trying to solve a hypothetical demand with a hypothetical machine translation.

2.2.3. Text Analytics

After the publication of the ALPAC report, many people thought that NLP research was no longer worthwhile. However, in the 1980s, this began to change. AI research started to pick back up steam with the advent of expert

systems and the deep learning research paradigm.It was conceivable to start NLP research again with the launch and rise in popularity of these new machine learning techniques that combined linguistics and statistics, although this time the approach was more statistical than machine translation.The primary issue at hand was the unstructured nature of the data, which made exploration challenging. Text analytics is the solution to this issue. Machine translation work started in the 1980s is continued with Text Classification (TC), also called Text Categorization. "TC began in the early 1960s, but it wasn't until the early 1990s that it gained prominence as a significant area within the information systems discipline." The task of text classification (TC) involves selecting predetermined classifications (or themes) to which a given text may belong.

In Boolean form, the technique of assigning values to each text involves taking a set of predefined categories and using true and false values to allocate a domain of documents to each of them.Text classification used to exist in a slightly different form in the past. Early text categorization research included email spam filtering, or "Ham vs. Spam." This is an issue with supervised learning, a type of machine learning where learning is developed under human supervision.

2.2.4 Sentiment Analysis

In 2002, the field of research on sentiment analysis began to take off. The first focus of this investigation was the Corpus, which consisted of movie reviews. Utilising these internet ratings aided in introducing the research topic. (Pang, Lee, and Vaithyanathan, 2002) produced the first actual paper based on this, which is among the earliest written publications to demonstrate that machine learning categorization algorithms can surpass human baselines.Texts from social media were not the only source of sentiment analysis. Jack Dorsey, the founder, sent out the first tweet on March 21st, 2006 (Abell, 2011). With Social Media's increasing popularity, it became clear that the tone of these websites would develop into a space where the subject might be improved. Through techniques like keyword mapping, text mining makes it possible to identify both new and preexisting relationships and trends

across a variety of unstructured materials (Aggarwal and Wang, 2011). This subject is covered in more detail starting in section 2.3.4.

2.3 Previous Research

The objective of this part is to examine the literature on text categorization, which ultimately brought us to the study of sentiment analysis.

2.3.1 Inverse Document Frequency (IDF)

Although others did not use the term "inverse document frequency" for Jones' study until a little while after she completed it, it was the subject of one of her papers. Jones desired toExamine the specificity of text index words to see whether it made sense to treat them as separate entities. She obtained phrase frequency variations and analysed them to determine whether termsthat appeared more frequently offered a chance to boost performance as a whole. She discovered that matching with term weighting produced significantly better results than matching with a simpler term.

2.3.2 APRIORI Algorithm

Among the first to investigate the impact of n-grams on text classification was Fürnkranz. In his research, n-grams characteristics for text categorization were investigated in two selected domains. Machine learning techniques were already in use at the time this paper was written, demonstrating its value in representing training data as feature vectors. Fürnkranz generates the characteristics in this work using an algorithm akin to APRIORI. An important data mining algorithm for Market Basket Analysis, which mines recurrent itemsets and their occurrence, is the Apriori algorithm. The algorithm locates feature pairs, determines where the position pointers of the two features meet for each pair, and then eliminates any features that are below the threshold indicated by the associated position pointers. In this particular experimental setting, n-grams of lengths ranging from one to five are used, and the number of n-grams in the data is reduced by the use of a stoplist, which is a list of predefined stopwords. The usage or absence of stopwords is a contentious topic in text analytics; Fürnkranz notes that eliminating stopwords may result in the loss of certain crucial data. In the two

domains they were dealing with—the 20-newsgroup data and the 21578 REUTERS newswire data—this technique and the efficient use of pruning with n-grams characteristics produced very high accuracy.

2.3.3 Bag-of-Words and N-grams

Mladenic also used bag-of-words and n-grams (see figure 2) in a new algorithm that he developed that year in an attempt to create an algorithm that could effectively classify text based on these attributes. Despite appearing promising, the Naïve Bayes method used for this didn't yield very good results; the average recall score was only 0.45 (Mladenic, 1998). according to recall ratings (Mladenic, 1998).

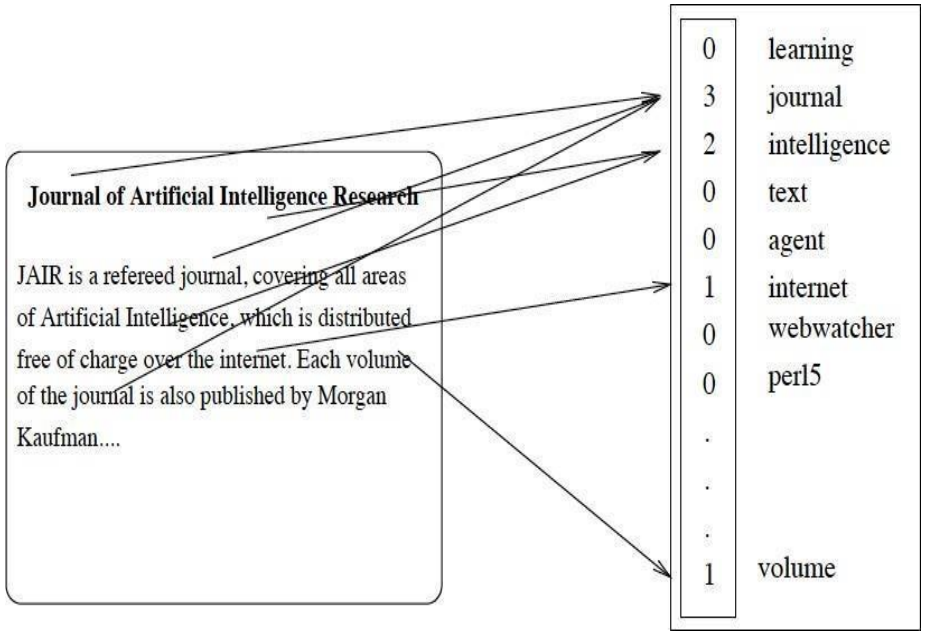


Fig:3 Mladenic 1998 – Example of Bag-of-words representation using frequency word-vector

Bigrams, or two-word phrases, have been used in certain research to improve text categorization.The authors of this paper are Tan, Wang, and Lee. They suggested that the best features for a Naïve Bayes classifier would be produced by an effective text categorization algorithm that creates bigrams

from the data. Tan, Wang, and Lee also explored with n-grams of lengths 2 and 3, as Fürnkranz had found that these were the optimal lengths for text categorization. They developed a method for bigram extraction that yields bigrams with high information gain and high domain occurrence.They found that the performance of their experiment was greatly improved by the usage of bigrams.

2.3.4 Sentiment Classification – New Beginnings

In their research report, Pang, Lee, and Vaithyanathan titled it "Thumbs Up? Sentiment Analysis research in the discipline is often regarded as starting with "Sentiment Classification using Machine Learning Techniques." The publication year of this study is significant; although it was published in 2002, sentiment analysis as a field did not really take off until 2003, mostly due to the contributions made in this work (Pang, Lee, and Vaithyanathan, 2002).

Text categorization work had been done on a few domains before to Pang and Lee ib id, but their work was among the first to employ movie reviews as the corpus for their research (a corpus is a domain of data). Because there is a lot of data in this domain that can be analysed online, they have outlined how to use it. They clarified how simple it is for a person to discern between a positive and a bad review when reading the reviews, citing instances in which they felt specific terms would tend to convey strong opinions. Two graduate students were asked to select, on their own, good and negative indicator words that they would anticipate seeing in movie reviews.Their baseline findings revealed accuracy rates of 58% and 64% for the 700 positive and 700 negative reviews, respectively. 69% of the findings were achieved by taking the words of another human and merging them with statistics from the domain. In their experiments, they looked into three different machine learning techniques. These included Support Vector Machines (SVM), Maximum Entropy, and Naïve Bayes.They got good results by combining the three approaches with three-fold cross-validation. The machine learning techniques with unigrams, unigrams with bigrams, unigrams with POS (Part-of- Speech tags), and adjectives and unigrams employing placement were all combined in their research. The foundational elements of a language, such as nouns and

adjectives, are known as POS tags; this was covered in the section on computational linguistics earlier.

SVM was the machine learning technique that performed the best in the majority of their tests; when assessing whether or not unigrams were present in the text, SVM produced the best results. Stopwords are a crucial term for this study; they were left in and still produced accuracy levels as high as 82%. It was noted that more effort needed to be done to better understand the subject of the reviews and determine whether doing so will increase accuracy metrics.

2.3.5 Subjectivity Extraction

A few years later, Pang and Lee published an extension of their results in their subsequent research report. This research was inspired by the idea that higher base accuracy outcomes may be obtained by using minimum cuts in sentiment summarization (Pang and Lee, 2004).

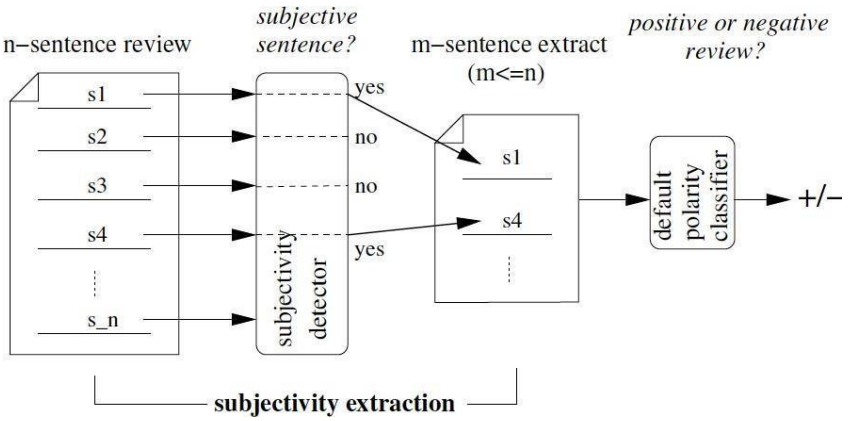
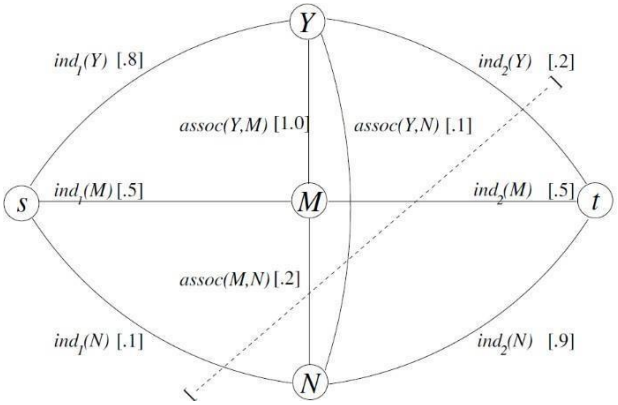


Figure 4 - Pang and Lee 2004 – Polarity Classification via Subjectivity Detection

In order to better comprehend the sentiment of the text, this study developed the concept of polarity and started to investigate the subjectivity of the text and how it affects the perceived polarity of the text. Previous

approaches have examined lexicons, such as "good" or "bad," however examining lexiconic words alone will not fully explain the polarity of texts. Additional research is needed. The subjectivity extraction procedure is illustrated in Figure 4. Their strategy was to categorise the writings as subjective or objective, then take out the objective texts and use a machine learning classifier on the extracts that were left. In place of feature engineering, as had been done previously, they devised a different approach. They employed individual scores and association scores in conjunction with the concept of cut-based classification to extract information from the texts.



C_1	Individual penalties	Association penalties	Cost
{Y,M}	$.2+.5+.1$	$.1+.2$	1.1
(none)	$.8+.5+.1$	0	1.4
{Y,M,N}	$.2+.5+.9$	0	1.6
{Y}	$.2+.5+.1$	$1.0+.1$	1.9
{N}	$.8+.5+.9$	$.1+.2$	2.5
{M}	$.8+.5+.1$	$1.0+.2$	2.6
{Y,N}	$.2+.5+.9$	$1.0+.2$	2.8
{M,N}	$.8+.5+.9$	$1.0+.1$	3.3

Figure 5- Pang and Lee 2004 – Graph for classifying three items, table shows examples of how cuts were made

Example values are enclosed in brackets; in this case, the individual scores are probabilities. We would place Y ("yes") in C1, N ("no") in C2, and M ("maybe") in C2 based just on individual scores. However, as the table illustrates, the association scores favour cuts that place Y and M in the same class. Consequently, M and Y are placed in C1 by the minimum cut, which is represented by the dashed line. Figure 4 (Pang and Lee, 2004).Using an additional subjectivity dataset to train their detectors, Pang and Lee ib id stayed with movie reviews as their corpus. They were able to extract and train on a more subjective dataset to provide sentiment analysis results by using

their default polarity classifiers and upgrading them later with the subjectivity dataset. They employed SVM and Naïve Bayes as their preferred machine learning models, and they found that adding subjectivity instead of polarity increased accuracy by, on average, two to five percent.

2.3.6 Continuous Space Use

Schwenk examined the concept of "Continuous Space language models," which classified spoken language using neural networks. "Attacking the data sparseness problem by performing the language model probability estimation in a continuous space" was the stated goal of this study. In order to improve the model's performance, three options were examined and addressed in the study. These included using huge projection layers, training many networks and then interpolating them together, and growing the size of the hidden layer (Schwenk, 2007).

2.3.7 Using Social Media for the task

Barbosa and Feng categorised the subjectivity of tweets (Twitter posts) using standard elements plus a few Twitter-specific cues like retweets, hashtags, links, capital letters, emoticons, question and exclamation marks. This was accomplished using a two-step sentiment analysis classification approach that first divides messages into categories based on their polarity— positive or negative— before classifying them as subjective or objective, as was previously seen (Barbosa and Feng, 2010). The objective is to use a robust classification detection method that can detect and remain effective across noisy or biassed data to extract sentiment from texts on Twitter. Data with a lot of information that isn't helpful for categorising the sentiment is referred to as noisy data. Their methodology aimed to do a similar task as that of Pang and Lee, who classified tweets as either positive or negative and then included the option of being classified as neutral.

Barbosa and Feng concentrated on and worked on considerably smaller pieces of texts, usually unigrams or bigrams (length of 2). Pang and Lee, on the other hand, completed their categorization work on larger pieces of texts that required the use of n-grams up to a length of 5. The idea behind the paper

is that specific POS tags are helpful and good indicators for sentiment tagging. To build upon the use of these types of tags that had shown some encouraging results in earlier publications, the POS dictionary taken from http://wordlist.sourceforge.net/pos-readme is used in the paper.They used data from three different sources, with the largest sum of tweets (254081), albeit there is a low probability that all of these are unique because retweets are included.Here, the training features were created using the syntax aspects of the tweets. A polarity classifier was the most significant classifier they employed.Though typically noisy, this work was a promising first step towards sentiment analysis of social media data. However, the findings obtained were good, if not state of the art, when compared to those obtained from more coherent, clean data derived from movie reviews.

CHAPTER-3

3 Technical Development

This section aims to elucidate the steps involved in the data analysis process, including the exploration of extraction and tokenization. These are the instruments required to complete this project, arranged in a methodical manner that describes their individual requirements.

3.1 Narrative Description

The project itself required the developer to learn a lot of topics that were unfamiliar to them before they started, in order to make meaningful progress towards the project's completion. Many of the topics were not covered in the first two years of the University of Hull's BSc Computer Science programme, which at first raised concerns about whether or not the project could be finished on schedule. This project required the learning of significant techniques, some of which were found during a summer school the developer attended overseas beforehand and in the Data Mining & Decision Systems module that was part of the year 3 criteria. These discoveries helped to fill in the gaps regarding the operation of machine learning algorithms. The developer was lucky to have selected this module ahead of time, since it allowed them to learn a lot of skills during the project's initial planning phase. This module made it easier to develop the various concepts that were needed. The developer discovered that the most helpful aspect of comprehending the concept of Neural Networks was the inclusion of a real-world machine learning classification challenge. Together with the experience from self-study, the developer's summer school education gave him a set of abilities that let him overcome problems and produce a product that far exceeded his accuracy goals.

3.2 System Requirements

3.2.1 A number of Python language libraries are used in the project's development. Because the project was a research topic, libraries that accelerate the processing of specific aspects are utilised in the software along with libraries that,

because of their capabilities, help to increase workload and produce high-quality results. There isn't a stand-alone software solution to go along with the Jupyter Notebook because of the nature of the project.

3.2.2 Python

Python was the programming language utilised for the project. This was caused by the abundance of excellent libraries available for Deep Learning and Machine Learning, as well as the language's natural ease of use for the job. The framework's library selection starts with some well-known ones, like Numpy, Pandas, and Tensorflow. These libraries make it possible to facilitate clear code and made it simple to follow along with the coding portion of the project. Additionally, the language's ease of use contributed to its popularity for this task. TensorFlow (2019), pandas (Python Data Analysis Library, 2019), and NumPy (2019)

3.2.3 Jupyter Notebook

The codebase of choice was Python, however an IDE was also suggested. Jupyter Notebook is an open-source web tool that facilitates coding in blocks, making code organisation and execution easy to understand. Although it was possible to preserve results because the code block containing them didn't need to be run again later, it was helpful to be able to run some parts of the code while leaving other parts unfinished. This allowed a model to be trained and the results to be seen.

```
In [263]: def remove_url(txt):
              #Replace the URL with nothing but a space
              """Replace URLs found in a text string with nothing
              (i.e. it will remove the URL from the string).

              Parameters
              ----------
              txt : string
                  A text string that you want to parse and remove urls.

              Returns
              -------
              The same txt string with url's removed.
              """

              return " ".join(re.sub("([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ", txt).split())

In [264]: def analyse_sentiment(tweet):
              analysis = TextBlob(tweet)

              if analysis.sentiment.polarity > 0:
                  return 1
              elif analysis.sentiment.polarity == 0:
                  return 0
              else:
                  return -1
```

Figure 6 – Visual Example of Jupyter Notebook Code-blocks

Figure 7 illustrates how code chunks could be viewed in a Jupyter notebook. The code number on the left aids in determining the sequence in which the code block was executed. The code blocks in the picture are in the correct sequence, however it is possible to run code out of order, which can occasionally result in issues like errors when variables are declared in code blocks after they are first used. Additionally, code blocks could be copied and pasted, which made it possible to run a different model with slightly different parameters or one with a different dataset without having to rewrite the entire code. As a result, results for multiple models could be displayed.

3.2.4 Tweepy

Data from social media that is noisy was used in this experiment. The project required a huge amount of data that was exclusive to the selected topic and could be obtained within the limits. Brexit was selected as the research topic because it was a hot topic in the United Kingdom. Twitter is a well-known website with millions of tweets posted every day. Sentiment analysis has also been done on this platform in the past. Twitter was used as the source in this instance for this reason. A scraping library, such as Tweepy, would be required in order to gather the data. Access to the Twitter Application Programming Interface (API), which describes how to connect to and use data from the Twitter internal system, was necessary in order to use Tweepy. This

would require developer access, which could only be obtained by submitting an application.

```
#Define your App access codes to access the API
auth = tw.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_key, access_secret)
api = tw.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True)
```

Figure 7 – Defining the Twitter API access credentials

The formatting needed to access the Twitter API and start accepting tweets is displayed in Figure 7. The authorization details (shown in figure 8) obtained by registering an app on the developer area of the website are the most crucial element that the developer account provides to the user for the gathering of data. These permission settings were configured to search for tweets about Brexit and filter out any retweets in order to prevent the taking of duplicates. A specific amount of 100,000 tweets was selected to be accepted. After URLs were first eliminated, 65,759 full-length tweets were eventually retrieved and saved to a text file.

3.2.5 Keras and Tensorflow GPU

Since Keras is merely a library that operates on top of Tensorflow, using Keras necessitates Tensorflow. Neural network models can be created with a variety of choices offered by Keras. Because of the substantial volume of data used in this research, Tensorflow GPU was selected as the preferred kernel flavour to run on a Jupyter Notebook. Utilising a GPU's processing capacity is made possible by Tensorflow GPU. A Recurrent Neural Network Sequential model made with Keras is displayed in Figure 9. This summary allowed one to examine the network's size as well as the quantity of predicted trainable parameters that would be added to the network from the dataset.

Figure 8 – A Keras Sequential Model created in Jupyter Notebook

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
layer_embedding (Embedding)  (None, 115, 8)            531384
_________________________________________________________________
gru_1 (GRU)                  (None, 115, 32)           3936
_________________________________________________________________
gru_2 (GRU)                  (None, 115, 16)           2352
_________________________________________________________________
gru_3 (GRU)                  (None, 115, 8)            600
_________________________________________________________________
gru_4 (GRU)                  (None, 4)                 156
_________________________________________________________________
dense_1 (Dense)              (None, 10)                50
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 33
=================================================================
Total params: 538,511
Trainable params: 538,511
Non-trainable params: 0
```

3.3 Data Extraction

This part tries to highlight the interesting results from the data and covers the steps involved in data extraction, exploration, and tokenization.

3.3.1 Exploring the data

Tweepy was used to extract the data from the Twitter API. The data might be saved to a text file after extraction and then further examined. Eliminating components that are not needed is part of the process of transferring the data from the text file to a data table. The cleansing of the data is one of the most important components of data exploration. Since hashtags are useless here, they were eliminated from Twitter, where they are useful for defining trends of the day or week.

Figure Figure 9 illustrates how Fig9 illustratesTwitter trends are defined by hashtags. These might be global trends or ones that only exist in one nation. What trends are displayed to users depends on where they are located. One significant aspect of the retrieved data was that none of it had any polarity associated with it. Sentiment analysis studies mostly used model training to enable recognition and association of words with polarity measures.

Figure 9- Visual example of trends on Twitter via Twitter.com

```
dfObj = pd.DataFrame(brexit)
df = pd.DataFrame(sentiment)

df = df.rename(columns={0: 'polarity'})

brexit_data = pd.concat([dfObj, df], axis=1)
brexit_data = brexit_data.rename(columns={0: 'tweet'})
brexit_data.head()
```

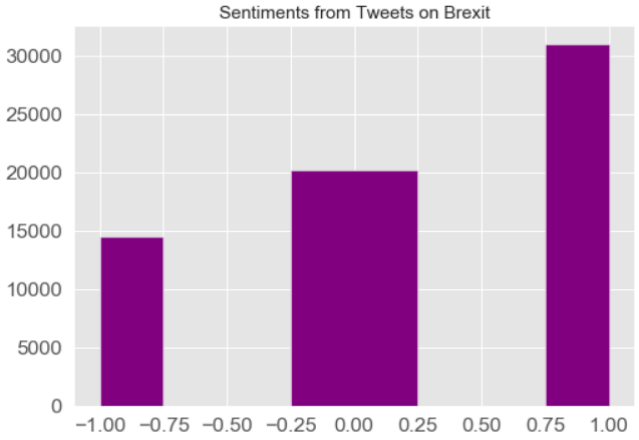
	tweet	polarity
0	If you re a Brit amp you thought Boris amp Bre...	-1
1	AndrewBowie4WAK 14m voted for your Prime Menda...	-1
2	Morte all EUrodittatura Brexit gt Italexit	0
3	Well done Boris We leave the EU 31st Jan 2020 ...	0
4	This Christmas BorisJohnson promises the gift ...	0

table 1 - Creating the Polarity Data table

3.3.2 Exploring Findings

3.3.2.1 Polarity Values

With the 65,759 tweets in total, it was important to have an idea how the polarity of Positive, Negative and Neutral was spread. By implementing a Python value_counts() function onto the data table, it was possible to visualise this.




```
 1    31024
 0    20250
-1    14485
Name: polarity, dtype: int64
```

Figure 10 - Graph showing the spread of Polarity across the data

In figure 10, positive is represented by a value of 1, neutrality is represented by a value of 0, and negativity is represented by a value of -1. With a total of 34,735 tweets, there were more positive polarity tweets than neutral or negative ones. The positive polarised tweets made up 47.17% of the data. There were more tweets with positive polarity overall, as can be seen in Figure 13, an easier- to-read graphical representation of the data dispersed in the form of a bar chart. It was intriguing to observe whether the models' bias when trained on this data would be affected by the nearly 50% dispersion of values in the positive direction.

3.3.2.2 Common Words Amongst Tweets

It was required to investigate the terms included in the tweets in order to determine which words were most frequently used. Every tweet was given a

Polarity value that was determined by analysing the entire tweet, taking into account every word. A tweet's bias may result from its high frequency of negative phrases, which may also cause problems with the polarity assignment process.

Figure 14 displays the numerical counts for the values in Figure 15, and Figure 15 reveals that the word "Brexit" was the second most frequently occurring word across all collected tweets. The fact that this was the "collection word"—the topic word—that was utilised to extract every tweet is what matters in this instance. Brexit replaced the hashtag #Brexit, which had been removed.

The most frequently occurring word was "the," which wasn't particularly helpful for determining a sentence's polarity. However, further investigation revealed that several of the most frequently occurring words were "stopwords." These are the words that connect the language to make it easier to understand, but other than that, they have no true significance.

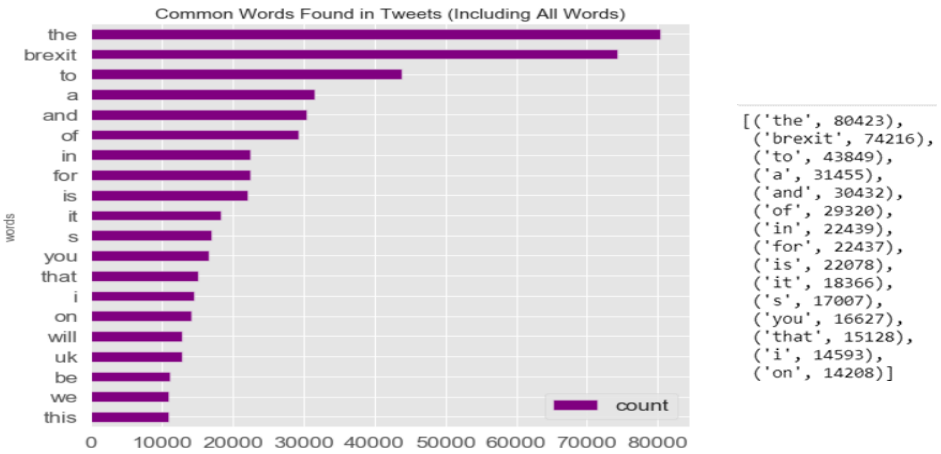


Fig:11 – Representation of common words found in tweets

First, all tweets had to be converted to lowercase, and then each tweet had to be divided into a list based on the spaces between words. This allowed for the collection and exploration of each individual word, such as Pull and pull. Pull and pull would have been regarded as distinct terms because one is capitalised and the other is not if the words had not been turned lowercase

before being divided.

3.3.3 Removing Stopwords & Collection Words

The elimination of stopwords and collection words from the data, as well as the justification for doing so, are covered in this section.

3.3.3.1 The NLTK Package

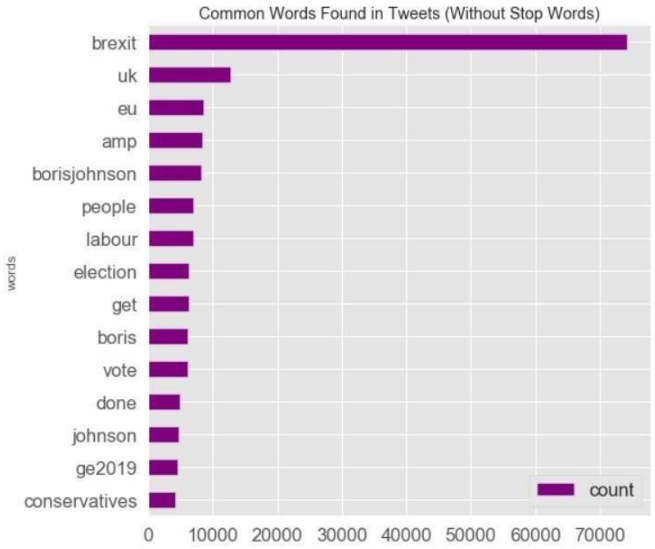
A collection of tools and libraries called the Natural Language Toolkit (NLTK) aids in the process of statistical or symbolic natural language processing. For its Englishstoplist stoplist library, NLTK was utilised. This stoplist's objective was to eliminate every stopword.

3.3.3.2 Using a Stoplist

A stoplist needs to be called from the NLTK library in order to be used. Setting a local variable with the list of all words found in the English stoplist was the first step required to do this. After that, a comparison for loop with an enclosed if statement could be made to check if any terms detected in a tweet were already on the stoplist. If they were, the words would be skipped over and not added back to the original. Figure 13 illustrates this.

```
#Remove the stop words from each tweet list of words
tweets_nsw =[[word for word in tweet_words if not word in stop_words]
             for tweet_words in words_in_tweet]
```

Fig:12 – for loop to remove stopwords




```
[('brexit', 74216),
 ('uk', 12777),
 ('eu', 8505),
 ('amp', 8438),
 ('borisjohnson', 8254),
 ('people', 6994),
 ('labour', 6979),
 ('election', 6200),
 ('get', 6191),
 ('boris', 6144),
 ('vote', 6040),
 ('done', 4862),
 ('johnson', 4613),
 ('ge2019', 4566),
 ('conservatives', 4104)]
```

Figure 13 - Representation of common words found in tweets,stopwords now removed

Figure 13 displays the outcomes of eliminating stopwords from every tweet, and Figure 14 displays the numerical values of Figure 14. The word Brexit continued to be heavily represented as the most common word by a significant margin, and it must be deleted as well to demonstrate a more even and well-represented spread of common words, aside from some predicted words that are related to the topic, such as UK and EU being among the most common. The fourth most prevalent term in figure 14, "amp," was the other anomaly under investigation. Twitter encrypts the & sign with amp. We are left with the unencoded form of this since the & symbol was not appropriately recovered throughout the tweet extraction process. Given that this isn't really

3.3.3.3 Collection word removal

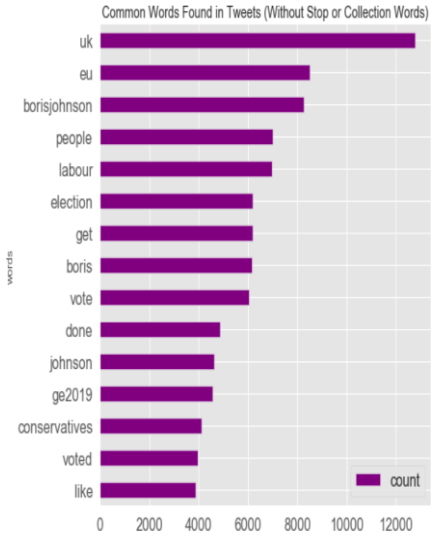
The procedure for eliminating collection words is the same as that for eliminating stopwords, with the exception that a list was constructed this time around rather than a predetermined one.

```
collection_words = ['brexit', 'amp']
tweets_nsw_nc = [[w for w in word if not w in collection_words]
                 for word in tweets_nsw]
```

Figure 14– for loop used to remove collections words from all tweets

The for loop that only keeps words that don't exist in the list of collection words is depicted in Figure

14. The collection word list now includes the word "brexit" in addition to "amp," which was previously defined as the & sign in its unencoded form. After this is finished, figures 20 and 21 show the findings. With the values of the common words being more closely reflected, the elimination now demonstrates that the proliferation of common terms was more influenced by the topic.




```
[('uk', 12777),
 ('eu', 8505),
 ('borisjohnson', 8254),
 ('people', 6994),
 ('labour', 6979),
 ('election', 6200),
 ('get', 6191),
 ('boris', 6144),
 ('vote', 6040),
 ('done', 4862),
 ('johnson', 4613),
 ('ge2019', 4566),
 ('conservatives', 4104),
 ('voted', 3957),
 ('like', 3877)]
```

3.3 Tokenisation

This section will go over the tokenization process and why it was a useful tool for preparing the data for the model's implementation. The data needed to be changed after it had been cleaned. The goal was to use a neural network to process the text, but the technique had a flaw in that neural networks can only comprehend numerical data. In order for the network that will be developed to recognise the data, it is necessary to convert it into numbers in order to overcome this difficulty.

We call this procedure "tokenization." For this operation, Keras includes a pre-processing package named Tokenizer.

3.4.1 Creating the Tokenizer

Tokenizing the words started with the creation of a memory-stored tokenizer variable. Since there are just spaces between words, the split parameter is set to a blank space and the word count is set to 50,000. After this is established, the texts are fitted with the tokenizer, which incorporates the words from the initial data into the tokenizer variable, resulting in the creation of a library. It is feasible to determine the number of unique individual words by looking up the length, which informed us that the tokenizer library has

66,422 unique words.

3.4.2 Creating three sets of Data

Three datasets were used for training and testing in order to facilitate good comparisons later on and to effectively train networks. The datasets included texts that were clean—that is, they contained no stopwords or collection words—texts that were left uncleaned, and texts that did not have a neutral Polarity value. Three new values were constructed and added to a new table containing the tweets and the polarity value in order to effectively eliminate the neutral polarity value from the clean data.The initial builds of the models were trained on clean data with neutral polarity, and then they were extended to incorporate unclean data. A 1 indicates that the value is not empty, and a 0 indicates that it is. These three values are Positive, Negative, and Neutral.

```
pos = []
neg = []
neut = []

for l in brexit_data_stop.polarity:
    if l == 0:
        pos.append(0)
        neg.append(0)
        neut.append(1)
    elif l == 1:
        pos.append(1)
        neg.append(0)
        neut.append(0)
    elif l == -1:
        pos.append(0)
        neg.append(1)
        neut.append(0)

brexit_data_stop['positive'] = pos
brexit_data_stop['negative'] = neg
brexit_data_stop['neutral'] = neut

brexit_data_stop = brexit_data_stop[['tweet', 'polarity', 'positive', 'negative', 'neutral']]
```

Figure 16 – for loop to create new data table values for polarity

	tweet	polarity	positive	negative	neutral
0	If you re a Brit amp you thought Boris amp Bre...	-1	0	1	0
1	AndrewBowie4WAK 14m voted for your Prime Menda...	-1	0	1	0
2	Morte all EUrodittatura Brexit gt Italexit	0	0	0	1
3	Well done Boris We leave the EU 31st Jan 2020 ...	0	0	0	1
4	This Christmas BorisJohnson promises the gift ...	0	0	0	1

Table2 – data table with added polarity values on uncleaned data

3.4.3 Filtering out Neutrals

Filtering can be used to eliminate the neutrals, enabling the creation of a dataset that entirely eliminates the neutral polarity. The elimination of the neutral value made it possible to evaluate and compare the results to see if the inclusion of the neutral value had any effect. This removal was motivated by previous study that only examined positive and negative polarity.

```
# Make another copy to maintain neutrals
brexit_data_withneutral = brexit_data.copy()
#Dropping neutrals column
brexit_data.drop(brexit_data[brexit_data['polarity'] == 0].index, inplace=True)
```

Figure 17 - filtering out neutral values

The filtering procedure is depicted in Figure 21. To ensure that the data including neutrals was preserved even after the neutrals were eliminated, a duplicate copy of the data was made. Following the creation of the copy, all polarity values with a value equal to 0, which denotes a neutral polarity, were dropped by calling the drop function. When inplace = false is used, Python returns a copy of the data instead of editing the original, as was the case when inplace = true was used at the end to tell Python that the query's output should be applied to the current data. Figure 17 displays the filtering results, indicating that no items corresponding to a polarity score of 0 remained.

```
# Show the dataframe with neutrals removed
brexit_data.polarity.value_counts()
```

```
 1    31024
-1    14485
Name: polarity, dtype: int64
```

Figure 18 – the result of filtering out neutral

3.4.4 Tokenizing and Padding

Holdout data had to be created in order for Neural Networks to use the data. Holdout is the term for data that is separated from training data and usually takes the form of test data that a machine learning model does not view during training. To make it simple to switch to a different dataset and to use the data solely for testing and training, three code blocks were developed. This method avoided the need to rewrite code by training various models at separate periods.

```
x = tokenizer.texts_to_sequences(brexit_data_withneutral['tweet'])
x = pad_sequences(x)
#y = polarity_all
y = brexit_data_withneutral[['positive', 'negative', 'neutral']]
#y = brexit_data_withneutral['polarity']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.15, random_state=0) #neutral class s
till within
```

Figure 19– code block sequence to create train and test split on the clean data with neutrals

The requirement that every piece of data in a layer have the exact same length is another crucial component of a neural network; otherwise, the model would protest and cease to function. By using a method called "padding," every data can have the same length. The train/test data preparation process using padding is depicted in Figure 19.To ensure that only words that the tokenizer knows are utilised, the texts_to_sequences pre-processing function was defined. This can be seen in the function's brackets, where it was instructed to utilise the terms that the tokenizer was familiar with based on the information in the "tweet" column. The sequences defined inside of x were

then padded when this was completed.

```
array([    0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,    20,
         917,     9,  5487,     5,  2575, 13158,     2], dtype=int32)
```

Figure 20 – a pre-padded tokenized tweet

A padded tokenized tweet is depicted in Figure 28. The term "target variable" refers to the value of y. This is what we want the model to guess, and accuracy is determined by how many times the model learns and provides the right response. In terms of padding, there are two approaches: pre-padding (beginning) and post-padding (end).This time, pre-padding was the method selected. This was due to our need to store the final result, or the hidden state, when utilising RNNs or CNNs and then utilise it to inform our predictions.

When the model reaches the conclusion of the data sequence if Post-Padding is applied, the final hidden state would be flushed out.

3.4.5 Tokenizer Inverse Map

Tokenizing words is fantastic, but if we want to be able to know which sentences are incorrect, for example, while assessing, we also need to be able to transform the text back. This was accomplished by developing an Inverse Map function that, upon call, transformed the chosen tokens and gave back the original text value without padding.

```
idx = tokenizer.word_index
inverse_map = dict(zip(idx.values(), idx.keys()))

def tokens_to_strings(tokens):
    #Map from tokens back to words
    words = [inverse_map[token] for token in tokens if token != 0]

    #Concatenate all words
    text = " ".join(words)

    return text
```

Figure 21 – function to change tokens back to words

```
x_train[10]
```

```
array([    0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,   834,   450,    20,   404,   244,   172,
         235,    14,  1469,    10,   440,    14,    83,    21,   133,
          32,    11,     1,   446,  1153,    10,    11,   477,    95,
       43723,     2, 43724, 43725, 43726, 43727,  5931], dtype=int32)
```

```
tokens_to_strings(x_train[10])
```

```
'started making this stupid thing last year i finished it tonight i don t know what s the point anyway it
s called british gothic brexit leagueofgentlemen reeceshearsmith stevepemberton americangothic illustratio
n'
```

Figure 22– A tokenized tweet before and after being reversed through an Inverse Map

To do this, the functions developed are displayed in Figure 22. The function takes the input and reverses the text by looking for any values that don't equal 0, where 0 is an empty value that was added during padding. It uses the number to get the textual value associated with the number from the

words variable; if the value is equal to 0, it is disregarded. Tokenization is reversed since this retains all of the words and their token values and shows the text instead of the number.

CHAPTER-4

4. PROJECT IMPLEMENTATION

4.1 Random Forest

Learning algorithms known as ensemble classification methods build a group of classifiers rather than a single classifier, and then categorise fresh data points by voting on their predictions. Bagging, Boosting, and Random Forest (RF) are the three ensemble classifiers that are most frequently employed. An algorithm for supervised machine learning based on ensemble learning is called random forest. In ensemble learning, distinct algorithm types or iterations of the same algorithm are combined to create a more potent prediction model. The term "Random Forest" comes from the random forest algorithm, which creates a forest of trees by combining several algorithms of the same kind, such as multiple decision trees.

The random forest algorithm is applicable to applications involving both classification and regression. The collection of tree-structured classifiers is known as the RF classifier. It is a more sophisticated kind of bagging that incorporates randomization. RF splits each node using the best among a subset of predictors randomly selected at that node, as opposed to utilising the best split among all variables. From the original data set, a new training data set is generated using replacement. The next step is to use random feature selection to grow a tree. Arboreal trees are not trimmed. With this tactic, RF achieves unprecedented accuracy. Additionally, RF is incredibly quick, resistant to overfitting, and allows the user to create as many trees as they desire. Create bootstrap samples for $ntree$ using the original data. Grow an unpruned classification or regression tree for each of the bootstrap samples, but change it so that at each node, randomly sample $mtry$ of the predictors and select the best split among those variables, instead of selecting the best split among all of the predictors. (Bagging can be considered the specific case of random forests that arise when the number of predictors, p, is equal to $mtry$.)

Assemble the forecasts of the $ntree$ trees to forecast fresh data (majority votes for classification, average for regression).

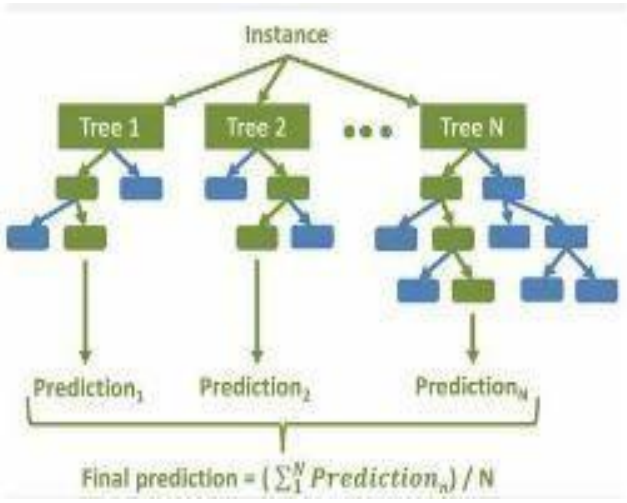


Fig:23 RF Prediction

F1 Points

An evaluation statistic for machine learning called the F1 score quantifies the accuracy of a model. It integrates a model's precision and recall ratings. The number of times a model correctly predicted throughout the whole dataset is calculated by the accuracy metric. Only when the dataset is class-balanced—that is, when each class contains an equal number of samples—can this be a trustworthy statistic

pd.read_csv('/content/train_tweet.csv') = train

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation

test = pd.read_csv('/content/test_tweets.csv')

	id	tweet
0	31963	#studiolife #aislife #requires #passion #dedic...
1	31964	@user #white #supremacists want everyone to s...
2	31965	safe ways to heal your #acne!! #altwaystohe...
3	31966	is the hp and the cursed child book up for res...
4	31967	3rd #bihday to my amazing, hilarious #nephew...

import from sklearn.ensemble RandomForestClassifier imports confusion_matrix and f1_score from sklearn.metrics and model = model.fit(x_train, y_train) returned by RandomForestClassifier() Model.predict(x_valid) = y_pred print(model.score(x_train, y_train)), "Training Accuracy:" model.score(x_valid, y_valid)) print("Validation Accuracy:").

figuring out the validation set's f1 score print("F1 score:", f1_score(y_valid, y_pred)) confusion_matrix(y_valid, y_pred) print(cm) # confusion matrix Accuracy of Training: 0.999123941429227 Accuracy of Validation: 0.9529470654486297 F1 score: 0.6163265306122448 [[257 302]] [7313 119]

4.2 DECISION TREE

The decision tree algorithm is an induction method for data mining that divides a set of records recursively using either a breadth-first or depth-first greedy approach until all of the data items are members of a specific class. Internal, leaf, and root nodes make up the structure of a decision tree. Unknown data records are classified using the tree structure. Impurity metrics

are used to determine the appropriate split at each internal node of the tree. The class labels that the data items have been grouped under make up the leaves of the tree. The two stages of the decision tree classification technique are tree building and tree pruning. The process of creating a tree is top-down.

The tree is recursively divided at this stage until every data item is associated with a single class label. Because the training data set is repeatedly accessed, it is extremely laborious and computationally demanding. Trees are pruned from the bottom up. It is employed to reduce over- fitting, or noise or excessive detail in the training data set, in order to increase the algorithm's prediction and classification accuracy. Misclassification error in decision tree algorithms is caused by over-fitting. As the training data set is only scanned once, tree pruning requires less work than the tree growth phase. The decision tree classification in the suggested system gives the user a better way to categorise tweets into positive and negative categories. The process involves comparing the maximum frequent things produced by the rules in the training data with the maximum frequent items in the test data. This allows for an easy categorization to be established.

This part includes comprehensive information on the experimental outcome, a discussion of the suggested system, as well as specifics on the experimental setup, performance metric, quantitative analysis, and comparison analysis. Python was used to create the suggested system, which has a 3.0 GHz Intel i5 processor, 4 GB of RAM, and a 1 TB hard drive. To evaluate the efficacy of the suggested methodology, its performance was contrasted with that of other categorization techniques and published research studies using the Twitter dataset. The accuracy of the suggested system's categorization, precision, recall, and f-measure were assessed. Measure of performance The regular measurement of outcomes and results that yields trustworthy data regarding the efficacy of the suggested method is known as the performance measure. The process of reporting, gathering, and evaluating data regarding an individual's or group's performance is also referred to as a performance measure.

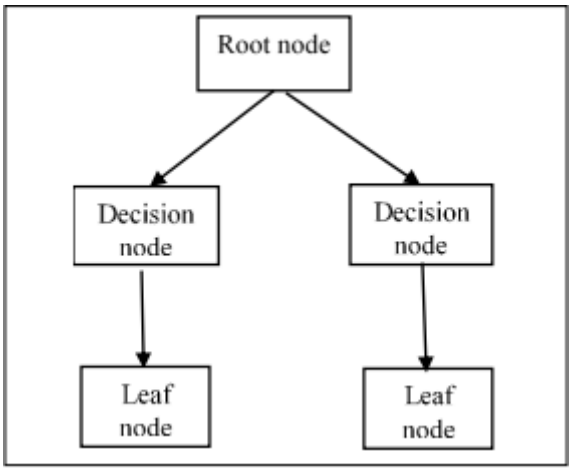


Fig:24- Structure of DT

Performance measure

The regular measurement of outcomes and results that yields trustworthy data regarding the efficacy of the suggested method is known as the performance measure. The process of reporting, gathering, and evaluating data regarding an individual's or group's performance is also referred to as a performance measure. The confusion metric used to assess the classifier for binary input is displayed in the table and can be explained as follows:

	Predicted positive	**Predicted Negative**
Actual positive	True Positive (TP)	False Positive (FP)
Actual Negative	False Negative (FN)	True Negative (TN)

Table 3- Confusion metric

False positives (FP) are real positives that are mistakenly labelled as negative, while true positives (TP) are actual positive situations that are accurately classified as positive. True negatives (TN) are real negative examples that are also appropriately classed as such, whereas false negatives (FN) are real negative examples that are mistakenly classified as positive. Eqs. (1), (2), (3), and (4) are the mathematical equations for accuracy, f-measure, precision, and recall.

The product of **K***lciacy* and $TP + TP + FN + FP \times 100$ A = $2TP$ $(2TP+FP+FN) \times 100$ (2) WP $(FP+TP) \times 100$ = $P\boldsymbol{U}ecision$ EI = TP $(FN+TP) \times 100$

In this case, FP denotes false positive, TN denotes true negative, TP stands for true positive, and FN denotes false negative.

Performance Analysis of Proposed Decision Tree

The performance evaluation of current approaches and the suggested strategy is compared in this experimental study using simulated Twitter data in terms of accuracy, F-measure, precision, and recall. STC files are used to run two rounds of tests under different experimental conditions. The main goal of the studies is to find the optimal supervised method in comparison to DT, MLP, and random forest (RF). Every topic-based text classified tweet in the Sanders corpus has its DT computed. The organisation can concentrate on tweets that have the biggest effects thanks to this classification. The discussion of Sanders' defined themes for DT is shown below for the ensuing sections.

Performance Analysis in terms of Precision, Recall and F-Measure

Methods	Parameters	Positive	Negative	Neutral
	Precision	0.56	0.27	0.66

This section compares the precision, recall, and F-measure of DT with that of MLP and RF for three classes: positive, negative, and neutral. The overall effectiveness of the DT algorithm's precision, recall, and F-measure in comparison to other methods like RF and MLP

Table 4-Performance

RF	Recall	0.39	0.04	0.86
	F-Measure	0.46	0.07	0.74
MLP	Precision	1.00	1.00	0.59
	Recall	0.06	0.02	1.00
	F-Measure	0.11	0.04	0.75
Proposed DT	Precision	**0.92**	0.46	0.86
	Recall	0.74	0.46	**0.95**
	F-Measure	0.82	**0.46**	0.91

Table 5- Values of proposed DT

The top suggested DT values for each of the three classes are indicated in bold in the above table. While the best recall value for the DT is 95% for neutral, 46% for negative, and 74% for positive, the DT achieved 92% precision for positive class, 46% for negative, and 86% for neutral. 82% of the DT F-Measure is positive, 91% is neutral, and 46% is negative. In terms of negative class of recall, MLP performs poorly when compared to current methods. The greatest neutral values for the RF technique were 66% for precision, 86% for recall, and 74% for F-Measure.

Methods	**Precision**	**Recall**	**F-Measure**
RF	81.16	80.85	80.83
MLP	74.74	74.20	74
Proposed DT	**88.12**	**92.31**	**90.17**

Table 6- Methods performance of DT

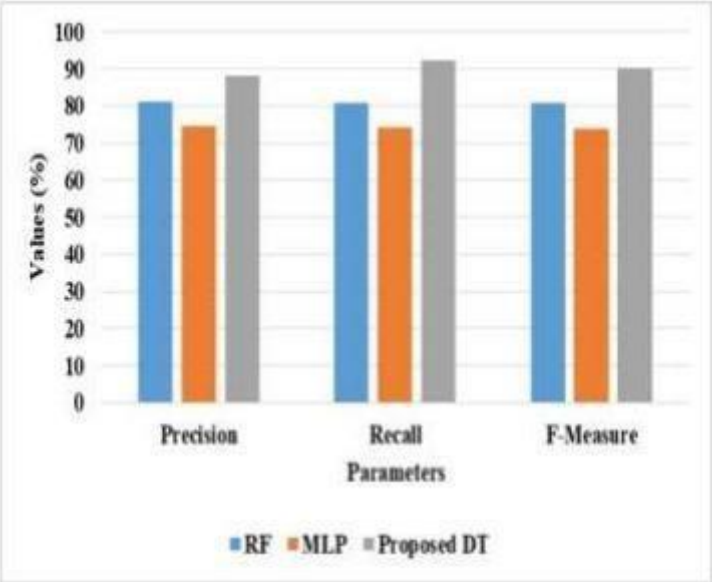


Fig:25-Performance of DT

It is evident from this that, in comparison to previous methods, the suggested DT produced better precision, recall, and F-Measure values. However, the weighting strategy allowed the RF to achieve 81.16% precision, 80.85% recall, and 80.83% F-Measure. MLP has a precision value of 74.74% and a recall value of 74.20%. The correctness of DT's performance will be discussed in the next section.

Performance Analysis of DT in terms of accuracy

This section compares the suggested DT's classification accuracy with current methods like RF and MLP.

Methods	**Accuracy**
RF	56.46
MLP	80.97
Proposed DT	**99.51**

Table 7- Accuracy

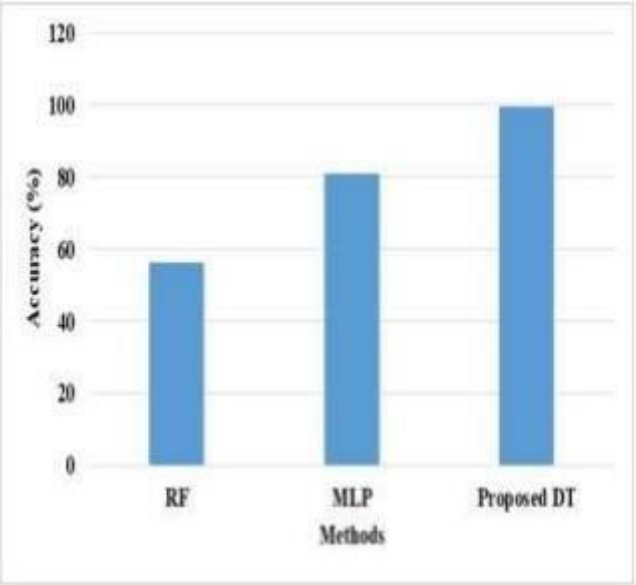


Fig 26-Performance of DT in terms of Accuracy

According to the experimental findings, the suggested DT classified tweets from STC datasets with a greater accuracy (i.e., 99.51%). The RF's accuracy was incredibly low in comparison to other methods now in use. While achieving 80.97% accuracy, the MLP technique was unable to concentrate the quantity of computing labour. The STC dataset's testing and training results are contrasted. When compared to other methods that are currently in use, including RF and MLP, the STC dataset yields the best results from the DT approach. It was evident that obtaining extremely precise findings was significantly influenced by the quantity of samples in the training and testing datasets.

The findings show that the suggested DT can improve words' capacity to discriminate when it comes to tweet classification. The suggested approach can help with sentiment analysis, spam identification, dimensionality reduction, and many other applications that deal with various tweet-related issues. As a result, the suggested strategy lessens the impact of these difficulties on the classification task's performance. We believe that this proposed strategy can also be used to other social media, even though our results were obtained using data from Twitter, which is one of the most

popular social media platforms.

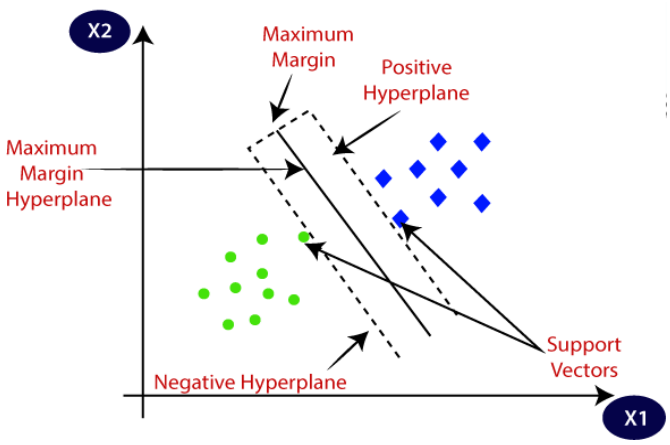
```
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier() model.fit(x_train, y_train)

y_pred = model.predict(x_valid)

print("Training Accuracy :", model.score(x_train, y_train)) print("Validation Accuracy :", model.score(x_valid, y_valid))

# calculating the f1 score for the validation set

print("f1 score :", f1_score(y_valid, y_pred))# confusion matrix cm = confusion_matrix(y_valid, y_pred) print(cm)
```

Training Accuracy :0.9991656585040257 Validation Accuracy : 0.9325491177574772 f1 score : 0.5373390557939915 [[7139 293] [246 313]]

4.3 SVM

One of the most widely used supervised learning techniques for both classification and regression issues is support vector machine, or SVM. But it's mostly applied to machine learning classification challenges. In order to make it simple to classify fresh data points in the future, the SVM method seeks to identify the optimal line or decision boundary that can divide n-dimensional space into classes. We refer to this optimal decision boundary as a hyperplane. SVM selects the extreme vectors and points to aid in the creation of the hyperplane. The algorithm is referred regarded as a Support Vector Machine since these extreme situations are known as support vectors. Hyperplane: To divide the classes in n-dimensional space, there can be a number of lines or decision boundaries; nevertheless, the most effective decision boundary for data point classification must be identified. The SVM hyperplane is the name given to this optimal boundary. The features in the dataset determine the hyperplane's dimensions; if there are two characteristics, the hyperplane will have a straight line as its shape. Additionally, a hyperplane will be a two-dimensional plane if there are three characteristics.

Support vectors are the data points or vectors that are closest to the hyperplane and have an impact on its position. These vectors are known as support vectors because they provide support to the hyperplane.



$g(X)=w^T\phi(X)+b$

The vectors "w," "b," and "X" represent weight, feature, and bias, respectively. From input space to high dimensional features space, ϕ() is a nonlinear mapping. In this case, "w" and "b" are automatically learned on the training set. SVM is an effective tool for recognising patterns.

```
from sklearn.svm import SVC

model = SVC() model.fit(x_train, y_train)

y_pred = model.predict(x_valid)

print("Training Accuracy :", model.score(x_train, y_train))
print("Validation Accuracy :", model.score(x_valid, y_valid))

# calculating the f1 score for the validation set

 print("f1 score :", f1_score(y_valid, y_pred))

# confusion matrix

cm = confusion_matrix(y_valid, y_pred) print(cm)
```

Training Accuracy : 0.978181969880272 Validation Accuracy :

0.9521962207483419 f1 score : 0.4986876640419947

[[7419 13]

[369 190]]

4.4 Logistic Regression

One of the most widely used machine learning algorithms, under the category of supervised learning, is logistic regression. With a given collection of independent factors, it is used to predict the categorical dependent variable. With logistic regression, the result of a categorical dependent variable is predicted. As a result, a discrete or category value must be the result. Instead of providing the exact values, which are 0 and 1, it provides the probabilistic values, which fall between 0 and 1. It can be either Yes or No, 0 or 1, true or False, etc. With the exception of how they are applied, logistic regression and linear regression are very similar. While logistic regression is used to solve classification difficulties, linear regression is used to solve regression problems.

Logistic Function (Sigmoid Function):

A mathematical function called the sigmoid function is utilised to convert expected values into probabilities.Any real value can be mapped into another value inside a The logistic regression's result must lie between 0 and 1, and as it cannot be greater than this, it takes the shape of a "S" curve. The logistic or sigmoid function is another name for the S-form curve. The idea of the threshold value, which indicates the likelihood of either 0 or 1, is used in logistic regression. For example, values above the threshold tend towards one, and those below the threshold tend towards zero.Logistic Regression Equation:

We are aware that the straight line equation can be expressed as follos:

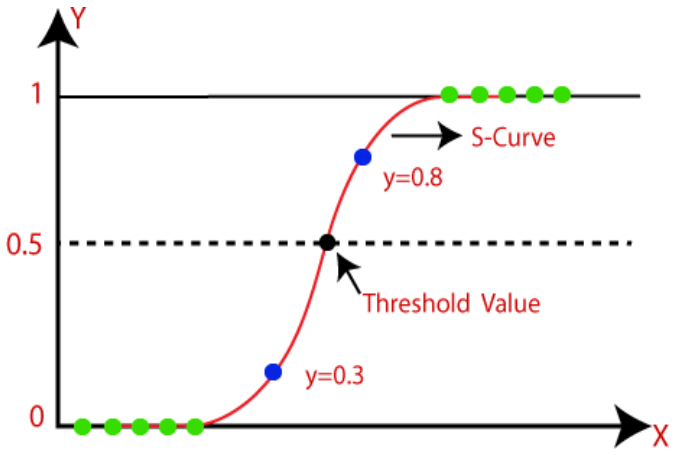
$$y= b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

Since y can only be between 0 and 1 in logistic regression, let's divide the following equation by (1- y):

$\frac{y}{}$ · 0 for y= 0 and infinity for y=1

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

However, we require the range from -[infinity] to +[infinity]. After taking the equation's logarithm, it becomes:



import sklearn.linear_model logicalLogistic regression modelRegression() y_pred = model.predict(x_valid) model.fit(x_train, y_train)

print(model.score(x_train, y_train)), "Training Accuracy:" model.score(x_valid, y_valid)) print("Validation Accuracy:").

figuring out the validation set's f1 score print("f1 score:", f1_score(y_valid, y_pred))

confusion_matrix(y_valid, y_pred) print(cm) # confusion matrix

Accuracy of Training: 0.9851487213716574 [[7185 247] [219 340]] Validation Accuracy:

0.9416843949443123 f1 score: 0.5933682373472949

CHAPTER-5

5. Critical Evaluation

This section will go over the projects' accomplishments, potential future directions for the research, the writer's next steps in this project, and how using quantum artificial intelligence can help the project even more.

5.1 Project Achievements

In terms of surpassing the initial accuracy target, this project was successful. The highest accuracy ever attained, surpassing the project's goal. The project itself had three goals, and it is believed that these goals were both reached and, in some cases, exceeded. The primary goal of the project was to use modelling techniques to classify textual data; however, the specific methodologies were left vague to allow for some degree of flexibility in the solution-finding process.

Because of its capabilities, deep learning was a fantastic tool for achieving this goal. While machine learning would have also made it possible to tackle the problem, deep learning was significantly superior because it made it possible to construct more intricate and comprehensive network architectures, such the convolutional neural network. The project's predicted natural progression led to the accomplishment of the secondary objectives, which transpired as anticipated.

Model Evaluation and Model Investigation were these. Using the research completed for the literature review part, the investigation was satisfied. While not all of the choices revealed in this research would be employed in the project, this review helped the networks to be decided and realised. The assessment of the models was likewise a resounding success, yielding positive results that exceeded expectations and text classification with very few failures, mostly related to sentence ambiguity (e.g., using the phrase "Not Great"), which was greeted with expected failure.

5.2 Further Development

It is thought that the study and effort towards bigram and trigram models should be done next in order to build on the work of this project. The models developed for this study were mostly focused on unigrams, which resulted in expected failures in regions where bigram and trigram investigation using Deep Learning methods may have produced superior results. Though it's also thought that the removal of stopwords didn't benefit the trained models' performance on unseen data in this case, this is still thought to be another area that could be examined in more detail to see where adjustments could be made to produce better results on data that doesn't use these stopwords.

5.3 Future Work

Although sentiment analysis research has come a long way since its beginning, there is still more work to be done on this issue. Even now, computers still struggle with sarcasm recognition and phrase ambiguity. However, there is hope that future research using more sophisticated neural networks and Tensors will help overcome these challenges. As of the project's launch, Quantum Tensorflow is accessible. The goal is to progress in this field of computing capacity to build more potent models and overcome the challenges associated with text classification to provide more perceptive computing tools.

5.4 Personal Reflection

The project proceeded smoothly, and keeping a weekly journal made it easier to produce. In order to keep a realistic picture of work during the early stages, this weekly record gave for the opportunity to establish deadlines for the following week and helped to comprehend the progression the project had taken over that week. One of the drawbacks of training recurrent neural networks was that the networks with 50 epochs would occasionally take up to 7 hours to complete, even on a GPU, necessitating nocturnal labour. Although this was a mistake made at the start of the project, it needed to be corrected

for.

Because models had to be trained over several hours and then another model had to be generated in order to modify, the incremental manner of model construction was also tremendously helpful.

Should this have been completed using a different methodology, there would have been problems with the project's pacing and the alternative flexible strategy that was

accessible. Bigram research would have been conducted in greater detail if there had been more time for this endeavour.Sadly, Quantum Tensorflow had not yet been made available to the general public when this project began, or else its incorporation would have been looked at as well.Further investigation and analysis into the field of automatic web scraping could be beneficial in the future. Making this process more automated might be a useful step in the direction of constructing sentient AI in the form of chat bots to assist people online. Sentiment analysis and web scraping are used in the context of understanding and responding to consumer questions.

Outputs

1.impoting module

```
Click here to ask Blackbox to help you code faster
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

import warnings
```

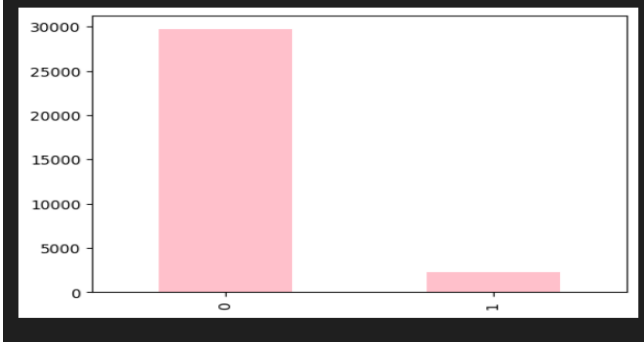
```
Click here to ask Blackbox to help you code faster
train.head()
```

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation

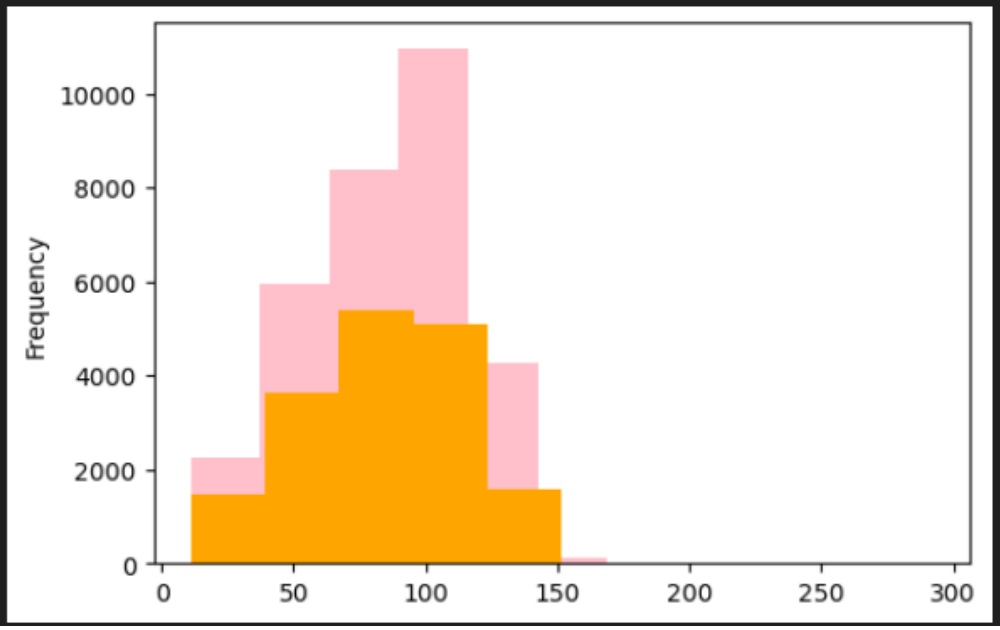
Test cases

```
Click here to ask Blackbox to help you code faster
train['label'].value_counts().plot.bar(color = 'pink', figsize = (6, 4))
```

```
<Axes: >
```




```
Click here to ask Blackbox to help you code faster
# checking the distribution of tweets in the data

length_train = train['tweet'].str.len().plot.hist(color = 'pink', figsize = (6, 4))
length_test = test['tweet'].str.len().plot.hist(color = 'orange', figsize = (6, 4))
```



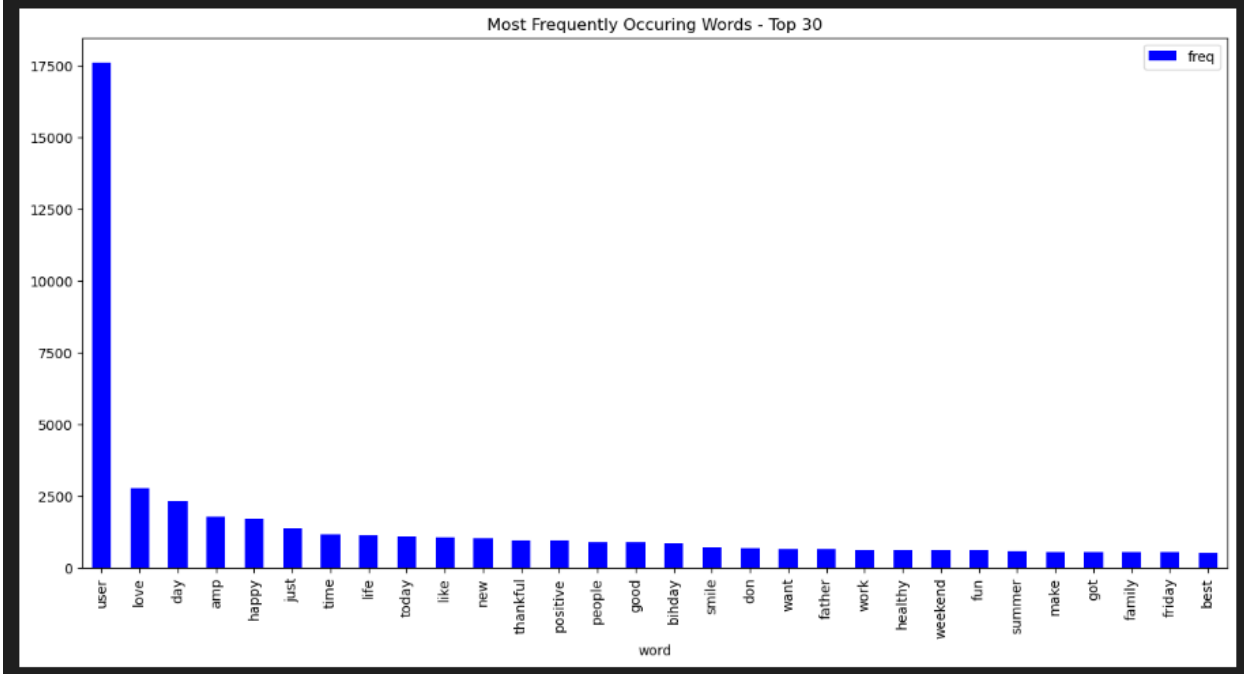

```
Click here to ask Blackbox to help you code faster
train.groupby('label').describe()
```

	id								len							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
label																
0	29720.0	15974.454441	9223.783469	1.0	7981.75	15971.5	23965.25	31962.0	29720.0	84.328634	29.566484	11.0	62.0	88.0	107.0	274.0
1	2242.0	16074.896075	9267.955758	14.0	8075.25	16095.0	24022.00	31961.0	2242.0	90.187779	27.375502	12.0	69.0	96.0	111.0	152.0

Testing repeated words in the datasets

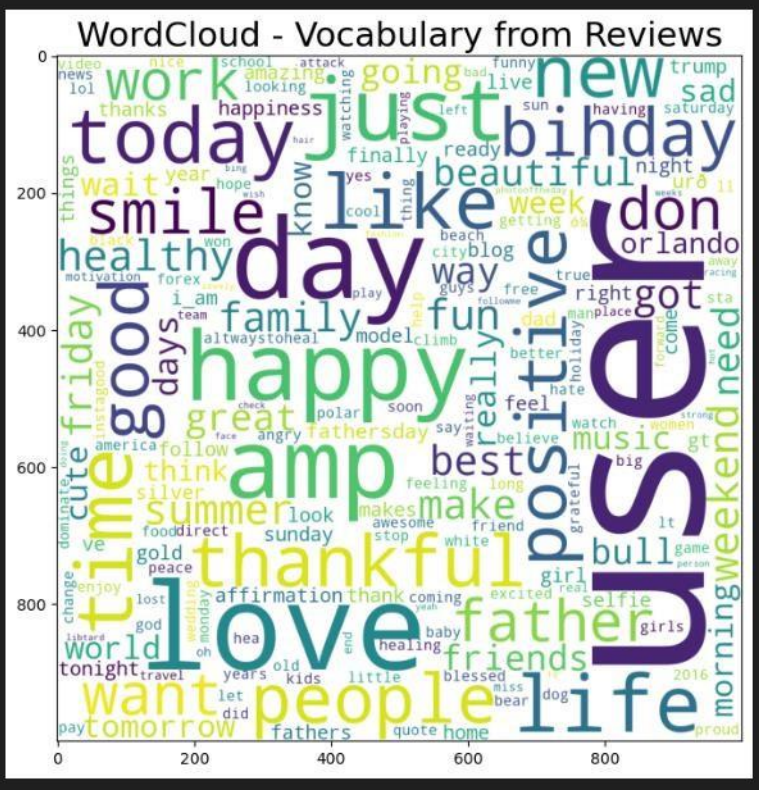
```
Click here to ask Blackbox to help you code faster
from sklearn.feature_extraction.text import CountVectorizer


cv = CountVectorizer(stop_words = 'english')
words = cv.fit_transform(train.tweet)

sum_words = words.sum(axis=0)

words_freq = [(word, sum_words[0, i]) for word, i in cv.vocabulary_.items()]
words_freq = sorted(words_freq, key = lambda x: x[1], reverse = True)

frequency = pd.DataFrame(words_freq, columns=['word', 'freq'])

frequency.head(30).plot(x='word', y='freq', kind='bar', figsize=(15, 7), color = 'blue')
plt.title("Most Frequently Occuring Words - Top 30")
```

```
Text(0.5, 1.0, 'Most Frequently Occuring Words - Top 30')
```



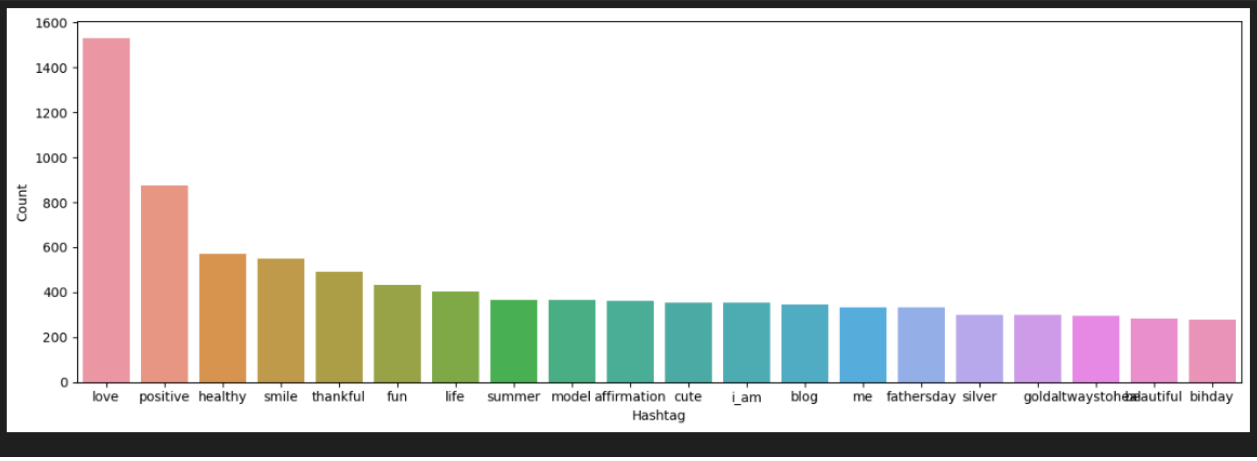
```
Click here to ask Blackbox to help you code faster
from wordcloud import WordCloud

wordcloud = WordCloud(background_color = 'white', width = 1000, height = 1000).generate_from_frequencies(dict(words_freq))

plt.figure(figsize=(10,8))
plt.imshow(wordcloud)
plt.title("WordCloud - Vocabulary from Reviews", fontsize = 22)
```

```
Text(0.5, 1.0, 'WordCloud - Vocabulary from Reviews')
```



Using different colors for different words in the dataset

```
Click here to ask Blackbox to help you code faster
import nltk
a = nltk.FreqDist(HT_regular)
d = pd.DataFrame({'Hashtag': list(a.keys()),
                  'Count': list(a.values())})

# selecting top 20 most frequent hashtags
d = d.nlargest(columns="Count", n = 20)
plt.figure(figsize=(16,5))
ax = sns.barplot(data=d, x= "Hashtag", y = "Count")
ax.set(ylabel = 'Count')
plt.show()
```



Testing the percent repeated words in the data set

```
Click here to ask Blackbox to help you code faster
model_w2v.wv.most_similar(positive = "cancer")
```

```
[('oâ\x80¦', 0.5829869508743286),
 ('wounded', 0.5783131122589111),
 ('worldâ\x80\x99s', 0.5597503185272217),
 ('pairs', 0.5417007207870483),
 ('surviving', 0.5375413298606873),
 ('diabetes', 0.531140923500061),
 ('buyers', 0.5309293866157532),
 ('lbs', 0.5280327200889587),
 ('leukemia', 0.5275989770889282),
 ('wanton', 0.5251265168190002)]
```

Testing the results for the dataset (Positive,Negative,Neutral)

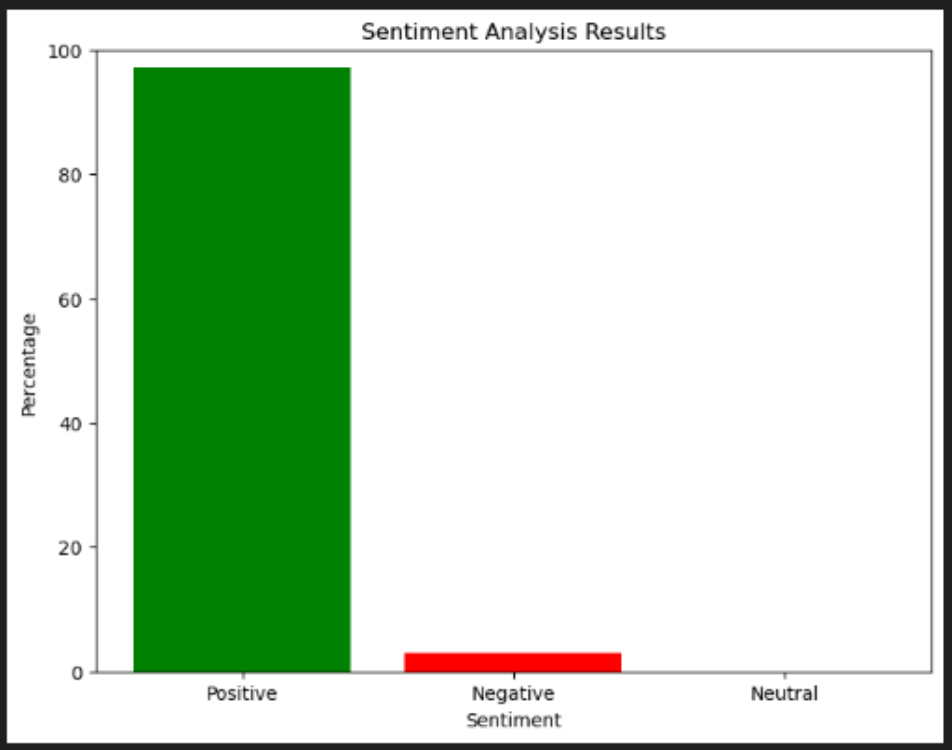
```
model.fit(tweet_vectors, labels)

# Analyze sentiment for each tweet
sentiments = []
sentiment_reviews = []
for tweet in tweets:
    sentiment = analyze_sentiment(tweet, model, tfidf_vectorizer)
    sentiments.append(sentiment)
    sentiment_reviews.append(get_sentiment_review(sentiment))

# Add sentiment predictions and reviews to the DataFrame
data['predicted_sentiment'] = sentiments
data['sentiment_review'] = sentiment_reviews

# Calculate accuracy (optional)
accuracy = accuracy_score(labels, sentiments)
print("Accuracy:", accuracy)

# Display results for each tweet
for index, row in data.iterrows():
    print("Tweet:", row['tweet'])
    print("Predicted Sentiment:", row['predicted_sentiment'])
    print("Sentiment Review:", row['sentiment_review'])
    print()

# Generate report
report = data.groupby('predicted_sentiment').size()
print("Sentiment Report:")
print(report)
```

```
Accuracy: 0.9565421437957574
Tweet:  @user when a father is dysfunctional and is so selfish he drags his kids into his dysfunction.   #run
Predicted Sentiment: 0
Sentiment Review: Negative review

Tweet: @user @user thanks for #lyft credit i can't use cause they don't offer wheelchair vans in pdx.    #disapointed #getthanked
Predicted Sentiment: 0
Sentiment Review: Negative review

Tweet:   bihday your majesty
Predicted Sentiment: 0
Sentiment Review: Negative review

Tweet: #model   i love u take with u all the time in urð±!!! ðððð¦ð¦ð¦ð¦ð¦ð¦ð¦ð¦ð¦ð¦
Predicted Sentiment: 0
Sentiment Review: Negative review

Tweet:  factsguide: society now    #motivation
Predicted Sentiment: 0
Sentiment Review: Negative review

Tweet: [2/2] huge fan fare and big talking before they leave. chaos and pay disputes when they get there. #allshowandnogo
Predicted Sentiment: 0
Sentiment Review: Negative review
...
predicted_sentiment
0    30985
1      977
dtype: int64
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

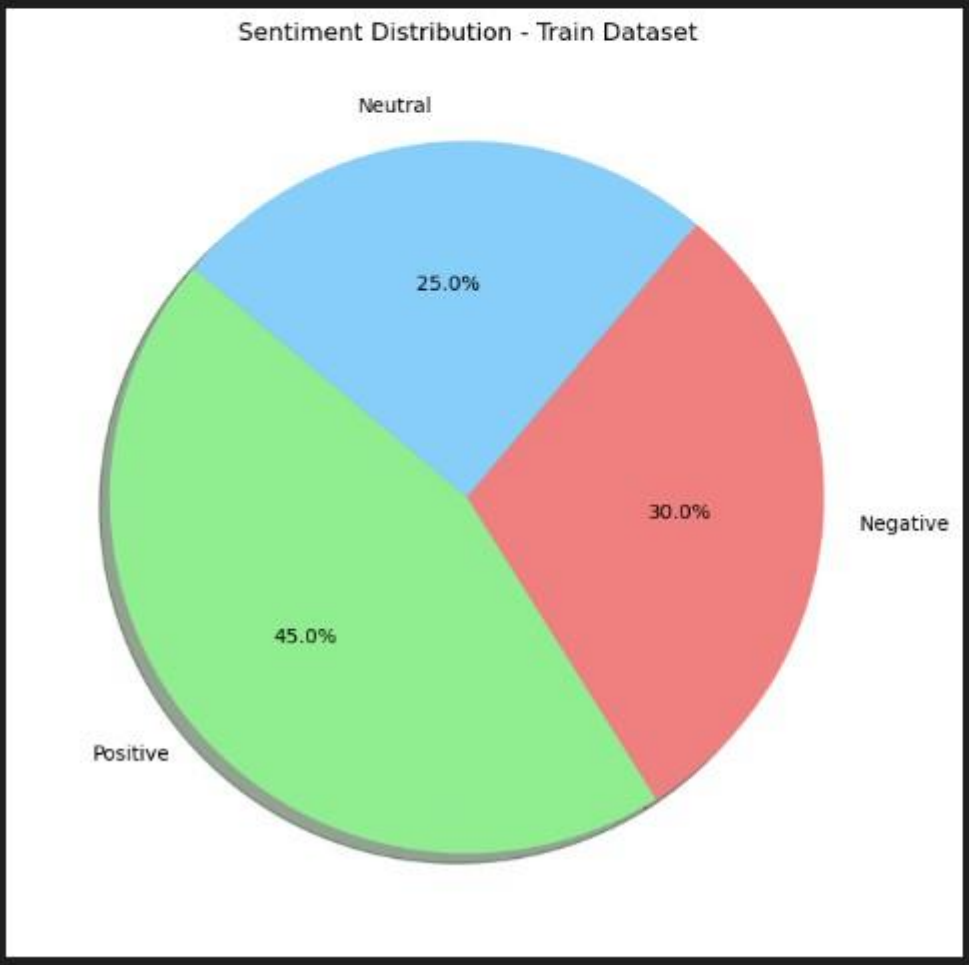
\

Visualization of the results for train dataset

```
Click here to ask Blackbox to help you code faster
import matplotlib.pyplot as plt

# Data for plotting
percentages = [positive_percentage, negative_percentage, neutral_percentage]
labels = ['Positive', 'Negative', 'Neutral']
colors = ['green', 'red', 'blue']

# Create bar graph
plt.figure(figsize=(8, 6))
plt.bar(labels, percentages, color=colors)
plt.xlabel('Sentiment')
plt.ylabel('Percentage')
plt.title('Sentiment Analysis Results')
plt.ylim(0, 100)

# Show the graph
plt.show()
```

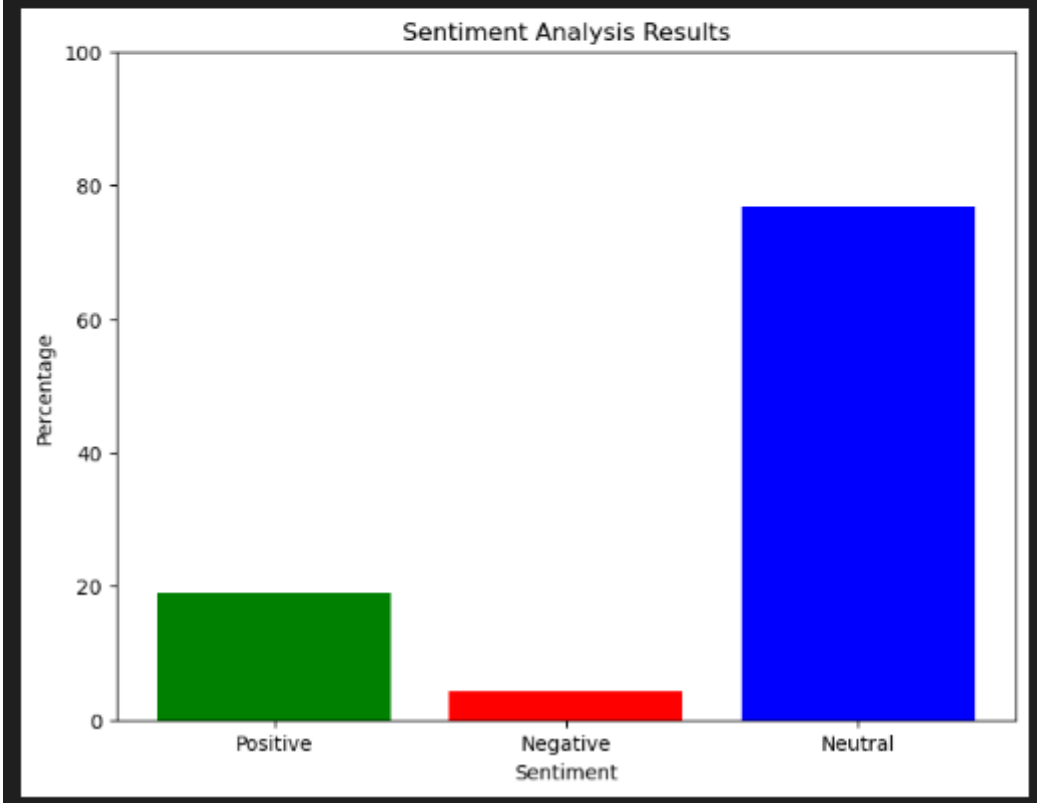


```
Click here to ask Blackbox to help you code faster
import matplotlib.pyplot as plt

# Example data: percentages of sentiments from sentiment analysis
train_percentages = {
    'Positive': 45,  # Example: 45% of data is Positive
    'Negative': 30,  # Example: 30% of data is Negative
    'Neutral': 25    # Example: 25% of data is Neutral
}

# Extracting percentages for pie chart
sentiments = ['Positive', 'Negative', 'Neutral']
percentages = [train_percentages.get(sentiment, 0) for sentiment in sentiments]
colors = ['lightgreen', 'lightcoral', 'lightskyblue']

# Creating the pie chart
plt.figure(figsize=(8, 8))
plt.pie(percentages, labels=sentiments, colors=colors, autopct='%1.1f%%', startangle=140, shadow=True)
plt.title('Sentiment Distribution - Train Dataset')
plt.show()
```

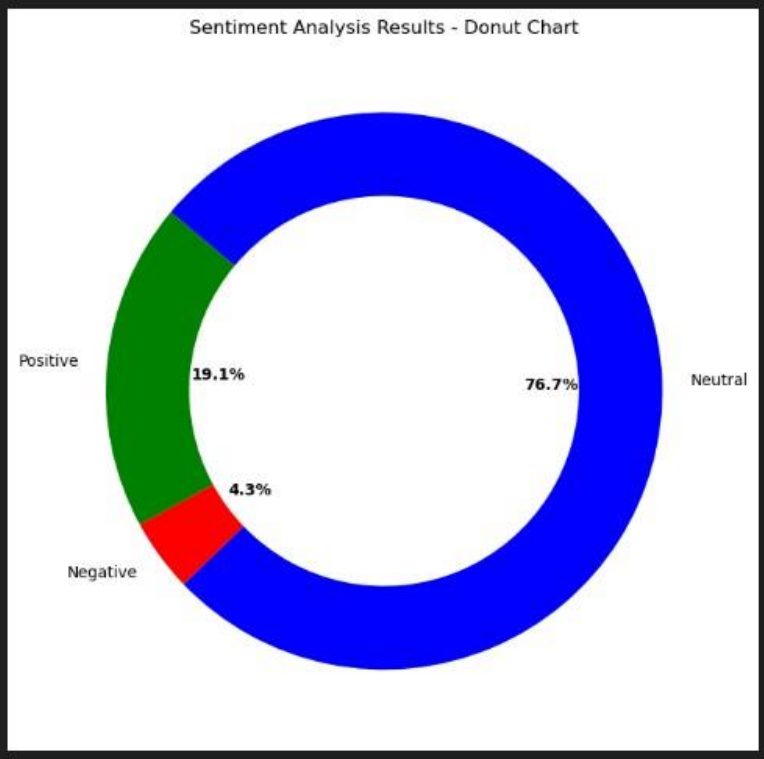


Visualization of the results for t

```
Click here to ask Blackbox to help you code faster
import matplotlib.pyplot as plt

# Data for plotting
percentages = [positive_percentage, negative_percentage, neutral_percentage]
labels = ['Positive', 'Negative', 'Neutral']
colors = ['green', 'red', 'blue']

# Create bar graph
plt.figure(figsize=(8, 6))
plt.bar(labels, percentages, color=colors)
plt.xlabel('Sentiment')
plt.ylabel('Percentage')
plt.title('Sentiment Analysis Results')
plt.ylim(0, 100)

# Show the graph
plt.show()
```

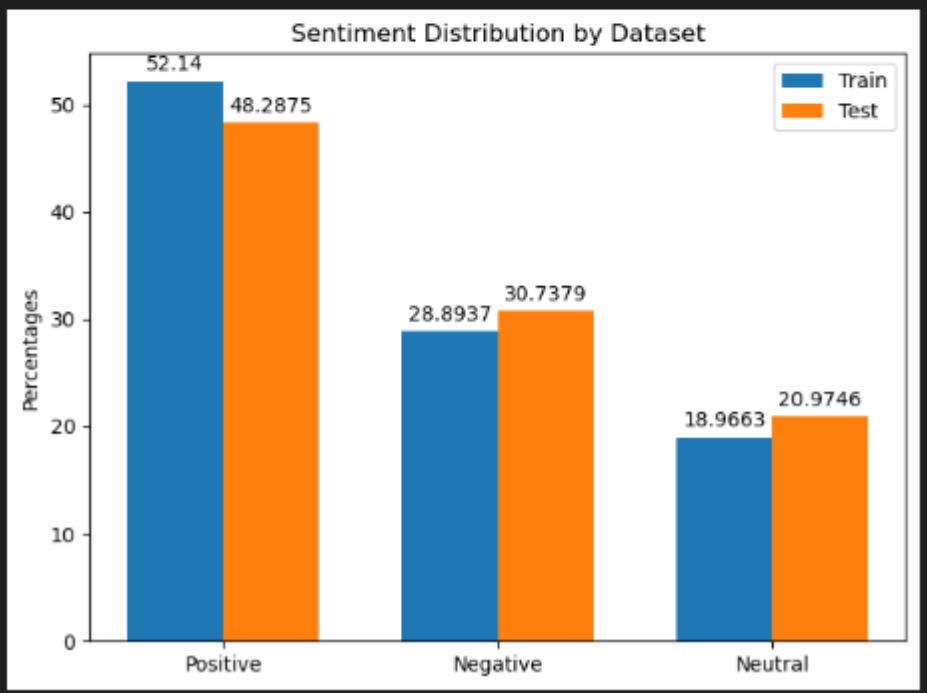


```
Click here to ask Blackbox to help you code faster
# Donut chart for sentiment distribution
fig, ax = plt.subplots(figsize=(8, 8), subplot_kw=dict(aspect="equal"))
wedges, texts, autotexts = ax.pie(percentages, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140, wedgeprops=dict(width=0.3))

plt.setp(autotexts, size=10, weight="bold")
ax.set_title('Sentiment Analysis Results - Donut Chart')

plt.show()
```

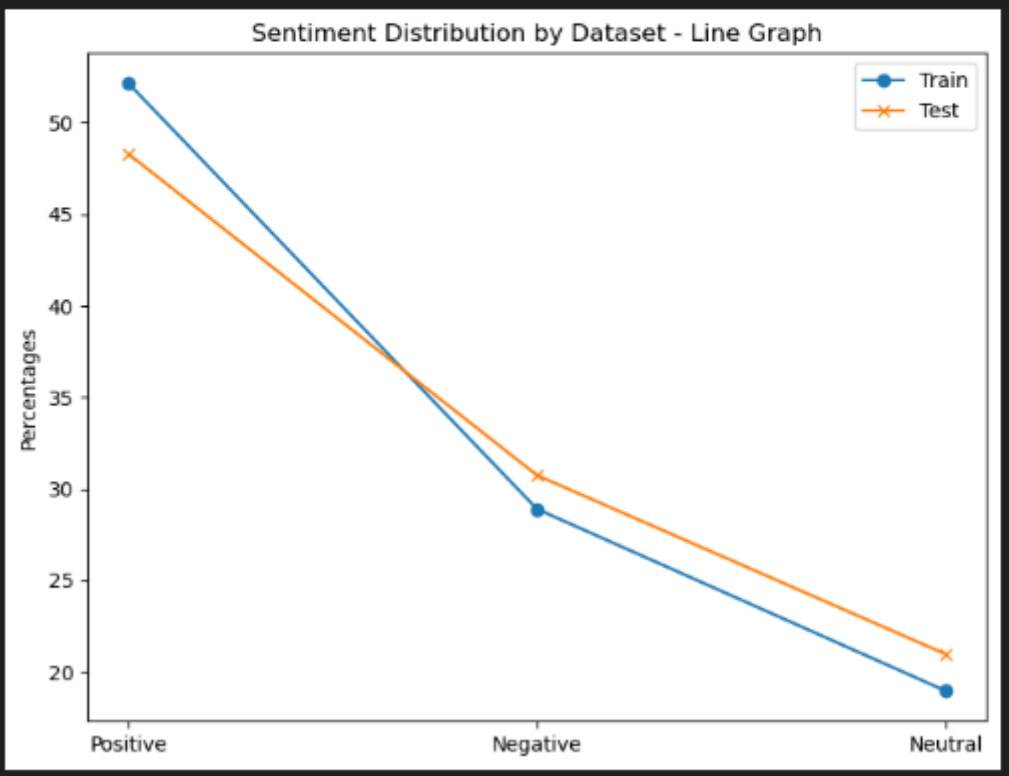


Difference between the train and test dataset results

```
Click here to ask Blackbox to help you code faster
import matplotlib.pyplot as plt

# Data for plotting
categories = ['Positive', 'Negative', 'Neutral']
train_values = [train_percentages.get(category, 0) for category in categories]
test_values = [test_percentages.get(category, 0) for category in categories]

# Plotting
x = np.arange(len(categories))  # the label locations
width = 0.35  # the width of the bars

fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, train_values, width, label='Train')
rects2 = ax.bar(x + width/2, test_values, width, label='Test')

# Add some text for labels, title and custom x-axis tick labels, etc.
ax.set_ylabel('Percentages')
ax.set_title('Sentiment Distribution by Dataset')
ax.set_xticks(x)
ax.set_xticklabels(categories)
ax.legend()

ax.bar_label(rects1, padding=3)
ax.bar_label(rects2, padding=3)

fig.tight_layout()

plt.show()
```

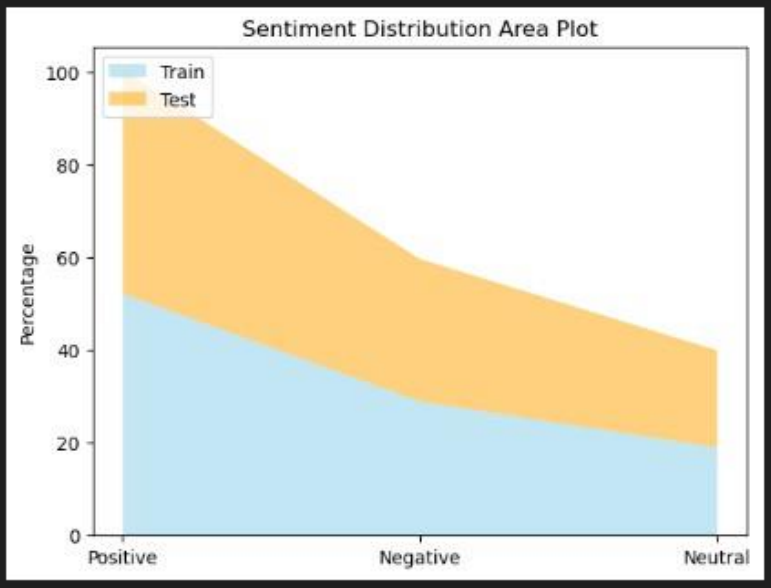


```
Click here to ask Blackbox to help you code faster
plt.figure(figsize=(8, 6))
plt.plot(categories, train_values, label='Train', marker='o')
plt.plot(categories, test_values, label='Test', marker='x')
plt.ylabel('Percentages')
plt.title('Sentiment Distribution by Dataset - Line Graph')
plt.legend()
plt.show()
```

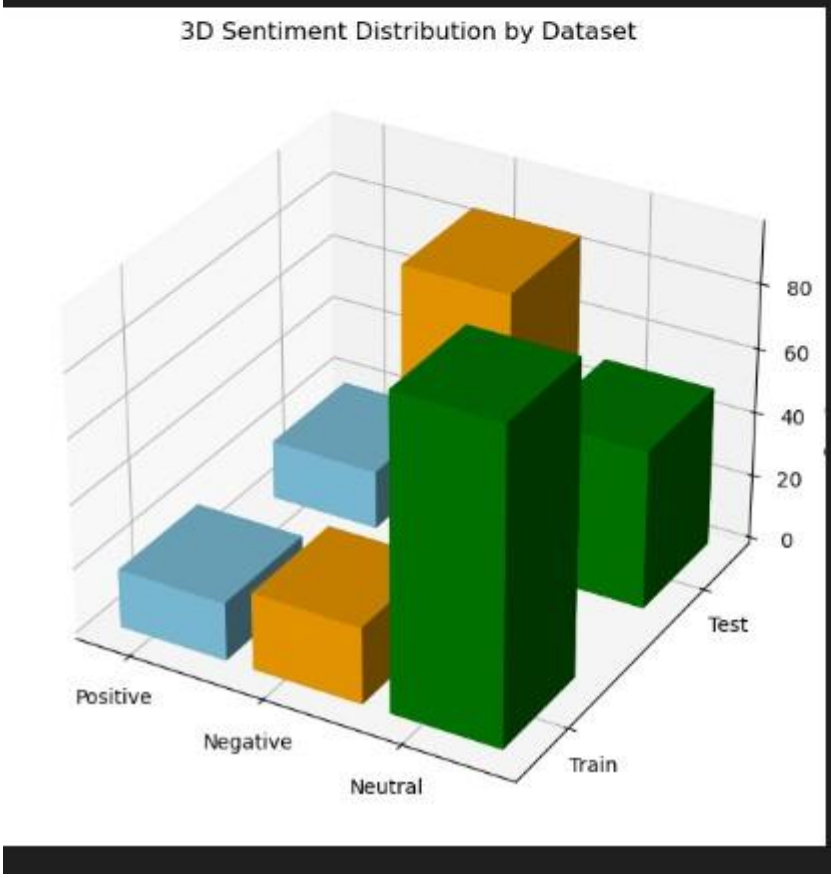


Click here to ask Blackbox to help you code faster

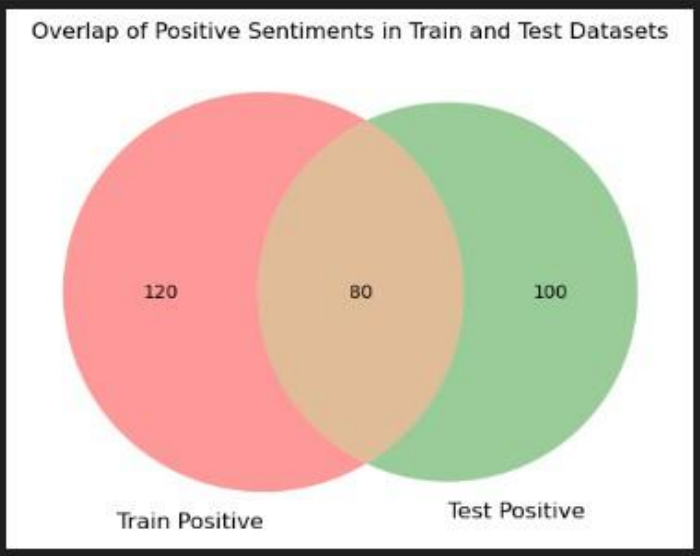
```
plt.stackplot(categories, train_values, test_values, labels=['Train', 'Test'], colors=['skyblue', 'orange'], alpha=0.5)
plt.legend(loc='upper left')
plt.title('Sentiment Distribution Area Plot')
plt.ylabel('Percentage')
plt.xticks(categories)
plt.show()
```



```python
# Flattening the grid arrays
x, y = x.flatten(), y.flatten()

# Setting the z positions and the depth (dz) for each bar
z = np.zeros_like(x)
dz = np.concatenate([train_values, test_values])

# Setting up the colors
colors = ['skyblue', 'orange', 'green']

# Drawing the bars
for i in range(len(categories)):
    ax.bar3d(x[i::len(categories)], y[i::len(categories)], z[i::len(categories)],
             dx=0.8, dy=0.5, dz=dz[i::len(categories)], color=colors[i % len(colors)])

# Setting the labels and titles
ax.set_xticks(x_pos + 0.4 / 2)
ax.set_xticklabels(categories)
ax.set_yticks([0.25, 1.25])
ax.set_yticklabels(['Train', 'Test'])
ax.set_zlabel('Sentiment Score')

ax.set_title('3D Sentiment Distribution by Dataset')

plt.show()
```



```
Click here to ask Blackbox to help you code faster
from matplotlib_venn import venn2
import matplotlib.pyplot as plt

# Simulated example of the number of positive, negative, and neutral sentiments in Train and Test datasets
# Let's assume these are counts of texts classified in each sentiment category
train_sentiments = {'Positive': 120, 'Negative': 80, 'Neutral': 60}
test_sentiments = {'Positive': 100, 'Negative': 90, 'Neutral': 70}

# For simplicity, let's focus on one sentiment category for the Venn diagram, e.g., 'Positive'
# The numbers represent counts of 'Positive' sentiments in each dataset and their hypothetical overlap
venn2(subsets=(train_sentiments['Positive'], test_sentiments['Positive'], 80),
      set_labels=('Train Positive', 'Test Positive'))

plt.title('Overlap of Positive Sentiments in Train and Test Datasets')
plt.show()
```



CHAPTER-6

Conclusion

A field of study called sentiment analysis, often known as opinion mining, examines people's attitudes, feelings, and sentiments towards particular things. Sentiment polarity categorization is a key challenge in sentiment analysis that is addressed in this study. The study's dataset consists of tweets pertaining to six airlines that were chosen from kaggle.com. Using the random forest technique, we were able to do sentiment analysis with an accuracy of about 95%. To see if you can get better results, I would advise you to try using some alternative machine learning techniques, such as logistic regression and SVM.One of the newest areas of study for determining and assessing user attitudes and perspectives is TSA. The suggested approach is divided into two phases, such as tweet classification and pre- processing. The obtained Twitter data is pre-processed by removing superfluous emojis from the messages and applying missing value handling.

The DT algorithm is used for TSA with the pre-processed data. The superiority of the suggested method was demonstrated by the simulated Sanders Twitter data used to validate the experimental examination of DT. The suggested methodology outperforms the earlier approaches in terms of the categorization rate of the Sander Twitter data. In terms of accuracy, F-Measure, precision, and recall, the suggested technique performed effectively when compared to other TSA systems currently in use. Compared to the earlier techniques, the developed strategy increased classification precision and recall rate by about 3–15%. Future research will build a hybrid sentiment technique for other social media data, such Facebook and YouTube, to determine people's sentiments towards specific subjects, with the goal of improving the categorization rate.

With logistic regression, the result of a categorical dependent variable is predicted. As a result, a discrete or category value must be the result. Instead of providing the exact values, which are 0 and 1, it provides the probabilistic values, which fall between 0 and 1. It can be either Yes or No, 0 or 1, true or False, etc. VM selects the extreme vectors and points to aid in the creation of the hyperplane. The algorithm is referred regarded as a Support Vector

Machine since these extreme situations are known as support vectors. Hyperplane: To separate the classes in n-dimensional space, there might be more than one line or decision boundary. The key is to determine which choice boundary best aids in the classification of the data points. The SVM hyperplane is the name given to this optimal boundary.

CHAPTER-7

References

[1] VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text, by C.

[2] J. Hutto and E. E. Gilbert. The proceedings of the Eighth International Conference on Weblogs and Social Media (ICWSM14) were published in 2014.

[3] N. Bahrawi, "Twitter Streaming API Features for Online Real-Time Sentiment Analysis of Tweets," J. Penelit. Pos dan Inform., vol. 9, no. 1, pp. 53–62, 2019.

An ensemble sentiment classification system utilising Twitter data for airline services analysis, Y. Wan and Q. Gao, 2015.

[1] "Sentiment Analysis of Review Datasets using Naïve Bayes' and K-NN Classifier," by L. Dey, S. Chakraborty, A. Biswas, B. Bose, and S. Tiwari

In J. Teknol. Inf. ITSmart, vol. 2, no. 2, p. 35, 2016, F. Nurhuda, S. Widya Sihwi, and A. Doewes, "Analisis Sentimen Masyarakat terhadap Calon Presiden Indonesia 2014 berdasarkan Opini dari Twitter Menggunakan Metode Naive Bayes Classifier."

[1] A. Hamzah, "Sentiment Analysis To Maximise Saran Kuesioner In Learning Assessment By Utilising Naive Bayes Classifier (NBC)," 2014.

[2] D. Setyawan and E. Winarko, "Analisis Opini Terhadap Fitur Smartphone Pada Ulasan Website Berbahasa Indonesia," Journal of Computer-supported collaborative systems (IJCCS) vol. 10, no. 2, pp. 183–194, 2016.

Dynamic profiles combining sentiment analysis and Twitter data for voting advice apps is what Luis Terán and José Mancera have published [2].The 2019 edition of Government Information Quarterly.

Using big data analytics to examine brand authenticity sentiments: The instance of Starbucks on Twitter, H. Shirdastian, M. Laroche, and M.O. Richard, International Journal of Information Management, 2017.

[1] Sanjib Kumar Sahu and Himja Khurana. "Bat inspired sentiment analysis of Twitter data."Advancements in Intelligent Engineering and Advanced Computing. 2018; Singapore: Springer, pp. 639-650.

In Expert Systems with Applications, vol.71, pp.111–124, 2017, M. Daniel, R.F. Neves, and N. Horta published "Company event popularity for financial markets using Twitter and sentiment analysis."

[3] "Using Twitter trust network for stock market analysis," by Y. Ruan, A. Durresi, and L. Alfantoukh, Knowledge-Based Systems, vol.145, pp.207-218, 2018

[1] "Big data analytics sentiment: US-China reaction to data collection by business and government," by R.C. LaBrie, G.H. Steinke, X. Li, and J.A. Cazier, Technological Forecasting and Social Change, 2017.

[2] C.C. Aggarwal and H. Wang (2011). Social Network Text Mining. In: Social Network Data Analytics, edited by C.C. Aggarwal. [Virtual] Springer US, Boston, MA, pp. 353–378. The following URL is accessible: [As of March 10, 2020].

[1] Anon in 2017. An introduction to data mining's apriori algorithm using R for beginners.The HackerEarth Blog.Apriori algorithm data mining R implementation tutorial for beginners is available at . [As of March 14, 2020].

[2] Anon 2019. NLTK 3.5 documentation for the Natural Language Toolkit. [Virtual] accessible at: [As of December 6, 2019].

[3] Anon in 2019. [online] NumPy — NumPy Acquired from: [As of October 25, 2019].Anon 2020. Seth Grimes's A Brief History of Text Analytics - BeyeNETWORK. [Virtual] At , accessible [As of March 10, 2020].

[1] Feng, J., and Barbara, L. (2010). Entire Twitter sentiment identification from skewed and noisy data. p. 9.K.D. Foote, 2019. An Overview of Natural Language Processing History (NLP). Variety of Data. [https://www.dataversity.net/a-brief-history-of-natural-language-processing-nlp/] is the URL where this can be found. [As of March 9, 2020].