

# **Twitter Sentiment Analysis to understand the public opinion towards Indian Agriculture Ministry using kNN and RandomForest algorithms**

A Mini Project Report Submitted to

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY**

In partial fulfilment of the requirements for the award of the degree of

**BACHELOR OF TECHNOLOGY**

**IN**

**INFORMATION TECHNOLOGY**

Submitted By

**PRATISH MASHANKAR      17BD1A1251**

**P SREEVARSH VASISTA      17BD1A1241**

**PREETHI RANGANATHAN      17BD1A1252**

**RADHIKA BILOLIKAR      17BD1A1253**

Under the guidance of

***Dr. K. Bhargavi***

***(Head of Department, IT Dept.)***

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY**



(Approved by AICTE, Affiliated to JNTUH)

Narayanaguda, Hyderabad,

**Academic Year: 2020-2021**



---

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**CERTIFICATE**

This is to certify that this is a bonafide record of the project report titled “**Twitter Sentiment Analysis to understand the public opinion towards Indian Agriculture Ministry using kNN and Random Forest algorithms**” which is being presented as the Mini Project report by

<b>1. PRATISH MASHANKAR</b>	<b>17BD1A1251</b>
<b>2. P SREEVARSH VASISTA</b>	<b>17BD1A1241</b>
<b>3. PREETHI RANGANATHAN</b>	<b>17BD1A1252</b>
<b>4. RADHIKA BILOLIKAR</b>	<b>17BD1A1253</b>

In partial fulfilment for the award of the degree of **Bachelor of Technology in Information Technology** affiliated to the **Jawaharlal Nehru Technological University, Hyderabad**.

**Internal Guide**

**(Dr. K. Bhargavi)**

**Head of the Department**

**(Dr. K. Bhargavi)**

**Submitted for Viva Voce Examination held on \_\_\_\_\_**

**External Examiner**

## **Vision of KMIT**

Producing quality graduates trained in the latest technologies and related tools and striving to make India a world leader in software and hardware products and services. To achieve academic excellence by imparting in depth knowledge to the students, facilitating research activities and catering to the fast growing and ever- changing industrial demands and societal needs.

## **Mission of KMIT**

- To provide a learning environment that helps students to enhance problem solving skills, be successful in their professional lives and to prepare students to be lifelong learners through multi model platforms and educating them about their professional, and ethical responsibilities
- To establish Industry Institute Interaction to make students ready for the industry.
- To provide exposure to students to the latest tools and technologies hardware and software. the area of hardware and software
- To promote research-based projects/activities in the emerging areas of technology convergence.
- To encourage and enable students to not merely seek jobs from the industry but also to create new enterprises.
- To induce in the students a spirit of nationalism which will enable the student to develop and understand India's problems and to encourage them to come up with effective solutions for the same.
- To support the faculty in their endeavours to accelerate their learning curve in order to continue to deliver excellent service to students.

## **Vision & Mission of IT**

### **Vision of the IT**

Producing quality graduates trained in the latest software technologies and related tools and striving to make India a world leader in software products and services.

### **Mission of the IT**

To create a faculty pool which has a deep understanding and passion for algorithmic thought process.

To impart skills beyond university prescribed to transform students into a well-rounded IT professional.

To inculcate an ability in students to pursue Information technology education throughout their lifetime by use of multimodal learning platform including e-learning, blended learning remote testing and skilling,

Exposure to different domains, paradigms and exposure to the financial and commercial underpinning of the modern business environment through the entrepreneur development cell.

To encourage collaboration with various organizations of repute for research, consultancy and industrial interactions.

To create socially conscious and emotionally mature individuals with awareness on India's challenges, opportunities, their role and responsibility as engineers towards achieving the goal of job and wealth creation.

## **PROGRAM OUTCOMES (POs)**

- 1. Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- 2. Problem analysis:** Identify formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
- 3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
- 4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- 5. Modern tool usage:** Create select, and, apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
- 6. The engineer and society:** Apply reasoning informed by the contextual knowledge to societal, health, safety. legal und cultural issues and the consequent responsibilities relevant to professional engineering practice.
- 7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of, and need for sustainable development.
- 8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- 9. Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams and in multidisciplinary settings.

**10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

**11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

**12. Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

## **PROGRAM SPECIFIC OUTCOMES (PSOs)**

**PSO1:** An ability to analyse the common business functions to design and develop appropriate Information

Technology solutions for social upliftment.

**PSO2:** Shall have expertise on the evolving technologies like Mobile Apps, CRM, ERP, Big Data, etc.

## **PROGRAM EDUCATIONAL OBJECTIVES (PEOs)**

**PEO1:** Graduates will have successful careers in computer related engineering fields or will be able to successfully pursue advanced higher education degrees.

**PEO2:** Graduates will try and provide solutions to challenging problems in their profession by applying computer engineering principles.

**PEO3:** Graduates will engage in life-long learning and professional development by rapidly adapting to the changing work environment.

**PEO4:** Graduates will communicate effectively, work collaboratively and exhibit high levels of professionalism and ethical responsibility.

## PROJECT OUTCOMES

**P1:** Compare the performance of k-Nearest Neighbors algorithm and Random Forest algorithm with different number of features

**P2:** Collection of Ministry of Agriculture Govt. of India dataset from Twitter through Tweepy API

**P3:** Classify the sentiment of tweets towards the Indian Ministry of Agriculture as positive, negative or neutral

**P4:** To visualize how the Agricultural ministry is perceived by the public and to better understand its effectiveness according to public opinion

L - LOW

M – MEDIUM

H – HIGH

### PROJECT OUTCOMES MAPPING PROGRAM OUTCOMES

PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
P1	H	H	H		M				H	H		H
P2	M				H	H		H	H	H		
P3	H	H	H	H	M	H		H	H	H		M
P4	M				H	H			H	H		H



### PROJECT OUTCOMES MAPPING PROGRAM SPECIFIC OUTCOMES

PSO	PSO1	PSO2
P1	L	H
P2	L	H
P3	H	H
P4	H	L

### PROJECT OUTCOMES MAPPING PROGRAM EDUCATIONAL OBJECTIVES

PEO	PEO1	PEO2	PEO3	PEO4
P1	H	H	H	H
P2	M	H	M	L
P3	H	H	H	M
P4	L	M	M	H

## DECLARATION

We hereby declare that the results embodied in the dissertation entitled “**Twitter Sentiment Analysis to understand the public opinion towards Indian Agriculture Ministry using k Nearest Neighbour and Random Forest algorithms**” has been carried out by us together during the academic year 2020-2021 as a partial fulfilment of the award of the B.Tech degree in Information Technology from JNTUH. I have not submitted this report to any other university or organization for award of any other degree.

<b>Student Name</b>	<b>Roll No.</b>
<b>PRATISH MASHANKAR</b>	<b>17BD1A1251</b>
<b>P SREEVARSH VASISTA</b>	<b>17BD1A1241</b>
<b>PREETHI RANGANATHAN</b>	<b>17BD1A1252</b>
<b>RADHIKA BILOLIKAR</b>	<b>17BD1A1253</b>

## ACKNOWLEDGEMENT

The satisfaction and euphoria that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant encouragement and guidance have been a source of inspiration throughout the course of the project work. I am glad to have the opportunity of expressing our gratitude to all of them.

We are thankful to **Dr. K. BHARGAVI**, Professor and Head of the Department of Information Technology, Keshav Memorial Institute of Technology. We thank her for her support and guidance throughout the project. She is a source of inspiration for innovative ideas and her kind support is well known to all her students and colleagues.

We would like to express our sincere gratitude to Technical Faculty, Mr. G. Rakesh Reddy, Assistant Professor and project coordinators for their support, guidelines and encouragement.

We also express our sincere gratitude to the Management and Principal of Keshav Memorial Institute of Technology for their encouragement, facilities provided and support.

Finally, we would like to make a special mention of all our family members and friends who helped us for the successful completion of the project report.

<b>Student Name</b>	<b>Roll No.</b>
<b>PRATISH MASHANKAR</b>	<b>17BD1A1251</b>
<b>P SREEVARSH VASISTA</b>	<b>17BD1A1241</b>
<b>PREETHI RANGANATHAN</b>	<b>17BD1A1252</b>
<b>RADHIKA BILOLIKAR</b>	<b>17BD1A1253</b>

# CONTENTS

DESCRIPTION	PAGE NO.
ABSTRACT	i
LIST OF DIAGRAMS	ii
LIST OF SCREENSHOTS	iii
LIST OF TABLES	iv
<b>CHAPTER - 1</b>	<b>1</b>
<b>1. INTRODUCTION</b>	<b>2</b>
1.1 Introduction to Twitter Sentiment Analysis	2
1.2 Problem with the Existing System	3
1.3 Proposed System	4
1.4 Working Principle	4
<b>CHAPTER - 2</b>	<b>7</b>
<b>2. SOFTWARE REQUIREMENT SPECIFICATIONS</b>	<b>8</b>
2.1 Introduction	9
2.2 Requirements specification document	10
2.3 Software requirements for Twitter Sentiment Analysis using KNN and Random Forest algorithms	10
2.3.1 Purpose	10
2.3.2 Intended Audience	10
2.3.3 Intended Use	10

2.3.4 Functional Requirements	10
2.3.5 Non Functional Requirements	11
2.3.6 System Features: Software Specifications	12
2.3.7 System Features: Hardware Specifications	13
 <b>CHAPTER - 3</b>	 <b>14</b>
 <b>3. LITERATURE SURVEY</b>	 <b>15</b>
 <b>CHAPTER - 4</b>	 <b>17</b>
 <b>4. SYSTEM DESIGN</b>	 <b>18</b>
4.1 UML Diagrams	18
4.1.1 Use Case Diagram	19
4.1.2 Context Diagram	21
4.1.3 Data Flow Diagram	23
4.1.4 Sequence Diagram	25
4.1.5 Activity Diagram	27
4.1.6 Class Diagram	29
4.1.7 Deployment Diagram	31
4.2 Technology And Tools Used	33
4.2.1 Term Frequency- Inverse Document Frequency Vectorizer	33
4.2.2 Confusion matrix	35
4.2.3 Classification Report	36
4.2.4 K Nearest Neighbors Algorithm	37
4.2.5 Random Forest Algorithm	39

## **CHAPTER - 5** **41**

### **5. IMPLEMENTATION AND CODE**

#### **SNIPPETS** **42**

5.1 Stage 1: Comparing K-Nearest Neighbors Algorithm  
and Random Forest algorithm **42**

5.2 Tabulating kNN and Random Forest Observations **52**

5.3 Results of the evaluation of kNN and Random Forest  
algorithms **53**

5.4 Stage 2: Extraction & Cleaning of Target Dataset:  
Tweets on Ministry of Agriculture **53**

5.5 Stage 3: Implementation Using Target Dataset **58**

5.6 Observations **66**

## **CHAPTER - 6** **68**

### **6. SOFTWARE TESTING** **69**

6.1 Testing **69**

6.2 Test Cases **71**

#### **BENEFITS** **77**

#### **FUTURE ENHANCEMENTS** **78**

#### **CONCLUSION** **79**

#### **REFERENCES** **80**

#### **BIBLIOGRAPHY** **81**

## ABSTRACT

Sentiment Analysis is the process of computationally determining whether the given text data is positive, negative or neutral. This analysis when done on the Tweets gathered from Twitter is called Twitter Sentiment Analysis. This analysis is used in many different fields such as politics, business etc. As of 2018, Twitter had more than 330 million monthly active users, with over 500 million tweets per day, thus the corpus for analysis is ample in amount. This project deals with the analysis of the Agricultural Ministry of Govt. Of India by analysing the feeling or sentiment behind the people expressing their views on twitter. The aim is to output how the Agricultural Ministry has performed and determine the response of the people. This project uses **Twitter API - Tweepy**, in order to gather the data from twitter, for which user authentication by Twitter is necessary. The initial step of the project is Data Collection, by using the Twitter API and collecting the data with the use of Cursor Object and its search parameters which determines the keywords to be searched for and the radius of the search, where the radius being 1607 kms, half the length of the nation. Second step being Data Cleaning, the gathered data is then cleaned to extract various features. Feature vectorization using **TFIDF Vectorizer** is performed to convert text to numbers. Machine Learning Algorithms namely **Random Forest Classification algorithm** and **kNN** are trained using **Twitter and Reddit Sentimental analysis Dataset** from Kaggle to determine the accuracy of each algorithm. Library functions of **scikit learn 2.0** are used for train-test splitting. Better algorithm among the two with higher accuracy rate is employed to output sentiment graphs which denote the percentage of positive, negative and neutral responses of the public towards the Agricultural ministry. This determines the public sentiment towards the Agricultural Ministry of India.

## LIST OF DIAGRAMS

<b>S.NO</b>	<b>Name Of the Diagram</b>	<b>Page No.</b>
<b>1</b>	Flowchart of the project	<b>5</b>
<b>2</b>	Use Case Diagram	<b>21</b>
<b>3</b>	Context Diagram	<b>23</b>
<b>4</b>	Data Flow Diagram (DFD)	<b>25</b>
<b>5</b>	Sequence Diagram	<b>26</b>
<b>6</b>	Activity Diagram	<b>28</b>
<b>7</b>	Class Diagram	<b>30</b>
<b>8</b>	Deployment Diagram	<b>32</b>
<b>9</b>	Flowchart for k Nearest Neighbors Algorithm	<b>38</b>
<b>10</b>	Flowchart for RandomForest Algorithm	<b>40</b>
<b>11</b>	Line Graph comparing kNN and RandomForest Algorithms	<b>66</b>
<b>12</b>	Pie Chart depicting public sentiment towards agricultural ministry	<b>67</b>



## LIST OF SCREENSHOTS

<b>S.NO</b>	<b>Name Of the Diagram</b>	<b>Page No.</b>
<b>1</b>	Comparing KNN and Random Forest algorithms	<b>42</b>
<b>2</b>	Output of KNN using 250 features	<b>44</b>
<b>3</b>	Output of Random Forest using 250 features	<b>46</b>
<b>4</b>	Output of KNN using 500 features	<b>47</b>
<b>5</b>	Output of Random Forest using 500 features	<b>48</b>
<b>6</b>	Output of KNN using 750 features	<b>49</b>
<b>7</b>	Output of Random Forest using 750 features	<b>50</b>
<b>8</b>	Extraction and Cleaning	<b>53</b>
<b>9</b>	Implementation using Target Dataset	<b>58</b>

## LIST OF TABLES

<b>S.No</b>	<b>Name of the Table</b>	<b>Page No.</b>
<b>1</b>	Software Specifications	<b>13</b>
<b>2</b>	Hardware Specifications	<b>14</b>
<b>3</b>	KNN and Random Forest Observations	<b>53</b>

# CHAPTER - 1

# 1. INTRODUCTION

## 1.1 Introduction to Sentiment Analysis

Twitter is a social networking platform with over 330 million active users across the globe. This number of users produce, on average, 350,000 tweets about various topics per minute all over the world. In India, there are more than 500 million tweets posted each day. Apart from posting, users can also perform actions such as liking and retweeting. Thus, it is clear that the corpus obtained from twitter under consideration is humongous and can provide satisfactory results for any demographic analysis. One such analysis is defined by a Twitter user's reaction or 'sentiment' expressed through a tweet, known as the Twitter Sentiment analysis.

Sentiment Analysis, which is also known as opinion mining or emotion AI, refers to the natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information. The main advantage of Sentiment Analysis is that it can be widely applied to understand customer reviews and survey results. The most essential task in sentiment analysis is to classify the polarity of a given text in the document, sentence, or feature/aspect level—we classify the given tweets into one of three categories, namely positive, negative and neutral. When this sentimental analysis is employed to analyse the tweets and determine their polarity, it is defined as Twitter Sentiment Analysis.

There are many uses of Twitter Sentiment Analysis, namely that it provides a wide range of use-cases ranging from politics to business. With the increased number of Netizens, a term defined for users of the Internet, and their subsequent involvement in politics through public opinion, there is a huge amount of data recorded in the form of tweets directed towards various ministries.

To begin with the gathering of the tweets from the Twitter database, it will be required to first send an application to the Twitter Team to grant the Developer access in order to use the Tweepy API. Tweepy is the Twitter API which helps to gather, and use the data of the tweets which have been tweeted by the users. Once the authentication is completed by the Twitter team and the access has been granted, four access keys from Twitter are received. After this, one can proceed with the data collection.

In this project, Random Forest Algorithm and k Nearest Neighbours algorithm are employed for classifying the sentiments of Twitter users. The KNN algorithm uses 'feature similarity' to predict the values of any new data points. This means that the new point is assigned a value based on how closely it resembles the points in the training set. The Random Forest algorithm builds multiple decision trees and merges them together to get a more accurate and stable prediction.

## **1.2 Problems with the Existing System**

- The current systems employ SVM and KNN algorithms in order to determine techniques to classify the sentiment label as accurately as possible. It introduces two methods: sentiment classification algorithm (SCA) based on k-nearest neighbor (KNN) and the support vector machine (SVM). Along with that, it evaluates their performance based on tweets. However the low accuracy of both the methods doesn't provide satisfactory results
- Another study conducts a real-time sentiment analysis of the public reaction towards the announcement of the Indian Union Budget 2020, or on only a few bills that the government has passed. This does not give any idea about the progress that the individual ministry is achieving. On social media platforms, the general public vents their opinions regarding popular issues, which includes various reforms that the state or the central government has implemented. The current system only gives a vague idea about the opinion or feeling of the people regarding some of the issues.

### 1.3 Proposed System

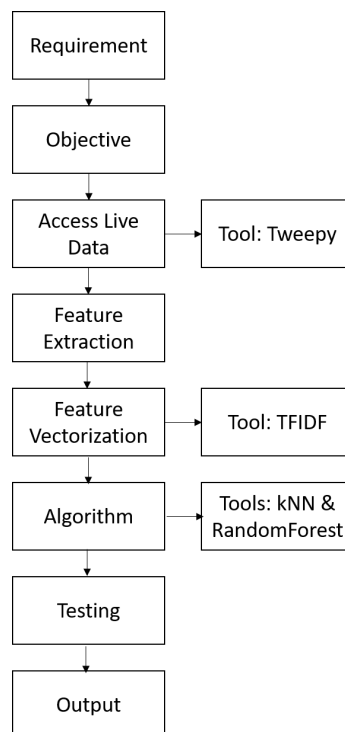
- The proposed system is to perform sentiment analysis on the latest Tweets on the Agricultural Ministry of India, using the Twitter API in order to analyse the public's opinion on the Agricultural Ministry by classifying them as either positive, negative or neutral by using machine learning classification algorithms: the RandomForest classifier and k Nearest Neighbors algorithm.
- The project is intended to showcase the opinion of the people on the Agricultural Ministry. This gives an idea as to how the changes implemented by the ministry are perceived and felt by the public. The Twitter Sentiment Analysis for the above mentioned ministry provides the data which can provide the knowledge of the change in public's opinion for the ministry.

### 1.4 Working Principle

- Twitter Sentiment Analysis determines the so called mood of the statement on which the analysis is being performed. It classifies the given statement into 3 categories:
  - Positive - when the value is 1.0
  - Neutral - when the value is 0.
  - Negative - when the value is -1.0
- Effectiveness of the Ministry of Agriculture of India is analysed by passing the dataset gathered by the Tweepy API calling it as Target Data, to a trained Machine learning algorithm in order to determine the sentiment of Twitter users about various initiatives taken by the respective Ministry
- This model analyses the Tweets by various users within the country with the help of a search location parameter within the Cursor class object, defined with a radius of 1607 kms from the city of Nagpur, the geographical centre of the country.
- This whole process is performed in three stages.
- In the first stage, the main aim is to compare the accuracy scores of Random

Forest and k-Nearest Neighbors algorithms. It begins by uploading the **Twitter and Reddit Sentimental analysis Dataset** from Kaggle. This is the **Training Dataset**. This dataset undergoes preprocessing and then goes through a TFIDF vectorizer. Then the different accuracy scores obtained by both Random Forest and KNN algorithms by using different features, are compared. Then the algorithm which provides a better accuracy score among the two is chosen.

- In the second stage, it is necessary to request access to tweets by using the Tweepy API. Twitter then gives the access keys namely the Consumer key, Consumer Secret, Access Token and Access Token Secret. This is followed by creating a dataset of tweets related to the sector of agriculture. The relevant tweets are extracted by using the right search words. This is followed by performing preprocessing, cleaning and feature vectorization on the resulting dataset. This is the **Target Dataset**. Assume, upon TFIDF vectorization, the Agriculture Dataset consists of total  $Fn$  features.



*Skeleton flowchart of working principle*

- Next, the third stage. Here, consider the algorithm which has been chosen from the first stage. Upload the Training Dataset which was employed in the first stage and perform the preprocessing and feature vectorization. During feature vectorization, give the TFIDF parameter *max\_features* a value of  $F_n$  since all the features from the Target Dataset are considered. Pass the vectorized Training Dataset with  $F_n$  features to the optimal algorithm. Depict the output in the form of a confusion matrix and classification report, and display the accuracy score of the updated Training Dataset with  $F_n$  features. After training, pass the value of Target (Agriculture) Dataset to the trained Machine Learning Model to find the sentiment of all tweets present in the dataset. Output a graphical representation of the overall public sentiment towards the Agricultural Ministry.



## CHAPTER - 2

## **2. SOFTWARE REQUIREMENTS SPECIFICATION(SRS)**

### **2.1 Introduction**

#### **What is SRS?**

A software requirements specification (SRS) is a document that describes what the software will do and how it will be expected to perform. It also describes the functionality the product needs to fulfill all stakeholders (business, users) needs.

A typical SRS includes:

- A purpose
- An overall description
- Specific requirements

The best SRS documents define how the software will interact when embedded in hardware — or when connected to other software. Good SRS documents also account for real-life users.

The SRS is a specification for a specific software product, program, or set of applications that perform particular functions in a specific environment. It serves several goals depending on who is writing it. First, the SRS could be written by the client of a system. Second, the SRS could be written by a developer of the system. The two methods create entirely various situations and establish different purposes for the document altogether. The first case, SRS, is used to define the needs and expectations of the users. The second case, SRS, is written for various purposes and serves as a contract document between customer and developer. The SRS phase consists of two basic activities:

1) Problem/Requirement Analysis The process is orderly and more nebulous of the two, deals with understanding the problem, the goal and constraints.

2) Requirement Specification Here, the focus is on specifying what has been found giving analysis such as representation, specification languages and tools, and checking the specifications are addressed during this activity. The Requirement phase terminates with the production of the validate SRS document. Producing the SRS document is the basic goal of this phase.

### **What is the role of SRS?**

- A software requirements specification is the basis for your entire project. It lays the framework that every team involved in development will follow.
- It's used to provide critical information to multiple teams — development, quality assurance, operations, and maintenance. This keeps everyone on the same page.
- Using the SRS helps to ensure requirements are fulfilled. And it can also help you make decisions about your product's lifecycle — for instance, when to retire a feature.
- Writing an SRS can also minimize overall development time and costs. Embedded development teams especially benefit from using an SRS.

## **2.2 REQUIREMENTS SPECIFICATION DOCUMENT**

A Software Requirements Specification (SRS) is a document that describes the nature of a project, software or application. In simple words, SRS document is a manual of a project provided it is prepared before you kick-start a project/application. This document is also known by the name SRS report, software document. A software document is primarily prepared for a project, software or any kind of application. There are a set of guidelines to be followed while preparing the software requirement specification document. This includes the purpose, scope, functional and nonfunctional requirements, software and hardware requirements of the project. In addition to this, it also contains the information about environmental conditions required, safety and security requirements, software quality attributes of the project etc. The purpose of SRS (Software Requirement Specification) document is to describe the external behavior of the application developed

or software. It defines the operations, performance and interfaces and quality assurance requirements of the application or software. The complete software requirements for the system are captured by the SRS. This section introduces the requirement specification document for Twitter Sentiment Analysis which enlists functional as well as non-functional requirements.

## **2.3 SRS for Twitter Sentiment Analysis using kNN and RandomForest Algorithms**

### **2.3.1 Purpose**

The purpose of the project is to compare the accuracy of KNN and Random Forest classifiers and to analyze the public sentiment towards the Indian agriculture ministry.

### **2.3.2 Intended Audience**

The intended audience includes the public and the Ministry of Agriculture of India.

### **2.3.3 Intended Use**

This project is intended to showcase how well the concerned ministry has worked and how the general public perceived this work. This helps us understand how well the particular ministry is working in accordance with the people's expectations.

### **2.3.4 Functional Requirements**

In systems engineering and requirements engineering, Functional requirements define the basic system behaviour. They are what the system does and can be defined as how the system responds to inputs. It usually defines if/then behaviours and includes calculations, data input, and business processes.

## USER

- The system should allow the user to analyse the positive, negative and neutral views of the agriculture ministry of the government.
- It should aid the user to determine how efficient the Indian Agricultural ministry has been and understand its impact on the people of the country.
- It should help to strengthen the public opinion by providing satisfactory analysis of the ministry

### **2.3.5 Non-functional Requirements**

In systems engineering and requirements engineering, a non-functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than behaviors. They are contrasted with functional requirements that define specific behavior or functions.

- **Reusability**-Reusability is the use of existing assets in some form within the software product development process; these assets are products and by-products of the software development life cycle and include code, software components, test suites, designs and documentation.

In this project, the code is being consistently reused for analysing and classifying the tweets into different sentiments.

- **Security-**

The software must remain resilient in the face of attacks. The behavior of the software must be correct and predictable. The software must be available and behave reliably even under DOS attacks. The software must ensure the integrity of the customer account information. The system is secured enough for the interface.

- **Portability**-Portability in high-level computer programming is the usability of the same software in different environments. The pre-requirement for portability is the generalized abstraction between the application logic and system interfaces. When software with the same functionality is produced for several computing platforms, portability is the key issue for development cost reduction. This system is compatible enough to perform on any platform.
- **Maintainability**- Maintainability refers to the ease with which you can repair, improve and understand software code that starts after the customer has received the product. This includes fixing bugs, optimizing existing functionality and adjusting code to prevent future issues. This system is easy to maintain.
- **Scalability**- Assesses the highest workloads under which the system will still meet the performance requirements. It is the measure of system ability to increase or decrease in performance and cost in response to changes in application and system processing demands. This system is scalable and large enough to analyse large amounts of data(tweets)

### 2.3.6 System Features: Software Specifications

<b>Operating System</b>	64-bit operating system, x64-based processor
<b>Programming IDE's</b>	Python
<b>Built with</b>	<ul style="list-style-type: none"> <li>• Google collab</li> <li>• Python 3</li> </ul>
<b>Prerequisites</b>	<ul style="list-style-type: none"> <li>• Python(Google collab)</li> <li>• Kaggle Twitter Database</li> <li>• Twitter Authentication Key</li> </ul>

### 2.3.6 System Features: Hardware Specifications

<b>Installed RAM</b>	8.00 GB
<b>Processor</b>	Intel(R) Core(TM) i5-7200U CPU @2.5GHz 2.71GHz
<b>Hard Disk Space</b>	70 GB or more
<b>Input Device</b>	Keyboard and mouse
<b>Internet Connection</b>	Network using 10 Mbps or higher speed

## **CHAPTER - 3**



### 3. LITERATURE SURVEY

Twitter Sentiment Analysis has been performed by various researchers and a lot of work has gone into it. It has been done to analyse the polarity of text. That is, it determines whether the tone of the textual writing in consideration is positive, negative or neutral. Studies have shown that social media platforms such as Twitter generate huge amounts of text containing political insights, which can be mined to analyze the public's opinion and predict the future trends in the elections. In this work, an attempt was made to mine tweets, capture the political sentiments from it and model it as a supervised learning problem<sup>[1]</sup>.

Another research applied Sentiment Analysis to define Twitter political users' classes and their homophily during the 2016 American presidential election. They collected 4.9 million tweets of 18,450 users and their contact network from August 2016 to November 2016. Next, they defined six user classes regarding their sentiment towards Donald Trump and Hillary Clinton. The users' political homophily was analysed and it was concluded that the negative users had homophily<sup>[2]</sup>.

In another study, they summarized the data set of Twitter messages related to the 14th Gujarat Legislative Assembly election, 2017 for predicting the chances of a party winning the elections by utilizing public's opinion. They used NRC Emotion Lexicon to determine the overall tone of the event by eight emotions and a Deep Learning tool named ParallelDots AI APIs by ParallelDots Inc that can analyze the sentiment into positive, negative and neutral. This tool helped to extract various peoples' sentiment and summarized the results for further decision making<sup>[3]</sup>.

Another study was done to understand the public's sentiment regarding gun policies. The study used framing theory, which assumed that both the news media and individuals use frames to construct perceptions and narratives about issues. Adopting an automated

content analysis as a method, the study examined 354 gun-related tweets downloaded from the Twitter accounts of three Arkansas politicians and 40 news articles about gun policy involving these politicians from three local newspapers. The results showed that political sentiment on Twitter was extremely negative while news media expressed a very neutral sentiment in their coverage of gun policy, suggesting a new venue for further investigation of current assumptions about the negative nature of news tone<sup>[4]</sup>.

## **CHAPTER - 4**

## 4. SYSTEM DESIGN

### 4.1 UML DIAGRAMS

Unified Modelling Language is a standard language used for specifying, visualizing, constructing, and documenting the artifacts of software systems. This language was created by the Object Management Group (OMG). The UML 1.0 specification draft was proposed to the OMG in January 1997. Ever since then, the OMG is consistently making efforts to create an industry standard.

1. UML stands for Unified Modelling Language.
2. UML is different from the other common programming languages such as C++, Java, COBOL, etc.
3. It is a pictorial language used to depict software in a blueprint format.
4. UML can be described as a general-purpose visual modelling language to visualize, specify, construct, and document software system.
5. Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc.

UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object-oriented analysis and design. After some standardization, UML has become an OMG standard

### **4.1.1 USE CASE DIAGRAM**

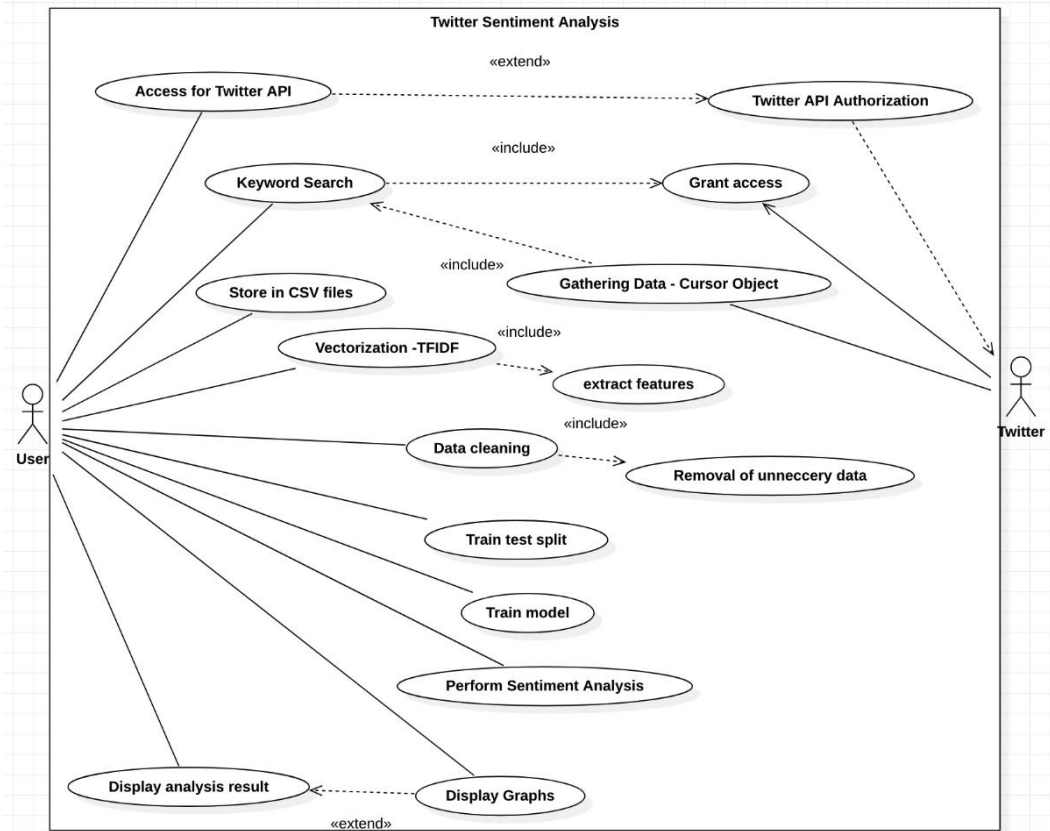
To model a system, the most important aspect is to capture the dynamic behavior. To clarify a bit in detail, dynamic behavior means the behavior of the system when it is running /operating.

So only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior. In UML there are five diagrams available to model dynamic nature and use case diagrams is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. So, use case diagrams consist of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system. So, to model the entire system numbers of use case diagrams are used.

#### **Purpose of Use Case Diagram**

- It is used to gather the requirements of a system.
- It is used to get an outside view of a system.
- It helps to identify the external and internal factors influencing the system.
- It shows the interaction among the requirements of the actors.



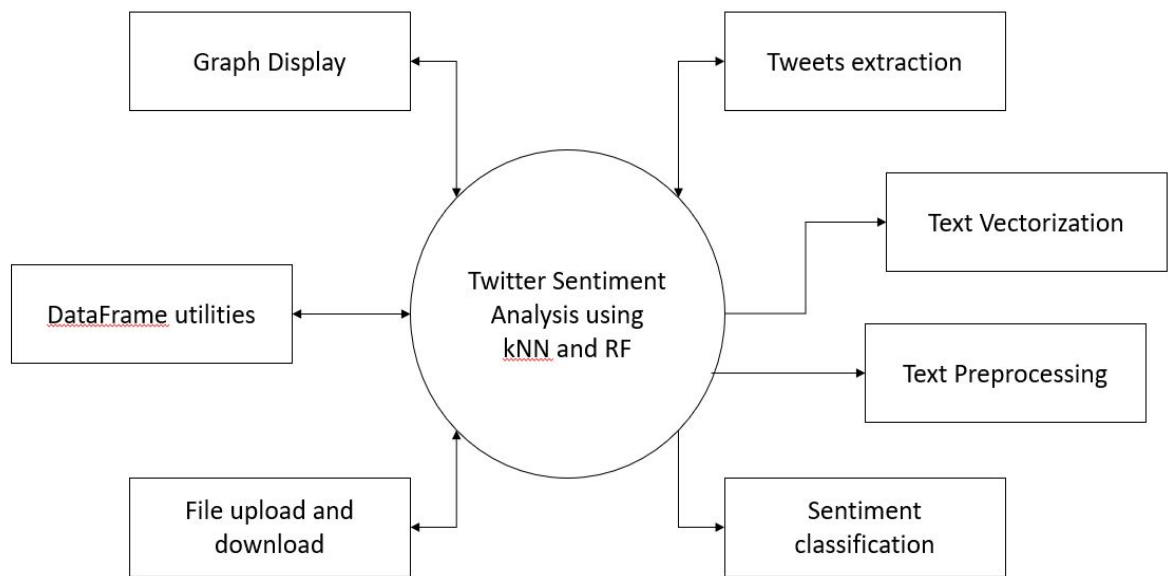
*Usecase diagram for Twitter Sentiment Analysis using kNN and RF*

#### **4.1.2 CONTEXT DIAGRAM :**

A context diagram is drawn in order to define and clarify the boundaries of the software system. It is sometimes called a level 0 data-flow diagram. It identifies the flows of information between the system and external entities.

##### **Purpose of Context Diagram**

- It shows the scope and boundaries of a system at a which includes the other systems that interface with it
- There is no technical knowledge required to understand the diagram
- It is easy to draw and amend due to its limited notation
- It benefits a wide audience including stakeholders, business analyst, data analysts, developers



*Context diagram for Twitter Sentiment Analysis using  $kNN$  and RF*

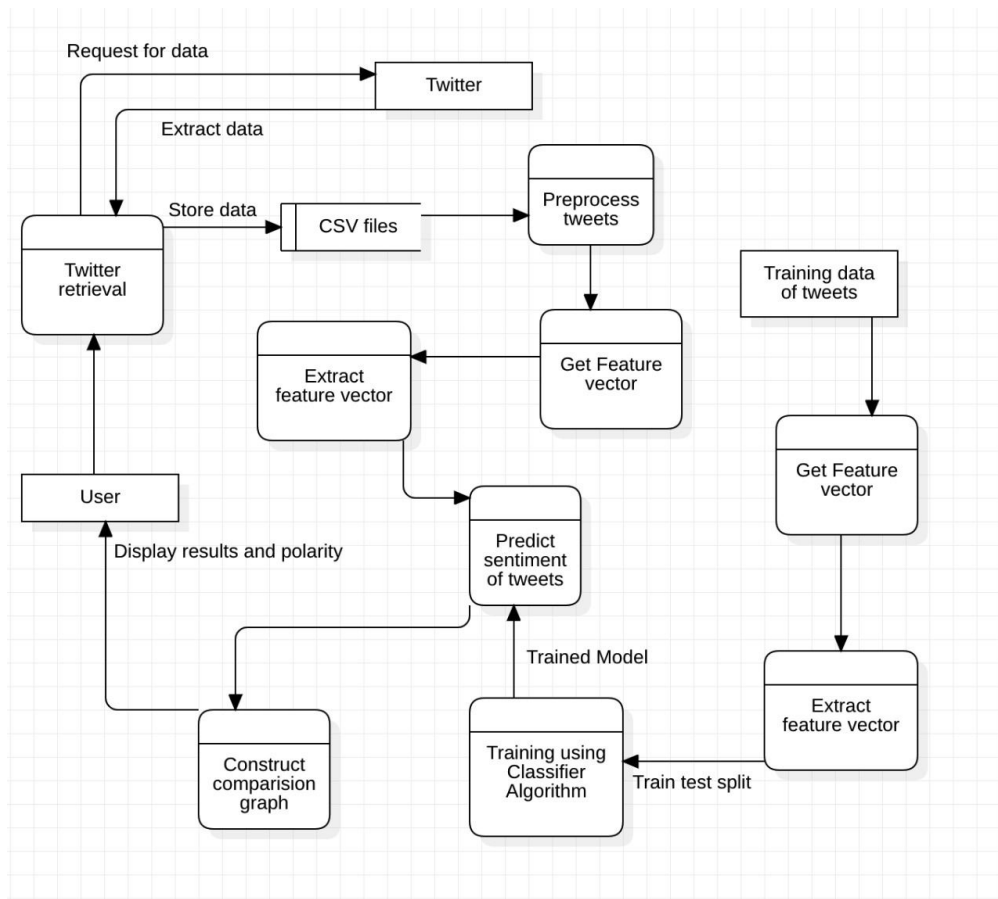


#### **4.1.3 DFD DIAGRAM :**

Data flow diagram represents the way in which the data flows from one process to another. This gives us an idea about how the control in the project goes from one process to another so that the user gets an idea as to what he/she would get as a result after a particular process.

#### **Purpose of DFD Diagram**

- Show the scope and boundaries of a system as a whole
- Can be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system.
- The DFD is also called a data flow graph or bubble chart.



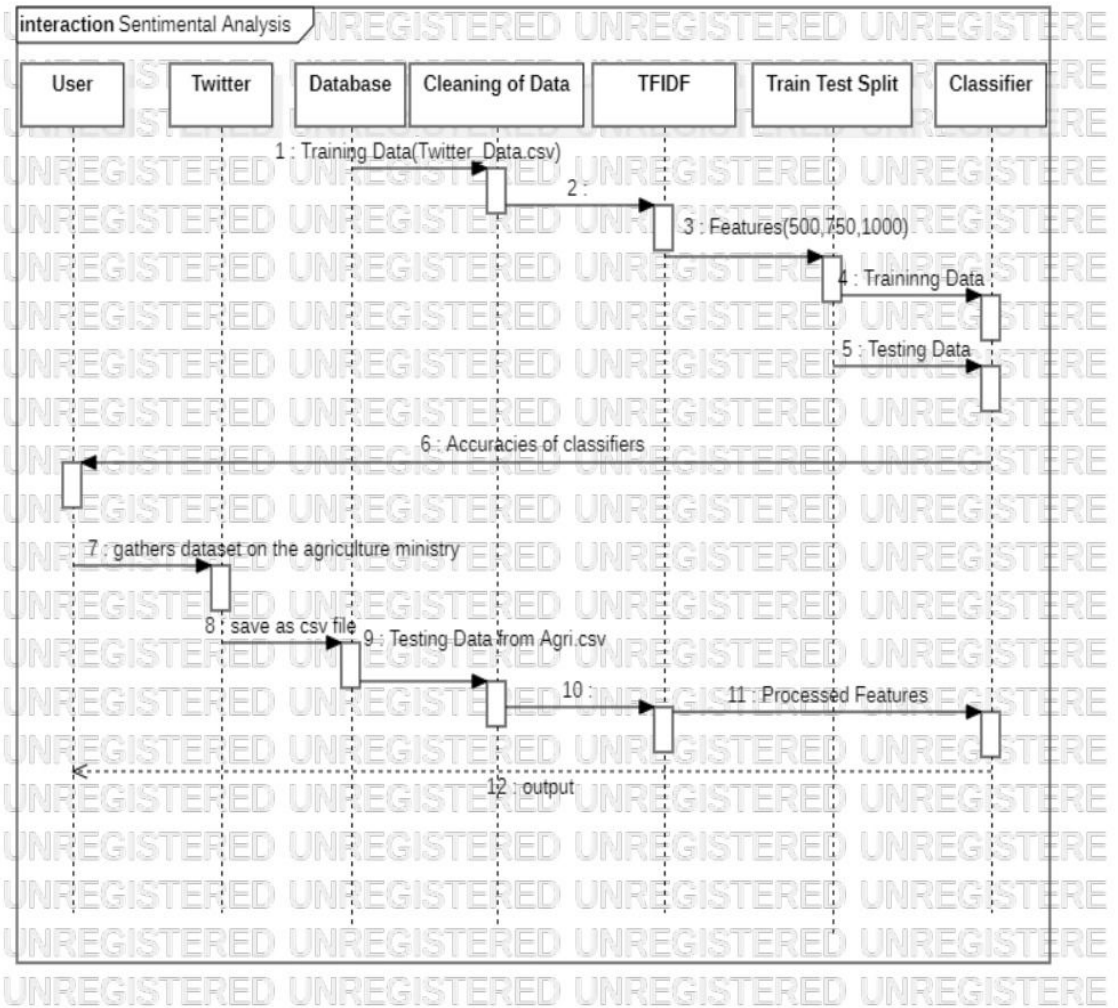
*DFD for Twitter Sentiment Analysis using kNN and RF*

#### **4.1.4 SEQUENCE DIAGRAM**

Sequence diagrams is a diagram which describes interactions among classes in terms of an exchange of messages over time. They're also called event diagrams. A sequence diagram is a good way to visualize and validate various runtime scenarios. These can help to predict how a system will behave and to discover responsibilities a class may need to have in the process of modeling a new system. The aim of a sequence diagram is to define event sequences, which would have a desired outcome. The focus is more on the order in which messages occur than on the message per se. However, the majority of sequence diagrams will communicate what messages are sent and the order in which they tend to occur.

#### **Purpose of Sequence Diagram**

- It is used for high-level interaction between active objects in a system.
- It is used to model the interaction between object instances within a collaboration that realizes a use case.
- It is used to model the interaction between objects within a collaboration that realizes an operation.
- It either shows model generic interactions (showing all possible paths through the interaction) or specific instances of an interaction (showing just one path through the interaction).



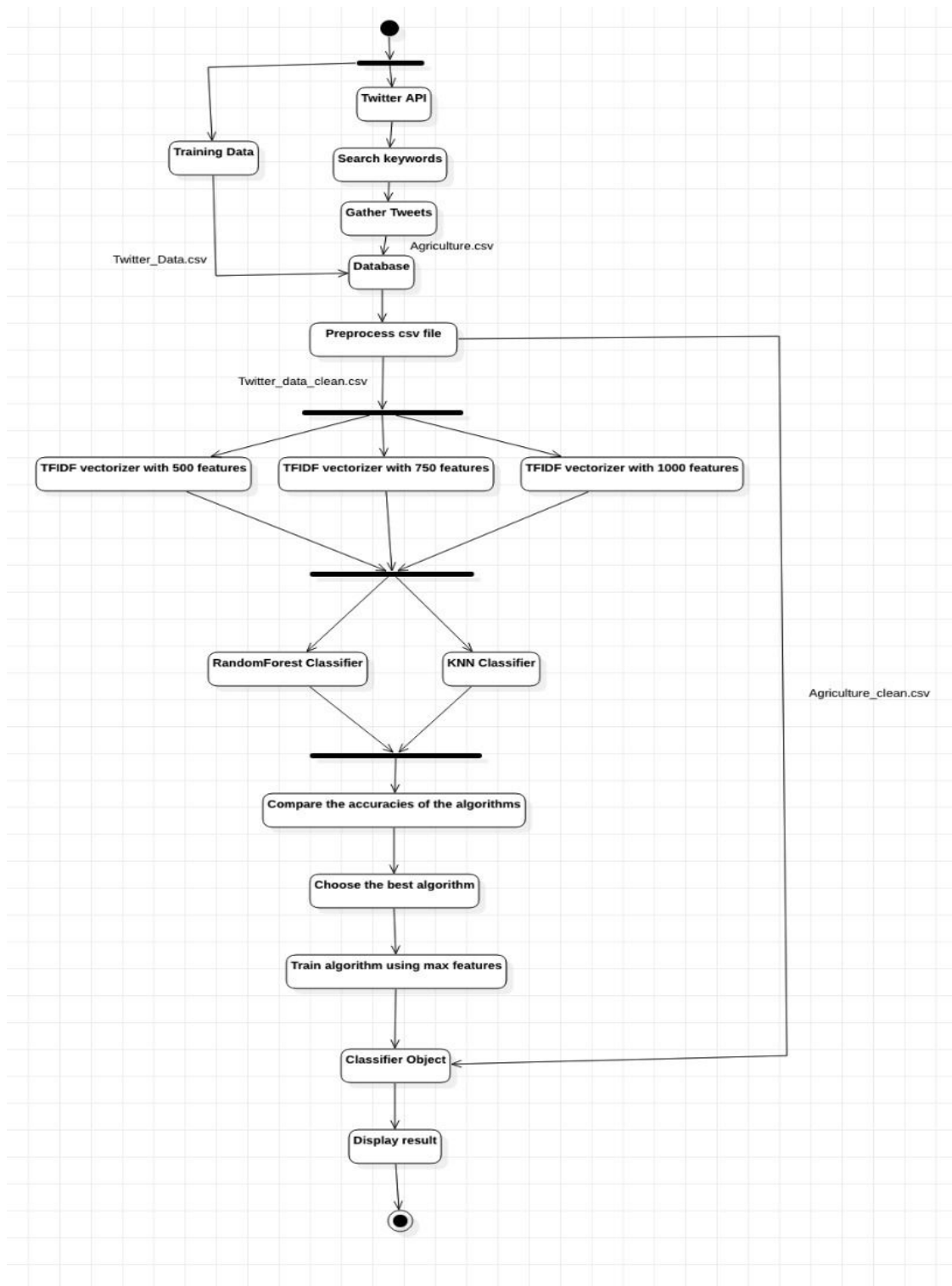
*Sequence diagram for Twitter Sentiment Analysis using kNN and RF*

#### **4.1.5 ACTIVITY DIAGRAM**

Activity diagram is a behavioral diagram in UML diagrams. It is used to describe the dynamic aspects of the system. It represents the flow from one activity to another. They are used to construct executable systems by using forward and reverse engineering techniques.

##### **Purpose of Activity Diagram**

- Draw the activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.



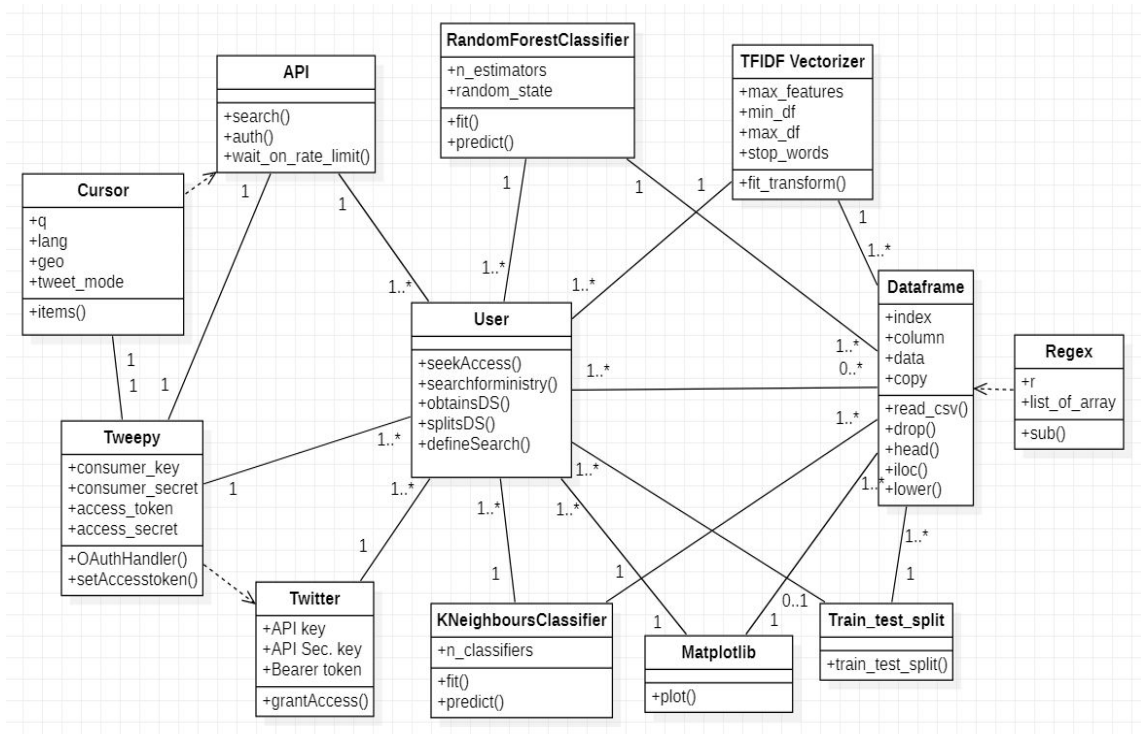
*Activity diagram for Twitter Sentiment Analysis using kNN and RF*

#### **4.1.6 CLASS DIAGRAM**

In UML, the class diagram is a static structural diagram which describes the system's structure. It is implemented using the system classes. It also describes the system's attributes, operations and the relationship between the objects of the classes. It provides basic notation ideas for the structural diagrams in the UML.

##### **Purpose of Class Diagram**

- It is used for analysis and design of the static view of an application.
- It describes responsibilities of a system.
- It is considered as the base for component and deployment diagrams.
- It is used for forward and reverse engineering.



*Class diagram for Twitter Sentiment Analysis using kNN and RF*

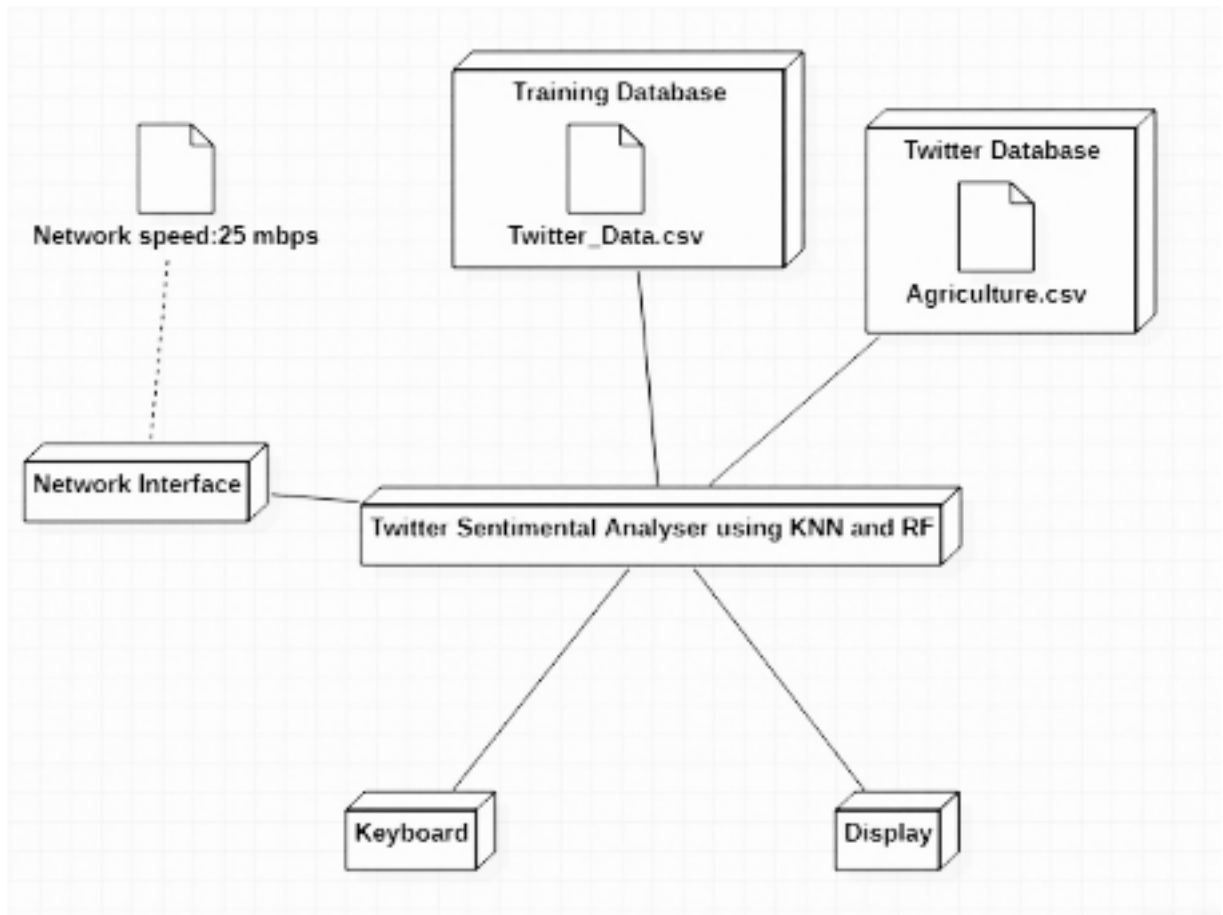


#### **4.1.7 DEPLOYMENT DIAGRAM**

Deployment diagrams are structural diagrams that show the components and shows the configuration of run time processing nodes. It is used to model the physical aspects of the object-oriented system. They are used to model the static deployment view of the system.

##### **Purpose of Deployment Diagram**

- It visualizes the hardware topology of a system.
- It is used to describe the hardware components used to deploy software components.
- It describes the runtime processing nodes.



*Deployment diagram for Twitter Sentiment Analysis using kNN and RF*

## 4.2 TECHNOLOGY AND TOOLS USED

### 4.2.1 TERM FREQUENCY - INVERSE DOCUMENT FREQUENCY VECTORIZER

**TF-IDF** stands for Term Frequency — Inverse Document Frequency transforms text to the feature vectors where feature vectors are used as an input for estimators. It defines how important a word is for a document, as well as taking into account the relation to other documents from the same corpus. This is performed by looking at the count of a word that appears into a document while paying attention to no. of times the same word appears in other documents in the corpus. The motive behind this is the following:

- a word that frequently appears in a document has more relevance for that document, i.e., there is a higher probability that the document is about or in relation to that specific word.
- a word that frequently appears in more documents may prevent us from finding the right document in a collection; the word is relevant either for all documents or for none. Either way, it will not help us filter out a single document or a small subset of documents from the whole set.

**TF-IDF is a score** which is applied to every word in every document in our dataset. For every word, the TF-IDF value increases with every appearance of the word in a document, but is gradually decreased with every appearance in other documents.

#### TF-IDF Formula Explained

The simple formula behind the TF-IDF statistical measure. The notations are:

- $N$  is the number of documents we have in our dataset
- $d$  is a given document from our dataset

- $D$  is the collection of all documents
- $w$  is a given word in a document

**Step 1:** calculate the term frequency, the first measure of the score.

$$tf(w,d)=\log(1+f(w,d))$$

- Here  $f(w,d)$  is the frequency of word  $w$  in document  $d$ .

**Step 2:** calculate the inverse term frequency.

$$idf(w,D)=\log(N/f(w,D))$$

- With  $N$  documents in the dataset and  $f(w, D)$  the frequency of word  $w$  in the whole dataset, this number will be lower with more appearances of the word in the whole dataset.

**Final Step:** compute the TF-IDF score by the following formula:

$$tfidf(w,d,D)=tf(w,d)*idf(w,D)$$

- tf-idf is used to classify documents, ranking in search engines.
- tf: term frequency(count of the words present in document from its own vocabulary)
- idf: inverse document frequency(importance of the word to each document).

The method addresses the fact that all words should not be weighted equally, using the weights to indicate the words that are most unique to the document, and best used to characterize it.

### 4.2.2 CONFUSION MATRIX

A Confusion matrix is an  $N \times N$  matrix used for evaluating the performance of a classification model, where  $N$  is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.

- The target variable has two values: **Positive** or **Negative**.
- The **columns** represent the **actual values** of the target variable.
- The **rows** represent the **predicted values** of the target variable

#### True Positive (TP)

- The predicted value matches the actual value
- The actual value was positive and the model predicted a positive value

#### True Negative (TN)

- The predicted value matches the actual value
- The actual value was negative and the model predicted a negative value

#### False Positive (FP) – Type 1 error

- The predicted value was falsely predicted
- The actual value was negative but the model predicted a positive value
- Also known as the **Type 1 error**

#### False Negative (FN) – Type 2 error

- The predicted value was falsely predicted
- The actual value was positive but the model predicted a negative value
- Also known as the **Type 2 error**

### 4.2.3 CLASSIFICATION REPORT

To evaluate the accuracy of the implementation of any classification algorithm an output is given in the form of a confusion matrix, which is a table that is generally used to show the output of a classification model on a set of data for which we already know the true values. The classification report contains the following parameters:

- **Precision:** In a classification model, Precision tells us the proportion of positive identification values that are the same as the true values. It is given by the formula:

$$\text{Precision} = \text{No. of true positives} / (\text{No. of true positives} + \text{No. of false positives})$$

If a model has a precision of 1.0, we know that it has no false positives.

- **Recall:** Recall is the proportion of true positive values that were identified correctly by the classification model. It is given as:

$$\text{Recall} = \text{No. of true positives} / (\text{No. of true positives} + \text{No. of false positives})$$

- **F1-score:** It is also known as F-score or F-measure. It is the weighted average of both precision and recall. It takes into consideration both false positives and false negatives. It is usually used when we have an uneven class distribution. It is given by:

$$F1\text{-score} = 2((\text{precision} * \text{recall}) / (\text{precision} + \text{recall}))$$

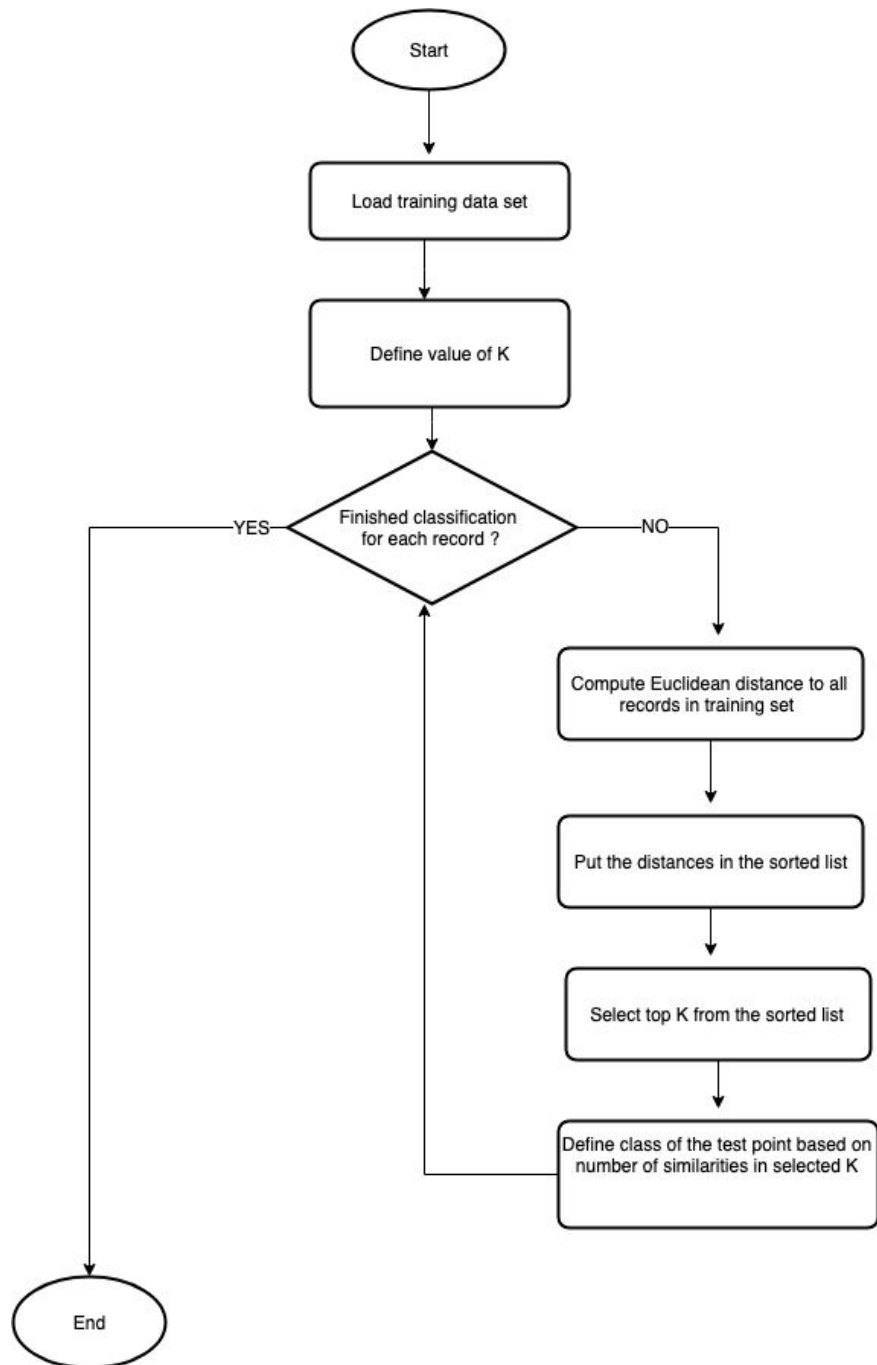
#### **4.2.4 K NEAREST NEIGHBORS ALGORITHM**

The K-nearest neighbors (KNN) algorithm is a supervised machine learning algorithm. The advantage of KNN is that it is quite simple to implement, and yet performs extremely complex classification problems. It is known as a lazy learning algorithm since it doesn't have a specialized training phase. Instead, it uses all of the data available for training. At the same time, it classifies a new data point or instance. KNN is a non-parametric learning algorithm, which means that it doesn't assume anything about the underlying data.

##### **ALGORITHM:**

1. Select the number K of the neighbors
2. Next, calculate the Euclidean distance of K number of neighbors
3. The K nearest neighbors are calculated according to the Euclidean distance.
4. After this, among these k neighbors, count the number of the data points in each category.
5. Next, assign the new data points to that category for which the number of the neighbor is maximum.

## FLOWCHART



*Flowchart of the k-nearest neighbours algorithm*



#### 4.2.5 RANDOM FOREST ALGORITHM

Random Forest is an ensemble classifier(methods that generate many classifiers and aggregate their results) that consists of many decision trees and outputs the class that is the mode of the class's output by individual trees.

The idea behind a tree is to search for a pair of variable-value within the training set and split it in such a way that will generate the “best” two child nodes. The goal is to create branches and leafs based on an optimal splitting criteria, a process called tree growing. Specifically, at every branch or node, a conditional statement classifies the data point based on a fixed threshold in a specific variable, therefore splitting the data.

#### ALGORITHM:

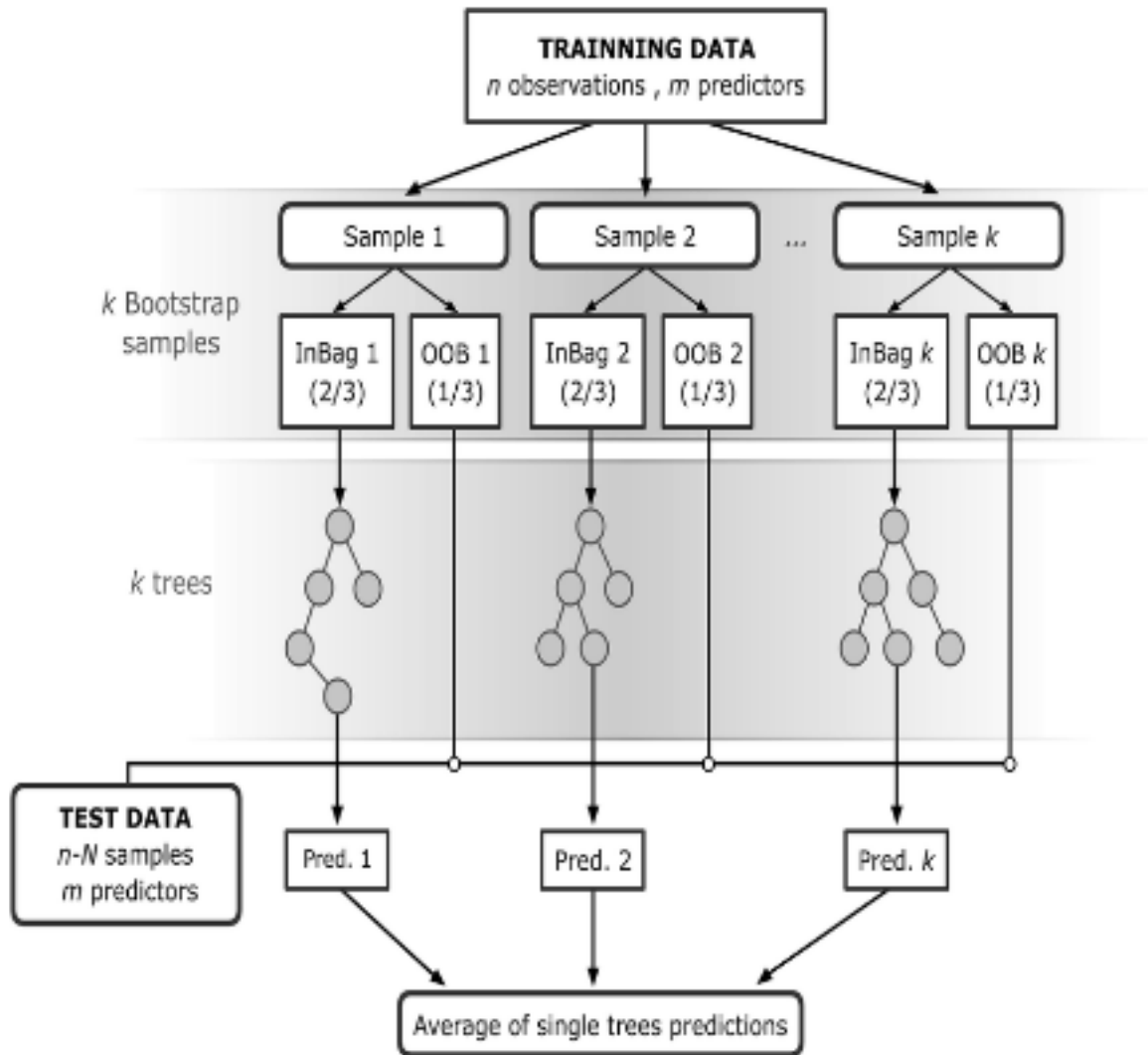
Each tree is constructed using the following algorithm:

Let the number of training cases be  $N$ , and the number of variables in the classifier be  $M$ .

1. The number  $m$  of input variables to be used to determine the decision at a node of the tree;  $m$  should be much less than  $M$ .
2. Choose a training set for this tree by choosing  $n$  times with replacement from all  $N$  available training cases (i.e. take a bootstrap sample).
3. For each node of the tree, randomly choose  $m$  variables on which to base the decision at that node. Calculate the best split based on these  $m$  variables in the training set.
4. Each tree is fully grown and not pruned (as may be done in constructing a normal tree classifier).

For prediction, a new sample is pushed down the tree. It is assigned the label of the training sample in the terminal node it ends up in. This procedure is iterated over all trees in the ensemble, and the average vote of all trees is reported as random forest prediction.

## FLOWCHART



*flowchart of the random forest algorithm*

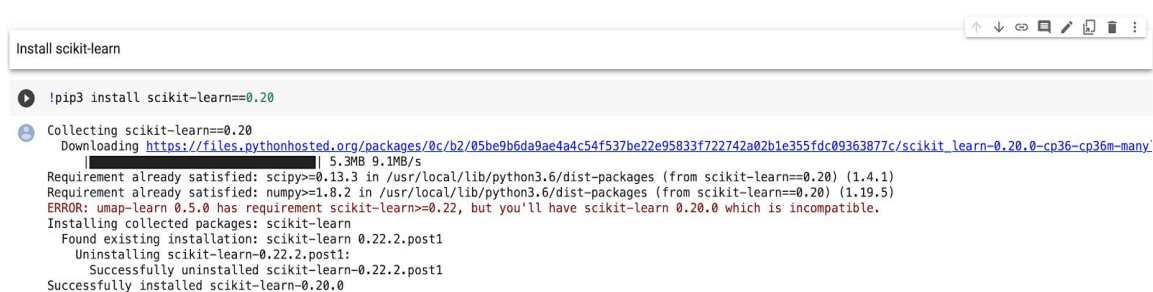
## **CHAPTER - 5**

## 5. IMPLEMENTATION AND CODE SNIPPETS

The project is executed over three stages namely: Comparing ‘k Nearest Neighbors Algorithm’ and ‘Random Forest Algorithm’, Extraction of Target Dataset: Tweets on Ministry of Agriculture, Implementation using target data, to find the desired output of the public sentiment towards the Ministry of Agriculture

### 5.1 STAGE 1: COMPARING ‘K-NEAREST NEIGHBORS ALGORITHM’ AND ‘RANDOM FOREST ALGORITHM’

#### 5.1.1 Installation of scikit-learn



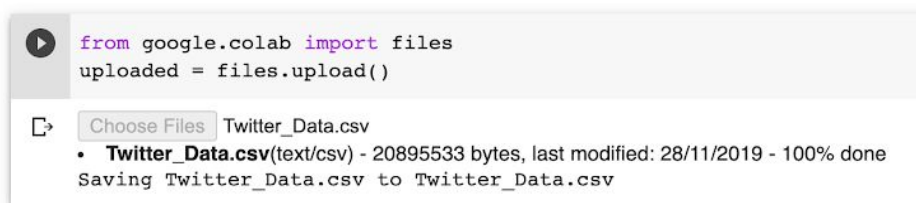
```
Install scikit-learn

!pip3 install scikit-learn==0.20

Collecting scikit-learn==0.20
  Downloading https://files.pythonhosted.org/packages/0c/b2/05be9b6da9ae4a4c54f537be22e95833f722742a02b1e355fdc09363877c/scikit_learn-0.20.0-cp36-cp36m-many...
    5.3MB 9.1MB/s
Requirement already satisfied: scipy>=0.13.3 in /usr/local/lib/python3.6/dist-packages (from scikit-learn==0.20) (1.4.1)
Requirement already satisfied: numpy>=1.8.2 in /usr/local/lib/python3.6/dist-packages (from scikit-learn==0.20) (1.19.5)
ERROR: umap-learn 0.5.0 has requirement scikit-learn>=0.22, but you'll have scikit-learn 0.20.0 which is incompatible.
Installing collected packages: scikit-learn
  Found existing installation: scikit-learn 0.22.2.post1
    Uninstalling scikit-learn-0.22.2.post1:
      Successfully uninstalled scikit-learn-0.22.2.post1
    Successfully installed scikit-learn-0.20.0
```

The first step is to install the Scikit-learn library in Python which contains efficient tools for classification.

#### 5.1.2 Import training data



```
from google.colab import files
uploaded = files.upload()

Choose Files Twitter_Data.csv
• Twitter_Data.csv(text/csv) - 20895533 bytes, last modified: 28/11/2019 - 100% done
Saving Twitter_Data.csv to Twitter_Data.csv
```

The dataset containing tweets is now uploaded. This dataset, named Twitter and Reddit Sentimental analysis Dataset, has been taken from Kaggle.

#### 5.1.3 Convert CSV file to DataFrame

```

import io
import pandas as pd
df1 = pd.read_csv('Twitter_Data.csv')

#clean dataframe of null rows
df1.dropna(
    axis=0,
    how='any',
    thresh=None,
    subset=None,
    inplace=True
)

```

Here, pandas is imported to read the CSV file containing the twitter dataset. This is then converted into the form of DataFrame. All the null rows are removed from this DataFrame.

#### 5.1.4 Splitting features and labels

```

[ ] features = df1.iloc[:, 0].values
    labels = df1.iloc[:, 1].values

```

Now, the features and labels are split.

#### 5.1.5 Constructing processed features

```

[ ] import re
    processed_features = []

    for sentence in range(0, len(features)):
        # Remove all the special characters
        processed_feature = re.sub(r'\W', ' ', str(features[sentence]))
        processed_features.append(processed_feature)

```

Next, all the special characters are removed. The next step is to implement using the TFIDF vectorizer for max\_features as 250, 500 and 750. This is followed by employing kNN and Random Forest classification Algorithms to determine their accuracy. The code for each of these is as follows:

#### 5.1.6 Feature vectorization & classification using TF-IDF max\_features=250

### 5.1.6.1 Using k Nearest Neighbours Algorithm

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

vectorizer = TfidfVectorizer(max_features = 250, min_df=7, max_df=0.8, stop_words = stopwords.words('english'))
processed_features = vectorizer.fit_transform(processed_features).toarray()

X_train, X_test, y_train, y_test = train_test_split(processed_features, labels, test_size=0.2)

classifier = KNeighborsClassifier(n_neighbors=300)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
print(accuracy_score(y_test, y_pred))
```

For a document (tweet), TFIDF Vectorizer is used in order to replace the text with its numerical value i.e with the weight of each word contained, by first considering max\_features as 250 features. The 'min\_df' attribute of the vectorizer specifies the minimum number of times a word must occur in order for the word to be considered as 'useful' and not remove that word. In a similar fashion, 'max\_df' is a value between 0 and 1.0 which specifies the maximum percentage of repetition of that word i.e if the word repeats more than 80% or 0.8 (according to the above code), it shall be considered as something which isn't of much value, hence min\_df and max\_df together specify the range in which the each 'useful' word should fall under, so that the vectorizer considers their weights. The weights for all other words are considered 0. TFIDF vectorizer also removes all the stopwords which do not add much meaning to the entire text.

The output of TFIDF vectorizer i.e the vectorizer object is then transformed into an array and stored in a variable called as the processed\_features.

After converting the vectorizer object into an array, we split this processed\_features array into 4 parts using the train\_test\_split function. These parts which are split are called

X\_train, X\_test, y\_train and y\_test which indicate x(the tweet or the text) and y(the polarity value of the text) data split into training and testing data. This process helps us train the model and find the accuracy of the model. The ratio of splitting of the array is specified by the text\_size attribute which is also a decimal number ranging from 0 to 1.0 which specifies the percentage of training data and testing data from the gathered data. Here, the value 0.2 specifies 20% of the entire data is split so that 20% of it is considered as the Testing Data and 80% of the entire data is considered as the Training Data. With the help of this 80% Training Data, we train the KNN model in order to help predict the values for the testing data. This model considered the k value as 7 indicating 7 nearest neighbours to determine its polarity. Once the model has been trained using X\_train and y\_train, the 20% testing data is passed to the model i.e X\_test is passed to the KNN classifier, which contains only the text but not the polarity values of the document (text). By performing KNN classification technique on the X\_test array, the classifier predicts the values for each document present in X\_test. Once these values have been predicted, these values are stored in a variable called y\_pred. Hence, for X\_test data, we have y\_test, which is the actual polarity values of the data present in the X\_test, and we have y\_pred which are the predictions of the algorithm made for x\_test. These both i.e y\_pred and y\_test are compared in order to find the confusion matrix, the classification\_report and determine the accuracy score of the algorithm, as shown below.

## Output:

```
[[2151 3651 1230]
 [ 925 8873 1217]
 [1332 6394 6821]]
      precision    recall  f1-score   support

    -1.0         0.49      0.31      0.38         7032
     0.0         0.47      0.81      0.59        11015
     1.0         0.74      0.47      0.57        14547

 accuracy                   0.55        32594
 macro avg              0.56      0.53      0.51        32594
 weighted avg           0.59      0.55      0.54        32594

0.5474934036939314
```

*Output for KNN accuracy, max\_features=250*

### 5.1.6.2 Using Random Forest Algorithm

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

vectorizer = TfidfVectorizer(max_features = 250, min_df=7, max_df=0.8, stop_words = stopwords.words('english'))
text_processed_features = vectorizer.fit_transform(text_processed_features).toarray()
X_train, X_test, y_train, y_test = train_test_split(text_processed_features, labels, test_size=0.2, random_state=0)
rfclassifier = RandomForestClassifier(n_estimators=300, random_state=0)
rfclassifier.fit(X_train, y_train)
rfpredictions = text_classifier.predict(X_test)

print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
```

This follows the same process as 5.1.6.1 but implements the RandomForest algorithm instead of the KNN algorithm in order to classify and predict the values for X\_test. Below



is the output of the process.

### Output:

```
[[ 2493  2516  2023]
 [   565  8809  1641]
 [   850  3603 10094]]
      precision    recall  f1-score   support

      -1.0         0.64      0.35      0.46       7032
         0.0         0.59      0.80      0.68      11015
         1.0         0.73      0.69      0.71      14547

 accuracy          0.66      32594
 macro avg          0.65      0.62      0.62      32594
weighted avg          0.66      0.66      0.65      32594

0.6564398355525557
```

*Output for RF accuracy, max\_features=250*

## 5.1.7 Feature vectorization & classification using TFIDF max\_features=500

### 5.1.7.1 Using kNearest Neighbours Algorithm

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

vectorizer = TfidfVectorizer(max_features = 500, min_df=7, max_df=0.8, stop_words = stopwords.words('english'))
processed_features = vectorizer.fit_transform(processed_features).toarray()

X_train, X_test, y_train, y_test = train_test_split(processed_features, labels, test_size=0.2)

classifier = KNeighborsClassifier(n_neighbors=300)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
print(accuracy_score(y_test, y_pred))
```

This follows the same procedure and 5.1.6.1 but implements KNN algorithm with 500 features. Below is the output of this process

### Output:

```
[[ 1721  4669   642]
 [   381 10123   511]
 [   782  8574 5191]]
           precision    recall  f1-score   support

    -1.0         0.60      0.24      0.35         7032
     0.0         0.43      0.92      0.59        11015
     1.0         0.82      0.36      0.50        14547

 accuracy                   0.52         32594
 macro avg              0.62      0.51      0.48         32594
 weighted avg           0.64      0.52      0.50         32594

0.5226422040866417
```

*Output for KNN accuracy, max\_features=500*

### 5.1.7.2 Using Random Forest Algorithm

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

vectorizer = TfidfVectorizer(max_features=500, min_df=7, max_df=0.8, stop_words=stopwords.words('english'))
text_processed_features = vectorizer.fit_transform(text_processed_features).toarray()
X_train, X_test, y_train, y_test = train_test_split(text_processed_features, labels, test_size=0.2, random_state=0)
rfclassifier = RandomForestClassifier(n_estimators=300, random_state=0)
rfclassifier.fit(X_train, y_train)
rfpredictions = text_classifier.predict(X_test)

print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
```

This follows the same procedure and 5.1.6.1 but implements the RandomForest algorithm with 500 features. Below is the output of this process.

## Output:

```
[[ 3296  2349  1387]
 [  254 10178   583]
 [  761  3005 10781]]
      precision    recall  f1-score   support

      -1.0         0.76      0.47      0.58         7032
       0.0         0.66      0.92      0.77        11015
       1.0         0.85      0.74      0.79        14547

 micro avg         0.74      0.74      0.74        32594
 macro avg         0.76      0.71      0.71        32594
weighted avg         0.76      0.74      0.74        32594

0.7441553660182856
```

*Output for RF classifier accuracy, max\_features=500*

## 5.1.8 Feature vectorization & classification using TFIDF max\_features=750

### 5.1.8.1 Using kNearest Neighbours Algorithm

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

vectorizer = TfidfVectorizer(max_features=750, min_df=7, max_df=0.8, stop_words=stopwords.words('english'))
processed_features = vectorizer.fit_transform(processed_features).toarray()

X_train, X_test, y_train, y_test = train_test_split(processed_features, labels, test_size=0.2)

classifier = KNeighborsClassifier(n_neighbors=300)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
print(accuracy_score(y_test, y_pred))
```

This follows the same procedure and 5.1.6.1 but implements KNN algorithm with 750 features. Below is the output of this process.

## Output

```
[[ 3296  2349  1387]
 [  254 10178   583]
 [  761  3005 10781]]
      precision    recall  f1-score   support

      -1.0         0.76      0.47      0.58        7032
         0.0         0.66      0.92      0.77       11015
         1.0         0.85      0.74      0.79       14547

   micro avg       0.74      0.74      0.74      32594
   macro avg       0.76      0.71      0.71      32594
weighted avg       0.76      0.74      0.74      32594

0.7441553660182856
```

*Output for KNN accuracy, max\_features=750*

### 5.1.8.2 Using Random Forest Algorithm

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

vectorizer = TfidfVectorizer(max_features = 750, min_df=7, max_df=0.8, stop_words = stopwords.words('english'))
text_processed_features = vectorizer.fit_transform(text_processed_features).toarray()
X_train, X_test, y_train, y_test = train_test_split(text_processed_features, labels, test_size=0.2, random_state=0)
rfclassifier = RandomForestClassifier(n_estimators=300, random_state=0)
rfclassifier.fit(X_train, y_train)
rfpredictions = text_classifier.predict(X_test)

print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
```

This follows the same procedure and 5.1.6.1 but implements the RandomForest algorithm with 750 features. Below is the output of this process.

**Output:**

```
[[ 3720  2072  1240]
 [   161 10464   390]
 [   724  2711 11112]]
      precision    recall  f1-score   support

-1.0         0.81         0.53         0.64         7032
 0.0         0.69         0.95         0.80        11015
 1.0         0.87         0.76         0.81        14547

 accuracy                   0.78         32594
 macro avg           0.79         0.75         0.75         32594
weighted avg           0.80         0.78         0.77         32594

0.7760937595876541
```

*Output for RandomForest accuracy, max\_features= 750*

## 5.2 TABULATING KNN & RANDOM FOREST OBSERVATIONS

S.No	Method	Number of Features	Polarity	Precision	Recall	F1-Score	Accuracy (%)
1	KNN	250	-1 0 1	0.49 0.47 0.74	0.31 0.81 0.47	0.38 0.59 0.57	54.75
2	KNN	500	-1 0 1	0.6 0.43 0.82	0.24 0.92 0.36	0.35 0.59 0.5	52.26
3	KNN	750	-1 0 1	0.65 0.41 0.84	0.22 0.94 0.29	0.32 0.57 0.43	49.13
4	RF	250	-1 0 1	0.64 0.059 0.73	0.35 0.80 0.69	0.46 0.68 0.71	65.64
5	RF	500	-1 0 1	0.76 0.66 0.85	0.47 0.92 0.74	0.58 0.77 0.79	74.41
6	RF	750	-1 0 1	0.81 0.69 0.87	0.53 0.95 0.76	0.64 0.80 0.81	77.61
7	RF	3038	-1 0 1	0.85 0.85 0.90	0.71 0.97 0.87	0.78 0.90 0.88	86.86





### 5.4.2 Setting up access to the API & Creating an API Object

```
def connect_to_twitter_OAuth():
    auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
    auth.set_access_token(ACCESS_TOKEN, ACCESS_SECRET)

    api = tweepy.API(auth, wait_on_rate_limit=True)
    return api

api = connect_to_twitter_OAuth()
```

After the retrieval of all the access and consumer keys, we move forward to authenticate these keys. This is done by the OAuth() method by using the OAuthHandler method of the tweepy API. This authenticates the user keys and the consumer keys given in order for us to access the tweets using this API.

### 5.4.3 Function for the creation of initial dataset

```
def create_dataset(search_words):
    tweets = tweepy.Cursor(api.search,
                           q= search_words,
                           lang="en", geo="21.1458°N, 79.0882°E, 1607km", tweet_mode='extended').items(3000)

    tweet_list=[]
    df=0
    for tweet in tweets:
        text = tweet.full_text # utf-8 text of tweet
        created_at = tweet.created_at # utc time tweet created
        source = tweet.source # utility used to post tweet
        retweets = tweet.retweet_count # number of times this tweet retweeted
        tweet_list.append({'text':text, 'created_at':created_at, 'source':source, 'retweets':retweets})
    df = pd.DataFrame(tweet_list, columns=['text', 'created_at', 'source', 'retweets'])

    return df
```

After the authentication of the user and consumer keys of the application along with creating an api reference in our project, we move forward to the retrieval of the tweets which follow certain protocols and satisfy the rules of search. We define these protocols



with the use of Cursor objects and the API Object references. By using the Cursor object, we get an Iterable object which references the location of the number of tweets which have been gathered, by using the items() method of the Cursor class which specifies the number of tweets to be searched. We specify this with the help of search\_words, the radius in which we need to collect the tweets from, the language of the tweets, and the tweet\_mode. The radius of the search is determined by the 'geo' attribute of the Cursor object which takes 2 parameters, which are the latitude and longitude of the center of the search radius as the first parameter and the radius of the search as the second parameter. As Twitter has increased the number of characters in a Tweet from 140 characters to 280 characters, hence we use the tweet\_mode in order to extract the full 240 characters and we use .full\_text in order to insert this tweet into a list. This user defined method which takes a search\_words parameter which specifies the keywords to be searched for, returns a DataFrame which contains the required fields such as the text of the tweet, the time of creation, the source of the tweet along with the retweet count.

#### 5.4.4 Creating a Dataset for the agriculture ministry as CSV

```
AgriGoI=create_dataset("#FarmBills2020")
agro_feat2=create_dataset("agriculture")
agro_feat3=create_dataset("@nstomar")
agro_feat4=create_dataset("farmers")

AgriGoI=AgriGoI.append(agro_feat2)
AgriGoI=AgriGoI.append(agro_feat3)
AgriGoI=AgriGoI.append(agro_feat4)

AgriGoI.to_csv("agriculture.csv")
```

In the above code snippet, an individual DataFrame is created for each search\_word which specifies the kind of tweets to be gathered. For tweets which can be used as a part of the Agricultural Ministry's dataset, we require the tweets which talk about the various

aspects of agriculture in India. To specifically gather these tweets, we use the `search_keywords` to look for these keywords in the tweet and if present, to fetch that tweet. Here, we can see that the use of various keywords such as `#Farmbills2020`, which is one of the top trending hashtags on Twitter, `@nstomar` which is the official Twitter handle of the Agricultural Minister, Shri Narendra Tomar, and keywords such as 'agriculture' and 'farmers' have been used to gather the dataset required for the Agricultural dataset. All these individual DataFrames are then appended into 1 final dataset which will be converted to a csv file with the use of `.to_csv()` function of the Pandas DataFrame.

#### 5.4.5 Converting CSV to dataframe

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
cols = ['no.', 'text', 'date', 'src', 'rt']
dfag = pd.read_csv("agriculture.csv", header=None, names=cols)
dfag=dfag.drop(0)
dfag.head()
```

After saving the DataFrames as CSV, we need to later turn these CSV files back to DataFrame in order for them to be cleaned. Hence, we use the `.read_csv(file_name, header, names)` as the function to convert the csv file back into a DataFrame, in order to perform data cleaning on this dataset.

### 5.4.6 Cleaning of dfag dataframe

```
#removing unnecessary columns
dfag.drop(['no.', 'date', 'src', 'rt'], axis=1, inplace=True)

import re
for i in range(0, len(dfag)):
    # Remove all the special characters
    dfag.iloc[i,0] = re.sub(r'\W', ' ', dfag.iloc[i,0])

    # remove all single characters
    dfag.iloc[i,0] = re.sub(r'\s+[a-zA-Z]\s+', ' ', dfag.iloc[i,0])

    # Remove single characters from the start
    dfag.iloc[i,0] = re.sub(r'\^[a-zA-Z]\s+', ' ', dfag.iloc[i,0])

    # Substituting multiple spaces with single space
    dfag.iloc[i,0] = re.sub(r'\s+', ' ', dfag.iloc[i,0], flags=re.I)

    # Removing prefixed 'b'
    dfag.iloc[i,0] = re.sub(r'^b\s+', '', dfag.iloc[i,0])

    # Converting to Lowercase
    dfag.iloc[i,0] = dfag.iloc[i,0].lower()

print(dfag.head())
print(len(dfag))
```

```
text
1  rt ritz444 rahul the fake gandhi says that far...
2  rt ritz444 rahul the fake gandhi says that far...
3  rt ani watch delhi group of people claiming to...
4  rt ritz444 rahul the fake gandhi says that far...
5  rt ritz444 rahul the fake gandhi says that far...
12000
```

After the dataset is converted back into a DataFrame, we clean the data by removing all the unwanted columns, and using re which is the module for Regular Expressions, we remove all the special characters, single characters, single characters from the starting of the tweet, removing double spaces and replacing them with single spaces, removing the prefixed 'b' character and finally turning all the tweets in lowercase. This process will result in a dataset which is free of unwanted data and is in a single case so that the chances of triggering an error is reduced as much as possible./

#### 5.4.7 Convert dataframe to csv which is exported to trained model

```
dfag.to_csv("CleanAgri.csv")
files.download("CleanAgri.csv")
```

After the dataset has been cleaned i.e all the unnecessary information from the dataset has been removed, we can move forward by saving this cleaned dataset as a new csv file and downloading and saving that file locally on the machine.

### 5.5 STAGE 3: IMPLEMENTATION USING TARGET DATASET

#### 5.5.1 Feature Vectorization



```
agrifeatures = dfagro.iloc[:, 1].values

import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer (min_df=7, max_df=0.8, stop_words = stopwords.words('english'))
agrifeatures = vectorizer.fit_transform(agrifeatures).toarray()

[ ] len(agrifeatures[0])

3038
```

The TFIDF Vectorizer is used in order to find the most relevant words in a document(tweet). The ‘min\_df’ attribute of the vectorizer specifies the minimum number of times a word must occur in order for the word to be considered as ‘useful’ and not remove that word. This min\_df is given a value of 7. In a similar fashion, ‘max\_df’ is a value between 0 and 1.0 which specifies the maximum percentage of repetition of that word i.e if the word repeats more than 80% or 0.8 (according to the above code), it shall be considered as something which isn’t of much value. Hence min\_df and max\_df together specify the range in which each ‘useful’ word should fall under, so that the vectorizer considers their weights. The weights for all other words are considered 0. TFIDF vectorizer also removes all the stopwords which do not add much meaning to the entire text.

The output of TFIDF vectorizer i.e the vectorizer object is then transformed into an array and stored in a variable called as the processed\_features.

The vectorizer object is then transformed and converted into the form of an array which is stored in a variable called agrifeatures. The length of agrifeatures is 3038.

### 5.5.2 Installing scikit-learn

Install scikit-learn

```
!pip3 install scikit-learn==0.20

Collecting scikit-learn==0.20
  Downloading https://files.pythonhosted.org/packages/0c/b2/05be9b6da9ae4a4c54f537be22e95833f722742a02b1e355fdc09363877c/scikit_learn-0.20.0-cp3.6-cp36m-macosx_10_14_x86_64.whl (5.3MB)
    Requirement already satisfied: numpy>=1.8.2 in /usr/local/lib/python3.6/dist-packages (from scikit-learn==0.20) (1.19.5)
    Requirement already satisfied: scipy>=0.13.3 in /usr/local/lib/python3.6/dist-packages (from scikit-learn==0.20) (1.4.1)
    ERROR: umap-learn 0.5.0 has requirement scikit-learn>=0.22, but you'll have scikit-learn 0.20.0 which is incompatible.
Installing collected packages: scikit-learn
  Found existing installation: scikit-learn 0.22.2.post1
    Uninstalling scikit-learn-0.22.2.post1:
      Successfully uninstalled scikit-learn-0.22.2.post1
    Successfully installed scikit-learn-0.20.0
```

The first step is to install the Scikit-learn library in Python which contains a lot of efficient tools for classification.

### 5.5.3 Uploading Twitter dataset

Upload csv

```
[ ] from google.colab import files
    uploaded = files.upload()
```

[Choose Files](#) no files selected

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving Twitter\_Data.csv to Twitter\_Data.csv

The dataset containing tweets is now uploaded. This dataset, named Twitter and Reddit Sentimental analysis Dataset, has been taken from Kaggle.

## 5.5.4 Converting CSV file to DataFrame

CSV to DataFrame

```
import io
import pandas as pd
rddf = pd.read_csv('Twitter_Data.csv')

#clean dataframe of null rows
rddf.dropna(
    axis=0,
    how='any',
    thresh=None,
    subset=None,
    inplace=True
)
```

Here, pandas is imported to read the CSV file containing the twitter dataset. This is then converted into the form of DataFrame. All the null rows are removed from this DataFrame.

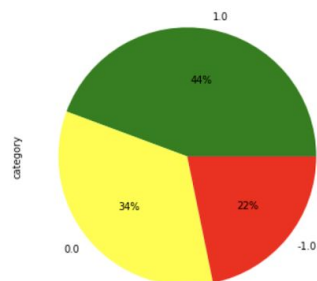
## 5.5.5 Data Visualization

```
#establishing plot dimensions
import matplotlib as plt
plot_size = plt.rcParams["figure.figsize"]
print(plot_size[0])
print(plot_size[1])

plot_size[0] = 8
plot_size[1] = 6
plt.rcParams["figure.figsize"] = plot_size

#piechart of sentiment
rddf.category.value_counts().plot(kind='pie', autopct='%1.0f%%', colors=["green", "yellow", "red"])
```

```
6.0
4.0
<matplotlib.axes._subplots.AxesSubplot at 0x7fcd55941940>
```



The matplotlib library is now imported to plot a pie chart. This graph depicts the initial classification present in the dataset obtained from Kaggle.

### 5.5.6 Splitting features and labels

```
[ ] text_features = rfd.f.iloc[:, 0].values
    labels = rfd.f.iloc[:, 1].values
```

Now, the features and labels are split.

### 5.5.7 Constructing processed features

```
[ ] import re
    text_processed_features = []

    for sentence in range(0, len(text_features)):
        # Remove all the special characters
        text_processed_feature = re.sub(r'\W', ' ', str(text_features[sentence]))
        text_processed_features.append(text_processed_feature)
```

Next, all the special characters are removed.

### 5.5.8 Training using Random Forest using 3038 features

```
[ ] from sklearn.model_selection import train_test_split

    X_train, X_test, y_train, y_test = train_test_split(text_processed_features, labels, test_size=0.2, random_state=0)
```

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier

vectorizer = TfidfVectorizer(max_features=3038, min_df=7, max_df=0.8, stop_words=stopwords.words('english'))
text_processed_features = vectorizer.fit_transform(text_processed_features).toarray()
X_train, X_test, y_train, y_test = train_test_split(text_processed_features, labels, test_size=0.2, random_state=0)
rfclassifier = RandomForestClassifier(n_estimators=300, random_state=0)
rfclassifier.fit(X_train, y_train)
rfpredictions = text_classifier.predict(X_test)

print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
```

```
[ ] predictions = text_classifier.predict(X_test)
```

For a document (tweet), TFIDF Vectorizer is used in order to replace the text with its numerical value i.e with the weight of each word contained, by considering max\_features

as 3808 features. The 'min\_df' attribute of the vectorizer specifies the minimum number of times a word must occur in order for the word to be considered as 'useful' and not remove that word. In a similar fashion, 'max\_df' is a value between 0 and 1.0 which specifies the maximum percentage of repetition of that word i.e if the word repeats more than 80% or 0.8 (according to the above code), it shall be considered as something which isn't of much value, hence min\_df and max\_df together specify the range in which the each 'useful' word should fall under, so that the vectorizer considers their weights. The weights for all other words are considered 0. TFIDF vectorizer also removes all the stopwords which do not add much meaning to the entire text.

The output of TFIDF vectorizer i.e the vectorizer object is then transformed into an array and is stored in a variable called the processed\_features.

After converting the vectorizer object into an array, we split this processed\_features array into 4 parts using the train\_test\_split function. These parts which are split are called x\_train, x\_test, y\_train and y\_test which are nothing but x(the tweet or the text) and y(the polarity value of the text) data split into training and testing data. This process helps us further train the model and figure out the accuracy of the model. The ratio of splitting of the array is specified by the test\_size attribute which is also a decimal number ranging from 0 to 1.0 which specifies the percentage of training data and testing data from the gathered data. Here, the value 0.2 specifies 20% of the entire data is split so that 20% of it is considered as the Testing Data and 80% of the entire data is considered as the Training Data. Since this model has already been trained, the testing data is passed to the model i.e x\_test is passed to the RF classifier, which contains only the text but not the polarity values of the document (text). By performing RF classification technique on the x\_test array, the classifier predicts the values for each document present in x\_test. Once these values have been predicted, these values are stored in a variable called y\_pred. Hence, for x\_test data, we have y\_test, which is the actual polarity values of the data present in the x\_test and we have y\_pred which are the predictions of the algorithm made for x\_test.

These both i.e rfpredictions and y\_test are compared in order to find the confusion



matrix, the classification\_report and determine the accuracy score of the algorithm, as shown below.

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
```

```
[[ 5021   777  1234]
 [  102 10683   230]
 [   766  1173 12608]]
```

	precision	recall	f1-score	support
-1.0	0.85	0.71	0.78	7032
0.0	0.85	0.97	0.90	11015
1.0	0.90	0.87	0.88	14547
accuracy			0.87	32594
macro avg	0.86	0.85	0.85	32594
weighted avg	0.87	0.87	0.87	32594

```
0.8686261275081303
```

*The accuracy rate of this prediction is 86.86%.*

### 5.5.9 Implementing using Agriculture dataset

After deciding on the Random Forest algorithm to analyse the public's sentiment with regards to the Ministry of Agriculture of India, we now upload the dataset containing tweets regarding agriculture. These tweets are cleaned and processed before being converted to a DataFrame format. Then, we test it by using the trained algorithm. The output of this process is the analysis of the public reaction to the Ministry of Agriculture. This we depict in the form of a pie chart.

### 5.5.10 Importing Agriculture dataset

```
[ ] from google.colab import files
    files.download('AgriResultSentiment.csv')
```

### 5.5.11 Converting CSV file to DataFrame

```
import io
import pandas as pd
dfagro = pd.read_csv('CleanAgri.csv')

#clean dataframe of null rows
dfagro.dropna(
    axis=0,
    how='any',
    thresh=None,
    subset=None,
    inplace=True
)
```

The CSV file containing the Agriculture dataset is converted to the form of DataFrame.

### 5.5.12 Running the trained algorithm

```
#dataframe output
agrotext=dfagro.iloc[:, 1].values
TweetDf=pd.DataFrame( agrotext, columns=['text'])
PredictionDf=pd.DataFrame( AgriPred, columns=['predicted_sentiment'])
result = pd.concat([TweetDf, PredictionDf], axis=1)
result
```

	text	predicted_sentiment
0	rt rit444 rahul the fake gandhi says that far...	1.0
1	rt rit444 rahul the fake gandhi says that far...	1.0
2	rt ani watch delhi group of people claiming to...	1.0
3	rt rit444 rahul the fake gandhi says that far...	1.0
4	rt rit444 rahul the fake gandhi says that far...	1.0
...	...	...
11995	mrsgandhi this could be deliberate attempt to...	0.0
11996	rt thedeshbhakt farmers say peacefulprotestcon...	0.0
11997	rt msinghbjp he is misguiding innocent farmers...	0.0
11998	rt deepakkhatri812 police is arresting the far...	1.0
11999	argentina is looking for ways to ensure ample...	1.0

12000 rows x 2 columns

```
[ ] result.to_csv("AgriResultSentiment.csv")
```

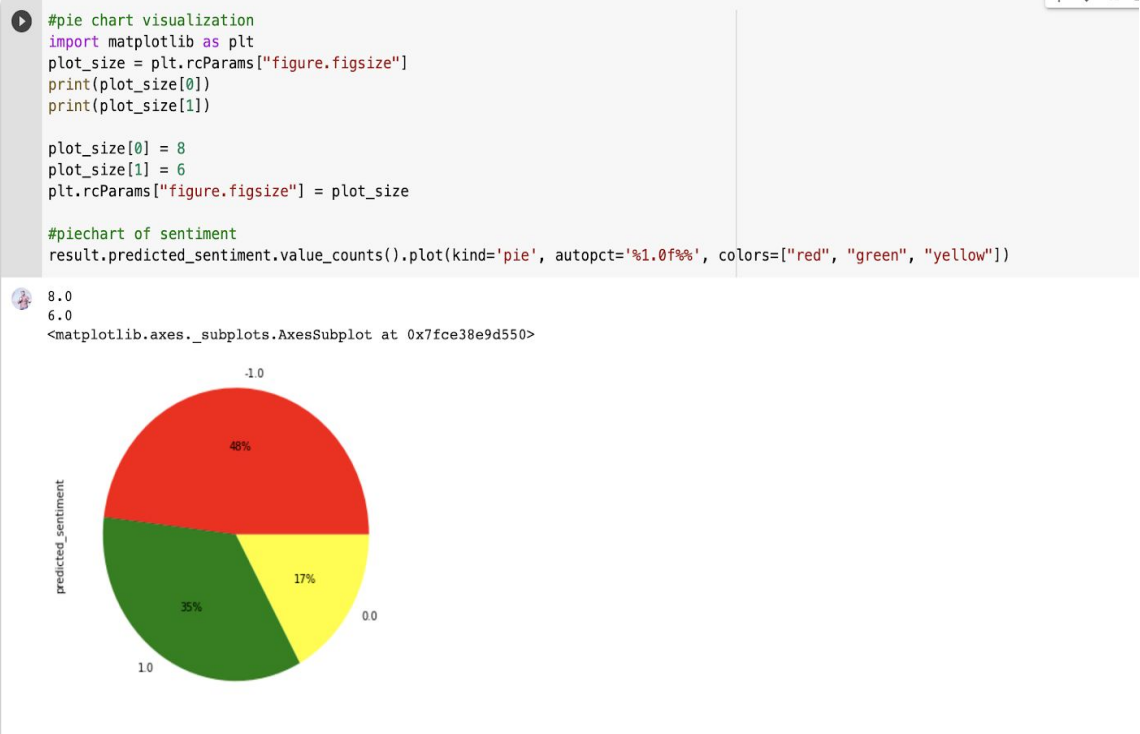
```
[ ] from google.colab import files
files.download('AgriResultSentiment.csv')
```

Run Trained ML algo

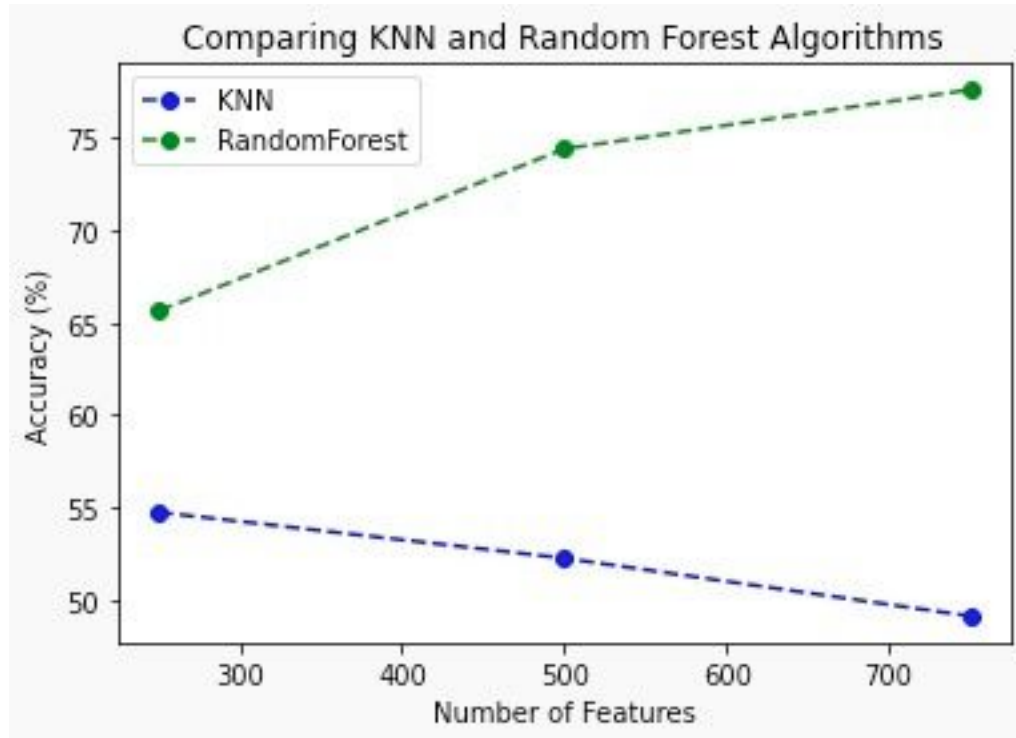
```
[ ] AgriPred = text_classifier.predict(agrifeatures)
```

The trained algorithm is now run using the Agriculture dataset. The predictions are stored in the object AgriPred.

### 5.5.13 Data visualization for Agriculture output

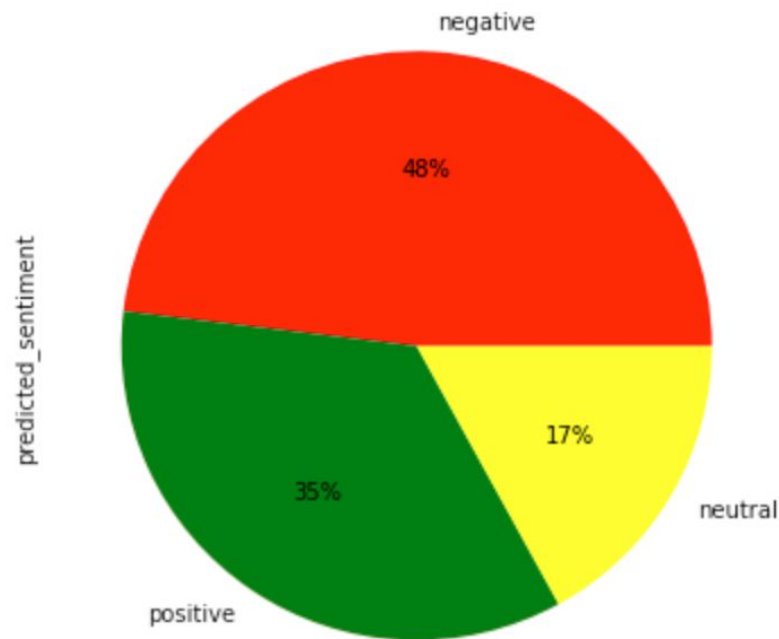


## 5.6 OBSERVATIONS



*Line graph Comparing kNN and RandomForest Algorithms*

In the above figure, we can see that the accuracy of the Random Forest Algorithm is much higher when considering 250, 500 and 750 features. The accuracy of the KNN algorithm has not proved to be upto the mark under these conditions hence Random Forest Algorithm has been used to classify the Target Data.



*Pie Chart depicting public sentiment towards agricultural ministry*

After running the cleaned agriculture dataset through the Random Forest Classifier algorithm, we get the values with which we can plot a pie-chart as depicted above. This pie-chart represents the percentage of tweets posted by the public regarding various schemes and implementations of the Agricultural ministry. The result of this project, as depicted by this pie-chart, shows that 48% of the tweets posted on Twitter are having a negative feeling towards the agricultural ministry while 35% of the tweets have a positive feeling towards the same. Only 17% of the tweets are neutral. This could be due to the backlash against the Farm Bills passed in September, 2020.

## CHAPTER - 6

## **6. SOFTWARE TESTING**

### **6.1 TESTING**

#### **6.1.1 Unit Testing**

In unit testing the different modules of software were tested independently to locate errors. This helps in locating errors, in coding and logic that were contained within a particular module.

#### **6.1.2 Integrated Testing**

After unit testing, combined testing of all modules was carried out. The purpose was to determine that all the modules are correctly integrating with each other.

#### **6.1.3 System Testing**

Final testing was done on the entire system to check whether the desired specification and requirements are met or not. The main aim here was to determine the inconsistencies in the developed system. Hence the whole system was tested. Put the new system into operation at this point, a final dress rehearsal sometimes runs.

#### **6.1.4 Final Testing**

The initial Dataset gathered contains the values for the text contained in the dataset. This dataset is split into Training Data and Testing Data. Both of these datasets contain the tweet text and its respective sentiment. The Training data is initially used to train the model.

This Testing Data is passed through the trained model using Random Forest Algorithm and KNN Algorithm. These algorithms, which have been trained by the Training Data first predicts the values for the Testing Data, and these predictions of the values for the

Testing Data are compared to the actual values of the Testing data, to produce the accuracy that the algorithm has with respect to the data set.

The dataset gathered by the use of various search words and using the Twitter API will be considered as the Target Data. Testing Data is then passed to this model in order to predict the values for the passed dataset. The result of this shall have the same accuracy as depicted when Training Data and Testing Data are used.

By using these datasets, the predictions are made and these predictions are converted into a pie-chart in order to visualise the mood of the public, i.e we can classify if the mood is positive, negative or neutral.



## 6.2 Test Cases

### 6.2.1 Test Case 1: KNN using 250 features

```
[ ] from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test, predictions))
```

```
[[2151 3651 1230]
 [ 925 8873 1217]
 [1332 6394 6821]]

              precision    recall  f1-score   support

    -1.0         0.49         0.31         0.38         7032
     0.0         0.47         0.81         0.59        11015
     1.0         0.74         0.47         0.57        14547

 accuracy                   0.55        32594
 macro avg              0.56         0.53         0.51        32594
 weighted avg           0.59         0.55         0.54        32594

0.5474934036939314
```

- Accuracy for KNN,max\_features=250 is 54.75%

### 6.2.2 Test Case 2: KNN using 500 features

```
[ ] from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test, predictions))
```

	precision	recall	f1-score	support
-1.0	0.60	0.24	0.35	7032
0.0	0.43	0.92	0.59	11015
1.0	0.82	0.36	0.50	14547
accuracy			0.52	32594
macro avg	0.62	0.51	0.48	32594
weighted avg	0.64	0.52	0.50	32594

0.5226422040866417

- Accuracy for KNN,max\_features=500 is 52.26%

### 6.2.3 Test Case 3: KNN using 750 features

```
[ ] from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test, predictions))
```

```

[[ 1513  5143   376]
 [  279 10348   388]
 [  551  9842 4154]]
      precision    recall  f1-score   support

-1.0         0.65      0.22      0.32       7032
 0.0         0.41      0.94      0.57      11015
 1.0         0.84      0.29      0.43      14547

 accuracy                   0.49      32594
 macro avg              0.63      0.48      0.44      32594
weighted avg              0.65      0.49      0.45      32594

0.4913481008774621

```

- Accuracy for KNN,max\_features=750 is 49.13%

#### 6.2.4 Test Case 4: Random Forest using 250 features

```

[ ] from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test, predictions))

```

```

      precision    recall  f1-score   support

-1.0         0.64      0.35      0.46       7032
 0.0         0.59      0.80      0.68      11015
 1.0         0.73      0.69      0.71      14547

0.6564398355525557

```

- Accuracy for Random Forest,max\_features=250 is 65.64%

### 6.2.5 Test Case 5: Random Forest using 500 features

```
[ ] from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test, predictions))
```

	precision	recall	f1-score	support
-1.0	0.76	0.47	0.58	7032
0.0	0.66	0.92	0.77	11015
1.0	0.85	0.74	0.79	14547
micro avg	0.74	0.74	0.74	32594
macro avg	0.76	0.71	0.71	32594
weighted avg	0.76	0.74	0.74	32594

0.7441553660182856

- Accuracy for Random Forest, max\_features=500 is 74.41%

### 6.2.6 Test Case 6: Random Forest using 750 features

```
[ ] from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test, predictions))
```

```

[[ 3720  2072  1240]
 [  161 10464   390]
 [  724  2711 11112]]
              precision    recall  f1-score   support

    -1.0         0.81         0.53         0.64         7032
     0.0         0.69         0.95         0.80        11015
     1.0         0.87         0.76         0.81        14547

 accuracy                   0.78        32594
 macro avg              0.79         0.75         0.75        32594
 weighted avg           0.80         0.78         0.77        32594

0.7760937595876541

```

- Accuracy for Random Forest ,max\_features=750 is 77.61%

### 6.2.7 Test Case 7: Random Forest using 3038 features

```

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test, predictions))

[[ 5021   777  1234]
 [  102 10683   230]
 [  766  1173 12608]]
              precision    recall  f1-score   support

    -1.0         0.85         0.71         0.78         7032
     0.0         0.85         0.97         0.90        11015
     1.0         0.90         0.87         0.88        14547

 accuracy                   0.87        32594
 macro avg              0.86         0.85         0.85        32594
 weighted avg           0.87         0.87         0.87        32594

0.8686261275081303

```

- Accuracy for Random Forest ,max\_features=3038 is 86.86%

## 6.2.8 Test Case 8: Random Forest on Agriculture Dataset



Sentiment	Percentage
Percentage of positive tweets	35%
Percentage of negative tweets	48%
Percentage of neutral tweets	17%

From the above table, we can conclude that the sentiment of the tweets of Indian Agriculture Ministry is mostly negative.

## **BENEFITS**

Ministries of the government can use this analysis in order to find out the acceptance rate or how the common people feel about various Ministries. For the sake of this project, the Agricultural Ministry of India has been implemented as a Use Case.

- This data can be used to depict which ministry has what kind of an opinion in the public. It helps the government focus on changing the feeling of the people posting the considered tweets.
- This gives us a visual representation of how well the chosen Ministry( in this case, the Agricultural Ministry of India) has been performing by analysing the outlook of the people with respect to that performance.

## **FUTURE ENHANCEMENTS**

- This can be extended to all the ministries under the Government, and can result in enabling more attention towards the ministries that are lacking according to the public's sentiment.
- This can be extended to not only to the central Ministries but also to various Political Parties in order to gather Data regarding that particular party and how the people feel about that party.
- Increase in the target dataset will increase the accuracy of the algorithm.
- This can be implemented using other Machine Learning algorithms with more feature extraction to check for a better accuracy score.



## CONCLUSION

We used two algorithms namely Random Forest and k Nearest Neighbors to determine which algorithm amongst the two is better able to find the sentiment of a text (tweets) by gradually increasing the number of features to be considered. We used the TFIDF vectorization method to convert textual data of tweets into numerical data. We observe that the Random Forest algorithm offers more accuracy when compared with the kNN algorithm. After performing Twitter Sentiment Analysis, we extract the tweets pertaining to our use case namely Ministry of Agriculture, Government of India using the Tweepy module. We find that the extracted tweets upon vectorization contain a total of 3038 features. We accordingly train the RandomForest Classifier and conduct the sentimental analysis for the Ministry of Agriculture Tweets. We hence find that the public's sentiment towards the Indian Ministry of Agriculture is mostly negative possibly because of the backlash to the newly introduced agricultural reforms in the country.

## REFERENCES

- [1] Mohd Zeeshan Ansari, M. B. Aziz, M. O. Siddiqui, H. Mehra and K. P. Singh. Analysis of Political Sentiment Orientations on Twitter.
- [2] Josemar A. Caetano, Hélder S. Lima, Mateus F. Santos & Humberto T. Marques-Neto in the Journal of Internet Services and Applications. Using sentiment analysis to define twitter political users' classes and their homophily during the 2016 American presidential election.
- [3] Sandip Roy. Analyzing Political Sentiment using Twitter Data.
- [4] Mohamed Lemine M'Bareck. Political Speech on Twitter: A Sentiment Analysis of Tweets and News Coverage of Local Gun Policy.
- [5] Hsuanwei Michelle Chen, Patricia C. Franks. Exploring Government Uses of Social media through Twitter Sentiment Analysis

## BIBLIOGRAPHY

1. <https://monkeylearn.com/blog/sentiment-analysis-of-twitter/>
2. <https://towardsdatascience.com/twitter-sentiment-analysis-classification-using-nltk-python-fa912578614c>
3. [Machine Learning with Python Tutorial - Tutorialspointwww.tutorialspoint.com › machine\\_learning\\_with\\_pyth...](http://www.tutorialspoint.com/machine_learning_with_python/)
4. [https://www.kaggle.com/cosmos98/twitter-and-reddit-sentimental-analysis-dataset?select=Twitter\\_Data.csv](https://www.kaggle.com/cosmos98/twitter-and-reddit-sentimental-analysis-dataset?select=Twitter_Data.csv)
5. [Twitter Sentiment Analysis using Python - GeeksforGeekswww.geeksforgeeks.org › twitter-sentiment-analysis-usi...](http://www.geeksforgeeks.org/twitter-sentiment-analysis-using-python/)
6. [Tutorials — pandas 0.15.2 documentationpandas.pydata.org › pandas-docs › version › tutorials](http://pandas.pydata.org/pandas-docs/version/0.15.2/tutorials/)
7. [Twitter Developer: Use Cases, Tutorials, & Documentationdeveloper.twitter.com › ...](https://developer.twitter.com/en/docs/)
8. [Developer Blog - Twitter Blogblog.twitter.com › developer › en\\_us](https://blog.twitter.com/developer/en_us/)