# Chapter 1

## PREAMBLE

### Introduction

According to World Health Organization (WHO), more than 300 million people worldwide are suffering from depression, which equals about 4.4% of the global population. While forms of depression are more common among females (5.1%) than males (3.6%) and prevalence differs between regions of the world, it occurs in any age group and is not limited to any specific life situation. Depression is therefore often described to be accompanied by paradoxes, caused by a contrast between the self-image of a depressed person and the actual facts. Latest results from the 2016 National Survey on Drug Use and Health in the United States report that, during the year 2016, 12.8% of adolescents between 12 and 17 years old and 6.7% of adults had suffered a major depressive episode (MDE). “depressed” has become frequently used in everyday language. In general, depression can be described to lead to an altered mood and may also be accompanied, for example, by a negative self-image, wishes to escape or hide, vegetative changes, and a lowered overall activity. The symptoms experienced by depressed individuals can severely impact their ability to cope with any situation in daily life and therefore differ drastically from normal mood variations that anyone experiences.

At its worst, depression can lead to suicide. Close to 800 000 people die due to suicide every year. Suicide is the second leading cause of death in 15-29-year-olds. In Europe, self-harm was even reported as the most common cause of death in the age group between 15 and 29 and the second most common between 30 and 49, again in results obtained by WHO in 2019.

While depression and other mental illnesses may lead to social withdrawal and isolation, it was found that social media platforms are indeed increasingly used by affected Based on these findings, peer to peer communities on social media can be able to challenge stigma, increase the likelihood to seek professional help, and directly offer help online to people with mental illness. A similar study in the USA came to the conclusion that internet users with stigmatized illnesses like depression or urinary incontinence are more likely to use online resources for health-related information and for communication about their illness than people with another chronic illness. All this emphasizes the importance of research toward ways to assist depressed individuals on social media platforms and on the internet in general.

This Paper will try to predict early signs of depression through social media text mining. For build the logic for that we are using Neural networks. Data set extracting from Social Media to feed into neural Networks.

## 1.2 Key facts

* Depression is a common mental disorder. Globally, more than 264 million people of all ages suffer from depression.
* Depression is a leading cause of disability worldwide and is a major contributor to the overall global burden of disease.
* More women are affected by depression than men.
* Depression can lead to suicide.
* There are effective psychological and pharmacological treatments for moderate and severe depression.

## 1.3 Types and symptoms

Depending on the number and severity of symptoms, a depressive episode can be categorized as mild, moderate or severe.

A key distinction is also made between depression in people who have or do not have a history of manic episodes. Both types of depression can be chronic (i.e. over an extended period) with relapses, especially if they go untreated.

# ****1.3.1 Recurrent depressive disorder****

This disorder involves repeated depressive episodes. During these episodes, the person experiences depressed mood, loss of interest and enjoyment, and reduced energy leading to diminished activity for at least two weeks. Many people with depression also suffer from anxiety symptoms, disturbed sleep and appetite, and may have feelings of guilt or low self-worth, poor concentration and even symptoms that cannot be explained by a medical diagnosis.

Depending on the number and severity of symptoms, a depressive episode can be categorized as mild, moderate or severe. An individual with a mild depressive episode will have some difficulty in continuing with ordinary work and social activities but will probably not cease to function completely. During a severe depressive episode, it is unlikely that the sufferer will be able to continue with social, work or domestic activities, except to a limited extent.

### ****1.3.2 Bipolar affective disorder****

This type of depression typically consists of both manic and depressive episodes separated by periods of normal mood. Manic episodes involve elevated or irritable mood, over-activity, pressure of speech, inflated self-esteem and a decreased need for sleep.

### Proposed System

What the result could mean? Positive, this mean that person is unlikely to have depression or anxiety. Neutral, this is the middle level wherein the user may or may not have depression but may also be more prone to being depress. At that stage the user may display some depression like symptoms. lastly, Negative is the lowest level where depression and anxiety symptoms are being detected through the user’s message. The more negative words the user uses mean the more negative emotion the message has. This project will try to predict early signs of depression through social media text mining. For build the logic for that we are using Neural networks. Data set extracting from Social Media to feed into neural Networks.

### Problem Statement

The objective of this proposed system is to develop an application which will predict the depression of the person prior. So, person can get treatment to cure. This project will try to predict the depression of the person on behalf of the text message delivered by the him/her.

### Objective of the Project

The objectives of the “Early Depression Detection” can be stated as follows:

1. Develop a model that can classify the given social media messages to depression or no

depression.

1. For the same, build a Convolutional Neural Network (CNN) model that can understand the underlying linguistic information and perform the classification task.

# Chapter 2

## LITERATURE SURVEY

1. **Early detection of depression by primary care physicians** John W. Feightner, MD, MSc, FCFP; Graham Worrall, MB, BS, CCFP, MRCGP. This paper provides the details about the early depression signs and treatment.

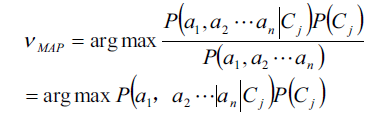
The early detection of depression is much more complex and challenging than the detection of other conditions seen in the primary care physician's office. Perhaps the most significant reason for this is how the natural history of depression differs from that of most conditions for which there are effective measures for prevention or early detection. The successful early detection of diseases or of patients at risk for adverse events has generally involved conditions that follow a continuous or progressive course and are detectable in the presymptomatic phase. Examples are cervical cancer (the Papanicolaou smear), breast cancer (clinical breast examination and mammography) and hypertension (screening for elevated blood pressure). The natural history of depression is not one of a continuous, unresolving disorder with a defined and detectable presymptomatic phase. In many cases the depression progresses continuously, becomes severe and is readily detected clinically. However, in some cases of mild and, indeed, more severe depression patients improve without specific intervention (or even recognition).

A second problem relates to the role that symptoms play in depression. The goal of early detection is to identify disorders in the presymptomatic stage. This is usually most successful when the condition has a relatively long presymptomatic stage, there are good objective confirmatory diagnostic tests and the symptoms often occur only at an advanced stage. In a sense depression has none of these three characteristics. It is diagnosed essentially on clinical grounds, the early stages are associated with symptoms (often nonspecific), and the presymptomatic" stage (even if there is one) is not necessarily prolonged, at least relative to other conditions. Since many early symptoms of depression also occur in other conditions (both physical and psychologic) the early detection of depression becomes even more complicated. Therefore, the development of an effective, accurate instrument is extremely difficult.

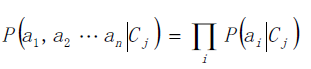
1. **NAIVE BAYES CLASSIFICATION ALGORITHM BASED ON SMALL SAMPLE SETH. Yuguang Huang, Lei Li** Beijing University of Posts and Telecommunications, Beijing, China huangyuguang1985@163.com, leili@bupt.edu.cn. This paper talks about Naïve Bayes Algorithms.

Naive Bayes classification algorithm is one of the most effective text classification methods, and in some areas its performance can be comparable with neural networks and decision tree learning. The Bayesian approach to classifying the new instance is to assign the most probable target value VMP, given the attribute values <a1,a2……,an> that describe the instance.

We can use Bayes theorem to rewrite this expression as: -



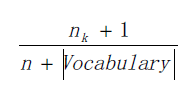
The Naïve Bayes classifier is based on the simplifying assumption that the attribute values are conditionally independent given the target value. In other words, the assumption is that given the target value of the instance, the probability of observing the conjunction a1,a2,……..an is just the product of the probabilities for the individual attributes:-



Substituting this into above Equation, we have the approach used by the Naïve Bayes classifier: -



Where VNBdenotes the target value output by the Naïve Bayes classifier. In the actual classification process, in order to avoid the case of p(ai/cj) =0, p(ai/cj) is estimated as the following formula: -



Where n is the total number of word positions in the target category, nk is the number of times word ai found among these n word positions, and |vocabulary| is the total number of distinct words found within the training data. In order to facilitate accurate computer storage, in the actual calculation, we finally adjust the Bayesian formula as: -

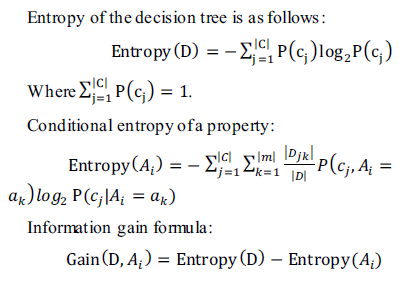


1. **Utilizing Neural Networks and Linguistic Metadata for Early Detection of Depression Indications in Text Sequences** Marcel Trotzek, Sven Koitka, and Christoph M. Friedrich, Member, IEEE. This paper provides the details about Emotions and Sentiment.

As sentiment analysis is focussed on extracting opinions, affects, and emotions from written texts, it seems natural that knowledge from this area can also be very useful to find emotional statements in the field of mental health text classification. Especially the emotions authors express towards their personal situation could be an important indicator. While it would be possible to use the output of any state-of-the-art sentiment classification model asan additional feature, this work has focussed on the use of lexicons to quickly analyze the general helpfulness of sentiment features in this dataset. First of all, the already described LIWC tool includes two features for positive and negative emotions and separate features that indicate anxiety, anger, or sadness. In addition to this, the NRC Emotion Lexicon and two general sentiment lexicons, namely the Opinion Lexicon12 and the VADER Sentiment Lexicon, have been used. There also exist several other lexicons that have not been evaluated, for example from the World Well-Being Project at University of Pennsylvania13. The NRC Emotion Lexicon contains 14,182 words that can be flagged as positive or negative and as belonging to one or more of the emotions anger, anticipation, disgust, fear, joy, sadness, surprise, and trust. The VADER lexicon includes 7,517 terms (including emoticons) and their mean sentiment value based on the judgement of ten human annotators on a scale between -4 (extremely negative) to 4 (extremely positive). Finally, the Opinion Lexicon consists of two lists with 2,006 positive and 4,783 negative words. The corresponding counts or scores obtained from these lexicons for the eRisk 2017 dataset were again averaged over all documents of a user. Unfortunately, for this specific dataset no relevant correlation between these features and the class label could be observed. Indeed, the positive (depressed) class contains slightly more emotions and sentiments of all kinds, which might again indicate the general difference of text quality and content between the depressed subjects and the control group. As the emotion and sentiment features were of no use in this specific case, they were not included in the final set of metadata features used in the experiments of this work. Nevertheless, it can be assumed that they would be more meaningful when used with a text corpus that generally included more sentiment and emotion in both classes by using a control group that more closely resembles the target group.

1. **Decision Tree Algorithm Optimization Research Based on MapReduce College of Computer and Information Engineering**, Inner Mongolia Normal University, Hohhot 010010, China, Xing An Vocational & Technical College, Wulanhaote 137400, China, CAS Key Laboratory of Genome Sciences and Information Beijing Institute of Genomics, Chinese Academy of Sciences Beijing 100101, China. This paper tells about the decision tree implementations.

Decision tree algorithm is one of inductive learning algorithms based on a set of training samples, mainly from a group of no rules, no order of sample set derived classification rules of decision tree representation. Decision tree is an automatic tree structure which is used to represent classification of data. It can be directly converted to the classification rules and seen as a tree-based prediction model. The root node of the tree is the entire set of data space, and each child node corresponds to a division problem that is a single variable of the training. The training data set is divided into two or more data blocks, each leaf node representing the category it belongs to. In the learning process, decision tree learning algorithm does not need to know all the variable factors, only need to know the tree node bifurcation discriminant being one kind corresponding node attribute values. How to get the most relevant variable properties is the main purpose of the introduction of information entropy, information gain maximum attribute as a selection test standard.



1. **Study of Depression Analysis using Machine Learning Techniques Devakunchari Ramalingam, Vaibhav Sharma, Priyanka Zar, International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-8, Issue-7C2, May 2019.** This paper provides the details about acoustic depression recognition.

**Chapter 3**

**THEORETICAL BACKGROUND**

Theoretical background highlighting some topics related to the project work is given below. The description contains several topics which are worth to discuss and also highlight some of their limitations that encourage going on finding solutions as well as highlights some of their advantages for which reason these topics and their features are used in this project.

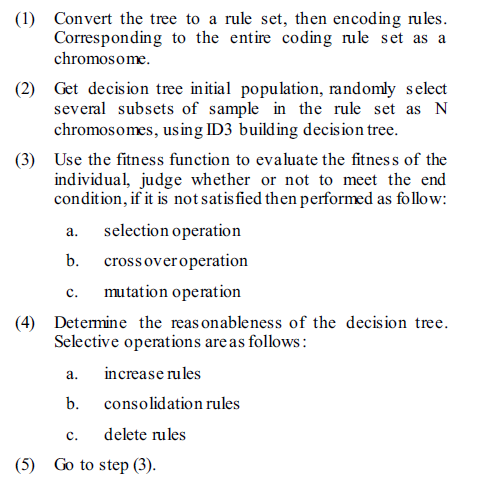
* 1. **Prediction Models**
     1. **Decision Tree**

With the explosive growth of information, “big data” is becoming more and more popular in all kinds of work which associated with large-scale data, including not only IT industry, but also business, finance, and bioinformatics, etc. Therefore, huge amount of data needs to be processed. In this environment, the cloud computing emerges as the times require. Cloud computing is a new calculation model blended multiple technologies, with the feather of high scalability, high reliability, high efficiency and low cost, etc. Hadoop is the computing platform of cloud computing, including MapReduce program model and Hadoop distribution file system (HDFS), which can realize large-scale distribution computing and parallel process. It is not necessary for programmers to know the low-level details of parallel processing. Decision tree algorithm is one of the important research topics in data mining. In recent years, the research mainly focused on the improvement of the decision tree algorithm, but facing the amount of data to be processed is growing exponentially, and from which to dig out useful information, the traditional decision tree algorithm has been difficult to meet requirement of reality. Emerging cloud computing technology which makes decision tree algorithm parallelization is one of the most effective methods. In the paper, on the basis of open source cloud computing platform, we study the parallel implementation of decision tree based on genetic algorithm (GA) optimization method used MapReduce framework.

* + - 1. **Basic principle of genetic decision tree algorithm**

Genetic algorithm is a swarm intelligence search optimization algorithm, inspired by evolution and genetics mechanism. The problem is encoded to generate the initial population, through the design of individual fitness function, genetic operators, the termination condition of the evolution, to find the way of global optimum. The ID3 is a kind of decision tree algorithm based on information gain size, lack of a global search strategy, it is difficult to find the optimal rule set corresponding tree.

The process of finding globally optimal rule set as follows:



* + 1. **Deep Neural Network**

The ﬁrst prediction model was based on a deep neural network (DNN). A DNN usually consists of an input layer, several hidden layers, and an output layer as shown in Figure1.

## 

**Figure 1:** An example of a fully-connected deep neural network (DNN)model.

In DNN prediction model, every node in one layer is connected to every node in the next layer, i.e., fully-connected. That is, each node in the hidden layer takes input vectors x1, . . …..., xn from the previous layer, where n is the number of nodes in the previous layer and the size of each Vector is m. Its output h is then defined as

where the weight vectors w1, . . ., wn and the bias b are parameters to be learned using training data. In addition, \* is an element-wise multiplication (Hadamard product) and σh is a nonlinear function, such as the logistic sigmoid, deﬁned by the activation function

and the rectiﬁed linear unit (ReLU), deﬁned as g(x) = max (0, x). σh is applied element-wise, and thus the size of his also m. In the output layer, the set of input vectors are ﬂattened into a single vector and then translated into a single value by using a dot product between the ﬂattened vector and a weight vector.

# The Confusion Matrix and Disagreement Score

A confusion matrix of size n x n associated with a classifier shows the predicted and actual classification, where n is the number of different classes. Table 1 shows a confusion matrix for n = 2, whose entries have the following meanings:

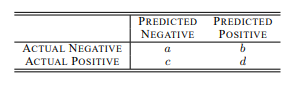
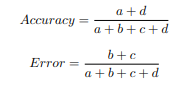


Figure 8: The confusion matrix for two-class classification problem.

* a is the number of correct negative predictions.
* b is the number of incorrect positive predictions.
* c is the number of incorrect negative predictions.
* d is the number of correct positive predictions.

The prediction accuracy and classification error can be obtained from this matrix as follows:



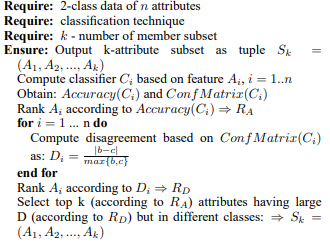
We define the disagreement score associated with a confusion matrix in below equation. According to this equation the disagreement is 1 when one of the quantities b or c is 0 (in this case the classifier misclassifies examples of one class only), and is 0 when b and c are the same



The attribute selection methodology proposed here selects attributes that not only have good discrimination power on their own, but more importantly are complementary to each other. For example, consider two attributes A1 and A2, having similar classification accuracy. Our approach will select them as a good subset of attributes if they have a large disagreement in terms of what examples they misclassify. A large disagreement is indicated by D values closer to 1 for both attributes, but distinct denominators in above equation.

Algorithm for Confusion Matrix-based Attribute Selection:

The pseudocode outlined below shows the steps to perform confusion matrix-based attribute selection for a 2-class classification problem. This method basically constructs attribute-subsets that: (1) have attributes with good individual classification power, and (2) have attributes that are complementary (i.e. they disagree in their misclassifications). Note that the algorithm may lead to several subsets of attributes to be further investigated, i.e. further the subset yielding higher classification accuracy may be selected. Also, the algorithm does not account for the possibility that two individually lower ranked attributes may combine in a high classification accuracy subset due to their high complementarity.



# Chapter 4

**SYSTEM REQUIREMENT SPECIFICATION**

A System Requirement Specification (SRS) is basically an organization’s understanding of a customer or potential client’s system requirements and dependencies at a particular point prior to any actual design or development work. The information gathered during the analysis is translated into a document that defines a set of requirements. It gives a brief description of the services that the system should provide and also the constraints under which the system should operate. Generally, SRS is a document that completely describes what the proposed software should do without describing how the software will do it. It’s a two-way insurance policy that assures that both the client and the organization understand the other’s requirements from that perspective at a given point in time.

SRS document itself states in precise and explicit language those functions and capabilities a software system (i.e., a software application, an ecommerce website and so on) must provide, as well as states any required constraints by which the system must abide. SRS also functions as a blueprint for completing a project with as little cost growth as possible. SRS is often referred to as the “parent” document because all subsequent project management documents, such as design specifications, statements of work, software architecture specifications, testing and validation plans, and documentation plans, are related to it.

Requirement is a condition or capability to which the system must conform. Requirement Management is a systematic approach towards eliciting, organizing and documenting the requirements of the system clearly along with the applicable attributes. The elusive difficulties of requirements are not always obvious and can come from any number of sources.

### 4.1 Functional Requirement

Functional Requirement defines a function of a software system and how the system must behave when presented with specific inputs or conditions. These may include calculations, data manipulation and processing and other specific functionality. In this system following are the functional requirements: -

Following are the functional requirements on the system:

1. The entire control model set must be translated to C output Code.
2. Inputs must be models designed using CLAW design components along with standard design components.
3. Multiple design models must be processed and the result must be combined to obtain a single output file.

### 4.2 Non-Functional Requirement

Non-functional requirements are the requirements which are not directly concerned with the specific function delivered by the system. They specify the criteria that can be used to judge the operation of a system rather than specific behaviors. They may relate to emergent system properties such as reliability, response time and store occupancy. Non-functional requirements arise through the user needs, because of budget constraints, organizational policies, the need for interoperability with other software and hardware systems or because of external factors such as:-

* Product Requirements
* Organizational Requirements
* User Requirements
* Basic Operational Requirements

### 4.2.1 Product Requirements

**Platform Independency: S**tandalone executables for embedded systems can be created sothe algorithm developed using available products could be downloaded on the actual hardware and executed without any dependency to the development and modeling platform.

**Correctness:** It followed a well-defined set of procedures and rules to compute and also rigorous testing is performed to confirm the correctness of the data.

**Ease of Use:** Model Coder provides an interface which allows the user to interact in an easy manner.

**Modularity:** The complete product is broken up into many modules and well- defined interfaces are developed to explore the benefit of flexibility of the product.

**Robustness:** This software is being developed in such a way that the overall performance is optimized and the user can expect the results within a limited time with utmost relevance and correctness

Non-functional requirements are also called the qualities of a system. These qualities can be divided into execution quality & evolution quality. Execution qualities are security & usability of the system which are observed during run time, whereas evolution quality involves testability, maintainability, extensibility or scalability.

### Organizational Requirements

**Process Standards:** The standards defined by DRDO are used to develop the application which is the standard used by the developers inside the defense organization.

Design Methods: Design is one of the important stages in the software engineering process. This stage is the first step in moving from problem to the solution domain. In other words, starting with what is needed design takes us to work how to satisfy the needs.

### 4.2.3 User Requirements

* + The coder must request the name of the model file to be processed
  + In case of multiple files, the coder must ask the names of the files sequentially.
  + The output file must be a C code translated from the model.
  + Only a single output file must be created even if multiple input files are provided.

### 4.2.4 Basic Operational Requirements

The customers are those that perform the eight primary functions of systems engineering, with special emphasis on the operator as the key customer. Operational requirements will define the basic need and, at a minimum, will be related to these following points: -

**Mission profile or scenario:** It describes the procedures used to accomplish mission objectives. It also finds out the effectiveness or efficiency of the system.

**Performance and related parameters:** It point out the critical system parameters to accomplish the mission.

**Utilization environments:** It gives a brief outline of system usage. Finds out appropriate environments for effective system operation.

**Operational life cycle:** It defines the system lifetime.

### 4.2.5 System Configuration

H/W System Configuration:

|  |  |  |
| --- | --- | --- |
| Processor | - | Pentium –IV |
| Speed | - | 1.1 Ghz |
| RAM | - | 4GB RAM |
| Hard Disk | - | 20 GB |
| Key Board | - | Standard Windows Keyboard |
| Mouse | - Two or Three Button Mouse | |
| Monitor | - | SVGA |
| S/W System Configuration: | | |
| Operating System | : XP/7/8/8.1/10 | |
| Coding Language | : Python-3.7.0 | |

# Chapter 5

**SYSTEM ANALYSIS**

Analysis is the process of finding the best solution to the problem. System analysis is the process by which we learn about the existing problems, define objects and requirements and evaluate the solutions. It is the way of thinking about the organization and the problem it involves, a set of technologies that helps in solving these problems. Feasibility study plays an important role in system analysis which gives the target for design and development.

## 5.1 Feasibility Study

All systems are feasible when provided with unlimited resources and infinite time. But unfortunately, this condition does not prevail in the practical world. So, it is both necessary and prudent to evaluate the feasibility of the system at the earliest possible time. Months or years of effort, thousands of rupees and untold professional embarrassment can be averted if an ill- conceived system is recognized early in the definition phase. Feasibility & risk analysis are related in many ways. If project risk is great, the feasibility of producing quality software is reduced. In this case three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

### .1 Economic Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

### Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

## 5.2 Analysis

### 5.2.1 Performance Analysis

For the complete functionality of the project work, the project is run with the help of a healthy networking environment. Performance analysis is done to find out whether the proposed system. It is essential that the process of performance analysis and definition must be conducted in parallel.

### 5.2.2 Technical Analysis

System is only beneficial only if it can be turned into information systems that will meet the organization’s technical requirement. Simply stated this test of feasibility asks whether the system will work or not when developed & installed, whether there are any major barriers to implementation. Regarding all these issues in technical analysis there are several points to focus on: -

**Changes to bring in the system:** All changes should be in a positive direction, there will be increased level of efficiency and better customer service.

**Required skills:** Platforms & tools used in this project are widely used. So,the skilled manpower is readily available in the industry.

**Acceptability:** The structure of the system is kept feasible enough so that there should not be any problem from the user’s point of view.

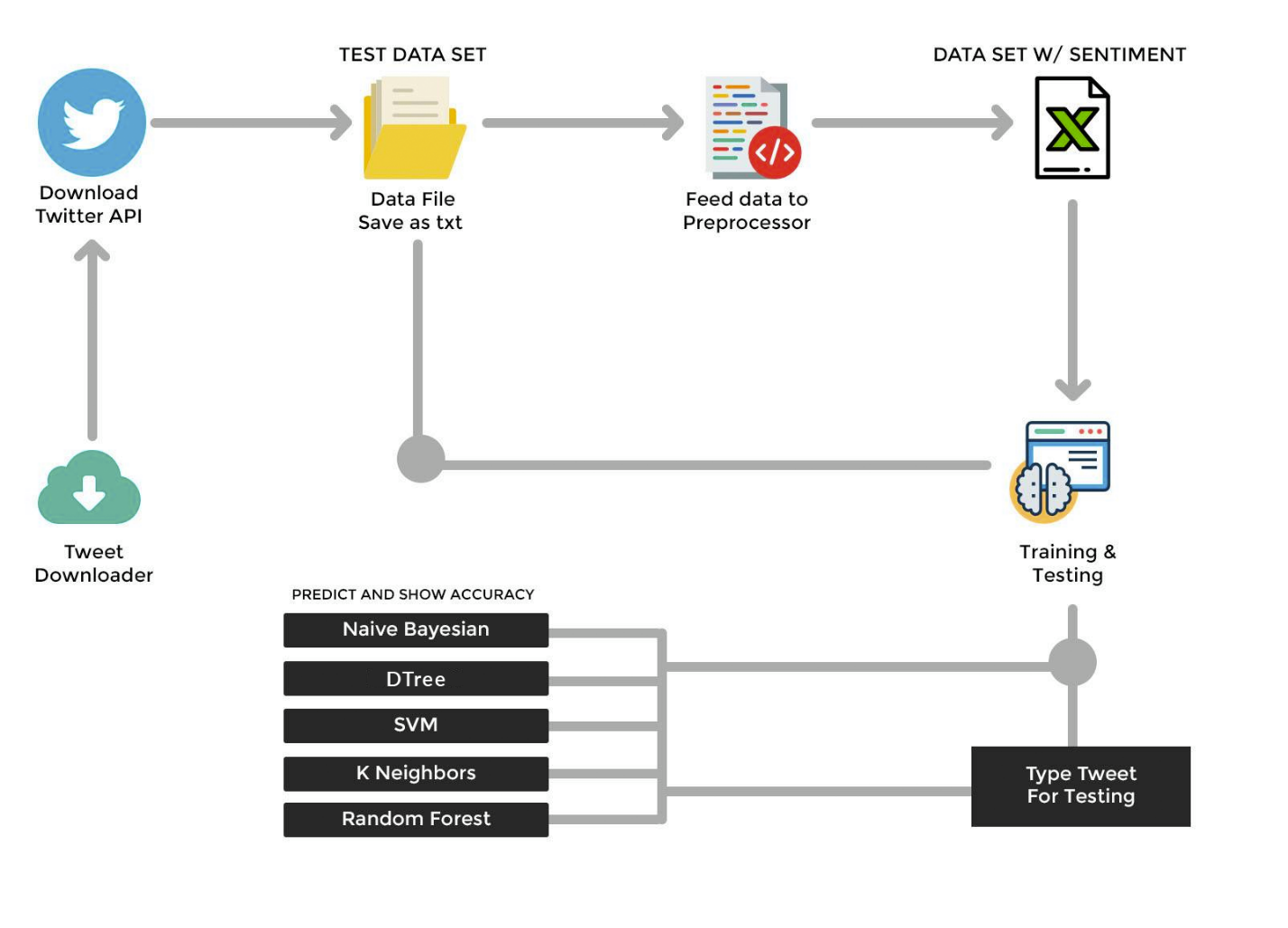
### 5.2.3 Economical Analysis

Economic analysis is performed to evaluate the development cost weighed against the ultimate income or benefits derived from the developed system. For running this system, we need not have any routers which are highly economical. So, the system is economically feasible enough.

**Chapter 6**

**SYSTEM DESIGN**

The aim of the project is to predict early signs of depression through **Social Media** text mining. Below are the steps to find the early signs of Depression of the person.



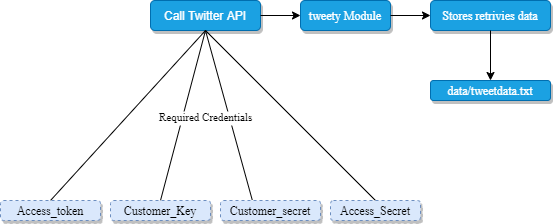
**Figure 6** : Flow Diagram to get Results

In the process to find the sing of detection follows few steps: -

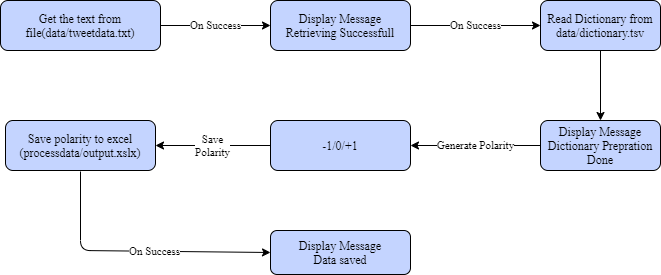
Step 1: Create a twitter developers account, from that account you would need 4 things.

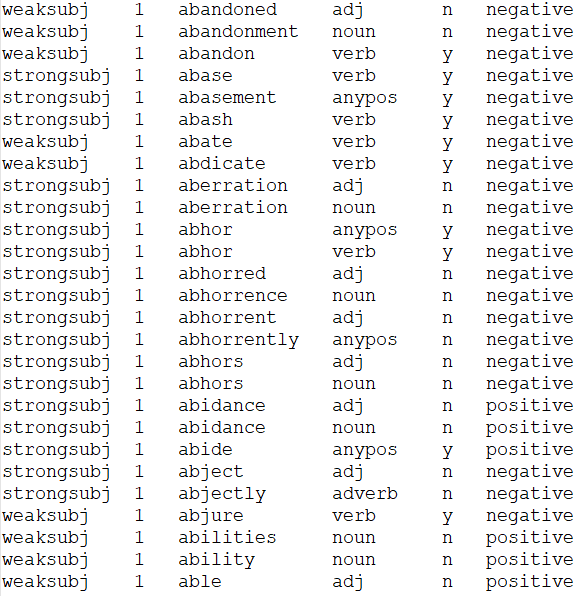
consumer\_key, consumer\_secret, access\_token, access\_secret.

Step 2: After the approval you will the data from twitter account that we will for text mining.



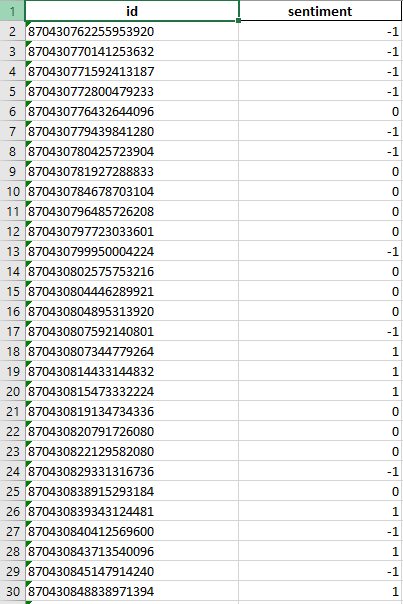
Step 3: This stage will go through your data sets and the given dictionary. The dictionary contains words with their corresponding polarity, which is essential to calculating the sentiment of each tweet, each word will be separated, tokenized and given its polarity. Every tweet will consist of the summation of all polarity of each word and divided by number of words in that tweet.





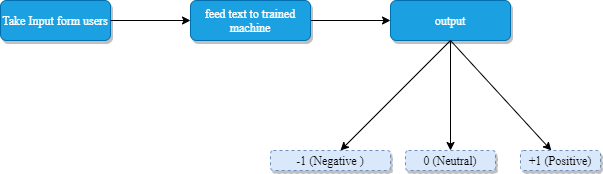
**Figure 7**: Dictionary

Step 4: Once preprocess is done. You can find the file in the directory "processed\_data/output.xlsx". Opening it you will find that the ID (tweet) and Sentiment of each tweet is separated into 2 columns. With this output you now have a twitter data set and its corresponding sentiment filtered by depress keywords. (Positive, Neutral and Negative).



**Figure 8**: Sentimental

Step 5: Now for training and Predicting. Make sure all files are located in proper folders, Run "depression\_sentiment\_analysis.py". The code will run through the output.xlsx file and at the same time recover the tweet corresponding to the id of each sentiment. using this we use the original data and feed them to our classifiers. Every moments output will be given on console.



# Chapter 7

# IMPLEMENTATION

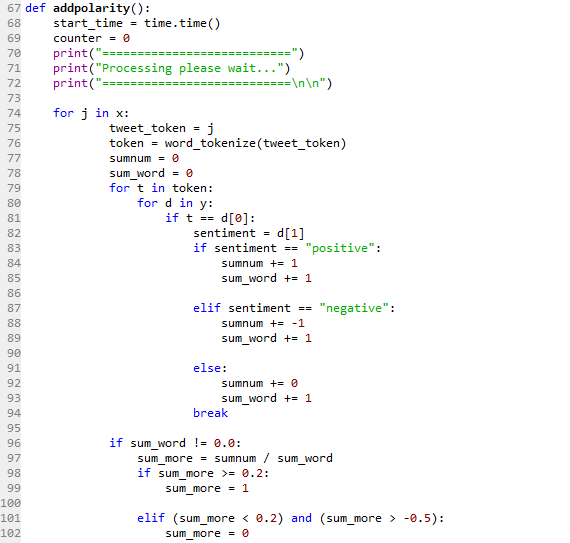
Step 1: Downlaod\_twitter\_api.py



Step 2: Preprocessor.py

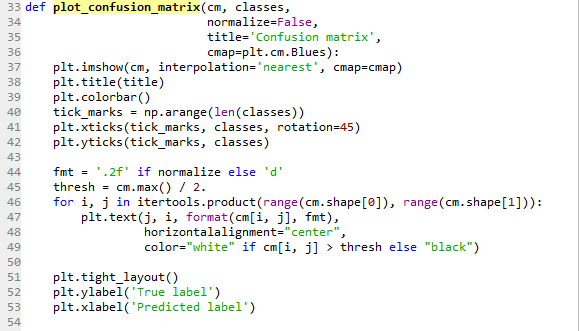


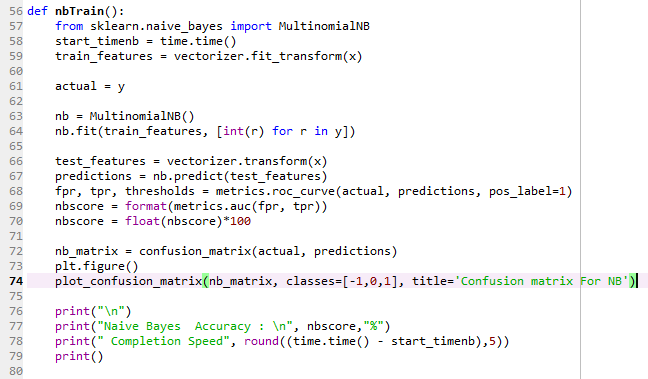


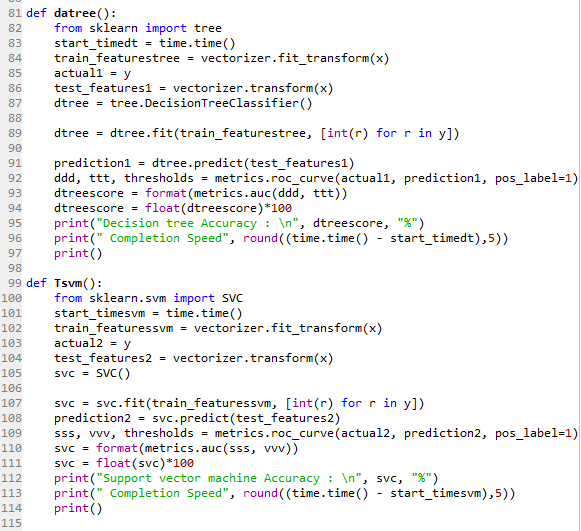


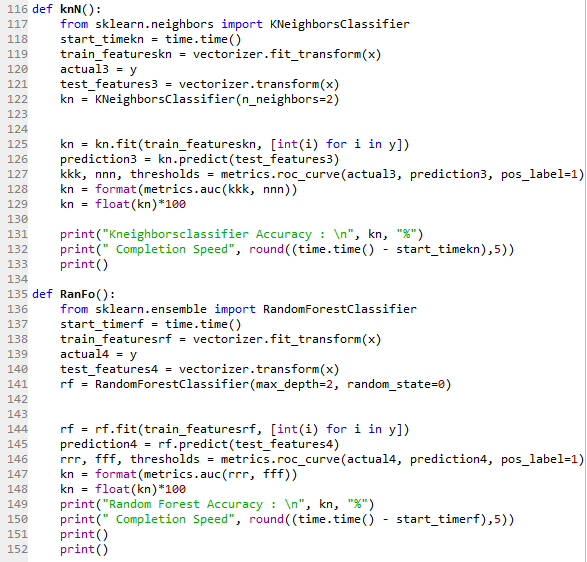


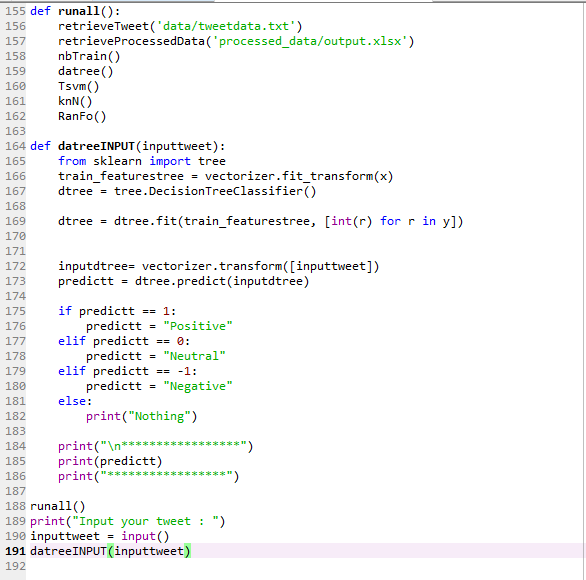
Step 3: Depression\_sentiment\_analysis.py







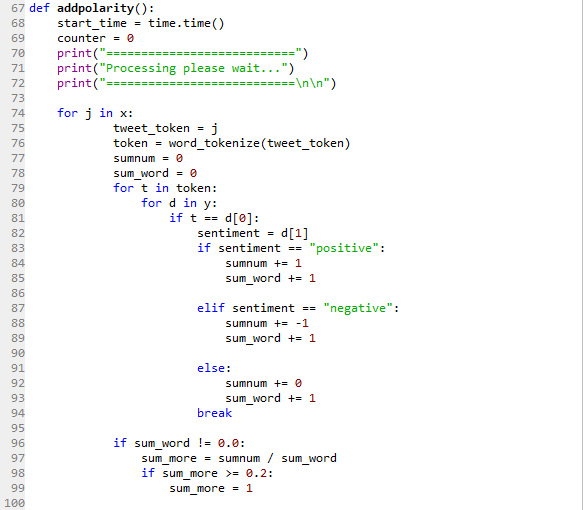


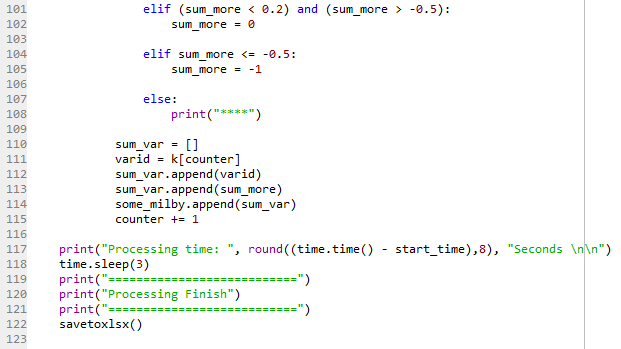


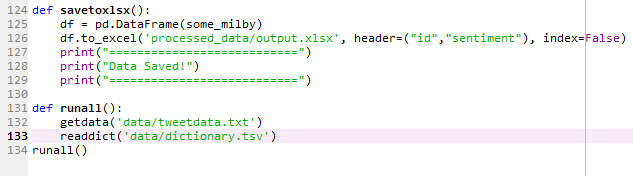
Step 4: Accuracy\_checker.py





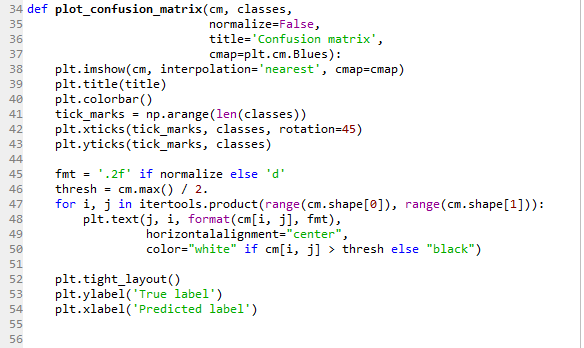
****

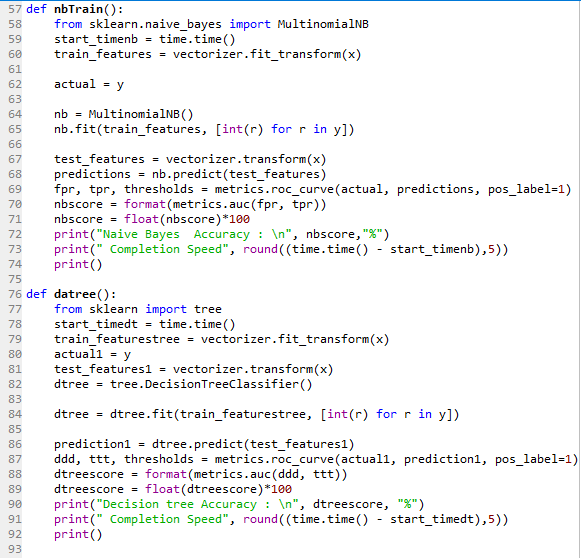
****

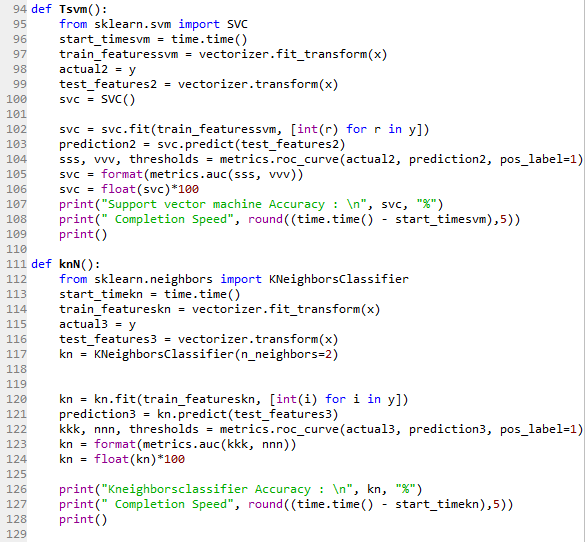
****

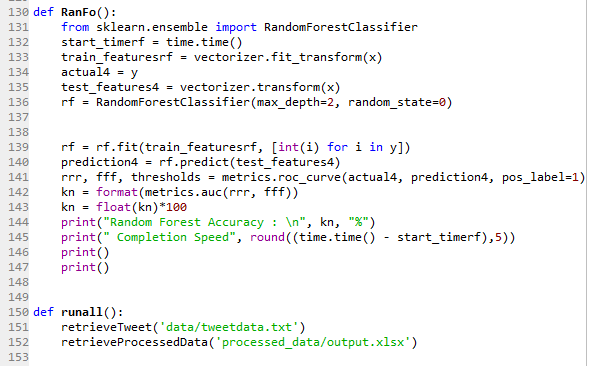
Step 5: Inset\_tweetDemo.py

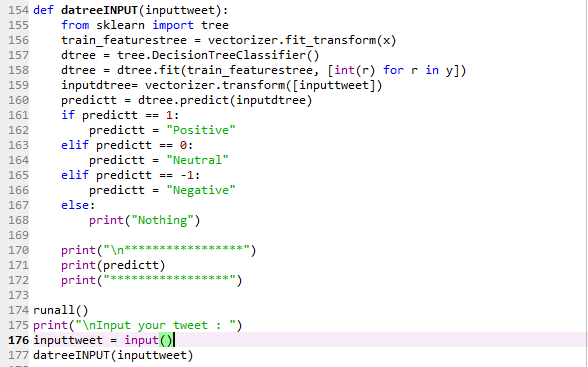


****

****

****

****

****

**Chapter 8**

**TESTING**

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that all the system elements have been properly integrated and perform allocated functions. The testing process is actually carried out to make sure that the product exactly does the same thing what is supposed to do. In the testing stage following goals are tried to achieve: -

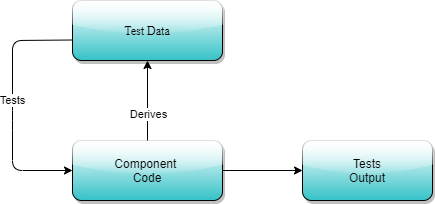
* + - To affirm the quality of the project.
    - To find and eliminate any residual errors from previous stages.
    - To validate the software as a solution to the original problem.
    - To provide operational reliability of the system.

## Testing Methodologies

There are many different types of testing methods or techniques used as part of the software testing methodology. Some of the important testing methodologies are:

### White box testing

White box testing (clear box testing, glass box testing, and transparent box testing or structural testing) uses an internal perspective of the system to design test cases based on internal structure. It requires programming skills to identify all paths through the software. The tester chooses test case inputs to exercise paths through the code and determines the appropriate outputs. While white box testing is applicable at the unit, integration and system levels of the software testing process, it is typically applied to the unit. While it normally tests paths within a unit, it can also test paths between units during integration, and between subsystems during a system level test.

****

**Fig 8.1.1:** White Box Testing

Though this method of test design can uncover an overwhelming number of test cases, it might not detect unimplemented parts of the specification or missing requirements, but one can be sure that all paths through the test object are executed. Using white box testing we can derive test cases that:

1. Guarantee that all independent paths within a module have been exercised at least once.
2. Exercise all logical decisions on their true and false sides.
3. Execute all loops at their boundaries and within their operational bounds.
4. Execute internal data structure to assure their validity

### 8.1.1.1 Advantages of White Box Testing

* To start the white box testing of the desired application there is no need to wait for user face (UI) to be completed. It covers all possible paths of code which will ensure a thorough testing.
* It helps in checking coding standards.
* Tester can ask about implementation of each section, so it might be possible to remove unused/deadlines of codes helps in reducing the number of test cases to be executed during the black box testing.
* As the tester is aware of internal coding structure, then it is helpful to derive which type of input data is needed to test the software application effectively.
* White box testing allows you to help in code optimization

### 8.1.1.2 Disadvantages of White Box Testing

* To test the software application a highly skilled resource is required to carry out testing who has good knowledge of internal structure of the code which will increase the cost.
* Updating the test script is required if there is change in requirement too frequently.
* If the application to be tested is large in size, then exhaustive testing is impossible.
* It is not possible for testing each and every path/condition of software program, which might miss the defects in code.
* White box testing is a very expensive type of testing.
* To test each paths or conditions may require different input conditions, so in order to test full application, the tester need to create range of inputs which may be a time consuming.

### 8.1.2 Black box testing

Black box testing focuses on the functional requirements of the software. It is also known as functional testing. It is a [software](http://www.webopedia.com/TERM/B/software.html) testing technique whereby the internal workings of the item being tested are not known by the tester. For example, in a black box test on software design the tester only knows the inputs and what the expected outcomes should be and not how the program arrives at those outputs.

The tester does not ever examine the programming [code](http://www.webopedia.com/TERM/B/code.html) and does not need any further knowledge of the program other than its specifications. It enables us to derive sets of inputs that will fully exercise all functional requirements for a program.



**Fig 8.1.2:** Black Box Testing

Black box testing is an alternative to white box technique. Rather it is a complementary approach that is likely to uncover a different class of errors in the following categories: -

* Incorrect or missing function.
* Interface errors.
* Performance errors.
* Initialization and termination errors.
* Errors in objects.

### 8.1.2.1 Advantages of Black Box Testing

* The test is unbiased as the designer and the tester are independent of each other.
* The tester does not need knowledge of any specific programming languages.
* The test is done from the point of view of the user, not the designer.
* Test cases can be designed as soon as the specifications are complete.

### 8.1.2.2 Disadvantages of Black Box Testing

* The test inputs need to be from large sample space. That is, from a huge set of data this willtake time.
* Also, it is difficult to identify all possible inputs in limited testing time. So writing test cases is slow and difficult.
* Chances are more that there will be unidentified paths during this testing.

## 8.2 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

## 8.3 System Testing

This information contributes towards reducing the ambiguity about the system. For example, when deciding whether to release a product, the decision makers would need to know the state of the product including aspects such as the conformance of the product to requirements, the usability of the product, any known risks, the product’s compliance to any applicable regulations,

Software testing enables making objective assessments regarding the degree of conformance of the system to stated requirements and specifications.

System testing checks complete end-end scenarios, as a user would exercise the system. The system has to be tested for correctness of the functionality by setting it up in a controlled environment. System testing includes testing of functional and nonfunctional requirements. It helps to verify and validate the system. All components

system should have been successfully unit tested and then checked for any errors after integration.

## 8.4 Quality Assurance

Quality assurance consists of the auditing and reporting functions of management. The goal of quality assurance is to provide management with the data necessary to be informed about product quality, thereby gaining insight and confident that the product quality is meeting its goals. This is an “umbrella activity” that is applied throughout the engineering process***.*** Software quality assurance encompasses: -

* Analysis, design, coding and testing methods and tools.
* Formal technical reviews that are applied during each software engineering.
* Multi-tiered testing strategy.
* Control of software documentation and the change made to it.
* A procedure to ensure compliance with software development standards.
* Measurement and reporting mechanisms.

### 8.4.1 Quality Factors

An important objective of quality assurance is to track the software quality and assess the impact of methodological and procedural changes on improved software quality. The factors that affect the quality can be categorized into two broad groups:

* Factors that can be directly measured.
* Factors that can be indirectly measured

These factors focus on three important aspects of a software product

* Its operational characteristics
* Its ability to undergo changes
* Its adaptability to a new environment.
* Effectiveness or efficiency in performing its mission
* Duration of its use by its customer.

### 8.5 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised. Systems/Procedures: Interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

## Chapter 9

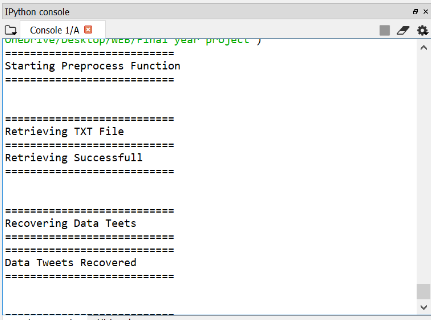
**RESULT AND PERFORMANCE ANALYSIS**

# SNAPSHOTS

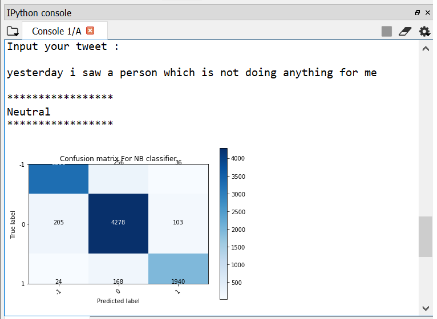
Step 1:Downlaod\_twitter\_api.py



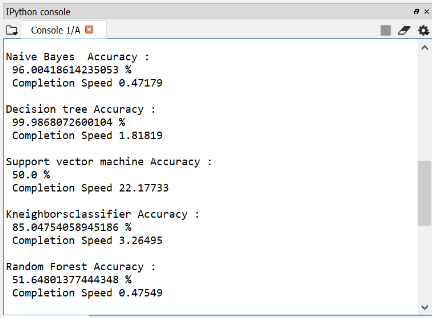
Step 2: Preprocessor.py



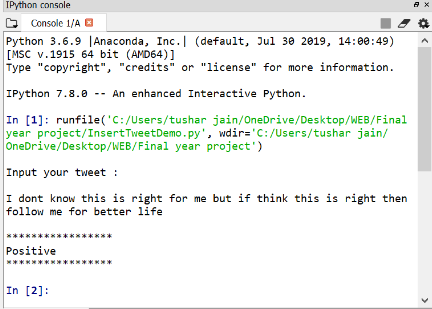
Step 3: Depression\_sentiment\_analysis.py

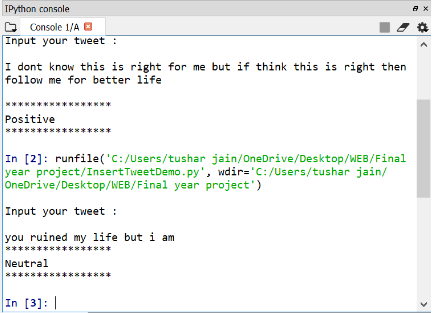


Step 4: Accuracy\_checker.py



Step 5: Inset\_tweetDemo.py





# Chapter 10

# 

# CONCLUSION AND FUTURE SCOPE

What the result could mean? **Positive**, this mean that person is unlikely to have depression or anxiety. **Neutral**, this is the middle level wherein the user may or may not have depression but may also be more prone to being depress. At that stage the user may display some depression like symptoms. lastly, **Negative** is the lowest level where depression and anxiety symptoms are being detected through the user’s tweets. The more negative words the user uses mean the more negative emotion the tweet has. In the given module we can add more words in dictionary to improve the result that leads to perfect results. To improvised the accuracy, we can add n-gram techniques for word mining, reduce the stop word eliminating frequency.

* + 1. **Download\_twitter\_API.py**

from tweepy.streaming import StreamListener

from tweepy import OAuthHandler

from tweepy import Stream

consumer\_key = 'd7RJDeV6M1TdKnXXdY29Zud5O'

consumer\_secret = '8LV35luiAco2mBnQ1W6erOnA8cbMwVgxblfHjP5zk5dmAXGwd6'

access\_token = '2206645458-9qlftwQ5eiovob7GCp21VrAoFRXi7AJLGt5ts3O'

access\_secret = 'Oc9ZKbHSL0reJhZYcU0Vk9UERbVvsTwerIfDUTwiRNGYf'

class StdOutListener(StreamListener):

def on\_data(self, data):

with open('data/tweetdata.txt','a') as tf:

tf.write(data)

print(data)

return True

def on\_error(self, status):

print(status)

if \_\_name\_\_ == '\_\_main\_\_':

l = StdOutListener()

auth = OAuthHandler(consumer\_key, consumer\_secret)

auth.set\_access\_token(access\_token, access\_secret)

stream = Stream(auth, l)

stream.filter(track=['depression', 'anxiety', 'mental health', 'suicide', 'stress', 'sad'])

* + 1. **preprocessor.py**

import json

import csv

from nltk.tokenize import word\_tokenize

import string

import re

import time

import pandas as pd

tweets\_data = []

x = []

y = []

k = []

some\_milby = []

print("===========================")

print("Starting Preprocess Function")

print("=========================== \n\n")

def getdata(dataurl):

print("===========================")

print("Retrieving TXT File")

tweets\_data\_path = dataurl

tweets\_file = open(tweets\_data\_path, "r")

for line in tweets\_file:

try:

tweet = json.loads(line)

tweets\_data.append(tweet)

except:

continue

print("===========================")

print("Retrieving Successfull")

print("=========================== \n \n")

time.sleep(3)

processdata()

def processdata():

print("===========================")

print("Recovering Data Teets")

print("===========================")

time.sleep(1)

RE\_EMOJI = re.compile('[\U00010000-\U0010ffff]', flags=re.UNICODE)

for i in range(len(tweets\_data)):

q = tweets\_data[i]['text']

o = tweets\_data[i]['id\_str']

q = RE\_EMOJI.sub(r'', q)

i = q.translate(str.maketrans('','',string.punctuation))

x.append(i)

k.append(o)

print("===========================")

print("Data Tweets Recovered")

print("===========================\n\n")

def readdict(dataurl):

print("===========================")

print("Reading Dictionary")

print("===========================")

with open(dataurl) as tsvfile:

reader = csv.reader(tsvfile, delimiter='\t')

for row in reader:

i = []

i.append(row[2])

i.append(row[5])

y.append(i)

print("===========================")

print("Dictionary Preparation Done")

print("===========================\n\n")

addpolarity()

def addpolarity():

start\_time = time.time()

counter = 0

print("===========================")

print("Processing please wait...")

print("===========================\n\n")

for j in x:

tweet\_token = j

token = word\_tokenize(tweet\_token)

sumnum = 0

sum\_word = 0

sum\_more = 0

for t in token:

for d in y:

if t == d[0]:

sentiment = d[1]

if sentiment == "positive":

sumnum += 1

sum\_word += 1

elif sentiment == "negative":

sumnum += -1

sum\_word += 1

else:

sumnum += 0

sum\_word += 1

break

if sum\_word != 0.0:

sum\_more = sumnum / sum\_word

if sum\_more >= 0.2:

sum\_more = 1

elif (sum\_more < 0.2) and (sum\_more > -0.5):

sum\_more = 0

elif sum\_more <= -0.5:

sum\_more = -1

else:

print("\*\*\*\*")

sum\_var = []

varid = k[counter]

sum\_var.append(varid)

sum\_var.append(sum\_more)

some\_milby.append(sum\_var)

counter += 1

print("Processing time: ", round((time.time() - start\_time),8), "Seconds \n\n")

time.sleep(3)

print("===========================")

print("Processing Finish")

print("===========================")

savetoxlsx()

def savetoxlsx():

df = pd.DataFrame(some\_milby)

df.to\_excel('processed\_data/output.xlsx', header=("id","sentiment"), index=False)

print("===========================")

print("Data Saved!")

print("===========================")

def runall():

getdata('data/tweetdata.txt')

readdict('data/dictionary.tsv')

runall()

* + 1. **depression\_sentiment\_analysis.py**

from warnings import simplefilter

simplefilter(action='ignore', category=FutureWarning)

import json

import pandas as pd

import time

import numpy as np

import itertools

import matplotlib.pyplot as plt

from sklearn.metrics import confusion\_matrix

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn import metrics

tweets\_data = []

x = []

y = []

vectorizer = CountVectorizer(stop\_words='english')

def retrieveTweet(data\_url):

tweets\_data\_path = data\_url

tweets\_file = open(tweets\_data\_path, "r")

for line in tweets\_file:

try:

tweet = json.loads(line)

tweets\_data.append(tweet)

except:

continue

def retrieveProcessedData(Pdata\_url):

sent = pd.read\_excel(Pdata\_url)

for i in range(len(tweets\_data)):

if tweets\_data[i]['id']==sent['id'][i]:

x.append(tweets\_data[i]['text'])

y.append(sent['sentiment'][i])

def plot\_confusion\_matrix(cm, classes,

normalize=False,

title='Confusion matrix',

cmap=plt.cm.Blues):

plt.imshow(cm, interpolation='nearest', cmap=cmap)

plt.title(title)

plt.colorbar()

tick\_marks = np.arange(len(classes))

plt.xticks(tick\_marks, classes, rotation=45)

plt.yticks(tick\_marks, classes)

fmt = '.2f' if normalize else 'd'

thresh = cm.max() / 2.

for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

plt.text(j, i, format(cm[i, j], fmt),

horizontalalignment="center",

color="white" if cm[i, j] > thresh else "black")

plt.tight\_layout()

plt.ylabel('True label')

plt.xlabel('Predicted label')

def nbTrain():

from sklearn.naive\_bayes import MultinomialNB

start\_timenb = time.time()

train\_features = vectorizer.fit\_transform(x)

actual = y

nb = MultinomialNB()

nb.fit(train\_features, [int(r) for r in y])

test\_features = vectorizer.transform(x)

predictions = nb.predict(test\_features)

fpr, tpr, thresholds = metrics.roc\_curve(actual, predictions, pos\_label=1)

nbscore = format(metrics.auc(fpr, tpr))

nbscore = float(nbscore)\*100

nb\_matrix = confusion\_matrix(actual, predictions)

plt.figure()

plot\_confusion\_matrix(nb\_matrix, classes=[-1,0,1], title='Confusion matrix For NB classifier')

print("\n")

print("Naive Bayes Accuracy : \n", nbscore,"%")

print(" Completion Speed", round((time.time() - start\_timenb),5))

print()

def datree():

from sklearn import tree

start\_timedt = time.time()

train\_featurestree = vectorizer.fit\_transform(x)

actual1 = y

test\_features1 = vectorizer.transform(x)

dtree = tree.DecisionTreeClassifier()

dtree = dtree.fit(train\_featurestree, [int(r) for r in y])

prediction1 = dtree.predict(test\_features1)

ddd, ttt, thresholds = metrics.roc\_curve(actual1, prediction1, pos\_label=1)

dtreescore = format(metrics.auc(ddd, ttt))

dtreescore = float(dtreescore)\*100

print("Decision tree Accuracy : \n", dtreescore, "%")

print(" Completion Speed", round((time.time() - start\_timedt),5))

print()

def Tsvm():

from sklearn.svm import SVC

start\_timesvm = time.time()

train\_featuressvm = vectorizer.fit\_transform(x)

actual2 = y

test\_features2 = vectorizer.transform(x)

svc = SVC()

svc = svc.fit(train\_featuressvm, [int(r) for r in y])

prediction2 = svc.predict(test\_features2)

sss, vvv, thresholds = metrics.roc\_curve(actual2, prediction2, pos\_label=1)

svc = format(metrics.auc(sss, vvv))

svc = float(svc)\*100

print("Support vector machine Accuracy : \n", svc, "%")

print(" Completion Speed", round((time.time() - start\_timesvm),5))

print()

def knN():

from sklearn.neighbors import KNeighborsClassifier

start\_timekn = time.time()

train\_featureskn = vectorizer.fit\_transform(x)

actual3 = y

test\_features3 = vectorizer.transform(x)

kn = KNeighborsClassifier(n\_neighbors=2)

kn = kn.fit(train\_featureskn, [int(i) for i in y])

prediction3 = kn.predict(test\_features3)

kkk, nnn, thresholds = metrics.roc\_curve(actual3, prediction3, pos\_label=1)

kn = format(metrics.auc(kkk, nnn))

kn = float(kn)\*100

print("Kneighborsclassifier Accuracy : \n", kn, "%")

print(" Completion Speed", round((time.time() - start\_timekn),5))

print()

def RanFo():

from sklearn.ensemble import RandomForestClassifier

start\_timerf = time.time()

train\_featuresrf = vectorizer.fit\_transform(x)

actual4 = y

test\_features4 = vectorizer.transform(x)

rf = RandomForestClassifier(max\_depth=2, random\_state=0)

rf = rf.fit(train\_featuresrf, [int(i) for i in y])

prediction4 = rf.predict(test\_features4)

rrr, fff, thresholds = metrics.roc\_curve(actual4, prediction4, pos\_label=1)

kn = format(metrics.auc(rrr, fff))

kn = float(kn)\*100

print("Random Forest Accuracy : \n", kn, "%")

print(" Completion Speed", round((time.time() - start\_timerf),5))

print()

print()

def runall():

retrieveTweet('data/tweetdata.txt')

retrieveProcessedData('processed\_data/output.xlsx')

nbTrain()

datree()

Tsvm()

knN()

RanFo()

def datreeINPUT(inputtweet):

from sklearn import tree

train\_featurestree = vectorizer.fit\_transform(x)

dtree = tree.DecisionTreeClassifier()

dtree = dtree.fit(train\_featurestree, [int(r) for r in y])

inputdtree= vectorizer.transform([inputtweet])

predictt = dtree.predict(inputdtree)

if predictt == 1:

predictt = "Positive"

elif predictt == 0:

predictt = "Neutral"

elif predictt == -1:

predictt = "Negative"

else:

print("Nothing")

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print(predictt)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

runall()

print("Input your tweet : ")

inputtweet = input()

datreeINPUT(inputtweet)

* + 1. **InsertTweetDemo.py**

import json

import pandas as pd

import time

import numpy as np

import itertools

import matplotlib.pyplot as plt

from sklearn.metrics import confusion\_matrix

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn import metrics

tweets\_data = []

x = []

y = []

vectorizer = CountVectorizer(stop\_words='english')

def retrieveTweet(data\_url):

tweets\_data\_path = data\_url

tweets\_file = open(tweets\_data\_path, "r")

for line in tweets\_file:

try:

tweet = json.loads(line)

tweets\_data.append(tweet)

except:

continue

def retrieveProcessedData(Pdata\_url):

sent = pd.read\_excel(Pdata\_url)

for i in range(len(tweets\_data)):

if tweets\_data[i]['id']==sent['id'][i]:

x.append(tweets\_data[i]['text'])

y.append(sent['sentiment'][i])

def plot\_confusion\_matrix(cm, classes,

normalize=False,

title='Confusion matrix',

cmap=plt.cm.Blues):

plt.imshow(cm, interpolation='nearest', cmap=cmap)

plt.title(title)

plt.colorbar()

tick\_marks = np.arange(len(classes))

plt.xticks(tick\_marks, classes, rotation=45)

plt.yticks(tick\_marks, classes)

fmt = '.2f' if normalize else 'd'

thresh = cm.max() / 2.

for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

plt.text(j, i, format(cm[i, j], fmt),

horizontalalignment="center",

color="white" if cm[i, j] > thresh else "black")

plt.tight\_layout()

plt.ylabel('True label')

plt.xlabel('Predicted label')

def nbTrain():

from sklearn.naive\_bayes import MultinomialNB

start\_timenb = time.time()

train\_features = vectorizer.fit\_transform(x)

actual = y

nb = MultinomialNB()

nb.fit(train\_features, [int(r) for r in y])

test\_features = vectorizer.transform(x)

predictions = nb.predict(test\_features)

fpr, tpr, thresholds = metrics.roc\_curve(actual, predictions, pos\_label=1)

nbscore = format(metrics.auc(fpr, tpr))

nbscore = float(nbscore)\*100

print("\n")

print("Naive Bayes Accuracy : \n", nbscore,"%")

print(" Completion Speed", round((time.time() - start\_timenb),5))

print()

def datree():

from sklearn import tree

start\_timedt = time.time()

train\_featurestree = vectorizer.fit\_transform(x)

actual1 = y

test\_features1 = vectorizer.transform(x)

dtree = tree.DecisionTreeClassifier()

dtree = dtree.fit(train\_featurestree, [int(r) for r in y])

prediction1 = dtree.predict(test\_features1)

ddd, ttt, thresholds = metrics.roc\_curve(actual1, prediction1, pos\_label=1)

dtreescore = format(metrics.auc(ddd, ttt))

dtreescore = float(dtreescore)\*100

print("Decision tree Accuracy : \n", dtreescore, "%")

print(" Completion Speed", round((time.time() - start\_timedt),5))

print()

def Tsvm():

from sklearn.svm import SVC

start\_timesvm = time.time()

train\_featuressvm = vectorizer.fit\_transform(x)

actual2 = y

test\_features2 = vectorizer.transform(x)

svc = SVC()

svc = svc.fit(train\_featuressvm, [int(r) for r in y])

prediction2 = svc.predict(test\_features2)

sss, vvv, thresholds = metrics.roc\_curve(actual2, prediction2, pos\_label=1)

svc = format(metrics.auc(sss, vvv))

svc = float(svc)\*100

print("Support vector machine Accuracy : \n", svc, "%")

print(" Completion Speed", round((time.time() - start\_timesvm),5))

print()

def knN():

from sklearn.neighbors import KNeighborsClassifier

start\_timekn = time.time()

train\_featureskn = vectorizer.fit\_transform(x)

actual3 = y

test\_features3 = vectorizer.transform(x)

kn = KNeighborsClassifier(n\_neighbors=2)

kn = kn.fit(train\_featureskn, [int(i) for i in y])

prediction3 = kn.predict(test\_features3)

kkk, nnn, thresholds = metrics.roc\_curve(actual3, prediction3, pos\_label=1)

kn = format(metrics.auc(kkk, nnn))

kn = float(kn)\*100

print("Kneighborsclassifier Accuracy : \n", kn, "%")

print(" Completion Speed", round((time.time() - start\_timekn),5))

print()

def RanFo():

from sklearn.ensemble import RandomForestClassifier

start\_timerf = time.time()

train\_featuresrf = vectorizer.fit\_transform(x)

actual4 = y

test\_features4 = vectorizer.transform(x)

rf = RandomForestClassifier(max\_depth=2, random\_state=0)

rf = rf.fit(train\_featuresrf, [int(i) for i in y])

prediction4 = rf.predict(test\_features4)

rrr, fff, thresholds = metrics.roc\_curve(actual4, prediction4, pos\_label=1)

kn = format(metrics.auc(rrr, fff))

kn = float(kn)\*100

print("Random Forest Accuracy : \n", kn, "%")

print(" Completion Speed", round((time.time() - start\_timerf),5))

print()

print()

def runall():

retrieveTweet('data/tweetdata.txt')

retrieveProcessedData('processed\_data/output.xlsx')

def datreeINPUT(inputtweet):

from sklearn import tree

train\_featurestree = vectorizer.fit\_transform(x)

dtree = tree.DecisionTreeClassifier()

dtree = dtree.fit(train\_featurestree, [int(r) for r in y])

inputdtree= vectorizer.transform([inputtweet])

predictt = dtree.predict(inputdtree)

if predictt == 1:

predictt = "Positive"

elif predictt == 0:

predictt = "Neutral"

elif predictt == -1:

predictt = "Negative"

else:

print("Nothing")

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print(predictt)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

runall()

print("\nInput your tweet : ")

inputtweet = input()

datreeINPUT(inputtweet)

* + 1. **Accuracy\_checker.py**

from warnings import simplefilter

simplefilter(action='ignore', category=FutureWarning)

import json

import pandas as pd

import time

import numpy as np

import itertools

import matplotlib.pyplot as plt

from sklearn.metrics import confusion\_matrix

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn import metrics

tweets\_data = []

x = []

y = []

vectorizer = CountVectorizer(stop\_words='english')

def retrieveTweet(data\_url):

tweets\_data\_path = data\_url

tweets\_file = open(tweets\_data\_path, "r")

for line in tweets\_file:

try:

tweet = json.loads(line)

tweets\_data.append(tweet)

except:

continue

def retrieveProcessedData(Pdata\_url):

sent = pd.read\_excel(Pdata\_url)

for i in range(len(tweets\_data)):

if tweets\_data[i]['id']==sent['id'][i]:

x.append(tweets\_data[i]['text'])

y.append(sent['sentiment'][i])

def plot\_confusion\_matrix(cm, classes,

normalize=False,

title='Confusion matrix',

cmap=plt.cm.Blues):

plt.imshow(cm, interpolation='nearest', cmap=cmap)

plt.title(title)

plt.colorbar()

tick\_marks = np.arange(len(classes))

plt.xticks(tick\_marks, classes, rotation=45)

plt.yticks(tick\_marks, classes)

fmt = '.2f' if normalize else 'd'

thresh = cm.max() / 2.

for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

plt.text(j, i, format(cm[i, j], fmt),

horizontalalignment="center",

color="white" if cm[i, j] > thresh else "black")

plt.tight\_layout()

plt.ylabel('True label')

plt.xlabel('Predicted label')

def nbTrain():

from sklearn.naive\_bayes import MultinomialNB

start\_timenb = time.time()

train\_features = vectorizer.fit\_transform(x)

actual = y

nb = MultinomialNB()

nb.fit(train\_features, [int(r) for r in y])

test\_features = vectorizer.transform(x)

predictions = nb.predict(test\_features)

fpr, tpr, thresholds = metrics.roc\_curve(actual, predictions, pos\_label=1)

nbscore = format(metrics.auc(fpr, tpr))

nbscore = float(nbscore)\*100

print("\n")

print("Naive Bayes Accuracy : \n", nbscore,"%")

print(" Completion Speed", round((time.time() - start\_timenb),5))

print()

def datree():

from sklearn import tree

start\_timedt = time.time()

train\_featurestree = vectorizer.fit\_transform(x)

actual1 = y

test\_features1 = vectorizer.transform(x)

dtree = tree.DecisionTreeClassifier()

dtree = dtree.fit(train\_featurestree, [int(r) for r in y])

prediction1 = dtree.predict(test\_features1)

ddd, ttt, thresholds = metrics.roc\_curve(actual1, prediction1, pos\_label=1)

dtreescore = format(metrics.auc(ddd, ttt))

dtreescore = float(dtreescore)\*100

print("Decision tree Accuracy : \n", dtreescore, "%")

print(" Completion Speed", round((time.time() - start\_timedt),5))

print()

def Tsvm():

from sklearn.svm import SVC

start\_timesvm = time.time()

train\_featuressvm = vectorizer.fit\_transform(x)

actual2 = y

test\_features2 = vectorizer.transform(x)

svc = SVC()

svc = svc.fit(train\_featuressvm, [int(r) for r in y])

prediction2 = svc.predict(test\_features2)

sss, vvv, thresholds = metrics.roc\_curve(actual2, prediction2, pos\_label=1)

svc = format(metrics.auc(sss, vvv))

svc = float(svc)\*100

print("Support vector machine Accuracy : \n", svc, "%")

print(" Completion Speed", round((time.time() - start\_timesvm),5))

print()

def knN():

from sklearn.neighbors import KNeighborsClassifier

start\_timekn = time.time()

train\_featureskn = vectorizer.fit\_transform(x)

actual3 = y

test\_features3 = vectorizer.transform(x)

kn = KNeighborsClassifier(n\_neighbors=2)

kn = kn.fit(train\_featureskn, [int(i) for i in y])

prediction3 = kn.predict(test\_features3)

kkk, nnn, thresholds = metrics.roc\_curve(actual3, prediction3, pos\_label=1)

kn = format(metrics.auc(kkk, nnn))

kn = float(kn)\*100

print("Kneighborsclassifier Accuracy : \n", kn, "%")

print(" Completion Speed", round((time.time() - start\_timekn),5))

print()

def RanFo():

from sklearn.ensemble import RandomForestClassifier

start\_timerf = time.time()

train\_featuresrf = vectorizer.fit\_transform(x)

actual4 = y

test\_features4 = vectorizer.transform(x)

rf = RandomForestClassifier(max\_depth=2, random\_state=0)

rf = rf.fit(train\_featuresrf, [int(i) for i in y])

prediction4 = rf.predict(test\_features4)

rrr, fff, thresholds = metrics.roc\_curve(actual4, prediction4, pos\_label=1)

kn = format(metrics.auc(rrr, fff))

kn = float(kn)\*100

print("Random Forest Accuracy : \n", kn, "%")

print(" Completion Speed", round((time.time() - start\_timerf),5))

print()

print()

def runall():

retrieveTweet('data/tweetdata.txt')

retrieveProcessedData('processed\_data/output.xlsx')

nbTrain()

datree()

Tsvm()

knN()

RanFo()

def datreeINPUT(inputtweet):

from sklearn import tree

train\_featurestree = vectorizer.fit\_transform(x)

dtree = tree.DecisionTreeClassifier()

dtree = dtree.fit(train\_featurestree, [int(r) for r in y])

inputdtree= vectorizer.transform([inputtweet])

predictt = dtree.predict(inputdtree)

if predictt == 1:

predictt = "Positive"

elif predictt == 0:

predictt = "Neutral"

elif predictt == -1:

predictt = "Negative"

else:

print("Nothing")

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print(predictt)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

runall()