SENTIMENTAL ANALYSIS ON TWITTER USING KIVY

A PROJECT REPORT

Submitted by

A AJITH	**(201217104007)**
S DELLIGANESH	**(210217104012)**
M MANIKANDAN	**(210217104028)**

in partial fulfillment of the requirements

for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

APOLLO ENGINEERING COLLEGE, CHENNAI

ANNA UNIVERSITY : CHENNAI 600 025

APRIL 2021

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"SENTIMENTAL ANALYSIS ON TWITTER USING KIVY"** is the bonafide work of **"Ajith. A (210217104007) & Delliganesh. S (210217104012) & ManiKandan. M (210217104028)"** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reportedhere in does not form part of any other project report or dissertation on the basis of which degree or award was conferred on an earlier occasion on this or any othercandidate.

Supervisor
Mr. E. Murali M. Tech., (Ph.D).
HOD and Professor,
Department of CSE
Apollo Engineering College
Kanchipuram-602 105

Head of the Department
Mr. E. Murali M. Tech., (Ph.D).
HOD and Professor,
Department of CSE
Apollo Engineering College
Kanchipuram-602 105

Submitted to Project and Viva Examination held

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

We express our profound gratitude to our Chairman, and Secretary of our college, for providing us the necessary facilities to carry out the mini-project work.

We would like to express my sincere thanks to **Dr. M. Ramkumar Prabhu M.E, (Ph.D).,** Principal of our college for their constant support and encouragement towards completion of this project.

We like to express our heartfelt thanks to **Mr. E. Murali M. Tech., (Ph.D).,** Head of the Department, Professor and internal guide, Department of Computer Science and Engineering, **Mrs. R. Sudha M. Tech.,** Project Coordinator for help in completing the project.

We would like to express my heartfelt thanks to the department staffs and family members for their continuous support and encourage us.

ABSTRACT

Twitter is a micro-blogging platform which provides a tremendous amount of data which can be used for various applications of Sentiment Analysis like predictions, reviews, elections, marketing, etc. Sentiment Analysis is a process of extracting information from large amount of data, and classifies them into different classes called sentiments. Python is simple yet powerful, high-level, interpreted and dynamic programming language, which is well known for its functionality of processing natural language data by using NLTK (Natural Language Toolkit). NLTK is a library of python, which provides a base for building programs and classification of data. NLTK also provide graphical demonstration for representing various results or trends. The goal of this thesis is to classify twitter data into sentiments (positive,neutral or negative) by using different supervised machine learning classifiers on data collected for different trending hashtaqs or other identifiers on twitter. We also concluded which classifier gives more accuracy during classification. KivyMd is a collection of Material Design compliant widgets for use with Kivy, a framework for cross-platform, touch-enabled graphical applications. The project's goal is to approximate Google's Material Design spec as close as possible without sacrificing ease of use or application performance. So, It can very useful to provide best GUI to the user.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF ABBREVIATIONS

SVM	-	Support Vector Machine
NB	-	Naïve Bayes
NLP	-	Natural Language Processing
NLTK	-	Natural Language Toolkit
SQL	-	Structured Query Language
MD	-	Material Design
CSV	-	Comma Separated Values
URL	-	Uniform Resource Locator
API	-	Appilcation Programming Interface

CHAPTER 1

INTRODUCTION

1.1 GENERAL

Sentiment Analysis is process of collecting and analyzing data based upon the person feelings, reviews and thoughts. Sentimental analysis often called as opinion mining as it mines the important feature from people opinions. Sentimental Analysis is done by using various machine learning techniques, statistical models and Natural Language Processing (NLP) for the extraction of feature from a large data.

By the help of KivyMd the user gets the best intraction from the program for better user experience. Also the additional security is provided by Login and Signup process before the execution of the Sentiment Analysis.

Sentiment Analysis can be done at document, phrase and sentence level. In document level, summary of the entire document is taken first and then it is analyze whether the sentiment is positive, negative or neutral. In phrase level, analysis of phrases in a sentence is taken in account to check the polarity. In Sentence level, each sentence is classified in a particular class to provide the sentiment.

Sentimental Analysis has various applications. It is used to generate opinions for people of social media by analyzing their feelings or thoughts which they provide in form of text. Sentiment Analysis is domain centered, i.e. results of one domain cannot be applied to other domain. Sentimental Analysis is used in many real life scenarios, to get reviews about any product or movies, to get the financial report of any company, for predictions or marketing.

Twitter is a micro blogging platform where anyone can read or write short form of message which is called tweets. The amount of data accumulated on twitter is very huge. This data is unstructured and written in natural language. Twitter Sentimental Analysis is the process of accessing tweets for a particular topic and predicts the sentiment of these tweets as positive, negative or neutral with the help of naïve bayes machine learning algorithm.

1.1.1 OBJECTIVE OF THE PROJECT

- The objective of this project is to obtain sentiment analysis from twitter data using Python programming.
- This project also increases the accuracy of the result and show them elaborately as possible with the help of Kivy material design framework.
- To make this program as platform independent because kivy application can be easily exported to any other devices like Android, IOS, etc.

1.1.2 EXISTING SYSTEM

In this system, a sentiment analysis application for twitter was conducted on Republic of Indonesia presidential candidates, using the python programming language.

Disadvantages

- Less Accuracy in obtaining results.
- Providing user data in the console was hard for user.
- Inflexible for the user due to no GUI(Graphical User Interface)The classroom must be always under light environment. In dark condition it becomes difficult to detect and recognize faces.

1.1.3 PROPOSED SYSTEM

In this application we can obtain sentiment analysis from twitter data using Python programming with kivy material design framework. We applied Naive bayes algorithm technique in this method to analysis twitter data. The results showed as Percentages of polarities as Positive, Negative, Neutral in the application.

Advantage

- Better GUI(Graphical User Interface) for flexible user interface.
- Kivy application can be easily exported to any other devices like Android, IOS, etc.
- Result shows as elaborately.

1.2 DOMAIN DESCRIPTION

The following are the domain description of the project:

1.2.1 DATA MINING

Data mining is the process of analyzing massive volumes of data to discover business intelligence that helps companies solve problems, mitigate risks, and seize new opportunities. This branch of data science derives its name from the similarities between searching for valuable information in a large database and mining a mountain for ore. Both processes require sifting through tremendous amounts of material to find hidden value.

Data mining can answer business questions that traditionally were too time consuming to resolve manually. Using a range of statistical techniques

to analyze data in different ways, users can identify patterns, trends and relationships they might otherwise miss. They can apply these findings to predict what is likely to happen in the future and take action to influence business outcomes.

Data mining is used in many areas of business and research, including sales and marketing, product development, healthcare, and education. When used correctly, data mining can provide a profound advantage over competitors by enabling you to learn more about customers, develop effective marketing strategies, increase revenue, and decrease costs.

You can use data mining to solve almost any business problem that involves data, including:

- Increasing revenue.
- Understanding customer segments and preferences.
- Acquiring new customers.
- Improving cross-selling and up-selling.
- Retaining customers and increasing loyalty.
- Monitoring operational performance.

1.2.2 SENTIMENT ANALYSIS

Sentiment Analysis is process of collecting and analyzing data based upon the person feelings, reviews and thoughts. Sentimental analysis often called as opinion mining as it mines the important feature from people opinions. Sentimental Analysis is done by using various machine learning techniques, statistical models and Natural Language Processing (NLP) for the extraction of feature from a large data. Sentiment Analysis can be done at document, phrase and sentence level. In document level, summary of the entire document is

taken first and then it is analyze whether the sentiment is positive, negative or neutral. In phrase level, analysis of phrases in a sentence is taken in account to check the polarity. In Sentence level, each sentence is classified in a particular class to provide the sentiment.

Sentimental Analysis has various applications. It is used to generate opinions for people of social media by analyzing their feelings or thoughts which they provide in form of text. Sentiment Analysis is domain centered, i.e. results of one domain cannot be applied to other domain. Sentimental Analysis is used in many real life scenarios, to get reviews about any product or movies, to get the financial report of any company, for predictions or marketing.

CHAPTER 2

LITERATURE SURVEY

2.1 "Sentiment Analysis of Twitter Data" by Sahar A.El_Rahman , eddah Alhumaidi AlOtaibi

This paper focus on mining tweets written in English. We are interested in seeing who people think is better Mcdonalds or KFC in terms of how good/bad reviews are. Analyzing people's opinions and what they think about a product from their tweets on social media could be a valuable thing for any business. In our project, we extracted tweets from Twitter using R language. R is a programming language used for statistical computing and machine learning algorithms. In order to extract tweets from Twitter, Twitter API were used to create Twitter application and get authorization. In Rstudio which is an environment and graphical user interface for R, we installed necessary packages and libraries. Some of the packages are (TwitteR, rtweet, R0Auth). By using twitter package you can extract tweets up to 4000 only .On the other hand, there is retweet package which is way better than twitter because it allows you to extract up to 20,000 tweets and in a suitable format.

2.2 L. Jiang, M. Yu, M. Zhou, X. Liu, T. Zhao et al

Twitter sentiment analysis was growing at faster rate as amount of data is increasing. They created a system which focuses on target dependent classification. It is based on Twitter in which a query is given first; they classify the tweets as positive, negative or neutral sentiments with respect to

that query that contain sentiment as positive, negative or neutral. In their research, query sentiment serves as target. The target- independent strategy is always adopted to solve these problems with the help of state- of-the-art approaches, which may sometime assign immaterial sentiments to target. Also, when state-of-the-art approaches are used for classification they only take tweet into consideration. These approaches ignore related tweet, as they classify based on current tweet. However, because tweets have property to be short and mostly ambiguous, considering current tweet only for sentiment analysis is not enough. They propose a system to improve target-dependent Twitter sentiment classification by:

1. Integrating target-dependent features, and
2. Taking related tweets into consideration.

According to their experimental results, these new advancement highly improves the efficiency and performance of target-dependent sentiment classification.

2.3 C. Tan, L. Lee, J. Tang, L. Jiang, M. Zhou, P. Li, et al

They show that information that can be used to improve user-level sentiment analysis. There base of research is social relationships, i.e. users that are connected in any social platform will somehow hold similar opinions, thoughts; therefore, relationship information can supplement what they extract from user's viewpoint. They use Twitter as there source of experimental data and they use semi-supervised machine learning framework to carry out analysis. They propose systems that are persuade either from the network of Twitter followers or from the network formed by users in Twitter in which users referring to each other using "@username". According to them, these semi-supervised learning results shows that by including this social- network

information leads to statistically significant improvement in performance of sentiment analysis classification over the performance based on the approach of SVM (Support Vector Machines) that have only access to textual features.

2.4 A. Pak, P. Paroubek et al

Micro-blogging nowadays has become very popular communication platform among users in social network. Billions of tweets share every year among millions of users, in which they share opinions, feelings on different aspects of daily life. Thus micro- blogging websites like Twitter, Friendfeed, Tumbler, etc. are rich sources of data for feature extraction and sentiment analysis. They also use Twitter one of the most popular micro-blogging website, for the implementation of sentiment analysis. They automatically collect a corpus (database) for training classifier to carry out sentiment analysis and opinion mining.

2.5 B. Sun, V. Ng, et al

Many efforts have been done to gather information from social networks to perform sentiment analysis on internet users. Their aim is to show how sentimental analysis influences from social network posts and they also compare the result on various topics on different social-media platforms. Large amount of data is generated every day, people are also very curious in finding other similar people among them. Many researchers' measures the influence of any post through the number of likes and replies it received but they are not sure whether the influence is positive or negative on other post. In their research some questions are raised and new methodologies are prepared for sentimental influence of post.

CHAPTER 3

LANGUAGE DESCRIPTION

3.1 PYTHON

Python is a high level, dynamic programming language which is used for this thesis. Python3.4 version was used as it is a mature, versatile and robust programming language. It is an interpreted language which makes the testing and debugging extremely quickly as there is no compilation step. There are extensive open source libraries available for this version of python and a large community of users.

Python is simple yet powerful, interpreted and dynamic programming language, which is well known for its functionality of processing natural language data, i.e. spoken English using NLTK. Other high level programming languages such as 'R' and 'Matlab' were considered because they have many benefits such as ease of use but they do not offer the same flexibility and freedom that Python can deliver.

ADVANTAGES

- Interpreted Language
- Dynamically Typed.
- Free and Open-Source.
- Vast Libraries Support.
- Portability.

3.2 NLTK

Natural Language Toolkit (NLTK) is library in Python, which provides a base for building programs and classification of data. NLTK is a collection of

resources for Python that can be used for text processing, classification, tagging and tokenization. This toolbox plays a key role in transforming the text data in the tweets into a format that can be used to extract sentiment from them.

NLTK provides various functions which are used in pre-processing of data so that data available from twitter become fit for mining and extracting features. NLTK support various machine learning algorithms which are used for training classifier and to calculate the accuracy of different classifier.

In our thesis we use Python as our base programming language which is used for writing code snippets. NLTK is a library of Python which plays a very important role in converting natural language text to a sentiment either positive or negative. NLTK also provides different sets of data which are used for training classifiers. These datasets are structured and stored in library of NLTK, which can be accessed easily with the help of Python.

3.3 KIVY

Kivy is a free and open source Python framework for developing mobile apps and other multitouch application software with a natural user interface. It is distributed under the terms of the MIT License, and can run on Android, iOS, GNU/Linux, macOS, and Windows. It can natively use most inputs, protocols and devices including WM_Touch, WM_Pen, Mac OS X Trackpad and Magic Mouse, Mtdev, Linux Kernel HID, TUIO. A multi-touch mouse simulator is included.

Kivy is 100% free to use, under an MIT license (starting from 1.7.2) and LGPL 3 for the previous versions. The toolkit is professionally developed, backed and used. You can use it in a commercial product. The framework is stable and has a well documented API, plus a programming guide to help you get started. The graphics engine is built over OpenGL ES 2, using a modern

and fast graphics pipeline. The toolkit comes with more than 20 widgets, all highly extensible. Many parts are written in C using Python, and tested with regression tests.

3.4 TWITTER API (APPLICATION PROGRAMMING INTERFACE)

Twitter allows users to collect tweets with the help of Twitter API(Application Programming Interface). Twitter provides two kinds of APIs: REST API and Streaming API. The differences between these are: REST APIs support connections for short time interval and only limited data can be collected at a time, whereas Streaming API provides tweets in real-time and connection for long time. We use Streaming API for our analysis. For collecting large amount of tweets we need long-lived connection and no limit data rate.

3.5 Supervised Machine learning Classifiers

Supervised machine learning is a technique whose task is to deduce a function from tagged training samples. The training samples for supervised learning consist of large set of examples for a particular topic. In supervised learning, every example training data comes in a pair of input (vector quantity) and output value (desired result). These algorithms analyze data and generate an output function, which is used to mapped new data sets to respective classes. The machine learning classifiers which we are going to use to build our classifier is Naïve-Bayes Classifier

3.5.1 Naïve-Bayes (NB) Classifier

Naïve-Bayes classifiers are probabilistic classifiers which come under

machine learning techniques. These classifiers are based on applying Bayes' theorem with strong (naïve) assumption of independence between each pair of features. Let us assume, there is a dependent vector from x1 to xn, and a class variable 'y'. Therefore, according to Bayes' :

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)}$$

Now according to assumption of independence

$$P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y),$$

For every 'i', this function become

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

In this $P(x_1, \ldots, x_n)$ on given input is constant, hence we can apply classification rule as:

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

And for estimating we can use MAP (Maximum A Posterior) estimation $P(y)$ and $P(x_i \mid y)$; the $P(y)$ of class 'y' in training sample is relative frequency.

CHAPTER 4

SYSTEM SPECFICATION

4.1 HARDWARE REQUIREMENTS

- RAM : 4 GB
- MEMORY : 250 MB
- PROCESSOR : INTEL i3
- KEYBOARD
- MOUSE

4.2 SOFTWARE REQUIREMENTS

- OPERATING SYSTEM : WINDOWS 10
- IDE : PYTHON
- CODING LANGUAGE : PYTHON 3.8.2
- LIBRAY PACKAGES :
  - TWEEPY
  - TEXTBLOB
  - KIVY
  - KIVYMD

CHAPTER 5

UML DIAGRAM

5.1 USE CASE DIAGRAM:

Use case diagram is a behavioural UML diagram type and frequently used to analyse various systems. They enable you to visualize the different types of roles in a system and how those roles interact with the system. Especially useful when presenting to managers or stakeholders. You can highlight the roles that interact with the system and the functionality provided by the system without going deep into inner workings of the system.

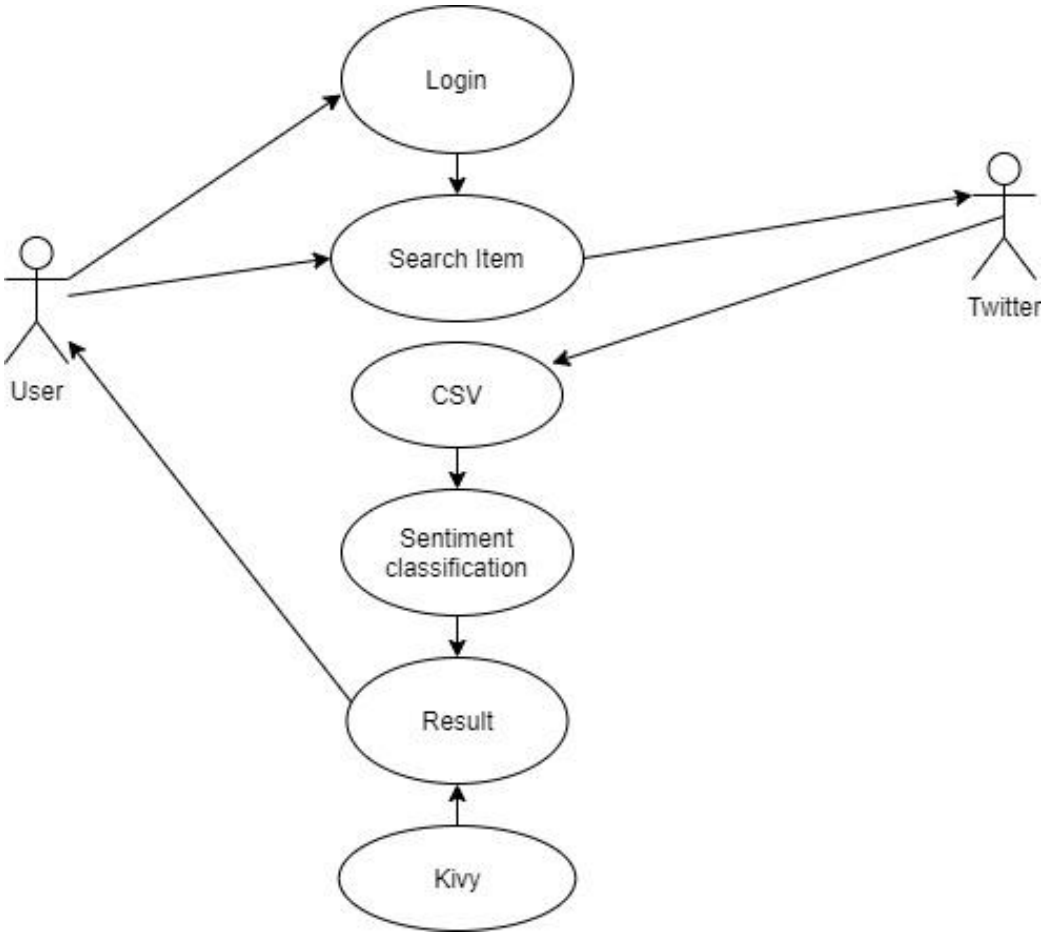


Fig.no 5.1 USE CASE DIAGRAM

5.2 DATA FLOW DIAGRAM:

A data-flow diagram is a way of representing a flow of data through

a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart. There are several notations for displaying data-flow diagrams. The notation presented above was described in 1979 by Tom DeMarco as part of Structured Analysis.For each data flow, at least one of the endpoints (source and / or destination) must exist in a process. The refined representation of a process can be done in another data-flow diagram, which subdivides this process into sub-processes. The data-flow diagram is part of the structured-analysis modeling tools. When using UML, the activity diagram typically takes over the role of the data-flow diagram. A special form of data-flow plan is a site-oriented data-flow plan. Data-flow diagrams can be regarded as inverted Petri nets, because places in such networks correspond to the semantics of data memories.

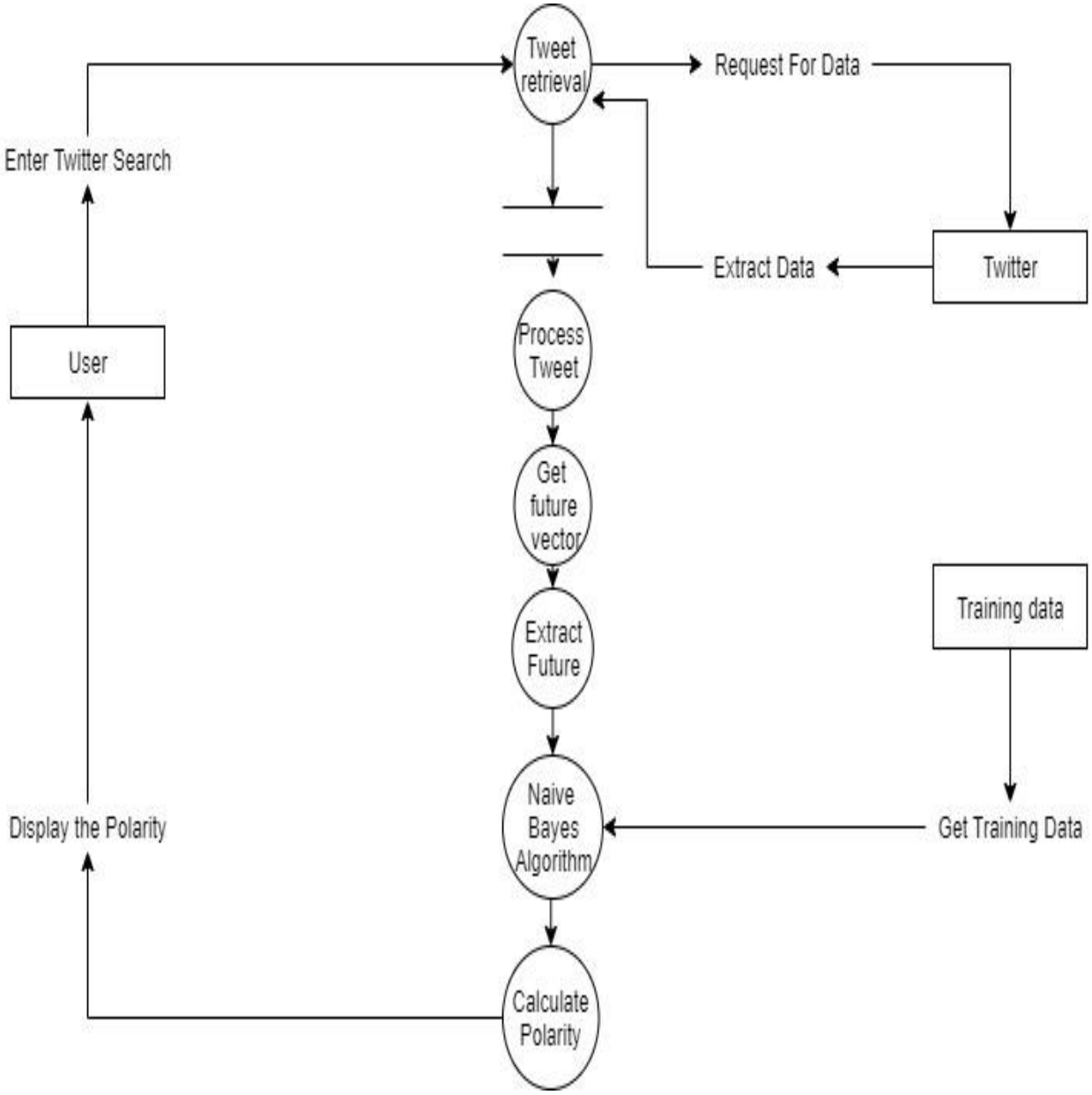


Fig.no 5.2 DATA FLOW DIAGRAM

5.3 CONTROL FLOW DIAGRAM:

A control-flow diagram can consist of a subdivision to show sequential steps, with if-then-else conditions, repetition, and/or case conditions. Suitably annotated geometrical figures are used to represent operations, data, or equipment, and arrows are used to indicate the sequential flow from one to another. A flow diagram can be developed for the process control system for each critical activity.

The application determines if the sensor information is within the predetermined (or calculated) data parameters and constraints. The results of this comparison, which controls the critical component. This feedback may control the component electronically or may indicate the need for a manual action. This closed-cycle process has many checks and balances to ensure that it stays safe. It may be fully computer controlled and automated, or it may be a hybrid in which only the sensor is automated and the action requires manual intervention. Further, some process control systems may use prior generations of hardware and software, while others are state of the art.

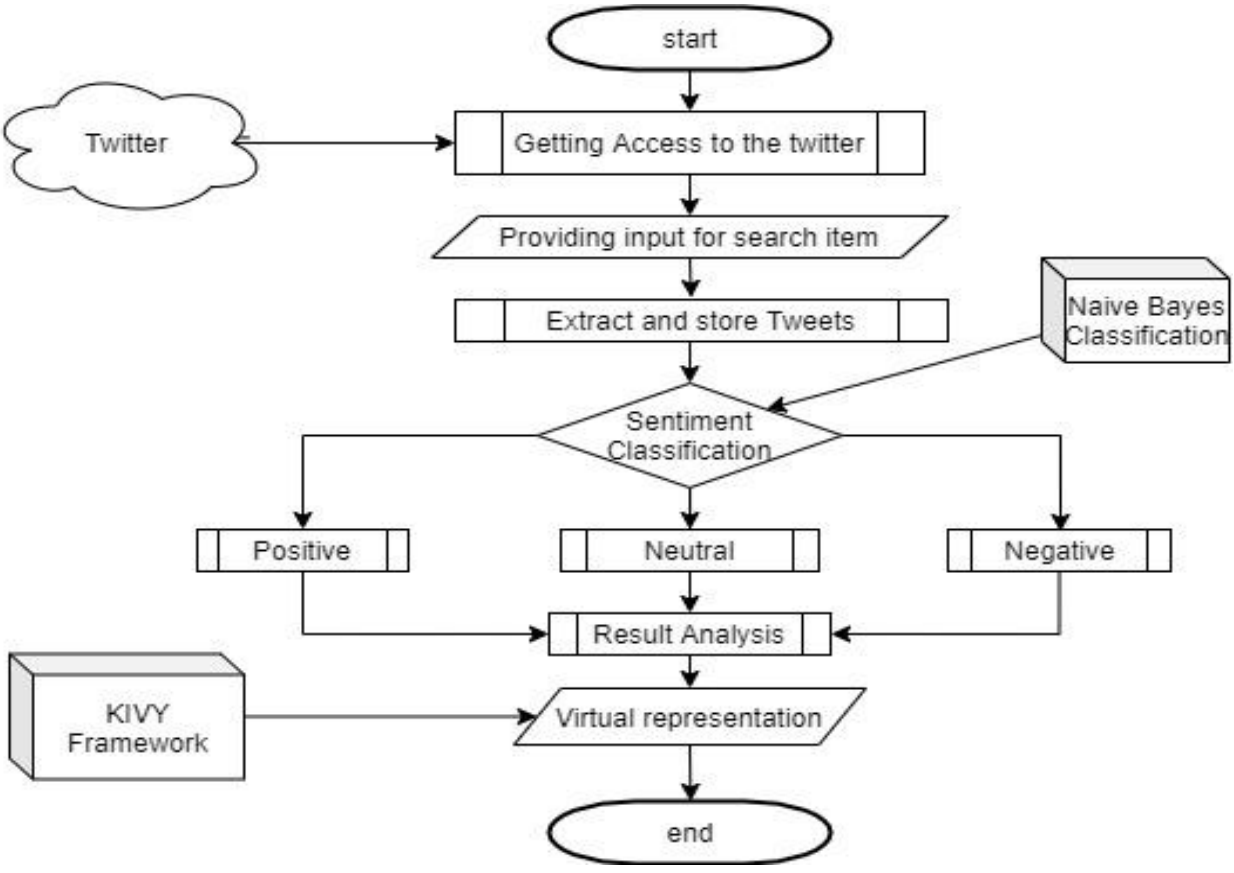


Fig.no 5.3 Control Flow Diagram

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 SYSTEM ARCHITECTURE

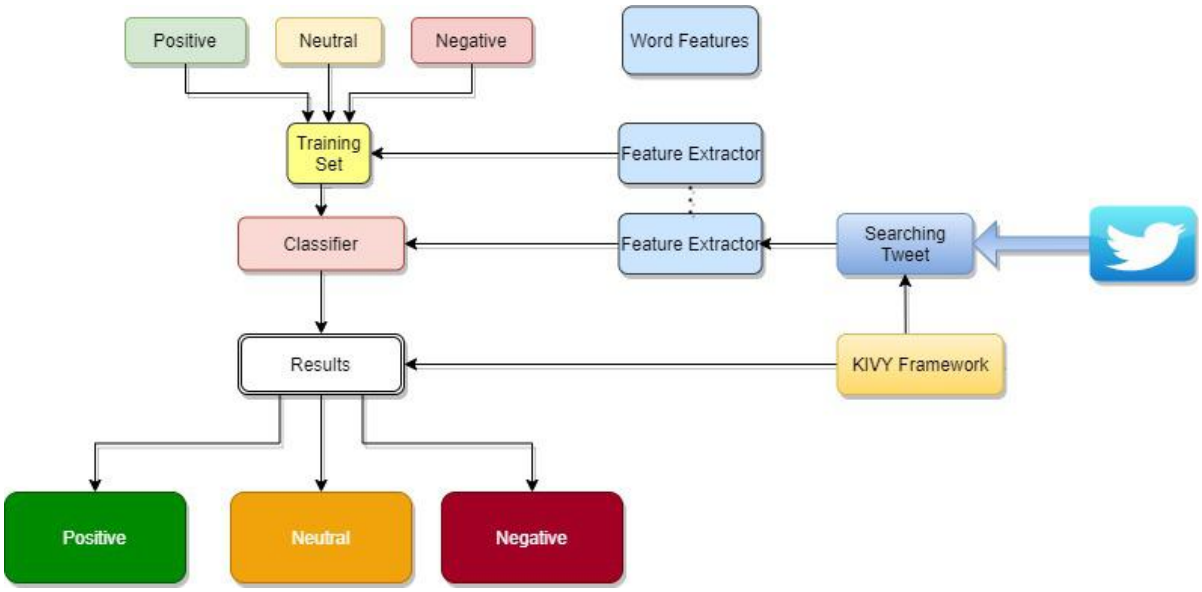


Fig.No 6.1 System Architecture

6.2 MODULES

The project is mainly divided into five modules they are

1. Authentication
2. Data collecting and processing
3. Sentiments of data
4. Results on the application

6.2.1 AUTHENTICATION

It is the process of validating user details in order obtain security in the application. The user first enter to the Login page when the application is start and

user needs to enter email address and password and press ‘Login’ to proceed. If user cannot have registered email and password then press ‘create account’ to create new user account. In create account page username,email address and password can be entered and press ‘create’. Then the user details can be stored in the ‘users.text’ file. So it can be again getted for authentication purpose when user enters details on login page.

6.2.2 DATA COLLECTION AND PROCESSING

First we are going to stream tweets in our build classifier with the help of Tweepy library in python. Then we pre-process these tweets, so that they can be fit for mining and feature extraction. Other data which we collected for this thesis is training data. This data is used to train the classifier which we are going to build. To collect this data we use NLTK library of Python. NLTK consists of corpora, which is very large and consists of structured set of text files which are used to perform analysis. In these corpora there are various types of text files like quotes, reviews, chat, history, etc.

6.2.3 SENTIMENTS OF DATA

Sentiment can be classified using NLP tool TextBlob library to analyse the polarity, as well as the subjectivity of a tweet on the specified subject or topic. he codes which we will specify will provide us with polarity and subjectivity. Polarity is the positivity or negativity of the text; it returns a float value in the range of “-1.0 to 1.0”, where ‘0.0’ indicates neutral, ‘+1’ indicates a very positive sentiment and ‘-1’ represents a very negative sentiment. Subjectivity is the text on the basis that how much of it is an opinion vs how factual it is. where ‘0.0’ is very objective and ‘1.0’ is very subjective. With the help of the NLP sentiments can be analysed and classified for the opinions.

6.2.4 RESULTS ON THE APPLICATION

The sentiments of the searched word on twitter can be found using NLP can be process to produce percentage of sentiments in each catagory as positive, neutral and negative. Then the results can be shown in the application build with the Kivy material designing framework. Then results can be shown as neatly and elaborately for best user interface.

CHAPTER 7

TESTING

7.1 SOFTWARE TESTING

Testing is the major quality control measure employed for software development. Its basic function is to detect errors in the software. During requirement analysis and design, the output is document that is usually textual and non-textual.

After the coding phase, computer programs are available that can be executed for testing purpose. This implies that testing has to uncover errors introduced during coding phases.

Thus, the goal of testing is to cover requirement, design, or coding errors in the program. The starting point of testing is unit testing. In this a module is tested separately and often performed by the programmer himself simultaneously while coding the module.

7.2 TESTING TECHNIQUES

7.2.1 Unit Testing

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements. Each module can be tested using the following two strategies:

7.2.1.1 Black Box Testing

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been uses to find errors in the following categories:

- Incorrect or missing functions
- Interface errors
- Errors in data structure or external database access

In this testing only the output is checked for correctness.

7.2.1.2 White Box Testing

In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases:

- Guarantee that all independent paths have been executed
- Execute all logical decisions on their true and false sides

7.2.2 Integration Testing

Integration testing ensures that software and subsystems work together as a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together.

7.2.3 System Testing

Involves in-house testing of the entire system before delivery to the user. Its aim is to satisfy the user the system meets all requirements of the client's specifications

7.2.4 Acceptance Testing

It is a pre-delivery testing in which entire system is tested at client's site on real world data to find errors.

7.2.5 Validation

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed.

CHAPTER 8
CONCLUSION AND FUTURE ENHANCEMENT

8.1 CONCLUSON

Sentiment analysis is used to identifying people's opinion, attitude and emotional states. The views of the people can be positive or negative. Commonly, parts of speech are used as feature to extract the sentiment of the text. An adjective plays a crucial role in identifying sentiment from parts of speech. Sometimes words having adjective and adverb are used together then it is difficult to identify sentiment and opinion. To do the sentiment analysis of tweets, the proposed system first extracts the twitter posts from twitter by user. The system can also computes the frequency of each term in tweet. Using machine learning supervised approach help to obtain the results. Twitter is large source of data, which make it more attractive for performing sentiment analysis.

8.2 FUTURE ENANCEMENT

Some of future scopes that can be included in our research work are the use of parser can be embedded into system to improve results. A web-based application can be made for our work in future. We can improve our system that can deal with sentences of multiple meanings. We can also increase the classification categories so that we can get better results. We can start work on multi languages like Hindi, Spanish, and Arabic to provide sentiment analysis to more local. We look forward to implement sentiment analysis for Facebook and Instagram also in this application. To build this application as better layout and graphical user interfaces more interactively.

CHAPTER 9

APPENDIX

9.1 CODING

9.1.1 “main.py”

```
from kivymd.app import MDApp
from kivy.lang import Builder
from kivy.uix.screenmanager import ScreenManager, Screen
from kivy.properties import ObjectProperty
from kivy.uix.popup import Popup
from kivy.uix.label import Label
from kivymd.icon_definitions import md_icons
from database import DataBase
from sentiment import SentimentAnalysis
class CreateAccountWindow(Screen,MDApp):
    namee = ObjectProperty(None)
    email = ObjectProperty(None)
    password = ObjectProperty(None)


    def submit(self):
        if self.namee.text != "" and self.email.text != "" and
self.email.text.count("@") == 1 and self.email.text.count(".") > 0:
            if self.password != "":
                db.add_user(self.email.text, self.password.text, self.namee.text)
                self.reset()
```

```
            sm.current = "login"
        else:
            invalidForm()
    else:
        invalidForm()
def login(self):
    self.reset()
    sm.current = "login"
def reset(self):
    self.email.text = ""
    self.password.text = ""
    self.namee.text = ""


class LoginWindow(Screen,MDApp):
    email = ObjectProperty(None)
    password = ObjectProperty(None)
    def loginBtn(self):
        if db.validate(self.email.text, self.password.text):
            MainWindow.current = self.email.text
            self.reset()
            sm.current = "main"
        else:
            invalidLogin()

    def createBtn(self):
```

```
        self.reset()
        sm.current = "create"
    def reset(self):
        self.email.text = ""
        self.password.text = ""

class MainWindow(Screen,MDApp):
    searchItem = ObjectProperty(None)
    noItems = ObjectProperty(None)
    current = ""
    def savegetdata(self):
        pass



    def on_exit(self, *args):
        pass

    def logout(self):
        sm.current = "login"
        pass

    def gosenti(self):
        self.manager.sw.search_Item.text = self.ids.searchItem.text
        sm.current = "senti"
        pass
```

```
class SentiWindow(Screen,MDApp):
    searchI = ObjectProperty(None)
    search_Item = ObjectProperty(MainWindow)
    a = ObjectProperty(SentimentAnalysis)
    b = ObjectProperty(SentimentAnalysis)
    c = ObjectProperty(SentimentAnalysis)
    d = ObjectProperty(SentimentAnalysis)
    e = ObjectProperty(SentimentAnalysis)
    a1 = ObjectProperty(None)
    b1 = ObjectProperty(None)
    c1 = ObjectProperty(None)
    d1 = ObjectProperty(None)
    e1 = ObjectProperty(None)
    current = ""

    def on_enter(self, *args):
        self.searchI.text = "Search Item : " + self.search_Item.text
        self.a, self.b, self.c, self.d = sa.DownloadData(self.search_Item.text)
        self.a1.text = "Overall Sentiment : " + self.a
        self.b1.text = "Percentage of Positive tweets : " + self.b
        self.c1.text = "Percentage of Negative tweets : " + self.c
        self.d1.text = "Percentage of Neutral tweets : " + self.d

    def gomain(self):
```

```
        sm.current = "main"
        pass

    def logout(self):
        sm.current = "login"
        pass

class WindowManager(ScreenManager):
    mw = MainWindow()
    sw = SentiWindow()
    pass

def invalidLogin():
    pop = Popup(title='Invalid Login',
                content=Label(text='Invalid username or password.'),
                size_hint=(None, None), size=(400, 400))
    pop.open()

def invalidForm():
    pop = Popup(title='Invalid Form',
                content=Label(text='Please fill in all inputs with valid
information.'),
                size_hint=(None, None), size=(400, 400))
    pop.open()

kv = Builder.load_file("my.kv")
```

```
sa = SentimentAnalysis()
sm = WindowManager()
db = DataBase("users.txt")

screens = [LoginWindow(name="login"),
CreateAccountWindow(name="create"),MainWindow(name="main"),
SentiWindow(name="senti")]
for screen in screens:
    sm.add_widget(screen)
sm.current = "login"

class MyMainApp(MDApp):
    def build(self):
        self.theme_cls.theme_style = "Light"
        self.theme_cls.primary_palette = "Indigo"
        self.theme_cls.primary_hue = "500"
        return sm

if __name__ == "__main__":
    MyMainApp().run()
```

9.1.2 “sentiment.py”

```
import sys, tweepy, csv, re
from textblob import TextBlob
class SentimentAnalysis:
    def __init__(self):
```

```
        self.tweets = []
        self.tweetText = []
    def DownloadData(self, searchTerm):
        # authenticating
        consumerKey = 'M5N69WvF5voLOTqlpMO10RE5e'
        consumerSecret =
'u8ZDalI3qCAlTCAJ0Fj6ynNC8D9zV43AMexXmpzF5Sz2u1Ne5S'
        accessToken = '1230104751250927616-
CVlNYnDRXa76al9W0FPklU5JSBXaZu'
        accessTokenSecret =
'E0IYZsFupOquVAbmRLugEIYJXxFDoSqAeslMViYDqsTAX'
        auth = tweepy.OAuthHandler(consumerKey, consumerSecret)
        auth.set_access_token(accessToken, accessTokenSecret)
        api = tweepy.API(auth)
        # input for term to be searched and how many tweets to search
        NoOfTerms = 1000
        # searching for tweets
        self.tweets = tweepy.Cursor(api.search, q=searchTerm, lang =
"en").items(NoOfTerms)

        # Open/create a file to append data to
        csvFile = open('result.csv', 'a')
        # Use csv writer
        csvWriter = csv.writer(csvFile)
        # creating some variables to store info
        polarity = 0
        positive = 0
```

```
        negative = 0
        neutral = 0


        # iterating through tweets fetched
        for tweet in self.tweets:
            #Append to temp so that we can store in csv later. I use encode UTF-
8
            self.tweetText.append(self.cleanTweet(tweet.text).encode('utf-8'))
            # print (tweet.text.translate(non_bmp_map))    #print tweet's text
            analysis = TextBlob(tweet.text)
            # print(analysis.sentiment)  # print tweet's polarity
            polarity += analysis.sentiment.polarity  # adding up polarities to find
the average later


            if (analysis.sentiment.polarity == 0):  # adding reaction of how
people are reacting to find average later
                neutral += 1
            elif (analysis.sentiment.polarity > 0):
                positive += 1
            elif (analysis.sentiment.polarity < 0):
                negative += 1


        # Write to csv and close csv file
        csvWriter.writerow(self.tweetText)
        csvFile.close()
```

```
        # finding average of how people are reacting
        positive = self.percentage(positive, NoOfTerms)
        negative = self.percentage(negative, NoOfTerms)
        neutral = self.percentage(neutral, NoOfTerms)

        # finding average reaction
        polarity = polarity / NoOfTerms
        po = 0

        # printing out data
        if (polarity == 0):
            po = "Neutral"
        elif (polarity > 0):
            po = "Positive"
        elif (polarity < 0):
            po = "Negative"

        popo = str(positive)
        pong = str(negative)
        ponu = str(neutral)
        return po, popo, pong, ponu

    def cleanTweet(self, tweet):
        # Remove Links, Special Characters etc from tweet
        return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t]) | (\w +:\ / \ / \S
+)", " ", tweet).split())
```

```
    # function to calculate percentage
    def percentage(self, part, whole):
        temp = 100 * float(part) / float(whole)
        return format(temp, '.2f')
```

9.1.3 “database.py”

```
import datetime
class DataBase:
    def __init__(self, filename):
        self.filename = filename
        self.users = None
        self.file = None
        self.load()

    def load(self):
        self.file = open(self.filename, "r")
        self.users = {}
        for line in self.file:
            email, password, name, created = line.strip().split(";")
            self.users[email] = (password, name, created)
        self.file.close()

    def get_user(self, email):
        if email in self.users:
```

```
            return self.users[email]
        else:
            return -1


    def add_user(self, email, password, name):
        if email.strip() not in self.users:
            self.users[email.strip()] = (password.strip(), name.strip(),
DataBase.get_date())
            self.save()
            return 1
        else:
            print("Email exists already")
            return -1


    def validate(self, email, password):
        if self.get_user(email) != -1:
            return self.users[email][0] == password
        else:
            return False


    def save(self):
        with open(self.filename, "w") as f:
            for user in self.users:
                f.write(user + ";" + self.users[user][0] + ";" + self.users[user][1] +
";" + self.users[user][2] + "\n")
    @staticmethod
```

```
    def get_date():
        return str(datetime.datetime.now()).split(" ")[0]
```

9.1.4 “my.kv”

```
<CreateAccountWindow>:
    name: "create"
    namee: namee
    email: email
    password: passw

    FloatLayout:
        cols:1
        canvas:
            Rectangle:
                pos: self.pos
                size: self.size

        FloatLayout:
            canvas:
                Color:
                    rgba: 0, 0, 1, .5
                Rectangle:
                    pos: self.pos
                    size: self.size
```

```
MDLabel:
    text: "Create an Account"
    size_hint: 0.3, 0.2
    pos_hint: {"x":0.1, "top":1}
    font_size: (root.width**2 + root.height**2) / 14**4

MDLabel:
    size_hint: 0.5,0.12
    pos_hint: {"x":0.2, "top":0.8}
    text: "Name: "
    font_size: (root.width**2 + root.height**2) / 14**4

MDTextField:
    mode: "fill"
    fill_color: 0, 0, 0.2, .2
    line_color_focus: 0, 0, 0.2, 1
    color_mode: 'custom'
    pos_hint: {"x":0.5, "top":0.8}
    icon_right: "account-edit"
    size_hint: 0.4, 0.12
    id: namee
    hint_text: "Enter New User Name"
    multiline: False
    font_size: (root.width**2 + root.height**2) / 14**4
```

```
MDLabel:
    size_hint: 0.5,0.12
    pos_hint: {"x":0.2, "top":0.8-0.13}
    text: "Email: "
    font_size: (root.width**2 + root.height**2) / 14**4

MDTextField:
    mode: "fill"
    fill_color: 0, 0, 0.2, .2
    line_color_focus: 0, 0, 0.2, 1
    color_mode: 'custom'
    pos_hint: {"x":0.5, "top":0.8-0.13}
    icon_right: "email"
    size_hint: 0.4, 0.12
    id: email
    hint_text: "@gmail.com"
    multiline: False
    font_size: (root.width**2 + root.height**2) / 14**4

MDLabel:
    size_hint: 0.5,0.12
    pos_hint: {"x":0.2, "top":0.8-0.13*2}
    text: "Password: "
    font_size: (root.width**2 + root.height**2) / 14**4
```

```
    MDTextField:
        mode: "fill"
        fill_color: 0, 0, 0.2, .2
        line_color_focus: 0, 0, 0.2, 1
        color_mode: 'custom'
        hint_text: "Enter the new password"
        icon_right: "lock"
        pos_hint: {"x":0.5, "top":0.8-0.13*2}
        size_hint: 0.4, 0.12
        id: passw
        multiline: False
        password: True
        font_size: (root.width**2 + root.height**2) / 14**4

MDFillRoundFlatIconButton:
    icon: "account-arrow-left"
    text_color: 1, 1, 1, 1
    pos_hint:{"x":0.3,"y":0.25}
    size_hint: 0.4, 0.1
    font_size: (root.width**2 + root.height**2) / 17**4
    text: "Already have an Account? Log In"
    on_release:
        root.manager.transition.direction = "left"
        root.login()
```

```
        MDFillRoundFlatIconButton:
            icon: "account-check"
            text_color: 1, 1, 1, 1
            pos_hint:{"x":0.2,"y":0.05}
            size_hint: 0.6, 0.15
            text: "Submit"
            font_size: (root.width**2 + root.height**2) / 14**4
            on_release:
                root.manager.transition.direction = "left"
                root.submit()


<LoginWindow>:
    name: "login"

    email: email
    password: password

    FloatLayout:
        canvas:
            Color:
                rgba: 0, 0, 1, .5
            Rectangle:
                pos: self.pos
                size: self.size
```

```
MDLabel:
    text:"Email: "
    font_size: (root.width**2 + root.height**2) / 13**4
    pos_hint: {"x":0.1, "top":0.9}
    size_hint: 0.35, 0.15

MDTextField:
    id: email
    mode: "fill"
    line_color_focus: 0, 0, 0.2, 1
    fill_color: 0, 0, 1, .2
    color_mode: 'custom'
    icon_right: "email"
    hint_text: "example@gmail.com"
    font_size: (root.width**2 + root.height**2) / 13**4
    multiline: False
    pos_hint: {"x": 0.45 , "top":0.9}
    size_hint: 0.4, 0.15

MDLabel:
    text:"Password: "
    font_size: (root.width**2 + root.height**2) / 13**4
    pos_hint: {"x":0.1, "top":0.7}
    size_hint: 0.35, 0.15
```

```
MDTextField:
    mode: "fill"
    fill_color: 0, 0, 1, .2
    line_color_focus: 0, 0, 0.2, 1
    color_mode: 'custom'
    id: password
    icon_right: "lock"
    hint_text: "Enter the Password"
    font_size: (root.width**2 + root.height**2) / 13**4
    multiline: False
    password: True
    pos_hint: {"x": 0.45, "top":0.7}
    size_hint: 0.4, 0.15

MDFillRoundFlatIconButton:
    icon: "login"
    text_color : 1,1,1,1
    pos_hint:{"x":0.2,"y":0.05}
    size_hint: 0.6, 0.2
    font_size: (root.width**2 + root.height**2) / 13**4
    text: "Login"
    on_release:
        root.manager.transition.direction = "up"
        root.loginBtn()
```

```
        MDFillRoundFlatIconButton:
            icon: "account-plus"
            text_color: 1, 1, 1, 1
            pos_hint:{"x":0.3,"y":0.3}
            size_hint: 0.4, 0.1
            font_size: (root.width**2 + root.height**2) / 17**4
            text: "Don't have an Account? Create One"
            on_release:
                root.manager.transition.direction = "right"
                root.createBtn()


<MainWindow>:
    searchItem : searchItem
    noItems : noItems
    search_Item : searchItem.text
    no_Items : noItems.text
    FloatLayout:
        canvas.before:
            Color:
                rgba: 0, 0, 1, .5
            Rectangle:
                pos: self.pos
                size: self.size
```

```
MDLabel:
    text: "Enter the search term: "
    font_size: (root.width ** 2 + root.height ** 2) / 14 ** 4
    pos_hint: {"x": 0.1, "top": 0.9}
    size_hint: 0.35, 0.15

MDTextField:
    mode: "fill"
    fill_color: 0, 0, 1, .2
    line_color_focus: 0, 0, 0.2, 1
    color_mode: 'custom'
    id: searchItem
    icon_right: "twitter-box"
    font_size: (root.width ** 2 + root.height ** 2) / 13 ** 4
    multiline: False
    pos_hint: {"x": 0.45, "top": 0.9}
    size_hint: 0.4, 0.15

MDLabel:
    text: "Number of items "
    font_size: (root.width ** 2 + root.height ** 2) / 15 ** 4
    pos_hint: {"x": 0.1, "top": 0.7}
    size_hint: 0.35, 0.15

MDTextField:
```

```
    mode: "fill"
    fill_color: 0, 0, 1, .2
    line_color_focus: 0, 0, 0.2, 1
    color_mode: 'custom'
    id: noItems
    input_filter: "int"
    font_size: (root.width ** 2 + root.height ** 2) / 13 ** 4
    multiline: False
    pos_hint: {"x": 0.45, "top": 0.7}
    size_hint: 0.4, 0.15

MDFillRoundFlatIconButton:
    icon: "logout"
    pos_hint: {"x": 0.3, "y": 0.02}
    size_hint: 0.3, 0.1
    font_size: (root.width ** 2 + root.height ** 2) / 13 ** 4
    text: "logout"
    on_release:
        root.logout()
        root.manager.transition.direction = "down"

MDFillRoundFlatIconButton:
    icon: "cloud-search"
    pos_hint: {"x": 0.2, "y": 0.3}
    size_hint: 0.6, 0.2
```

```
            font_size: (root.width ** 2 + root.height ** 2) / 13 ** 4
            text: "Submit"

            on_release:
                root.manager.transition.direction = "right"
                root.gosenti()


<SentiWindow>

    searchI : searchI
    a1 : a1
    b1 : b1
    c1 : c1
    d1 : d1

    FloatLayout:
        canvas.before:
            Color:
                rgba: 0, 0, 1, .5
            Rectangle:
                pos: self.pos
                size: self.size
        rows:6
```

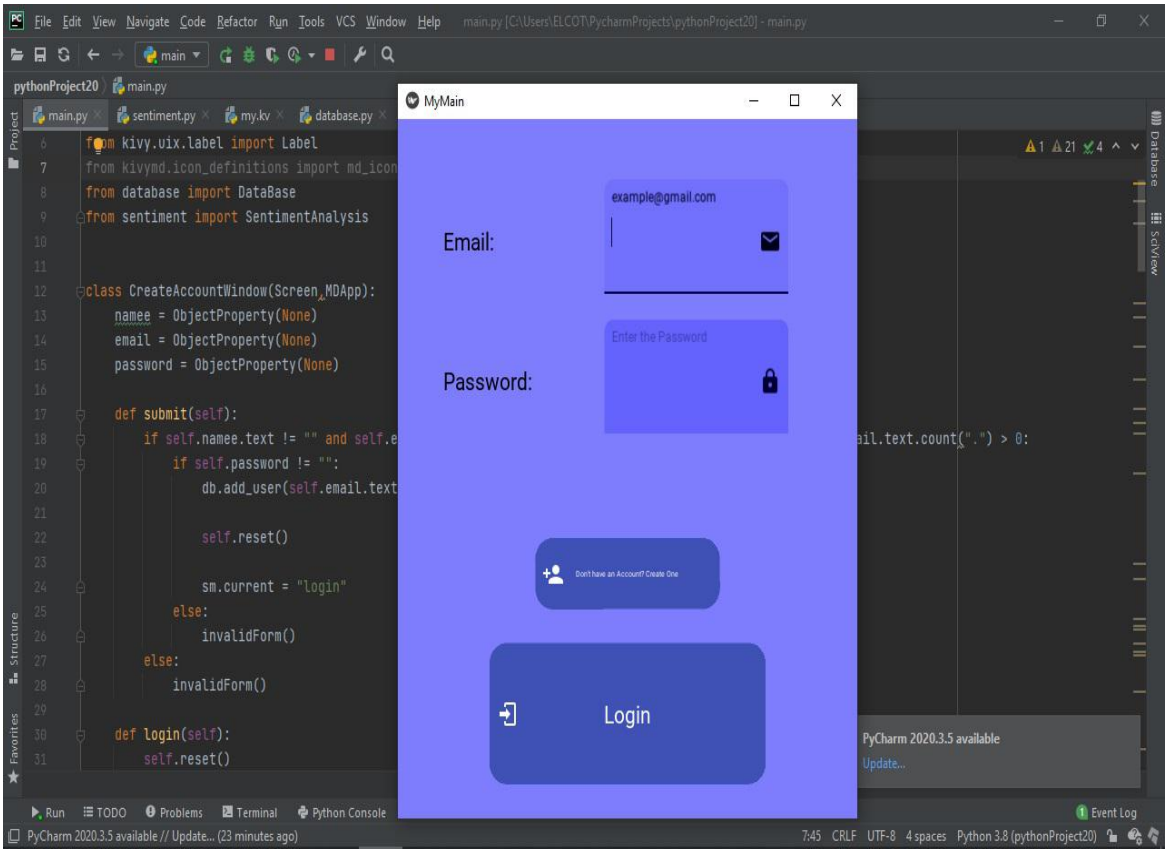
```
MDLabel:
    halign: "center"
    id: searchI
    text: "Search Item : "
    pos_hint: {"x": 0.2, "top": 0.9}
    size_hint: 0.50, 0.20
    font_size: (root.width ** 2 + root.height ** 2) / 13 ** 4

MDLabel:
    halign: "center"
    id: a1
    text: "Overall Sentiment : "
    pos_hint: {"x": 0.2, "top": 0.8}
    size_hint: 0.50, 0.15
    font_size: (root.width ** 2 + root.height ** 2) / 14 ** 4

MDLabel:
    id: b1
    text: "Percentage of Positive tweets : "
    pos_hint: {"x": 0.2, "top": 0.7}
    size_hint: 0.50, 0.15
    theme_text_color: "Custom"
    text_color: 0.1, 0.4, 0, 1

MDIconButton:
```

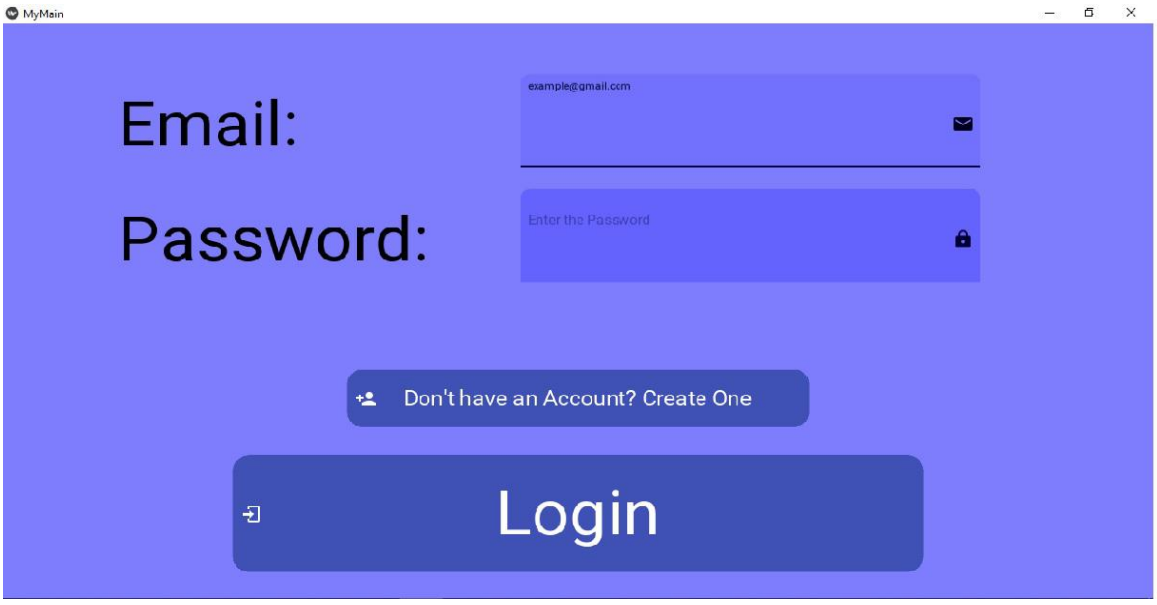
```
    icon: "emoticon"
    pos_hint: {"x": 0.1, "top": 0.67}
    theme_text_color: "Custom"
    text_color: 0.1, 0.4, 0, 1

MDLabel:
    id: c1
    text: "Percentage of Negative tweets : "
    pos_hint: {"x": 0.2, "top": 0.6}
    size_hint: 0.50, 0.15
    theme_text_color: "Custom"
    text_color: 0.6, 0, 0, 1

MDIconButton:
    icon: "emoticon-sad"
    pos_hint: {"x": 0.1, "top": 0.57}
    theme_text_color: "Custom"
    text_color: 0.6, 0, 0, 1


MDLabel:
    id: d1
    text: "Percentage of Neutral tweets : "
    pos_hint: {"x": 0.2, "top": 0.5}
    size_hint: 0.50, 0.15
```

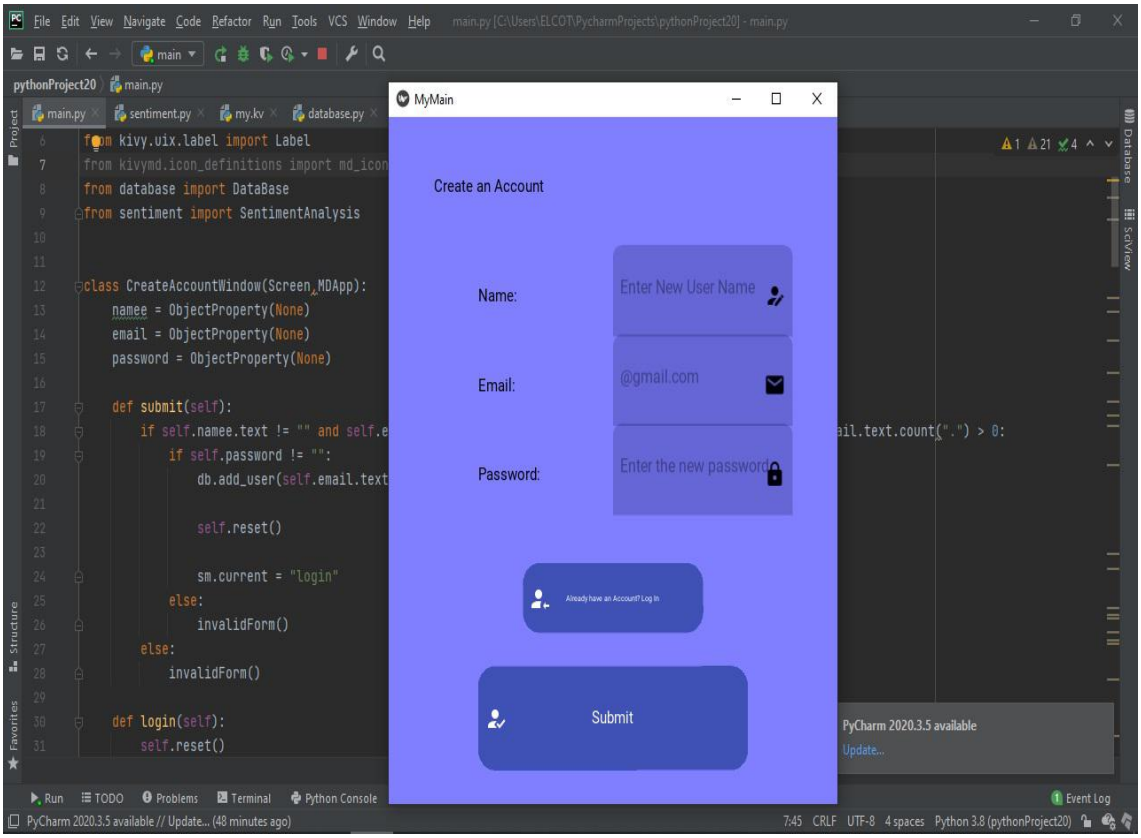
```
    theme_text_color: "Custom"
    text_color: 0.7, 0.4, 0.2, 1
MDIconButton:
    icon: "emoticon-neutral"
    pos_hint: {"x": 0.1, "top": 0.47}
    theme_text_color: "Custom"
    text_color: 0.7, 0.4, 0.2, 1

MDFillRoundFlatIconButton:
    icon: "twitter-retweet"
    pos_hint: {"x": 0.3, "top": 0.35}
    size_hint: 0.3, 0.15
    font_size: (root.width ** 2 + root.height ** 2) / 15 ** 4
    text: "search again"
    on_release:
        root.gomain()
        root.manager.transition.direction = "down"

MDFillRoundFlatIconButton:
    icon: "logout"
    pos_hint: {"x": 0.33, "top": 0.15}
    size_hint: 0.25, 0.1
    font_size: (root.width ** 2 + root.height ** 2) / 16 ** 4
    text: "logout"
    on_release:
```

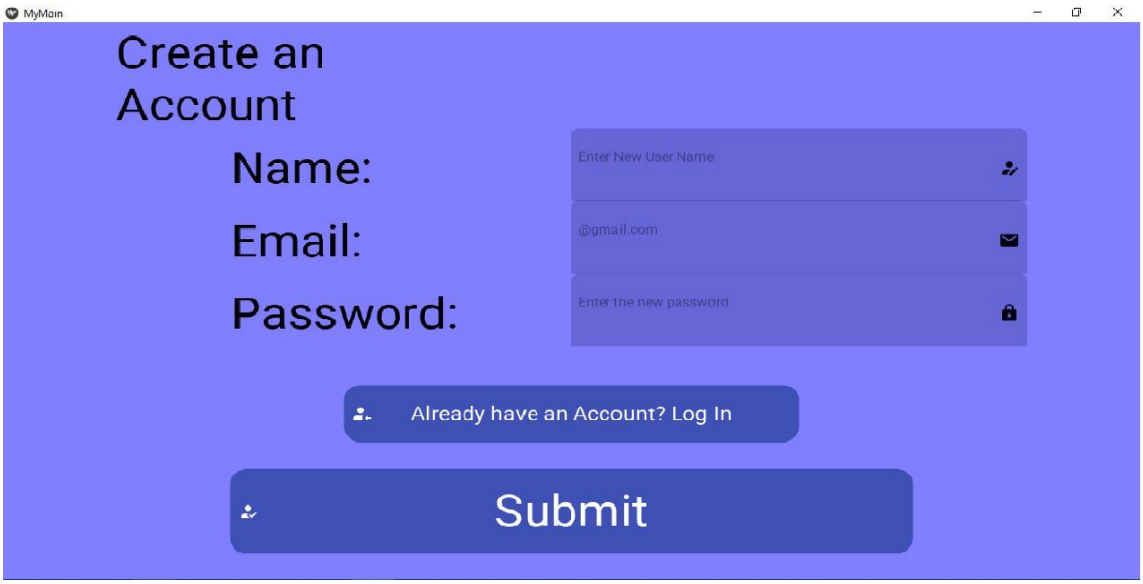
```
root.logout()
root.manager.transition.direction = "down"
```

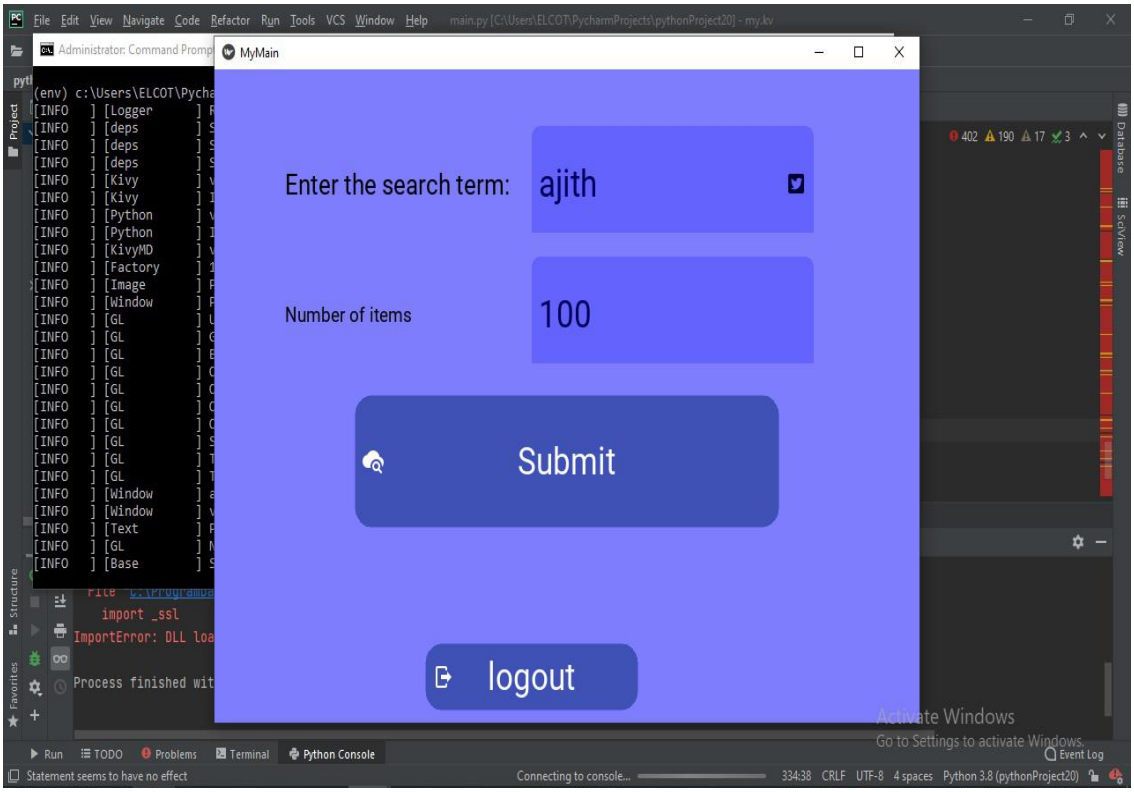
9.2 SCREENSHOTS

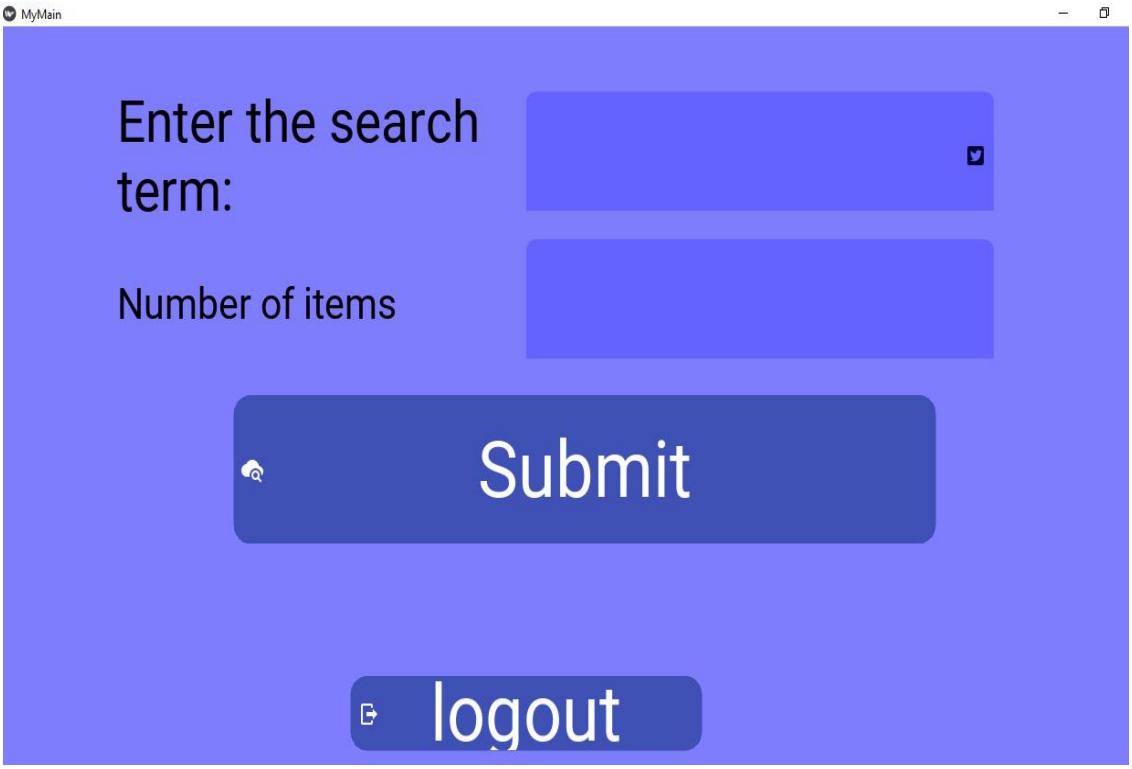


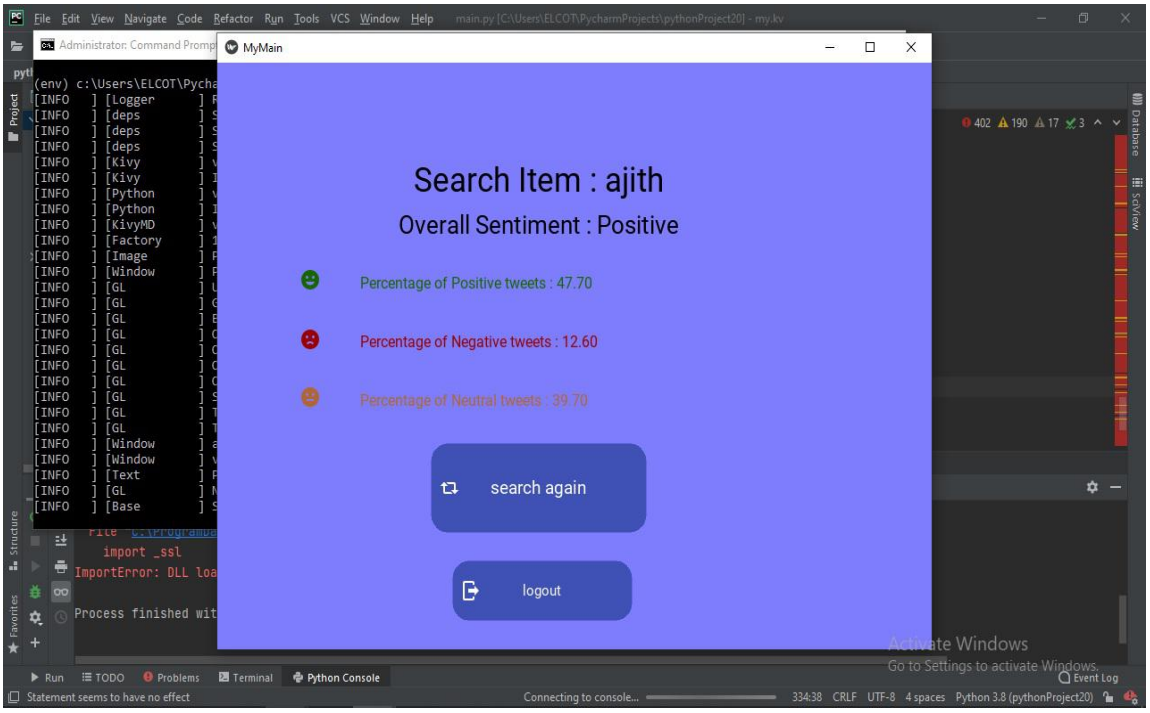


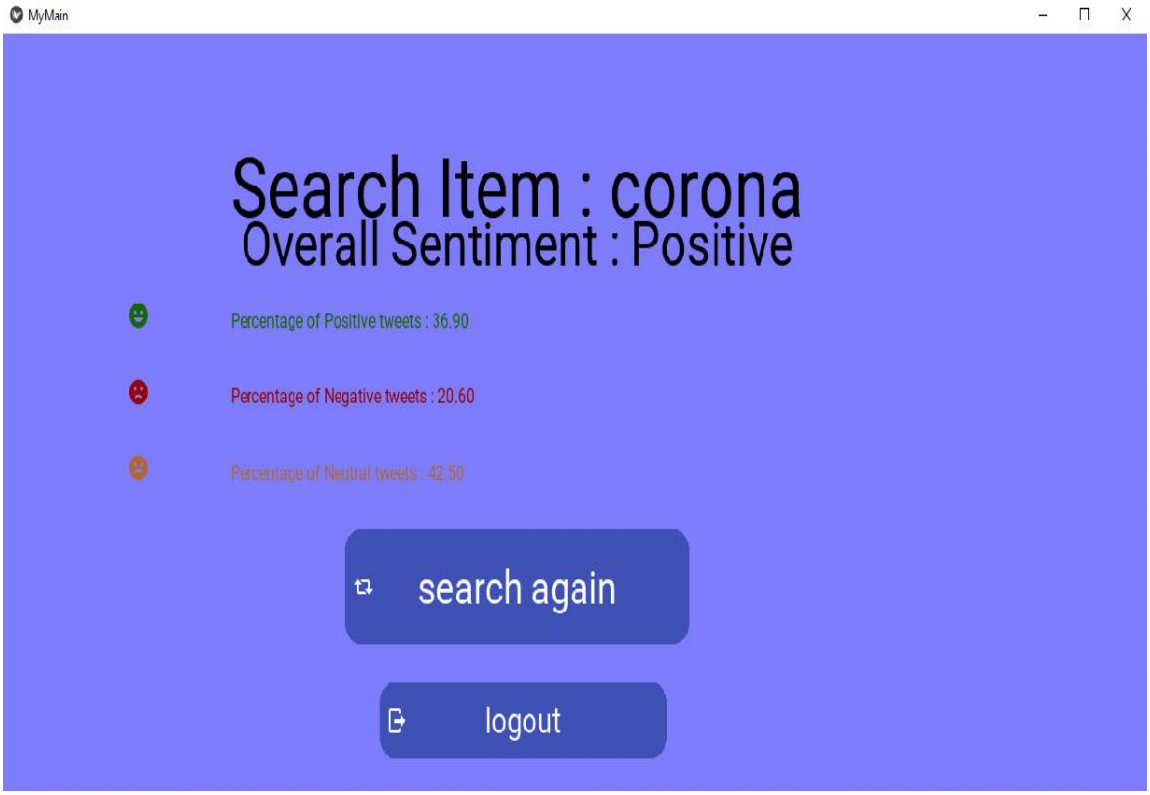

MyMain
Create an Account
Name:
Enter New User Name
Email:
@gmail.com
Password:
Enter the new password
Already have an Account? Log In
Submit


MyMain
Create an Account
Name:
Enter New User Name
Email:
@gmail.com
Password:
Enter the new password
Already have an Account? Log In
Submit


MyMain
Enter the search term:
ajith
Number of items
100
Submit
logout


MyMain
Enter the search term:
Number of items
Submit
logout


MyMain
Search Item : ajith
Overall Sentiment : Positive
Percentage of Positive tweets : 47.70
Percentage of Negative tweets : 12.60
Percentage of Neutral tweets : 39.70
search again
logout


MyMain
Search Item : corona
Overall Sentiment : Positive
Percentage of Positive tweets : 36.90
Percentage of Negative tweets : 20.60
Percentage of Neutral tweets : 42.50
search again
logout

CHAPTER 10
REFERENCES

[1] H. Zang, “The optimality of Naïve-Bayes”, Proc. FLAIRS, 2004.

[2] C.D. Manning, P. Raghavan and H. Schütze, “Introduction to Information Retrieval”, Cambridge University Press, pp. 234-265, 2008.

[3] McCallum and K. Nigam, “A comparison of event models for Naive Bayes text classification”, Proc. AAAI/ICML-98 Workshop on Learning for Text Categorization, pp. 41-48, 1998.

[4] M. Schmidt, N. L. Roux and F. Bach, “Minimizing finite Sums with the Stochastic Average Gradient”, 2002.

[5] Y. LeCun, L. Bottou, G. Orr and K. Muller, “Efficient BackProp”, Proc. In Neural Networks: Tricks of the trade 1998.

[6] T. Wu, C. Lin and R. Weng, “Probality estimates for multi-class classification by pairwise coupling”, Proc. JMLR-5, pp. 975-1005, 2004.

[7] “Support Vector Machines” [Online], http://scikit-learn.org/stable/modules/svm.html#svm-classification, Accessed Jan 2016.

[8] P. Pang, L. Lee and S. Vaithyanathan, “Thumbs up? sentiment classification using machine learning techniques”, Proc. ACL-02 conference on Empirical methods in natural language processing, vol.10, pp. 79-86, 2002.

[9] P. Pang and L. Lee, “Opinion Mining and Sentiment Analysis. Foundation and Trends in Information Retrieval”, vol. 2(1-2), pp.1-135, 2008.

[10] E. Loper and S. Bird, “NLTK: the Natural Language Toolkit”, Proc. ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics ,vol. 1,pp. 63-

70, 2002.

[11] H. Wang, D. Can, F. Bar and S. Narayana, “A system for real-time Twitter sentiment analysis of 2012 U.S.presidential election cycle”, Proc. ACL 2012 System Demostration, pp. 115-120, 2012.

[12] O. Almatrafi, S. Parack and B. Chavan, “Application of location-based sentiment analysis using Twitter for identifying trends towards Indian general elections 2014”. Proc. The 9th International Conference on Ubiquitous Information Management and Communication, 2015.

[13] L. Jiang, M. Yu, M. Zhou, X. Liu and T. Zhao, “Target-dependent twittersentiment classification”, Proc. The 49th Annual Meeting of the Association.