A PROJECT REPORT ON

WHATSAPP GROUP CHAT ANALYSIS AND PREDICTION USING NAÏVE BAYES CLASSIFICATION

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

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by

SHOVON RAUL	(19P31A0550)
S.D.J RAMA KOTI REDDY	(19P31A0546)
T. SAI KIRAN	(19P31A0556)
K. VEERA GANESH	(20P35A0505)

Under the esteemed supervisor of

Mr. Ch. S.V.V.S.N. Murty, M. Tech, (Ph.D).

Associate Professor

ADITYA COLLEGE OF ENGINEERING & TECHNOLOGY

(Approved by AICTE, New Delhi & Affiliated to JNTUK, Kakinada)

Accredited by NAAC (A+) and NBA

Surampalem, Kakinada District, Andhra Pradesh - 533 437

2019-2023

VISION

To induce higher planes of learning by imparting technical education with

- ✓ International standards
- ✓ Applied research
- ✓ Creative Ability
- ✓ Value based instruction and to emerge as a premiere institute

MISSION

Achieving academic excellence by providing globally acceptable technicaleducation by forecasting technology through

- ✓ Innovative Research And development
- ✓ Industry Institute Interaction
- ✓ Empowered Manpower

PRINCIPAL
PRINCIPAL
Aditya College of
Engineering & Technology
SURAMPALEM- 533 437

ADITYA
COLLEGE OF ENGINEERING & TECHNOLOGY
Approved by AICTE, New Delhi • Permanently Affiliated to JNTUK, Kakinada
Accredited by NBA, & NAAC (A+) with CGPA of 3.4
Recognized by UGC Under Section 2(f) and 12(B) of the UGC Act, 1956
Ph: 99591 76665
Email: office@acet.ac.in
Website: www.acet.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION:

To become a center for excellence in Computer Science and Engineering education and innovation.

MISSION:

- Provide state of art infrastructure
- Adapt skill-based learner centric teaching methodology
- Organize socio cultural events for better society
- Undertake collaborative works with academia and industry
- Encourage students and staff self-motivated, problem-solving individuals using Artificial Intelligence
- Encourage entrepreneurship in young minds.

Head of the Department

Head of the Department
Dept.of CSE
Aditya College of Engineering
& Technology
SURAMPALEM-533437

ADITYA
COLLEGE OF ENGINEERING & TECHNOLOGY
Approved by AICTE, New Delhi • Permanently Affiliated to JNTUK, Kakinada
Accredited by NBA, & NAAC (A+) with CGPA of 3.4
Recognized by UGC Under Section 2(f) and 12(B) of the UGC Act, 1956
Ph: 99591 76665
Email: office@acet.ac.in
Website: www.acet.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM EDUCATIONAL OBJECTIVES

PEO1: Capability to design and develop new software products as per requirements of the various domains and eligible to take the roles in various government, research organizations and industry

PEO2: More enthusiastic to adopt new technologies and to improve existing solutions by reducing complexity which serves society requirements as per timeline changes

PEO3: With good hands-on basic knowledge and ready improve academic qualifications in India or Abroad

PEO4: Ability to build and lead the team to achieve organization goals.

Head of the Department

Head of the Department
Dept.of CSE
Aditya College of Engineering
& Technology
SURAMPALEM-533437

ADITYA
COLLEGE OF ENGINEERING & TECHNOLOGY
Approved by AICTE, New Delhi • Permanently Affiliated to JNTUK, Kakinada
Accredited by NBA, & NAAC (A+) with CGPA of 3.4
Recognized by UGC Under Section 2(f) and 12(B) of the UGC Act, 1956
Ph: 99591 76665
Email: office@acet.ac.in
Website: www.acet.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM SPECIFIC OUTCOMES

PSO 1: The ability to design and develop computer programs for analyzing the data.

PSO 2: The ability to analyze data & develop Innovative ideas and provide solution by adopting emerging technologies for real time problems of software industry.

PSO 3: To encourage the research in software field that contribute to enhance the standards of human life style and maintain ethical values.

Head of the Department

Head of the Department
Dept.of CSE
Aditya College of Engineering
& Technology
SURAMPALEM-533437

ADITYA
COLLEGE OF ENGINEERING & TECHNOLOGY
Approved by AICTE, New Delhi • Permanently Affiliated to JNTUK, Kakinada
Accredited by NBA, & NAAC (A+) with CGPA of 3.4
Recognized by UGC Under Section 2(f) and 12(B) of the UGC Act, 1956
Ph: 99591 76665
Email: office@acet.ac.in
Website: www.acet.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM OUTCOMES

1. ENGINEERING KNOWLEDGE: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. PROBLEM ANALYSIS: Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. DESIGN/DEVELOPMENT OF SOLUTIONS: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. CONDUCT INVESTIGATIONS OF COMPLEX PROBLEMS: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. MODERN TOOL USAGE: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

6. THE ENGINEER AND SOCIETY: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues, and the consequent responsibilities relevant to the professional engineering practice:

7. ENVIRONMENT AND SUSTAINABILITY: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. ETHICS: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. INDIVIDUAL AND TEAM WORK: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

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

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

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

Head of the Department
Head of the Department
Dept.of CSE
Aditya College of Engineering
& Technology
SURAMPALEM-533437

ADITYA COLLEGE OF ENGINEERING & TECHNOLOGY

(Approved by AICTE, New Delhi & Affiliated to JNTUK, Kakinada)

Accredited by NAAC (A+) and NBA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project work entitled, "WHATSAPP GROUP CHAT ANALYSIS AND PREDICTION USING NAÏVE BAYES CLASSIFICATION", is a bonafide work carried out by SHOVON RAUL (19P31A0550), S.D.J RAMA KOTI REDDY (19P31A0546),T. SAI KIRAN (19P31A0556), K. VEERA GANESH (20P35A0505)**,** in partial fulfillment of the requirements for the award **of the degree of** BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING **from** Aditya College of Engineering and Technology**, Surampalem, during the academic year 2022-2023.**

This project work has not been submitted in full or part to any other University or educational institute for the award of any degree or diploma.

PROJECT SUPERVISOR

Mr. Ch. S.V.V.S.N. Murty,

M. Tech, (Ph.D.)

Associate Professor

HEAD OF THE DEPARTMENT

Dr. M. Anil Kumar,

M. Tech., Ph.D.

Professor

EXTERNAL EXAMINER

DECLARATION

We hereby declare that this project entitled "WHATSAPP GROUP CHAT ANALYSIS AND PREDICTION USING NAÏVE BAYES CLASSIFICATION" has been undertaken by us and this work has been submitted to **ADITYA COLLEGE OF ENGINEERING AND TECHONOLOGY**, Surampalem affiliated to JNTUK, Kakinada, in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING.**

We further declare that this project work has not been submitted in full or part to any other University or educational institute for the award of any degree or diploma**.**

PROJECT ASSOCIATES

SHOVON RAUL	**(19P31A0550)**
S.D.J RAMAKOTI REDDY	**(19P31A0546)**
T. SAI KIRAN	**(19P31A0556)**
K. VEERA GANESH	**(20P35A0505)**

ACKNOWLEDGEMENT

It is with immense pleasure that we would like to express our indebted gratitude to my project supervisor**,** Mr. CH. S.V.V.S.N MURTY**,** M.Tech., (Ph.D.), who has guided us a lot and encouraged us in every step of project work, his valuable moral support and guidance has been helpful in successful completion of this Project.

We wish to express our sincere thanks to Dr. M. ANIL KUMAR M.Tech.,Ph.D., Head of the Department of CSE, for his valuable guidance given to us throughout the period of the project work.

We feel elated to thank Principal**,** Dr. DOLA SANJAY S M.Tech.,Ph.D., of Aditya College of Engineering and Technology for his cooperation in completion of our project and throughout our course.

We feel elated to thank Dr. A. RAMA KRISHNA M.Tech.,Ph.D.**,** Dean (Academics & Administration) of Aditya College of Engineering and Technology for his cooperation in completion of our project work.

We wish to express our sincere thanks to all faculty members, and lab programmers for their valuable guidance given to us throughout the period of the project.

We avail this opportunity to express our deep sense and heart full thanks to the Management of Aditya College of Engineering & Technology for providing a great support for us by arranging the trainees, and facilities needed to complete our project and for giving us the opportunity for doing this work.

PROJECT ASSOCIATES

SHOVON RAUL	**(19P31A0550)**
S.D.J RAMA KOTI REDDY	**(19P31A0546)**
T. SAI KIRAN	**(19P31A0556)**
K. VEERA GANESH	**(20P35A0505)**

ABSTRACT

Recently WhatsApp has become the most used and efficient method of communication and people are more interested in doing important collaborate tasks using WhatsApp Group Chat facilities. As group can be created for various purposes in different field of work, these group chats can be sources of lot of information for further analysis and problem solving. Our project “WhatsApp group chat analysis and prediction using naïve bayes classification” can provide an in-depth statistical analysis based on the given group chat data and a better understanding about people’s behaviours through chats which can be further useful for sentiment analysis later. We have included some special features to existing systems like, our tool can future predict the most suitable hour of the day for admin’s reply in the group, Categorize the subject of the group, giving warnings after recognizing media overload and hate speech, Linguistic Manifestations of WhatsApp conversations, language detection etc. In brief the aim of the project is doing statistical analysis as well as future predictions to include all possible feature in a single application. The naïve bayes classification algorithm used here are easy to handle, efficient and less resource consuming algorithm, therefore it will succeed while applying to larger datasets also. This project will help individuals, group administrators and any kind of organizations for efficient analysis of WhatsApp group chats for their specific purposes.

INDEX

CHAPTER	PAGE NO
ABSTRACT	**ix**
LIST OF FIGURES	**xii**
LIST OF TABLES	**xiii**
1. INTRODUCTION	1
1.1 Existing System	2
1.2 Scope of the Existing System	2
1.3 Proposed System	3
1.4 Novelty of Proposed System	4
2. REQUIREMENT ANALYSIS	5
2.1 Functional Requirements	6
2.2 Non-Functional Requirements	6
2.2.1 User Interface and Human Factors	6
2.2.2 Software Requirements	7
2.2.3 Hardware Requirements	7
2.2.4 Usability	7
2.2.5 Reliability	8
2.2.6 Performance	8
2.2.7 Supportability	8
2.2.8 Physical Environment	9
2.2.9 Security Requirements	9
2.2.10 Resource Requirements	9
3. SYSTEM ANALYSIS	11
3.1 Introduction	12
3.2 Use Cases	12
3.2.1 Actors	12
3.2.2 List of use cases	13
3.3.3 Use case diagrams	14
4. SYSTEM DESIGN	15
4.1 Introduction	16
4.2 System Architecture	16

4.3 System Object Model 17
4.3.1 Introduction 17
4.3.2 Subsystems 17
4.4 Object Description 18
4.4.1 Objects 18
4.4.2 Class Diagrams 19
4.5 Dynamic Model 20
4.5.1 Sequence Diagrams 21
4.5.2 Activity Diagrams 22
4.6 Static Model 23
4.6.1 Component Diagrams 23
4.6.2 Deployment Diagrams 24
5. IMPLEMENTATION
5.1 Software Environment 26
5.2 Liraries Used 31
5.3 Naive Bayes Classification 33
5.3.1 Multinomial Naïve Bayes 34
5.3.2 Count Vectorizer 35
5.3.3 Pipeline in multinomial Naïve Bayes 36
5.3.4 Advantages of multinomial 37
5.4 Source Code 38
6. TESTING
6.1 Introduction to testing 60
6.2 Types of testing 61
6.2.1 Unit Testing 61
6.2.2 Integration Testing 61
6.2.3 Functional Testing 62
6.2.3.1 Test Case Templates 63
6.2.4 Load Testing 66
7. OUTPUT SCREENS 70
8. DEPLOYMENT AND MAINTENANCE 80
9. CONCLUSION 81
10. BIBLIOGRAPHY 82

LIST OF FIGURES

S.No.	NAME OF FIGURE	PAGE No.
1	1.1 Demo Chat Analysis	3
2	2.1 Sentiment Analysis Dataset	9
3	2.2 Language Dataset	10
4	2.3 WhatsApp Dataset	10
5	3.1 Use Case Diagram	14
6	4.1 System Architecture	16
7	4.2 Subsystems	18
8	4.3 Class Diagram	20
9	4.4 Sequence Diagram	21
10	4.5 Activity Diagram	22
11	4.6 Component Diagram	23
12	4.7 Deployment Diagram	24
13	5.1 Language Detection Model	38
14	5.2 Sentiment Detection Model	43
15	6.1 Load Test Interface	67
16	6.2 Load Test Report	68
17	7.1 User Interface	70
18	7.2 Whatsapp Chat Dataframe	70
19	7.3 Total Messages,words,url and Monthly Timeline	71
20	7.4 Daily Timeline	71
21	7.5 Activity Map	72
22	7.6 Most Busy User	72
23	7.7 WordCloud	73
24	7.8 Frequent Words	73
25	7.9 Most Common Emojis	74
26	7.10 Language Partitioned dataframe	74
27	7.11 English and non-English message Count	75
28	7.12 User Partition on using other languages	75

29	7.13 Message Sentimental Dataframe	76
30	7.14 Sentimental Percentage	76
31	7.15 WordCloud Joy	77
32	7.16 WordCloud Sadness	77
33	7.17 WordCloud Fear	78
34	7.18 WordCloud anger	78
35	7.19 WordCloud Surprise	79
36	7.20 WordCloud neutral	79

LIST OF TABLES

S.No.	NAME OF TABLE	PAGE No.
1	3.1 Use Cases of Admin	13
2	3.2 Use Cases of actor user	13
3	3.3 Use Cases of actor system	14
4	6.1 Test Case Template-1	63
5	6.2 Starting Test Case	63
6	6.3 Test Case Template-2	64
7	6.4 Statistical Test Case	64
8	6.5 Test Case Template-3	65
9	6.6 Language & Sentimental Test Case	65

CHAPTER-1

INTRODUCTION

1.INTRODUCTION

Today one of the trendy social media platforms is…. guess what? One and only WhatsApp. It is one of the favorite social media platforms among all of us because of its attractive features. It has more than 2B users worldwide and "According to one survey an average user spends more than 195 minutes per week on WhatsApp". How terrible the above statement is. Leave all these things and let's understand what actually WhatsApp analyzer means?

WhatsApp Analyzer means we are analyzing our WhatsApp group activities. It tracks our conversation and analyses how much time we are spending or saying it as "wasting" on WhatsApp. The aim of this project is to provide step by step guide to build our own WhatsApp analyzer using python. Here we used different python libraries which helps us to extract useful information from raw data. As group can be created for various purposes in different field of work, these group chats can be sources of lot of information for further analysis and problem solving.

In our society we come across problems based on the communications in social media where social violence occurs. As mentioned, WhatsApp is used by billions of people definitely we should have a system for detecting such events so that we can solve this before it is creating more trouble. If we build a system for sentiment analysis exclusively for WhatsApp chat then it will solve this problem.

We can add features like language detection in analysis as well so that admins can detect messages and labeled them with particular language for WhatsApp groups having connection with different linguistic persons.

This project will provide an in-depth statistical analysis based on the given group chat data and a better understanding about people's behavior through chats which can be further useful for sentiment analysis later.

1.1 Existing System

There are some basic applications available which can do basic analysis of WhatsApp chats by getting the input data from export chat feature of WhatsApp, but these apps are not so much popular due to limited analysis power, some third-party apps required so much of permissions of devices and trust issues and can't use for sentiment analysis. WhatsApp has in build end to end encryption features. So, we can't add any features without considering the security. WhatsApp has not such features available for their end users till now. The scope of the existing systems as followed.

1.2 Scope of the Existing System

The existing systems available till now is to perform only statistical analysis on the data given by the WhatsApp using the export data of any particular chat for a contact. But we are in the time of artificial intelligence and there is no automation available for this particular analysis. We can't do any kind of prediction about what the group member wants to tell without manually reading the whole WhatsApp chats which are very time consuming as well as hard working process. As mentioned earlier sentiment analysis is a very important step of evolving in digital communication regarding messaging applications like WhatsApp. As WhatsApp is used by millions of customers in a group communication, any kind of conceptual problems and hate speeches can create so much social issues. Sometimes they are using different languages while chatting and it is very crucial to understand what language they are talking about. Certainly, systems with intellectual abilities are urgent to society to solve this problem automatically.

Investigation organizations may have such trustworthy applications but they are used for very special purposes and not on demand of any particular individuals. We can not go directly to police station each time we came across any small conflict happens due to silly jokes or discussion on nonsense topics in any community group in WhatsApp. As well as we cannot share such confidential chats to some persons with data analytics skills for manual analysis or to some third-party applications where we don't know which kind of operations they are doing on our information. They may not trustworthy and if we will do that there is no use of end-to-end encryption. Our proposed system can solve this problem preciously.

1.3 Proposed System

In our proposed system we will include two main features

- First of all, the initial part of the project is to understand implementation and usage of various python- built modules so that the input data can be formatted for future use and statistical analysis will be done using Numpy, Pandas modules.

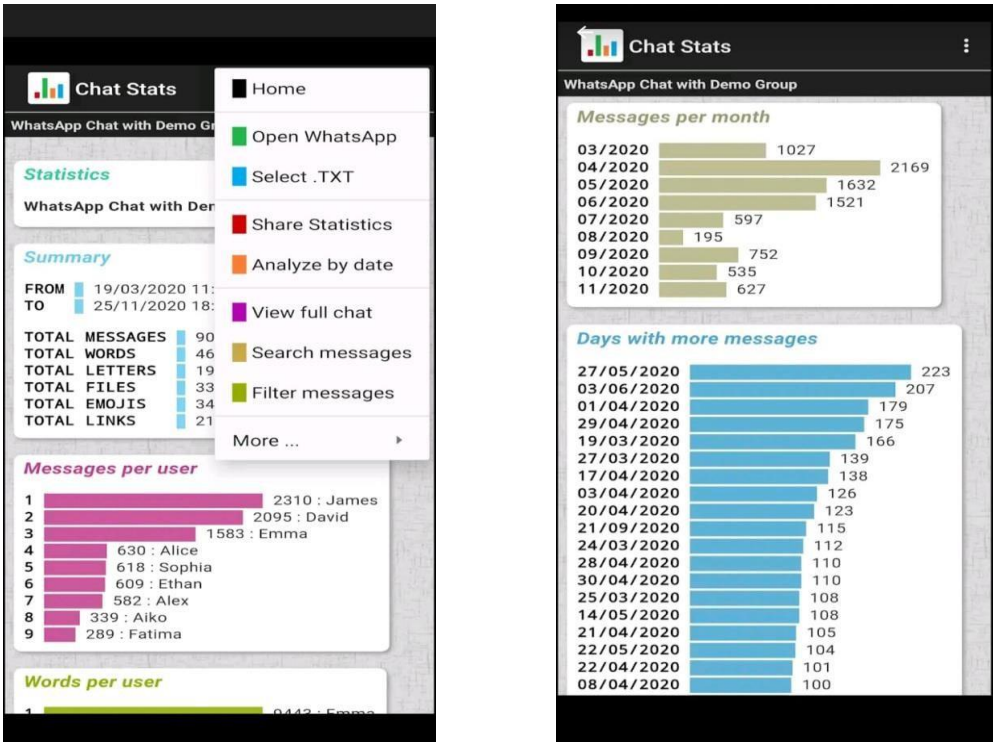


Fig:1.1 Demo Chat Analysis

1. The above picture demonstrates the analysis made in one what's app group for a period of time i.e 19-03-2020 to 23-11-2020. From the picture we observed that total of 8446 messages are shared in the group by 9 different people, among which James shared 2310, David shared 2095, Brownie shared 1583, Alice, Sophie, Ethan, Alex, Alko, Fatima accounts for 630,618,609,582,339,289 respectively. Further it also represents the number of words , letters, files, emojis, links shared in the group. In the similar way our project will be depicting the sentimental analysis of a whatsapp group in a graphical manner using naive bayes algorithm.By representing the outcome in the above manner one can easily understand the various statistical views of our application.

- In the next part, which is an exploratory data analysis part we will apply sentiment analysis algorithm based on naïve bias classification and advanced python modules which will differentiate messages into positive, negative and neutral labels so that we can understand the sentiment of the chat based on some mathematical value. We are building a language model based on naïve bayes classification to check the languages in the chat as well.

1.4 Novelty of Proposed System

Novelty of our proposed system are as followed

- Our proposed system works in such a way that any WhatsApp user can use and enjoy the power of analysis and they can get their required result in short amount of time.
- No such systems are implemented till now for society that can do both statistical and sentimental analysis for end users of WhatsApp and WhatsApp has not included any such features in it as well.
- Some of the systems which are specially designed for such analytical purpose of WhatsApp are usually limited to big organization and government agencies. Our system will make this available to all WhatsApp users, explicitly for the group admins having number of WhatsApp group with lot of daily conversation. So, they no need to depend on any kind of lawful actions for small problems.
- In case of countries like India there are people with different languages and cultures. For a group with different linguistic people, sometimes we face some messages with different languages. Our project can solve this by identifying people and labeling their languages using language detection model and make it east for group admins to do their work.

CHAPTER-2

REQUIREMENT ANALYSIS

2. REQUIREMENT ANALYSIS

2.1 Functional Requirements

- There has to be a proper administrative interface to collect the WhatsApp data
- Proper Datasets should be available as training datasets for Language detection
- We have used the language datasets and sentimental analysis dataset for training the different models using Naïve Bayes Classifier.
- Imported chat data must be cleaned to remove inconsistent data and perform statistical analysis on it.
- Later the data must be processed by sentimental analysis to give sentimental data outcomes.
- After Extracting all the outcomes from the analysis done a proper visual representation format should be available to show which is human understandable.

2.2 Non Functional Requirements

2.2.1 User Interface and Human Factors

User interface is actual blueprint and human factors is the study of human behavior as users interact with the physical world / with a system. The user interface must be simple and easy to understand. Admin should have the permissions to use that particular WhatsApp Data available and he needs to give the input data to the application interface from WhatsApp application built in export chat data The outcome of the input chat will be organized and diagrametical representation with the help of graphs increases readability and make user familiar to the software in less time.

Human Factors includes job factors,timeline,organizational. User have to input the data and the outcome will be visible to the user directly. The user interface for our system is very easy. User has only one work in user interface that ios inputting WhatsApp data. The admin have the direct access to the code and he/she can update the training data for better accuracy.

2.2.2 Software Requirements

- Pandas
- streamlit
- Matplotlib
- Seaborn
- Sklearn
- Wordcloud
- NLTK libraries
- Pycharm IDE
- Google colab

2.2.3 Hardware Requirements

- Windows: Windows 7/8/10
- Processor (CPU): Intel Core i3/i5 (5th generation or newer)
- Memory: Minimum 8GB RAM’

2.2.4 Usability

Sentiment analysis applications are the future of our society, as we are mainly focusing on artificial intelligenece especially natural language processing for automations. Human language understanding is a great topic of machine learning and have a better future. Our applications have usuabilities like

- This can be used to detect information such as most active user,most active time.
- Helpful in detecting hate speech and sentiments of the group members with their messages.
- This can be used for language detection as well.
- The report should have proper visualization graphs to understand in short amount of time.

2.2.5 Reliability

Reliability is a most important concept while considering the software development. Softwares should be trustworthy his particular application will be more secure and reliable because if the application is not trustworthy who will like to use that. The application will fail before implementation. The reliabilities of our applications are as followed.

1. The person who is using giving the WhatsApp data has the ownership on the data.
2. The data analysis is directly showed to that individual, nothing is stored in any database, third party servers or cloud.
3. The application never stores the data locally also. So none other person can steal the information.

2.2.6 Performance

Performance makes one application successful. It will be more scalable and performance of the system will be more because

- We are using textual data which can be analyzed easily.
- Large amount of data can be imported in very short time so scalable.
- Any features relative to our system can be included in future
- It will never be a bulky application so execution speed will be more.

2.2.7 Supportability

The application will be able to process large amount of data in order to provide accurate results in terms of analyzing the goals of the group chat and classifying it as positive,negative or neutral.

Cost of Implementation is very less. No need to have greater resources for execution. We can use this application in mobile devices as well.

2.2.8 Physical Environment

Now-a-days more number of people are using WhatsApp. It becomes the main messeging app. Getting the requiered data for analysis is easy for this system. The application also can be used in mobile devices and desktop systems as well. It requires a normal room temperature with proper power supply to the system and a well functioning system to run the software.

2.2.9 Security Requirements

In order to export the chat the user must be a member in the group and so no other person outside the group can export it. The in build end-to-end encryption of whatsapp is also can be added as security requirements as we are getting the data set which is confidential. We are doing the operation locally without using any third party application which means the security requirement of the system is preserved

2.2.10 Resource Requirements

Very less resources are required in this project. A single system with the mentioned hardware and software requirements is enough for this project.

Sentiment analysis Dataset:

# Index	A review Stemmed and Lemmatized review using nltk	# polarity Polarity score generated using TextBlob	A division Categorical label generated using polarity score
0 4156	good 4% good product 2% Other (3868) 95%	-1 1	positive 75% neutral 13% Other (482) 12%
3870	able play youtube alexa	0.5	positive
62	able recognize indian accent really well drop function helpful call device talk person near device s...	0.2794	positive
487	absolute smart device amazon connect external sub woofer sound amaze recons voice even close room li	0.1827	positive

Fig 2.1: Sentiment Analysis Dataset

Source: https://www.kaggle.com/datasets/pradeeshprabhakar/preprocessed-dataset-sentiment-analysis

LanguageDetection Dataset: https://www.kaggle.com/datasets/basilb2s/language-detection

Text	Language
Text Details	Language
10267 unique values	English 13% French 10% Other (7938) 77%
Nature, in the broadest sense, is the natural, physical, material world or universe.	English
"Nature" can refer to the phenomena of the physical world, and also to life in general.	English
The study of nature is a large, if not the only, part of science.	English
Although humans are	English

Fig 2.2: Language Dataset

WhatsApp Chat Analysis Dataset:

https://www.kaggle.com/datasets/sarthaknautiyal/whatsappsample

```
25/6/15, 1:42:12 AM: Vishnu Gaud created this group
25/6/15, 1:42:12 AM: You were added
18/12/16, 1:57:38 AM: Shahain: <image omitted>
21/12/16, 9:54:46 PM: Pankaj Sinha: <image omitted>
21/12/16, 9:57:45 PM: Shahain: Wow
21/12/16, 10:48:51 PM: Sakshi: <image omitted>
21/12/16, 10:49:00 PM: Sakshi: <image omitted>
21/12/16, 10:50:12 PM: Neha Wipro: Awsum😃😃👍🏻👍🏻
21/12/16, 10:51:21 PM: Sakshi: 🙈
21/12/16, 10:57:01 PM: Ganguly: 🙂🙂👍🏻👍🏻
21/12/16, 11:28:51 PM: Vishnu Gaud: Waste out of wealth 😂
21/12/16, 11:48:42 PM: Venu Wipro: Fancy dress competition?
22/12/16, 12:08:04 AM: Kushbhu: 😂😂😂
22/12/16, 12:24:00 AM: Messages you send to this group are now secured with end-to-end enc
22/12/16, 12:25:02 AM: Nauty's phone: Superrrrrrr se bhiiiiiiii uperrrrrrrr
22/12/16, 12:36:54 AM: Sakshi: We were Divided into four groups..

Each group had to use newspapers only for dressing up one of their team members
22/12/16, 6:27:38 AM: Preeti: 👌🏻👌🏻👌🏻👌🏻
22/12/16, 12:10:31 PM: Kushbhu: Dunia ka sbse khatarnak proposal.. Must watch 😂😂😂😂😂
22/12/16, 12:10:45 PM: Kushbhu: <video omitted>
```

Fig 2.3: WhatsApp Dataset

CHAPTER-3

SYSTEM ANALYSIS

3. SYSTEM ANALYSIS

3.1 Introduction

System analysis is aa process by which individual (s) studies a system such that an information system can be analyzed, modeled, and a logical alternative can be chosen. Systems analysis projects are initiated for three reasons: problems, opportunities, and directives. The people involved include systems analysts, sponsors, and users. Here in case of our project the system analysis is very easy. We will be more focusing on the use cases and structure of the system. System analysis represents the first diagrammatic representation for creation of any project. The use cases of our project are as followed.

3.2 Use cases

A use case is a methodology used in system analysis to identify, clarify and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. The method creates a document that describes all the steps taken by a user to complete an activity. In our project the working is based on the WhatsApp data so the use cases can be data extraction to the giving all kind of results, all functionalities. To describe use cases, we need 2 main components which are actors and use case. Brief explanations about all the components are as followed.

3.2.1 Actors

An Actor models a type of role played by an entity that interacts with the subject (e.g., by exchanging signals and data), but which is external to the subject. Actors may represent roles played by human users, external hardware, or other subjects. Actors do not necessarily represent specific physical entities but merely particular facets (i.e., “roles”) of some entities that are relevant to the specification of its associated use cases. A single physical instance may play the role of several different actors and a given actor may be played by multiple different instances.

Here our project can be described using mainly three actors, those are admin user and system.

Admin actor is administrator of the system who collects the datasets and upload to the system for creating all kind of models for sentiment analysis and language detection.

User actor refers to the end user who need some analysis from the application, so he/she decides to feed the system with proper format WhatsApp data to get the result.

System is another actor which is doing all kind of internal working for the chat analysis applying algorithm and build in methods and giving the output.

Each actors have some use cases relevant to the projects. Those use cases are as followed.

3.2.2 List of Use Cases

A list of use cases is given in a tabular format for three actors.

Use Cases	**Description**
Load Dataset	Admin has use case as loading dataset to the system for analysis, datasets are of two types, so this use case has connection to sub use cases as followed.
Sentiment Dataset	Admin gives separate dataset for sentimental analysis for each user.
Language Dataset	Admin gives a separate dataset for language detection of chats.

Table 3.1: Use Cases of Admin

Use Cases	**Description**
Input WhatsApp Data	User has to input real time WhatsApp chat data into the system for further analysis for that particular chat data.
Wrong Input	If the data inputted by the user is not in proper format the system will show an error, so it is <<exclude>> association with above use case.
Getting result	User gets the report after analysis is over.

Table 3.2: Use Cases of actor user

Use cases	Description
Chat Segmentation	After getting the chats from the user system has to segment the chats in a proper format for further operation.
Statistical Analysis	System does statistical analysis about the given data
Sentimental Analysis	System does sentimental analysis about the given data for users
Language Detection	Language Detection for chats is also available in the system
Getting Result	Result is given to the user by the system.

Table 3.3: Use Cases of actor system

3.3 USE CASE DIAGRAM

Fig 3.1: Use Case Diagram

CHAPTER-4
SYSTEM DESIGN

4. SYSTEM DESIGN

4.1 Introduction

System Design is the process of designing the architecture, components, and interfaces for a system so that it meets the end-user requirements. Our project is based on WhatsApp Data Chat Analysis and in that we are doing both statistical analysis and sentimental analysis both together. We have added language detection feature to extend the actions of our application. Our Project is based on Natural Language Processing. Natural Language Processing is a part of machine learning which mainly focuses on human language, apply algorithms on that and come out with information from that input. Here we are using Naïve Bayes Classification algorithm for creating this project. We have used 3 datasets for building this project. So, language dataset and sentimental datasets are used for building the detection models and real time WhatsApp data is used for analysis of that particular data. The whole process is done using python modules like sklearn, pandas, numpy, matplotlib and many more. The detailed system design is as followed.

4.2 System Architecture

A system architecture is a representation of a system in which there is a mapping of functionality onto hardware and software components, a mapping of the software architecture onto the hardware architecture, and human interaction with these components. In case of this system, we are using a very easy system architecture. The architecture is as followed.

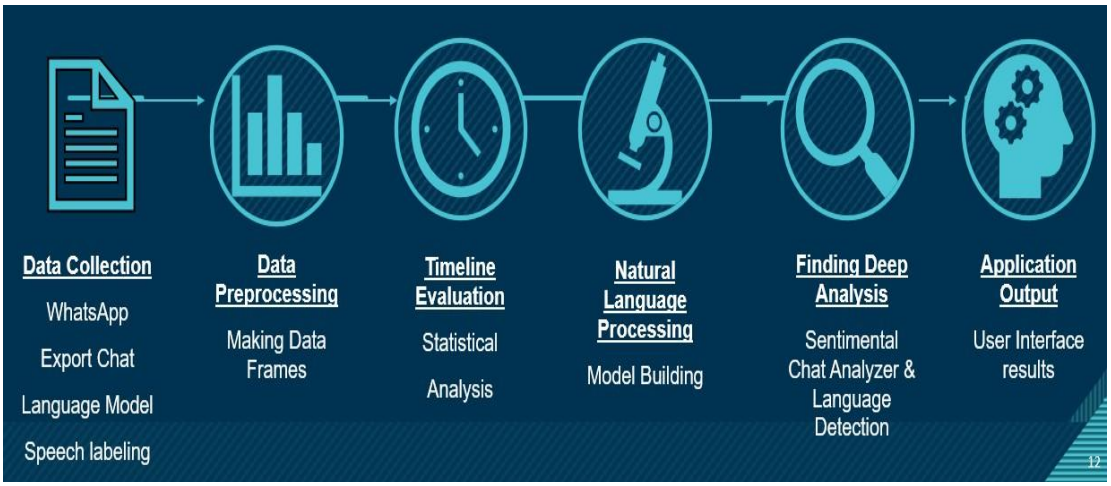


Fig 4.1: System Architecture

Here as given in the above diagram the system has to take input from outside, and in such case there are mainly three kind of inputs which are WhatsApp text data for analysis, sentiment and language detection training data for training the detection model for doing the future operation.

In the second step the Data (WhatsApp Data) which is actually a real time data is to be preprocessed to form predefined data frames so the system can do further operations on it. There will be three kind of operations which are statistical analysis, sentiment analysis and language detection.

Timeline evaluation basically refers to the statistical analysis using build in modules like pandas and matplotlib.

In case of natural language processing a chat sentiment detection model and a language detection model is built. In the next step based on the model deep analysis is done on WhatsApp data and later it is shown to the user.

4.3 System Object Model

4.3.1 Introduction

It serves as an object-oriented model that can be distinguished from other models contained in object-oriented programming languages. SOM basically includes an interface definition language, a runtime environment with procedure calls and a set of enabling frameworks. In our system object model we are preparing our model using Naïve Bayes Classification algorithm. We have compared the WhatsApp text data with some regular expression and prepared data frames from it. Later in the case we are using transforms text to feature vectors that can be used as input to estimator. So that the sentiment analyzer and language detector models can use this input for further process.

4.3.2 Subsystems

Here our project has mainly three purposes which are Chat statistical analysis, Sentimental Analysis and Language detection. So, the three subsystems here are Statistical Analyze, Sentiment Analyze and Language Detection. For sentiment analyze mainly we are using vectorization. We are getting all kind of output a report with all kind of analysis results and visualizations graphs. A pictorial representation of subsystems as followed**.**

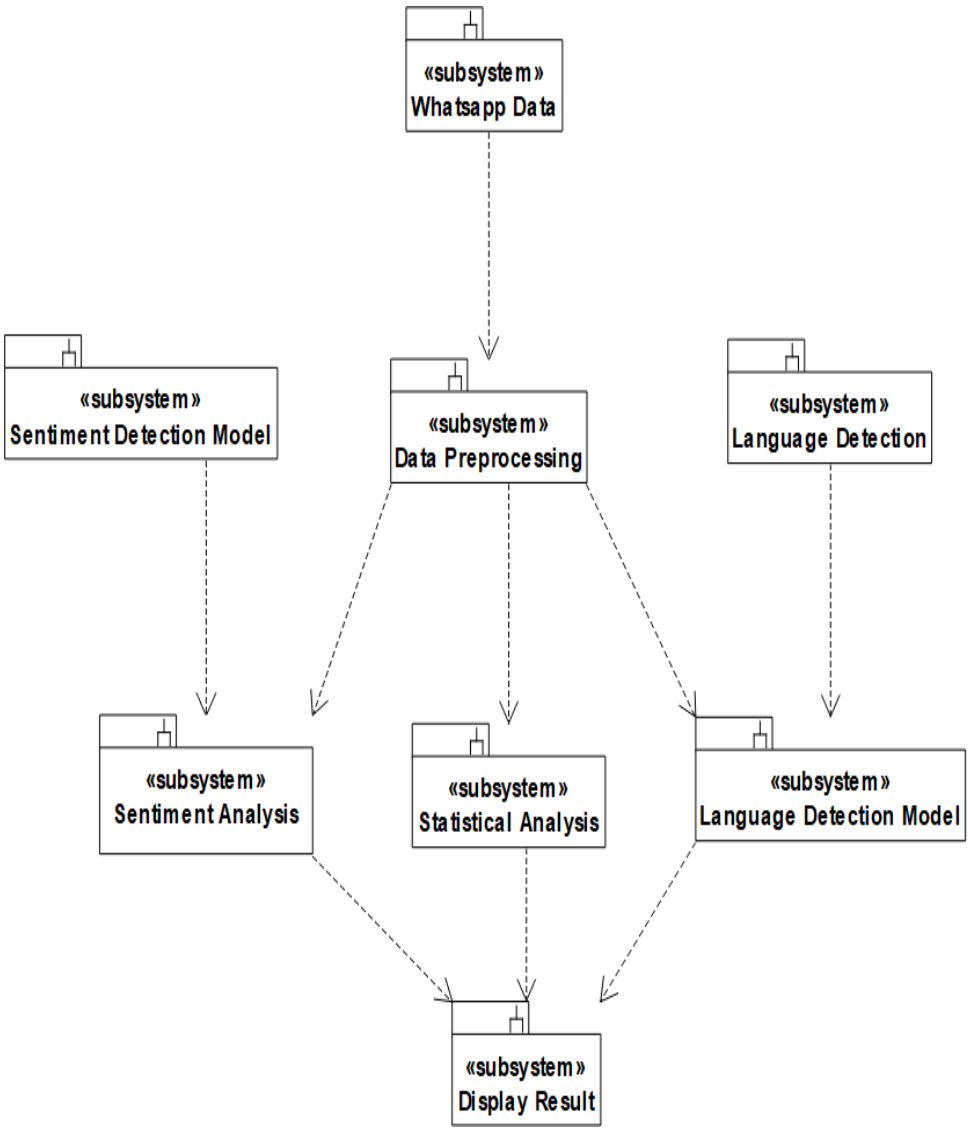


Fig 4.2: Subsystems

4.4 Object Description:

4.4.1 Objects

Objects play a major role in system design for any kind of software. Object basically refers to the runtime entity of class variable with proper functionalities and properties. Properties refer to the attributes and functionalities are refers to the operations. Objects have relations among themselves sometimes for their combinatorial working. Our WhatsApp Group Chat Analyzer and prediction application have relative objects for class variables for creating the whole system. The objects available are belongs to the classes which are as followed.

1. Data processing – object containing all the functionalities and attribute to work with data given by the user
2. Statistics of chat – functionalities required for all kind of statistical analysis with built in python modules are available here in this particular object
3. Sentiment model – It contains attributes like normalization vectors required for sentiment analysis and required datasets
4. Language model – It contains attributes like datasets required for language detection with available datasets for training and testing
5. Sentiment of chat – contains group of operations for sentimental analysis
6. Language of chat – contains group of operations for language detection
7. Display Report – showing a proper visualization about the whole analysis.

These objects which are actually the name of the particular classes are described below with a class diagram.

4.4.2 Class Diagram

A class diagram relative to our proposed system is as followed. Here the required classes are mentioned with the proper functionalities and attribute which should be available for building the total system. The diagram is very close to the coding implementation as the operations we are added for each class are the actual functions for the object in our software program. The class objects are described before in brief.

Classes are as followed

- Data Preprocessing
- Statistics Of Chats
- Language Model
- Sentiment Model
- Sentiment Of Chat
- Interface Report

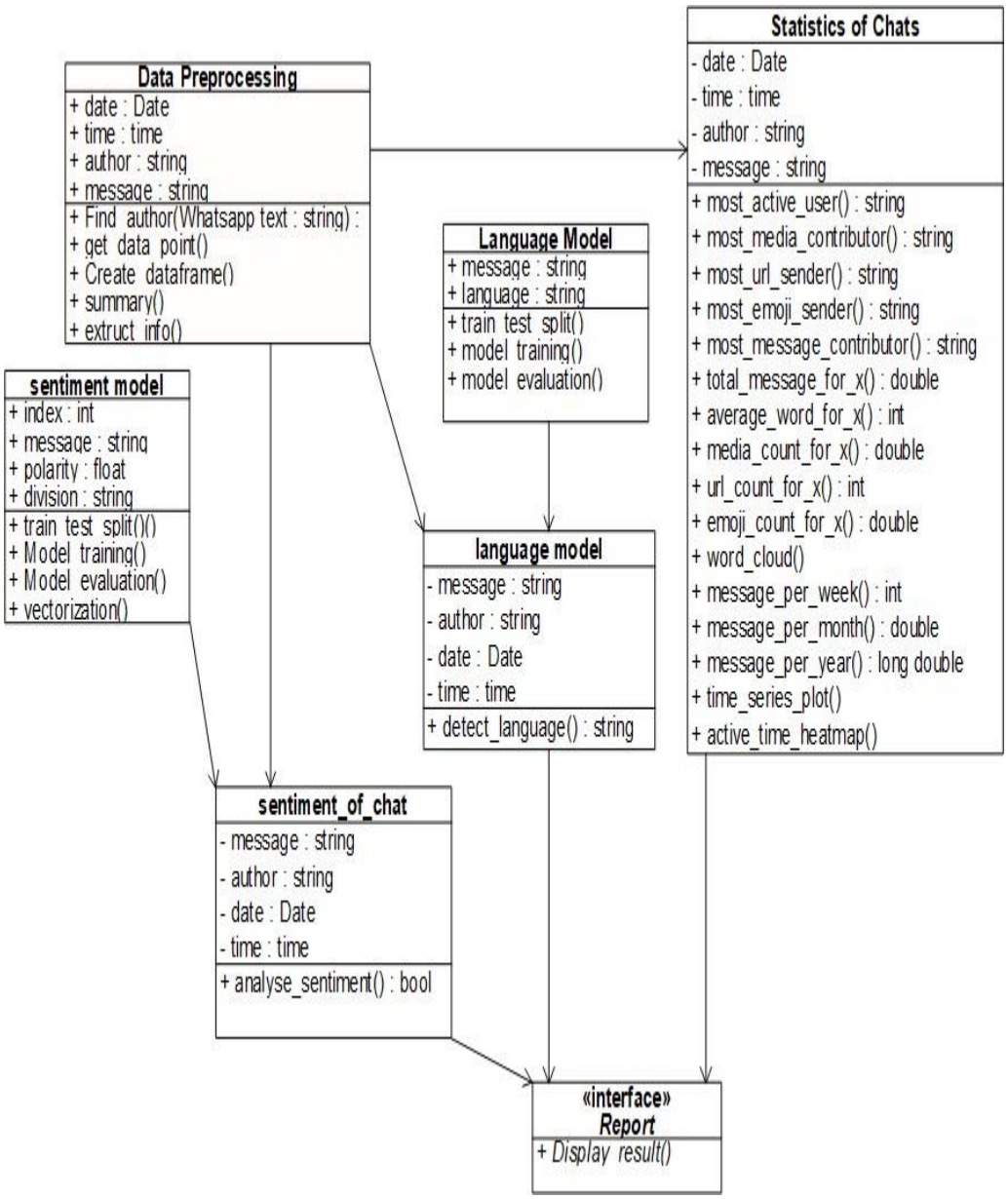


Fig 4.3: Class Diagram

4.5 Dynamic Model

The dynamic model is used to express and model the behavior of the system over time. The proposed system has been expressed by two dynamic model diagrams which are sequence diagram and activity diagram. The representation of both of them are as followed.

4.5.1 Sequence Diagrams

Sequence diagrams are used to display the interaction between users, screens, objects and entities within the system. It provides a sequential map of message passing between objects over time.

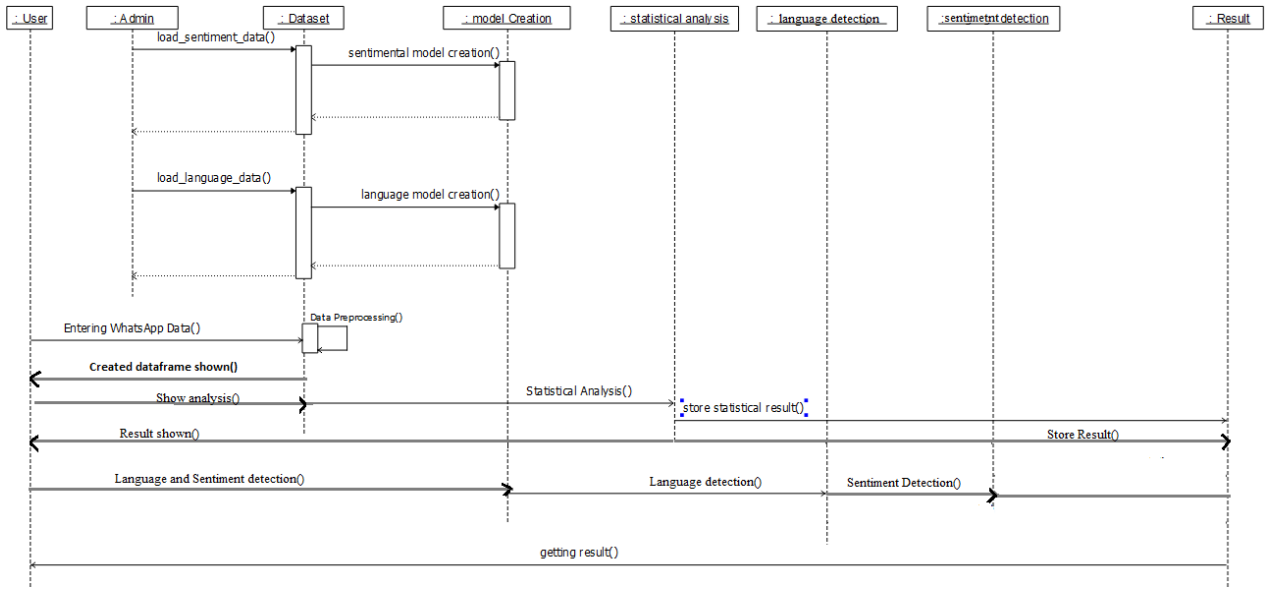


Fig 4.4: Sequence Diagram

As represented in the above sequence diagrams the objects are user, admin, dataset, model creation, statistical analysis, sentimental analysis, language detection and lastly the result.

In the first two dynamic processes the admin has to give sentimental analysis training dataset followed by language detection training dataset for creating the detection model for each individually. It should be synchronous signal as the model is created synchronously and we can update the dataset to update the model.

Coming to the working of the user, the user is giving WhatsApp data to the user for getting the output and after that all the processing is done by the system itself.

The system is taking doing statistical Analysis on the data followed by sentimental analysis and language detection using the created models and lastly the result is shown to the user.

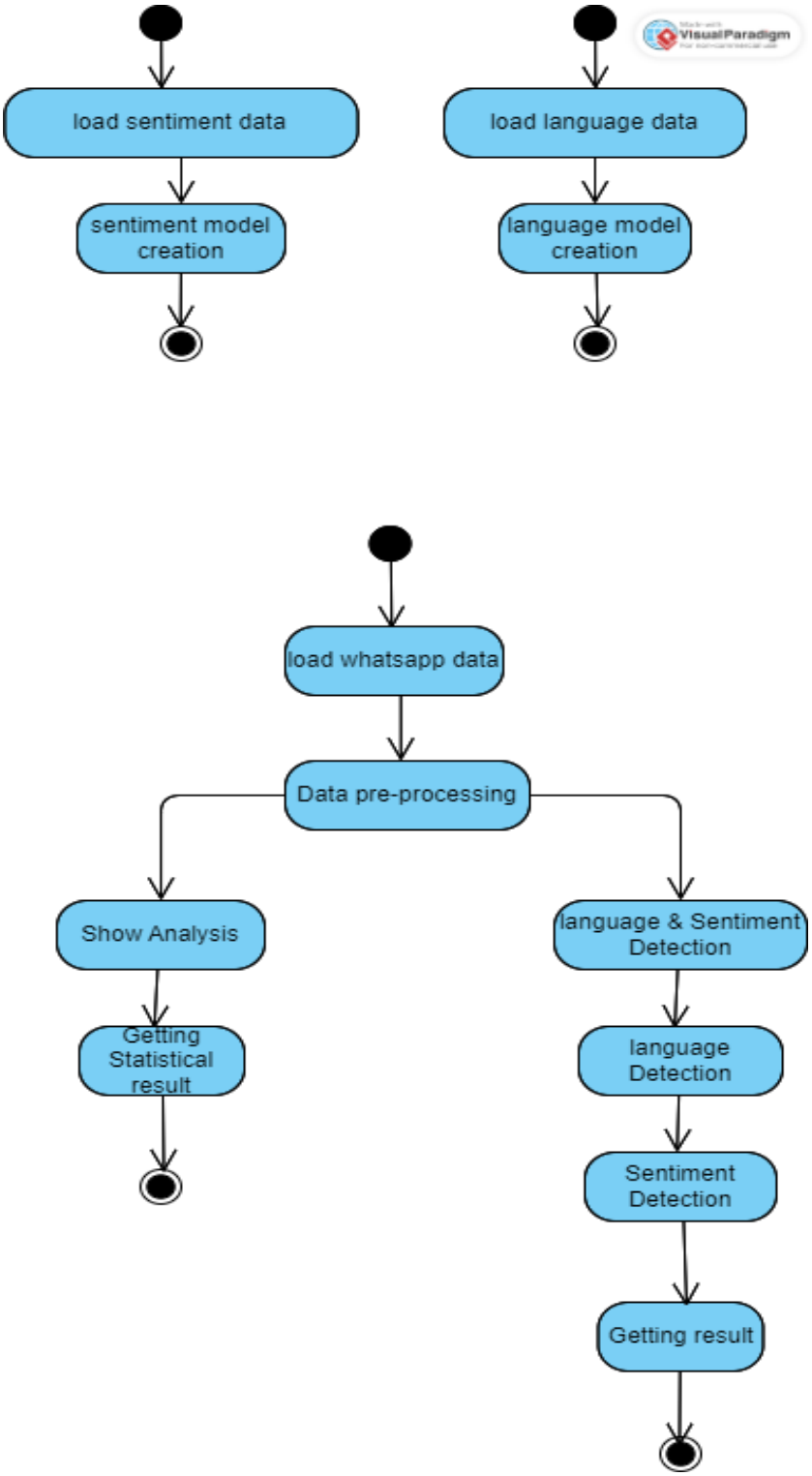


Fig 4.5: Activity Diagram

The above is the activity diagram of WhatsApp chat analysis and prediction application. It is the next step after the sequence diagram. Activity diagrams are used to show how different workflows in the system are constructed, how they start and the possibly many decision paths that can be taken from start to finish. The activity diagram is defined in a proper way showing the staring to the ending activities for the system So the model creation is easy. The datasets

collected are one after another, firstly sentimental analysis training dataset followed by language labelling data and lasting the WhatsApp chats. The application terminates after showing the result to the user.

4.6 Static Model

4.6.1 Component Diagrams

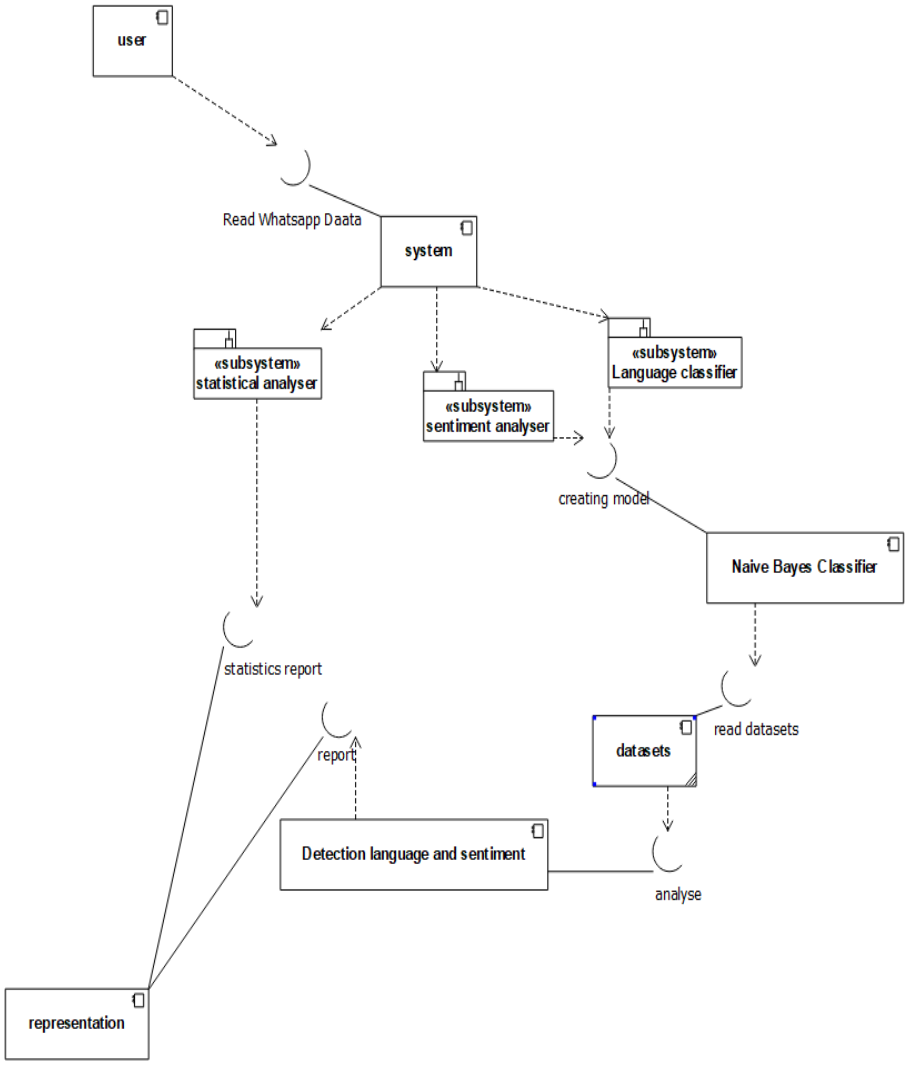


Fig 4.6: Component Diagram

Component diagrams are used to describe the physical artifacts of a system. This artifact includes files, executables, libraries, etc. In case of the proposed system the components are user, system, naïve bayes classifier, datasets, detection language and sentiment and lastly representation. System has three subsystem as statistical analyzer, sentiment analyzer and language classifier. All the systems are connected with proper interfaces and they are named to denote the working..

4.6.2 Deployment Diagrams

Deployment diagrams are used to visualize the hardware processors/ nodes/ devices of a system, the links of communication between them and the placement of software files on that hardware. The above describes the deployment diagram with the nodes as user system and the report. Deployment diagrams denotes the system is very easy to implement.

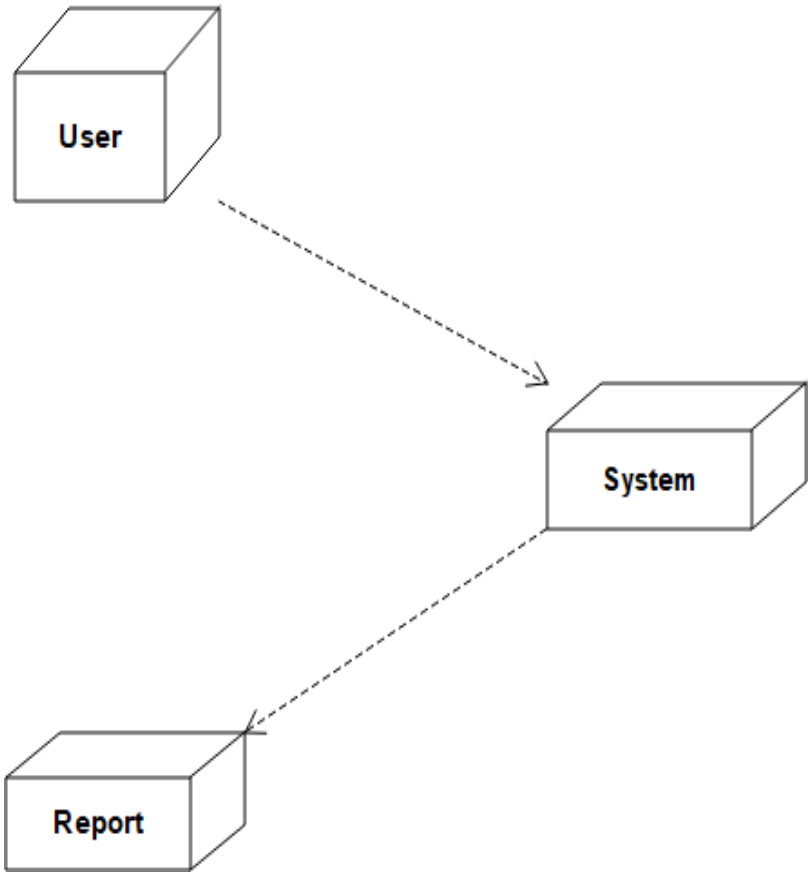


Fig 4.7: Deployment Diagram

CHAPTER-5
IMPLEMENTATION

5.IMPLEMENTATION

5.1 SOFTWARE ENVIRONMENT:

What is Python :-

The following are some Python facts.

Python is the most widely used multi-purpose, high-level programming language at the moment.Python supports both Object-Oriented and Procedural programming paradigms. Python programmers are typically smaller than those written in other programming languages such as Java.Programmers must type relatively little, and the language's indentationrequirement ensures that their code is always readable.

Python is used by almost all tech giants, including Google, Amazon, Facebook, Instagram, Dropbox, Uber, and others.

Python's greatest strength is its vast collection of standard libraries, which can be used for the following—

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQtetc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like OpenCV, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

Advantages of Python :-

Let’s see how Python dominates over other languages.

1. **Extensive Libraries**

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manually.

2. **Extensible**

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. **Embeddable**

Python is also embeddable, which is complementary to extensibility. You can include Python code in the source code of another language, such as C++. This enables us to add scripting capabilities to our other language code**.**

4. **Improved Productivity**

The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. **IOT Opportunities**

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

6. **Simple and Easy**

When working with Java, you may have to create a class to print 'Hello World'. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. **Readable**

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. This further aids the readability of the code.

8. **Object-Oriented**

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions
into one.

9. Free and Open-Source

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Advantages of Python Over Other Languages

1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

The reason it is not so famous despite the existence of Brython is that it isn't that secure.

3. Design Restrictions

As you know, Python is dynamically-typed. This means that you don't need to declare the type of variable while writing the code. It uses duck-typing. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple

No, we’re not kidding. Python’s simplicity can indeed be a problem. Take my example. I don’t do Java, I’m more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language**.**

History of Python : -

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde& Informatica). The best fulfillment of ABC was once to have an effect on the format of Python. Python used to be conceptualized in the late 1980s. Guido van Rossum labored that time in a mission at the CWI, known as Amoeba, a dispensed running system. In an interview with Bill Venners1, Guido van Rossum said: "In the early 1980s, I labored as an implementer on a crew constructing a language referred to as ABC at Centrum voorWiskundeen Informatica (CWI). I do not understand how properly humans understand ABC's have an impact on on Python. I strive to point out ABC's affect due to the fact I'm indebted to the whole thing I discovered throughout that undertaking and to the human beings who labored on it." Later on in the equal Interview, Guido van Rossum continued: "I remembered all my trip and some of my frustration with ABC. I determined to attempt to graph a easy scripting language that possessed some of ABC's higher properties, however barring its problems. So I began typing. I created a easy digital machine, a easy parser, and a easy runtime. I made my very own model of the a number ABC components that I liked. I created a simple syntax, used indentation for declaration grouping alternatively of curly braces or begin-end blocks, and developed a small range of effective statistics types: a hash desk (or dictionary, as we name it), a list, strings, and numbers."

5.2 LIBRARIES USED:

STREAMLIT: Streamlit is an open source app framework in Python language. It helps us create web apps for data science and machine learning in a short time. It is compatible with major Python libraries such as scikit-learn, Keras, PyTorch, SymPy(latex), NumPy, pandas, Matplotlib etc. With Streamlit, no callbacks are needed since widgets are treated as variables. Data caching simplifies and speeds up computation pipelines. Streamlit watches for changes on updates of the linked Git repository and the application will be deployed automatically in the shared link

PANDAS:

Pandas is an open-source Python Library supplying high-performance statistics manipulation and analysis device the use of its effective records structures. Python used to be majorly used for records munging and preparation. It had very little contribution toward statistics analysis. Pandas solved this problem. Using Pandas, we can accomplish 5 ordinary steps in the processing and evaluation of data, regardless of the starting place of statistics load, prepare, manipulate, model, and analyze. Python with Pandas is used in a extensive vary of fields inclusive of educational and business domains such as finance, economics, Statistics, analytics, etc.

Matplotlib:

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

Create publication quality plots

Make interactive figures that can zoom, pan, update.

Customize visual style and layout.

Export to many file formats.

Embed in JupyterLab and Graphical User Interfaces.

Use a rich array of third-party packages built on Matplotlib.

Seaborn:

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.

Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them

Sklearn:

Scikit-learn is a key library for the Python programming language that is typically used in machine learning projects. Scikit-learn is focused on machine learning tools including mathematical, statistical and general purpose algorithms that form the basis for many machine learning technologies. As a free tool, Scikit-learn is tremendously important in many different types of algorithm development for machine learning and related technologies.

Wordcloud:

Word clouds or tag clouds are graphical representations of word frequency that give greater prominence to words that appear more frequently in a source text. The larger the word in the visual the more common the word was in the document(s).

NLTK LIBRARY:

The Natural Language Toolkit (NLTK) is a platform used for building Python programs that work with human language data for applying in statistical natural language processing (NLP). It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning.

5.3 Naïve bayes classification:

Naive Bayes classification is a probabilistic machine learning algorithm that is commonly used for classification problems. It is based on Bayes' theorem and makes a "naive" assumption that the features (or attributes) of the input data are conditionally independent of each other given the class label.

The basic idea behind Naive Bayes is to calculate the probability of each class label given the input data, and then predict the class label with the highest probability. This is done by first calculating the prior probability of each class label, which is the proportion of training instances that belong to each class. Then, the likelihood of the input data given each class label is calculated by assuming that the features are independent given the class label. This is where the "naive" assumption comes in - it assumes that the features are independent, even though they may not be in reality.

Once the prior probabilities and likelihoods have been calculated, Bayes' theorem is used to calculate the posterior probability of each class label given the input data. The class label with the highest posterior probability is then predicted as the output of the algorithm.

Naive Bayes is a fast and simple algorithm that works well in many real-world situations, especially when the number of features is large compared to the number of training instances. It is also a popular algorithm for text classification tasks, where the features are often words or tokens.

The simple form of the calculation for Bayes Theorem is as follows:

P(A|B) = P(B|A) * P(A) / P(B)
Where the probability that we are interested in calculating P(A|B) is called the posterior probability and the marginal probability of the event P(A) is called the prior.

We can frame classification as a conditional classification problem with Bayes Theorem as follows:

P(yi | x1, x2, …, xn) = P(x1, x2, …, xn | yi) * P(yi) / P(x1, x2, …, xn)
The prior P(yi) is easy to estimate from a dataset, but the conditional probability of the observation based on the class P(x1, x2, …, xn | yi) is not feasible unless the number of examples is

extraordinarily large, e.g. large enough to effectively estimate the probability distribution for all different possible combinations of values.

As such, the direct application of Bayes Theorem also becomes intractable, especially as the number of variables or features (n) increases.

There are three main types of Naive Bayes classifiers: Gaussian Naive Bayes, Multinomial Naive Bayes, and Bernoulli Naive Bayes. Gaussian Naive Bayes is used for continuous input variables, while Multinomial Naive Bayes and Bernoulli Naive Bayes are used for discrete input variables, such as word counts in text classification tasks.

Overall, Naive Bayes is a simple and effective algorithm that is often used as a baseline for comparison with more complex classification algorithms.

5.3.1 Multinomial Naïve Bayes:

Multinomial Naive Bayes is a type of Naive Bayes classifier that is commonly used for text classification tasks where the input features are discrete counts, such as the frequency of words in a text document. It is particularly suited for classification tasks where each document can belong to one or more classes, as it can handle multi-class and multi-label classification problems.

In Multinomial Naive Bayes, the input data is represented as a vector of non-negative integers, where each element of the vector corresponds to the count of a particular feature (e.g. the count of a word) in the document. The assumption made by Multinomial Naive Bayes is that the features are generated from a multinomial distribution, hence the name "Multinomial" Naive Bayes.

To train a Multinomial Naive Bayes classifier, the algorithm estimates the prior probability of each class label, as well as the conditional probability of each feature given the class label. This is done by counting the occurrences of each feature in the training data, and using these counts to estimate the probabilities.

When making a prediction for a new document, the classifier calculates the probability of each class label given the features in the document, using Bayes' theorem. The class label with the highest probability is then predicted as the output of the classifier.

One limitation of Multinomial Naive Bayes is that it does not take into account the order or context of the words in a document, as it treats each feature independently. This means that it may not perform well for tasks that require an understanding of the semantic meaning of the text, such as sentiment analysis or topic modeling.

Despite this limitation, Multinomial Naive Bayes is a popular and effective algorithm for text classification tasks, especially when the number of features is large compared to the number of training instances. It is also relatively fast and simple to implement, making it a good choice for real-world applications.

5.3.2 Count Vectorizer:

In Naive Bayes classification, count vectorizer is used as a text preprocessing step to convert text data into a numeric representation that can be used as input for the Naive Bayes algorithm.

The Naive Bayes algorithm is a probabilistic classification algorithm that calculates the probability of each class for a given input data, and predicts the class with the highest probability. In text classification, the input data is usually a collection of documents, and the classes are the different categories or labels that the documents can be assigned to.

CountVectorizer is used to convert the text data into a matrix of token counts, where each row represents a document and each column represents a token in the vocabulary. The value in each cell represents the number of times the token appears in the document. This matrix is then used as input for the Naive Bayes algorithm.

Naive Bayes works by calculating the conditional probability of each class given the input data, using Bayes' theorem. The "Naive" assumption in Naive Bayes is that the features (tokens in this case) are conditionally independent given the class label. This allows the algorithm to estimate the probability of each class label based on the frequencies of the tokens in the input data.

In summary, CountVectorizer is used to convert the text data into a numerical representation that can be used as input for the Naive Bayes algorithm, which uses the frequencies of the tokens to estimate the probability of each class label for the input data.

5.3.3 Pipeline in multinomial Naïve Bayes :

In machine learning, a pipeline is a sequence of data processing steps that are chained together to form a complete workflow. A pipeline in Multinomial Naive Bayes usually includes a series of preprocessing steps, such as data cleaning, feature extraction, and normalization, followed by a Multinomial Naive Bayes classifier.

The purpose of a pipeline is to automate the entire process of model training and prediction, from data preprocessing to model evaluation. By combining multiple steps into a pipeline, the workflow can be streamlined and the likelihood of error is reduced.

Here is an example of a pipeline for Multinomial Naive Bayes:

1. Load the raw text data.
2. Preprocess the text data by removing stopwords, stemming or lemmatizing the words, and removing any special characters or punctuation.
3. Convert the preprocessed text data into a numerical representation using a vectorization techniquesuch as CountVectorizer or TF-IDF.
4. Split the data into training and testing sets.
5. Train a Multinomial Naive Bayes classifier on the training data using the numerical representationof the text data.
6. Evaluate the performance of the classifier on the testing data using metrics such as accuracy,precision, recall, and F1 score.
7. Use the trained classifier to make predictions on new, unseen data.

By using a pipeline in Multinomial Naive Bayes, the entire process from loading the raw data to making predictions on new data can be automated, making it easier to iterate on the model and improve its performance. Additionally, using a pipeline can help ensure that the same preprocessing steps are applied consistently to all data points, which can reduce the risk of overfitting and improve the generalization performance of the model.

5.3.4 Advantages of multinomial :

Multinomial Naive Bayes (MNB) is a popular algorithm for text classification tasks, such as language detection and sentiment analysis, because of several advantages it offers. Here are some of the advantages of using MNB for language detection and sentiment analysis of text:

Efficient and Scalable: MNB is a simple and efficient algorithm that can handle large datasets with high-dimensional feature spaces. It is well-suited for text classification tasks where the number of classes (i.e., languages or sentiment categories) is large, and the number of features (i.e., words or n-grams) is also large.

Handles Discrete Features: MNB is specifically designed to handle count-based data, such as word frequency counts or TF-IDF values, which are commonly used in language detection and sentiment analysis tasks. The algorithm assumes that the frequency of each feature (i.e., word or n-gram) is generated from a multinomial distribution, which makes it well-suited for text classification tasks.

Naive Bayes Assumption: MNB assumes that the features (i.e., words or n-grams) are conditionally independent given the class label (i.e., language or sentiment category), which means that it is able to handle high-dimensional feature spaces without suffering from the "curse of dimensionality." This assumption is often violated in practice, but MNB can still perform well in many cases.

Good Performance: MNB has been shown to perform well on a variety of text classification tasks, including language detection and sentiment analysis. It is often used as a baseline algorithm for comparison with more complex models, as it can achieve good accuracy with relatively little training data.

Overall, Multinomial Naive Bayes is a simple, efficient, and effective algorithm for text classification tasks, particularly when dealing with large datasets and high-dimensional feature spaces. Its ability to handle count-based data and its naivety with regards to the conditional independence assumption make it a good choice for many real-world applications in language detection and sentiment analysis of text.

5.4 Source Code:

Language Detection Model:

Language Detection Model Creation

- using naive bayes classification

```
#import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import re
import joblib
import warnings
warnings.filterwarnings("ignore")
```

[1] Python

```
#pip install -U scikit-learn
import sklearn
sklearn.__version__
#1.0.2
```

[10] Python

```
'1.0.2'
```

```
data=pd.read_csv("LanguageDetection.csv")
data.head(10)
```

[11] Python

	Text	Language
0	Nature, in the broadest sense, is the natural...	English
1	"Nature" can refer to the phenomena of the phy...	English
2	The study of nature is a large, if not the onl...	English
3	Although humans are part of nature, human acti...	English
4	[1] The word nature is borrowed from the Old F...	English
5	[2] In ancient philosophy, natura is mostly us...	English
6	[3][4] \nThe concept of nature as a whole, the...	English
7	During the advent of modern scientific method ...	English
8	[5][6] With the Industrial revolution, nature ...	English
9	However, a vitalist vision of nature, closer t...	English

```
data.shape
```

[12] Python

```
(10337, 2)
```

```
data.info()
```

[36] Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10337 entries, 0 to 10336
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Text      10337 non-null  object
 1   Language  10337 non-null  object
dtypes: object(2)
memory usage: 161.6+ KB
```

```
[13] data.isnull().sum()
```
Python

```
Text        0
Language    0
dtype: int64
```

```
[14] len(data[data.duplicated()])
```
Python

```
66
```

```
[15] data.drop_duplicates(inplace=True)
```
Python

```
[16] data.shape
```
Python

```
(10271, 2)
```

```
[17] data["Language"].nunique()
```
Python

```
17
```

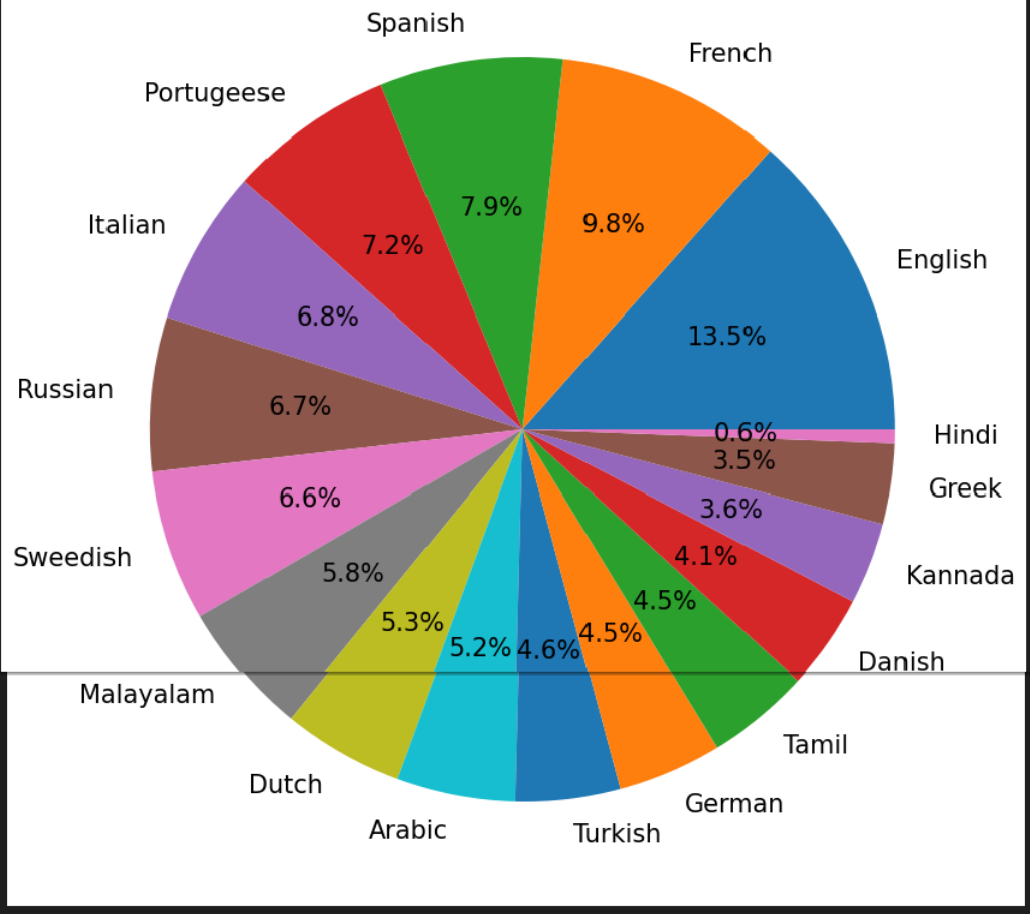
```
[18] data["Language"].value_counts()
```
Python

```
English       1382
French        1007
Spanish        816
Portugeese     736
Italian        694
Russian        688
Sweedish       673
Malayalam      591
Dutch          542
Arabic         532
Turkish        471
German         465
Tamil          464
Danish         424
Kannada        366
Greek          358
Hindi           62
Name: Language, dtype: int64
```

```
language = data["Language"].value_counts().reset_index()

plt.figure(figsize=(10,10))
labels= language['index']

plt.pie(language["Language"], labels= labels, autopct='%.1f%%', textprops={'fontsize': 15})
plt.show()
```
[19] Python




```
data1 = data.copy()
data1["cleaned_Text"] = ""
data1
```

[20] Python

	Text	Language	cleaned_Text
0	Nature, in the broadest sense, is the natural...	English	
1	"Nature" can refer to the phenomena of the phy...	English	
2	The study of nature is a large, if not the onl...	English	
3	Although humans are part of nature, human acti...	English	
4	[1] The word nature is borrowed from the Old F...	English	
...	...	...	...
10332	ನಿಮ್ಮ ತಪ್ಪು ಏನು ಬಂದಿದೆಯೆಂದರೆ ಆ ದಿನದಿಂದ ನಿಮಗೆ ಒ...	Kannada	
10333	ನಾರ್ಸಿಸಾ ತಾನು ಮೊದಲಿಗೆ ಹೆಣಗಾಡುತ್ತಿದ್ದ ಮಾರ್ಗಗಳನ್...	Kannada	
10334	ಹೇಗೆ ' ನಾರ್ಸಿಸಿಸಮ್ ಈಗ ಮರಿಯನ್ ಅವರಿಗೆ ಸಂಭವಿಸಿದ ಎ...	Kannada	
10335	ಅವಳು ಈಗ ಹೆಚ್ಚು ಚಿನ್ನದ ಬೈಡ್ ಬಯಸುವುದಿಲ್ಲ ಎಂದು ...	Kannada	
10336	ಟೆರ್ರಿ ನೀವು ನಿಜವಾಗಿಯೂ ಆ ದೇವದೂತನಂತೆ ಸ್ವಲ್ಪ ಕಾಣು...	Kannada	

10271 rows × 3 columns

```python
def clean_func(Text):
    Text = re.sub(r'[\([{})\]!@#$,"%^*?:;~`0-9]', ' ', Text)   # removing the symbols and numbers
    Text = Text.lower()   # converting the text to lower case
    Text = re.sub('#\S+', '', Text)  # remove hashtags

    return Text
```

[21] Python

```python
data1["cleaned_Text"] = data1["Text"].apply(lambda x:clean_func(x))
data1
```

[22] Python

	Text	Language	cleaned_Text
0	Nature, in the broadest sense, is the natural...	English	nature in the broadest sense is the natural...
1	"Nature" can refer to the phenomena of the phy...	English	nature can refer to the phenomena of the phy...
2	The study of nature is a large, if not the onl...	English	the study of nature is a large if not the onl...
3	Although humans are part of nature, human acti...	English	although humans are part of nature human acti...
4	[1] The word nature is borrowed from the Old F...	English	the word nature is borrowed from the old f...
...	...	...	...
10332	ನಿಮ್ಮ ತಪ್ಪು ಏನು ಬಂದಿದೆಯೆಂದರೆ ಆ ದಿನದಿಂದ ನಿಮಗೆ ಒ....	Kannada	ನಿಮ್ಮ ತಪ್ಪು ಏನು ಬಂದಿದೆಯೆಂದರೆ ಆ ದಿನದಿಂದ ನಿಮಗೆ ಒ....
10333	ನಾರ್ಸಿಸಾ ತಾನು ಮೊದಲಿಗೆ ಹೆಣಗಾಡುತ್ತಿದ್ದ ಮಾರ್ಗಗಳನ್...	Kannada	ನಾರ್ಸಿಸಾ ತಾನು ಮೊದಲಿಗೆ ಹೆಣಗಾಡುತ್ತಿದ್ದ ಮಾರ್ಗಗಳನ್...
10334	ಹೇಗೆ ' ನಾರ್ಸಿಸಿಸಮ್ ಈಗ ಮರಿಯನ್ ಅವರಿಗೆ ಸಂಭವಿಸಿದ ಎ...	Kannada	ಹೇಗೆ ' ನಾರ್ಸಿಸಿಸಮ್ ಈಗ ಮರಿಯನ್ ಅವರಿಗೆ ಸಂಭವಿಸಿದ ಎ...
10335	ಅವಳು ಈಗ ಹೆಚ್ಚು ಚಿನ್ನದ ಬ್ರೆಡ್ ಬಯಸುವುದಿಲ್ಲ ಎಂದು ...	Kannada	ಅವಳು ಈಗ ಹೆಚ್ಚು ಚಿನ್ನದ ಬ್ರೆಡ್ ಬಯಸುವುದಿಲ್ಲ ಎಂದು ...
10336	ಟೆರ್ರಿ ನೀವು ನಿಜವಾಗಿಯೂ ಆ ದೇವದೂತನಂತೆ ಸ್ವಲ್ಪ ಕಾಣು...	Kannada	ಟೆರ್ರಿ ನೀವು ನಿಜವಾಗಿಯೂ ಆ ದೇವದೂತನಂತೆ ಸ್ವಲ್ಪ ಕಾಣು...

10271 rows × 3 columns

```python
X = data1["cleaned_Text"]
y = data1["Language"]
```

[23] Python

```python
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
```

[24] Python

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

[25] Python

```python
pipe_lr1 = Pipeline(steps=[('cv',CountVectorizer()),('nb',MultinomialNB())])
```

[26] Python

```python
pipe_lr1.fit(X_train,y_train)
```

[27] Python

```
Pipeline(steps=[('cv', CountVectorizer()), ('nb', MultinomialNB())])
```

```python
pipe_lr1
```

[28] Python

```
Pipeline(steps=[('cv', CountVectorizer()), ('nb', MultinomialNB())])
```

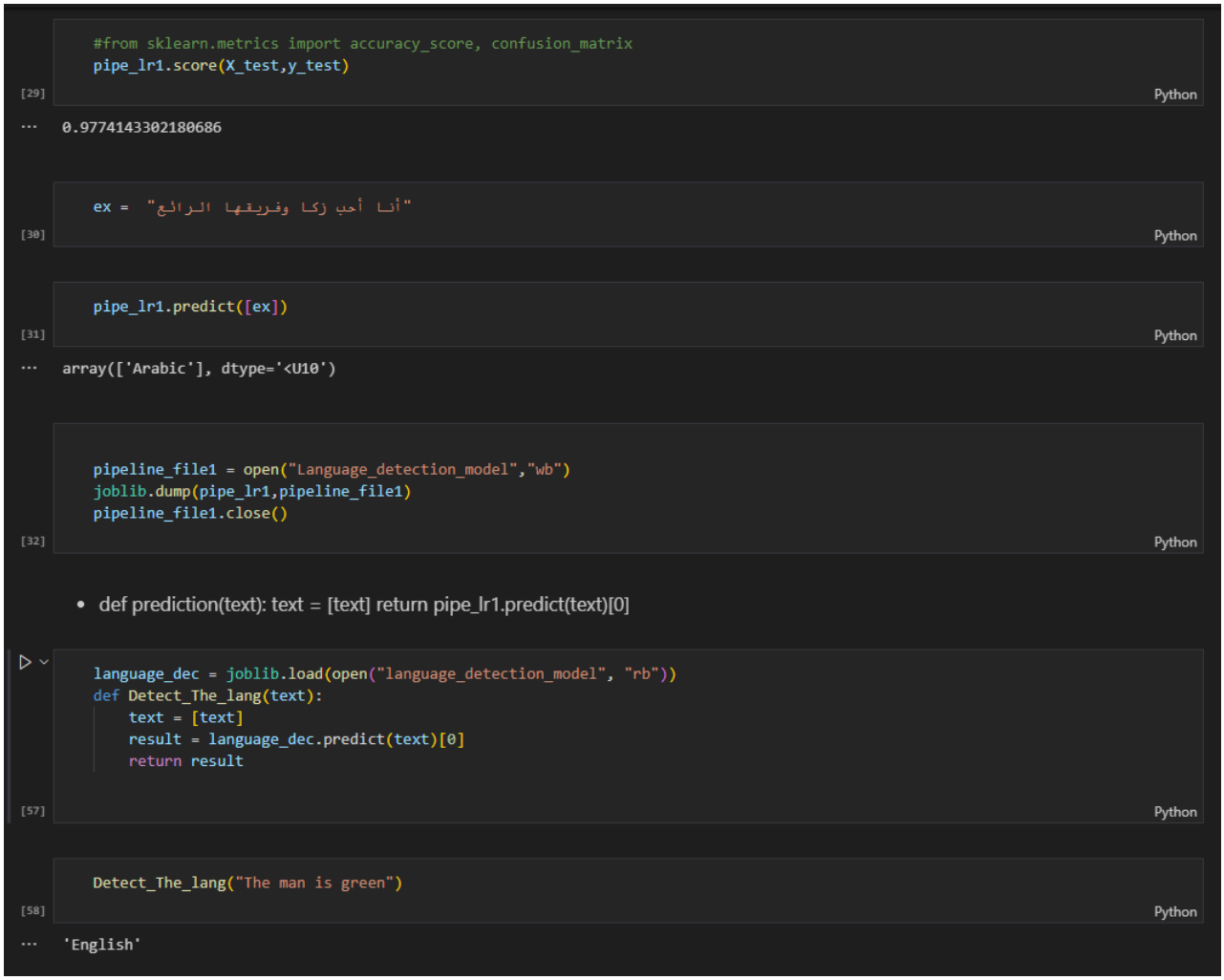


Fig 5.1: Language Detection Model

Sentiment Analysis Model:

Sentiment Analysis Model Building:

- Text Classifier using Naive bayes classification
- model is saved in pickle file using joblib

```
[1]
#pip install neattext
```
Python

```
[3]
import pandas as pd
import numpy as np


import seaborn as sns

# Load Text Cleaning Pkgs
import neattext.functions as nfx

import joblib
```
Python

```
[4]
# Load ML Pkgs

# Estimators
from sklearn.naive_bayes import MultinomialNB

# Transformers
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix

# Build Pipeline
from sklearn.pipeline import Pipeline
```
Python

```
[5]
# Load Dataset
df_s = pd.read_csv("emotion_dataset_raw.csv")
```
Python

```
[6]
#view the dataset
df_s.head()
```
Python

	Emotion	Text
0	neutral	Why ?
1	joy	Sage Act upgrade on my to do list for tommorow.
2	sadness	ON THE WAY TO MY HOMEGIRL BABY FUNERAL!!! MAN ...
3	joy	Such an eye ! The true hazel eye-and so brill...
4	joy	@Iluvmiasantos ugh babe.. hugggzzz for u .! b...

+ Code + Markdown

```
[7]
#get the information
df_s.info()
```
Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34792 entries, 0 to 34791
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Emotion  34792 non-null  object
 1   Text     34792 non-null  object
dtypes: object(2)
memory usage: 543.8+ KB
```

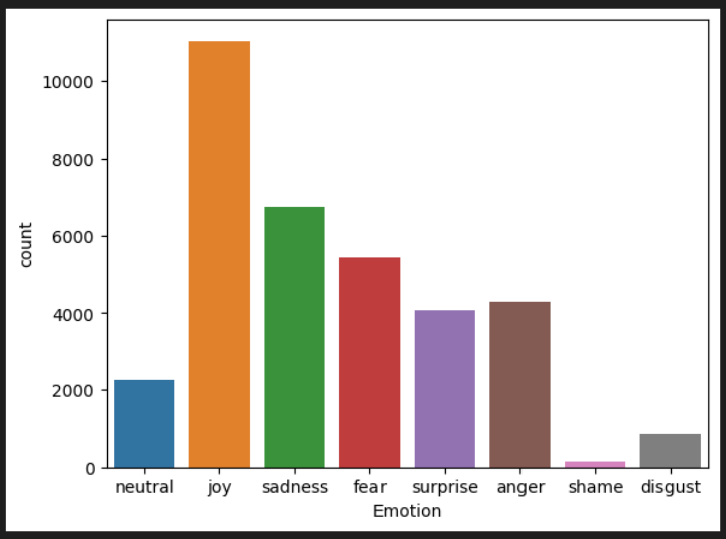
```
[8]
# Value Counts
df_s['Emotion'].value_counts()
```
Python

```
joy         11045
sadness      6722
fear         5410
anger        4297
surprise     4062
neutral      2254
disgust       856
shame         146
Name: Emotion, dtype: int64
```

```
# Plot
sns.countplot(x='Emotion',data=df_s)
```

[9] Python

```
<AxesSubplot:xlabel='Emotion', ylabel='count'>
```




```
# Data Cleaning
dir(nfx)
```

[10] Python

```
Output exceeds the size limit. Open the full output data in a text editor
['BTC_ADDRESS_REGEX',
 'CURRENCY_REGEX',
 'CURRENCY_SYMB_REGEX',
 'Counter',
 'DATE_REGEX',
 'EMAIL_REGEX',
 'EMOJI_REGEX',
 'HASTAG_REGEX',
 'MASTERCard_REGEX',
 'MD5_SHA_REGEX',
 'MOST_COMMON_PUNCT_REGEX',
 'NUMBERS_REGEX',
 'PHONE_REGEX',
 'PoBOX_REGEX',
 'SPECIAL_CHARACTERS_REGEX',
 'STOPWORDS',
 'STOPWORDS_de',
 'STOPWORDS_en',
 'STOPWORDS_es',
 'STOPWORDS_fr',
 'STOPWORDS_ru',
 'STOPWORDS_yo',
 'STREET_ADDRESS_REGEX',
 'TextFrame',
 'URL_PATTERN',
...
 'term_freq',
 'to_txt',
 'unicodedata',
 'word_freq',
```

```
'word_length_freq']
```

```
# remove anomalies

# User handles
df_s['Clean_Text'] = df_s['Text'].apply(nfx.remove_userhandles)
```
[11] Python

```
# Stopwords
df_s['Clean_Text'] = df_s['Text'].apply(nfx.remove_stopwords)
```
[12] Python

```
df_s
```
[13] Python

	Emotion	Text	Clean_Text
0	neutral	Why ?	?
1	joy	Sage Act upgrade on my to do list for tommorow.	Sage Act upgrade list tommorow.
2	sadness	ON THE WAY TO MY HOMEGIRL BABY FUNERAL!!! MAN ...	WAY HOMEGIRL BABY FUNERAL!!! MAN HATE FUNERALS...
3	joy	Such an eye ! The true hazel eye-and so brill...	eye ! true hazel eye-and brilliant ! Regular f...
4	joy	@lluvmiasantos ugh babe.. hugggzzz for u .! b...	@lluvmiasantos ugh babe.. hugggzzz u .! babe n...
...	...	...	...
34787	surprise	@MichelGW have you gift! Hope you like it! It'...	@MichelGW gift! Hope like it! hand wear ! It'l...
34788	joy	The world didnt give it to me..so the world MO...	world didnt me..so world DEFINITELY cnt away!!!
34789	anger	A man robbed me today .	man robbed today .
34790	fear	Youu call it JEALOUSY, I call it of #Losing YO...	Youu JEALOUSY, #Losing YOU...
34791	sadness	I think about you baby, and I dream about you ...	think baby, dream time

34792 rows × 3 columns

```
# Features & Labels
Xfeatures_s = df_s['Clean_Text']
ylabels_s = df_s['Emotion']
```
[14] Python

```
#  Split Data
xs_train,xs_test,ys_train,ys_test = train_test_split(Xfeatures_s,ylabels_s,test_size=0.3,random_state=42)
```
[15] Python

```
# Naive Bayes Classification pipeline
pipe_lr = Pipeline(steps=[('cv',CountVectorizer()),('nb',MultinomialNB())])
```
[16] Python

```
# Train and Fit Data
pipe_lr.fit(xs_train,ys_train)
```
[17] Python

```
Pipeline(steps=[('cv', CountVectorizer()), ('nb', MultinomialNB())])
```

```
pipe_lr
```
[18] Python

```
Pipeline(steps=[('cv', CountVectorizer()), ('nb', MultinomialNB())])
```

```
# Check Accuracy
pipe_lr.score(xs_test,ys_test)
```
[19] Python

```
0.5713738264035256
```

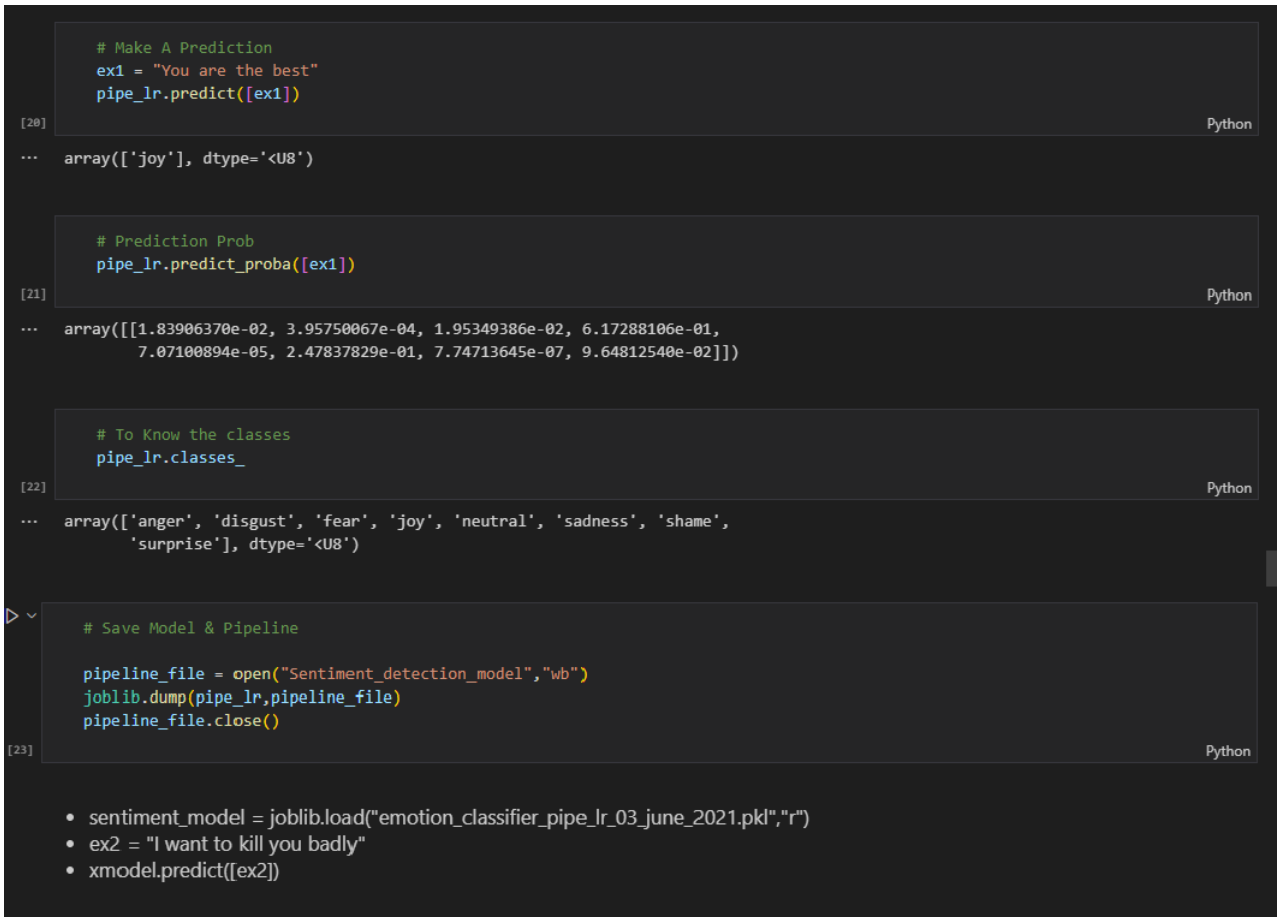


Fig 5.2: Sentiment Detection Model

Preprocessor.py:

```
import re
import pandas as pd
import Detection_Function


def pre_process(data):
    pattern = '\d{1,2}/\d{1,2}/\d{2,4},\s\d{1,2}:\d{2}\s[APap][mM]\s-\s'
    messages = re.split(pattern, data)[1:]
    dates = re.findall(pattern, data)

    df = pd.DataFrame({'user_message': messages, 'message_date': dates})
    df['message_date'] = pd.to_datetime(df['message_date'], format='%d/%m/%Y, %I:%M %p - ')
    df.rename(columns={'message_date': 'date'}, inplace=True)

    users = []
    messages = []
    for message in df['user_message']:
        entry = re.split('([\w\W]+?):\s', message)
        if entry[1:]:
            users.append(entry[1])
            messages.append(entry[2])
        else:
            users.append('group_notification')
            messages.append(entry[0])

    df['user'] = users
    df['message'] = messages

    df.drop(columns=['user_message'], inplace=True)

    df['only_date'] = df['date'].dt.date
    df['year'] = df['date'].dt.year
    df['month_num'] = df['date'].dt.month
    df['month'] = df['date'].dt.month_name()
    df['day'] = df['date'].dt.day
    df['day_name'] = df['date'].dt.day_name()
    df['hour'] = df['date'].dt.hour
    df['minute'] = df['date'].dt.minute

    period = []
    for hour in df[['day_name', 'hour']]['hour']:
        if hour == 23:
            period.append(str(hour) + "-" + str('00'))
        elif hour == 0:
            period.append(str('00') + "-" + str(hour + 1))
        else:
            period.append(str(hour) + "-" + str(hour + 1))

    df['period'] = period
```

```
    # without lambda pipeline attributes can't be called
    #df['language'] = df['message'].apply(lambda x: Detection_Function.Detect_The_lang(x))
    #df['sentiment'] = df['message'].apply(lambda x: Detection_Function.Detect_The_senti(x))

    return df
```

helper.py:

```
import pandas as pd
from urlextract import URLExtract
from wordcloud import WordCloud
from collections import Counter
import emoji
import Detection_Function

# from nltk.sentiment.vader import SentimentIntensityAnalyzer
# import nltk

# nltk.downloader.download('vader_lexicon')

# sentiments = SentimentIntensityAnalyzer()

extract = URLExtract()


def fetch_stats(selected_user, df):
    if selected_user != 'Overall':
        df = df[df['user'] == selected_user]

    num_message = df.shape[0]

    words = []
    for message in df['message']:
        words.extend(message.split())

    media_count = df[df['message'] == '<Media omitted>'].shape[0]
    delecetd_message_count = df[df['message'] == 'This message was deleted'].shape[0]

    urls = []
    for message in df['message']:
        urls.extend(extract.find_urls(message))

    return num_message, len(words), len(urls), media_count, delecetd_message_count


def most_busy_user(df):
```

```
    x = df['user'].value_counts().head()
    dfx = round((df['user'].value_counts() / df.shape[0]) * 100, 2).reset_index().rename(
        columns={'index': 'name', 'user': 'percentage'})
    return x, dfx


def create_word_cloud(selected_user, df):
    f = open('stop_hinglish.txt', 'r')
    stop_words = f.read()

    if selected_user != 'Overall':
        df = df[df['user'] == selected_user]

    temp = df[df['user'] != 'group_notification']
    # temp = temp[temp['message'] != '<media omitted>\n']
    temp = temp[temp['message'] != '<Media omitted>']
    temp = temp[temp['message'] != 'This message was deteted\n']

    def remove_stop_words(message):
        y = []
        for word in message.lower().split():
            if word not in stop_words:
                y.append(word)
        return " ".join(y)

    if selected_user != 'Overall':
        df = df[df['user'] == selected_user]

    wc = WordCloud(width=500, height=500, min_font_size=5, background_color="white")
    temp['message'] = temp['message'].apply(remove_stop_words)
    df_wc = wc.generate(df['message'].str.cat(sep=" "))
    return df_wc


def most_common_words(selected_user, df):

    f = open('stop_hinglish.txt', 'r')
    stop_words = f.read()

    if selected_user != 'Overall':
        df = df[df['user'] == selected_user]

    temp = df[df['user'] != 'group_notification']
    # temp = temp[temp['message'] != '<media omitted>\n']
    temp = temp[temp['message'] != '<Media omitted>\n']
    temp = temp[temp['message'] != 'This message was deteted\n']

    words = []

    for message in temp['message']:
        for word in message.lower().split():
```

```
            if word not in stop_words:
                words.append(word)

    word_df = pd.DataFrame(Counter(words).most_common(20))

    return word_df


def emoji_analysis(selected_user, df):
    if selected_user != 'Overall':
        df = df[df['user'] == selected_user]

    emojis = []
    for message in df['message']:
        x = emoji.distinct_emoji_list(message)
        emojis.extend([c for c in x])

    emoji_df = pd.DataFrame(Counter(emojis).most_common(len(Counter(emojis))))

    return emoji_df


def monthly_timeline(selected_user, df):
    if selected_user != 'Overall':
        df = df[df['user'] == selected_user]

    timeline = df.groupby(['year', 'month_num', 'month']).count()['message'].reset_index()

    time = []
    for i in range(timeline.shape[0]):
        time.append(timeline['month'][i] + "-" + str(timeline['year'][i]))

    timeline['time'] = time

    return timeline


def daily_timeline(selected_user, df):
    if selected_user != 'Overall':
        df = df[df['user'] == selected_user]

    dailytimeline = df.groupby('only_date').count()['message'].reset_index()

    return dailytimeline


def week_activity_map(selected_user, df):
    if selected_user != 'Overall':
        df = df[df['user'] == selected_user]

    return df['day_name'].value_counts()
```

```
def month_activity_map(selected_user, df):
    if selected_user != 'Overall':
        df = df[df['user'] == selected_user]

    return df['month'].value_counts()


def activity_heatmap(selected_user, df):
    if selected_user != 'Overall':
        df = df[df['user'] == selected_user]

    user_heatmap = df.pivot_table(index='day_name', columns='period', values='message',
aggfunc='count').fillna(0)

    return user_heatmap


'''
def sentiment_analyse(selected_user, df):
    if selected_user != 'Overall':
        df = df[df['user'] == selected_user]

    temp = df[df['user'] != 'group_notification']
    temp = temp[temp['message'] != '<media omitted>\n']
    temp = temp[temp['message'] != '<Media omitted>\n']

    sen_df = pd.DataFrame(df, columns=['date', 'user', 'message'])
    sen_df['positive'] = [sentiments.polarity_scores(i)['pos'] for i in sen_df['message']]
    sen_df['negative'] = [sentiments.polarity_scores(i)['neg'] for i in sen_df['message']]
    sen_df['neutral'] = [sentiments.polarity_scores(i)['neu'] for i in sen_df['message']]

    return sen_df

'''


def message_language_count(selected_user, df):
    if selected_user != 'Overall':
        df = df[df['user'] == selected_user]
    df['language'] = df['message'].apply(lambda x: Detection_Function.Detect_The_lang(x))

    temp = df[df['user'] != 'group_notification']
    # temp = temp[temp['message'] != '<media omitted>\n']
    temp = temp[temp['message'] != '<Media omitted>\n']
    temp = temp[temp['message'] != 'This message was deteted\n']
    temp['message'] = temp['message'].str.replace('http[s]?://(?:[a-zA-Z]|[0-9]|[$-
_@.&+]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', 'This is an url\n')
    temp = temp[temp['message'] != 'This is an url\n']
```

```
    df_eng = temp[temp['language']=='English']
    df_non_eng = temp[temp['language'] != 'English']

    eng_count = df_eng.shape[0]
    non_eng_count = df_non_eng.shape[0]

    return df_eng, df_non_eng, eng_count, non_eng_count


def message_sentiment_count(selected_user, df):
    if selected_user != 'Overall':
        df = df[df['user'] == selected_user]
    df['sentiment'] = df['message'].apply(lambda x: Detection_Function.Detect_The_senti(x))

    temp = df[df['user'] != 'group_notification']
    # temp = temp[temp['message'] != '<media omitted>\n']
    temp = temp[temp['message'] != '<Media omitted>\n']
    temp = temp[temp['message'] != 'This message was deteted\n']
    '''
    temp['message'] = temp['message'].str.replace(
        'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', 'This is an
url\n')
    temp = temp[temp['message'] != 'This is an url\n']
    '''

    return temp


def seeSentiment(selected_user,df):
    if selected_user != 'Overall':
        df = df[df['user'] == selected_user]

    x = df['sentiment'].value_counts().head()

    dfx = round((df['sentiment'].value_counts() / df.shape[0]) * 100, 2)

    return x, dfx
```

Detection_Function.py:

```
import joblib
from sklearn.pipeline import Pipeline


language_dec = joblib.load(open("language_detection_model", "rb"))
sentiment_dec = joblib.load(open("Sentiment_detection_model", 'rb'))


def Detect_The_lang(text):
    text = [text]
    result = language_dec.predict(text)[0]
    return result


def Detect_The_senti(text):
    text = [text]
    result = sentiment_dec.predict(text)[0]
    return result


#print(Detect_The_senti("The man is green"))
#print(Detect_The_lang(ex))#ex = "ffffffff ffffffff ffff ffff ffff"

'''
        # Langugae Detection Part
        st.title("Language Detection of WhatsApp Chat: ")
        eng, neng, c_eng, c_ncng = helper.message_language_count(selected_user, df)
        com, con = st.columns(2)

        with com:
            st.header("English message Count")
            st.title(num_messages)

        with con:
            st.header("Non English message Count")
            st.title(no_words)
    '''
```

App.py:

```
import matplotlib.pyplot as plt
# import pandas as pd
import seaborn as sns
import streamlit as st

import helper
import preprocessor
import Detection_Function

st.sidebar.title("WhatsApp Chat Analyzer and Prediction Application")

# accu = Language_detection.model_creation()
# st.title("Accuracy of Language Detection model is"+str(accu))
timeFormat = st.radio(
    "What\'s your chat's time format:",
    ('12h', '24h'))
uploaded_file = st.sidebar.file_uploader("Choose a file")
if uploaded_file is not None:
    bytes_data = uploaded_file.getvalue()
    data = bytes_data.decode("utf-8")
    # st.text(data) to view the inputted data
    df = preprocessor.pre_process(data,timeFormat)
    df.index += 1

    st.title("Created DataFrame is: ")
    st.dataframe(df)

    # fetch unique user
    if len(df.axes[0]!=0):
        user_list = df['user'].unique().tolist()
        user_list.remove('group_notification')
        user_list.sort()
        user_list.insert(0, "Overall")

        selected_user = st.sidebar.selectbox("Show analysis with respect to: ", user_list)

        if st.sidebar.button("Show Analysis"):

            num_messages, no_words, no_urls, media_count, deleted_count =
helper.fetch_stats(selected_user, df)
            col1, col2, col3 = st.columns(3)

            with col1:
                st.header("Total Messages")
                st.title(num_messages)

            with col2:
                st.header("Total Words")
```

```
    st.title(no_words)

with col3:
    st.header("Total URLs")
    st.title(no_urls)
# not working

# monthly timeline
st.title("Monthly Timeline")
timeline = helper.monthly_timeline(selected_user, df)
fig, ax = plt.subplots()
ax.plot(timeline['time'], timeline['message'], color='green')
plt.xticks(rotation='vertical')
st.pyplot(fig)

# daily timeline
st.title("Daily Timeline")
daily_timeline = helper.daily_timeline(selected_user, df)
fig, ax = plt.subplots()
ax.plot(daily_timeline['only_date'], daily_timeline['message'], color='black')
plt.xticks(rotation='vertical')
st.pyplot(fig)

# activity map
st.title('Activity Map')
col1, col2 = st.columns(2)

with col1:
    st.header("Most busy day")
    busy_day = helper.week_activity_map(selected_user, df)
    fig, ax = plt.subplots()
    ax.bar(busy_day.index, busy_day.values, color='purple')
    plt.xticks(rotation='vertical')
    st.pyplot(fig)

with col2:
    st.header("Most busy month")
    busy_month = helper.month_activity_map(selected_user, df)
    fig, ax = plt.subplots()
    ax.bar(busy_month.index, busy_month.values, color='orange')
    plt.xticks(rotation='vertical')
    st.pyplot(fig)

st.title("Weekly Activity Map")
user_heatmap = helper.activity_heatmap(selected_user, df)
fig, ax = plt.subplots()
ax = sns.heatmap(user_heatmap)
st.pyplot(fig)

# group level
if selected_user == "Overall":
```

```
            st.title("Most Busy User: ")
            x, y = helper.most_busy_user(df)
            fig, ax = plt.subplots()

            cola, colb = st.columns(2)

            with cola:
                y.index += 1
                st.dataframe(y)

            with colb:
                ax.bar(x.index, x.values, color='Red')
                plt.xticks(rotation='vertical')
                st.pyplot(fig)

        # word_cloud
        st.title("WordCloud")
        wordcloud_image = helper.create_word_cloud(selected_user, df)
        fig, ax = plt.subplots()
        ax.imshow(wordcloud_image)
        st.pyplot(fig)

        # most_common_words
        st.title("Most Frequent Words: ")
        most_common_word = helper.most_common_words(selected_user, df)
        fig, ax = plt.subplots()
        ax.barh(most_common_word[0], most_common_word[1])
        plt.xticks(rotation='vertical')
        st.pyplot(fig)
        # st.dataframe(most_common_word)

        # emoji analysis
        st.title("Most Common Emojis: ")
        emoji_df = helper.emoji_analysis(selected_user, df)


        #fig, ax = plt.subplots()
        #ax.pie(emoji_df['Frequency'], labels=emoji_df['Emoji'])
        #st.pyplot(fig)
        st.dataframe(emoji_df)

    if st.sidebar.button("Language & Sentiment Detection:"):
        eng_dataframe, noneng_dataframe, eng_count_message, non_eng_count_message =
helper.message_language_count(selected_user,df)

        if(noneng_dataframe.shape[0]!=0):
            st.title("Messages other than english are shown below: ")
            if(noneng_dataframe.shape):
                noneng_dataframe.index += 1
                st.dataframe(noneng_dataframe)
```

```
colx, coly = st.columns(2)

with colx:
   st.header("Total number of english messages: ");
   st.title(eng_count_message)

   st.header("Total Number of non-english messages: ");
   st.title(non_eng_count_message)

if selected_user == "Overall":
   st.title("User using non-english languages mostly: ")
   x, y = helper.most_busy_user(noneng_dataframe)
   y.index += 1
   fig, ax = plt.subplots()

   colm, coln = st.columns(2)

   with colm:
      st.dataframe(y)

   with coln:
      ax.bar(x.index, x.values, color='Red')
      plt.xticks(rotation='vertical')
      st.pyplot(fig)

st.title("Sentiment Analysis Results: ")
st.title("Message dataset with corresponding sentiments: ")

sentimentDataset = helper.message_sentiment_count(selected_user,df)
st.dataframe(sentimentDataset)
m, n = helper.seeSentiment(selected_user,sentimentDataset)
fig, ax = plt.subplots()

colx, coly = st.columns(2)

with colx:
      st.title("Messages parcentage based on sentiment type:")
      st.dataframe(n)

with coly:
   st.title("Graphical sentiment analysis: ")
   ax.bar(m.index, m.values, color='green')
   plt.xticks(rotation='vertical')
   st.pyplot(fig)


joy_data,sadness_data,fear_data,anger_data,surprise_data,neutral_data,disgust_data,shame_data
= helper.word_in_emotion(selected_user,sentimentDataset)
```

```
        if(joy_data.shape[0]!=0):
            st.title("WordCloud Joy")
            wordcloud_image_joy = helper.create_word_cloud(selected_user, joy_data)
            fig, ax = plt.subplots()
            ax.imshow(wordcloud_image_joy)
            st.pyplot(fig)
        if(sadness_data.shape[0]!=0):
            st.title("WordCloud sadness")
            wordcloud_image_sadness = helper.create_word_cloud(selected_user, sadness_data)
            fig, ax = plt.subplots()
            ax.imshow(wordcloud_image_sadness)
            st.pyplot(fig)
        if(fear_data.shape[0]!=0):
            st.title("WordCloud fear")
            wordcloud_image_fear = helper.create_word_cloud(selected_user, fear_data)
            fig, ax = plt.subplots()
            ax.imshow(wordcloud_image_fear)
            st.pyplot(fig)
        if(anger_data.shape[0]!=0):
            st.title("WordCloud anger")
            wordcloud_image_anger = helper.create_word_cloud(selected_user, anger_data)
            fig, ax = plt.subplots()
            ax.imshow(wordcloud_image_anger)
            st.pyplot(fig)
        if(surprise_data.shape[0]!=0):
            st.title("WordCloud surprise")
            wordcloud_image_surprise = helper.create_word_cloud(selected_user, surprise_data)
            fig, ax = plt.subplots()
            ax.imshow(wordcloud_image_surprise)
            st.pyplot(fig)
        if(neutral_data.shape[0]!=0):
            st.title("WordCloud neutral")
            wordcloud_image_neutral = helper.create_word_cloud(selected_user, neutral_data)
            fig, ax = plt.subplots()
            ax.imshow(wordcloud_image_neutral)
            st.pyplot(fig)
        if(disgust_data.shape[0]!=0):
            st.title("WordCloud disgust")
            wordcloud_image_disgust = helper.create_word_cloud(selected_user, disgust_data)
            fig, ax = plt.subplots()
            ax.imshow(wordcloud_image_disgust)
            st.pyplot(fig)
        if(shame_data.shape[0]!=0):
            st.title("WordCloud shame")
            wordcloud_image_shame = helper.create_word_cloud(selected_user, shame_data)
            fig, ax = plt.subplots()
            ax.imshow(wordcloud_image_shame)
            st.pyplot(fig)
    else:
        st.title("Please Check, Is the input is correct??")
```

CHAPTER-6
TESTING

6.Test Cases Identification

6.1 INTRODUCTION TO TESTING

System testing is the stage of implementation, which aimed at ensuring that system works accurately and efficiently before the live operation commence. Testing is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an error. A successful test is one that answers a yet undiscovered error. The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product.

The software more or less confirms to the quality and reliable standards Testing is one of the most important phases in the software development activity. In software development life cycle (SDLC), the main aim of testing process is the quality, the developed software is tested against attaining the required functionality and performance. The success of the testing process in determining the errors is mostly depends upon the test cases criteria, for testing any software we need to have a description of the expected behavior of the system and the method of determining whether the observed behavior confirmed to the expected behavior.

A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. The underlying motivation of program testing is to affirm software quality with methods that can economically and effectively apply to both strategic to both large and small-scale systems.

To perform effective testing, a software team should conduct effective formal technical reviews. By doing this, many errors will be eliminated before testing commences.

> Testing begins at the component level and works "outward" toward the integration of entire computer-based system.

> Different testing techniques are appropriate at different points in time. The developer of the software and an independent test group conducts testing.

>Testing and debugging are different activities, but debugging must accommodate in a testing strategy.

6.2 TYPES OF TESTING

6.2.1 UNIT TESTING

Unit testing is the testing of each module and the integration of the overall system is done. Unit testing becomes verification efforts on the smallest unit of software design in the module. This is also known as ‘module testing’. The modules of the system are tested separately. This testing is carried out during the programming itself. In this testing step, each model is found to be working satisfactorily as regard to the expected output from the module. There are some validation checks for the fields. For example, the validation check is done for verifying the data given by the user where both format and validity of the data entered is included. It is very easy to find error and debug the system.

6.2.2 INTEGRATION TESTING

Data can be lost across an interface, one module can have an adverse effect on the other sub function, when combined, may not produce the desired major function. Integrated testing is systematic testing that can be done with sample data. The need for the integrated test is to

find the overall system performance. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components. Cyber Bullying Detection on Social Media Using Machine Learning.

6.2.3 FUNCTIONAL TESTING

Functional testing is a formal type of testing performed by testers. unctional testing focuses on testing software against design document, Use cases and requirements document. Functional testing is a black box type of testing and does not require internal working of the software unlike white box testing. Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes.

6.2.3.1 TEST CASE TEMPLATES:

Test Case Template-1
Test case name- Whatsapp Analyser starting test case Test case type- Functional test case Requirement no-1 Module-App interface Status-passed Severity-low Release-1.0 Version-1.0 Pre-condition- Should have an exported chat Test data-WhatsApp-chat1.txt Summary-To check the starting of website

Table 6.1 : Test Case Template-1

Step No	Description	Input	Expected result	Actual result	Status
1	Open browser and enter url	Write the url here	Default display should be shown to collect the data	Default display should be shown to collect the data	pass
2	Enter the Whatsapp exported chat	WhatsApp-chat1.txt and choose the time format	Valid: Dropdown shown, dataset will be shown, buttons shown Invalid: No data in dataframe Error Message: Please check is the input correct?	Valid: Dropdown shown, dataset will be shown, buttons shown Invalid: No data in dataframe Error Message: Please check is the input correct?	pass

Table 6.2 : Starting Test Case

Test Case Template-2
Test Case name-Whatsapp chat statistical test case Test case type- Functional test case Requirement no-2 Module-Whatsapp chat Statistics Status-passed Severity-low Release-1.0 Version-1.0 Pre-condition- Should have an exported chat Test data-WhatsApp-chat1.txt Summary-To check the working of the statistical Analysis

Table 6.3 : Test Case Template-2

Step No	**Description Input**	**Expected Result**	**Actual Result**	**Status**
1	1.Choosing dropdown contact 2.Show Analysis button click	Message count Monthly timeline Daily timeline Activity Map Weekly Activity Map Most busy user Word cloud Most frequent words Most common emoji	Message count Monthly timeline Daily timeline Activity Map Weekly Activity Map Most busy user Word cloud Most frequent words Most common emoji	Pass

Table 6.4 : Statistical Test Case

Test Case Template-3
Test Case name-Whatsapp chat language & sentimental test case Test case type- Functional test case Requirement no-3 Module-Language detection & Sentiment detection Status-passed Severity-low Release-1.0 Version-1.0 Pre-condition- Should have an exported chat Test data-WhatsApp-chat1.txt Summary-To check the working of the Language and sentimental Analysis

Table 6.5 : Test Case Template-3

Step No	Description Input	Expected Result	Actual Result	Status
1	1.Choosing dropdown contact 2.Language & Sentiment Detection button click	Language detection Total English messages English message count Non-English message count Users using other languages Sentiment of the message Overall percentage of each sentiment Wordcloud on each sentiment	Language detection Total English messages English message count Non-English message count Users using other languages Sentiment of the message Overall percentage of each sentiment Wordcloud on each sentiment	Pass

Table 6.6: Language & Sentimental Test Case

6.2.4 LOAD TESTING :

Load testing is a type of software testing that is conducted to check the tolerance/behavior of the system under a specific expected load. Locust is an open-source load-testing tool.
The target of locust is load-testing websites and checking the number of concurrent users a system can handle. During a locust test, a swarm of locusts will attack the target i.e website. The behavior of each locust is configurable and the swarming process is monitored from a web UI in real time.

Specialty of locust are:

1. Test scenarios can be written in Python.
2. Distributed and scalable
3. Web-based UI
4. Any system can be tested using this tool

Locust can be installed with pip.
pip install locust

Once the locust is successfully installed, a locust command should be available in shell.

Code:

```
from locust import HttpUser,task,between

class AppUser(HttpUser):
   wait_time = between(2,5)

   #end point
   @task
   def home_page(self):
      self.client.get("/")
```

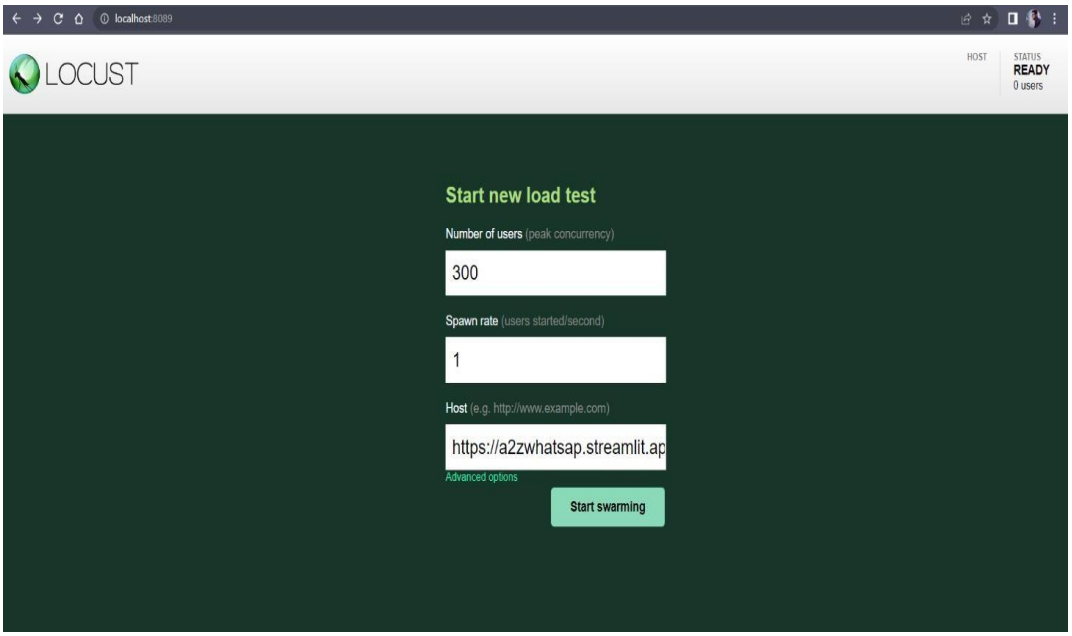


Fig 6.1 Load test Interface

Test Report For Load Testing:

4/1/23, 7:36 PM Test Report for loadTestFile.py

Locust Test Report

During: 4/1/2023, 7:31:06 PM - 4/1/2023, 7:36:34 PM

Target Host: https://a2zwhatsap.streamlit.app/

Script: loadTestFile.py

Download the Report

Request Statistics

Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	//	12418	0	832	474	6936	5616	37.8	0.0
	Aggregated	12418	0	832	474	6936	5616	37.8	0.0

Response Time Statistics

Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	//	710	760	840	950	1300	1700	2400	6900
	Aggregated	710	760	840	950	1300	1700	2400	6900

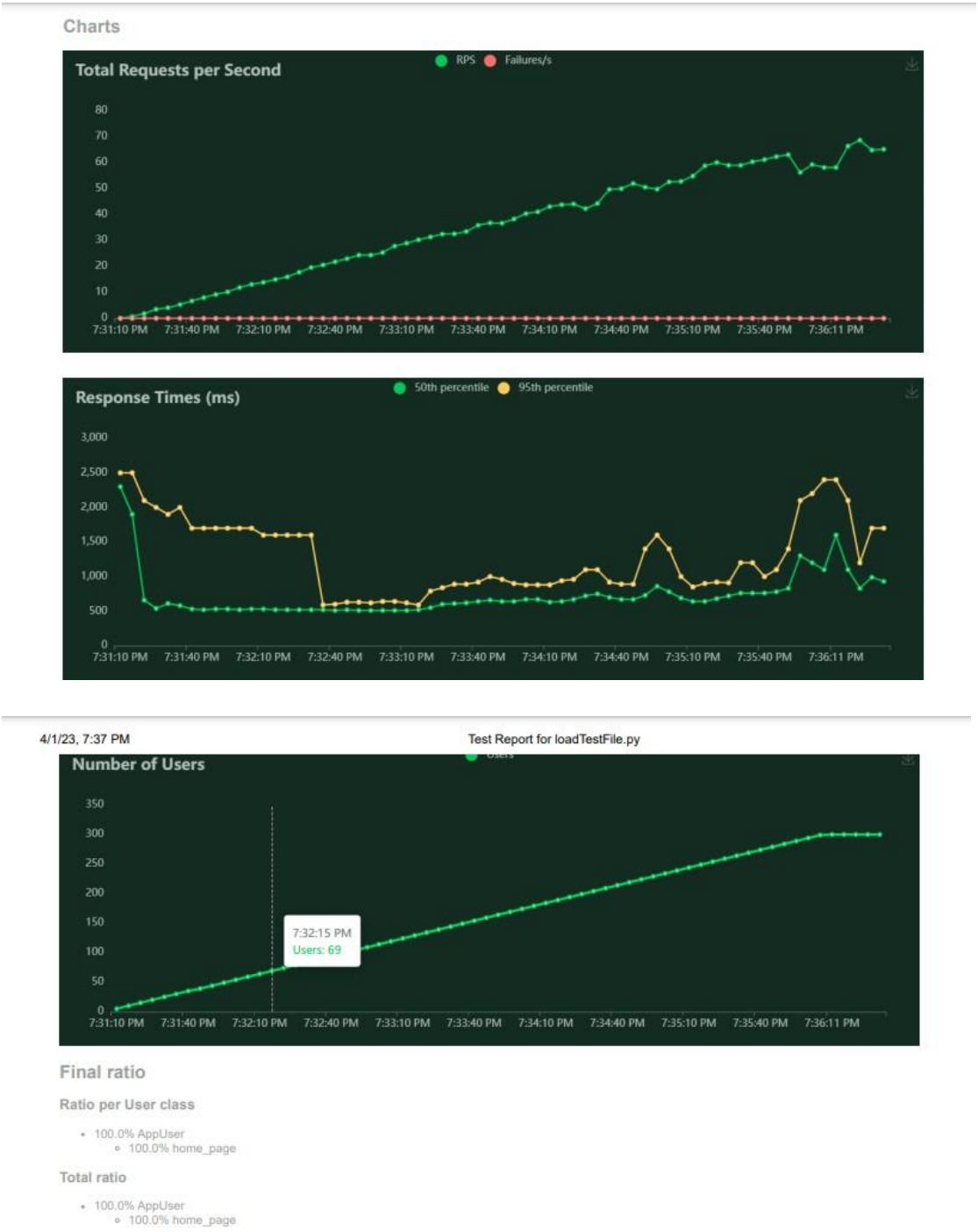


Fig 6.2 : Load Test Report

CHAPTER-7
OUTPUT SCREENS

7.OUTPUT SCREENSHOTS OF THE PROJECT EXECUTION

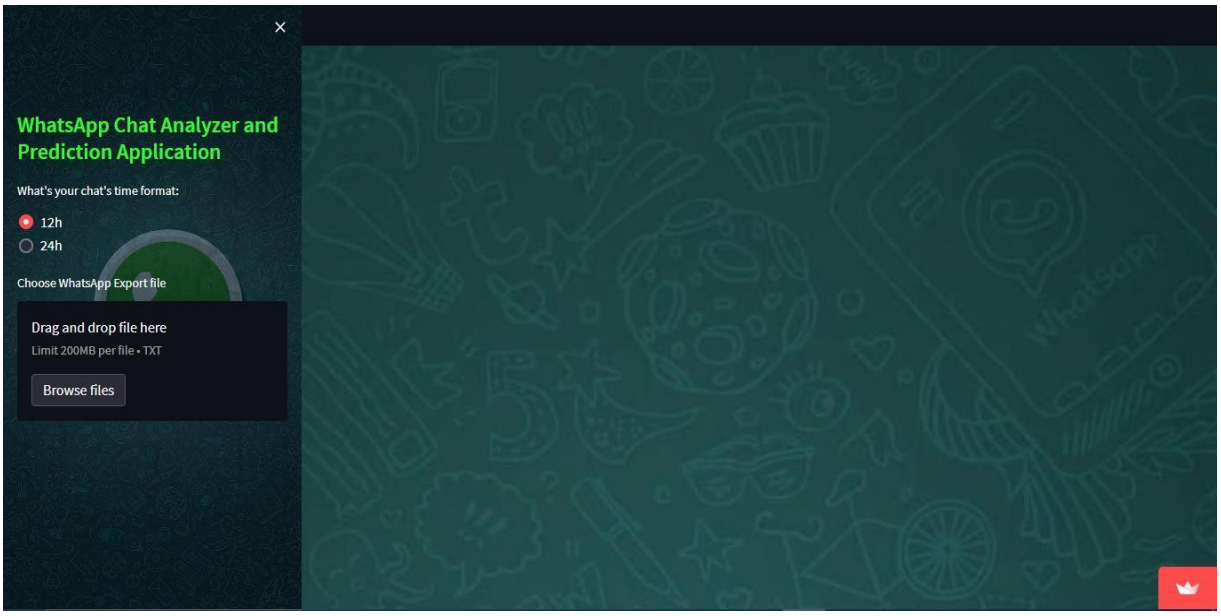


Fig 7.1: User Interface

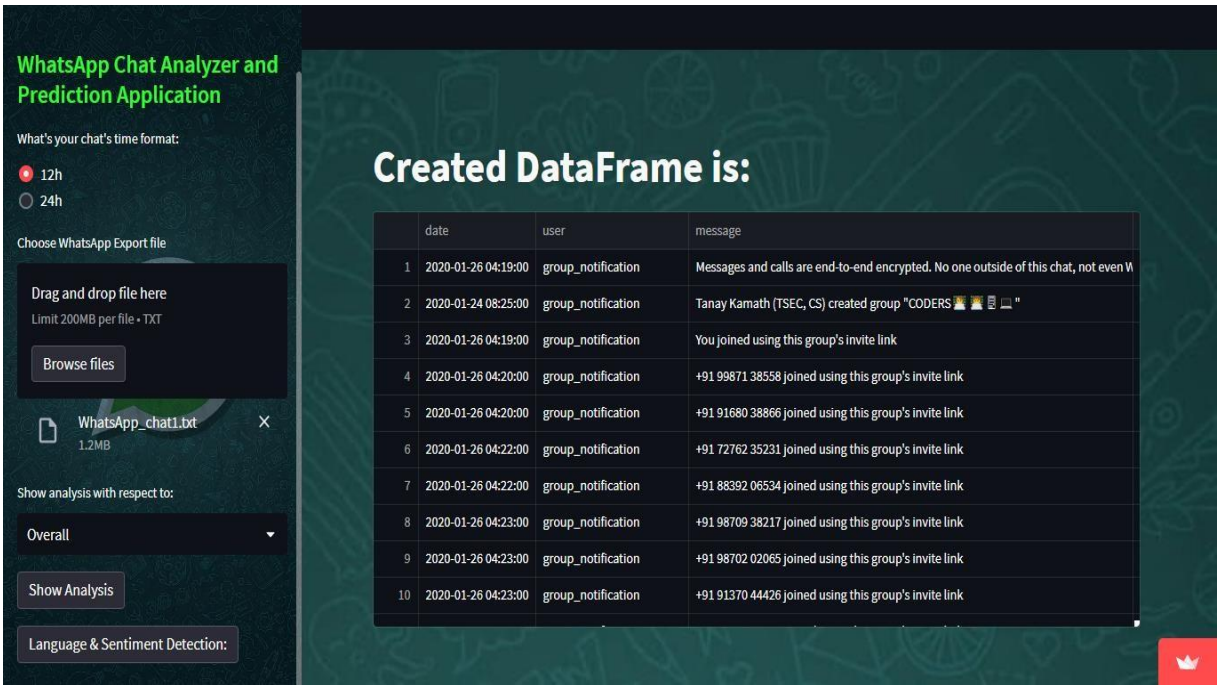


Fig 7.2: Whatsapp Chat Dataframe

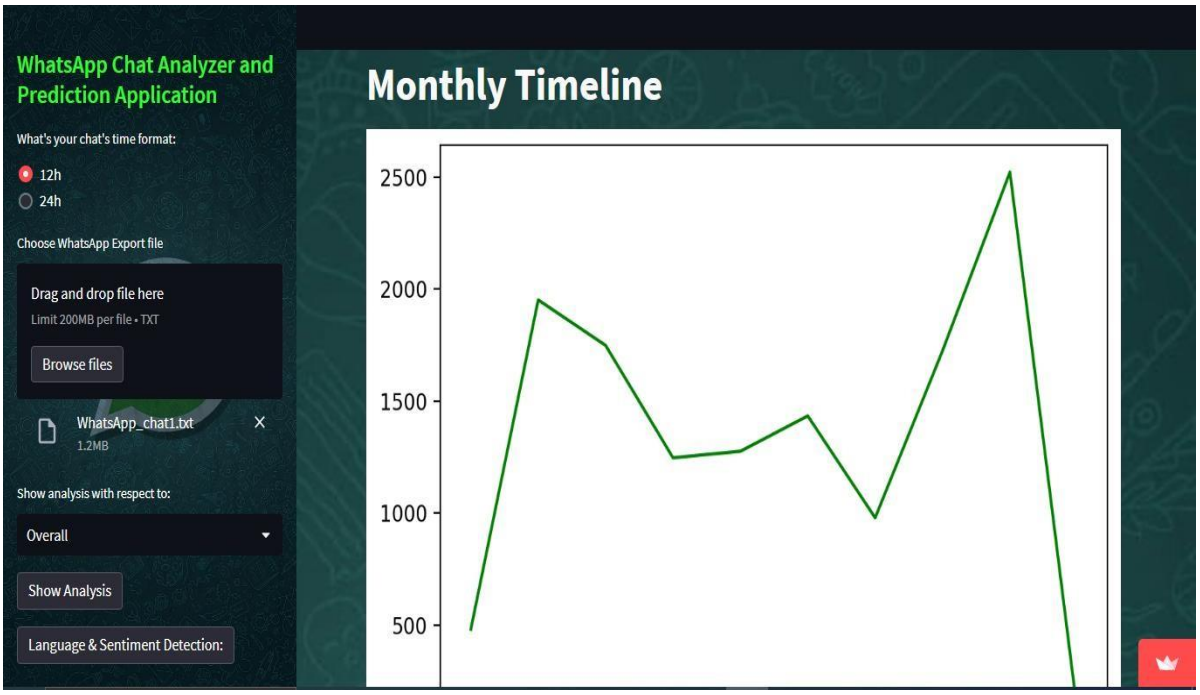


Fig 7.3:Total Messages,words,urls and Monthly Timeline

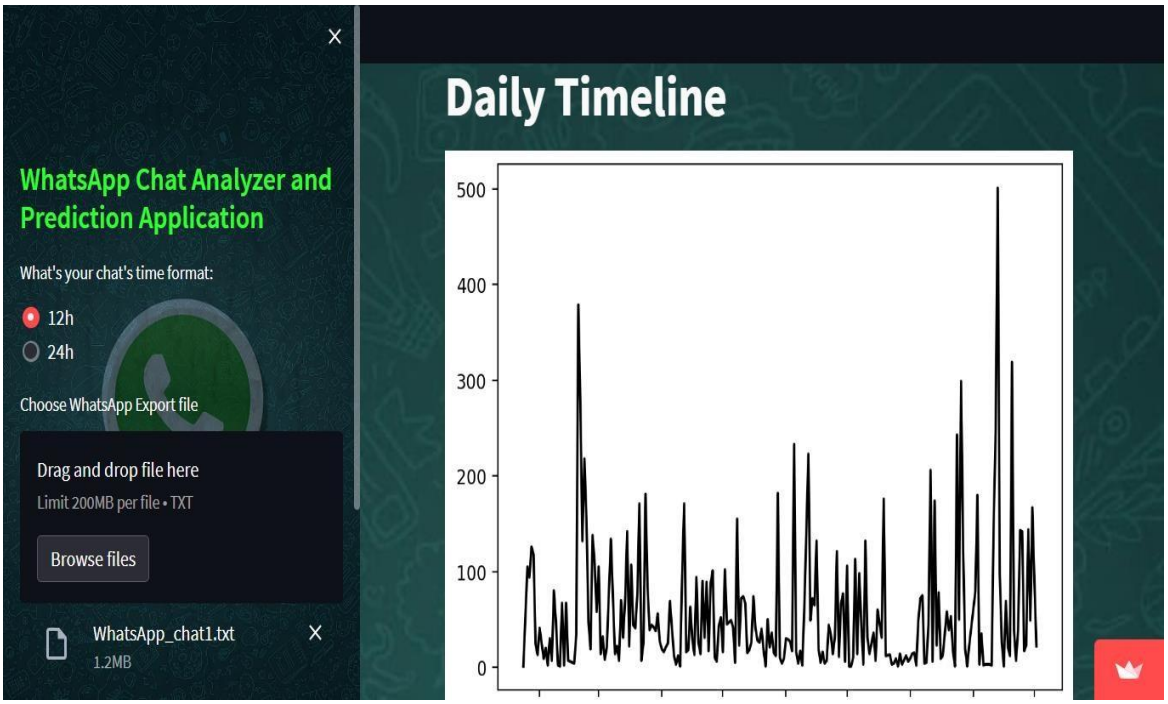


Fig 7.4:Daily Timeline

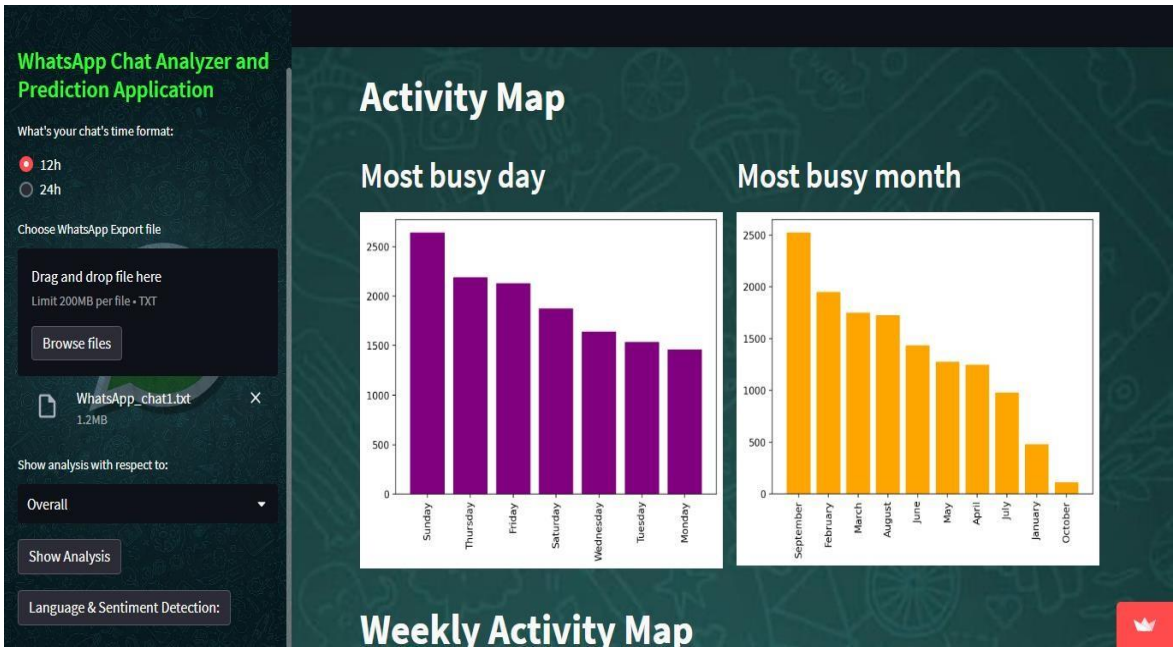


Fig 7.5: Activity Map

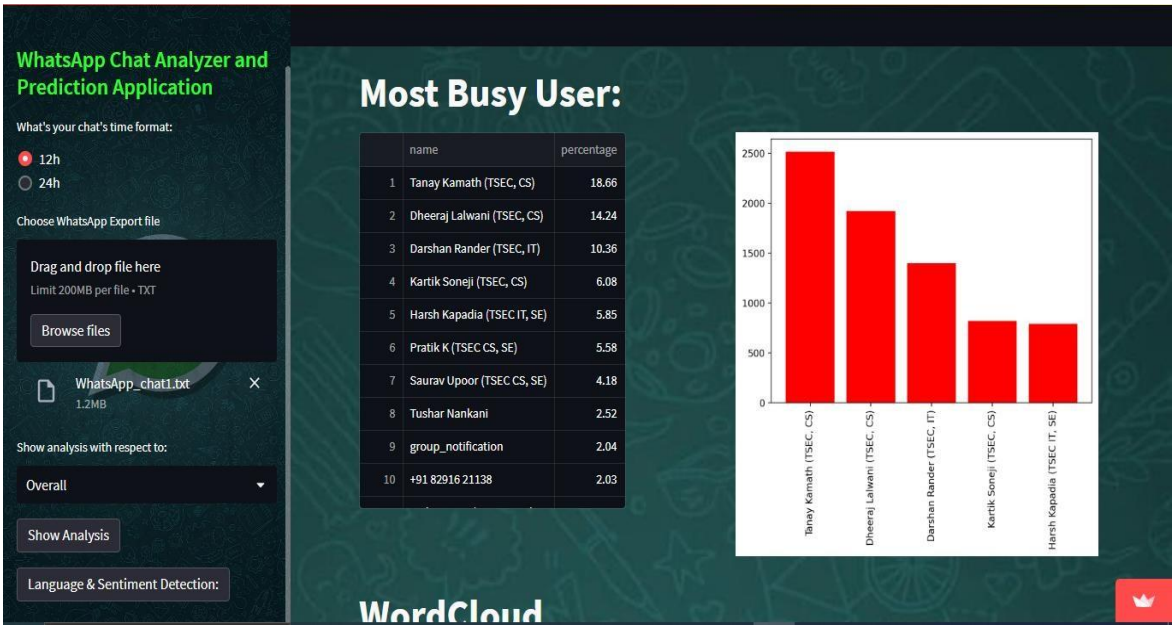


Fig 7.6: Most Busy User

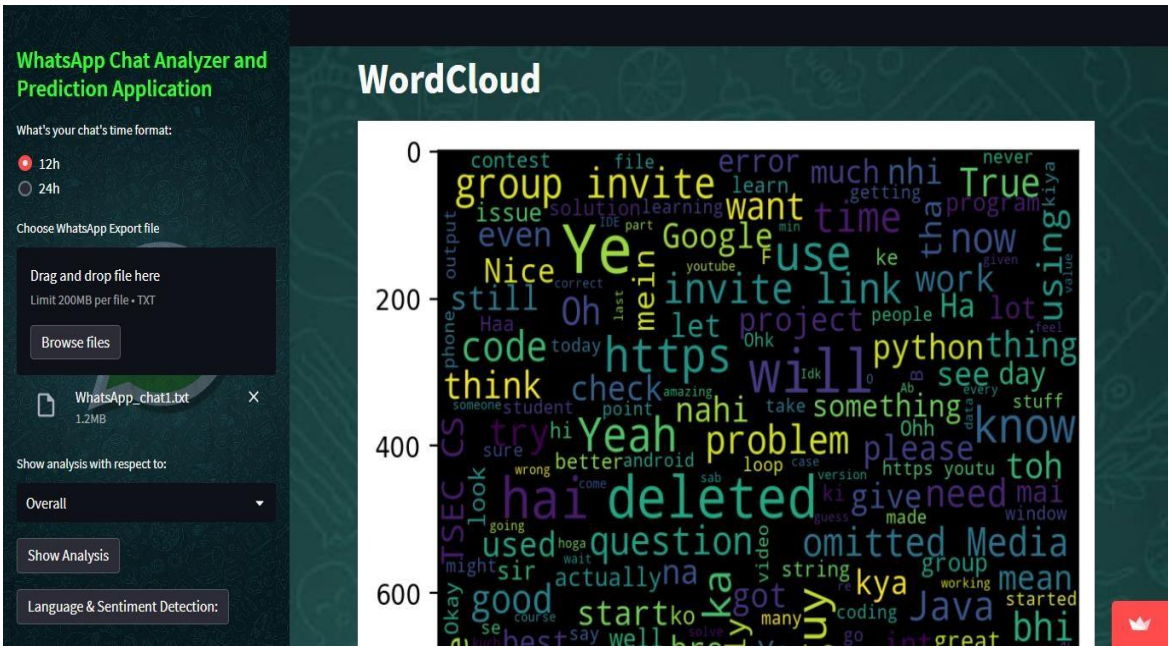


Fig 7.7: WordCloud

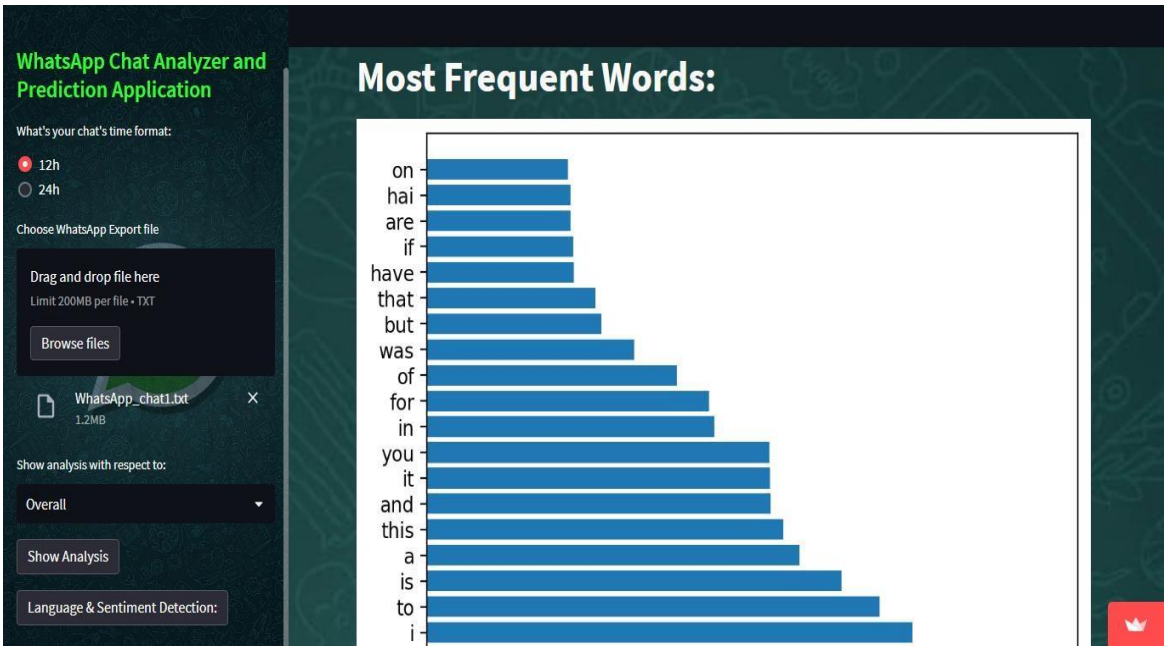


Fig 7.8: Frequent Words

Fig 7.9: Most Common Emojis

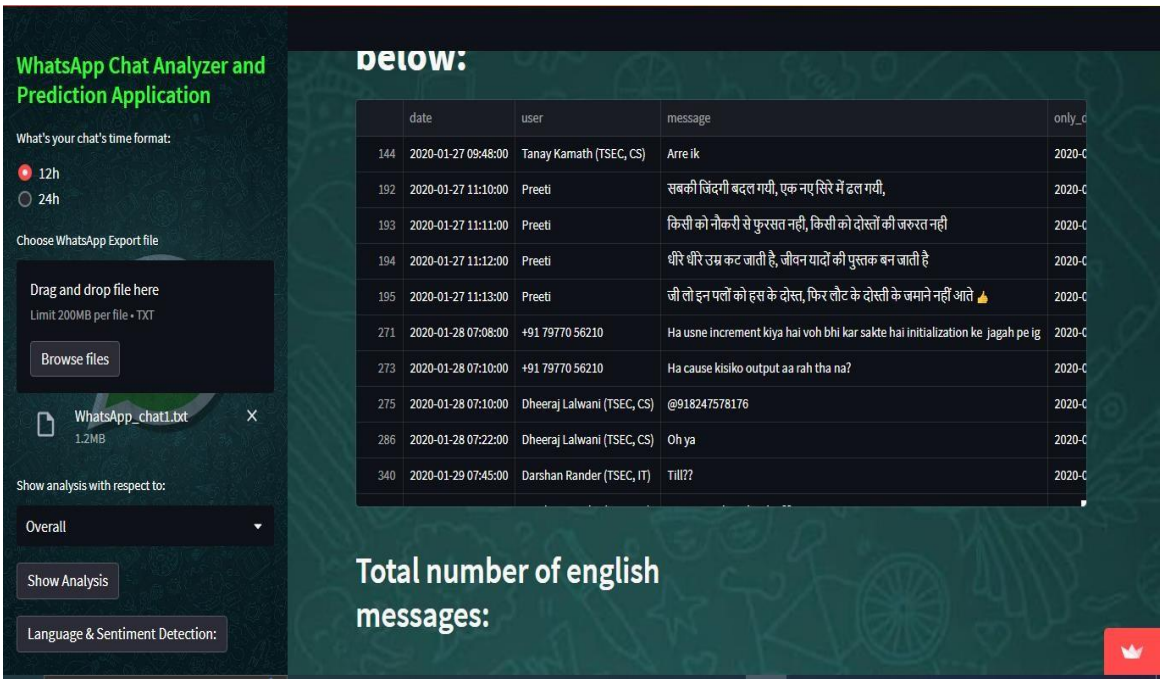


Fig 7.10: Language partitioned dataframe

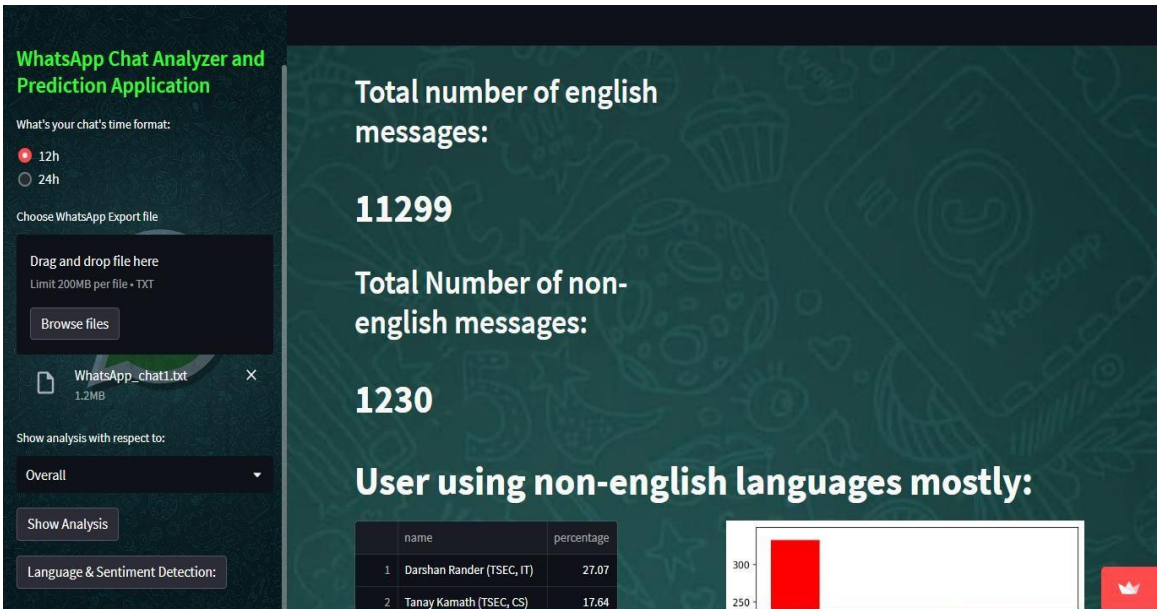


Fig 7.11: English and non-English message count

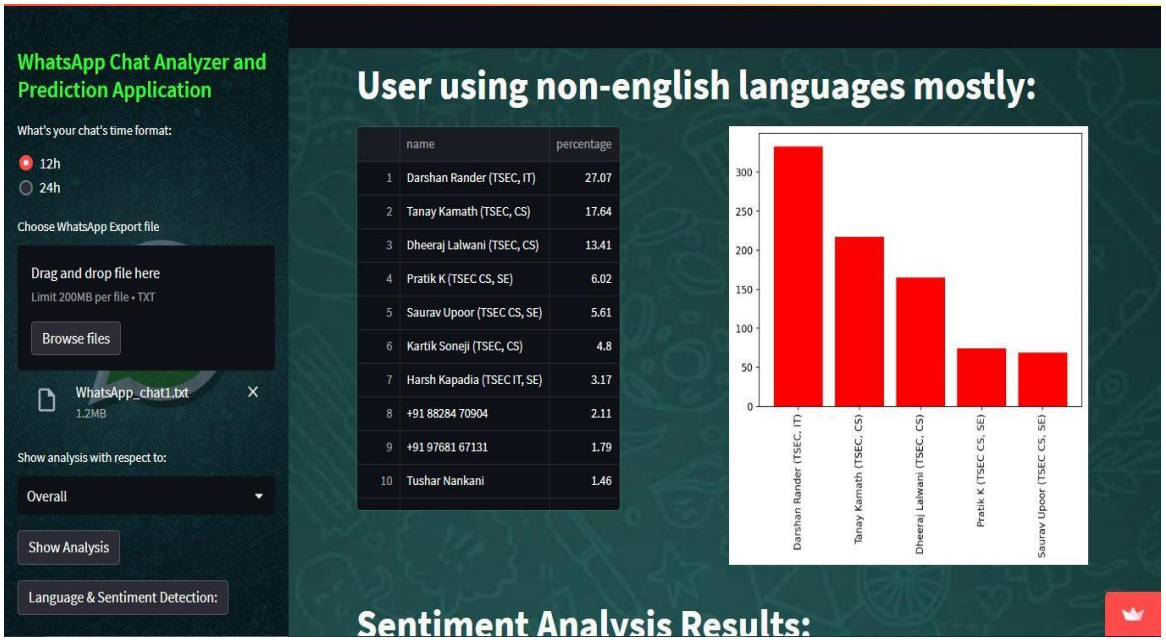
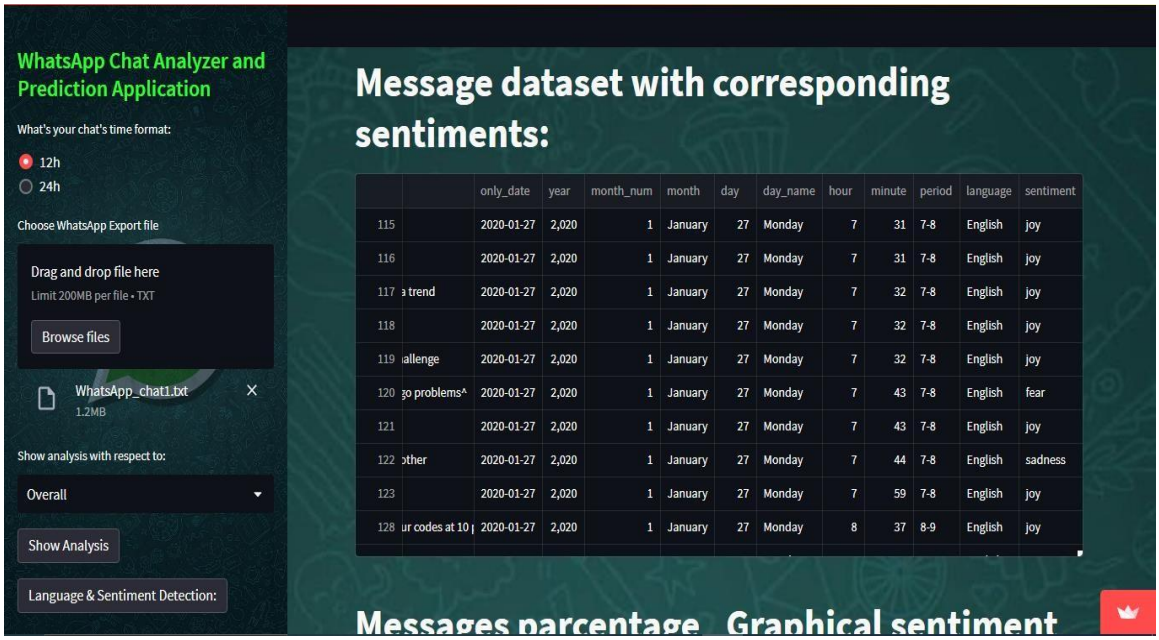
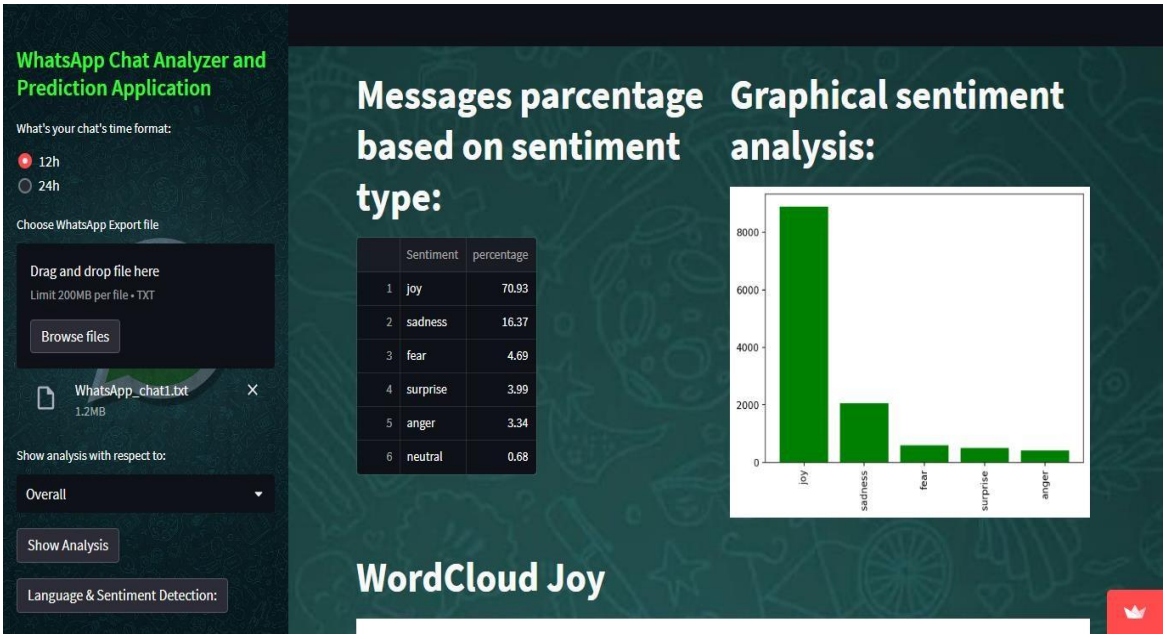


Fig 7.12: User Partition on using other languages



		only_date	year	month_num	month	day	day_name	hour	minute	period	language	sentiment
115		2020-01-27	2,020	1	January	27	Monday	7	31	7-8	English	joy
116		2020-01-27	2,020	1	January	27	Monday	7	31	7-8	English	joy
117	a trend	2020-01-27	2,020	1	January	27	Monday	7	32	7-8	English	joy
118		2020-01-27	2,020	1	January	27	Monday	7	32	7-8	English	joy
119	allenge	2020-01-27	2,020	1	January	27	Monday	7	32	7-8	English	joy
120	go problems^	2020-01-27	2,020	1	January	27	Monday	7	43	7-8	English	fear
121		2020-01-27	2,020	1	January	27	Monday	7	43	7-8	English	joy
122	other	2020-01-27	2,020	1	January	27	Monday	7	44	7-8	English	sadness
123		2020-01-27	2,020	1	January	27	Monday	7	59	7-8	English	joy
128	ur codes at 10	2020-01-27	2,020	1	January	27	Monday	8	37	8-9	English	joy

Fig 7.13: Message Sentiment dataframe



	Sentiment	percentage
1	joy	70.93
2	sadness	16.37
3	fear	4.69
4	surprise	3.99
5	anger	3.34
6	neutral	0.68

Fig 7.14: Sentiment percentage

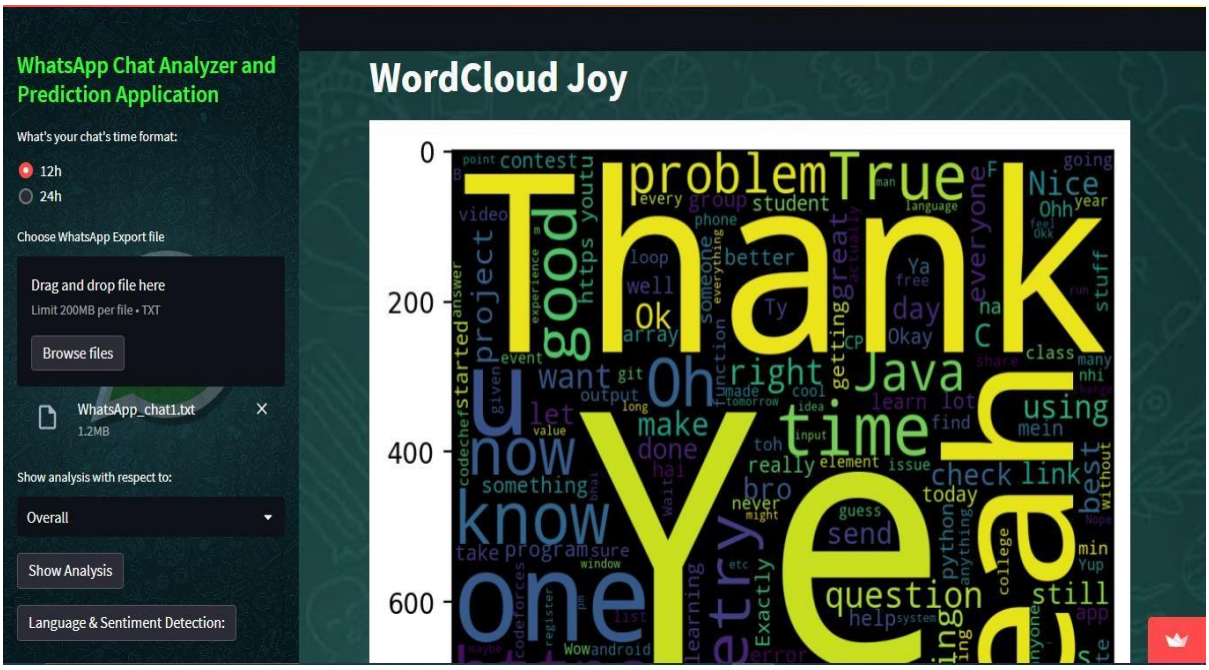


Fig 7.15: WordCloud Joy

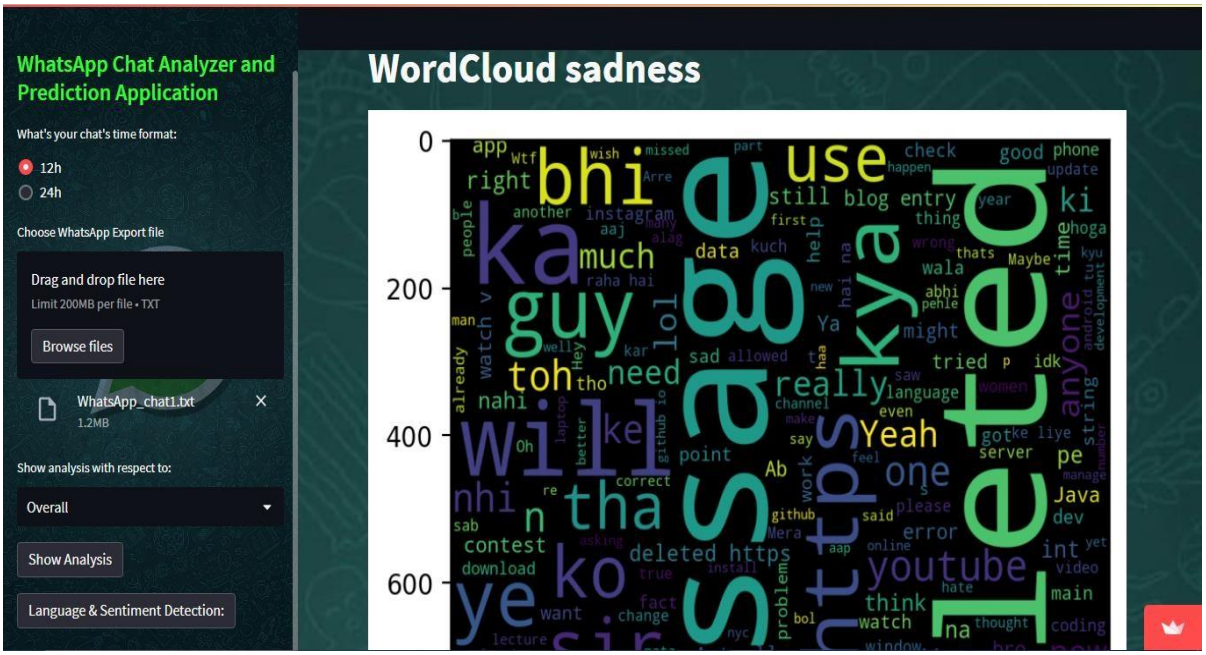


Fig 7.16: WordCloud Sadness

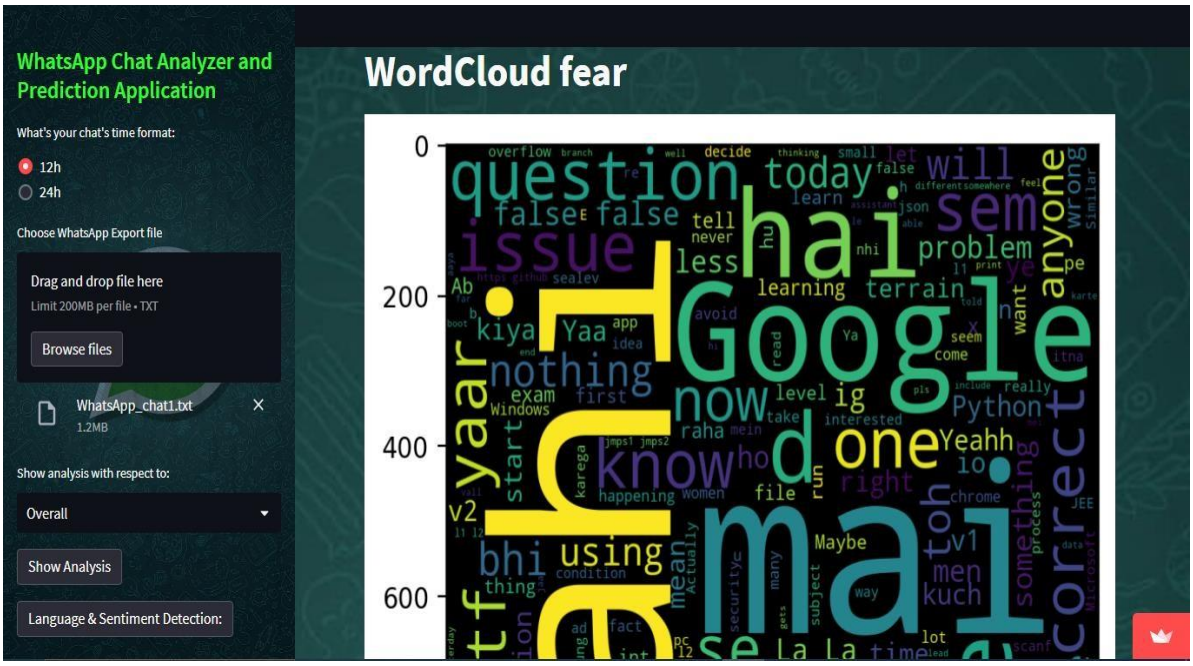


Fig 7.17: WordCloud fear

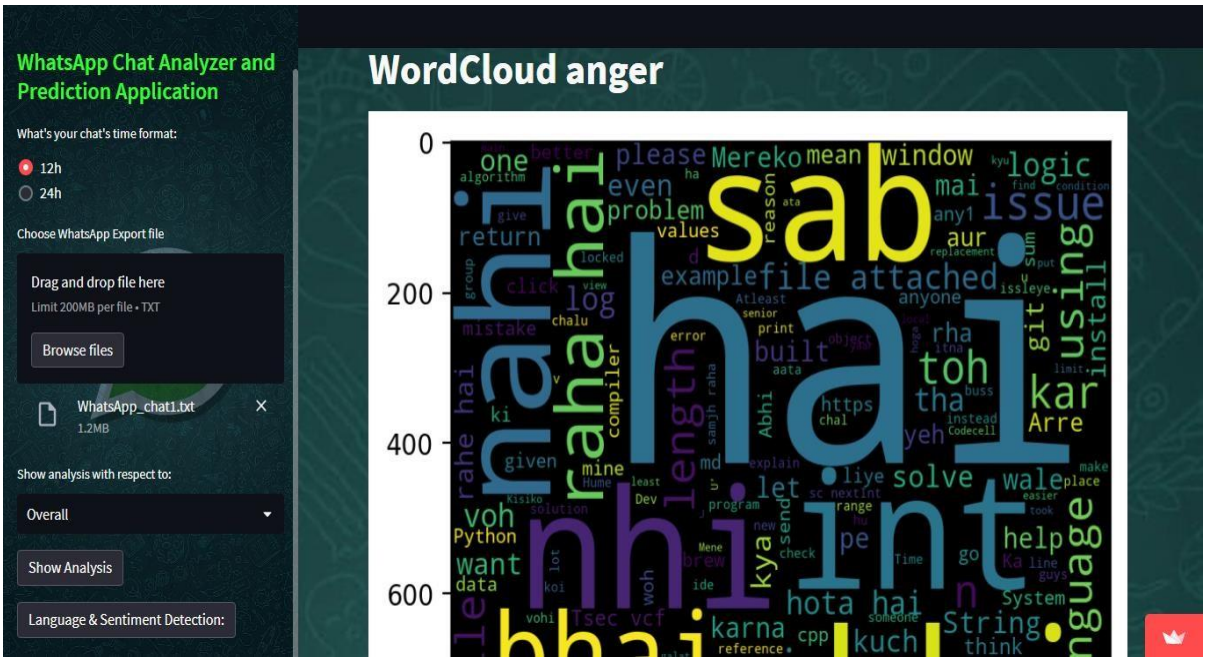


Fig 7.18: WordCloud anger

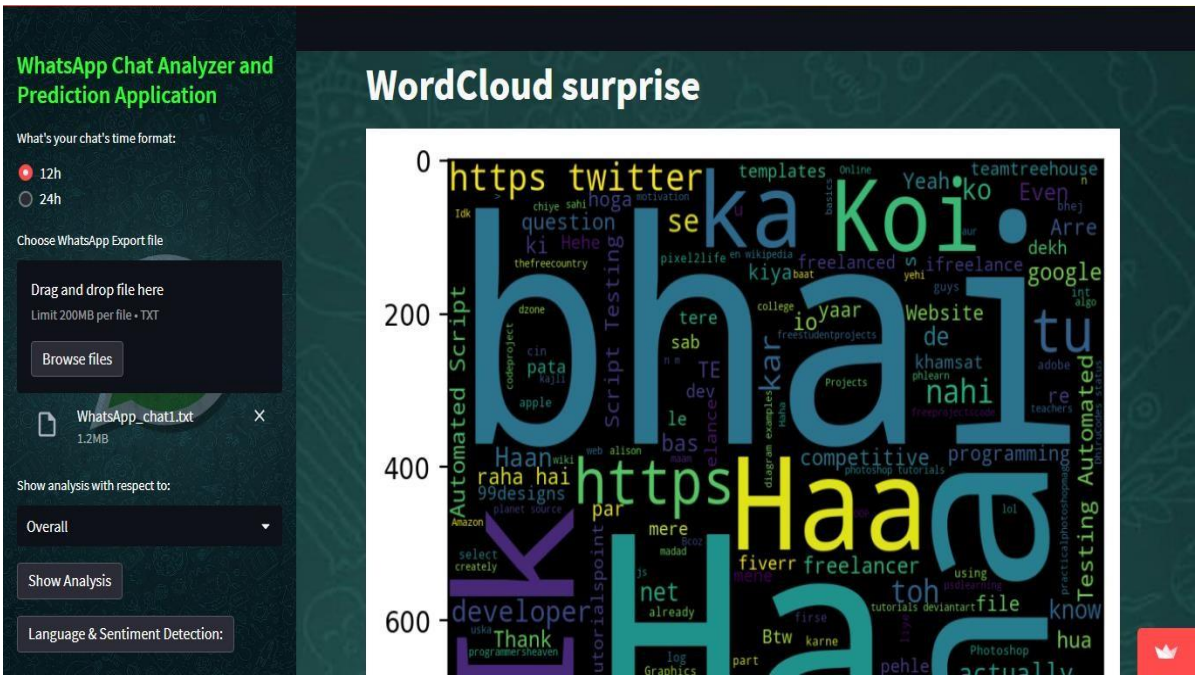


Fig 7.19: WordCloud surprise

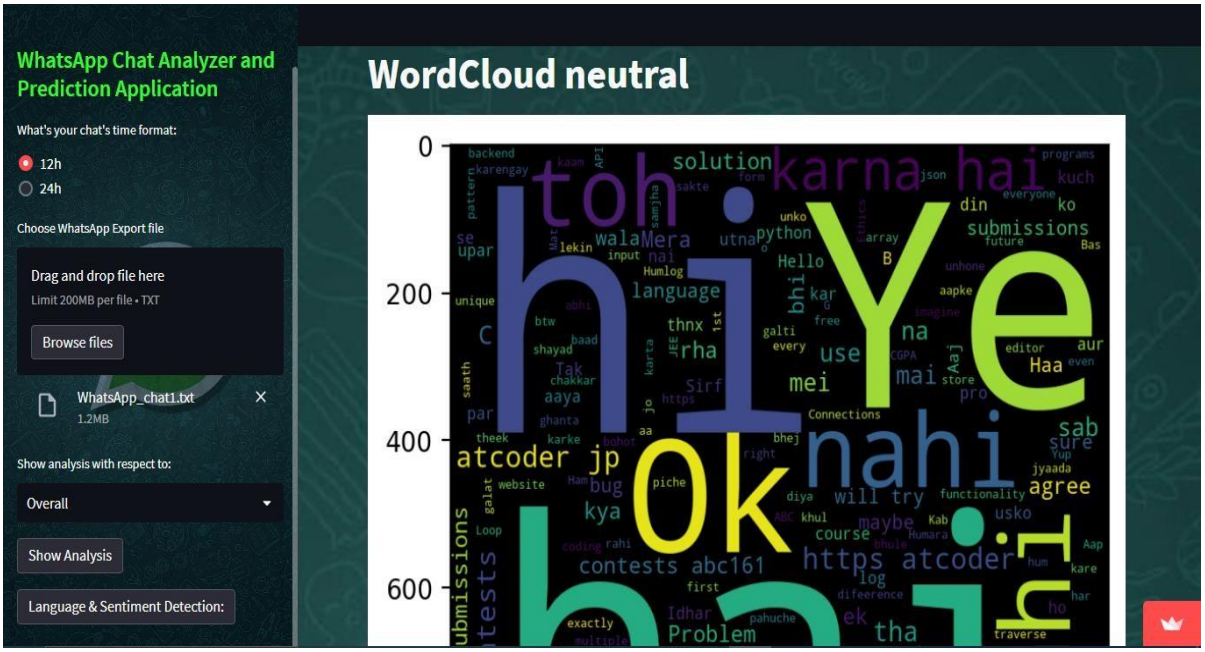


Fig 7.20: WordCloud neutral

8. Deployment and Maintenance Details

Streamlit Community Cloud is a workspace for your team to deploy, manage, and collaborate on your Streamlit apps. We can connect our Streamlit Community Cloud account directly to your GitHub repository (public or private) and then Streamlit Community Cloud launches the apps directly from the code we've stored on GitHub. This creates a fast iteration cycle for our deployed apps, so that developers and viewers of apps can rapidly prototype, explore, and update apps.

Under the hood Streamlit Community Cloud handles all of the containerization, authentication, scaling, security and everything else so that all we need to worry about is creating the app. Maintaining Streamlit apps is easy. Containers get the latest security patches, are actively monitored for container health. We are also building the capability to observe and monitor apps.

Steps followed while deploying the app in the cloud:

1. We have created our requirement.txt file containing all the dependencies and python modules required for our app to run, so that the cloud platform can recognize that and install all the dependencies.
2. Then we have to add a setup.sh file containing all kind of dependencies required to execute the files.
3. Optionally we need to have a procfile which is basically a text file without any extension containing the starting command for starting the app or setup commands. It always works in the root directory.

We can also collaborate as a team by having multiple pushing to the same GitHub repo. Whenever anyone on our team updates the code on GitHub, the app will also automatically update.

Maintenance:

All kind of maintenance like HTTPS connection , Data hosting, virtual private cloud creation and maintaining security practices like confidentiality , Integrity and availability are maintained by streamlit cloud itself.

9. CONCLUSION

In conclusion about our project, we can express that, there is not an existing system available with WhatsApp data sentiment analysis for end users. Conflicts in WhatsApp group conversations may create more social problems. Peoples are using delivering hate speeches and fake information to create panic in society through WhatsApp. People can use WhatsApp freely in developing countries so more security concerns is to be present for our society regarding this app. The used algorithms in the proposed system are very easy to implement and no need to have large infrastructures to implement this. We have implemented same Naïve bayes classification algorithm for creating both the models. Further any user can get a nice interaction with system with user friendly interfaces. After successful implementation of the system, it will help a large number of group admins who have so much daily work with WhatsApp chats. In future we can add other advanced features to the system as well. We hope this will solve the existing problems in certain extend and will express the power of artificial intelligence in society.

10. BIBLIOGRAPHY

[1] Dahiya, S., Mohta, A., & Jain, A. (2020, June). Text classification based behavioural analysis of whatsapp chats. In 2020 5th international conference on communication and electronics systems (ICCES) (pp. 717-724). IEEE.

[2] Mohta, A., Jain, A., Saluja, A., & Dahiya, S. (2020, October). Pre-Processing and Emoji Classification of WhatsApp Chats for Sentiment Analysis. In 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) (pp. 514-519). IEEE.

[3] Kanakia, H., Raundale, P., Britto, R., & Sawardekar, R. (2019, May). Analysis of Social Networks using Naive Bayes. In 2019 International Conference on Intelligent Computing and Control Systems (ICCS) (pp. 88-91). IEEE.

[4] Shafiq, M., Yu, X., & Laghari, A. A. (2016, September). WeChat text messages service flow traffic classification using machine learning technique. In 2016 6th International Conference on IT Convergence and Security (ICITCS) (pp. 1-5). IEEE.

[5] Liu, B., Blasch, E., Chen, Y., Shen, D., & Chen, G. (2013, October). Scalable sentiment classification for big data analysis using naive bayes classifier. In 2013 IEEE international conference on big data (pp. 99-104). IEEE**.**