

SPAM DETECTION AND BLOCKING OF SPAMMER BASED ON TWITTER

Submitted in partial fulfillment of the requirements for the
award of
Bachelor of Engineering degree in Computer Science and Engineering

by

VELAGACHERLA LOKESHKUMAR REDDY (Reg.no. 37110815)

TADIKONDA AKHIL NAGA SAI KRISHNA (Reg.no. 37110765)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

**Accredited with Grade “A” by NAAC JEPPIAAR NAGAR, RAJIV GANDHI
SALAI, CHENNAI – 600 119**

MARCH – 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **VELAGACHERLA LOKESHKUMAR REDDY (Reg. No. 37110815)** and **TADIKONDA AKHIL NAGA SAI KRISHNA (Reg. No. 37110765)** who carried out the project entitled "**SPAM DETECTION AND BLOCKING OF SPAMMER BASED ON TWITTER**" under my supervision from **November 2020 to March 2021**.

Internal Guide

Dr. L.K.JOSHILA GRACE, M.E., Ph.D.,

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **VELAGACHERLA LOKESHKUMAR REDDY, TADIKONDA AKHIL NAGA SAI KRISHNA** here by declare that the Project Report entitled “**SPAM DETECTION AND BLOCKING OF SPAMMER BASED ON TWITTER**” is done by me under the guidance of **Dr. L. K. JOSHILA GRACE, M.E., Ph.D.**, Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr.T.Sasikala M.E., Ph.D ., Dean**, School of Computing and **Dr.L.Lakshmanan M.E., Ph.D.,and Dr.S.Vigneshwari M.E., Ph.D.**, Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.L.K.JOSHILA GRACE, M.E.,Ph.D., Assistant Professor**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Online review systems play an important role in affecting consumers' behaviors and decision making, attracting many spammers to insert fake reviews to manipulate review content and ratings. To increase utility and improve user experience, some online review systems allow users to form social relationships between each other and encourage their interactions. In this paper, we aim at providing an efficient and effective method to identify review spammers by incorporating social relations based on two assumptions that people are more likely to consider reviews from those connected with them as trustworthy, and review spammers are less likely to maintain a large relationship network with normal users. The contributions of this paper are two-fold: (1) We elaborate how social relationships can be incorporated into review rating prediction and propose a trust-based rating prediction model using proximity as trust weight, and (2) We design a trust-aware detection model based on rating variance which iteratively calculates user-specific overall trustworthiness scores as the indicator for spam city. Experiments on the dataset collected from Yelp.com show that the proposed trust-based prediction achieves a higher accuracy than standard CF method, and there exists a strong correlation between social relationships and the overall trustworthiness scores.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	vi
	LIST OF FIGURES	viii
1	INTRODUCTION	1
	1.1 OUTLINE OF PROJECT	1
	1.2 SOCIAL NETWORKING SERVICES	2
2	LITERATURE SURVEY	3
	2.1 LITERATURE REVIEW	6
3	AIM AND SCOPE	6
	3.1 AIM	6
	3.2 OBJECTIVE	6
	3.3 RESEARCH QUESTIONS	7
	3.4 REPORT STRUCTURE	7
4	SYSTEM DESIGN AND METHODOLOGY	9
	4.1 EXISTING SYSTEMS	9
	4.1.1 DISADVANTAGES OF EXISTING SYSTEM	9
	4.2 PROPOSED SYSTEM	9
	4.2.1 ADVANTAGES OF PROPOSED SYSTEM	10
	4.3 REQUIREMENT ANALYSIS	10
	4.3.1 FUNCTIONAL REQUIREMENTS	10
	4.3.2 NON-FUNCTIONAL REQUIREMENTS	11
	4.4 SYSTEM DESIGN AND TESTING	12
	4.4.1 INPUT DESIGN	12
	4 4.2 OUTPUT DESIGN	13
	4.5 SYSTEM REQUIREMENTS	13
	4.5.1 HARDWARE REQUIREMENTS	13

	4.5.2 SOFTWARE REQUIREMENTS	13
	4.6 MODULE DESCRIPTION	14
5	RESULT AND DISCUSSION	23
6	CONCLUSION AND FUTURE WORK	26
	REFERENCES	27
	APPENDIX	
	A. SOURCE CODE	29
	B. SCREEN SHOT	44
	C. PALAGRISM REPORT	48

List of Figures

FIGURE NO.	FIGURE NAME	PAGE NO.
4.6	ARCHITECTURAL DESIGN	16
4.6.1	DATA FLOW DIAGRAM	17
4.6.2	USE CASE DIAGRAM	19
4.6.3	CLASS DIAGRAM	20
4.6.4	SEQUENCE DIAGRAM	21
4.6.5	ACTIVITY DIAGRAM	22
5.1	IMPORTING LIBRARIES	24
5.2	TRAINED DATA	25
5.3	DETECTION	25
6.1	COMMAND PROMPT WITH REFERENCE ADDRESS	44
6.2	LOGIN PAGE	45
6.3	DATA COLLECTION	45
6.4	USER CHECK	46
6.5	OUTPUT	46
6.6	FAKE OR LEGITIMATE USE	47

CHAPTER 1

INTRODUCTI

ON

1.1 OUTLINE OF PROJECT

Spam filters detect unsolicited, unwanted, and virus-infested email (called spam) and stop it from getting into email inboxes. Internet Service Providers (ISPs) use spam filters to make sure they aren't distributing spam. Small- to medium- sized businesses (SMBs) also use spam filters to protect their employees and networks.

Spam filters are applied to both inbound email (email entering the network) and outbound email (email leaving the network). ISPs use both methods to protect their customers. SMBs typically focus on inbound filters.

There are many spam filtering solutions available. They can be hosted in the "cloud," on computer servers, or integrated into email software such as Microsoft Outlook

Wikipedia defines a social network service as a service which "focuses on the building and verifying of online social networks for communities of people who share interests and activities, or who are interested in exploring the interests and activities of others, and which necessitates the use of software."

Support for learning: Social networks can enhance informal learning and support social connections within groups of learners and with those involved in the support of learning. **Support for members of an organization:** Social networks can potentially be used by all members of an organization, and not just those involved in working with students. Social networks can help the development of communities of practice.

Engaging with others: Passive use of social networks can provide valuable business intelligence and feedback on institutional services (although this may give rise to ethical concerns).

Ease of access to information and applications: The ease of use of many social networking services can provide benefits to users by simplifying access to other tools

and applications. The Facebook Platform provides an example of how a social networking service can be used as an environment for other tools.

Common interface: A possible benefit of social networks may be the common interface which spans work / social boundaries. Since such services are often used in a personal capacity the interface and the way the service works may be familiar, thus minimizing training and support needed to exploit the services in a professional context. This can, however, also be a barrier to those who wish to have strict boundaries between work and social activities.

1.2 Social Networking Services:

Facebook: Facebook is a social networking Web site that allows people to communicate with their friends and exchange information. In May 2007 Facebook launched the Facebook Platform which provides a framework for developers to create applications that interact with core Facebook features

Myspace: Myspace is a social networking Web site offering an interactive, user-submitted network of friends, personal profiles, blogs and groups, commonly used for sharing photos, music and videos.

Ning: An online platform for creating social Web sites and social networks aimed at users who want to create networks around specific interests or have limited technical skills.

Twitter: Twitter is an example of a micro-blogging service. Twitter can be used in a variety of ways including sharing brief information with users and providing support for one's peers. Note that this brief list of popular social networking services omits popular social sharing services such as Flickr and YouTube.

CHAPTER-2

LITERATURE SURVEY

2.1 Literature Review:

1) Glided Twitter spam:

Now a days Twitter fraud has been as critical downside. Recent present works specialize in applying artificial intelligence or ML techniques for spam detection, that create use of the statistical options of tweets. In the labeled tweets data set. The work tends to observe that the proposed properties of spam tweets differ by time, and thus, the performance of existing machine learning-based classifiers is reduced. This issue is brought up as "Twitter Spam Drift". so as to tackle this downside, it tends to first perform a deep analysis on the proposed system of 1 Lakh spam tweets and 1,000,000 non- Fraud tweets, then propose a scheme called Lfun. The proposed arrangement can discover "changed" spam tweets from unlabeled tweets and merge them into classifier's planning method. Different type of experiments are done to predict the projected theme. The results show that the projected Lfun theme will considerably improve the spam detection accuracy in real- world situations.

2) Automatically perceiving fake news in notable Twitter strings:

Information quality on social media is becoming increasingly relevant, but experts' ability to analyses and correct much of the misleading material, or "fake news," on these sites is limited by web-scale data. In this work, we will provide an automated way to detect fake news on Twitter by learning how to accurately predict false statistics in a reliable Twitter database.: CREDBANK is a crowdsourcing site.

PHEME, a journalism research journal on possible rumors and correlations on Twitter and a rating scale for Twitter cases. Used this approach to evaluate BuzzFeed's fake data Twitter content is a product that is trained for a large number of employees, both trained in the media system and affiliates. and journalistic results All three datasets are now publicly accessible, having been aligned into a standard format. The findings of a feature study, which are consistent with previous work, classify functions which can be maximum predictive for crowdsourced and journalistic accuracy assessments. The work end with a summary of the differences between accuracy and integrity, as well as why non-expert models outperform journalistic models.

3) Streaming spam tweets acknowledgment :

Twitter's scale of success attracts more fake advertisers. Spammers send spam messages to Twitter users to promote websites and services that are harmful to everyday users. Researchers have suggested a number of different methods to combat scammers. Applying machine learning technology to search for spam has been the subject of recent research, but tweets have been restored in the stream. Developers and researchers can use Twitter's ongoing API to access public tweets in real time. No spam filter uses the current learning curve. We fill in the gaps in this work by conducting an evaluation process based on three different areas: data, functions, and products. By using a URL-based security tool, we've reached over 600 million key real estate properties. Public tweets have been improved and 12 weighted features have been launched to represent tweets for real-time spam detection. Spam Search In addition, n translates into a binary distribution problem in the feature space. This can be solved using standard machine learning algorithms. We then examine how various variables such as spam-to-spam optimization, role classification, training data size, sample data, time-related data, and machine learning algorithms affect spam detection. This is still a daunting task and force detection techniques need to take into account three areas: data, function, and products.

4) A model-based system for recognizing spammers :

In the work it suggest a mathematical method for detecting spammers based on the Dirichlet distribution. Current unsupervised methods require human involvement to set informal threshold criteria to identify spammers, while the proposed method would automatically differentiate between fake person or genuine person. It approach the issue of finding fake persons in social networks from the perspective of mixture modelling and develop a principled unsupervised approach to do so. In this approach, first create a feature vector for and social network user that represents their behaviour and interactions with other users. Then, based on the users' estimated feature vectors. Further more, this methodology is generic in that it can be found on a number of online social networking sites. It performed experiments on real data derived from Instagram and Twitter to demonstrate the suitability of the proposed process

5) Spam area of Twitter traffic:

Law Implementation Offices cover a pivotal job in the examination of open information and need accurate strategies to channel inconvenient information. In a real situation, Law Implementation Organizations investigate Informal communities, for ex: Twitter Manage access and account information. Tragically, allying the tremendous measure of social users, there are individuals that utilization micro blogs for badgering others or spreading malign substance. Client's order and spammer's distinguishing proof is a useful technique for reduce Twitter traffic from uninformative s. This work proposes a structure that abuses a non-uniform element testing inside a dim box AI Framework, utilizing a variation of the Irregular Timber lands Calculation to distinguish spammers inside Twitter traffic. Investigation are made on a well known Twitter data set and on another data set of Twitter clients. The new given Twitter data set is made of clients named as spammers or legitimate users, narrated by 54 highlights

CHAPTER 3

AIM AND SCOPE

3.1 Aim:

The aim of this paper is to identify fake user detection on Twitter and to present a framework by classifying these approaches into several categories. For classification, we have identified four means of reporting spammers that can be helpful in identifying fake identities of users. Spammers can be identified based on: (i) fake content, (ii) URL based spam detection, (iii) detecting spam in trending topics, and (iv) fake user identification.

Moreover, the analysis also shows that machine learning-based techniques can be effective for identifying fake user on Twitter. However, the selection of the most feasible techniques and methods is highly dependent on the available data.

3.2 Objective:

The objectives of the project are provided below:

Objective 1: Performing secondary research to identify similar methods used in identifying the Twitter spam accounts

Objective 2: Using Twitter API to collect raw account details and clean up the data to perform the analysis.

Objective 3: Performing analysis of the clean data using binary classification methods such as Support Vector Machine (SMV), Naïve Bayes, and Decision Tree, and Random Forest

Objective 4: Comparing the analysis and suggesting the binary classifier that can be used to detect spam account on twitter.

3.3 Research questions:

Objectives	Research Questions
Objective 1: Performing secondary research to identify the similar methods used in identifying the Twitter Spam accounts	Research question 1: What are the current methods used to identify the spam accounts in Twitter?
Objective 2: Using Twitter API to collect raw account details and clean up the data to perform the analysis.	Research question 2: What details of the account need to be collected from Twitter to identify whether the account is spam or not?
Objective 3: Performing analysis of the clean data using binary classification methods such as Support Vector Machine (SMV), Naïve Bayes, and Decision Tree, and Random Forest	Research question 3: What machine learning algorithm can be used to achieve the appropriate results in detecting spam accounts on twitter? Research question 4: What are the properties or factors that distinguish spam accounts from non-real accounts?
Objective 4: Comparing the analysis and suggesting the binary classifier that can be used to detect the spam account on Twitter	Research question 5: What is the best binary classifier to detect the spam accounts on Twitter.

3.4 Report Structure:

The report structure is provided below

Section 2 discusses the background study of the project.

Section 3 provides a detailed information regarding the methodology used for this project.

Section 4 discusses the data collection and categorisation

Section 5 focuses on the data processing steps involved in the project

Section 6 is the complete analysis of the data results.

Section 7 provides the discussion regarding the results derived in this project.

Section 8 contains the conclusion and future works

CHAPTER 4

SYSTEM DESIGN AND METHODOLOGY

4.1 EXISTING SYSTEM

- Tingminet *al.* provide a survey of new methods and techniques to identify Twitter spam detection. The above survey presents a comparative study of the current approaches.
- On the other hand, S. J. Somanet. *al.* conducted a survey on different behaviors exhibited by spammers on Twitter social network. The study also provides a literature review that recognizes the existence of spammers on Twitter social network.
- Despite all the existing studies, there is still a gap in the existing literature. Therefore, to bridge the gap, we review state-of-the-art in the spammer detection and fake user identification on Twitter

4.1.1 DISADVANTAGES EXISTING SYSTEM

No efficient methods used.

No real time data's used.

More complex

4.2 PROPOSED SYSTEM

- The aim of this paper is to identify fake user detection on Twitter and to present a framework by classifying these approaches into several categories. For classification, we have identified four means of reporting spammers that can be helpful in identifying fake identities of users. Spammers can be identified based on: (I) fake content, (ii) URL based spam detection, (iii) detecting spam in trending topics, and (iv) fake user identification.
- Moreover, the analysis also shows that machine learning-based techniques can be effective for identifying fake user on Twitter. However, the selection of the

most feasible techniques and methods is highly dependent on the available data.

4.2.1 ADVANTAGES OF PROPOSED SYSTEM

This study includes machine learning methodology proposed using real time datasets and with different characteristics and accomplishments.

The proposed system is more effective and accurate than other existing systems.

Tested with real time data's.

4.3 REQUIREMENT ANALYSIS

Requirement analysis, also called requirement engineering, is the process of determining user expectations for a new modified product. It encompasses the tasks that determine the need for analysing, documenting, validating and managing software or system requirements. The requirements should be documentable, actionable, measurable, testable and traceable related to identified business needs or opportunities and define to a level of detail, sufficient for system design.

4.3.1 FUNCTIONAL REQUIREMENTS

It is a technical specification requirement for the software products. It is the first step in the requirement analysis process which lists the requirements of particular software systems including functional, performance and security requirements. The function of the system depends mainly on the quality hardware used to run the software with given functionality.

Usability

It specifies how easy the system must be use. It is easy to ask queries in any format which is short or long, porter stemming algorithm stimulates the desired response for user.

Robustness

It refers to a program that performs well not only under ordinary conditions but also under unusual conditions. It is the ability of the user to cope with errors for irrelevant queries during execution.

Security

The state of providing protected access to resource is security. The system provides good security and unauthorized users cannot access the system there by providing high security.

Reliability

It is the probability of how often the software fails. The measurement is often expressed in MTBF (Mean Time Between Failures). The requirement is needed in order to ensure that the processes work correctly and completely without being aborted. It can handle any load and survive and survive and even capable of working around any failure.

Compatibility

It is supported by version above all web browsers. Using any web servers like localhost makes the system real-time experience.

Flexibility

The flexibility of the project is provided in such a way that is has the ability to run on different environments being executed by different users.

Safety

Safety is a measure taken to prevent trouble. Every query is processed in a secured manner without letting others to know one's personal information.

4.3.2 NON- FUNCTIONAL REQUIREMENTS

Portability

It is the usability of the same software in different environments. The project can be run in any operating system.

Performance

These requirements determine the resources required, time interval, throughput and everything that deals with the performance of the system.

Accuracy

The result of the requesting query is very accurate and high speed of retrieving information. The degree of security provided by the system is high and effective.

Maintainability

Project is simple as further updates can be easily done without affecting its stability.

Maintainability basically defines that how easy it is to maintain the system. It means that how easy it is to maintain the system, analyse, change and test the application. Maintainability of this project is simple as further updates can be easily done without affecting its stability.

4.4 SYSTEM DESIGN AND TESTING PLAN

4.4.1 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?

- How the data should be arranged or coded?

- The dialog to guide the operating personnel in providing input.

- Methods for preparing input validations and steps to follow when error occur.

4.4.2 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making. The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the Future.

- Signal important events, opportunities, problems, or warnings.

- Trigger an action.

- Confirm an action.

4.5 SYSTEM REQUIREMENTS:

4.5.1 HARDWARE REQUIREMENTS:

- System : Pentium Dual Core.

- Hard Disk : 120 GB.

- Monitor : 15" LED

- Input Devices : Keyboard, Mouse

- Ram : 4 GB.

4.5.2 SOFTWARE REQUIREMENTS:

- Operating system: Windows 7/10.

- Coding Language :Python

- Tool : Pi-champ

- Database : MYSQL

4.6 MODULE DESCRIPTIONS:

Admin Module:

In the first module, we develop the Online Social Networking (OSN) system module. We build up the system with the feature of Online Social Networking System, Twitter. Where, this module is used for admin login with their authentication.

Data Collection:

We will be using a Python Library called *Tweepy* to connect to the Twitter API and collect the data. We download tweets containing certain key words, to incorporate the words or hash tags that contain relevant keyword related to fake users.

Some of the most important fields are:

- *text*, which contains the text included in the tweet.
- *created_at*, which is a timestamp of when the tweet was created.
- *user*, which contains information about the user that created the tweet, like the username and user id.

Train and Test:

- We present the proposed framework for metadata features are extracted from available additional information regarding the tweets of a user, whereas content-based features aim to observe the message posting behavior of a user and the quality of the text that the user uses in posts.

Machine Learning Technique:

- The number of features, which are associated with tweet content, and the characteristics of users are recognized for the detection of spammers. These features are considered as the characteristics of machine learning process for categorizing users, i.e., to know whether they are spammers or not.
- In order to recognize the approach for detecting spammers on Twitter, the labelled collection in pre-classification of fake user and legitimate user has been done. Next, those steps are taken which are needed for the construction of labeled collection and acquired various desired properties.
- In other words, steps which are essential to be examined to develop the collection of users that can be labelled as fake user or legitimate user. At the

end, user attributes are identified based on their behavior, e.g., who they interact with and what is the frequency of their interaction.

- In order to confirm this instinct, features of users of the labelled collection has been checked. Two attribute sets are considered, i.e., content attributes and user behavior attributes, to differentiate one user from the other.

Detection of Fake User:

- In this module, we implement the collection of tweets with respect to trending topics on Twitter. After storing the tweets in a particular file format, the tweets are subsequently analyzed.
- Labelling of fake user is performed to check through all datasets that are available to detect the malignant.
- Feature extraction separates the characteristics construct based on the language model that uses language as a tool and helps in determining whether the user is fake or not.
- The classification of data set is performed by shortlisting the set of tweets that is described by the set of features provided to the classifier to instruct the model and to acquire the knowledge for spam detection.
- The fake user detection uses the classification technique to accept tweets as the input and classify the fake user and legitimate user.

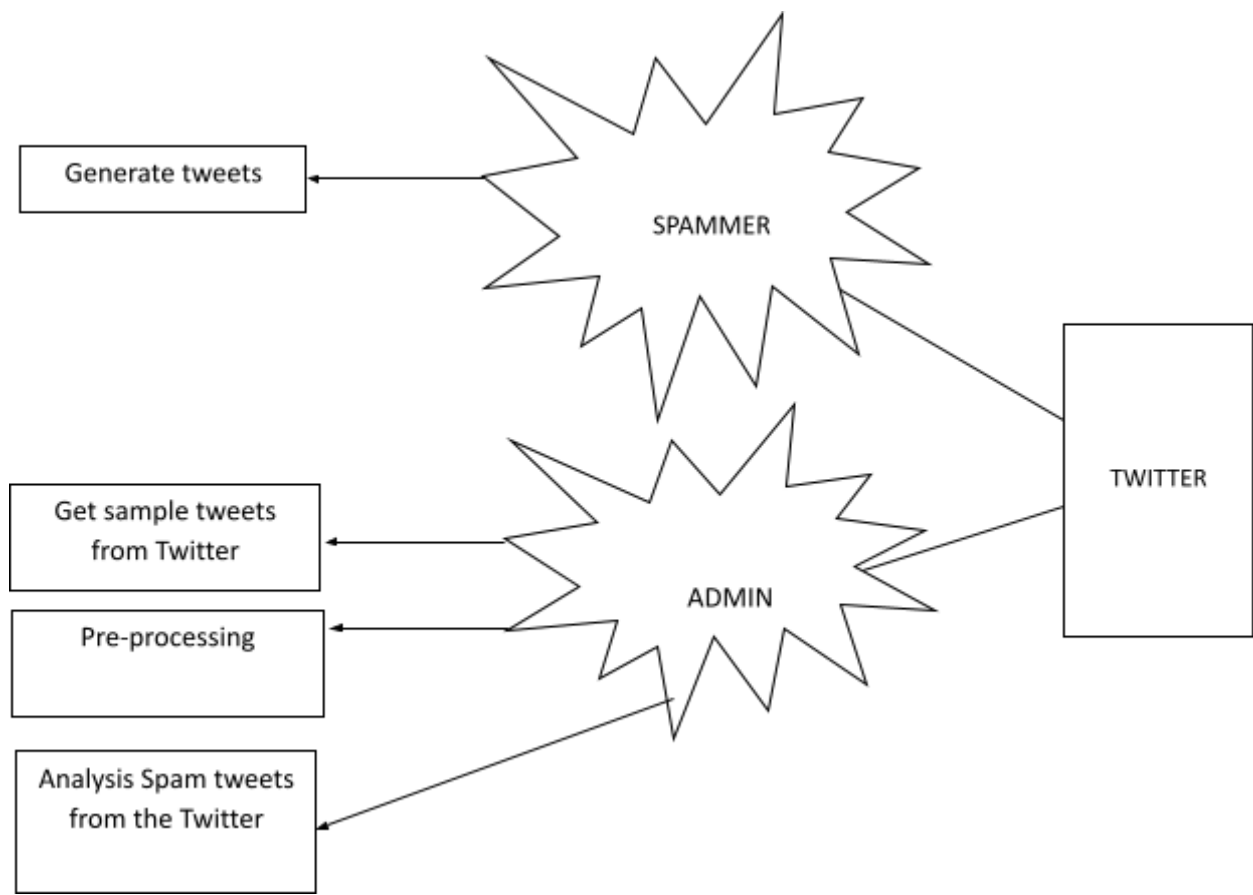


Fig 4.6 Architectural Design

Data flow diagram:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.

3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

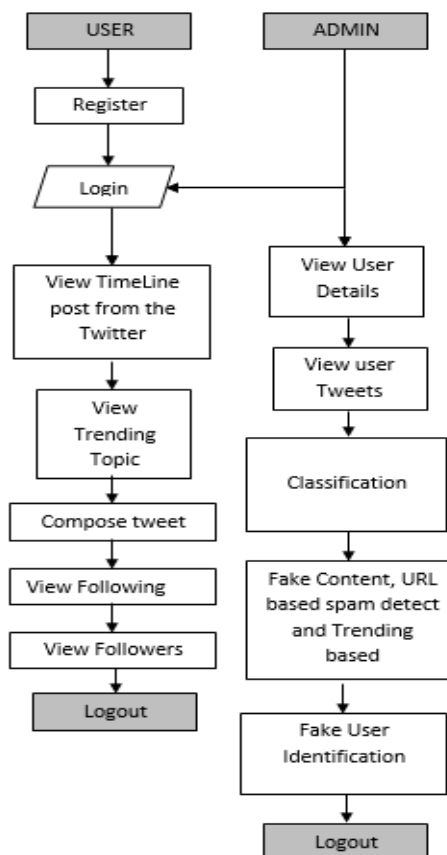


Fig 4.6.1 Data flow diagram

UML diagrams:

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major

components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

Goals:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

use case diagram:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

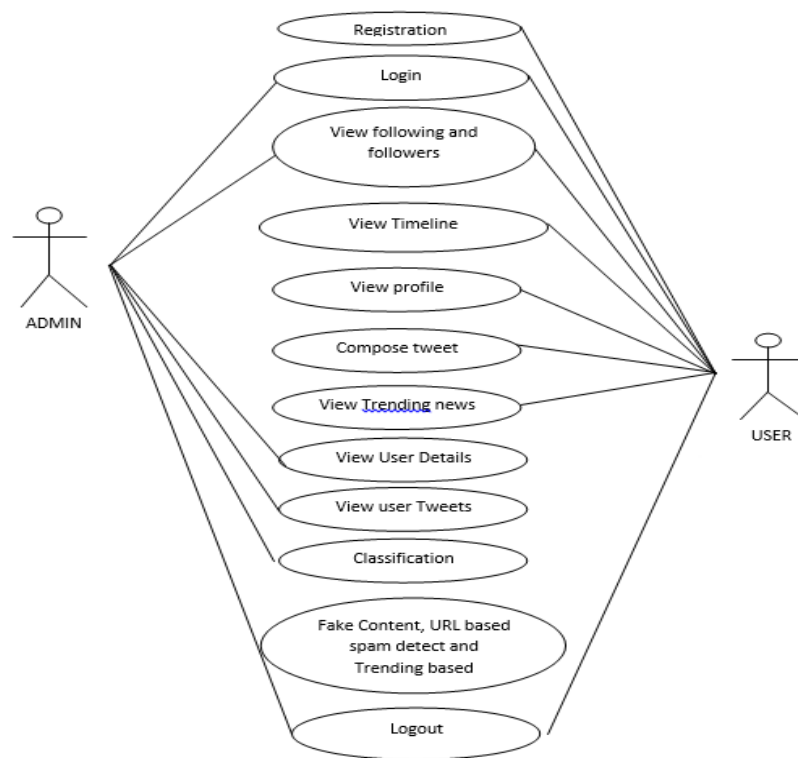


Fig 4.6.2 Use Case Diagram

Class diagram:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

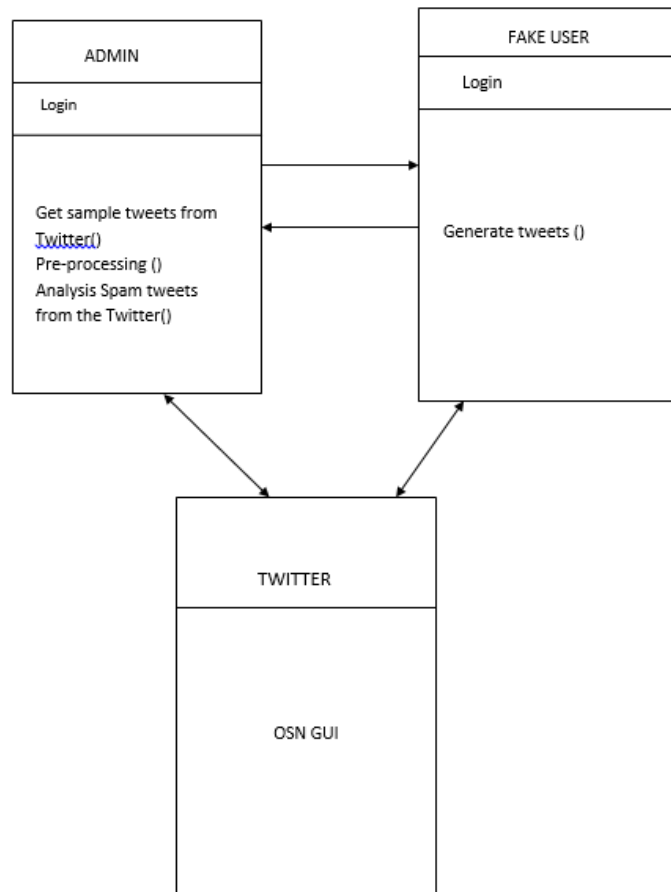


Fig 4.6.3 Class Diagram

Sequence diagram:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

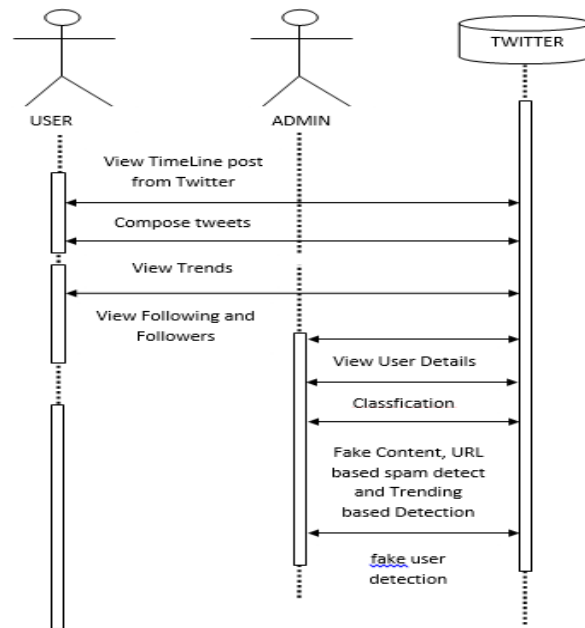


Fig 4.6.4 Sequence Diagram

Activity diagram:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

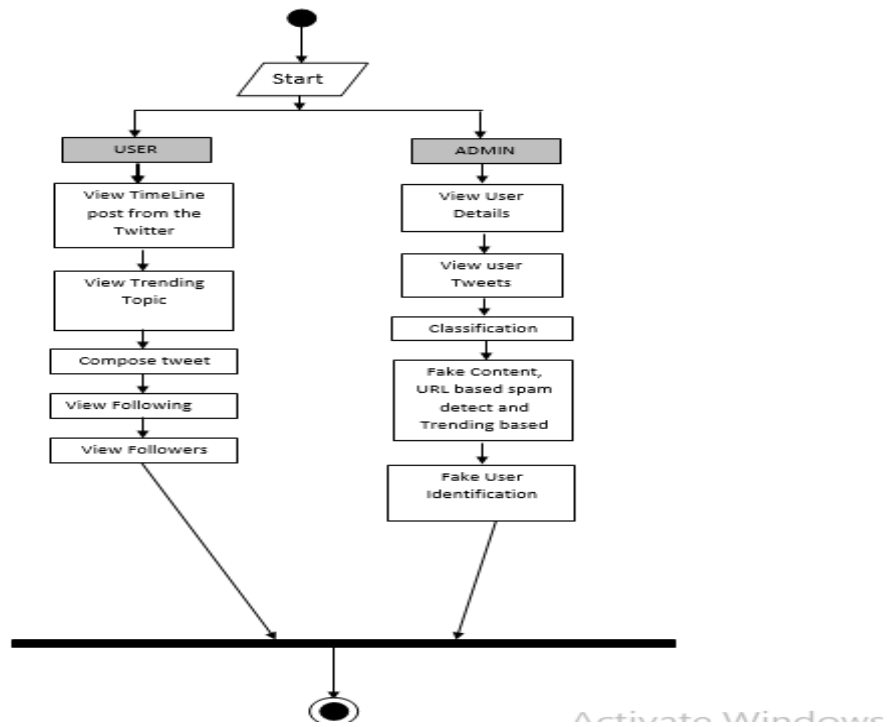


Fig 4.6.5 Activity Diagram

CHAPTER 5

RESULT AND DISCUSSION

This chapter presents the results that are obtained from the experiment. In this work, have developed and compared various algorithms based on machine learning. more specially work tested in random forest , Convolutional Neural Networks and combined both and also regression type but that algorithms are much more effective in other algorithms ,even in sequence data analysis but not in bit-coin prediction, why because when work did with random forest ,first it will convert in decision tree it will take high time to train data, in the same if work did in support vector machine (SVM) in this algorithm the main advantage is keras function .By using this function it is easy to calculate or manipulating the number functions, even it will take less time to train data when compared to other function and another advantage is time complexity is low when compared to Convolutional neural network and random forest, next about space complexity even in this category Support Vector Machine is very good. Finally if go to the accuracy, when work use support vector machine algorithm it was getting high when compared to other any algorithms so, in every case and support vector machine is very good and user friendly and even easy to understand.

Support vector machine is the classifier that maximizes the margin. The goal of a classifier in our example below is to find a line or $(n-1)$ dimension hyper-plane that separates the two classes present in the n -dimensional space. The margin is calculated as the perpendicular distance from the line to only the closest points. Only these points are relevant in defining the line and in the construction of the classifier. These points are called the support vectors. They support or define the hyperplane. Supervised machine learning model that uses classification algorithms for two-group classification problems. After giving an SVM model sets of labelled training data for each category, they're able to categorize new text. So you're working on a text classification problem. The support vector uses a mathematical function, often called a kernel function which is a math function that matches the new data to the best image

from the training data in order to predict the unknown picture's label

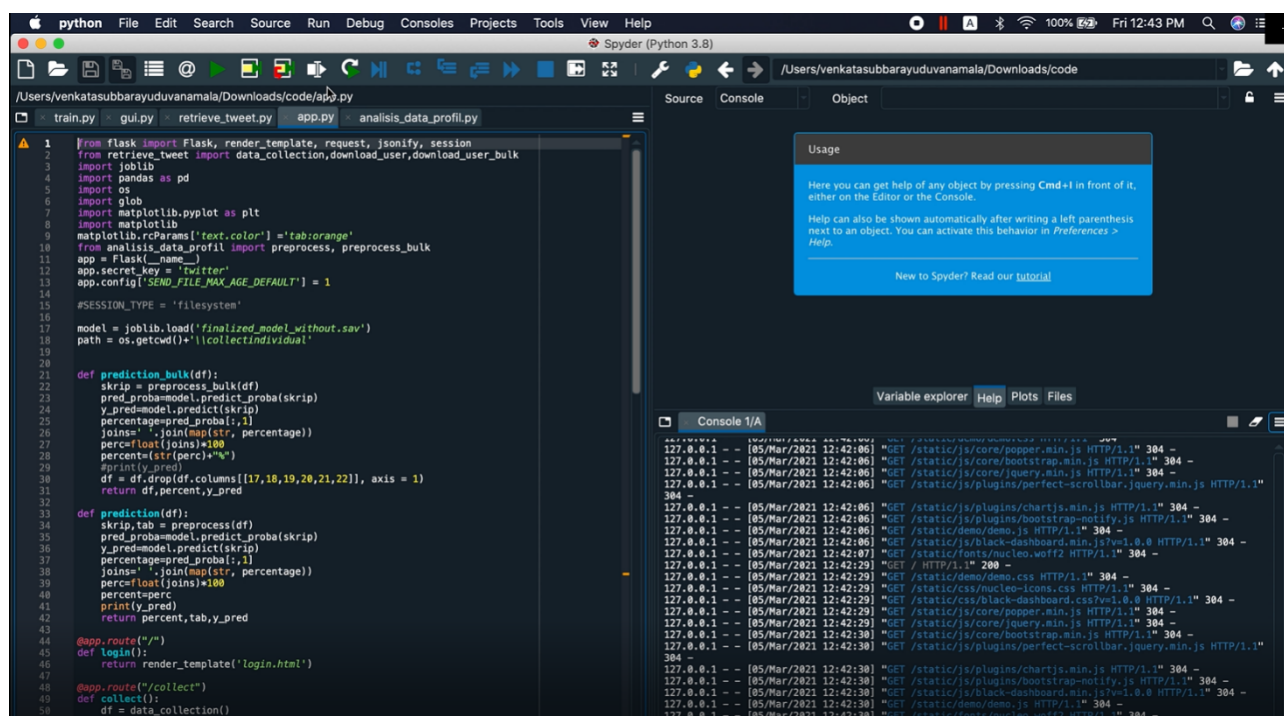


Fig 5.1 Importing Libraries

So by observing the above figure5.1 at the starting of the program all the libraries are imported .

The image shows a web browser window displaying the 'Spammer Detection & Blocking' application. The page has a sidebar with navigation links: FUD, SPAMMER DETECTION, DATA COLLECTION, TESTING, USER CHECK, and DETECTION. The main content area shows 'TESTING' results in a table.

1	827676697919180800	827676697919180800	Band Name Generator	band_name_gen	30	3	1	2
2	2602311594	2602311594	☆ Ruby-Dooby-Doo ☆	MoodyMrMembrane	59	6	137	133
3	135949983111151621	135949983111151621	Calefyb	CalefybTech	293	10	6	6799
4	3022689500	3022689500	Cha-Ching	TreJames76	108	0	169	6140
5	1262098367686967296	1262098367686967296	Stream Epik High on Spotify	coffeeNberries	0	0	46	42
6	1112454257372995584	1112454257372995584	Order of the G.A.C.R	MitchellMoffatt	3076	13	3598	152392
7	1243271238295998464	1243271238295998464	rakim-bot	BotRakim	2	0	1	0
8	175388536	175388536	ProTexting	ProTexting	17877	708	18924	885
9	1351323674079621120	1351323674079621120	Delaney Grey Management	DelaneyGreyMgmt	509	1	519	335
10	2642702533	2642702533	Kermit The Frog	ActualKermitBot	18	3	4	0

Fig 5.2 Trained Data

After collecting data set from the excel sheet we have to train the data

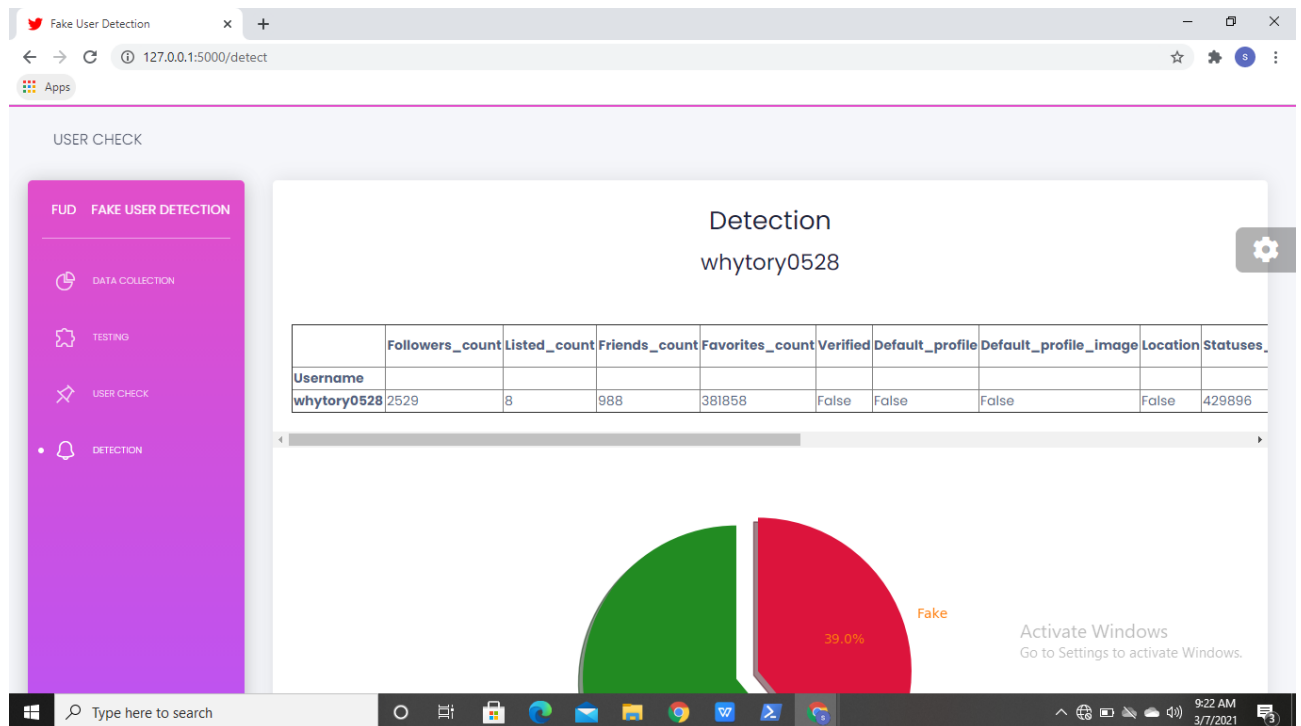


Fig 5.3 Detection

After completing the training then we can search the user after finding that user we can detect that whether that person is fake or legitimate.

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

CONCLUSION:

In this paper, we performed a review of techniques used for detecting spammers on Twitter. In addition, we also presented taxonomy of Twitter spam detection approaches and categorized them as fake content detection, URL based spam detection, spam detection in trending topics, and fake user detection techniques. We also compared the presented techniques based on several features, such as user features, content features, graph features, structure features, and time features. Moreover, the techniques were also compared in terms of their specified goals and datasets used. It is anticipated that the presented review will help researchers find the information on state- of-the-art Twitter spam detection techniques in a consolidated form. Despite the development of efficient and effective approaches for the spam detection and fake user identification on Twitter , there are still certain open areas that require considerable attention by the researchers. The issues are briefly highlighted as under: False news identification on social media networks is an issue that needs to be explored because of the serious repercussions of such news at individual as well as collective level .Another associated topic that is worth investigating is the identification of rumor sources on social media. Although a few studies based on statistical methods have already been conducted to detect the sources of rumors, more sophisticated approaches, e.g., social network-based approaches, can be applied because of their proven effectiveness.

REFERENCES

- [1] C. Chen, S. Wen, J. Zhang, Y. Xiang, J. Oliver, A. Alelaiwi, and M. M. Hassan, "Investigating the deceptive information in Twitter spam," *Future Gener. Comput. Syst.*, vol. 72, pp. 319–326, Jul. 2017.
- [2] I. David, O. S. Siordia, and D. Moctezuma, "Features combination for the detection of malicious Twitter accounts," in *Proc. IEEE Int. Autumn Meeting Power, Electron. Comput. (ROPEC)*, Nov. 2016, pp. 1–6.
- [3] M. Babcock, R. A. V. Cox, and S. Kumar, "Diffusion of pro- and anti-false information tweets: The black panther movie case," *Comput. Math. Org. Theory*, vol. 25, no. 1, pp. 72–84, Mar. 2019.
- [4] S. Keretna, A. Hossny, and D. Creighton, "Recognising user identity in Twitter social networks via text mining," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2013, pp. 3079–3082.
- [5] C. Meda, F. Bisio, P. Gastaldo, and R. Zunino, "A machine learning approach for Twitter spammers detection," in *Proc. Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2014, pp. 1–6.
- [6] W. Chen, C. K. Yeo, C. T. Lau, and B. S. Lee, "Real-time Twitter content polluter detection based on direct features," in *Proc. 2nd Int. Conf. Inf. Sci. Secur. (ICISS)*, Dec. 2015, pp. 1–4.
- [7] H. Shen and X. Liu, "Detecting spammers on Twitter based on content and social interaction," in *Proc. Int. Conf. Netw. Inf. Syst. Comput.*, pp. 413–417, Jan. 2015.
- [8] G. Jain, M. Sharma, and B. Agarwal, "Spam detection in social media using convolutional and long short term memory neural network," *Ann. Math. Artif. Intell.*, vol. 85, no. 1, pp. 21–44, Jan. 2019.
- [9] M. Washha, A. Qaroush, M. Mezghani, and F. Sedes, "A topic-based hidden Markov model for real-time spam tweets filtering," *Procedia Comput. Sci.*, vol. 112, pp. 833–843, Jan. 2017.
- [10] F. Pierri and S. Ceri, "False news on social media: A data-driven survey," 2019, arXiv:1902.07539. [Online]. Available: <https://arxiv.org/abs/1902.07539>

- [11] S. Sadiq, Y. Yan, A. Taylor, M.-L. Shyu, S.-C. Chen, and D. Feaster, "AAFA: Associative affinity factor analysis for bot detection and stance classification in Twitter," in *Proc. IEEE Int. Conf. Inf. Reuse Integr. (IRI)*, Aug. 2017, pp. 356–365.
- [12] M. U. S. Khan, M. Ali, A. Abbas, S. U. Khan, and A. Y. Zomaya, "Segregating spammers and unsolicited bloggers from genuine experts on Twitter," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 4, pp. 551–560, Jul./Aug. 2018.
- [13] Mccord, M. and Chuah, M., 2011, September. Spam detection on twitter using traditional classifiers. In *international conference on Autonomic and trusted computing* (pp. 175-186). Springer, Berlin, Heidelberg.
- [14] Perveen, N. Missen, M.M.S., Rasool, Q. and Akhtar, N., 2016. Sentiment Based Twitter Spam Detection. *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS*, 7(7), pp.568-573.
- [15] Wang, A.H., 2010, July. Don't follow me: Twitter spam detection. In *Proc. 5th International Conf. on Security and Cryptography (SECRYPT)*.
- [16] Wang, A.H., 2010, July. Don't follow me: Spam detection in twitter. In *Security and Cryptography (SECRYPT), Proceedings of the 2010 International Conference on* (pp. 1-10). IEEE.

SOURCE CODE:

Analysis of data profile

```
import pandas as
pd import tweedy
from time import
sleep import csv
from datetime import datetime, date, time, time
delta import datetime as dt
import ion
import NumPy as
np import jilbab
import
pickle
import re

def preprocess bulk(df):
    #df=pd.read_csv('coba.csv')
    df=pd.DataFrame(df)
    df["Created_at"]= pd.to_datetime(df["Created_at"])

    #Count age in days
    now = pd.Timestamp('now')
    df['age_in_days'] = (now -
```

```
df['Created_at']).dt.days #Number of  
tweets/age
```

```

df['Ratio Statuses_count/age']=(df['Statuses_count']/df['age_in_days']).round(3)

#Number of favourites/age

df['Ratio

Favorites/age']=(df['Favorites_count']/df['age_in_days']).round(3)

#Ratio friends/follower

df['Ratio Friends/Followers']=(df['Friends_count']/df['Followers_count']).round(3)

#Count length of bio

df['Length_of_Bio']=df['Description'].astype(str).str.le

n() #Measure reputation

df['Reputation']= df['Followers_count']/(df['Followers_count']+ df['Friends_count'])

df1=df.drop(['Geo_enabled','id_str','Created_at'], axis=1)

df1['URL'] = pd.notnull(df1['URL'])
df1['Location']=pd.notnull(df1['Location'])
bag_of_word
='bot|b0t|updates|hourly|automatically|generating|generated|every|compu
ter- generated|twitterbot|automated|'
df2=df1

df2['contains_bot_name']=df2['Description'].astype(str).str.contains(bag_of_word,

flags=re.IGNORECASE, regex=True)

df2=df2.drop(['Description'], axis=1)

df2['Length_of_Bio'].fillna(0, inplace = True)

df2['Reputation'].fillna(0, inplace = True)

df2['contains_bot_name'].fillna(False, inplace=True)

df2=pd.DataFrame(df2)

db=df2

```

```

db=pd.DataFrame(db)
db['URL'].fillna(False,inplace=True)
db.replace(np.inf, 0)
db.replace(-np.inf, 0)
db = db.replace(np.NaN, 0)
db = db.replace([np.inf, -np.inf], 0)
db=db.drop(['id'],
axis=1) db =
db[['Followers_count',
    'Listed_count',
    'Friends_count',
    'Favorites_count',
    'Verified',
    'Default_profile',
    'Default_profile_image', 'Location',
    'Statuses_count',
    'URL',
    'age_in_days',
    'Ratio
Statuses_count/age',
    'Ratio Favorites/age',
    'Ratio
Friends/Followers',

```

'Length_of_Bio',
'contains_bot_name',
'Reputation',

```

    ]]

    data_prob=db.values.tolist()

    st()

    db=pd.DataFrame(db)

    return data_prob


def preprocess(df):

    df=pd.read_csv('coba.csv')

    df=pd.DataFrame(df)

    df["Created_at"]= pd.to_datetime(df["Created_at"])


    #Count age in days

    now = pd.Timestamp('now')

    df['age_in_days'] = (now -

    df['Created_at']).dt.days #Number of

    tweets/age

    df['Ratio Statuses_count/age']=(df['Statuses_count']/df['age_in_days']).round(3)

    #Number of favourites/age

    df['Ratio Favorites/age']=(df['Favorites_count']/df['age_in_days']).round(3)

    #Ratio friends/follower

    df['Ratio Friends/Followers']=(df['Friends_count']/df['Followers_count']).round(3)

    #Count length of bio

    df['Length_of_Bio']=df['Description'].astype(str).str.le

```

```
n() #Measure reputation
```

```
df['Reputation']= df['Followers_count']/(df['Followers_count']+ df['Friends_count'])
```

```

df1=df.drop(['Geo_enabled','id_str','Created_at'],
axis=1) df1['URL'] = pd.notnull(df1['URL'])

df1['Location']=pd.notnull(df1['Location'])

bag_of_word
='bot|b0t|updates|hourly|automatically|generating|generated|every|compu
ter- generated|twitterbot|automated|'
df2=df1

df2['contains_bot_name']=df2['Description'].astype(str).str.contains(bag_of_word,
flags=re.IGNORECASE, regex=True)

df2=df2.drop(['Description'], axis=1)

df2['Length_of_Bio'].fillna(0, inplace = True)

df2['Reputation'].fillna(0, inplace = True)

df2['contains_bot_name'].fillna(False, inplace=True)

df2=pd.DataFrame(df2)

db=df2

db=pd.DataFrame(db)

db['URL'].fillna(False,inplace=True)

db.replace(np.inf, 0)

db.replace(-np.inf, 0)

db = db.replace([np.inf, -np.inf], 0)

db=db.drop(['id'],
axis=1) db =

db[['Followers_count',
'Listed_coun

```

t',
'Friends_co
unt',

```

        'Favorites_count',
        'Verified',
        'Default_profile',
        'Default_profile_ima
ge', 'Location',
        'Statuses_count',
        'URL',
        'age_in_days',
        'Ratio
Statuses_count/age',
        'Ratio Favorites/age',
        'Ratio
Friends/Followers',
        'Length_of_Bio',
        'contains_bot_name',
        'Reputation',
    ]]

data_prob=db.values.toli
st()

db=pd.DataFrame(db)

return data_prob,db

```

Retrieve tweet:

```
import tweepy
```

```
from tweepy import  
Cursor import  
unicodecsv  
import numpy as np
```

```

import pandas as pd

from unidecode import
unidecode import json

import os

try:
    os.stat(os.getcwd()+"\\collectindividu
al') path =
    os.getcwd()+"\\collectindividual'
except:
    os.mkdir(os.getcwd()+"\\collectindivid
ual') path =
    os.getcwd()+"\\collectindividual'

pd.set_option('display.max_columns', 1000)
pd.set_option('display.max_rows', 1000)
#pd.set_option('display.width', 1000)
#pd.set_option('max_colwidth', 30)
pd.set_option('colheader_justify',
'center')

# Authentication and connection to Twitter API.

```

consumer_key

='YBAUoHlkgYKQ9fDlJywpM1DGd'

consumer_secret ='67yxNaSftHGGKfbqYe4G8nYeXZIXSe9F3il68HML0SX5ysU047'

access_key ='120092298-stEI5XDozGPotvSYLFonT1HlbK4TAuofi9NzDm7x'

```
access_secret ='UiAKoMghPOCoRZ9D3BHFaKBHaVcnAJJLV3h0aBmMH8aoR'
```

```
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
```

```
auth.set_access_token(access_key, access_secret)
```

```
api = tweepy.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True)
```

```
words =
```

```
["#bot", "#b0t", "#updates", "#hourly", "#automatically", "#generating", "#generated", "#ever  
y"
```

```
, "#computer-generated", "#twitterbot", "#automat
```

```
ed"] date_since = "2019-11-01"
```

```
my_list_of_dicts
```

```
= []
```

```
my_demo_list =
```

```
[]
```

```
# Collect tweets
```

```
def data_collection():
```

```
    for search_words in
```

```
        words: tweets =
```

```
tweepy.Cursor(api.search,q=search_words,lang="en",since=date_since).items(5)
```

```
    print(search_words)
```

```
    for json_tweet in tweets:
```

```
        my_list_of_dicts.append(json_tweet._json)
```

```
    with open('tweet_json.txt','w') as file:
```

```
        file.write(json.dumps(my_list_of_dicts,
```

```
    indent=4))  
with open('tweet_json.txt', encoding='utf-8') as json_file:
```

```

all_data =
json.load(json_file)
for
each_dictionary in all_data:
    created_at =
    each_dictionary['created_at']
    tweet_id = each_dictionary['id']
    text = each_dictionary['text']
    retweet_count =
    each_dictionary['retweet_count']
    favorite_count =
    each_dictionary['favorite_count']
    place =
    each_dictionary['place']
    lang = each_dictionary['lang']
    username =
    each_dictionary['user']['screen_name']
    my_demo_list.append({'created_at':
    created_at,
        'tweet_id':
        str(tweet_id), 'text':
        str(text),
        'favorite_count':
        int(favorite_count),
        'retweet_count':
        int(retweet_count), 'place':

```

```

        str(place),

        'lang': str(lang),

        'username':str(username)

    })

#print(my_demo_list)

df = pd.DataFrame(my_demo_list)

df = df.drop_duplicates(subset ="tweet_id", keep
='first').reset_index(drop=True)

df =
df[['created_at','username','tweet_id','text','favorite_count','retweet_count','place','lang
']]

```

```

        #print(
df) return df
# for tweet in tweets:
#     print(tweet)
#     tweet =
pd.DataFrame() #
print(tweet)

```

```

def
    getExceptionMessage(ms
g): words = msg.split(' ')
    errorMsg = ""
    for index, word in
        enumerate(words): if index not
        in [0,1,2]:
            errorMsg = errorMsg + ' ' +
word errorMsg =
errorMsg.rstrip("\'}") errorMsg =
errorMsg.lstrip(" \'")

    return errorMsg

```

```
def
    download_user_bulk(dl_use
r): try:
    user_obj = api.get_user(dl_user)
```

```
with open(os.path.join(path,'coba'+dl_user+'.csv'), 'wb') as
```

```
file: writer = unicodcsv.writer(file, delimiter = ',',
```

```
quotechar = '"') # Write header row.
```

```
writer.writerow(["id",
```

```
    "id_str",
```

```
    "Name",
```

```
    "Username",
```

```
    "Followers_count
```

```
    t",
```

```
    "Listed_count",
```

```
    "Friends_count",
```

```
    "Favorites_count
```

```
    ", "Created_at",
```

```
    "Verified",
```

```
    "Default_profile",
```

```
    "Default_profile_image
```

```
    ", "Location",
```

```
    "Statuses_count",
```

```
    "Description",
```

```
    "URL",
```

```
    "Geo_enabled",
```

```
    ])
```

```
# Gather info specific to the current user.
```

```
user_info = [user_obj.id,
```

```

        user_obj.id_str,
        user_obj.name,
        user_obj.screen_name,
        user_obj.followers_count,
        user_obj.listed_count,
        user_obj.friends_count,
        user_obj.favourites_count,
        user_obj.created_at,
        user_obj.verified,
        user_obj.default_profile,
        user_obj.default_profile_im
age, user_obj.location,
        user_obj.statuses_count,
        user_obj.description,
        user_obj.url,
        user_obj.geo_enabled,
    ]

writer.writerow(user_i
nfo) # Show status if
success

print("Wrote tweets by %s to CSV." %
dl_user) except tweepy.TweepError as e:

    print ("\n"+str(e.api_code) + ":" + getExceptionMessage(e.reason)+"\n")

```

```

def
download_user(dl_use
r): try:
    user_obj = api.get_user(dl_user)
    with open('coba.csv', 'wb') as file:
        writer = unicodcsv.writer(file, delimiter = ',', quotechar =
        '"') # Write header row.

    writer.writerow(["id",
                    "id_str",
                    "Name",
                    "Username",
                    "Followers_count",
                    "t",
                    "Listed_count",
                    "Friends_count",
                    "Favorites_count",
                    ", "Created_at",
                    "Verified",
                    "Default_profile",
                    "Default_profile_image",
                    ", "Location",
                    "Statuses_count",

```

"Description",

"URL",

```

        "Geo_enabled",
    ])

# Gather info specific to the current user.
user_info = [user_obj.id,
              user_obj.id_str,
              user_obj.name,
              user_obj.screen_name,
              user_obj.followers_count,
              user_obj.listed_count,
              user_obj.friends_count,
              user_obj.favourites_count,
              user_obj.created_at,
              user_obj.verified,
              user_obj.default_profile,
              user_obj.default_profile_image, user_obj.location,
              user_obj.statuses_count,
              user_obj.description,
              user_obj.url,
              user_obj.geo_enabled,
          ]

writer.writerow(user_info) # Show status if
                           # success

print("Wrote tweets by %s to CSV." % dl_user)

```

```
except tweepy.TweepError as e:  
    print ("\n"+str(e.api_code) + ":" + getExceptionMessage(e.reason)+"\n")
```

SCREENSHOTS

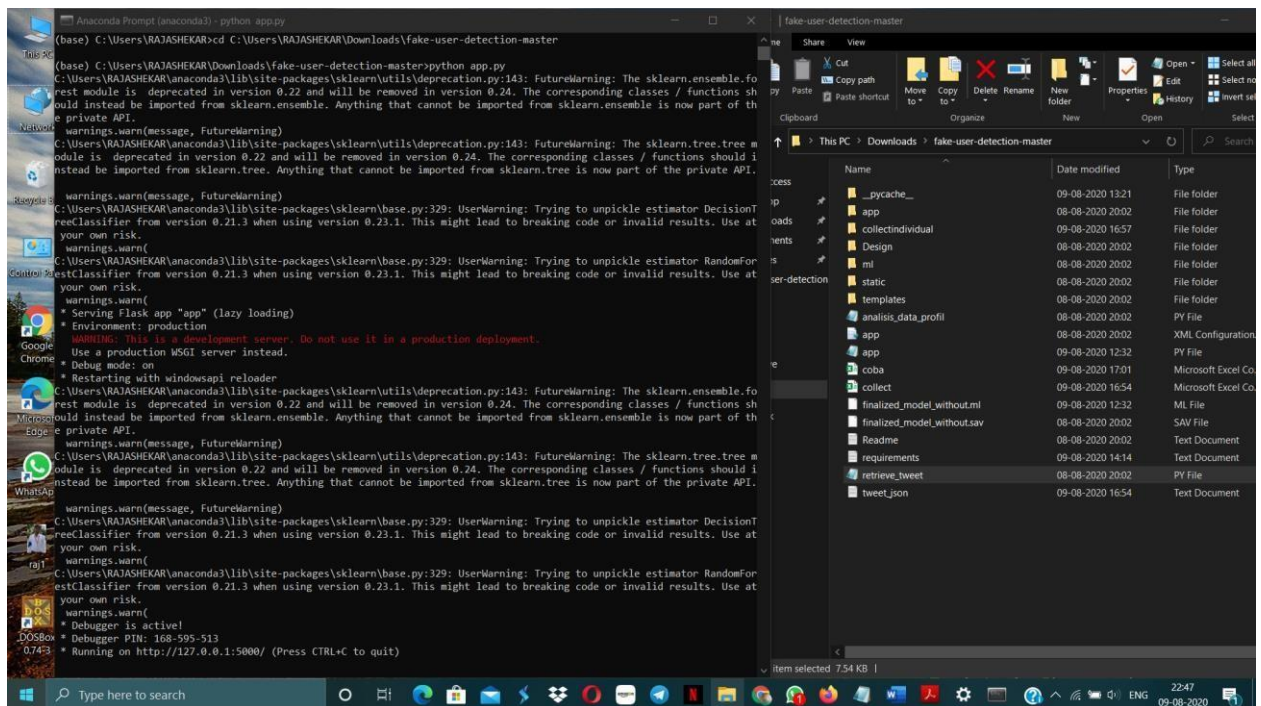


Fig 6.1 command prompt with reference address

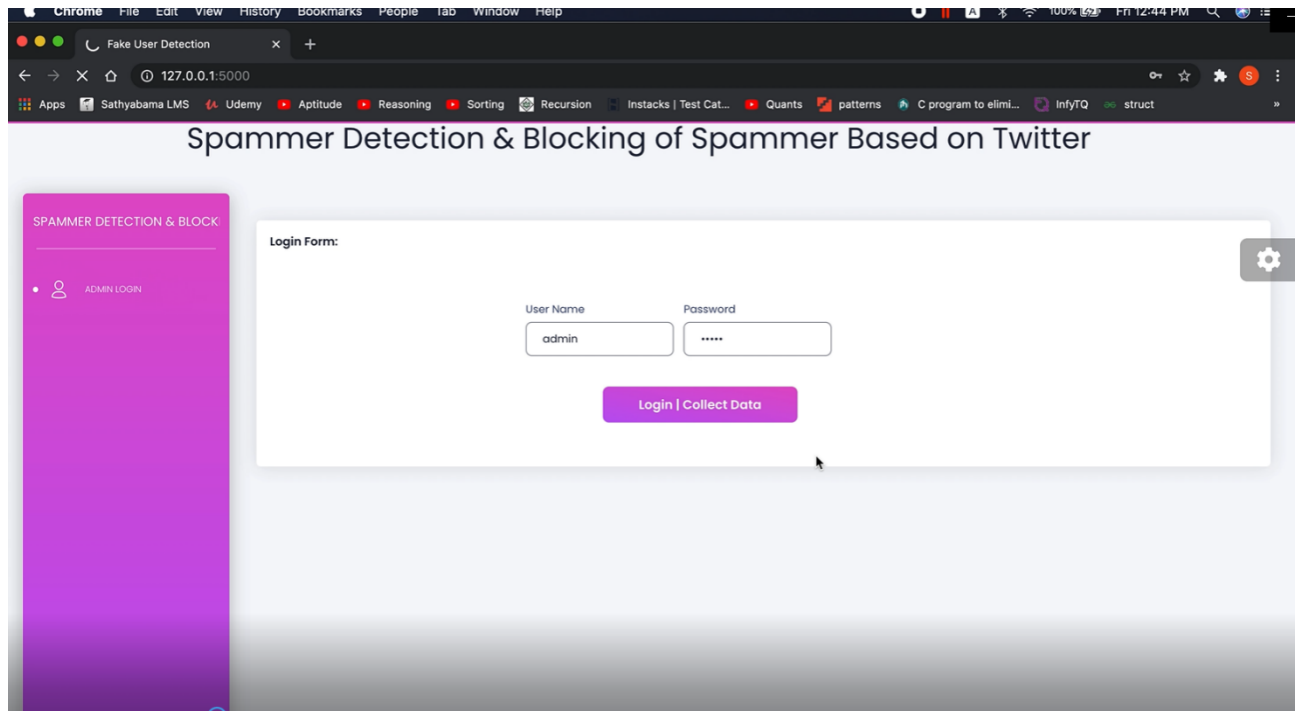


Fig 6.2 Login Page

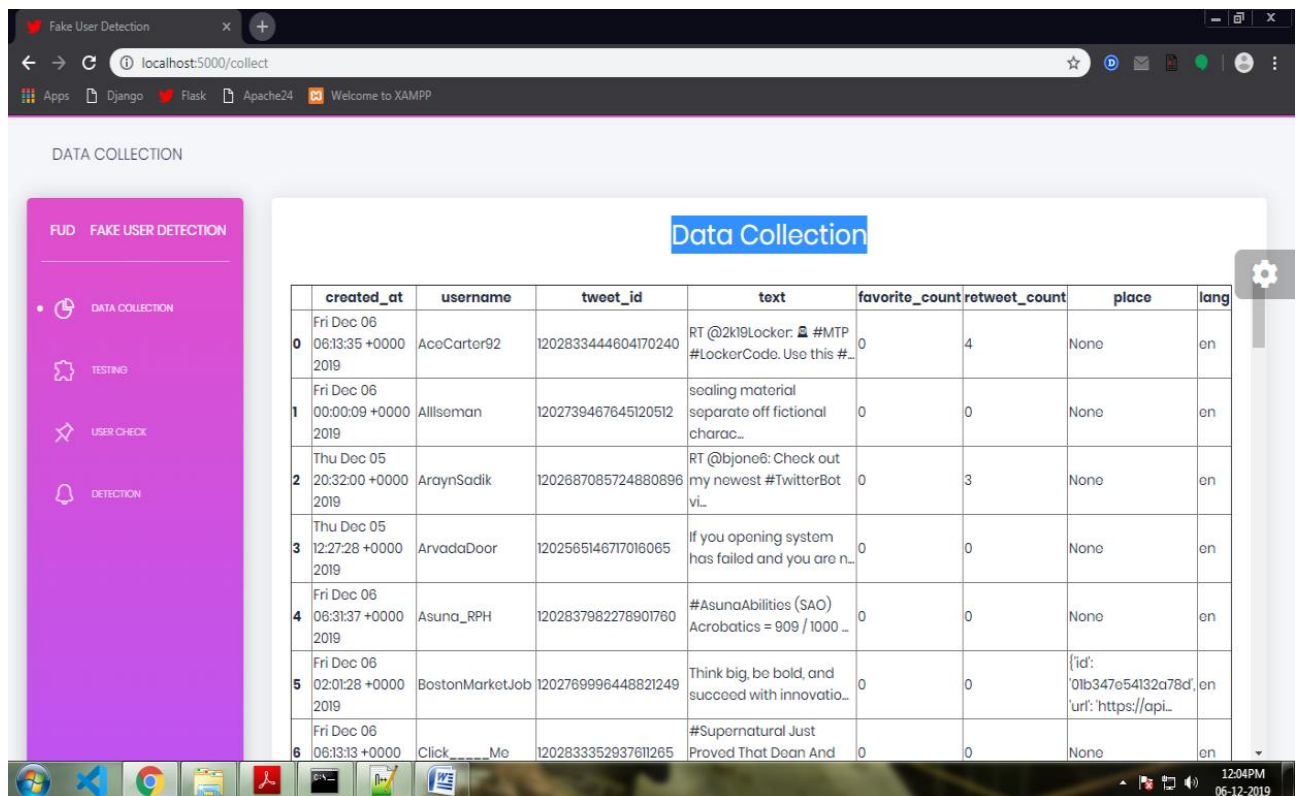


Fig 6.3 Data collection

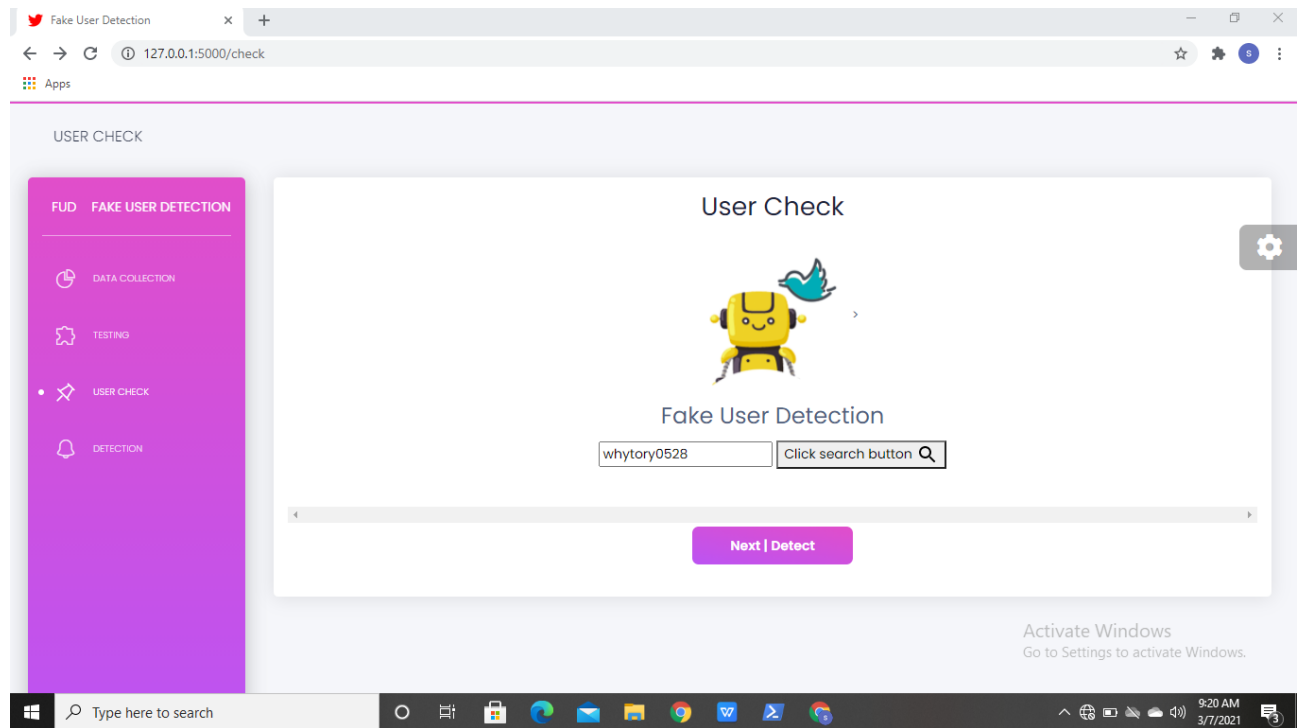


Fig 6.4 user check

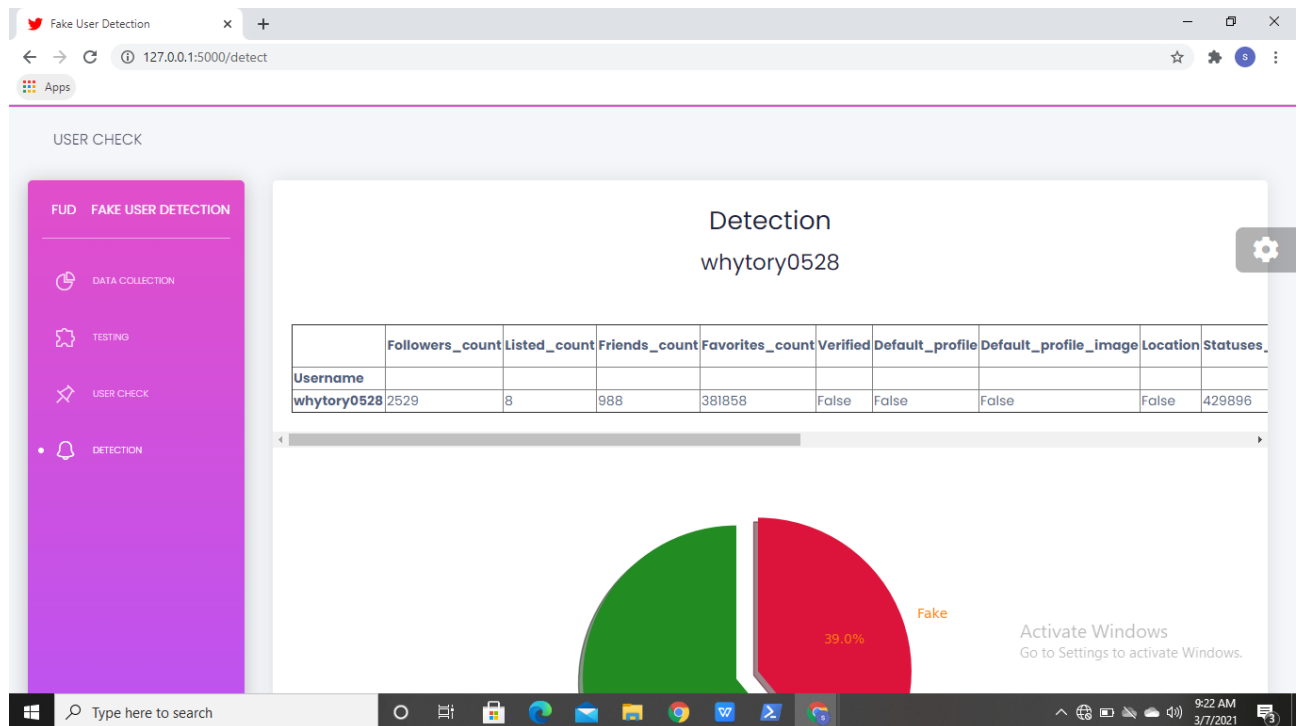


Fig 6.5 Output

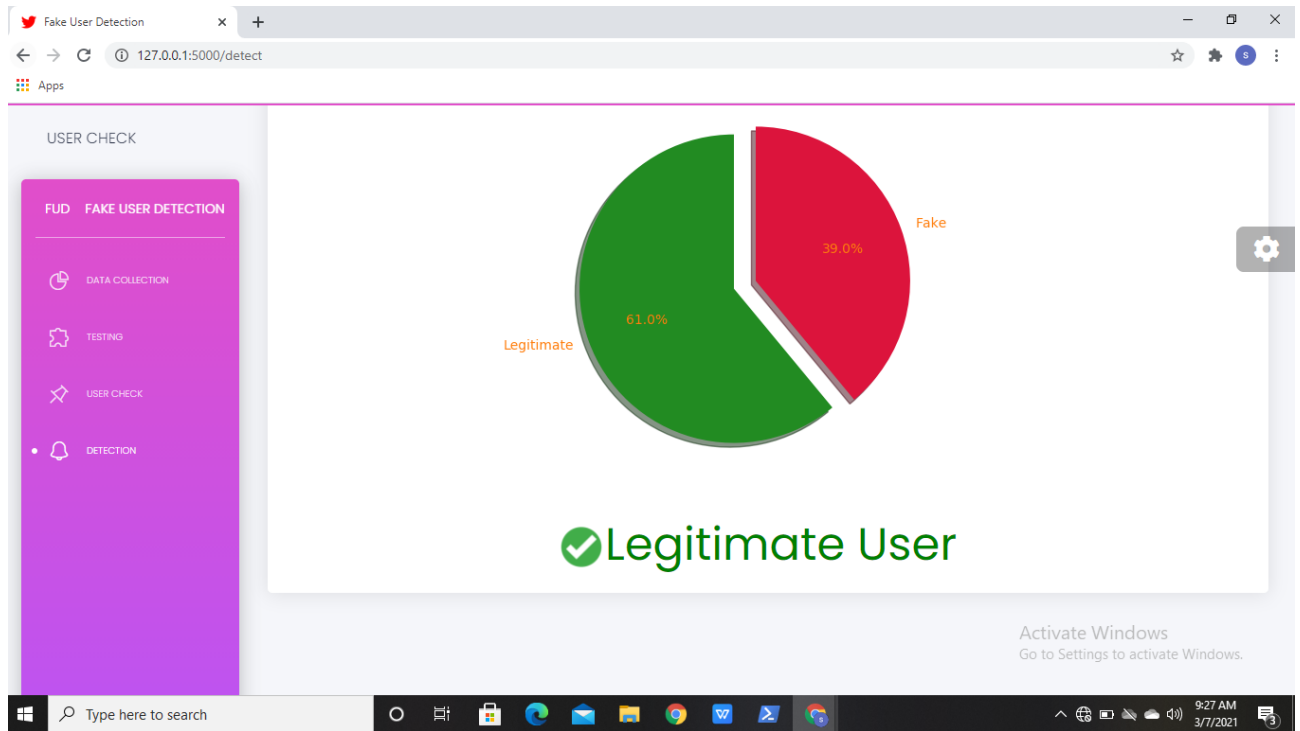


Fig 6.6 Fake or Legitimate user

Plagiarism report:

