LOCATION PREDICTION ON TWITTER USING MACHINE LEARNING TECHNIQUES

A major project report submitted in partial fulfillment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING

By

ABBAS HUSSAIN MUZAMMIL
(18841A0562)

Under the esteemed guidance of

Ms. K. Kavitha

(Head of the Department, IT)

AURORA'S TECHNOLOGICAL AND RESEARCH INSTITUTE

(Affiliated to JNTU, Hyderabad and approved by AICTE, New Delhi)

Parvathapur, Uppal, Hyderabad – 500098

2021-2022

AURORA'S TECHNOLOGICAL & RESEARCH INSTITUTE

(Affiliated to JNTU, Hyderabad)

Parvathapur, Uppal, Hyderabad-500098

DECLARATION

I hereby declare that the work described in this project, entitled '**LOCATION PREDICTION ON TWITTER USING MACHINE LEARNING TECHNIQUES"** which is being submitted by me in partial fulfilment for the award of Bachelor of Technology in Computer Science and Engineering to **AURORA'S TECHNOLOGICAL AND RESEARCH INSTITUTE** is the result of investigation carried by me under the guidance of **Ms. K. Kavitha, Head of the Department, IT.**

The work is original and has not been submitted for any degree of this or any other university.

Place: Hyderabad

Date:

Abbas Hussain Muzammil (18841A0562)

AURORA'S TECHNOLOGICAL & RESEARCH INSTITUTE

(Affiliated to JNTU, Hyderabad)

Parvathapur, Uppal, Hyderabad-500098

CERTIFICATE

This is to certify that the major project report entitled "**LOCATION PREDICTION ON TWITTER USING MACHINE LEARNING TECHNIQUES**" that is being submitted by **Abbas Hussain Muzammil (18841A0562)** in partial fulfilment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering to the Jawaharlal Nehru Technological University is a record of bonafide work carried out by him under our guidance and supervision.

GUIDE	**PROJECT COORDINATOR**
Ms. K. Kavitha	**Ms. V. Shilpa**
Head of the Department	Associate Professor
Department of IT	Department of CSE

HEAD OF THE DEPARTMENT	**DIRECTOR**
Ms. Durga Pavani	**Mr. Srikanth Jatla**

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

This work has been done during the project period and it was a very good opportunity to put theoretical knowledge into planned exercise with an aim to solve a real time problem and also to develop a confidence to face various practical situations.

I would like to express my gratitude to **Mr. Srikanth Jatla**, Director, Aurora's Technological and Research Institute for providing me congenial atmosphere and encouragement.

I express my sincere thanks to Head of the Department **Ms. Durga Pavani**, **HOD/CSE** for giving me the support and her kind attention and valuable guidance to me throughout the course.

I express my sincere thanks to Project Coordinator **Ms. V. Shilpa, Associate Professor and Project Coordinator** for helping me to complete my project work by giving valuable suggestions.

I convey thanks to my project guide **Ms. K. Kavitha**, **Head of the Department**, Department of Information Technology, for providing encouragement, constant support and guidance which was of a great help to complete this project successfully.

Abstract

Location prediction of users from online social media brings considerable research these days. Automatic recognition of location related with or referenced in records has been investigated for decades. As a standout amongst the online social network organization, Twitter has pulled in an extensive number of users who send a million of tweets on regular schedule. Because of the worldwide inclusion of its users and continuous tweets, location prediction on Twitter has increased noteworthy consideration in these days. Tweets, the short and noisy and rich natured texts bring many challenges in research area for researchers. In proposed framework, a general picture of location prediction using tweets is studied. In particular, tweet location is predicted from tweet contents. By outlining tweet content and contexts, it is fundamentally featured that how the issues rely upon these text inputs. In this work, wepredict the location of user from the tweet text exploiting machine learning techniques namely Naive Bayes, Support Vector Machine and Decision Tree.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Twitter is an American micro blogging and social networking service on which users post and interact with messages known as **tweets**. Registered users can post, like, and retweet tweets, however, unregistered users have the ability to only read tweets that are publicly available. Users interact with Twitter through browser or mobile frontend software, or programmatically via its APIs.
Users may post explicitly their location on the tweet text they post, whereas in certain cases the location may be available implicitly by including certain relevant criteria. Tweets are not a strongly typed language, in which users may post casual with emotion images. Abbreviated form of text, misspellings, and extra characters of emotional words makes tweet texts noisy.

1.1 Impact of social media

Social media platforms can be efficiently used for supply chain management by professionals, organizations, and retailers for their operations. Social networks like Twitter and Facebook allow users to update information on social activities that they undertake.

1.2 Tweet Classification

The major work in this project is a tweet classification system to classify tweets into high and low priority.
High Priority tweets are those, which ask for help, such as food, shelter, medicine etc. during a disaster.
Example: Two sample tweets of high priority. Tweet is in the English script, but the words used here are in the Hindi language. The translation of the tweet is, "Mr. @Narendramodi, heavy floods in Chapra Bihar, please arrange for administrative

help, people here are very worried.”

Low priority tweets convey information related to a disaster, such as “Rescue team has done a good job.”

Example A user thanks Twitter for its help during a disaster.

The other contribution of this project work is location prediction of high priority tweets.

If Geo-tagging information is missing in a tweet

- We predict location by making use of historical Geo-tagged tweets of the specificusers and build a Markov chain.

- The low priority tweets are analyzed to find the spread of the disaster. These mayalso be used to evaluate the performance of different agencies during a disaster.

1.3 Analyzing Tweets

The techniques applied for normal documents are not suited for analyzing tweets. The character limitations of tweets about 140 characters may make the tweet uneasy to understand, if the tweet context is not studied. The issue of location prediction related named as Geo location prediction is examined for Wikipedia and web page documents. Entity recognition from these formal documents has been researched for years. Different types of content and context handling on these documents are also studied extensively. However, the location prediction problem from twitter depends highly on tweet content. Users living in specific regions, locations may examine neighborhood tourist spots, landmarks and buildings and related events.

- **Home Location:** User’s residential address given by user or location given by user on account creation is considered as home location. Home location prediction can be used in various application namely recommendation systems, location-based advertisements, health monitoring, and polling etc. Home location can be specified as
    - administrative location

  - geo graphical location or co-ordinates.

- **Tweet Location:** Tweet location refers to the region from where the tweet is posted by user. By construing tweet location, one can get tweet person's mobility. Usually, home location is collected from user profile, whereas tweet location can be arrived from user's Geo tag.

1.4 Importance of predicting location

Social networks, like Twitter and Facebook, are valuable sources to monitor real-time events, such as earthquakes and epidemics. For this type of surveillance, user's location is an essential piece of information, but a substantial number of users choose not to disclose their geographical information.
However, characteristics of the users' behavior, such as the friends they associate with and the types of messages published can hint on their spatial location.

1.5 Applications of Location Prediction from Twitter

When composing tweets, user may make reference to the names of a few locations in tweet texts. Referenced location prediction may encourage better understanding of tweet content, and applications like

- recommendation systems
- location based advertisements
- health monitoring and polling etc.

The use of social media is being explored as a tool for disaster management bydevelopers, researchers, government agencies and businesses. The disaster-affected area requires both, cautionary and disciplinary measures For this we need a **computerized decision-making system during emergencies.**

Twitter plays a major role in informing people, acquiring their status of information, and also gathering information on different rescue activities taking place during both, natural disasters (tsunamis/floods) and man-made disasters (terrorist attack/food contamination)

1.5.1 Why Twitter?

Twitter provides the space where both official and common people can post their experiences and advice regarding disasters., which makes it a popular choice for disaster management.

- A lot of research work is going on to make this platform more suitable for disaster management.
- However, a more systematic study of social media is needed to improve public response.

We can appeal to the research community to devise methods to improve citizen-engagement during emergencies. Quick and accurate responses from the leaders during disaster may boost their personal political standing and Several agencies such as **BMKG in Indonesia** are actively engaged in providing updates and warnings to public through Twitter.
Social media is also used by various agencies to coordinate rescue efforts and help victims.

1.5.2 Impact of Twitter

Twitter is a micro blog where users send brief text messages, photographs and audio clips. Since users write small messages, they regularly send it and check for updates from others.
Twitter updates include social events such as

- Parties
- Cricket matche

- Disastrous events such as storms, heavy rainfall, earthquakes.
- Political camoaigns.
- Traffic jams etc.

A lot of work has been done to detect events, both social as well as disastrous from Twitter messages. Most disastrous event detection systems are confined to detect whether a tweet is related to the disaster or not, based on textual content. The related tweets are further used to warn and inform people about precautionary measures These tweets are also used to study the tweeting behavior of users during disasters.

We view Twitter not only as an awareness platform, but a place where people can ask for help during disaster.

The tweets asking for help need to be separated from other tweets related to the disaster. These tweets then can be used to guide the rescue personnel.
To help victims in need, one needs to have his/her exact location in their tweet, which is another important issue in emergency situations.

Distribution centers play a big role in helping victims. We can propose a **multi-objective location routing-model** to minimize the cost of opening a distribution center for relief routing.
The real time location estimation plays a big role in logistics, stockpiles, and medical supply planning. The growing number of location-based Social Networks provide the spatiotemporal data that has substantial potential to increase situational awareness and enhance, both planning and investigation.

1.5.3 Twitter Location Prediction Analysis

The analysis shows that only 26% users mention their location at a city level or below, and the remaining are mostly a country name, or even words with not much meaning, such as Wonderland.
Only 0.42% tweets have geo-tags, but it is found that about 3.17% tweets are geo-

tagged.

These analyses reveal that Twitter has limited applicability as a location-based sensing system.

1.5.4 Utilizing GPS

The rise of mobile Internet users in the last couple of years has significantly increased the number of mobile twitter users.

According to a report by statista (2020), the mobile Internet users in India will be 749 million by the end of 2020. The same report also highlights the fact that in rural areas, 39% of users are using social media, whereas in urban areas, this percentage is much higher. Mobile Twitter users can switch on and off their geo-tagging, as and when preferred.

The battery power of smart phones plays a significant role here, as the **global positioning system (GPS)** consumes significant amount of battery power.

Users prefer switching off their GPS to save power.

On the other hand, applications such as taxi hiring services and e-commerce sites such as flipkart.com require GPS to work properly. The analysis of mobile Twitter users thus shows some tweets with geo-tagging, and others without geo-tagging.

1.5.4 Down Fall of Geo Tag

During emergencies, people want to preserve the battery power of their phones. Hence, tweets with geo-tags will be very few on such occasions.

1.5.5 Challenges in a Country

For Example: India is a multilingual country, where English is used as the main language for communicating on social media websites. However, users of these sites also use their regional languages.

Hence, event detection in the Indian context also needs to identify variations in the language used.

CHAPTER 2

LITERATURE SURVEY

2.1 Existing System

In the Existing system to the problem of finding location from social media content. The Social Networks from and motivated by Term frequency (TF) and inverse document frequency (IDF), they arrived Inverse City Frequency (ICF) and Inverse Location Frequency (ILF) respectively.

They raked the features by using these frequency values and TF then by TF values. From this they arrived those local words spread in document in few places and have high ICF and ILF values.

They approached model for identifying local words indicative or used in certain locations only. They aimed to identify automatically by ranking the local words by their location, and they find their degree of association of location words associated to particular location or cities.

As a standout amongst the online social network organization, Twitter has pulled in an extensive number of users who send a million of tweets on regular schedule. Because of the worldwide inclusion of its users and continuous tweets, location prediction on Twitter has increased noteworthy consideration in these days. In this work, we predict the location of user from the tweet text exploiting machine learning techniques namely naive bayes, Support Vector Machine and Decision Tree [1].

Social networks, like Twitter and Facebook, are valuable sources to monitor Real - time events, such as earthquakes and epidemics. For this type of surveillance, the user's location is an essential piece of information, but a substantial number of users choose not to disclose their geographical information. However, characteristics of the users' behavior, such as the friends they associate with and the types of messages published can hint on their spatial location. In this, we present a method to infer the spatial location of Twitter users. Unlike the approaches presented so far, we incorporate two sources of information to learn the geographical position: the text posted by users and their friendship network. We propose a probabilistic approach that jointly models the geographical labels and the Twitter texts of the users organized in the form of a graph representing the friendship network. We use Markov random field probability model to represent the network and learning is carried out through a Markov chain Monte Carlo simulation technique to approximate the posterior probability distribution of the missing geographical labels. We demonstrate the utility of this model in a large dataset of Twitter users, where the ground truth is the location given by GPS dispositive. The method is evaluated and compared to two baseline algorithms that user these two types of information separately, the accuracy rates obtained are significantly better than those of the baseline methods. Index Terms—Network Learning; Geographic Targeting; Geolocation Estimation; Spatial Data Mining [2].

Getting data comes as the second step in any data science/machine learning project lifecycle, right after framing the problem you want to solve, which would make this step be the backbone of the rest of the phases. Also, social media are great places to collect data, especially for competitor analysis, topic research, sentiment analysis, etc. This article aims to perform a step-by-step implementation on how to get credentials and the implementation on a simple use case [3].

Geo location prediction is vital to geospatial applications like localized search and local event detection. Predominately, social media geo location models are based on full text data, including common words with no geospatial dimension (e.g., today) and noisy strings (tomorrow), potentially hampering prediction and leading to slower/more memory-intensive models. In this paper, we focus on finding location indicative words (LIWs) via feature selection, and establishing whether the reduced feature set boosts geo location accuracy. Our results show that an information gain ratio-based approach surpasses other methods at LIW selection, outperforming state-of the art geo location prediction methods by 10.6% in accuracy

10.6% in accuracy and reducing the mean and median of prediction error distance by 45km and 209km, respectively, on a public dataset. We further formulate notions of prediction confidence, and demonstrate that performance is even higher in cases where our model is more confident, striking a trade-off between accuracy and coverage. Finally, the identified LIWs reveal regional language differences, which could be potentially useful for lexicographers [4].

From the book we can Understand the importance of applying spatial relationships in data science, Select and apply data layering of both raster and vector graphics, Apply location data to leverage spatial analytics, Design informative and accurate maps, automate geographic data with Python scripts, Explore Python packages for additional functionality, Work with atypical data types such as polygons, shape files, and projection, Understand the graphical syntax of spatial data science to stimulate curiosity [5].

You are Here is a compelling read perfect for anyone interested in learning about the history of location technology. This book examines how having constant access to location data has come to be a part of modern life, and highlights the key players in the development of location and mapping technologies. Informative and interesting, you are Here discusses how solving one problem can sometimes usher in a new set of concerns [6].

2.2 Proposed System

Live stream of twitter data is collected as dataset using authentication keys.

The aim of proposed system is to predict the user location from twitter content considering

– user home location,

-- tweet location

-- tweet content.

To handle this, we used three machine learning approaches to make prediction easier and
finding the best model amongst them. Live tweet stream from twitter for keyword "apple" is collected and stored in Tweet table.

live twitter data can be collected by registering a consumerkey, consumer_secret, access token, access token secret for authentication and collecting live stream of tweets. we have collected more than 1000 tweets of particular keywords such as Indian city hashtag names. We can also search tweets based on hashtag.

2.2 System Study

2.3.1 Feasibility Study

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- **Economic Feasibility:** This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.
- **Technical Feasibility:** This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed systemmust have a modest requirement, as only minimal or null changes are required for implementing this system.
- **Social Feasibility:** The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must

be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER 3

REQUIREMENT SPECIFICATION

3.1 Software Requirements

Following are the Software's required followed by their detailed description

3.1.1 Python

- Python is a high-level, general-purpose and a very popular programming language.
- Python programming language (latest Python 3) is being used in web development, Machine Learning applications, along with all cutting-edge technology in Software Industry.
- Python allows programming in Object-Oriented and Procedural paradigms.
- Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
- Python is dynamically typed and has automatically garbage been collected.
- It supports multiple programming paradigms, including procedural, object-oriented, and functional programming

.

3.1.2 Operating System

An operating system (OS) is system software that manages computer hardware, software resources, and provides common services for computer programs. The operating system should support Python 3.7.0 and all the libraries mentioned above.

3.1.3 Libraries Used

3.1.3.1 Pandas

Pandas is an open-source library that is used for data analysis and manipulation. It is a Python package that offers various data structures and operations for manipulating numerical data and time series. It is mainly popular for importing and analyzing data much easier. Pandas is fast and it has high-performance productivity for users. In this project we have used panda's version 0.25.3.

3.1.3.2 NumPy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. In this project we have used NumPy version 1.81.1.

3.1.3.3 Sklearn

Sklearn is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

3.1.4 Frameworks Used

3.1.3.4 Django

The Django web framework is a free, open-source framework that can speed up development of a web application being built in the Python programming language. Django is designed in such a way that encourages developers to develop websites fast, clean and with practical design. Django's practical approach to getting things done is where it stands out from the crowd. If we are planning to build a highly customize app, such as social media website, Django is one of the best frameworks to consider. Django strength lies in its interactionbetween users or its ability to share different types of media.

3.1.3.5 Why Django?

It allows developers to use modules for faster development.

It has built-in protection for some common security issues such as cross-site scripting, request forgery, click jacking, and SQL injection. django releases new security patches quite often and it immediately responds to the security vulnerabilities and alerts other frameworks.

3.1.5 Packages

3.1.3.6 Tweepy

Tweepy is an open-source Python package that gives you a very convenient way to access the Twitter API with Python.

3.1.3.7 Geopy

Geopy makes it easy for Python developers to locate the coordinates of addresses, cities, countries, and landmarks across the globe using third-party geocoders and other data sources.

3.2 Hardware Requirements

For developing the application, following are the Hardwar Requirements

- **Processor**: Intel i3 or above, AMD Ryzen 3 or above
- **RAM:** 8 GB Minimum
- **Hard Disk:** 5 GB free hard disk space.

CHAPTER 4

SOFTWARE REQUIREMENT ANALYSIS

4.1 Problem Definition

- Locating the location through GPS may sometimes not feasible as the GPS can be turned off.
- We can make use Python's **Geopy** web service for locating the coordinates of the location.
- With coordinates we can locate the exact location and verify the accuracy of the location using Machine Learning Algorithms.
- To fetch real time data without any extra space we can call tweepy - an api of twitter which gives tweet data.

4.2 Functional Requirements

4.2.1 Modules

4.2.1.1 Data Collection

Live tweet stream from twitter for keyword “apple” is collected and stored in 'twitter.json’ file. Live twitter data can be collected by registering a consumer_key, consumer_secret, access_token, access_token_secret for authentication and collecting live stream of tweets. We have collected more than 1000 tweets of particular keyword such as ‘Chennai, Mumbai and Kerala’. The information extracted from live includes tweetid, name, screen_name, tweet_text, HomeLocation, TweetLocation, MentionedLocation, Lvalue.

4.2.1.2 Data Preprocessing

- Extra characters are removed from tweet text.
- Capitalize all words to find for geo location.
- Remove the tweet if user home location not mentioned.
- Mention home location in tweet location, if user tweet location is null.
- Removes tweets if no location is mentioned in tweet text.
- Final extract geodata from tweet text.

4.2.1.3 Search Tweets

In this module we enter tweet content and then system will classify the tweet content based on the dataset from tweepy API .

4.2.1.4 Geopy

Geopy makes it easy for Python developers to locate the coordinates of addresses cities, countries, and landmarks across the globe using third-party geo coders and other data sources. The geopy package is not geocoding service provider. It just provides an interface to connect to several services under a single package.

4.2.1.5 Tweepy

Tweepy is an open-source Python package that gives you a very convenient way to access the Twitter API with Python. It gives you an interface to access the API from your Python application.

4.3.1.6 Naive Bayes Classification

Naive Bayes classifier is the most popular and simple classifier model used commonly. This model finds the posterior probability based on word distribution in the document. Naïve Bayes classifier work with Bag of Words (BOW) feature extraction model, which do not consider the position of word inside the document. This model used Bayes Theorem for prediction of particular label from the given feature set. The dataset is split into train set and test set. Upon test set, NB_model is applied to find the location prediction.

There are three types of Naive Bayes Model, which are given below:

- **Gaussian**: The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.
- **Multinomial**: The Multinomial Naïve Bayes classifier is used when the data is multinomial distributed. It is primarily used for document classification problems, it means a particular document belongs to which category such as Sports, Politics, education etc. The classifier uses the frequency of words for the predictors.
- **Bernoulli**: The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Booleans variables. Such as if a particular word is present or not in a document. This model is also famous for document classification tasks.

4.2.1.7 Support vector machine

Support vector machine is one of most common used supervised learning techniques, which is commonly used for both classification and regression problems. The algorithm works in such a way that each data is plotted as point in n dimensional space with the feature values represents the values of each co-ordinate.

An SVM model is basically a representation of different classes in a hyperplane in multidimensional space. The hyperplane will be generated in an iterative manner by SVM so that the error can be minimized. The goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane.

The followings are important concepts in SVM –

- **Support Vectors** − Datapoints that are closest to the hyperplane is called support vectors. Separating line will be defined with the help of these data points.
- **Hyperplane** − As we can see in the above diagram, it is a decision plane or space which is divided between a set of objects having different classes.
- **Margin** − It may be defined as the gap between two lines on the closet data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. Large margin is considered as a good margin and small margin is considered as a bad margin.
- The main goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (MMH) and it can be done in the following two steps –
- First, SVM will generate hyperplanes iteratively that segregates the classes in best way.
- Then, it will choose the hyperplane that separates the classes correctly.

4.2.1.8 Decision Tree

Decision tree is the learning model, which utilizes classifications problem. Decision tree module works by splitting the dataset into minimum of two sets. Decision tree's internal nodes indicates a test on the features, branch depicts the result and leaf 's are decisions made after succeeding process on training.

Decision Tree works as follows-

- Decision tree starts with all training instances linked with the root node.
- It splits the dataset into train set and test set.
- It uses information to gain and chooses attributes to label the each node. Subsets made contain information with a similar feature attribute.
- Above process is repeated till in all subset until leafs get generated in tree.

Attribute Selection Measures –

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as Attribute selection measure or ASM**.** By this measurement, we can easily select the best attribute for the nodes of the tree.

There are two popular techniques for ASM, which are:

- Information Gain

  Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute. It calculates how much information a feature provides us about a class. According to the value of information gain, we split the node and build the decision tree. A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first.

- Gini Index

  Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm. An attribute with the low Gini index should be preferred as compared to the high Gini index. It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.

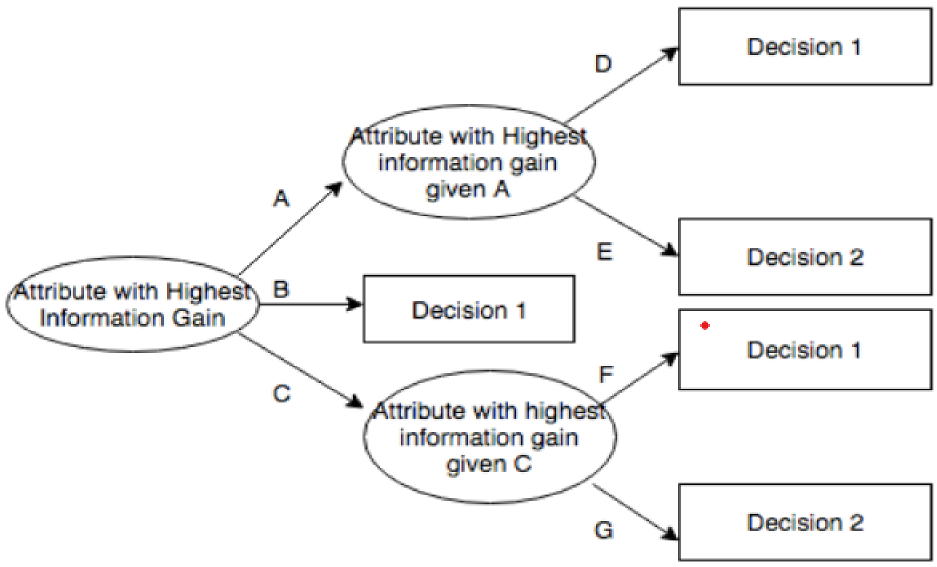

Attribute with Highest Information Gain
A
B
C
Attribute with Highest information gain given A
Decision 1
Attribute with highest information gain given C
D
E
F
G
Decision 1
Decision 2
Decision 1
Decision 2

Figure 4.3.1.8: Decision Tree Model

4.3 Non – Functional Requirements

4.3.1 Performance

The performance of the project depends upon the algorithms we use in this project It is a non-functional requirement that deals with the measure of time of the application under different load conditions.

4.3.2 Reliability

Reliability of a software system in our project can be seen from the values of Mean Square Error and Root mean Square which tells the quality of our prediction.

4.3.3 Availability

We have a data-set of millions of tweets for classification. With the help of this data – set we can classify our new tweet content

4.3.4 Security

We are securing out application by giving access to concerned professional person by providing authentication credentials.

4.3.5 Interoperability

We can make use of our project information by integrating it with several other software product for real time updates of tweet.

CHAPTER 5

SOFTWARE DESIGN

5.1 System Architecture

- The tweet data is collected and loaded into the Json File by making an API call to Twitter through tweepy and requesting the real time tweet.
- The Collected data then processed and redundant and indispensable data is identified and loaded into the CSV file.
- The data set loaded into the CSV file is used for classification for new Tweet content.
- Location is identified using Python's geopy package and accuracy of location is predicted using Machine Learning Algorithms.
- The accuracy results are shown which built using Django.

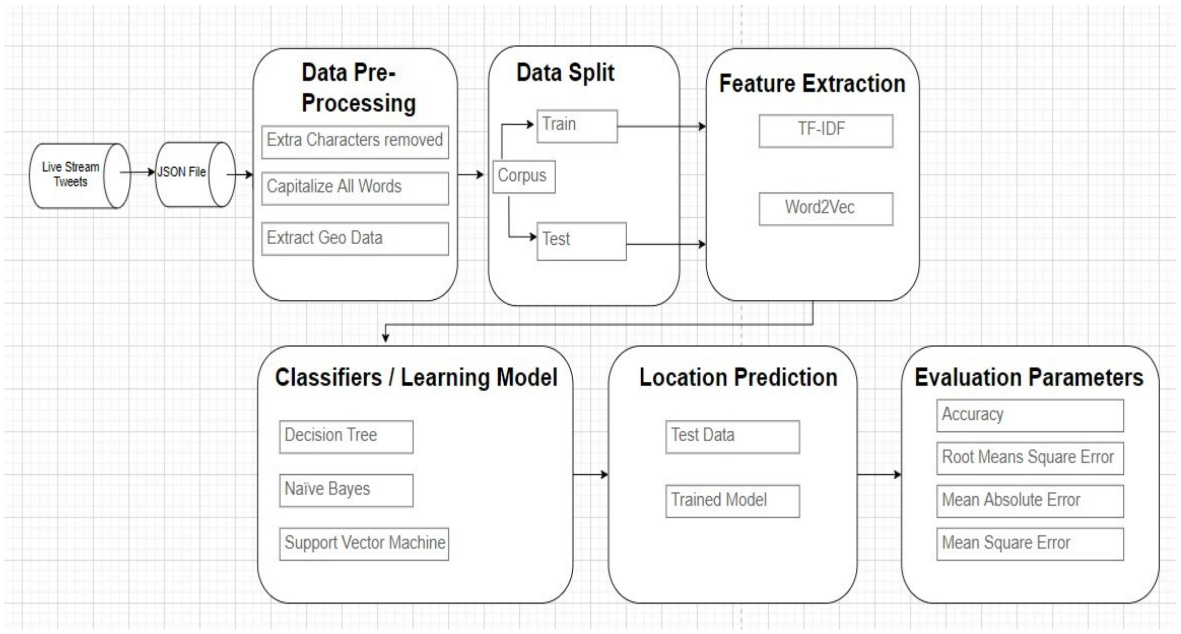


Figure 5.1 System Architecture

5.2 UML Diagrams

The Unified Modelling Language (UML) is the industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as other no software systems. UML simplifies the complex process of software design, making a "blueprint" for construction, and is now the standard notation for software architecture. UML provides both the structural views and behavioral views of the system.

5.2.1 Use Case Diagram

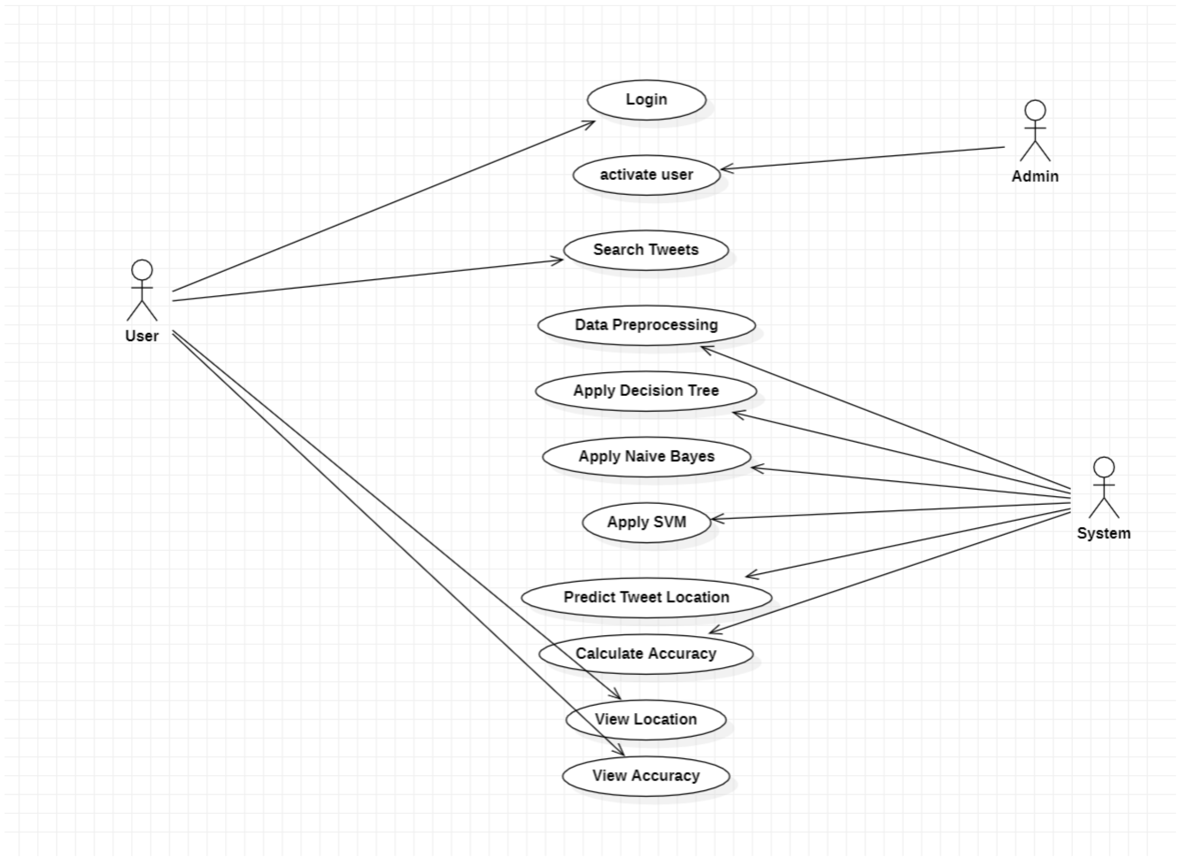


Figure 5.2.1: Use Case Diagram

They are three actors in our Use Case Diagram, User, Admin and System. The User can do operations like login and search tweets, The Admin activates the registered professional user and the System does operations such as Data Preprocessing, developing a Machine Learning model, predicting tweet location and calculating accuracy. Then the user can view the results.

5.2.2 Class Diagram

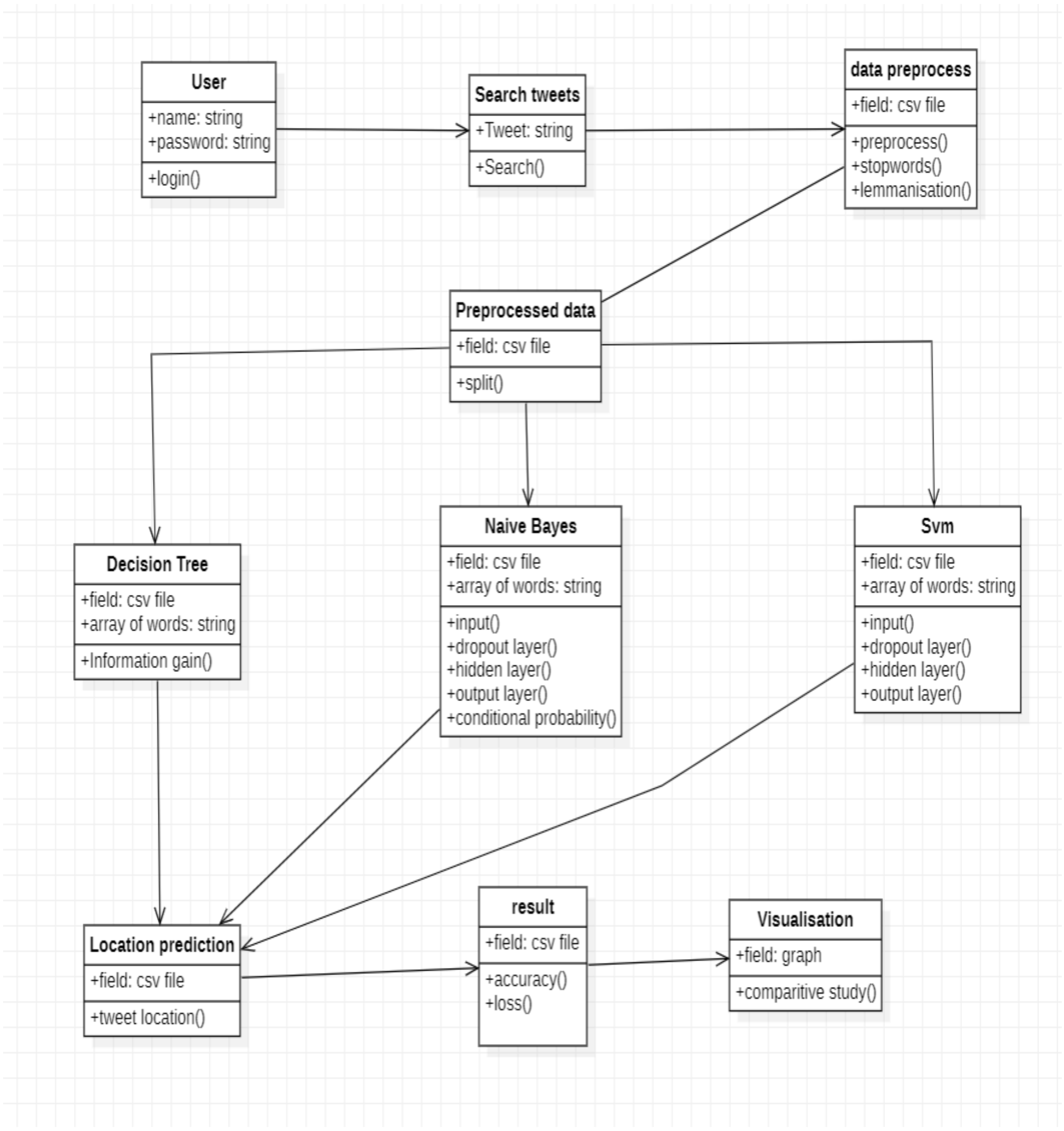


Figure 5.2.2: Class Diagram

They are 10 classes involved in our Class Diagram, User class and Search Tweet class are associated with the Data Pre-processing class with attributes which remove noisy data. The Data Pre-processing class is then associated with Pre-processed data wherein the data is split and stored in the CSV file. The Pre-processed data class is then associated with three different machine learning models, these models have the same operations such as a field containing a CSV file but different attributes. The Decision Tree class has attributes such as

Information Gain and the node having the highest Information Gain is split first. The Naïve Bayes class has attributes such as input layer, dropout layer, hidden layer, output layer and conditional probability layer which tells how the probability of an event is based on the occurrence of another event. The SVM class has similar attributes like Naïve Byae. In SVM models are effective in high dimensional spaces and are still effective in cases where the number of dimensions is greater than the number of samples.

All three machine learning models are associated with the Location Prediction class with the attribute predicting location. The Location Prediction class is associated with the result and Visualization class. The result class have operations such as the accuracy of models being calculated. In visualization class operation such as the comparative study of Machine Learning is done.

5.2.3 Sequence Diagrams

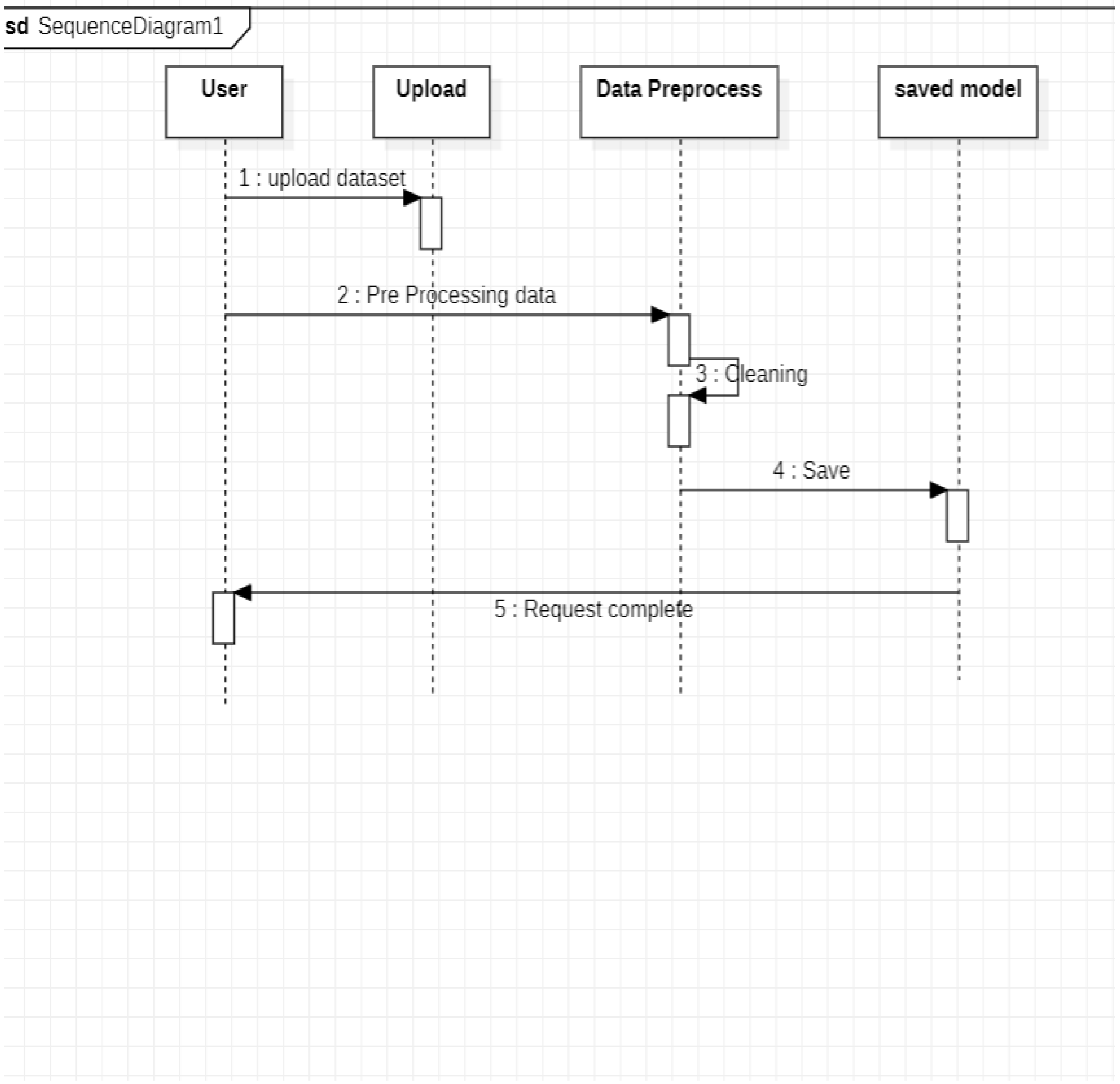


Figure 5.2.3: Sequence Diagram Preprocessing

The Sequence Diagram of Preprocessing have four lifelines User, Upload, Data Preprocess and Saved model. The User sends the message upload dataset and preprocessed data. The Data Preprocess lifeline removes all the noisy data and saves the model and the user request is completed.

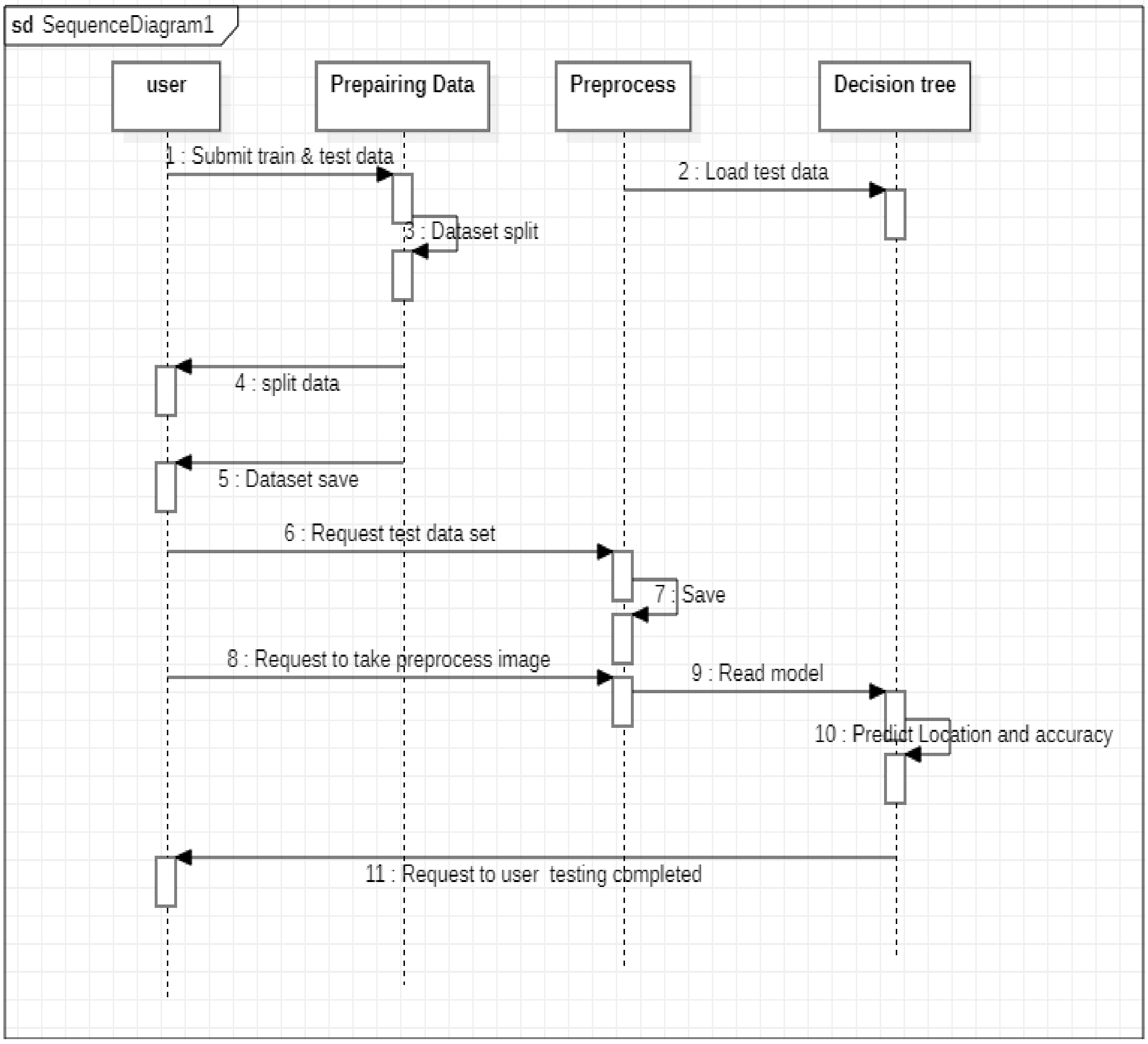

sd SequenceDiagram1
user
Prepairing Data
Preprocess
Decision tree
1 : Submit train & test data
2 : Load test data
3 : Dataset split
4 : split data
5 : Dataset save
6 : Request test data set
7 : Save
8 : Request to take preprocess image
9 : Read model
10 : Predict Location and accuracy
11 : Request to user testing completed

Figure 5.2.4: Sequence Diagram Testing Decision Tree

The Sequence Diagram for Testing for Decision Tree has four lifelines where in the user submits the train and the test data set and the data set is loaded into the decision tree model. The model is saved and test data set is tested and the location is predicted and accuracy is calculated and the request for User testing is completed.

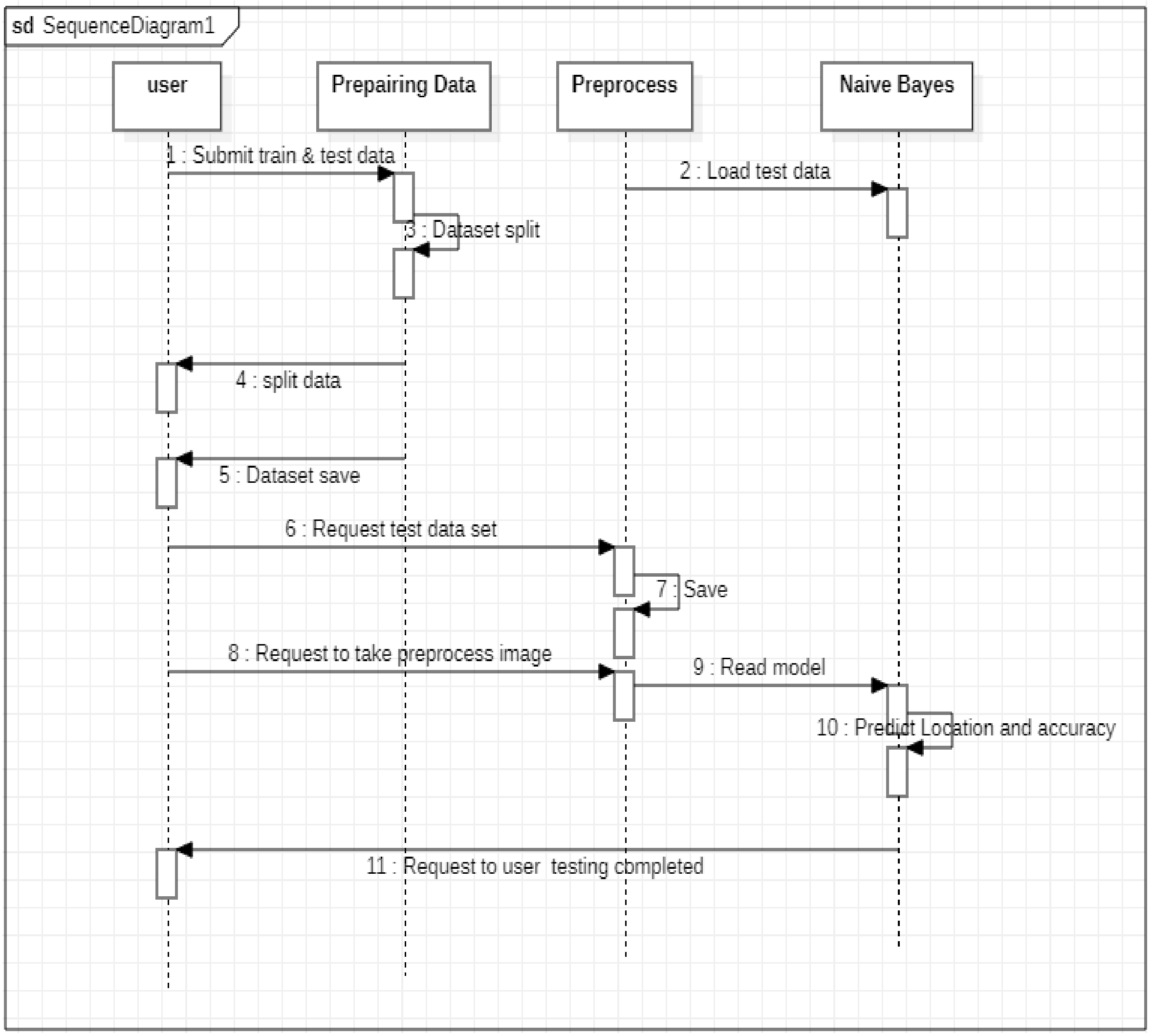

sd SequenceDiagram1
user
Prepairing Data
Preprocess
Naive Bayes
1 : Submit train & test data
2 : Load test data
3 : Dataset split
4 : split data
5 : Dataset save
6 : Request test data set
7 : Save
8 : Request to take preprocess image
9 : Read model
10 : Predict Location and accuracy
11 : Request to user testing completed

Figure 5.2.5: Sequence Diagram Testing Naïve Bayes

The Sequence Diagram for Testing for Naïve Bayes has four lifelines wherein the user submits the train and the test data set and the data set is loaded into the decision tree model. The model is saved and test data set is tested and the location is predicted and accuracy is calculated and the request for User testing is completed.

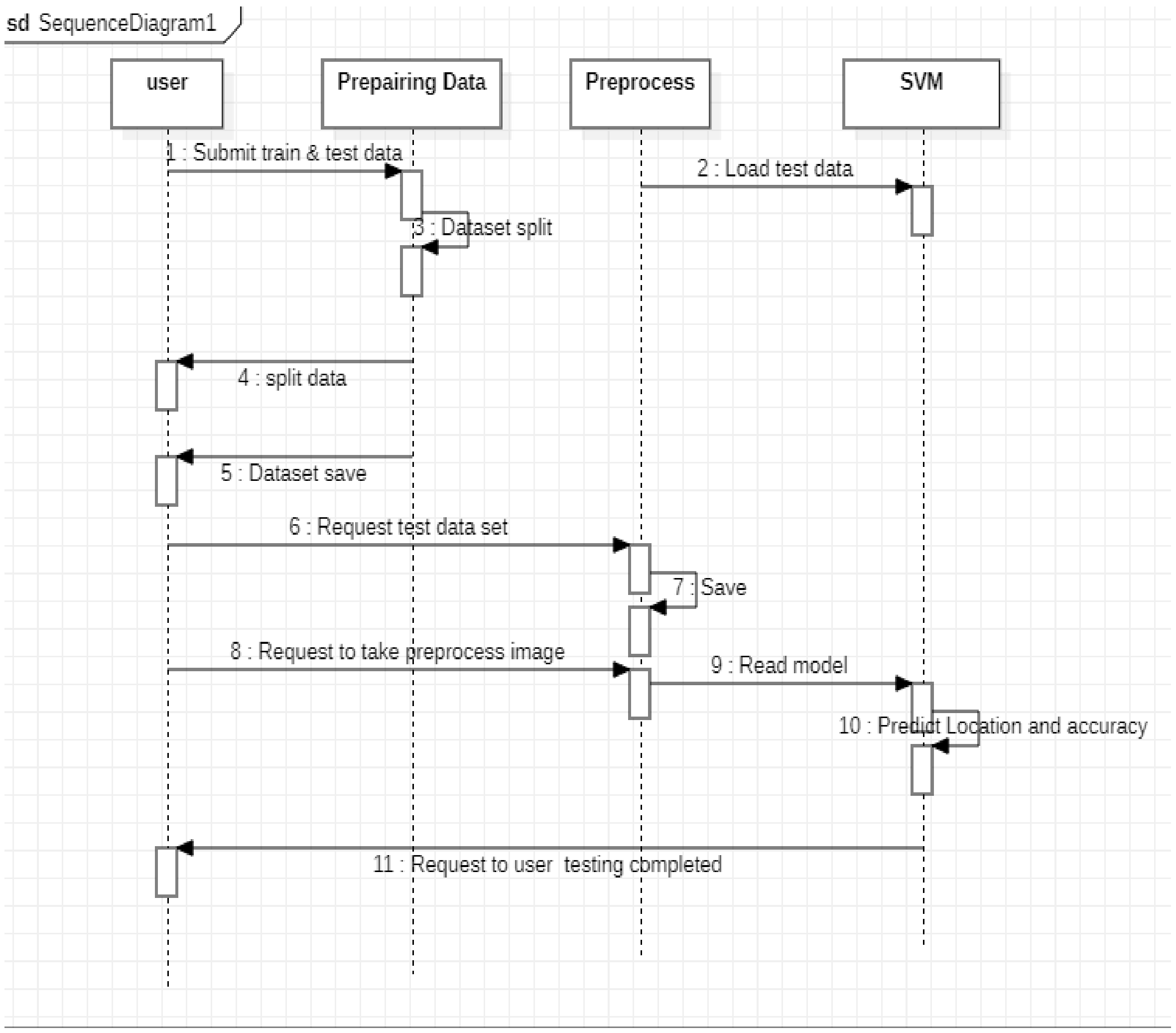

sd SequenceDiagram1
user
Prepairing Data
Preprocess
SVM
1 : Submit train & test data
2 : Load test data
3 : Dataset split
4 : split data
5 : Dataset save
6 : Request test data set
7 : Save
8 : Request to take preprocess image
9 : Read model
10 : Predict Location and accuracy
11 : Request to user testing completed

Figure 5.2.6: Sequence Diagram Testing SVM

The Sequence Diagram for Testing for Support Vector Machine has four lifelines wherein the user submits the train and the test data set and the data set is loaded into the decision tree model. The model is saved and test data set is tested and the location is predicted and accuracy is calculated and the request for User testing is completed.

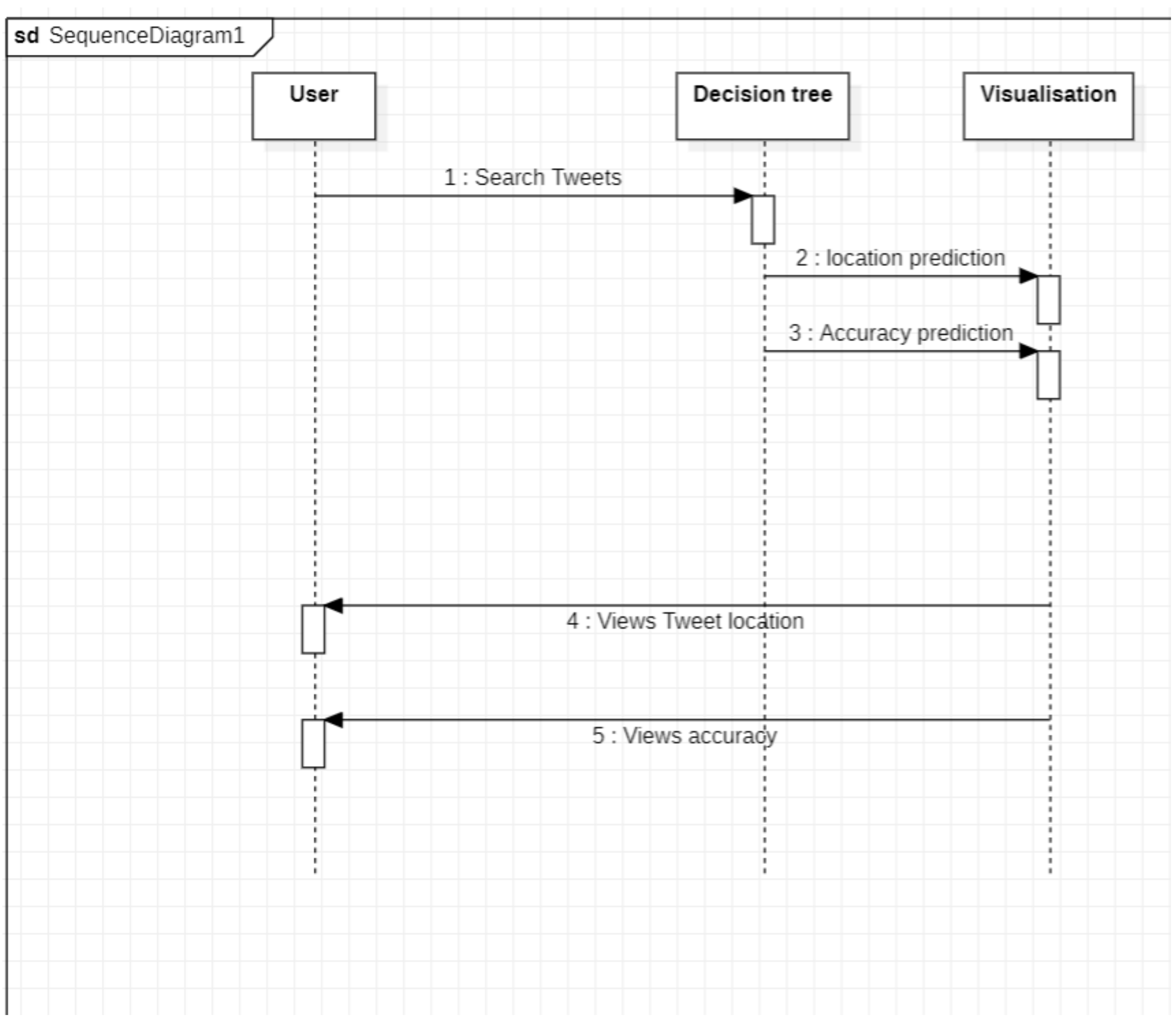


Figure 5.2.7: Sequence Diagram Result Decision Tree

The Sequence Diagram Result for Decision Tree has three lifelines where the User lifeline sends Search Tweets message to Decision Tree model which in turn sends a message to visualization class which return with Tweet location and View accuracy message to User Lifeline.

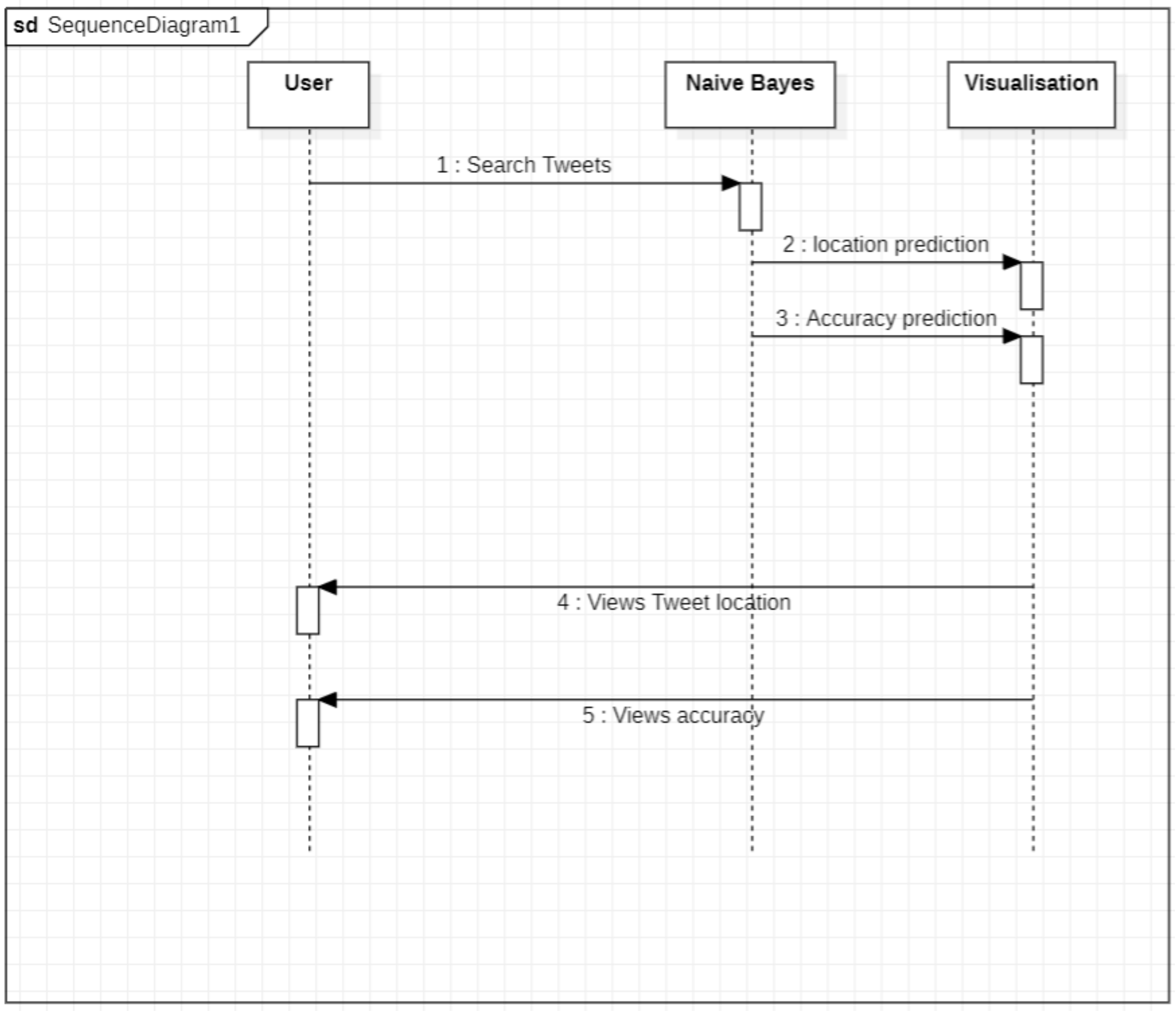


Figure 5.2.8: Sequence Diagram Result Naïve Bayes

The Sequence Diagram Result for Nive Bayes has three lifelines where the User lifeline sends Search Tweets message to Naïve Bayes model which in turn sends a message to visualization class which return with Tweet location and View accuracy message to User Lifeline.

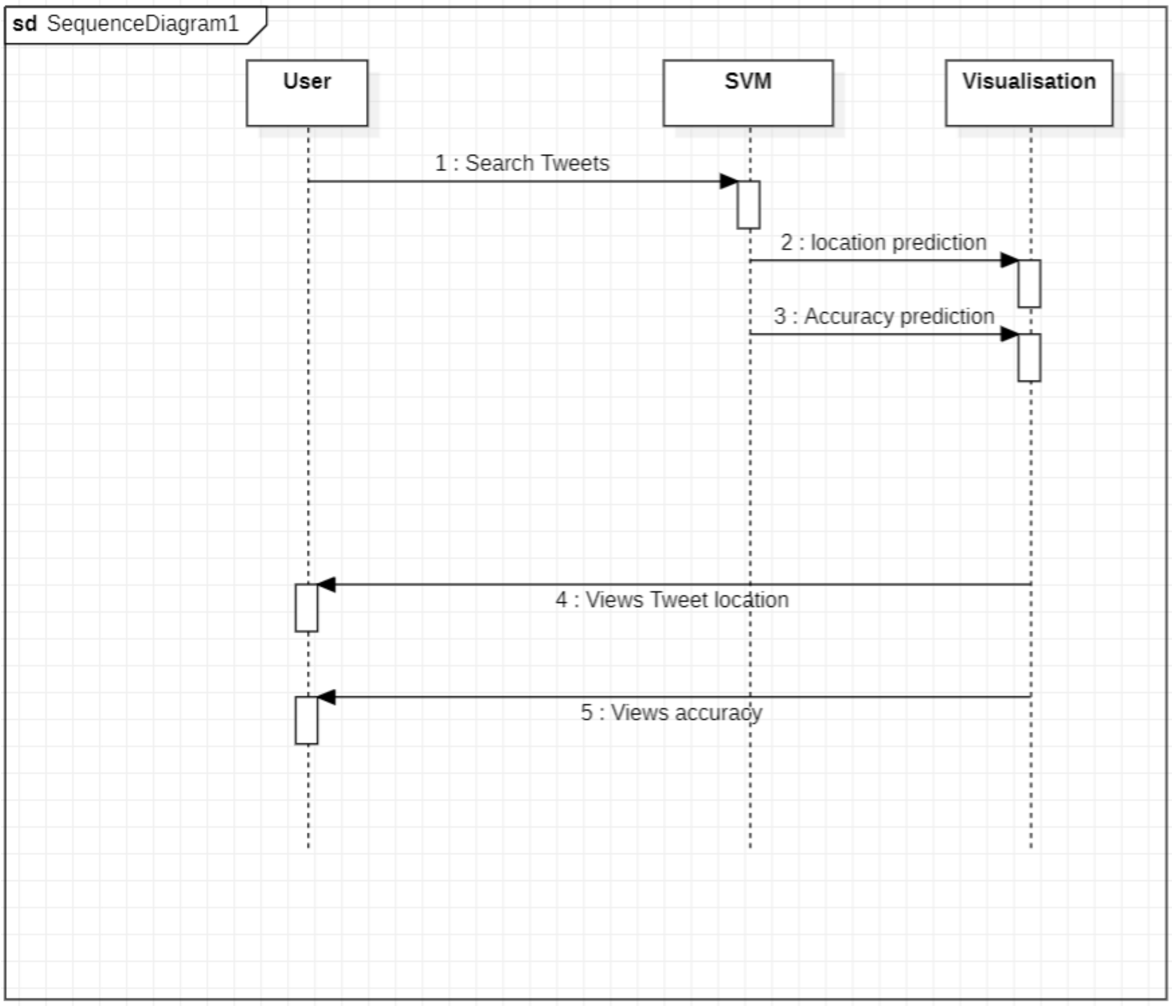


Figure 5.2.9: Sequence Diagram Result SVM

The Sequence Diagram Result for Support Vector Machine (SVM) has three lifelines where the User lifeline sends Search Tweets message to SVM model which in turn sends a message to visualization class which return with Tweet location and View accuracy message to User Lifeline.

CHAPTER 6

SOURCE CODE

views.py

```
from django.shortcuts import render,
HttpResponsefrom django.contrib
import messages
from .forms import UserRegistrationForm
from .models import UserRegistrationModel, UserSearchTweetsLocationModel,
UserAlgorithmResultsModel
from .TwitterClientAlgo import
TwitterClientfrom .Tweetinfo
import GetTweetLocatin from
django_pandas.io import
read_frame
from django.core.paginator import Paginator, EmptyPage,
PageNotAnIntegerfrom .algorithms.UserNaiveBayes import
UserNaiveBayesClass
from .algorithms.UserSVMAlgorithm import UserSVMClass
from .algorithms.UserDecisionTree import UserDecisionTreeClass

# Create your views here.
def
UserRegisterActions(r
equest):if
request.method ==
'POST':
form =
UserRegistrationForm(request.POST)if
form.is_valid():
```

```
messages.success(request, 'You have been successfully registered')
form = UserRegistrationForm()
return render(request, 'UserRegistrations.html', {'form':
form})else:
messages.success(request, 'Email or Mobile Already
Existed')print("Invalid form")
else:
form = UserRegistrationForm()
return render(request, 'UserRegistrations.html', {'form': form})


def UserLoginCheck(request):
if request.method == "POST":
loginid =
request.POST.get('loginname')
pswd = request.POST.get('pswd')
print("Login ID = ", loginid, ' Password = ', pswd)
 try:
check = UserRegistrationModel.objects.get(loginid=loginid, password=pswd) status =
check.status
print('Status is = ', status)if status == "activated":
request.session['id'] = check.id request.session['loggeduser'] = check.name
request.session['loginid'] = loginid request.session['email'] = check.email print("User id
At", check.id, status)
return render(request, 'users/UserHome.html', {})else:
messages.success(request, 'Your Account Not at activated') return render(request,
'UserLogin.html')
except Exception as e: print('Exception is ', str(e))pass
messages.success(request, 'Invalid Login id and password') return render(request,
'UserLogin.html', {})

def UserHome(request):
return render(request, 'users/UserHome.html', {})
```

```
def UserGetTweetsForm(request):
return render(request,'users/GetTweetForm.html',{})

def GetTweets(request):
if request.method=='POST':
hashtag = request.POST.get('tweettag')print("Working")
#api = TwitterClient()#limit = 200
            # calling function to get tweets
            # tweets = api.get_tweets(query=tagname, count=200) #tweets =
            api.get_tweets(query=hashtag, count=limit)#print(type(tweets))
            obj = GetTweetLocatin()
            dataList,dataframe = obj.getLocations(hashtag) dataframe =
            dataframe.to_html()
              for x in dataList: tweetId = x[0] username = x[1] created_at = x[2]
              user_screen_name = x[3]tweettext = x[4] tweet_location = x[5] user_loc =
              x[6]
#print("Date type ",type(created_at))if len(tweet_location)!=0:
#print("Tweet Location ", tweet_location)
lattitude,longitude,address = obj.getLatitudeLongitude(tweet_location)
#print(tweet_location,"==",lattitude,address)
flag = 0
if user_loc==None:flag = 0
else:
flag = 1
UserSearchTweetsLocationModel.objects.create(tweetid=tweetId, username=username,
userscreenname=user_screen_name,tweettext=tweettext,

createdat=created_at,address=address,latitude=lattitude,longitude=longitude,userloc=flag)
return render(request,"users/GetTweetsinfo.html",{'data':dataframe})
def UserViewDataset(request):
data_list = UserSearchTweetsLocationModel.objects.all()page = request.GET.get('page', 1)
```

```
paginator = Paginator(data_list, 20)try:
users = paginator.page(page)except PageNotAnInteger:
users = paginator.page(1)except EmptyPage:
users = paginator.page(paginator.num_pages)
return render(request, 'users/UserViewDataSet.html',{'users':users})

def UserNaiveBayes(request):
data_list = UserSearchTweetsLocationModel.objects.all()df = read_frame(data_list)
obj = UserNaiveBayesClass() accuracy,mae,mse,rmse,r_squared = obj.getNaiveResults(df)
algorithmname = "Naive Bayes"
username = request.session['loginid']

UserAlgorithmResultsModel.objects.create(username=username,algorithmname=algorithmn
ame,accu racy=accuracy,mae=mae,mse=mse,rmse=rmse,r_squared=r_squared)
return
render(request,'users/NaiveResults.html',{"accuracy":accuracy,"mae":mae,"mse":mse,"rmse"
:rmse,"r
_squared":r_squared})

def UserSVM(request):
data_list = UserSearchTweetsLocationModel.objects.all()df = read_frame(data_list)
obj = UserSVMClass()
accuracy, mae, mse, rmse, r_squared = obj.getSVM(df)algorithmname = "SVM"
username = request.session['loginid']
UserAlgorithmResultsModel.objects.create(username=username,
algorithmname=algorithmname,
accuracy=accuracy, mae=mae,mse=mse, rmse=rmse, r_squared=r_squared)
return render(request, 'users/SVMResults.html',{"accuracy": accuracy, "mae": mae, "mse":
mse,"rmse": rmse, "r_squared": r_squared})
def UserDecisionTree(request):
data_list = UserSearchTweetsLocationModel.objects.all()df = read_frame(data_list)
obj = UserDecisionTreeClass()
```

```
accuracy, mae, mse, rmse, r_squared = obj.getDecisionTree(df)algorithmname = "Decision
Tree"
username=request.session['loginid']
UserAlgorithmResultsModel.objects.create(username=username,
algorithmname=algorithmname,
accuracy=accuracy, mae=mae,
mse=mse,        rmse=rmse,        r_squared=r_squared)        return        render(request,
'users/DecisionTreeResults.html',
{"accuracy": accuracy, "mae": mae, "mse": mse, "rmse": rmse, "r_squared": r_squared})
```

Decision tree.py

```
from sklearn.model_selection import train_test_splitfrom sklearn import tree
from sklearn.metrics import accuracy_score,
mean_absolute_error,mean_squared_error,r2_scoreimport math
class UserDecisionTreeClass: def getDecisionTree(self,df):
df = df[['latitude', 'longitude', 'userloc']]print("df=", df.head())
X = df[['latitude', 'longitude']]y = df[['userloc']]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1 / 3, random_state=0)
model = tree.DecisionTreeClassifier()
model.fit(X,y)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred) mae = mean_absolute_error(y_pred, y_test)mse
= mean_squared_error(y_pred, y_test) rmse = math.sqrt(mse)
r_squared = r2_score(y_pred, y_test)
print("Decision Tree", "Accuracy = ", accuracy, "\t MAE=", mae, "\t MSE=", mse, "\t
RMSE=",rmse,
"\t r_squared = ", r_squared)


                    # return
                    round(accuracy,2),round(mae,2),round(mse,2),round(rmse,2),rou
                    nd(r_squared,2)return accuracy, mae, mse, rmse, r_squared
```

NaiveBayes.py

```
from sklearn.model_selection import train_test_splitfrom sklearn.naive_bayes import
GaussianNB
from sklearn.metrics import accuracy_score,
mean_absolute_error,mean_squared_error,r2_scoreimport math
import numpy as np
class UserNaiveBayesClass: def getNaiveResults(self,df):
df = df[['latitude','longitude', 'userloc']]print("df=",df.head())
X = df[['latitude', 'longitude']]y = df[['userloc']]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1 / 3, random_state=0)
model = GaussianNB()
model.fit(X_train, y_train) y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test,y_pred) mae = mean_absolute_error(y_pred,y_test)mse
= mean_squared_error(y_pred,y_test)
rmse = math.sqrt(mse)
r_squared =r2_score(y_pred,y_test)
print("Naivebayes","Accuracy = ",accuracy,"\t MAE=",mae,"\t MSE=",mse,"\t
RMSE=",rmse,"\t r_squared = ",r_squared)
#return round(accuracy,2),round(mae,2),round(mse,2),round(rmse,2),round(r_squared,2)
return accuracy,mae,mse,rmse,r_squared
```

SVMalgorithm.py

```
from sklearn.model_selection import train_test_splitfrom sklearn import svm
from sklearn.metrics import accuracy_score,
mean_absolute_error,mean_squared_error,r2_scoreimport math
import numpy as np class UserSVMClass:
def getSVM(self,df):
df = df[['latitude', 'longitude', 'userloc']]print("df=", df.head())
X = df[['latitude', 'longitude']]y = df[['userloc']]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1 / 3, random_state=0)
model = svm.SVC()
model.fit(X,y)
```

```
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
mae = mean_absolute_error(y_pred, y_test)mse = mean_squared_error(y_pred, y_test)
rmse = math.sqrt(mse)
r_squared = r2_score(y_pred, y_test)
print("SVM", "Accuracy = ", accuracy, "\t MAE=", mae, "\t MSE=", mse, "\t RMSE=",
rmse,"\t r_squared = ", r_squared)
# return round(accuracy,2),round(mae,2),round(mse,2),round(rmse,2),round(r_squared,2)
return accuracy, mae, mse, rmse, r_squared
```

Tweeinfo.py

```
import os
import tweepy as twimport pandas as pd
from geopy import geocoders
from geopy.geocoders import Nominatim
class GetTweetLocatin:
        def getLocations(self, tweet):
                search_words = "#" + tweet
                date_since = "2018-11-16"



consumer_key = "aKBt8eJagd4PumKz8LGmZw"
consumer_secret = "asFAO5b3Amo8Turjl2RxiUVXyviK6PYe1X6sVVBA"
access_token = "1914024835-
dgZBlP6Tn2zHbmOVOPHIjSiTabp9bVAzRSsKaDX"access_token_secret =
"zCgN7F4csr6f3eU5uhX6NZR12O5o6mHWgBALY9U4"

auth = tw.OAuthHandler(consumer_key,
consumer_secret)
auth.set_access_token(access_token,
access_token_secret) api = tw.API(auth,
wait_on_rate_limit=True)
# Collect tweets
tweets = tw.Cursor(api.search, q=search_words, lang="en", since=date_since).items(50)
print("=====>", tweets._dict_)

users_locs = [[tweet.id, tweet.user.name, tweet.created_at, tweet.user.screen_name,
tweet.text,tweet.user.location, tweet.coordinates] for tweet in tweets]
# print(users_locs)
dataframe = pd.DataFrame(data=users_locs,
columns=['Tweet ID', 'User Name', 'Created at', 'User Screen Name', "Tweets",'Tweet
Location', 'User Location'])
#gn = geocoders.GeoNames()
#print(gn.geocode("Cleveland, OH
44106"))return users_locs, dataframe

# return users_locs

def getLatitudeLongitude(self,cityname):
geolocator = Nominatim(user_agent="datapointprojects13@gmail.com")location =
geolocator.geocode(cityname)
try:
return location.latitude, location.longitude,location.addressexcept Exception as ex:
return 0,0,None
```

Adminside Views.py

```
from django.shortcuts import render from django.contrib import messages
from users.models import UserRegistrationModel,UserAlgorithmResultsModel

# Create your views here.
```

```
def AdminLoginCheck(request):if request.method == 'POST':
usrid = request.POST.get('loginid')pswd = request.POST.get('pswd') print("User ID is = ",
usrid)
if usrid == 'admin' and pswd == 'admin':
    return render(request, 'admins/AdminHome.html')elif usrid == 'Admin' and pswd ==
'Admin':
    return render(request, 'admins/AdminHome.html')else:

messages.success(request, 'Please Check Your Login Details')return render(request,
'AdminLogin.html', {})
def AdminHome(request):
return render(request, 'admins/AdminHome.html')



def ViewRegisteredUsers(request):
data = UserRegistrationModel.objects.all()
return render(request, 'admins/RegisteredUsers.html', {'data': data})



def
AdminActivaUsers(r
equest):if
request.method ==
'GET':
id =
request.GET.get('ui
d')status =
'activated'
print("PID = ", id,
status)
UserRegistrationModel.objects.filter(id=id).update(status=stat
us) data = UserRegistrationModel.objects.all()
return render(request, 'admins/RegisteredUsers.html', {'data': data})
```

```
def AdminNaiveBayes(request):
data = UserAlgorithmResultsModel.objects.filter(algorithmname="Naive Bayes")return
render(request,'admins/AdminNaiveBayes.html',{'data':data})

def AdminSVM(request):
data = UserAlgorithmResultsModel.objects.filter(algorithmname="SVM")return
render(request,'admins/AdminSVM.html',{'data':data})

def AdminDecisionTree(request):
data = UserAlgorithmResultsModel.objects.filter(algorithmname="Decision Tree")return
render(request, 'admins/AdminDecisionTree.html', {'data': data})
```

base.html

```
<!doctype html>
{%load static%}
<html class="no-js" lang="zxx">
<head>
  <meta charset="utf-8">
  <meta http-equiv="x-ua-compatible" content="ie=edge">
  <title>Tweets of locations </title>
  <meta name="description" content="">
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <link rel="shortcut icon" type="image/x-icon" href="{%static 'img/favicon.ico'%}">

  <!-- CSS here -->
  <link rel="stylesheet" href="{%static 'css/bootstrap.min.css'%}">
  <link rel="stylesheet" href="{%static 'css/owl.carousel.min.css'%}">
  <link rel="stylesheet" href="{%static 'css/slicknav.css'%}">
  <link rel="stylesheet" href="{%static 'css/flaticon.css'%}">
  <link rel="stylesheet" href="{%static 'css/animate.min.css'%}">
  <link rel="stylesheet" href="{%static 'css/magnific-popup.css'%}">
  <link rel="stylesheet" href="{%static 'css/fontawesome-all.min.css'%}">
  <link rel="stylesheet" href="{%static 'css/themify-icons.css'%}">
  <link rel="stylesheet" href="{%static 'css/slick.css'%}">
  <link rel="stylesheet" href="{%static 'css/nice-select.css'%}">
  <link rel="stylesheet" href="{%static 'css/style.css'%}">
</head>

<body class="body-bg">
<!--? Preloader Start -->
<div id="preloader-active">
  <div class="preloader d-flex align-items-center justify-content-center">
    <div class="preloader-inner position-relative">
      <div class="preloader-circle"></div>
      <div class="preloader-img pere-text">
        <img src="{%static 'img/logo/myloader.png'%}" alt="">
      </div>
    </div>
  </div>
</div>
<header>
  <!-- Header Start -->
  <div class="header-area">
    <div class="main-header ">
      <div class="header-top d-none d-lg-block">


      </div>
      <div class="header-bottom  header-sticky">
        <div class="container">
          <div class="row align-items-center">
            <!-- Logo -->
            <div class="col-xl-2 col-lg-2">
              <div class="logo">
                <a href="about.html" class="btn post-btn">Location Prediction on
```

```
Twitter</a>
                              </div>
                         </div>
                         <div class="col-xl-10 col-lg-10">
                              <div class="menu-wrapper  d-flex align-items-center justify-content-
end">
                                   <!-- Main-menu -->
                                   <div class="main-menu d-none d-lg-block">
                                        <nav>
                                             <ul id="navigation">
                                                  <li><a href="{%url 'index'%}">Home</a></li>
                                                  <li><a href="{%url 'UserLogin'%}">User</a></li>
                                                  <li><a href="{%url 'AdminLogin'%}">Admin</a></li>
                                                  <li><a href="{%url 'UserRegister'%}">Registrations</a></li>
                                             </ul>
                                        </nav>
                                   </div>
                              </div>
                         </div>
                         <!-- Mobile Menu -->
                         <div class="col-12">
                              <div class="mobile_menu d-block d-lg-none"></div>
                         </div>
                    </div>
               </div>
          </div>
     </div>
    </div>
    <!-- Header End -->
</header>
<main>
    {%block contents%}
    {%endblock%}
</main>
<footer>
    <!--? Footer Start-->
    <div class="footer-area section-bg" data-background="{%static
'img/gallery/footer_bg.jpg'%}">
        <div class="container">
            <div class="footer-top footer-padding">

            </div>
            <div class="footer-bottom">
                <div class="row d-flex justify-content-between align-items-center">
                    <div class="col-xl-9 col-lg-8">
                        <div class="footer-copy-right">
                            <p><!-- Link back to Colorlib can't be removed. Template is licensed
under CC BY 3.0. -->
  Copyright &copy;<script>document.write(new Date().getFullYear());</script> All rights
reserved | This template is made with <i class="fa fa-heart" aria-hidden="true"></i> by
Alex Corporation
  <!-- Link back to Colorlib can't be removed. Template is licensed under CC BY 3.0. --
></p>
                        </div>
                    </div>


                </div>
            </div>
        </div>
    </div>
    <!-- Footer End-->
</footer>

    <!-- JS here -->
    <script src="{%static 'js/vendor/modernizr-3.5.0.min.js'%}"></script>
    <!-- Jquery, Popper, Bootstrap -->
    <script src="{%static 'js/vendor/jquery-1.12.4.min.js'%}"></script>
    <script src="{%static 'js/popper.min.js'%}"></script>
    <script src="{%static 'js/bootstrap.min.js'%}"></script>
    <!-- Jquery Mobile Menu -->
    <script src="{%static 'js/jquery.slicknav.min.js'%}"></script>

    <!-- Jquery Slick , Owl-Carousel Plugins -->
    <script src="{%static 'js/owl.carousel.min.js'%}"></script>
    <script src="{%static 'js/slick.min.js'%}"></script>
    <!-- One Page, Animated-HeadLin -->
    <script src="{%static 'js/wow.min.js'%}"></script>
    <script src="{%static 'js/animated.headline.js'%}"></script>
    <script src="{%static 'js/jquery.magnific-popup.js'%}"></script>

    <!-- Nice-select, sticky -->
    <script src="{%static 'js/jquery.nice-select.min.js'%}"></script>
    <script src="{%static 'js/jquery.sticky.js'%}"></script>

    <!-- counter , waypoint -->
    <script
src="http://cdnjs.cloudflare.com/ajax/libs/waypoints/2.0.3/waypoints.min.js"></script>
    <script src="{%static 'js/jquery.counterup.min.js'%}"></script>

    <!-- contact js -->
    <script src="{%static 'js/contact.js'%}"></script>
    <script src="{%static 'js/jquery.form.js'%}"></script>
    <script src="{%static 'js/jquery.validate.min.js'%}"></script>
    <script src="{%static 'js/mail-script.js'%}"></script>
    <script src="{%static 'js/jquery.ajaxchimp.min.js'%}"></script>
    <!-- Jquery Plugins, main Jquery -->
    <script src="{%static 'js/plugins.js'%}"></script>
    <script src="{%static 'js/main.js'%}"></script>

    </body>
</html>
```

CHAPTER 7

TESTING

Testing is the process of executing a program with the aim of finding errors. To make our software perform well it should be error-free. If testing is successful, it will remove all the unforeseen errors from the product.

Software testing is the process of executing a program with the intention of finding errors in the code. It is a process of evolution of system or its parts by manual or automatic means to verify that it is satisfying specified or requirements or not. Generally, no system is perfect due to communication problems between user and developer, time constraints, or conceptual mistakes by developer.

7.1 Test Cases

7.1.1 Login

Object	**Functionality**	**Input**	**Output**	**Result**
User Login	Verify user details	User id Password	Login successful, If authorized	Pass
User Login	Verify user details	User id Password	Login unsuccessful, If unauthorized	Fail

7.1.2 Activate Registered Users

Object	**Functionality**	**Input**	**Output**	**Result**
Admin	Activate requested user accounts and provide authentication keys	Successfully registered users	Activated if legitimate user	Pass
Admin	Activate requested user accounts and provide authentication keys	Successfully registered users	Reject activation request if not legitimate user	Fail

7.1.3 Upload Data Set

Object	**Functionality**	**Input**	**Output**	**Result**
Fetching the data set from twitter	Input Dataset	CSV File	Dataset uploaded	Pass
Fetching the dataset from twitter	Input Dataset	CSV File	Unable to upload Dataset	Fail

7.1.4 Preprocess

Object	Functionality	Input	Output	Result
Dataset	Preprocess the dataset	Labelled Data	Noisy Data removed	Pass
Dataset	Preprocess the dataset	Labelled Data	Noisy Data not removed	Fail

7.1.5 Run Decision Tree Algorithm

Object	Functionality	Input	Output	Result
Dataset	Run Decision Tree Algorithm	Preprocess ed Data	Classified Data	Pass
Dataset	Run Decision Tree Algorithm	Preprocess ed Data	Unclassified Data	Fail

7.1.6 Run Naïve Bayes Algorithm

Object	**Functionality**	**Input**	**Output**	**Result**
Dataset	Run Naïve Bayes Algorithm	Preprocess ed Data	Classified Data	Pass
Dataset	Run Naïve Bayes Algorithm	Preprocess ed Data	Unclassified Data	Fail

7.1.7 Run Support Vector Machine

Object	**Functionality**	**Input**	**Output**	**Result**
Dataset	Run Support Vector Machine Algorithm	Preprocess ed Data	Classified Data	Pass
Dataset	Run Support Vector Machine Algorithm	Preprocess ed Data	Unclassified Data	Fail

7.1.8 Decision Tree Algorithm Accuracy

Object	**Functionality**	**Input**	**Output**	**Result**
Dataset	Run Decision Tree Algorithm	Preprocess ed Data	Classified Data	Pass
Dataset	Run Decision Tree Algorithm	Preprocess ed Data	Unclassified Data	Fail

7.1.9 Naïve Bayes Algorithm Accuracy

Object	**Functionality**	**Input**	**Output**	**Result**
Dataset	Run Naïve Bayes Algorithm	Preprocessed Data	Classified Data	Pass
Dataset	Run Naïve Bayes Algorithm	Preprocessed Data	Unclassified Data	Fail

7.1.10 Support Vector Machine Algorithm Accuracy

Object	**Functionality**	**Input**	**Output**	**Result**
Dataset	Run Support Vector Machine Algorithm	Preprocessed Data	Classified Data	Pass
Dataset	Run Support Vector Machine Algorithm	Preprocessed Data	Unclassified Data	Fail

CHAPTER 8

OUTPUT SCREENS

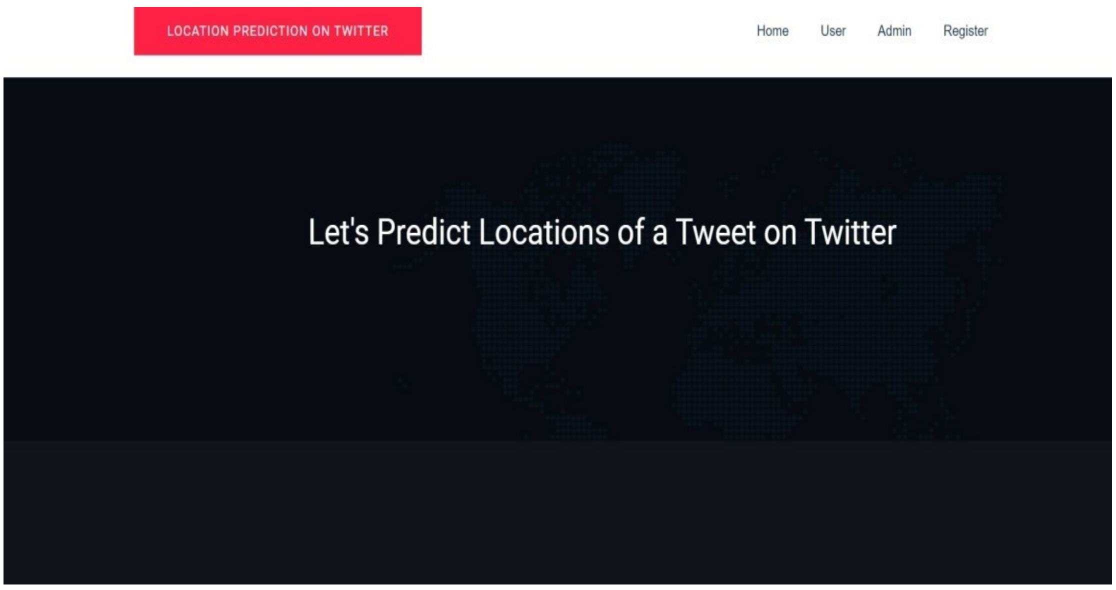

LOCATION PREDICTION ON TWITTER
Home
User
Admin
Register
Let's Predict Locations of a Tweet on Twitter

Figure 8.1: Main Screen

This Web application is opened and is ready to work

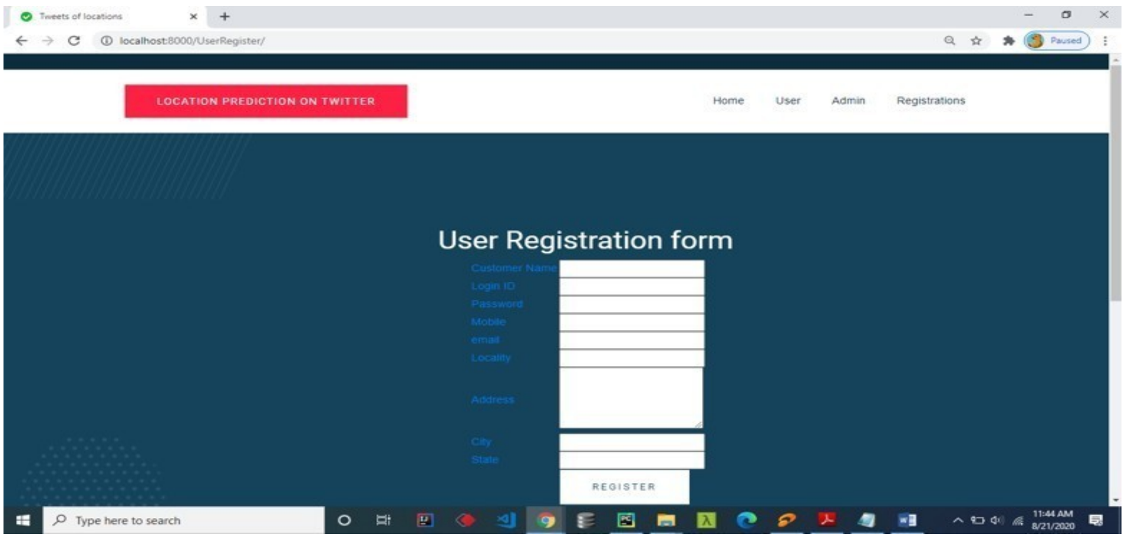


Figure 8.2: User Registration

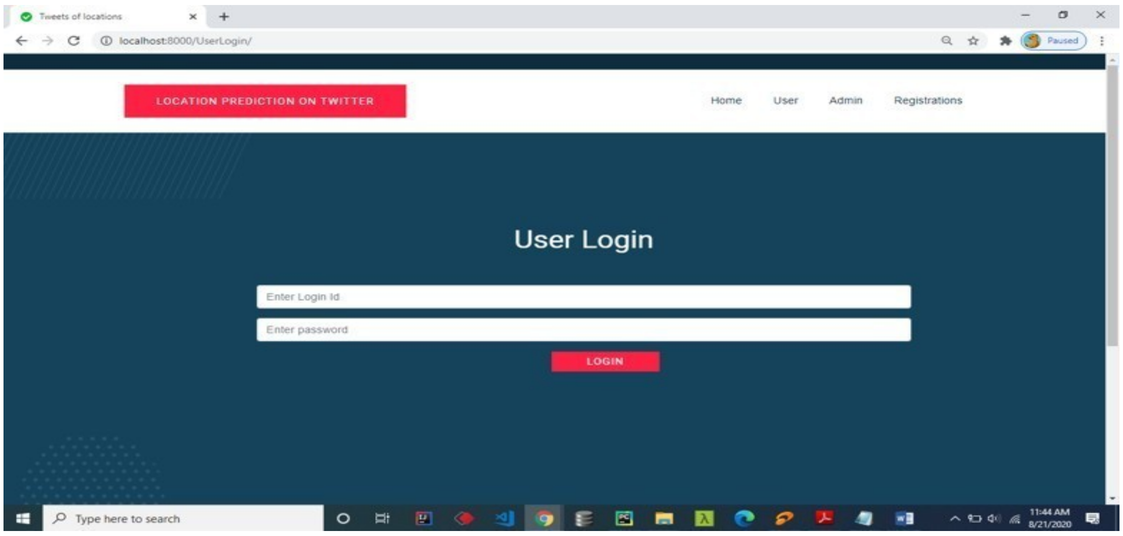
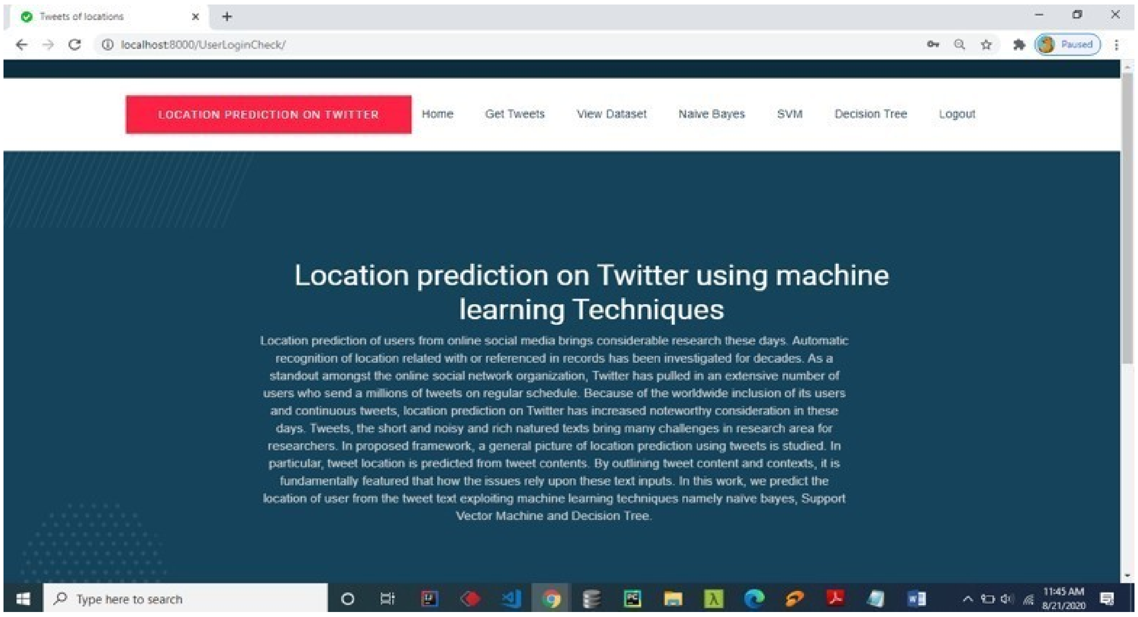


Figure 8.3: User Login



Location prediction on Twitter using machine learning Techniques

Location prediction of users from online social media brings considerable research these days. Automatic recognition of location related with or referenced in records has been investigated for decades. As a standout amongst the online social network organization, Twitter has pulled in an extensive number of users who send a millions of tweets on regular schedule. Because of the worldwide inclusion of its users and continuous tweets, location prediction on Twitter has increased noteworthy consideration in these days. Tweets, the short and noisy and rich natured texts bring many challenges in research area for researchers. In proposed framework, a general picture of location prediction using tweets is studied. In particular, tweet location is predicted from tweet contents. By outlining tweet content and contexts, it is fundamentally featured that how the issues rely upon these text inputs. In this work, we predict the location of user from the tweet text exploiting machine learning techniques namely naive bayes, Support Vector Machine and Decision Tree.

Figure 8.4: User Home

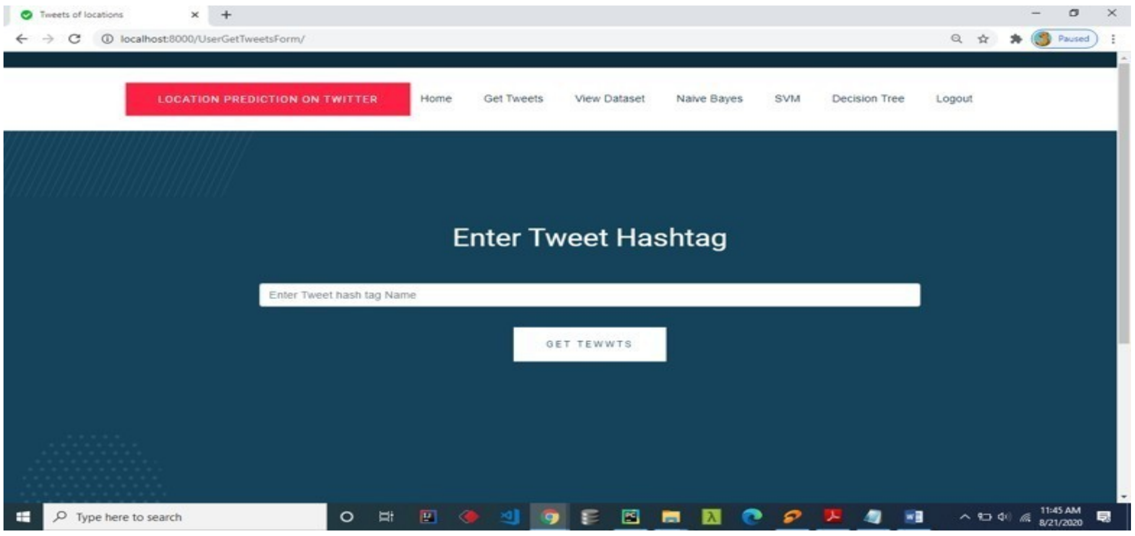
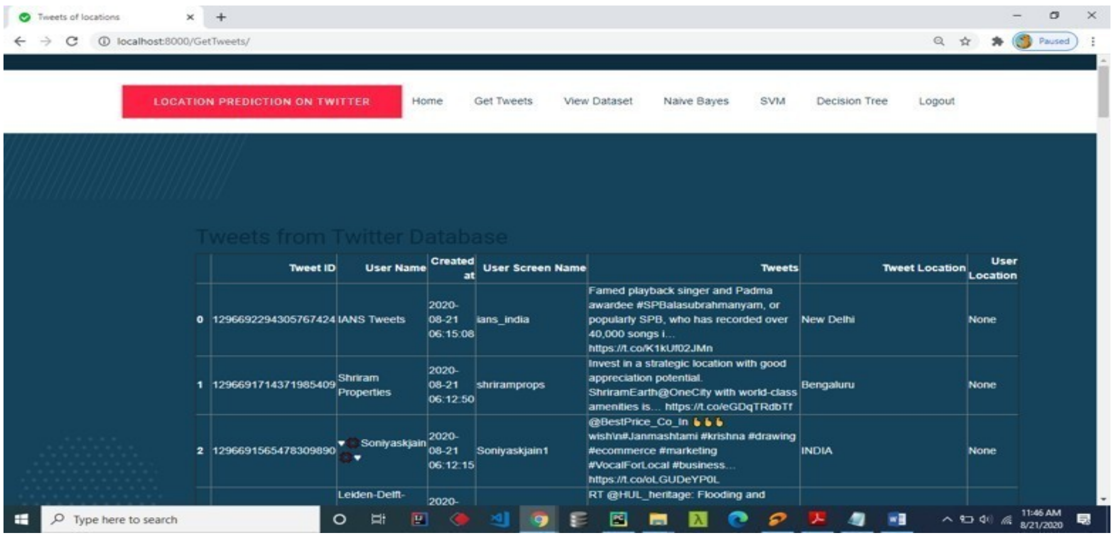


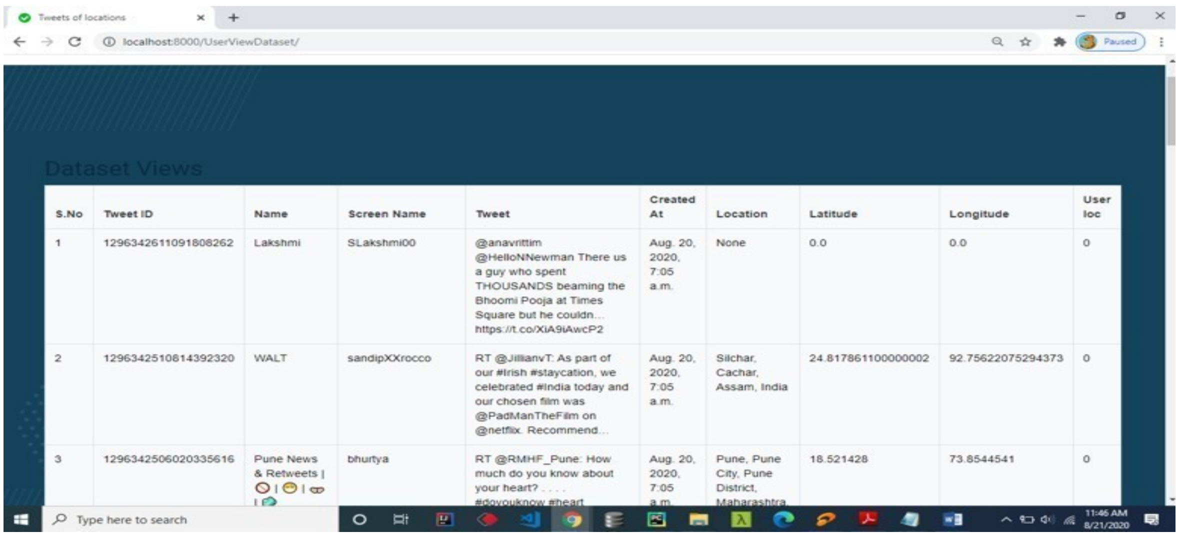
Figure 8.5: Enter Tweet



Tweets from Twitter Database

	Tweet ID	User Name	Created at	User Screen Name	Tweets	Tweet Location	User Location
0	1296692294305767424	IANS Tweets	2020-08-21 06:15:08	ians_india	Famed playback singer and Padma awardee #SPBalasubrahmanyam, or popularly SPB, who has recorded over 40,000 songs i… https://t.co/K1kUf02JMn	New Delhi	None
1	1296691714371985409	Shriram Properties	2020-08-21 06:12:50	shriramprops	Invest in a strategic location with good appreciation potential. ShriramEarth@OneCity with world-class amenities is… https://t.co/eGDqTRdbTf	Bengaluru	None
2	1296691565478309890	Soniyaskjain	2020-08-21 06:12:15	Soniyaskjain1	@BestPrice_Co_In 👌👌👌 wish\n#Janmashtami #krishna #drawing #ecommerce #marketing #VocalForLocal #business… https://t.co/oLGUDeYP0L	INDIA	None
		Leiden-Delft-	2020-		RT @HUL_heritage: Flooding and		

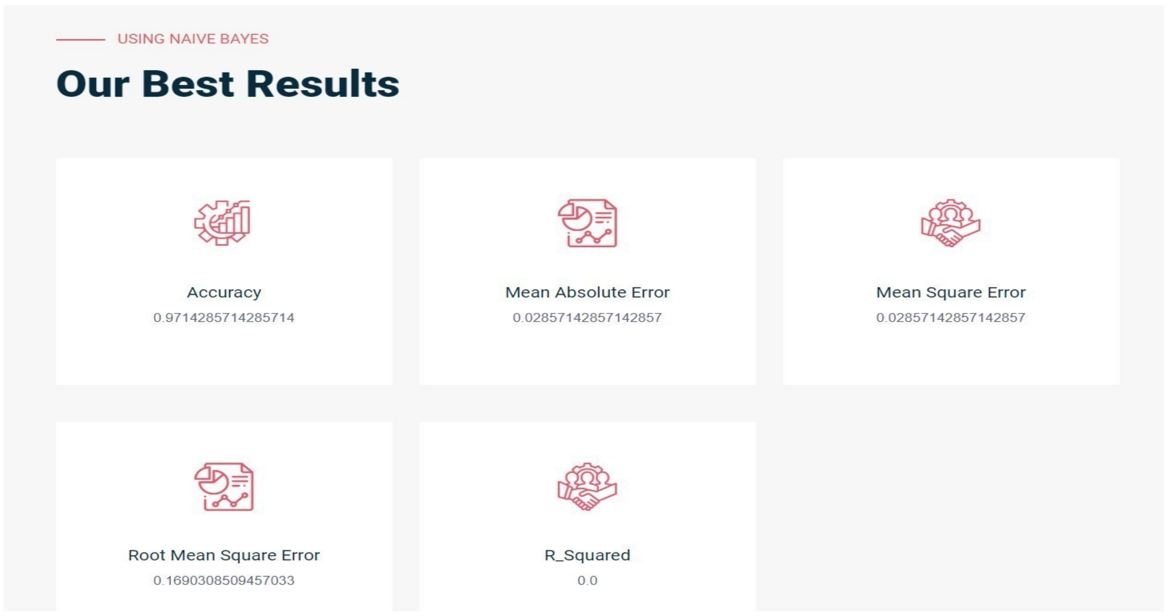
Figure 8.6: Tweet result



Dataset Views

S.No	Tweet ID	Name	Screen Name	Tweet	Created At	Location	Latitude	Longitude	User loc
1	1296342611091808262	Lakshmi	SLakshmi00	@anavrittim @HelloNNewman There us a guy who spent THOUSANDS beaming the Bhoomi Pooja at Times Square but he couldn… https://t.co/XiA9iAwcP2	Aug. 20, 2020, 7:05 a.m.	None	0.0	0.0	0
2	1296342510814392320	WALT	sandipXXrocco	RT @JillianvT: As part of our #Irish #staycation, we celebrated #India today and our chosen film was @PadManTheFilm on @netflix. Recommend…	Aug. 20, 2020, 7:05 a.m.	Silchar, Cachar, Assam, India	24.817861100000002	92.75622075294373	0
3	1296342506020335616	Pune News & Retweets | 🚫 | 😁 | 👓 | 🗺	bhurtya	RT @RMHF_Pune: How much do you know about your heart? . . . . #doyouknow #heart	Aug. 20, 2020, 7:05 a.m.	Pune, Pune City, Pune District, Maharashtra	18.521428	73.8544541	0

Figure 8.7: Tweet Training Data Set


USING NAIVE BAYES

Our Best Results

Accuracy
0.9714285714285714

Mean Absolute Error
0.02857142857142857

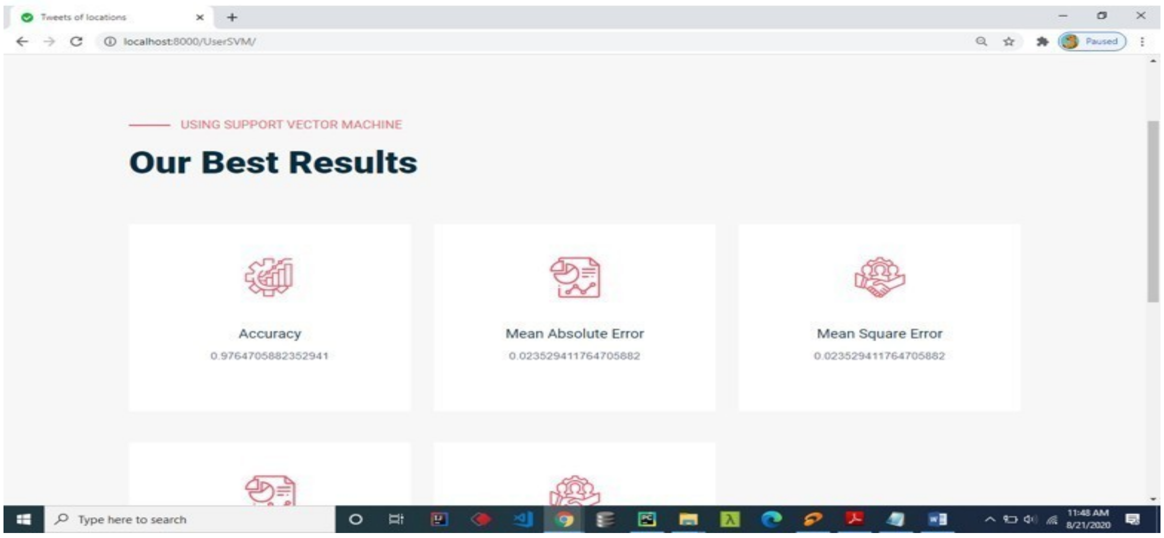
Mean Square Error
0.02857142857142857

Root Mean Square Error
0.1690308509457033

R_Squared
0.0

Figure 8.8: NB result user end

Support Vector Machine Results:


Tweets of locations

localhost:8000/UserSVM/

USING SUPPORT VECTOR MACHINE

Our Best Results

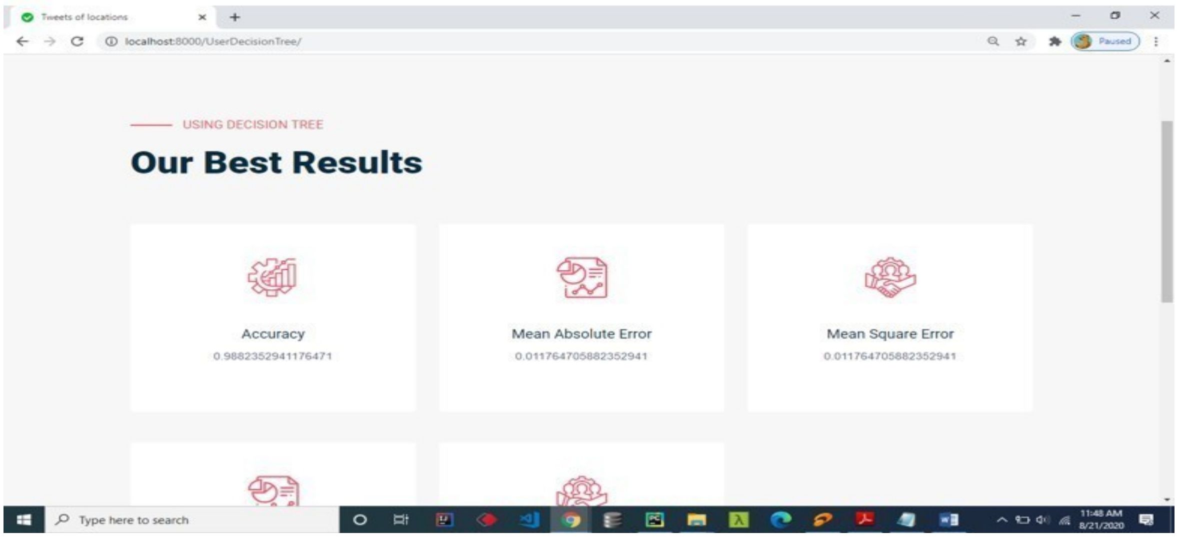
Accuracy
0.9764705882352941

Mean Absolute Error
0.023529411764705882

Mean Square Error
0.023529411764705882

Figure 8.9: SVM Result user end

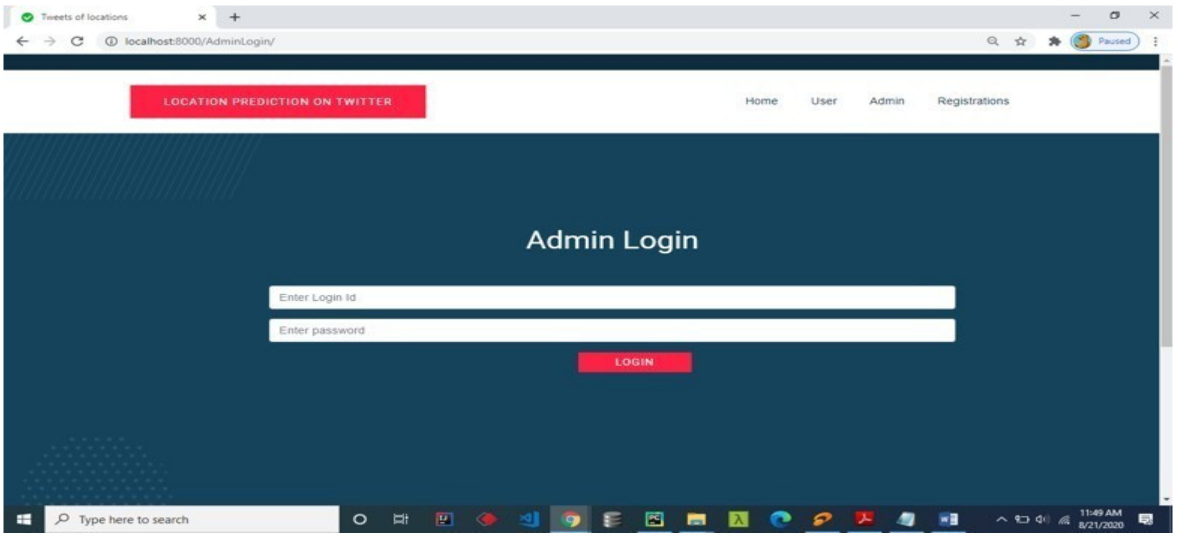
Decision Tree Results:


USING DECISION TREE

Our Best Results

Accuracy	Mean Absolute Error	Mean Square Error
0.9882352941176471	0.011764705882352941	0.011764705882352941

Figure 8.10: Decision Tree Result user end


LOCATION PREDICTION ON TWITTER

Home User Admin Registrations

Admin Login

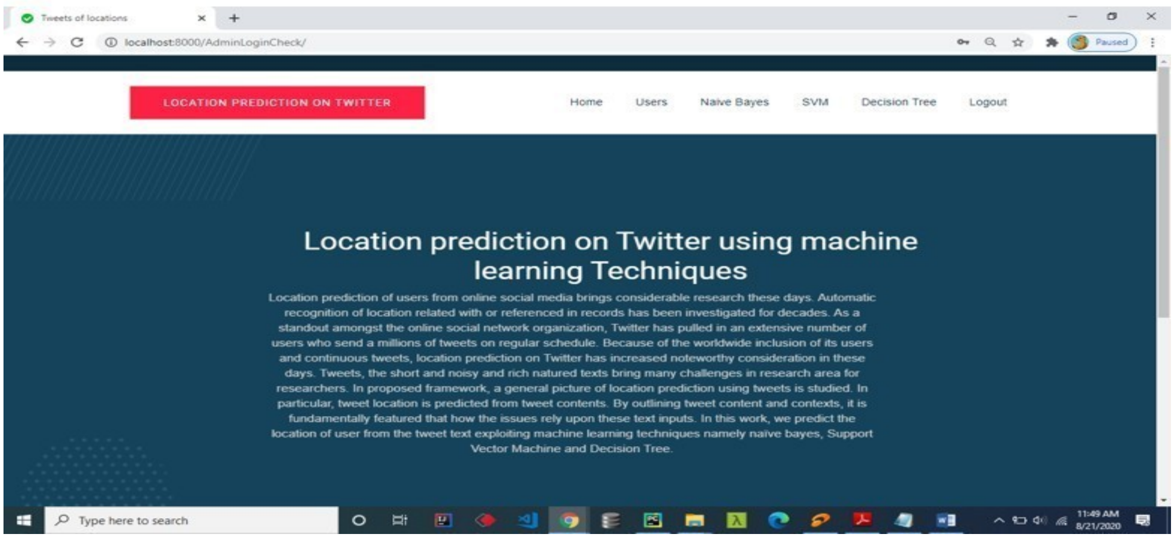
Enter Login Id

Enter password

LOGIN

Figure 8.11: Admin Login page

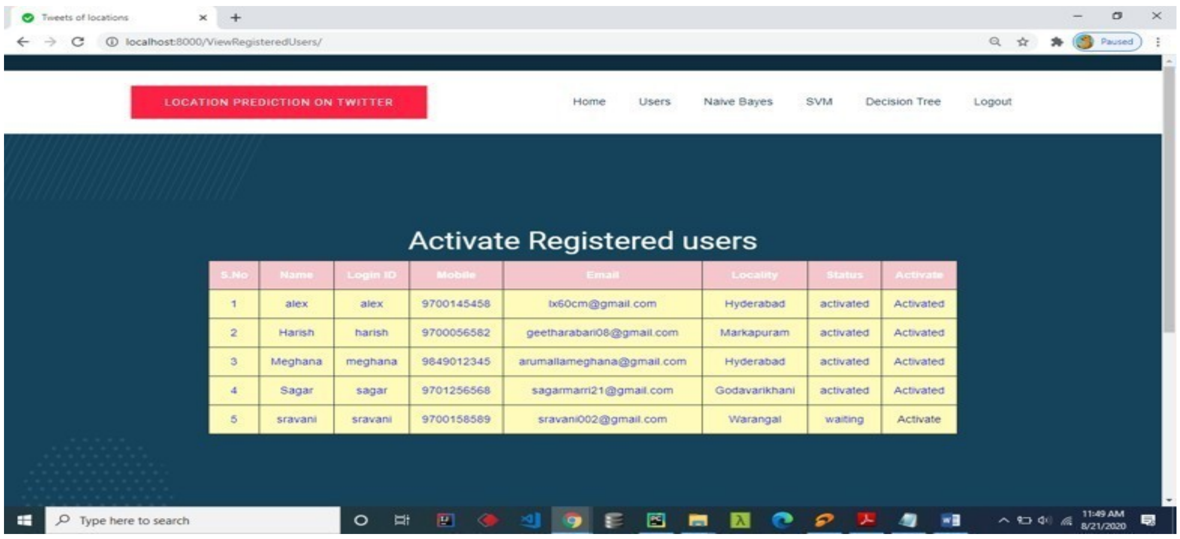
Admin Home Page:



Location prediction on Twitter using machine learning Techniques

Location prediction of users from online social media brings considerable research these days. Automatic recognition of location related with or referenced in records has been investigated for decades. As a standout amongst the online social network organization, Twitter has pulled in an extensive number of users who send a millions of tweets on regular schedule. Because of the worldwide inclusion of its users and continuous tweets, location prediction on Twitter has increased noteworthy consideration in these days. Tweets, the short and noisy and rich natured texts bring many challenges in research area for researchers. In proposed framework, a general picture of location prediction using tweets is studied. In particular, tweet location is predicted from tweet contents. By outlining tweet content and contexts, it is fundamentally featured that how the issues rely upon these text inputs. In this work, we predict the location of user from the tweet text exploiting machine learning techniques namely naive bayes, Support Vector Machine and Decision Tree.

Figure 8.12: Admin Login Home page

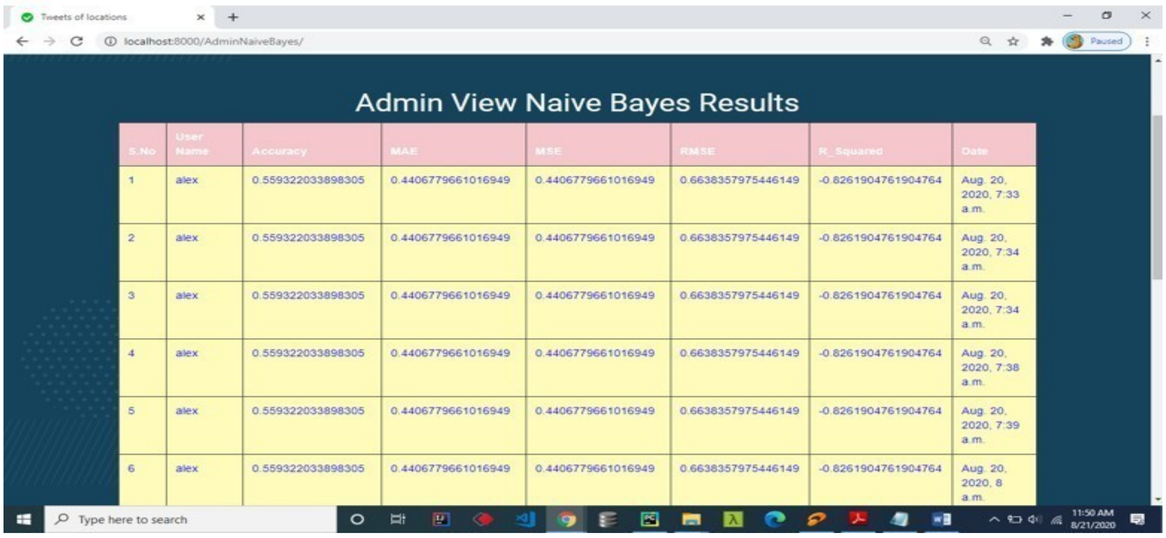


Activate Registered users

S.No	Name	Login ID	Mobile	Email	Locality	Status	Activate
1	alex	alex	9700145458	lx60cm@gmail.com	Hyderabad	activated	Activated
2	Harish	harish	9700056582	geetharabari08@gmail.com	Markapuram	activated	Activated
3	Meghana	meghana	9849012345	arumallameghana@gmail.com	Hyderabad	activated	Activated
4	Sagar	sagar	9701256568	sagarmarri21@gmail.com	Godavarikhani	activated	Activated
5	sravani	sravani	9700158589	sravani002@gmail.com	Warangal	waiting	Activate

Figure 8.13: Activate User

Admin View naïve bayes Results

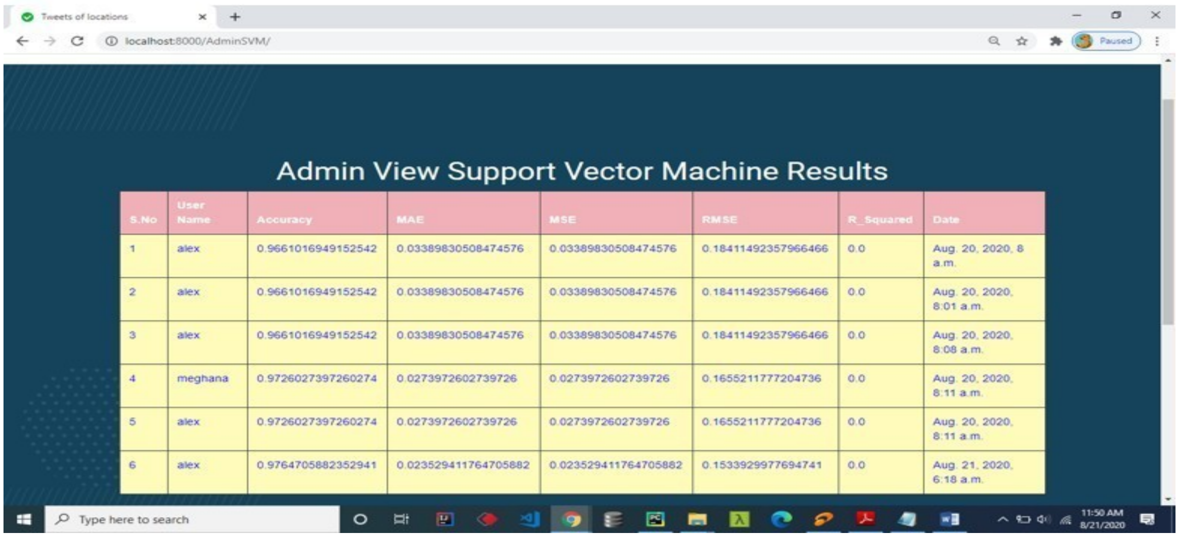


Admin View Naive Bayes Results

S.No	User Name	Accuracy	MAE	MSE	RMSE	R_Squared	Date
1	alex	0.559322033898305	0.4406779661016949	0.4406779661016949	0.6638357975446149	-0.8261904761904764	Aug. 20, 2020, 7:33 a.m.
2	alex	0.559322033898305	0.4406779661016949	0.4406779661016949	0.6638357975446149	-0.8261904761904764	Aug. 20, 2020, 7:34 a.m.
3	alex	0.559322033898305	0.4406779661016949	0.4406779661016949	0.6638357975446149	-0.8261904761904764	Aug. 20, 2020, 7:34 a.m.
4	alex	0.559322033898305	0.4406779661016949	0.4406779661016949	0.6638357975446149	-0.8261904761904764	Aug. 20, 2020, 7:38 a.m.
5	alex	0.559322033898305	0.4406779661016949	0.4406779661016949	0.6638357975446149	-0.8261904761904764	Aug. 20, 2020, 7:39 a.m.
6	alex	0.559322033898305	0.4406779661016949	0.4406779661016949	0.6638357975446149	-0.8261904761904764	Aug. 20, 2020, 8 a.m.

Figure 8.14: Admin View Naive Bayes Result

Admin View SVM Results:

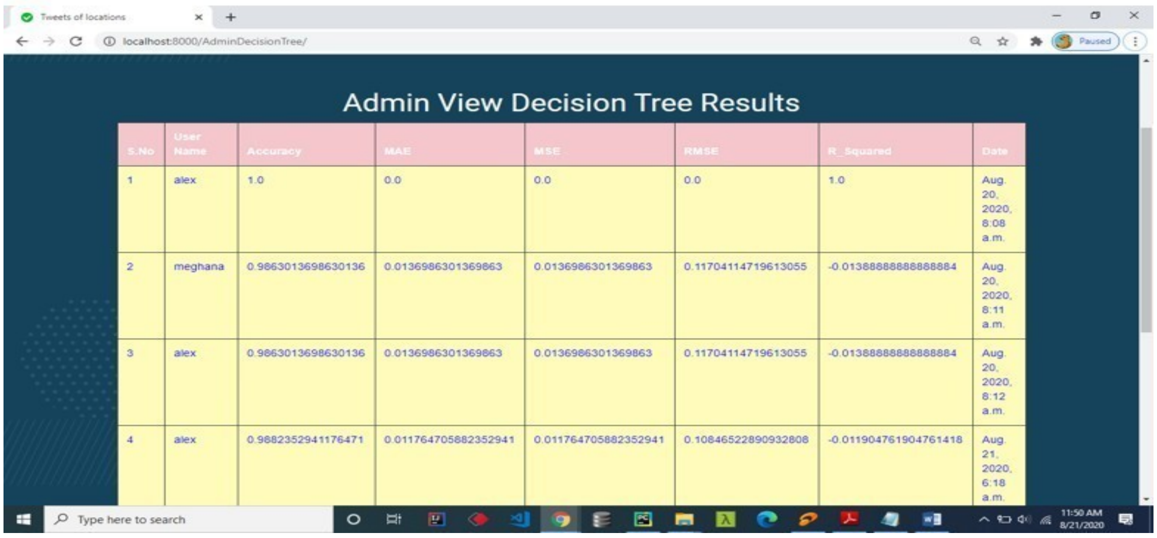


Admin View Support Vector Machine Results

S.No	User Name	Accuracy	MAE	MSE	RMSE	R_Squared	Date
1	alex	0.9661016949152542	0.03389830508474576	0.03389830508474576	0.18411492357966466	0.0	Aug. 20, 2020, 8 a.m.
2	alex	0.9661016949152542	0.03389830508474576	0.03389830508474576	0.18411492357966466	0.0	Aug. 20, 2020, 8:01 a.m.
3	alex	0.9661016949152542	0.03389830508474576	0.03389830508474576	0.18411492357966466	0.0	Aug. 20, 2020, 8:08 a.m.
4	meghana	0.9726027397260274	0.0273972602739726	0.0273972602739726	0.1655211777204736	0.0	Aug. 20, 2020, 8:11 a.m.
5	alex	0.9726027397260274	0.0273972602739726	0.0273972602739726	0.1655211777204736	0.0	Aug. 20, 2020, 8:11 a.m.
6	alex	0.9764705882352941	0.023529411764705882	0.023529411764705882	0.1533929977694741	0.0	Aug. 21, 2020, 6:18 a.m.

Figure 8.15: Admin View SVM Result

Admin View Decision Tree Results:



Admin View Decision Tree Results

S.No	User Name	Accuracy	MAE	MSE	RMSE	R_Squared	Date
1	alex	1.0	0.0	0.0	0.0	1.0	Aug. 20, 2020, 8:08 a.m.
2	meghana	0.9863013698630136	0.0136986301369863	0.0136986301369863	0.11704114719613055	-0.01388888888888884	Aug. 20, 2020, 8:11 a.m.
3	alex	0.9863013698630136	0.0136986301369863	0.0136986301369863	0.11704114719613055	-0.01388888888888884	Aug. 20, 2020, 8:12 a.m.
4	alex	0.9882352941176471	0.011764705882352941	0.011764705882352941	0.10846522890932808	-0.011904761904761418	Aug. 21, 2020, 6:18 a.m.

Figure 8.16: Admin View Decision Tree Result

CHAPTER 9

CONCLUSION

Three locations are considered from twitter data, namely home location, mentioned location and tweet location. When the twitter data is considered, geolocation prediction becomes a challenging problem. The tweet text nature and number of characters limitation make it hard to understand and analyze. In this work, we have predicted the geolocation of user from their tweet text using machine learning algorithms. We have implemented three algorithms to show the better performed one, which is suitable for geolocation prediction problem. Our experiment analysis concluded that decision tree is suitable for tweet text analysis and location prediction problem.

Future Scope

- We can locate the predicted location on a web mapping platform such as Google Maps through which we can navigate to the predicted location by taking shortest path in less time.
- Share the location predicted to officials such as the Disaster Management team, Police etc.
- In the near future, adding multiple real-time information layers such as traffic flow, incidents, signals from smart city infrastructure, … could be extremely beneficial in keeping track of an increasingly complex society.

CHAPTER 10

REFERENCES

[1] “Location Prediction on Twitter Using Machine Learning”By Indira K.Brumancia and E. Siva Kumar DOI: 10.1109/ICOEI.2019.8862768.

[2] “Uncovering the location of Twitter User ” By Renato Assunc and Wagner Me- Ira Jr. , DOI: 10.1109/BRACIS.2013.47.

[3] “Collect Data From Twitter : A step by step implementation using tweepy” By Zoumana Keitahttps ,://towardsdatascience.com/collect-data-from-twitter-a-step-by-step-implementation-using-tweepy-7526ffff2cb31.

[4] “Geo location Prediction in Social Media Data by Finding Location Indicative Words” By Han, Cook & Paul Baldwin https://aclanthology.org/C12-1064.

[5] “Python for Geospatial Data Analysis” By Bonny P. McClain.

[6] “You are Here: From the Compass to GPS, the History and Future of How We Find Ourselves” By Hiawatha Bray.