**Location prediction on Twitter using machine learning Techniques**

**1. INTRODUCTION:**

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. The techniques applied for normal documents are not suited for analysing 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 geolocation precition 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, geographical 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 collected from user profile, whereas tweet location can be arrived from user’s geo tag. Because of the first perspectives on tweet location, POIs are comprehensively received as representation of tweet regions. Mentioned Location: 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 advantage applications like recommendation systems, location based advertisements, health monitoring, and polling etc. In this study, we include two sub-modules of mentioned location: First one is recognizing the mentioned location in tweet text, which can be achieved by extracting text content from a tweet that refers to geography names. Second one is identifying the location from tweet text by solving them to entries in a geographical database.

**1.1 Objective of the project:**

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 naïve bayes, Support Vector Machine and Decision Tree.

**2. LITERATURE SURVEY:**

**“Geolocation Prediction in Social Media Data by Finding Location Indicative Words”**

Geolocation prediction is vital to geospatial applications like localised search and local event detection. Predominately, social media geolocation models are based on full text data, including common words with no geospatial dimension (e.g. today) and noisy strings (tmrw), 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 geolocation accuracy. Our results show that an information gain ratiobased approach surpasses other methods at LIW selection, outperforming state-of-the-art geolocation prediction methods by 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.

**“Geo-locating Twitter Users Based on Tweets and Social Networks”**

In this paper, we investigate the advantages of taking two dimensions of tweet content and social relationships to construct models for predicting where people settle down as their profiles reveal city- and town-level data. Based on the users who voluntarily reveal their locations in their profiles, we propose two local word filters - Inverse Location Frequency (ILF) and Remote Words (RW) filter - to identify local words in tweets content. We also extract separately the place name mentioned in tweets using the Named Entity Recognition application and then filter them by computing the city distance. We consider users’ friends and 2-hop of followings. In our experiment, we finally combine these two dimensions to estimate user location and achieve an Accuracy of 56.6% within 100 miles in city-level and 45.2% within 25 miles in town-level of their actual location which outperforms the single dimension prediction and the baseline.

**“Text-Based Twitter User Geolocation Prediction.”**

Geographical location is vital to geospatial applications like local search and event detection. In this paper, we investigate and improve on the task of text-based geolocation prediction of Twitter users. Previous studies on this topic have typically assumed that geographical references (e.g., gazetteer terms, dialectal words) in a text are indicative of its author’s location. However, these references are often buried in informal, ungrammatical, and multilingual data, and are therefore non-trivial to identify and exploit. We present an integrated geolocation prediction framework and investigate what factors impact on prediction accuracy. First, we evaluate a range of feature selection methods to obtain “location indicative words”. We then evaluate the impact of nongeotagged tweets, language, and user-declared metadata on geolocation prediction. In addition, we evaluate the impact of temporal variance on model generalisation, and discuss how users differ in terms of their geolocatability. We achieve state-of-the-art results for the text-based Twitter user geolocation task, and also provide the most extensive exploration of the task to date. Our findings provide valuable insights into the design of robust, practical text-based geolocation prediction systems.

**“Multiple Location Profiling for Users and Relationships from Social Network and Content. Proceedings of the VLDB Endowment.”**

Users' locations are important for many applications such as personalized search and localized content delivery. In this paper, we study the problem of profiling Twitter users' locations with their following network and tweets. We propose a multiple location profiling model (*MLP*), which has three key features: 1) it formally models how likely a user follows another user given their locations and how likely a user tweets a venue given his location, 2) it fundamentally captures that a user has multiple locations and his following relationships and tweeted venues can be related to any of his locations, and some of them are even noisy, and 3) it novelly utilizes the home locations of some users as partial supervision. As a result, *MLP* not only discovers users' locations *accurately* and *completely*, but also "explains" each following relationship by revealing users' true locations in the relationship. Experiments on a large-scale data set demonstrate those advantages. Particularly, 1) for predicting users' home locations, *MLP* successfully places 62% users and out-performs two state-of-the-art methods by 10% in accuracy, 2) for discovering users' multiple locations, *MLP* improves the baseline methods by 14% in recall, and 3) for explaining following relationships, *MLP* achieves 57% accuracy.

**“Home Location Identification of Twitter Users.”**

We present a new algorithm for inferring the home location of Twitter users at different granularities, including city, state, time zone or geographic region, using the content of users’ tweets and their tweeting behavior. Unlike existing approaches, our algorithm uses an ensemble of statistical and heuristic classifiers to predict locations and makes use of a geographic gazetteer dictionary to identify place-name entities. We find that a hierarchical classification approach, where time zone, state or geographic region is predicted first and city is predicted next, can improve prediction accuracy. We have also analyzed movement variations of Twitter users, built a classifier to predict whether a user was travelling in a certain period of time and use that to further improve the location detection accuracy. Experimental evidence suggests that our algorithm works well in practice and outperforms the best existing algorithms for predicting the home location of Twitter users.

**“A Simple Scalable Neural Networks based Model for Geolocation Prediction in Twitter.”**

This paper describes a model that we submitted to W-NUT 2016 Shared task #1: Geolocation Prediction in Twitter. Our model classifies a tweet or a user to a city using a simple neural networks structure with fully-connected layers and average pooling processes. From the findings of previous geolocation prediction approaches, we integrated various user metadata along with message texts and trained the model with them. In the test run of the task, the model achieved the accuracy of 40.91% and the median distance error of 69.50 km in message-level prediction and the accuracy of 47.55% and the median distance error of 16.13 km in user-level prediction. These results are moderate performances in terms of accuracy and best performances in terms of distance. The results show a promising extension of neural networks based models for geolocation prediction where recent advances in neural networks can be added to enhance our current simple model.

**“A multi-indicator approach for geolocalization of tweets,”**

Real-time information from microblogs like Twitter is useful for different applications such as market research, opinion mining, and crisis management. For many of those messages, location information is required to derive useful insights. Today, however, only around 1% of all tweets are explicitly geotagged. We propose the first multi-indicator method for determining (1) the location where a tweet was created as well as (2) the location of the user's residence. Our method is based on various weighted indicators, including the names of places that appear in the text message, dedicated location entries, and additional information from the user profile. An evaluation shows that our method is capable of locating 92% of all tweets with a median accuracy of below 30km, as well as predicting the user's residence with a median accuracy of below 5.1km. With that level of accuracy, our approach significantly outperforms existing work.

# “Towards social user profiling: Unified and discriminative influence model for inferring home locations”

Users' locations are important to many applications such as targeted advertisement and news recommendation. In this paper, we focus on the problem of profiling users' home locations in the context of social network (Twitter). The problem is nontrivial, because signals, which may help to identify a user's location, are scarce and noisy. We propose a unified discriminative influence model, named as UDI, to solve the problem. To overcome the challenge of scarce signals, UDI integrates signals observed from both social network (friends) and user-centric data (tweets) in a unified probabilistic framework. To overcome the challenge of noisy signals, UDI captures how likely a user connects to a signal with respect to 1) the distance between the user and the signal, and 2) the influence scope of the signal. Based on the model, we develop local and global location prediction methods. The experiments on a large scale data set show that our methods improve the state-of-the-art methods by 13%, and achieve the best performance.

**“On the accuracy of hyper-local geotagging of social media content,”**

Social media users share billions of items per year, only a small fraction of which is geotagged. We present a data-driven approach for identifying non-geotagged content items that can be associated with a hyper-local geographic area by modeling the location distributions of n-grams that appear in the text. We explore the trade-off between accuracy and coverage of this method. Further, we explore differences across content received from multiple platforms and devices, and show, for example, that content shared via different sources and applications produces significantly different geographic distributions, and that it is preferred to model and predict location for items according to their source. Our findings show the potential and the bounds of a data-driven approach to assigning location data to short social media texts, and offer implications for all applications that use data-driven approaches to locate content.

**“Spatially aware term selection for geotagging,”**

The task of assigning geographic coordinates to textual resources plays an increasingly central role in geographic information retrieval. The ability to select those terms from a given collection that are most indicative of geographic location is of key importance in successfully addressing this task. However, this process of selecting spatially relevant terms is at present not well understood, and the majority of current systems are based on standard term selection techniques, such as (χ2) or information gain, and thus fail to exploit the spatial nature of the domain. In this paper, we propose two classes of term selection techniques based on standard geostatistical methods. First, to implement the idea of spatial smoothing of term occurrences, we investigate the use of kernel density estimation (KDE) to model each term as a two-dimensional probability distribution over the surface of the Earth. The second class of term selection methods we consider is based on Ripley's K statistic, which measures the deviation of a point set from spatial homogeneity. We provide experimental results which compare these classes of methods against existing baseline techniques on the tasks of assigning coordinates to Flickr photos and to Wikipedia articles, revealing marked improvements in cases where only a relatively small number of terms can be selected.

**3. SYSTEM ANALYSIS**

**3.1 Existing System**

Most of the techniques used in existing works are machine learning, whereas few works in deep learning also proposed. Miura et al. on his work used neural network is implemented for twitter location prediction

**Disadvantages of Existing System:**

* Less Accuracy
* we don’t know how and why the neural network came up with a certain output.

**3.2 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 and tweet content. To handle this we used three machine learning approaches to make prediction easier and finding the best model amongst them.

**Advantages of Proposed System:**

* High Accuracy.
* Easy to find

**Modules Information:**

To implement this project author has designed following modules

1. Upload Dataset: using this module we will upload tweets dataset to application
2. Preprocess Dataset: tweets often contains raw data with special symbols stop words and URL’s and to make location prediction model we need to clean dataset. So dataset preprocess module will remove special symbols and stop words from dataset and make it clean. After cleaning data we will split dataset into train and test part where application used 80% dataset for training and 20% dataset to test trained model accuracy.
3. Run Machine Learning Algorithm: using this module we will train above 3 algorithms and then apply test data to check how may test records models are predicting correctly and based on that accuracy will be evaluated for each machine learning model.
4. Accuracy Comparison Graph: using this module we will accuracy comparison graph between all 3 algorithms
5. Predict Location from Test Tweets: using this module we will upload test tweets and then machine learning model will predict location of that test tweet.

**3.3. PROCESS MODEL USED WITH JUSTIFICATION**

**SDLC (Umbrella Model):**

**Umbrella Activity**

**Umbrella Activity**

**Umbrella Activity**

1. Feasibility Study
2. TEAM FORMATION
3. Project Specification PREPARATION

Business Requirement Documentation

ANALYSIS & DESIGN

CODE

UNIT TEST

DOCUMENT CONTROL

ASSESSMENT

TRAINING

INTEGRATION & SYSTEM TESTING

DELIVERY/INSTALLATION

ACCEPTANCE TEST

Requirements Gathering

SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

**Stages in SDLC:**

* Requirement Gathering
* Analysis
* Designing
* Coding
* Testing
* Maintenance

**Requirements Gathering** **stage:**

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

* Feasibility study is all about identification of problems in a project.
* No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
* Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

**Analysis Stage:**

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

**Designing Stage:**

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

  
When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

**Development (Coding) Stage:**

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.



The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

**Integration & Test Stage:**

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.



The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

* **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loa ded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

**Maintenance:**

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category. For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

**3.4. Software Requirement Specification**

**3.4.1. Overall Description**

A Software Requirements Specification (SRS) – a [requirements specification](http://en.wikipedia.org/wiki/Requirements_specification) for a [software system](http://en.wikipedia.org/wiki/Software_system) is a complete description of the behavior of a system to be developed. It includes a set of [use cases](http://en.wikipedia.org/wiki/Use_case) that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. [Nonfunctional requirements](http://en.wikipedia.org/wiki/Non-functional_requirements) are requirements which impose constraints on the design or implementation (such as [performance engineering](http://en.wikipedia.org/wiki/Performance_engineering) requirements, [quality](http://en.wikipedia.org/wiki/Quality_%28business%29) standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A [business analyst](http://en.wikipedia.org/wiki/Business_analyst), sometimes titled [system analyst](http://en.wikipedia.org/wiki/System_analyst), is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the [systems development lifecycle](http://en.wikipedia.org/wiki/Systems_development_life_cycle) domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

* [Business requirements](http://en.wikipedia.org/wiki/Business_requirements) describe in business terms what must be delivered or accomplished to provide value.
* Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
* Process requirements describe activities performed by the developing organization. For instance, process requirements could specify .Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:
* **ECONOMIC FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

* **Operational Feasibility**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

* **TECHNICAL FEASIBILITY**

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to .the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

**3.4.2. External Interface Requirements**

**User Interface**

The user interface of this system is a user friendly python Graphical User Interface.

**Hardware Interfaces**

The interaction between the user and the console is achieved through python capabilities.

**Software Interfaces**

The required software is python.

**SYSTEM REQUIREMENT:**

**HARDWARE REQUIREMENTS:**

# Processor - Intel i3(min)

* Speed - 1.1 GHz
* RAM - 4GB(min)
* Hard Disk - 500 GB
* Key Board - Standard Windows Keyboard
* Mouse - Two or Three Button Mouse
* Monitor - SVGA

**SOFTWARE REQUIREMENTS:**

* Operating System - Windows10(min)
* Programming Language - Python

**4. SYSTEM DESIGN**

**CLASS DIAGRAM:**

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake

****

**USECASE DIAGRAM:**

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.



**SEQUENCE DIAGRAM:**

A **sequence diagram** 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. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams**, **event scenarios**, and timing diagrams.



**COLLABORATION DIAGRAM:**

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behaviour of a system.



**COMPONENT DIAGRAM:**

In the Unified Modelling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.



**DEPLOYMENT DIAGRAM:**

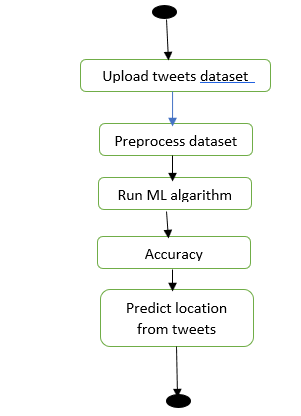
A **deployment diagram** in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.



**ACTIVITY DIAGRAM:**

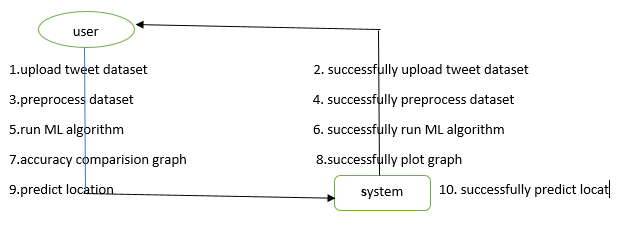
Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent.



**Data flow :**

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.



**5. IMPLEMETATION**

**5.1 Python**

Python is a general-purpose language. It has wide range of applications from Web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D). The syntax of the language is clean and length of the code is relatively short. It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

**History of Python:**

Python is a fairly old language created by Guido Van Rossum. The design began in the late 1980s and was first released in February 1991.

**Why Python was created?**

In late 1980s, Guido Van Rossum was working on the Amoeba distributed operating system group. He wanted to use an interpreted language like ABC (ABC has simple easy-to-understand syntax) that could access the Amoeba system calls. So, he decided to create a language that was extensible. This led to design of a new language which was later named Python.

**Why the name Python?**

No. It wasn't named after a dangerous snake. Rossum was fan of a comedy series from late seventies. The name "Python" was adopted from the same series "Monty Python's Flying Circus".

**Features of Python:**

**A simple language which is easier to learn**

Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax.

If you are a newbie, it's a great choice to start your journey with Python.

**Free and open-source**

You can freely use and distribute Python, even for commercial use. Not only can you use and distribute software’s written in it, you can even make changes to the Python's source code.

Python has a large community constantly improving it in each iteration.

**Portability**

You can move Python programs from one platform to another, and run it without any changes.

It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.

**Extensible and Embeddable**

Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code.

This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.

**A high-level, interpreted language**

Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on.

Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations.

**Large standard libraries to solve common tasks**

Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server? You can use MySQLdb library using import MySQLdb .

Standard libraries in Python are well tested and used by hundreds of people. So you can be sure that it won't break your application.

**Object-oriented**

Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively.

With OOP, you are able to divide these complex problems into smaller sets by creating objects.

**Applications of Python:**

**1. Simple Elegant Syntax**

Programming in Python is fun. It's easier to understand and write Python code. Why? The syntax feels natural. Take this source code for an example:

a = 2

b = 3

sum = a + b

print(sum)

**2. Not overly strict**

You don't need to define the type of a variable in Python. Also, it's not necessary to add semicolon at the end of the statement.

Python enforces you to follow good practices (like proper indentation). These small things can make learning much easier for beginners.

**3. Expressiveness of the language**

Python allows you to write programs having greater functionality with fewer lines of code. Here's a link to the source code of Tic-tac-toe game with a graphical interface and a smart computer opponent in less than 500 lines of code. This is just an example. You will be amazed how much you can do with Python once you learn the basics.

**4. Great Community and Support**

Python has a large supporting community. There are numerous active forums online which can be handy if you are stuck.

**5.2 Sample Code:**

from tkinter import messagebox

from tkinter import \*

from tkinter import simpledialog

import tkinter

import matplotlib.pyplot as plt

import numpy as np

from tkinter import ttk

from tkinter import filedialog

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from string import punctuation

from nltk.corpus import stopwords

import nltk

from nltk.stem import WordNetLemmatizer

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.preprocessing import LabelEncoder

import os

from sklearn.tree import DecisionTreeClassifier

from sklearn.svm import SVC

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score

main = Tk()

main.title("Location prediction on Twitter using machine learning Techniques")

main.geometry("1300x1200")

global filename

global X, Y

global X\_train, X\_test, y\_train, y\_test

global tfidf\_vectorizer

accuracy = []

stop\_words = set(stopwords.words('english'))

lemmatizer = WordNetLemmatizer()

textdata = []

labels = []

global classifier

location\_name = ['Arizona', 'Brazil', 'Brooklyn', 'Chennai', 'Florida', 'India', 'Indonesia',

'Kerala', 'Kirkwall', 'Pune', 'Sweden', 'United States', 'mexico', 'uk']

def cleanPost(doc):

tokens = doc.split()

table = str.maketrans('', '', punctuation)

tokens = [w.translate(table) for w in tokens]

tokens = [word for word in tokens if word.isalpha()]

tokens = [w for w in tokens if not w in stop\_words]

tokens = [word for word in tokens if len(word) > 1]

tokens = [lemmatizer.lemmatize(token) for token in tokens]

tokens = ' '.join(tokens)

return tokens

def uploadDataset():

global filename

text.delete('1.0', END)

le = LabelEncoder()

filename = filedialog.askopenfilename(initialdir="Dataset")

textdata.clear()

labels.clear()

dataset = pd.read\_csv(filename)

print(np.unique(dataset['location']))

dataset['location'] = pd.Series(le.fit\_transform(dataset['location'].astype(str)))

print(np.unique(dataset['location']))

for i in range(len(dataset)):

msg = dataset.get\_value(i, 'text')

label = dataset.get\_value(i, 'location')

msg = str(msg)

msg = msg.strip().lower()

labels.append(label)

clean = cleanPost(msg)

textdata.append(clean)

text.insert(END,clean+"\n")

def preprocess():

text.delete('1.0', END)

global X, Y

global tfidf\_vectorizer

global X\_train, X\_test, y\_train, y\_test

stopwords=stopwords = nltk.corpus.stopwords.words("english")

tfidf\_vectorizer = TfidfVectorizer(stop\_words=stopwords, use\_idf=True, ngram\_range=(1,2),smooth\_idf=False, norm=None, decode\_error='replace')

tfidf = tfidf\_vectorizer.fit\_transform(textdata).toarray()

df = pd.DataFrame(tfidf, columns=tfidf\_vectorizer.get\_feature\_names())

text.insert(END,str(df))

print(df.shape)

df = df.values

X = df[:, 0:df.shape[1]]

Y = np.asarray(labels)

indices = np.arange(X.shape[0])

np.random.shuffle(indices)

X = X[indices]

Y = Y[indices]

print(X)

print(Y)

print(Y.shape)

print(X.shape)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2)

text.insert(END,"\n\nTotal tweets found in dataset : "+str(len(X))+"\n")

text.insert(END,"Total tweets used to train machine learning algorithms : "+str(len(X\_train))+"\n")

text.insert(END,"Total tweets used to test machine learning algorithms : "+str(len(X\_test))+"\n")

def runML():

global X, Y

global tfidf\_vectorizer

global classifier

global X\_train, X\_test, y\_train, y\_test

global accuracy

accuracy.clear()

text.delete('1.0', END)

cls = GaussianNB()

cls.fit(X\_train, y\_train)

predict = cls.predict(X\_test)

a = accuracy\_score(y\_test,predict)\*100

accuracy.append(a)

text.insert(END,"Naive Bayes Accuracy : "+str(a)+"\n\n")

cls = SVC()

cls.fit(X, Y)

predict = cls.predict(X\_test)

a = accuracy\_score(y\_test,predict)\*100

accuracy.append(a)

text.insert(END,"SVM Accuracy : "+str(a)+"\n\n")

cls = DecisionTreeClassifier()

cls.fit(X, Y)

predict = cls.predict(X\_test)

a = accuracy\_score(y\_test,predict)\*100

accuracy.append(a)

text.insert(END,"Decision Tree Accuracy : "+str(a)+"\n\n")

classifier = cls

def graph():

height = accuracy

bars = ('Naive Bayes','SVM','Decision Tree')

y\_pos = np.arange(len(bars))

plt.bar(y\_pos, height)

plt.xticks(y\_pos, bars)

plt.title('Accuracy Comparison Graph')

plt.show()

def predict():

global tfidf\_vectorizer

global classifier

testfile = filedialog.askopenfilename(initialdir="Dataset")

testData = pd.read\_csv(testfile)

text.delete('1.0', END)

testData = testData.values

print(testData)

for i in range(len(testData)):

msg = testData[i]

msg1 = testData[i]

msg = msg[0]

print(msg)

review = msg.lower()

review = review.strip().lower()

review = cleanPost(review)

testReview = tfidf\_vectorizer.transform([review]).toarray()

predict = classifier.predict(testReview)[0]

print(predict)

text.insert(END,str(msg1)+" === LOCATION PREDICTED AS "+location\_name[predict]+"\n\n")

font = ('times', 15, 'bold')

title = Label(main, text='Location prediction on Twitter using machine learning Techniques')

title.config(bg='gold2', fg='thistle1')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=0,y=5)

font1 = ('times', 13, 'bold')

ff = ('times', 12, 'bold')

uploadButton = Button(main, text="Upload Tweets Dataset", command=uploadDataset)

uploadButton.place(x=20,y=100)

uploadButton.config(font=ff)

processButton = Button(main, text="Preprocess Dataset", command=preprocess)

processButton.place(x=20,y=150)

processButton.config(font=ff)

dtButton = Button(main, text="Run Machine Learning Algorithm", command=runML)

dtButton.place(x=20,y=200)

dtButton.config(font=ff)

graphButton = Button(main, text="Accuracy Comparison Graph", command=graph)

graphButton.place(x=20,y=250)

graphButton.config(font=ff)

predictButton = Button(main, text="Predict Location from Test Tweets", command=predict)

predictButton.place(x=20,y=300)

predictButton.config(font=ff)

font1 = ('times', 12, 'bold')

text=Text(main,height=30,width=120)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=330,y=100)

text.config(font=font1)

main.config(bg='DarkSlateGray1')

main.mainloop()

**6. TESTING:**

**Implementation and Testing:**

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

## Implementation

## The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifies as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

## Testing

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

### System Testing

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to use the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

**Module Testing**

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

**Integration Testing**

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

**Acceptance Testing**

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

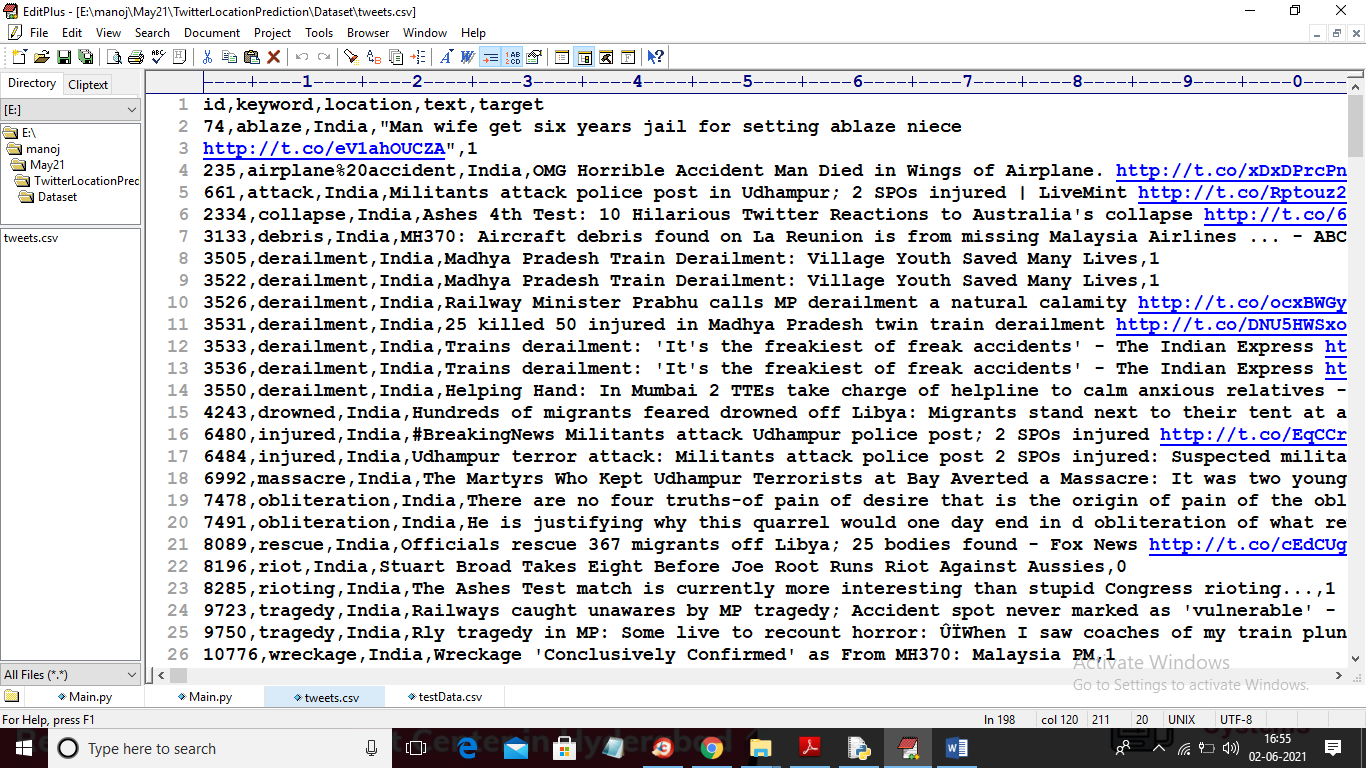
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case Id** | **Test Case Name** | **Test Case Desc.** | **Test Steps** | | | | **Test Case Status** | **Test Priority** |
| **Step** | **Expected** | | **Actual** |
| 01 | upload tweets dataset | Verify  Uploaded twitter or not | If User may not uploaded | we cannot do any further operations | we can do further operations | | High | High |
| 02 | preprocess dataset | Verify preprocess dataset or not | If preprocess dataset may not be Done | we cannot do any further operations | we can do further operations | | High | High |
| 03 | run ML algorithm | Verify run ML algorithm or not | If run ML algorithm may not be run | we cannot do any further operations | we can do further operations | | High | High |
| 04 | accuracy comparison graph | Verify accuracy comparison graph or not | If accuracy comparison graph may not Run | We cannot run  operation | We can Run the Operation | | High | High |
| 05 | predict location from tweets | Verify predict location from tweets or not | If predict location from tweets May not be Recorded | we cannot do any further operations | we can do further operations | | High | High |

**7. SCREENSHOTS:**

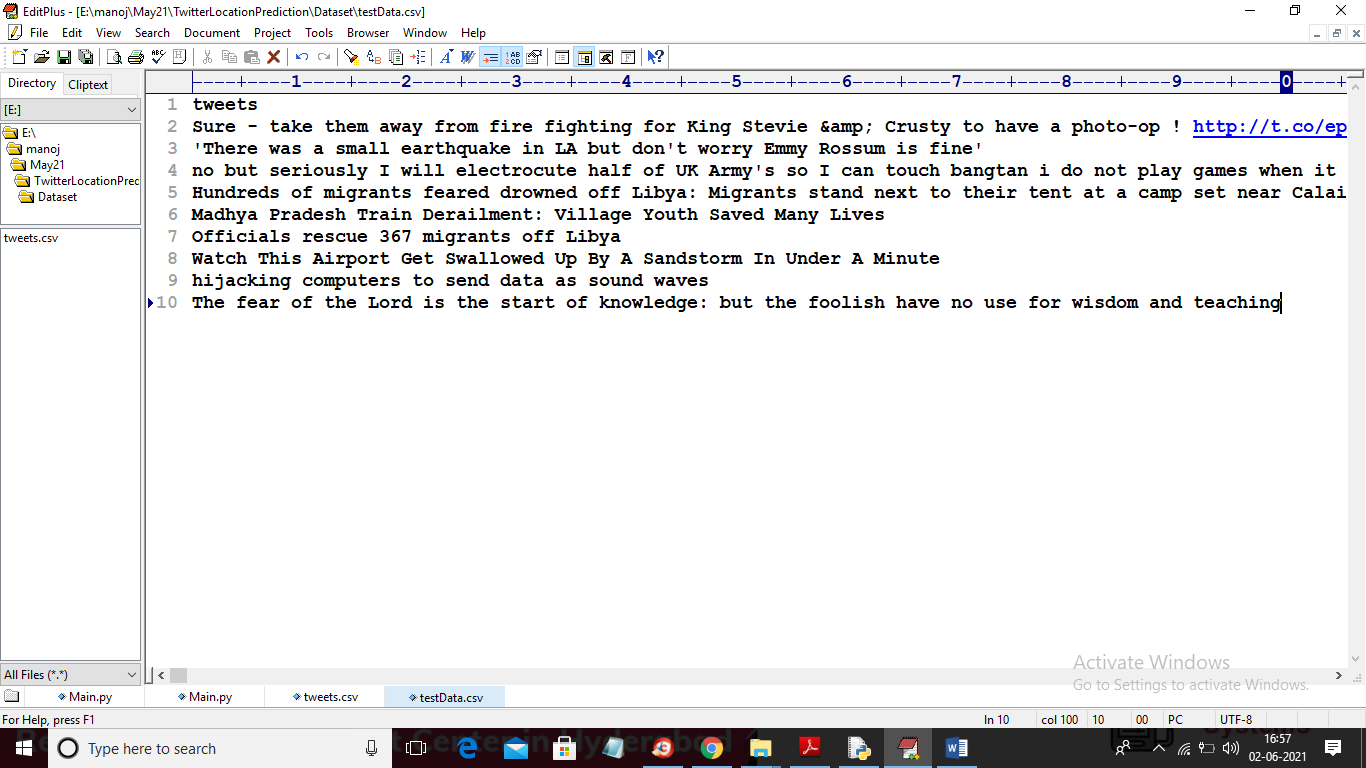
Location prediction on Twitter using machine learning Techniques

In this paper author is using machine learning algorithms such as SVM, Naïve Bayes and Decision Tree to predict location of twitter user by analysing his tweets. To predict location we have downloaded some tweets from twitter which consists of tweet messages and tweet location. We will train above machine learning algorithms with tweets dataset to predict user location. To implement this project we have designed following modules

We are using below dataset to train above machine learning algorithms and this dataset is saved inside ‘Dataset’ folder



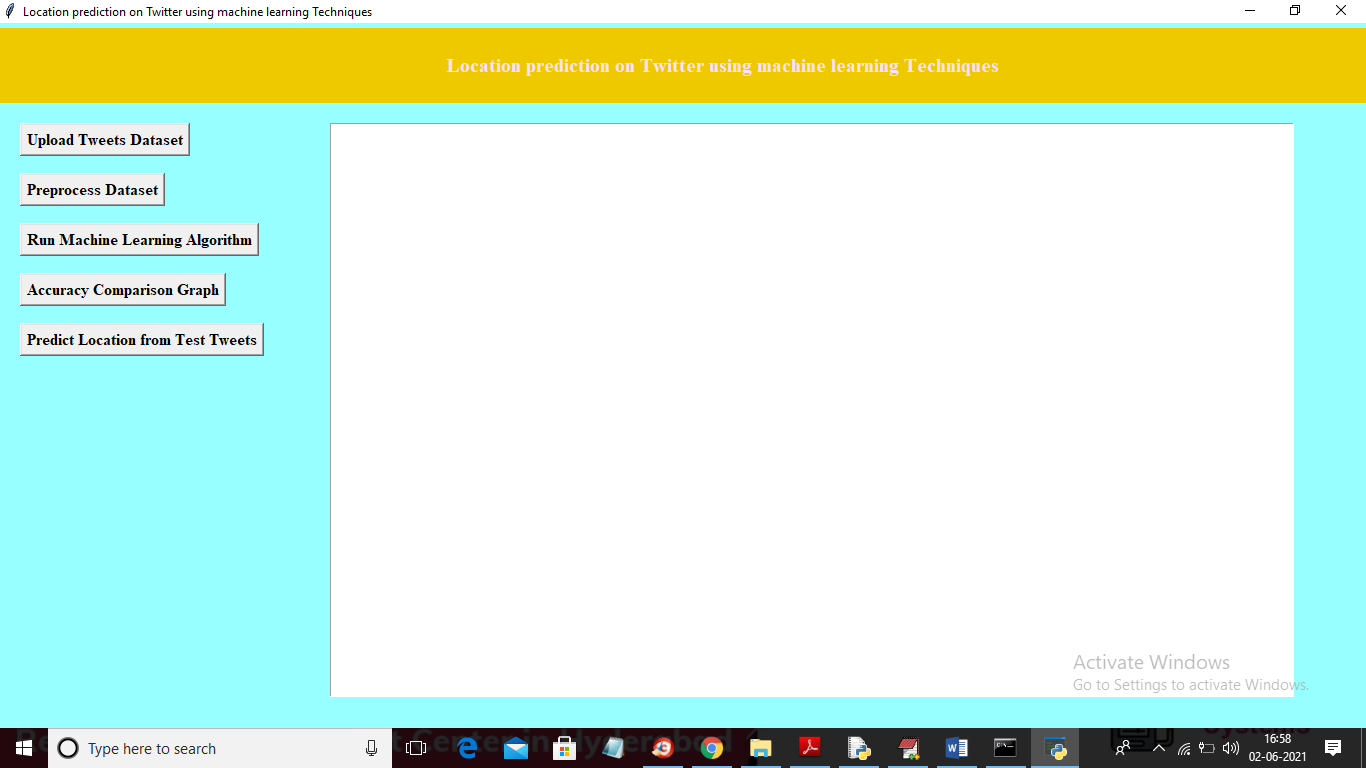
In above dataset screen first row contains dataset column names and remaining rows contains dataset values and in above dataset we are using ‘text and location’ column where text columns contains tweet message and location column contains tweet location. Below is the test dataset which contains only tweets and ML will predict location for that tweet



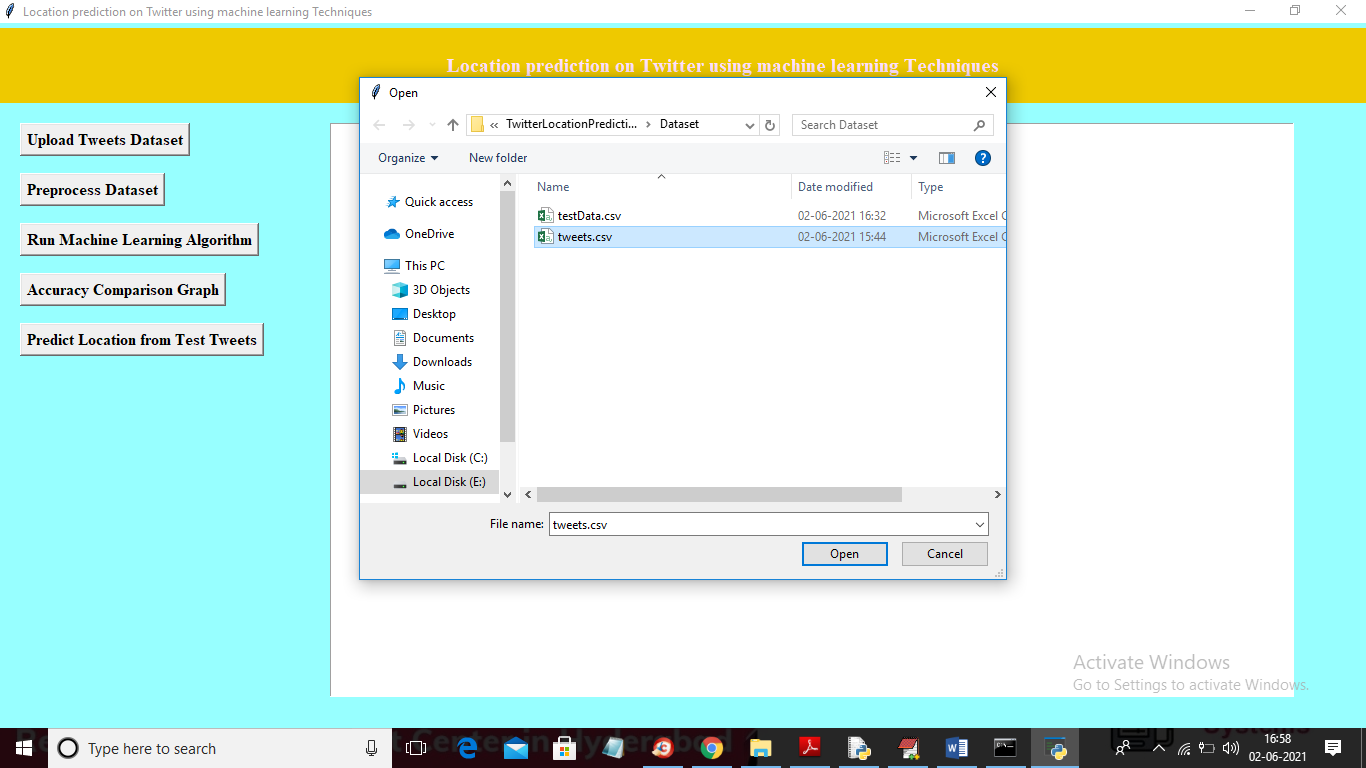
In above test dataset we can see only tweets messages are there and ML will predict location for each tweet

SCREEN SHOTS

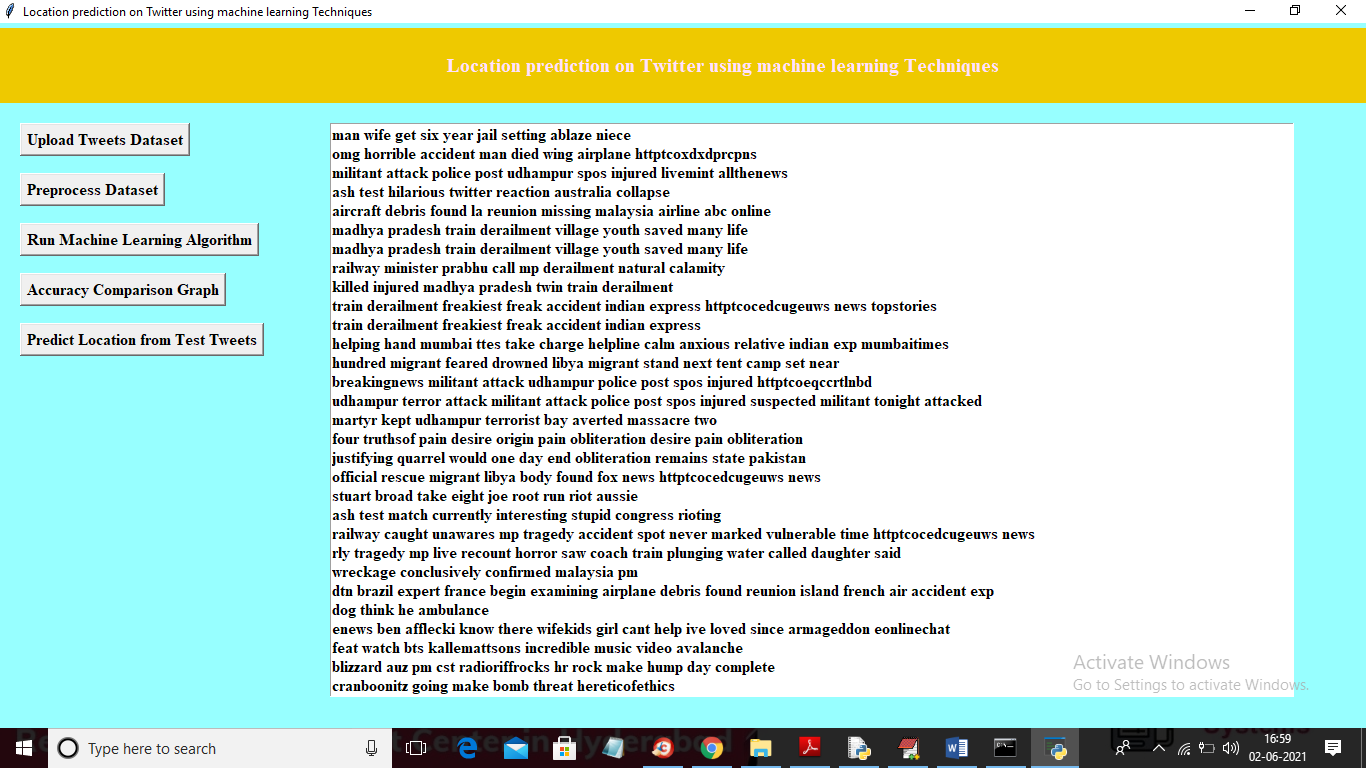
To run project double click on ‘run.bat’ file to get below screen



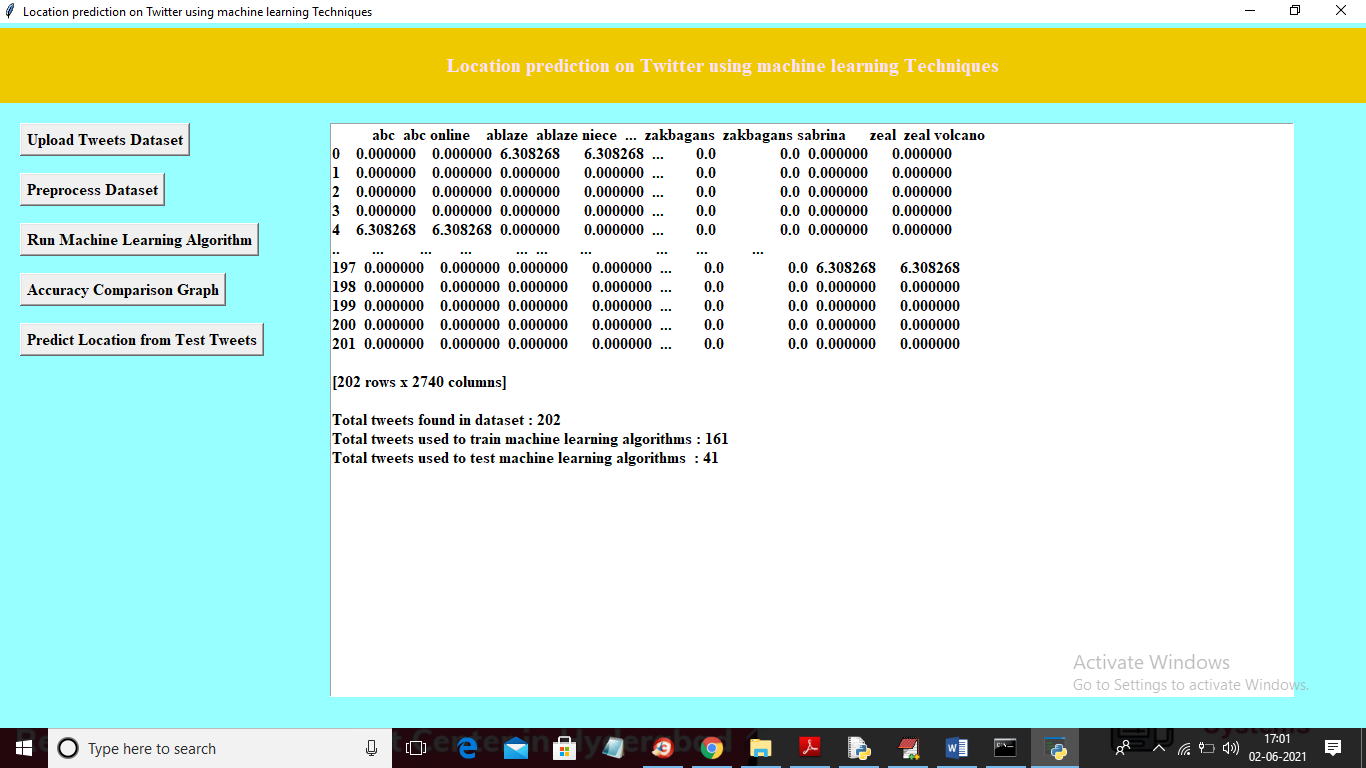
In above screen click on ‘Upload Tweets Dataset’ button to upload dataset



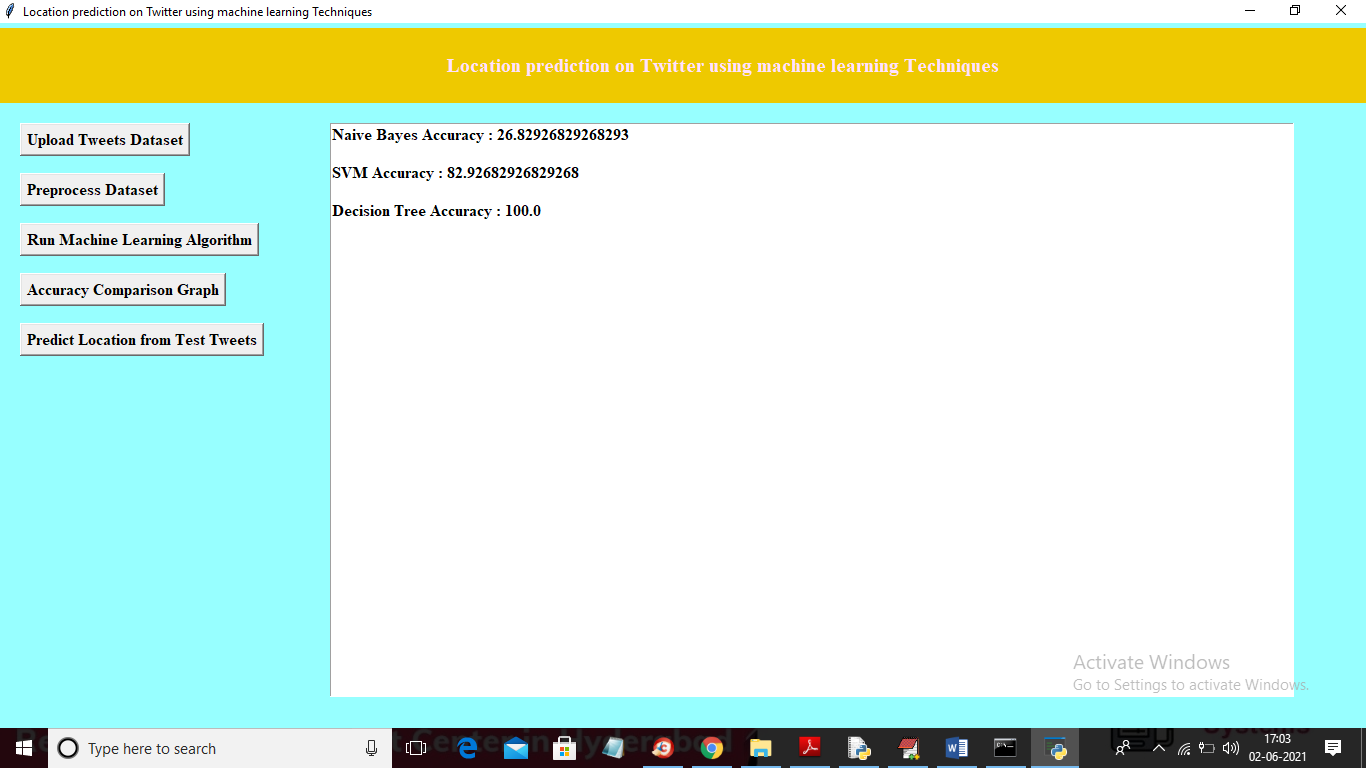
In above screen selecting and uploading ‘tweets.csv’ file and then click on ‘Open’ button to load dataset and to get below screen



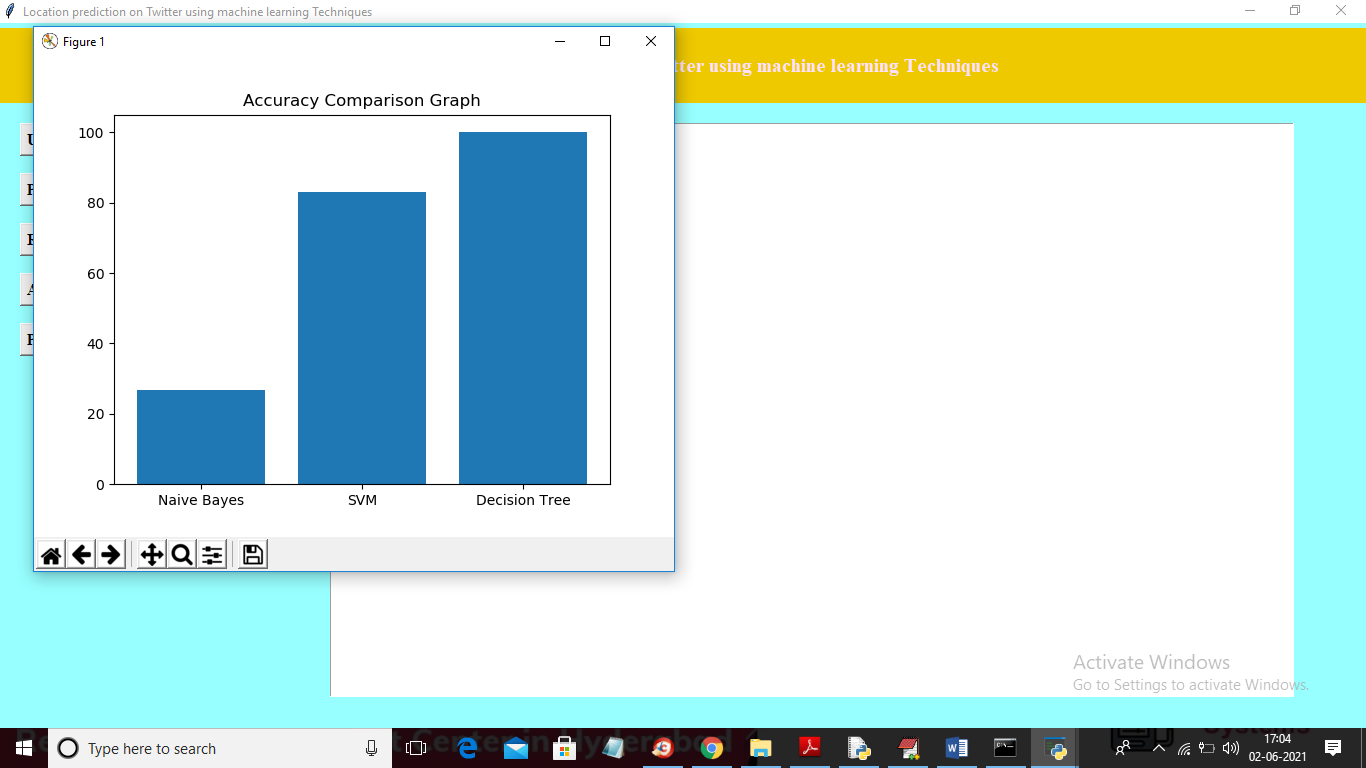
In above screen we can see all tweets are cleaned and displaying only tweets messages without special symbols and stop words and now click on ‘Preprocess Dataset’ to convert all tweets into TF-IDF vector and this vector will put all unique words into vector column and if word appear in tweet then that vector row will be filled with average of that words count and if word not appear then 0 will be put in vector column. So to convert tweets into vector click on ‘Preprocess Dataset’ button



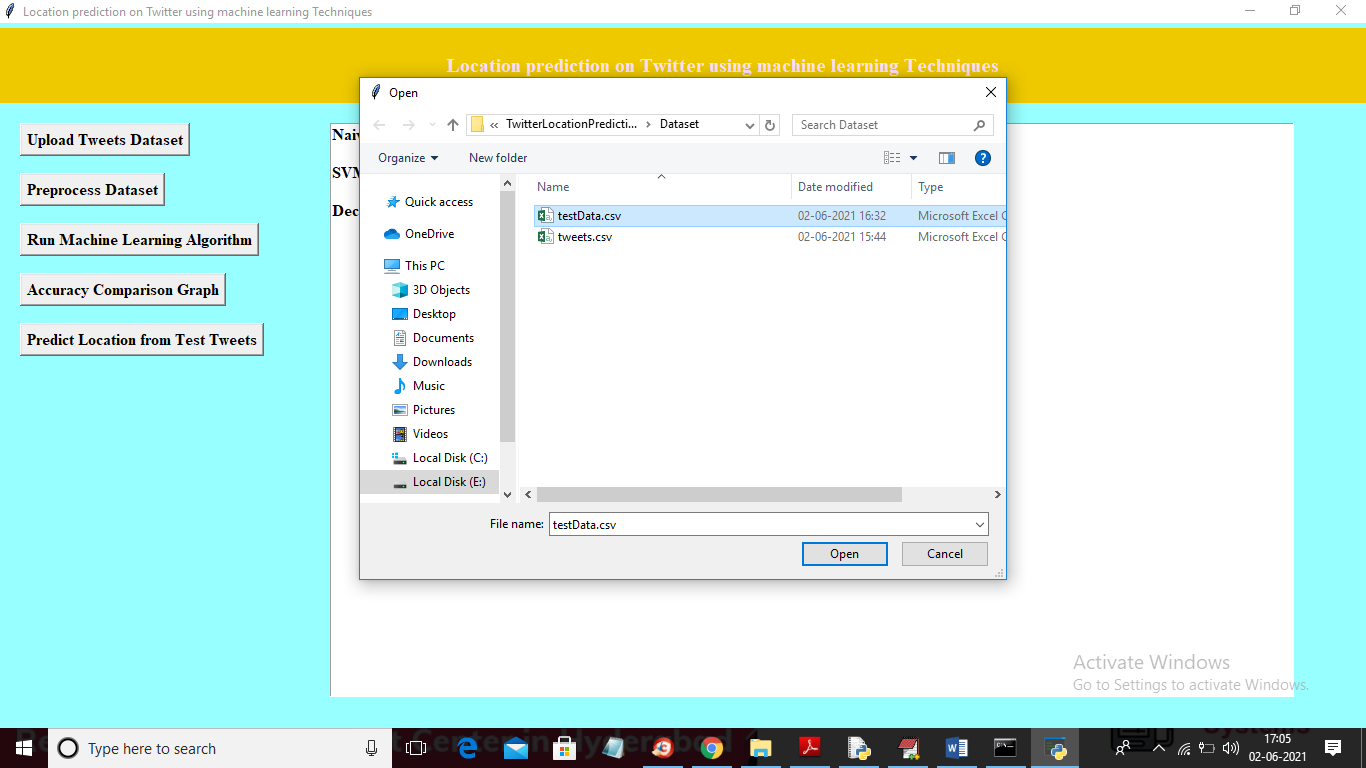
In above screen we can see in first row we have words in vector and remaining rows showing count average of that word. Now vector is ready and now click on ‘Run Machine Learning Algorithms’ button to train all 3 ML with above vector and calculate accuracy



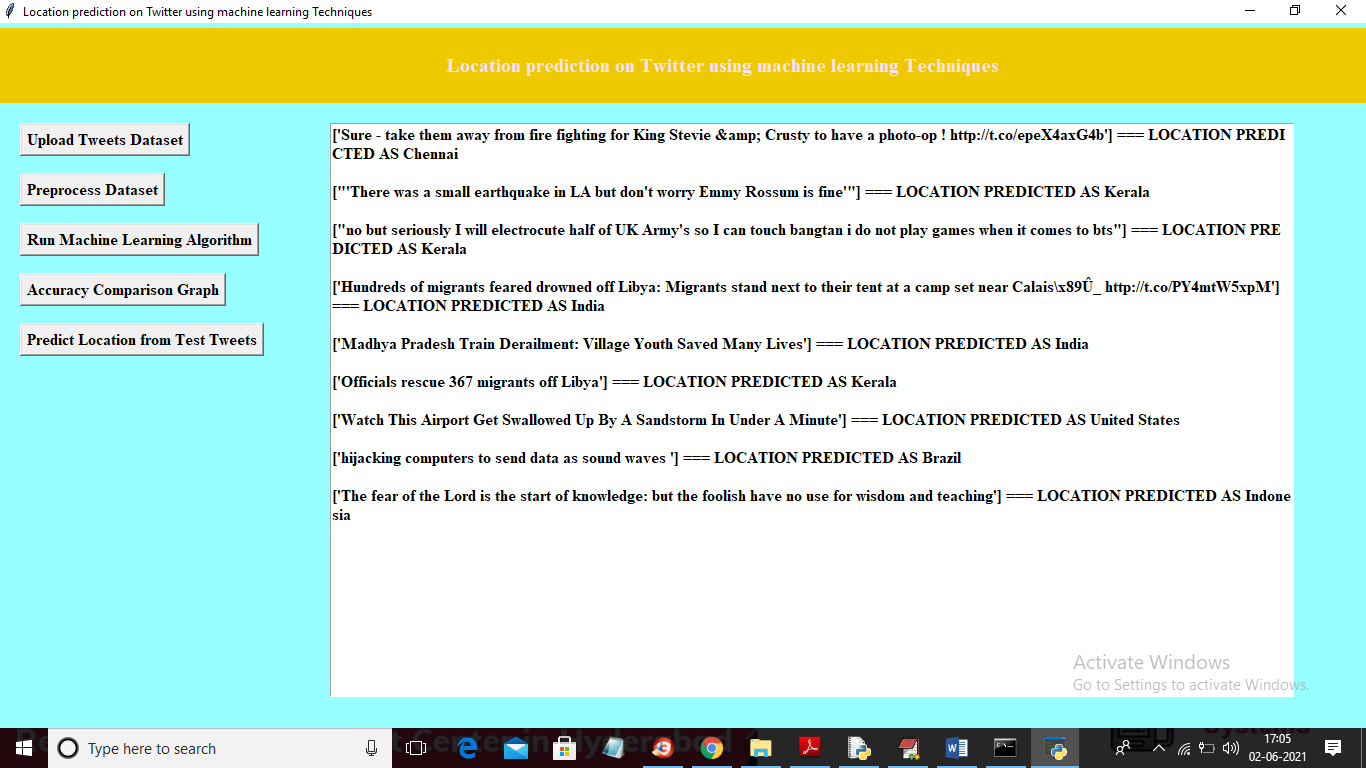
In above screen Naïve Bayes got 26% accuracy and SVM got 82 and Decision Tree got 100% accuracy and now ML models are ready and now click on ‘Accuracy Comparison Graph’ button to get below graph



In above graph x-axis represents algorithm name and y-axis represents accuracy of those algorithms and now click on ‘Predict Location from Test Tweets’ button to upload test tweets and then ML will predict location



In above screen selecting and uploading ‘testData.csv’ file and then click on ‘Open’ button to get below prediction result



In above screen in square bracket we can see tweet message and after square bracket we can see predicted location for that tweet. In above screen first tweet message location predicted as ‘Chennai'

**8. 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.

**9. REFERENCES:**

[1] Han, Bo & Cook, Paul & Baldwin, Timothy. (2012). Geolocation Prediction in Social Media Data by Finding Location Indicative Words. 24th International Conference on Computational Linguistics - Proceedings of COLING 2012: Technical Papers. 1045-1062.

[2] Ren K., Zhang S., Lin H. (2012) Where Are You Settling Down: Geo-locating Twitter Users Based on Tweets and Social Networks. In: Hou Y., Nie JY., Sun L., Wang B., Zhang P. (eds) Information Retrieval Technology. AIRS 2012. Lecture Notes in Computer Science, vol 7675. Springer, Berlin, Heidelberg.

[3] Han, Bo & Cook, Paul & Baldwin, Timothy. (2014). Text-Based Twitter User Geolocation Prediction. The Journal of Artificial Intelligence Research (JAIR). 49. 10.1613/jair.4200.

[4] Li, Rui & Wang, Shengjie & Chen-Chuan Chang, Kevin. (2012). Multiple Location Profiling for Users and Relationships from Social Network and Content. Proceedings of the VLDB Endowment. 5. 10.14778/2350229.2350273.

[5] Jalal Mahmud, Jeffrey Nichols, and Clemens Drews. 2014. Home Location Identification of Twitter Users. ACM Trans. Intell. Syst. Technol. 5, 3, Article 47 (July 2014), 21 pages.

[6] Miura, Yasuhide, Motoki Taniguchi, Tomoki Taniguchi and Tomoko Ohkuma. “A Simple Scalable Neural Networks based Model for Geolocation Prediction in Twitter.” NUT@COLING (2016).

[7] A. Schulz, A. Hadjakos, H. Paulheim, J. Nachtwey, and M. M¨ uhlh¨auser, “A multi-indicator approach for geolocalization of tweets,” in Proc. 7th Int. Conf. on Weblogs and Social Media, 2013.

[8] R. Li, S. Wang, H. Deng, R. Wang, and K. C.-C. Chang, “Towards social user profiling: unified and discriminative influence model for inferring home locations,” in Proc. 18th ACM Int. Conf. on Knowledge Discovery and Data Mining, 2012, pp. 1023–1031.

[9] B. Han, P. Cook, and T. Baldwin, “A stacking-based approach to twitter user geolocation prediction,” in Proc. 51st Annual Meeting of the Association for Computational Linguistics System Demonstrations, 2013, pp. 7–12.

[10] D. Flatow, M. Naaman, K. E. Xie, Y. Volkovich, and Y. Kanza, “On the accuracy of hyper-local geotagging of social media content,” in Proc. 8th ACM Int. Conf. on Web Search and Data Mining, 2015, pp. 127–136. [11] O. V. Laere, J. A. Quinn, S. Schockaert, and B. Dhoedt, “Spatially aware term selection for geotagging,” IEEE Trans. Knowl. Data Eng., vol. 26, no. 1, pp. 221–234, 2014.

[12] J. Mahmud, J. Nichols, and C. Drews, “Where is this tweet from? inferring home locations of twitter users,” in Proc. 6th Int. Conf. on Weblogs and Social Media, 2012.