

1. INTRODUCTION

1.1 Introduction To Analysis Of Women Safety In Indian Cities Using ML On Tweets

There are certain types of harassment and Violence that are very aggressive including staring and passing comments and these unacceptable practices are usually seen as a normal part of the urban life. There have been several studies that have been conducted in cities across India and women report similar type of sexual harassment and passing off comments by other unknown people. The study that was conducted across most popular Metropolitan cities of India including Delhi, Mumbai and Pune, it was shown that 60 % of the women feel unsafe while going out to work or while travelling in public transport. Women have the right to the city which means that they can go freely whenever they want whether it be too an Educational Institute, or any other place women want to go. But women feel that they are unsafe in places like malls, shopping malls on their way to their job location because of the several unknown Eyes body shaming and harassing these women point Safety or lack of concrete consequences in the life of women is the main reason of harassment of girls.

There are instances when the harassment of girls was done by their neighbours while they were on the way to school or there was a lack of safety that created a sense of fear in the minds of small girls who throughout their lifetime suffer due to that one instance that happened in their lives where they were forced to do something unacceptable or was sexually harassed by one of their own neighbour or any other unknown person. Safest cities approach women safety from a perspective of women rights to the affect the city without fear of violence or sexual harassment. Rather than imposing restrictions on women that society usually imposes it is the duty of society to imprecise the need of protection of women and also recognizes that women and girls also have a right same as men have to be safe in the City. Analysis of twitter texts collection also includes the name of people and name of women who stand up against sexual harassment and unethical behaviour of men in Indian cities which make them uncomfortable to walk freely. The data set that was obtained through Twitter about the status of women safety in Indian society was for the processed through machine learning algorithms for the purpose of smoothening the data by removing zero values and using Laplace and porter's theory is to developer method of analyzation of data and remove retweet and redundant data from the data set that is obtained so that a clear and original view of safety status of women in Indian society is obtained.

1.2 Existing System

In the present world, social media usage has been increasing a lot in platforms such as Twitter, Instagram, Facebook and many more. The advertisements in televisions, radio about women harassment is not so effective neither to stop such activities nor to analyse the safety of women. So here the disadvantage is that as it is one way communication, so it fails to know about other people's view. In the existing system TWEETPY package is used to download data, but everytime internet is not available.

1.3 Motivation

Staring at women and passing comments can be certain types of violence and harassments and these practices, which are unacceptable, are usually normal especially on the part of urban life. Many researches that have been conducted in India shows that women have reported sexual harassment and other practices as stated above. Such studies have also shown that in popular metropolitan cities like Delhi, Pune, Chennai and Mumbai, most women feel they are unsafe when surrounded by unknown people.

1.4 Proposed System

TWEETPY package from python is used to download tweets from twitter but every time INTERNET will not be available to download tweets online so MEETOO tweets data set must be downloaded. Machine learning algorithms such as Support Vector Machine(SVM), Decision Tree(DT), Random Forest (RF) are used and message is given as input and so the system predicts whether the message is Positive or Negative.

2. LITERATURE SURVEY

2.1 Sentiment Analysis of Top Colleges in India Using Twitter Data

Authors: Nehal Mangain, Ekta Mehta, Ankush Mittal, Gaurav Bhatt

The advent and growth of social media in the world, stack holders often take to expressing their opinions on popular social medias. It presents a challenge for analysis because of its humongous and disorganized nature.

Taking the additional pre-processing methods like the expansion of net lingo and removal of duplicate tweets, a probabilistic model based on bayes theorem was used for spelling correction. This also highlights a comparison between the results obtained by exploiting the machine learning algorithms such as Naive Bayes, SVM.

People are extensively using this platform to share their thoughts loud and clear. Twitter is one such well known micro-blogging site getting around 500 million tweets per day. Twitter user post everyday about various subjects like products, services, places, personalities etc. A number of studies focus upon the popularity and reviews of the products and services offered by different organizations.

Researchers have also been working upon prediction of accuracy of tested dataset using machine learning algorithms. And some other researchers used Natural Language Processing techniques for Sentiment Analysis and compared machine learning methods and Ensemble methods to improve on the accuracy of the classification. The tweets also provide information about the user, location, time-zone etc.

The general sentiment derived from the dataset regarding the three colleges that are AIIMS, IIT, NIT. AIIMMS had the highest positivity average sentiment and the ratio for positive to negative tweets. The predictions made by the machine learning algorithms showed high accuracy. For the AIIMS dataset, Naïve Bayes achieved an accuracy of 90%, SVM an accuracy of 91% and ANN an accuracy of 92.6%.

2.2 Design of women safety device.

Authors: Divya Chitkara, Nipun Sachideva, Yash Dev Vashist

Women have been facing a lot of problems in the society though there are unprecedented number of laws against domestic violence, sexual assault and other violence. But there are major

challenges in implementing such laws. it is making the society insecure for the women and the oppressors are not getting punishment.

There are many strategies for women to get rid of oppressors but they are not much safe to carry with them and it's a tedious task to carry also. So for this reason this device is used which is also cost efficient when compared to other approaches. This device which has been proposed in this system is the solution for all women who deserve a safe and secure world. It is a type of attacking strategy against women violence and eve teasing. This device has been made in the form of glove and it is completely electronic but people may doubt that as it is electronic it might cause harm for women but it is completely safe because of the usage of insulator as the glove is insulated.

Device is in the form of a glove consisting of electronic circuitry. Person using this glove has to activate the circuitry installed within the glove to attack the oppressor and protect herself from any danger. The conductive layer on activation gives a daunting shock to the oppressor, it doesn't kill the oppressor but it gives a shock which will be tight and muscle contraction takes place.

This will render the oppressor stunned and intimidated. So the wearer can easily overcome any aggressor with absolute ease. So this tool has become effective for eradicating violence against women.

2.3 Determining the sentiment of opinions

Authors: Soo-Min-Kim, and Eduard Hovy

Identifying sentiments is a challenging problem. Here the system automatically finds the opinions and sentiments of people. We describe an opinion as a quadruple [Topic, Holder, Claim, Sentiment] in which the Holder believes a Claim about the Topic and in many cases associates a Sentiment, such as good or bad, with a belief. Sentiments always involve the Holder's emotions or desires, and may be present explicitly or implicitly. To avoid the problem of differentiating between shades of sentiments, we simplify the problem based on the expressions of positive, negative or neutral together with their holders.

Sentences without sentiments are considered as separate set. Problem is approached in stages, starting with words and moving on to sentences. The system operates in four steps. First, sentences containing both phrase and holder candidates are selected. Next, the holder-based regions of

opinions are delimited. Then, the sentence sentiment classifier calculates the polarity of all sentiment-bearing words individually. Finally, the system combines them to produce the holder sentiment for the whole sentence.

Algorithms used are Word Sentiment Classifier [Word Classification Models], Sentence Sentiment Classifier [Holder Identification, Sentiment Region, Classification Models]. Building three models in Classification Models namely Model 0: considering the polarities of the sentiments and not the strengths, Model 1: harmonic mean of the sentiment strengths in the region, Model 2: harmonic mean of the sentiment strengths in the region. Word Sentiment Classifier is done based on considering Human-Human Agreement and Human-Machine agreement. Sentence Sentiment Classifier is tested based on Human Annotated Data. The best overall performance is provided by Model 0, though Model 1 average is best. Concluding, Sentiment recognition is a challenging and difficult part of understanding opinions. This can be further done by extending to more difficult cases such as sentences with weak-opinion-bearing words or sentences with multiple opinions about a topic. To improve identification, parser can be used and explore other learning techniques such as decision lists or SVMs. Nonetheless, encouraging results can be obtained even with relatively simple models and only a small amount of manual seeding effort.

2.4 Contextual phrase-level polarity analysis using lexical affect scoring and syntactic n-grams

Authors: Apoorv Agarwal, Fadi Biadisy, and Kathleen R. Mckeown

We present a classifier to predict contextual polarity of subjective phrases in a sentence. Our approach features lexical scoring derived from the Dictionary of Affect in Language (DAL) and extended through WordNet, allowing us to automatically score the vast majority of words in our input avoiding the need for manual labeling. We augment lexical scoring with n-gram analysis to capture the effect of context. We combine DAL scores with syntactic constituents and then extract n-grams of constituents from all sentences. We also use the polarity of all syntactic constituents within the sentence as features. Our results show significant improvement over a majority class baseline as well as a more difficult baseline consisting of lexical n-grams.

2.5 Robust sentiment detection on twitter from biased and noisy data

Authors: Luciano Barbosa and Junlan Feng

In this paper, we propose an approach to automatically detect sentiments on Twitter messages (tweets) that explores some characteristics of how tweets are written and meta-information of the words that compose these messages. Moreover, we leverage sources of noisy labels as our training data. These noisy labels were provided by a few sentiment detection websites over twitter data. In our experiments, we show that since our features are able to capture a more abstract representation of tweets, our solution is more effective than previous ones and also more robust regarding biased and noisy data, which is the kind of data provided by these sources.

2.6 Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis.

Authors: Michael Gamon.

We demonstrate that it is possible to perform automatic sentiment classification in the very noisy domain of customer feedback data. We show that by using large feature vectors in combination with feature reduction, we can train linear support vector machines that achieve high classification accuracy on data that present classification challenges even for a human annotator. We also show that, surprisingly, the addition of deep linguistic analysis features to a set of surface level word n-gram features contributes consistently to classification accuracy in this domain.

2.7 Study of Twitter sentiment analysis using machine learning algorithms on Python

Authors: Gupta B, Negi M, Vishwakarma K, Rawat G & Badhani P

Twitter is a platform widely used by people to express their opinions and display sentiments on different occasions. Sentiment analysis is an approach to analyze data and retrieve sentiment that it embodies. Twitter sentiment analysis is an application of sentiment analysis on data from Twitter (tweets), in order to extract sentiments conveyed by the user. In the past decades, the research in this field has consistently grown. The reason behind this is the challenging format of the tweets which makes the processing difficult. The tweet format is very small which generates a whole new dimension of problems like use of slang, abbreviations etc. In this paper, we aim to review some papers regarding research in sentiment analysis on Twitter, describing the

methodologies adopted and models applied, along with describing a generalized Python based approach.

3. SYSTEM ANALYSIS

3.1 SYSTEM REQUIREMENTS

3.1.1 SOFTWARE REQUIREMENTS

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation. The appropriation of requirements and implementation constraints gives the general overview of the project in regards to what the areas of strength and deficit are and how to tackle them.

- Python idle 3.7 version
- Anaconda 3.7
- Jupiter
- Google colab

3.1.2 HARDWARE REQUIREMENTS

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

- Operating system: windows, linux
- Processor : minimum intel i3
- Ram: minimum 4 gb
- Hard disk: minimum 250gb

FUNCTIONAL REQUIREMENTS

- 1.Data Collection
- 2.Data Preprocessing
- 3.Training and Testing
- 4.Modelling
- 5.Predicting

NON FUNCTIONAL REQUIREMENTS

NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system.

Example of non functional requirement, “how fast does the website load?” Failing to meet non-functional requirements can result in systems that fail to satisfy user needs.

Non- functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users are > 10000. Description of non-functional requirements is just as critical as a functional requirement.

- Usability requirement
- Serviceability requirement
- Manageability requirement
- Recoverability requirement
- Security requirement
- Data Integrity requirement
- Capacity requirement
- Availability requirement
- Scalability requirement
- Interoperability requirement
- Reliability requirement
- Maintainability requirement
- Regulatory requirement
- Environmental requirement

3.2 DOMAIN

3.2.1 Python

Python is a general purpose, dynamic, high level, and interpreted programming language It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures. Python is easy to learn yet powerful and versatile scripting language, which makes it attractive for Application Development. Python's syntax and dynamic typing with its interpreted nature make it an ideal language for scripting and rapid application development. Python supports multiple programming pattern, including object

oriented, imperative, and functional or procedural programming styles. Python is not intended to work in a particular area, such as web programming. That is why it is known as multipurpose programming language because it can be used with web, enterprise, 3D CAD, etc. We don't need to use data types to declare variable because it is dynamically typed so we can write `a=10` to assign an integer value in an integer variable. Python makes the development and debugging fast because there is no compilation step included in Python development and edit-test-debug cycle is very fast.

3.2.2 Python's Feature Set

The factors that played an important role in moulding the final form of the language and are given by

1.Easy to code: Python is very easy to code. Compared to other popular languages like Java and C++, it is easier to code in Python.

2.Easy to read: Being a high-level language, Python code is quite like English. Looking at it, you can tell what the code is supposed to do. Also, since it is dynamically-typed, it mandates indentation. This aids readability.

3.Expressive: Suppose we have two languages A and B, and all programs that can be made in A can be made in B using local transformations. However, there are some programs that can be made in B, but not in A, using local transformations. Then, B is said to be more expressive than A. Python provides us with a myriad of constructs that help us focus on the solution rather than on the syntax.

4.Free and Open-Source: Firstly, Python is freely available. You can download it from the link. Secondly, it is open source. This means that its source code is available to the public. You can download it, change it, use it, and distribute it. This is called FLOSS (Free/Libre and Open Source Software).

5.High- Level: This means that as programmers, we don't need to remember the system architecture. Nor do we need to manage the memory. This makes it more programmer- friendly and is one of the key python features.

6.Portable: We can take one code and run it on any machine, there is no need to write different code for different machines. This makes Python a portable language.

7.Interpreted: If you are any familiar with languages like C++ or Java, you must first compile it, and then run it. But in Python, there is no need to compile it. Internally, its source code is

converted into an immediate form called bytecode. So, all you need to do is to run your Python code without worrying about linking to libraries, and a few other things. By interpreted, we mean the source code is executed line by line, and not all at once. Because of this, it is easier to debug your code. Also, interpreting makes it just slightly slower than Java, but that does not matter compared to the benefits it has to offer.

8.Object-Oriented: A programming language that can model the real world is said to be object-oriented. It focuses on objects and combines data and functions. Contrarily, a procedure-oriented language revolves around functions, which are code that can be reused. Python supports both procedure-oriented and object-oriented programming which is one of the key python features. It also supports multiple inheritance, unlike Java. A class is a blueprint for such an object. It is an abstract data type and holds no values.

9.Extensible: If needed, you can write some of your Python code in other languages like C++. This makes Python an extensible language, meaning that it can be extended to other languages.

10.Embeddable: We just saw that we can put code in other languages in our Python source code. However, it is also possible to put our Python code in a source code in a different language like C++. This allows us to integrate scripting capabilities into our program of the other language.

11.Large Standard Library: Python downloads with a large library that you can use so you don't have to write your own code for every single thing. There are libraries for regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and a lot of other functionality.

12.GUI Programming: It also supports graphical user interface. Dynamically Typed Python is dynamically-typed. This means that the type for a value is decided at runtime, not in advance. This is why we don't need to specify the type of data while declaring it.

Modules used in Project :-

Numpy:

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code

- Useful linear algebra, Fourier transform, and random number capabilities Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Scikit-learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use

3.2.3 Anaconda Distribution

Anaconda Navigator is a free open source software distribution of the languages like Python and R for Machine learning and Data Science. The aim is to simplify the package management and deployment. The Anaconda distribution is used by over 6 million users, and it includes more than 250 popular data science packages suitable for Windows, Linux, and MacOS. Anaconda distribution comes with more than 1,000 data packages as well as the conda package and virtual environment manager, called Anaconda Navigator [6], so it eliminates the need to learn to install each library independently.

Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux. Navigator is automatically included with Anaconda version 4.0.0 or higher.

Anaconda Cloud

Anaconda Cloud is a package management service by Anaconda where you can find, access, store and share public and private notebooks, environments, and conda and PyPI packages. Cloud hosts useful Python packages, notebooks and environments for a wide variety of applications. You do not need to log in or to have a Cloud account, to search for public packages download and install them. You can build new packages using the Anaconda Client command line interface (CLI), then manually or automatically upload the packages to Cloud.

Conda

Conda is an open source package management system and environment management system that runs on Windows, macOS and Linux. Conda quickly installs, runs and updates packages and their dependencies. Conda easily creates, saves, loads and switches between. Parts of Anaconda distribution

1. Jupyter Lab
2. Jupyter Notebook
3. QtConsole
4. Spyder
5. Glueviz
6. Orange
7. RStudio
8. Visual Studio Code

Jupyter

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Language of choice

It supports more than 40 programming languages including Python, Spyder etc.

Share Notebooks

Notebooks can be shared with others using email, Dropbox, GitHub and the Jupyter Notebook Viewer.

Interactive Output

Your code can produce rich, interactive output, HTML, images, videos, LaTeX, and custom MIME types



Fig 3.1 Anaconda Jupyter Notebook

VS Code

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages and runtimes (such as C++, C#, Java, Python, PHP, Go, .NET).

4. METHODOLOGY

Various methodology can be applied to implement the system that analyses women safety. Few of those methodology are discussed below.

4.1 Textblob

Textblob is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more. Sentiment Analysis can assist us in determining the mood and feelings of the general public as well as obtaining useful information about the setting. Sentiment Analysis is the process of assessing data and categorizing it according to the needs. The polarity and subjectivity of a statement are returned by TextBlob. The range of polarity is $[-1,1]$, with -1 indicating a negative sentiment and 1 indicating a positive sentiment. Negative words are used to change the polarity of a sentence. Semantic labels in TextBlob aid in fine-grained analysis. Emoticons, exclamation marks, and emojis, for example. subjectivity falls under the numeric range of $[0,1]$. The degree of personal opinion and factual information in a text is measured by subjectivity. Because of the text's heightened subjectivity, it contains personal opinion rather than factual information. There's one more setting in TextBlob: intensity. The 'intensity' is used by TextBlob to calculate subjectivity. The intensity of a word influences whether it modifies the next word. Adverbs are used as modifiers in English. By providing an input sentence, the TextBlob's sentiment property returns a named tuple with polarity and subjectivity scores. The polarity score ranges from -1.0 to 1.0 and the subjectivity ranges from 0.0 to 1.0 where 0.0 is an objective statement and 1 is a subjective statement.

4.2 Tokenization using textblob

In any NLP pipeline, tokenization is considered to be the first step of the pipeline process. A tokenizer breaks down unstructured data and natural language text into chunks of information that can be considered discrete elements. The document's token occurrences can be used to generate a vector that reflects the document. An unstructured string (text document) is turned into a numerical data structure suitable for machine learning in a matter of seconds. They can also be utilized to direct a computer's helpful operations and responses. They could potentially be used as features in a machine learning pipeline to prompt more

complex decisions or actions. Recent deep learning-powered NLP algorithms have interpreted tokens in the context in which they appear, even in very extensive contexts. Because the system can infer the “meaning” of unusual tokens based on their context, this ability mitigates the “Heteronyms” problem and makes NLP systems more robust in the face of rare tokens. The way we tokenize text has altered as a result of these relatively new capabilities in the realm of NLP. A pipeline approach is commonly used to tokenize non-deep learning systems. After separating the text into token candidates (by splitting on white spaces or using more complex heuristics), related tokens are merged and noisy tokens are removed.

4.3 Lemmatization using textblob

Lemmatization is the process of aggregating together the derived forms of a word into a single element or item. The “lemmatize” property helps us achieve this functionality. In Natural Language Processing (NLP) and machine learning, lemmatization is one of the most used text techniques during the pre-processing phase. Stemming is a Natural Language Processing concept that is nearly comparable to this one. We strive to reduce a given term to its root word in both stemming and lemmatization. In the stemming process, the root word is termed a stem, and in the lemmatization process, it is called a lemma. Lemmatization has the advantage of being more precise. Lemmatization is important if you’re working with an NLP application like a chat bot or a virtual assistant when comprehending the meaning of the discussion is critical. However, this precision comes at a price. Because lemmatization entails determining a word’s meaning from a source such as a dictionary, it takes a long time. As a result, most lemmatization methods are slower than stemming techniques. Although there is a processing expense for lemmatization, computational resources are rarely a consideration in an ML challenge.

4.4 Flask

Flask is a web application framework written in Python. Armin Ronacher, who leads an international group of Python enthusiasts named Pocco, develops it. Flask is based on Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

What is Web Framework?

Web Application Framework or simply Web Framework represents a collection of libraries and modules that enables a web application developer to write applications without having to bother about low-level details such as protocols, thread management etc.

What is Flask?

Flask is a web application framework written in Python. It is developed by **Armin Ronacher**, who leads an international group of Python enthusiasts named Pocco. Flask is based on the Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

WSGI

Web Server Gateway Interface (WSGI) has been adopted as a standard for Python web application development. WSGI is a specification for a universal interface between the web server and the web applications.

Werkzeug

It is a WSGI toolkit, which implements requests, response objects, and other utility functions. This enables building a web framework on top of it. The Flask framework uses Werkzeug as one of its bases.

Jinja2

Jinja2 is a popular templating engine for Python. A web templating system combines a template with a certain data source to render dynamic web pages.

Flask is often referred to as a micro framework. It aims to keep the core of an application simple yet extensible. Flask does not have built-in abstraction layer for database handling, nor does it have form a validation support. Instead, Flask supports the extensions to add such functionality to the application.

Various Machine Learning algorithms are also used such as Decision Tree(DT). Support Vector Machine(SVM), Random Forest(RF).

4.5 Decision Tree

Decision Tree as the name suggest the main idea behind decision tree algorithm is to make a tree like structure and get the answers in form of true or false. The model begins from a root node and ends on the decision. Each node receives a Yes/No question and answer is passed on to the next node. Root node gets all the input of the training dataset. The challenge to assembling such a tree is that question to ask at a node and when. To do this, decision tree algorithmic program uses accepted indices like entropy or Gini-impurity to quantify an

uncertainty or impurity related to an explicit node. Equations (1) and (2) show however entropy and Gini impurity are calculated, severally, for a setoff information. Within the equations, C is that the variety of classes[1]. There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree: Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand. The logic behind the decision tree can be easily understood because it shows a tree-like structure.

Decision tree terminologies

Root Node: Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.

Leaf Node: Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.

Splitting: Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.

Branch/Sub Tree: A tree formed by splitting the tree. **Pruning:** Pruning is the process of removing the unwanted branches from the tree.

Parent/Child node: The root node of the tree is called the parent node, and other nodes are called the child nodes.

How does the Decision Tree algorithm Work? In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node. For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm and as shown in figure 5.1.

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).

Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

$$H(s) = - \sum p(c) \log p(c) \text{ for all } c \in C \quad H(s) = 1 - \sum p(c)^2 \text{ for all } c \in C$$

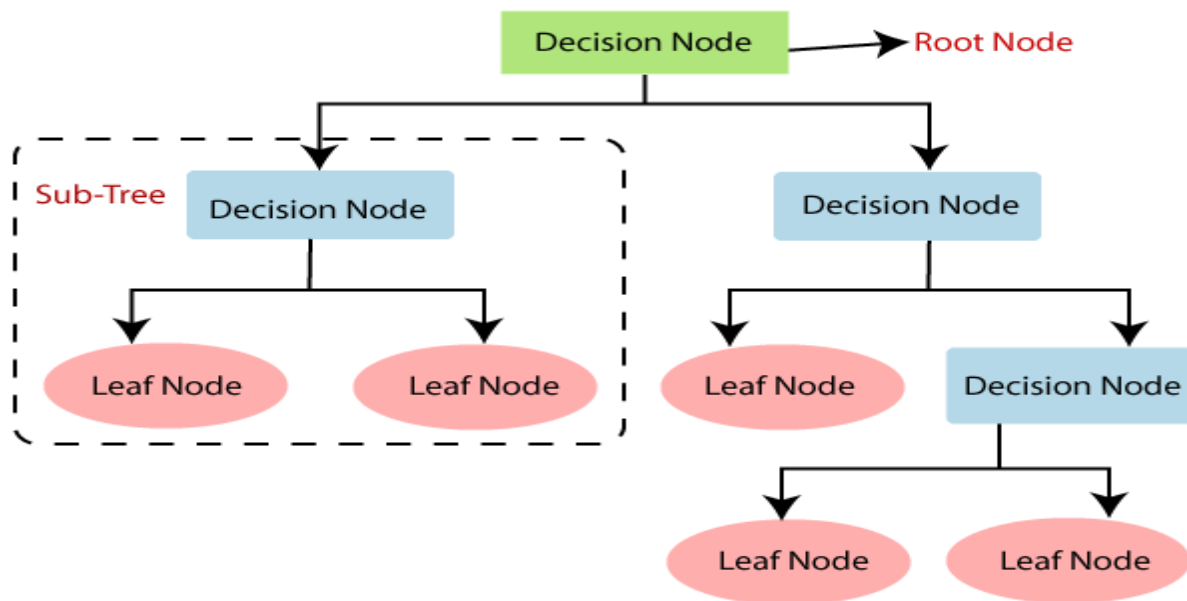


Fig 4.1 Decision Tree

4.6 Support Vector Machine

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane.

SVM can be of two types:

Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

Non-linear SVM: Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

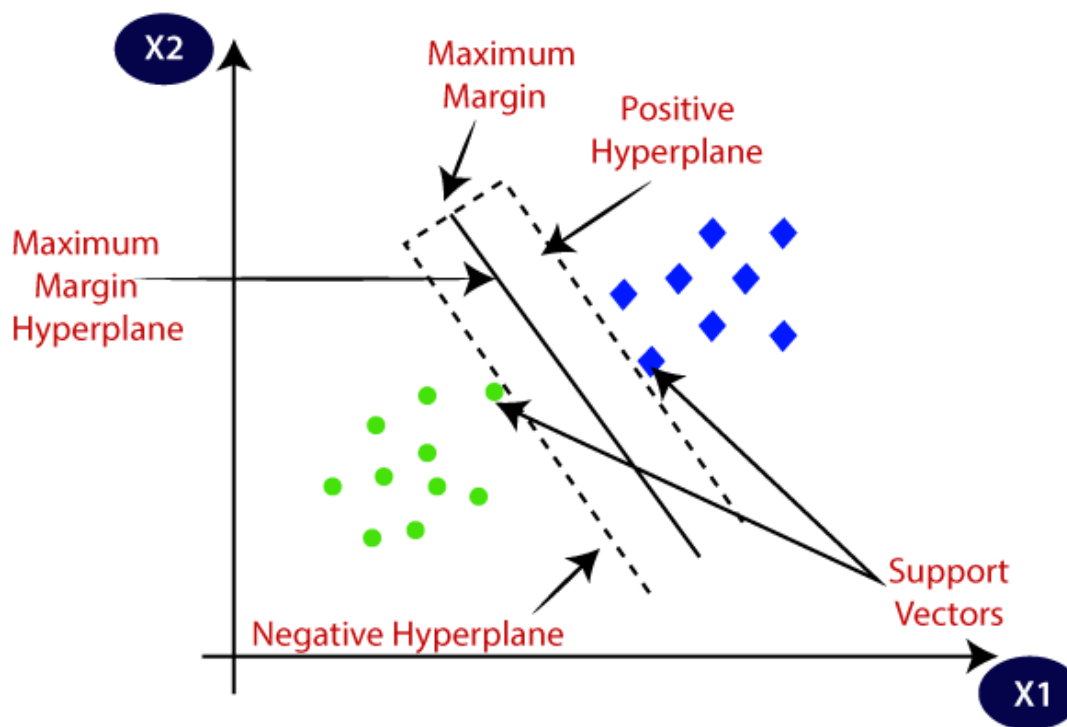


Fig 4.2 Support Vector Machine

Example: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat.

Hyperplane and Support Vectors in the SVM algorithm:

Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane. We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

Support Vectors:

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

4.7 Random Forest

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

How does Random forest work?

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Below are some points that explain why we should use the Random Forest algorithm:

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

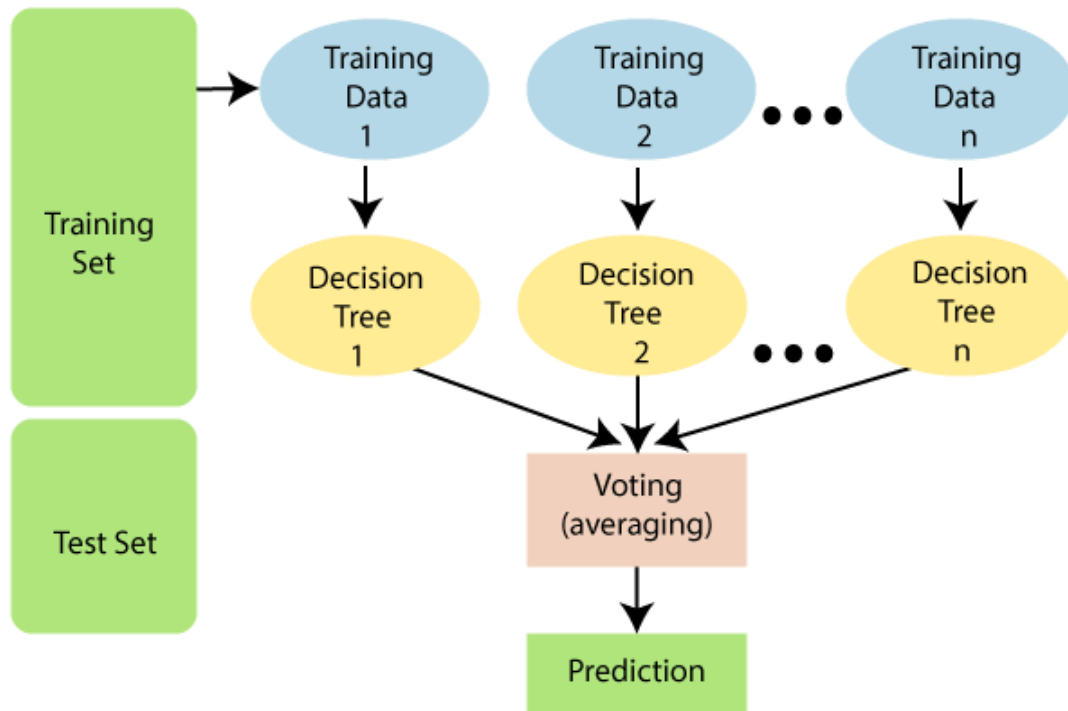


Fig 4.3 Random Forest

Applications of Random Forest

1. **Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.
2. **Medicine:** With the help of this algorithm, disease trends and risks of the disease can be identified.
3. **Land Use:** We can identify the areas of similar land use by this algorithm.
4. **Marketing:** Marketing trends can be identified using this algorithm.

5. MODULE DESIGN

5.1 Data Collection

Data Collection is one most important and crucial aspects of the Sentiment Analysis application. Due to the wide adoption of machine learning models, simply having large datasets on a domain specific task does not ensure superior performance. The performance of the model depends on the quality of dataset and labelling/annotation.

Ways to collect data for sentiment analysis:

Using API provided by social media platform which allows to collect data in a streaming fashion. Example: Twitter API to extract tweets by hashtags, NewsAPI to extract news by category from different news publishers.

Using Web scrapers that crawl up web data and collect specified information. It extracts data from webpage(HTML document). Example: Scrapy, BeautifulSoup is a Python web-scraping package to extract any information from the web like news articles or comments from blogs by parsing HTML tags.

Using a Web browser plugin with which users can extract information from any public website using HTML and export the data to the desired file format. Example: Webscraper.io is a free extension for the Google Chrome web browser.

Using existing open-source repositories of data that are cleaned and compiled which can be used directly. Example: Rotten Tomatoes, IMDB movie review, Yelp, Amazon product review, Twitter tweets on Kaggle and from other websites.

In our project, we have collected the data set from Kaggle named “METOOTWEETS.csv”. Our Data set contains 15054 rows and 8 columns. The attributes in our data set are Text Id, Length, created at, source, Favourite, count, Retweet count, Language.

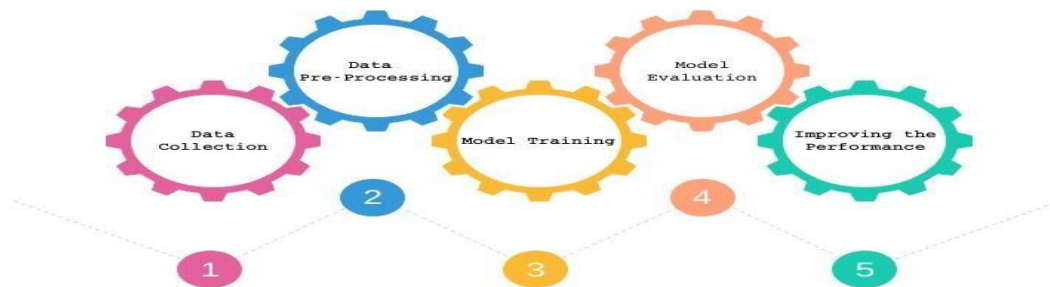


Fig 5.1 Data Collection

5.2 Data Pre-processing

Text cleaning is task-specific and one needs to have a strong idea about what they want their end result to be and even review the data to see what exactly they can achieve.

Having too many typos or spelling mistakes in the text

Having too many numbers and punctuations (E.g. Hello!!!!)

Text is full of emojis and emoticons and username and links too. (If the text is from Twitter or Facebook)

Some of the text parts are not in the English language. Data is having a mixture of more than one language

Some of the words are combined with the hyphen or data having contractions words. (E.g. text-processing)

Repetitions of words (E.g. Data)



Fig 5.2 Data preparation

Most common methods for Cleaning the Data

We will see how to code and clean the textual data for the following methods.

Lowercasing the data: The idea is to convert the input text into the same casing format so that it converts 'DATA', 'Data', 'DaTa', 'DATa' into 'data'. In some use cases, like the tokenizer and vectorization processes, the lower casing is done beforehand. But choose the lower casing precisely because if we are doing sentiment analysis on the text then if we make the text in lower case then sometimes we might miss what the word is actually stating. For example, if the word is in the upper case then it refers to anger and so on.

Removing Punctuations: The second most common text processing technique is removing punctuations from the textual data. The punctuation removal process will help to

treat each text equally. For example, the word data and data! are treated equally after the process of removal of punctuations.

We need to take care of the text while removing the punctuation because the contraction words will not have any meaning after the punctuation removal process. Such as ‘don’t’ will convert to ‘dont’ or ‘don t’ depending upon what you set in the parameter.

Removing Numbers: Sometimes number doesn’t hold any vital information in the text depending upon the use cases. So it is better to remove them than to keep them.

For example, when we are doing sentiment analysis then the number doesn’t hold any specific meaning to the data but if the task is to perform NER (Name Entity Recognition) or POS (Part of Speech tagging) then use the removing of number technique carefully.

Here, we are using the `isdigit()` function to see if the data has a number in it or not, and if we encountered the number then we are replacing the number with the blank.

Removing extra space: Well, removing the extra space is good as it doesn’t store extra memory and even we can see the data clearly.

Replacing the repetitions of punctuations: Having knowledge of regular expression will help to code faster and easier. To remove the repetition of punctuations is very helpful because it doesn’t hold any vital information if we keep more than one punctuation in the word, for example, data!!! need to convert to data.

Removing Contractions: There are so many contractions in the text we type so to expand them we will use the contractions library. The Twitter and Instagram data has so many contractions in them and if we remove the punctuations from that text then it would look like this. For example, the text “I’ll eat pizza” and if we remove the punctuations then the text will look like this “I ll will eat pizza”. Here, “I ll” doesn’t hold any information to the text that’s why we use the contraction.

5.3 Modelling

A sentiment model is composed by a collection of **entries**; entries are defined by a word or multiwords (group of words that appear together in the text). In some cases, we will want to define more complex scenarios, not just a word or multiword, so for that we will use **subentries** (which are always related to an entry). Both entries and subentries consist on their **definition** and the **sentiment behavior** associated to them.

Data handling has been done using various steps such as data collection, data preparation, modelling and evaluation.



Fig 5.3 Modelling

Take for instance the word "beaten". In general it has a negative connotation: "They were beaten badly by the other team", "He declared he was beaten with a bat", but if you were analyzing recipes, you could easily find the expression "Once the eggs are beaten...", which is not negative at all.

Another good example of the need of customized sentiment models is the variants you can find in the same language depending on the region of origin of the speaker. For instance, while "dill" is in general used as the name of an aromatic herb, in Australia and New Zealand, it's also used as an insult.

5.4 Evaluation

This step includes visualizing the results. To visualize the results of Sentiment Analysis, many people employ well-known techniques, such as graphs, histograms, and confusion matrices. Because of present multiple data domains and tasks, visualizations approaches like word cloud, pie charts, interactive maps, sparkline-style plots are also very popular.

6. SYSTEM DESIGN

6.1 System Architecture

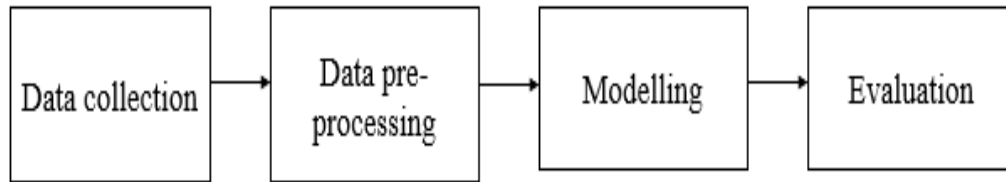


Fig 6.1 System architecture

6.2 Input and Output:

Input Design: Input design is a part of overall system design. The main objective during the input design is as given below

- To produce a cost-effective method of input.
- To achieve the highest possible level of accuracy.
- To ensure that the input is acceptable and understood by the user.

Input stages:

1.Data collection: Collecting data is the first step. Data is pulled from available sources, including data lakes and data warehouses. It is important that the data sources available are trustworthy and well- built so the data collected (and later used as information) is of the highest possible quality.

2.Data preparation: Once the data is collected, it then enters the data preparation stage. Data preparation, often referred to as “pre-processing” is the stage at which raw data is cleaned up and organized for the following stage of data processing. During preparation, raw data is diligently checked for any errors. The purpose of this step is to eliminate bad data (redundant, incomplete, or incorrect data) and begin to create high-quality data for the best business intelligence.

3.Data input: The clean data is then entered into its destination and translated into a language that it can understand. Data input is the first stage in which raw data begins to take the form of usable information.

4.Processing: During this stage, the data inputted to the computer in the previous stage is actually processed for interpretation. Processing is done using machine learning algorithms, though the process itself may vary slightly depending on the source of data being processed (data lakes, social networks, connected devices etc.) And its intended use (examining advertising patterns, medical diagnosis from connected devices, determining customer needs, etc.).

5.Data output/interpretation: The output/interpretation stage is the stage at which data is finally usable to non-data scientists. It is translated, readable, and often in the form of graphs, videos, images, plain text, etc.). Members of the company or institution can now begin to self-serve the data for their own data analytics projects.

6.Data storage: The final stage of data processing is storage. After all of the data is processed, it is then stored for future use. While some information may be put to use immediately, much of it will serve a purpose later on. Plus, properly stored data is a necessity for compliance with data protection. When data is properly stored, it can be quickly and easily accessed when needed.

Output design: Outputs from computer systems are required primarily to communicate the results of processing to users. The system is accepted by the user only by the quality of its output. Therefore, an effective output design is the major criteria for deciding the overall quality of the system.

- User's main interface with the computer.
- Output should be with relevant data.
- It must be easily understandable by the user.

Output Definition:

The outputs should be defined in terms of the following points:

- Type of the output.
- Content of the output.
- Format of the output.
- Location of the output.
- Frequency of the output.
- Sequence of the output

6.3 UML Diagrams

6.3.1 Use Case diagram

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

Use-case diagrams are helpful in the following situations:

- Before starting a project, you can create use-case diagrams to model a business so that all participants in the project share an understanding of the workers, customers, and activities of the business.
- While gathering requirements, you can create use-case diagrams to capture the system requirements and to present to others what the system should do.
- During the analysis and design phases, you can use the use cases and actors from your use case diagrams to identify the classes that the system requires.
- During the testing phase, you can use use-case diagrams to identify tests for the system.

The following topics describe model elements in use-case diagrams:

- **Usecases:** A use case describes a function that a system performs to achieve the user's goal. A use case must yield an observable result that is of value to the user of the system
- **Actors:** An actor represents a role of a user that interacts with the system that you are modeling. The user can be a human user, an organization, a machine, or another external system.
- **Subsystems:** In UML models, subsystems are a type of stereotyped component that represent independent, behavioral units in a system. Subsystems are used in class, component, and use-case diagrams to represent large-scale components in the system that you are modeling.
- **Relationships in use-case diagrams:** In UML, a relationship is a connection between model elements. A UML relationship is a type of model element that adds semantics to a model by defining the structure and behavior between the model elements.

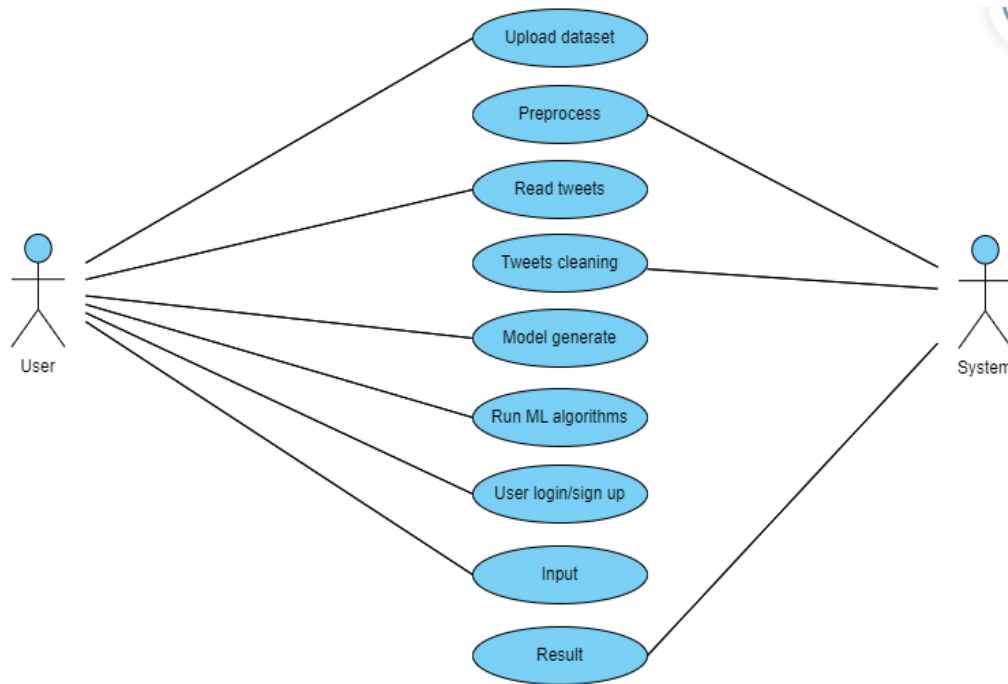


Fig 6.2 Use case diagram

6.3.2 Class Diagram

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

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object oriented systems because they are the only UML diagrams, which can be mapped directly with object oriented languages. Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

Purpose of Class Diagrams

The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus

widely used at the time of construction. UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application, however class diagram is a bit different. It is the most popular UML diagram in the coder community. The purpose of the class diagram can be summarized as –

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.

How to Draw a Class Diagram?

Class diagrams are the most popular UML diagrams used for construction of software applications. It is very important to learn the drawing procedure of class diagram. Class diagrams have a lot of properties to consider while drawing but here the diagram will be considered from a top level view. Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. A collection of class diagrams represent the whole system. The following points should be remembered while drawing a class diagram

- The name of the class diagram should be meaningful to describe the aspect of the system.
- Each element and their relationships should be identified in advance.
- Responsibility (attributes and methods) of each class should be clearly identified
- For each class, minimum number of properties should be specified, as unnecessary properties will make the diagram complicated.
- Use notes whenever required to describe some aspect of the diagram. At the end of the drawing it should be understandable to the developer/coder.
- Finally, before making the final version, the diagram should be drawn on plain paper and reworked as many times as possible to make it correct.

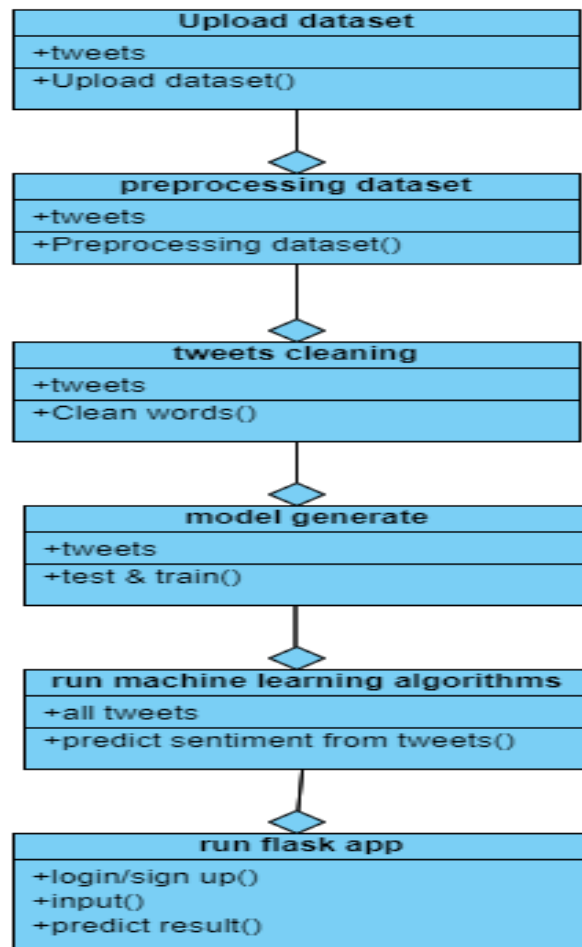


Fig 6.3 Class Diagram

6.3.3 Sequence diagram

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".

Purpose of a Sequence Diagram

1. To model high-level interaction among active objects within a system.
2. To model interaction among objects inside a collaboration realizing a use case.
3. It either models generic interactions or some certain instances of interaction.

Notations of a Sequence Diagram

Lifeline: An individual participant in the sequence diagram is represented by a lifeline. It is positioned at the top of the diagram.

Actor: A role played by an entity that interacts with the subject is called as an actor. It is out of the scope of the system. It represents the role, which involves human users and external hardware or subjects. An actor may or may not represent a physical entity, but it purely depicts the role of an entity. Several distinct roles can be played by an actor or vice versa.

Activation: It is represented by a thin rectangle on the lifeline. It describes that time period in which an operation is performed by an element, such that the top and the bottom of the rectangle is associated with the initiation and the completion time, each respectively.

Messages: The messages depict the interaction between the objects and are represented by arrows. They are in the sequential order on the lifeline. The core of the sequence diagram is formed by messages and lifelines.

Following are types of messages enlisted below:

- o **Call Message:** It defines a particular communication between the lifelines of an interaction, which represents that the target lifeline has invoked an operation.

- o **Return Message:** It defines a particular communication between the lifelines of interaction that represent the flow of information from the receiver of the corresponding caller message.

- o **Self Message:** It describes a communication, particularly between the lifelines of an interaction that represents a message of the same lifeline, has been invoked.

- o **Recursive Message:** A self message sent for recursive purpose is called a recursive message. In other words, it can be said that the recursive message is a special case of the self message as it represents the recursive calls.

- o **Create Message:** It describes a communication, particularly between the lifelines of an interaction describing that the target (lifeline) has been instantiated.

- o **Destroy Message:** It describes a communication, particularly between the lifelines of an interaction that depicts a request to destroy the lifecycle of the target.

- o **Duration Message:** It describes a communication particularly between the lifelines of an interaction, which portrays the time passage of the message while modeling a system.

Note: A note is the capability of attaching several remarks to the element. It basically carries useful information for the modelers.

Sequence Fragments

1. Sequence fragments have been introduced by UML 2.0, which makes it quite easy for the creation and maintenance of an accurate sequence diagram.
2. It is represented by a box called a combined fragment, encloses a part of interaction inside a sequence diagram.
3. The type of fragment is shown by a fragment operator.

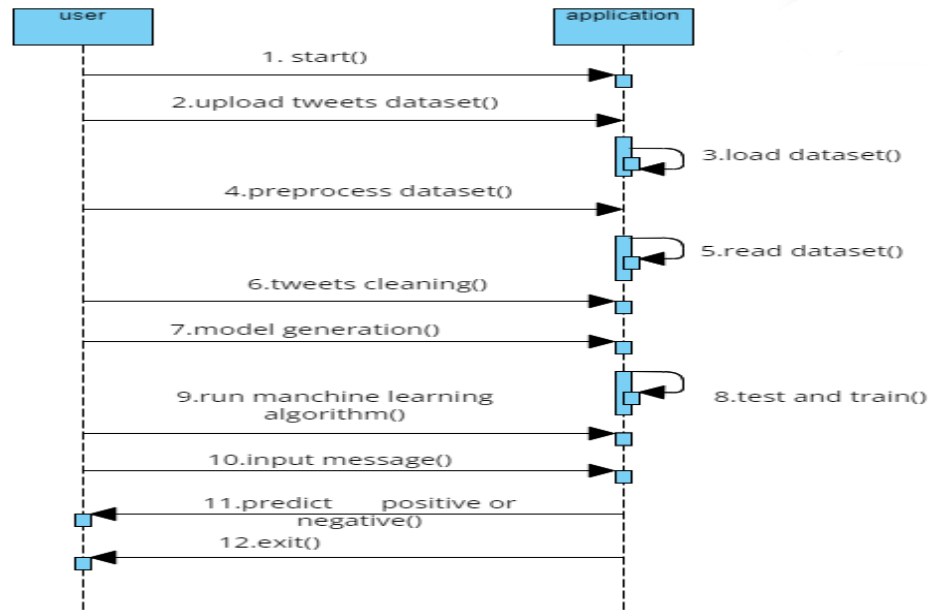


Fig 6.4 Sequence diagram

6.3.4 Activity diagram

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.

Purpose of Activity Diagrams

The basic purposes of activity diagrams is similar to other four diagrams. It captures the dynamic behavior of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another. Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the

executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.

The purpose of an activity diagram can be described as –

- Draw the activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.

How to Draw an Activity Diagram?

Activity diagrams are mainly used as a flowchart that consists of activities performed by the system.

Activity diagrams are not exactly flowcharts as they have some additional capabilities. These additional capabilities include branching, parallel flow, swim lane, etc. Before drawing an activity diagram, we must have a clear understanding about the elements used in activity diagram. The main element of an activity diagram is the activity itself. An activity is a function performed by the system. After identifying the activities, we need to understand how they are associated with constraints and conditions.

Before drawing an activity diagram, we should identify the following elements –

- Activities
- Association
- Conditions
- Constraints

Once the above-mentioned parameters are identified, we need to make a mental layout of the entire flow. This mental layout is then transformed into an activity diagram. Following is an example of an activity diagram for order management system. In the diagram, four activities are identified which are associated with conditions. One important point should be clearly understood that an activity diagram cannot be exactly matched with the code. The activity diagram is made to understand the flow of activities and is mainly used by the business users. Following diagram is drawn with the four main activities –

- Send order by the customer
- Receipt of the order

- Confirm the order
- Dispatch the order

After receiving the order request, condition checks are performed to check if it is normal or special order. After the type of order is identified, dispatch activity is performed and that is marked as the termination of the process. Where to Use Activity Diagrams? The basic usage of activity diagram is similar to other four UML diagrams. The specific usage is to model the control flow from one activity to another. This control flow does not include messages. Activity diagram is suitable for modeling the activity flow of the system. An application can have multiple systems. Activity diagram also captures these systems and describes the flow from one system to another. This specific usage is not available in other diagrams. These systems can be database, external queues, or any other system. We will now look into the practical applications of the activity diagram. From the above discussion, it is clear that an activity diagram is drawn from a very high level. So it gives high level view of a system. This high level view is mainly for business users or any other person who is not a technical person.

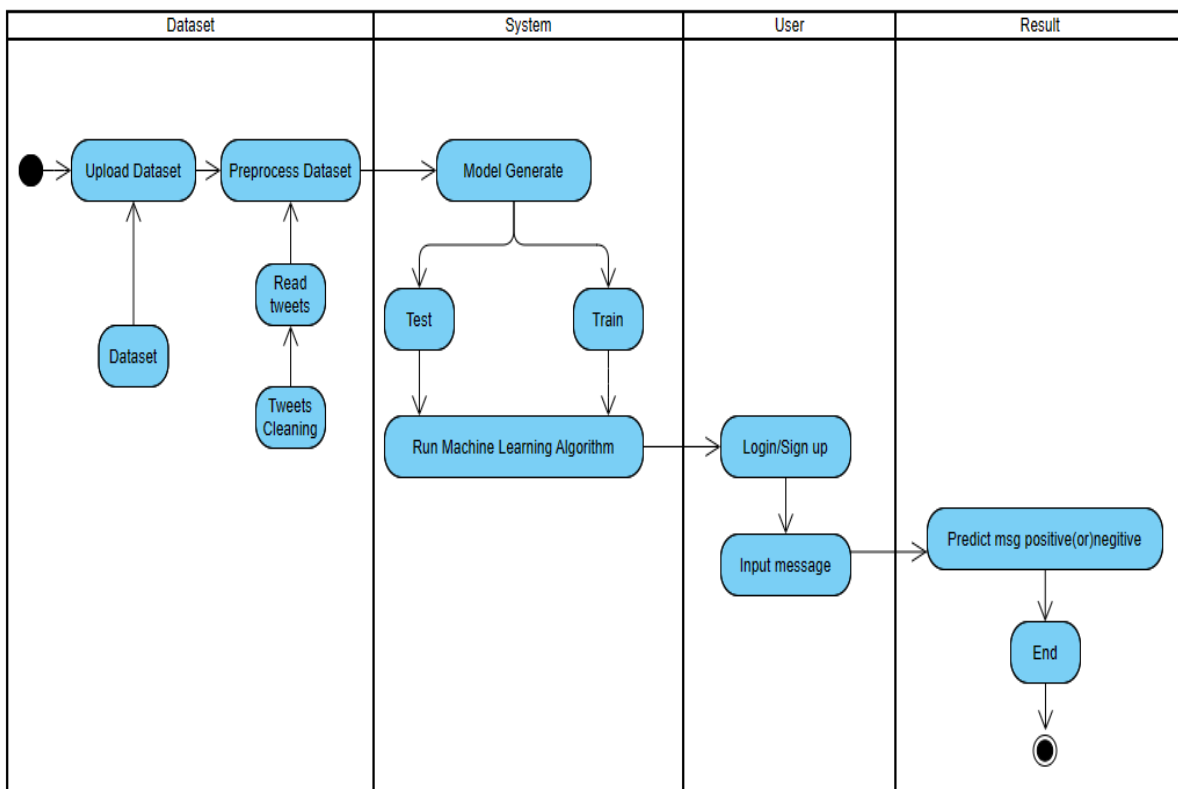


Fig 6.5 Activity diagram

7. IMPLEMENTATION

```
from textblob import TextBlob
from tkinter import *
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from string import punctuation
from nltk.corpus import stopwords
import re
import string

from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
from collections import Counter, defaultdict
import plotly.graph_objects as go
import nltk

from nltk.corpus import stopwords
from textblob import TextBlob
from tkinter import *
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from string import punctuation
from nltk.corpus import stopwords
import re
import string

from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
from collections import Counter, defaultdict
import plotly.graph_objects as go
import nltk

from nltk.corpus import stopwords
tweets_list = []
clean_list = []

train = pd.read_csv("dataset/MeToo_tweets.csv", encoding='iso-8859-1')
```

```
train
train.info()
train.isnull().sum()
train.describe()

train
alphanumeric = lambda x: re.sub("\w*\d\w*", ' ', x)
# '[%s]' % re.escape(string.punctuation), ' ' - replace punctuation with white space
# .lower() - convert all strings to lowercase
punc_lower = lambda x: re.sub('[%s]' % re.escape(string.punctuation), ' ', x.lower())
    # Remove all '\n' in the string and replace it with a space
remove_n = lambda x: re.sub("\n", " ", x)
# Remove all non-ascii characters
remove_non_ascii = lambda x: re.sub(r'[\x00-\x7f]', r' ', x)
# Apply all the lambda functions wrote previously through .map on the comments column
train["Text"] =
train["Text"].map(alphanumeric).map(punc_lower).map(remove_n).map(remove_non_
ascii)
train
def polarity(text):
    testimonial = TextBlob(text)
    polarity = testimonial.sentiment.polarity
    return polarity
def subjectivity(text):
    testimonial = TextBlob(text)
    subjectivity = testimonial.subjectivity
    return subjectivity
def senti(text, polarity_threshold=0.2):
    testimonial = TextBlob(text)
    senti = testimonial.sentiment.polarity
    if senti >= polarity_threshold:
        return 'Positive'
    elif np.abs(senti) < polarity_threshold:
```

```
        return 'Neutral'
    else:
        return 'Negative'
train['polarity'] = train['Text'].apply(lambda x: polarity(x))
train['subjectivity'] = train['Text'].apply(lambda x: subjectivity(x))
train['sentiment'] = train['Text'].apply(lambda x: senti(x))
train
train['score'] = train['sentiment'].map({'Neutral': 0, 'Positive' : 1, 'Negative' : 2})
train
from sklearn.feature_extraction.text import CountVectorizer
cv=CountVectorizer(max_features=1500)
X=cv.fit_transform(train['Text']).toarray()
y = train['score']
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size=0.20)
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
from sklearn import metrics
def evaluate(model, X_train, X_test, y_train, y_test):
    y_test_pred = model.predict(X_test)
    y_train_pred = model.predict(X_train)
    print("TRAINING RESULTS: \n=====")
clf_report=pd.DataFrame(classification_report(y_train,y_train_pred,output_dict=True))
    print(f"CONFUSION MATRIX:\n{confusion_matrix(y_train, y_train_pred)}")
    print(f"ACCURACY SCORE:\n{accuracy_score(y_train, y_train_pred):.4f}")
    print(f"CLASSIFICATION REPORT:\n{clf_report}")
    print("TESTING RESULTS: \n=====")
    clf_report=pd.DataFrame(classification_report(y_test,y_test_pred, output_dict=True))
    print(f"CONFUSION MATRIX:\n{confusion_matrix(y_test, y_test_pred)}")
    print(f"ACCURACY SCORE:\n{accuracy_score(y_test, y_test_pred):.4f}")
    print(f"CLASSIFICATION REPORT:\n{clf_report}")
    from sklearn import svm
    sv = svm.SVC()
```

```
sv.fit(x_test, y_test)
svm_score = sv.score(x_test, y_test) * 100
evaluate(sv,x_train, x_test, y_train, y_test)
from sklearn.tree import DecisionTreeClassifier
DT = DecisionTreeClassifier()
DT.fit(x_train, y_train)
dt_score = DT.score(x_test, y_test) * 100
evaluate(DT,x_train, x_test, y_train, y_test)
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(max_depth=2, random_state=0)
clf.fit(x_train, y_train)
clf_score = clf.score(x_test, y_test) * 100
evaluate(clf,x_train, x_test, y_train, y_test)
from flask import Flask,request, url_for, redirect, render_template
import pandas as pd
import pickle
import numpy as np
import sqlite3
from textblob import TextBlob
app = Flask(__name__)
tweets_list = []
clean_list = []
@app.route('/')
def hello_world():
    return render_template("home.html")
@app.route('/logon')
def logon():
    return render_template('signup.html')
@app.route('/login')
def login():
    return render_template('signin.html')
@app.route('/note')
```



```
def note():
return render_template('notebook.html')
def tweetCleaning(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()]
    stop_words = set(stopwords.words('english'))
    tokens = [w for w in tokens if not w in stop_words]
    tokens = [word for word in tokens if len(word) > 1]
    tokens = ' '.join(tokens) #here upto for word based
    return tokens
def clean():
    text.delete('1.0', END)
    clean_list.clear()
    for i in range(len(tweets_list)):
        tweet = tweets_list[i]
        tweet = tweet.strip("\n")
        tweet = tweet.strip()
        tweet = tweetCleaning(tweet.lower())
        clean_list.append(tweet)
@app.route("/signup")
def signup():
    username = request.args.get('user', "")
    name = request.args.get('name', "")
    number = request.args.get('mobile', "")
    email = request.args.get('email', "")
    password = request.args.get('password', "")
    con = sqlite3.connect('signup.db')
    cur = con.cursor()
    cur.execute("insert into `info` (`user`,`email`,`password`,`mobile`,`name`) VALUES
    (?, ?, ?, ?, ?)",(username,email,password,number,name))
```

```
con.commit()
con.close()
return render_template("signin.html")
    @app.route("/signin")
def signin():
    mail1 = request.args.get('user', '')
    password1 = request.args.get('password', '')
    con = sqlite3.connect('signup.db')
    cur = con.cursor()
    cur.execute("select `user`, `password` from info where `user` = ? AND `password` = ?",(mail1,password1,))
    data = cur.fetchone()
    if data == None:
        return render_template("signin.html")
    elif mail1 == 'admin' and password1 == 'admin':
        return render_template("index.html")
        elif mail1 == str(data[0]) and password1 == str(data[1]):
            return render_template("index.html")
    else:
        return render_template("signup.html")
@app.route('/predict',methods=['POST','GET'])
def predict():
    if request.method == 'POST':
        text = request.form['message']
        blob = TextBlob(text)
        if blob.polarity <= 0.2:
            prediction = 0
        elif blob.polarity > 0.2 and blob.polarity <= 0.5:
            prediction = 2
        elif blob.polarity > 0.5:
            prediction = 1
        if prediction == 0:
```

```
        return render_template('result.html',pred=f'Negative')
    elif prediction == 1:
        return render_template('result.html',pred=f'Positive')
    elif prediction == 2:
        return render_template('result.html',pred=f'NEUTRAL')
@app.route('/index')
def index():
    return render_template('index.html')
if __name__ == '__main__':
    app.run(debug=True)
```

8. OUTPUT SCREENS

This is the data set depicting the twitter data in which we have collected it from Kaggle.

	Text	Id	Lenght	Created_at	Source	Favorite_count	Retweet_count	Lang
0	@Rightsatbirth: "What he did was manipulative...	1.184600e+18	140	10/16/2019 22:33	Twitter for Android	0	2	en
1	Happy 94th Birthday!0x9fx8elx82 to @_AngelaL...	1.184600e+18	144	10/16/2019 22:33	Twitter for iPhone	0	0	en
2	@RepLoriTrahan: Two years ago #MeToo woke up ...	1.184600e+18	139	10/16/2019 22:33	Twitter for iPhone	0	36	en
3	@RituG15: #TimesUp #MeTooVoter #MenToo #Meto...	1.184600e+18	139	10/16/2019 22:33	Twitter for Android	0	7	en
4	@ambertamblyn: Months before MeToo broke us w...	1.184600e+18	140	10/16/2019 22:33	Twitter for iPhone	0	161	en
...
15049	advocate4victim: #DYK: October 15 is the secon...	1.184160e+18	140	10/15/2019 17:46	Twitter for Android	0	3	en
15050	TaranaBurke: Tonight, on the anniversary of #m...	1.184160e+18	140	10/15/2019 17:46	Twitter Web App	0	345	en
15051	Meet Alki David: The self-appointed ambassador...	1.184160e+18	133	10/15/2019 17:46	Twitter for iPhone	0	1	en
15052	IreneWoodbury: 2lx80lx9cGreat Read2lx806will L...	1.184160e+18	140	10/15/2019 17:46	Twitter for Android	0	17	en
15053	manny_ottawa: Canadians don2lx80lx99t TRUST Tr...	1.184160e+18	140	10/15/2019 17:46	Twitter Web App	0	65	en

15054 rows × 8 columns

Fig 8.1 Dataset

```
1 train.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15054 entries, 0 to 15053
Data columns (total 8 columns):
#   Column          Non-Null Count  Dtype
---  -
0   Text             15054 non-null  object
1   Id               15054 non-null  float64
2   Lenght           15054 non-null  int64
3   Created_at       15054 non-null  object
4   Source           15054 non-null  object
5   Favorite_count   15054 non-null  int64
6   Retweet_count    15054 non-null  int64
7   Lang             15054 non-null  object
dtypes: float64(1), int64(3), object(4)
memory usage: 941.0+ KB
```

```
1 train.isnull().sum()

Text             0
Id               0
Lenght           0
Created_at       0
Source           0
Favorite_count   0
Retweet_count    0
Lang             0
```

Fig 8.2 Data Cleaning

1	train											
		Text	Id	Lenght	Created_at	Source	Favorite_count	Retweet_count	Lang	polarity	subjectivity	sentiment
0	rights	atbirth what he did was manipulative...	1.184600e+18	140	10/16/2019 22:33	Twitter for Android	0	2	en	0.10	0.666667	Neutral
1		happy birthday to angelalansbury ...	1.184600e+18	144	10/16/2019 22:33	Twitter for iPhone	0	0	en	0.65	1.000000	Positive
2	replori	trahan two years ago metoo woke up ...	1.184600e+18	139	10/16/2019 22:33	Twitter for iPhone	0	36	en	0.50	0.500000	Positive
3	timesup	metoo voter mentoo metoo bre...	1.184600e+18	139	10/16/2019 22:33	Twitter for Android	0	7	en	0.00	0.000000	Neutral
4	ambert	amblyn months before metoo broke us w...	1.184600e+18	140	10/16/2019 22:33	Twitter for iPhone	0	161	en	-0.05	0.450000	Neutral
...	
15049	dyk	october is the second anniversary o...	1.184160e+18	140	10/15/2019 17:46	Twitter for Android	0	3	en	0.00	0.000000	Neutral
15050	taran	burke tonight on the anniversary of m...	1.184160e+18	140	10/15/2019 17:46	Twitter Web App	0	345	en	0.00	0.000000	Neutral
15051	meet	alki david the self appointed ambassador...	1.184160e+18	133	10/15/2019 17:46	Twitter for iPhone	0	1	en	0.00	0.000000	Neutral
15052	irenewoodbury	leave you wanting mor...	1.184160e+18	140	10/15/2019 17:46	Twitter for Android	0	17	en	0.50	0.500000	Positive
15053	manny	ottawa canadians trust trudeau ...	1.184160e+18	140	10/15/2019 17:46	Twitter Web App	0	65	en	0.00	0.000000	Neutral
15054

Fig 8.3 Dataset obtained after applying polarity, subjectivity and sentiment

```

1 evaluate(sv,x_train, x_test, y_train, y_test)

TRAINING RESULTS:
=====
CONFUSION MATRIX:
[[7502  96   0]
 [ 872 2828   0]
 [ 480   8 257]]
ACCURACY SCORE:
0.8791
CLASSIFICATION REPORT:
              0              1              2  accuracy  macro avg  \
precision    0.847301    0.964529    1.000000    0.8791    0.937277
recall       0.987365    0.764324    0.344966    0.8791    0.698885
f1-score     0.911986    0.852835    0.512974    0.8791    0.759265
support      7598.000000  3700.000000  745.000000    0.8791  12043.000000

              weighted avg
precision    0.892763
recall       0.879100
f1-score     0.869130
support      12043.000000

TESTING RESULTS:
=====
CONFUSION MATRIX:
[[1916   5   0]
 [ 105  813   0]
 [  91   1  80]]
ACCURACY SCORE:
0.9329
CLASSIFICATION REPORT:
              0              1              2  accuracy  macro avg  \
precision    0.907197    0.992674    1.000000    0.932913    0.966624
recall       0.997397    0.885621    0.465116    0.932913    0.782711
f1-score     0.950161    0.936097    0.634921    0.932913    0.840393
support      1921.000000  918.000000  172.000000    0.932913  3011.000000

              weighted avg
precision    0.938559
recall       0.932913
f1-score     0.927865
support      3011.000000

```

Fig 8.4 Model building using Support Vector Machine(SVM)

```

1 evaluate(DT,x_train, x_test, y_train, y_test)

TRAINING RESULTS:
=====
CONFUSION MATRIX:
[[7598    0    0]
 [   2 3698    0]
 [   1    0  744]]
ACCURACY SCORE:
0.9998
CLASSIFICATION REPORT:
              0              1              2  accuracy  macro avg \
precision    0.999605    1.000000    1.000000  0.999751    0.999868
recall       1.000000    0.999459    0.998658  0.999751    0.999372
f1-score     0.999803    0.999730    0.999328  0.999751    0.999620
support      7598.000000  3700.000000  745.000000  0.999751  12043.000000

              weighted avg
precision    0.999751
recall       0.999751
f1-score     0.999751
support      12043.000000

TESTING RESULTS:
=====
CONFUSION MATRIX:
[[1802   85   34]
 [   82  829    7]
 [   46   16  110]]
ACCURACY SCORE:
0.9103
CLASSIFICATION REPORT:
              0              1              2  accuracy  macro avg \
precision    0.933679    0.891398    0.728477  0.910329    0.851184
recall       0.938053    0.903050    0.639535  0.910329    0.826879
f1-score     0.935861    0.897186    0.681115  0.910329    0.838054
support      1921.000000  918.000000  172.000000  0.910329  3011.000000

              weighted avg
precision    0.909066
recall       0.910329
f1-score     0.909518
support      3011.000000

```

Fig 8.5 Model building using Decision Tree


```
1 evaluate(clf,x_train, x_test, y_train, y_test)
```

TRAINING RESULTS:

=====

CONFUSION MATRIX:

```
[[7598  0  0]
 [3268 432  0]
 [ 745  0  0]]
```

ACCURACY SCORE:

0.6668

CLASSIFICATION REPORT:

	0	1	2	accuracy	macro avg \
precision	0.654379	1.000000	0.0	0.666777	0.551460
recall	1.000000	0.116757	0.0	0.666777	0.372252
f1-score	0.791088	0.209100	0.0	0.666777	0.333396
support	7598.000000	3700.000000	745.0	0.666777	12043.000000

	weighted avg
precision	0.720084
recall	0.666777
f1-score	0.563344
support	12043.000000

=====

CONFUSION MATRIX:

```
[[1921  0  0]
 [ 815 103  0]
 [ 172  0  0]]
```

ACCURACY SCORE:

0.6722

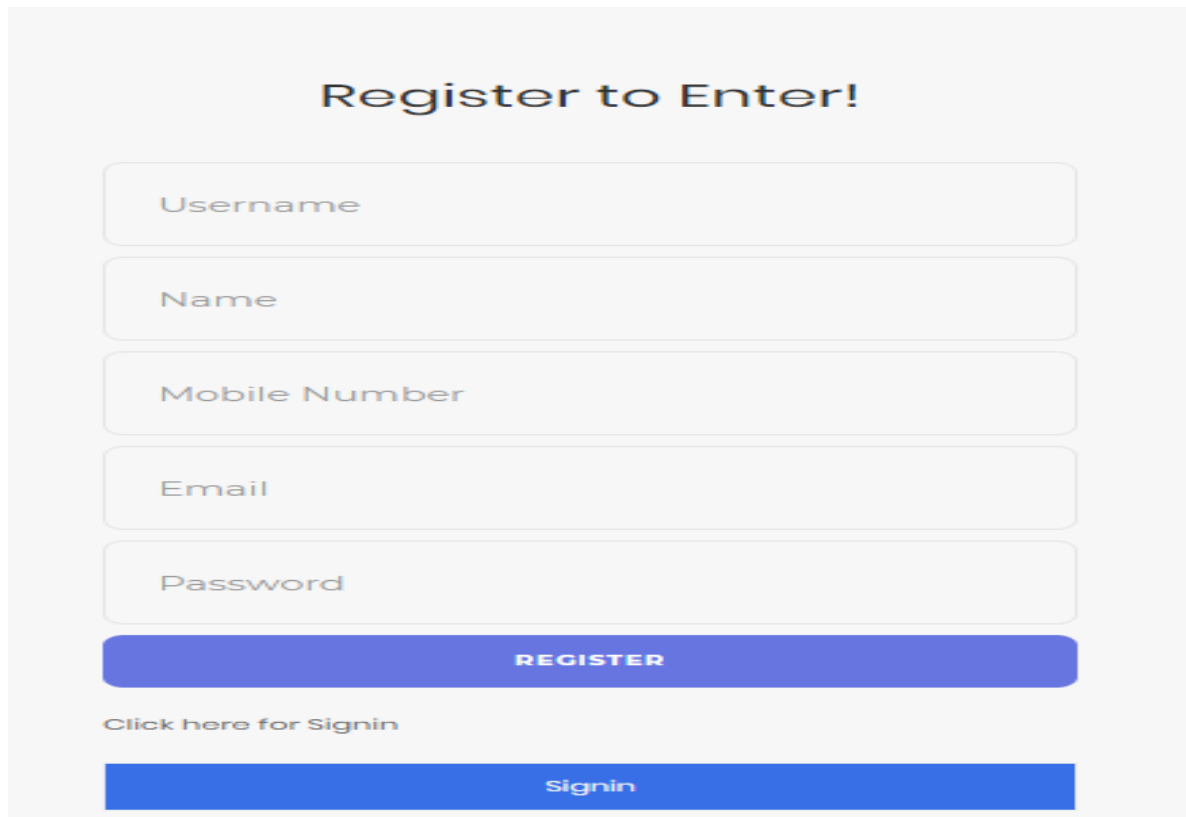
CLASSIFICATION REPORT:

	0	1	2	accuracy	macro avg	weighted avg
precision	0.660591	1.000000	0.0	0.672202	0.553530	0.726336
recall	1.000000	0.112200	0.0	0.672202	0.370733	0.672202
f1-score	0.795610	0.201763	0.0	0.672202	0.332458	0.569108
support	1921.000000	918.000000	172.0	0.672202	3011.000000	3011.000000

Fig 8.6 Model Building using Random Forest (RF)



Fig 8.7 Home page



Register to Enter!

Username

Name

Mobile Number

Email

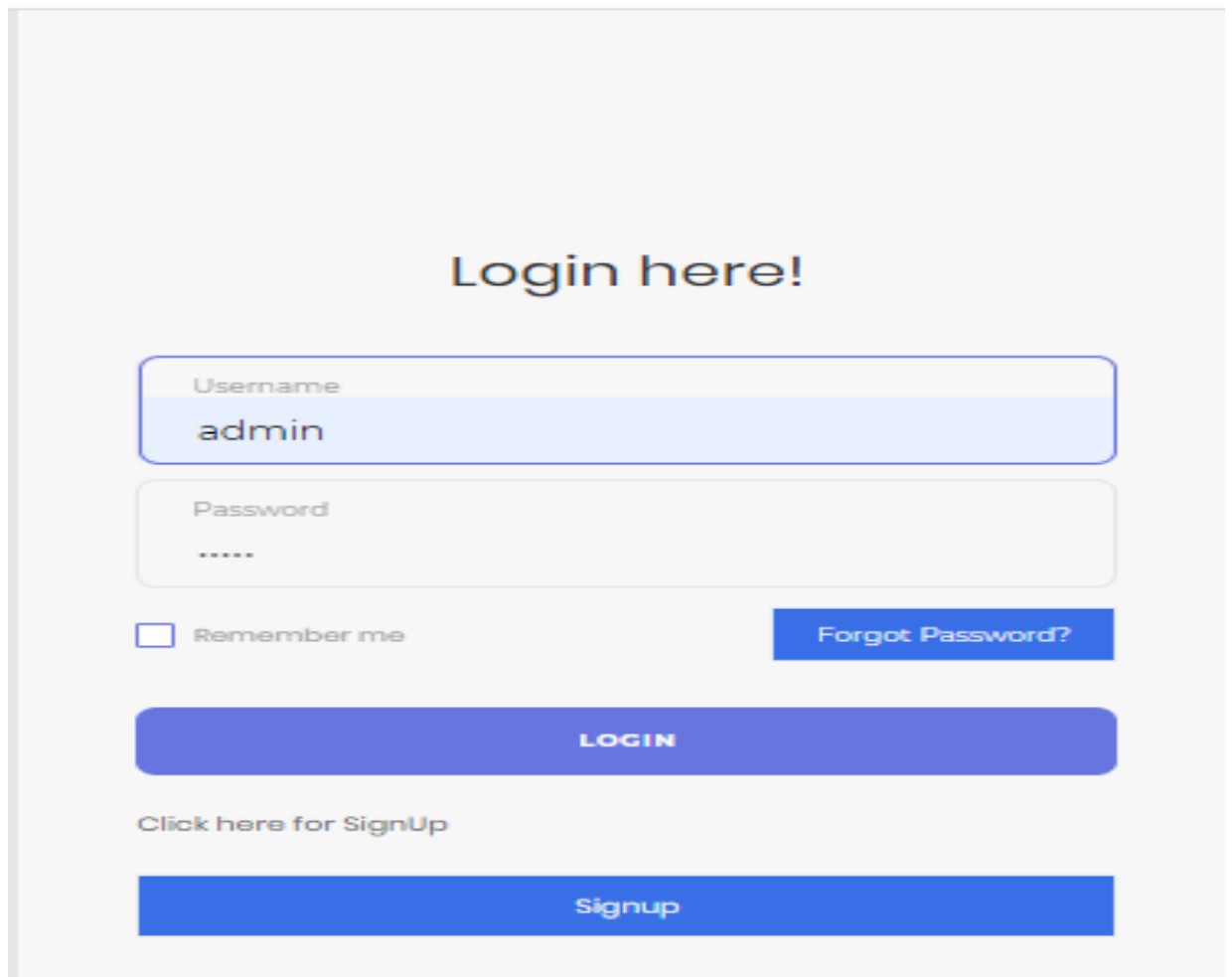
Password

REGISTER

[Click here for Signin](#)

Signin

Fig 8.8 Signup page



The image shows a login page with a light gray background. At the top center, the text "Login here!" is displayed in a large, bold, black font. Below this, there are two input fields: "Username" and "Password". The "Username" field contains the text "admin". The "Password" field contains six dots. To the left of the "Password" field, there is a checkbox labeled "Remember me". To the right of the "Password" field, there is a blue button labeled "Forgot Password?". Below these fields, there is a large blue button labeled "LOGIN". Below the "LOGIN" button, there is a link that says "Click here for SignUp". At the bottom, there is a blue button labeled "Signup".

Login here!

Username
admin

Password

☐ Remember me


Forgot Password?

LOGIN

Click here for SignUp

Signup

Fig 8.9 Login Page



Enter Your Message Here

A women got harassed by a man

SUBMIT

Fig 8.10 User Input



Fig 8.11 Predict Whether message is positive or Negative

9. TESTING

9.1 Introduction

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

Steps in testing

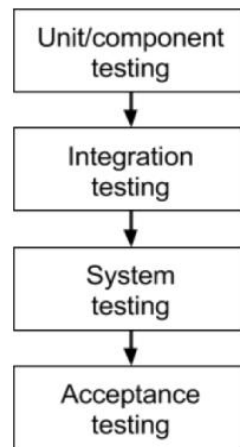


Fig 9.1 Types of Testing

9.2 TYPES OF TESTING

1. Unit Testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process

performs accurately to the documented specifications and contains clearly defined inputs and expected results.

2.Integration Testing:

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successful unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

3.Functional Testing:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

Valid Input: identified classes of valid input must be accepted. **Invalid Input:** identified classes of invalid input must be rejected.

Functions: identified functions must be exercised.

Output: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

4.System Testing:

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

5.White Box Testing:

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

6.Black Box Testing:

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot—see into it. The test provides inputs and responds to outputs without considering how the software works.

9.3 Test Cases:

1.Unit testing: Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases. Test strategy and approach: Field testing will be performed manually and functional tests will be written in detail.

Test objectives:

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be verified:

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

2.Integration Testing: Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g., components in a software system or – one step up – software applications at the company level – interact without error. Test Results: All the test cases mentioned above passed successfully. No defects encountered.

3.Acceptance Testing: User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements. Test Cases: All the test cases mentioned above passed successfully.

S.NO	INPUT	OUTPUT	RESULT
Test Case 1 (Unit testing of Dataset)	The user gives the input in the form of dataset.	An output is whether given input message is positive or negative.	An output is whether given input message is positive or negative.
Test Case 2 (Unit testing of Accuracy)	The user gives the input in the form of dataset.	An output is whether given input message is positive or negative.	A result is test data using machine learning algorithm
Test Case 3 (Unit testing of Machine Learning Algorithms)	The user gives the input in the form of dataset.	An output is whether given input message is positive or negative..	A result is test data using machine learning algorithm to predict sentiments from tweets.
Test Case 4 (Integration testing of Dataset)	The user gives the input in the form of dataset.	An output is whether given input message is positive or negative.	An output is whether given input message is positive or negative.
Test Case 5 (Big Bang testing)	The user gives the input in the form of dataset.	An output is whether given input message is positive or negative.	An output is whether given input message is positive or negative.

Test Case 6 (Data Flow Testing)	The user gives the input in the form of dataset.	An output is whether given input message is positive or negative.	A result is test data using machine learning algorithm to predict sentiments from tweets
Test Case 7 (User interface Testing)	The user gives the input in the form of dataset.	An output is whether given input message is positive or negative.	An output is whether given input message is positive or negative.
Test Case 8 (User interface Testing-Event based)	The user gives the input in the form of dataset. .	An output is whether given input message is positive or negative.	A result is test data using machine learning algorithm to predict sentiments from tweets

10. CONCLUSION AND FUTURE WORK

Now a days women safety is the most concerned factor in various cities. Women have been facing a lot of problems through social media also so this “Analysis of women safety in Indian cities using machine learning on tweets” predicts whether the given message is positive or negative and also gives a comparative analysis of machine learning algorithms such as Decision Tree, Support Vector Machine, Random Forest and decision tree gives the best accuracy of 91%.

Further it can be extended using deep learning techniques which gives better results and can also be extended for other social media platforms such as Facebook and Instagram. Present ideology which is proposed can be integrated with the twitter application interface to reach larger extent and apply sentimental analysis on millions of tweet to provide more safety.

11. REFERENCES

- [1] Agarwal, Apoorv, Fadi Biadisy, and Kathleen R. Mckeown. "Contextual phrase-level polarity analysis using lexical affect scoring and syntactic n-grams." Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, 2009.
- [2] Barbosa, Luciano, and Junlan Feng. "Robust sentiment detection on twitter from biased and noisy data." Proceedings of the 23rd international conference on computational linguistics: posters. Association for Computational Linguistics, 2010.
- [3] Bermingham, Adam, and Alan F. Smeaton. "Classifying sentiment in microblogs: is brevity an advantage?." Proceedings of the 19th ACM international conference on Information and knowledge management. ACM, 2010.
- [4] Gamon, Michael. "Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis." Proceedings of the 20th international conference on Computational Linguistics. Association for Computational Linguistics, 2004.
- [5] Kim, Soo-Min, and Eduard Hovy. "Determining the sentiment of opinions." Proceedings of the 20th international conference on Computational Linguistics. Association for Computational Linguistics, 2004.
- [6] Divya Chitkara, Nipun Sachideva, Yash Dev Vashist, "Design of a Women Safety Device".
2016 IEEE Region 10 Humanitarian Technology Conference.
- [7] Charniak, Eugene, and Mark Johnson. "Coarse-to-fine n-best parsing and MaxEnt discriminative reranking." Proceedings of the 43rd annual meeting on association for computational linguistics. Association for Computational Linguistics, 2005.
- [8] Sahayak, V., Shete, V., & Pathan, A. (2015). Sentiment analysis on twitter data. International Journal of Innovative Research in Advanced Engineering (IJIRAE), 2(1), 178-183.
- [9] Mamgain, N., Mehta, E., Mittal, A., & Bhatt, G. (2016, March). Sentiment analysis of top colleges in India using Twitter data. In Computational Techniques in Information and Communication Technologies (ICCTICT), 2016 International Conference on (pp. 525-530). IEEE.

