**AUTOMATIC HATE SPEECH DETECTION ON SOCIAL MEDIA**

**A project report submitted in the partial fulfillment of the requirements for the**

**Award of the Degree of**

**MASTER OF**

**COMPUTER SCIENCE AND APPLICATION**

Submitted By

**vanga aparna**

**Reg.No: 113519862003**

1Under the guidance of

Mr. Naveen Kumar

**Assistant Professor**

****

**DEPARTMENT OF MASTER OF COMPUTER SCIENCE AND APPLICATION**

**NANAKRAM BHAGWANDAS SCIENCE COLLEGE**

**2019-2022**

**NANAKRAM BHAGWANDAS SCIENCE COLLEGE**

**CERTIFICATE**

This is to certify that this project work entitled “ **AUTOMATIC HATE SPEECH DETECTIO ON SOCIAL MEDIA ”** is the bonafide work carried out by **Vanga Aparna , Reg.No: 113519A862003** submitted in Partial fulfillment of the requirement for the Award of Degree of **Master of Computer Application**, during the acadamic year 2019-2022.

The results submitted in this project have been verified and are found to be satisfactory.

**Signature of Project Guide Signature of Head of the Department**

**Signature of External Examiner**

**ACKNOWLEDGEMENT**

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crowned the efforts with success. It is a pleasant aspect that I have now the opportunity to express my gratitude for all of them.

The first person I would like to thank is my project guide **Mr. Naveen Kumar, Assistant Professor**, Department of MCA , had given continuous critical suggestions and extension of proper working atmosphere, abiding interest has finally evolved into this research work.

It is indeed with a great sense of pleasure and immense sense of guidance that I acknowledge the help and I am highly indebted to our **HOD Smt. KL Madhuri** provided to accomplish this work.

I owe special thanks to Departmental committee for the successful completion of this project. I am grateful to all staff members, Department of MCA, for their valuable suggestions and encouragement.

**VANGA APARNA**

**11351986200**



Date: 5 - 0 6 - 2 0 2 2

**Certificate**

This is to certify that **Ms: VANGA APARNA (113519862003)**

final year M.C.A student of **Nanakram Bhagwandas Science College**

**and PG Center** completed her project in **1000PROJECTS IT**

**Technologies PVT LTD** on **“Automatic Hate Speech Detection on**

**Social Media”** under guidance and supervision of technical team for the

award of the degree of Master of Computer Applications



**ABSTRACT**

The increasing use of social media and information sharing has given major benefits to humanity. However, this has also given rise to a variety of challenges including the spreading and sharing of hate speech messages. Thus, to solve this emerging issue in social media sites, recent studies employed a variety of feature engineering techniques and machine learning algorithms to automatically detect the hate speech messages on different datasets. However, to the best of our knowledge, there is no study to compare the variety of feature engineering techniques and machine learning algorithms to evaluate which feature engineering technique and machine learning algorithm outperform on a standard publicly available dataset. Hence, the aim of this paper is to compare the performance of three feature engineering techniques and eight machine learning algorithms to evaluate their performance on a publicly available dataset having three distinct classes. The experimental results showed that the bigram features when used with the support vector machine algorithm best performed with 79% off overall accuracy. Our study holds practical implication and can be used as a baseline study in the area of detecting automatic hate speech messages. Moreover, the output of different comparisons will be used as state-of-art techniques to compare future researches for existing automated text classification techniques.

**TABLE OF CONTENTS**

#### INTRODUCTION

* 1. [MACHINE LEARNING](#_TOC_250029) **[1](#_TOC_250029)**
  2. [PROBLEM DEFINITION](#_TOC_250028) **[2](#_TOC_250028)**
  3. [PROJECT PURPOSE](#_TOC_250027) **[2](#_TOC_250027)**
  4. [PROJECT FEATURES](#_TOC_250026) **[3](#_TOC_250026)**
  5. [MODULES DESCRIPTION](#_TOC_250025) **[3](#_TOC_250025)**

1. LITERATURE SURVEY
   1. [MACHINE LEARNING](#_TOC_250024) **[6](#_TOC_250024)**
   2. [SOFTWARE DESCRIPTION](#_TOC_250023) **[7](#_TOC_250023)**
2. REQUIREMENT ANALYSIS
   1. [FUNCTIONAL REQUIREMENTS](#_TOC_250022) **[9](#_TOC_250022)**
   2. [NON FUNCTIONAL REQUIREMENTS](#_TOC_250021) **[10](#_TOC_250021)**
   3. [HARDWARE REQUIREMENTS](#_TOC_250020) **[12](#_TOC_250020)**
   4. [SOFTWARE REQUIREMENTS](#_TOC_250019) **[12](#_TOC_250019)**
3. DESIGN
   1. [DESIGN GOALS](#_TOC_250018) **[13](#_TOC_250018)**
   2. [DATA FLOW DIAGRAM](#_TOC_250017) **[15](#_TOC_250017)**
   3. SEQUENCE DIAGRAM **16**
4. IMPLEMENTATION
   1. [META DATA GENERATION](#_TOC_250016) **[18](#_TOC_250016)**
   2. CREATION OF PROCESSED TWEETS AND FEATURE VECTOR

19

* 1. [USE EXTRACT FEATURES TO GET FEATURES](#_TOC_250015) **[20](#_TOC_250015)**
  2. [APPLY FEATURES TO GET TRAINING SET](#_TOC_250014) **[21](#_TOC_250014)**
  3. [USE TRAINING SET TO TRAIN CLASSIFIER](#_TOC_250013) **[21](#_TOC_250013)**
  4. [TRAINED CLASSIFIER IS STORED IN PICKLE](#_TOC_250012) **[21](#_TOC_250012)**
  5. [USER INPUT SENTENCE TO CLASSIFY](#_TOC_250011) **[21](#_TOC_250011)**
  6. [USER INPUT TOPIC TO FETCH TWEET AND CLASSIFY](#_TOC_250010) **[22](#_TOC_250010)**
  7. [CALCULATING CONFIDENCE](#_TOC_250009) **[22](#_TOC_250009)**
  8. [STORING MORE TRAINING DATA](#_TOC_250008) **[23](#_TOC_250008)**

1. TESTING
   1. [UNIT TESTING](#_TOC_250007) **[24](#_TOC_250007)**
   2. [INTEGRATION TESTING](#_TOC_250006) **[24](#_TOC_250006)**
   3. [VALIDATION TESTING](#_TOC_250005) **[25](#_TOC_250005)**
   4. [SYSTEM TESTING](#_TOC_250004) **[26](#_TOC_250004)**
   5. [TESTING OF INITIALIZATION AND UICOMPONENTS 27](#_TOC_250003)
2. SNAPSHOT 32
3. CONCLUSION AND FUTURE ENHANCEMENT
   1. [CONCLUSION](#_TOC_250002) **[47](#_TOC_250002)**
   2. [FUTURE ENHANCEMENT](#_TOC_250001) **[48](#_TOC_250001)**

[BIBLIOGRAPHY 49](#_TOC_250000)

**LIST OF FIGURE**

|  |  |  |
| --- | --- | --- |
| S NO. | NAME | PAGE NO. |
| 1 | System Model | 13 |
| 2 | Data flow Diagram | 15 |
| 3 | Sequence Diagram for Twitter streaming | 16 |
| 4 | Implementation steps of Hate Speech Prediction analysis | 17 |
| 5 | CSV file | 18 |
| 6 | Testing Process | 25 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| S NO. | NAME | PAGE NO |
| 1 | Test case for train option | 27 |
| 2 | Test case for test classifier option | 28 |
| 3 | Test case for try option | 29 |
| 4 | Test case for tweet option | 30 |
| 5 | Test case for store option | 31 |

# 1.INTRODUCTION

## INTRODUCTION

In recent years, hate speech has been increasing in-person and online communication. The social media as well as other online platforms are playing an extensive role in the breeding and spread of hateful content – eventually which leads to hate crime. For example, according to recent surveys, the rise in online hate speech content has resulted in hate crimes including Trump's election in the US, the Manchester and London attacks in the UK, and terror attacks in New Zealand. To tackle these harmful consequences of hate speech, different steps including legislation have been taken by the European Union Commission. Recently, the European Union Commission also enforced social media networks to sign an EU hate speech code to remove hate speech content within 24 hours. However, the manual process to identify and remove hate speech content is labor-intensive and time consuming. Due to these concerns and widespread hate speech content on the internet, there is a strong motivation for automatic hate speech detection. The automatic detection of hate speech is a challenging task due to disagreements on different hate speech definitions. Therefore, some content might be hateful to some individuals and not to others, based on their concerned definitions. According to hate speech is: “the content that promotes violence against individuals or groups based on race or ethnic origin, religion, disability, gender, age, veteran status, and sexual orientation/gender identity”. Despite these different definitions, some recent studies claimed favorable results to detect automatic hate speech in the text. The proposed solutions employed the different feature engineering techniques and ML algorithms to classify content as hate speech. Regardless of this extensive amount of work, it remains difficult to compare the performance of these approaches to classify hate speech content. To the best of our knowledge, the existing studies lack the comparative analysis of different feature engineering techniques and ML algorithms. Therefore, this study contributes to solving this problem by comparing three feature engineering and eight ML classifiers on standard hate speech datasets. Table I shows major concepts related to automatic text classification along with their explanations and references. This study holds practical importance and served as a reference for new researchers in the domain of automatic hate speech detection.

**MACHINE LEARNING**

*“Optimizing a performance criterion using example data and past experience”* gives an easy but faithful description about machine learning.

In machine learning, data plays an indispensable role, and the learning algorithm is used to discover and learn knowledge or properties from the data. The quality or quantity of the data set will affect the learning and prediction performance. We might, for instance, be interested in learning to complete a task, make accurate predictions or behave intelligently. The learning that is being done is always based on some sort of observations or data, such as examples, direct experience, or instruction. So in general, machine learning is about learning to do better in the future based on what was experienced in the past. The emphasis of machine learning is to automate methods. In other words, the goal is to devise learning algorithms that do the learning automatically without human intervention or assistance.

In machine learning, we seek methods by which the computer will come up with its own program based on examples that we provide. Machine learning is a core subarea of artificial intelligence. It is very unlikely that we will be able to build any kind of intelligent system capable of any of the facilities that we associate with intelligence, such as language or vision, without using machine learning. These tasks are otherwise simply too difficult to solve. Further, we would not consider a system to be truly intelligent if it were incapable of learning since learning is the core of intelligence.

## PROBLEM DEFINITION

In recent years, hate speech has been increasing in-person and online communication. The social media as well as other online platforms are playing an extensive role in the breeding and spread of hateful content – eventually which leads to hate crime. For example, according to recent surveys, the rise in online hate speech content has resulted in hate crimes including Trump's election in the US, the Manchester and London attacks in the UK, and terror attacks in New Zealand. To tackle these harmful consequences of hate speech, different steps including legislation have been taken by the European Union Commission. Recently, the European Union Commission also enforced social media networks to sign an EU hate speech code to remove hate speech content.

## PROJECT PURPOSE

The automatic detection of hate speech is a challenging task due to disagreements on different hate speech definitions. Therefore, some content might be hateful to some individuals and not to others, based on their concerned definitions. According to, hate speech is: “the content that promotes violence against individuals or groups based on race or ethnic origin, religion, disability, gender, age, veteran status, and sexual orientation/gender identity”. Despite these different definitions, some recent studies claimed favorable results to detect automatic hate speech in the text. The proposed solutions employed the different feature engineering techniques and ML algorithms to classify content as hate speech. Regardless of this extensive amount of work, it remains difficult to compare the performance of these approaches to classify hate speech content. To the best of our knowledge, the existing studies lack the comparative analysis of different feature engineering techniques and ML algorithms.

## PROJECT FEATURES

Our project was developed to train a machine about Hate Speech Prediction using machine learning algorithm and later use the trained machine to classify the Hate Speech Prediction of tweets. In our consideration classifier is our machine that will classify sentences or tweets based on Hate Speech Prediction. We took a good amount of data to train classifier for better accuracy. It takes huge amount of time to train classifier with good amount of data. We reduced training time by pre processing the training data set which removes unwanted words and symbols. The prE processing generates list of important words called feature vector that affects Hate Speech Prediction.

Reducing the training time was not enough for our project. We also required best accuracy possible. For better accuracy we solved some of the problem such as negation problem. Negation problem is about Hate Speech Prediction of negation sentence having negation words like not, don’t, neither in the sentence. To be more confident on Hate Speech Prediction of tweet or sentence, we used machine learning algorithm to classify Hate Speech Prediction. These algorithms were used to give the confidence about the outcome. We called it confidence level.

**Existing system:**

* Existing system employed a dictionary-based approach to identify cyber hate on Twitter. In this research, they employed an N-gram feature engineering technique to generate the numeric vectors from the predefined dictionary of hateful words.
* also used a dictionary based approach for the automatic detection of racism in Dutch Social Media. In this study, the authors used the distribution of words over three dictionaries as features. They fed the generated features to the SVM classifier. Their experimental results obtained 0.46 F-Score.

**Disadvantages:**

* However, the major disadvantage of such type of approach is that it requires a dictionary, based on the large corpus to look for domain words. To overcome this drawback, many of the researchers have used a BOW-based approach which is similar to a dictionary-based approach but the word features are obtained from training data and not from the predefined dictionaries.
* However, the major disadvantage of this technique is, the word-order is ignored and causes miss classification as different words are used in different contexts. To overcome this limitation, researchers have proposed an N-grams-based approach

**Proposed system:**

* The proposed solutions employed the different feature engineering techniques and ML algorithms to classify content as hate speech and predict given text as hate speech or not. Training , testing on data set and accuracy calculation is done by algorithms.

**Advantages:**

* Compare the variety of feature engineering techniques and machine learning algorithms to evaluate which feature engineering technique and machine learning algorithm outperform on a standard publicly available data set.

## MODULES DESCRIPTION

### PROCESS TWEET

Pre processing is done to remove all unwanted characters, which does not lead us to determine the Hate Speech Prediction of the tweet. Some of the actions performed in pre processing module are:

Lower cases: We converted the tweets to lowercase.

URL’s: We eliminate the URL’s via regular expressions and replace it with the word URL.

@username: We eliminate “@username” and replace it with the word “AT\_USER”.

#hashtag: We eliminate “#tag word” and replace it with the word “tag\_word”. Additional whitespaces: We eliminate all the additional white spaces.

### GET FEATUREVECTOR

This function takes processed tweets and gives list of meaning words that has some Hate Speech Prediction called as feature vector. It refines processed tweet with the following functions to generate the feature vector:

Stop words: These words don't indicate any Hate Speech Prediction. Hence we remove it.

Repeating letters: In tweets, sometimes people repeat letters to stress the emotion. Two or more repetitive letters in words are replaced by two of the same letter.

Punctuation: We removed punctuation such as **:;,”, ’,,?**

Words must start with an alphabet: We removed all those words which don’t start with an alphabet. The word starting with number or symbol usually don’t have any Hate Speech Prediction.

### NEGATE\_WORD

Negation identification is not a simple task and its complexity increases, since negation words such as not, nor etc., (syntactic negation) are not the only criterion for negation calculation. The linguistic patterns also introduce the context of negation in textual data. The negation rules are designed in order to improve the Hate Speech Prediction text analysis. The Hate Speech Prediction probability for any frequently used positive word is very high and hence even when used with negation words gives positive Hate Speech Prediction. For example “I am not a good boy” gives positive Hate Speech Prediction as Hate Speech Prediction probability of “not” is nearly equal in positive and negative class. So, this function changes “not good” into single word “not\_good”. This single word will occur more only in negative sentence and hence gives “negative” as Hate Speech Prediction outcome.

### EXTRACT\_FEATURES

Extract features function just marks all the words in feature list as true or false for all class separately. This list is used for probability calculation for each word in the sentence. The words is marked as true for positive class if it is present in any sentence of positive class and false if it is not present. Similarly, it is done for all classes.

# 2.LITERATURE SURVEY

## MACHINE LEARNING

Literature [survey](http://www.blurtit.com/q876299.html) is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy and team strength. Once these things are satisfied, then next steps is to determine which operating system and language can be used for developing the tool. Once the [programmers](http://www.blurtit.com/q876299.html) start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from [book](http://www.blurtit.com/q876299.html) or from websites. Before building the system the above consideration are taken into account for developing the proposed system. We have to analyze Machine learning and Hate Speech Predicational analysis:

With the rise of Social Media internet users became able to easily express and share their opinions about companies, products, services, events etc. Thus, companies became interested in monitoring what people say about their brands in order to get feedback or enhance their marketing efforts.

Machine Learning has several interesting applications in Social Media Monitoring. It is used in order to evaluate the opinions of the users and classify them as positive, negative or neutral (Also known as Hate Speech Prediction Analysis). In addition, it can be used to detect whether the posts are objective or subjective, what is the natural language of the posts and whether the posts were written by a man or woman.

### HATE SPEECH PREDICTION ANALYSIS

Hate Speech Predicational Analysis is the process of computationally identifying and categorizing opinions expressed in a piece of text, especially in order to determine whether the writer's attitude towards a particular topic, product, etc. is positive, negative, or neutral**.** Hate Speech Prediction analysis is becoming a popular area of research and social media analysis, especially

around user reviews and tweets. It is a special case of text mining generally focused on identifying opinion polarity, and while it’s often not very accurate, it can still be useful. For simplicity (and because the training data is easily accessible) we will be focusing on 3 possible Hate Speech Prediction classifications: positive, negative and neutral.

#### Benefits of Hate Speech Predicational analysis:

Hate Speech Prediction analysis on tweets can be an excellent source of information and can provide insights that can:

* + - * Determine marketing strategy
      * Improve campaign success
      * Improve product messaging
      * Improve customer service
      * Test business KPIs

## SOFTWARE DESCRIPTION

### PYTHON

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library. Python interpreters are available for many operating systems, allowing Python code to run on a wide variety of systems.

Python uses dynamic typing and a mix of reference counting and a cycle-detecting garbage collector for memory management. An important feature of Python is dynamic name resolution (late binding), which binds method and variable names during program execution. Python has a large standard library, commonly cited as one of Python's greatest strengths, providing tools suited to many tasks

The Python Package Index, the official repository of third-party software for Python, contains more than 72,000 packages offering a wide range of functionality, including: graphical user interfaces, web frameworks, multimedia, databases, networking and communications…test frameworks, automation and web scraping, documentation tools, system administration…scientific computing, text processing, image processing.

### NLTK

The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for the Python programming language. NLTK includes graphical demonstrations and sample data. It is accompanied by a book that explains the underlying concepts behind the language processing tasks supported by the toolkit, plus a cookbook.

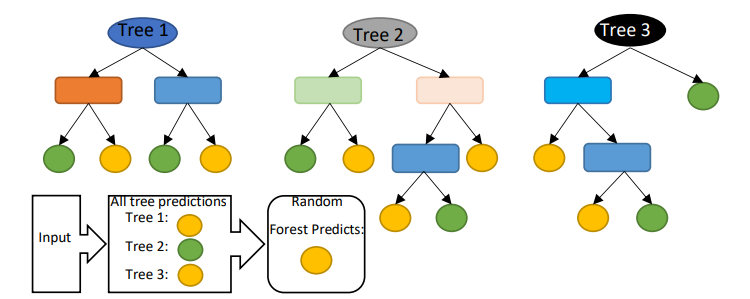
NLTK is intended to support research and teaching in NLP or closely related areas, including empirical linguistics, cognitive science, artificial intelligence, information retrieval, and machine learning. NLTK has been used successfully as a teaching tool, as an individual study tool, and as a platform for prototyping and building research systems.

**Algorithms:**

**RANDOM FOREST**

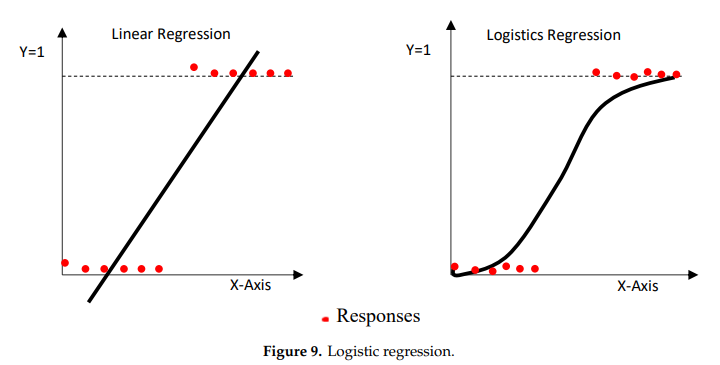
The Random Forest classification model is made up of several decision trees. In simple terms, it combines the results from numerous decision trees to reach a single result. The main difference between decision trees and random forests is that decision trees consider all the possible feature splits, however, random forests will only select a subset of those features.

RF was developed by Breiman, L. This is an ensemble learning algorithm made up of several DT classifiers, and the output category is determined collectively by these individual trees. When the number of trees in the forest increases, the fallacy in generalization error for forests converges. There are also important benefits of the RF. For example, it can manage high-dimensional data without choosing a feature; trees are independent of each other during the training process, and implementation is fairly simple; however, the training speed is generally fast and, at the same time, the generalization functionality is good enough [4]. Random forest algorithm for machine learning has tree predictions, and based on tree predictions, the RF provides random forest predictions [61]. The RF model is visualized in Figure



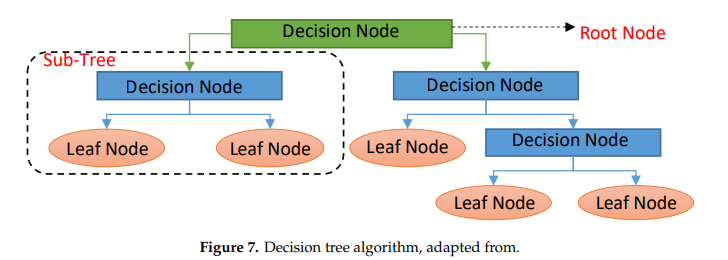
**Logistic Regression (LR):**

reported a study on forecasting the downtime of a printing machine based on real time predictions of imminent failures. In their study, they utilized unstructured historical machine data to train the ML classification algorithms including RF, XG Boost, and LR in predicting the machine failures. Various metrics were analyzed to determine the goodness of fit of the models. These metrics include empirical cross-entropy, area under the receiver operating characteristic curve (AUC), receiver operating characteristic curve itself (ROC), precision-recall curve (PRC), number of false positives (FP), true positives (TP), false negatives (FN), and true negatives (TN) at various decision thresholds, and calibration curves of the estimated probabilities. Based on the results obtained, in terms of ROC, all the algorithms performed significantly better and almost similar. But in terms of decision thresholds, RF and XG Boost perform better than LR. Using a given set of independent variables, linear regression is used to estimate the continuous dependent variations. However, using a given set of independent variables, logistic regression is used to estimate the categorical contingent variations [68]. Graph of the linear regression model and logistics regression model are shown in Figure 9.

****

**Decision Tree (DT)**:

Decision Tree is a network system composed primarily of nodes and branches, and nodes comprising root nodes and intermediate nodes. The intermediate nodes are used to represent a feature, and the leaf nodes are used to represent a class label [52]. DT can be used for feature selection [57]. DT algorithm is presented in Figure



DT classifiers have gained considerable popularity in a number of areas, such as character identification, medical diagnosis, and voice recognition. More notably, the DT model has the potential to decompose a complicated decision-making mechanism into a series of simplified decisions by recursively splitting co variate space into subs paces, thereby offering a solution that is sensitive to interpretation

#### Support Vector Machine

A support vector machine (SVM) is a type of supervised machine learning classification algorithm which outputs an optimal hyperplane that categorizes new examples given labeled training data [[15](#_bookmark72)]. SVM’s were introduced initially in 1960s and were later refined in 1990s. However, it is only now that they are becoming very popular, owing to their ability to achieve outstanding results.

Simple SVM: In case of linearly separable data in two dimensions, as shown in Figure [2.6](#_bookmark5), a typical machine learning algorithm tries to find a boundary that divides the data in such a way that the miss classification error can be minimized. If you closely look at Figure [2.6](#_bookmark5), there can be several boundaries that correctly divide the data points. The two dashed lines as well as one solid line classify the data correctly.

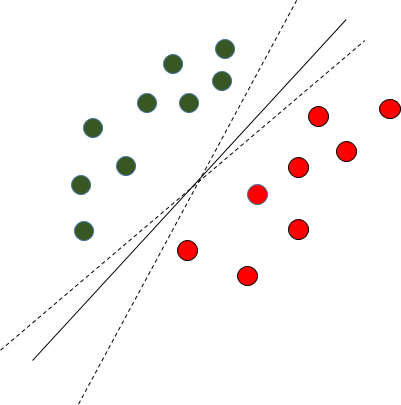


Fig. 2.6 Multiple Decision Boundaries

SVM differs from the other classification algorithms in the way that it chooses the decision boundary that maximizes the distance from the nearest data points of all the classes. An SVM doesn’t merely find a decision boundary; it finds the most optimal decision boundary. The most optimal decision boundary is the one which has maximum margin from the nearest points of all the classes. The nearest points from the decision boundary that maximize the distance between the decision boundary and the points are called support vectors as seen in Figure¬[2.7](#_bookmark6). The decision boundary in case of support vector machines is called the maximum margin classifier, or the maximum margin hyper plane.

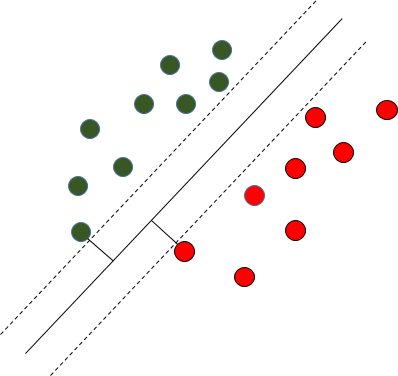


Fig. 2.7 Decision Boundary with Support Vectors

Kernel SVM: In the previous two figures Figure [2.6](#_bookmark5) and Figure [2.7](#_bookmark6) it was shown how the simple SVM algorithm can be used to find decision boundary for linearly separable data. However, in the case of non-linearly separable data, such as the one shown in Figure [2.8](#_bookmark7), a straight line cannot be used as a decision boundary.

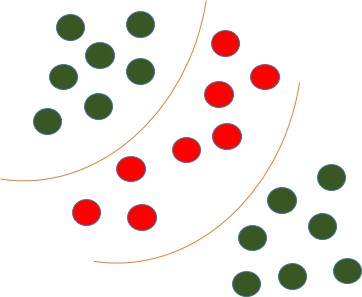


Fig. 2.8 Non-linearly Separable Data

In case of non-linearly separable data, the simple SVM algorithm cannot be used. Rather, a modified version of SVM, called Kernel SVM, is used. Basically, the kernel SVM projects the non-linearly separable data lower dimensions to linearly separable data in higher dimen- sions in such a way that data points belonging to different classes are allocated to different dimensions.

#### Random Forest

Random forest is a tree-based algorithm which involves building several trees (decision trees), then combining their output to improve generalization ability of the model. The method of combining trees is known as an ensemble method. Ensembling is nothing but a combination of weak learners (individual trees) to produce a strong learner

.

Definition: A random forest is a classifier consisting of a collection of tree structured classifiers *h*(*x,* Θ*k*)*, k* = 1*, ...* where the Θ*k* are independent identically distributed (*i.i.d*) random vectors and each tree casts a unit vote for the most popular class at input [[4](#_bookmark61)].

Random Forest Algorithm: The following are the basic steps involved in performing the random forest algorithm:

* + - * Pick N random records from the dataset.
      * Build a decision tree based on these N records.
      * Choose the number of trees you want in your algorithm and repeat steps (i) and (ii).
      * In case of a classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.

Figure [2.1](#_bookmark0) shows different trees labelling the class differently. What ensemble does is take the mode (maximum occurring class) of the output produced by n different trees to create a better model. To say it in simple words: Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

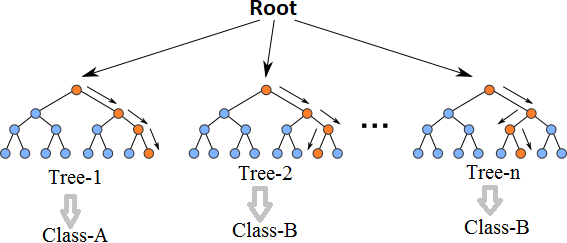


Fig. 2.1 Multiple decision trees [[12](#_bookmark69)]

Even though decision trees are pretty intuitive and easier to understand, they can be very noisy. Few changes in the data can lead to different splits and completely different models. The instability of the tree makes it unrealistic as a prediction model by itself. A single decision tree is insufficient and generally overfits the data, that is it can capture the structure of the in-sample data very well, but it tends to work poorly out-of-sample. In the context of statistics, decisions trees have low bias (as it can fit the data well) but high variances (the predictions are noisy).

Understanding the working principle of decision trees is imperative in the understanding of Random Forest Algorithm. The most popular algorithm for decision trees is ID3 algorithm. It finds the best attributes/features that best classifies the target attribute. One of the most commonly used way to figure out the best attribute is by calculating Information Gain which is, in turn, calculated using another property called Entropy.

The calculation of entropy of a system is done as follows:

*c*

*Entropy*(*S*) = ∑ *pilog*2 *pi*

−

*i*=1

Here, c is the total number of classes or attributes and *pi* is number of examples belonging to the *ith* class. Information gain is simply the expected reduction in entropy caused by partitioning all our examples according to a given attribute. Mathematically, it is defined as:

*Gain*(*S, A*) ≡ *Entropy*(*S*) − ∑ |*Sv*| *Entropy*(*Sv*)

*v*∈*Values*(*A*) |*S*|

S refers to the entire set of examples that we have. A is the attribute we want to partition or split. |S| is the number of examples and |*Sv*| is the number of examples for the current value of attribute A. The attribute with the highest information gain sits at the root node, and the tree is first split based on that attribute.

#### XG Boost:

XG Boost is another ensemble learning method. As it is almost never sufficient to reply upon the results of just one model, it combines the predictive powers of multiple learners to reach a conclusion. The base learners are weak learners in which the bias is high, and the predictive power is just slightly better than random guessing. But each of these weak learners add some vital information for prediction, resulting in a strong learner by effectively combining these weak learners. The final strong learner brings down both the bias and the variance.

The tree ensemble model consists of a set of classification and regression trees (CART). Figure [2.2](#_bookmark1) shows a simple example of a CART that classifies whether someone will like an app or not. The original figure from [[5](#_bookmark62)] had been modified to paint a better picture of our data set.

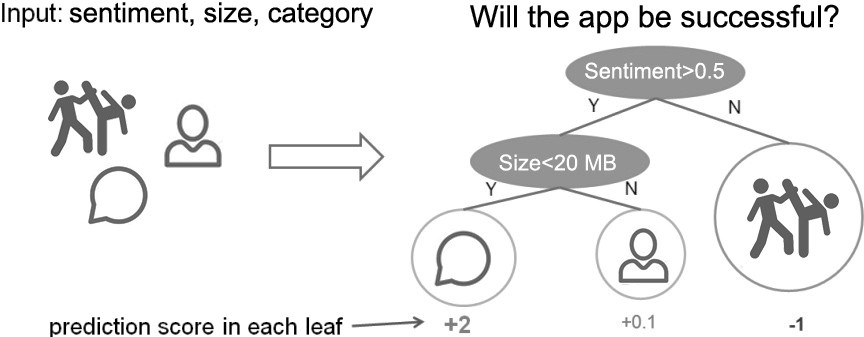


Fig. 2.2 CART Model Representation

Suppose, the many app categories available are classified into different leaves and as- signed a score on the corresponding leaf. Unlike decision trees, in which the leaf only contains decision values, in CART, a real score is associated with each of the leaves, which gives a better interpretation.

The task of training the model involves finding the best parameters *θ* that best fit the

training data *xi* and labels *yi*. This is done via the objective function which measures how well the model fits the training data. Objective functions are composed of two parts: training loss and regularization term which can be denoted by:

*ob j*(*θ* ) = *L*(*θ* ) + Ω(*θ* ) (2.3)

where *L* is the training loss function, and Ω is the regularization term. The regularization term controls the complexity of the model, helping to avoid overfitting.

While trees are built in a parallel manner in bagging, boosting builds trees sequentially such that each subsequent tree aims to reduce the errors of the previous tree. Figure [2.3](#_bookmark2) perfectly illustrates the concept. Due to each tree learning from its predecessors and updating the residual errors (difference between an observed y-value and the corresponding predicted y-value), the tree that grows next in the sequence will always learn from an updated version of the residuals. This is known as an additive strategy where what has already been learned is fixed, and a new tree is added one at a time.

The boosting process in its absolute basic can be broken down into the following steps:

* + - * Fit a model to the data: *F*1(*x*) = *y*
      * Fit a model to the residuals: *h*1(*x*) = *y* − *F*1(*x*)

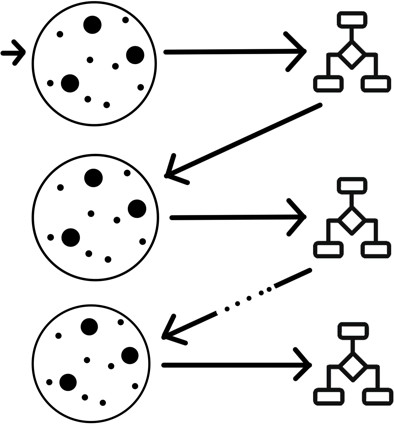


Fig. 2.3 Sequential Tree Structure

* Create a new model: *F*2(*x*) = *F*1(*x*) + *h*1(*x*)

By creating more models that correct the errors of the previous models, this can be generalized to:

*F*(*x*) = *F*1(*X* ) → *F*2(*x*) = *F*1(*x*) + *h*1(*x*)*. . . ..* → *FM*(*x*) = *FM*−1(*x*) + *hM*−1(*x*)*.* (2.4)

At each step, the residual would also need to be calculated: *hm*(*x*) = *y* − *Fm*(*x*) where *hm*(*x*) can be any model, but in our case, it is a tree-based learner. With this in mind, suppose that instead of training *h*0 on the residuals of *F*0, we train *h*0 on the gradient of the loss function, *L*(*y, F*0(*x*)) with respect to the prediction values produced by *Fm*(*x*). With samples in *hm* grouped into leaves, an average gradient can be calculated and then scaled by some factor, *γ*, such that *Fm* + *γhm* minimizes the loss function for the samples in each leaf. In practice, a different factor is chosen for each leaf. For iteration m = 1 to M:

* Calculate the gradient of L at the point *sm*−1
* “Step” in the direction of greatest descent (the negative gradient) with step size *γ*. That is, *sm* = *sm*−1 − *γL*(*sm*−1). If *γ* is small and *M* is sufficiently large, *sM* will be the location of *L* ‘s minimum value.

Most of these are true for all previous gradient boosting algorithms that came before XG Boost, but what really separates it from the others is [[22](#_bookmark79)]:

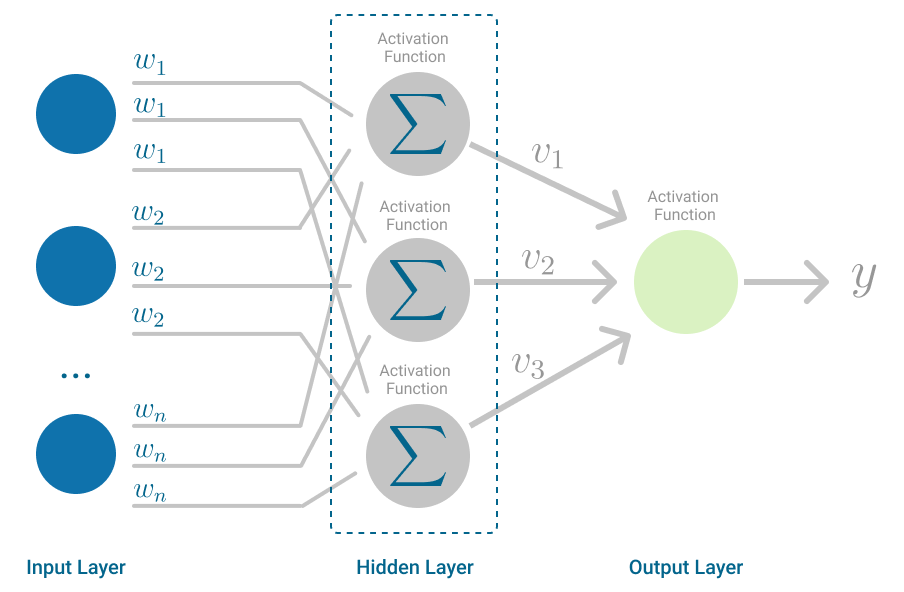
Regularization: XG Boost can penalize complex models through both L1 and L2 regularization which helps prevent over-fitting.

* + Handling sparse data: Missing values or data processing steps like one-hot encoding can make data sparse. XG Boost incorporates a sparsity-aware split finding algorithm that can handle different types of sparsity patterns in the data.
  + Weighted quantile sketch: Most existing tree based algorithms can find the split points when the data points are of equal weights (using quantile sketch algorithm). However, they can not handle weighted data. XG Boost has a distributed weighted quantile sketch algorithm that can effectively handle weighted data.
  + Block structure for parallel learning: For faster computing, XG Boost can utilize multiple cores on the CPU. Unlike other algorithms, this enables the data layout to be reused by subsequent iterations, instead of computing it again.
  + Cache awareness: In XG Boost, non-continuous memory access is required to get the gradient statistics by row index. Hence, XG Boost has been designed to make optimal use of hardware.
  + Out-of-core computing: This feature optimizes the available disk space and maximizes its usage when handling huge datasets that do not fit into memory

**Multi layer Perception :**

The**Multi layer Perceptron** was developed to tackle this limitation. It is a neural network where the mapping between inputs and output is non-linear.

A Multi layer Perceptron has input and output layers, and one or more **hidden layers** with many neurons stacked together. And while in the Perceptron the neuron must have an activation function that imposes a threshold, like Relu or sigmoid, neurons in a Multi layer Perceptron can use any arbitrary activation function.



**Multi layer Perceptron:**

Multi layer Perceptron falls under the category of [feedforward algorithms](https://en.wikipedia.org/wiki/Feedforward_neural_network" \t "_blank), because inputs are combined with the initial weights in a weighted sum and subjected to the activation function, just like in the Perceptron. But the difference is that each linear combination is propagated to the next layer.

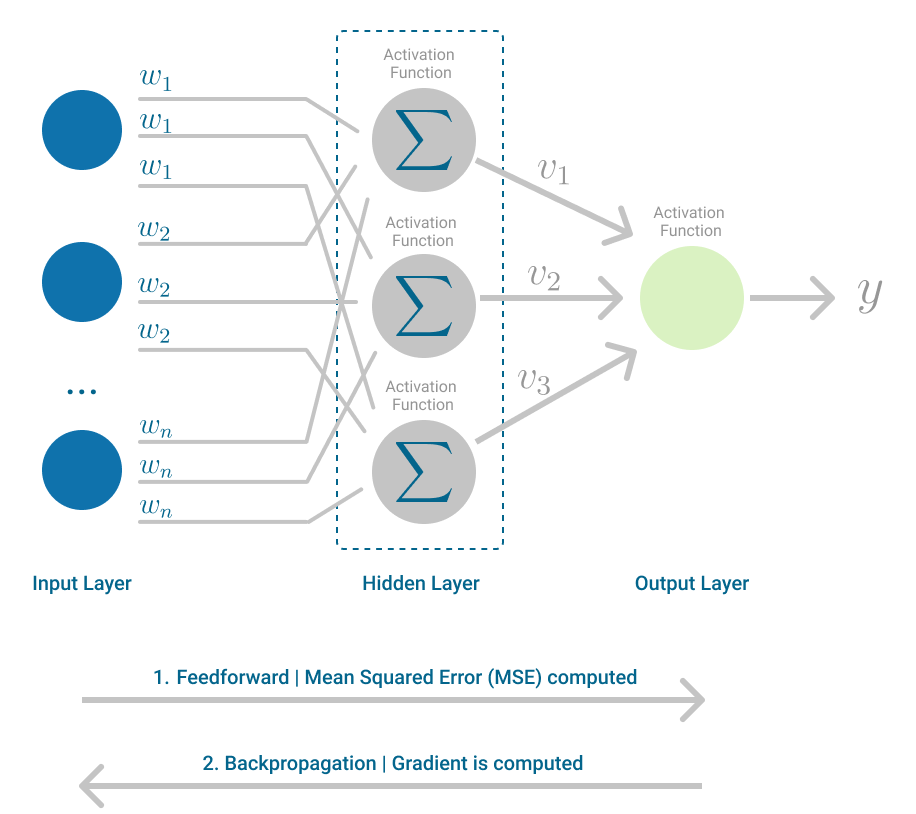
Each layer is feeding the next one with the result of their computation, their internal representation of the data. This goes all the way through the hidden layers to the output layer.But it has more to it.If the algorithm only computed the weighted sums in each neuron, propagated results to the output layer, and stopped there, it wouldn’t be able to learn the weights that minimize the cost function. If the algorithm only computed one iteration, there would be no actual learning.

This is where **[Backpropagation](https://en.wikipedia.org/wiki/Backpropagation" \t "_blank)** comes into play.

# Backpropagation:

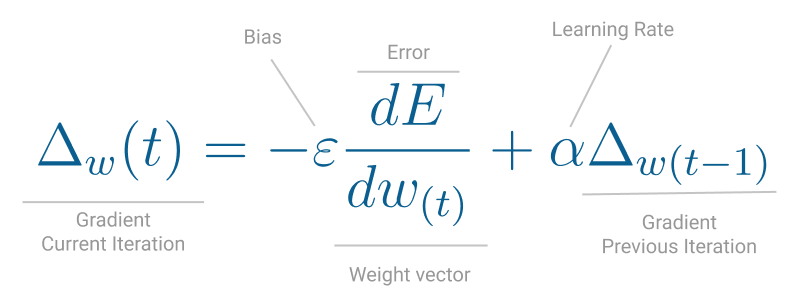
Back propagation is the learning mechanism that allows the Multi layer Perceptron to iteratively adjust the weights in the network, with the goal of minimizing the cost function.

There is one hard requirement for backpropagation to work properly. The function that combines inputs and weights in a neuron, for instance the weighted sum, and the threshold function, for instance ReLU, must be differentiable. These functions must have a **bounded derivative**, because [Gradient Descent](https://towardsdatascience.com/stochastic-gradient-descent-explained-in-real-life-predicting-your-pizzas-cooking-time-b7639d5e6a32" \t "_blank) is typically the optimization function used in MultiLayer Perceptron.



Multilayer Perceptron, highlighting the Feedforward and Backpropagation steps. (Image by author)

In each iteration, after the weighted sums are forwarded through all layers, the gradient of the **Mean Squared Error** is computed across all input and output pairs. Then, to propagate it back, the weights of the first hidden layer are updated with the value of the gradient. That’s how the weights are propagated back to the starting point of the neural network!



One iteration of Gradient Descent. (Image by author)

This process keeps going until gradient for each input-output pair has converged, meaning the newly computed gradient hasn’t changed more than a specified convergence threshold, compared to the previous iteration.

# 3.REQUIREMENT ANALYSIS

## FUNCTIONAL REQUIREMENTS

In software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behavior, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases. The following list shows functional requirement of our project:

* + - We should be able to train classifier with huge amount of data in sufficient time.
    - Comparison graph for training time for different algorithm is required.
    - Classifier should be able to give Hate Speech Prediction of input given by user or from twitter.
    - Fetch live tweets from twitter on mentioned topic and classify their Hate Speech Prediction.
    - Twitter will block streaming for frequent connection and streaming. So we should be able to store sufficient tweet for each execution.
    - Program should have testing option with accuracy graph as outcome of different algorithm.
    - Project must have functionality to store new training data to learn more every time.
    - Classifier should be trained only once to work anytime.

## NON-FUNCTIONAL REQUIREMENTS

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that define specific behavior or functions. The plan for implementing functional requirements is detailed in the system design. The plan for implementing non-functional requirements is detailed in the system architecture.

Other terms for non-functional requirements are "constraints", "quality attributes", "quality goals", "quality of service requirements" and "non-behavioral requirements".

Some of the quality attributes are as follows:

* + - ACCESSIBILITY
    - MAINTAINABILITY
    - SCALABILITY
    - PORTABILITY

### ACCESSIBILITY:

Accessibility is a general term used to describe the degree to which a product, device, service, or environment is accessible by as many people as possible.

In our project people who have registered with the apps on twitter and have all four Oath key should be able to fetch tweets from twitter using internet anytime.

Our program interface is simple and efficient and easy to use.

### MAINTAINABILITY:

* + - * In software engineering, maintainability is the ease with which a software product can be modified in order to:Correct defects
      * Meet new requirements

New functionalities can be added in the project based on the user requirements just by adding the appropriate module and functions to existing project using simple Python GUI.

Since the programming is very simple, it is easier to find and correct the defects and to make the changes in the project.

### SCALABILITY:

System is capable of handling increase total throughput under an increased load of training data when resources (typically hardware) are added.

System can work normally under situations such as low bandwidth and using less hardware resources.

### PORTABILITY:

Portability is one of the key concepts of high-level programming. Portability is the software code base feature to be able to reuse the existing code instead of creating new code when moving software from an environment to another.

Project can be executed under different operation conditions provided it meet its minimum configurations. Only python environment and common libraries would have to be configured in such case.

## HARDWARE REQUIREMENTS

Processor : Any Processor above 1.5 GHz

RAM : 2 GB

Hard Disk : 10 GB

Input device : Standard Keyboard and Mouse Output device : VGA or High-Resolution Monitor

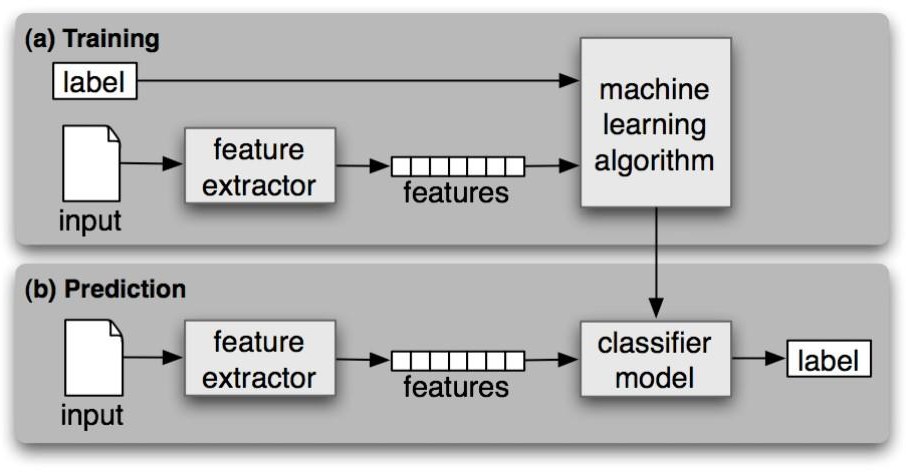
## SOFTWARE REQUIREMENTS

* Operating system : Windows XP or any higher version
* IDE : Python 2.7 (64bit)
* Internet : Low Bandwidth is enough

# 4.DESIGN

## DESIGN GOALS

To implement an algorithm for automatic classification of tweets into Positive, Negative or Neutral with highest accuracy possible. The tweets cab be gathered to summarize the overall Hate Speech Prediction on a particular topic.



**Fig 4.1 System Model**

### TRAINING DATA

Training data should consist of tweets and its Hate Speech Prediction. In our case we have csv file with two columns. First column is Hate Speech Prediction and second column is tweet.

### FEATURES

All the symbols, URL are removed and word starting with number and word without Hate Speech Prediction is removed. The remaining list of word is called features which is list of word that have some Hate Speech Prediction.

### MACHINE LEARNING ALGORITHM

The algorithm that is used to train classifier is called machine learning algorithm. It creates trained classifier. In our case we have algorithms with Naive Bayes as important and six others are Multinomial Naive Bayes, Bernoulli Naive Bayes, Logistic Regression, Stochastic Gradient Descent, Support vector clustering, Linear Support vector clustering.

### INPUT

The input to program is sentence or tweet.

### CLASSIFIER MODEL

We have seven classifier model. One for each machine learning algorithm that is used to classify input and gives output as label

### LABEL

Label is the Hate Speech Prediction output generated by classifier for the given input

## DATA FLOW DIAGRAM

Start

User

|  |  |  |
| --- | --- | --- |
| processTweet() | | processTweet() |
| getFeaturevector( | | getFeaturevector() |
| negateWords() | | negateWords() |
| extractFeatures() | | extractFeatures() |
|  | Accuracy | Confidence level (based on UI) |

|  |  |
| --- | --- |
| processTweet() | |
| getFeaturevector() | |
|  | |
|  | negateWords() |
|  | extractFeatures() |
|  | |

Pie chart based

on UI

Pie chart

based on twitter data

**Fig 4.2: DATA FLOW DIAGRAM**



UI==1

UI==2

UI==3

UI==4

UI==5

Exit

Confidence level

(based on twitter tweets)

`

Loading pickle

Bar graph w.r.t

accuracy

Sentiment (positive,

negative, neutral)

EOP

Trained pickle

getFeaturevector()

Bar graph w.r.t time

extractFeatures()

negateWords()

processTweet()

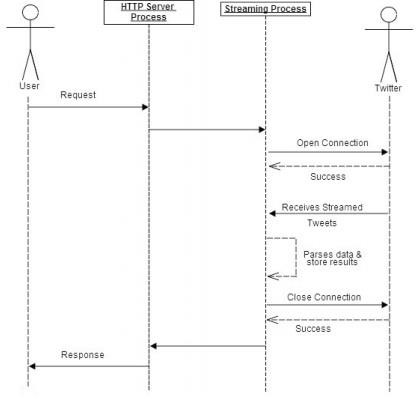
Twitter

Try

Test

Train

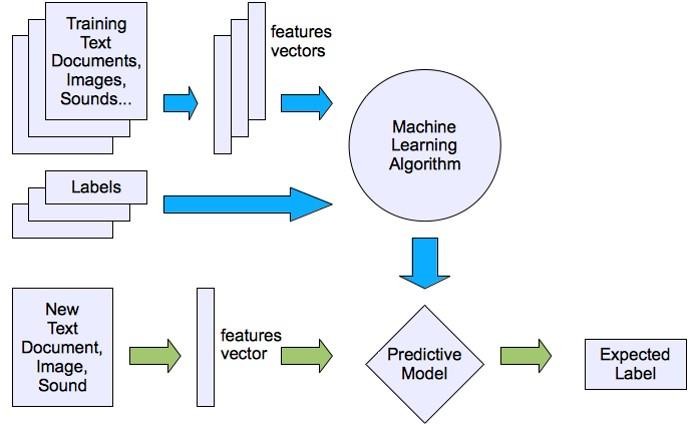
## SEQUENCE DIAGRAM FOR TWITTER STREAMING



**Fig 4.3: SEQUENCE DIAGRAM**

The above diagram shows how to fetch tweet from twitter to use it as input to classifier and calculate Hate Speech Prediction.

# 5.IMPLEMENTATION



**Fig 5.1: Implementation Steps of Hate Speech Prediction Analysis**

The figure above shows implementation steps that is required in Hate Speech Prediction analysis problem. The detail explanation about each step in Hate Speech Prediction analysis on twitter data are given below.

## 

## SYSTEM ARCHITECTURE

## 

## The purpose of the design phase is to arrange an answer of the matter such as by the necessity document. This part is that the opening moves in moving the matter domain to the answer domain. The design phase satisfie s the requirements of the system. The design of a system probably the foremost crucial issue warm heart edness the standard of the software package. It’s a serious impact on the later part, notably testing and maintenance.

## The output of this part is that the style of the document. This document is analogous to a blueprint of answer and is employed later throughout implementation, testing and maintenance. The design activity is commonly divided into 2 separate phases System Design and Detailed Design.

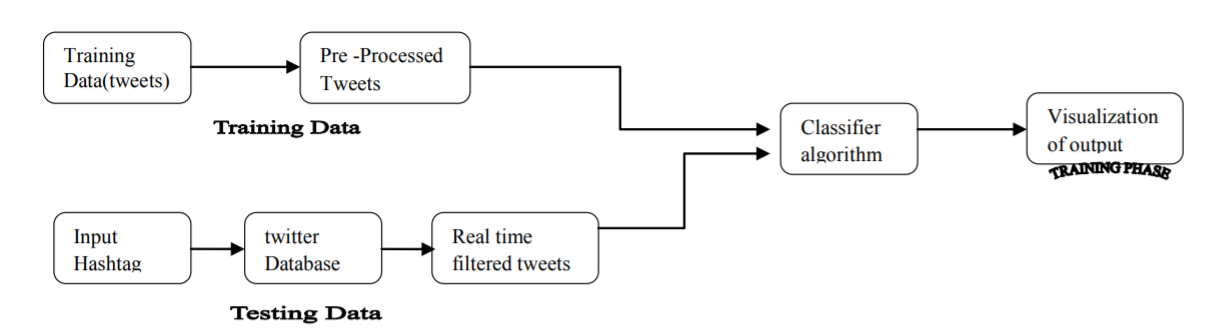
## System Design conjointly referred to as top-ranking style aims to spot the modules that ought to be within the system, the specifications of those modules, and the way them move with one another to supply the specified results.

## At the top of the system style all the main knowledge structures, file formats, output formats, and also the major modules within the system and their specifications square measure set. System design is that the method or art of process the design, components, modules, interfaces, and knowledge for a system to satisfy such as needs. Users will read it because the application of systems theory to development.

## Detailed Design, the inner logic of every of the modules laid out in system design is determined. Throughout this part, the small print of the info of a module square measure sometimes laid out in a high-level style description language that is freelance of the target language within which the software package can eventually be enforced.

## In system design the main target is on distinguishing the modules, whereas throughout careful style the main target is on planning the logic for every of the modules.

## 



## Figure 7.1: Architecture diagram

## Here first we collect the data sets and process the data and we remove if there are any impurities in the data sets. Next the data is normalized if needed like it can be converted to smaller volume of data. Next the data is converted to supporting format. And then it is stored in the databases. Next the required method is applied. Now we get the final results.

## DATA FLOW DIAGRAMS

## Data Flow Diagram can also be termed as bubble chart. It is a pictorial or graphical form, which can be applied to represent the input data to a system and multiple functions carried out on the data and the generated output by the system.

## A graphical tool accustomed describe and analyze the instant of knowledge through a system manual or automatic together with the method, stores of knowledge, and delays within the system. The transformation of knowledge from input to output, through processes, is also delineate logically and severally of the physical elements related to the system. The DFD is also known as a data flow graph or a bubble chart. The Basic Notation used to create a DFD’s are as follows:

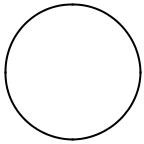
* **Dataflow:**





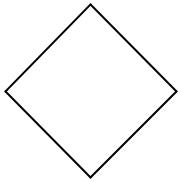
* **Process:**

.



* **Source:**
* **Data Store:**



* **Rhombus**: decision

**UML DIAGRAMS**

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

**User Model View**

This view represents the system from the user’s perspective. The analysis representation describes a usage scenario from the end-user’s perspective.

**Structural Model view**

In this model the data and functionality are arrived from inside the system. This model view models the static structures.

**Behavioral Model View**

It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

**Implementation Model View**

In this the structural and behavioral as parts of the system are represented as they are to be built.

**USE CASE DIAGRAM**

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

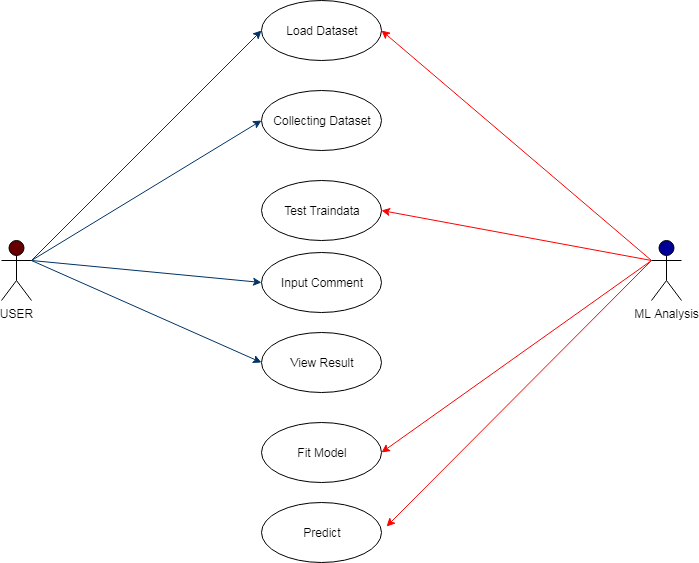


Figure 7.3.1 Use Case Diagram

**CLASS DIAGRAM**

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

The upper part holds the name of the class

The middle part contains the attributes of the class

The bottom part gives the methods or operations the class can take or undertake.

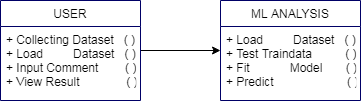
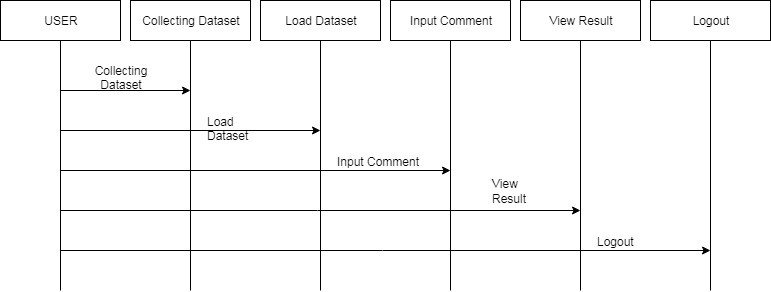


Figure 7.3.2: Class Diagram.

## SEQUENCEDIAGRAM

A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



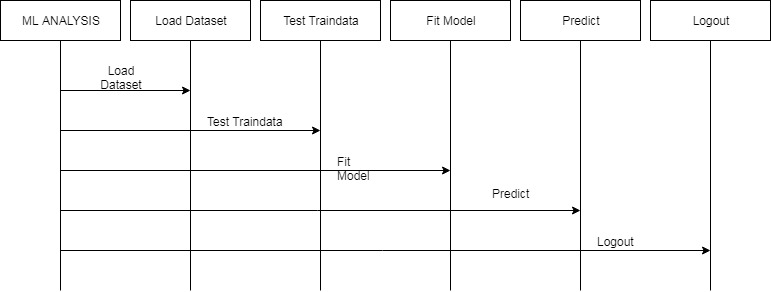


Figure 7.3.3: Sequence diagram

## ACTIVITY DIAGRAM

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

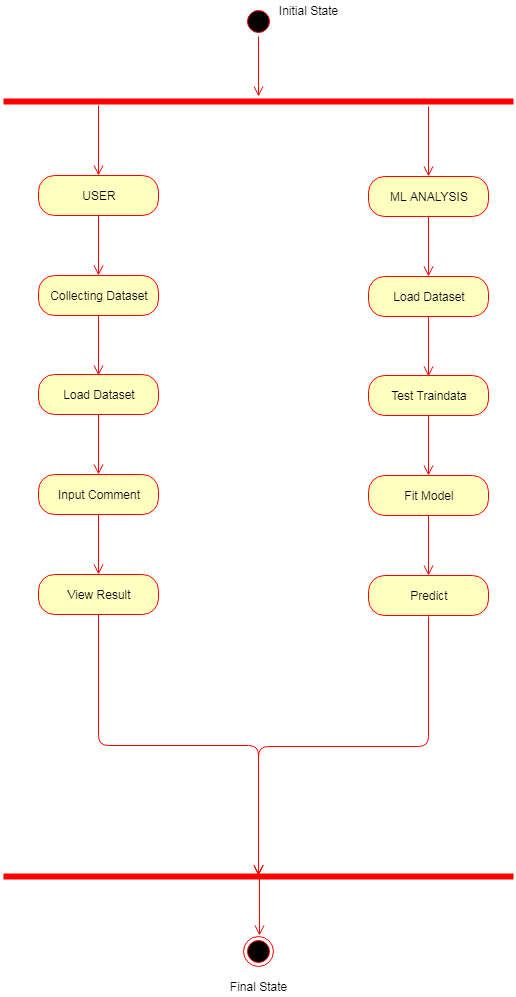
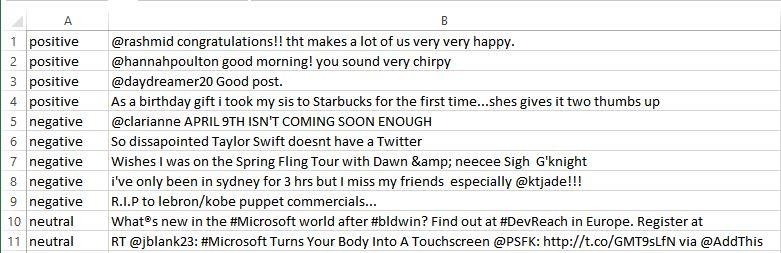


Figure 7.3.4: Activity Diagram

## META-DATA GENERATION

Let the user wishes to use program to train classifier with training data for Hate Speech Prediction analysis with three classes namely positive, negative and neutral. Let this file be CSV file which consist of two columns containing Hate Speech Prediction (positive, negative, neutral) in first column (A) and Tweet of max 140 character in second column (B). Each row is training data and we can have huge number of training data.



**Fig 5.2: CSV file**

We initially read the file using csv reader and create list of dataset name it as Tweets. Each element in list have Hate Speech Prediction and a Tweet.

Each row is then separated into two part. First as Hate Speech Prediction and second as Tweet. Consider first row.

Hate Speech Prediction = “positive”

Tweet = “@sooraj \\ I can't afford flight tickets of [http://www.jetways.com](http://www.jetways.com/) but I can buy train ticket. #sad”

## CREATION OF PROCESSED TWEETS AND FEATURE LIST

### PRE-PROCESSING

* + - 1. Convert to lowercase
      2. Replace @user with at user.
      3. Replace #Tag with Tag.
      4. Replace [http://www.url.com](http://www.url.com/) with URL.
      5. Remove special characters that doesn’t affect Hate Speech Prediction.

Tweet before preprocessing is as follows:

Tweet = “@sooraj \\ I can't afford flight tickets of [http://www.jetways.com](http://www.jetways.com/) but I can buy train ticket. #sad”

Tweet after preprocessing is as follows:

Tweet =“at user i can't afford flight tickets of URL but i can buy train ticket. sad”

### CREATING FEATURE VECTOR

* + - 1. Tokenization
      2. Replace two or more with two occurrences
      3. Remove the word that doesn’t start with alphabet
      4. Remove all stop word which doesn’t have any Hate Speech Prediction

Tweet after preprocessing before creating feature vector is as follows:

Tweet =“at user i can't afford flight tickets of URL but i can buy train ticket. sad”

Tweet after creating feature vector is as follows: FeatureVector = “ can't, afford, but, can, buy., sad ”

### CREATE NEGATED FEATURE VECTOR

* + - 1. Attach “not\_” to words which have negation impact.
      2. Remove negation words.
      3. Remove the remaining punctuation.

Tweet after creating feature vector is as follows:

FeatureVector = “ Not\_afford, can, buy, sad ” Hate Speech Prediction = “positive”

### CREATE FEATURE LIST

FeatureList = FeatureList + FeatureVector

FeatureList = FeatureList + [Not\_afford, can, buy, sad]

### APPENDING HATE SPEECH PREDICTION AND FEATURE VECTOR

Tweets = Tweets + [“positive, Not\_afford, can, buy, sad”]

### REPEAT STEP 5.2.1 TO 5.2.5

Step from 5.2.1 till 5.2.5 is repeated for each tweet in training set to get complete list of Processed Tweets and Feature List

## USE EXTRACT FEATURES TO GET FEATURES

Extract features function just marks all the words in feature list as true or false for all class separately. This list is used for probability calculation for each word in the sentence.

The words is marked as true for positive class if it is present in any sentence of positive class and false if it is not present. Similarly, it is done for all classes. This outcome is called as Features.

## APPLY FEATURES TO GET TRAINING SET

The apply feature function from NLTK (Natural Language Processing Tool) is used to create Training set from Features.

## USE TRAINING SET TO TRAIN CLASSIFIER

The function train () in Naïve Bayes Classifier class of NLTK library takes training set created in 5.2.6 as parameter and returns trained classifier. Similarly other six classifier (Multinomial Naïve Bayes, Bernoulli Naïve Bayes, Logistic Regression, Stochastic Gradient Descent, Support vector clustering, Linear Support vector clustering) are created using train () function from sklearn library of different machine learning classes.

## 

## TRAINED CLASSIFIER IS STORED IN PICKLE

The trained classifier is stored in the system to avoid training classifier every time. Whenever it is required to classify the tweet or sentence then the classifier is loaded from pickle and used to give the output without training once again.

## USER INPUT SENTENCE TO CLASSIFY

Now comes the user part. User enters a sentence and the preprocessing that is 5.2.1 to

5.2.3 is done to create features. The function classify () takes features and does the classification to give Hate Speech Prediction as output. All seven classifier gives they output as positive or negative or neutral. The output may vary of different classifier which is based on methodology used to classify.

## USER INPUT

In this module hate speech dataset is collected from Kaggle website which has hate and normal text and features and o and 1 as labels**.**

## CALCULATING CONFIDENCE

Hate Speech Prediction output from seven classifier can be used for voting to give which is the best Hate Speech Prediction. The Hate Speech Prediction with the majority vote wins. For example, 4 out of seven classifier gives positive as output and other 3 as negative or neutral then confidence is calculated as follows:

No. of positive = 4 No. of negative = 3 No. of neutral = 0

Output = Max (No. of positive, No. of negative, No. of neutral) Output = 4 (positive)

Confidence Level = (output\Total) \* 100 % Confidence Level = (4\7) \* 100 % = 57.14 %

**Hence the output will be positive with 57.14 confidenc**

## MORE TRAINING DATA

If output given of the classifier is wrong then we can teach them the correct output by storing correct output of the tweet. After every execution it ask whether outcome was correct or not. If user enters as wrong then control ask the correct answer and stores it for next training. After training classifier next time, it learns about that tweet and STORING increases its correctness. Hence this is a continuous learning process like human

.

# 6.TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product 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 tests. Each test type addresses a specific testing requirement.

## TYPES OF TESTS

## 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.

## 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 satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

## VALIDATION TESTING

An engineering validation test (EVT) is performed on first engineering prototypes, to ensure that the basic unit performs to design goals and specifications. It is important in identifying design problems, and solving them as early in the design cycle as possible, is the key to keeping projects on time and within budget. Too often, product design and performance problems are not detected until late in the product development cycle — when the product is ready to be shipped. The old adage holds true: It costs a penny to make a change in engineering, a dime in production and a dollar after a product is in the field.

Verification is a Quality control process that is used to evaluate whether or not a product, service, or system complies with regulations, specifications, or conditions imposed at the start of a development phase. Verification can be in development, scale-up, or production. This is often an internal process

Validation is a Quality assurance process of establishing evidence that provides a high degree of assurance that a product, service, or system accomplishes its intended requirements. This often involves acceptance of fitness for purpose with end users and other product stakeholders.

The testing process overview is as follows:



**VALIDATION TESTING**

**INTEGRATION TESTING**

**UNIT TESTING**

**Fig 6.1: The testing process**

## SYSTEM TESTING

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

As a rule, system testing takes, as its input, all of the "integrated" software components that have successfully passed integration testing and also the software system itself integrated with any applicable hardware system(s).

System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

System testing is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS).

System testing tests not only the design, but also the behavior and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification(s).

## TESTING OF INITIALIZATION AND UI COMPONENTS

**Table 6.1: Test case for train option**

|  |  |
| --- | --- |
| Serial Number of Test Case | TC 01 |
| Module Under Test | Train |
| Description | When the train option is selected, it takes size as input and trains the classifiers. |
| Output | If the training size is valid, it generates bar chart, which shows training time taken for training the classifiers. If size is incorrect, an exception is thrown. |
| Remarks | Test Successful. |

#### Table 6.2: Test Case for test classifier option

|  |  |
| --- | --- |
| Serial Number of Test Case | TC 02 |
| Module Under Test | Test |
| Description | An option where user enter the size for testing the classifier and it generates the accuracy score. |
| Input | Size for testing the classifier |
| Output | If the size is valid, it generates the accuracy of all the classifiers, along with bar chart for the same. If the size is invalid, an exception is thrown. |
| Remarks | Test Successful. |

**Table 6.3: Test Case for try option**

|  |  |
| --- | --- |
| Serial Number of Test Case | TC 03 |
| Module Under Test | Try |
| Description | When user enter the tweet, it generates the corresponding Hate Speech Prediction along with the confidence level of the user tweet. |
| Input | User sentence(tweet) |
| Output | The user is shown with the Hate Speech Prediction of the sentence and pie chart for the same is displayed. |
| Remarks | Test Successful. |

**Table 6.4: Test Case for tweet option**

|  |  |
| --- | --- |
| Serial Number of Test Case | TC 04 |
| Module Under Test | Tweet |
| Description | When the user selects this option, user is asked for tweet topic. This topic is fetched from Twitter and corresponding Hate Speech Prediction is generated |
| Input | User enters the topic to be fetched from Twitter. |
| Output | The Hate Speech Prediction of the tweet fetched from Twitter based on user entered topic and pie chart are generated. |
| Remarks | Test Successful. |

**Table 6.5: Test Case for store option**

|  |  |
| --- | --- |
| Serial Number of Test Case | TC 05 |
| Module Under Test | Store |
| Description | User is provided with option to correct the Hate Speech Prediction generated, if user feels the generated Hate Speech Prediction is wrong. |
| Input | Save option, Correct Hate Speech Prediction |
| Output | Correct Hate Speech Prediction of the tweet is saved |
| Remarks | Test Successful. |

# 7.CONCLUSION AND FUTURE ENHANCEMENT

## CONCLUSION

The applications for Hate Speech Prediction analysis are endless. It is used in social media monitoring and Voice of Customers (VOC) to track customer reviews, survey responses, competitors, humanoid robots etc. However, it is also practical to use it in business analytics and situations for text analysis.

As we know every coin have two sides, Hate Speech Prediction analysis is great but it’s a difficult task. The difficulty increases with increase in complexity of opinions expressed. In some of the fields like product reviews, face recognition, span filter etc. are relatively easy whereas fields like books, movies, art, music, indirect expressions of opinion are more difficult.

Hate Speech Prediction analysis is in demand because of its efficiency. Thousands of text documents can be processed for Hate Speech Prediction in seconds, compared to the hours by a team of people to manually complete it. It is so efficient, accurate and fast that many businesses are adopting text and Hate Speech Prediction analysis and incorporating it into their business processes.

The great thing about social media Hate Speech Prediction analysis is that you're not looking for the needle in the hay. Hate Speech Prediction mining is looking into trends and large numbers of people. It means that you can account for some degree of fuzziness in Hate Speech Prediction classification with the raw amount of data otherwise we will come to know that trends we searching is not popular or important.

## FUTURE ENHANCEMENT

We took existing algorithm and tried to enhance the accuracy with different preprocessing work. We build a confidence level to increase accuracy and solved negation problem. But there are lot more enhancement that we can do with this project. We removed a problem of training every time using pickle which have ability to store trained classifier. Some of the enhancement that we were not able to do due to time constraint are as follows:

* + - Training with large and better data for best accuracy.
    - Enhancing Naïve Bayes using bi-gram, tri-gram, n-gram etc.
    - Training with dataset of different lexicon for improving accuracy of varieties of sentences.
    - There are some complex machine learning algorithm like neural network that have great accuracy.
    - Reducing the execution time by training the different algorithm classifier in parallel way.
    - Creating a better user interface.
    - Creating an executable file that can be executed without requiring python environment.

# BIBLIOGRAPHY

Good Teachers are worth more than thousand books, we have them in Our Department. Specially our guide

## REFERENCE PAPERS:

* + - * Twitter Hate Speech Prediction Analysis: The Good the Bad and the OMG! By Efthymios Kouloumpis, Theresa Wilson, Johanna Moore
      * Hate Speech Prediction Analysis of Twitter Data by Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow and Rebecca Passonneau
      * Modeling and Representing Negation in Data-driven Machine Learning- based Hate Speech Prediction Analysis by Robert Remus
      * Hate Speech Prediction Analysis of Twitter Data Using Machine Learning Approaches and Semantic Analysis by Geetika Gautam
      * A Framework for Fast-Feedback Opinion Mining on Twitter Data Streams by Lokmanyathilak Govindan Sankar Selvan and Teng-Sheng Moh
      * Hate Speech Prediction Analysis on Twitter by Akshi Kumar and Teeja Mary Sebastian
      * Hate Speech Prediction Analysis on Twitter through Topic-based Lexicon Expansion by Zhixin Zhou, Xiuzhen Zhang, and Mark Sanderson
      * Serendio: Simple and Practical lexicon based approach to Hate Speech Prediction Analysis by Prabu Palanisamy,Vineet Yadav and Harsha Elchuri

### TEXT BOOKS:

1. Learning Python, Fifth Edition by Mark Lutz
2. Python Essential Reference by David Beazley
3. Python for Data Analysis by Wes McKinney
4. Introduction to Python for Econometrics, Statistics and Data Analysis by Kevin Sheppard from University of Oxford
5. A Python 2.7 programming tutorial Version 2.0 by John W. Shipman

### SITES REFFERED:

* + - https:/[/www.p](http://www.python.org/)y[thon.org/](http://www.python.org/)
    - <http://scikit-learn.org/>
    - <http://www.nltk.org/>
    - <http://www.lfd.uci.edu/~gohlke/pythonlibs/>
    - https://dev.twitter.com/overview/documentation
    - <http://matplotlib.org/>
    - [http://nlp.stanford.edu](http://nlp.stanford.edu/)