

Twitter Spam Classification Using Machine Learning Techniques

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

G. VENKATESH (Reg No.38119503)

D. SANTOSH REDDY (Reg No.38119504)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI – 600119, TAMILNADU



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" Grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600 119.

Phone: 044 - 2450 3150 / 51 / 52 / 54 / 55 Fax: 044 - 2450 2344
www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **GALI VENKATESH (Reg No: 38119503)** and **DWARAMPUDI SANTOSH REDDY (Reg No: 38119504)** who carried out the project entitled **"TWITTER SPAM CLASSIFICATION USING MACHINE LEARNING TECHNIQUES"** under our supervision from **NOVEMBER 2021** to **APRIL 2022**.

INTERNAL GUIDE

Dr. C. KAVITHA, M.E.

HEAD OF THE DEPARTMENT

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, D.santosh reddy (Reg No: 38119504) ,hereby declare that the Project Report entitled as **“TWITTER SPAM CLASSIFICATION USING MACHINE LEARNING TECHNIQUES”** done by me under the guidance of **Dr. C. KAVITHA, M.E., Ph.D.,** Department of Information Technology at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering.

DATE: 07/05/2022

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

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

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

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. T. SASIKALA, M.E., Ph.D., Dean**, Department of Computer Science and Engineering for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

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

ABSTRACT

Stream clustering methods have been repeatedly used for spam filtering in order to categorize input messages/tweets into spam and nonspam clusters. These methods assume each cluster contains a number of neighbor small (micro) clusters, where each microcluster has a symmetric distribution. Nonetheless, this assumption is not necessarily correct and big microclusters might have asymmetric distribution. To enhance the assigning accuracy of former methods in their online phase, we suggest replacing the Euclidean distance by a set of classifiers in order to assign incoming samples to the most relative microcluster with arbitrary distribution. Here, a set of incremental Naïve Bayes (INB) classifier is trained for microclusters whose population exceeds a threshold. These INBs can capture the mean and boundary of microclusters, while the Euclidean distance just considers the mean of clusters and acts inaccurate for asymmetric big microclusters. In this paper, DenStream was promoted by the proposed framework, called here as INB DenStream. To show the effectiveness of INB-DenStream, state-of-the-art methods such as DenStream, StreamKM++, and CluStream were applied to the Twitter datasets and their performance was determined in terms of purity, general precision, general recall, F1 measure, parameter sensitivity, and computational complexity. The compared results implied the superiority of our method to the rivals in almost the datasets.-

TABLE OF CONTENTS

CHAPTER No	TITLE	PAGE
	ABSTRACT	5
	List of figures	8
1	INTRODUCTION	10
1.1	About the Project	10
2	SYSTEM ANALYSIS	11
2.1	Existing system	11
2.2	Proposed system	11
3	REQUIREMENTS SPECIFICATION	12
3.1	Introduction	12-14
3.2	Hardware and Software specification	14
3.3	Technologies Used	14-21
4.1	Design and Implementation Constraints	22
4.2	Other Nonfunctional Requirements	23-24
5	Architecture Diagram	25

6	Modules	32
6.1	Modules	32
6.2	Module explanation	32
7	CODING AND TESTING	33
7.1	Coding	33
7.2	Coding standards	33-35
7.3	Test procedure	35-36
7.4	Test data and output	36-38
	SOURCE CODE	43-75
	SNAP SHOTS	76-79
	REFERENCES	80-84

LIST OF FIGURES

5.1 Sequence Diagram

5.2 Use Case Diagram

5.3 Activity Diagram

5.4 Collaboration Diagram:

List of figures

FIGURE NO	TITLE	PAGE NO
5.1	Sequence Diagram	26
5.2	Use Case Diagram	28
5.3	Activity Diagram	30
5.4	Collaboration Diagram:	31
	Comparison of among the four Classifiers snapshot	77
	Predict the tweet with Spam or Ham snapshot	79
	spam word or ham word snapshots	80

CHAPTER 1

INTRODUCTION

Aim:

To enhance the assigning accuracy of former methods in spam detection in Twitter using advanced methods.

Synopsis:

Social networking sites have become very popular in recent years. Users use them to find new friends, updates their existing friends with their latest thoughts and activities. Among these sites, Twitter is the fastest growing site. Its popularity also attracts many spammers to infiltrate legitimate user's accounts with a large amount of spam messages. In this paper, we discuss some user-based and content-based features that are different between spammers and legitimate users. Then, we use these features to facilitate spam detection. Using the API methods provided by Twitter, we crawled active Twitter users, their followers/following information and their most recent 100 tweets. Then, we evaluated our detection scheme based on the suggested user and content-based features. Our results show that among the four classifiers we evaluated, the Random Forest classifier produces the best results. Our spam detector can achieve 95.7% precision and 95.7% F-measure using the Naive Bayes Algorithm

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

Stream clustering methods have been repeatedly used for spam filtering in order to categorize input messages/tweets into spam and non-spam clusters. These methods assume each cluster contains a number of neighbor small (micro) clusters, where each micro-cluster has a symmetric distribution. Nonetheless, this assumption is not necessarily correct and big micro clusters might have asymmetric distribution. To enhance the assigning accuracy of former methods in their online phase, we suggest replacing by machine learning classifiers. Our results show that among the four classifiers we evaluated, the Random Forest classifier produces the best results.

2.2 PROPOSED SYSTEM

We discuss some user-based and content-based features that are different between spammers and legitimate users. Then, we use these features to facilitate spam detection. Using the API methods provided by Twitter, we crawled active Twitter users, their followers/following information and their most recent 100 tweets. Then, we evaluated our detection scheme based on the suggested user and content-based features. Our results show that among the four classifiers we evaluated, the Random forest classifier produces the best results.

CHAPTER 3

REQUIREMENT SPECIFICATIONS

3.1 INTRODUCTION

ONLINE social networks (OSNs) like Facebook, Instagram, and Twitter have been tremendously grown and become the most popular websites in recent years. According to the latest statistics, Twitter has 200 million members, who generate over 400 million tweets per day [1]. A considerable amount of these tweets is spams like advertising messages, fishing attack, spreading malwares links, and financial fraud [2]. Spam tweets have common indicators like “Hashtags,” “mentions,” and shortened URLs, whereas every message containing these indicators is not necessarily a spam. This is, therefore, the task of spam filtering is really a challenging issue. Due to the character limitation policy in Twitter, shortened URL is frequently utilized by spammers to deceive the members. Since the destination of URLs is unknown in Twitter, the capability of disguising URL destination makes it a charming target for spammers [3]. Besides investigating the message content, the behavior of members can be monitored for identifying the source of spams. For instance, if the number of sent messages by a member exceeds his number of connected friends, all of his messages can be categorized as spam. To deceive the investigators, some spammers randomly send limited amount of spams to members or use fake trending hashtags to make spams more visible in searches. Considering the existence of spam indicators, machine

learning methods have been utilized for spam detection through stream of messages in different OSNs [3]–[6]. So far, both supervised and unsupervised methods have been deployed, extended, and also customized for the detection of spams/spammers in OSNs. It is obvious that supervised methods are able to produce better results than the unsupervised ones; however, the high cost of labelling for big datasets limits the applicability of this approach for spam filtering [7]. Nonetheless, the results of the conventional stream clustering methods implied their insufficient flexibility in confronting with concept drift [8], [9].

In this paper, we developed a novel stream clustering framework, which is able to promote all conventional stream clustering methods by replacing the Euclidean distance, with a set of incremental Naïve Bayes (INB) classifiers in their online phase. In this paper, we chose DenStream [10] due to its performance (among the stream clustering methods) in order to enhance its performance by our framework. The modified version of DenStream by INBs is called “INB-DenStream.” For each susceptible microcluster, which its population exceeds a threshold, an INB is trained to capture the center and boundary of that cluster, and therefore, these set of INBs are able to more accurately assign incoming samples to the clusters. By tracing the rate of changes along with population of microclusters through the time, our scheme is able to change the status of susceptible and nonsusceptible microclusters in order to digest the concept change in stream of messages. Moreover, to enrich the elicited features from tweets, a combination of user behavior dependent and tweet content dependent features [5] were elicited from Twitter datasets. The performance of our method was compared to state-of-the-art stream clustering methods in terms of purity, general precision, general recall, F1 measure, and computational complexity. The rest of this paper is structured as follows. Section II reviews the conventional stream clustering methods and carefully analyzes their pros and cons. Section III explains the

employed datasets in this paper, and then the stages of our proposed method are illustrated. Section IV presents the results achieved by the proposed method and compares them to the results of the implemented counterparts and then the advantages and disadvantages of the methods are discussed. Finally, this paper is concluded in Section V and some outlines are suggested as future work.

3.2 HARDWARE AND SOFTWARE SPECIFICATION

3.2.1 HARDWARE REQUIREMENTS

- Hard Disk : 500GB and Above
- RAM : 4GB and Above
- Processor : I3 and Above

3.2.2 SOFTWARE REQUIREMENTS

- ✓ Operating System : Windows 7 , 8, 10 (64 bit)
- ✓ Software : Python 3.7
- ✓ Tools : Anaconda (Jupyter Note Book IDE)

3.3 TECHNOLOGIES USED

- Python

3.3.1 Introduction to Python

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was

mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- System scripting.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

Good to know

- The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- Python 2.0 was released in 2000, and the 2.x versions were the prevalent releases until December 2008. At that time, the development team made the decision to release version 3.0, which contained a few relatively small but significant changes that were not backward compatible with the 2.x versions. Python 2 and 3 are very similar, and some features of Python 3 have been backported to Python 2. But in general, they remain not quite compatible.
- Both Python 2 and 3 have continued to be maintained and developed, with periodic release updates for both. As of this writing, the most recent versions available are 2.7.15 and 3.6.5. However, an official End Of Life date of January 1, 2020 has been established for Python 2, after which time it will no longer be maintained.
- Python is still maintained by a core development team at the Institute, and Guido is still in charge, having been given the title of BDFL (Benevolent Dictator For Life) by the Python community. The name Python, by the way, derives not from the snake, but from the British comedy troupe Monty Python's Flying Circus, of which Guido was, and presumably still is, a fan. It is common to find references to Monty Python sketches and movies scattered throughout the Python documentation.
- It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages

- Python was designed to for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.

- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Python is Interpreted

- Many languages are compiled, meaning the source code you create needs to be translated into machine code, the language of your computer's processor, before it can be run. Programs written in an interpreted language are passed straight to an interpreter that runs them directly.
- This makes for a quicker development cycle because you just type in your code and run it, without the intermediate compilation step.
- One potential downside to interpreted languages is execution speed. Programs that are compiled into the native language of the computer processor tend to run more quickly than interpreted programs. For some applications that are particularly computationally intensive, like graphics processing or intense number crunching, this can be limiting.
- In practice, however, for most programs, the difference in execution speed is measured in milliseconds, or seconds at most, and not appreciably noticeable to a human user. The expediency of coding in an interpreted language is typically worth it for most applications.
- For all its syntactical simplicity, Python supports most constructs that would be expected in a very high-level language, including complex dynamic data types, structured and functional programming, and object-oriented programming.
- Additionally, a very extensive library of classes and functions is available that provides capability well beyond what is built into the language, such as database manipulation or GUI programming.
- Python accomplishes what many programming languages don't: the language itself is simply designed, but it is very versatile in terms of what you can accomplish with it.

Naive Bayes Algorithm

Introduction

What will you do? You have hundreds of thousands of data points and quite a few variables in your training data set. In such situation, if I were at your place, I would have used ‘**Naive Bayes**’, which can be extremely fast relative to other classification algorithms. It works on Bayes theorem of probability to predict the class of unknown data set.

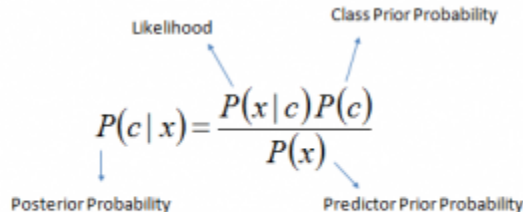
So that next time when you come across large data sets, you can bring this algorithm to action.

Naive Bayes algorithm

It is a classification technique based on Bayes’ Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as ‘Naive’.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability $P(c|x)$ from $P(c)$, $P(x)$ and $P(x|c)$. Look at the equation below:



The diagram shows the equation $P(c|x) = \frac{P(x|c)P(c)}{P(x)}$ with four labels and arrows: 'Likelihood' points to $P(x|c)$, 'Class Prior Probability' points to $P(c)$, 'Posterior Probability' points to $P(c|x)$, and 'Predictor Prior Probability' points to $P(x)$.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$
$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$$

Above,

- $P(c|x)$ is the posterior probability of class (c, target) given predictor (x, attributes).
- $P(c)$ is the prior probability of class.
- $P(x|c)$ is the likelihood which is the probability of predictor given class.

- $P(x)$ is the prior probability of predictor.

Naive Bayes algorithm works

Let's understand it using an example. Below I have a training data set of weather and corresponding target variable 'Play' (suggesting possibilities of playing). Now, we need to classify whether players will play or not based on weather condition. Let's follow the below steps to perform it.

Step 1: Convert the data set into a frequency table

Step 2: Create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

Step 3: Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

Problem: Players will play if weather is sunny. Is this statement is correct?

We can solve it using above discussed method of posterior probability.

$$P(\text{Yes} \mid \text{Sunny}) = P(\text{Sunny} \mid \text{Yes}) * P(\text{Yes}) / P(\text{Sunny})$$

$$\text{Here we have } P(\text{Sunny} \mid \text{Yes}) = 3/9 = 0.33, P(\text{Sunny}) = 5/14 = 0.36, P(\text{Yes}) = 9/14 = 0.64$$

$$\text{Now, } P(\text{Yes} \mid \text{Sunny}) = 0.33 * 0.64 / 0.36 = 0.60, \text{ which has higher probability.}$$

Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple classes.

The Pros and Cons of Naive Bayes

Pros:

- It is easy and fast to predict class of test data set. It also perform well in multi class prediction
- When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
- It performs well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

Cons:

- If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as “Zero Frequency”. To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
- On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict_proba are not to be taken too seriously.
- Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

Four Applications of Naive Bayes Algorithms

- **Real time Prediction:** Naive Bayes is an eager learning classifier and it is sure fast. Thus, it could be used for making predictions in real time.
- **Multi class Prediction:** This algorithm is also well known for multi class prediction feature. Here we can predict the probability of multiple classes of target variable.
- **Text classification/ Spam Filtering/ Sentiment Analysis:** Naive Bayes classifiers mostly used in text classification (due to better result in multi class problems and independence rule) have higher success rate as compared to other algorithms. As a result, it is widely used in Spam filtering (identify spam e-mail) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)

- **Recommendation System:** Naive Bayes Classifier and Collaborative Filtering together builds a Recommendation System that uses machine learning and data mining techniques to filter unseen information and predict whether a user would like a given resource or not

Tips to improve the power of Naive Bayes Model

Here are some tips for improving power of Naive Bayes Model:

- If continuous features do not have normal distribution, we should use transformation or different methods to convert it in normal distribution.
- If test data set has zero frequency issue, apply smoothing techniques “Laplace Correction” to predict the class of test data set.
- Remove correlated features, as the highly correlated features are voted twice in the model and it can lead to over inflating importance.
- Naive Bayes classifiers has limited options for parameter tuning like $\alpha=1$ for smoothing, `fit_prior=[True|False]` to learn class prior probabilities or not and some other options (look at detail here). I would recommend to focus on your pre-processing of data and the feature selection.
- You might think to apply some *classifier combination technique like* ensembling, bagging and boosting but these methods would not help. Actually, “ensembling, boosting, bagging” won’t help since their purpose is to reduce variance. Naive Bayes has no variance to minimize.

CHAPTER 4

4.1 Design and Implementation Constraints

4.1.1 Constraints in Analysis

- ◆ Constraints as Informal Text
- ◆ Constraints as Operational Restrictions
- ◆ Constraints Integrated in Existing Model Concepts
- ◆ Constraints as a Separate Concept
- ◆ Constraints Implied by the Model Structure

4.1.2 Constraints in Design

- ◆ Determination of the Involved Classes
- ◆ Determination of the Involved Objects
- ◆ Determination of the Involved Actions
- ◆ Determination of the Require Clauses
- ◆ Global actions and Constraint Realization

4.1.3 Constraints in Implementation

A hierarchical structuring of relations may result in more classes and a more complicated structure to implement. Therefore it is advisable to transform the hierarchical relation structure to a simpler structure such as a classical flat one. It is rather straightforward to transform the developed hierarchical model into a bipartite, flat model,

consisting of classes on the one hand and flat relations on the other. Flat relations are preferred at the design level for reasons of simplicity and implementation ease. There is no identity or functionality associated with a flat relation. A flat relation corresponds with the relation concept of entity-relationship modeling and many object oriented methods.

4.2 Other Nonfunctional Requirements

4.2.1 Performance Requirements

The application at this side controls and communicates with the following three main general components.

- embedded browser in charge of the navigation and accessing to the web service;
- Server Tier: The server side contains the main parts of the functionality of the proposed architecture. The components at this tier are the following.

Web Server, Security Module, Server-Side Capturing Engine, Preprocessing Engine, Database System, Verification Engine, Output Module.

4.2.2 Safety Requirements

1. The software may be safety-critical. If so, there are issues associated with its integrity level
2. The software may not be safety-critical although it forms part of a safety-critical system. For example, software may simply log transactions.
3. If a system must be of a high integrity level and if the software is shown to be of that integrity level, then the hardware must be at least of the same integrity level.
4. There is little point in producing 'perfect' code in some language if hardware and system software (in widest sense) are not reliable.

5. If a computer system is to run software of a high integrity level then that system should not at the same time accommodate software of a lower integrity level.
6. Systems with different requirements for safety levels must be separated.
7. Otherwise, the highest level of integrity required must be applied to all systems in the same environment.

CHAPTER 5

5.1 Architecture Diagram:

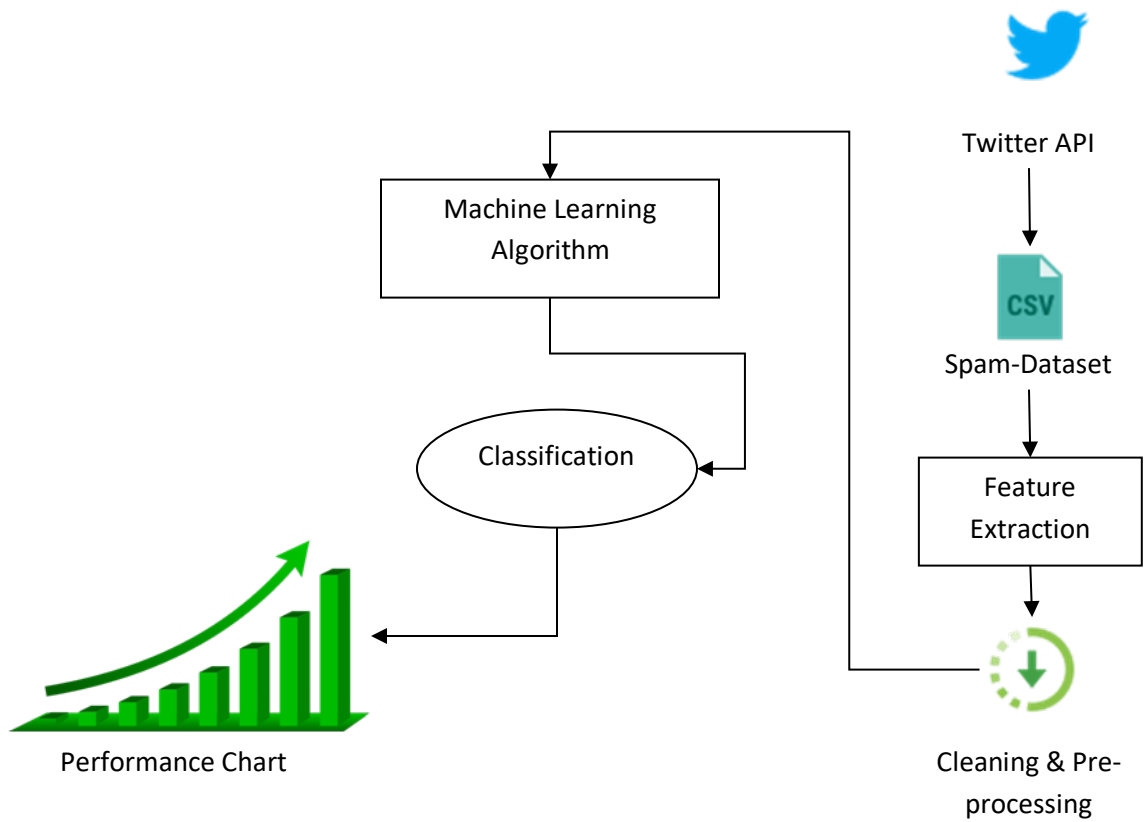
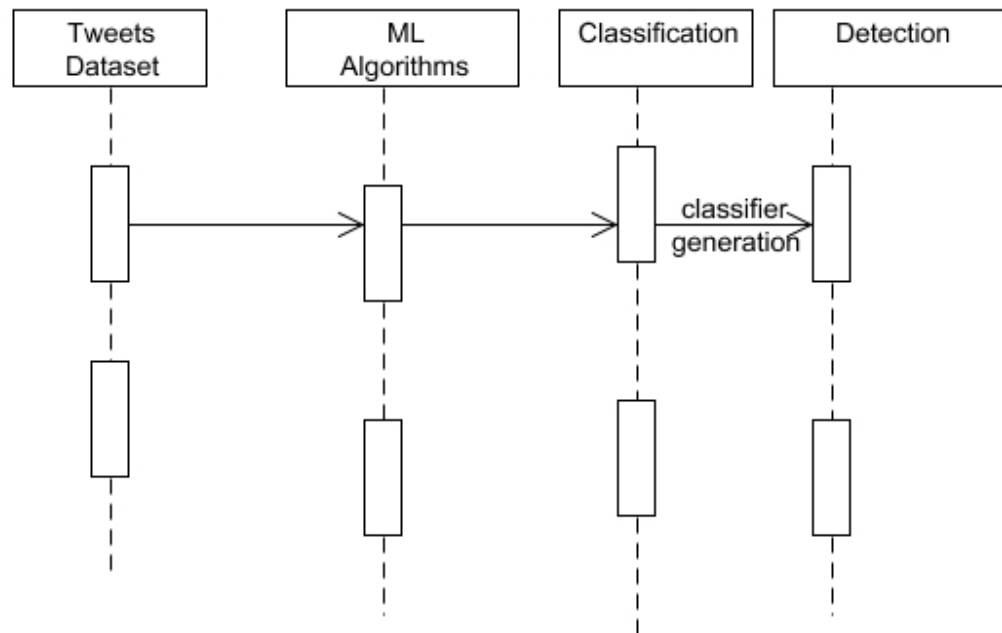


Fig: 5.1

5.2 Sequence Diagram:

A Sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of Message Sequence diagrams are sometimes called event diagrams, event sceneries and timing diagram.



5.3 Use Case Diagram:

Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of software engineering. The standard is managed and was created by the Object Management Group. UML includes a set of graphic notation techniques to create visual models of software intensive systems. This language is used to specify, visualize, modify, construct and document the artifacts of an object oriented software intensive system under development.

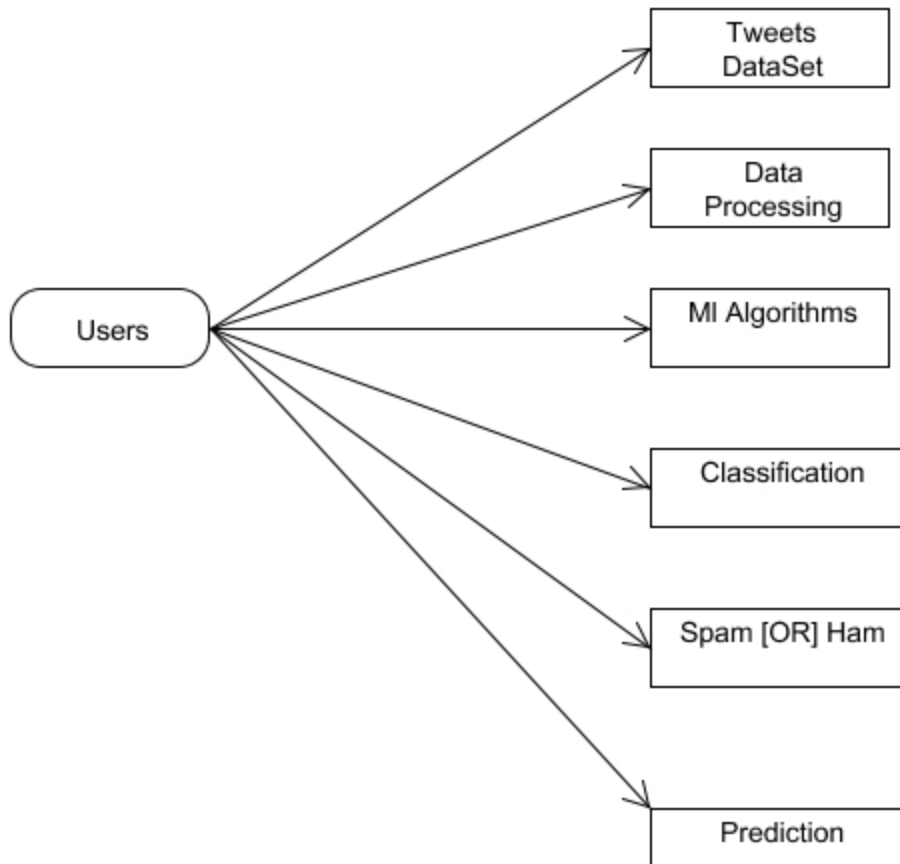
5.3.1. USECASE DIAGRAM

A Use case Diagram is used to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases.

Use case diagram consists of two parts:

Use case: A use case describes a sequence of actions that provided something of measurable value to an actor and is drawn as a horizontal ellipse.

Actor: An actor is a person, organization or external system that plays a role in one or more interaction with the system.

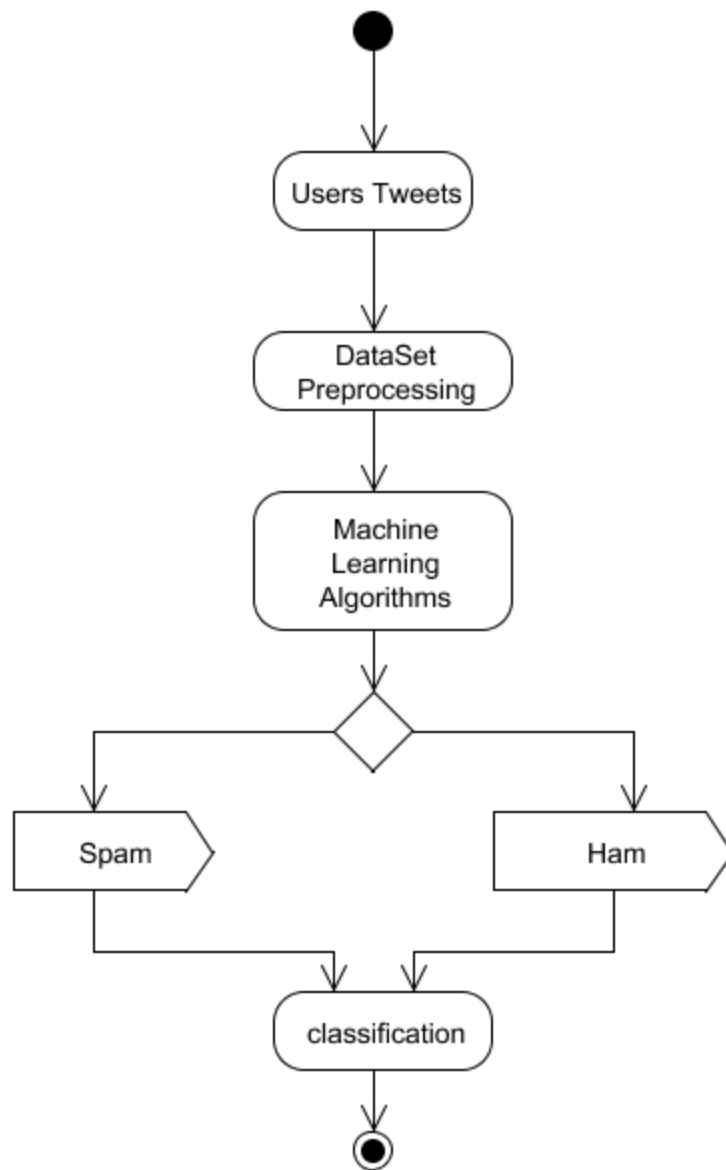


5.4 Activity Diagram:

Activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows the overall flow of control.

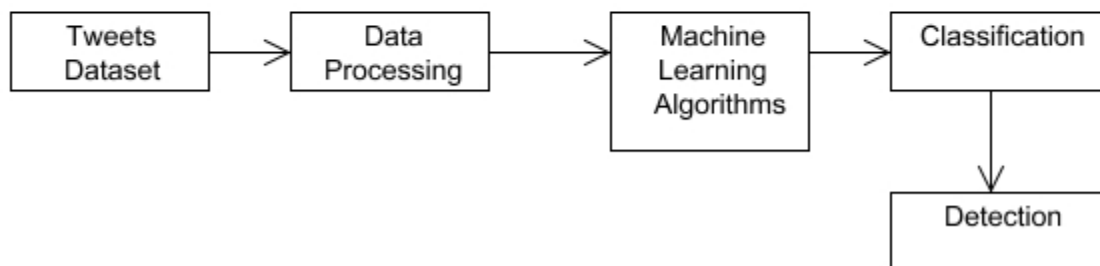
The most important shape types:

- Rounded rectangles represent activities.
- Diamonds represent decisions.
- Bars represent the start or end of concurrent activities.
- A black circle represents the start of the workflow.
- An encircled circle represents the end of the workflow.



5.5 Collaboration Diagram:

UML Collaboration Diagrams illustrate the relationship and interaction between software objects. They require use cases, system operation contracts and domain model to already exist. The collaboration diagram illustrates messages being sent between classes and objects.



CHAPTER 6

6.1 MODULES

- Data Preprocessing
- Machine Learning Mechanism
- Performance Statistics

6.2 MODULE EXPLANATION:

6.2.1 Data Preprocessing ÷

The first and foremost step in data processing is collecting the dataset. We have collected a dataset based on Twitter spam data. The dataset is a CSV file format data which consists of n number of Twitter spam Data. We need to select or extract the features from the collected dataset. Then the Data Cleaning should be initiated. Thus in this module data preprocessing will be completed.

Machine Learning Mechanism:

We evaluated our detection scheme based on the suggested user and content-based features. Our results show that among the four classifiers

- Decision Trees Classifier
- Support Vector Classifier
- Random Forest Classifier
- Naive Bayes algorithm

Performance Statistics:

This project results show that among the four classifiers we evaluated, the Naive Bayes algorithm produces the best results.

CHAPTER 7

CODING AND TESTING

7.1 CODING

Once the design aspect of the system is finalized the system enters into the coding and testing phase. The coding phase brings the actual system into action by converting the design of the system into the code in a given programming language. Therefore, a good coding style has to be taken whenever changes are required it easily screwed into the system.

7.2 CODING STANDARDS

Coding standards are guidelines to programming that focuses on the physical structure and appearance of the program. They make the code easier to read, understand and maintain. This phase of the system actually implements the blueprint developed during the design phase. The coding specification should be in such a way that any programmer must be able to understand the code and can bring about changes whenever felt necessary. Some of the standard needed to achieve the above-mentioned objectives are as follows:

Program should be simple, clear and easy to understand.

Naming conventions

Value conventions

Script and comment procedure

Message box format

7.2.1 NAMING CONVENTIONS

Naming conventions of classes, data member, member functions, procedures etc., should be **self-descriptive**. One should even get the meaning and scope of the variable by its name. The conventions are adopted for **easy understanding** of the intended message by the user. So it is customary to follow the conventions. These conventions are as follows:

Class names

Class names are problem domain equivalence and begin with capital letter and have mixed cases.

Member Function and Data Member name

Member function and data member name begins with a lowercase letter with each subsequent letters of the new words in uppercase and the rest of letters in lowercase.

7.2.2 VALUE CONVENTIONS

Value conventions ensure values for variable at any point of time. This involves the following:

- Proper default values for the variables.
- Proper validation of values in the field.
- Proper documentation of flag values.

7.2.3 SCRIPT WRITING AND COMMENTING STANDARD

Script writing is an art in which indentation is utmost important. Conditional and looping statements are to be properly aligned to facilitate easy understanding. Comments are included to minimize the number of surprises that could occur when going through the code.

7.2.4 MESSAGE BOX FORMAT

When something has to be prompted to the user, he must be able to understand it properly. To achieve this, a specific format has been adopted in displaying messages to the user. They are as follows:

- X – User has performed illegal operation.
- ! – Information to the user.

7.3 TEST PROCEDURE

SYSTEM TESTING

Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example the design must not have any logic faults in the design is detected before coding commences, otherwise the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as walkthrough.

Testing is one of the important steps in the software development phase. Testing checks for the errors, as a whole of the project testing involves the following test cases:

- Static analysis is used to investigate the structural properties of the Source code.
- Dynamic testing is used to investigate the behavior of the source code by executing the program on the test data.

7.4 TEST DATA AND OUTPUT

7.4.1 UNIT TESTING

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

7.4.2 FUNCTIONAL TESTS

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values and special values, such as logically related inputs, files of identical elements, and empty files.

Three types of tests in Functional test:

- Performance Test
- Stress Test
- Structure Test

7.4.3 PERFORMANCE TEST

It determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit.

7.4.4 STRESS TEST

Stress Test is those test designed to intentionally break the unit. A Great deal can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unit breaks.

7.4.5 STRUCTURED TEST

Structure Tests are concerned with exercising the internal logic of a program and traversing particular execution paths. The way in which White-Box test strategy was employed to ensure that the test cases could Guarantee that all independent paths within a module have been have been exercised at least once.

- Exercise all logical decisions on their true or false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structures to assure their validity.
- Checking attributes for their correctness.
- Handling end of file condition, I/O errors, buffer problems and textual errors in output information

7.4.6 INTEGRATION TESTING

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The

objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that, interface dispenses can be easily found and corrected.

The major error that was faced during the project is linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

7.5 TESTING TECHNIQUES / TESTING STRATERGIES

7.5.1 TESTING

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet –undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise the program or the project is not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that as a probability of finding an yet undiscovered error. A successful test is one that uncovers an yet undiscovered error. Any engineering product can be tested in one of the two ways:

7.5.1.1 WHITE BOX TESTING

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis path testing:

- Flow graph notation
- Cyclometric complexity
- Deriving test cases
- Graph matrices Control

7.5.1.2 BLACK BOX TESTING

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

- Graph based testing methods
- Equivalence partitioning
- Boundary value analysis
- Comparison testing

7.5.2 SOFTWARE TESTING STRATEGIES:

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

- Testing begins at the module level and works “outward” toward the integration of the entire computer based system.
- Different testing techniques are appropriate at different points in time.
- The developer of the software and an independent test group conducts testing.
- Testing and Debugging are different activities but debugging must be accommodated in any testing strategy.

7.5.2.1 INTEGRATION TESTING:

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together. The problem of course, is “putting them together”- interfacing. There may be the chances of data lost across on another’s sub functions, when combined may not produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; global data structures can present problems.

7.5.2.2 PROGRAM TESTING:

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that in violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax error. These errors are shown through error messages generated by the computer. A logic error on the other hand deals with the incorrect data fields, out-off-range items and invalid combinations. Since the compiler s will not deduct logical error, the programmer must examine the output. Condition testing exercises the logical conditions contained in a module. The possible types of elements in a condition include a Boolean operator, Boolean variable, a pair of Boolean parentheses A relational operator or on arithmetic expression. Condition testing method focuses on testing each condition in the program the purpose of condition test is to deduct not only errors in the condition of a program but also other a errors in the program.

7.5.2.3 SECURITY TESTING:

Security testing attempts to verify the protection mechanisms built in to a system well, in fact, protect it from improper penetration. The system security must be tested for invulnerability

from frontal attack must also be tested for invulnerability from rear attack. During security, the tester places the role of individual who desires to penetrate system.

7.5.2.4 VALIDATION TESTING

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.

- * The function or performance characteristics confirm to specifications and are accepted.
- * A validation from specification is uncovered and a deficiency created.

Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily. Though there were deficiencies in the system they were not catastrophic

7.5.2.5 USER ACCEPTANCE TESTING

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required. This is done in regarding to the following points.

- Input screen design.
- Output screen design.

Source Code

```
module1:
```

```
import re
```

```
import sys
```

```
from tweepy import OAuthHandler
```

```
import tweep
```

```
def text_clean(text):
```

```
    remove_space = '\s+'
```

```
    find_url = ('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|'
```

```
        '![*\\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+')'
```

```
    find_mention = '@[\\w\\-]+''
```

```
    process_text = re.sub(remove_space, ' ', text)
```

```
process_text = re.sub(find_url, '', process_text)

process_text = re.sub(find_mention, '', process_text)

process_text = process_text.encode('ascii','ignore')

return process_text
```

```
con_key = 'PG2atN7eFZIfjqPugmrQ0DTHN'
```

```
con_secret =
```

```
'BcKjeAsjePUjw5SqOj7DqQy7QgTvNo6fbNXoT6PHoV68CwqWSX'
```

```
acc_token = '93137600-fD6r6VCcwx2f4ahKrqTIImxhFicNYcpy17HkL8Ne4'
```

```
acc_token_secret = 'lYUwaYoI6j5nHk7pijChJUxzDuY99osg64rs9vjhOejWU'
```

```
try:
```

```
    auth = OAuthHandler(con_key, con_secret)
```

```
    auth.set_access_token(acc_token, acc_token_secret)
```

```
    api = tweepy.API(auth)
```

```
except:
```

```
    print("Error: Twitter Not Connected")
```

```

if len (sys.argv) != 2 :

    print ("Usage: python fetchTweet.py <search text>")

    sys.exit (1)

searchTxt = sys.argv[1]

query = "#" + searchTxt

count = 40

tweets = []

clean_tweets=[]

try:

    fetchd_tweets = api.search(q = query, count = count)

    for tweet in fetchd_tweets:

        parsed_tweet = {}

        ss=tweet.text

        parsed_tweet['text'] = ss#clean_tweet(ss)

        clean_tweets.append(text_clean(ss))

```

```

        tweets.append(parsed_tweet)

except tweepy.TweepError as e:

    print("Error : " + str(e))


print ("*****")

print ("Total tweets read from twitter server ")

print (len(tweets))


import pandas as pd

df = pd.DataFrame(tweets)

df.to_csv("tweets.csv")


df = pd.DataFrame(clean_tweets)

df.to_csv("clean_tweets.csv")


module2:

#!/usr/bin/env python

# coding: utf-8

```

Installing Dependencies

In[1]:

get_ipython().run_line_magic('matplotlib', 'inline')

import matplotlib.pyplot as plt

import csv

import pandas

import sklearn

import pickle

from wordcloud import WordCloud

import pandas as pd

import numpy as np

import nltk

from nltk.corpus import stopwords

**from sklearn.feature_extraction.text import CountVectorizer,
TfidfTransformer**

```
from sklearn.tree import DecisionTreeClassifier  
  
#from sklearn.learning_curve import learning_curve  
  
from sklearn.model_selection import learning_curve, GridSearchCV
```

```
# Importing the Dataset spam.csv
```

```
# In[2]:
```

```
data = pd.read_csv('./data/spam.csv', encoding='latin-1')  
  
data.head()
```

```
# Removing unwanted columns
```

```
# In[3]:
```

```
data = data.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1)
```

```
data = data.rename(columns={"v2" : "text", "v1":"label"})
```

```
# In[4]:
```

```
data[1990:2000]
```

```
# In[5]:
```

```
data['label'].value_counts()
```

```
# In[6]:
```

```
data['length'] = data['text'].map(lambda x: len(x))
```

```
data.hist(column='length',bins=50, figsize=(10,7))
```

```
# ### Some Visualisations
```

```
# In[7]:
```

```
data.hist(column='length', by='label', bins=100, figsize=(20,7))
```

```
# In[8]:
```

```
import nltk
```

```
nltk.download("punkt")
```

```
import warnings
```

```
warnings.filterwarnings('ignore')
```

```
# ### WordClouds- to see which words are common in SPAM and NOT SPAM  
mesaages
```

```
# In[9]:
```

```
ham_words = "
```

```
spam_words = "
```

```
# In[10]:
```

```
for val in data[data['label'] == 'spam'].text:
```

```
    text = val.lower()
```

```
    tokens = nltk.word_tokenize(text)
```

```
    for words in tokens:
```

```
spam_words = spam_words + words + ' '
```

```
for val in data[data['label'] == 'ham'].text:
```

```
    text = val.lower()
```

```
    tokens = nltk.word_tokenize(text)
```

```
    for words in tokens:
```

```
        ham_words = ham_words + words + ' '
```

```
# In[11]:
```

```
spam_wordcloud = WordCloud(width=500,  
height=300).generate(spam_words)
```

```
ham_wordcloud = WordCloud(width=500, height=300).generate(ham_words)
```

```
# In[12]:
```

#Spam Word cloud

```
plt.figure( figsize=(10,8), facecolor='w')
```

```
plt.imshow(spam_wordcloud)
```

```
plt.axis("off")
```

```
plt.tight_layout(pad=0)
```

```
plt.show()
```

In[13]:

```
plt.figure( figsize=(10,8), facecolor='g')
```

```
plt.imshow(ham_wordcloud)
```

```
plt.axis("off")
```

```
plt.tight_layout(pad=0)
```

```
plt.show()
```

```
# In[14]:
```

```
data = data.replace(['ham','spam'],[0, 1])
```

```
# In[15]:
```

```
data.head(10)
```

```
# ### Removing Stopwords from the messages
```

```
# In[16]:
```

```
import string
```

```
def text_process(text):
```

```
text = text.translate(str.maketrans('', '', string.punctuation))
```

```
text = [word for word in text.split() if word.lower() not in  
stopwords.words('english')]
```

```
return " ".join(text)
```

```
# In[17]:
```

```
data['text'] = data['text'].apply(text_process)
```

```
# In[18]:
```

```
data.head()
```

```
# In[19]:
```

```
text = pd.DataFrame(data['text'])
```

```
label = pd.DataFrame(data['label'])
```

```
# In[20]:
```

```
## Counting how many times a word appears in the dataset
```

```
from collections import Counter
```

```
total_counts = Counter()
```

```
for i in range(len(text)):
```

```
    for word in text.values[i][0].split(" "):
```

```
        total_counts[word] += 1
```

```
print("Total words in data set: ", len(total_counts))
```

```
# In[21]:
```

```
## Sorting in decreasing order (Word with highest frequency appears first)
```

```
vocab = sorted(total_counts, key=total_counts.get, reverse=True)
```

```
print(vocab[:60])
```

```
# In[22]:
```

```
# Mapping from words to index
```

```
vocab_size = len(vocab)
```

```
word2idx = {}
```

```

#print vocab_size

for i, word in enumerate(vocab):

    word2idx[word] = i


# In[23]:


### Text to Vector

def text_to_vector(text):

    word_vector = np.zeros(vocab_size)

    for word in text.split(" "):

        if word2idx.get(word) is None:

            continue

        else:

            word_vector[word2idx.get(word)] += 1

    return np.array(word_vector)

```

In[24]:

Convert all titles to vectors

word_vectors = np.zeros((len(text), len(vocab)), dtype=np.int_)

for ii, (_, text_) in enumerate(text.iterrows()):

word_vectors[ii] = text_to_vector(text_[0])

In[25]:

word_vectors.shape

Converting words to vectors using TFIDF Vectorizer

In[26]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
vectorizer = TfidfVectorizer()
```

```
vectors = vectorizer.fit_transform(data['text'])
```

```
vectors.shape
```

```
# In[27]:
```

```
#features = word_vectors
```

```
features = vectors
```

```
# ### Splitting into training and test set
```

```
# In[28]:
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(features, data['label'],  
test_size=0.15, random_state=111)
```

```
# In[29]:
```

```
print (X_train.shape)
```

```
print (X_test.shape)
```

```
print (y_train.shape)
```

```
print (y_test.shape)
```

```
# ## Classifying using sklearn pre built classifiers
```

```
# In[30]:
```

```
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.svm import SVC
```

```
from sklearn.naive_bayes import MultinomialNB
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.metrics import accuracy_score
```

```
# In[31]:
```

```
svc = SVC(kernel='sigmoid', gamma=1.0)
```

```
knc = KNeighborsClassifier(n_neighbors=49)
```

```
mnb = MultinomialNB(alpha=0.2)
```

```
dtc = DecisionTreeClassifier(min_samples_split=7, random_state=111)
```

```
lrc = LogisticRegression(solver='liblinear', penalty='l1')
```

```
rfc = RandomForestClassifier(n_estimators=31, random_state=111)
```

In[32]:

```
clfs = {'SVC' : svc, 'KN' : knn, 'NB': mnb, 'DT': dtc, 'LR': lrc, 'RF': rfc}
```

In[33]:

```
def train(clf, features, targets):
```

```
    clf.fit(features, targets)
```

```
def predict(clf, features):
```

```
    return (clf.predict(features))
```

In[34]:

```

pred_scores_word_vectors = []

for k,v in clfs.items():

    train(v, X_train, y_train)

    pred = predict(v, X_test)

    pred_scores_word_vectors.append((k, [accuracy_score(y_test , pred)]))

```

Predictions using custom vocabulary

In[35]:

```

predictions_word_vectors                                     =
pd.DataFrame.from_items(pred_scores_word_vectors,orient='index',
columns=['Score_'])

predictions_word_vectors

```

```
# ## Plot of scores custom vocabulary
```

```
# In[36]:
```

```
predictions_word_vectors.plot(kind='bar',    ylim=(0.85,1.0),    figsize=(9,6),  
align='center', colormap="Accent")
```

```
plt.xticks(np.arange(6), predictions_word_vectors.index)
```

```
plt.ylabel('Accuracy Score')
```

```
plt.title('Distribution by Classifier - Word Vectors')
```

```
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

```
# In[37]:
```

```
def find(p):
```

```
    if p == 1:
```

```
print ("Message is SPAM")
```

```
else:
```

```
print ("Message is NOT Spam")
```

```
# In[38]:
```

```
text = ["Free tones Hope you enjoyed your new content"]
```

```
integers = vectorizer.transform(text)
```

```
# In[39]:
```

```
p = mnbc.predict(integers)[0]
```

```
find(p)
```

In[40]:

```
text = ["good"]
```

```
integers = vectorizer.transform(text)
```

In[41]:

```
q = mnw.predict(integers)[0]
```

```
find(q)
```

Saving the Model

In[47]:

```
from sklearn.externals import joblib
```

```
joblib.dump(mnb, 'model.pkl')
```

```
print ("Model Saved")
```

```
# In[48]:
```

```
best = joblib.load('model.pkl')
```

```
# In[ ]:
```

```
module3:
```

```
from flask import Flask ,render_template,request,jsonify,session
```

```
from flask import Flask, render_template, url_for, request
```

```
import pandas as pd

import pickle

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.naive_bayes import MultinomialNB

from sklearn.externals import joblib

import sqlite3 as sql

import base64

import pandas as pd

from sklearn.preprocessing import LabelEncoder

#from flask_bootstrap import Bootstrap

import numpy as np

from sklearn.utils import shuffle

import os

from flask import Flask, render_template, request, url_for, send_from_directory

import os

#from geo import getTweetLocation
```

```

app = Flask(__name__)

app.secret_key = 'any random string'

PEOPLE_FOLDER = os.path.join('static', 'people_photo')

@app.route('/', methods=['GET', 'POST'])

def home():

    return render_template('index.html')


def validate(username,password):

    con = sql.connect('static/chat.db')

    completion = False

    with con:

        cur = con.cursor()

        cur.execute('SELECT * FROM persons')

        rows = cur.fetchall()

        for row in rows:

            dbuser = row[1]

            dbpass = row[2]

            if dbuser == username:

```

```
        completion = (dbpass == password)

    return completion
```

```
@app.route('/login', methods=['GET', 'POST'])
```

```
def login():
```

```
    error = None
```

```
    if request.method == 'POST':
```

```
        username = request.form['username']
```

```
        password = request.form['password']
```

```
        completion = validate(username,password)
```

```
        if completion == False:
```

```
            error = 'invalid Credentials. please try again.'
```

```
        else:
```

```
            session['username'] = request.form['username']
```

```
            return render_template('index.html')
```

```
    return render_template('search.html', error=error)
```

```

@app.route('/register', methods = ['GET','POST'])

def register():

    if request.method == 'POST':

        try:

            name = request.form['name']

            username = request.form['username']

            password = request.form['password']

            with sql.connect("static/chat.db") as con:

                cur = con.cursor()

                cur.execute("INSERT INTO persons(name,username,password)
VALUES (?,?)" ,(name,username,password))

                con.commit()

                msg = "Record successfully added"

        except:

            con.rollback()

            msg = "error in insert operation"

    finally:

```

```

        return render_template("index.html",msg = msg)

    con.close()

    return render_template('register.html')


@app.route('/list')

def list():

    con = sql.connect("static/chat.db")

    con.row_factory = sql.Row

    cur = con.cursor()

    cur.execute("select * from persons")

    rows = cur.fetchall();

    return render_template("list.html",rows = rows)

app.config['UPLOAD_FOLDER'] = PEOPLE_FOLDER


@app.route('/predict', methods=['POST'])

```

```

def predict():

    df = pd.read_csv("spam.csv", encoding="latin-1")

    #df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], axis=1, inplace=True)

    # Features and Labels

    df['label'] = df['class'].map({'ham': 0, 'spam': 1})

    X = df['message']

    y = df['label']


    # Extract Feature With CountVectorizer

    cv = CountVectorizer()

    X = cv.fit_transform(X) # Fit the Data

    from sklearn.model_selection import train_test_split

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
random_state=42)

    # Naive Bayes Classifier

    from sklearn.naive_bayes import MultinomialNB

    clf = MultinomialNB()

    clf.fit(X_train, y_train)

```

```
clf.score(X_test, y_test)
```

```
# Alternative Usage of Saved Model
```

```
# joblib.dump(clf, 'NB_spam_model.pkl')
```

```
# NB_spam_model = open('NB_spam_model.pkl','rb')
```

```
# clf = joblib.load(NB_spam_model)
```

```
if request.method == 'POST':
```

```
    message = request.form['message']
```

```
    data = [message]
```

```
    vect = cv.transform(data).toarray()
```

```
    my_prediction = clf.predict(vect)
```

```
    if int(my_prediction)==1:
```

```
        prediction='Spam'
```

```
        return render_template('result2.html', prediction=prediction)
```

```
    else:
```

```
        prediction='Not a Spam (It is a Ham)'
```

```
        return render_template("result3.html",prediction=prediction)
```

```
if __name__ == '__main__':
```

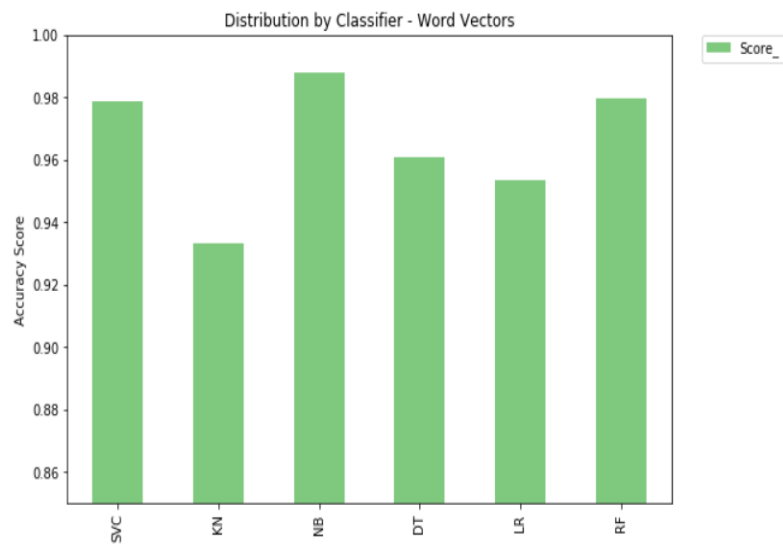
```
app.run(debug= )
```

Screenshots:

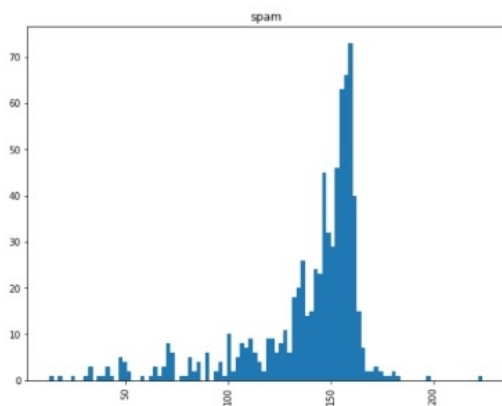
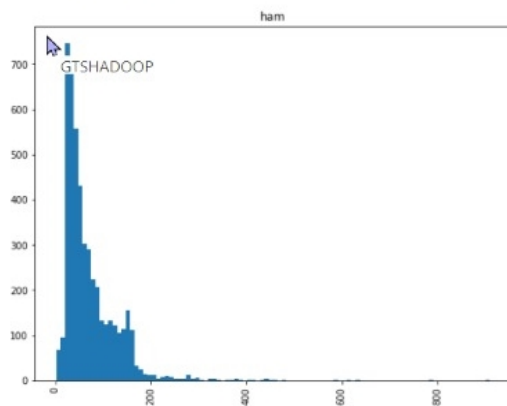
Clean the tweets

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
	0																				
	0	b'RT : #TDP supremo and leader of opposition N #ChandrababuNaidu slammed the #JaganMohanReddy government for betraying farmers'																			
	1	b' ...I #CMYSJagan #YSJaganEmpowersWomen #ysjagan #ysrcp '																			
	2	b'! #APAssembly #CMYSJagan #ysjagan #ysrcp #ysjaganhmohreddy #andhrapradesh #ysr '																			
	3	b' , '																			
	4	b' #amaravati #andhrapradesh '																			
	5	b'TDP leader slams Jagan Reddy's advisor for remarks over Amaravati farmers' agitation #TDP #JaganMohanReddy #YSRCP "																			
	6	b' - #APPolitics, #cmysjagan, #devineniurna, #jaganhmohreddy, '																			
	7	b'RT : , #'																			
	8	b'RT : #TDP supremo and leader of opposition N #ChandrababuNaidu slammed the #JaganMohanReddy government for betraying farmers'																			
	9	b'RT : #TDP supremo and leader of opposition N #ChandrababuNaidu slammed the #JaganMohanReddy government for betraying farmers'																			
	10	b' , '																			
	11	b'#TDP supremo and leader of opposition N #ChandrababuNaidu slammed the #JaganMohanReddy government for betraying f '																			
	12	b'LIVE : YCP 9 ,49 Actor Shivaji ABN Watch Live Here - '																			
	13	b' ' ! #jaganhmohreddy '																			
	14	b'# # ... # # # # # # # , '																			
	15	b'RT : BEST CM OF ANDHRA PRADESH... #YSJaganMarkGovernance #TDPTwitter #JaganMohanReddy #CBN Retweet '																			
	16	b'"In a setback to #JaganMohanReddy-led government's plans to develop three state capitals, the #AndhraPradesh High Co "																			
	17	b'"In a setback to #JaganMohanReddy-led government's plans to develop three state capitals, the #AndhraPradesh "																			
	18	b'AP High Court orders Amaravati Masterplan to be implemented #amaravati #amaravatimasterplan #andhrapradeshhc '																			
	19	b'RT : Parents Association requests #ChiefMinister #JaganMohanReddy to evacuate students from #Belarus as well '																			
	20	b'Parents Association requests #ChiefMinister #JaganMohanReddy to evacuate students from #Belarus as well '																			
	21	b'RT : #APYouthWantJobNotifications #Jagan #JaganMohanReddy ... '																			
	22	b'#NitishKumar #Vijay Kumar Choudhary #NarendraModi #Jaganhmohreddy #manishkashyap #sudhirchoudhary '																			
	23	b'#APCMYSJagan #apcm #jaganhmohreddy #Help #UkraineRussiaWar sir please help chesi yelaga ay '																			
	24	b'Honble Chief Minister dedicates ins Visakhapatnam to the city of destiny #AndhraPradesh '																			
	25	b' Vs #JaganMohanreddy #Pavankalyan #Bheemlanaik '																			

Comparison of among the four Classifiers



In []:



Predict the tweet with Spam or Ham

Spam Detector For TWITTER Tweets

A Machine Learning SPAM detection APP with Flask

Enter Your Tweet Here

Predict



Spam word



Ham word



REFERENCES

- [1] H. Tsukayama, Twitter Turns 7: Users Send Over 400 Million Tweets Per Day. Washington, DC, USA: Washington Post, Mar. 2013.
- [2] F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida, “Detecting spammers on Twitter,” in Proc. 7th Annu. Collaboration, Electron. Messaging, Anti Abuse Spam Conf., Redmond, WA, USA, Jul. 2010.
- [3] Z. Miller, B. Dickinson, W. Deitrick, W. Hu, and A. H. Wang, “Twitter spammer detection using data stream clustering,” *Inf. Sci.*, vol. 260, pp. 64–73, Mar. 2014.
- [4] C. Chen et al., “A performance evaluation of machine learning-based streaming spam tweets detection,” *IEEE Trans. Comput. Social Syst.*, vol. 2, no. 3, pp. 65–76, Sep. 2015.
- [5] X. Zheng, Z. Zeng, Z. Chen, Y. Yu, and C. Rong, “Detecting spammers on social networks,” *Neurocomputing*, vol. 159, pp. 27–34, Jul. 2015.
- [6] A. Mukherjee et al., “Spotting opinion spammers using behavioral footprints,” in Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Chicago, IL, USA, 2013, pp. 632–640.
- [7] O. Kurasova, V. Marcinkevicius, V. Medvedev, A. Rapecka, and P. Stefanovic, “Strategies for big data clustering,” in Proc. IEEE 26th Int. Conf. Tools Artif. Intell., Nov. 2014, pp. 740–747.

- [8] M. Taheri and R. Boostani, “Novel auxiliary techniques in clustering,” in Proc. World Congr. Eng., London, U.K., 2007.
- [9] H. Tajalizadeh and R. Boostani, “A Novel Clustering Framework for Stream Data Un nouveau cadre de classifications pour les données de flux,” Can. J. Elect. Comput. Eng., vol. 42, no. 1, pp. 27–33, 2018.
- [10] F. Cao, M. Ester, W. Qian, and A. Zhou, “Density-based clustering over an evolving data stream with noise,” in Proc. SIAM Int. Conf. Data Mining, 2006, pp. 59–70.
- [11] A. H. Wang, “Don’t follow me: Spam detection in twitter,” in Proc. Int. Conf. Secur. Cryptogr. (SECRYPT), Jul. 2010, pp. 1–10.
- [12] G. Stringhini, C. Kruegel, and G. Vigna, “Detecting spammers on social networks,” in Proc. 26th Annu. Comput. Secur. Appl. Conf., Austin, TX, USA, 2010, pp. 1–9.
- [13] K. Lee, J. Caverlee, and S. Webb, “Uncovering social spammers: Social honeypots + machine learning,” in Proc. 33rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr., Geneva, Switzerland, 2010, pp. 435–442.
- [14] C. Yang, R. Harkreader, and G. Gu, “Empirical evaluation and new design for fighting evolving Twitter spammers,” IEEE Trans. Inf. Forensics Security, vol. 8, no. 8, pp. 1280–1293, Aug. 2013.

- [15] J. Song, S. Lee, and J. Kim, “Spam filtering in twitter using senderreceiver relationship,” in Proc. 14th Int. Conf. Recent Adv. Intrusion Detection, Menlo Park, CA, USA, 2011, pp. 301–317.
- [16] I. Inuwa-Dutse, M. Liptrott, and I. Korkontzelos, “Detection of spamposting accounts on Twitter,” *Neurocomputing*, vol. 315, pp. 496–511, Nov. 2018.
- [17] K. Lee, B. Eoff, and J. Caverlee, “Seven months with the devils: A longterm study of content polluters on twitter,” in Proc. ICWSM, 2011, pp. 1–8.
- [18] S. Sedhai and A. Sun, “Semi-supervised spam detection in Twitter stream,” *IEEE Trans. Comput. Social Syst.*, vol. 5, no. 1, pp. 169–175, Mar. 2018.
- [19] S. Sedhai and A. Sun, “HSpam14: A collection of 14 million tweets for hashtag-oriented spam research,” in Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr., Santiago, Chile, 2015, pp. 223–232.
- [20] N. Jindal and B. Liu, “Opinion spam and analysis,” in Proc. Int. Conf. Web Search Data Mining, Palo Alto, CA, USA, 2008, pp. 219–230.
- [21] M. Ott, Y. Choi, C. Cardie, and J. T. Hancock, “Finding deceptive opinion spam by any stretch of the imagination,” in Proc. 49th Annu. Meeting Assoc. Comput. Linguistics, Hum. Lang. Technol., vol. 1, Portland, Oregon, 2011, pp. 309–319.
- [22] T. Wu, S. Liu, J. Zhang, and Y. Xiang, “Twitter spam detection based on deep learning,” in Proc. Australas. Comput. Sci. Week Multiconf., Geelong, Australia, 2017, p. 3.

- [23] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, p. 436, 2015.
- [24] C. Yang, R. C. Harkreader, and G. Gu, “Die free or live hard? Empirical evaluation and new design for fighting evolving twitter spammers,” in *Proc. 14th Int. Conf. Recent Adv. Intrusion Detection*, Menlo Park, CA, USA, 2011, pp. 318–337.
- [25] M. R. Ackermann, M. Mörtens, C. Raupach, K. Swierkot, C. Lammersen, and C. Sohler, “StreamKM++: A clustering algorithm for data streams,” in *Proc. 12th Workshop Algorithm Eng. Exp. (ALENEX)*, 2012, pp. 2–4.
- [26] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, “A framework for projected clustering of high dimensional data streams,” in *Proc. 13th Int. Conf. Very Large Data Bases*, vol. 30, Toronto, ON, Canada, 2004, pp. 852–863.
- [27] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining*, 1996, pp. 1–6.
- [28] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, “A framework for clustering evolving data streams,” in *Proc. 29th Int. Conf. Very Large Data Bases*, 2003, pp. 81–92.

- [29] S. Ren, Y. Lian, and X. Zou, “Incremental naïve Bayesian learning algorithm based on classification contribution degree,” *J. Comput.*, vol. 9, no. 8, pp. 1967–1974, 2014.
- [30] K. Lee, J. Caverlee, Z. Cheng, and D. Z. Sui, “Content-driven detection of campaigns in social media,” in *Proc. 20th ACM Int. Conf. Inf. Knowl. Manage.*, Glasgow, U.K., 2011, pp. 551–556.
- [31] G. Moise, J. Sander, and M. Ester, “P3C: A robust projected clustering algorithm,” in *Proc. 6th Int. Conf. Data Mining (ICDM)*, Dec. 2006, pp. 414–425.
- [32] G. Roffo, S. Melzi, U. Castellani, and A. Vinciarelli, “Infinite latent feature selection: A probabilistic latent graph-based ranking approach,” in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 1398–1406.