Spammer Detection from Twitter using Machine Learning

FINAL PROJECT REPORT

Submitted by

Radhesyam Saharan

M160579CA

In the partial fulfilment for the award of the degree of

MASTER OF COMPUTER APPLICATION

BATCH 2016-2019

Under the guidance of

Dr. Lijiya A

(Assistant professor , Dept. of Computer Science and Engineering)

Department of Computer Science and Engineering

NATIONAL INSTITUTE OF TECHNOLOGY,CALICUT

NIT CAMPUS PO KERALA INDIA, 673601

MAY 2019

Acknowledgment

I am grateful to all those who have contributed towards shaping this project. At the outset, I would like to express my sincere thanks to Assistant Professor **Dr. Lijiya A** for her advice during my project work. As my supervisor, she has constantly encouraged me to remain focused on achieving my goal. Her observation and comments helped me to establish the overall direction of the research and to move forward with an investigation in depth. She has helped me greatly and been a source of knowledge.

I would also like to express my gratitude towards my parents and classmates who provided me with a different insight into my ideas and willingly helped me with their abilities.

DECLARATION

"I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institutes of higher learning, except where due acknowledgment has been made in the text."

Place: Calicut **Signature**

Date: 16 May 2019 **Name: Radheshyam Saharan**

Reg.No: M160579CA

CERTIFICATE

*This is to certify that the project report "***Spammer detection from twitter using machine learning***" submitted by* ***Mr. Radheshyam Saharan*** *to the National Institute of Technology Calicut towards partial fulfillment of the requirements for the award of the Degree of Master of Computer Application is a bona fide record of the work carried out by him under our supervision and guidance.*

Dr. Lijiya A

Assistant Professor
Department of CSE NITC

Place:

Date:

Signature of Head of the Department

(*Office seal*)

Content

LIST OF FIGURES

LIST OF TABLES

1.Introduction

1.1. Problem Definition

Many researchers around the world are working to find a feasible solution to the problem of online impersonation and the use of spam users. In this project, we intend to give a framework using machine learning techniques with which the detection of spam users can be done so that the social life of people can become a little more secure.

1.2. Motivation

I have been in multiple instances wherein I read a review online for a movie or a restaurant which was not as suggested. These reviews are usually done by anonymous fake users reviewing places to promote it. This is a simple example that I have come across but there are similar issues where spam users spread false content online. This may also lead in harassment of a real user or try to retrieve account information of real users. These instances have been increasing day by day and thus require a solution.

2.Literature survey

With the wide spreading of Online Social Networks(OSNs), the privacy of the users involved in such services is becoming a major concern. Several researchers already proposed solutions to mitigate the privacy threat for those people having profiles in OSNs (which, for example, count more than 800 million for Facebook, one of the most popular OSN). As an example of such privacy-threat mitigation solutions, in [1] the authors proposed the concept of Virtual Private Social Network (VPSN). VPSN basically reflects the concept of Virtual Private Network (known in computer networks) within OSNs: only friends within the VPSN are able to see the real information of a person. Other people in the OSN, as well as the OSN manager, do not have the means to access the same information. Most of the work in the literature aimed at protecting the information the OSN is aware of—to be accessed only by the authorized people in an authorized way. For example, in [2] the authors show how an adversary might get access to the information that the victim shares in the profile, against the victim's wishes. While the problem of protecting the privacy of the information that a real person put on the OSN has been at least already considered, we believe that too little attention has been put in protecting the privacy of the people that might not even use at all a specific OSN. In fact, an adversary willing to get private information of a victim might run a "social engineering" type of attack. As an example, one might create the profile of the victim, and then try to get private information of the victim, while interacting with the victim's real friends connected to the fake profile. We refer to this malicious behavior as a Fake Profile Attack (FPA). In [3], the authors first demonstrated the feasibility of mounting identity threat attacks on OSN with two variants: single OSN and cross-sites OSNs. In the first case, the victim already has a profile in the OSN where the adversary will create the clone profile. In cross-site OSNs, the victim does not have a profile in the same OSN where the attack is run, but the profile of the victim exists in other OSNs. In [4], the authors present a detection framework based on profiles similarities: attribute and friends network similarity. In [4], Facebook is considered to be the OSN where the adversary runs the attack. A similar approach was used in [5], considering Twitter as the target OSN. We underline that the current solutions for detecting leverage the assumption that a profile of the victim is available in some

OSN, which might not always be the case. Furthermore, if the OSN in which the attack is run and the one considered as a reference for similarity are not of the same type, we expect the performances of the detection solutions to be lower, compared to the case when considering, for example, the reference profile being the same OSN as the cloned one. In [4], authors already pointed out that their approach (measuring similarities) is very specific to the existence of an original profile where the victim already has a presence in. In fact, since there is no pre-existing profile, the similarity measurements and other techniques proposed so far for the detection of cannot be applied to FPA. Recently, another issue has been introduced, it concerns multiple identities in OSNs [6]. It proposes a framework for grouping similar identities which refer to the same person. We focus our study on characterizing the behavior of users, based on dynamic structural information of the OSN of the user. While static aspects of OSNs has been widely considered, less interest has been put so far in dynamic features. However, some investigation has been already made. For example, in [16] authors propose a way to evaluate the global structure of a large scale network by analyzing microscopic behavior of nodes in an OSN. Their model observes the edge-by-edge evolution of the graph by adopting the maximum-likelihood principle. Given a new node arrival in the graph, the creation of new edges is affected by the degree and age of the node. In [7], the authors get a macroscopic view of OSNs graph evolution. It studies OSN growth by evaluating the evolution of three types of groups: singletons, giant components, and the middle region. Other researchers [8] considered and combined two important features of OSNs: service recommendation and friendship prediction. The work shows a correlation between user-service interactions (interests networks) and the link between users (friendship network). However, none of these work leveraged dynamic features of
OSNs for privacy purposes.

3.Proposed Model Description

#Objective:

To detecting spam users from Twitter and block their account from Twitter. It may reduce many online spamming incidents in the future.

#Description:

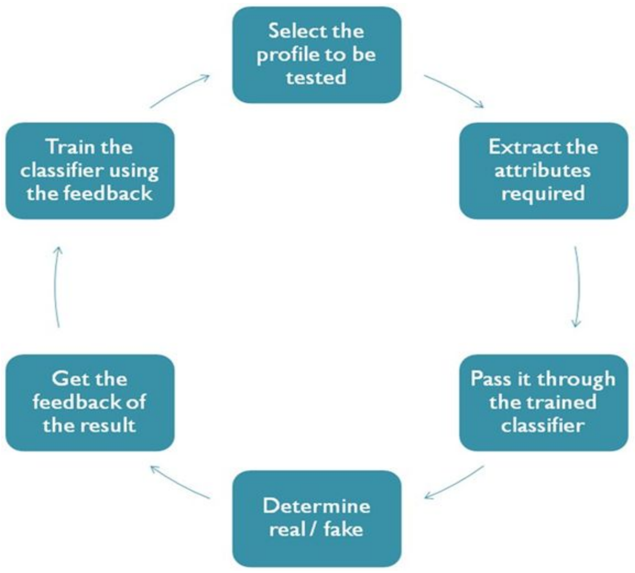
In this proposed model, we will explore the data and will try to understand the relation between class labels, for data analysis and prediction we will use most sophisticated way/algorithm to implementation so that we could have the best accuracy for prediction.

3.1 Proposed framework

The proposed framework in Fig. 1 shows the sequence of processes that need to be followed for continues detection of fake profiles with active learning from the feedback of the result given by the classification algorithm. This framework can easily be implemented by social networking companies.

1. The detection process starts with the selection of the profile that needs to be tested.
2. After the selection of the profile, the suitable attributes (i.e. features) (detailed discussed in chapter 5) is selected on which the classification algorithm is implemented.
3. The attributes extracted is passed to the trained classifier. The classifier gets trained regularly as new training data is feed into the classifier.
4. The classifier determines whether the profile is spam or real.
5. The classifier may not be 100% accurate in classifying the profile so; the feedback of the result is given back to the classifier. For example, if the profile is identified as spam, social networking site can send a notification to the profile to submit identification. If the valid identification is given, feedback is sent to the classifier that the profile was not fake.

6. This process repeats and as the time proceeds, the no. of training data increases and the classifier becomes more and more accurate in predicting the fake profiles.



Select the profile to be tested
Extract the attributes required
Pass it through the trained classifier
Determine real / fake
Get the feedback of the result
Train the classifier using the feedback

Fig.1 Process Sequence

4.About Dataset

Real Time data is collected from Twitter using the Twitter API. For Authenticate user, data collected from the verified account of Twitter and for Spammer data collected from those users who recently reported as spam account by other users. We used 11459 tweets from genuine users and 5789 tweets from spam users.

5.Feature Extraction and Analyzing

We extracted many features from Twitter like TextData, TweetCreatedAt, RetweetCount, TweetFavouriteCount, TweetSource, UserID, UserScreenName, UserName, UserCreatedAt, UserDescription, UserDescriptionLength, UserFollowersCount, UserFriendsCount, UserLocation, HttpCount, HashtagCount, MentionCount, TweetCount. That features we collected from Twitter and a few more we will generate and add by using these features.

We will add a column to our dataset Current_Time by using that feature we can find out the age of Account feature that will help in classification.

Few more feature we will add by evaluating existing features that formula is given below alongside that features.

- AgeOfAccount = Current_time - UserCreateAt
- Reputation Of an User = UserFollowersCount divided by the number of lists.
- TweetPerDay = TweetCount / AgeOfAccount
- TweetPerFollower = TweetCount / UserFollowersCount
- AgeByFollowing = AgeOfAccount / UserFriendsCount
- AvgHashtag = Sum(HashtagCount) / 30
- AvgURLCount = Sum(HttpCount) /30
- AvgFavoriteCount = Sum(TweetFavouriteCount) /30
- AvgMention = Sum(MentionCount) / 30
- AvgRetweetCount = Sum(RetweetCount) / 30

(Division by 30 is done because we have collected 30 tweets from each user's profile, thus average is calculated)

5.1. Analyzing the TextData feature:

If we analyze every tweet from both the legitimate and spam user's profile, we find out that appropriate words are used by both users, but inappropriate words are more often found in the spam user’s tweets. In Fig 2, we can see how much a word ‘sex’ used by the legitimate user in their tweets.

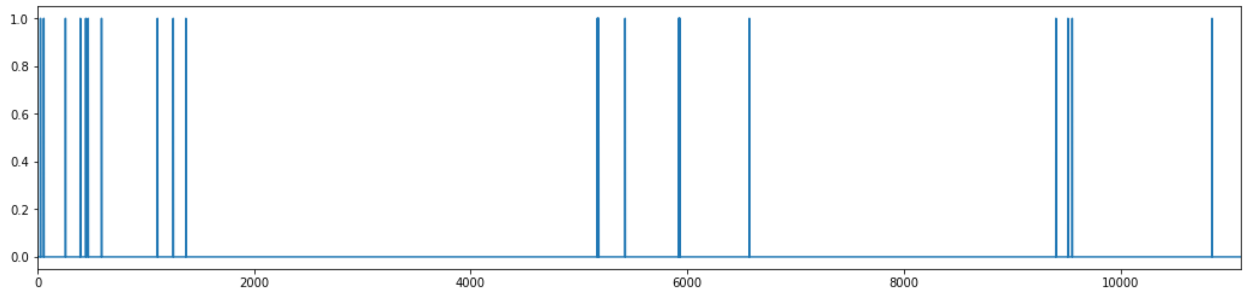


Fig 2. Shows how many times a word used by Legitimate users.

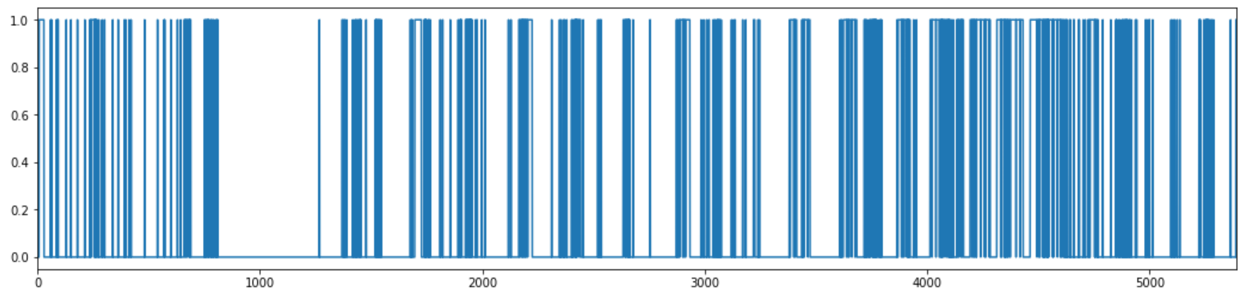


Fig 3. Shows how many times a word used by Spam users.

5.2.Analyzing Tweet Count feature:

The hypothesis is that Real user tweet more than fake user if we calculate mean, Standard deviation, minimum, 25%, 50%, 75%, max, and the result is given below.

● Legitimate User’s TweetCount:

```
count      375.000000
mean     41288.162667
std      93281.144477
min          1.000000
25%        324.000000
50%       3883.000000
75%      20650.000000
max     596778.000000
```

● Spam User's TweetCount:

```
count    1.770000e+02
mean     2.532717e+04
std      9.549593e+04
min      1.000000e+00
25%      6.410000e+02
50%      4.744000e+03
75%      1.185200e+04
max      1.150378e+06
```

5.3. Analyzing Age By Account feature:

The hypothesis is that Real user AgeByAccount more than fake user if we calculate mean, Standard deviation, minimum, 25%, 50%, 75%, max, and the result is given below.

● Legitimate User's AgeByAcount:

```
count     369.000000
mean       57.947560
std       204.846550
min         0.101799
25%         5.720546
50%        13.004804
75%        43.425175
max      3007.774225
```

● Spam User's AgeByAccount:

```
count     171.000000
mean       63.834844
std       289.162889
min         0.008907
25%         0.577691
50%         1.880437
75%         7.191746
max      2914.926181
```

6. Attributes Selection for prediction

Below mentioned are the features that we have selected from our data set as a feed to classifiers , that we got from feature extraction.

- Reputation
- AvgHashtag
- AvgRetweet
- UserFollowersCount
- UserFriendsCount
- AvgFavCount
- AvgMention
- AvgURLCount
- TweetCount
- AgeOfAccount
- TweetPerDay
- TweetPerFollower
- AgeByFollowing

Target Class

We use binary values for fake and real profile 0 for real and 1 for fake that is our target class classifiers classify on basis of that target.

- FakeOrReal

7. Cross k-Validation for splitting training and test data

Cross-validation is a technique used to evaluate predictive models. In this technique, the original samples are divided into two categories: training set for model training and test set for evaluation. The original sample is randomly divided into k subsamples with equal size. One of these subsamples is considered as evaluative data in order to test the model, and the rest of them, k-1 subsamples, are considered as training data. The cross-validation process is repeated k times for k subsamples, each time for one of them as evaluative data. The first advantage of this method is that all samples are used for both training and validation process, and the second one is that each sample is used for validation just once.

8. Evaluation Metrics

The evaluation is based on a confusion matrix associated metrics variables TP, FP, TN, and FN in the confusion matrix refer
to the following:

– True positive (TP): number of fake nodes that are identified as fake nodes.

– False positive (FP): number of normal nodes that are identified as fake nodes.

– True negative (TN): number of normal nodes that are identified as normal nodes.

– False negative (FN): number of fake nodes that are identified as normal nodes.

To evaluate the classifier, accuracy, and area under the curve (AUC) are used. The AUC is performance metrics for binary classifiers; the closer this AUC is to one, the more favorable the final performance of the classification will be. By comparing the ROC curves with the AUC, it captures the extent to which the curve is up in the northwest corner. The metrics which are introduced below are used to calculate the ROC.

– True negative rate (TNR) =TN/ (TN+FP).

– False positive rate (FPR) =FP/ (FP+TN).

– True positive rate (TPR) = TP/ (TP+FN).

– False negative rate (FNR) =FN/ (FN+TP)

9. Classification

Classification is a technique of categorizing an object into a particular class based on the training data set that was used to train the classifier. We feed the classifier with data set so that we can train it to identify related objects with as best accuracy as possible. The classifier is an algorithm used for classification. In this project, we have used four classifiers namely **KNeighbor**, **Decision Tree, Random Forest, Navie Bayes.** Details about the working of classifiers are given below.

9.1. K Nearest Neighbor Classifier :

KNN is a non-parametric and lazy learning algorithm. Non-parametric means there is no assumption for underlying data distribution. In other words, the model structure determined from the dataset. This will be very helpful in practice where most of the real world datasets do not follow mathematical theoretical assumptions. The lazy algorithm means it does not need any training data points for model generation. All training data is used in the testing phase. This makes training faster and testing phase slower and costlier. Costly testing phase means time and memory. In the worst case, KNN needs more time to scan all data points and scanning all data points will require more memory for storing training data.

9.1.1.Result of KNN

Training Accuracy: 0.91534
Test Accuracy: 0.85802

classifier performance report:

Table I

parameter	precision	recall	f1-score	support
0	0.90	0.90	0.90	115
1	0.75	0.77	0.76	47
Micro avg	0.86	0.86	0.86	162
Macro avg	0.83	0.83	0.83	162

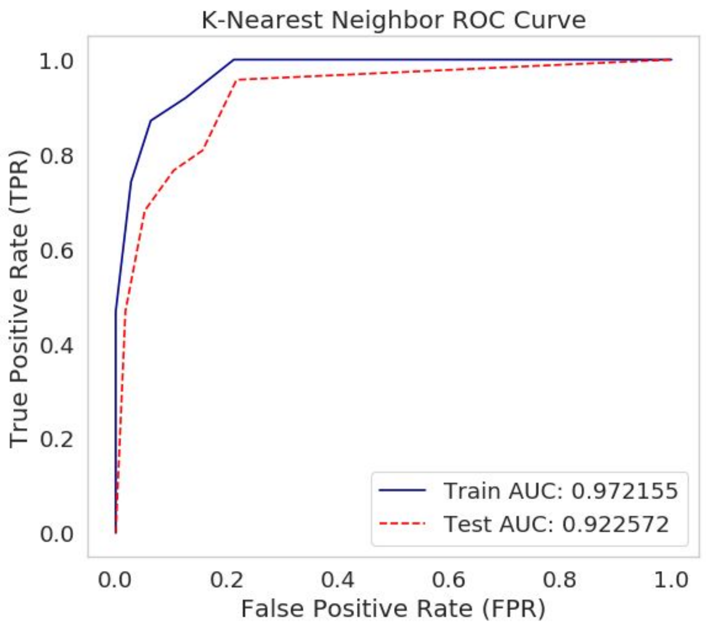


Fig 4. ROC Curve

9.2. Decision Tree Algorithm

A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in recursive manner called recursive partitioning. This flowchart-like structure helps you in decision making. It's visualization like a flowchart diagram which easily mimics the human level thinking. That is why decision trees are easy to understand and interpret. Decision Tree is a white box type of ML algorithm. It shares internal decision-making logic, which is not available in the black box type of algorithms such as Neural Network. Its training time is faster compared to the neural network algorithm. The time complexity of decision trees is a function of the number of records and the number of attributes in the given data. The decision tree is a distribution-free or non-parametric method, which does not depend upon probability distribution assumptions. Decision trees can handle high dimensional data with good accuracy.

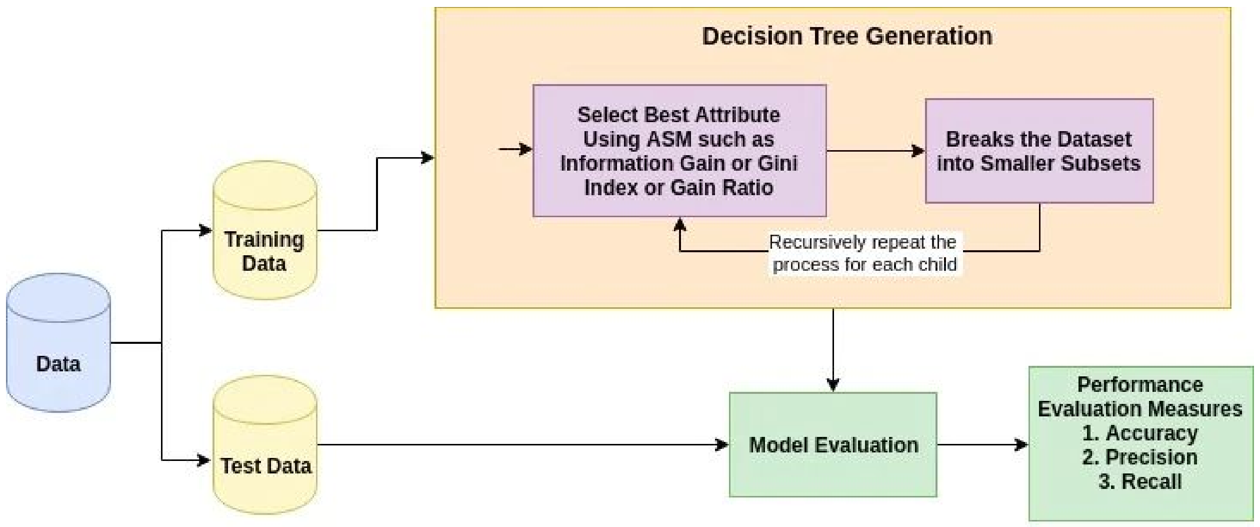

Decision Tree Generation
Select Best Attribute Using ASM such as Information Gain or Gini Index or Gain Ratio
Breaks the Dataset into Smaller Subsets
Recursively repeat the process for each child
Training Data
Data
Test Data
Model Evaluation
Performance Evaluation Measures
1. Accuracy
2. Precision
3. Recall

Fig.5 Procedure of Decision Tree Generation and result.

9.2.1.Result of Decision Tree

Trainig Accuracy: 0.86243
Test Accuracy: 0.91358

Classifier performance report:

Table II

	precision	recall	f1-score	support
0	0.91	0.97	0.94	115
1	0.92	0.77	0.84	47
Micro avg	0.91	0.91	0.91	162
Macro avg	0.92	0.87	0.89	162
weighted	0.91	0.91	0.91	162

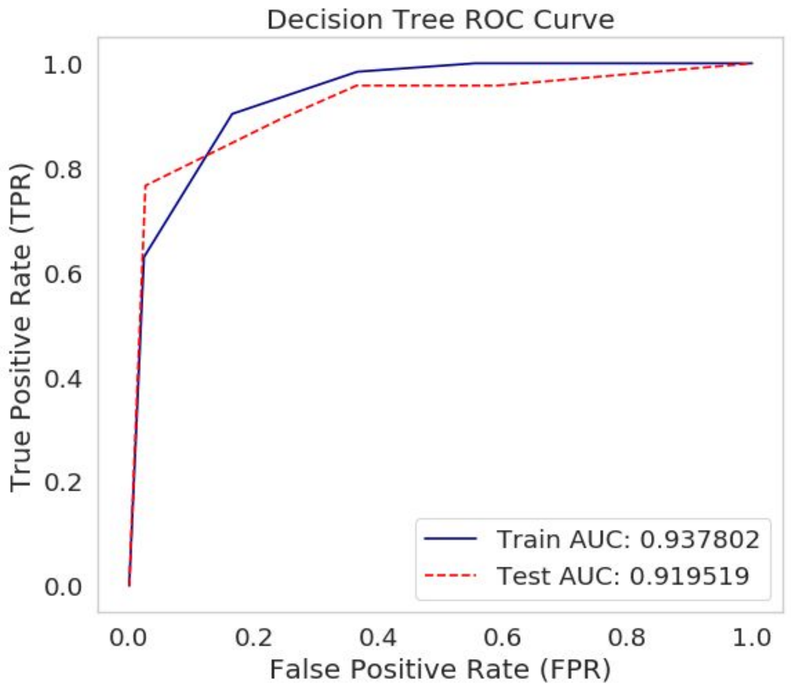


Fig.6 ROC Curve

9.3. Random Forest

Random forests is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use an algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests create decision trees on randomly selected data samples, gets a prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance. Random forests have a variety of applications, such as recommendation engines, image classification, and feature selection. It can be used to classify loyal loan applicants, identify fraudulent activity and predict diseases. It lies at the base of the Boruta algorithm, which selects important features in a dataset. In this tutorial, you are going to learn about all of the following:

9.3.1. How does the algorithm work?

It works in four steps:

1. Select random samples from a given dataset.
2. Construct a decision tree for each sample and get a prediction result from each decision tree.
3. Perform a vote for each predicted result.
4. Select the prediction result with the most votes as the final prediction.

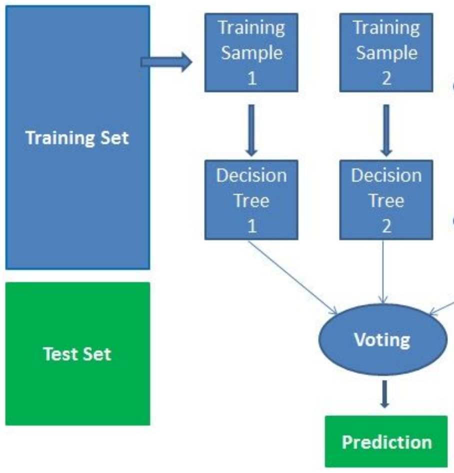


Fig .7 Random Forest Classifier

9.3.2 Result of Random Forest

Training Accuracy: 0.99206
Test Accuracy: 0.92593
Classifier performance report:

Table III

	precision	recall	f1-score	support
0	0.95	0.95	0.95	70
1	0.87	0.87	0.87	38
Micro avg	0.93	0.93	0.93	162
Macro avg	0.89	0.91	0.91	162
weighted	0.93	0.93	0.93	162

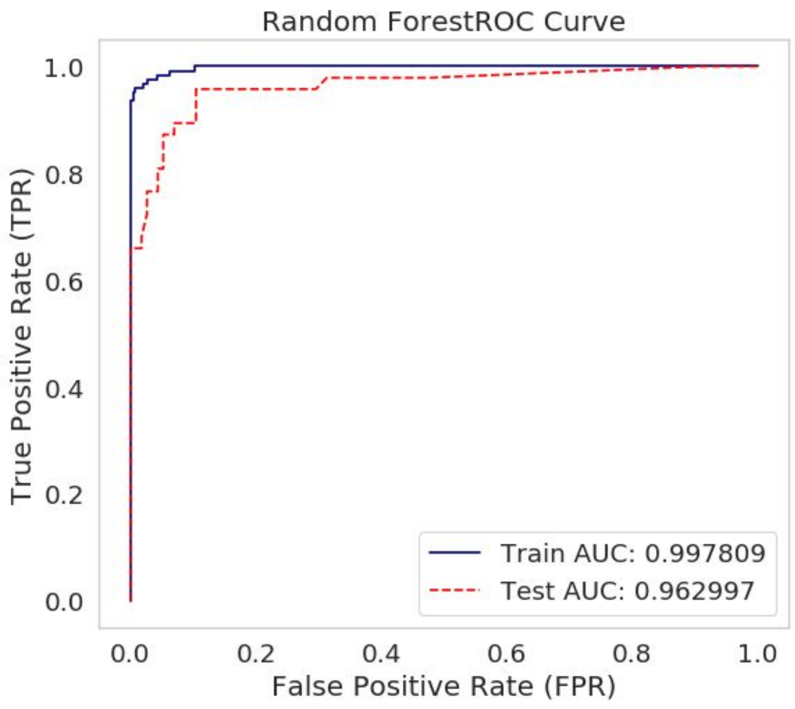


Fig.8 ROC Curve

9.4 Naive Bayes Algorithm

Naive Bayes is a statistical classification technique based on Bayes Theorem. It is one of the simplest supervised learning algorithms. Naive Bayes classifier is the fast, accurate and reliable algorithm. Naive Bayes classifiers have high accuracy and speed on large datasets.
Naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other features. For example, a loan applicant is desirable or not depending on his/her income, previous loan and transaction history, age, and location. Even if these features are interdependent, these features are still considered independently. This assumption simplifies computation, and that's why it is considered as naive. This assumption is called class conditional independence.

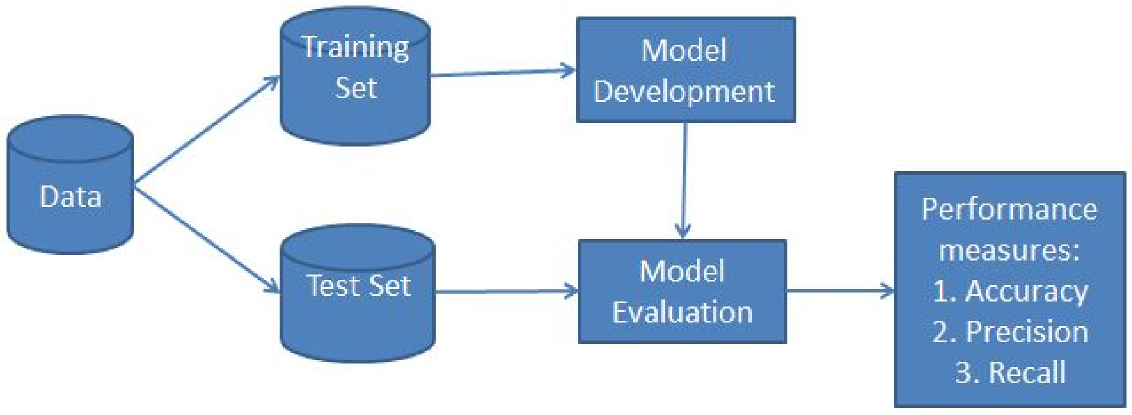


Fig.9 Procedure of classification

9.4.1.Result of Naive Bayes Algorithm

Training Accuracy: 0.65079
Test Accuracy: 0.54321

Classifier performance report:

Table IV

	precision	recall	f1-score	support
0	0.96	0.37	0.53	115
1	0.39	0.98	0.55	47
Micro avg	0.56	0.54	0.54	162
Macro avg	0.68	0.57	0.54	162
weighted	0.81	0.54	0.54	162

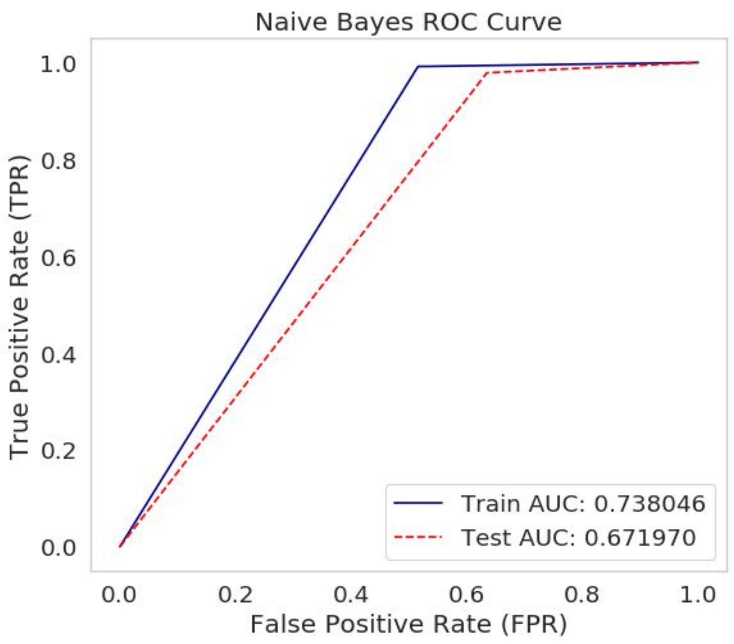


Fig. 10 ROC Curve

10. COMPARISON OF PERFORMANCE OF CLASSIFIER

Table V

Accuracy	**Precision**	**Recall**	**F1-Score**	**Classifier**
0.92	0.97	0.89	0.93	**K- Nearest Neighbor**
0.91	0.97	0.96	0.96	**Decision Tree**
0.67	0.77	0.87	0.82	**Naive Bayes**
0.96	0.97	0.94	0.96	**Random Forest**

10.1. ROC Curve of comparisons

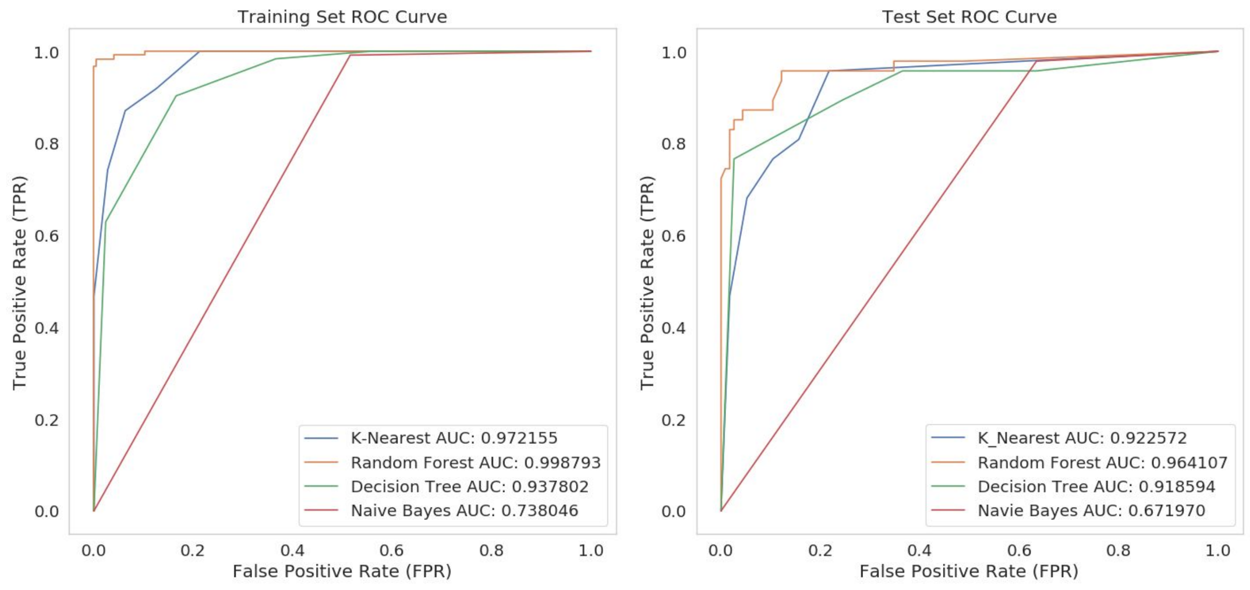


Fig . 11 Comparison of classifier by AUC curve

11. Conclusion and Future Work

We have given a framework which collects data from Twitter using Twitter API and from every tweet, we extract features that we need to feed our classifiers, that binary classification through the Random Forest is more efficient than through any other classifier. Using Decision tree, we have achieved the efficiency of 96%. In the future, we wish to classify profiles by analyzing the behavior of the user by his tweets find out a pattern and classify.

12. References

[1] Lei Jin, Hassan Takabi, and James B.D. Joshi, “Towards active detection of identity clone attacks on online social networks”, in Proceedings of the first ACM CODASPY, 2011

[2] Shah Mahmood and Yvo Desmedt, “Your Facebook deactivated friend or a cloaked spy”, in Proceedings of the 4th IEEE International Workshop on SESOC ’12, 2012

[3] Leyla Bilge, Thorsten Strufe, Davide Balzarotti, and Engin Kirda, “All your contacts belong to us: automated identity theft attacks on social networks”, in Proceedings of the 18th WWW, 2009, pp. 551–560.

[4] Lei Jin, Hassan Takabi, and James B.D. Joshi, “Towards active detection of identity clone attacks on online social networks”, in Proceedings of the first ACM CODASPY, 2011, pp. 27–38.

[5] Georgios Kontaxis, Iasonas Polakis, Sotiris Ioannidis, and Evangelos P. Markatos, “Detecting social network profile cloning.”, in Proceedings of PerCom Workshop, 2011, pp. 295–300.

[6] Kahina Gani, Hakim Hacid, and Ryan Skraba, “Towards multiple identity detection in social networks”, in Proceedings of the 21st international conference companion on World Wide Web, 2012, pp. 503–504.

[7] Ravi Kumar, Jasmine Novak, and Andrew Tomkins, “Structure and evolution of online social networks”, in Proceedings of the 12th ACM SIGKDD, 2006, pp. 611–617.

[8] Shuang-Hong Yang, Bo Long, Alex Smola, Narayanan Sadagopan, Zhaohui Zheng, and Hongyuan Zha, “Like like alike: joint friendship and interest propagation in social networks”, in Proceedings of the 20th WWW, 2011, pp. 537–546.