**KL deemed to be University**

**Green Fields, Vaddeswaram, Guntur (Dt) : 522502**

**Department of Computer Science Engineering**

**Final year Minor Project**



**Submitted by**

K.SRIVATHSA 170030498

P.MANI TEJA 170030978

P.S.S.N.CHANDRA 170030998

V.LAKSHMI PRIYA 170031339

**Under Guidance of**

Dr.CHAYAN PAUL

**IV Year Engineering Course Work**

**2019-2020**



**KL deemed to be University**

**Green fields,Vaddeswaram,Guntur Dist**.

**CERTIFICATE**

This is to certify, that this project work entitled “**DETECTION OF SPAM TWEETS IN TWITTER USING ML ALGORITHMS**” in the BIG DATA specialization is a bonafide work carried out, as part of final year Minor Project by

K.SRIVATHSA-170030498, P.MANI TEJA-170030978,

P.S.S.N.CHANDRA-170030998, V.LAKSHMI PRIYA-170031339 in Department of Computer Science Engineering.

Project Guide Head of the Department

Dr.Chayan Paul Dr. V. HARI KIRAN

**DECLARATION**

We declare that the project work entitled “**DETECTION OF SPAM TWEETS IN TWITTER USING ML ALGORITHMS”** was carried out by us during 2020, and this work is not same as that of any other and has not been submitted for the award of any other degree/diploma

Place: Vaddeswaram.

Date:11-12-2020

K.SRIVATHSA

P.MANI TEJA

P.S.S.N.CHANDRA

V.LAKSHMI PRIYA

**ACKNOWLEDGEMENT**

We express our sincere gratitude to our guide **Mr.Chayan Paul** sir for encouraging and guiding us to undertake this project work.

We express our gratitude to V.Hari Kiran , Head of the Department for Computer Science and Engineering. For providing us with adequate facilities, ways, and means by which we are able to complete this Project.

We would like to place on record the deep sense of gratitude to the honourable Vice-Chancellor, K L University, for providing the necessary facilities to carry the concluded Project

Last but not least, we thank all Teaching and Non-Teaching Staff of our department for their support in the completion of our Project

Place: Vaddeswaram.

Date:11-12-2020

**ABSTRACT**

Twitter has adult hugely over the past few years. With sites like Google, YouTube, Twitter and Facebook, amongst them twitter is hierarchical within the high ten most visited sites. In February 2009, twitter was the fastest-growing web site with a rate of one,382 per. In 2011, folks sent concerning one hundred forty million tweets per day and 460,000 new accounts were created per day. Right now i.e 2020 ,Globally, Twitter has over 330 million monthly active users and a hundred million daily active users the large growth of twitter permits several users to share their data and communicate with one another.

Twitter spam has long been a crucial however troublesome drawback to be addressed . on-line Social Media platforms, like Twitter, change all users, severally of their characteristics, to freely generate and consume vast amounts of information. whereas this information is being exploited by people and organisations to realize competitive advantage, a considerable quantity of information is being generated by spam or pretend users. One in each two hundred social media messages and one in each twenty one tweets is calculable to be spam. The ascension within the volume of world spam is predicted to compromise analysis works that use social media information, thereby questioning information believability. During this project , we have steered a machine learning-based spam detection system that determines whether or not or not a selected message within the real-time dataset is spam or not by employing a set of machine learning algorithms.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **CONTENTS** | **PAGE.NO** |
| **1** | INTRODUCTION | 8-9 |
| **2** | LITERATURE REVIEW | 10-14 |
| **3** | THEORETICAL ANALYSIS | 15-41 |
| **4** | EXPERIMENTAL INVESTIGATIONS AND RESULTS | 42-66 |
| **5** | DISCUSSION OF RESULTS | 67-69 |
| **6** | SUMMARY,CONCLUSION AND RECOMMENDATIONS | 70-71 |
| **7** | REFERENCES | 72-73 |

**1.INTRODUCTION**

An Online Social Network i.e., a Web-based service that allows individuals to construct a public or semi-public profile within a bounded system, articulate a list of other users with whom they share a connection, and view and traverse their list of connections and those made by others within the system. Some of the Online Social Networks (OSNs) that are widely popular right now are Facebook, WhatsApp, Twitter, Instagram etc.

Along with the growth of the social networks, increased the number of spammers. Spammers are the users who manipulate the platforms to broad cast unwanted or malicious messages.

Twitter is a microblogging service where users can post 280-character messages called tweets. Unlike Facebook and MySpace, Twitter is directed, meaning that a user can follow another user, but the second user is not required to follow back. Most accounts are public and can be followed without requiring the owner’s approval. With this structure, spammers can easily follow legitimate users as well as other spammers. Now-a-days, Twitter (An Online Social Network) has become an integral part in a daily of everyone. Twitter was Publicly Introduced on July 15,2006.The Success of social networking services can be seen in the dominance of today's society with Twitter having 330 million monthly active users by 2020

As of May 2020, every second, on average, around 6,000 tweets. or, 350,000 tweets sent per minute or, 500 million tweets sent each day or, 200 billion tweets per year are present facts. Due to this huge growing trend, this Online Social Network has attracted many users along with spammers. [1]

Web Attacks that have appeared on Twitter are Scan , Spam , Phishing etc., Spam is a form of Platform Manipulation. Platform Manipulation is considered as an activity that is intended to negatively impact the people’s experience on Twitter. This includes unsolicited or repeated actions. Spam can include malicious automation and other forms of platform manipulation such as fake accounts. [2]

Shortened URL is included in most of the Spam Tweets to trick users into clicking on it. Also they tend to tweet similar trending topics to attract a larger audience since resources, such as tweets can be shared with each other. This type of Web Attacks not only disturbs the user experience but also causes a whole internet damage which may possibly cause temporary of Internet Services all over the Globe

To handle the consequences, User can report a spam by clicking their home page. Then Accordingly the spam accounts are suspended. However, as the Total number of Tweets sent per Day are 500 million in 2020, Among which 10%(Approx.) are of Spam Tweets. This has become a major problem on finding an appropriate Solution. In this Project, the one way of solving this problem is given. We have used the approach of Machine Learning Algorithms. Classification is used here, Classification refers to a predictive modelling problem where a class label is predicted for a given example of input data. here we have used it to get whether a tweet is a spam or not.

The Solution for solving this problem is to identify the helpful features that can be used in machine learning algorithms to classify messages as Spam or ham. In this Project we have tried the following steps: Decision Tree, Random Forest, XGboost, ADABoost, Logistic Regression. Among all the Performed Classification Technique, XGboost returns much better performances comparatively to other Techniques. In this project we try to train machine learning models which can detect messages in spam and non-spam category. We used an open source data set to train the models and then use the trained models to detect the real world tweets as either spam or ham.

**2.LITERATURE REVIEW**

🡪Resul Kara et el [3]., in their paper titled A Survey of Spam Detection Methods on Twitter, tried to identify new features to identify the tweets as Spams. They opined that is in order to provide a spam-free environment, tweets of spammers are needed to be detected and filtered as well as the owners. By doing this, it is critical to reduce false positive detections in order to prevent legitimate users to be classified as spammers. In their paper, the features of Twitter spam detection and proposed approaches in the literature are discussed with considering their advantages and disadvantages. Also, the outdated features of Twitter which are commonly used by Twitter spam detection approaches are highlighted. Some new features of Twitter which, to the best of our knowledge, have not been mentioned by any other works are also presented.

🡪Nasira Perveen et el [4].,in their paper titled Sentiment Based Twitter Spam Detection, tried to Propose a spam detection approach for twitter based on sentimental features. They opined that By using twitter API they collected their dataset of 29 most trending topic in 2012. They proposed sentimental and some content based features which will help in identifying spam tweets and return spam filtered result set when user visit twitter with good accuracy rate. They evaluate the usefulness of their suggested features in spam detection by using five traditional classifiers like BayesNet, Naive Bayes, Random Forest, Support Vector Machine (SVM) and J48 schemes. Their experiments results shows that Naive Bayes, J48 and Random Forest classifier gives over all best performance than the others.

🡪Monika Verma et el [5].,in their paper titled Techniques to Detect Spammers in Twitter- A Survey, tried to detect Spam profiles in OSNs. In this paper they opined many methods that have been developed and used by various researchers to find out spammers in different social networks. From the papers reviewed it can be concluded that most of the work has been done using classification approaches like SVM, Decision Tree, Naive Bayesian, and Random Forest. Detection has been done on the basis of user based features or content based features or a combination of both. Few authors also introduced new features for detection. All the approaches have been validated on very small dataset and have not been even tested with different combinations of spammers and non spammers. Combination of features for detection of spammers has shown better performance in terms of accuracy, precision, recall etc. as compared to using only user based or content based features.

🡪Rohini et el [6].,in their paper titled Improving Spam Detection on Online Social Media with hybrid classification techniques on Twitter platform, tried to use the Naïve Bayes theorem classifier and build a speaker organization to exclude spam and not spam.

In this paper they opined that Using ML algorithm SVM ( Support vector machine) and NB are used to Improving Spam Detection on Online Social Media with hybrid classification techniques on the Twitter platform.In this dissertation, the System provides a fundamental evaluation of ML algorithms on the detection of streaming spam tweets. In this evaluation, the system works on offline tweets and real-time tweets which are timely updated. The system identified that Feature discretization was an important pre-process to ML-based spam detection.

🡪N.Noor Allema et el [7].,in their paper titled Spam Detection Framework for Twitter using ML, tried to Detect Spam Accounts using ML Algorithms. In this paper they opined that spam has become one of the main issue and it should be solved in every social networking site. Spam detection framework in twitter using machine learning help people to solve their spam issues in a easy and accurate way. This project detects the spam to 95% and spammers can be easily blocked.

🡪Claudia Meda et el [8].,in their paper titled Machine Learning Techniques applied to Twitter Spammers Detection, tried to Detect spam tweets using machine learning techniques. In this paper they opined that SVM and Random Forest algorithm are used to find the spam in the data set and they came to the conclusion that random forest performance is more when compared with other techniques.

🡪Vanyashree Mardi et el [9].,in their paper titled Text-Based Spam Tweets Detection Using Neural Networks, tried to Detect Spam Tweets Using Neural Network. In this Paper is methods such as Naive-Bayes Classifier and Artificial Neutral Network are used.They opined that Performance study of these two algorithms, shows that Artificial Neural Network performs better than Naive Bayes Classification algorithm.

🡪Nan Sun et el [10].,in their paper titled Near real-time twitter spam detection with machine learning techniques, tried to Detect Real-Time Spam Tweets using ML Algorithms utilizing Parallel Computing Techniques. In this paper, Machine Learning along with utilizing parallel computing techniques are used. Machine Learning Techniques such as Random Forest, SVM are used. They opined that among them, Random Forest has more stable performance when compared to other techniques.

🡪Jagtap Kalyani Laxman et el [11].,in their paper titled Machine Learning Approach For Spam Tweets Detection, tried to Detect Of Real-Time Spam Tweets Using Naïve Bayes or SVM. In this Journal they have opined that NB and SVM cannot accurately give us the output as NB and SVM have different options including the choice of kernel function for each. They are both sensitive to parameter optimization (i.e. different parameter selection can significantly change their output). The work of Naive Bayes classifier is better than SVM classifier for the taken dataset.

🡪K Subba Reddy et el [12].,in their paper titled Detecting Spam Messages in Twitter Data by Machine learning Algorithms using Cross-Validation, tried of Classifying messages as Spam or Ham. In their paper to avoid spam messages, they proposed a methodology by using machine learning algorithms and to develop an approach using a set of content-based features and prepared a spam detection model using the Support vector machine algorithm(SVM) and Naive Bayes classification algorithm and made use of precision, recall, and F measure metrics to measure the performance of the model.

🡪Miss.Shukla Twinkle Kailas et el [13].,in their paper titled Design of Machine Learning Approach For Spam Tweet Detection, tried to categorize the Spam and Non-spam tweets by machine learning approach. In their paper classifier system-based approach is used to solve the detection of spam messages, the classification model is mainly based on a machine learning algorithm which gives the output in the form of the binary value. The most important phase of the project is feature extraction to add

more benefits to the system. They opined that the performance evaluation is carried out on a large dataset which includes around 600 tweets to identify the spammer also system helps to categories the spam and non-spam message.

🡪Vinodhini.M et el [14].,in their paper titled Spam Detection Framework using ML Algorithm, tried to Determine whether or not a specific message in the dataset is spam using a set of machine learning algorithms. They have used Bayes Network, Naive Bayes, K-nearest neighbor identified the spams and spammers present in a twitter dataset with the help of machine learning algorithms and NLP concepts. They opined that by reviewing the spam, the entire details about the spammer are accessed and displayed, which in turn helps in determining other spams, spammers, and their way of writing messages.

**3.THEORETICAL ANALYSIS**

**3.1 Twitter:-**

Twitter has grown tremendously over the past few years. With sites such as Google, YouTube, Twitter and Facebook, amongst them twitter is ranked in the top 10 most visited sites. In February 2009, twitter was the fastest-growing website with a growth rate of 1,382 per. In 2011, people sent about 140 million tweets per day and 460,000 new accounts were created per day. Right now i.e in 2020 ,Globally, Twitter has over 336 million monthly active users and 100 million daily active users [1].The enormous growth of twitter allows many users to share their information and communicate with each other. Online Social Media platforms, such as Twitter, enable all users, independently of their characteristics, to freely generate and consume huge amounts of data.

**3.2 Spam:-**

[Spam](https://blog.malwarebytes.com/glossary/spam/) is any sort of unwanted, unsolicited digital communication, regularly an email, that receives sent out in bulk. Spam is a large waste of time and resources. The Internet service providers (ISP) convey and save the data. When hackers can’t scouse borrow data bandwidth from the ISPs, they scouse borrow it from individual users, hacking computer systems and enslaving them in a zombie [botnet](https://blog.malwarebytes.com/threats/botnets/).

**3.3 Twitter Spam:-**

The enormous growth of twitter allows many users to share their information and communicate with each other. Twitter spam has long been a critical but difficult problem to be addressed. Online Social Media platforms, such as Twitter, enable all users, independently of their characteristics, to freely generate and consume huge amounts of data. While this data is being exploited by individuals and organisations to gain competitive advantage, a substantial amount of data is being generated by spam or fake users. One in every 200 social media messages and one in every 21 tweets is estimated to be spam. The rapid growth in the volume of global spam is expected to compromise research works that use social media data, thereby questioning data credibility.

**3.4 ALGORITHMS**

* **LOGISTIC REGRESSION:**

Logistic regression is a classification algorithm. It is wont to predict a binary outcome supported a group of independent variables. A binary outcome is one where there are only two possible scenarios either the event happens or not happen as an outcome. Logistic regression is employed to calculate the probability of a binary event occurring, and to affect problems with classification.

The different types of logistic regression

• Binary logistic regression, For example, the output can be Success/Failure, 0/1, True/False, or Yes/No.

• Multinomial logistic regression example The transport type is going to be the variable like train, bus, tram, and bike

• Ordinal logistic regression Examples of such as answers on an opinion poll (good/avg/bad)

However, it is far easier to implement than alternative strategies, particularly within the context of machine learning, and works well for cases wherever the dataset is linearly dissociable and provides helpful insights

**Sigmoid Function**

It is a function having a characteristic which will take any real value and map it to between 0 to 1 shaped just like the letter “S”. The sigmoid function also called a logistic function.

Y = 1 / 1+e -z

So, if the worth of z goes to positive infinity then the anticipated value of y will become 1 and if it goes to negative infinity then the anticipated value of y will become 0. And if the result of the sigmoid function is quite 0.5 then we classify that label as class 1 or positive class and if it's but 0.5 then we will classify it to negative class or label as class 0.Chart

Description automatically generated

[15]

Step by Step Process

1) We need to build the logistic regression model and fit it to the training data set. First, we will need to import the logistic regression algorithm from Sklearn.

2)Next, we need to create an instance classifier and fit it to the training data.

3)Next, we need to create predictions on the test dataset.

4)Lastly, we can check the performance of our model by using the Confusion matrix.

5)Finally we check Accuracy of the model we made.

In this project we used Logistic Regression classifier to classify different tweets extracted from twitter supported various features into spam or quality, but the twitter is suffering from an excellent deal of options

.

Initially we'd like to make a training dataset that contains information about the tweets, including some features required like following, followers, actions, isretweet, location and specific labels i.e., spam, quality.

We must separate the columns (attributes or features) of the dataset into input patterns (X) and output patterns (Y). we will do this easily by specifying the column indices within the NumPy array format. Finally, we must split the X and Y data into a training and test dataset. The training set are going to be wont to prepare the Logistic Regression model and therefore the test set are going to be wont to make new predictions, from which we will evaluate the performance of the model.

For this we'll use the train\_test\_split() function from the scikit-learn library. We also specify a seed for the random number generator so as that we always get the same split of data whenever this instance is executed.

.

This means we will use the complete scikit-learn library with Logistic Regression models.

The Logistic Regression model for classification is named Logistic Regression()

.

We can create and and fit it to our training dataset. Models are fit using the scikit-learn API and therefore the log.fit() function.

We can make predictions using the fit model on the test dataset.

To make predictions we use the scikit-learn function log.predict()

By default, the predictions made by Logistic Regression are probabilities. Because this is often often a binary classification problem, each prediction is that the probability of the input pattern belonging to the first class. we will easily convert them to binary class values by rounding them to 0 or 1.

We can use confusion\_matrix() function by taking the test dataset and predict model for this.Now that we've used the fit model to form predictions on new data, we will evaluate the performance of the predictions by comparing them to the expected values. For this we'll use the inbuilt accuracy\_score() function in scikit-learn.

In this project we used Logistic Regression classifier to classify different tweets extracted from twitter supported various features into spam or quality, but the twitter is suffering from an excellent deal of options

.

Initially we'd like to make a training dataset that contains information about the tweets, including some features required like following, followers, actions, isretweet, location and specific labels i.e., spam, quality.

We must separate the columns (attributes or features) of the dataset into input patterns (X) and output patterns (Y). we will do this easily by specifying the column indices within the NumPy array format. Finally, we must split the X and Y data into a training and test dataset. The training set are going to be wont to prepare the Logistic Regression model and therefore the test set are going to be wont to make new predictions, from which we will evaluate the performance of the model.

For this we'll use the train\_test\_split() function from the scikit-learn library. We also specify a seed for the random number generator so as that we always get the same split of data whenever this instance is executed.

.

This means we will use the complete scikit-learn library with Logistic Regression models.

The Logistic Regression model for classification is named Logistic Regression()

.

We can create and and fit it to our training dataset. Models are fit using the scikit-learn API and therefore the log.fit() function.

We can make predictions using the fit model on the test dataset.

To make predictions we use the scikit-learn function log.predict()

By default, the predictions made by Logistic Regression are probabilities. Because this is often often a binary classification problem, each prediction is that the probability of the input pattern belonging to the first class. we will easily convert them to binary class values by rounding them to 0 or 1.

We can use confusion\_matrix() function by taking the test dataset and predict model for this.Now that we've used the fit model to form predictions on new data, we will evaluate the performance of the predictions by comparing them to the expected values. For this we'll use the inbuilt accuracy\_score() function in scikit-learn.

* **RANDOM FOREST:**

Random Forest is one in all the popular and most used supervised learning algorithm because it can perform both multivariate analysis in predicting continuous values and also classification in predicting class labels. Random forest algorithm was built by Tin Kam Ho by random subspace method in his formulation is an approach to implement “Stochastic discrimination” in order to classify which was proposed by Eugene Kleinberg. Leo Breiman and Adele Culter developed the continuation of the algorithm.

Further the extension was carried out by Breiman which included the combination of bagging and random selection of features, introduced by Tin Kam Ho and later by Amit and Geman so to construct a group of decision trees with lower variance.

As we all know that the bigger variety of trees within the forest ends up in higher accuracy and prevents the matter of overfitting. Random forests were used as blackbox models in business field as random forest is capable of generating legitimate prediction beyond large amount of data. Overfitting occurs once we have a really flexible data model the model features a high capability that mainly memorizes the training data by fitting it closely. The matter is that the model learns not solely the factual relationships from the training data, however conjointly associate the noise that is present in the training data. A flexible model is supposed to own high variance because the learned parameters resembling the structure of the decision tree can vary significantly with the training data. Whereas an inflexible model is said to have high bias as it makes assumptions about the training data it is biased towards prepossessed ideas of the data a linear classifier always makes the inference that the information is linear and doesn't have the flexibleness to suit non-linear relationships. Associate in inflexible model might not have the capacity to fit even the training data and in each case high variance and high bias the model is not ready to generalize well to new data samples. Furthermore, the random forest algorithm creates decision trees on knowledge data samples then gets the prediction from every of them and eventually it selects the foremost effective answer by means that of vote.

It is an ensemble technique that is highly efficient than one decision tree as a result of it reduces the over-fitting by averaging the result. The ensemble could be a method of combining multiple classifiers to resolve a complicated problem and to boost the performance of the model.

This model makes use of key principles that offers it the call random are random sampling of training data samples while constructing trees. Random subsets of features taken into consideration while splitting nodes. When training, every tree in a random forest learns from a random pattern of the data samples. The samples are drawn with replacement, recognised as bootstrapping, this means that that a few samples may be used a couple of times in a single tree. The concept is that through training every tree on one-of-a-kind samples, despite the fact that every tree may have big variance with recognize to a specific set of the training records, overall, the whole forest could have decrease variance however now no longer on the growing the bias. During the testing phase the average of the predictions of each decision tree is considered to predict the result and this process of training each individual learner on completely independent bootstrapped subsets of the data so averaging the predictions made is understood as bagging abbreviation for bootstrap aggregation.

Random forests work well for an oversized vary of data than one decision tree does. The random forest has less variance than one decision tree and is extremely versatile and possesses very high accuracy. Random Forest algorithms maintain sensible accuracy even a huge proportion of the information is missing.

When it comes to model performance, each parameter plays a vital role.

1. General Parameter:

The parameter that takes care of the overall functioning of the model.

* n\_estimators : *int, default=100*

The number of trees in the forest.

* . criterion : *{“gini”, “entropy”}, default=”gini”*

The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain. Note: this parameter is tree-specific.

* max\_depth : *int, default=None*

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min\_samples\_split samples.

* min\_samples\_split  *: int or float, default=2*

The minimum number of samples required to split an internal node:

* If int, then consider min\_samples\_split as the minimum number.
* If float,then min\_samples\_splits is a fraction andceil(min\_samples\_split\*n\_samples) are the minimum number of samples for each split.
* min\_samples\_leaf : *int or float, default=1*

The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min\_samples\_leaf  training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

* If int, then consider min\_samples\_leaf as the minimum number.
* If float, then min\_samples\_leaf  is a fraction and  ceil(min\_samples\_leaf\*n\_samples) are the minimum number of samples for each node.
* min\_weight\_fraction\_leaf : *float, default=0.0*

The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample\_weight is not provided.

* max\_features : *{“auto”, “sqrt”, “log2”}, int or float, default=”auto”*

The number of features to consider when looking for the best split:

* If int, then consider max\_features features at each split.
* If float, then max\_features is a fraction and int(max\_features\*n\_features) features are considered at each split.
* If “auto”, then max\_features=sqrt(n\_features).
* If “sqrt”, then max\_features=sqrt(n\_features) (same as “auto”).
* If “log2”, then max\_features=log2(n\_features).
* If None, then max\_features=n\_features.
* max\_leaf\_nodes : *int, default=None*

Grow trees with max\_leaf\_nodes in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes.

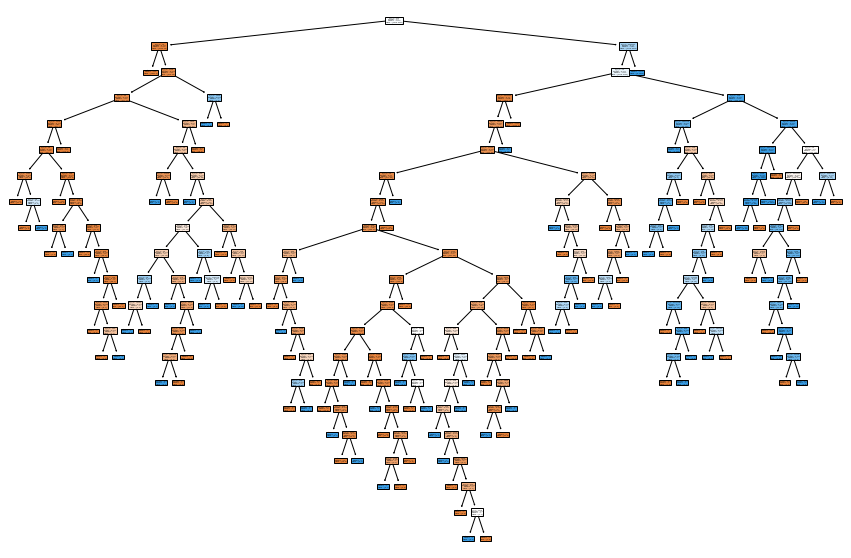
* min\_impurity\_decrease : *float, default=0.0*

A node will be split if this split induces a decrease of the impurity greater than or equal to this value.

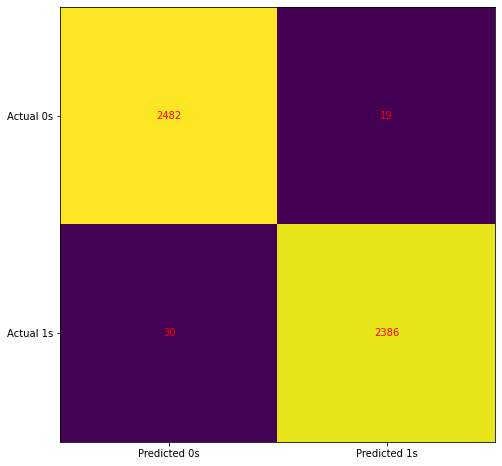
In this project we used Random forest classifier to classify different tweets extracted from twitter based on various features into spam or quality, but the twitter is littered with a great deal of options.

Initially we need to create a training dataset that contains information about the tweets, including some features required like following, followers, actions, isretweet, location and specific labels i.e., spam, quality. We had then ought to split the data by sorting out in order that we will be able to split it within the biggest manner possible. The split data broadly contains the input data and output data based on feature extraction. We might want to start by splitting Tweets by following and so by actions. Then would keep splitting until that particular node no longer needs it and can predict a specific selected tweet with its correct label based on averaging the majority pool of votes of each decision tree created by the Random Forest classifier that's highly efficient, test set will be used to make new predictions, from which we can evaluate the performance of the model.

In order to generate predictions on the unseen new tweets from the info we used the scikit-learn function **model.predict()**. To visualize one of the decision trees in this Random forest model we can make use of “tree” module from sklearn library an open-source library in python. The function plot\_tree() is used here to visualize 11th decision tree of random forest model .



We can use confusion\_matrix() function by taking the test dataset and predict model for this.Finally, we evaluate our trained model using confusion matrix. The model finished with 19 false positives and 30 false negatives.The plot of confusion matrix looks like:



Now that we have used the fit model to predict on new data, we will evaluate the performance of the predictions by comparing them to the expected values. For that purpose we will be using the built-in function accuracy\_score() in scikit-learn.

* **XGBOOST:**

XGBoost has mature to be a extensively used and truly noted device amongst Kaggle competition and information Scientists in trade, as a result of it's been warfare examined for producing on large-scale problems. it's a particularly bendy and versatile device that may paintings through most regression, category and rating problems additionally to user-constructed goal functions. As AN open-supply computer code, it's miles simply handy and it may well be used through exceptional systems and interfaces.

The very sensible movability and compatibility of the appliance permits its utilization on all three Windows, Linux and OS X. It to boot helps education on assigned cloud systems like AWS, Azure, GCE amongst others and it's miles simply associated with large-scale cloud dataflow structures that embody Flink and Spark. though it become created and to begin with used withinside the program line Interface (CLI) via method of suggests that of its creator (Tianqi Chen), it may to boot be loaded and used in various languages and interfaces that embody Python, C++, R, Julia, Scala and Java.

Its decision stands for eXtreme Gradient Boosting, it become advanced via method of suggests that of Tianqi Chen and now's a locality of {a much|a method|a far} wider series of open-supply libraries advanced via way of suggests that of the Distributed Machine Learning Community (DMLC). XGBoost could be a scalable and proper implementation of gradient boosting machines and it's verified to push the boundaries of computing electricity for boosted bushes algorithms as a result of it become created and advanced for the sole reason of version overall performance and process speed.

Specifically, it become built to create the foremost every bit of reminiscence and hardware assets for tree boosting algorithms.

The implementation of XGBoost provides various superior functions for version calibration, computing environments and set of rules sweetening. it's able to acting the three vital kinds of gradient boosting (Gradient Boosting (GB), random GB and regularised GB) and it's miles robust spare to help satisfactory calibration and addition of regularization parameters. per Tianqi Chen, the latter is what makes it advanced and exceptional to completely different libraries.

The set of rules become advanced to with success reduce computing time ANd allot an most useful utilization of reminiscence assets. vital functions of implementation embrace addressing of lacking values (Sparse Aware), Block Structure to help parallelization in tree creation and also the capability to in form and increase on new statistics delivered to a talented version.

It includes out the gradient boosting call tree algorithmic program. it's various exceptional names like gradient boosting, gradient boosting machine, etc.

Boosting isn't something but ensemble methods within which preceding version mistakes square measure resolved withinside the new models. These models square measure brought instantly until no completely different development is seen. one among the great samples of such AN set of rules is the AdaBoost algorithmic program.

Gradient boosting could be a technique within which the novel models square measure created that computes the error withinside the preceding version once which leftover is dropped at create the end prediction.

It makes use of a gradient descent algorithmic program this can be the aim it's miles stated as a “Gradient Boosting Algorithm”. Weather classification or regression methods square measure supported for every kinds of prophetical modelling issues.

When it involves model performance, every parameter plays an important role.

XGBoost, as per the creator, parameters are widely divided into three different classifications that are stated below [16] -

 1.General Parameter:

The parameter that takes care of the overall functioning of the model.

* Booster[default=gbtree]

Assign the booster type like gbtree, gblinear or dart to use. Use gbtree or dart for classification problems and for regression, you can use any of them.

* nthread[default=maximum cores available]

The role of nthread is to activate parallel computation. It is set as maximum only as it leads to fast computation.

* silent[default=0]

It is better not to change it if you set it to 1, your console will get running messages.

1. Booster Parameter:

A parameter that powers the selected booster performance.

Parameters for Tree Booster

* nrounds[default=100]

It controls the maximum number of iterations. For classification, it is similar to the number of trees to grow. Should be tuned using CV

* eta[default=0.3][range: (0,1)]

It commands the learning rate i.e the rate at which the model learns from the data. The computation will be slow if the value of eta is small. Its value is between 0.01-0.03.

* gamma[default=0][range: (0,Inf)]

Its function is to take care of the overfitting. Its value is dependent on the data. The regularization will be high if the value of gamma is high.

* max\_depth[default=6][range: (0,Inf)]

Its function is to control the depth of the tree, if the value is high, the model would be more complex.  There is no fixed value of max\_depth. The value depends upon the size of data. It should be tuned using CV.

* subsample[default=1][range: (0,1)]

Its values lie between (0.5-0.8) and it controls the samples given to the tree.

* colsample\_bytree[default=1][range: (0,1)]

It checks about the features supplied to the tree.

* lambda[default=0]

It is used to avoid overfitting and controls L2 regularisation.

* alpha[default=1]

Enabling alpha, it results in feature selection by which it is more useful for high dimension dataset. It controls L1 regularization on weights.

Parameters for Linear Booster

 Its computation is high as it has relatively fewer parameters to tune.

* nrounds[default=100]

It powers the iteration that is required by gradient descent to converge. It should be tuned using CV.

* lambda[default=0]

Its function is to permit Ridge Regression.

* alpha[default=1]

Its function is to permit  Lasso Regression.

1. Learning Tak Parameters:

A parameter that validates the learning process of the booster.

* Objective[default=reg:linear]
* Reg: linear - It is used for linear regression.
* Binary: logistic - It is used for logistic regression for binary classification that returns the class probabilities.
* Multi: softmax - It is used for multi-classification using softmax that returns predicted class labels.
* Multi: softprob - It is used for multi-classification using softmax that returns predicted class probabilities.
* **eval\_metric [no default, depends on objective selected]**

These metrics are used to validate a model's capability to generalize. For the classification type of problem, the default is an error and for regression, the default metric is RMSE.

#### **Error functions are listed below:**

* **mae -**Used in regression.
* **Log loss -**Negative log-likelihood that is used in classification.
* **AUC -**Area under curve used in classification
* **RMSE -**Root mean square error used in regression
* **Error -**Binary classification error rate.
* **mlogloss -**multiclass log loss used for classification again.

In this project we tend to used XGBoost classifier to classify completely different tweets extracted from twitter supported varied options into spam or quality, however the twitter is laid low with a good deal of choices.

Initially we want to make a coaching dataset that contains data regarding the tweets, as well as some options needed like following, followers, actions, isretweet, location and specific labels i.e., spam, quality. we tend to should separate the columns (attributes or features) of the dataset into input patterns (X) and output patterns (Y). we are able to try this simply by specifying the column indices within the NumPy array format. Finally, we tend to should split the X and Y data into a coaching and check dataset. The coaching set are going to be accustomed prepare the XGBoost model and also the check set are going to be accustomed create new predictions, from that we are able to assess the performance of the model.

For this we'll use the train\_test\_split() function from the scikit-learn library. we tend to additionally specify a seed for the random range generator in order that we tend to continually get constant split of information anytime this instance is executed.

XGBoost provides a wrapper category to permit models to be treated like classifiers or regressors within the scikit-learn framework.

This means we are able to use the full scikit-learn library with XGBoost models.

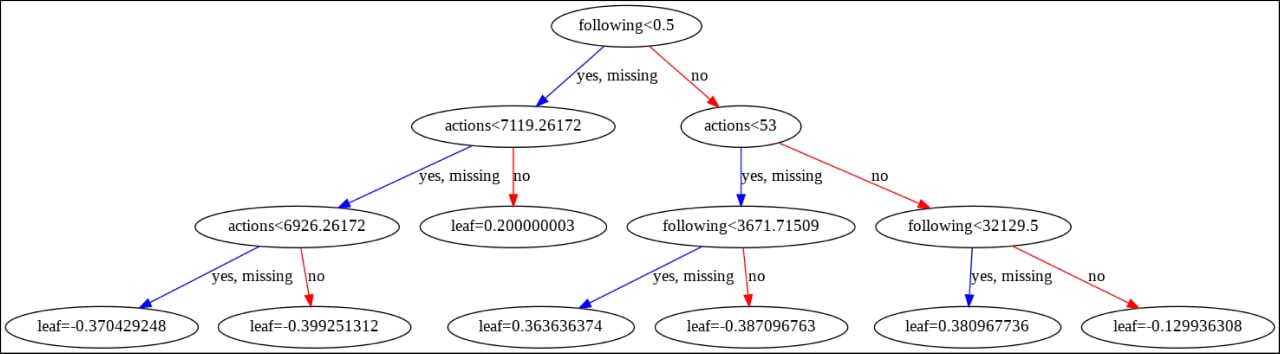
The XGBoost model for classification is called **XGBClassifier**. Here We have used Objective=Binary Logistic, random\_state=42 , n\_estimators=800, learning\_rate=2.

We can create and and fit it to our training dataset. Models are fit using the scikit-learn API and the **model.fit()** function.

We can make predictions using the fit model on the test dataset.

To make predictions we use the scikit-learn function **model.predict()**.

By default, the predictions made by XGBoost are probabilities. Because this is a binary classification problem, each prediction is the probability of the input pattern belonging to the first class. We can easily convert them to binary class values by rounding them to 0 or 1.We can use confusion\_matrix() function by taking the test dataset and predict model for this.This is how Plot of XGBoost will look like:-



Now that we have used the fit model to make predictions on new data, we can evaluate the performance of the predictions by comparing them to the expected values. For this we will use the built in accuracy\_score() function in scikit-learn.

* **ADA BOOST:**

AdaBoost, transient for adaptive Boosting, is a machine learning meta-algorithm formulated via means of means that of Yoav Freund and Robert Schapire , United Nations agency received the 2003 Gödel Prize for their work. It are often used in conjunction with many alternative forms of learning algorithms to reinforce overall performance. The output of the other gaining data of algorithms ('vulnerable inexperienced persons') is combined right into a weighted total that represents the previous output of the boosted classifier. AdaBoost is adaptive withinside the sense that next weak inexperienced persons square measure tweaked in favor of those instances misclassified via means of means that of preceding classifiers. AdaBoost is sensitive to hissing information and outliers , during a few troubles it are often abundant less liable to the overfitting hassle than totally different learning algorithms. The individual inexperienced persons could also be vulnerable, but ciao because the overall performance of each one is barely beyond random guesswork, the ultimate version could also be established to converge to a strong learner.

Every learning rule features a tendency to suit some trouble varieties beyond others, and frequently has several distinctive parameters and configurations to manage prior it achieves optimum overall performance on a dataset. AdaBoost (with decision trees because the weak inexperienced persons) is commonly referred to as the simplest out-of-the-box classifier. once used with selection tree gaining data of, facts collected at each degree of the AdaBoost rule some the relative 'hardness' of each coaching sample is fed into the tree growing rule such later trees have a tendency to awareness on harder-to-classify examples.

It combines multiple classifiers to extend the accuracy of classifiers. AdaBoost is Associate in Nursing repetitive ensemble technique. AdaBoost classifier builds sturdy|a robust|a powerful} classifier by combining multiple poorly playacting classifiers in order that you may get high accuracy strong classifier. the fundamental thought behind Adaboost is to line the weights of classifiers and coaching the info sample in every iteration such it ensures the correct predictions of bizarre observations.

At a high level, AdaBoost is analogous to Random Forest in this they each tally up the predictions created by every call trees at intervals the forest to come to a decision on the ultimate classification. There square measure but, some refined variations. for example, in AdaBoost, the choice trees have a depth of one (i.e. 2 leaves). additionally, the predictions created by every call tree have variable impact on the ultimate prediction created by the model.

It works in the following steps [17]:

1. Initially, Adaboost selects a coaching set indiscriminately.

2. It iteratively trains the AdaBoost machine learning model by choosing the coaching set supported the correct prediction of the last coaching.

3. It assigns the upper weight to wrong classified observations in order that within the next iteration these observations can get the high likelihood for classification.

4. Also, It assigns the load to the trained classifier in every iteration in line with the accuracy of the classifier. The additional correct classifier can get high weight.

5. This method repeat till the entire coaching information fits with none error or till reached to the desired most range of estimators.

6. To classify, perform a "vote" across all of the educational algorithms you engineered.

In this project we tend to used ADABoost classifier to classify totally different tweets extracted from twitter supported numerous options into spam or quality, however the twitter is tormented by an excellent deal of choices.

Initially we want to form a coaching dataset that contains info concerning the tweets, as well as some options needed like following, followers, actions, isretweet, location and specific labels i.e., spam, quality. we tend to should separate the columns (attributes or features) of the dataset into input patterns (X) and output patterns (Y). we are able to try this simply by specifying the column indices within the NumPy array format. Finally, we tend to should split the X and Y data into a coaching and test dataset.

The training set will be used to prepare the ADABoost model and the test set will be used to make new predictions, from which we can evaluate the performance of the model.

For this we will use the **train\_test\_split()** function from the scikit-learn library. We also specify a seed for the random number generator so that we always get the same split of data each time this example is executed.

The ADABoost model for classification is called **AdaBoostClassifier**. Here We have used random\_state=42 , n\_estimators=800, learning\_rate=2.

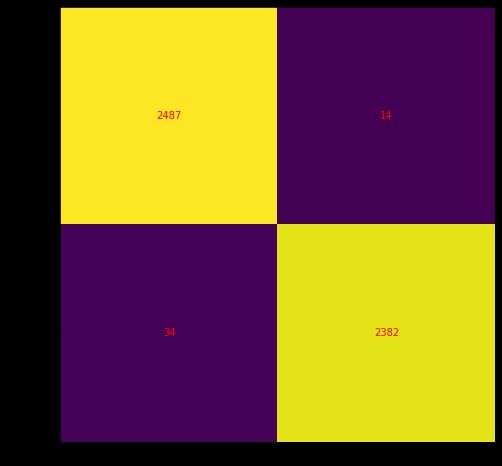
We can create and and fit it to our training dataset. Models are fit using the scikit-learn API and the **ada.fit()** function.

We can make predictions using the fit model on the test dataset.

To make predictions we use the scikit-learn function **model.predict()**.

We can use confusion\_matrix() function by taking the test dataset and predict model for this.

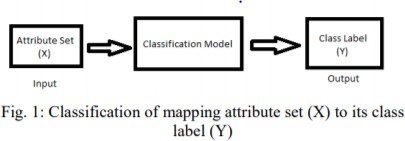
Finally, we evaluate the model using a confusion matrix. The model finished with with 14 false positives and 34 false negatives.This is how plot of confusion matrix looks like



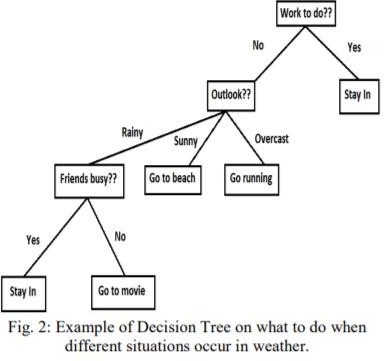
Now that we have used the fit model to make predictions on new data, we can evaluate the performance of the predictions by comparing them to the expected values. For this we will use the built in **accuracy\_score()** function in scikit-learn.

* **DECISION TREES:**

Classification is the task of giving objects to categories which have many diverse

applications.

[18]Decision Tree –It is a general tree that has a root, branches and leaf nodes. The similar design is present in the Decision Tree. It contains a root node, branches, and leaf nodes. Testing an attribute is on every internal node, the outcome of the test is on branch and class label as a result is on leaf node. The root is the parent of every node present and as per the name it is the first node in the tree. A decision tree may be a tree where each node shows a feature (attribute), each link (branch) shows a choice (rule) and every leaf shows an outcome (categorical or continuous value) . As decision trees mimic the human level thinking so it’s so simple to grab the data and make some good interpretations. The whole idea is to form a tree like this for the complete data and process one outcome at every leaf.

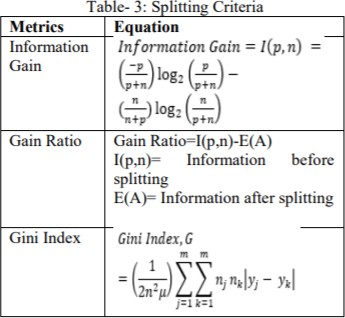


# [18]

The algorithms help to classify the attributes to test at any node in order to prove if the classification is good in individual classes. The partition at each branch is independent as possible, for that classifying rules are identical.

|  |  |
| --- | --- |
| **Algorithm name** | **Classification** |
| CART | Gini Index as metric |
| ID3 | Entropy and IG as metrics |
| C4.5 | Similar to ID3 |
| C5.0 | Improved version of C4.5 |
| CHAID | Precedes the ID3 |
| MARS | Finds the best classification |

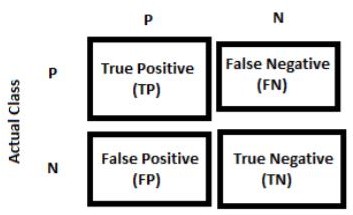
# Metrics



[18]

**Evaluation:-**

The values are said to be precise when they are very near to each other. If their averages are very near to each other the set is accurate. Only if given a group of knowledge from repeated measurements of the equivalent quantity then one can measure above two terms.



**Confusion Matrix in a DT**

[18]

Here in our classification we use DT classifier to classify tweets as spam or quality based on many features.

Firstly the training dataset is created that has all info related to features and specific labels which is the final output. Then the testing dataset is prepared in the same way as the other classification algorithms.

The training set is useful in building the model whereas the testing set is involved in making new predictions, from which the accuracy and precision are calculated.

The train\_test\_split method is used to split them into two datasets.

The parameters like random state estimators and learning rate are given accordingly.

The dt.fit method is used to fit the model into our dataset so that the evaluation is done on the test set.

The predictions are made using dt.predict() method.

The confusion matrix is thus obtained by using confusion\_matrix() giving the test dataset and model as parameters.

**2236, 12**

**6, 1835**

Now we can evaluate the performance of the model using accuracy\_score() method from the scikit-learn package.

Accuracy: 0.6503965832824893

precision recall f1-score support

0 0.70 0.55 0.61 2501

1 0.62 0.76 0.68 2416

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| accuracy |  | 0.65 | 4917 |  |
| macro avg | 0.66 | 0.65 | 0.65 | 4917 |
| weighted avg | 0.66 | 0.65 | 0.65 | 4917 |

**4. EXPERIMENTAL INVESTIGATIONS AND RESULTS**

**4.1 Dataset**

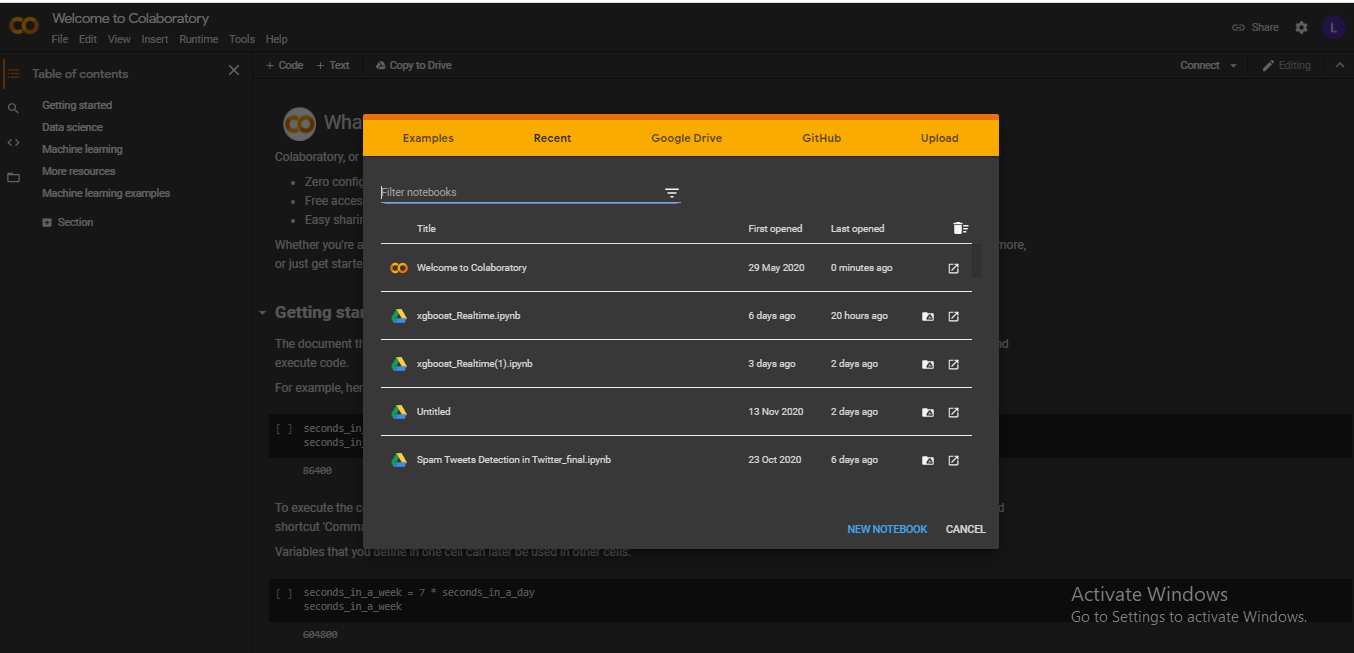
The data base that has been used in this project is collected from the Kaggle by that we had started to understand the data and tried to classify the tweets through classification algorithms and suggest the best algorithm for the prediction of spam tweets from twitter.

**Graphical user interface, application, table, Excel

Description automatically generated**

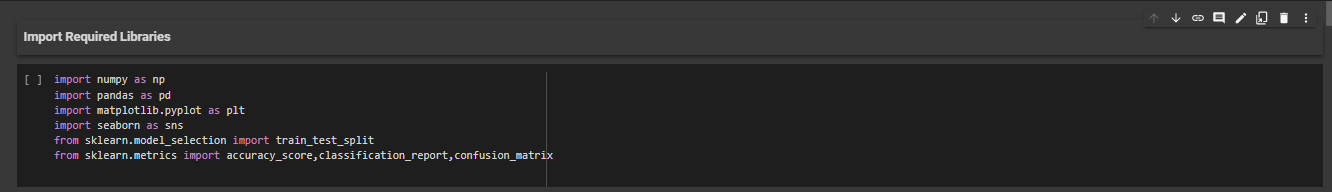
**Fig 1: Dataset**

**4.2 Opening** **Google Colab**

****

**Fig 2: Opening Google Colab**

**4.3 Importing Required Libraries**

****

**Text

Description automatically generated**

**4.4 Loading & Exploring the Data**

We loaded the dataset into the notebook using the required libraries and then explored the dataset, we notice that the dataset is well balanced as we can see in the below countplot there is equal class distribution of the output variable i.e. “Type”.

**A screenshot of a computer

Description automatically generated**

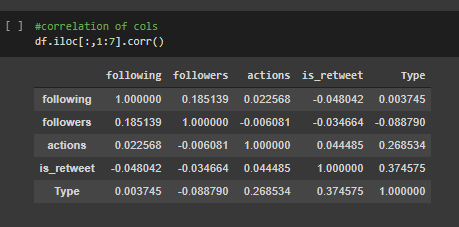
**Graphical user interface, text, application

Description automatically generated**

**Fig 4: Loading & Exploring the Data**

**4.5 Data Preprocessing and Visualization**

After Preprocessing the data by performing Exploratory Data Analysis i.e., EDA which involved replacing the null values with their respective mean values and created a **Correlation Matrix** using Pandas and we plotted a visual representation of the **Correlation Matrix** using Seaborn and Matplotlib.

****

**Fig 5: Correlation Matrix**

**Text

Description automatically generated Text

Description automatically generated**

**Fig 6: Checking the NULL values Fig 7: Scalar Values of all Columns**

**Chart, treemap chart

Description automatically generated**

**Fig 8: Visualizing Correlation Matrix using Seaborn Heatmap**

**A picture containing window

Description automatically generated**

**Fig 9: Visualizing Data with Pairs Plot to see the distribution of Data**

**Chart, histogram

Description automatically generated**

**Fig 10: Visualizing Histogram plot****with respect to tweet length**

**4.6 Text Pre- Processing with NLTK**

To perform textual data analysis, we need to preprocess all the tweets for that we used NLTK library which is **Python's Natural Language** Toolkit used for Text Preprocessing which involved in the steps like:

|  |  |
| --- | --- |
| * Convert text to lowercase | * Punctuation removal |
| * White spaces removal | * Tokenization |
| * Remove stop words | * Stemming |
| * Lemmatization | * Part of speech tagging (POS) |
| * Build Corpus |  |

Text

Description automatically generated

**Fig 10: Text Preprocessing using NLTK and building Corpus**

**4.7 Principal Component Analysis on Embedded Tweets data**

As we here working on the tweets data form a large dataset of each word. To reduce the dimensionality, we usedPCA as a dimension reduction tool to reduce large set of attributes to a small set without loss of information.

Graphical user interface, text

Description automatically generated

**Fig 11: Dimensionality Reduction using PCA**

**4.8 Create training and testing data**

To train a model and measure its accuracy we need to split the dataset into training (67%) and testing (33%) subsets of data.

A screenshot of a computer

Description automatically generated

**Fig 12: Training and Testing Subsets**

**4.9 Model Building**

To predict the twitter tweets, we need to build a model using the existing dataset made use of Classification algorithms of Datamining techniques. In our project we use classification algorithms like XGBoost, AdaBoost, Random Forest, Decision Tree and Logistic Regression to classify the tweets into labels i.e., Spam, Quality.

Graphical user interface, text

Description automatically generated

**Fig 13: Model Building using XGBoost**

Text

Description automatically generated

**Fig 14: Model Building using AdaBoost**

**Text

Description automatically generated**

**Fig 15: Model Building using Random Forest**

**Text

Description automatically generated**

**Fig 16: Model Building using Decision Tree**

**Text

Description automatically generated**

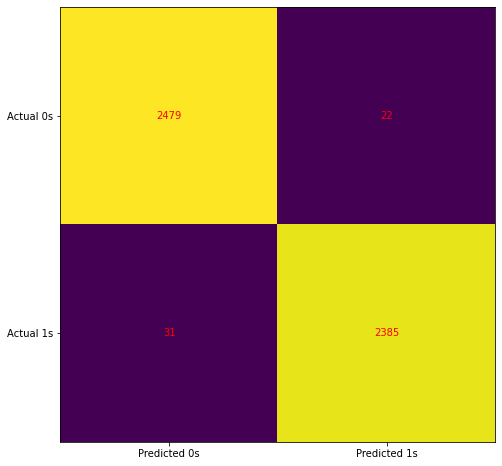
**Fig 17: Model Building using Logistic Regression**

**4.10 Evaluating Model and finding Accuracy using Testing data**

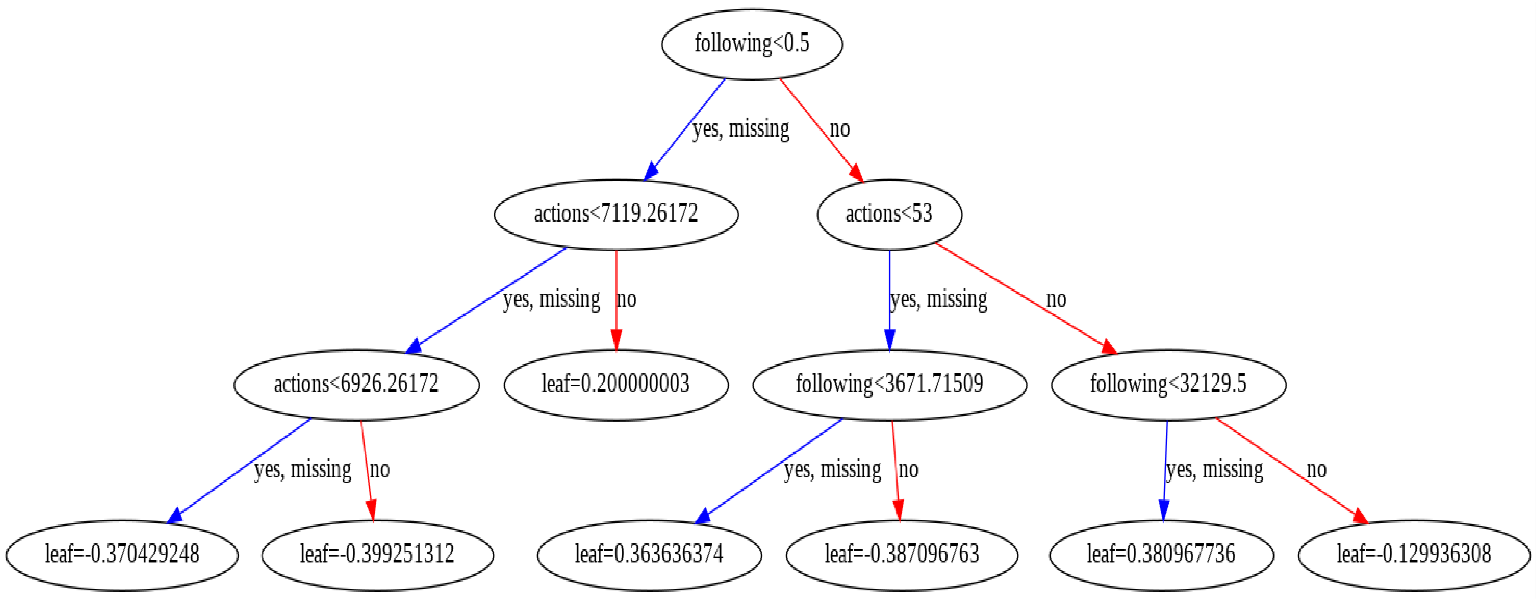
Text

Description automatically generated

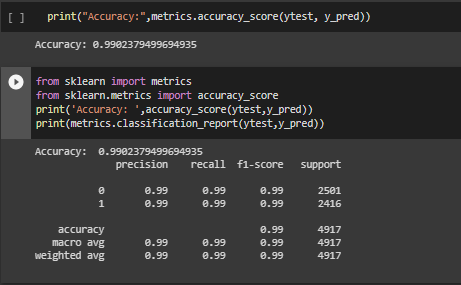
**Fig 18: Accuracy score for XGBoost Classifier**

****

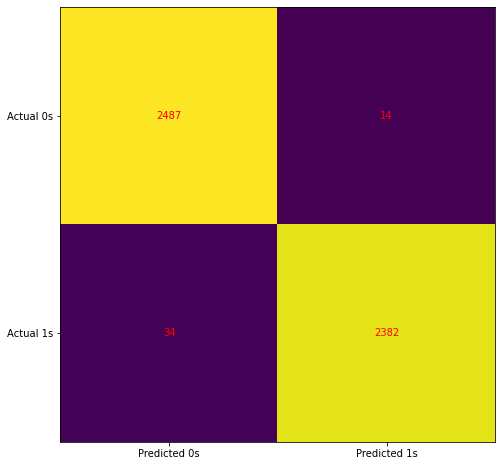
**Fig 19: Confusion Matrix plot for XGBoost Classifier**

****

**Fig 20: Visualization of XGBoost Model**

****

**Fig 21: Accuracy score for ADABoost Classifier**

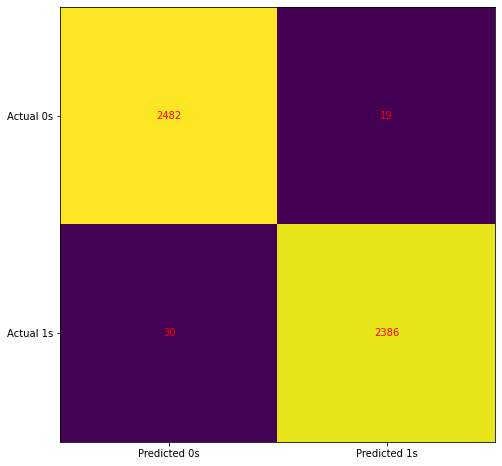
****

**Fig 22: Confusion Matrix plot for ADABoost Classifier**

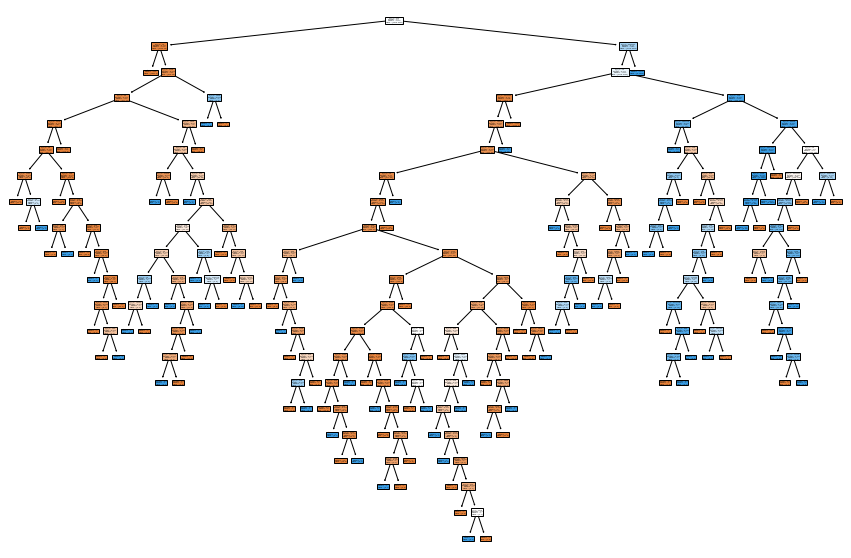
**A picture containing text

Description automatically generated**

**Fig 23: Accuracy score for Random Forest Classifier**

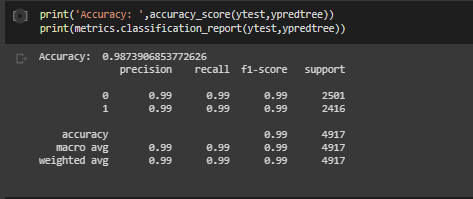
****

**Fig 24: Confusion Matrix plot for Random Forest Classifier**

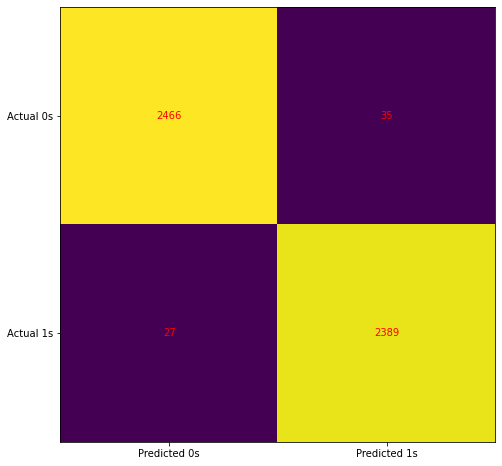
****

**Fig 25: Visualising 11th decision tree of random forest Classifier.**

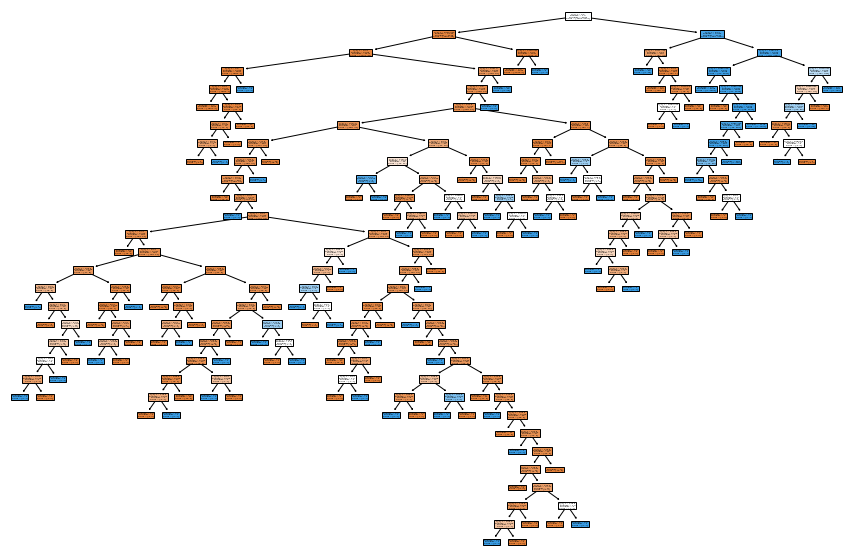
**Fig 23: Accuracy score for Random Forest Classifier**

****

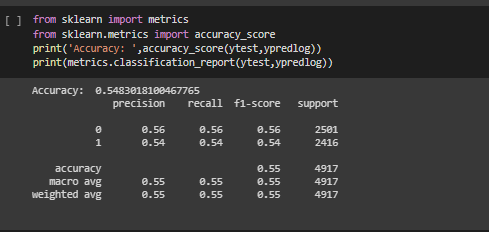
**Fig 26: Accuracy score for Decision Tree Classifier**

****

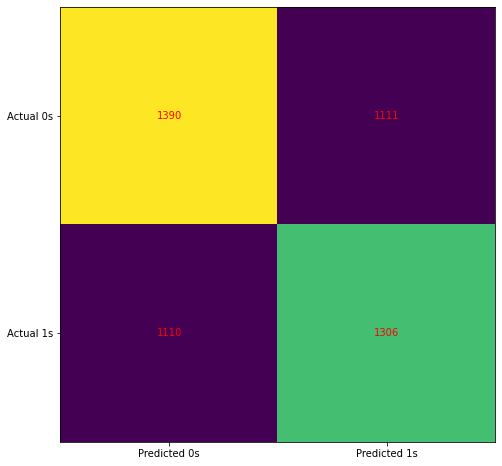
**Fig 27: Confusion Matrix plot for Decision Tree Classifier**

****

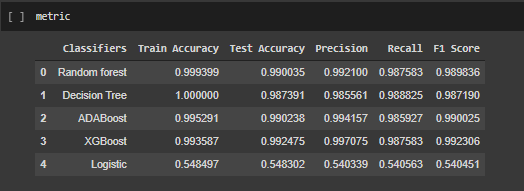
**Fig 28: Visualising Decision Tree Classifier.**

****

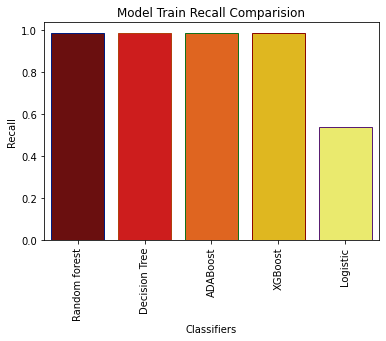
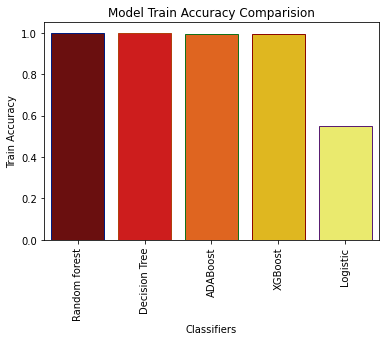
**Fig 29: Accuracy score for Logistic Regression Classifier**

****

**Fig 30: Confusion Matrix plot for Logistic Regression Classifier**

****

**Fig 31: Matrix of metrics for used Algorithms**

****

**Fig 32: Comparison of Accuracy and Recall between Classifiers**

**4.11 Extracting Tweets from Twitter**

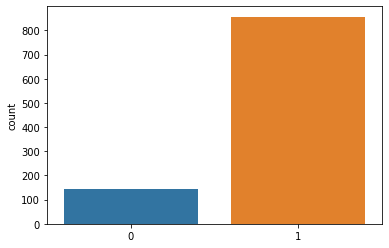
To predict on real time tweets we collect tweets from twitter through developer account. We collect 1000 tweets with a hashtag #TAEYANG and created the dataset with required fields for our model to predict. To extract these, we used “rtweet” package from R.

**Table

Description automatically generated**

**Fig 33: Dataset of Extraction of Tweets from twitter**

**4.12 Prediction On Real-Time Tweets**

** **

**Fig 34: Predictions using XGBoost Fig 35: Predictions using ADABoost**

****

**Fig 36: Prediction Using Random Forest Fig 37: Prediction Using Decision Tree**

**A picture containing text, crossword puzzle, receipt

Description automatically generated**

**Fig 38: Predicted Tweets**

**5. DISCUSSION OF RESULTS**

In this section we discuss about the experimental analysis and evaluation of results of the proposed model for detecting spam in tweets. We made use of machine learning Classification Algorithms to classify these tweets. Generally, these classification algorithms are used for predictions, finding hidden patters from the data. In this project to predict the tweets and classify them to class labels of output i.e., Quality and Spam we used machine learning datamining classifiers like Random forest, Decision tree, XGBoost, AdaBoost and Logistic regression in our approach for building our model for proposed system.

From our models we have measure the performance of Random forest, Decision tree, XGBoost, AdaBoost and Logistic regression Machine Classifiers. Then we compared all the constructed methodologies performance. For evaluating models, we used standard accuracy metrics such as testing accuracy, precision, recall and F measure.

* **Testing accuracy** is checking the accuracy of new data which is provided for the trained model.
* **Precision** is one of the standard metrics that is a measure of classifier’s exactness. The lower precision denotes that there are large number of false positives in the result. Precision is calculated by considering the number of True Positives predicted to that of the number of true positives and false positives.
* **Recall** it is the measure of our model correctly identifying True Positives. Simply it measures the classifier’s completeness. A lower recall represents presence of many false negatives in our predicted result. Its is the ratio of total true positives to that of true positives and false negatives.
* **F1 Score** is a metric used when both precision and recall metric are required to measure the performance of the classifiers. This metric measures the association between recall and precision.

To classify tweets with the discussed models we need to reduce the dimensionality of the processed data here we used Principal Component Analysis to make it a smaller dataset without loss of true information. Then we divided our dataset into training and testing subsets of 67% and 33% respectively.

To train and evaluate the results obtained when constructing machine learning classifiers, we collected the Twitter dataset from Kaggle which consists of tweets extracted from twitter and grouped into classes like Quality and Spam and the obtained results are as follow:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Classifier** | **Testing Accuracy** | **Precision** | **Recall** | **F1 Score** |
| Random Forest | 0.990 | 0.992 | 0.987 | 0.989 |
| Decision Tree | 0.985 | 0.985 | 0.988 | 0.987 |
| AdaBoost | 0.994 | 0.994 | 0.985 | 0.990 |
| XGBoost | 0.997 | 0.997 | 0.987 | 0.992 |
| Logistic Regression | 0.540 | 0.540 | 0.540 | 0.540 |

**Tab 1: Comparison on performance of Classifiers**

From the above obtains results we have been observed that, for most of the datasets samples the performance measure is above 98% for Random forest, Decision tree, XGBoost, AdaBoost classifiers when compared logistic regression. We conferred a comparative study of our approach with the constructed classifiers in which they have been trained with various spam detection algorithms on Twitter data. For every classifier, the same evaluation metrics are calculated for spam detection. The performance comparison of the proposed classifiers is plotted below, we can be observed that from the below visualization that the proposed method outpaced in terms of testing accuracy, precision, recall and F1 score metrics which are considered as standard measures for evaluating a model.

Chart, bar chart

Description automatically generated

**Fig 1: Performance comparison of results**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Classifier** | **Precision** | | **Recall** | |
| Spam | Quality | Spam | Quality |
| Random Forest | 0.99 | 0.99 | 0.99 | 0.99 |
| Decision Tree | 0.99 | 0.99 | 0.99 | 0.99 |
| AdaBoost | 0.99 | 0.99 | 0.99 | 0.99 |
| XGBoost | 0.99 | 0.99 | 0.99 | 0.99 |
| Logistic Regression | 0.54 | 0.56 | 0.54 | 0.56 |

**Tab 2: Comparison on classification based on precision and recall**

**6. SUMMARY, CONCLUSION & RECOMMENDATIONS**

**6.1 Summary:-**

In this project we have used Machine Learning techniques to identify Spam tweets and Quality tweets. Initially we found a dataset on the internet on which we have performed the analysis based on various features like repost that will help us in identifying the type of tweet(spam/quality). The usefulness of the tweet has been decided with the help of various classifiers like Decision Tree Classifier, AdaBoost Classifier, Logistic Regression, Random Forest Classifier, XGBoost Classifier. The Principal Component Analysis was performed on the dataset in order to increase the performance of the model by us.We then built the respective model upon the training and testing datasets. We calculated the evaluation metrics like Testing accuracy, Precision, Recall, F1 score to know the working of the model. Finally we chose the model that gives the highest result to be the accurate one. In order to increase the scope of the model built we have performed a real time analysis of tweets by extracting the everyday tweets from the Twitter API using tweepy package. This needed a developer account because Twitter has some restrictions regarding the access of tweets. This made our project model even more fascinating that it can be used as a Spam classifier in real time.

**6.2 Conclusion:-**

Twitter has been a very popular and lay-man-usable microblogging social media platform. It has a very beautiful architecture that any kind of malpractices can happen to users who are unaware of the consequences. Various kinds of attacks like phishing, hacking, stealing of private data, use of child-pornography can happen on this platform. For the fact that Twitter is very different from general mailing websites and service providers the spam filters cannot identify it very easily. Hence all the above mentioned algorithms are used in spam detection. Of all the models built the logistic regression did not perform so well that its accuracy was 54%. The Random Forest gave an accuracy of 82.7% initially. After performing the PCA it was improved to 99%. Similarly Decision Tree gave an initial accuracy of 80.02% which was later improved to 98.5%. Later two new models were added to the project. They are AdaBoost and XGBoost. They gave an overwhelming accuracy of 99.4% and 99.7% respectively. Discussing the XGBoost model we can achieve a precision of 0.997 and an F1 score of 0.992. Thus We Conclude our Project.

**6.3 Recommendations:-**

Spammers are the matter in any online social networking sites. Once a spammer is detected it's easy to suspend his/her account or block their IP address. This research deals with the study of spam classification techniques in twitter. Twitter API is developed to gather real dataset from Twitter public available information. the important time tweet dataset was obtained and therefore the document was pre-processed. Various algorithms were utilized for classifying the tweets dataset as spam and non-spam. The results are found to be quite satisfactory in terms of accuracy, precision and recall. The results have also been compared to other algorithms which are implemented. In future, other algorithms are often implemented for real time tweets comparison purpose and for better results than these classifiers. Also this model are often made into an API and used for not only twitter but also various applications for instance Messaging in Mobile Phones or WhatsApp Spam Detection and lots of others too.

**7.REFERENCES**

|  |  |
| --- | --- |
| [1] | David Sayce, "Dsayce Digital Marketing Consultant," 10 December 2020. [Online]. Available: https://www.dsayce.com/social-media/tweets-day/. |
| [2] | K. O. Hinesley, "Twitter," 11 February 2019. [Online]. Available: https://blog.twitter.com/en\_au/topics/company/2019/Spamm-ybehaviour-and-platform-manipulation-on-Twitter.html. |
| [3] | K. Resul and T. K. Abdullah, "A Survey of Spam Detection Methods on Twitter," *IJACSA,* vol. 8, no. 3, pp. 29-38, 2017. |
| [4] | P. Nasira , S. M. Malik.M and R. Qaisar, "Sentiment Based Twitter Spam Detection," *IJACSA,* vol. 7, no. 7, pp. 568-573, 2016. |
| [5] | V. Monika , Divya and S. Sanjeev, "Techniques to Detect Spammers in Twitter- A Survey," *International Journal of Computer Applications,* vol. 85, no. 10, pp. 27-32, 2014. |
| [6] | Rohini, K. Sunil and Jaiswal, "Improving Spam Detection on Online Social Media with hybrid classification techniques on Twitter platform," *IJSDR,* vol. 3, no. 6, pp. 26-32, 2018. |
| [7] | A. N.Noor, C. S.Vishnu, J. Suman and G.Tejaswi, "Spam Detection Framework for Twitter using ML," *IJITEE,* vol. 9, no. 6, pp. 216-219, 2020. |
| [8] | M. Claudia, B. Federica, G. Paolo and Z. D. Rodolfo, "Machine Learning Techniques applied to Twitter Spammers Detection," *IEEE,* p. 6, 2014. |
| [9] | M. Vanyashree, K. Anvaya, V.M.Sukanya and S.Rachana, Text-Based Spam Tweets Detection Using Neural Networks, Singapore: Springer, 2020. |
| [10] | S. Nan, L. Guanjun, Q. Junyang and R. Paul, "Near real-time twitter spam detection with machine learning techniques," *International Journal Of Computer and Applications,* pp. 1-11, 2020. |
| [11] | K. L. Jagtap, Dr.Bamu and B.A.Khansole, "Machine Learning Approach For Spam Tweets Detection," *IJIRCCE,* vol. 5, no. 7, pp. 13114-13122, 2017. |
| [12] | R. K.Subba and R. E.Srinvasa, "Detecting Spam Messages in Twitter Data by Machine learning Algorithms using Cross-Validation," *IJITEE,* vol. 8, no. 12, pp. 2941-2946, 2019. |
| [13] | T. K. Miss.Shukla and Prof.D.B.Kshirsagar, "Design of Machine Learning Approach For Spam Tweet Detection," *IJARIIE,* vol. 2, no. 5, pp. 626-631, 2016. |
| [14] | Vinodhini.M, Prithvi.D and Balaji.S, "Spam Detection Framework using ML Algorithm," *IJRTE,* vol. 8, no. 6, pp. 5326-5329, 2020. |
| [15] | Dinesh, "analyticssteps," 21 August 2019. [Online]. Available: https://www.analyticssteps.com/blogs/introduction-logistic-regression-sigmoid-function-code-explanation. |
| [16] | R. Dwivedi, "analyticssteps," 03 July 2020. [Online]. Available: https://www.analyticssteps.com/blogs/introduction-xgboost-algorithm-classification-and-regression. |
| [17] | A. Navlani, "datacamp," 20 November 2018. [Online]. Available: https://www.datacamp.com/community/tutorials/adaboost-classifier-python. |
| [18] | H. Harsh and P. Purvi, "Study and Analysis of Decision Tree Based Classification Algorithms," *JCSE,* vol. 6, no. 10, pp. 74-78, 2018. |