© 2018 IJSRSET | Volume 4 | Issue 8 | Print ISSN: 2395-1990 | Online ISSN : 2394-4099
Themed Section : Engineering and Technology

Sentiment Analysis on Tweets

K. Amaravathi*, N. Lokeswari
CSE, Anil Neerukonda Institute of Technology & Sciences, Visakhapatnam, Andhra Pradesh, India

ABSTRACT

The rapid increase of textual data is overwhelming. The huge amount of data generated from the different call sites, customer reviews on different products, and so on is in the unstructured form. So the amount of textual centre logs, emails, blogs, documents on the internet, tweets form twitter, customer comments from different social networking data rapidly increasing makes the summarization task challenging for businesses. In this paper we provide techniques that how to organize textual data and analyze textual data for extracting intuitive customer intelligence from a large collection of data. The predominant and different types of information on twitter make it one of the most appropriate virtual environments for information monitoring and tracking. In this paper, starting with the analysis of different hash-tags, categorization of different areas, identification of influence, and finally analysis of sentiment. This paper tries to draw some important conclusions based on the sentiment analyzed.

Keywords: Data Mining, Machine Learning, Natural Language Processing, Twitter, Prediction.

I. INTRODUCTION

Sentiment Analysis is the process of determining whether a piece of writing (product/movie review, tweet, etc.) is positive or negative. It can be used to identify the customer or follower's attitude towards a brand through the use of variables such as context, tone, emotion, etc. Sentiment analysis could be used to analyze or predict. It has seen a vast number of applications in recent time as 80% of data available is unstructured in the form of text, audios, videos etc[6]. Here is the step by step approach on how it was done:

Gathering data: Kaggle is a platform for predictive modelling and analytics competitions in which statisticians and data miners compete to produce the best models for predicting and describing the datasets uploaded by companies and users.
We have used a dataset from Kaggle on demonetization.

The details of the dataset are as follows.
Context of the dataset - The demonetization of 500rs and 1000rs banknotes was a step taken by the Government of India on 8 November 2016, ceasing the usage of all 500rs and 1000rs banknotes of the Mahatma Gandhi Series as a form of legal tender in India from 9 November 2016. The announcement was made by the Prime Minister of India Narendra Modi in an unscheduled live televised address to the nation at 20:15 Indian Standard Time (IST) the same day. In the announcement, Modi declared circulation of all 500rs and 1000rs banknotes of the Mahatma Gandhi Series as invalid and announced the issuance of new 500rs and 2000rs banknotes of the Mahatma Gandhi New Series in exchange for the old banknotes. Content - The data contains 14941 most recent tweets on #demonetization. There are 14941 rows(one for each tweet) and 14 columns. Metadata, Text (Tweets), favorited, favoriteCount, replyToSN,created,

truncated, replyToSID, id, replyToUID, statusSource, screenName, retweetCount, isRetweet, retweeted[5]. From the above detailed dataset we have used text column for our analysis and generated a class column for it.

Text cleaning: Stop-words removal, punctuation removal, blank spaces removal and so on were few of the techniques used to clean the text. This is common to every text analysis problem.

Sentiment generation: It is then important to fine the sentiment of the statement, there are many smart and reliable algorithms to tag statements with complex or simple sentiments. Some sentiment algorithm would also give us sentiments like positive and negative. Natural language processing is used to tag each word with a sentiment and the overall score or sentiment that the statement would get depends on the underlying word sentiments.

Prediction: This was found by prediction method: data was divided into 2 sets training and testing (70%–30%). Different algorithms were used to learn the behavior of sentiments and then 30% is used as unseen data to test the extent to which it could predict.

The organization of this document is as follows. In Section 2 (**classification algorithms**), describes different classification algorithms. In Section 3 (**Experimental Results**), present execution time of different classification algorithm. Final Section 4(**Conclusion**) ,concludes the paper.

II. CLASSIFICATION ALGORITHM

In the current system, the sentiment of the tweets is obtained by using one machine learning algorithm. This sentiment analysis is done with the help of fuzzy logic (deals with reasoning and gives closer views to the exact sentiment values).

In the proposed system, we are implementing various algorithms for the classification, such as Naive Bayes, Logistic Regression, SVM, Random Forest, Decision Tree and analyse the outputs of each one.

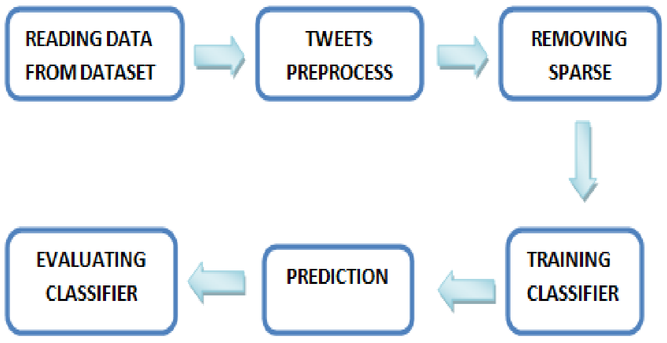

Figure 1. Architecture

Algorithms used

A. Naive Bayes
B. Logistic Regression
C. Random Forest
D. Decision Tree
E. SVM

A. Naive Bayes

The **Naive Bayes Classifier** technique is based on the so-called Bayesian theorem and is particularly suited when the dimensionality of the inputs is high. Despite its simplicity, Naive Bayes can often outperform more sophisticated classification methods. One of the easiest ways of selecting the most probable hypothesis given the data that we have that we can use as our prior knowledge about the problem. Bayes' Theorem provides a way that we can calculate the probability of a hypothesis given our prior knowledge. In a classification problem, our hypothesis (h) may be the class to assign for a new data instance (d) [2].

Bayes' Theorem is stated as:

$$P(h|d) = (P(d|h) * P(h)) / P(d)$$

Where,

- ✓ **P(h|d)** is the probability of hypothesis h given the data d. This is called the posterior probability.

- ✓ **P(d|h)** is the probability of data d given that the hypothesis h was true.
- ✓ **P(h)** is the probability of hypothesis h being true (regardless of the data). This is called the prior probability of h.
- ✓ **P(d)** is the probability of the data (regardless of the hypothesis).
- ✓ It is called *Naive Bayes* because the calculation of the probabilities for each hypothesis are simplified on the assumption that every attribute is independent of the other making the calculation tractable.

B. Logistic Regression

Logistic regression is named for the function used at the core of the method, the logistic function. The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve shown in figure 2 that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

$$1 / (1 + e^{-value})$$

Where e is the base of the natural logarithms and value is the actual numerical value that we want to transform. Below is a plot of the numbers between -5 and 5 transformed into the range 0 and 1 using the logistic function.

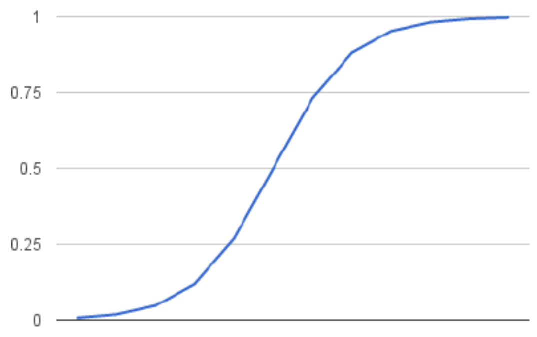

Figure 2. S- Shaped Curve

Representation Used for Logistic Regression-

Logistic regression uses an equation as the representation, very much like linear regression.

Input values (x) are combined linearly using weights or coefficient values (referred to as the Greek capital letter Beta) to predict an output value (y). A key difference from linear regression is that the output value being modeled is a binary values (0 or 1) rather than a numeric value.

Logistic regression is a linear method, but the predictions are transformed using the logistic function. The impact of this is that we can no longer understand the predictions as a linear combination of the inputs as we can with linear regression, for example, continuing on from above, the model can be stated as:

$$p(X) = e^{(b0 + b1*X)} / (1 + e^{(b0 + b1*X)})$$

Learning the Logistic Regression Model -

The coefficients (Beta values b) of the logistic regression algorithm must be estimated from your training data. This is done using maximum-likelihood estimation.

Maximum-likelihood estimation is a common learning algorithm used by a variety of machine learning algorithms, although it does make assumptions about the distribution of your data.

Making Predictions with Logistic Regression -

Making predictions with a logistic regression model is as simple as plugging in numbers into the logistic regression equation and calculating a result.

Prepare Data for Logistic Regression -

The assumptions made by logistic regression about the distribution and relationships in your data are much the same as the assumptions made in linear regression.

- ✓ Binary Output Variable: This might be obvious as we have already mentioned it, but logistic regression is intended for binary (two-class) classification problems. It will predict the probability of an instance belonging to the default class, which can be snapped into a 0 or 1 classification.

- ✓ Remove Noise: Logistic regression assumes no error in the output variable (y), consider removing outliers and possibly misclassified instances from your training data.
- ✓ Gaussian distribution: Logistic regression is a linear algorithm (with a non-linear transform on output). It does assume a linear relationship between the input variables with the output. Data transforms of your input variables that better expose this linear relationship can result in a more accurate model. For example, you can use log, root, Box-Cox and other univariate transforms to better expose this relationship.
- ✓ Remove Correlated Inputs: Like linear regression, the model can overfit if you have multiple highly-correlated inputs. Consider calculating the pairwise correlations between all inputs and removing highly correlated inputs.
- ✓ Fail to Converge: It is possible for the expected likelihood estimation process that learns the coefficients to fail to converge. This can happen if there are many highly correlated inputs in your data or the data is very sparse (e.g. lots of zeros in your input data).

C. Random Forest

Random Forest is an ensemble learning (both classification and regression) technique. It is one of the commonly used machine learning technique. Ranger is a fast implementation of random forests.

In a random forest algorithm number of decision trees are built during the process. A vote from each of the decision trees is considered in deciding the final class of a case or an object, this is called ensemble process.

Since, many decision trees are built and used in a process of Random Forest algorithm, it is called a forest.

For a building a decision tree, samples of a data frame are selected with replacement along with selecting a subset of variables for each of the decision tree. Both sampling of data frame and selection of subset of the variables are done randomly, so first word "random" is arrived.

Key advantages of using Random Forest:

- Reduce chances of over-fitting
- Higher model performance or accuracy

Random Forest uses Gini Index based impurity measures for building decision tree.

Some of the commonly used parameters of Random Forest functions are -

x : Random Forest Formula

data: Input data frame

ntree: Number of decision trees to be grown

D. Decision Tree

Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, decision tree algorithm can be used for solving regression and classification problems too.

The general motive of using Decision Tree is to create a training model which can use to predict class or value of target variables by learning decision rules inferred from prior data (training data).

Decision Tree Algorithm Pseudo code

1. Place the best attribute of the dataset at the root of the tree.
2. Split the training set into subsets. Subsets should be made in such a way that each subset contains data with the same value for an attribute.
3. Repeat step 1 and step 2 on each subset until you find leaf nodes in all the branches of the tree.

In decision trees, for predicting a class label for a record we start from the root of the tree. We compare the values of the root attribute with record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node. We continue comparing our record's attribute values with other internal nodes of the tree until we reach a leaf node with predicted class value [1].

The below are the some of the assumptions we make while using Decision tree:

- ✓ At the beginning, the whole training set is considered as the root.
- ✓ Feature values are preferred to be categorical. If the values are continuous
- ✓ At the beginning, the whole training set is considered as the root.
- ✓ Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.
- ✓ Records are distributed recursively on the basis of attribute values.

Figure 3 shows order to placing attributes as root or internal node of the tree is done by using some statistical approach.

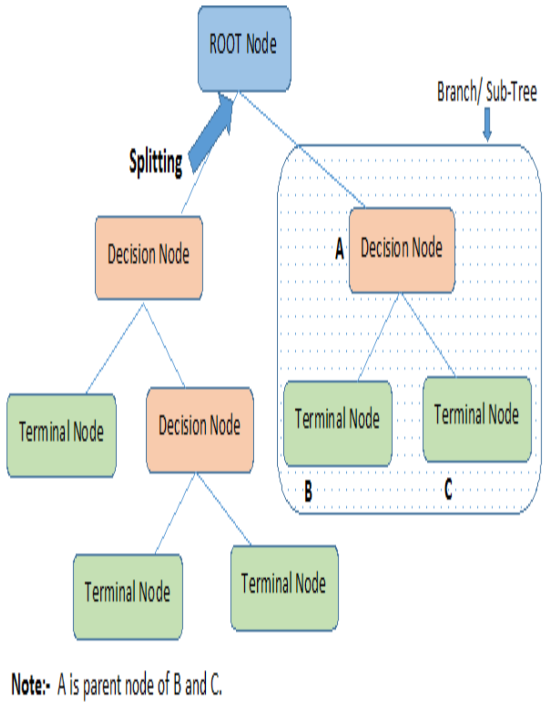

Figure 3. Node Structure of Decision Tree

E. SVM Classifier

What is Support Vector Machine?

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification and regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well [3,7].

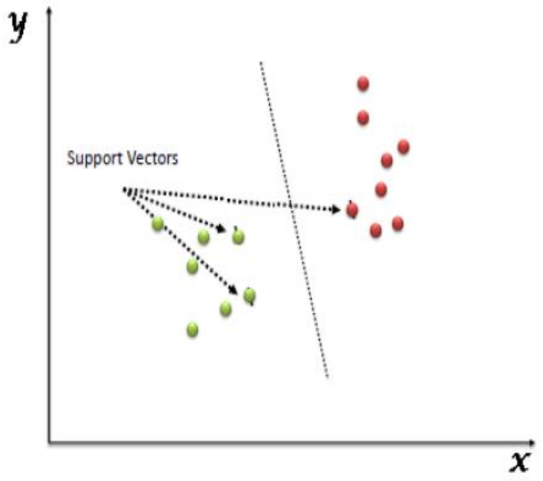

Figure 4. SVM classifier

Support Vectors are simply the co-ordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line).

The e1071 package in R is used to create Support Vector Machines with ease. It has helper functions as well as code for the Naive Bayes Classifier.

Tuning parameters value for machine learning algorithms effectively improves the model performance.

We should always look at the cross-validation score to have effective combination of these parameters and avoid over-fitting.

Mentioned below are the respective parameters for e1071 package:

- ✓ The kernel parameter can be tuned to take "Linear","Poly","rbf" etc.
- ✓ The gamma value can be tuned by setting the "Gamma" parameter.
- ✓ The C value in Python is tuned by the "Cost" parameter in R.

III. EXPERIMENTAL RESULTS

A. Divided Dataset into Modules

The results of accuracy for the test modules considered are (in percentage) –

TEST NAME	NAÏVE BAYES	SVM	DECISION TREE	RANDOM FOREST	LOGISTIC REGRESSION
TEST 1	89.32	91.83	90.00	93.33	88.15
TEST 2	88.33	89.83	90.83	92.50	81.50
TEST 3	88.35	93.67	91.33	94.67	96.33
TEST 4	88.61	92.07	90.41	92.40	83.33
AVERAGE	88.65	91.85	90.64	93.22	87.32

Figure 5. Results of accuracy for the test modules

B. Execution Time

Algorithm name	Classifier training time			Prediction time		
	User	System	Elapsed	User	System	Elapsed
Naive Bayes	1.93	0.17	2.19	132.47	0.05	133.23
Logistic Regression	6.75	0.05	7.06	0.04	0.00	0.05
Decision Tree	10.95	0.01	11.10	0.08	0.00	0.08
Random Forest	7.92	0.10	2.34	0.43	0.02	0.24
SVM Classifier	253.64	0.17	255.11	2.57	0.01	2.59

Figure 6. Execution time of Classification algorithms

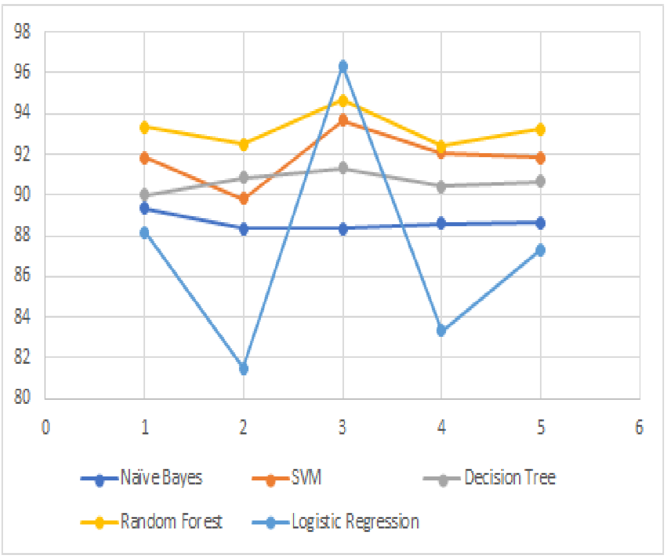

Figure 7. Accuracy result of Dataset

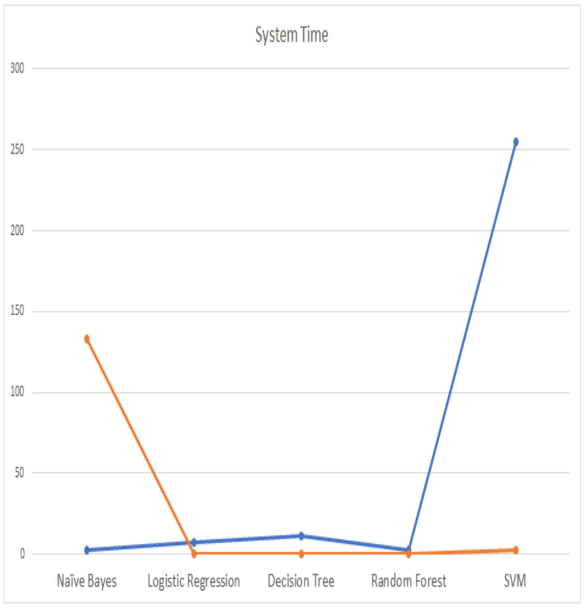

Figure 8. Graphical representation Execution time of Classification algorithms

IV. CONCLUSION

In this, we have implemented sentiment analysis algorithms like Naive Bayes, Logistic Regression, Random Forest, Decision Tree, SVM over the demonetization dataset that we used from Kaggle.
The accuracies on different algorithms for the dataset are as follows -

- ✓ Naive Bayes - 92.32%
- ✓ Logictic Regression - 88.24%
- ✓ Random Forest - 93.22%
- ✓ Decision Tree - 90.64%
- ✓ SVM Classifier - 91.85%

For our dataset considered, the highest accuracy is obtained in the Random forest algorithm implementation and the least is Logistic Regression.

V. REFERENCES

[1]. https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial- tree-based-modeling-scratch-in-python/

[2]. https://machinelearningmastery.com/naive-bayes-for-machine- learning/

[3]. https://rpubs.com/aka7h/simple-sentiment-analysis

[4]. Jiawei Han, Micheline Kamber and Jian Pei, "Data Mining - Concepts and Techniques" (3rd edition)

[5]. Amit G. Shirbhate1 , Sachin N. Deshmukh, "Feature Extraction for Sentiment Classification on Twitter Data", International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064 Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

[6]. Chakraborty, Goutam, and Murali Krishna Pagolu. "Analysis of unstructured data: Applications of text analytics and sentiment mining." In SAS global forum, pp. 1288-2014. 2014.

[7]. Huq, Mohammad Rezwanul, Ahmad Ali, and Anika Rahman. "Sentiment Analysis on Twitter Data using KNN and SVM." IJACSA International Journal of Advanced Computer Science and Applications 8, no. 6 (2017): 19-25.