Harshal Shah 1020849

Medha Mishra 976189

Chitra Naga Durga Sindhusha Veluguleti 1045391

SML Assignment 1: Who Tweeted That?
Project Report: Project Group 136

Abstract: **With the advent of social media, the amount of data being generated that can be processed has skyrocketed. With droves of new users joining social media every day, the amount of unique data specific to themselves that they generate in terms of the content is ever growing. Twitter is at the frontlines of doing this, and an excellent candidate for this project due to the added structure provided by the 280-character upper limit. To the human eye, there is little structure outside of the character limit confining the content of tweets. This report presents various techniques used to predict the author of an unlabelled tweet after using an exhaustive training dataset and some machine learning techniques.**

Keywords: Twitter, feature, word, tweets, SVM, Decision Tree, Gaussian NB, Bagging, Classifier.

INTRODUCTION:

Anyone who has experienced the maelstrom of human opinion and sentiment will tell you there is little uniformity on Twitter. With more than 320 million users, it becomes a Goliath of a task to wrangle any data generated on twitter for the entire platform due to the incredibly diverse content generated on in. Between updates about breakfast, ad bots, Twitter feuds between users and news updates, tweets have very diverse features and content. The project of “Who tweeted that?” as delivered to us, tasks us the with the task of predicting the authors of the tweets by implementing various statistical machine learning techniques using the datasets provided by Kaggle and the University of Melbourne.

FEATURE ENGINEERING:

Due to being bound by the large dataset and limited processing capacity of our computers, the following features listed in Table 2 were extracted. However, unbound by such constraint all the following features should be considered to improve accuracy. N-grams should only be implemented once ‘@handle’ and stop words are removed from the tweet and a matrix of generated N-grams should be used as features (Rani, Anuradha and Reddy, 2016). We can also use Vectorizer in Scikit learn which helps us to convert all the text into numerical features.

Basic Tweet Features- Features about tweet	
tweet Length	Number of words in a tweet
tweetCharacterLength	Total number of characters in tweet or Length of the tweet in characters
averageWordLength	Calculated as (Total Number of characters in words/Number of words).
hashtagCount	Number of hashtags in the tweet
is_ascii	Checks if the tweet has any non-ASCII characters. If any found, it is False, else True. Useful in checking if user is using a different language.
Author tweeting habits- Features about the tweeting preferences of the author	
is_directed	Checks if the tweet is directed at one or more other users. Is true if tweet is directed to other user(s) else False.
handles_mentioned	Number of other users referenced with ‘@’ in a tweet.
is_rt	Checks if the tweet is a retweet. If a tweet is a retweet, it is True, else False
emoji_count	Number of emoji used by author
urls	Number of URLs mentioned in the tweet.
Author Stylistic habits- Features about the twitter user’s tweeting lexicon	
exaggerated_writing	Checks if the twitter author tends to write with exaggerated spelling, e.g. “WOOOOOW”. Is true if exaggerated spelling is used else False
exaggerated_punctuation	Checks if the twitter author tends to write with exaggerated punctuation, e.g. “Nice!!!!”. Is true if exaggerated punctuation is used else False
isTitle	Checks if the first alphabet of every word in the tweet begins with an uppercase alphabet. Is true if such writing styles are employed else false
isDigit	Checks if the entire tweet only consists of digits. Is true if no nonnumeric characters present in tweet else false.
isUpper	Checks if entire tweet is written in the upper case. Is true if no lowercase alphabets in tweet else false.
numUpper	Number of words in upper case
numStopWords	Number of stop words used in the tweet

Table 1: All Features

Harshal Shah Medha Mishra Chitra Naga Durga Sindhusha Veluguleti
1020849 976189 1045391

Features Extracted	
tweetLength	tweetCharacterLength
hashtagCount	AverageWordlength
urls	numStopWords
isTitle	sentiment
isUpper	isDigit

Table 2: Features Extracted

MACHINE LEARNING MODELS:

This section describes the various statistical machine learning techniques used to predict the author of the tweet. Several models were used to train the data based on the features extracted. We tried implementing Support vector Machine using two kernels (linear, polynomial). Support vector machine using linear kernels helps us to run our code faster, but it is not a good approach as the data provided is not linearly separable. Support Vector machine using the polynomial kernel is a good choice as it helps us deal with nonlinear data. Higher the degree of freedom, the better it deals with the nonlinear data. We implemented the algorithm but did not successfully execute it because SVM takes a very long computation time which is approximately n^3 where n is the no of rows in the dataset and as our data is very huge it becomes an inefficient technique, unless used cleverly. We further implemented Decision Tree Classifier model which follows a tree like structure where the leaf represents the outcome and the internal node represents the feature. This model allows us to quantify the items based on probability which helps the final decision makers to make decisions on the basis calculated values which avoids any random errors. Decision Tree is a very good approach as it helps us understand the complex data in a visual format. This approach worked for us and we were successfully able to train our model and use it for further predictions as well. After, the Decision Tree approach, we tried implementing the group of Decision Tree named as the random forest classifier model, which we had to discard for training because of its endless running and our poor implementation. Going ahead, Bayes theorem is stated as:

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

where p(h|d) = posterior probability, p(d|h) = probability that the thesis is true, p(h) = prior probability and p(d) = probability of the data point (Brownlee, 2016). This classifier belongs to the family of “probabilistic classifiers”. We implemented the Gaussian naive Bayes model, which is based on the principle of prior, posterior distributions. We assume that each class follows a gaussian distribution. Another major requirement for the Bayesian model is the data must be numeric and that is why we had to use the label encoder to encode all the text features in numbers, which eventually helped us training our model but reducing the accuracy of our model as well. And finally, we implemented the Bagging Classifier along with the Multi-layer perceptron model and as MLP does not work on very large data, we used the bagging classifier along with it. Bagging classifier helps us divide the dataset into sub samples and train the model, make the predictions and thereby merging all the output into one before giving the outcome of the task to the user. This is how the bagging classifier helps us to work on the large data even with the simplest models. The reason behind choosing the Multi-Layer perceptron is that it is a neural network which can learn the function, **f(.):** $\mathbf{R^m \to R^o}$, by training on a dataset (Scikit Learn, 2019). All the models were implemented in python using the ski learn and keras library.

MODEL SELECTION:

Based on the accuracy and the execution time taken by all the models we tried and implemented, we chose Decision Tree classifier as our final model. Decision Tree classifier helps us dealing with nonlinear data and complex data in a very simple, visual tree-based manner. In this project, we need to identify the author of the tweet, which is a typical classification problem. Decision Tree helps us to analyse our decision in a broad spectrum of possibilities thereby getting better results at the end. The major issue with Decision Tree is the overfitting of the model. This model justifies the data by using more than one parameter. While using the Decision Tree approach, pruning of the tree can be done to reduce the overfitting of the model. Pruning is done by eliminating the branches of the tree which provide very little classification power and are of less importance. Pruning can be further divided into pre and post pruning. Also, there are many hyperparameters in the decision tree approach which can be tuned as and when needed for better model training. Some of them are: Max Depth, Min Samples, leaf, etc. Due to processing capacity constraints, we had very few options to choose our final model, so we decided Decision Tree as it worked the best for us. If any of the algorithm which we tried to implement such as multi-layer perceptron, support vector machine – polynomial kernel, random forest etc. would have worked perfectly for us then, we would have chosen that as our final model. The reason for selecting any of those models is, they work more efficiently, work faster and gives better results along with good accuracy. The figure 1 & 2 in the Critical Analysis section shows the execution time & accuracy taken by Decision Tree and gaussian naïve Bayes algorithm.

CRITICAL ANALYSIS:

As shown in the images below, Gaussian naïve Bayes takes 5 minutes execution time and gives us an accuracy of 0.8% while Decision Tree takes approximately 15 minutes of execution time but provides us with a better accuracy of 4.5%. There are many reasons for obtaining these results. As our features are basic, the execution time needed is not very high. We mentioned earlier that we have implemented only limited features and excluded features such as emoji count, the exaggerated alphabets and the exaggerated punctuations for training the model because of the time constraint, etc. If we consider those features as well, then the execution time will increase for Decision Tree, but it will improve the accuracy significantly. As far as Gaussian naïve Bayes is considered, it takes less execution time because we have used the label encoding technique. Label encoder converts every type of data into numbers. We did this because Naïve Bayes requires categorical data. But converting text to numerical data is of course not a very good strategy to work on and therefore it reduces accuracy to a very small number even though it takes less execution time. The features we have used to train the model are appropriate at the basic level and will work well if the dataset is not very large. All the features extracted are classified as the text-based features which can be generally extracted from any tweet (excluding pictures, videos, etc.). Our choice in terms of the model which we have implemented is restricted to a basic level, as we selected slow and suboptimal models over optimal models which we could not implement due to the constraints, but using the extended features and models which didn't work for us because of the time constraint will help to improve the accuracy of the system and predict the authors more accurately.

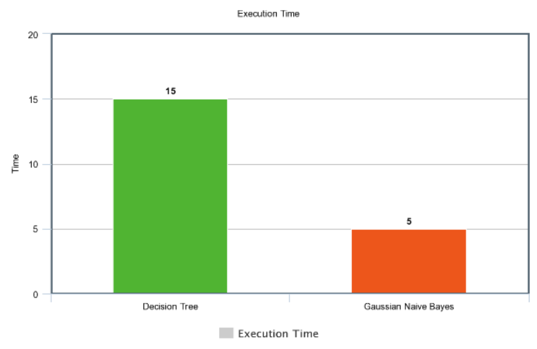

Figure 1: Execution time for Decision Tree and Gaussian Naïve Bayes Algorithm

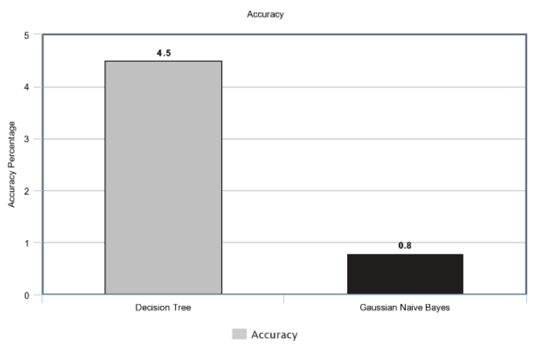

Figure 2: Accuracy for Decision Tree and Gaussian Naïve Bayes Algorithm

CONCLUSION AND FUTURE WORK:

The Suggested Decision tree and Gaussian Naïve Bayes approach that was implemented is a rudimentary approach towards solving the provided "who tweeted that?" problem which can be further improved a lot by being implemented for the full set features discussed in the report. Alternatively, Bagging Classifiers along with some other fast algorithms can be used. The basic learners such as SVM, Logistic Regression, Random Forest, etc. if used cleverly can help us achieve our goals. Neural networks and deep learning systems are ideal for tasks like this and would solve the problem in a more efficient manner.

REFERENCES:

1. Machine Learning Mastery (2016). Naïve Bayes for Machine Learning. Retrieved from https://machinelearningmastery.com/naive-bayes-for-machine-learning/
2. Neural Network models (2019). Multi-layer perceptron. Retrieved from https://scikit-learn.org/stable/modules/neural_networks_supervised.html
3. T. Rani, K. Anuradha and P. Reddy, "Sentiment Classification on Twitter data using word N Gram model", International Journal of Technology and Engineering Science IJTES, vol. 4, no. 1, pp. 7032-7035, 2016.