Text Mining and, Election Campaign Monitoring

Submitted to:

Dr. James Doherty

Dept. Computer Science

UCC Cork, Ireland.

Report prepared by:

Praveer Kumar

115220186@umail.ucc.ie

(M.Sc. Data Science and Analytics)

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Praveer Kumar
(M.Sc. Data Science and Analytics)

Signature: _____ Date: _____

Abstract

UNIVAC (Universal Automatic Computer) was the first computer to predict the election results in 1952. The UNIVAC was in Philadelphia where statistician Max Woodbury worked on the algorithm and did all the data entry by himself. The predictions turned out to be correct [1]. People realized the potential of the computers, but the importance of the statistics was yet to be re-discovered until recent years.

Almost half a century ago, the prediction science using computers depended on the accuracy of the data provided to the computers, which was completely manual. Today with the advent of the mobile "internet hooked devices", the amount of information has exploded. The social networking sites like Twitter, Facebook, and LinkedIn etc. are producing large amounts of data, easily accessible to developers and statisticians. Exploiting such data to get relevant insight into what people are thinking, general trends or, campaigns etc. is quiet natural and hence data analytics has taken the centre stage.

Text mining the social networking sites to monitor them is therefore the obvious way to go forward, but there are a lot of pitfalls that needs to be considered while exploring through such huge amounts of data. Nate Silver in the same context points out that IBM estimated that they are generating 2.5 quintillion bytes of data most of which was created in recent years. However, sheer volume of data is mistaken for the accuracy of the prediction. Data driven predictions can succeed and they can fail. It is our responsibility to construe them with meaning and also measure the odds of failure of the prediction [2].

So, in order to check the quality of the data produced on the internet, we analysed the social media especially twitter Feeds by extracting the data, measuring the user's opinion, and hence predicted the outcome and also measured its accuracy. In the given project, we used the example of election campaigns and, analysed user's opinion on these campaigns using the twitter feeds. Tools that were used to extract, process, save and visualize data are Python, MongoDB and Web Technologies.

Table of Contents

List of Figures

List of Tables

1. Introduction

a. Objective

The objective of the project is to build a model that can predict the likelihood of the votes by users from different places in a given country. The aim is to monitor the social networking sites to detect user's inclination for different parties in any bilateral governmental system and also measure the overall efficiency of the model in predicting the results thus helping to create better election campaigns.

b. Motivation

With the advent of cheap computing power, better networking capabilities and lesser processing time, there has been an outburst in the amount of data floating on the internet. The consumption of online information has grown exponentially. People today are voluntarily posting their views through the use of social media and micro blogging sites. This wealth of data has opened new horizons for everybody to develop better marketing, services and campaign strategies.

Less than a decade ago interviews and surveys were the only methods to gain feedback from the people. This methodology is time consuming, expensive and in a rapidly changing environment will also date rapidly and be unreliable.

Natural language processing and sentiment analysis today play a key role in identifying the behaviour and inclination of people towards a particular topic and therefore might act as a key factor in predicting their behaviour. So it will be important to analyse whether these behavioural patterns as posted on the social media could truly help in building better campaigns and strategies.

c. Contribution to knowledge

Analysis of sentiment and the voting patterns available on social networking sites aids understanding whether these sentiments help in predicting overall votes received. Consideration is also given to whether negative or positive sentiments help more in deciding the voting patterns of the total population. A detailed report is created for a "pre-known campaign" and its results are predicted from the chosen model. It is then compared to the actual result for its accuracy and a remark on the prediction quality and limitation is made. Also a generalised solution is proposed to recreate similar searches on the social networking site (Twitter).

2. Background

a. Data Mining

'Data mining' is a subdomain of computer science which focuses on discovering patterns in large datasets. It is the process of collection, cleaning and retrieval of relevant information through the use of advanced statistical techniques. It allows us to analyse data from different dimensions, categorize it and summarise the relationships identified [3].

In general, a data mining task can be divided into the following process [3]:

1. Business Understanding
2. Data Understanding
3. Data Preparation (pre-processing mainly cleaning or imputing)
4. Modelling
5. Evaluation/Diagnosis
6. Results and,
7. Conclusion

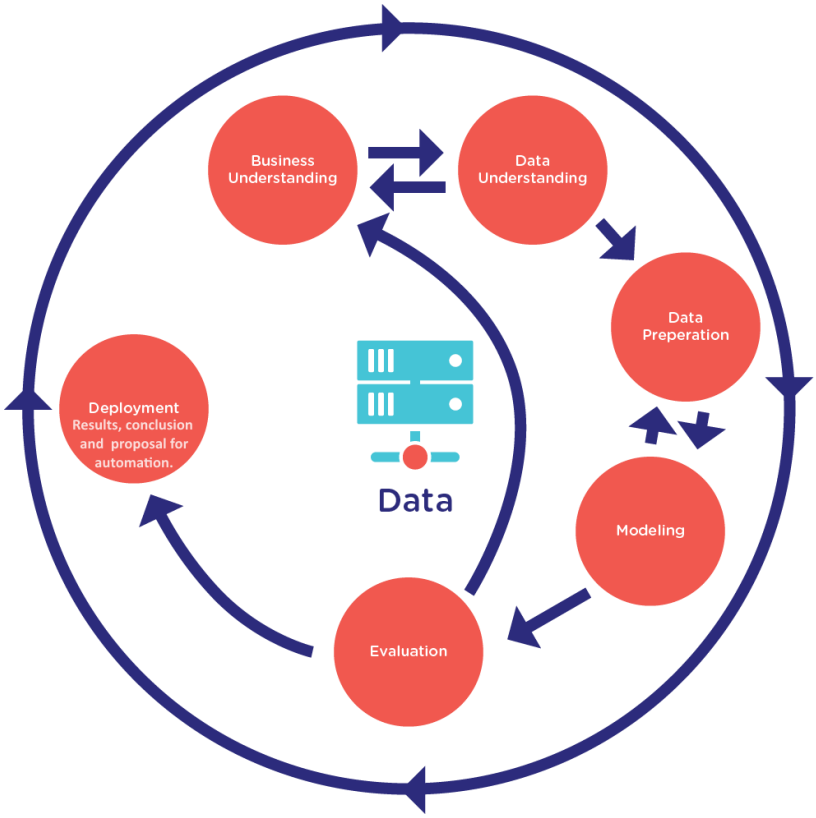

Figure 1: Data Mining Projects development/process cycle **[4]**

More technically, any data mining task can be categorised into six different classes:

1. **Anomaly Detection (Outliers/Change/Deviation Detection)** – It is the process of identifying unusual data records, that might be interesting or data errors that require further investigation. An unusual data point can either be an outlier, a case of high leverage or a case of high influence.
    i. **Outlier** - By definition an outlier is any point that falls outside 1.5 times the interquartile range above the third or below the first quartile. Outliers usually are the data points that do not fit the model under study. It can also be an error in the measurement. An outlier can also occur when comparing relationships between two sets of data. Figure 1 shows an example of an outlier [5].

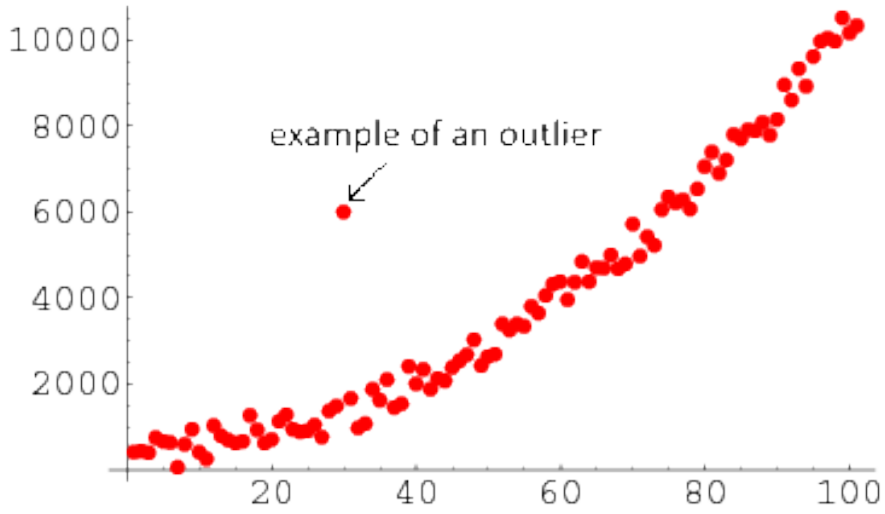

Figure 2: Example of an outlier on a scatter plot **[5]**

    ii. **Cases of high leverage** – A data point is said to be a case of high leverage if the corresponding x-value is far from the mean value of x i.e. if the x-value is unusually large or unusually small. Figure 2 shows a case of high leverage. By convention any h_{ii} value above $2p'/n$ is said to be a case of high leverage or,

$$h_{ii} > \frac{2p'}{n}$$

Where, h_{ii} is the hat matrix, p' is the number of predictor and the intercept i.e. $p + 1$and n is the sample size [6].

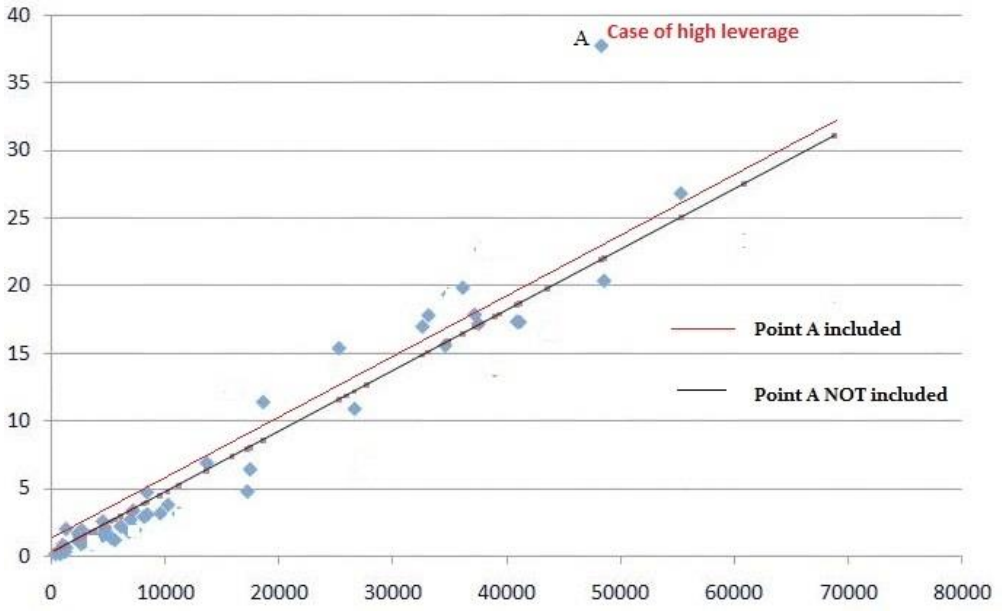

Figure 3: Shows the change in fitted model if point A (a case of high leverage) is included and when not included **[7]**

iii. **Cases of high influence** – omitting a case of high influence from the dataset results into a large change in the values of the estimates of the fitted model. Figure 3 shows a case of high influence [6].

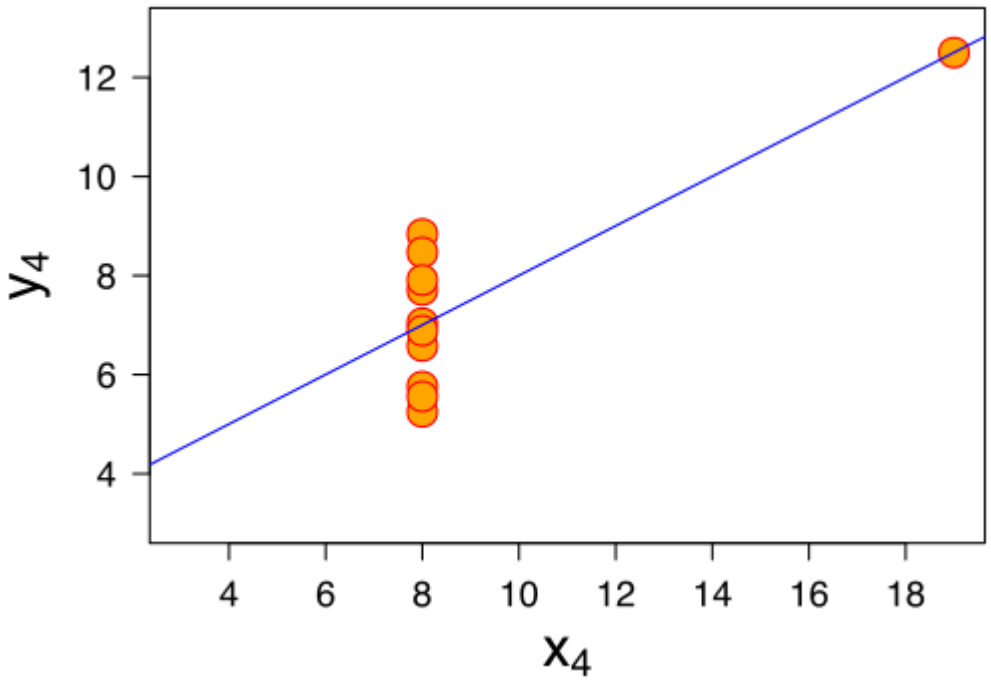

Figure 4: The farthest point in the graph above shows a case of high influence

2. **Association rule learning** – Searches for relationships between variables. Usually a correlation matrix between the different variables give a good insight into understanding such relationships between them. In statistics these correlation matrices show the correlation between various predictor. However, the correlation matrix only gives positive or negative association information between predictor. In general, two highly correlated variables give the same information in statistics as a single variable will do and so one of them is usually dropped from the predictor set. Graphically, the dependence of each of the different predictor can be modelled before selection and usually helps in picking up the right variables [8, 9]. In Figure 4 we show how different variables used the given project can be linked together and grouped according to their properties. We can notice that the predictor user id, name and screen name are bound to be highly correlated because they give the same information i.e. users primary information. Similarly, place id and subdivision give the same information etc.

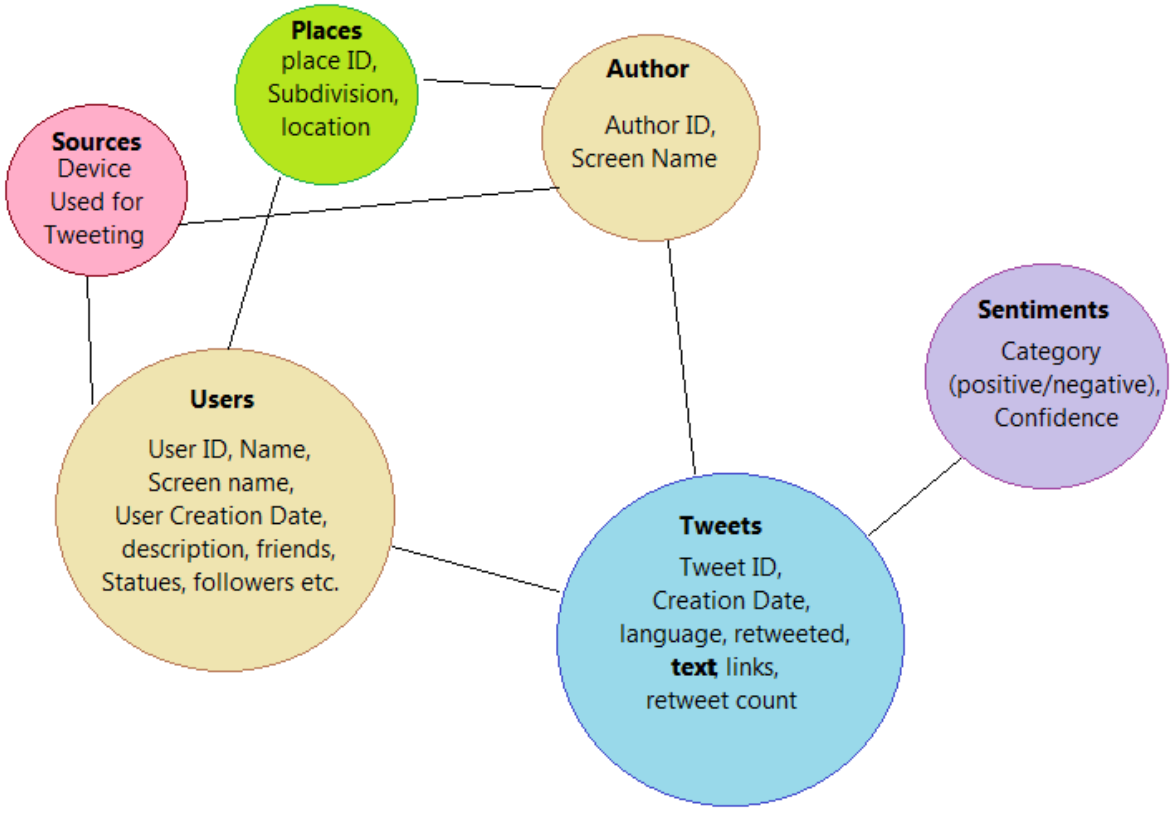

Figure 5: Dependency graph for various predictor in the given project

3. **Clustering** – It is a task of discovering groups and structures in the data that are similar in some way or another, without using known structures in the data. It has no mechanism to differentiate between relevant and irrelevant variables. Therefore, the choice of variables included in a cluster analysis must be underpinned by conceptual considerations [10]. This is very important because the clusters formed can be very dependent on the variables included. Figure 5 shows a dendrogram that separates the data using cluster analysis:

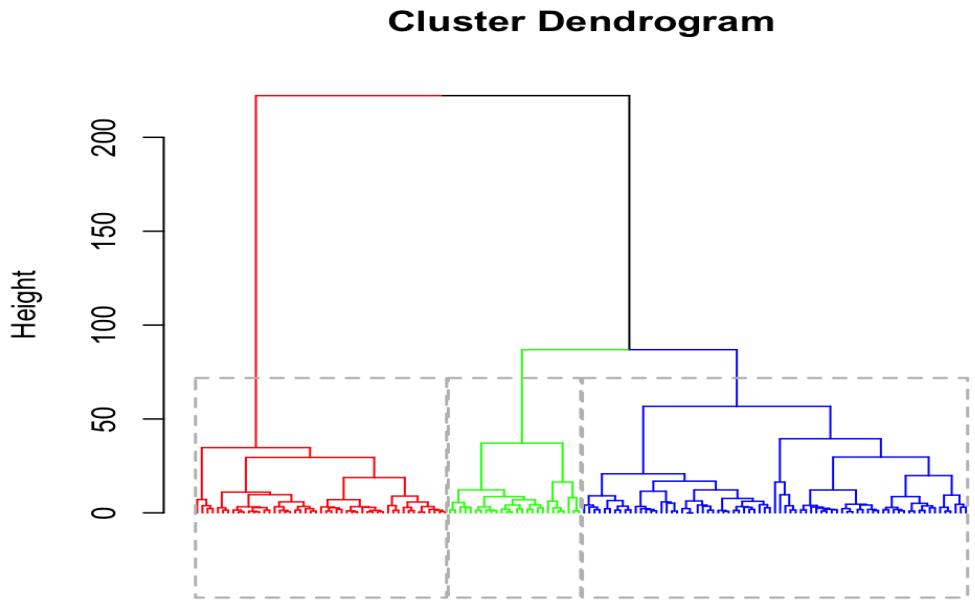

Figure 6: An example of cluster analysis on a given predictor set **[11]**

4. **Classification** – It is the task of generalising known structures to new data. Most of the data mining domain's fame comes from the expertise people have gained in handling this task. There are many classification techniques or, classifiers that one might use to predict a qualitative response. The most commonly used classification techniques are logistic regression, linear discriminant analysis, K-nearest neighbour, Naïve Bayes etc. Some of the more computer intensive methods include generalised additive model, trees, random forest, boosting and support vector machines. Classification problems occur more often than any regression problem. Some of the examples are as follows [12, 13]:
    i. A person arrives at the emergency room with a set of symptoms that could possibly be attributed to one of the three possible medical conditions. Which of the three conditions does the individual have.

ii. Given the users past records and IP address etc. a bank may need to identify whether a particular transaction is fraudulent or not.
iii. On the basis of DNA sequence and the knowledge of patient having a disease or not, a biologist might want to find out whether a particular DNA mutation is deleterious or not.

In the present case, we are given with the user's information and what they are tweeting about. Knowing this information, we need to find out whether the person is going to vote for or against a particular governmental party.

5. **Regression** – It attempts to find a function which models the data with the least error estimates. In particular, regression is a useful tool for predicting any quantitative response. The best example for a regression model can be "the marketing plan for any company". Given the sales for a particular product and its Advertising budget for TV, radio and newspaper media, we need to figure out the best marketing plan for the company. In order to achieve the best model, we need to answer few important questions, some of which are stated as follows:
    i. Is there any relationship between the advertising budget and the sales of the product?
    ii. How strong is the relationship between the advertising budget and the sales of the given product?
    iii. Which of the given Media contribute towards the sales?
    iv. How accurately can we estimate the effect of each medium on the sales?
    v. How accurately can we predict the future sales?
    vi. Is there a relationship linear and do they need transformation?
    vii. Is there synergy among the advertising media? How does the combined effect of each contribute to the sales of the product?

    Finding the surrogate predictor is also important. For example, consider the famous example of shark attacks and ice cream sales. It is counter intuitive to say that during the summers there is an increase in the shark attacks with the increase in the ice cream sales. Both the variables show that they are significant in predicting the sales of ice cream but only when the effect of temperature is introduced in the model that the shark attacks variable become insignificant. In the current project it will be important to check the effects of each predictor, its relationships with other predictor, the synergy effect among predictor, and the surrogate variables [12].

6. **Summarization** – We always need to imbue a statistical result with meaning. It is therefore, important to visualize, interpret and point out the weakness of the proposed model in order to understand the results meaningfully. The numbers do not speak for themselves and it is our responsibility to construe them with sensible interpretation by linking the observed inference with the knowledge of the real world.

b. Sentiment Analysis

Sentiment analysis is a part of natural language processing which is used to identify and extract subjective information from the source material. It can be regarded as a classification technique, either binary (polarity classification into positive/negative) or multi-class categorization (e.g. positive/neutral/negative). These sentiments can be considered as one of the best methods to analyse public's inclination towards a particular campaign. Knowing the response, we can measure how much a person is likely to vote if a negative sentence is expressed and how much is he likely to vote if a positive word is spoken.

Natural Language processing is a field of computer science that deals with lexical analysis and interpretation of the human language. It models the methods through which natural language can be interpreted by the computers in a meaningful manner. NLTK or the Natural Language Processing Tool Kit is a package created in Python which serves the purpose of these computational linguistics. It is an amazing library to play with natural language and is extensively used in this project for lexical analysis [14].

Following are some of the features that help in processing the text data and summarize them:

1. **Word tokenizing** – a process of breaking down a stream of text into words, phrases and symbols. There are mainly two kinds of tokenizers available in NLTK:
    a. **Sent_tokenize –** It is used to split a document into set of sentences. It is useful at places where we need to tag each word in a sentence with parts of speech tags.

    b. **Word_tokenize –** It is used to split a sentence into tokens of words.

    One more important derived class available in NLTK is the **TweetTokenizer**. It is really helpful in creating tokens from the tweets on the twitter. It is noteworthy to understand that Standard English language does not contain smiley faces in

order understand the emotion of the user. Tweet Tokenizer is a specialised tokenizer that can detect these tokens as well, making the analysis much easier when it comes to analysing text coming from twitter or any other social networking sites. There are many other tokenizers that uses machine learning algorithms in order to tokenize the words coming from different sources. Such algorithms are useful when there is a mixed source of data and, where language of the text is not a barrier [15].

2. **Parts of speech tagging** – a process through which some text is read and each word is tagged with its parts of speech (noun, verb, adjective etc.). A tag set corpus uses many different conventions for tagging words. A universal Parts of speech set is mentioned in the table 1 below. These are the most frequent and common to all languages tag set used while tagging words.

Table 1: Sample set of annotations used in Parts of speech tagging **[14]**

Tag	Meaning	English Examples
ADJ	adjective	new, good, high, special, big, local
ADP	adposition	on, of, at, with, by, into, under
ADV	adverb	really, already, still, early, now
CONJ	conjunction	and, or, but, if, while, although
DET	determiner, article	the, a, some, most, every, no, which
NOUN	noun	year, home, costs, time, Africa
NUM	numeral	twenty-four, fourth, 1991, 14:24
PRT	particle	at, on, out, over per, that, up, with
PRON	pronoun	he, their, her, its, my, I, us
VERB	Verb	is, say, told, given, playing, would
.	punctuation marks	. , ; !
X	other	ersatz, esprit, dunno, gr8, univeristy

3. **Stop words removal** – are the most common occurring words in the language. They are usually removed if the sentence to be parsed is not required to be evaluated for phrases and parts of speech detection. Since they do not contribute towards understanding the sentiment of the text using statistical methods, they are called stop

words and hence removed before any analysis is done. Below is the set of stop words found in Standard English language:

Table 2: Set of stop words found in Standard English corpora as mentioned in NLTK package **[15]**

ourselves, hers, between, yourself, but, again, there, about, once, during, out, very, having, with, they, own, an, be, some, for, do, its, yours, such, into, of, most, itself, other, off, is, s, am, or, who, as, from, him, each, the, themselves, until, below, are, we, these, your, his, though, don, nor, me, were, her, more, himself, this, down, should, our, their, while, above, both, up, to, ours, had, she, all, no, when, at, any, before, them, same, and, been, have, in, will, on, does, yourselves, then, that, because, what, over, why, so, can, did, not, now, under, he, you, herself, has, just, where, too, only, myself, which, those, I, after, few, whom, t, being, if, theirs, my, against, a, by, doing, it, how, further, was, here, than

4. **Stemming** – Stemmers are used to remove the morphological affixes from the words or simply, it is the process of converting a word into its root or the base form. Two main algorithms that are used for stemming are Porter stemmer algorithm and Snowball stemmer algorithm. The main difference between these two stemming algorithm is that the first converts the word into its root form without preserving the word meaningfully where the snowball stemmer algorithm preserves the word. For example, if we stem the word 'generous' using snowball stemmer algorithm it will give us 'generous' while the porter stemmer algorithm will give 'gener'. Snowball stemmer algorithm is more generalised compared to porter stemmer algorithm [15].
5. **Chunking** – It is the method of identifying the entities and relationships in any given text. It helps in grouping the data into meaningful sets such as nouns, adverbs, names, etc. It makes the analysis easier when there is a large document and we need to analyse it subparts or structure. NLTK provides with tools to visualise these chunks into meaningful tree structures, thus help is determining key points in the document easily [15].

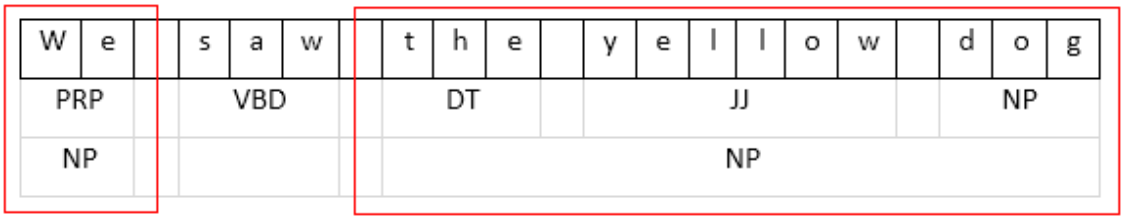

Figure 7: Segmentation and Labeling at both the Token and Chunk Levels **[15]**

The smaller boxes show the word-level tokenization and parts of speech tagging, while the large boxes show higher level of chunking. Each of the larger box in Figure 7 is called a chunk. Like tokenization, which omits whitespaces, chinking usually selects a subset of the tokens.

6. **Frequency Distribution** – It is one of the most useful features that enables us to identify the most common occurring words in a given text and hence gives us a sense of context on which the text is written. However, in a large document it might also be important to determine the location of a word in the given text. These positional information is usually obtained using the dispersion plot as mentioned in Figure 8.

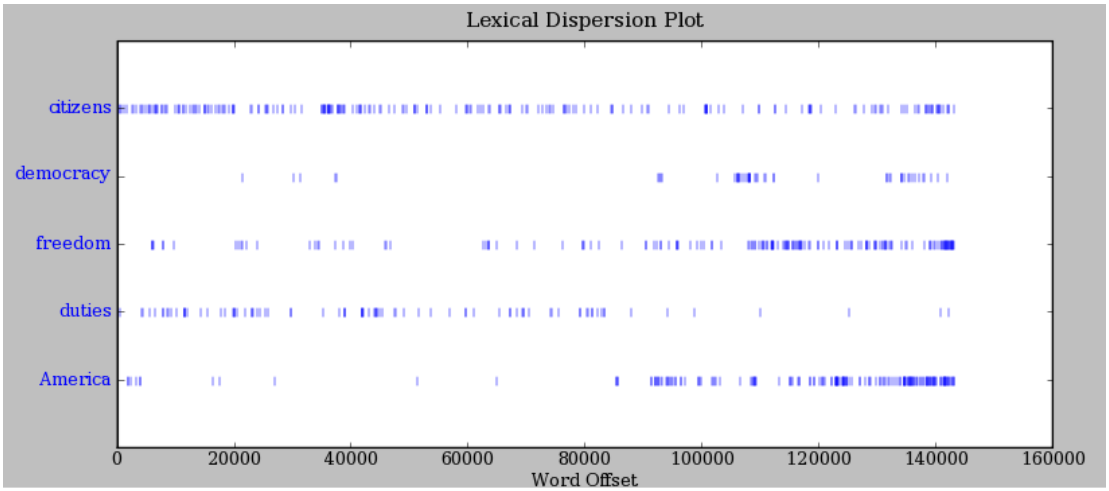

Figure 8: Lexical Dispersion Plot for Words in U.S. Presidential Inaugural Addresses: This can be used to investigate changes in language use over time **[15]**

7. **Visualizing text** – Tree structures are usually very useful in identifying the various parts of speech in a given text and thus help in identifying, the correlation between words in a given document. A simple chunked sentence can also be visualized using chunk gram parsers. Figure 9 shows a chunked sentence using regular expression "NP: {<DT>?<JJ>*<NN>}" [15].

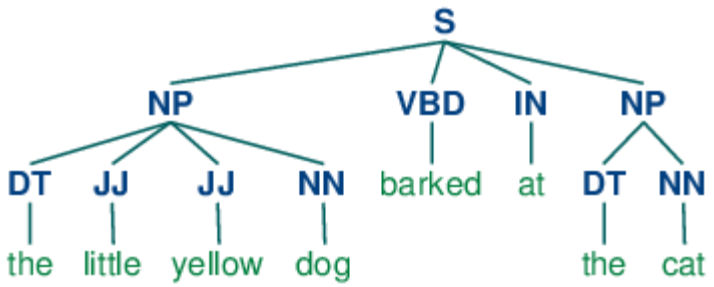

Figure 9: Example of a Simple Regular Expression Based NP Chunker **[15]**.

c. Machine Learning Algorithms

Machine learning is a subfield of computer science that has evolved from the study of pattern recognition, computational intelligence and, computational learning theory in artificial intelligence. It refers to a vast set of tools for understanding data. These tools can be classified as supervised or unsupervised techniques. Supervised statistical learning involves one or more predictor and a guiding response variable. In supervised learning we estimate the output when a population of data is known beforehand. Unsupervised learning techniques on the other hand does not have any pre-known response set and is used to find relationship between the set of data and group them [16].

The project concentrates on supervised learning techniques for both text classification and analysis of the twitter as a viable source for predicting voting patterns. Further supervised learning can be broken down into two sub parts:

1. Regression – It is a technique of modelling relationship between scalars or a discrete binary response variable Y and a set of predictor X also called the explanatory variables. Usually these techniques are helpful when a geometric pattern can be observed in the scatter plot of a data set w.r.t response.
2. Classification – It is the problem of understanding the classes to which an observed data belongs to. This classification is done on the basis of the training data set containing observations whose category membership is already known. Example, given a few examples of an email being a spam or not we identify whether a new email will be a spam or not. A classification problem might have two response labels as mentioned in the above example or might have multiple labels e.g. sentiment analysis with three emotions “Happy/Neutral/Sad”.

The project involves different classification and regression techniques to classifying a sentiment into positive or negative. These sentiments are then used to measure how each sentiment contribute towards a +1 vote.

i. Logistic Regression

It models the probability that Y belongs to a particular category given a certain condition. Rather than directly modelling the variable Y, logistic regression models the probability of that Y belongs to a particular category. For example, in voting contribution of sentiment can be calculated as [12]:

$$Pr(vote = Yes|sentiment = +ve)$$

The above eq. is abbreviated as $p(sentiment = +ve)$ whose value ranges from 0 to 1. It is modelled as:

$$p(x) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}} \qquad (eq.1)$$

To fit the model, we use maximum likelihood ratio instead of least square method because of better statistical properties. The likelihood function is given by:

$$l(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_i'=0} (1 - p(x_i))$$

The logistic regression model follows a sigmoidal function and is represented by the following graph:

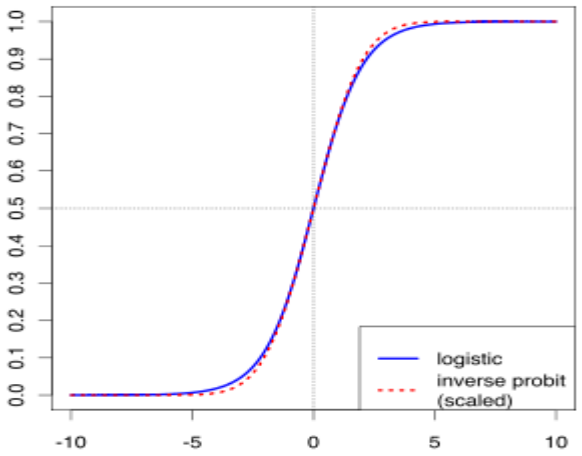

Figure 10: Comparison between Logit and Inverse Probit Function **[17]**

The above graph states the how a logit function changes in comparison to a similar inverse probit function, which is a similar kind of prediction model. Various other regression models do exist but logit and probit are the most common.

A little bit of manipulation of eq. 1 we can find out that:

$$\frac{p(x)}{1-p(x)} = e^{\beta_0+\beta_1 X} \qquad (eq.2)$$

The quantity $p(x) - 1/p(x)$ is called the odds, and can hold any value between $0\ and\ \infty$. Values of the odds close to $0\ and\ \infty$ show a very high probabilities of default, respectively. These odds are traditionally used as it denotes a more natural way of relating win and loss. Taking a natural log on both sides of the eq.2 yields the following equation:

$$\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 X \qquad (eq.3)$$

The equation 3 is synonymous with the linear regression. The left hand side of the equation is called the log-odds or logit. So, now we can say that β_1 gives the change in the log odds with one-unit increase in X. It is noteworthy that the relationship between $p(x)\ and\ X$

is not in a straight line, hence one-unit increase in X does not correspond to the change in $p(x)$.

Generalising the above equations to multiple logistic regression, we get:

$$\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p \qquad (eq.3)$$

Where, $X = (X_1 \ldots X_n)$ are p predictor.

Similarly, the probability can be expressed as follows:

$$p(x) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}} \qquad (eq.4)$$

The estimates of $\beta_0, \beta_1, \ldots\ \beta_p$ is done using the maximum likelihood method.

ii. Naive Bayes

A Naïve Bayes classifier is a simple probabilistic model which relies on the assumption of the feature independence in order to classify input data. It is one of the most popular algorithms and is known for its simplistic implementation, low computational cost and relatively high accuracy. It is however known to over fit data and usually requires close examination of the results obtained [18].

Naïve Bayes is a conditional probability model and assigns an instance probability $p(C_k|x_1, x_3, \ldots . x_n)$ for each of the K possible outcomes or classes C_k. Therefore, applying the Bayes theorem, conditional probability, $p(C_k|x) = \frac{p(C_k).p(x|C_k)}{p(x)}$, we obtain the conditional distribution over the class variable C under the complete independence assumption as follows:

$$p(C_k|x_1, \ldots x_n) = \frac{1}{Z} p(C_k) \prod_{i=1}^{n} p(x_i|C_k)$$

Where the evidence $Z = p(x)$ is a scaling factor dependent only on $x_1, \ldots . x_n$, that is a constant if the values of the feature variables are known.

The two different types of Naïve Bayes used in the project are [18]:

- **Multinomial Naïve Bayes –** It is a Naïve Bayes algorithm for multinomial distributed data, and is one of the two classic naïve Bayes variant used in text classification. The distribution is parameterized by vectors $\theta_y = (\theta_{y1}, \dots \theta_{yn})$ for each class y, where n is the number of features (in text classification, the size of the vocabulary) and θ_{yi} is the probability $P(x_i|y)$ of feature i appearing in a sample belonging to class Y. The parameters θ_y is estimated by a smoothed version of the maximum likelihood, i.e. relative frequency counting.

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

Where $N_{yi} = \sum_{x \epsilon T} x_i$ is the number of times feature i appears in a sample of class y in the training set T, and $N_y = \sum_{i=1}^{|T|} N_{yi}$ is the total count of all features for class Y. The smoothing priors' $\alpha \geq 0$ accounts for features not present in the learning samples and prevents zero probabilities in the further computations. Settings $\alpha = 1$ is called Laplace smoothing, while $\alpha < 1$ is called Lidstone smoothing [18].

- **Bernoulli Naïve Bayes –** is the Naïve Bayes training and classification algorithm for data that is distributed according to multivariate Bernoulli distributions, i.e. there may be multiple features but each one is assumed to be a binary valued (Bernoulli, Boolean) variable. Therefore, this class requires samples to be represented as a binary valued feature vectors. The decision rule for Bernoulli Naïve Bayes is based on:

$$P(x_i|y) = P(i|y)x_i + \big(1 - P(i|y)\big)(1 - x_i)$$

Which differs from multinomial Naïve Bayes rule in that it explicitly penalizes the non-occurrence of a feature i that is an indicator for class Y, where the multinomial variant would simply ignore a non-occurring feature [18].

iii. Stochastic gradient Descent Classifier

It is a simple yet very efficient approach to discriminative learning of linear classifiers under convex loss functions such as (linear) Support Vector Machines and Logistic Regression. Even though SGD has been around in the machine learning community for long time, it has received a considerable amount of attention just recently in the context of large-scale learning. SGD has been successfully applied to large scale and sparse machine learning problems often encountered in the text classification and natural language processing. Given the data is sparse, the classifiers in this method easily scales to the problem with more than 10^5 training examples and more than 10^5 features. The advantage of the stochastic gradient descent are efficiency and ease of implementation. The disadvantage is that it requires quite a number of hyper parameters such as regularization parameter and the number of iterations. It is also sensitive to feature scaling [19].

In comparison to the simple gradient descent function, stochastic gradient descent function can overcome the computing cost. Even if the dataset is very large, it can still lead to a faster convergence.

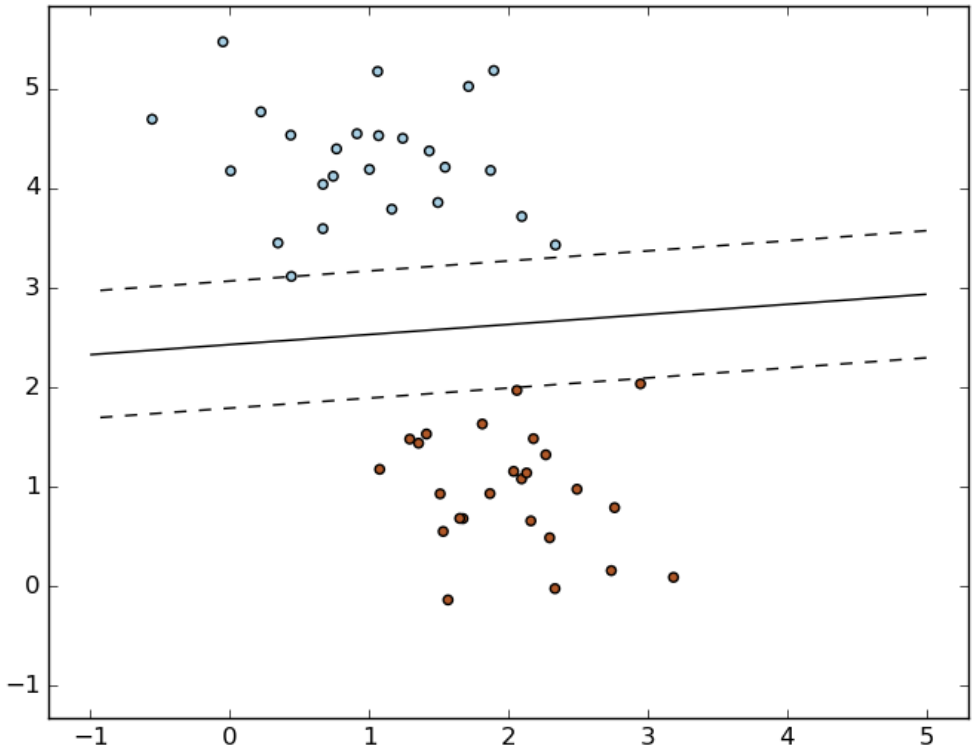

Figure 11: Stochastic Gradient Descent **[19]**

The standard gradient descent algorithm updates the parameters θ of the objective function $J(\theta)$ as,

$$\theta = \theta - \alpha \nabla_\theta E[J(\theta)]$$

Where the expectation in the above equation is approximated by evaluating the cost and the gradient over the full training set. Stochastic Gradient Descent (SGD) simply does away with the expectation in the update and computes the gradient parameters using only a single or a few training examples. The new update is given by:

$$\theta = \theta - \alpha \nabla_\theta J(\theta; x^{(i)}, y^{(i)})$$

With a pair $(x^{(i)}, y^{(i)})$ from the training set.

Generally, each parameter update in SGD is computed w.r.t. a few training examples or a mini batch as opposed to a single example. This reduces the variance in the parameter update and can lead to more stable convergence. It also allows the computation to take advantage of highly optimized matrix operations that should be used in a well vectorised computation of the cost and the gradient. A typical mini batch size of 256, although the optimal size of mini batch can vary for different applications and architectures. In SGD the learning rate α is typically much smaller than a corresponding learning rate in the batch gradient descent beacause there is much more variance in the update. Choosing the proper learning rate and schedule can be fairly difficult. One standard method that works well in practice is to use a small enough constant learning rate that gives stable convergence in the initial epoch (full pass through the training set) or two of training and then halve the value of the learning rate as convergence slows down. An even better approach is to evaluate a held out set after each epoch and anneal the learning rate when the change in objective between epochs is below a small threshold. This tends to give good convergence to local optima. Another commonly used schedule is to anneal the learning rate at each iteration t as $ab + t$ where $a \; and \; b$ dictate the initial learning rate and when the annealing begins respectively. More sophisticated methods include using a backtracking line search to find the optimal update. One final but important point regarding SGD is the order in which we present the data to the algorithm. If the data is given in some meaningful order, this can bias the gradient

and lead to poor convergence. Generally, a good method to avoid this is to randomly shuffle the data prior to each epoch of training. If the objective has the form of a long shallow ravine leading to the optimum and steep walls on the sides, standard SGD will tend to oscillate across the narrow ravine since the negative gradient will point down one of the steep sides rather than along the ravine towards the optimum. The objectives of deep architectures have this form near local optima and thus standard SGD can lead to very slow convergence particularly after the initial steep gains. Momentum is one method for pushing the objective more quickly along the shallow ravine. The momentum update is given by,

$$\upsilon = \gamma\upsilon + \alpha\nabla_\theta J(\theta; x^{(i)}, y^{(i)})$$

$$\theta = \theta - \upsilon$$

In the above equation υ is the current velocity vector which is of the same dimension as the parameter vector θ. The learning rate α is as described above, although when using momentum, α may need to be smaller since the magnitude of the gradient will be larger. Finally, $\gamma\epsilon(0,1]$ determines for how many iterations the previous gradients are incorporated into the current update. Generally, γ is set to 0.5 until the initial learning stabilizes and then is increased to 0.9 or higher [20].

iv. Support Vector Machines

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. It is based on the concept of decision planes that define the decision boundaries. A decision plane is one that separates between a set of objects having different class memberships. Instead of constructing a complex curve to separate two objects, the original objects are mapped or rearranged using a set of mathematical functions, known as kernels and transformed such that they are linearly separable [21].

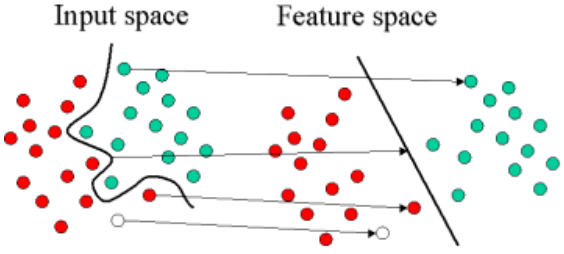

Figure 12: Support vector machine basic transformation **[21, 22]**

Following graph shows a simple 2D linearly separable points. The graph is an over simplification of the problem having lines and points in the Cartesian plane instead of hyperplanes and vectors in higher dimensions. Here, we are needed to find the optimal line that separates the two classes.

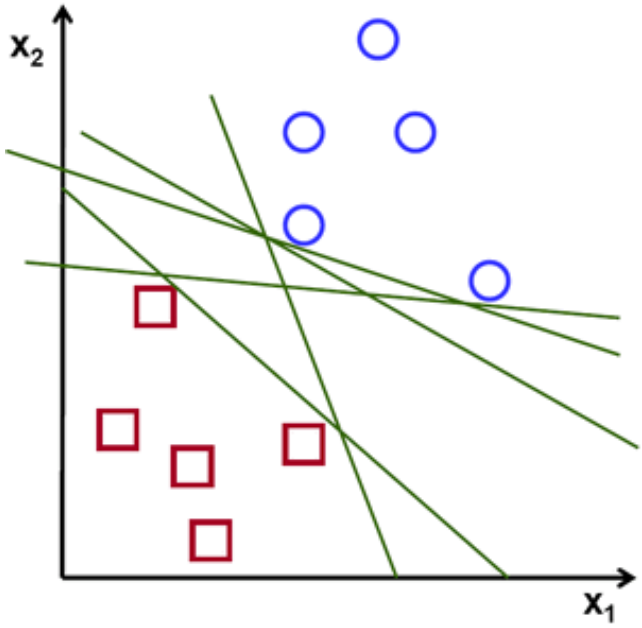

Figure 13: Graph showing two separable classes of objects that need to be classified using SVM **[23]**

A line will be poor if it passes too close to the points because it will be noise sensitive and will not generalize correctly. Therefore, our goal is to find the line passing as far as possible from all points. The optimal separating hyperplane (in this case a line) should be the largest minimum distance to the training examples. Similar to the example below:

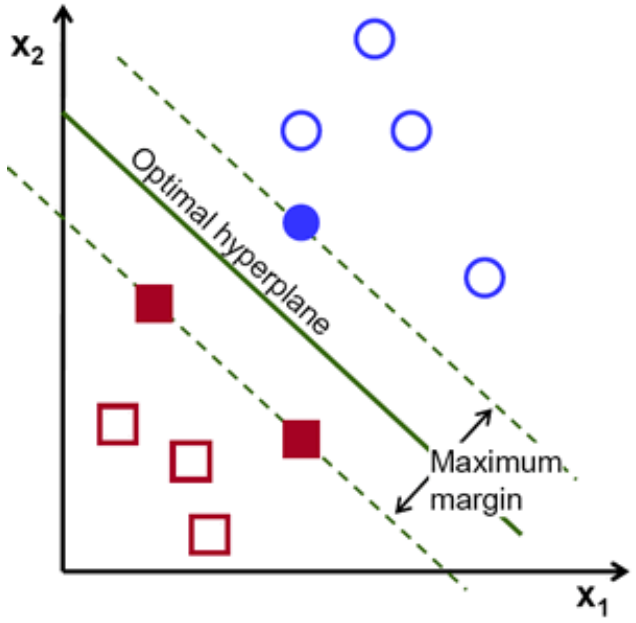

Figure 14: Optimal hyperplane separating two different classes in a Cartesian plane **[23]**.

A hyperplane is formally defined as:

$$f(x) = \beta_0 + \beta_x^T$$

Where β is known as the weight vector and β_0 as the bias.

The optimal hyperplane can be represented in an infinite number of different ways by scaling of β and β_0. As a matter of convention, among all the possible representations of the hyperplane, the one chosen is:

$$|\beta_0 + \beta_x^T| = 1$$

Where x symbolises the training examples closest to the hyperplane. In general, the training examples that are closest to the hyperplane are called support vectors. This representation is known as the canonical hyperplane. So, the distance between the point x and a hyperplane (β, β_0) is given by:

$$distance = \frac{|\beta_0 + \beta_x^T|}{||\beta||}$$

In particular, for the canonical hyperplane, the numerator is equal to one and the distance to the support vectors is given by:

$$distance_{support\ vectors} = \frac{|\beta_0 + \beta_x^T|}{||\beta||} = \frac{1}{||\beta||}$$

So, the maximum margin as shown in Figure 14 is twice the distance to the closest examples. Therefore, we can say that:

$$\mathcal{M} = \frac{2}{||\beta||}$$

Since, $\mathcal{M}$ is inversely proportional to β, therefore maximizing $\mathcal{M}$ is equal to minimizing function $L(\beta)$ subject to some constrains. The constraints model the requirement for the hyperplane to classify correctly all the training examples x_i . Formally,

$$\min_{\beta,\beta_0} L(\beta) = \frac{1}{2}||\beta||^2 \qquad subject\ to \quad y_i\left(\beta_{x_i}^T + \beta_0\right) \geq 1 \quad \forall\, i$$

Where y_i, represents each of the labels of the training examples [23].

So, we can say that SVM is a non-probabilistic binary linear classifier. The basic idea of separating two objects is two introduce a hyperplane. If the objects are not immediately linearly separable in a multi-dimensional space, it introduces a new dimension and continues until they are linearly separable in a hyperspace.

The advantages of support vector machines are:

- Effective in high dimensional spaces
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the detection function (called support vectors), so it is also memory efficient.

The disadvantages of support vector machines are:

- If the number of features is much greater than the number of samples, the method is likely to give poor performance.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross validation.

NuSVC and LinearSVC are part of the sklearn package in Python. They accept slightly different parameters. For LinearSVC the kernel is linear, and has less number of options than NuSVC [24].

3. Implementation

Scraping data online has become easier with the increasing processing power and interconnectivity and with the advent of social networking sites, internet has exploded with information. It also has given every individual the freedom of speech. People tend to talk about agendas, campaigns and share their experience using these platforms.

Knowing the word of mouth on the street is not simple and asking a sample of the population about a campaign is expensive and time consuming. Social Networking platforms comes handing in such a case. Twitter in that sense provides one of the best and easy to use API to scrape data. The word limit of 140 characters makes it easier to analyse information more predictably and efficiently.

a. Platform and Software

The application requires a minimum of 2GB RAM, and a 32bit operating system. The system with lower end 32-bit system shows issues in Windows operating system because some of the packages are not pre-complied for 32-bit version of them. It is therefore suggested to use a 64bit system if on Windows, otherwise system with Linux/MacOS installed shall work perfectly fine. For statistical analysis and to monitor election campaigns, we use the following tools:

i. Python

There are many statistical analysis and visualization tools available online which include SPSS, SAS, R, RapidMiner, Statistica etc. but Python is one of the most versatile tools of all as it can analyse, visualise, automate and even create useful applications. It is a light weight open source tool. One of the biggest advantage of the Python language is its large community and a pseudo code like scripting language which creates a low learning curve for both beginners and statisticians.

ii. Packages

Various packages are available in the Python language which aids in statistical modelling and analysis. They include – numpy, pandas, scipy, sklearn, nltk, matplotlib etc. Apart from these packages Twitter API tweepy and pymongo help really a lot in data scraping and downloading JSON formatted data and saving in the Mongo database.

Below is the description of each of them:

a. **Numpy** – is a fundamental package for scientific computing with Python. It has the capabilities to create N-dimensional array objects, matrices, useful linear algebra, Fourier transformation and random data generation capabilities. ndim, shape, size, dtype, itemsize, data etc. are some of the basic functions inbuilt [25].

b. **Pandas** – It adds into Python, R like capabilities. The basic building blocks of pandas data structure are Series and DataFrame. It provides some of the fantastic tools to read and write data using its IO Tools [26].

c. **Scipy** – contains additional routines needed in the scientific work. For example, routines for computing integrals numerically, solving differential equations, optimization and sparse matrices [27].

d. **Sklearn** – scikit-learn or sklearn for short is a tool for data mining and analysis. It mainly contains the learning algorithms that can be used for classification, clustering, regression, dimensionality reduction, model selection and pre-processing. It is built on top of the packages like Numpy, Scipy and Matplotlib. Statsmodel is another package which offers R like formatting for analysis using the sklearn package and is also used in building the model extensively [28].

e. **NLTK** – It is a package for text processing. The library contains functions such as tokenization, stemming, tagging, parsing, and semantic reasoning. One of the cornerstone of the package is that it contains specialised functions for Twitter text processing including tokenizing and parsing features [15].

f. **Matplotlib** – is a plotting library for Python used to plot publication quality figures and is useful tool while generating graphs and charts in between the analysis. Histograms and frequency distribution charts especially help in identifying the best approach to tackle the problem at hand [29].

g. **Tweepy** – It provides a simple to use interface for twitter data scraping and facilitating exception handling and url encoding using core urllib available in Python language [30].

h. **pymongo** – makes it really easy to connect the Python interface with mongoDB database server. One of the key features of mongoDB is that it allows us to store data in JSON format. The set of tables is called collections. Each collection has multiple documents similar to row entries in a database table [31].

iii. MongoDB

It is an open-source document database that provides high performance, high availability, and automatic scaling. It obviates the need for an Object Relational Mapper (ORM) to facilitate development. ORM basically is a technique in computer science that facilitates converting data between two incompatible type systems namely relational database and programming languages to enable programming easier. In mongoDB this problem is removed by keeping JSON formatted data as the mode of data exchange, so there is no immediate need for any ORMs. It is also scalable and can be used in distributed systems making it a lot easier to handle load when application usage grows [31].

iv. Spreadsheets

These are convenient tools for simple visual inspection for parts of the data. Pandas package makes the life easier by creating an IO tool to import and export spreadsheets using simple interface. It uses in the background openpyxl package to handle Excel sheets [26].

v. Web Interface

While automating the whole process it also seemed a good idea to create a simple web interface for interacting with the application. Once the statistical inferences and predictor selection is in place we can use an interface to trigger and generate reports from a centralised location. As a proof of concept, the reports might serve as a tool to regenerate and check the quality of “Twitter as channel for campaign monitoring”. For creating the interface, we use CSS tool ‘bootstrap’, bi-directional event handling JavaScript Framework, ‘AngularJS’ and a micro-server ‘bottlepy’ to trigger data exchange between the core software and the user interface.

a. **Bootstrap -** It is a free and open-source front end web frame work for designing websites and web applications. It contains HTML and CSS based design templates for typography, forms, buttons, navigation, and other interface components, as well as optional JavaScript extensions. Unlike many web frameworks, it concerns itself with front-end development only. It is also one of the most popular projects on GitHub [32].

b. **AngularJS -** It is a complete JavaScript based open-source front-end web application framework mainly maintained by google and supported by communities and corporations. It works by first reading the HTML page which has embedded into it additional custom tag attributes. Angular interprets those attributes as directives to bind input output parts of the page to a model that is represented in standard JavaScript variables [33].

c. **Bottlepy -** It is a WSGI micro web-framework for Python programming language. It is designed to be fast, simple and lightweight, and is distributed as a single file module with no dependencies other than the Python Standard Library [34].

b. Twitter Data Collection

Twitter provides us with many APIs to search and stream data. Stream APIs are useful when we need to track a particular keyword on the real-time basis and does not care about the place of origin of the text. Search API on the other hand is a very powerful tool that allows us to track keywords from different places using geo search and place ids. Also allows us to look through the user's timeline. We can track keywords and limit them to particular place of origin.

Twitter however comes with the limitation that we can only search or stream information that is not more than a week old. This constrains analysts who want to analyse an information that is from the past. However, it allows us to download user's timeline which can have data that is not constrained by the 1-week cap. This gives us the opportunity that if a keyword is still floating on twitter about a campaign, we can go back in time and track those users who are still talking about the topic and track what they said about the same campaign in the near past. Tracking each person's tweets and analysing it for positive or negative sentiment we can predict how likely is it for a person to vote for a campaign when he tweets a negative sentiment or when he tweets a positive sentiment.

Twitter also limits the rate of searches that we can do in a particular day. Usually, it is around 180 searches in 15 minutes. Also, to limit the use according to the users, it provides us with an OAUTH tokens to make sure that the users do not put immense load on its servers by making multiple connections. To get an OAUTH token we need to register on the developers' site of twitter and create an app and generate an app id, access token and consumer token in order to use the APIs [35].

Tweepy provides us with OAUTH and Twitter objects that can be used from within a Python script, initialised with the given access and consumer secret tokens and used to download data. In this project we created a Python script to download the data. While looking through a single response received from the twitter search API it was found that there are many parameters that seemed to be repetitive or irrelevant. So, in order to analyse such tweets the following parameters were found useful and the data was filtered accordingly.

Table 3: Variables grouped according to users, places, tweet, author and source

User	Tweet	Author	Place	Source	Sentiment
user_id	tweet_id	author_id	place_id	source	category
name	created_at	screen_name	subdivision		confidence
screen_name	lang		location		
user_created_at	retweeted				
description	text				
friends_count	links				
statuses_count	retweet_count				
followers_count					
favourites_count					
contributors_enabled					

As can be seen from the above table, a few of the downloaded data are redundant or unnecessary. For example, user_id, name, screen_name refer to the same individual so they are dropped and only the screen_name is taken into consideration while analysing the data. However the original information is retained if required. It should be noted that the sentiment analysed on our end is not retrieved from the server. This information is re-arranged in-terms of their characteristics such as the redundant information that's needed to be dropped, the variables that

have multi-factors or factor levels equal to two, time series or a numeric data, and the response variable. The table below show these re-arranged variables.

Table 4: Variables grouped according to their characteristics/usefulness in the model

Dropable	Factors		Numeric	Time series	Response
	Multiple	Two			
user_id name description place_id location tweet_id lang links author_id	Author screen_name source screen_name subdivision	contributors_ enabled retweeted	friends_count statuses_count followers_count favourites_count retweet_count confidence	created_at user_created_at	text

It can be seen from Table 4 that the variables 'user_id', 'name' are duplicates for 'screen_name'. Also, the variable 'description' does say anything about the voting pattern. The variable 'place_id' gives the same information as subdivision. The variable 'location' that is downloaded from twitter is a user input therefore 'subdivision' is better candidate variable to keep because it drives away the need for recoding duplicate entries. Variable 'tweet_id' is unique but has no significance in the prediction so should be dropped. The variable 'lang' contains only English because we are using only English corpus to train the system and hence it remains the same for all entries. Variable 'links' is used to back track tweets. However, it has no significance in predicting data. Variable 'author_id' is a repetition for 'author_screen_name' and hence dropped.

Also, we consider each spoken keyword (response 'text') for a campaign as a potential vote for it, so it makes the comparison between two competing campaigns easier using logistic regression analysis. Once this data is collected they are saved into mongoDB and retrieved from there for any further analysis. The scripts that were created in this phase can be found in the Appendix A.

c. Training the System with Corpus

NLTK contains many corpora used to model sentiments and analyse text documents. Data samples made available by NLTK packages in Python aids in creating supervised learning model. The project uses the twitter data samples provided by NLTK to model positive and negative sentiments. The corpus is downloaded using 'import nltk' and 'ntlk.download()' function. The downloaded corpus is usually saved in 'Users/~/AppData/Roaming/' in Windows and '/home/<user>/' in Linux [15]. In the 'twitter_samples' folder, there are two files. One of them is 'positive_tweets.json' and the other is 'negative_tweets.json'. Each line in these document is a JSON object containing 'text'. These objects are imported from the documents and used for training the classifiers.

The following folder structure shows how the nltk_data folder is organized. The project uses English 'stop words' from the corpora to remove unwanted words from each text. The feature set has 3000 words for prediction and a sentiment attached to it. It is similar to "choosing a set of words from the bag of words" and labelling it with a "sentiment".

```
|+-nltk_data
 |+-chunkers
 |+-corpora
  |+-twitter_samples
   |--negative_tweets.json
   |--positive_tweets.json
   |--...
  |+-stopwords
   |--english
   |--...
 |+-grammars
 |+-help
 |+-misc
 |+-models
 |+-stemmers
 |+-taggers
 |+-tokenizers
```

The following classifier algorithms were used for the training purpose:

a. Multinomial Naïve Bayes

b. Bernoulli Naïve Bayes

c. Logistic Regression

d. Stochastic gradient descent

e. Support Vector Machines (variants as mentioned in the background)

To streamline the process of parsing and creating a feature set a class 'Tweets' is created. This class is initialised with TweetTokenizer(), SnowballStemmer(), stopwords() in the English language. The number of features/predictor are then initialised to 5000 words. The imported samples are then pushed into a data structure along with their sentiment (pos/neg). The feature sets or words are then pickled for future referencing in the analysis part of the project. Then the first 3000 featuring words in the bag of words are taken to create a feature set [36]. This data is then shuffled and used for training the system. A leave-p-out cross validation approach is performed to check the accuracy of the model created. The classifiers are then trained using these feature sets and saved for future use.

The classifiers were trained twice in order to achieve better accuracy. In the first iteration because the training and test data were not normally distributed, and gave poor results as shown below:

Table 5: Accuracy of the trained classifiers, when data is not shuffled properly.

Learning Algorithm	Abbreviation	Accuracy
Multinomial Naïve Bayes	MultinomialNB	61.75%
Bernoulli Naïve Bayes	BernoulliNB	61.75%
Logistic Regression	LogisticRegression	61.2%
Stochastic Gradient Descent	SGDClassifier	60.15%
Support Vector Classifier (SVC)	SVC	49.45%
Nu-Support Vector Classification	NuSVC	54.95%
Linear Support Vector Classification	LinearSVC	61.05%

The data was then reshuffled to train all the models again. In this case, since they were normally distributed and gave better results.

Table 6: Accuracy of the trained classifier and, abbrebreations of each model.

Learning Algorithm	Abbreviation	Accuracy
Multinomial Naïve Bayes	MultinomialNB	93.10%
Bernoulli Naïve Bayes	BernoulliNB	94.75%
Logistic Regression	LogisticRegression	95.10%
Stochastic Gradient Descent	SGDClassifier	95.10%
Support Vector Classifier (SVC)	SVC	94.95%
Nu-Support Vector Classification	NuSVC	94.95%
Linear Support Vector Classification	LinearSVC	94.90%

The training set had a total of 8000 samples out of 10000 whereas testing had the remaining 2000 samples. The Stochastic gradient Classifier works well for sparse data unlike other classifiers. It might fail to predict with great accuracy the class of the response properly but definitely keeps a check that the model gives good results in the presence of sparse data.

d. Data Cleaning

The aim of the data cleaning process is to remove the unwanted content from the training data. The term unwanted refers to any piece of information within the tweets that will not be useful for statistical analysis. In our attempt to find the relationship between predictor of the dataset we are using logistic regression. It gives a detailed analysis on the influence of each predictor on the decision or the odds of winning between two competing brands. As mentioned in the “Twitter Data Collection” section there are some predictor which give the same information as others and hence are dropped from the model. For the remainder of the predictor frequency distribution plots are generated to check for the overall distribution pattern. In a large dataset, it is always a good idea to plot frequency distribution graphs in order to visually the predictor for any potential anomalies. So, before the data cleaned it is visualised as follows:

i. Tweets 'Text' Analysis

The following histogram show the frequency distribution of different text that has been retweeted by people over time. If the text occurs more than once in the dataset, then they are categorized as 'same tweet' while the tweets that are unique are categorized as independent tweets.

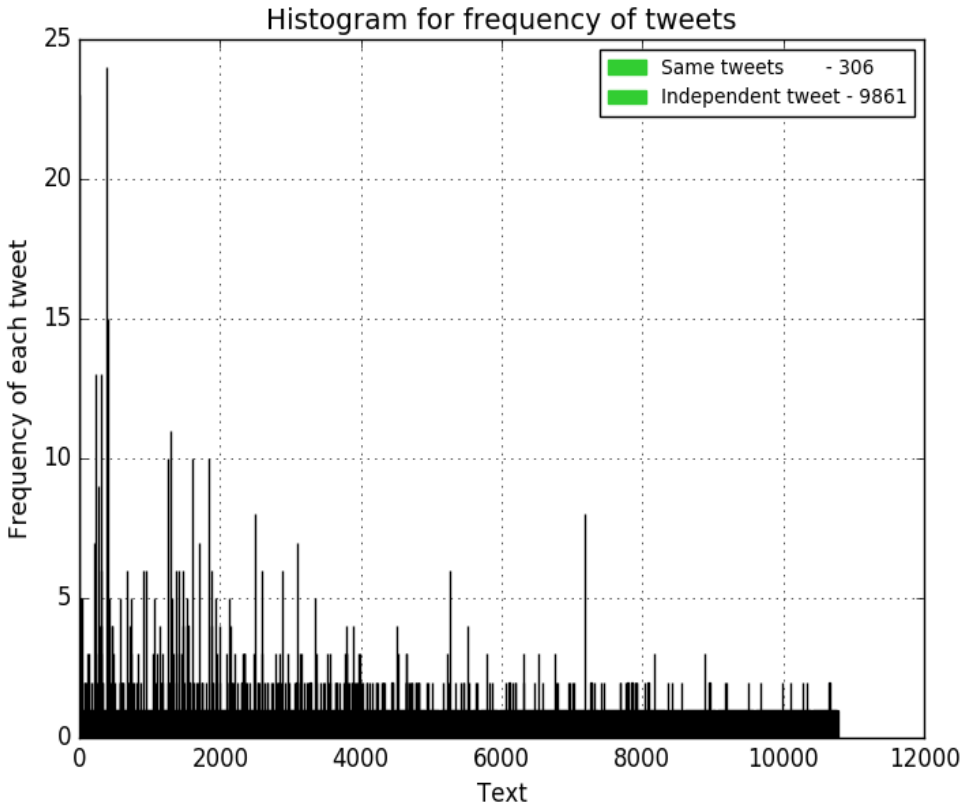

Figure 15: Independent and Duplicate Tweets

It is noted that a few texts have been floating more than other text on twitter, with the highest frequency of 23 in the given sample space. The Number of tweets that were same but with different "Tweet ID" were 306 and independent tweets were 9861 in count. Statistically there is no significant contribution of multiple copies of the same tweets. This does not appear to create any anomalies or skewness in the dataset so, we keep the tweets and do not change anything in it.

Twitter appears to add a 'RT' symbol in front of the tweets that are retweeted. In order to analyse this thing we compare the box plot with counts of 'individual' and 'retweeted' types of text.

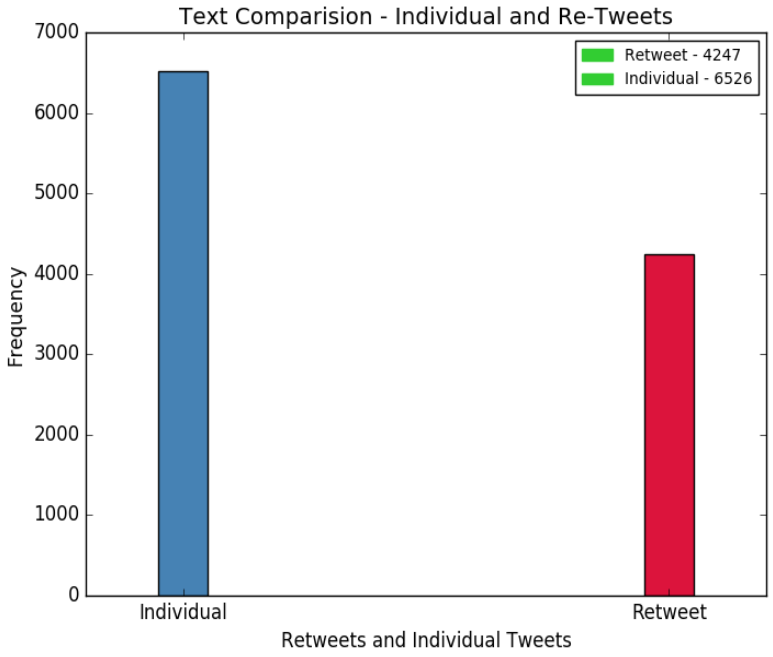

Figure 16: Text comparison of individual and retweeted text

Here, we see that there is a distribution of 40% and 60% between the individual and retweets which is a normal distribution.

ii. Tweets according to places

The following box plot shows the distribution of tweets coming from various parts of the country.

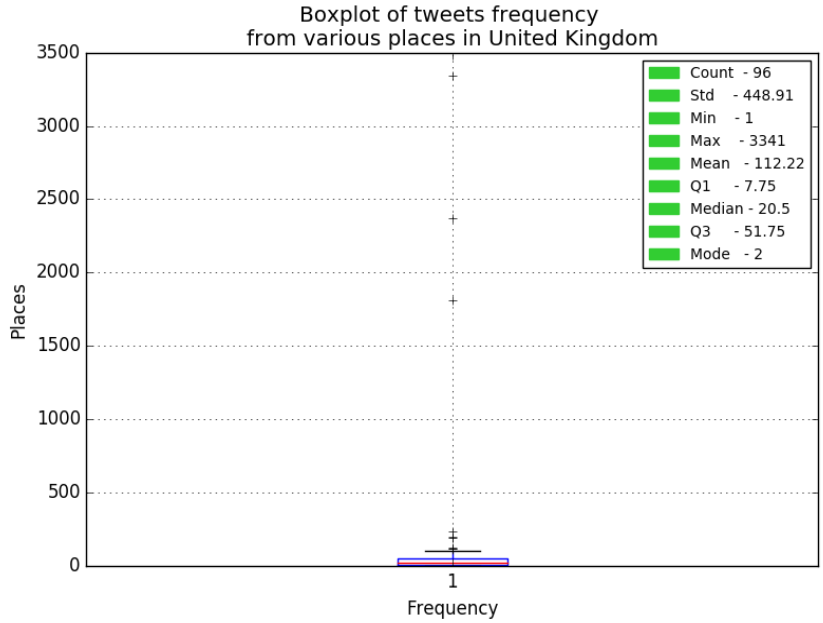

Figure 17: Box plot for tweets coming from different places/subdivisions of a country

It definitely shows that the tweets coming from the highest frequency regions will have a greater impact on the overall model. It also suggests that the tweets coming from the places with lower tweet frequency can be combined together in order to reduce the factor levels of the subdivisions. The major areas of the country might also help in understanding the bias in the voting patterns from the major subdivisions.

Figure 18 shows the highest to lowest frequency of tweets coming from various parts of the country.

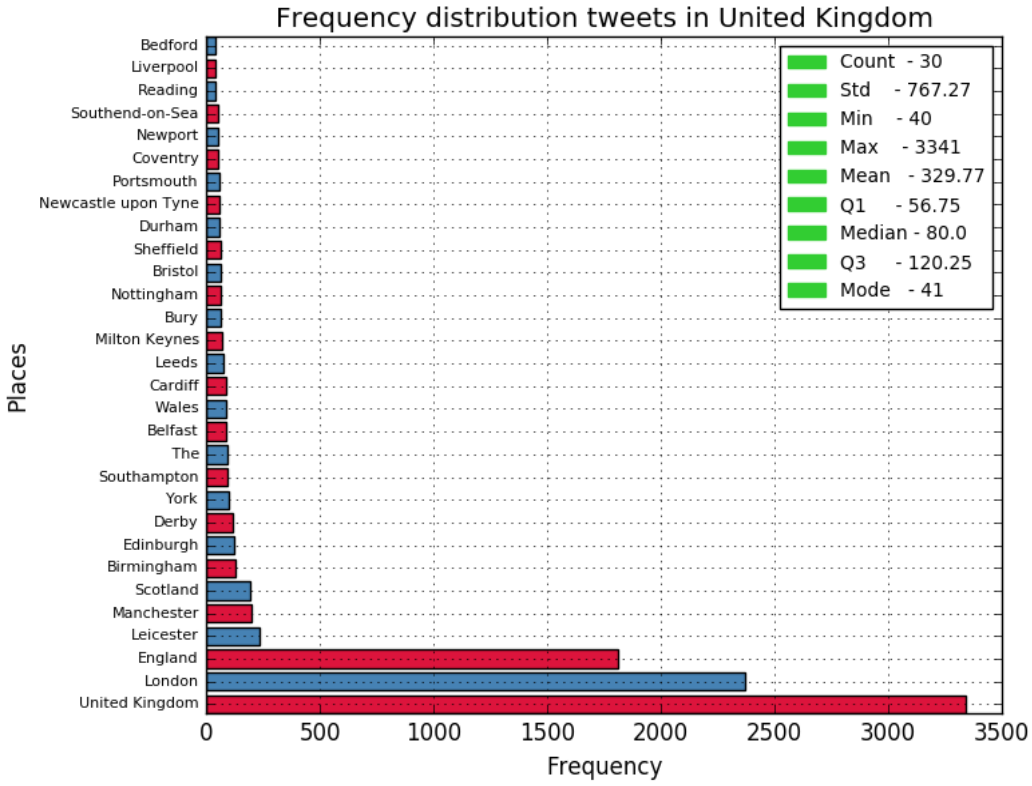

Figure 18: The Box plot for Frequency of tweets from various places

It suggests that in the United Kingdom overall, London, England, Leicester, Manchester, Scotland are the major areas from which tweets are coming and will definitely have significant impact on the overall model. However, others might not contribute significantly so, they might be combined together as one. Since, United Kingdom occurs in the subdivisions of the dataset. It will be a better that we group the data coming from other parts of UK into the same set i.e. United Kingdom.

iii. Tweets Analysis w.r.t Sources

Similar to the case above, the data received from various sources is skewed. The following three graphs also suggest that the people tend to use mobile devices to tweet more often than any other.

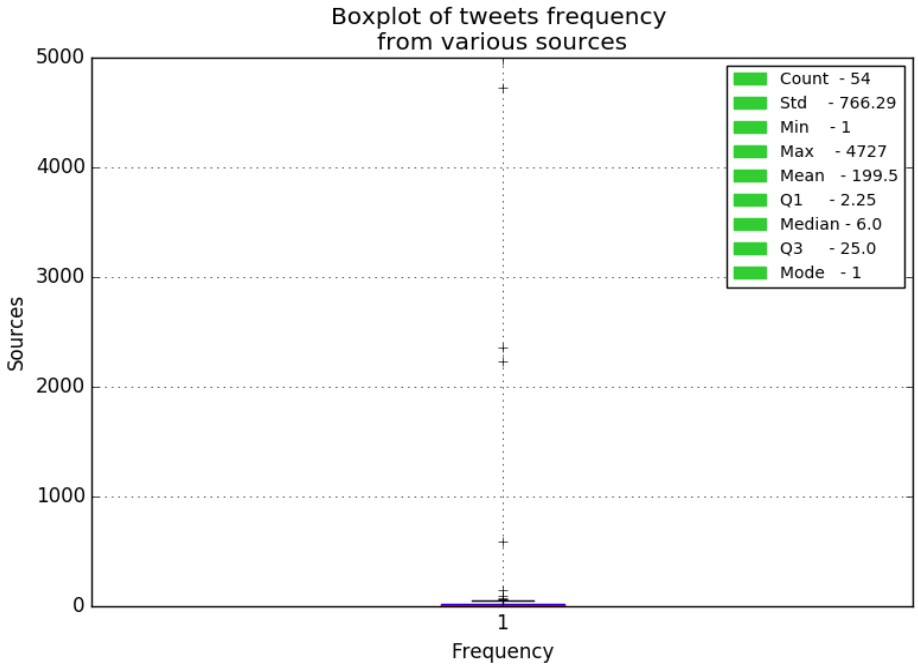

Figure 19: The Box plot of tweets frequency from various Sources/Devices

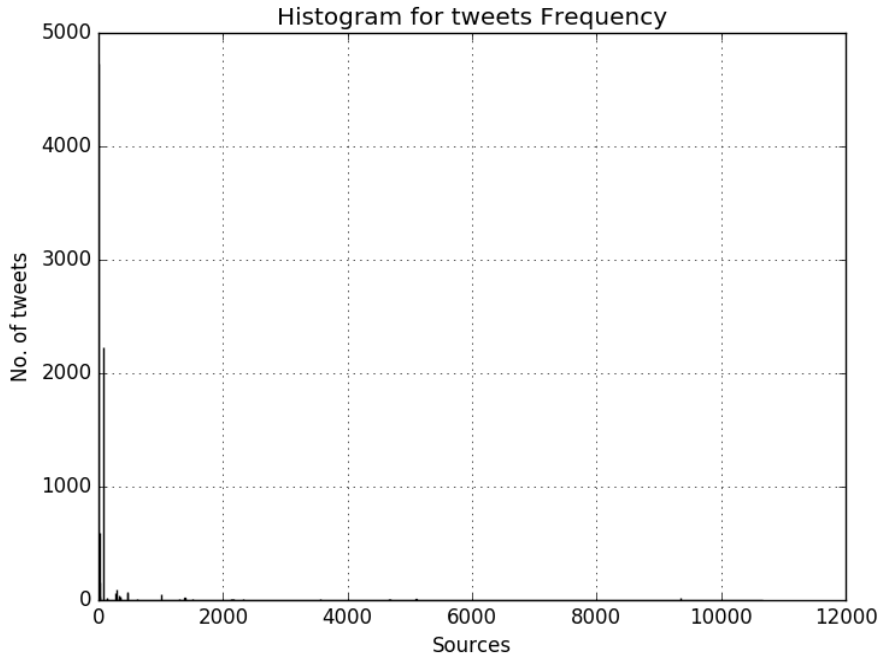

Figure 20: Histogram of tweets from various devices

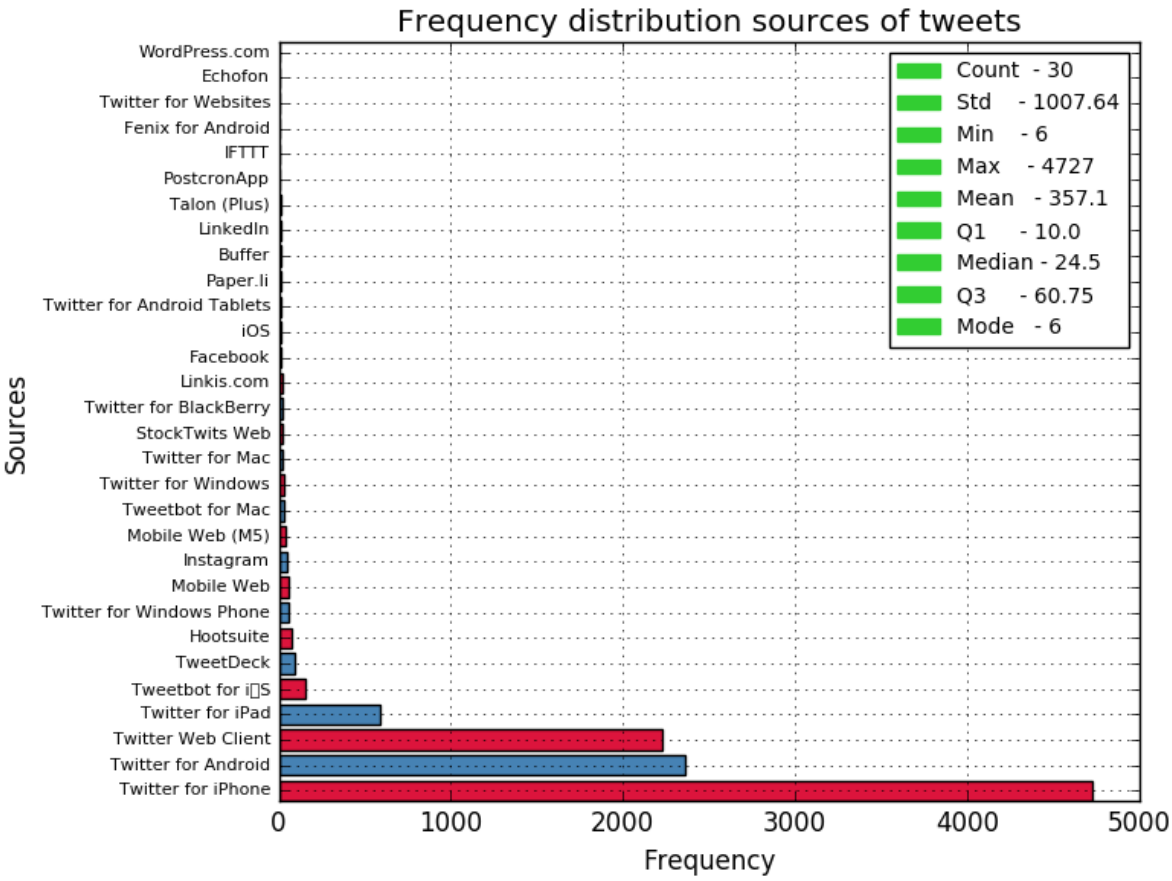

Figure 21: Box plot for Frequency distribution of tweets from various sources

It is seen from the above plots Fig. 3.5, 3.6 and, 3.7 that the iPhone, Android, web client and iPad users are the most impactful sources of tweets and plays a significant role. It might be useful to combine others into a single factor and reduce the factor levels. The significance of each factor level can be grouped into first few highest frequency devices and others can be grouped into a common set labelled as 'others'.

iv. Tweet Analysis w.r.t Users

The tweets coming from various users should be approximately normally distributed otherwise may create bias in the overall model. Although this is also similar to the cases above. A similar analysis shows that a few users tend to tweet more than others and hence lower frequency users might be combined together to reduce the factor levels.

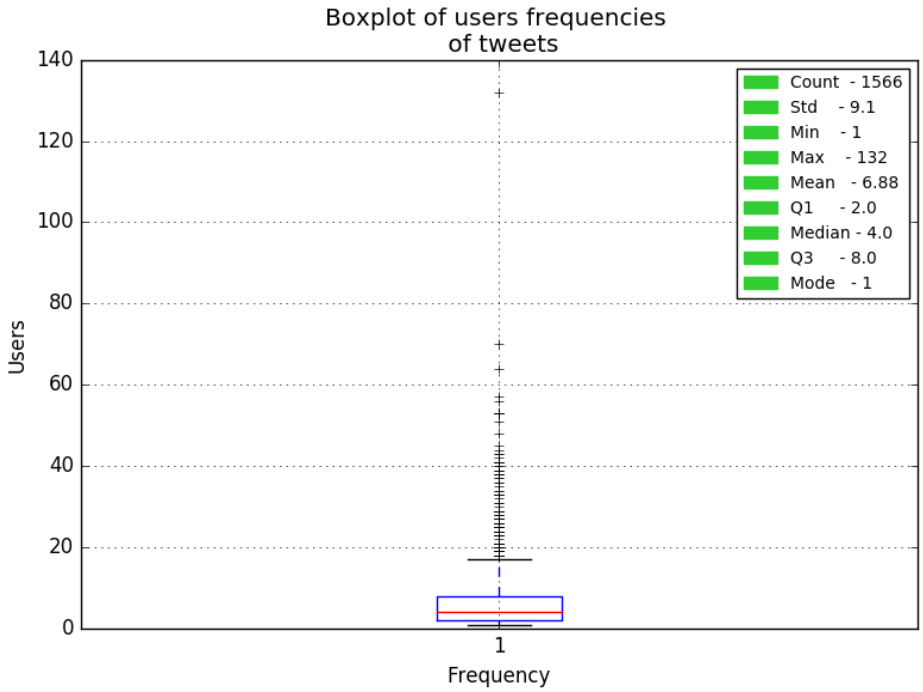

Figure 22: Box plot for users tweeting frequency

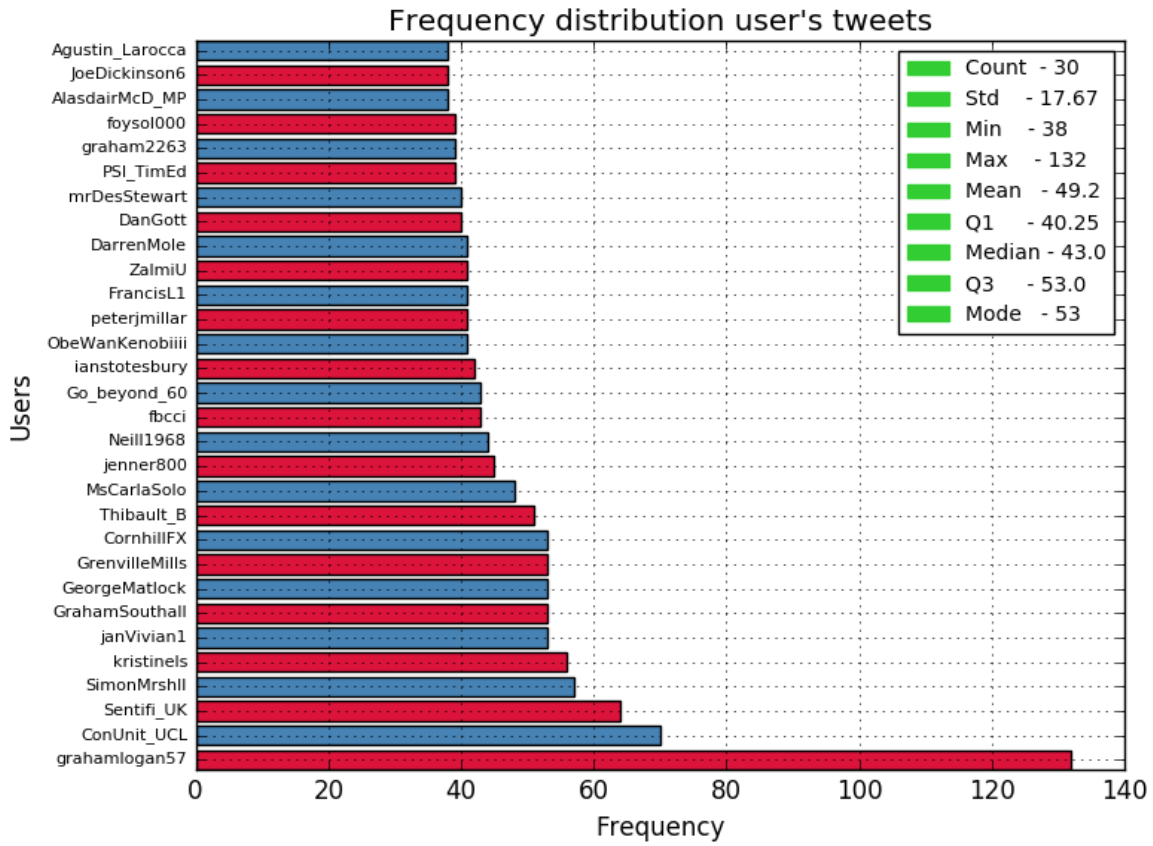

Figure 23: Box plot of users tweets

It is noted that the user grahamlogan57 appears to tweet more than others in the overall sample. It might be significant in the overall model and hence shall be kept in the model.

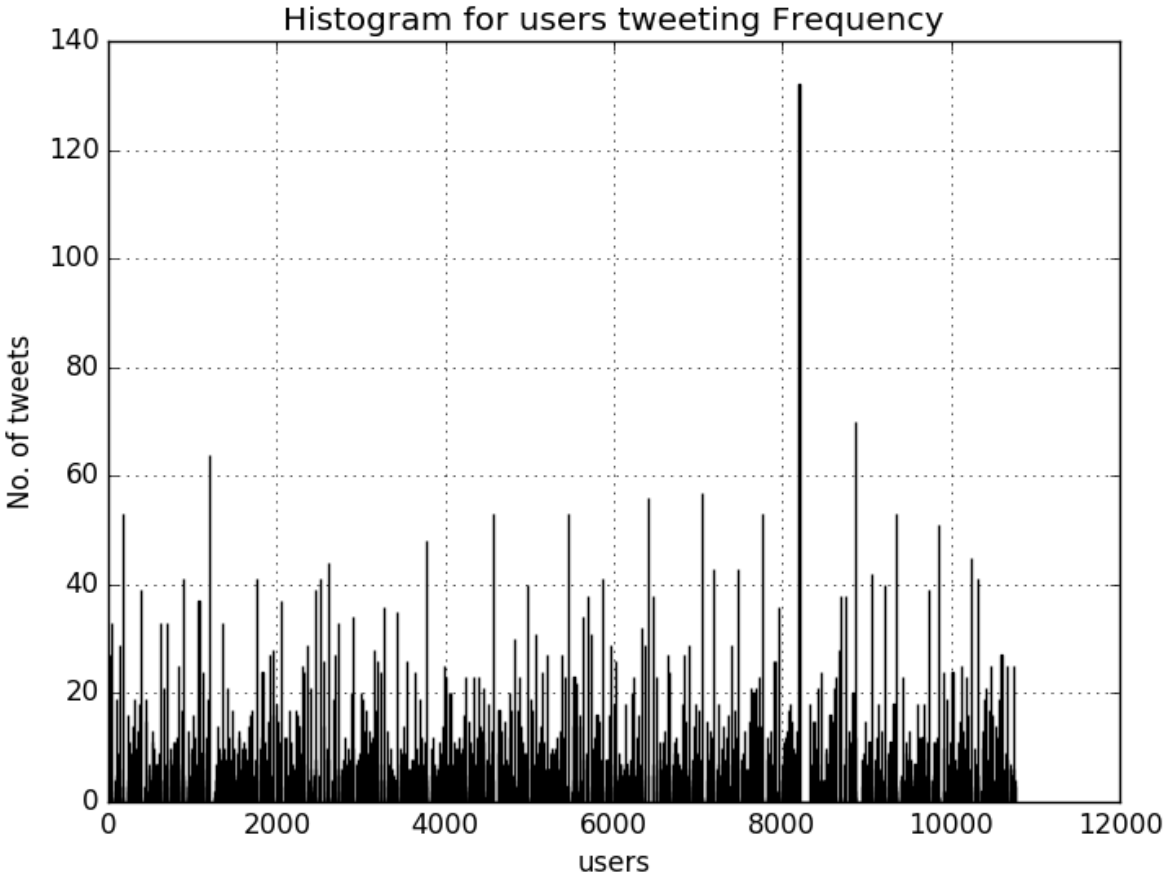

Figure 24: Histogram of users tweeting frequency

The histogram mentioned above shows the tweeting frequency of each user. It can be seen from the graph that these frequencies are normally distributed.

v. Time Series Analysis

Finally, we analyse the text on the basis of time. If we see a keyword that is for the first competing person/brand in the election, we mark it as zero and others as one. This is called recoding the data and is done in order to visualise the changes in conversation over time. The initial plot obtained follows:

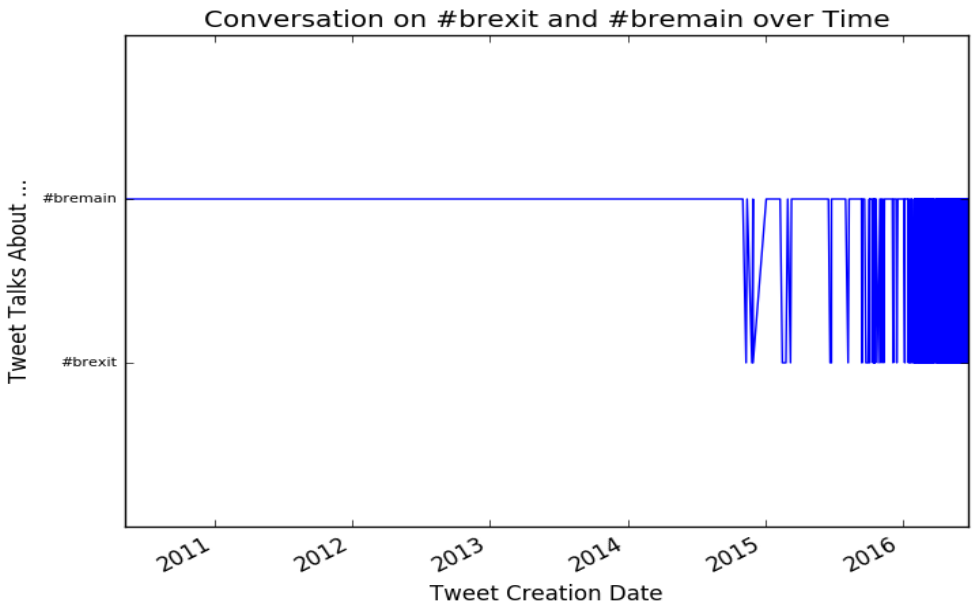

Figure 25: Time Series analysis of the frequency of the plots

It is seen that 2nd keyword impacts the overall analysis and creates a biased inference in the overall statistical analysis. So, the data is trimmed to contain only the conversations or tweets that were done in the year 2015 and 2016. After trimming of the data the following output was generated from the plot.

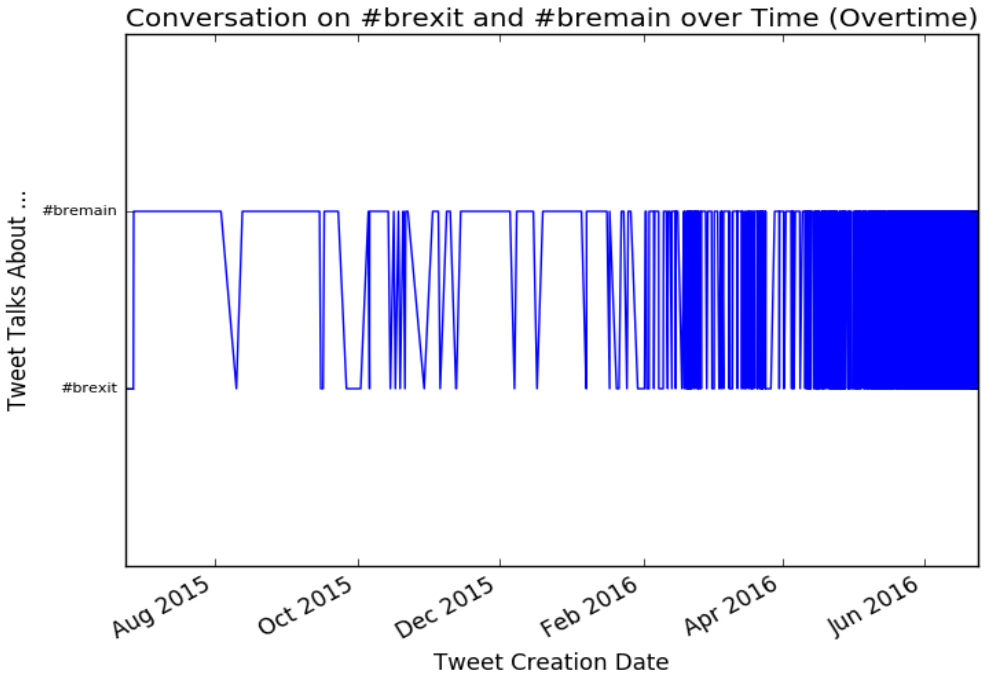

Figure 26: Time Series analysis of the frequency of the plots

e. Modelling

For modelling, the backward selection process was used in conjunction with interaction terms for dates, category and confidence in order to identify the statistically significant predictor in the model. The 'statsmodel' package in Python provides with 'patsy.dmatrices()' function that allows creation of dummy variables for the model with categorical data without manually assigning values by using pandas 'get_dummies()' function [37]. So, after reducing the factor levels and creating dummy variables, the logit function was applied on these models. The significance level of each predictor is then checked to test the hypothesis.

In order to evaluate each model, we used the AIC or the 'Akaike Information Criteria'. The model with least AIC value was chosen. Following were the scores of each model:

Table 7: Akaikie Information Criteria for each of the chosen models

Model	AIC
Model 1	12795.5
Model 2	12783.8
Model 3	12788.5
Model 4	12787.4

It can be seen from the above table that Model 2 turned out to be the best of all. It was also interesting to see that the predictor 'user – screen_name' was insignificant in all models tried. The following model was obtained having the least AIC value:

```
text ~ source + subdivision + statuses_count + followers_count +
retweet_count + contributors_enabled + category:confidence +
created_at_day:created_at_mon:created_at_year
```

Once the model is obtained, it is used to fit a logit function. With results summarised in the table below:

Table 8: Logistic regression model results

Logit Regression Results

Dep. Variable:	text	**No. Observations:**	10686
Model:	Logit	**DF Residuals:**	10667
Method:	MLE	**DF Model:**	18
Date:	Wed, 24	**Pseudo R-squ.:**	0.02809
Time:	Aug 2016	**Log-Likelihood:**	-6374.7
converged:	04:22:30	**LL-Null:**	-6558.9
	True	**LLR p-value:**	3.352e-67

	coef	Std err	z	P>|z|
Intercept	1.273	0.392	3.251	0.001
source[T.Tweetbot for iOS]	-0.400	0.321	-1.248	0.212
source[T.Twitter Web Client]	-0.640	0.264	-2.424	0.015
source[T.Twitter for Android]	-0.330	0.265	-1.250	0.211
source[T.Twitter for iPad]	-0.675	0.275	-2.459	0.014
source[T.Twitter for iPhone]	-0.297	0.262	-1.135	0.256
source[T.others]	-0.522	0.275	-1.904	0.057
subdivision[T.Leicester]	-1.670	0.151	-11.091	0.000
subdivision[T.London]	0.122	0.070	1.744	0.081
subdivision[T.Manchester]	0.370	0.177	2.085	0.037
subdivision[T.Scotland]	0.301	0.185	1.626	0.104
subdivision[T.United Kingdom]	0.017	0.060	0.296	0.768
contributors_enabled[T.True]	-0.198	0.154	-1.289	0.197
statuses_count	1.3e-05	4.11e-06	3.265	0.001
followers_count	-1.0e-05	6.56e-06	-1.637	0.102
retweet_count	0.000	2.71e-05	5.123	0.000
category[neg]:confidence	-0.007	0.003	-2.169	0.030
category[pos]:confidence	-0.006	0.004	-1.736	0.083
created_at_day:created_at_mon:created_at_year	2.3e-06	2.83e-07	8.199	0.000

Since the log likelihood ratio p-value (LLR p-value) is equal to 3.35e-67 which is less than 0.05 therefore, we reject the null hypothesis and conclude that the votes are dependent on the descriptor variables.

The confidence intervals and the odds ratios for the model are as follows:

Table 9: Confidence Interval and odds ratio for the selected model

Discriptors	Lower CI	Upper CI	Odds Ratio
Intercept	1.65818	7.70053	3.57335
source[T.Tweetbot for iOS]	0.356763	1.25721	0.669722
source[T.Twitter Web Client]	0.314199	0.884565	0.52719
source[T.Twitter for Android]	0.427717	1.20651	0.718361
source[T.Twitter for iPad]	0.297028	0.871931	0.508909
source[T.Twitter for iPhone]	0.443986	1.24154	0.742447
source[T.others]	0.346174	1.01534	0.592862
subdivision[T.Leicester]	0.140089	0.252801	0.188188
subdivision[T.London]	0.984923	1.2971	1.13029
subdivision[T.Manchester]	1.02243	2.04994	1.44773
subdivision[T.Scotland]	0.940015	1.94475	1.35207
subdivision[T.United Kingdom]	0.905354	1.14424	1.01781
contributors_enabled[T.True]	0.606731	1.10868	0.820163
statuses_count	1.00001	1.00002	1.00001
followers_count	0.999976	1	0.999989
retweet_count	1.00009	1.00019	1.00014
category[neg]:confidence	0.986506	0.999312	0.992888
category[pos]:confidence	0.98691	1.0008	0.993833
created_at_day:created_at_mon:created_at_year	1	1	1

So, it can be seen from the model above the following descriptor variables are statistically significant- “statuses count, retweet count, subdivision – Leicester and, Manchester, combined effect of category and confidence, source - Twitter Web Client and, Twitter for iPad, combined effect of ‘created_at_day, created_at_mon, created_at_year’ and, Intercept”.

However, variables “followers count, contributors enabled - True, source - Tweetbot for iOS, Twitter for Android, Twitter for iPhone and, others, category[pos], confidence and, subdivision - London, Scotland and, United Kingdom” are insignificant.

The following table shows the predictor in descending order w.r.t. its effect on the overall model. The table shows the likelihood of people voting for #bremain:

Table 10: Shows likelihood of people voting for #bremain in presence of each predictor

Predictor	Influence	Orientation	Reference label
Intercept	257.34%	more likely	When there are no tweets
Leicester	81.18%	less likely	Compared to England
iPad	49.11%	less likely	Compared to Tweet deck
Web Client	47.28%	less likely	Compared to Tweet deck
Manchester	44.77%	more likely	Compared to England
Effect of negative sentiment	0.01%	less likely	Than a positive sentiment
Retweet count	0.01%	more likely	With increase in no. of tweets
Statuses count	0.00%	equally likely	With increase in no. of tweets
date of the tweet	0.00%	equally likely	With increase in no. of tweets
followers count	Insignificant		With increase in no. of tweets
contributors enabled	Insignificant		contributors disabled
iOS	Insignificant		Compared to Tweet deck
Android	Insignificant		Compared to Tweet deck
iPhone	Insignificant		Compared to Tweet deck
Positive sentiment	Insignificant		Than a negative sentiment
Confidence	Insignificant		With increase in no. of tweets
London	Insignificant		Compared to England
Scotland	Insignificant		Compared to England
Other places in United Kingdom	Insignificant		Compared to England

f. Diagnostics

Dependent variable (y), predicted values of y(ypred), error terms (e), hat values (h), Pearson's residual values (rpear), deviance residual values (rdev) and, cook's distance values (D) are combined together for closer investigation. Below is a small portion of the data frame generated by combining these values:

Table 11: Diagnostics summary from a small portion of the data.

h	y	rdev	rpear	e	ypred	d
0.000644226	0	-1.62055	-1.64853	-0.731012	0.731012	0.00087595
0.000964369	1	0.892366	0.699343	0.328445	0.671555	0.000236055
0.000835228	1	0.74653	0.566877	0.243198	0.756802	0.000134312
0.00141618	1	0.729724	0.552322	0.233751	0.766249	0.000216316
0.00080248	1	0.790287	0.605419	0.268221	0.731779	0.000147186

From the plots of Pearson's residual and Deviance residual it can be seen that the values are randomly scattered, having bands of rectangular clouds. So, the systematic component of the model is correct.

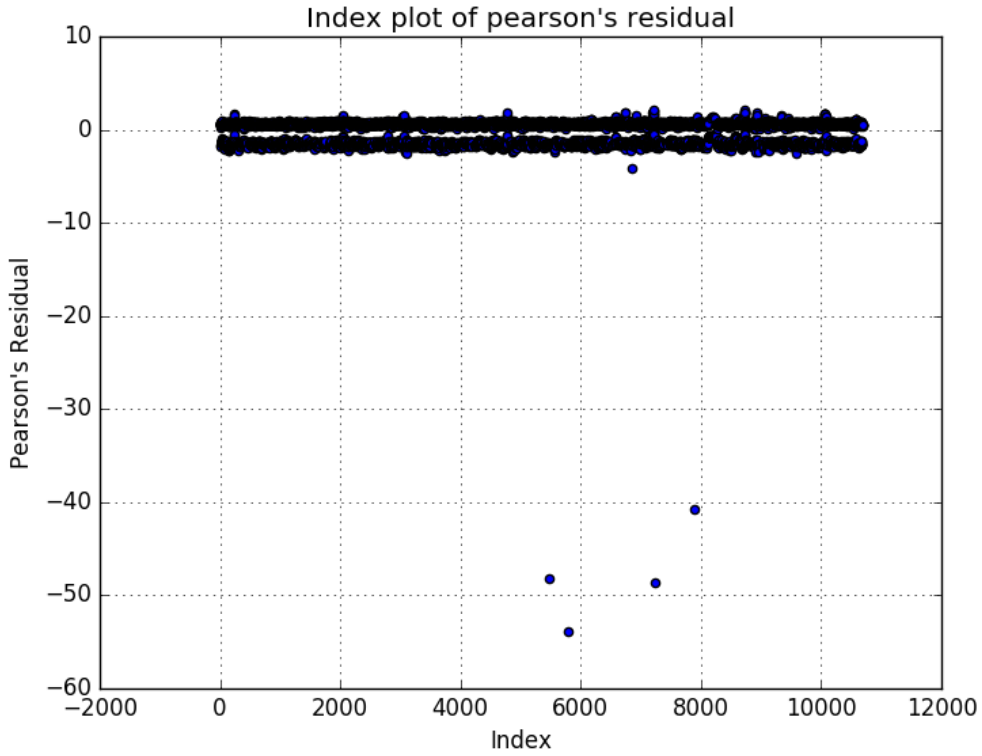

Figure 27: Pearson's Residual vs. Index plot for verifying the systematic component

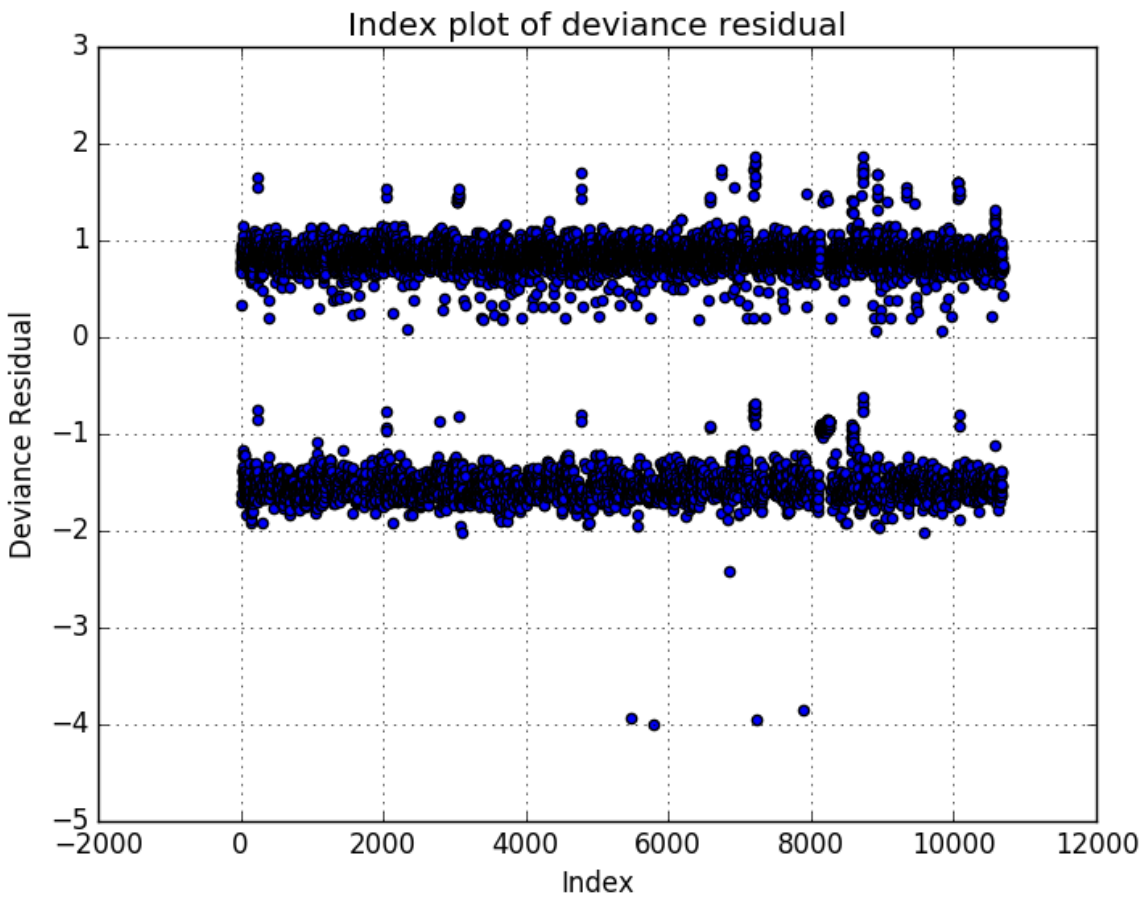

Figure 28: Index plot of Deviance Residual

The following graphs show the cases of high leverage and cases of high influence.

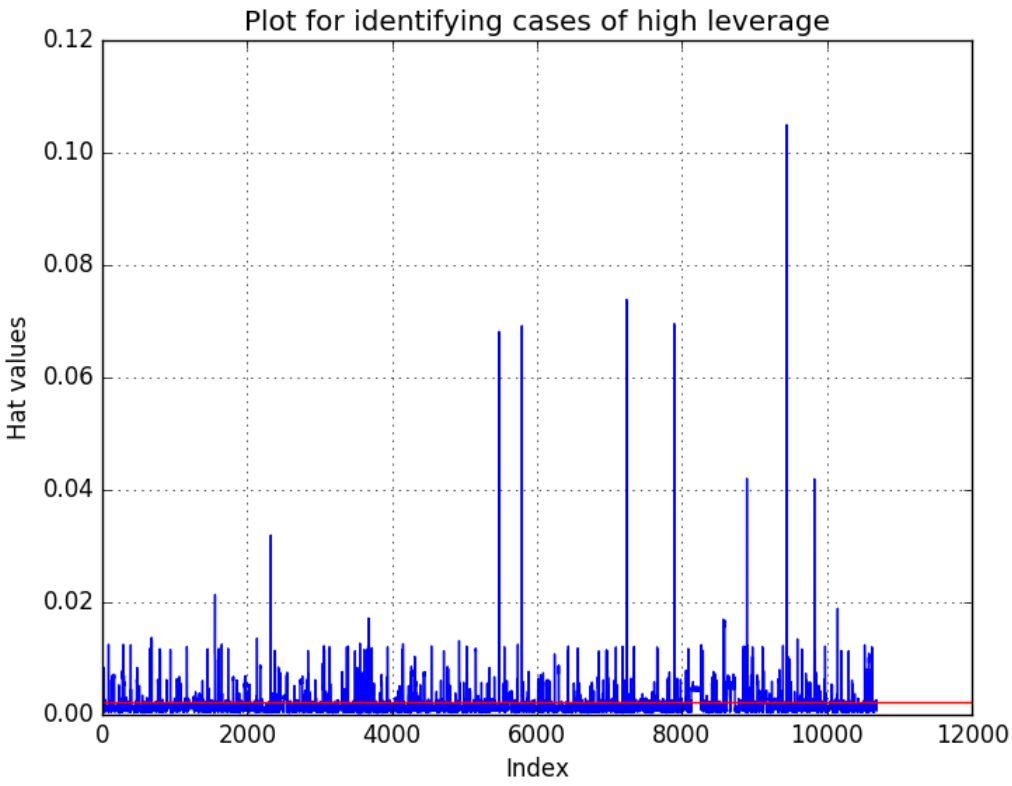

Figure 29: Plot for identifying cases of high leverage

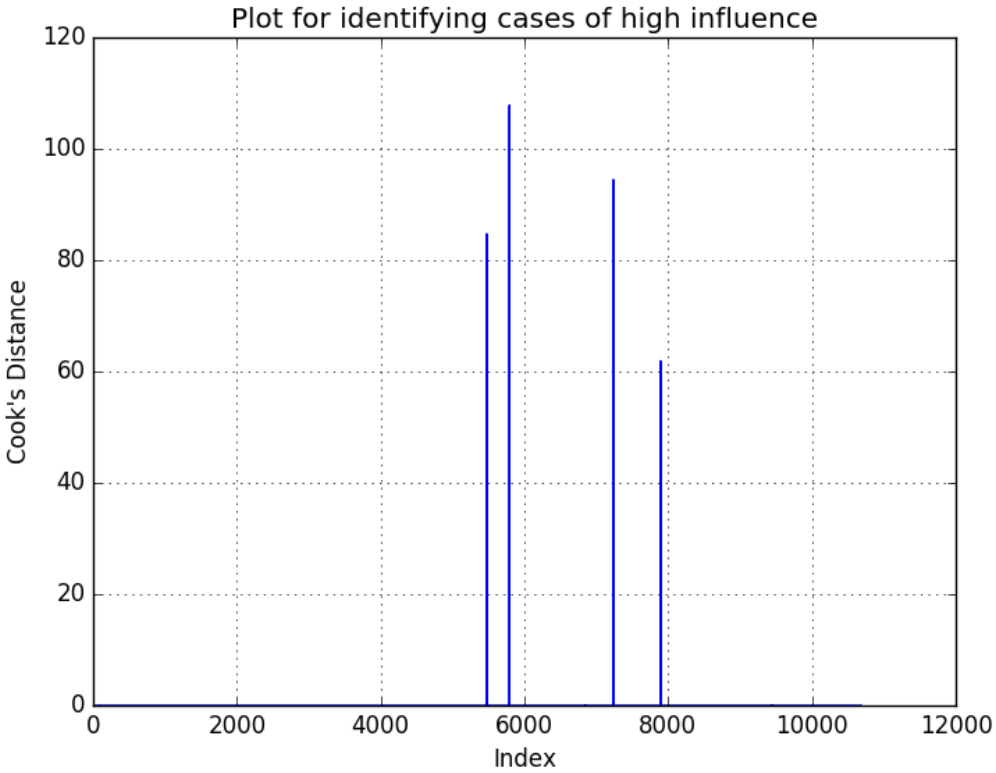

Figure 30: Plot for identifying cases of high influence

Fig 1.15 above shows the cases of high leverage. The values above the cut-off point marked in red are the cases of high leverage. In this case, we have 1951 cases of high leverage.

The Fig 1.16 shows the cases of high influence. In this case, we have 4 cases of high influence and they are indexed as 5475, 5792, 7236, and 7894 in the dataset.

g. Accuracy

The accuracy of the model is obtained by considering the predicted values above 0.5 as #bremain while, values equal to or below 0.5 as #Brexit. A confusion matrix is plotted in order to obtain the overall accuracy of the model. The following table shows the confusion matrix:

Table 12: Confusion Matrix for the selected Model

		Predicted		Total
Actual	#brexit	177	3066	3243
	#bremain	84	7359	7443
	Total	261	10425	10686

So, the model is accurately able to identify 5.46% of the results for #brexit and 98.87% for #bremain. Also, the overall accuracy of the model is 70.52%. Consequently, it can be said that 48.56% of the people are sure to vote for #bremain.

4. Results

There are a few interesting facts that can be obtained from the interpretation of the given model.

- We see that people writing about #brexit tend to use more of the web clients and ipads. This might suggest that people who think that it is necessary to make others realize the importance of #brexit, substantially took out their time from the daily schedule to sit around a less mobile device to tweet about their opinion on #brexit.
- The data also suggested that people from southern part (Leicester) of the United Kingdom were more likely to vote for #brexit than people in the northern part (Manchester) of the United Kingdom. This interpretation is also in line with the actual voting pattern of the referendum, which strongly supports that the model is good enough to predict quality results. However, It is surprising to see that the tweets coming from London are insignificant which might be because of assertive expressions or different languages used for tweeting.
- We also see that the 'retweeting' has a very small impact (0.01%) on changing the opinion of the people. This is quite obvious that people who tend to agree on a particular tweet, are the ones tweeting back. So, it does not help in changing their mindset.
- We also see that the status count is significant in the model but has no effect on the votes. This is because of the fact that people who tweet, tend to tweet more than others. So, they a significantly visible but no matter how much they tweet it does not change other people's point of view. It might however, be useful spreading the word around but has no overall impact.
- The sentiment of the tweets shows some obvious facts. A negative tweet has a significant impact (though by a very small percentage i.e. 0.01%) on the voting patterns of the users. It can be seen from the interpretation that people tend to vote more for #brexit if the sentiment of the tweet is negative. This inference is also in line with the human behavior in general and so, we can conclude that sentiments do help in predicting the voting patterns in any election campaign. We can also conclude that if a negative sentiment has a greater impact on a particular topic of discussion, it is bound to be more controversial/aggressive.
- Date and time is also significant in the model. It is quite obvious that tweet and time of arrival of tweet are correlated. However, it is seen that date and time do not contribute towards user's vote. So, it can be said that the tweets are not automatically generated at any particular instance of time and hence might be authentic user's tweets.

5. Conclusion

This work suggest that the regional voting patterns can be predicted with reasonable accuracy based on twitter sentiment analysis alone. The results however also show different aspects of the data collected. It is clear from the interpretation of data that it is difficult to identify what percentage of the votes were for #brexit. It is unlikely that this is a statistical difference due to insufficient data for #brexit relative to #bremain, but is probably more a social issue in terms of assertive expression or language used in the face of cosmopolitan media based in London, in contrast to the country folk who may keep views private to avoid upsetting rural neighbours, whose families and politics are known for generations.

People on the social web tend to talk more about things that are favored by a broader class of society than things that need more thoughtful and rigid decision making mindset. The given model predicts that almost 48.56% of the population will vote for #bremain which is quite close to the overall votes received in the actual referendum i.e. 49% for #bremain. Therefore, the accuracy of the model cannot be denied. However, the overall picture of any election campaign is difficult to obtain may be because “people tend to refrain from talking about any controversial topic in general” or perhaps it is merely “a manifestation of the classic city/country divide”.

6. Future Work

Better information can be obtained by keeping track of the voting patterns observed on twitter. Also other learning algorithms can be implemented to get better solutions that give clearer pictures of a given scenario. For example, knowing the sentiment as positive, negative or neutral can give a more accurate depiction of the patterns. Better visualization techniques can be used to bring the results to life. Moreover, the present model can be extended to compare multiple campaigns such as using decision trees, random forest and other learning techniques, which can extend the model to fit in a multilateral governmental system. Furthermore, other social networking sites can be scraped for obtaining similar kinds of data and checked for the authenticity and quality of results obtained. Once, results from different social networking sites are available, they can be compared for quality of data and hence a rating metrics can be created for each one of them which might be dynamic in nature.

Appendices

Appendix A: Python Programs

Below is an overview of the source code. Further information about it is maintained in the GitHub repository: https://github.com/praveer-k/campaign-monitor .

1. **Packages**

```
Default:
sys, os, json, pickle, zipfile, statistics, datetime, collections, re, html, time, random

Installed:
tweepy, nltk, sklearn, tabulate, progress, openpyxl, pymongo, pandas, statsmodels, numpy,
pycountry, yaml, matplotlib

User-Defined:
lib.helper, lib.graphs
```

2. **Constants**

```
KEYWORD1              # for first competing brand
KEYWORD2              # for second competing brand
SINCE                 # start date for downloading tweets
UNTIL                 # end date for downloading tweets
DBName                # new name given to the database for tweets
COUNTRY               # Country from where tweets are downloaded
CACHE                 # whether to use cached data for each country or not
PORT                  # port number for mongodb server.
```

3. **Helper functions**

```
get_all_countries()
get_variables(path)
wait_to_recover(func)

@wait_to_recover
find_place_ids(api, country_code, subdivision)
download_place_ids(api, country)

@wait_to_recover
find_user_ids(api, keyword, country_code, place_id)
download_user_ids(api, keyword, country)

@wait_to_recover
download_users_timeline(api, data)
save_data_to_mongoDB(port_number, DBName, dirpath, user_ids)
```

4. Graph functions

```
patches(legend_labels)

summary_legend(vals)

Hist(df, colname, xlabel, ylabel, title, figpath, legend=None)

HBar(df, colname, xlabel, ylabel, title, figpath)

Boxp(df, colname, xlabel, ylabel, title, figpath)
```

5. User-Defined Class

```
class Tweets:
    def __init__(self, max_features=5000):
        self.__tokenizer  = TweetTokenizer()
        self.__stemmer    = SnowballStemmer("english")
        self.__stop_words = stopwords.words('english')
        self.__all_words  = []
        self.__document   = []
        self.  max features = max features # Predictor for the algorithm
        self.featuresets = []  # Publicly available data

    def push(self, tweets, category):
        for tweet in tweets:
            self.  document.append( (tweet, category))
            words = self.__tokenizer.tokenize(tweet)
            words = [w.lower() for w in words if w.lower() not in self.__stop_words]
            self.__all_words += words

    def pickle_featuring_words(self, path):
        wordFreq = FreqDist(self.__all_words)
        featuring_words = list(wordFreq.keys())[:self.__max_features]
        f = open(path, 'wb')
        pickle.dump(featuring_words, f)
        f.close()

    def create_feature_sets(self):
        wordFreq = FreqDist(self.__all_words)
        featuring_words = list(wordFreq.keys())[:self.__max_features]
        self.featuresets = []
        for text, category in self.__document:
            words = self.__tokenizer.tokenize(text)
            words = [w.lower() for w in words if w.lower() not in self.__stop_words]
            words = [self.__stemmer.stem(w) for w in words]
            features = {}
            for w in featuring_words:
                features[w] = (w in words)
            self.featuresets += [(features, category)]
```

Appendix B: JSON data format

```
statuses = { 'user': None, 'timeline' : [] }

statuses['user'] =  { # --------- User Info -----------
                      'user_id'               : status.user.id_str,
                      'name'                  : status.user.name,
                      'screen_name'           : status.user.screen_name,
                      'user_created_at'       : status.user.created_at,
                      'description'           : status.user.description,
                      'friends_count'         : status.user.friends_count,
                      'statuses_count'        : status.user.statuses_count,
                      'followers_count'       : status.user.followers_count,
                      'favourites_count'      : status.user.favourites_count,
                      'contributors_enabled'  : status.user.contributors_enabled,
                       # ----------- Place Info -----------
                      'place_id'              : data['place_id'],
                      'subdivision'           : data['subdivision'],
                      'location'              : status.user.location
                    }
```

```
statuses['timeline'].append({ # ----------- Tweet Info ----------
                              'tweet_id'        : status.id,
                              'created_at'      : status.created_at,
                              'lang'            : status.lang,
                              'retweeted'       : status.retweeted,
                              'text'            : text,
                              'links'           : url,
                              'retweet_count'   : status.retweet_count,
                               # --------- Author Info ----------
                              'author_id'       : status.author.id_str,
                              'author_screen_name': status.author.screen_name,
                               # --------- Source Info ----------
                              'source'          : source
                             })
```

Here, **status** is the data received from the **Twitter API** whereas the **statuses** is the JSON object stored in the MongoDB.

Appendix C: Models

After removal of the variables that were either singular labels or did not have any of its factors as significant, few of the following given formulas stood out as the potential candidates for modelling. One of them were then selected using the least AIC.

```
formulas = [ 'text ~ source + subdivision + statuses_count + followers_count +
retweet_count + contributors_enabled + category + confidence + created_at_day +
created_at_mon + created_at_year',
             'text ~ source + subdivision + statuses_count + followers_count +
retweet_count + contributors_enabled + category + confidence + category:confidence +
created_at_day + created_at_mon + created_at_year +
created_at_day:created_at_mon:created_at_year',
             'text ~ source + subdivision + statuses_count + followers_count +
retweet_count + contributors_enabled + category + category:confidence +
created_at_day:created_at_mon:created_at_year',
             'text ~ source + subdivision + statuses_count + followers_count +
retweet_count + contributors_enabled + category:confidence +
created_at_day:created_at_mon:created_at_year'
           ]
results = []
print('Model Selection...')
for formula in formulas:
    formula = helper.removeUnwanted(formula, df.columns)
    y,X = patsy.dmatrices(formula, df, return_type='dataframe')
    logit = sm.Logit(y, X)
    result = logit.fit()
    results.append( (formula, y, X, result) )

table = {'Model':[],'AIC':[]}
i = 1
for result in results:
    table['Model'].append('Model '+str(i))
    table['AIC'].append(result[3].aic)
    i += 1

results = [result for result in results if result[3].mle_retvals['converged']==True]
formula, y, X, result = min(results, key=lambda x: x[3].aic)
```

Appendix D: Project Management

Figure 29 shows the folder structure used in the project.

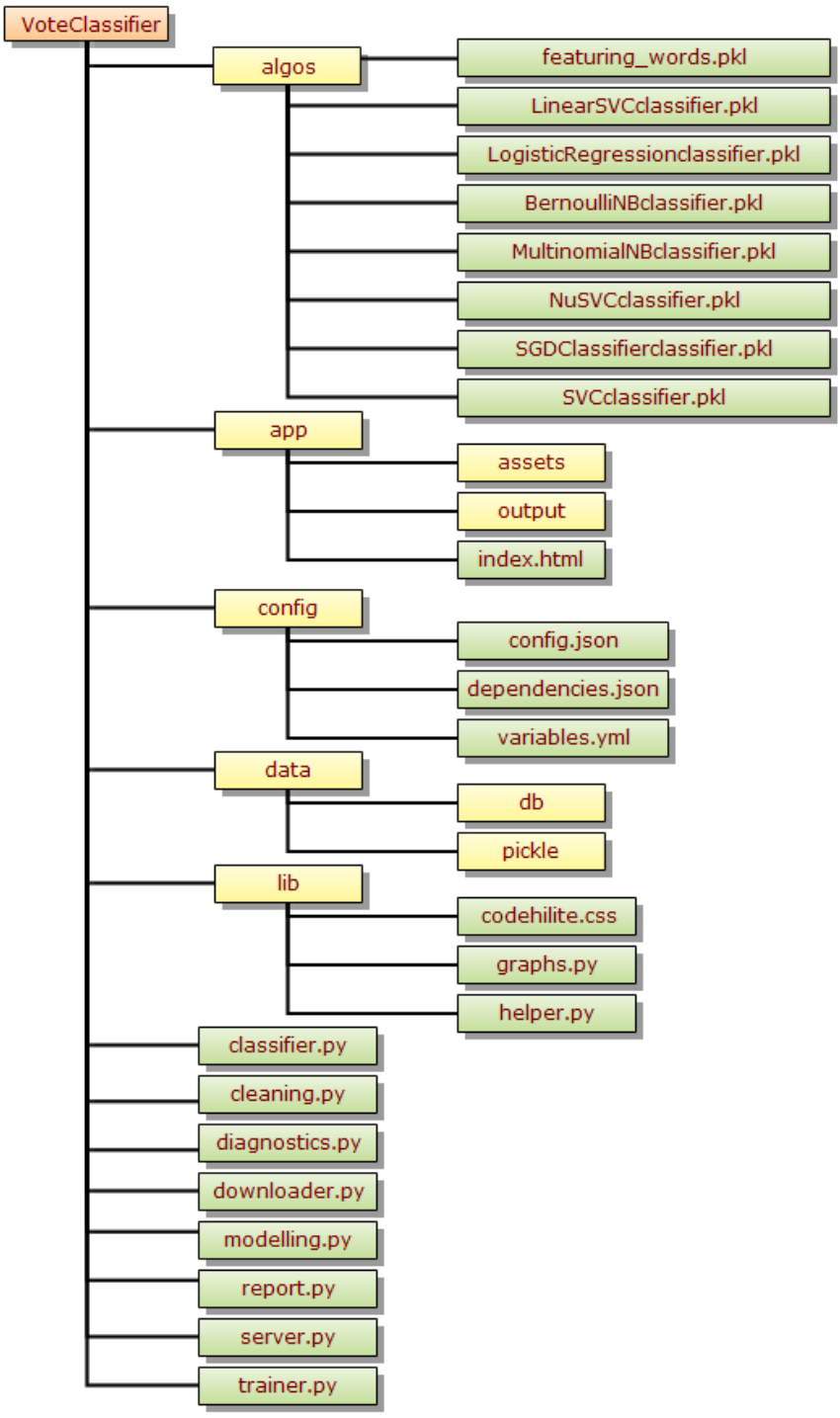

Figure 31: Folder Structure for the Project

The directory structure was carefully structured in order to make the project management easier.

The project spans over a time period of 101 days from 13th May 2016 to 31th August. Following plan was followed in order to achieve the targets.

Job	Start Date	End Date	Duration	Comments	Completed
Literature Review	13-May-16	13-Jun-16	31	Experimented with codes and Read books	Yes
Submission of the Abstract	14-Jun-16	15-Jun-16	1	Initial Idea	Yes
Proposing Initial Project Plan	16-Jun-16	28-Jun-16	12	Initial draft	Yes
Data Extraction	28-Jun-16	3-Jul-16	5	Download data from social media	Yes
Proposing Prediction Model/s	4-Jul-16	11-Jul-16	7	Initial Working Model	Yes
Re-Extraction of Data	8-Jul-16	11-Jul-16	3	Anamolies detected in the downloaded data	Yes
Writing Code for First Working Model and Visualise Output	12-Jul-16	16-Jul-16	4	Initial Draft	Yes
Work on Changes	16-Jul-16	20-Jul-16	4	Source Code changes	Yes
Submission of Thesis (Initial Draft)	21-Jul-16	28-Jul-16	7	Stager through 21st till 28th July	Yes
Cross validate the output	29-Jul-16	31-Jul-16	2	Report for anamolies	Yes
Stating the Result and Concluding from the Prediction	1-Aug-16	8-Aug-16	7	Initial Results	Yes
Finalise Analysis and Conclusion	9-Aug-16	12-Aug-16	3	Finalise Analysis and results	Yes
Full Thesis Report Submission (2nd Iteration)	12-Aug-16	18-Aug-16	6	2nd Iteration	Yes
Thesis Reviews and changes	19-Aug-16	27-Aug-16	8	Stager the required changes and work on critical remarks	Yes
Final Report Completion	27-Aug-16	31-Aug-16	4	Reviews and Corrections	Yes

Figure 32: Project schedule

Following graph shows the Gantt chart of the above mentioned schedule.

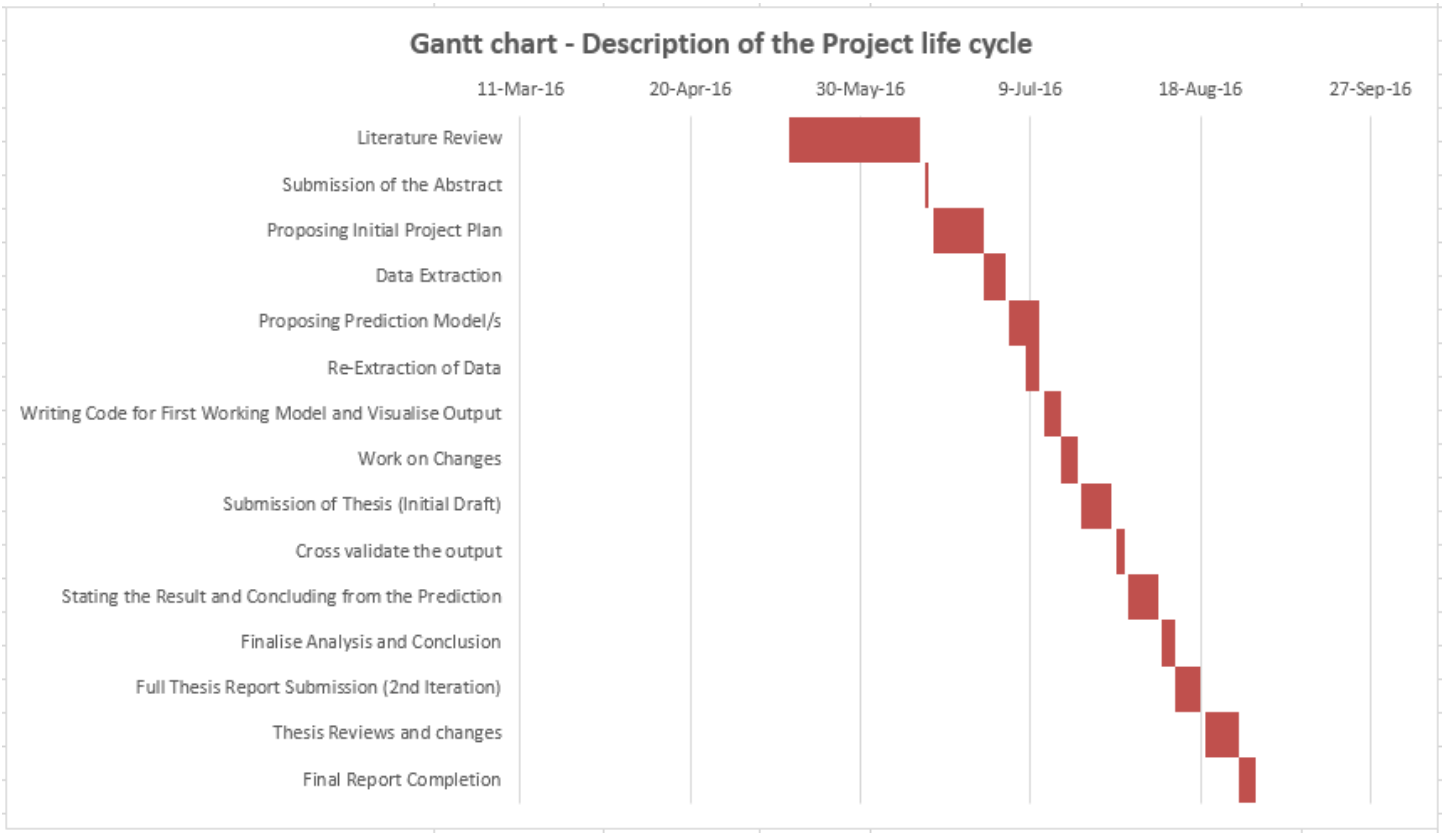

Figure 33: Gantt chart to show description of the project life cycle

References

[1] J. MacNeil, "UNIVAC predicts election results," 04 November 2015. [Online]. Available: http://www.edn.com/electronics-blogs/edn-moments/4423752/UNIVAC-predicts-election-results--November-4--1952.

[2] N. Silver, "The Signal and The Noise," in *The Signal and The Noise*, New York, The Penguin Press, 2012, pp. 9-10.

[3] "https://en.wikipedia.org/wiki/Data_mining," https://en.wikipedia.org/, [Online]. Available: https://en.wikipedia.org/wiki/Data_mining. [Accessed 10 07 2016].

[4] "http://www.qrious.co.nz/products/advanced-analytics/," [Online]. Available: http://www.qrious.co.nz/products/advanced-analytics/. [Accessed 21 07 2016].

[5] "wolfram mathworld," Wolfram Research, Inc., 2016. [Online]. Available: http://mathworld.wolfram.com/Outlier.html. [Accessed 16 07 2016].

[6] S. Weisberg, Applied Linear Regression, Hoboken, New Jersey: A JOHN WILEY & SONS, INC., PUBLICATION, 2005.

[7] "http://www.statisticshowto.com/how-to-find-a-linear-regression-equation/," Statistics How To, [Online]. Available: http://www.statisticshowto.com/how-to-find-a-linear-regression-equation/. [Accessed 10 07 2016].

[8] "Association_rule_learning," [Online]. Available: https://en.wikipedia.org/wiki/Association_rule_learning. [Accessed 20 08 2016].

[9] "https://en.m.wikiversity.org/wiki/Correlation," Wikiversity, [Online]. Available: https://en.m.wikiversity.org/wiki/Correlation#Coefficient_of_determination. [Accessed 10 07 2016].

[10] R. Cornish, *clusteranalysis,* http://www.statstutor.ac.uk, 2007.

[11] "http://www.sthda.com/english/wiki/clustering-validation-statistics-4-vital-things-everyone-should-know-unsupervised-machine-learning," [Online]. Available: http://www.sthda.com/english/wiki/clustering-validation-statistics-4-vital-things-everyone-should-know-unsupervised-machine-learning. [Accessed 20 07 2016].

[12] D. W. Gareth James, "Classification," in *An Introduction to Statistical Learning*, Springer, 2013, pp. 130-137.

[13] D. w. Gareth James, "An overview of statistical learning," in *An Introduction to Statistical Learning*, Springer, 2013, p. 1.

[14] "http://www.nltk.org/," [Online]. Available: http://www.nltk.org/book/. [Accessed 10 07 2016].

[15] "http://www.nltk.org/," [Online]. Available: http://www.nltk.org/. [Accessed 10 07 2016].

[16] "https://en.wikipedia.org/wiki/Machine_learning," [Online]. Available: https://en.wikipedia.org/wiki/Machine_learning. [Accessed 10 07 2016].

[17] "https://en.wikipedia.org/wiki/Logistic_regression," [Online]. Available: https://en.wikipedia.org/wiki/Logistic_regression#Bayesian_logistic_regression. [Accessed 10 07 2016].

[18] "http://scikit-learn.org/stable/modules/naive_bayes.html," [Online]. Available: http://scikit-learn.org/stable/modules/naive_bayes.html. [Accessed 10 07 2016].

[19] "http://scikit-learn.org/stable/modules/sgd.html," [Online]. Available: http://scikit-learn.org/stable/modules/sgd.html. [Accessed 10 07 2016].

[20] "http://ufldl.stanford.edu/tutorial/supervised/OptimizationStochasticGradientDescent/," [Online]. Available: http://ufldl.stanford.edu/tutorial/supervised/OptimizationStochasticGradientDescent/. [Accessed 10 08 2016].

[21] "support-vector-machines-svm," [Online]. Available: https://documents.software.dell.com/statistics/textbook/support-vector-machines-svm. [Accessed 02 08 2016].

[22] "https://www.statsoft.com/textbook/support-vector-machines," Dell/statsoft, [Online]. Available: https://www.statsoft.com/textbook/support-vector-machines. [Accessed 10 07 2016].

[23] "http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html," [Online]. Available: http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html. [Accessed 10 08 2016].

[24] "http://scikit-learn.org/stable/modules/svm.html," sklearn project, [Online]. Available: http://scikit-learn.org/stable/modules/svm.html. [Accessed 10 07 2016].

[25] "http://www.numpy.org/," [Online]. Available: http://www.numpy.org/. [Accessed 10 07 2016].

[26] "http://pandas.pydata.org/," [Online]. Available: http://pandas.pydata.org/pandas-docs/version/0.18.1/overview.html. [Accessed 10 07 2016].

[27] "https://www.scipy.org/," [Online]. Available: https://www.scipy.org/getting-started.html. [Accessed 10 07 2016].

[28] "http://scikit-learn.org/," Open-Source, [Online]. Available: http://scikit-learn.org/stable/. [Accessed 10 07 2016].

[29] J. D. Hunter, "http://matplotlib.org/," [Online]. Available: http://matplotlib.org/. [Accessed 10 07 2016].

[30] "http://www.tweepy.org/," [Online]. Available: http://tweepy.readthedocs.io/en/v3.5.0/getting_started.html. [Accessed 10 07 2016].

[31] "https://api.mongodb.com/python/current/," MongoDB, [Online]. Available: https://api.mongodb.com/python/current/. [Accessed 10 07 2016].

[32] "Bootstrap (front-end_framework)," [Online]. Available: https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework). [Accessed 15 07 2016].

[33] "AngularJS," [Online]. Available: https://en.wikipedia.org/wiki/AngularJS. [Accessed 15 07 2016].

[34] "Bottle (web_framework)," [Online]. Available: https://en.wikipedia.org/wiki/Bottle_(web_framework). [Accessed 15 07 2016].

[35] "https://dev.twitter.com/," Twitter, [Online]. Available: https://dev.twitter.com/rest/public/search. [Accessed 25 05 2016].

[36] Harrison, "https://pythonprogramming.net/," SentDex, [Online]. Available: https://pythonprogramming.net/tokenizing-words-sentences-nltk-tutorial/. [Accessed 12 06 2016].

[37] "http://statsmodels.sourceforge.net/," statsmodels, [Online]. Available: http://statsmodels.sourceforge.net/devel/example_formulas.html. [Accessed 15 06 2016].