Predicting Political Partisanship on Social Media

Machine Learning | Final Project Report

MSc Data Science & Business Analytics – ESSEC Business School | CentraleSupélec

Rafaëlle Aygalenq B00724587 | Sarah Lina Hammoutene B00712035 | Dóra Linda Kocsis B00714326 | Noémie Quéré B00719656

ABSTRACT

The broad appropriation of social media as a vector of political communication during the past few years opened the door to new opportunities for the continuous screening of active political representatives' opinions. Additionally, the shift of usage in social media platforms such as Twitter and Facebook towards a - sometimes unique - information source, can allow biased messages to be considered as objective truth-holders by a number of users. In this article we portray a few strategies to foresee the bias of political representatives' social media activities in light of the substance of their message. Using a dataset of 5,000 messages, we could classify message substance with a 66% accuracy using a Naive Bayes model along with features selection and predict partisan bias with known message substance with an accuracy of 80%. After trying different approaches, Naives Bayes and XGBoost both gave acceptable results for message classification. Computing power capacity was one of the limits of our study, whence XGBoost was dropped from modeling. Applying Semantic Orientation (SO) to determine the orientation of tweets helped us create a new feature that later showed a strong importance in the results. However, due to a lack of data to accurately predict the number of classes originally considered, we ran into several problems during the modelisation. We thus decided to conclude with limits that arise using our methodology, coupled with a few higher-level observations.

MACHINE LEARNING CONCEPTS

Natural Language Processing, Logistic Regression, Support Vector Machines, Naive Bayes, Random Forests, Neural Network, Boosting, Maximum Entropy

KEYWORDS

text mining, sentiment analysis, social media, politics, partisan bias, classification, machine learning models, Natural Language Processing

1 INTRODUCTION | MOTIVATION

During a time when social media constitutes a key source of news outlet - about 59%[1] of Americans prefer to read news online - it is gaining increasing importance to study how individual influencers use these channels to weigh in on the way we form our opinions. Social media in fact play a key role in transforming political dialogue across the United States.

The network surrounding tweets with a political message substance shows a highly partisan structure with low connectivity among users from opposing sides (Conover et al., (2011) [1]).

This segregated network structure raises concerns primarily for those 59% of individuals consuming their news online. Via their choices of news outlets or influencers, their opinions tend to diverge into further extremes. The time they are exposed to homogeneous viewpoints and the decrease of a plausible opposition keep reinforcing their convictions and in turn deteriorate democracy (Sunstein (2002, 2007) [2][3]).

In this study, we examine whether different message substances of social media posts from US legislators convey partisan bias or not. More specifically, we decided to apply a two-levered classification model. We aim to classify first the legislators' social media posts into 9 message classes using text mining and sentiment analysis techniques. Second, we would like to see if this can further reinforce the existence of bias based on the first classification.

The results of this study could provide inference about the social media behavior of the two opposing sides, and whether gender or the social media platform used (Facebook or Twitter) play any significance in political polarization.

It is important to highlight, that our preliminary data analysis showed, that Democrat legislators tend to express partisan views more often. However, the interpretation of the results is of great importance in this case. The database contains posts from 2015, a year when Republicans gained control of the

[1]Source: http://www.journalism.org/2016/07/07/the-modern-news-consumer/pj_2016-07-07_modern-news-consumer_1-02/

House of Congress, giving their opposition more reason to opinionate partisan messages on their social media outlets.

The following section aims at explaining the structure of our optimization problem more in depth. In section 3, we examine existing research in place specifically focusing on text mining related to social media posts conveying political messages. Then, section 4 is dedicated to describing the methodology in place. Finally, Section 5 and 6 give an overall overview on model evaluation techniques and conclusions.

2 PROBLEM DEFINITION

Given the objective of this project is a classification task, the optimization problem in place will aim to produce a classification function, that returns the correlation between a feature (D) and a class (C).

This classifier is first trained using a training sample and tested on an unseen sample after validation allowed us to optimize model parameters. Testing permits to calculate the prediction error and compare the capacity of the different models we applied. We decided to take 80% of the dataset as the training and 20% as the testing sample. Because of the small size of our dataset, the validation sample has been removed and replaced by cross-validation. The process consists of splitting the training sample in K folds and for each fold, we estimate the model on K − 1 folds using the last fold for the validation step. The number of the folds (K) was set by testing different values and looking at the variance of the prediction error.

When it comes to natural language processing such a classifier is expected to perform fundamentally three types of tasks: topic classification, spam filtering and sentiment analysis. The primary scope of our project focuses on the first capability while our secondary aim is more connected to the sentiment analysis aspect. The purpose of this usage is to establish the subjective value of a text, in our case whether a tweet contains partisan bias or not. Classification accuracy, however, tends to underperform given the subtleties of human language (e.g.: irony, context interpretation, slang, cultural differences). For this reason, we decided to allow other attributes such as party affiliation, gender, position or audience to influence the secondary classification.

The following operations need to be performed upstream given any text classification task:

- **tokenization:** the process of breaking up sentences into smaller pieces
- **stop word removal:** filtering out commonly used words (e.g.: the, a, and, in) before indexing the entries by their frequencies
- **word normalization or stemming:** process of obtaining the base / stem form of a word by reducing any affixes.
- **bag-of-words model**: the process of segmenting sentences into words building a unique dictionary for the dataset in place.

The obtained dictionary constitutes as a basis for our feature vectors, computing the Term Frequency-Inverse Document Frequency (Tf-idf) for each word. Additionally, the Semantic Orientation (SO) was added in every case to determine the relative positivity or negativity of the words. The process is explained in further details in section 4.3.

The project then aims to maximize the classification accuracy (that is, the number of correct assessments in relation to the number of all assessments) of the models which is defined by:

$$Accuracy = \frac{(TN + TP)}{(TN + TP + FN + FP)}$$

In addition, to determine which model is the best, other metrics are used such as the precision (positive predictive values). This metric considers the number of correct assessments in relation to the number of correct assessments:

$$Precision = \frac{(TP)}{(TP + FP)}$$

And the recall, the true positive rate or sensitivity (the number of correct assessments vs the number of training assessments):

$$Recall = \frac{(TP)}{(TP + FN)}$$

Under these cases, we consider a classification “true positive” (TP) when the test makes a positive prediction, and the subject has a positive result under the gold standard, and "false positive" (FP) is the event that the test makes a positive prediction, and the subject has a negative result under the model constraints. Our final tool of model evaluation was the F-score:

$$F = \frac{2\ x\ Precision\ x\ Recall}{(Precision + Recall)}$$

3 RELATED WORK

Text pre-processing techniques have been widely implemented in the context of social media posts as well. Gokulakrishnan et al. (2012) [4] proceeded in their study with 8 different levels: emoticon replacement, uppercase identification, lowercasing, URL extraction, detection of pointers (usernames and hashtags), identification of punctuations, removal of stop

words, removal of query terms, compression of words and removal of skewness in dataset.

While Calvo et al. (2012) [5] studied the method used to implement categorical models for pseudo-documents. They converted these text fragments to vector space using Term Frequency / Inverse Document Frequency (tf-idf). A dimensionality reduction technique was then applied using amongst others, Latent Semantic Analysis (LSA), a method based on Singular Value Decomposition.

A study, fundamentally similar to this one – Patodkar et al. (2016) [6] – used a multinomial Naïve Bayes classifier to build their sentiment detecting algorithm. They deiced to train two Bayes classifiers, using different features: presence of n-grams and part-of-speech distribution information. Where essentially, the n-gram based classifier checks for its presence in the post and uses it as a binary feature. While, the other classifier estimates the probability of POS-tags presence within different sets of texts and uses it to calculate posterior probability.

Even though sentiment detection techniques have rather been applied on subjects related to reviews of products or services, there is a wide range of its applicability related to blog entries, comments, forum postings or tweets. A study by Boiy and Moens [7] considers this technique as a classification problem, whereby they carry out feature selection on multilingual digital contents. Unfortunately, there has only been slight progress measuring bias such digital collections.

These techniques are commonly divided into two major groups: lexicon-based methods and machine-learning methods. While the prior rely on a sentiment lexicon – a collection of pre-collected sentiment terms-, the latter use syntactic and linguistic features in compiling the final algorithm. The different approaches when it comes to detecting political opinions were studied and summarized by Maynard, Funk (2011) [8]. In fact, they made use of a variety of NLP techniques to extract opinions form a pre-election Twitter corpus focused on the UK, whereby they were able to identify opinion holders via mining for opinionated, positive or negative statements.

Finally, Abdullah et al. (2017) [9] provide a comprehensive overview on the most suitable machine learning techniques and sentiment analysis approaches on social media data. Specifically, they consider a hybrid approach for politics related Twitter posts, namely a combination of lexicon of dictionary and SVM.

4 METHODOLOGY

4.1 DATASET

Our primary dataset comes from Crowdflower's Data For Everyone Library. Political Social Media Posts provides 5,000 Twitter and Facebook posts collected from US Senators, Congressmen and Congresswomen, along with human judgments on the intended audience, existence of partisan bias and 9 different types of message substance. The latter constitutes as our primary classification target, ranging from informational, support, media appearance to attack and policy related messages.

Our secondary dataset includes all the current and historic legislators for both the Congress and the Senate. The dataset is part of GovTrack.us [2] congress-legislators project, which provides a free repository on US federal legislative information. Each legislator is recorded with id's relating the record to other databases, name information, biographical information (`birthday`, `gender`), party affiliation and terms served in Congress. We decided to only keep the name, state, gender and party affiliation from this dataset before merging it with our primary source, further described in section 4.2.1.

4.2 DATA PRE-PROCESSING

4.2.1 Merging Data Sources

To enrich our dataset, we included two new features extracted from our secondary data source: party affiliation and gender. This way, we hope to uncover new correlations between the features and expand the scope of our research.

We chose to aggregate our two datasets by matching each observation according to the combination of the two variables `last_name` and `state` - our composite key - as the information was present in both sources. In the original dataset, the author's last name and state were extracted from the label variable. After applying multiple splits, 6 new features were created, including 4 `author_name` and 1 `author_state` variable. In the second dataset, the `last_name` feature was already present but was split into two, `last_name1` and `last_name2`, to handle cases of individuals with multiple last names. Politicians with three last names, special characters or different state names were treated manually.

The aggregation was done on observations with an exact matching of both states and the iterated list of last names. Additional variables created for the name transformation were dropped.

4.2.2 Feature Engineering

As a part of feature engineering on the existing attributes of the dataset, we created the feature: text_length, displaying the number of characters per tweet. Furthermore, by extracting

[2] Source: https://www.govtrack.us/about-our-data

information from the label attribute, we recovered the author's name, position (Representative, Senator) and the state serve. The following variables were transformed to binary scalars: audience, bias, source, position, gender were transformed to numeric levels, taking value 0 or 1. The feature `author_party` was transformed into numeric scalar taking value 0 for Independent, 1 for Democrat and 2 for Republican. Finally, redundant features, that always take the same value regardless of the observation (e.g.: as_`golden`), empty columns (e.g.: `orig_golden`), attributes containing identifiers (e.g.: `_unit_id`) or related to time (e.g.: `_last_judgement_at`) were dropped from the analysis.

4.3 TEXT MINING

The goal of the text processing performed here is to create a Bag-of-Words model. As this model takes word in the dataset as features, it is essential to transform the text upstream to work with the barest corpus possible.

4.3.1 Cleaning

Firstly, all numbers, punctuation, special characters etc. were removed so that only text would remain in the message. We then looked for repetitions of characters and replaced them with the character itself. We also replaced tweet-specific characters (@, #, URL) by their corresponding full name. Finally, we changed the text to be lowercase at all time.

4.3.2 Text Processing

We started by tokenizing the remaining text by applying the tokenize_word function of the `nltk library`. Tokenization is the concept of splitting up a string into different substrings, here words, called tokens. This step is essential for our modeling since each token will later be used as a feature in our Bag-of-words.

Additionally, we stemmed each of the words present in the corpus, again with the `nltk library`. **Stemming** refers to the linguistic technique of reducing a word back to its root, or stem. This process allows us to aggregate words that have similar stem, and thus meaning, in one category. For instance, 'government', 'governance' or 'governor' all became 'govern' after the stemming process. This aggregation allows us to obtain more accurate frequency for words with similar meaning.

The next step was to remove all stopwords, once again thanks to an integrated `nltk` function. It removed all generic words that were not essential for the message substance by iterating through a predefined stopword list. Finally, we de-tokenized our cleaned data frame to vectorize it for the modeling.

4.3.3 Bag-of words model

In a Bag-of-words model, each of the word is considered as a feature. The frequency of the words in each message is computed to obtain a frequency matrix. In our model, we chose to construct our feature vectors using tf-idf (Term Frequency-Inverse Document Frequency) instead of the frequency. Tf-idf allows to compute the relevance of a word within the corpus as its value increases with the frequency of a word in a document, but is offset with the frequency of the word in the corpus. Thus, for a word i in document j:

$$w_{i,j} = tf_{i,j} + log(\frac{N}{df_i})$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of tweets containing i
N = total number of tweets

That way, only the frequent and distinctive words stay as a marker. After transforming our data into a BoW, we normalized it so it adds up to one.

4.3.4 Semantic orientation of a word

We decided to add one more feature by calculating the Semantic Orientation (SO) of the words, e.g. whether they were positive or negative. To implement SO, we need to measure how close a word is to being positive or negative. This can be done with Pointwise Mutual Information (PMI), calculated as follow:

$$PMI(t_1, t_2) = \log(\frac{P(t_1 t_2)}{P(t_1)P(t_2)})$$

where t_1, t_2 represent terms, $P(t)$ is the probability of appearance of the term t and $P(t_1 \wedge t_2)$ the probability that the two terms are observed together which are defined such that:

$$P(t) = \frac{DF(t)}{|t|} \text{ and } P(t_1 \wedge t_2) = \frac{DF(t_1 \wedge t_2)}{|D|}$$

Then, let V^+ (respectively V^-) be a predefined list of positive (respectively negative) words, we defined the Semantic Orientation of a term t as:

$$SO(t) = \sum_{t' \in V^+} PMI(t, t') - \sum_{t' \in V^-} PMI(t, t')$$

4.4 MODEL TUNING & RESULTS

4.4.1 Overview of deployed models

Logistic Regression

Logistic regression belongs to the same family as linear regression, or the generalized linear models. In both cases its main aim is to link an event to a linear combination of features. However, for logistic regression we assume that the target

variable is following a Binomial distribution. For the first part of this project, to predict the message substance, we will use a multinomial logistic regression due to the multi-class aspect of our problem.

Support Vector Machines (SVM)

Support Vector Machines are based on finding the optimal margin hyperplanes, which, if possible, classifies or separates the data correctly while keeping a sufficiently large distance from all other observations. This objective can be achieved either by:

1. Defining the hyperplane as a solution of a constrained optimization problem whose objective function is only expressed through vector scalar products. Here, the number of active constraints or support vectors is controlling the model's complexity.
2. Introducing a kernel function in the scalar product inducing implicitly a nonlinear transformation of the data into an intermediate space of higher dimension. These kernels (specifically adapted to a given problem) are giving more flexibility to the model to adapt to diverse situations.

Naive Bayes Classifier

Naive Bayes classifiers are a family of probabilistic classifiers based on Bayes' theorem with an assumption of independence between predicting features, that means that this classifier assumes that the presence of one feature in each class is unrelated to the presence of any other feature. It is a model, that is easy to implement. This model usually works well on large datasets. One major drawback of this method is the independence assumption which usually does not hold in reality.

Random Forests (RF)

Random Forest is an ensemble machine learning method, that works by developing a multitude of decision trees. The aim is to make decision trees more independent by adding randomness in features choice. For each tree, it firstly draws a bootstrap sample, obtained by repeatedly sampling observations from the original sample with replacement. Then, the tree is estimated on that sample by applying feature randomization. This means, that searching for the optimal node is preceded by a random sampling of a subset of predictors. Finally, the result may either be an average or weighted average of all the terminal nodes that are reached, or, in the case of categorical variables, a voting majority.

Neural Networks (NNET)

A neural network is an association into a graph, complex, of artificial neurons. Neural networks are distinguished by their architecture (layers, complete), their level of complexity (number of neurons) and their activation function. An artificial neuron is a mathematical function conceived as a model of biological neurons. It is characterized by an internal state, input signals and an activation function which is operating a transformation of an affine combination of input signals. This affine combination is defined by a weight vector associated to the neuron and for which the values are estimating during the training part. For this project, we used a multilayer perceptron. This is a network composed of successive layers which are ensembles of neurons with no connection between them.

Boosting techniques

Boosting principle is to construct a family of models which are then aggregate together with a weighted average of estimations or a vote but it is different from other ensemble method on the way of constructing the family. In this case, each model is an adaptive version of the previous one by giving more weight to wrongly predicted observations. Intuitively, this algorithm focusses on observations which are more difficult to adjust, in other words, the idea is to improve capacities of weak learners. In this project we implemented two different versions of boosting, the first one is the Gradient Boosting (GBM), an iterative algorithm based on the gradient descent, and the XGBoost algorithm (XGB), which is a more regularized version of gradient boosting to control overfitting and it is supposed to give better results.

K Nearest Neighbors (KNN)

KNN is a simple algorithm, which computes a similarity measure across observations, such as Euclidean distance, and makes predictions by searching in the dataset for the K most similar cases, called the neighbors. The final output is obtained by taking the most common value among these K instances.

4.4.2 Message Substance Modelling

The first part of the project consists in modelling the message substance of social media posts. This is a multi-classification problem with 9 different classes that we tried to address using several approaches.

Firstly, we started by predicting all classes simultaneously with a direct application on the whole dataset of the algorithms described previously. This method gave poor results, which could have been foreseen because of the small amount of data per class to train the models. As the following figure shows, the best accuracy score that we obtained was around 40%.

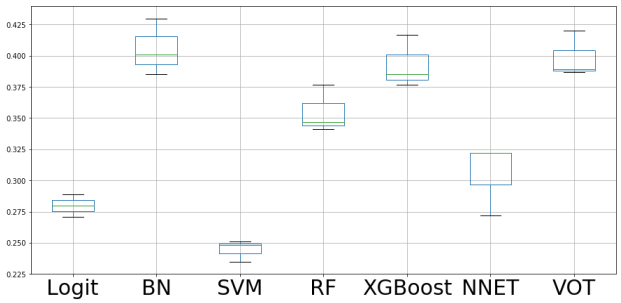

Figure 1. Model Performance Predicting 9 classes

That is why we decided to implement the One vs All method which consists in considering one class at the time by fixing it to 1 and the rest to 0, and repeat it for all the different classes. For this technique we also tried different models and most of the time three of them stand out: Naive Bayes, Random Forest and XGBoost. We obtained a very high accuracy with this approach as shown in the figure below that represents the results obtained for the category 7 (Personal).

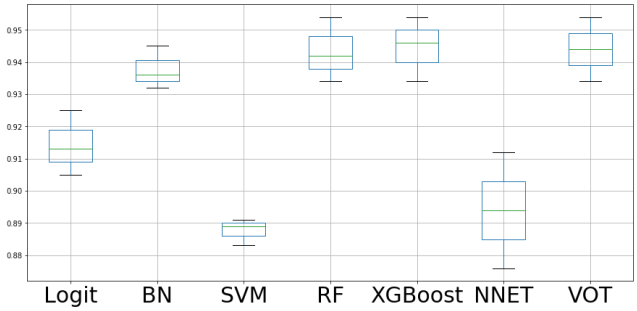

Figure 2. Model Performance Using One vs All

The high results obtained can be attributed to the fact, that the models were predicting a 0 value in most cases, caused by the lack of data in some classes. Indeed, sometimes only 125 observations were observed for a class on a dataset of 5000. This problem became important when we tried to predict the whole dataset to help model the bias because more than 4000 tweets remained unclassified.

To tackle this issue, we chose to merge the categories into 4, and then 3 groups to get bigger samples for each group. After applying all models, we obtained accuracy results between 30% and 45% which is not considered as sufficient. These poor results are explained by the fact that the categories grouped together were not truly similar, thus models could not train properly.

Our final attempt was to select only the top 3 categories and to use the same process as before. We first tried the One vs All approach by predicting one out of three classes at the time. As previously, by doing so the accuracy obtained for each category was high but when merged with the predictions, more than 4000 tweets remained unclassified. Then, we performed a classification for the 3 categories using the models presented previously.

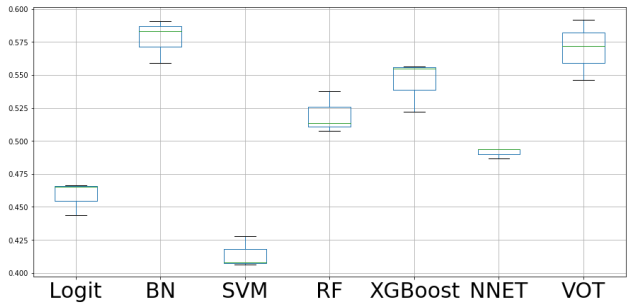

Figure 3. Model Performance Predicting 3 classes

We can notice that Naive Bayes classifier for multivariate Bernoulli model gives the best results reaching almost 60% of accuracy. Moreover, we tested a Deep Learning technique - Convolutional Neural Network - using the Keras library, but results were not better. This was due to the aforementioned reasons as well as the limited computation capacity of our computers.

Since we are dealing with a high dimensionality problem (10343 features and 3502 messages after keeping the top 3 categories), one way to improve the accuracy of our model is to perform dimensionality reduction and features selection.

Dimensionality Reduction:

Given that we are dealing with a sparse matrix, PCA cannot be used since the algorithm will not converge. However, there exists an adapted version of SVD called TruncatedSVD that works on term count/tf-idf matrices. Unfortunately, due to our limited computation capacity we could not use it.

Feature Selection:

To keep the most significant features according to the highest scores, we used SelectKBest. To determine the number of features that needed to be kept, we performed the selection for several values. Then, we computed the accuracy with the subset obtained for each. To create the classification, we decided to use the models that gave the best accuracy for the top 3 categories: Naive Bayes and XGBoost.

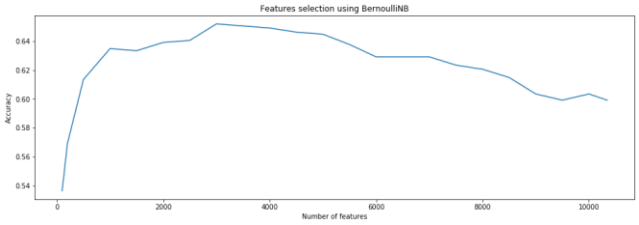

Figure 4. Feature Selection Using Naïve Bayes

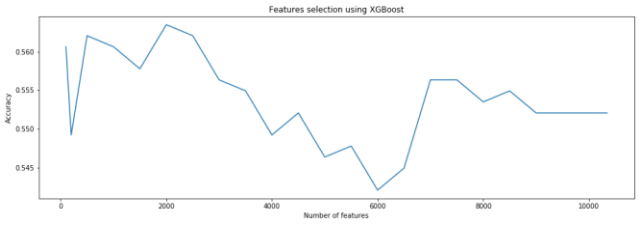

Figure 5. Feature Selection Using XGBoost

The results given by the features selection is interesting: as we can see the accuracy of the two models varies with the number of features selected. However, considering more features doesn't improve necessarily the models. Both Naive Bayes and XGboost reach their top accuracy at around 2500 features, with a better one for Naive Bayes (66%).

The reason why the accuracy decreases when more features are kept is overfitting: with a lot of features, the train dataset tends to be overfitted and thus give poor results on the test dataset.

Note: we can also improve the accuracy of our model by performing tuning parameters. However, it is time consuming and not feasible given the computation capacity that we have.

4.4.3 Bias Modelling

The second part of this project is about identifying whether a post is biased so it can be treated as a binary classification problem.

As in the previous section, different models have been implemented and as it can be seen on the Figure below, three of them stand out from the other: the two boosting methods and the random forests algorithm. This Figure represents accuracy distribution for each model and has been obtained by implementing a Monte Carlo cross validation, 10 different train and test samples have been randomly sampled and the accuracy for each method and each test sample has been computed.

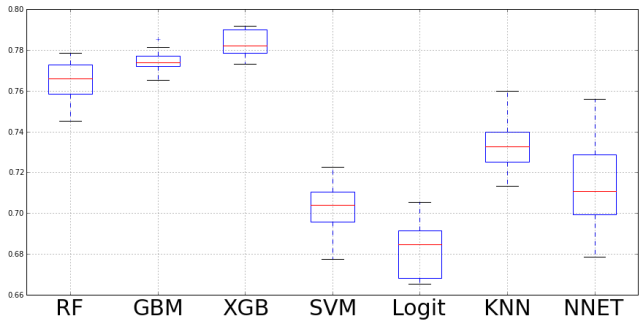

Figure 6. Model Performance Predicting Partisan Bias

We also evaluated every model by computing the ROC (Receiver Operating Characteristic) curve which highlights the true positive rate against the false positive rate. This curve leads to the AUC (Area Under Curve) score giving an idea of the model's performance. These curves are drawn on the Figure below from which we can reach the same conclusion, RF, GBM and XGBoost seem to be the best models.

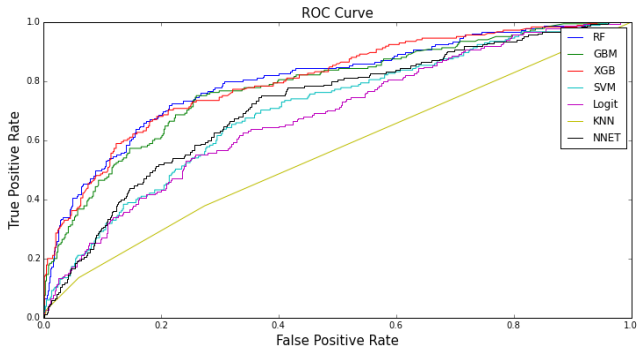

Figure 7. ROC Curve of Selected Models

According to these results we decided to implement another ensemble model which is the Voting Classifier that combines conceptually different estimators and use a majority vote, hard voting, or the average predicted probabilities, soft voting, to predict the class. We used the three best models as estimators for the voting classifier and tested both hard and soft voting for predicting. Finally, this model has given the best results, which are always close to 80% accuracy.

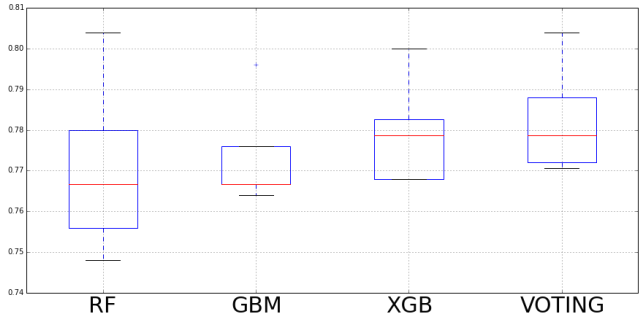

Figure 8. Model Performance Using Voting Classifier

5 LIMITS AND CONCLUSION

In this project we have demonstrated that politically-active social media users generate text information through their posts that can be used to effectively predict the political partisanship of large numbers of individuals. We detailed several approaches to tackle this issue that we divided into two different tasks: predict the message substance first, and the bias in a second step.

We showed that message substance is extremely important to predict bias, as it was the most important feature for every model we implemented when it came to predict partisanship. Therefore, accurately predicting message substance is essential, even though it is by far the most difficult task since it relates to a multi-classification problem. With higher computational power, predicting bias using only text from social media posts could have been possible, also following our initial goal. Unfortunately, we ran into two limits, that we detail further below.

The first problem we faced was the size of the dataset. Indeed, it did not contain enough observations per category to run the multi-classification part. More data would have helped to better train models for all categories and thus would have yield better end results.

Secondly, we were also limited by the computation capacity of our computers. We tried to use the Keras library to implement a Convolutional Neural Network, a method that requires an important computing power, which we do not have. However, results obtained with this technique were promising.

REFERENCES

[1] M. D. Conover, J. Ratkiewicz, M. Francisco, B. Gonc¸alves, A. Flammini, F. Menczer (2011). Political Polarization on Twitter. Association for the Advancement of Artificial Intelligence. Pp.89-96.

[2] Sunstein, C. (2002). The law of group polarization. Journal of Political Philosophy 10(2):175–195.

[3] Sunstein, C. R. (2007). Republic.com 2.0. Princeton University Press.

[4] Gokulakrishnan, B., Privanthan, P., Ragavan, T., Prasath, N. and Perera, A. (2012). Opinion mining and sentiment analysis on a Twitter data stream. Advances in ICT for Emerging Regions (ICTer), 2012 International Conference on, pp.182 - 188

[5] Calvo, R. and Mac Kim, S. (2012). EMOTIONS IN TEXT: DIMENSIONAL AND CATEGORICAL MODELS. Computational Intelligence, 29(3), pp.527-543.

[6] Patodkar, V. and I.R, S. (2016). Twitter as a Corpus for Sentiment Analysis and Opinion Mining. IJARCCE, 5(12), pp.320-322.

[7] Boiy, E., Moens, M.F.: A machine learning approach to sentiment analysis in multilingual web texts. (2009). pp.526-558.

[8] Diana Maynard and Adam Funk. (2011). Automatic detection of political opinions in Tweets. pp.1-12.

[9] Nur Atiqah Sia Abdullah, Nurul Iman Shaari and Abd Rasid Adb Rahman. (2017). Journal of Engineering and Applied Sciences 12 (3). pp.462-467.