4/16/2019

Project Report P12

Canada 2015 General Election Analysis

Gurjot Singh (B00811724)
Bhavneet Kaur Sachdeva (B00809769)

Abstract

Elections have always been a major event which involves every person of the country. Everyone is always anxious to know who the next leader will be and what changes they will bring. The competing leaders' campaign about their intentions for the upcoming term and people vote and choose their next leader. But there are also a lot of sentiments and opinions of people on social media about all the leaders and party. Hence, in this project, we aimed to mine the sentiments of the people using the tweets and then use those to find the accuracy of our model and compare it with different models.

Introduction

Predicting the elections have always been one of the hot topics in the data mining field. There are different ways to predict the elections. One of the methods is using sentiment analysis of tweets to predict the election results. Sentiment analysis is the method of extracting the tweets and performing text processing on the tweets to convert the text into a machine-readable format using which we can predict the outcome or perform analysis on the tweets.

In this project, we have used two datasets. The first dataset was obtained from Elections Canada - Elections Canada is the independent, non-partisan agency responsible for conducting federal elections and referendums [1]. This dataset contained the election information like the province, district, polling results, candidate's and party data and the other dataset used was the twitter tweets.

After creating the model and training the model, three algorithms were applied to compare the accuracy of the results. The three algorithms used were Naïve Bayes, Random Forest and Decision tree classifier.

Related Work

Elections have been already used for prediction using user activity on Twitter. The research paper [12] provides the comprehensive use of Twitter-based election forecasting in the Indonesia's presidential elections of2015 using the Twitter-predictor of traditional polls at the national level. Also, the research paper [13] used a tidy approach of a massive dataset collected on Twitter using 3.5 million tweets. A sentiment analysis was performed regarding each political leader.

Data Set Used

Two datasets have been used in the development of the project.

1) Canada Elections 2015 Official Voting Results: The Raw Dataset of the results of Canada Elections 2015 was used in the first analysis of the algorithms [2]. The data set consists of the results of all the provinces of Canada in detail.
2) Twitter Tweets: The twitter tweets related to Canada elections in 2015 were fetched using Twitter API [3]. There were more than 7 lakh tweets and the analysis were done using them.

Technologies Used

The technologies used are as follows:

- **Language Used**: The project has been developed in Python Programming Language.

- **Tool Used**: Jupyter Notebook and Spyder was used.
- **Libraries Used**:
    - **Pandas:** Pandas Library was used to fetch the JSON and CSV files.
    - **Google-translate**: Google-translate library was used in converting French text into English.
    - **Scikit-learn**: This library was used to apply machine learning algorithms.
    - **Numpy**: Numpy was used in array processing of the text.

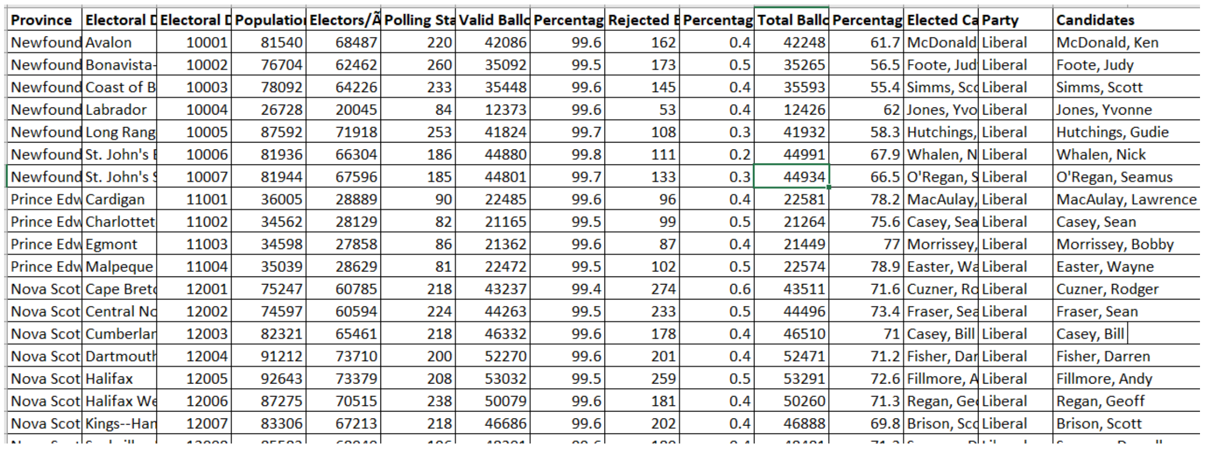
Data Cleaning

"No great marketing decisions have ever been made on qualitative data." – John Sculley [4]. We have performed data cleaning to ensure we have the best dataset available to perform the analysis. We have performed different types of data cleaning on both the datasets.

Data Set 1 – Election Voting Results

The dataset fetched from the Election voting results [2] contained the Elected Candidate/Candidat élu column had both French and English, hence we had to segregate it to obtain them separately. To identify the party name, we converted the French to English using the Google-translate library [5] in python.

The data set was loaded using pandas and using data frames the data cleaning was performed on the dataset.



Province	Electoral D	Electoral D	Populatio	Electors/Ã	Polling Sta	Valid Ballo	Percentag	Rejected E	Percentag	Total Ballo	Percentag	Elected Ca	Party	Candidates
Newfound	Avalon	10001	81540	68487	220	42086	99.6	162	0.4	42248	61.7	McDonald	Liberal	McDonald, Ken
Newfound	Bonavista-	10002	76704	62462	260	35092	99.5	173	0.5	35265	56.5	Foote, Jud	Liberal	Foote, Judy
Newfound	Coast of B	10003	78092	64226	233	35448	99.6	145	0.4	35593	55.4	Simms, Sco	Liberal	Simms, Scott
Newfound	Labrador	10004	26728	20045	84	12373	99.6	53	0.4	12426	62	Jones, Yvo	Liberal	Jones, Yvonne
Newfound	Long Rang	10005	87592	71918	253	41824	99.7	108	0.3	41932	58.3	Hutchings,	Liberal	Hutchings, Gudie
Newfound	St. John's E	10006	81936	66304	186	44880	99.8	111	0.2	44991	67.9	Whalen, N	Liberal	Whalen, Nick
Newfound	St. John's S	10007	81944	67596	185	44801	99.7	133	0.3	44934	66.5	O'Regan, S	Liberal	O'Regan, Seamus
Prince Edw	Cardigan	11001	36005	28889	90	22485	99.6	96	0.4	22581	78.2	MacAulay,	Liberal	MacAulay, Lawrence
Prince Edw	Charlottet	11002	34562	28129	82	21165	99.5	99	0.5	21264	75.6	Casey, Sea	Liberal	Casey, Sean
Prince Edw	Egmont	11003	34598	27858	86	21362	99.6	87	0.4	21449	77	Morrissey,	Liberal	Morrissey, Bobby
Prince Edw	Malpeque	11004	35039	28629	81	22472	99.5	102	0.5	22574	78.9	Easter, Wa	Liberal	Easter, Wayne
Nova Scot	Cape Bret	12001	75247	60785	218	43237	99.4	274	0.6	43511	71.6	Cuzner, Ro	Liberal	Cuzner, Rodger
Nova Scot	Central No	12002	74597	60594	224	44263	99.5	233	0.5	44496	73.4	Fraser, Sea	Liberal	Fraser, Sean
Nova Scot	Cumberlar	12003	82321	65461	218	46332	99.6	178	0.4	46510	71	Casey, Bill	Liberal	Casey, Bill
Nova Scot	Dartmouth	12004	91212	73710	200	52270	99.6	201	0.4	52471	71.2	Fisher, Dar	Liberal	Fisher, Darren
Nova Scot	Halifax	12005	92643	73379	208	53032	99.5	259	0.5	53291	72.6	Fillmore, A	Liberal	Fillmore, Andy
Nova Scot	Halifax We	12006	87275	70515	238	50079	99.6	181	0.4	50260	71.3	Regan, Ge	Liberal	Regan, Geoff
Nova Scot	Kings--Har	12007	83306	67213	218	46686	99.6	202	0.4	46888	69.8	Brison, Sco	Liberal	Brison, Scott

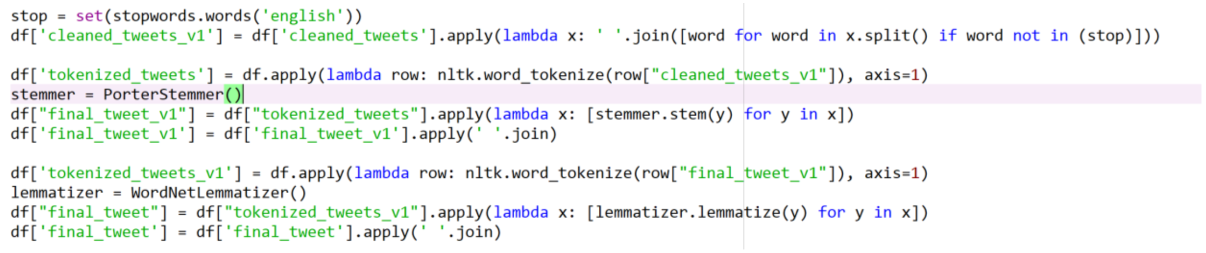
Fig 1: Sample Final Cleaned Dataset

Data Set 2 – Twitter Tweets of Election 2015

After extracting the twitter tweets, cleaning of the tweets was to be performed for the dataset to be used in the text processing. Hence, first the retweets were removed from the dataset as it contained a lot of retweets which would hamper the results. Then, the special characters were removed except for the hash as it was to be used to identify the hashtags. Also, a set of stop words was identified from the nltk corpus and they were used to remove the stop words from the dataset as they are repeated a lot of times and hence, they unnecessarily affect the frequencies and counts of the words.

Thereafter, stemming of the dataset was done. To perform stemming, first a sentence had to be broken down into tokens. Stemming was performed to extract the root words so that words written in different tenses is also assigned the same vector. This increases the accuracy of the prediction as it extracts the root of the word.

Another step to increase the accuracy further was the lemmatization of the data. Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item. Lemmatization is like stemming but it brings context to the words. So, it links words with similar meaning to one word. [7] Hence, after stemming, lemmatization was performed to further process the text to identify the context of the words.



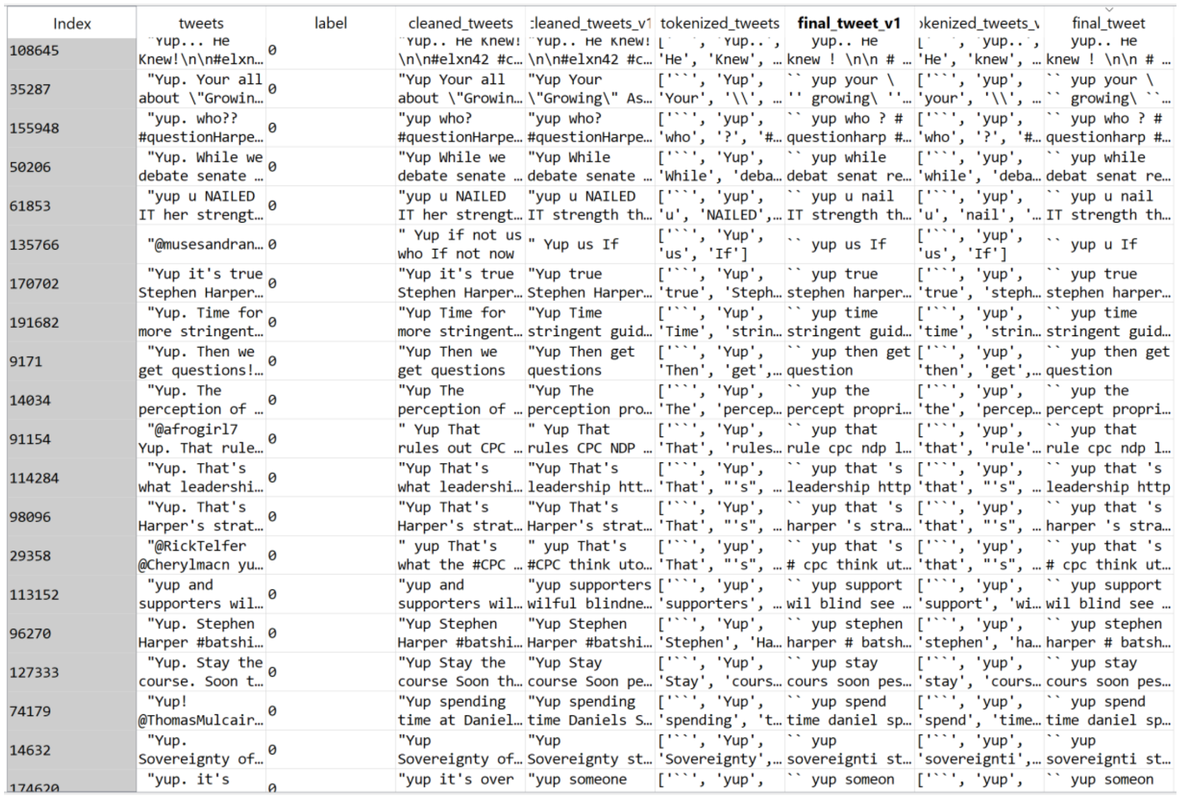
```
stop = set(stopwords.words('english'))
df['cleaned_tweets_v1'] = df['cleaned_tweets'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop)]))

df['tokenized_tweets'] = df.apply(lambda row: nltk.word_tokenize(row["cleaned_tweets_v1"]), axis=1)
stemmer = PorterStemmer()
df["final_tweet_v1"] = df["tokenized_tweets"].apply(lambda x: [stemmer.stem(y) for y in x])
df['final_tweet_v1'] = df['final_tweet_v1'].apply(' '.join)

df['tokenized_tweets_v1'] = df.apply(lambda row: nltk.word_tokenize(row["final_tweet_v1"]), axis=1)
lemmatizer = WordNetLemmatizer()
df["final_tweet"] = df["tokenized_tweets_v1"].apply(lambda x: [lemmatizer.lemmatize(y) for y in x])
df['final_tweet'] = df['final_tweet'].apply(' '.join)
```

Fig 2: Python Data Cleaning Snippet

The final dataset obtained after following the above steps was:



Index	tweets	label	cleaned_tweets	:leaned_tweets_v1	tokenized_tweets	**final_tweet_v1**	›kenized_tweets_\	final_tweet
108645	"Yup... He Knew!\n\n#elxn…	0	"Yup.. He Knew! \n\n#elxn42 #c…	"Yup.. He Knew! \n\n#elxn42 #c…	['``', 'Yup..', 'He', 'Knew', …	yup.. He knew ! \n\n # …	['``', 'yup..', 'He', 'knew', …	yup.. He knew ! \n\n # …
35287	"Yup. Your all about \"Growin…	0	"Yup Your all about \"Growin…	"Yup Your \"Growing\" As…	['``', 'Yup', 'Your', '\\', …	`` yup your \ '' growing\ ''…	['``', 'yup', 'your', '\\', …	`` yup your \ `` growing\ ``…
155948	"yup. who?? #questionHarpe…	0	"yup who? #questionHarpe…	"yup who? #questionHarpe…	['``', 'yup', 'who', '?', '#…	`` yup who ? # questionharp #…	['``', 'yup', 'who', '?', '#…	`` yup who ? # questionharp #…
50206	"Yup. While we debate senate …	0	"Yup While we debate senate …	"Yup While debate senate …	['``', 'Yup', 'While', 'deba…	`` yup while debat senat re…	['``', 'yup', 'while', 'deba…	`` yup while debat senat re…
61853	"yup u NAILED IT her strengt…	0	"yup u NAILED IT her strengt…	"yup u NAILED IT strength th…	['``', 'yup', 'u', 'NAILED',…	`` yup u nail IT strength th…	['``', 'yup', 'u', 'nail', '…	`` yup u nail IT strength th…
135766	"@musesandran…	0	" Yup if not us who If not now	" Yup us If	['``', 'Yup', 'us', 'If']	`` yup us If	['``', 'yup', 'us', 'If']	`` yup u If
170702	"Yup it's true Stephen Harper…	0	"Yup it's true Stephen Harper…	"Yup true Stephen Harper…	['``', 'Yup', 'true', 'Steph…	`` yup true stephen harper…	['``', 'yup', 'true', 'steph…	`` yup true stephen harper…
191682	"Yup. Time for more stringent…	0	"Yup Time for more stringent…	"Yup Time stringent guid…	['``', 'Yup', 'Time', 'strin…	`` yup time stringent guid…	['``', 'yup', 'time', 'strin…	`` yup time stringent guid…
9171	"Yup. Then we get questions!…	0	"Yup Then we get questions	"Yup Then get questions	['``', 'Yup', 'Then', 'get',…	`` yup then get question	['``', 'yup', 'then', 'get',…	`` yup then get question
14034	"Yup. The perception of …	0	"Yup The perception of …	"Yup The perception pro…	['``', 'Yup', 'The', 'percep…	`` yup the percept propri…	['``', 'yup', 'the', 'percep…	`` yup the percept propri…
91154	"@afrogirl7 Yup. That rule…	0	" Yup That rules out CPC …	" Yup That rules CPC NDP …	['``', 'Yup', 'That', 'rules…	`` yup that rule cpc ndp l…	['``', 'yup', 'that', 'rule'…	`` yup that rule cpc ndp l…
114284	"Yup. That's what leadershi…	0	"Yup That's what leadershi…	"Yup That's leadership htt…	['``', 'Yup', 'That', "'s", …	`` yup that 's leadership http	['``', 'yup', 'that', "'s", …	`` yup that 's leadership http
98096	"Yup. That's Harper's strat…	0	"Yup That's Harper's strat…	"Yup That's Harper's strat…	['``', 'Yup', 'That', "'s", …	`` yup that 's harper 's stra…	['``', 'yup', 'that', "'s", …	`` yup that 's harper 's stra…
29358	"@RickTelfer @Cherylmacn yu…	0	" yup That's what the #CPC …	" yup That's #CPC think uto…	['``', 'yup', 'That', "'s", …	`` yup that 's # cpc think ut…	['``', 'yup', 'that', "'s", …	`` yup that 's # cpc think ut…
113152	"yup and supporters wil…	0	"yup and supporters wil…	"yup supporters wilful blindne…	['``', 'yup', 'supporters', …	`` yup support wil blind see …	['``', 'yup', 'support', 'wi…	`` yup support wil blind see …
96270	"Yup. Stephen Harper #batshi…	0	"Yup Stephen Harper #batshi…	"Yup Stephen Harper #batshi…	['``', 'Yup', 'Stephen', 'Ha…	`` yup stephen harper # batsh…	['``', 'yup', 'stephen', 'ha…	`` yup stephen harper # batsh…
127333	"Yup. Stay the course. Soon t…	0	"Yup Stay the course Soon th…	"Yup Stay course Soon pe…	['``', 'Yup', 'Stay', 'cours…	`` yup stay cours soon pes…	['``', 'yup', 'stay', 'cours…	`` yup stay cours soon pes…
74179	"Yup! @ThomasMulcair…	0	"Yup spending time at Daniel…	"Yup spending time Daniels S…	['``', 'Yup', 'spending', 't…	`` yup spend time daniel sp…	['``', 'yup', 'spend', 'time…	`` yup spend time daniel sp…
14632	"Yup. Sovereignty of…	0	"Yup Sovereignty of…	"Yup Sovereignty st…	['``', 'Yup', 'Sovereignty',…	`` yup sovereignti st…	['``', 'yup', 'sovereignti',…	`` yup sovereignti st…
174620	"yup. it's	0	"yup it's over	"yup someone	['``', 'yup',	`` yup someon	['``', 'yup',	`` yup someon

Fig 3: Sample Final Cleaned Dataset

Data Labeling

To use the data set in the Machine Learning models, we had to setup a classifier for labels. The labels were setup for the dataset 1 as the following:

Party Name	**Labels Assigned**
Liberal	0
NDP- New Democratic Party	1
Bloc Quebecois	2
Conservative	3

For the dataset 2, we used to follow labels:

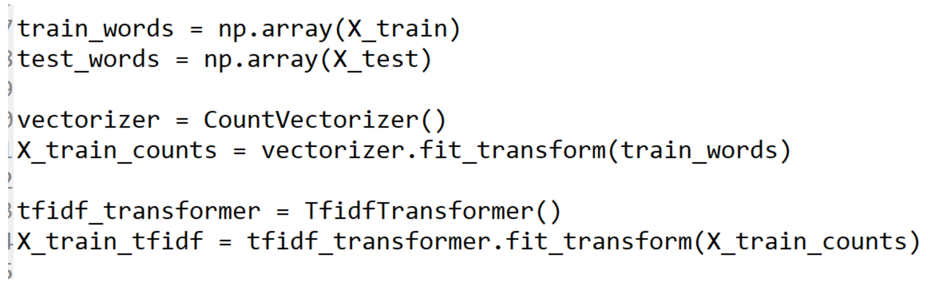
Candidate	Labels Assigned
Justin Trudeau & Liberal Party(lpc)	1
NDP- New Democratic Party	-1
Other Tweets	0

Feature-set Selection and Cross-Validation

Feature selection methods aid you in your mission to create an accurate predictive model. They help you by choosing features that will give you as good or better accuracy whilst requiring less data. Feature selection methods can be used to identify and remove unneeded, irrelevant and redundant attributes from data that do not contribute to the accuracy of a predictive model or may in fact decrease the accuracy of the model. [7]

For the first dataset, the features selected were the province and the district to identify which party wins the election in a district. This will help in deducing a pattern for the districts and thus help us in predicting the winning party. For the second dataset, the features were the tweets. To reach to this point, we split the dataset into training and testing data with a 33% given to the test data set.

After splitting the data, we had to train the model using the training data set. But since we had text in the dataset and for machine learning models, it is desired for the input to be in vector or numbers, we had to convert this text to vectors. To do the conversion, we used CountVectorizer and tfidfvectorizer. CountVectorizer converts a collection of text documents to a matrix of token counts. Term Frequency-Inverse Document Frequency (TF-IDF) is an algorithm to transform text into a meaningful representation of numbers and this technique is widely used to extract features across various NLP applications [6]. These both are feature vectors and can be used independently, but in some cases the count works better and in some cases the frequency works better. Hence, using them both combined could give us a complete set of feature vector which could help us increase our accuracy.



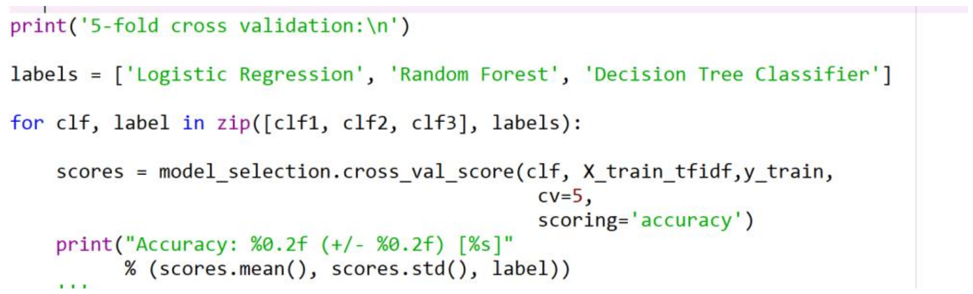
```
train_words = np.array(X_train)
test_words = np.array(X_test)

vectorizer = CountVectorizer()
X_train_counts = vectorizer.fit_transform(train_words)

tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
```

Fig 4: Feature Selection and training the train dataset

After extracting the feature vector, we trained our model by fitting the dataset into these models. Since we had to compare the algorithms, we created three models and fit the same dataset into the three models. This way, all the three models were trained on the same dataset, thus providing consistency while comparing the accuracies of the three models. After training this model using the training dataset, the same steps were repeated on the test data set, and the accuracy was predicted. To further enhance the accuracy, 5-fold cross validation was used for each model. Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. [8] This helps us in achieving higher accuracy as it performs the same process multiple times with different datasets and then provides the accuracy as a mean of all results.



```
print('5-fold cross validation:\n')

labels = ['Logistic Regression', 'Random Forest', 'Decision Tree Classifier']

for clf, label in zip([clf1, clf2, clf3], labels):

    scores = model_selection.cross_val_score(clf, X_train_tfidf,y_train,
                                              cv=5,
                                              scoring='accuracy')
    print("Accuracy: %0.2f (+/- %0.2f) [%s]"
          % (scores.mean(), scores.std(), label))
    ...
```

Fig 5: 5-fold Cross Validation

Algorithm 1 – Decision Tree Classifier

The general motive of using Decision Tree is to create a training model which can be used to predict class or value of target variables by learning decision rules inferred from prior data. A decision tree is a flowchart-like tree structure where an internal node represents feature (or attribute), the branch represents a decision rule, and each leaf node represents the outcome. [9]

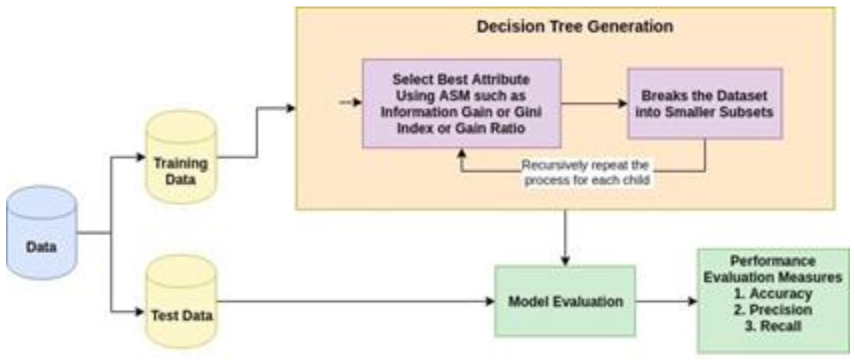
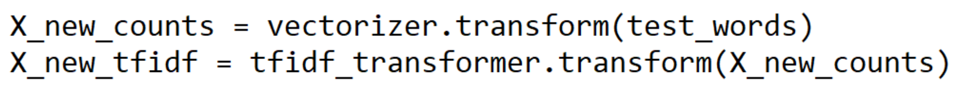

Decision Tree Generation
Select Best Attribute Using ASM such as Information Gain or Gini Index or Gain Ratio
Breaks the Dataset into Smaller Subsets
Recursively repeat the process for each child
Training Data
Data
Test Data
Model Evaluation
Performance Evaluation Measures
1. Accuracy
2. Precision
3. Recall

Fig 6: Decision Tree Classifier

The working of the decision tree algorithm is as shown above. Selecting the best attribute is the first step. The default attribute selection method is gini index. Gini index is calculated using the formula mentioned where pi is the probability that a tuple belongs to class Ci.

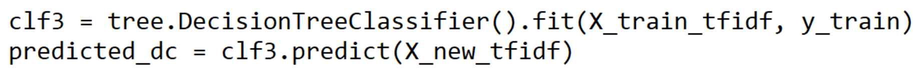
$$\text{Gini(D)}=1-\sum_{i=1}^{m} \text{Pi}^2$$

This attribute is then used to split the dataset into smaller subsets to form a tree like structure. The model is built as shown below and the prediction is then made on the testing dataset.



```
X_new_counts = vectorizer.transform(test_words)
X_new_tfidf = tfidf_transformer.transform(X_new_counts)
```

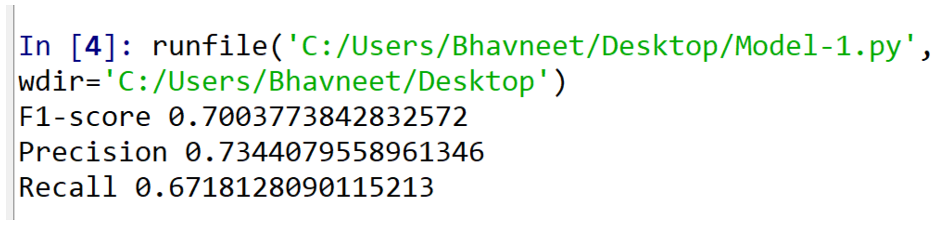
Fig 7: Training the test dataset



```
clf3 = tree.DecisionTreeClassifier().fit(X_train_tfidf, y_train)
predicted_dc = clf3.predict(X_new_tfidf)
```

Fig 8: Decision Tree Algorithm

Finally, after creating the model, we need to evaluate the performance of the model using the evaluation metrics namely, f1-score, precision and recall. This will help us in identifying the correctness of the model and will give us an idea of whether to hyper tune the parameters and how to hyper tune them.



```
In [4]: runfile('C:/Users/Bhavneet/Desktop/Model-1.py',
wdir='C:/Users/Bhavneet/Desktop')
F1-score 0.7003773842832572
Precision 0.7344079558961346
Recall 0.6718128090115213
```

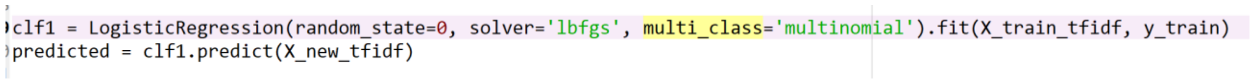
Fig 9: Evaluation Metrics

The above figure shows the f1-score, precision and recall of the decision tree classifier using the model we built.

Algorithm 2 – Logistic Regression Model

Logistic Regression is a statistical machine learning algorithm that classifies the data by considering the outcome variables of both ends and makes a logarithmic line that distinguishes them. This algorithm provides great efficiency.

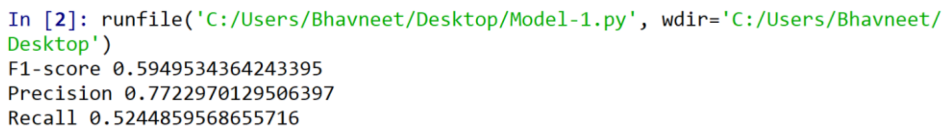
Logistic regression is normally used for binary classification, but to set it to multi-class classification as is desired in our program, we set the multi_class parameter to multinomial. Solver is the algorithm which is used to handle the optimization. To perform multiclass classification, only 'newton-cg', 'sag', 'saga' and 'lbfgs' [10] solvers can be used. The model is built for logistic regression as shown:



```
clf1 = LogisticRegression(random_state=0, solver='lbfgs', multi_class='multinomial').fit(X_train_tfidf, y_train)
predicted = clf1.predict(X_new_tfidf)
```

Fig 10: Logistic Regression Model

Once the model is built and trained on the training set, the model is evaluated using a few metrics. The evaluation of the model using the metrics is as shown:



```
In [2]: runfile('C:/Users/Bhavneet/Desktop/Model-1.py', wdir='C:/Users/Bhavneet/
Desktop')
F1-score 0.5949534364243395
Precision 0.7722970129506397
Recall 0.5244859568655716
```

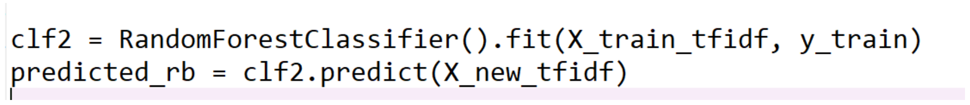
Fig 11: Evaluation metrics

This gives us a precision of 77%, I.e., when our model predicts a class, 77% of the time, the prediction of the class is correct.

Algorithm 3 – Random Forest Classifier

Random Forest Classifier is a supervised learning algorithm that can be used for both classification and regression. A random forest is a meta estimator that fits several decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size, but the samples are drawn with replacement if bootstrap=True (default). [11] It creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It does not suffer from the overfitting problem. The main reason is that it takes the average of all the predictions, which cancels out the biases. [12]

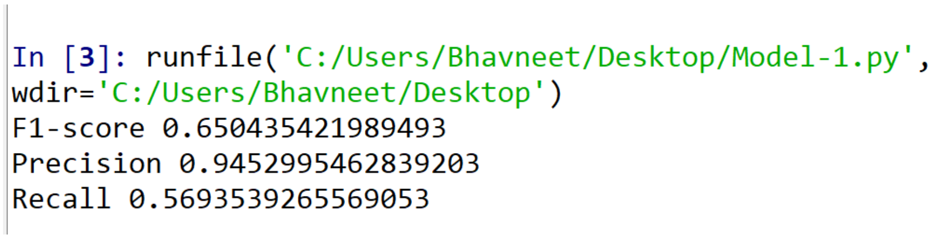
The model for the random forest is built as shown below, and then the prediction is made on the test data set. Here, the default attribute selection criterion is the same as decision tree classification, gini index. But, along with that, it also has some other criterion such as the max depth of the trees or min_samples_split for an internal node. These parameters can be used to improve the evaluation metrics of the model.



```
clf2 = RandomForestClassifier().fit(X_train_tfidf, y_train)
predicted_rb = clf2.predict(X_new_tfidf)
```

Fig 12: Random Forest Classifier

As you can see from the evaluation metrics below, the precision of the random forest classifier is the highest among the three models on the same dataset. This shows that the model for the random forest has been fit correctly and does predict the outcomes with proper precision, which is also important to consider. Only selecting the model based on the accuracy can be misleading.

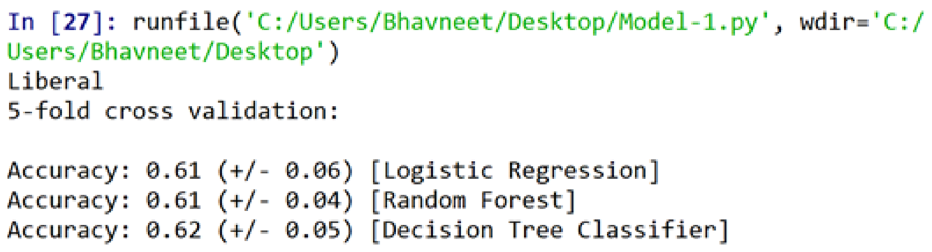


```
In [3]: runfile('C:/Users/Bhavneet/Desktop/Model-1.py',
wdir='C:/Users/Bhavneet/Desktop')
F1-score 0.650435421989493
Precision 0.9452995462839203
Recall 0.5693539265569053
```

Fig 13: Evaluation Metrics

Results

The following are the results of analysis on dataset 1:

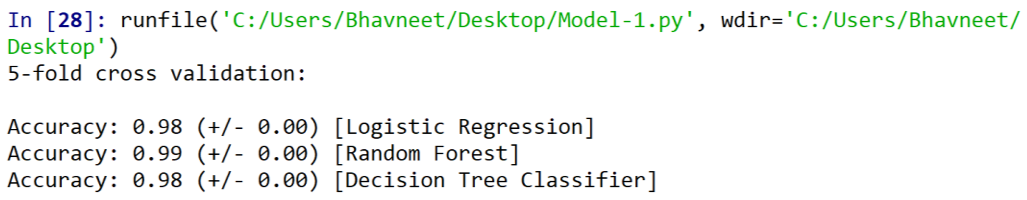


```
In [27]: runfile('C:/Users/Bhavneet/Desktop/Model-1.py', wdir='C:/
Users/Bhavneet/Desktop')
Liberal
5-fold cross validation:

Accuracy: 0.61 (+/- 0.06) [Logistic Regression]
Accuracy: 0.61 (+/- 0.04) [Random Forest]
Accuracy: 0.62 (+/- 0.05) [Decision Tree Classifier]
```

Fig 14: Complete Results for Dataset 1

Similarly, the results for the Dataset 2 are as follows:



```
In [28]: runfile('C:/Users/Bhavneet/Desktop/Model-1.py', wdir='C:/Users/Bhavneet/
Desktop')
5-fold cross validation:

Accuracy: 0.98 (+/- 0.00) [Logistic Regression]
Accuracy: 0.99 (+/- 0.00) [Random Forest]
Accuracy: 0.98 (+/- 0.00) [Decision Tree Classifier]
```

Fig 15: Complete Results for Dataset 2

Conclusion

As you can see from the results above, all the three algorithms are giving almost similar accuracy. Hence, by just having a look at the accuracy a person would assume that all the three algorithms are the same and have the same performance. But other than accuracy we also need to go into in-depth metrics evaluation to determine the correctness of the prediction. That can only be determined if we analyze all the evaluation metrics for the algorithms. From the results, you can see that after analyzing all the evaluation metrics, it is found that the random forest classifier produces the most accurate and the most correct results as well. The other two algorithms have a precision of around 70%, whereas the random forest has a precision of around 90% which is way better than the other two.

Thus, we can conclude that any one evaluation criteria cannot be the sole measure of the correctness and the efficiency of the algorithm. A proper comparison of different evaluation criteria is required to choose the appropriate algorithm.

References

[1]"Élections Canada en ligne - Elections Canada On-line", *Elections.ca*, 2019. [Online]. Available: https://www.elections.ca/. [Accessed: 15- Apr- 2019].

[2]"Elections Canada - Official Website", *Elections Canada*, 2019. [Online]. Available: https://www.elections.ca/res/rep/off/ovr2015app/home.html#15. [Accessed: 15- Apr- 2019].

[3] "#elxn42 tweets (42nd Canadian Federal Election) - Nick Ruest Dataverse", *Dataverse.scholarsportal.info*, 2019. [Online]. Available: https://dataverse.scholarsportal.info/dataset.xhtml?persistentId=hdl:10864/11270. [Accessed: 16- Apr- 2019].

[4]R. Inc, "20 Inspirational Quotes About Data", *RingLead*, 2019. [Online]. Available: https://www.ringlead.com/blog/20-inspirational-quotes-about-data/. [Accessed: 16- Apr- 2019].

[5]"googletrans", *PyPI*, 2019. [Online]. Available: https://pypi.org/project/googletrans/. [Accessed: 16- Apr- 2019].

[6]"TF-IDF: Vector representation of Text | CommonLounge", *Commonlounge.com*, 2019. [Online]. Available: https://www.commonlounge.com/discussion/99e86c9c15bb4d23a30b111b23e7b7b1. [Accessed: 16- Apr- 2019].

[7] J. Brownlee, "An Introduction to Feature Selection", *Machine Learning Mastery*, 2019. [Online]. Available: https://machinelearningmastery.com/an-introduction-to-feature-selection/. [Accessed: 16- Apr- 2019].

[8]J. Brownlee, "A Gentle Introduction to k-fold Cross-Validation", Machine Learning Mastery, 2019. [Online]. Available: https://machinelearningmastery.com/k-fold-cross-validation/. [Accessed: 16- Apr- 2019].

[9] "Decision Tree Classification in Python", DataCamp Community, 2019. [Online]. Available: https://www.datacamp.com/community/tutorials/decision-tree-classification-python. [Accessed: 16- Apr- 2019].

[10]"sklearn.linear_model.LogisticRegression — scikit-learn 0.20.3 documentation", Scikit-learn.org, 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression. [Accessed: 16- Apr- 2019].

[11]"3.2.4.3.1. sklearn.ensemble.RandomForestClassifier — scikit-learn 0.20.3 documentation", Scikit-learn.org, 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. [Accessed: 16- Apr- 2019].

[12]2019. [Online]. Available: https://www.researchgate.net/publication/299867566_Twitter-based_Election_Prediction_in_the_Developing_World. [Accessed: 16- Apr- 2019].

[13]"The 2015 Canadian Election on Twitter: A Tidy Algorithmic Analysis", *Socialdatasciencelab.org*, 2019. [Online]. Available: https://socialdatasciencelab.org/research/policyImpactsResources/canElection2015LdaDashboardEng.html. [Accessed: 16- Apr- 2019].