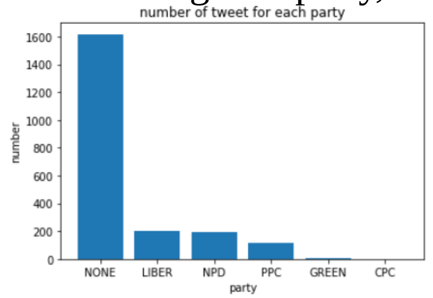
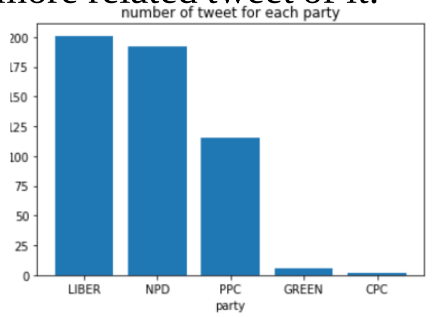
Exploratory analysis : Politics parties and tweet length

In this part we use key word to divide Twitter into different parties. The key word we use to judge a tweet in which party is as below. We count the frequency of the key word and a tweet will be divide into the party by the frequency of keywords. If the frequency of each party is the same or zero, it will be put into “NONE”

NDP: voteforchang, votendp, ndp ,pro-ndp, ndp2019, jackharrisndp, jaedendt_ndp
LIBER: voteliber, ivotedliber, voteliberalstopthecon, liber, liberal_parti, liberalparti, liber, liberal_parti, liberal_parti
GREEN: votegreen, votekooy, votegreen2019
PPC: voteppc2019, voteforjo, voteppc, votetrudeauout2019, cpacvote2019, ppc, northppc, ppc2019, ppcdbn, ppc2019alltheway, ppctilden, ppc
CPC: votecpc, votecarol

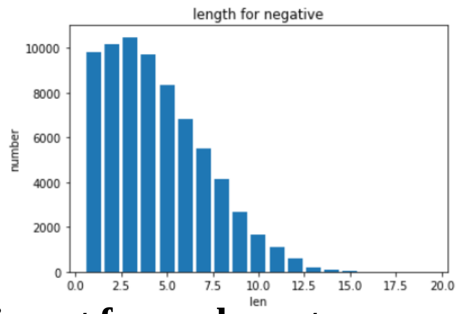
According to the graph showed in botton we can see that none is dominating, that is because most of the tweet is like "vote for animal“ "student vote " which is not exactly refer to political party. so we remove the "none” and take a closer look. We can see it clearly that the LIBER has most related tweet, NDP second, next is PPC. We can conclude that the large the party, the more related tweet of it.

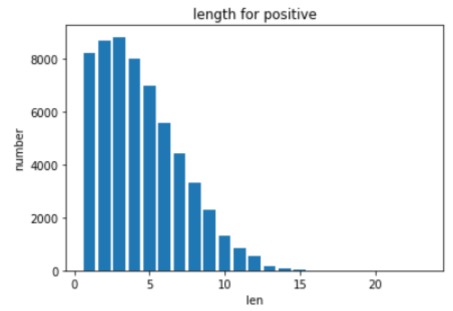


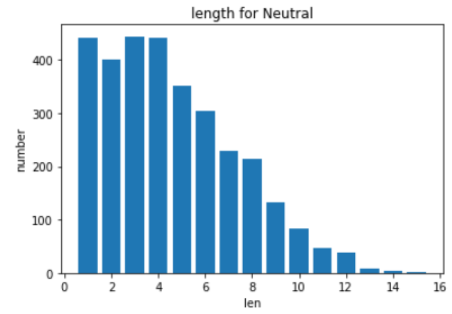


Length of tweet

For the sentiment dataset we want to figure out the relationship between tweet length and its sentiment. We plot the length and the number of each sentiment. According to these 3 pic we can see that to the positive and negative tweet, the distribution is the same. They all seems to have a same trend they all reach the peak at about len=3 then drop. The only different between negative and positive is the total number of each of them is a little different, but the distribution seems to be exactly the same. There is also a 3rd class Neutral, the trend is also tends to be the same while there is another peak around len 1, perhaps because that the sample size is a little some and contains a little disturbance.

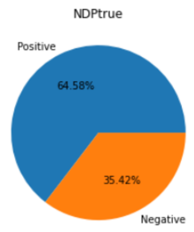


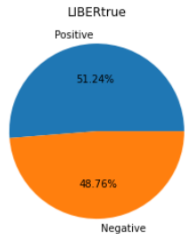


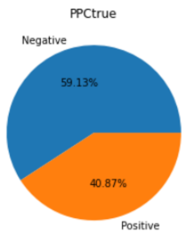


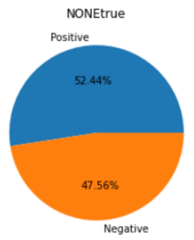
Sentiment for each party

All the three parties(which is 4 in above , we add NDP), the positive sample in NDP seems much more than the negetive sample in NDP, And for the liber and NONE class, the postive sentiment takes about 50% and the rest is nagetive. what we need to point out is that for the PPC, the public tends to be more friendly, as we can see there are 60% is postive with 40% is negative. perhaps guys think PPC will win the vote or PPC have a better reputation among the people.







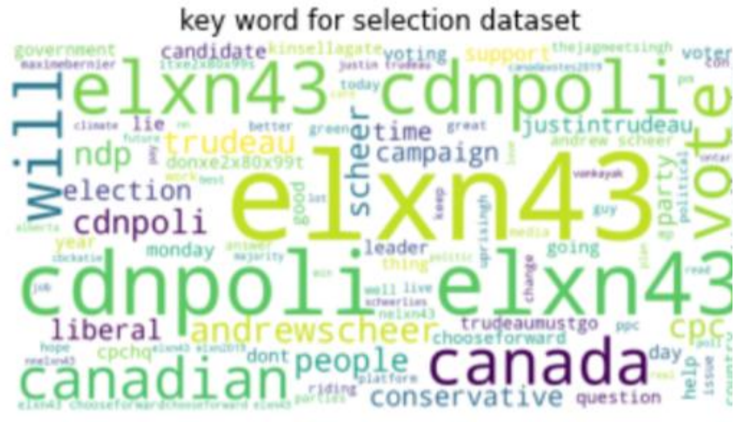


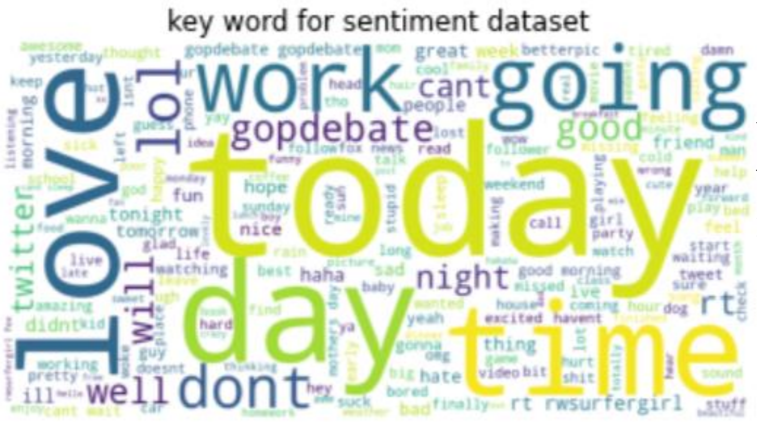
Exploratory analysis : Word frequency of each dataset and feature selection

Word frequency:

The words with highest frequency in election data set. As shown in the wordcloud, the tag about election is the word with highest frequency like “elxn43” and “vote”, Then the name and the principle ideas of each politics parties like “people”, “ndp” . Then is some word about Governing commitment like “will”.

For the dataset of sentiment the problem comes complicated. Words like working and going have higher word frequency, which can not tell the sentiment is negative or positive directly. While there is also some words with obvious emotion like “haha” “well” “love”. Some useless word still involved in the word cloud like ”rt” which means the user just repeat others’ tweet.





Feature selection:

we only pick up the word whose WF is higher than 50 when we training the model for predicting sentiment, due to the risk of crash of the code. For the dataset concludes 130000 data points, even the word whose WF is lower than 50 have significant relationship with sentiment, it may be considered as an outliner due to very little sample number. (why we choose 50 is that 50 is the min value we can aviod the crash happen(according to practical experiment, we still want to get as more information as possible).

Advantages: we can avoid the occurrence of crash and with less features we can improve the training speed of model. With these words we still contains main of the information.

Disadvantages: we drop many words and the words left is only 10-20%. We still lose information. And it may cause lower accuracy. What’s more is that we will use this “useful word list ” in the process of encoding of election dataset, which is not exactly same with the sentiment dataset and that may cause disturbing to the second model .

Implementation, tuning and result of Model 1 :Predicting the sentiment of tweet

Subset:

For the extremely long running time and we want to use cross-validation to prevent our models from overfitting, we choose a dataset with 10000 samples in it, a “mini set”. We use code “df.sample(10000)” to make sure that the mini set is chosen randomly.

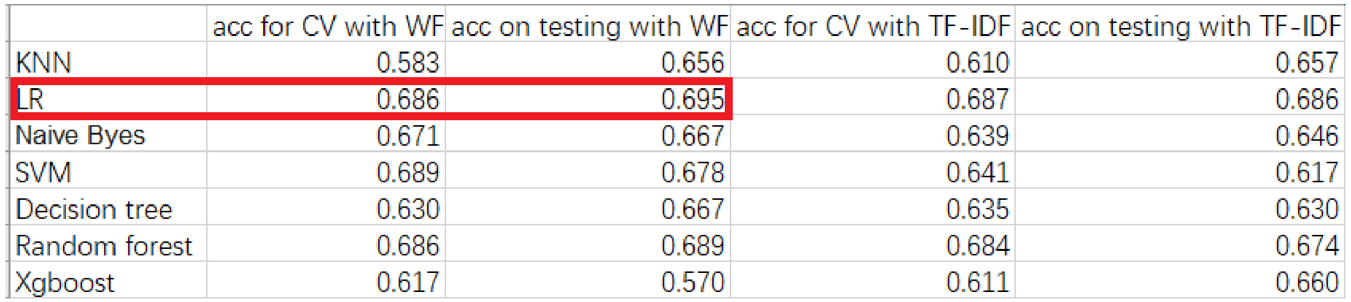
Advantage: “mini set” make it possible for us to use cross-validation on every single algorithm so that it can somehow prevent models from overfitting. The time is large shorter than use the whole set.

Disadvantage: “mini set” only contains 10000 samples which may cause the “mini set ” can not representative of all data. This will cause some noise of the model training and may lead to low accuracy when the models work on testing set.

Verification: after run this process for many times the accuracies of each model of CV(except SVM and Xgboost, they costs too much time, so we just fit train and predict testing) maintain at a similar level the accuracy. And we tried increase the size of this “mini set” to 20000 and 50000 to verify that the size of subset is enough or not, the accuracies of each model is still around 60%. So we can conclude that the size of “mini set” is practical and feasible.

Tuning:

We use cross-validation to tune the hyperparameter and the result we got is as belows. We choose the logistic regression with C=0.1,solver=‘lbfgs’ and apply it on the election set.



	acc for CV with WF	acc on testing with WF	acc for CV with TF-IDF	acc on testing with TF-IDF
KNN	0.583	0.656	0.610	0.657
LR	0.686	0.695	0.687	0.686
Naive Byes	0.671	0.667	0.639	0.646
SVM	0.689	0.678	0.641	0.617
Decision tree	0.630	0.667	0.635	0.630
Random forest	0.686	0.689	0.684	0.674
Xgboost	0.617	0.570	0.611	0.660

Result:

Finally we choose logistic regression with C=0.1,solver=‘lbfgs’ to predict the election. And the accuracy we get is **60.1%**

How NLP works on twitter and election

the accuracy is just 60% on election set and to avoid crash we drop the word whose frequency is lower than 50. Which cause lose some important information(like gooooooooooal, It can be 100% positive but we drop it. So if we use NLP analytics based on tweets can somehow predict the result but it can't be quiet useful because language is really unpredictable and will cause to much noise for the model (even we use stem)

Implementation, tuning and result of Model 2 :Predicting the reason of negetive

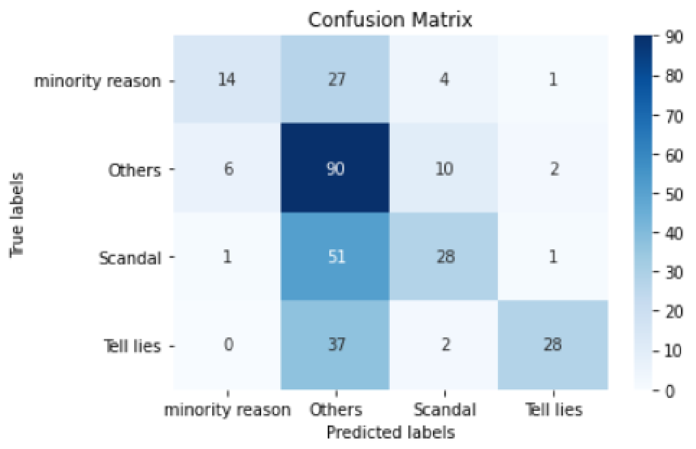
Combine similar reasons into fewer categories:
First of all, we will see the number counts of each class.
 In this table we have already combine
“Healthcare and Marijuana” and ”Healthcare” together.

```
Others                                  364
Scandal                                 270
Tell lies                               198
Economy                                  51
Women Reproductive right and Racism      45
Climate Problem                          41
Separation                               16
Privilege                                12
Healthcare                                9
```

According to the table above we can see that first 3 reasons are Others(364),Scandal(270), and Tell lies(198). In this situation, the sum of all the rest is only 174. and a small number of these class will make it difficult for the classifier to give them a class especially for algorithm like kNN and SVM having a deep relationship with distance. So we decide to combine these reason together as a new reason “minority reason”.

Tuning:
We use cross-validation to tune the hyperparameter and the result we got is as belows. The three algorithms are logistic regression, naive byes and decision tree.

	acc for CV with best hyprameter
Decision tree	0.463
Naive Byes	0.489
LR	0.506



Result:
Finally we choose logistic regression with C=0.1,solver=‘liblinear’ to predict the election. And the accuracy we get is **52.9%**

Here is a confusion matrix to show the difference between y_true and y_pred, we can see that a lot of samples is identified as class’ other’ which is wrong.

Discussion and how to improve 2 models

For the first model:
As we can see above the accuracy of the first model for predicting the sentiment of select-relating data is around 60%. When the model work on the training set the accuracy can reach 69.8%. It is a little overfitting.
Reason:
The first reason causing this situation is that when we are doing the data preparation we choose the word whose WF is higher 50 as features and drop all the rest. Only 1000 unique words left. By doing this the crash is avoid while some useful information is lost either. The second reason is that the data set we choose to train and we choose to test is not exactly the same. Which means in the sentiment training set we do not 100% target on election-sentiment analyzing. So perhaps the features we choose to do is not perfect match the test set and it causes decrease in accuracy.

Solution:
1.Better data cleaning. A better data cleaning can reduce the number of feature so that there is no need for us to use “mini set” and we can cover all the information in the data set.
2. Get a subset from sentiment which contains all the tweets about politics, so that the data we use to train is more match to testing set and will solve the question state in reason #2
3. A different thinking method from solution#1 we can drop the tweet whose length is lower than a number (for example: 3). According to the graph shown in the first page , the distribution of length of both class is same, so perhaps it can give us a cleaner data set which is good to our model building and predicting.

For the second model:
Reason :
There are 2 reasons why the accuracy of the second model is relatively low. The first reason is same with the reason of the first model. We drop the word whose WF is lower than 50 and use it to transform the election data which will cause information losing in the process, especial some politics words have low frequency in daily life will appear more in politics-relate tweet. The second reason is that the negative reason given by the dataset not have a clear boundary. according to the confusion matrix we build in the last step, it can be shown clearly that all other class will be class into class "other". this perhaps because that when the data is collecting, any sample that can tell the reason will be put into class "other" and it cause that "other" class has various characteristic and cause a lot of noise for classification.

Solution:
1.for the second reason, it is caused by collection of the data so perhaps there is not only one center in this class perhaps we can use mixture gaussian distribution to improve this question.
2.We can use the election dataset to do the vectorizer.fit_transform() instead of vectorizer.transform(). And do the feature selection again. By doing this we can create a model which is more suitable for the election data.(what we apply now the result of doing feature selection on sentiment dataset) It can contains as much information as possible and will reduce the noise. And it can lead to a higher accuracy.