Title
Sentiment classification Using different machine learning algorithms and different word vectorizer

Author
Cui, Shuyang

Publication Date
2021

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Sentiment classification Using different machine learning algorithms and different word vectorizer

A thesis submitted in partial satisfaction

of the requirements for the degree

Master of Applied Statistics

by

Shuyang Cui

2021

ABSTRACT OF THE THESIS

Sentiment classification Using different machine learning algorithms and different word vectorizer

by

Shuyang Cui

Master of Applied Statistics

University of California, Los Angeles, 2021

Professor Yingnian Wu, Chair

The goal of this thesis is to apply different machine learning algorithms and different word vectorizer methods on twitter messages to do the sentiment classification. By applying two word vectorizer methods and three machine learning algorithms, I will get six different models' results. After that, I will analyze the model results to find out the best machine learning algorithm and best word vectorizer method for the data.

The thesis of Shuyang Cui is approved.

Frederic R. Paik Schoenberg

Nicolas Christou

Yingnian Wu, Committee Chair

University of California, Los Angeles

2021

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

As more and more people are willing to share thoughts and opinions on social media, interpreting the voices with efficiency and accuracy becomes crucial to many industries. By automatically sorting and clustering the conversations, tweets, and comments, decision makers can retrieve information about a customer's perception of a topic or a product faster and easier. This thesis aims to apply the machine learning techniques to predict the sentiment classification pertaining to the well-known topic – Climate change. Climate change could affect the society through impacts on human health, energy, and even water supplies. Some people hold the opinion that global warming is primarily as a result of human activities. Oppositely, others think global warming has nothing to do with human beings. With the use of sentiment classification, people could investigate the perceptions of people on climate change which could potentially benefit decision-makers with the understanding of public voice. Another benefit of using sentiment classification, in this case, is that it can significantly increase the efficiency of the process. If people have to manually go through each tweet to draw conclusions and con- duct statistical analysis, it will be extremely time-consuming and even inaccurate. Sentiment classification can help us to systematically gather information, identify potential trends, and extract useful information with efficiency and limited human efforts from an objective point of view.

However, it is tough for machine learning methods to train the textual con-

tents directly. The numerical contents are the best friend with the computer.To solve this problem, there are several message vectorization methods which are widely used. In this thesis, we will focus on TF-IDF and Countvectorizer.

In this thesis, TF-IDF vectorizer and Countvectorizer are two methods to vectorize the textual message to vectors and count the importance of different words in determining sentiment. In the following thesis, I will apply TF-IDF and Countvectorizer methods to vectorizer the textual message, and after that, the two groups of vectors will be separately trained by the same group of machine learning approaches. At the end, we will do the comparison between the results made by both different word vectorizers and machine learning algorithms.

CHAPTER 2

Problem Statement

The goal of the thesis is to analyze different machine learning models and different word vectorizers on predicating the sentiment classification. The prediction involves one input and one output. The input will be tweets' textual messages, and the output is classifications. At the end, we will do analysis on the predicted classification results. In order to achieve this goal, the following steps will be needed:

- Retrieve the data from website
- Grub the stop word and symbols to clean up our data
- Do the explanatory data analysis to know the data
- Utilize the TF-IDF vectorizer to vector the textual messages
- Training the machine learning models on vectors
- Utilize the Countvectorizer technique to vector the textual messages
- Training the vectors with the same machine learning algorithms
- Perform analysis on results grouped by different vectorization methods
- Based on the result analysis, make conclusion

CHAPTER 3

Data Analysis

3.1 Date set

The data set was collected by Edward Qian, and was funded by the University of Waterloo. It contains twitter comments related to the global warming or climate change topic. There are 43943 tweets in total which are from April 2015 to February 2018. Data set contains three columns: sentiment(-1, 0, 1, 2), message(tweets), and tweet id. For the sentiment column, it has four classifications. Every tweet is read by three reviewers, and the classified labels are agreed by all reviewers. Detailed meaning for the numerical classifications is following:

- Anti(-1): The tweet shows it doesn't believe in human-made global warming or climate change
- Neutral(0): The tweet shows the indifferent between whether the global warming is human-made or not
- Pro(1): The tweet shows it believes that the global warming or climate change is a man-made issue
- News(2): the tweet is linked with a factual account about climate change or global warming

3.2 EDA

3.2.1 EDA on the sentiment part

First of all, we look at the last five rows of data to have a better understanding of this data set.

	sentiment	message	tweetid
43938	1	Dear @realDonaldTrump,\nYeah right. Human Medi...	791307031919550464
43939	1	What will your respective parties do to preven...	791316857403936768
43940	2	RT @MikkiL: UN Poll Shows Climate Change Is th...	791357509101621249
43941	0	RT @taehbeingextra: i still cangt believe th...	791390042136641537
43942	1	@Likeabat77 @zachhaller \n\nThe wealthy + foss...	791401610308038656

Figure 3.1 Display of the last five rows

As we can see, the sentiment is labeled by the number in the original data set, which is easy for a machine learning step, but classification of numbers would make data visualization harder to understand. Hence, I change the numbers to its actual meaning while doing the EDA, where “Anti” represents -1, “neutral” represents 0, “Pro” represents 1, “News” represents 2. After the replacement, the last five rows become:

	sentiment	message	tweetid
43938	Pro	Dear @realDonaldTrump,\nYeah right. Human Medi...	791307031919550464
43939	Pro	What will your respective parties do to preven...	791316857403936768
43940	News	RT @MikkiL: UN Poll Shows Climate Change Is th...	791357509101621249
43941	Neutral	RT @taehbeingextra: i still cangt believe th...	791390042136641537
43942	Pro	@Likeabat77 @zachhaller \n\nThe wealthy + foss...	791401610308038656

Figure 3.2 Display of the last five rows after replacement

We have three columns: sentiment, message, and tweet id. Tweet id does not contain any useful information, so the data analysis part will mainly focus on the sentiment and message parts. I will do the EDA on the sentiment part first.

Sentiment	Number
Anti	3990
Neutral	7715
Pro	22962
News	9276

Table 3.1

From the table 3.1, we noticed there are 22962 tweets hold the “Pro” sentiment; 9276 tweets hold the “News” sentiment; 7715 tweets hold “Neutral” sentiment, while 3990 tweets shows “Anti” sentiment. As we can see, the sentiment is not perfectly balanced, and there is no way to combine any of two classifications to one due to the actual meaning of labels. Unbalanced may lead to the low accuracy of our model.

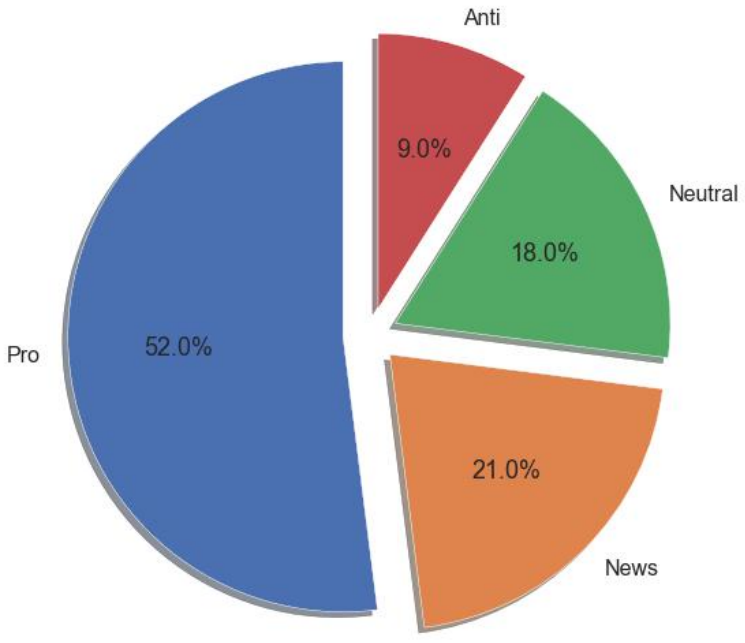

Figure 3.3: Distribution of sentiment

The rank of sentiment from the largest group to the lowest group are: Pro, News, Neutral, Anti. This means that more than half of tweets' publishers prefer to believe that climate change is caused by human activities. To develop a better understanding of message columns, the following EDA contains two parts: the first part will be the analysis of the message length which tries to find connection between tweet length and sentiment. The second part will be the analysis of the textual content of frequent words or tags. EDA is an essential step to help us know our data set, and a better understanding of data would save us great energy in the model building process.

Count	43943
Mean	17.4
Std	4.6
Min	1
25%	14
50%	18
75%	21
Max	97

Table 3.2

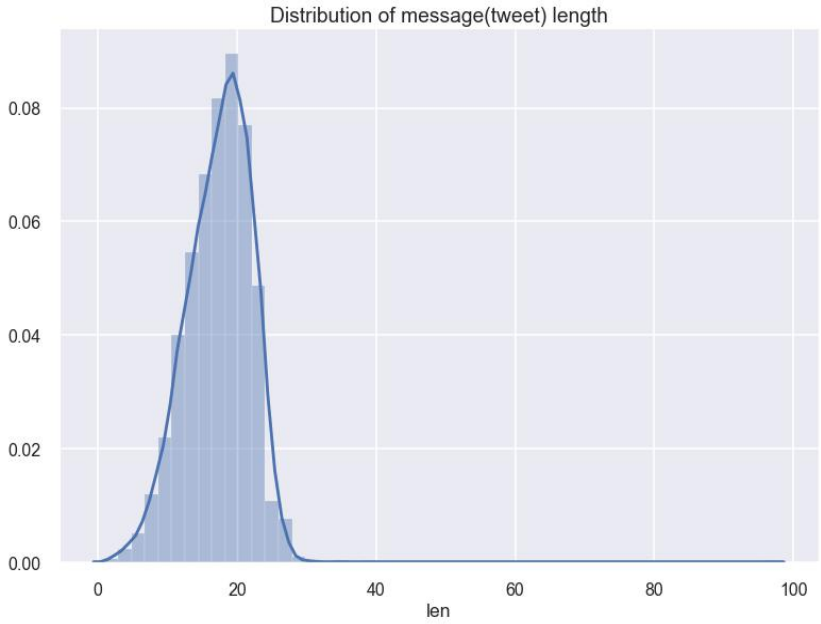

Figure 3.4: Distribution of message length

From the table 3.2 and figure 3.4, we can see that the 25% to 75% interval of messages' length is from 14 to 21 with a mean of 17.4 and standard deviation of 4.6, and only a few of messages contain more than 25 words. The maximum length of message is 97 while the minimum length is 1. In the next step, it is

interesting to check what is the relationship between message length and sentiment classification.

Sentiment	Mean length
Anti	18.14
Neutral	16.26
Pro	18.5
News	15.32

Table 3.3

From the table 3.3, we can see the mean length of "Anti" message is the longest with the length of 18.14, and the mean length of "News" message is the shortest with the length of 15.32. However, there is no evidence showing there exists a statistically significant difference between the mean of four lengths. In other words, adding length as a variable into the model training needs more evidence to support.

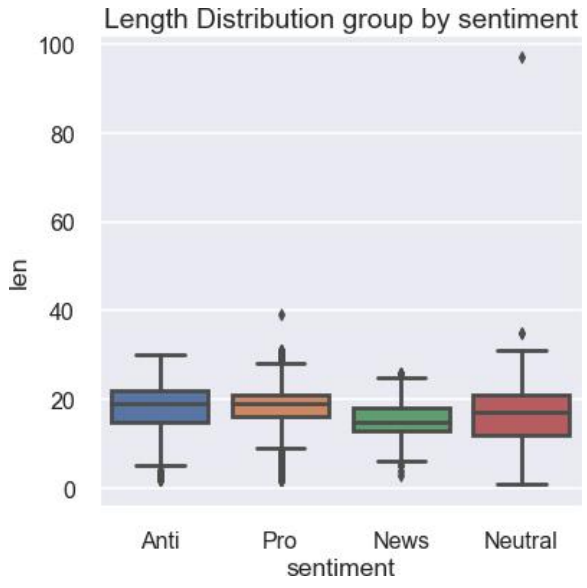

Figure 3.5: Length distribution grouped by sentiment

From the figures 3.5, all the four distributions look fine without too many outliers. Both of four sentiments' distributions are not too skewed. For the next part, I will introduce the exploratory data analysis on the message part.

3.2.2 EDA on the message part

Before drawing any plot related to the message, we need to clean the data. First of all, the original textual messages contain too many useless signals. To get more accurate EDA result and also for better model training, we need to clean the data by filtering out the extra signals and stop words.

Word and hash-tags inside the message are the essential indicators in sentiment classification. To have better useful word frequency plot, we cleaning up the messages with removing the stop words, then lemmatizered the words.

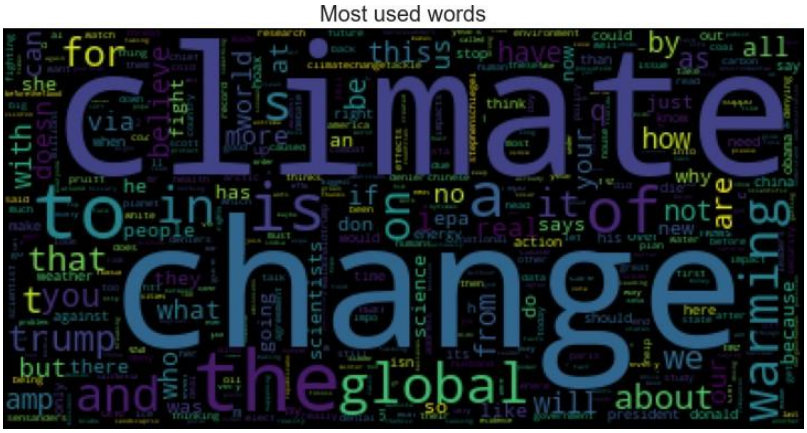

Figure 3.6: Word cloud

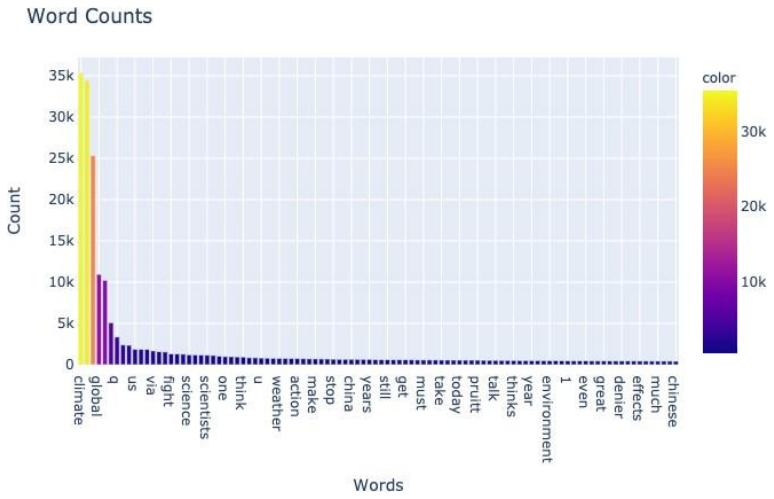

Figure 3.7: Word count plot

It is reasonable for words such as: “climate”, “warming”, “change”, “global” , being the top frequently in the messages. Also, considering the data collected during Donald Trump’s term of office, seeing that “trump”, “president” are the top frequent words are also not surprising.

Since the data will be used to do the sentiment classification, it is crucial to have a basic understanding on word frequency in different sentiment classes. Hence, I will locate data by its sentiment labels, and draw the word-cloud plots for four different sentiment classifications to check the common and difference.

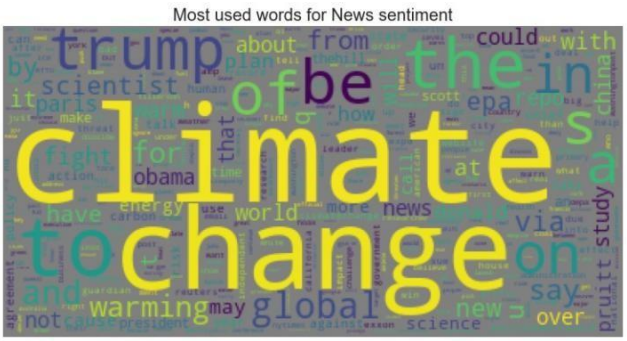

Figure 3.8: Word cloud for News

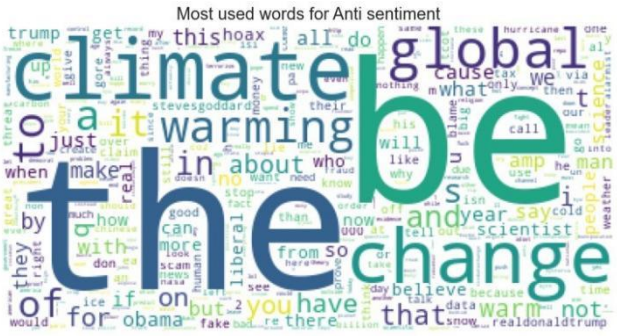

Figure 3.9: Word cloud for Anti

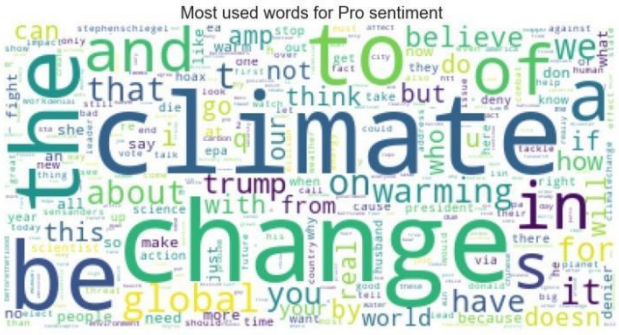

Figure 3.10: Word cloud for Pro

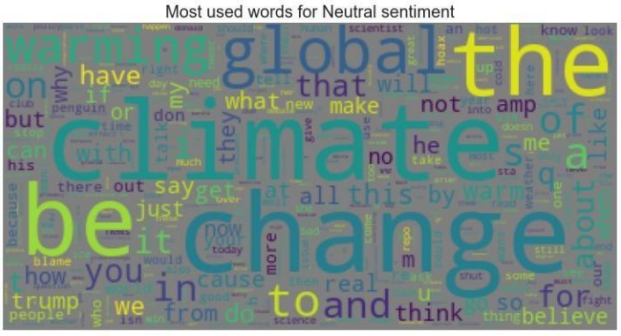

Figure 3.11: Word cloud for Neutral

From the figures, the messages from all four classifications share lots of common top frequent words, such as: “climate”, “change”, “warm”, and etc. However, there are still some differences between messages from various classifications. For example, we can see that the messages with “Anti” class contain more political words, such as:

“realdonaldtrump”, “Obama”, and they also have more negative emotional words compared to others, such as: “hoax”, “scam”.

In the following chapter, the thesis will mainly focus on methodologies which are used to clean the data and train the model.

CHAPTER 4

Methodology

For this thesis, the methodologies can be divided into two major parts. The first part is the data(text) pre-processing, involving the tokenization, countvectorizer, and term frequency - inverse document frequency (TF-IDF). The second part consists of three different machine learning techniques: Logistic Regression, Random Forest, Naive Bayes.

4.1 Text Pre-processing

For the text pre-processing part, I will first apply the tokenization to split the sentence into a set of words. Then for the word embedding part, two methods: countvectorizer and TF-IDF will be applied. Hence, we will have two different groups of vectors: one is applied with a countvectorizer and the other one is applied with TF-IDF.

4.1.1 Tokenization

Tokenization is regarded as the very first step of the natural language process(NLP), and it has the function of splitting a sequence of texts into units with its semantic meaning [1]. Also it is necessary to do the tokenization since generally textual content is the set of characters at the very beginning, and all text analysis processes require the textual data is readable to the computer [2]. In this thesis, we will apply the tokenization inside the TF-IDF and countvectorizer algorithms by setting the parameter “analyzer = ‘word’ ”.

Take the first message as an example, the original text message is “climate change be an interesting hustle”. After applying the tokenization from NLTK, it

becomes [’climate’, ’change’, ’be’, ’an’, ’interesting’, ’hustle’].

After the tokenization, we still need to vectorizer the textual content, and this step enables the data for machine learning training. TF-IDF and Countvectorizer are the two techniques involved in this thesis. Also, one of the goals of our thesis is to find which word vectorization technique is the best fit for our model.

4.1.2 Countvectorizer

For this thesis, I will apply the countvectorizer as the first word embedding technique. Countvectorizer is a simple but efficient way of word vectorization. The basic idea for countvectorizer is to create a vector that has the same dimension as the size of sentence, and for each word, we put it as one into the dimension. For every time this vocabulary appears, we increase the count in the dimension by one[3].

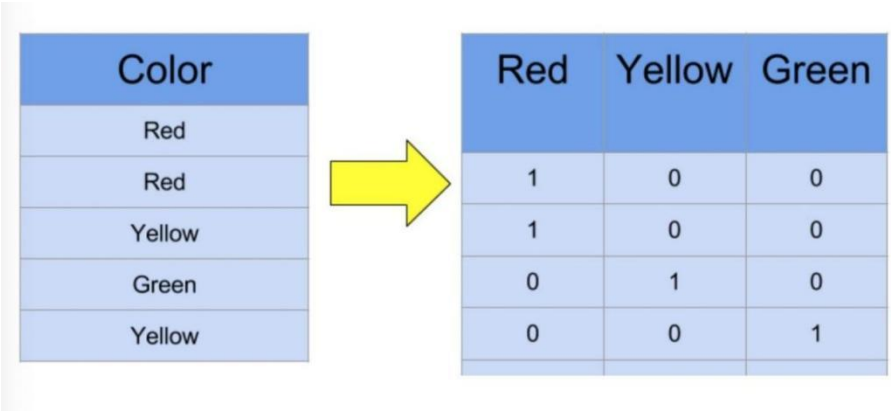

Figure4.1: countvectorizer (‘https://towardsdatascience.com/natural-language-processing-count-vectorization-with-scikit-learn-e7804269bb5e’)

The above figure shows us the basic idea of the countvectorizer, and different from the picture, in the practice, the countvectorizer also should transfer the column names, in this example “red”, “yellow”, “green”, to numerical index for

model training convenience. For the next part, I will introduce the second word vectorization method in this thesis, the term frequency - inverse document frequency (TF-IDF).

4.1.3 TF-IDF

Term frequency - inverse document frequency (TF-IDF) is considered as one of the most commonly used term weight schemes in the nature language process for the words retrieving [4]. Compared to the countvectorizer, term frequency not only considers how many times the words appear, but also takes how many times the same words appear in other documents into account. Although the words may appear frequently in the document, if they also appear frequently in other documents, they are likely to have lower TF-IDF score. Hence, the TF-IDF is aimed to have more precise measure of how important the words for the documents are and avoid to overrate the stop words and common frequent words.

To show some mathematical background of TF-IDF, the TF-IDF score is calculated by determining the relative frequency of words in the specific document compared to the words' inverse proportion over the entire document corpus [5]. The calculation of TF-IDF can be divided into two parts. The first part is TF(term-frequency), and second part is IDF(inverse document frequency).

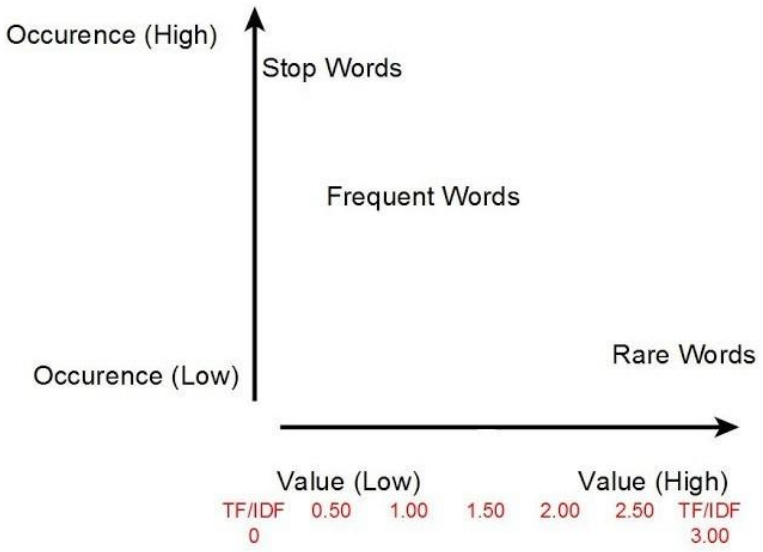

Figure 4.2: TF-IDF ('https://towardsdatascience.com/introduction-to-word-embeddings-4cf857b12edc')

We first calculate the TF part, this calculation shows us about how many times a word is showing in its document, the formula is following:

$$tf(w,d) = log(1 + f(w,d))$$

In this formula, where:

- d is the given document
- w is a given word in this document

Secondly, we calculate the inverse term frequency, and the formula is following:

$$idf(w,D) = log(\frac{N}{f(w,D)})$$

In this formula, where:

- N is the number of total documents in the data set

- D is the set of all documents which contain the word “w”

After have TF and IDF parts, we finally calculate our TF-IDF by the following:

$$tfidf(w,d,D) = tf(w,d) * idf(w,D)$$

As we can see, the TF-IDF contains both TF and IDF scores. If the word “w” appears frequently in other documents, it will have low IDF score which result in the low TF-IDF score[6].

4.2 Machine Learning Models

After cleaning and vectorizing the text, the data is ready for the machine learning training. Machine learning training can be divided into the supervised training and unsupervised training. All methods in this thesis are grouped in the super- vised training. Furthermore, the supervised model training is set into two parts: one is regression and the other one is classification. The prediction outcomes are four labels in this case: -1(Anti), 0(Neutral), 1(Pro), 2(News), so the super- vised model training would be the classification in this project. We will have four machine learning classifications to introduce: logistic regression, random forest classifier, multi-nomial naive bayes, and KNN.

4.2.1 Logistic Regression

The logistic regression is designed to solve the binary classification problem. How- ever, for the multi-class classifications, logistic regression is also a very popular and efficient method. The basic idea for the multi-class logistic regression classification is the same as for the binary class.

For the binary class logistic regression, the mathematical algorithm for the

logistic regression is based on the probability prediction lies between 0 to 1. The function of the sigmoid function is to transfer the predicted values to the probabilities.

The mathematical formula for the sigmoid function is following:

$$f(x) = \frac{1}{1 + e^{-x}}$$

After mapping the predicted values to the probabilities, the probability prediction function looks like following:

As the outcome is binary, either 0 or 1, a threshold value is necessary, which is also known as cutoff point. Usually the 0.5 is an ideal cutoff point, any probability higher than 0.5 is assigned to one classification, and any probability lower than 0.5 is assigned to another classification.

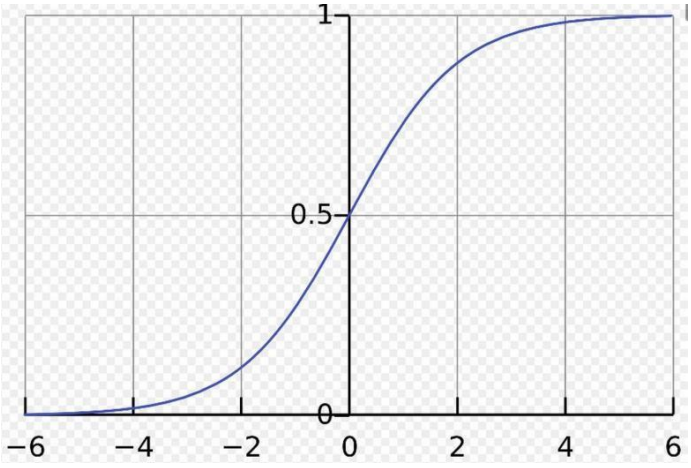

Figure 4.3: Sample of cut-off point

For multi-class classification, the basic algorithm is similar to the binary case. I will take our thesis project as an example: the data has four possible outcomes:

-1(Anti), 0(Neutral), 1(Pro), 2(News). When the logistic regression works on the -1(Anti), it will treat -1(Anti) as a classification, and regard the rest of classes as another classification. If the logistic regression works on the 0(Neutral), it will treat 0(Neutral) as a classification, and treat others as another. Hence, we can also think of multi-class logistic regression as multi-step binary logistic regression.

4.2.2 Random Forest Classifier

Random forest is developed based on the decision tree algorithm. Hence, to understand the random forest, it is crucial to learn the basics about decision tree. Decision tree is one of the most commonly used supervised machine learning algorithms. It is a "tree " structure algorithm which contains root nodes, interior nodes, leaf nodes and branches. Root node is the original sample of data. Interior node is the subset from the upper level node which satisfies specific conditions. Leaf node is the outcome node. Branches are the condition rules and connect the root node, interior node and leaf node together. By weighing the cost of splits (the cost is calculated by the cost function), the decision tree algorithm chooses the split with the lowest cost. This process is also known as greedy algorithm.

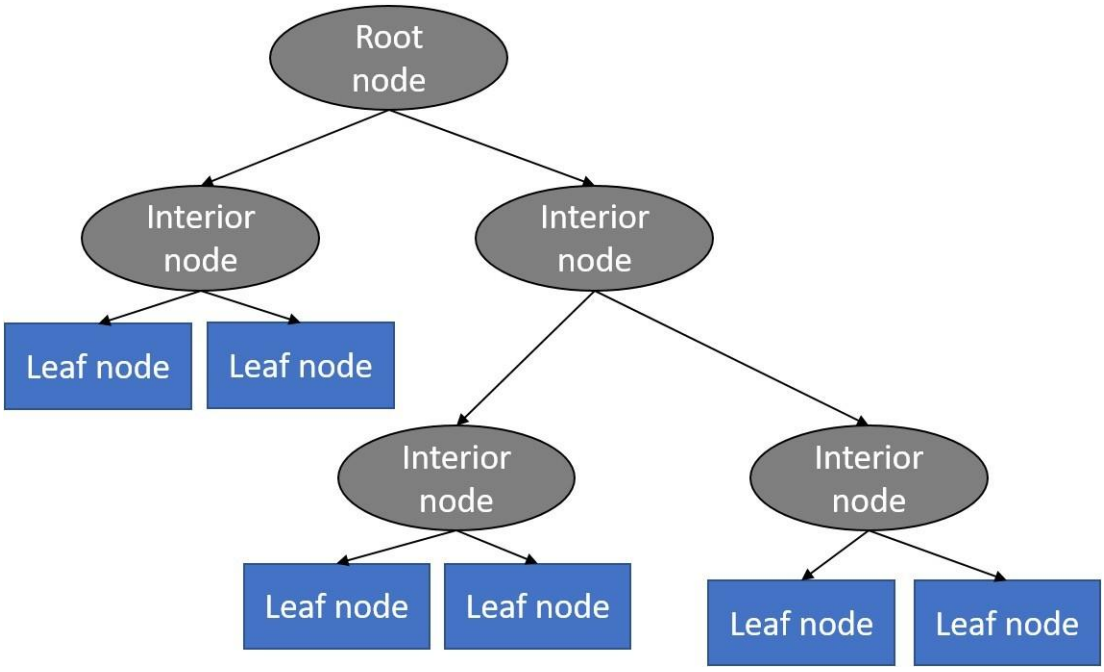

Figure 4.4: Decision Tree('https://www.python-course.eu/Decision_Trees.php')

Random forest is basically a collection of many decision trees. It is developed by Leo Breiman, a statistician from University of California at Berkeley, and Random forest is a machine learning algorithm, which aims to improve classification of data by applying the random sampling and attributes selection [7]. Compare to the decision tree, the random forest has two major differences:

- Random forest apply the random subsets of features for each time splitting nodes, this aims to avoid some features to become too dominated
- Random forest train different samples of data which aims to reduce the variance compare to the decision tree.

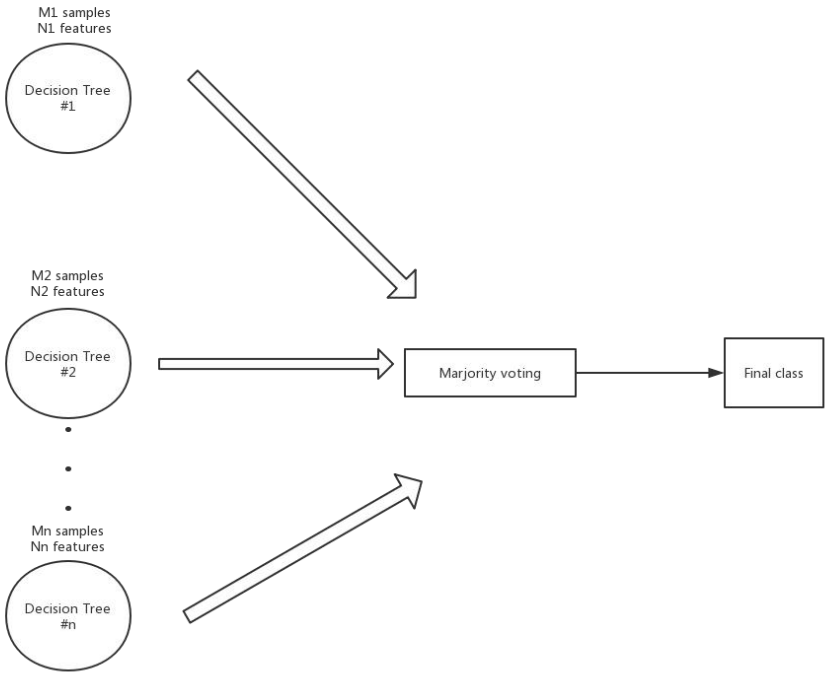

Figure 4.5: Random Forest

As we can see from the picture, random forest assigns the random samples and subset of features to each decision tree, and gets the final class based on the results of all decision trees. Due to the Law of Large Numbers, random forests are more less likely to have an overfitting problem, and also the random inputs and random features are more likely to give good results in classification [8].

4.2.3 Multinomial Naive Bayes Classifier

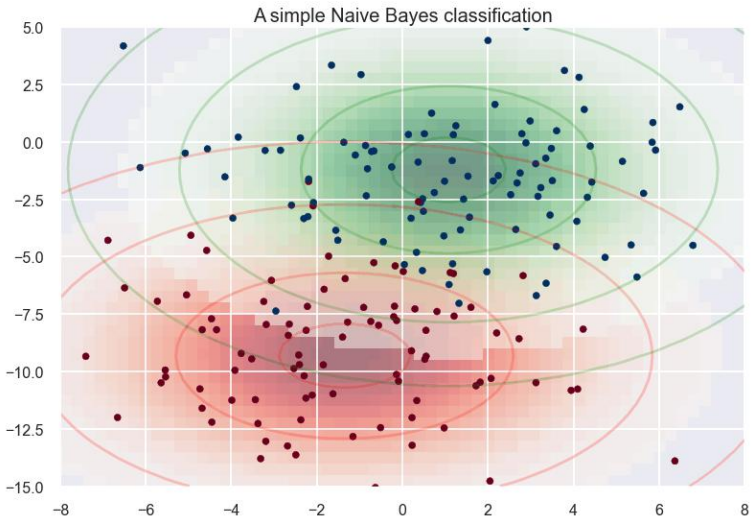

Figure 4.6: Sample of Naive Bayes

Multinomial Naive Bayes (MNB) addresses the task of text classification from a Bayesian principle. MNB makes a simple assumption that word occurrences are conditionally in- dependent of each other given the class of the document. [9]

$$P(C_a|x_1, x_2, ..., x_n) = \frac{P(C_a) * P(x_1, x_2, ..., x_n|c)}{P(x_1, x_2, ..., x_n)}$$

$where\ C_a\ is\ the\ class, x_1\ to\ x_n\ are\ conditional\ independent\ features.$

so we can transform the equation to:

$$P(C_a|x_1, x_2, ..., x_n) = \frac{P(C_a) * \prod_{i=1}^{n} P(x_i|c_a)}{P(x_1, x_2, ..., x_n)}$$

In this case, the features are all available, and we are using them to predict its' class. Hence, given the feature in a specific document, the MNB algorithm is able to give how likely the document belongs to a specific class. This time, we

have four classes: -1(Anti), 0(Neutral), 1(Pro) and 2(News). The MNB algorithm will calculate all four possibilities for a document, and assign the document to the class with the largest probability among these four.

CHAPTER 5

Model Analysis

```
X_all = texts["message"]
y_all = texts["sentiment"]

xtrain, xvalid, ytrain, yvalid = train_test_split(X_all, y_all, random_state=42,
                                                  test_size=0.1, shuffle=True)
```

```
print (xtrain.shape)
print (xvalid.shape)
```

```
(39548,)
(4395,)
```

Figure 5.1: Code

Data split is an essential step in modeling. In the exploratory data analysis part, the existing evidences did not show the length of message is statistically different between four classifications. Hence, in this case, the input is only texts message, while the output is sentiment. I divide 90% percent data to the model training group, and the 10 percent data to the testing group.

The reason that I assign 90% percent data to the training group is the data set contains 43943 observations with four possible outcomes. This is also a text classification, the most of observations has more than 10 features. To avoid over-fitting or underfitting problem, in the consideration of relative small data samples compare to the number of features and outcomes, I decided to assign more percent of data to the training group than usual. At the end, the training group contains 39548 samples and testing group contains 4395 samples. I set the random_state to be 42, and the random_state is a seed to the random generator which aims to keep split remain unchanged. Also, I do the data split with shuffling.

Then, I will apply two text vectorization methods to the message. These two methods(countvectorizer and TF-IDF) will transfer the same group of text to

the vectors. Hence, we will have two groups of vectors, and these two groups will be separately trained by the same machine learning algorithms. One of our model analysis goal is to compare the results grouped by the different vectorization methods and find which one is the best fit to the data. The countvectorizer and TF-IDF are imported from sklearn package.

For the next step, I will apply three machine learning algorithms(Logistic Regression, Random Forest Classifier, Multinomial Naive Bayes Classifier) to two groups of vectors(Countvectorizer, TF-IDF). Hence, at the end, we will have six results, and I will do the model analysis based on these six results.

For the model comparison and analysis part, we will plot the confusion matrix and get the F1 score, accuracy score, recall score and precision score. In text categorization, the F1-score is often used to evaluate classifier performance [10]. Precision score penalizes false positives but does not penalize false negatives. Recall score penalizes false negatives but it do not penalizes the false positives [11]. F1 score is an average of the precision and recall score. The accuracy score is a simple but efficient measurement which equals to the ratio of correct predictions to all prediction.

The formula for accuracy is:

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + False\ Positive + False\ Negative + True\ Negative}$$

The formula for precision is:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

The formula for recall is:

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

The formula for F1 score is:

$$F1\ score = \frac{2 * (Recall * Precision}{Recall + Precision}$$

5.1 Random Forest Classifier

In this thesis, Random Forest Classifier algorithm is imported from the sklearn package. To have more precise outcomes, I set the parameter "max_depth" to be 200, which means the maximum depth of a tree will be 200. Big tree can carry more information than the smaller one, which is more likely to make better prediction. I also set the number of trees to as many as the computer can handle which is also more likely to give me better prediction. In the EDA part, we find the data in unbalanced, so I set the class_weight= 'balanced' to reduce the effect of unbalanced data. As I mentioned before, I will apply the same machine learning techniques to train two different groups of vectors: one is created by countvectorizer, and the other one is created by TF-IDF.

5.1.1 Random Forest Classifier with countvectorizer

First, I will show the classification report of predictions made by random forest classifier with countvectorizer:

	precision	recall	f1-score	support
-1	0.57	0.48	0.52	402
0	0.44	0.59	0.50	803
1	0.84	0.49	0.62	2251
2	0.49	0.87	0.62	939
accuracy			0.59	4395
macro avg	0.58	0.61	0.57	4395
weighted avg	0.67	0.59	0.59	4395

Figure 5.2: Classification report

As from the classification report, we can see the overall classification accuracy score is only 0.59, which means the ratio of correct prediction to all prediction is barely more than half. The precision score is high for "1(Pro)", and for f1-score, labels "1(Pro)" and "2(News)" have better performance than other two.

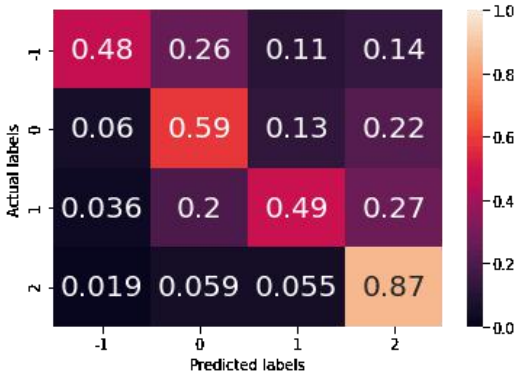

Figure 5.3: Confusion matrix

From the confusion matrix, we can see that the label "2"(News) has pretty good performance on the recall score, but it is fair on precision score due to large number of false positive.

5.1.2 Random Forest Classifier with TF-IDF

Here is the classification report of prediction made by random forest classifier with TF-IDF:

	precision	recall	f1-score	support
-1	0.64	0.38	0.48	402
0	0.53	0.54	0.53	803
1	0.78	0.65	0.71	2251
2	0.54	0.83	0.65	939
accuracy			0.64	4395
macro avg	0.62	0.60	0.59	4395
weighted avg	0.67	0.64	0.64	4395

Figure 5.4: Classification report

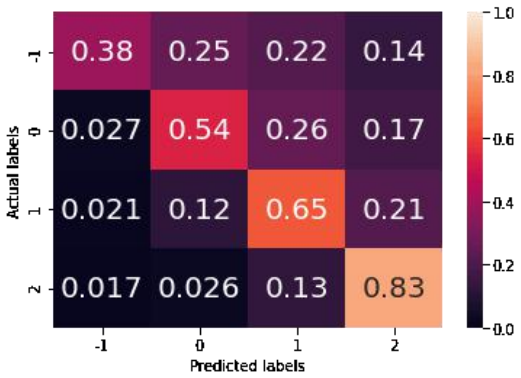

Figure 5.5: Confusion matrix

As from the classification report, we can see that the overall classification accuracy has improved compare to the random forest classifier with countvectorizer. Also, except for "-1(Anti)", the f1-score for other labels has improved.

From the confusion table, we can see that the recall for label "2(News)" is still remains at a high level.

5.2 Multinomial Naive Bayes Classifier

For the multinomial naive bayes classifier, I import this algorithm from sklearn. In the practice, I did not change its parameters since the default one has the better performance. For the default setting, it sets the "alpha" to 1, the "class_prior" to none, and "fit_prior" to true. I also set the class_weight='balanced' due to the unbalanced data. I will also apply multinomial naive bayes classifier to both vector groups created by countvectorizer and TF-IDF.

5.2.1 Multinomial Naive Bayes Classifier with countvectorizer

I will show the classification report for prediction made by Multinomial Naive Bayes Classifier with countvectorizer first:

	precision	recall	f1-score	support
-1	0.74	0.39	0.51	402
0	0.58	0.44	0.50	803
1	0.77	0.79	0.78	2251
2	0.62	0.83	0.71	939
accuracy			0.70	4395
macro avg	0.68	0.61	0.62	4395
weighted avg	0.70	0.70	0.69	4395

Figure 5.6: Classification report

From the classification report, the classification accuracy score is 0.7 which is not bad compare to the result of the random forest model with countvectorizer. Recall score for "-1(Anti)" and "0(Neutral)" are pretty low meaning there are too many false negative on these two classifications.

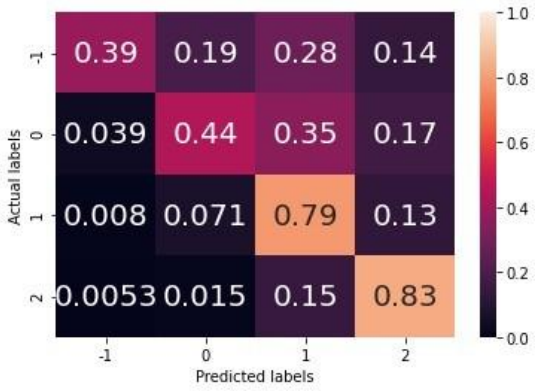

Figure 5.7: Confusion matrix

From the confusion plot, it shows the same issues as in the classification report table: the multinomial naive bayes model makes much better prediction for label "1(Pro)" and "2(News)". The performance for the label "-1(Anti)" and "0(Neutral)" is not as well as expected.

5.2.2 Multinomial Naive Bayes Classifier with TF-IDF

	precision	recall	f1-score	support
-1	0.92	0.19	0.32	402
0	0.72	0.25	0.37	803
1	0.65	0.95	0.77	2251
2	0.82	0.62	0.71	939
accuracy			0.68	4395
macro avg	0.78	0.50	0.54	4395
weighted avg	0.72	0.68	0.64	4395

Figure 5.8: Classification report

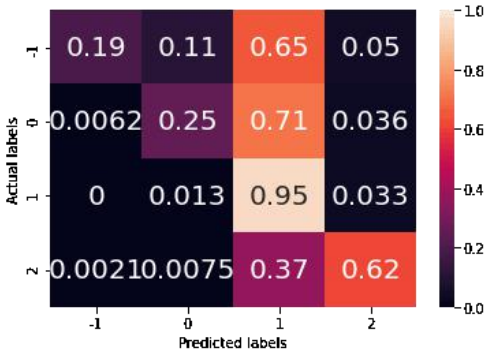

Figure 5 .9: Confusion matrix

Unlike the comparison between TF-IDF an countvectorizer with the random forest classifier, this time the TF-IDF with multinomial naive bayes classifier make less accurate prediction compare to the model with countvectorizer. The classification accuracy is 0.68 and the model performance for four labels are pretty skewed. Model makes much better prediction for label "1(Pro)" and fair prediction for "2(News)". As for label "-1(Anti)" and "0(Neutral)", the performance is not good.

5.3 Logistic Regression

For the logistic regression, I also set the class_weight='balanced' to reduce the effect of unbalanced data.

5.3.1 Logistic Regression with countvectorizer

	precision	recall	f1-score	support
-1	0.72	0.55	0.62	402
0	0.60	0.58	0.59	803
1	0.80	0.82	0.81	2251
2	0.76	0.81	0.78	939
accuracy			0.75	4395
macro avg	0.72	0.69	0.70	4395
weighted avg	0.75	0.75	0.75	4395

Figure 5 .10: Classification report

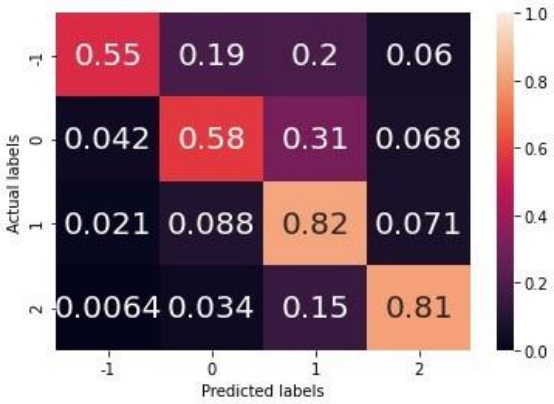

Figure 5.11: Confusion matrix

From the classification report and confusion matrix, the logistic regression with countvectorizer makes pretty good prediction performance. The overall classification accuracy is 0.75 which is the highest score I have so far. The f1-scores are also relatively balanced compared to the previous results. Recall scores for label "1(Pro)" and "2(News)" are still much higher than other two, which means this model still makes many false negative for "-1(Anti)" and "0(Neutral)".

5.3.2 Logistic Regression with TF-IDF

	precision	recall	f1-score	support
-1	0.58	0.67	0.62	402
0	0.53	0.61	0.57	803
1	0.85	0.71	0.78	2251
2	0.70	0.84	0.76	939
accuracy			0.72	4395
macro avg	0.67	0.71	0.68	4395
weighted avg	0.74	0.72	0.72	4395

Figure 5.12: Classification table

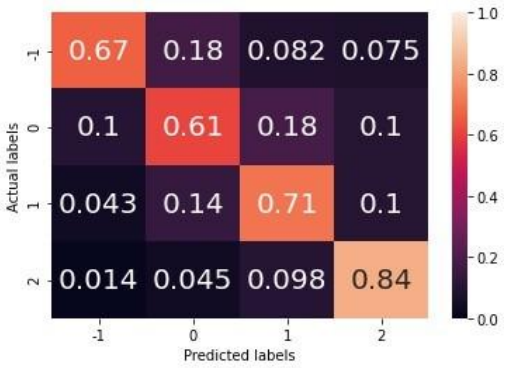

Figure 5.13: Confusion matrix

As we can see from the classification report and confusion matrix, the classification accuracy of model on testing group is 0.72, which is lower than the accuracy score for the same machine learning model but with the countvectorizer vectors. This model also makes high recall scores for label "1(Pro)" and label "2(News)". However, compare to the previous models, the recall scores for label "-1(Anti)" and "0(Neutral)" have improved a lot.

CHAPTER 6

Conclusion

The goal of this thesis is to use two different vectorization methods(TF-IDF and Countvectorizer) on three machine learning algorithms(Random Forest, Naive Bayes, Logistic Regression) to get six predictions of sentiment classification on tweets related to the global warming or climate change topics.

Firstly, according to the EDA, I get better understanding of the data. Secondly, with the word vectorizers, I transfer the same textual messages into two different group of vectors: one is vectorized by TF-IDF and the other one is vectorized by countvectorizer. By applying three different machine learning algorithms, we finally get six model results.

When we analyzing the results based on the different vectorizers, I find it is hard to judge which vectorizer is better fit for the data. This is because that for random forest classifier, the model with TF-IDF vectors has much better performance than model with countvectorizer. However, for logistic regression and multinomial naive bayes, the models with countvectorizer have better performance. Hence, it is really hard to say which vectorizer is better fit for this data based on the evidence we have yet.

When we analyzing the results based on the different machine learning algorithms, it is no doubt that logistic regression makes the best prediction performance among three machine learning algorithms. For both classification accuracy score, f1 score, precision and recall score, the logistic regression models have better performance than other machine learning techniques do.

There is also some drawbacks of this data set which I could improve in the future. For example, the data set is unbalanced. For the sentiment part, there are too many more “1(Pro)” sentiment compare to others, which lead to unbalanced model training result. How to solve the model unbalanced is a potential topic to continue studying on.

References

[1] Ane Berasategi, "*Overview of tokenization algorithms in NLP, Introduction to tokenization methods, including subword, BPE, WordPiece and SentencePiece*", Aug 2020

[2] Vijayarani Mohan, "*Text Mining: Open Source Tokenization Tools: An Analysis*", Advanced Computational Intelligence: An International Journal (ACII), Vol.3, No.1, January 2016

[3] Hunter Heidenreich, "*Natural Language Processing: Count Vectorization with scikit-learn*", Aug 2018

[4] Akiko Aizawa, "*An information-theoretic perspective of tf–idf measures*", Information Processing and Management 39 (2003) 45–65

[5] Juan Ramos, "*Using TF-IDF to Determine Word Relevance in Document Queries* ", 2003

[6]Marius Borcan, "*TF-IDF Explained And Python Sklearn Implementation*", Jun 2020

[7]Frederick Livingston, "*Implementation of Breiman's Random Forest Machine Learning*

Algorithm", ECE591Q Machine Learning Journal Paper. Fall 2005.

[8]Leo Breiman, *"RANDOM FORESTS",* Jan 2001

[9]Li Zhao, Minlie Huang, Ziyu Yao, Rongwei Su*, Yingying Jiang*, Xiaoyan Zhu, "*Semi-Supervised Multinomial Naive Bayes for Text Classification by Leveraging Word-Level Statistical Constraint* ", Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

[10] Akinori Fujino, Hideki Isozaki, and Jun Suzuki, "*Multi-label Text Categorization with Model Combination based on F1-score Maximization",* NTT Communication

Science Laboratories NTT Corporation 2-4, Hikaridai, Seika-cho, Soraku-gun, Kyoto, Japan 619-0237

[11]Sergio A. Alvarez, "*An exact analytical relation among recall, precision, and classification accuracy in information retrieval",* Boston College, 2002