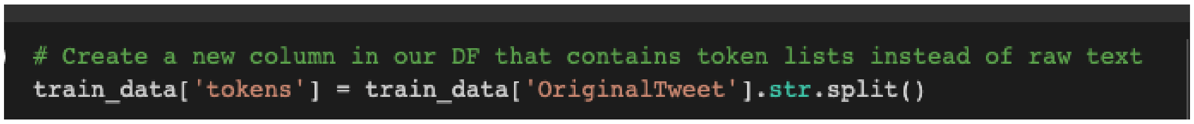
JSC270 Assignment 4

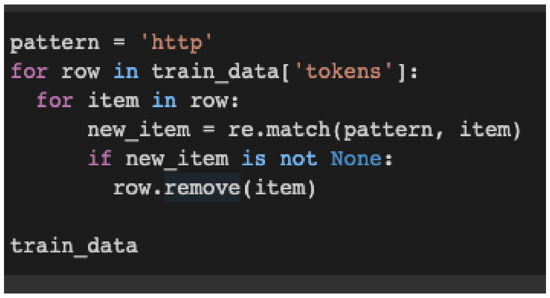
Author: Hongshuo Zhou, Chun Yin Yan
Contribution: All the work was done jointly with equal distribution.

Github link:
https://github.com/TheTonyZhou/JSC270-A4.git

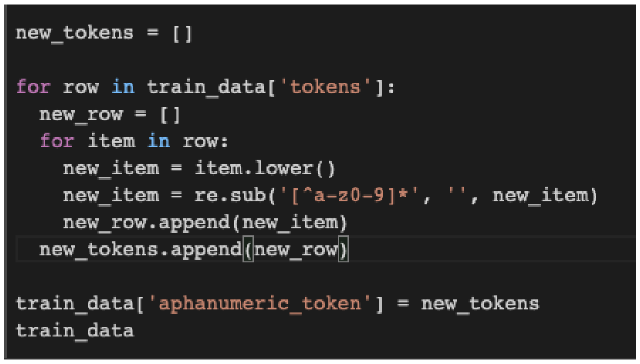
Part 1:
A:
There are 15397 negative observations, 7712 neutral observations, and 18042 positive observations in the training set. Thus, in the training data set, 48.84% are positive observations, 37.42% are negative observations, and 18.72% are neutral observations, which suggests the data is not well balanced.

B:


```
# Create a new column in our DF that contains token lists instead of raw text
train_data['tokens'] = train_data['OriginalTweet'].str.split()
```

C:


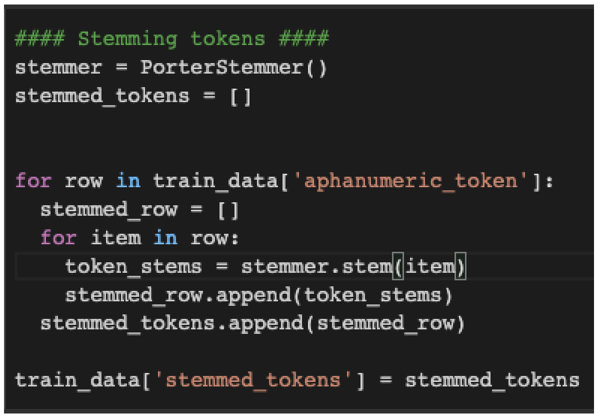
```
pattern = 'http'
for row in train_data['tokens']:
  for item in row:
      new_item = re.match(pattern, item)
      if new_item is not None:
        row.remove(item)

train_data
```

D:


```
new_tokens = []

for row in train_data['tokens']:
  new_row = []
  for item in row:
    new_item = item.lower()
    new_item = re.sub('[^a-z0-9]*', '', new_item)
    new_row.append(new_item)
  new_tokens.append(new_row)

train_data['aphanumeric_token'] = new_tokens
train_data
```

A scenario where we should keep some forms of punctuation is that at the end of a sentence, a question mark or exclamation mark can be helpful because they can suggest the sentiment of the text. So, it should be better to keep the punctuation in this case.

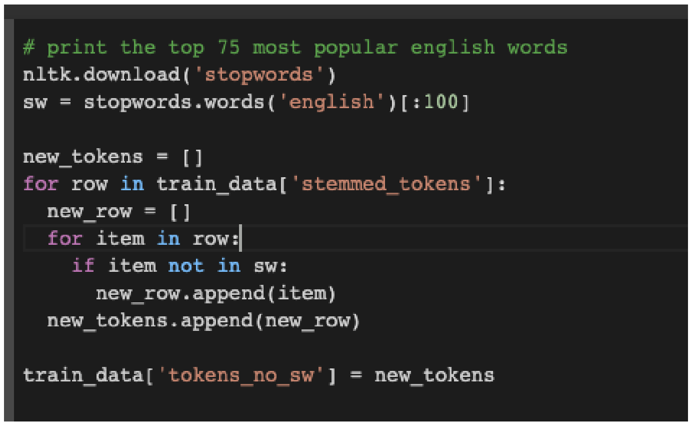
E:



```python
#### Stemming tokens ####
stemmer = PorterStemmer()
stemmed_tokens = []


for row in train_data['aphanumeric_token']:
  stemmed_row = []
  for item in row:
    token_stems = stemmer.stem(item)
    stemmed_row.append(token_stems)
  stemmed_tokens.append(stemmed_row)

train_data['stemmed_tokens'] = stemmed_tokens
```

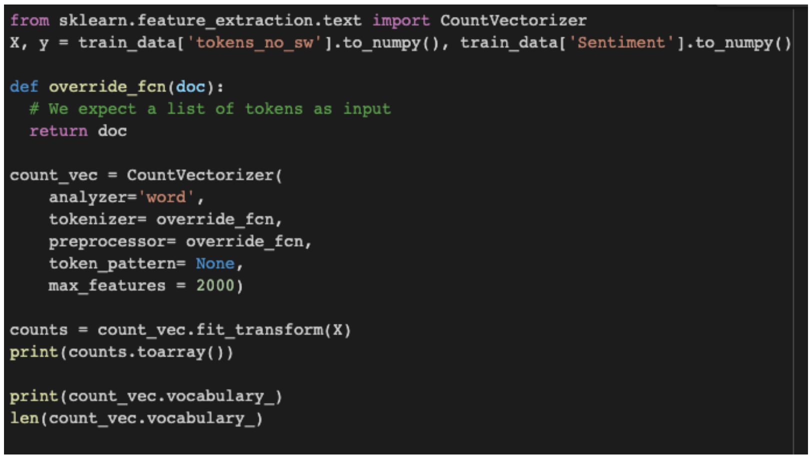
The stemmer we used is PorterStemmer.

F:



```python
# print the top 75 most popular english words
nltk.download('stopwords')
sw = stopwords.words('english')[:100]

new_tokens = []
for row in train_data['stemmed_tokens']:
  new_row = []
  for item in row:
    if item not in sw:
      new_row.append(item)
  new_tokens.append(new_row)

train_data['tokens_no_sw'] = new_tokens
```

G:

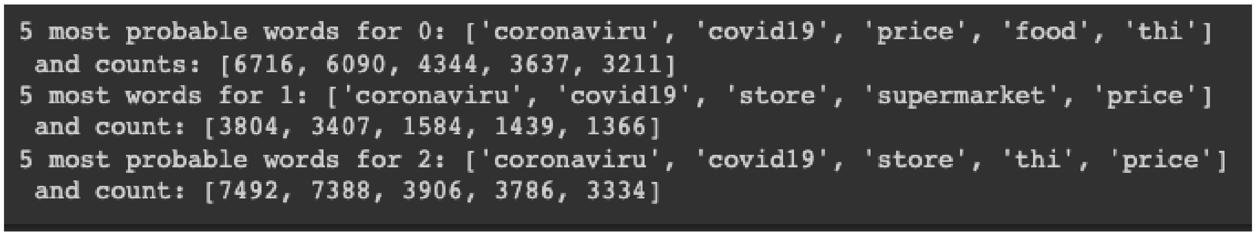

```
from sklearn.feature_extraction.text import CountVectorizer
X, y = train_data['tokens_no_sw'].to_numpy(), train_data['Sentiment'].to_numpy()

def override_fcn(doc):
  # We expect a list of tokens as input
  return doc

count_vec = CountVectorizer(
    analyzer='word',
    tokenizer= override_fcn,
    preprocessor= override_fcn,
    token_pattern= None,
    max_features = 2000)

counts = count_vec.fit_transform(X)
print(counts.toarray())

print(count_vec.vocabulary_)
len(count_vec.vocabulary_)
```

The total length of our vocabulary is 52572, but we only used the top 2000 to save space and running time.

H:

The code is too long so please refer to the google colab. The training accuracy of this model is 0.71.20 and testing accuracy is 0.3873.

Top frequency words in each class:


```
5 most probable words for 0: ['coronaviru', 'covid19', 'price', 'food', 'thi']
 and counts: [6716, 6090, 4344, 3637, 3211]
5 most words for 1: ['coronaviru', 'covid19', 'store', 'supermarket', 'price']
 and count: [3804, 3407, 1584, 1439, 1366]
5 most probable words for 2: ['coronaviru', 'covid19', 'store', 'thi', 'price']
 and count: [7492, 7388, 3906, 3786, 3334]
```

I:

I think we should not fit an ROC curve in this scenario because we have three classes. However, the ROC curve can only be fitted between three classes. If one really wants to fit a ROC curve, they can use the approach like one class versus rest classes, but it is not very informative.

J:

The training accuracy is 0.6920 and the testing accuracy is 0.4104. Compared to count vectors, the training accuracy decreases but the testing accuracy increases. However, the overall changes are relatively small so the performance of the two are roughly the same.

K:

The training accuracy is 0.6889 and the testing accuracy is 0.4120. Compared to stemming, the training accuracy decreases but the testing accuracy increases. However, the overall changes are relatively small so the performance of the two are roughly the same.

Bonus:
Naive Bayes models are generative because they are based on the joint probability of the classes to build our model, which is a key feature of generative models.

Sentiment analysis of people's opinion on Amber Alert using Twitter API

01 April 2022

Chun Yin Yan, Hongshuo Zhou

Problem Description and Motivation:

An Amber Alert is a message which informs the public that a child has been abducted[1]. In Canada, when an Amber Alert is received on a mobile device, a loud sound will be played, the device will vibrate, and a message will appear on the screen. The message includes information of the abducted child and the last seen location.

Some mobile devices do not have the ability to turn off Amber Alerts or prevent themselves from ringing when it receives an Amber Alert. This sparked controversies about how Amber Alert are delivered. Some argued that the ringtone of the Alert disrupts people's activities, such as being distracted or shocked when driving, which could cause lethal accidents. Some argued that the Amber Alert system is sometimes faulty, and messages are delayed or from other regions. There is any related research on the public's opinion of the Amber Alert system, so this is a difficult problem which has not been solved or researched into.

We would like to investigate what the public think about the Amber Alert system. Also, we want to identify which Tweets on Twitter are informational and which are opinion-based regarding Amber Alert. To answer this question, we used Tweepy, a Python library, to extract relevant Tweets, performed data analysis, and built several machine learning models which helped us answer these two questions.

[1] https://en.wikipedia.org/wiki/Amber_alert

Data Description:

Using Tweepy, we extracted seven days of Tweets which mentioned “Amber Alert” from 2022-03-25 to 2022-04-01. For the extraction parameters used in the API, our search words are “Amber Alert” and we set the language to English, so we only extracted English tweets which contained the words “Amber Alert”. Retweets without comments are filtered out and removed, as they do not provide additional information about the stance of the Retweeters. The dataset contains 2837 observations and 2 features: an observation from the data collected contains a datetime field (e.g. 25/3/2022 23:23) and a full_tweet field, which contains the whole tweet including URLs and hashtags.

The strength of our dataset is that these Tweets are highly relevant to Amber Alert. The limitation of our dataset is that it only includes Tweets which explicitly mentioned the phrase “Amber Alert”, and it doesn’t capture any comments on these Tweets or any Retweets. Moreover, Tweets are collected over seven days only, which contain similar words (e.g. Tennessee) since Amber Alerts and the Tweets have similar messages.

Exploratory data analysis:

We manually labelled 1223 Tweets from 2022-03-25 to 2022-03-27 to represent the sentiment of the Tweet. A value of ‘1’ means the Tweet contains a neutral opinion on the Amber Alert system or is purely informational, while a ‘0’ means the Tweet has a negative sentiment and a ‘2’ means the Tweet has a positive sentiment towards Amber Alert.

Upon inspecting the Tweets in our dataset, we have discovered several interesting findings. First, “Amber Alert” in some Tweets does not mean the message regarding child abduction emergencies. In some contexts, “Amber Alert” means a weather warning signal which warns people of extreme weather events. Second, more than half of the Tweets collected have a sentiment score of ‘1’, while the rest of the Tweets mostly contain negative sentiments. We expected this to be the case because the authority uses Twitter as a platform to spread information about Amber Alert, and people are likely to complain about their inconvenience more than complimenting the system when it works[2]. Lastly, the Tweets which are in favour of Amber Alert in our labelled dataset often show support towards the Amber Alert by using sarcasm to mock people who are against Amber Alerts. An example is: “Why are people complaining about the Amber Alert?”.

Before we trained our machine learning models, we cleaned the data by carrying out these steps in the following order: remove replies (@user), URLs (https://...) and hashtags (#AmberAlert) from the Tweets, removed non-alphanumeric characters and punctuations, turn all alphabets to lowercase, tokenized the cleaned Tweets, and stop words from tokens, and stemmed the tokens using Count Vectorizer and TF-IDF Vectorizer.

Machine learning model description

The machine learning models we fitted to predict the sentiment of the Tweets are: Naive Bayes Classifier, Multiclass Logistic Regression, Support Vector Machine, and Linear Discriminant. These four machine learning models are all classification models which help us to classify the Tweets into the three sentiment categories.

All these four models are supervised machine learning models. The properties, strengths and weaknesses of these four models are summarized below:

ML Models	Type of Model	Strengths	Weaknesses
Naive Bayes Classifier	Generative, Probabilistic	Simple, Efficient for Text Classification	Assumed words in Tweets are independent

[2] https://goldfayn.com/why-customers-complain-but-do-not-compliment/#:~:text=People%20naturally%20tend%20to%20complain,as%20a%20waste%20of%20time.

Multiclass Logistic Regression	Discriminative, Probabilistic	Gives measurements of the influences of predictors[3]	Assumed sentiment of Tweets have a linear relationship with Vectorized Matrices[4]
Support Vector Machine	Discriminative, Non-probabilistic	Memory Efficient[5]	Low performance with imbalanced Twitter data[6]
Linear Discriminant Analysis	Generative, Probabilistic	Highly efficient even when we have many Tweets.	Assumed Normality of Tweets, Sensitive to Outliers[7]

The baseline model chosen is Naïve Bayes Classifier, since it is the simplest model among these four and usually has good performance. We used these four models because they are all similar classification models which have similar properties (see the second column of the table above), which let us closely compare their performance during model fitting. Since we are interested in classifying Tweets into the three sentiment types, the performance metric of interest is the accuracy of the model.

Results and conclusion

We fitted four machine learning models to predict the sentiment of the Tweets. The table below summarizes the properties and performance of the four models:

ML Model	Training Accuracy	Test Accuracy
Naive Bayes Classifier	0.837	0.746
Multiclass Logistic Regression	0.897	0.738
Support Vector Machine	0.964	0.746
Linear Discriminant Analysis	0.989	0.632

[3] https://www.i2tutorials.com/what-are-the-advantages-and-disadvantages-of-logistic-regression/
[4] https://medium.com/analytics-vidhya/pros-and-cons-of-popular-supervised-learning-algorithms-d5b3b75d9218
[5] https://dhirajkumarblog.medium.com/top-4-advantages-and-disadvantages-of-support-vector-machine-or-svm-a3c06a2b107
[6] https://towardsdatascience.com/support-vector-machines-imbalanced-data-feb3ecffbb0e
[7] https://www.knowledgehut.com/blog/data-science/linear-discriminant-analysis-for-machine-learning

From above, we can see that all four models have very excellent train accuracy, and the test accuracy is moderately lower than the training accuracy. Moreover, we observe that the training accuracy of the support vector machine and linear discriminant analysis model both are very close to 100%, which we suspect is an indication of overfitting.

There are several ways we can increase the performance of these models and reduce the chance of overfitting. First, we can collect more data over a longer period of time, so that the Tweets collected have more diverse messages. Since our dataset is collected over a week, we found that some Tweets have extremely similar messages since the Amber Alert is highly specific to location, that is, the age of the child and the area is repeated in many Tweets. Alternatively, we can remove certain tokens such as locations (e.g. Tennessee) to reduce the bias in our data. This can potentially solve the overfitting issue suspected in the LDA and SVM models. Lastly, it would be a good idea to make our training dataset to be more balance in terms of the distribution of the sentiments. This could reduce bias towards the largest class, in this case, the neutral sentiment.

In conclusion, the Naïve Bayes Classifier and Multiclass Logistic Regression both offer very fine predictive power for identifying the sentiment of Tweets about Amber Alerts. Moreover, based on the count of tokens and our manual inspection many Tweets with negative sentiment, it is observed that most people are frightened by the sound made by their mobile devices when they received an Amber Alert, especially late at night or when driving. We think that the authority should take into consideration of the time and location to send out Amber Alerts. Alternatively, Amber Alerts should be able to be silent during certain periods of the day, such as during driving or after midnight.

References:

1. https://en.wikipedia.org/wiki/Amber_alert
2. https://goldfayn.com/why-customers-complain-but-do-not-compliment/#:~:text=People%20naturally%20tend%20to%20complain,as%20a%20waste%20of%20time.
3. https://www.i2tutorials.com/what-are-the-advantages-and-disadvantages-of-logistic-regression/
4. https://medium.com/analytics-vidhya/pros-and-cons-of-popular-supervised-learning-algorithms-d5b3b75d9218
5. https://dhirajkumarblog.medium.com/top-4-advantages-and-disadvantages-of-support-vector-machine-or-svm-a3c06a2b107
6. https://towardsdatascience.com/support-vector-machines-imbalanced-data-feb3ecffbb0e
7. https://www.knowledgehut.com/blog/data-science/linear-discriminant-analysis-for-machine-learning