Create Sentiment Analysis Model of English-Language Twitter Tweets Concerning COVID-19 During the Course of April 2020

Problem Statement:

This capstone project aims to create a sentiment analysis model from the English-language, coronavirus disease 2019 (COVID-19) Tweets to predict sentiment towards the circumstances of the pandemic during the course of April 2020. The sentiment in this project will be categorized as positive, neutral, and negative. The primary clients this project targets are those that work in or deal with mental health and social analysis. Particularly, these clients would want to see how the social circumstances surrounding the COVID-19 pandemic and its lockdowns affect the social-welfare and mental health of the populace. COVID-19 is an infectious disease caused by the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) that primarily affects the lungs. However, it was discovered that it can also negatively affect other organs like the heart, kidneys, and brain, causing a multitude of other health complications that can leave a lasting impact even after recovery from the initial infection. It was first identified in Wuhan, China on December 2019 and has since spread worldwide, causing long-term lockdowns in many parts of the world as early as March 9, 2020. Besides taking a toll on medical health, it is indicated that the COVID-19 pandemic is also taking a toll on mental and social health, especially due to the stay-at-home orders. Hence, this project could give mental healthcare workers and sociologists insights on how the COVID-19 pandemic situation is affecting the mental health of the populace who are heavily limited in their movements and social interactions to prevent the spread of the virus. Also, the resulting model of this project could give indications on how resources of psychological care could be distributed and utilized to mitigate any negative effects long-term pandemics and lockdowns may have on emotional health.

The datasets that were used for this project were created by Shane Smith (https://www.kaggle.com/smid80) and posted on Kaggle.com under the CC0: Public Domain. They are available for download via the following links:

- Main Page: https://www.kaggle.com/smid80/coronavirus-covid19-tweets
- Early April: https://www.kaggle.com/smid80/coronavirus-covid19-tweets-early-april
- Late April: https://www.kaggle.com/smid80/coronavirus-covid19-tweets-late-april

Dataset Description:

It was decided that a supervised, classification prediction model using NLP will be created to predict whether the sentiment of a Tweet will be determined to be positive, negative, or neutral. However, as the original data does not have the sentiment labels, the sentiment analysis model will be trained with the pre-labeled corpus of Twitter samples from the Natural Language Toolkit (NLTK) Python library to predict and create sentiment labels for the original data. To create and clean the df_tweets.csv file for data exploration and for the main analysis of this project, multiple csv files pertaining to each day from March 29, 2020 to April 30, 2020 were concatenated together. This section describes the data wrangling and cleaning steps and methods used to create the df_tweets.csv file.

Using Python 3 in IPython Notebook, the multiple csv files containing tweets pertaining to the COVID-19 pandemic during the month of April 2020 were concatenated together via the glob module into one pandas dataframe denoted as df_tweets which contained 22 columns and

14,607,013 rows. Rows in which the Tweet text were not in the English language were dropped via a Boolean array. Also, columns in the df_tweets dataframe were removed by a Boolean array based on a 60% missing (null) percentage criteria as well as whether or not the columns would contribute to the data exploration and the main purpose of this project. It was decided that status_id (the ID of the actual Tweet) and user_id (The ID of the user account that Tweeted) would be dropped while the other variables were kept. This left the df_tweets dataframe with 13 columns and 8,133,785 rows. Due to the limited computational power of the hardware used to analyze the data for this project, 90% of the rows were randomly dropped from the df_tweets dataframe. This left the dataframe with 13 columns and 813,378 rows.

Text cleaning was done on the text column to prepare the df_tweets dataframe for the NLP analysis of the project. First, a function called, emoticon_text, was created to convert emoticons into their respective texts via a custom dictionary containing commonly used emoticons with their respective textual meanings (ex. :) : happy / smile, xD : laugh, :(: frown / sad / pouting, etc.) before it was applied on the text column with the new column called, cleaned_text, which will contain the cleaned text of the Tweets. Next, a function called, replace_contractions, was written to convert contractions in the text of the cleaned_text column into their basic words (ex. don't : do not, hadn't : had not, hadn't've : had not have, etc.). Then, a function called, tweet_cleaner, was made to remove url links and Twitter handles (@user), convert emojis (ex. 😄, 😆, 😍, etc.) into their respective text, keep only alphabetical words, remove whitespaces, and convert the text to lowercase under the cleaned_text column. Additionally, functions called, lemmatize_text and stem_text, were written to help to lemmatize and stem the words of the text under the cleaned_text column. Finally, the remove_stopwords function was created to remove the stop words of the text under the cleaned_text column. After applying the text cleaning funtions, the rows containing null values of the cleaned_text column were removed from the df_final dataframe. The final df_tweets dataframe has 14 columns and 813,378 rows.

The final df_tweets dataframe was exported as a CSV file named, df_tweets.csv.

Initial Statistical Findings:

According to Figure 1 and 2 below, the word frequency plots of both the text and cleaned_text columns of the df_tweets dataframe show how well the text cleaning methods worked in removing url links, Twitter handles, special characters, and stop words of the text as well as lemmatizing and stemming them. Also, Figure 2 shows how many more meaningful words appear in the word frequency count of the text from the text column after the text cleaning methods were applied to it.

Even though this project is an NLP analysis, additional exploratory data analysis was done to see how other features of the df_tweets dataset might affect the sentiment of the COVID-19 Tweets once the sentiment labels were made for the Tweets later in the project. Verified Twitter accounts are accounts that received the blue verified badge on Twitter, letting people know that an account of public interest is authentic. According to Figure 3 below, there are far less verified Twitter accounts than unverified Twitter accounts in the df_tweets dataset. However, according to Figure 4 and 5 below, verified Twitter accounts have more followers and friends than unverified Twitter accounts, especially for number of followers. Thus, there is a possibility that sentiment might follow the verified Twitter accounts as they contain more

Figure 1: 100 Most Frequent Words of “text” in df_tweets

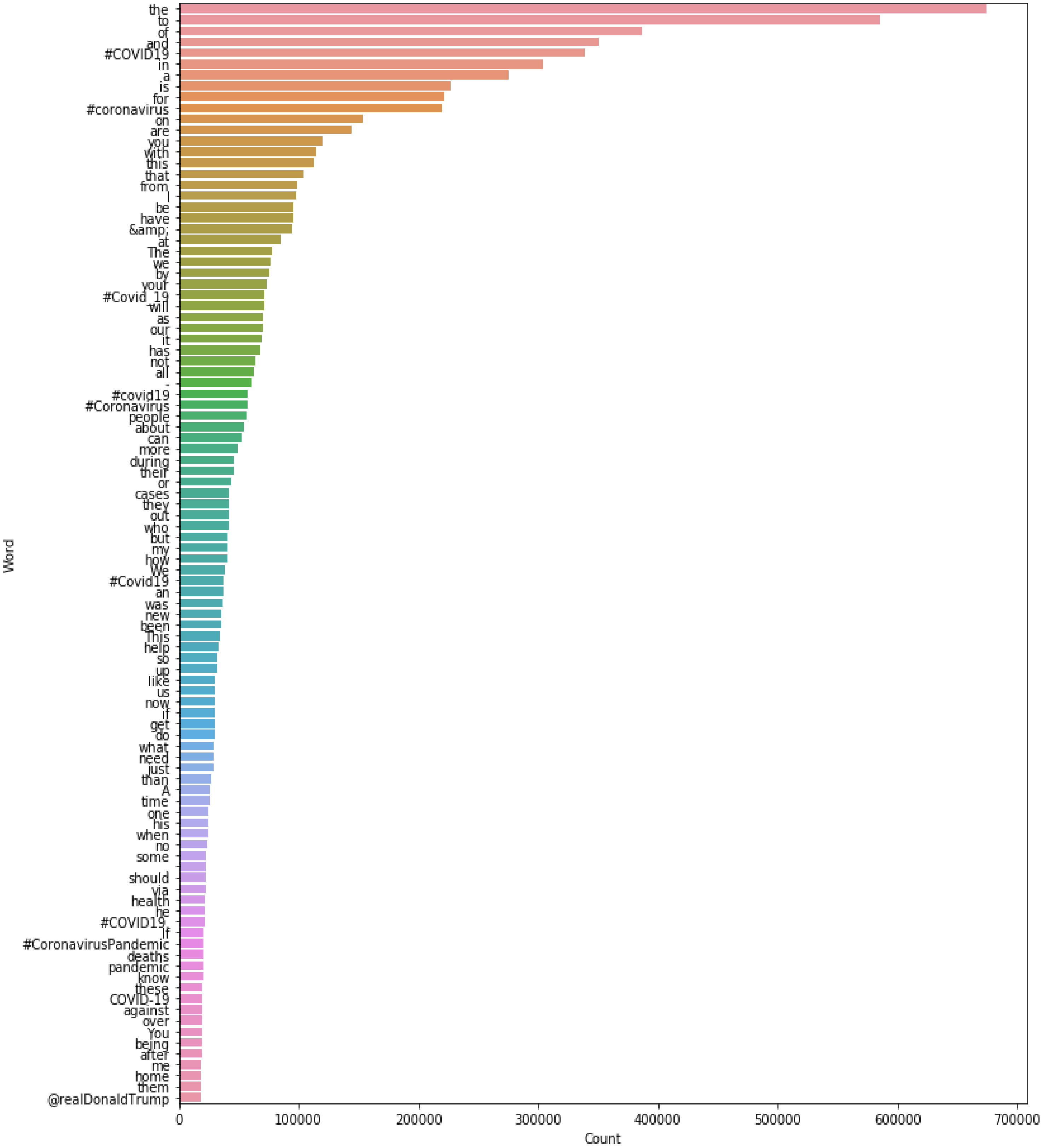


Figure 2: 100 Most Frequent Words of “cleaned_text” in df_tweets

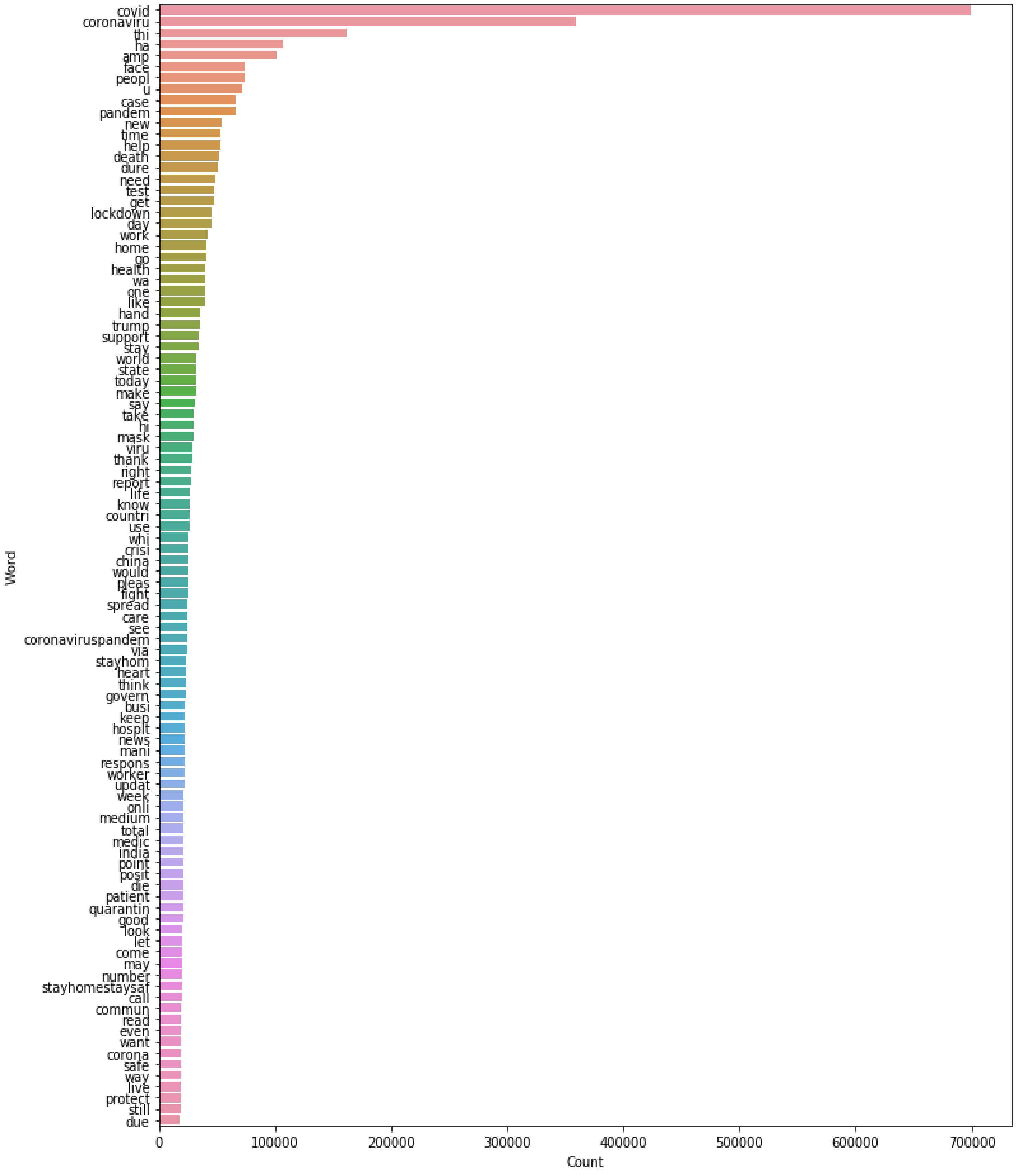


Figure 3:

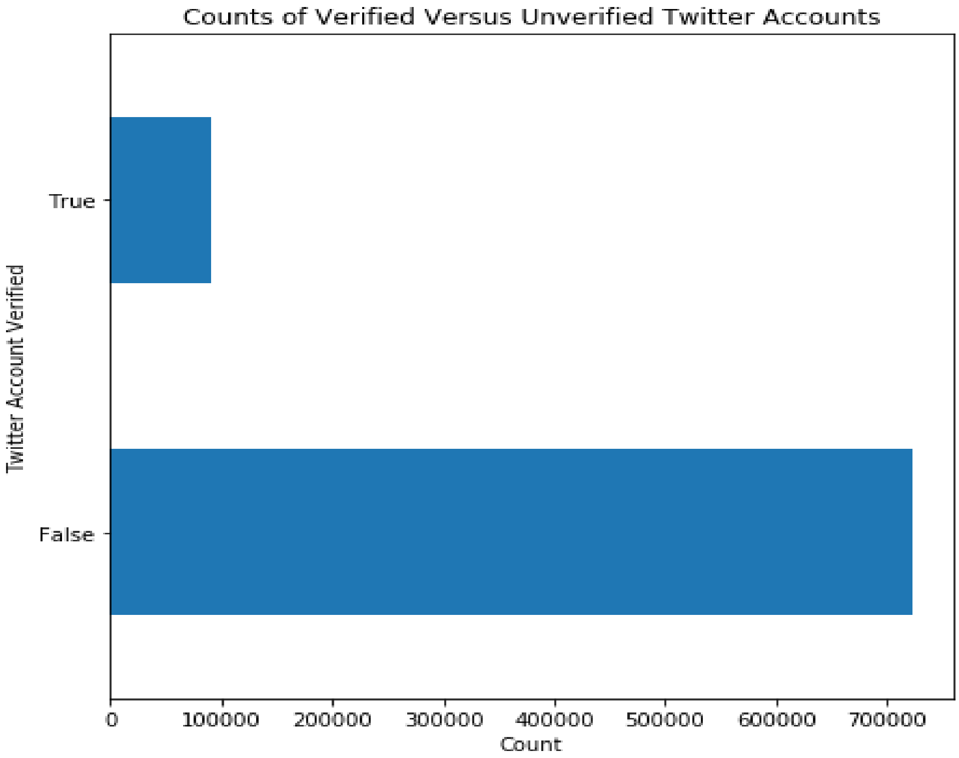


Figure 4:

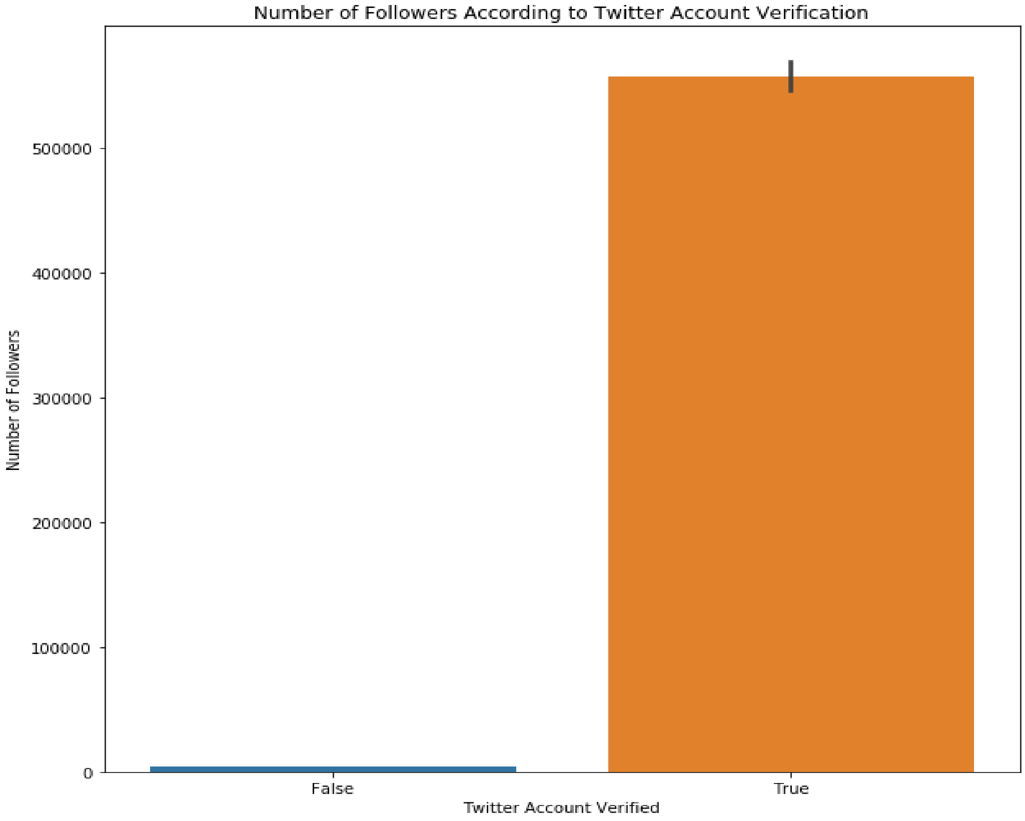
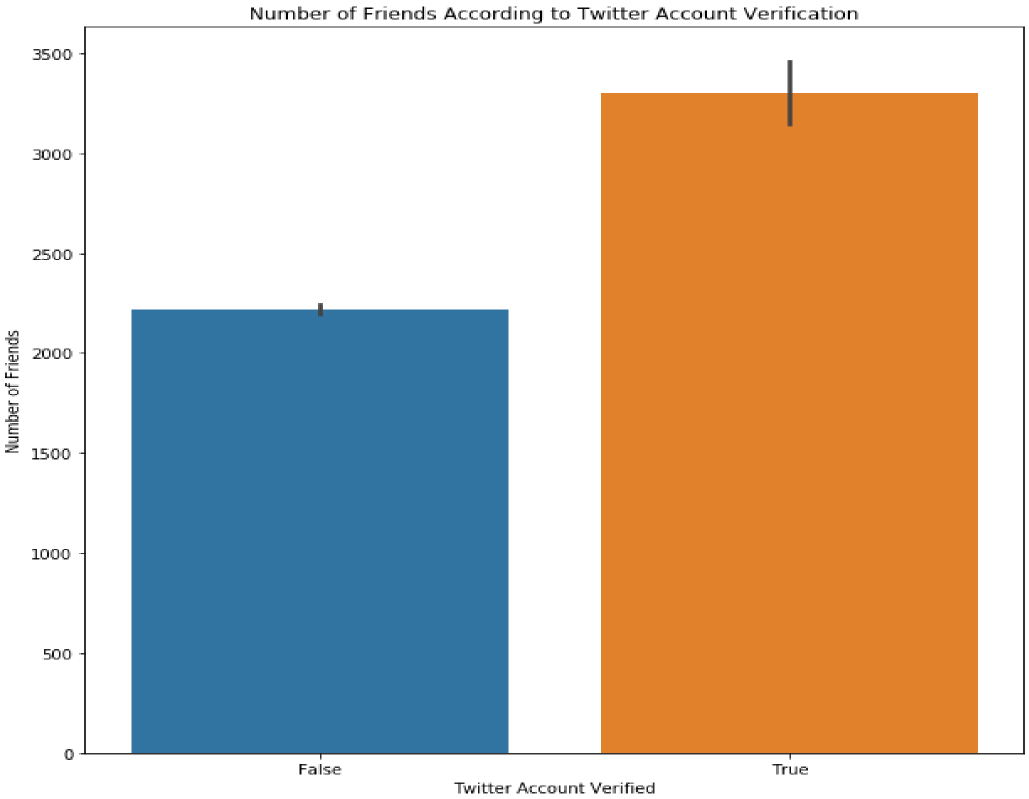


Figure 5:



numbers of followers and friends than the unverified Twitter accounts. Based on the scatterplots of Figure 6 and 7 below, the date in which the Twitter account was created does somewhat have an effect on the number of friends and followers as the higher count points tend to correlate with the older Twitter accounts. However, those are only a small number of data points, and a large number of the data points do not show any correlation with the age of the Twitter account. There is one noticeable outlier in the scatterplots in which a Twitter account was created in the year of 1970. This is supposed to be impossible as Twitter was launched on July 15, 2006. It can be assumed that this outlier was caused by a technical error and thus can be ignored.

To further explore the data, an unsupervised learning algorithm was used to analyze if there are any distinct classes in the text under the cleaned_text column in the dataset of df_tweets. The algorithm that was used was K-Means clustering since it can handle big data well due to its linear time complexity. Due to limitations of the RAM memory space of the computing computer, 97.5% of rows were randomly dropped from the df_tweets dataset which left only 20,334 of its observations to be analyzed under K-Means clustering. Below documents how this inferential statistics technique was used to analyze the data of df_tweets and the inferences made based on the results.

In order to perform K-Means clustering on the cleaned text under the cleaned_text column of the df_tweets dataframe, TF-IDF (term frequency–inverse document frequency) values were computed to vectorize the text of the cleaned_text column in order to create the features and X matrices. Then, the K-Means clustering algorithm was applied to the vectorized

Figure 6:

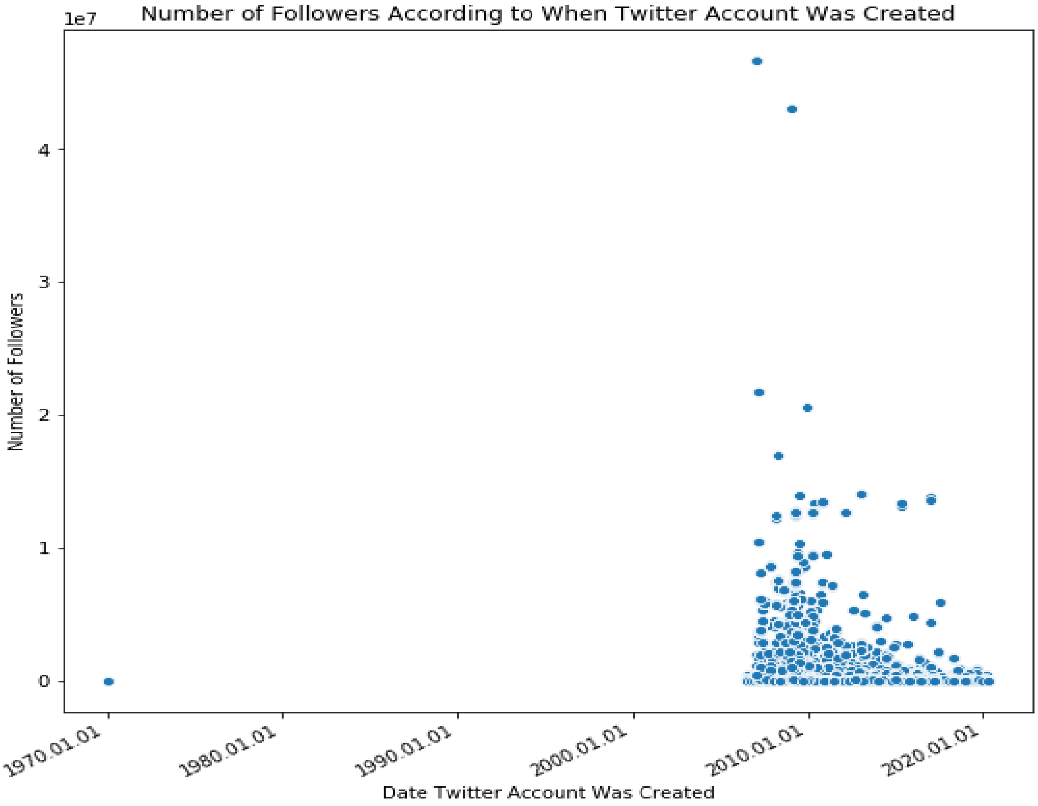
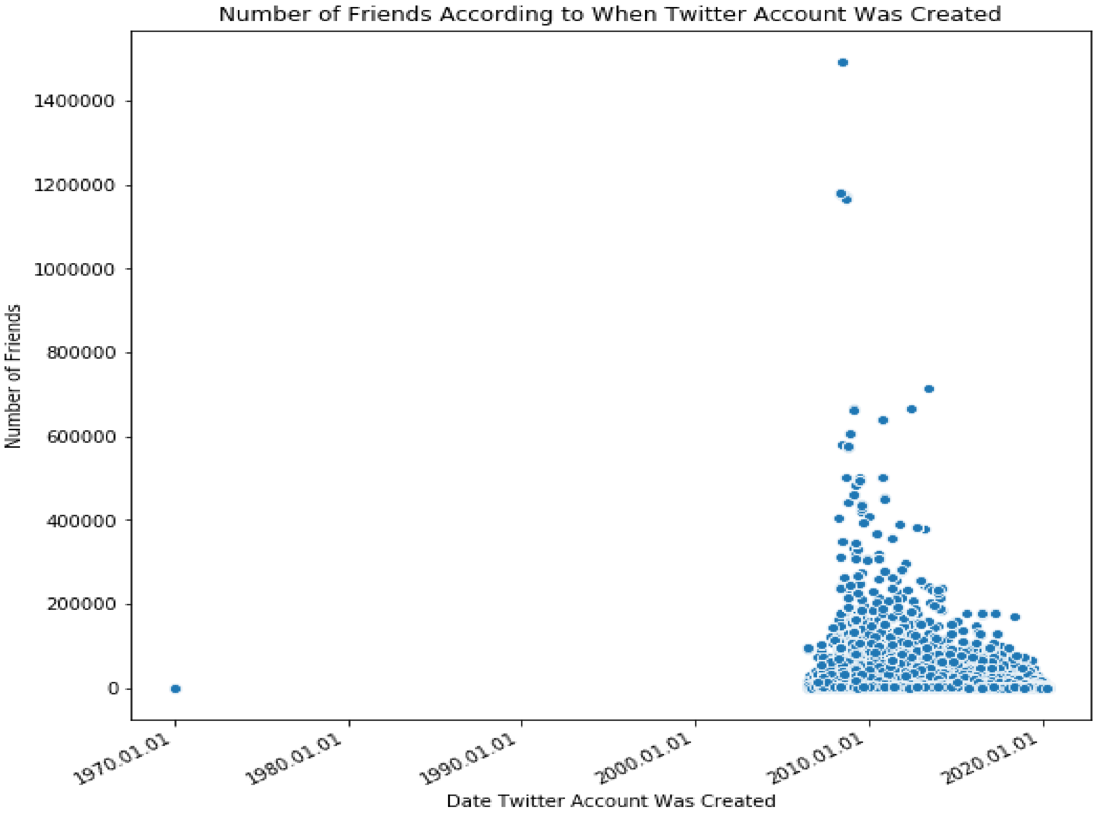


Figure 7:



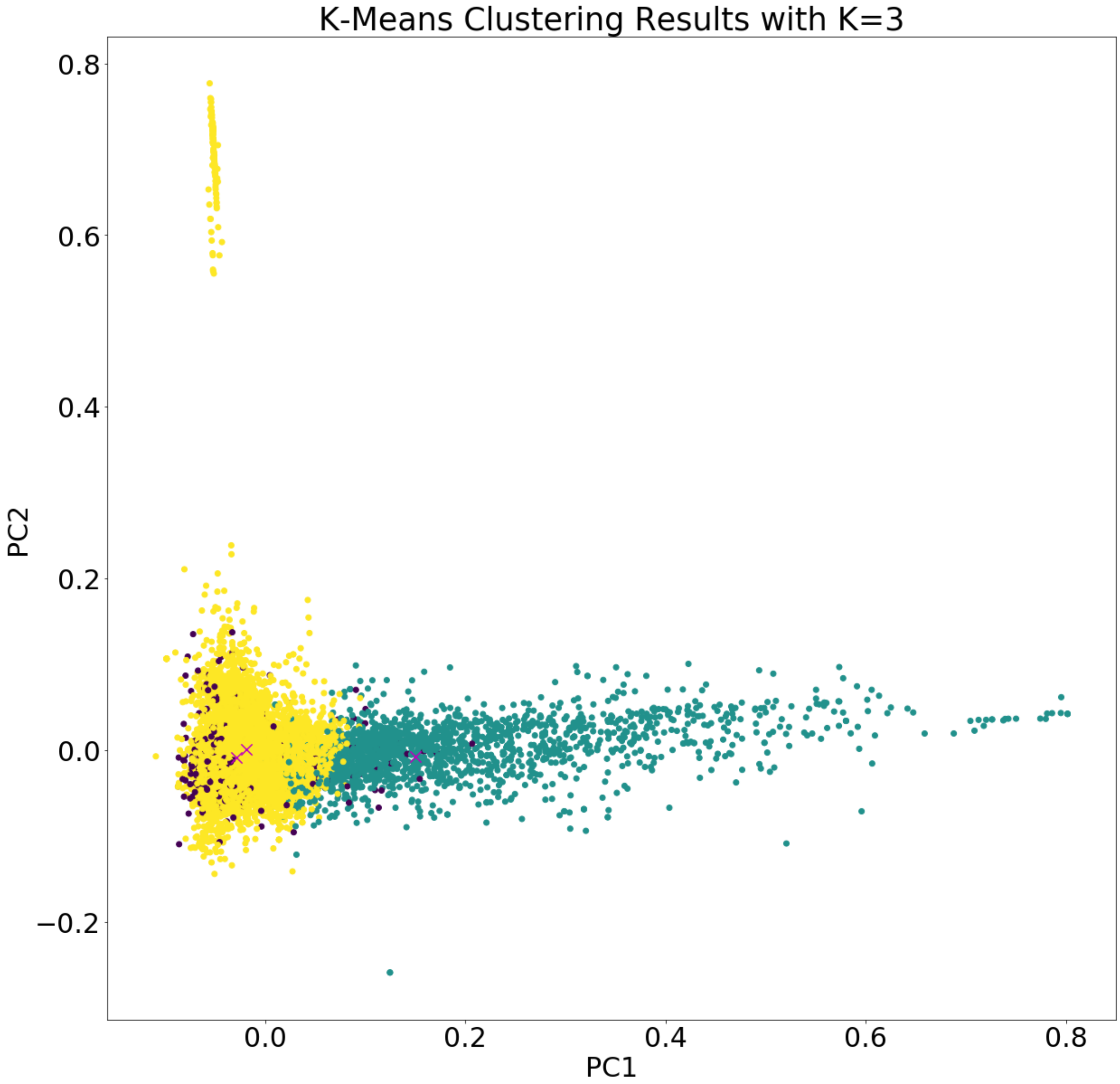
text. The number of clusters in the algorithm was set to 3 since there are 3 sentiment classes to be analyzed for this project. The centroids and features of the K-Means clustering analysis were also calculated. Table 1 below shows which clusters the centroids, based on the words of the text in the cleaned_text column, belong to.

Table 1: Centroids in the Three K-Means Clusters

Cluster	Centroids
0	covid thi amp ha face peopl help dure time pandem
1	coronaviru covid thi ha pandem peopl trump lockdown test amp
2	case death total new covid report coronaviru confirm deliv number

The K-Means clustering plot in Figure 8 was made by creating an instance of K-Means and then using the PCA function in Python 3 to reduce the calculated features and the cluster centers to 2D. According to Figure 8 below, the scatterplot shows the points overlap instead of forming into three distinct clusters. As the text under the cleaned_text column in the df_tweets dataset is unlabeled for sentiment, the Silhouette Coefficient is used to check the clustering model in Figure 8. The best value of the Silhouette Coefficient is 1 while the worst value is -1, and values near 0 indicate overlapping clusters. As the Silhouette Coefficient was calculated to be about 0.0017, the clusters very much overlap rather than being distinct from each other. The calculations of K-Means so far indicate that the cleaned_text column in the df_tweets dataset does not have the distinct classes to accurately predict which sentiment label of positive, negative, or neutral would likely occur for a Tweet pertaining COVID-19 in April 2020.

Figure 8:



Create Sentiment Labels:

Even though it was originally intended to use the pre-labeled corpus of Twitter samples from the Natural Language Toolkit (NLTK) Python library to train the sentiment analysis model of this project to predict and create sentiment labels for the df_tweets dataset, it was discovered that the Twitter samples from the NLTK library have only two sentiment labels which are positive and negative, and thus, they would not be able to train the sentiment analysis model to predict Tweets with neutral sentiment. It was found that Python's TextBlob package has a sentiment function that returns two properties: polarity and subjectivity. The polarity score is a float within the range [-1.0, 1.0] where 1 means a statement is positive and -1 means a statement is negative. The subjectivity score is a float within the range [0.0, 1.0] where 0.0 is very objective and 1.0 is very subjective. To create the sentiment labels for the df_tweets dataset, a function called, sentimentize, was written to measure the subjectivity and polarity of the Tweets' text under the cleaned_text column to determine whether they are positive, negative, or neutral. In the function, the subjectivity of the text is first measured. If the text has a subjectivity score of less than or equal to 0.05, it is labeled as neutral. If not, then the text's polarity is measured. If the polarity score is greater than 0.1, the text is labeled as positive. Otherwise, it is labeled as negative. The results of the sentimentize function were saved in a new column called, sentiment. There were 315,501 neutral labels, 255,354 negative labels, and 242,523 positive labels under the sentiment column in the df_tweets dataset. The df_tweets dataframe with the sentiment labels was exported as a CSV file named, tweets_sentiment.csv.

Create Sentiment Analysis Model:

To create the sentiment analysis model for this project, multi-label classification in NLP was used to create a classification model to predict what sentiment would be labeled for a Tweet concerning COVID-19 in April 2020. However, there are three well-known text classification algorithms that are used to create multi-class text classification models. These algorithms are Naive Bayes Classifier for Multinomial Models, Linear Support Vector Machine, and Logistic Regression. Thus, it was decided to create three different sentiment analysis models with each of these classification algorithms to evaluate which of these models would be the best to analyze the data for this project. This section documents how in-depth analysis was used to create the sentiment analysis model via multi-label classification with NLP performed on the cleaned_text column from the df_tweets dataset and with the sentiment column from the same dataset being denoted as the target variable.

Before the sentiment analysis model was created, the data from the df_tweets dataset needed its text data and target variable converted into features and formats respectively that can be used in the text classification model. The target variable was formatted to be one-hot encoded by sklearn's MultiLabelBinarizer function which transforms a list of sets or tuples into the supported multilabel format - a (samples x classes) binary matrix indicating the presence of a class label. This was first done by splitting the sentiment column's values into separate lists.

```
#Converting values in target variable into lists
y = df_tweets['sentiment'].str.split()
```

Then, they were fitted and transformed by the MultiLabelBinarizer function into the binary matrix format that can be used in the classification model for classifying the text data.

```
#One hot encode the target variable by using sklearn’s MultiLabelBinarizer( )
multilabel_binarizer = MultiLabelBinarizer()
multilabel_binarizer.fit(y)
```

```
MultiLabelBinarizer(classes=None, sparse_output=False)
```

```
# transform target variable
y = multilabel_binarizer.transform(y)
```

As the data from the df_tweets dataset is very large (813,378 observations), the training and validation sets of the text data and target variable were simply split with the train/test split method with an 80/20 split.

```
# split dataset into training and validation set
xtrain, xval, ytrain, yval = train_test_split(df_tweets['cleaned_text'], y,
test_size=0.2, random_state=42)
```

The train and validation sets of the text data were converted into TF-IDF (Term Frequency — Inverse Document Frequency) features.

```
#Using TF-IDF to extract features from the cleaned version of the text data
tfidf_vectorizer = TfidfVectorizer(max_df=0.8, max_features=10000) # set
10,000 most frequent words in the data as features

#Create TF-IDF features
xtrain_tfidf = tfidf_vectorizer.fit_transform(xtrain)
xval_tfidf = tfidf_vectorizer.transform(xval)
```

This means that the words in the text were quantified based on the computed weight of each word which signifies the importance of the word in the corpus.

With the features of the text data (cleaned_text column from the df_tweets dataset) extracted, and the target variable (sentiment column from the df_tweets dataset) formatted, the NLP text classification model is ready to be created. Building a model for every one-hot encoded target variable would take a considerable amount of time. Thus, the text classification model would be built with sklearn's OneVsRestClassifier class used to solve the model's problem as a Binary Relevance or a one-vs-all problem. The sentiment analysis models were built with the Naive Bayes Classifier for Multinomial Models, Linear Support Vector Machine, and Logistic Regression text classification algorithms and with sklearn's OneVsRestClassifier class as shown below before they were fitted on the validation sets of the TF-IDF features and the formatted target variable. After the models were trained, they were used to make predictions with the validation (or test) TF-IDF features set.

TF-IDF and Naive Bayes Classifier for Multinomial Models with OneVsRestClassifier

```
#Use sklearn's OneVsRestClassifier class to solve the Naive Bayes Classifier
model's problem as a Binary Relevance or one-vs-all problem
nb = MultinomialNB()
clf_nb = OneVsRestClassifier(nb)

#Fit model on TF-IDF train data
clf_nb.fit(xtrain_tfidf, ytrain)

#Make predictions for validation set
y_pred_tfidf_nb = clf_nb.predict(xval_tfidf)
```

TF-IDF and Linear Support Vector Machine with OneVsRestClassifier

```
#Use sklearn’s OneVsRestClassifier class to solve the Linear Support Vector
Machine model's problem as a Binary Relevance or one-vs-all problem
lsvm = SGDClassifier(loss='hinge', penalty='l2',alpha=1e-3, random_state=42,
tol=None)
clf_lsvm = OneVsRestClassifier(lsvm)

#Fit model on TF-IDF train data
clf_lsvm.fit(xtrain_tfidf, ytrain)

#Make predictions for validation set
y_pred_tfidf_lsvm = clf_lsvm.predict(xval_tfidf)
```

TF-IDF and Logistic Regression with OneVsRestClassifier

```
#Use sklearn’s OneVsRestClassifier class to solve the Logistic Regression
model's problem as a Binary Relevance or one-vs-all problem
lr = LogisticRegression(max_iter = 4000)
clf_lr = OneVsRestClassifier(lr)

#Fit model on TF-IDF train data
clf_lr.fit(xtrain_tfidf, ytrain)

#Make predictions for validation set
y_pred_tfidf_lr = clf_lr.predict(xval_tfidf)
```

To evaluate the models’ overall performances, all of the predictions and the entire target variable of the validation set would need to be taken into consideration. Thus, the metrics of the confusion matrix for multi-class classification were used to evaluate the three sentiment analysis models created. Namely, the micro, macro, and weighted F1-scores were emphasized to determine which model was best to analyze the data used for this project. In general, the F1 score is interpreted as a weighted average of the precision and recall where 1 is its best score, and 0 is its worst score. More specifically, the micro F1-score is calculated by considering the total TP (true positives), total FP (false positives), and total FN (false negatives) of the confusion matrix of the model. It does not consider each class individually. It calculates the metrics globally. For the macro F1-score however, it calculates the metrics for each class (sentiment label in this case) individually and then takes unweighted mean of the measures. In other words, it calculates the precision, recall, and micro F1-score for each class. Unlike the macro F1-score, the weighted F1-score takes a weighted mean of the measures. In other words, the weights for each class are the total number of samples of that class.

TF-IDF and Naive Bayes Classifier for Multinomial Models with OneVsRestClassifier

```
Confusion Matrix

[[49999   322   836]
 [31499 30901   549]
 [28604    73 19892]]
```

```
Accuracy: 0.45

Micro Precision: 0.97
Micro Recall: 0.47
Micro F1-score: 0.63

Macro Precision: 0.97
Macro Recall: 0.47
Macro F1-score: 0.63

Weighted Precision: 0.97
Weighted Recall: 0.47
Weighted F1-score: 0.63

Classification Report

              precision    recall  f1-score   support

    positive       0.96      0.44      0.61     51157
     neutral       0.99      0.49      0.66     62949
    negative       0.93      0.41      0.57     48569

   micro avg       0.96      0.45      0.62    162675
   macro avg       0.96      0.45      0.61    162675
weighted avg       0.96      0.45      0.61    162675
 samples avg       0.45      0.45      0.45    162675
```

TF-IDF and Linear Support Vector Machine with OneVsRestClassifier

```
Confusion Matrix

[[50659   257   241]
 [56892  6056     1]
 [40072   155  8342]]

Accuracy: 0.11

Micro Precision: 0.94
Micro Recall: 0.08
Micro F1-score: 0.15

Macro Precision: 0.94
Macro Recall: 0.08
Macro F1-score: 0.14

Weighted Precision: 0.94
Weighted Recall: 0.08
Weighted F1-score: 0.15

Classification Report

              precision    recall  f1-score   support

    positive       0.94      0.06      0.11     51157
```

```
     neutral       0.94      0.10      0.17     62949
    negative       0.97      0.17      0.29     48569

   micro avg       0.95      0.11      0.19    162675
   macro avg       0.95      0.11      0.19    162675
weighted avg       0.95      0.11      0.19    162675
 samples avg       0.11      0.11      0.11    162675
```

TF-IDF and Logistic Regression with OneVsRestClassifier

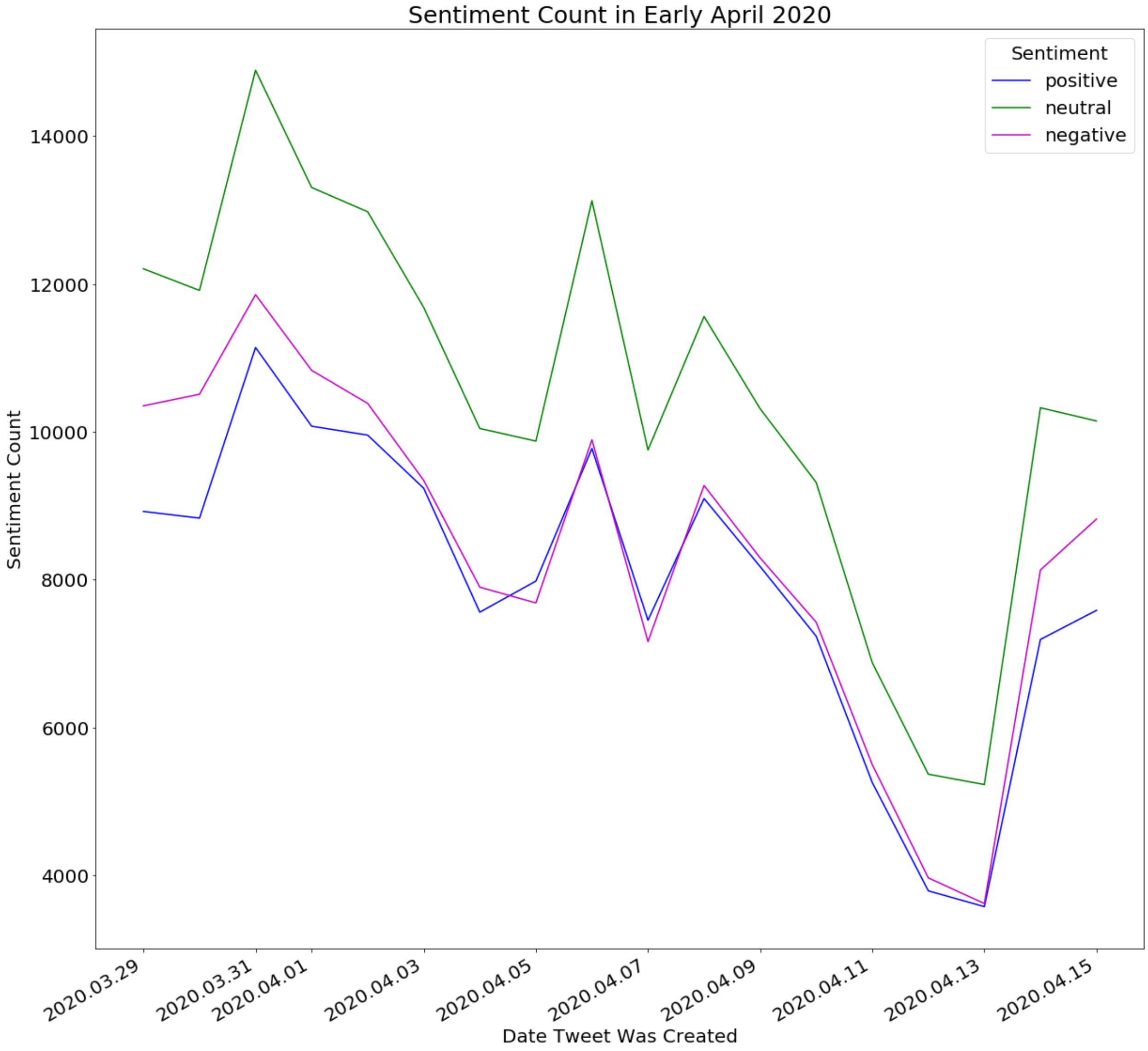
```
Confusion Matrix

[[50014   394   749]
 [  883 62032    34]
 [ 4440   205 43924]]

Accuracy: 0.93

Micro Precision: 0.97
Micro Recall: 0.95
Micro F1-score: 0.96

Macro Precision: 0.97
Macro Recall: 0.95
Macro F1-score: 0.96

Weighted Precision: 0.97
Weighted Recall: 0.95
Weighted F1-score: 0.96

Classification Report

              precision    recall  f1-score   support

    positive       0.94      0.91      0.93     51157
     neutral       0.99      0.99      0.99     62949
    negative       0.96      0.93      0.95     48569

   micro avg       0.97      0.95      0.96    162675
   macro avg       0.96      0.94      0.95    162675
weighted avg       0.97      0.95      0.96    162675
 samples avg       0.94      0.95      0.94    162675
```

According to the evaluation results of the three models above, the Logistic Regression with OneVsRestClassifier model had the highest micro, macro, and weighted F1-scores than the other two models. Thus, the Logistic Regression with OneVsRestClassifier model was the best model for the sentiment analysis model of this project.

Post Statistical Analysis:

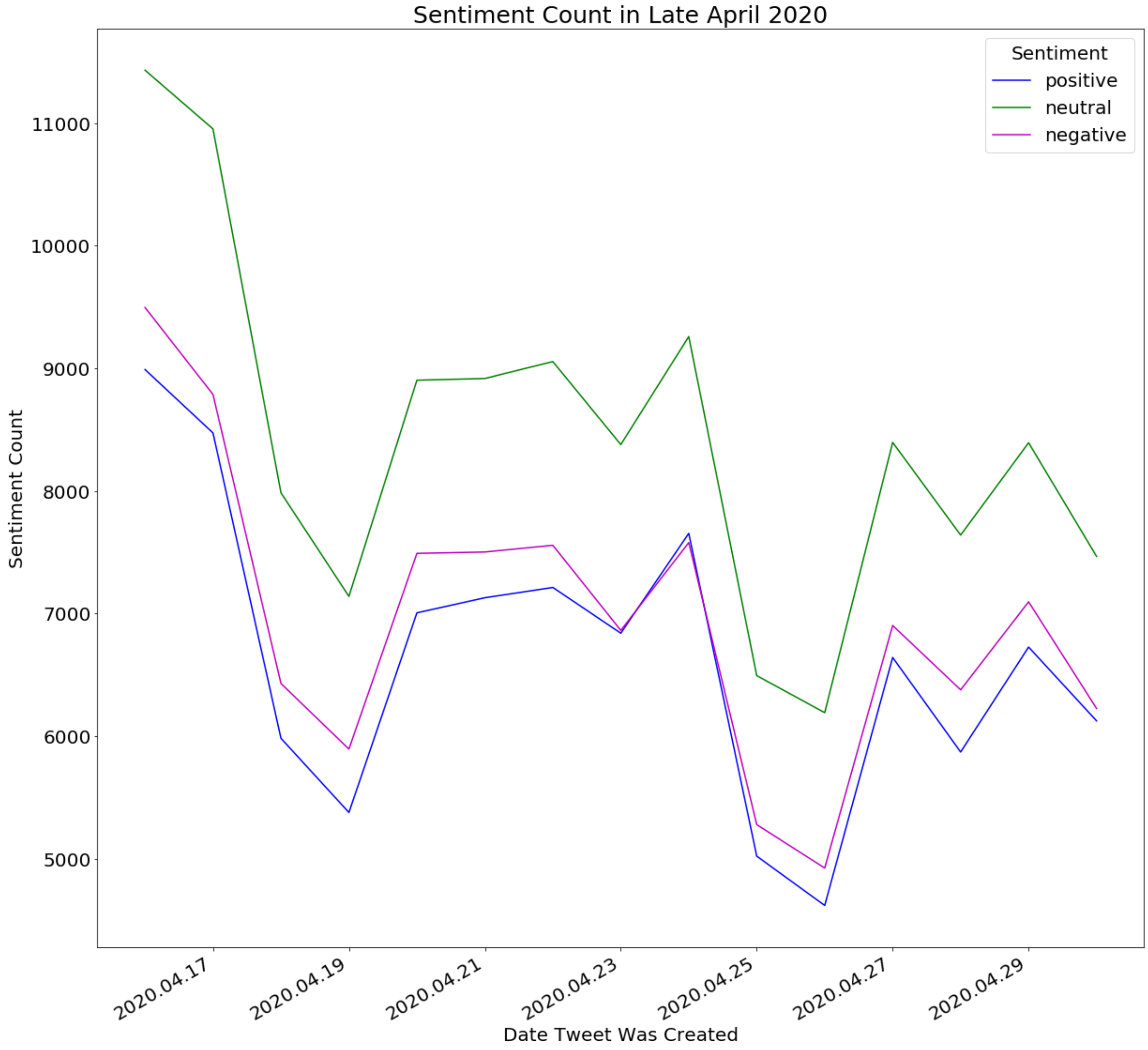
Additional statistical analysis was done on the dataset in tweets_sentiment.csv to see if there were any relationships between the counts of sentiment labels of positive, neutral, and negative with the time duration of April 2020 Tweets concerning COVID-19 and whether or not a Twitter account is verified due to verified Twitter accounts tending to have more numbers of followers and friends as analyzed in the initial statistical findings of this project. The dataset imported from tweets_sentiment.csv was denoted as tweets during the post statistical analysis.

Figure 9:



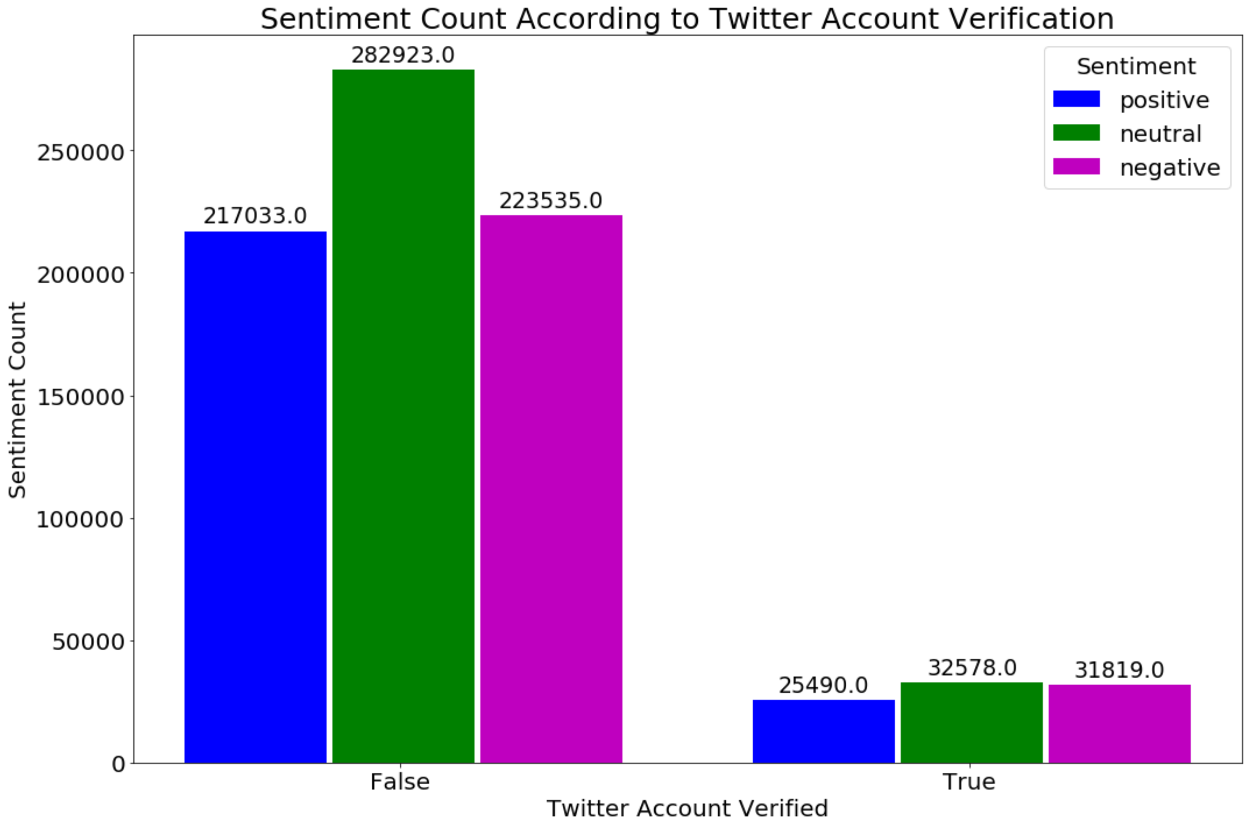
According to Figure 9 and 10, there appears to be no linear correlation between the sentiment label counts and the dates the Tweets were created during the course of April 2020. It is interesting to note though that the counts of sentiment labels peaked highest on 3/31 in Figure 9 (early April) and 4/16 in Figure 10 (late April) while they dipped lowest on 4/13 in Figure 9 and 4/26 in Figure 10. Perhaps Tweets from those days can be analyzed to see what sort of events happened on those days to cause the peaks and dips in the plots.

Figure 10:



According to Figure 11 below, there are far less sentiment label counts for the verified Twitter accounts compared to the unverified Twitter accounts. This indicates that a Twitter account does not need to be verified to affect the sentiment concerning COVID-19 during April 2020. This in turn might indicate that the number of friends and the number of followers also do not affect the sentiment towards the pandemic. It would be interesting to further analyze these results to see which features of the tweets dataset significantly affect the sentiment concerning COVID-19 during April 2020. However, it is worth noting that the results of the analysis might change if more data of Tweets concerning COVID-19 were obtained beyond April 2020. Then a more accurate analysis could be given of how the sentiment towards COVID-19 changes during the course of the pandemic.

Figure 11:



Discussion:

As the best sentiment analysis model (Logistic Regression with OneVsRestClassifier model) of this project has micro, macro, and weighted F1-scores of 0.96, it looks to be serviceable in predicting the sentiment of Tweets. However, this model was only worked on English Tweets. Thus, it would have limitations analyzing Tweets written in different languages as different languages have different grammatical structures. It is also worth noting that the text cleaning functions in this project could not clean Tweet text in different formats such as, 𝖎 𝖆𝖒 𝖑𝖊𝖆𝖗𝖓𝖎𝖓𝖌 𝖓𝖊𝖜 𝖜𝖆𝖞 𝖔𝖋 𝖙𝖜𝖊𝖊𝖙𝖎𝖓𝖌 #𝕮𝖔𝖗𝖔𝖓𝖆𝖁𝖎𝖗𝖚𝖘 #𝕷𝖔𝖈𝖐𝖉𝖔𝖜𝖓. Thus, these types of Tweets were excluded from the analysis of the model. Perhaps a function could be created to convert these Tweets back into the standardized text format? It is also worth noting that even if the sentimentize function written in this project largely did a good job in labeling the Tweets as positive, negative, and neutral, there were still a number of misclassifications such as "whitehous project devast covid death toll" being labeled as neutral. Perhaps dictionaries categorizing words and phrases as positive, negative, and neutral would increase the accuracy of the sentiment labeling function especially concerning the COVID-19 pandemic. Overall, the sentiment analysis model created in this project would most successfully work on English-language Tweets. However, might it be possible to train this model to incorporate the different languages as well as fine-grained sentiments such as angry, sad, happy, sarcasm, etc.?