INDIANA UNIVERSITY

Sentimental Analysis On Covid Vaccine Data

CSCI - B 565 Data Mining Project Report

Spring 2022

- Shiwani Rajagopalan (srajago@iu.edu)
- Meet Jatin Shah (meetshah@iu.edu)
- Ayush Sanghavi (sanghavi@iu.edu)

INDEX

ABSTRACT

The year 2020 has been a disastrous year for the entire world due to the breakout of the Covid-19 pandemic. The disease has spread across multiple nations, and has infected over 100k people. The transmission of the disease is direct touch, droplet and potential aerosol ways. The only way to suppress the same was to bring out an invention of the vaccine. Many research and development centers require a large sample size of infected subjects, in order to invent and test a vaccine on the same.

The three major companies namely, Pfizer, Moderna and Johnson&Johnson have worked hard day and night in order to bring out a vaccine for the entire world. There are a couple of other companies too, but the top 3 were the ones widely used.

We know that Twitter is a social media platform used to portray freedom of speech for all. In this project, we are going to use some of the machine learning models in order to check the sentiments of tweets based on categorizing them as positive and negative by performing an analysis about the covid vaccines.

KEYWORDS

Pandas, NumPy, sci-kit learn, Twitter, Sentimental analysis, Support Vector Machine, snscrape, Logistic Regression, Bagging, Adaboost, K-Nearest Neighbors, Word Cloud, Visualization, lemmatization, tokenization, stop words, data preprocessing, data modelling.

INTRODUCTION

We will be using a labeled dataset from Kaggle (Link here) in order to train our models. The labeled dataset basically contains a balanced list of about 100K tweets which are marked as positive or negative depending on a few keywords. We will preprocess the dataset, so that our models output high accuracy with a lot of efficiency. The data was preprocessed by removing the following :

a) Stop words
b) Usernames (words that start with @ on Twitter)
c) Punctuation
d) URLs
e) Numbers

We need to remove all of the above since they do not contribute towards sentimental analysis.

Furthermore, we perform tokenization of tweets (convert them to a list of independent words) separated by comma, so that we can perform lemmatization. On all these words, TF-IDF algorithm was applied in order to get the frequency of the used words and to also know how important the word is.

After the data preprocessing, we feed our dataset to the models which will run on our testing dataset to calculate the accuracy.

Once we get the model that outputs the best accuracy, we then create our own dataset by scraping tweets from Twitter using the python package snscrape. We give the following query keywords “Pfizer, pfizer, Moderna, moderna, Johnson&Johnson,

johnson&johnson". We scraped data between the timeframe 1st January 2021 and 30th June 2021, since that was the time when the vaccines rolled out and Twitter was flooded with vaccine tweets. We wanted to find out the sentiments of these tweets whether they are positive or negative. Lastly, we perform some data visualization and infer some insights from our results and also the frequency of the top "x" words.

METHODS

1) Data Preprocessing :

Below are some highlights of how we preprocess the data. Once the data is cleaned, we move on to the next steps of modelling. We will use some Machine learning algorithms and check which one produces the best output.

Removing Stopwords ; Stopwords are all words that are trivial and do not affect the sentiment of the tweet. These words are extra and it needs to be removed

In [ ]:
```
stops = set(stopwords.words('english'))
def remove_stopwords(text):
    return " ".join([word for word in str(text).split() if word not in stops])
df_final['tweet'] = df_final['tweet'].apply(lambda text: remove_stopwords(text))
```

Removing Usernames; all usernames in the tweets will start with @ , and those need to be removed

In [ ]:
```
def remove_usernames(text):
    return re.sub(r"@[A-Za-z0-9_]+",'',text)
df_final['tweet'] = df_final['tweet'].apply(lambda text: remove_usernames(text))
```

Removing Punctuations, punctuations are not necessary to find out the sentiments of the tweets and thereby removing it

In [ ]:
```
eng_punctuations = string.punctuation
def remove_punctuations(text):
    dict_translate = str.maketrans('', '', eng_punctuations)
    return text.translate(dict_translate)
df_final['tweet'] = df_final['tweet'].apply(lambda text: remove_punctuations(text))
```

In [ ]:
```
eng_punctuations
```

Out[ ]:
```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

Removing URLs

In [ ]:
```
def removing_URLs(text):
    return re.sub('[^\w\s]|http\S+',' ',text)
df_final['tweet'] = df_final['tweet'] .apply(lambda text: removing_URLs(text))
```

Removing Numbers

In [ ]:
```
def removing_numbers(text):
    return re.sub('[0-9]+', '', text)
df_final['tweet'] = df_final['tweet'].apply(lambda text: removing_numbers(text))
```

2) Model evaluation :

For model evaluation, we have written a common function that displays the confusion matrix along with precision, recall, f1-score and support.

Below is the screenshot of the function that we will use and call and set the parameters as our model and the testing data.

In [ ]:
```
def model_evaluate(model,X_test):
    y_pred = model.predict(X_test)
    print(classification_report(y_test, y_pred))
    cf_matrix = confusion_matrix(y_test, y_pred)
    categories = ['Negative','Positive']
    group_names = ['True Neg ','False Pos ', 'False Neg ','True Pos' ]
    group_percentages = ['{0:.2%}'.format(value) for value in cf_matrix.flatten() / np.sum(cf_matrix)]
    labels = [f'{v1}n{v2}' for v1, v2 in zip(group_names,group_percentages)]
    labels = np.asarray(labels).reshape(2,2)
    sns.heatmap(cf_matrix, annot = labels, cmap = 'Greens',fmt = '',
    xticklabels = categories, yticklabels = categories)
    plt.xlabel("Predicted values", fontdict = {'size':16}, labelpad = 10)
    plt.ylabel("Actual values" , fontdict = {'size':16}, labelpad = 10)
    plt.title ("Confusion Matrix", fontdict = {'size':16}, pad = 25)
```

Now we will check the confusion matrix for each of the models we run.

a) Logistic Regression :

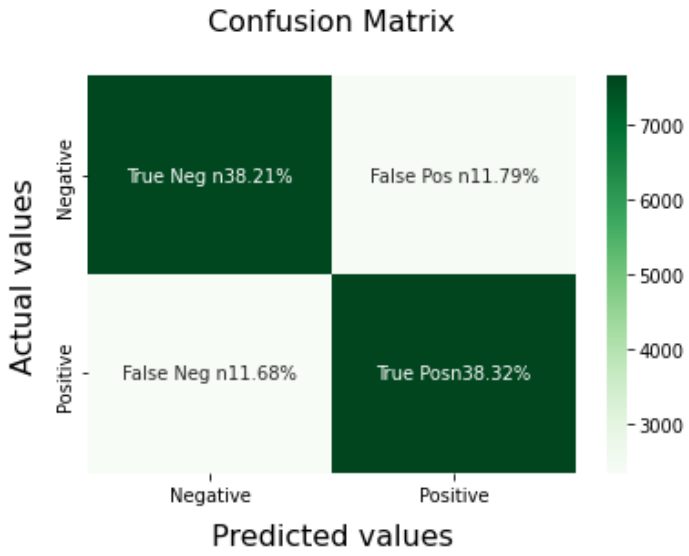
Logistic Regression Model

```
lr_model = LogisticRegression(C = 1, max_iter = 1000, penalty = 'l2', n_jobs=-1)
lr_model.fit(X_train_tfidf  ,y_train)
```

```
LogisticRegression(C=1, max_iter=1000, n_jobs=-1)
```

```
model_evaluate(lr_model ,X_test_tfidf)
```

	precision	recall	f1-score	support
0	0.77	0.76	0.77	10000
1	0.76	0.77	0.77	10000
accuracy			0.77	20000
macro avg	0.77	0.77	0.77	20000
weighted avg	0.77	0.77	0.77	20000



We notice that the accuracy we got here is 77%, which is good enough for this data, where we are doing some NLP.

b) Support Vector Machine

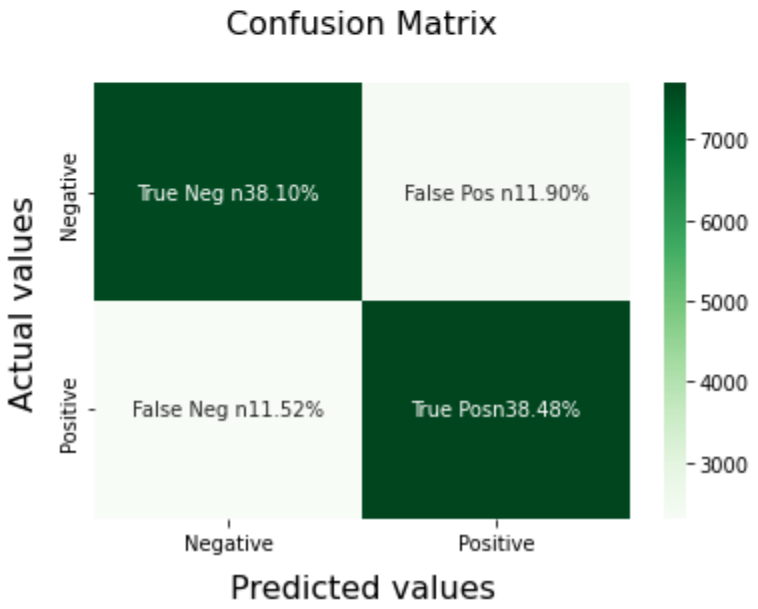
SVM Model

```
### Tried C=0.01,0.1,1
svm_model = LinearSVC(C=0.1)
svm_model.fit(X_train_tfidf  ,y_train)
```

```
LinearSVC(C=0.1)
```

```
model_evaluate(svm_model ,X_test_tfidf)
```

	precision	recall	f1-score	support
0	0.77	0.76	0.76	10000
1	0.76	0.77	0.77	10000
accuracy			0.77	20000
macro avg	0.77	0.77	0.77	20000
weighted avg	0.77	0.77	0.77	20000



We find out that SVM outputs an accuracy of 77%. This is as good as logistic regression, but we go ahead with SVM when compared to logistic regression since, when we have more and more data, SVM will definitely perform better.

c) K-Nearest Neighbors

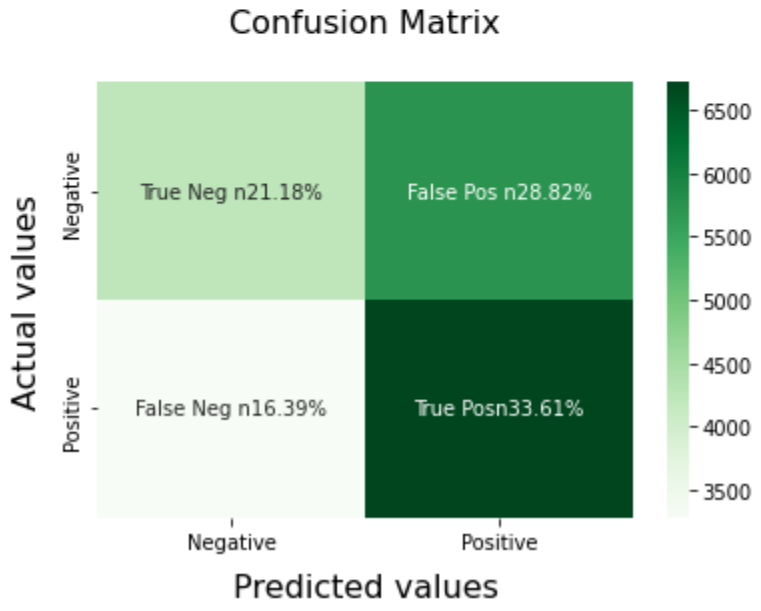
KNN

```
## tried k=3,5,7,9
knn_model = KNeighborsClassifier(n_neighbors=9)
knn_model.fit(X_train_tfidf ,y_train)
```

```
KNeighborsClassifier(n_neighbors=9)
```

```
model_evaluate(knn_model ,X_test_tfidf)
```

	precision	recall	f1-score	support
0	0.56	0.42	0.48	10000
1	0.54	0.67	0.60	10000
accuracy			0.55	20000
macro avg	0.55	0.55	0.54	20000
weighted avg	0.55	0.55	0.54	20000



For KNN, the accuracy is 55%. Hence we do not choose this one.

d) Ada-Boost Classifier

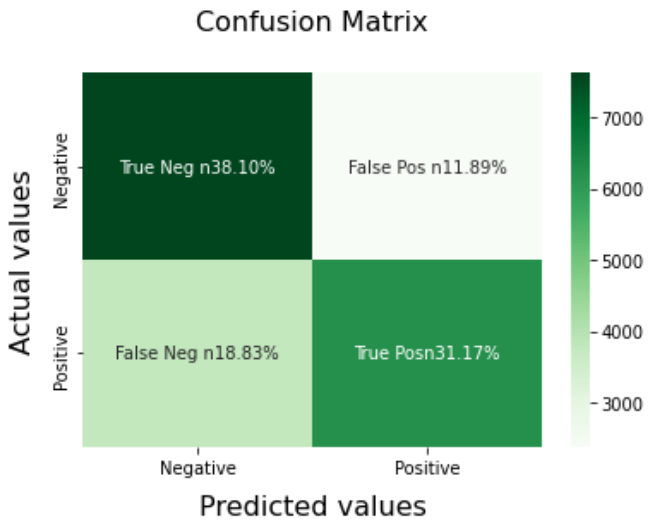
Ada Boost Classifier

```
## Tried n_estimators = 10,25,50,75,100
pipeline =  DecisionTreeClassifier(random_state=1)
boostclassifier_model = AdaBoostClassifier(base_estimator=pipeline, n_estimators=2,
                                random_state=1,learning_rate =0.1)
boostclassifier_model.fit(X_train_tfidf ,y_train)
```

```
AdaBoostClassifier(base_estimator=DecisionTreeClassifier(random_state=1),
                   learning_rate=0.1, n_estimators=2, random_state=1)
```

```
model_evaluate(boostclassifier_model ,X_test_tfidf)
```

	precision	recall	f1-score	support
0	0.67	0.76	0.71	10000
1	0.72	0.62	0.67	10000
accuracy			0.69	20000
macro avg	0.70	0.69	0.69	20000
weighted avg	0.70	0.69	0.69	20000



ADA boost outputs 69%, hence we discard the same.

e) Bagging classifier

Bagging Classifier

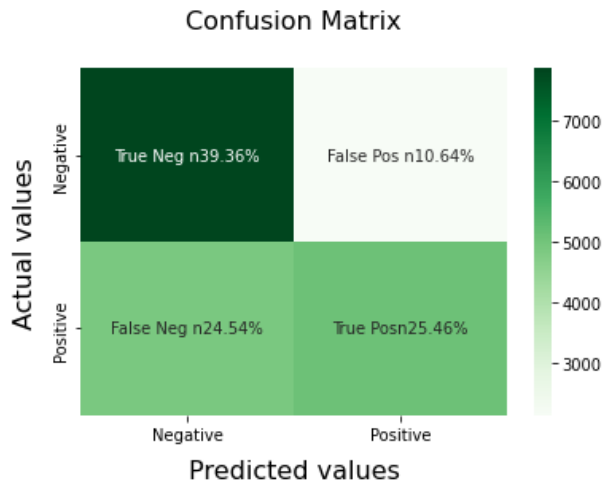
```
## Tried n_estimators = 10,25,50,75,100
pipeline = make_pipeline(DecisionTreeClassifier(random_state=1))

bgclassifier = BaggingClassifier(base_estimator=pipeline, n_estimators=2,
                                 max_features=X_train_tfidf.shape[1]//2,
                                 max_samples=X_train_tfidf.shape[0]//2,
                                 random_state=1, n_jobs=5)
bgclassifier.fit(X_train_tfidf, y_train)
```

```
BaggingClassifier(base_estimator=Pipeline(steps=[('decisiontreeclassifier',
                                                  DecisionTreeClassifier(random_state=1))]),
                  max_features=194280, max_samples=40000, n_estimators=2,
                  n_jobs=5, random_state=1)
```

```
model_evaluate(bgclassifier ,X_test_tfidf)
```

	precision	recall	f1-score	support
0	0.62	0.79	0.69	10000
1	0.71	0.51	0.59	10000
accuracy			0.65	20000
macro avg	0.66	0.65	0.64	20000
weighted avg	0.66	0.65	0.64	20000



The accuracy when bagging is used is 65%. We discard this as well.

Hence, we pick SVM, since for huge NLP data it gives the best accuracy. We will now create a new dataset and put it into a dataframe using pandas.

So far we have only performed supervised machine learning algorithms for pre-labelled dataset, so here we have decided to scrape tweets on our own, build our own dataset and label the tweets as positive or negative.

Loading the covid vaccine data

```
!pip3 install snscrape
```

```
Collecting snscrape
  Downloading snscrape-0.3.4-py3-none-any.whl (35 kB)
Collecting lxml
  Downloading lxml-4.8.0-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_24_x86_64.whl (6.4 MB)
     |████████████████████████████████| 6.4 MB 29.5 MB/s
Collecting beautifulsoup4
  Downloading beautifulsoup4-4.11.1-py3-none-any.whl (128 kB)
     |████████████████████████████████| 128 kB 62.7 MB/s
Requirement already satisfied: requests[socks] in /usr/local/lib/python3.7/site-packages (from snscrape) (2.25.1)
Collecting soupsieve>1.2
  Downloading soupsieve-2.3.2.post1-py3-none-any.whl (37 kB)
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/local/lib/python3.7/site-packages (from requests[socks]->snscrape)
(4.0.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/site-packages (from requests[socks]->snscrape)
(2021.10.8)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.7/site-packages (from requests[socks]->snscrap
e) (1.26.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/site-packages (from requests[socks]->snscrape) (2.10)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python3.7/site-packages (from requests[socks]->snscrap
e) (1.7.1)
Installing collected packages: soupsieve, lxml, beautifulsoup4, snscrape
Successfully installed beautifulsoup4-4.11.1 lxml-4.8.0 snscrape-0.3.4 soupsieve-2.3.2.post1
```

```
import pandas as pd
import snscrape.modules.twitter as sntwitter
import itertools
```

Extracting Tweets for the search keyword Pfizer between January 01 2020 to June 30 2020

```
df_pf1 = pd.DataFrame(itertools.islice(sntwitter.TwitterSearchScraper(
    'pfizer Pfizer vaccine lang:en since:2020-01-01 until:2020-01-31').get_items(), 1000))[['date', 'content']]
df_pf2 = pd.DataFrame(itertools.islice(sntwitter.TwitterSearchScraper(
    'pfizer Pfizer vaccine lang:en since:2020-02-01 until:2020-02-28').get_items(), 1000))[['date', 'content']]
df_pf3 = pd.DataFrame(itertools.islice(sntwitter.TwitterSearchScraper(
    'pfizer Pfizer vaccine lang:en since:2020-03-01 until:2020-03-31').get_items(), 1000))[['date', 'content']]
df_pf4 = pd.DataFrame(itertools.islice(sntwitter.TwitterSearchScraper(
    'pfizer Pfizer vaccine lang:en since:2020-04-01 until:2020-04-30').get_items(), 1000))[['date', 'content']]
df_pf5 = pd.DataFrame(itertools.islice(sntwitter.TwitterSearchScraper(
    'pfizer Pfizer vaccine lang:en since:2020-05-01 until:2020-05-31').get_items(), 1000))[['date', 'content']]
df_pf6 = pd.DataFrame(itertools.islice(sntwitter.TwitterSearchScraper(
    'pfizer Pfizer vaccine lang:en since:2020-06-01 until:2020-06-30').get_items(), 1000))[['date', 'content']]
```

```
df_pf = df_pf1.append([df_pf2,df_pf3,df_pf4,df_pf5,df_pf6],ignore_index=True)
```

We now collect data and store it into 6 dataframes. We collect data at a time only in one dataframe since snscrape allows only about 1000 tweets to be collected.

After we collect all the pfizer tweets in different data frames, we concatenate all of them and create a data frame for pfizer.

We do the same for Moderna and J&J.

In the end we concatenate all the three sub dataframes into one and add a column so we know which tweet is for which vaccine.

We then fit in our SVM model since it performed the best into this dataset and check the accuracy.

Now we will label the dataset that we just created by scraping the tweets from Twitter using snscrape.

RESULTS & DISCUSSIONS

Below is the SVM results for the dataset that was labeled after testing on the pre-trained model.

	date	content	vaccine	content_comb	month_num	month	sentiment_num	sentimen
0	2020-01-30 23:17:54+00:00	[certain, disease, serious, pregnant, mother, ...	pfizer	certain disease serious pregnant mother develo...	1	January	1	Positiv
1	2020-01-30 20:44:05+00:00	[one, pharma, company, like, come, coronarviru...	pfizer	one pharma company like come coronarvirus vacc...	1	January	0	Negativ
2	2020-01-30 20:15:46+00:00	[safe, effective, go, phrase, safe, effective,...	pfizer	safe effective go phrase safe effective apply ...	1	January	1	Positiv
3	2020-01-30 18:39:49+00:00	[whereas, happily, betrayed, all, promised, pa...	pfizer	whereas happily betrayed all promised parent f...	1	January	0	Negativ
4	2020-01-30 17:35:31+00:00	[russia, chorona, virus, medicine, vaccine, ho...	pfizer	russia chorona virus medicine vaccine hospital...	1	January	0	Negativ
...	...	...	...	...	...	...	...	...
13986	2020-06-10 17:53:42+00:00	[johnson, johnson, announces, acceleration, co...	johnson&johnson	johnson johnson announces acceleration covid v...	6	June	1	Positiv
13987	2020-06-10 17:49:55+00:00	[johnson, johnson, begin, human, trial, corona...	johnson&johnson	johnson johnson begin human trial coronavirus ...	6	June	0	Negativ
13988	2020-06-10 17:43:33+00:00	[johnson, johnson, move, start, coronavirus, v...	johnson&johnson	johnson johnson move start coronavirus vaccine...	6	June	1	Positiv
13989	2020-06-10 17:43:32+00:00	[johnson, johnson, move, start, coronavirus, v...	johnson&johnson	johnson johnson move start coronavirus vaccine...	6	June	1	Positiv
13990	2020-06-10 17:43:07+00:00	[know, johnsonjohnson, made, baby, shampoo, ye...	johnson&johnson	know johnsonjohnson made baby shampoo yeah yea...	6	June	0	Negativ

13991 rows × 8 columns

We found out that from all the tweets we scraped :

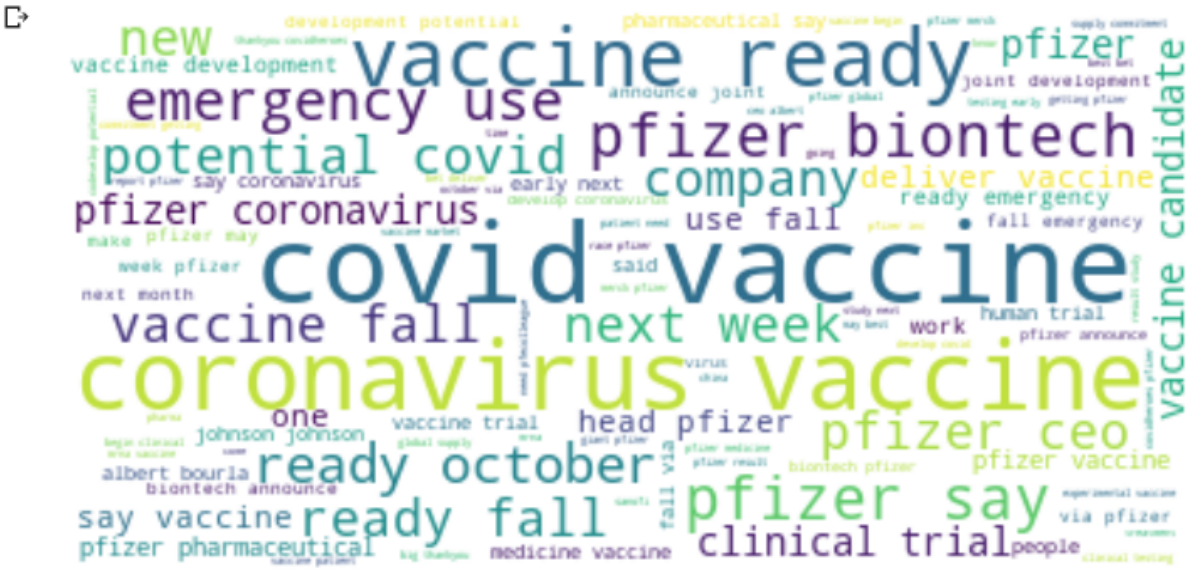
9412 were positive and 4579 were negative tweets.

Total Number of Positive and Negative tweets across all vaccines

```
d['sentiment'].value_counts()
```

```
Positive    9412
Negative    4579
Name: sentiment, dtype: int64
```

```
#text = " ".join(c for c in d.content)
text = " ".join(review for review in d_pf.content_comb)
# Creating word_cloud with text as argument in .generate() method
word_cloud = WordCloud(background_color = 'white').generate(text)
# Display the generated Word Cloud
plt.figure(figsize=(12,12))
plt.imshow(word_cloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



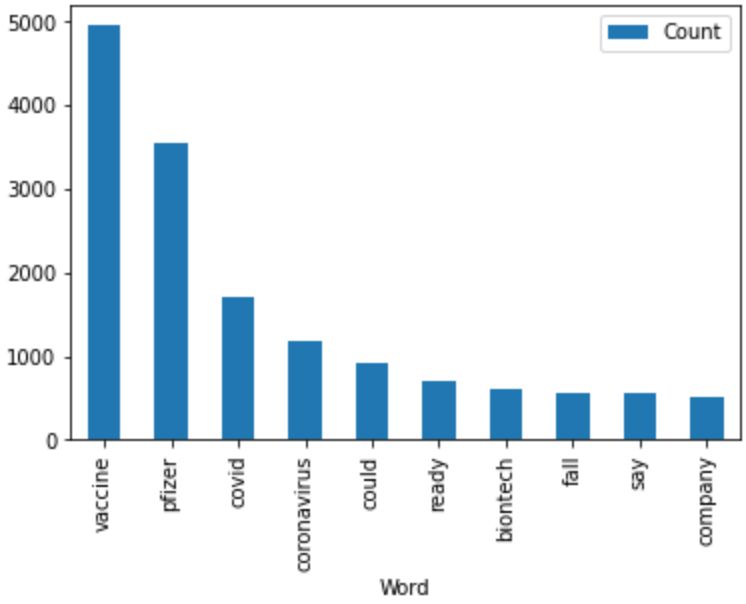
Generated a word cloud for Pfizer tweets.

Similarly did the same for Moderna as well as Johnson&Johnson.

```
How many most common words to print: 10

The 10 most common words are as follows

vaccine :  4954
pfizer :  3550
covid :  1702
coronavirus :  1171
could :  931
ready :  705
biontech :  617
fall :  565
say :  551
company :  509
<matplotlib.axes._subplots.AxesSubplot at 0x7fd9d87d1450>
```



Found out the top 10 frequently used words in Pfizer tweets, similarly did the same for Moderna as well as Johnson&Johnson.

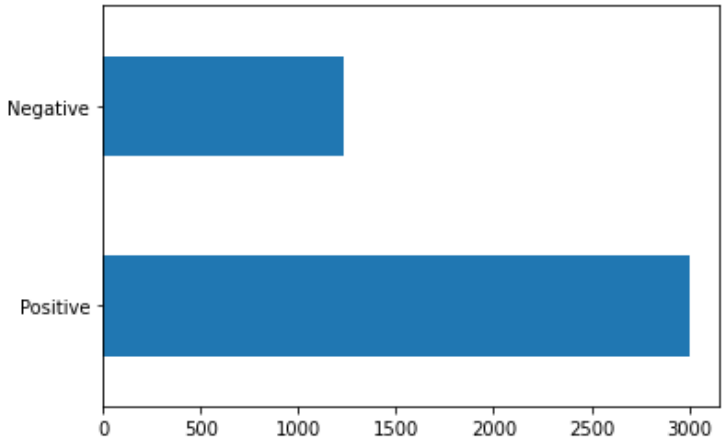
Below is the graph we have used to find the number of positive and negative words for Pfizer.

```
[ ] d_pf = d[d['vaccine']=='pfizer']
    d_pf['sentiment'].value_counts()
```

```
Positive    3003
Negative    1233
Name: sentiment, dtype: int64
```

```
d_pf['sentiment'].value_counts().plot(kind='barh')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd9d7b03e10>
```

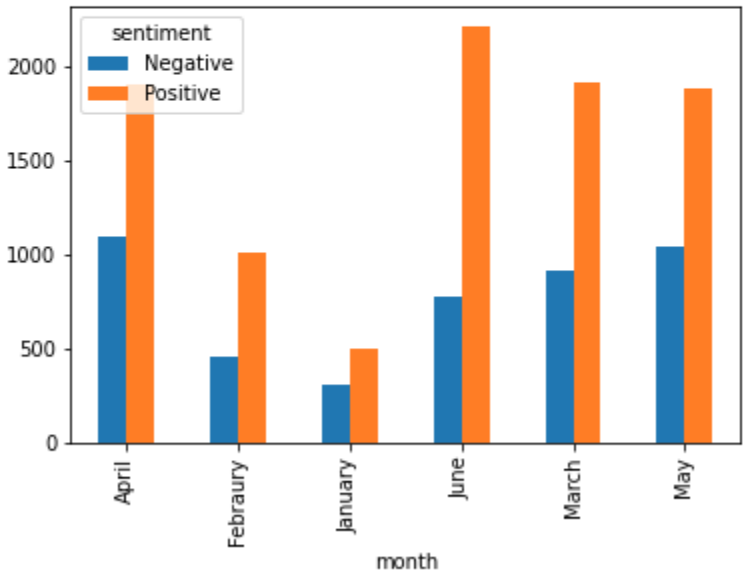


We did the same for Moderna as well as Johnson&Johnson.

Next, we found out the overall sentiment distribution over months for each vaccine, below is the picture for the “Pfizer” tweets.

```
(d
 .groupby(['month', 'sentiment'])
 .size()
 .unstack()
 .plot.bar()
)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd9d7443dd0>
```



CONCLUSION AND FUTURE SCOPE

We noticed that usually a model to be trained on a dataset with a lot of text gets a little difficult. Models used for NLP and NLU usually have a lower accuracy as compared to models that are fed on numbers/integers/continuous values.

In future, we can use a larger dataset and apply neural networks to find out the sentiments of the tweets.

We also plan to build a webapp over which given the keyword and the count we are able to find out the number of tweets positive and/or negative in the given time frame.

The good thing is, most tweets regarding the vaccines on Twitter were positive!

REFERENCES

1) https://journals.sagepub.com/doi/full/10.1177/2050168420982128

2) https://www.researchgate.net/publication/301649777_Analyzing_Scientific_Papers_Based_on_Sentiment_Analysis_First_Draft

3) http://www.cs.columbia.edu/~julia/papers/Agarwaletal11.pdf

4) https://ieeexplore.ieee.org/document/6758829

5) https://www.researchgate.net/publication/320903527_Sentiment_Analysis_A_Comparative_Study_of_Supervised_Machine_Learning_Algorithms_Using_Rapid_miner

6) https://d1wqtxts1xzle7.cloudfront.net/68071049/24898_48714_1_PB-with-cover-page-v2.pdf?Expires=1651539331&Signature=cMAHP0vI-vsBz5VrCawwExnnopXKjQy0ahHqJXTLzXnwzzDzNOLJkGTJxW69-JEm2cerLNhRQtgJQO5KE4NBJCLuM-nAnS37lB9YjikzJBPwr1qoT39u3un1XnjFNKCgVMk1qLyzslep~pWVKi316LzODLnUcFSpCVvetWkzQme0spFamVI8bAtje20a5wKoTGkbSLakfG4lP-n2ZtYUKNyp9xvxWJjuOpPnGLWN5ksZGWalXbopx3oz3RxziH22wjLcc4XXDRWEP2MlwfxjxpXVkar1I36aOwu2O7hIOmsyhrT-c45BU12W8RZc3SpGR2Qlpl2CJLr74BmsjR9tdg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

7) https://d1wqtxts1xzle7.cloudfront.net/68071049/24898_48714_1_PB-with-cover-page-v2.pdf?Expires=1651539331&Signature=cMAHP0vI-vsBz5VrCawwExnnopXKjQy0ahHqJXTLzXnwzzDzNOLJkGTJxW69-JEm2cerLNhRQtgJQO5KE4NBJCLuM-nAnS37lB9YjikzJBPwr1qoT39u3un1XnjFNKCgVMk1qLyzslep~pWVKi316LzODLnUcFSpCVvetWkzQme0spFamVI8bAtje20a5wKoTGkbSLakfG4lP-n2ZtYUKNyp9xvxWJjuOpPnGLWN5ksZGWalXbopx3oz3RxziH22wjLcc4XXDRWEP2MlwfxjxpXVkar1I36aOwu2O7hIOmsyhrT-c45BU12W8RZc3SpGR2Qlpl2CJLr74BmsjR9tdg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA