Public opinion and sentiment analysis on the COVID-19 vaccine using Twitter data

Project report

Project Report on
Public opinion and sentiment analysis on the COVID-19 vaccine using Twitter data
Subject: Data Mining (CMPE 255)
M.S in Software Engineering (Semester – 1)
(User Defined Project)

Submitted by:
Team DATAGEEKS
Sanjay Bhargav Madamanchi (016421587)
Chirudeep Gorle ()
Raparla Sravani ()
TanujaReddy Maligireddy ()

Guided by:
Prof. Vijay Eranti
Academic Year: 2022-23

Abstract

Covid-19 has a large impact on people and a lot of difficulties were present. The objective of this project is to collect tweets relating to the COVID-19 vaccine either from Twitter or from another data source and analyze the data to gain a deeper understanding of the data. Ultimately, the goal is to analyze public sentiments and opinions regarding COVID-19 vaccine by applying different approaches and understanding the important features of this data.

Introduction

Millions of people were affected by Covid during the pandemic. Today there is a large amount of data available on relevant statistics as well as on additional factors, and it is natural to try to make use of these to improve our understanding of the industry based on the people's opinions. Every drug manufacturer or the governments must understand the opinions of the people weather they are inclined towards vaccines, or it has any other effects. In our project, we have implemented machine learning model which will predict the sentiment of the tweet posted by user weather its positive or negative or neutral based on the description provided. A model like this would be very much valuable for vaccine manufacturers or government who can make use of the information to check the sentiment of people on vaccines.

Related work

For this project, we have used the CRISP-DM methodology.

1. Covid Vaccination Progress: a. Progress of covid vaccines per day, per week, month, to an year visualizing using plotly for every country

2. Vaccines Success rate: a. Sentiment analysis of predictive as well as subjective data on vaccines such that people couldn't be affected by the powerful toxins that they are allergic to.

3. Date Time Features: these are components of the time step itself for each observation: a. 'Series' refers to each time step (you will see that in the next cell) b. 'Target' refers to the target value at the current time step.

4. Change emojis to either positive or negative: a. This will add value to sentiment analysis appropriately

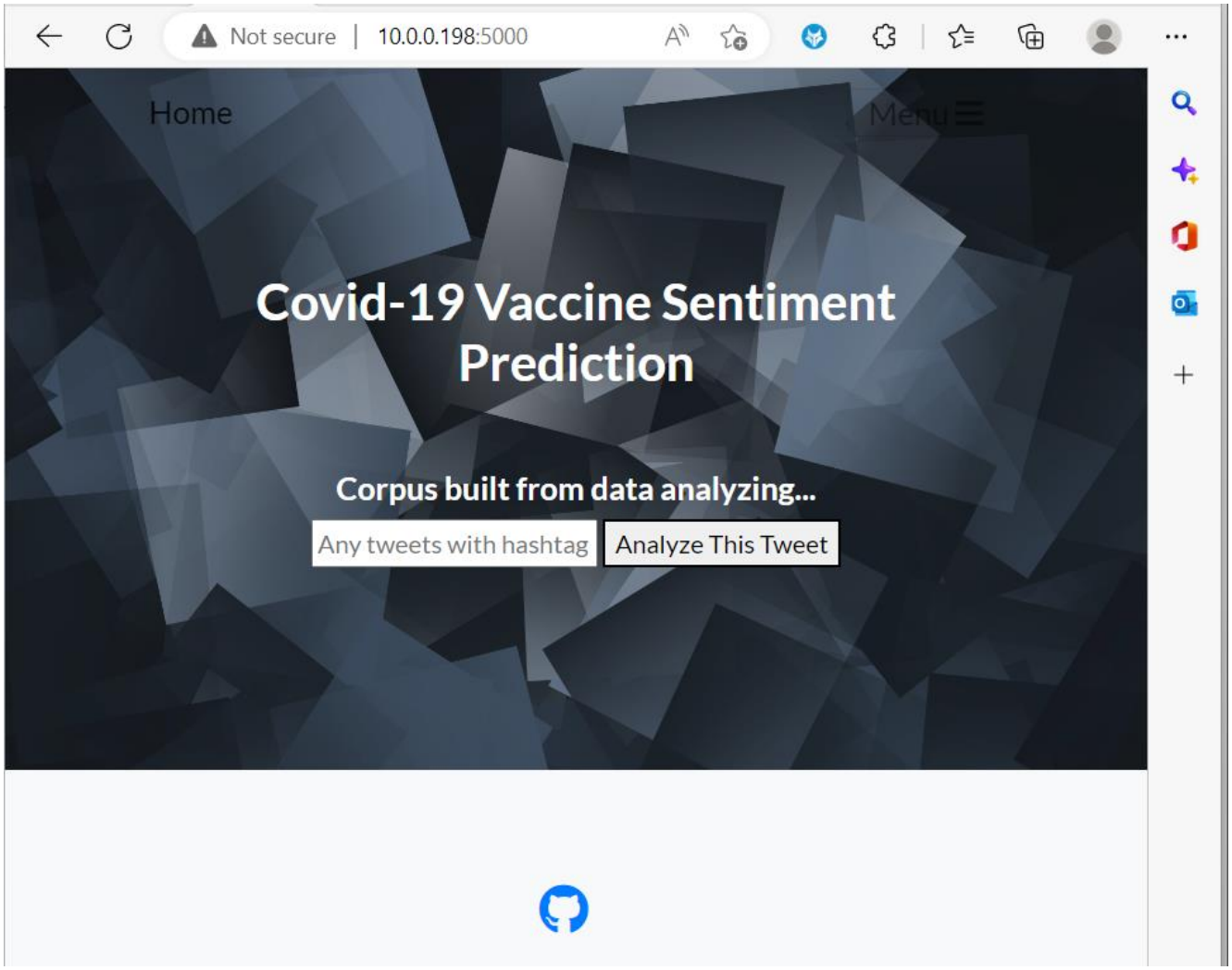
5. Lower all the text or tweets: a. This will ensure tweets with capitalization and non-capitalization are tested as the same.

6. Delete numbers, quotes, links, user information, one characters and punctuation: a. This only adds noise or is unhelpful to NLP analysis.,

7. Clean double spaces and multiple treats: a. This is just to make sure non characters and consecutive repeated words aren't used for NLP analysis.

8. Replace common words: a. This is to update contraction words with their appropriate expansion which can be used in NLP analysis.

9. We build a flask rest API with user interface. We used python to develop front end.



Data

Data for this project, we selected raw data from Kaggle Sources:

1) https://www.kaggle.com/gpreda/all-covid19-vaccines-tweets

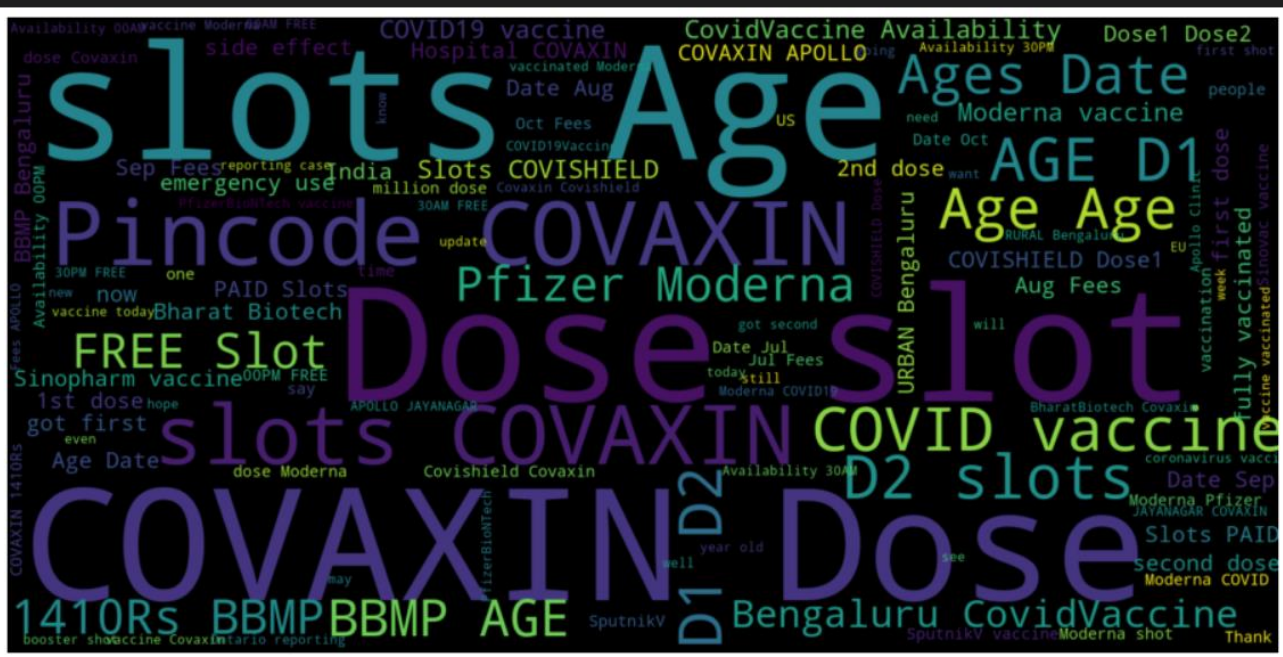
2) https://www.kaggle.com/gpreda/covid-world-vaccination-progress.

We have preprocessed the data by removing unwanted columns, null values. Also, duplicate values were deleted, and appropriate column values were casted to relevant data type.

Methods:

Generating word clouds of prevalent words:

```
show_wordcloud(df_vacc_tweet['text'], title = 'Prevalent words in tweets')
```



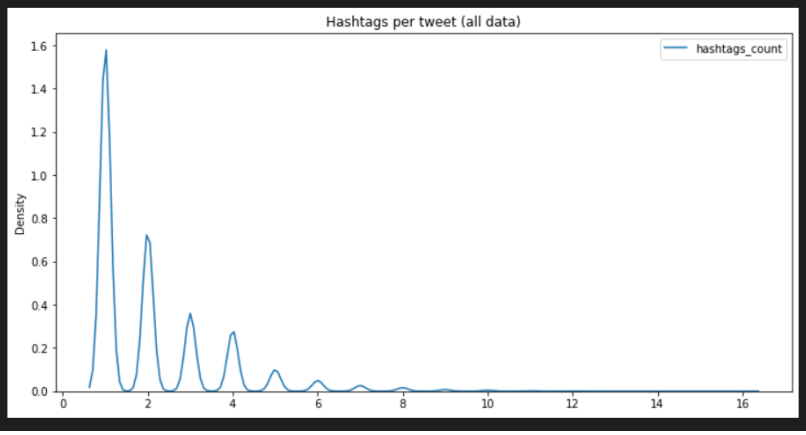
Methods

Performing hashtag analysis on the tweet data to find the overall count of hashtags in a tweet

```
df_vacc_tweet['hashtags'] = df_vacc_tweet['hashtags'].replace(np.nan, "['None']", regex=True)
df_vacc_tweet['hashtags'] = df_vacc_tweet['hashtags'].apply(lambda x: x.replace('\\N',''))
df_vacc_tweet['hashtags_count'] = df_vacc_tweet['hashtags'].apply(lambda x: len(x.split(',')))
plot_features_distribution(['hashtags_count'], 'Hashtags per tweet (all data)', df_vacc_tweet)
```

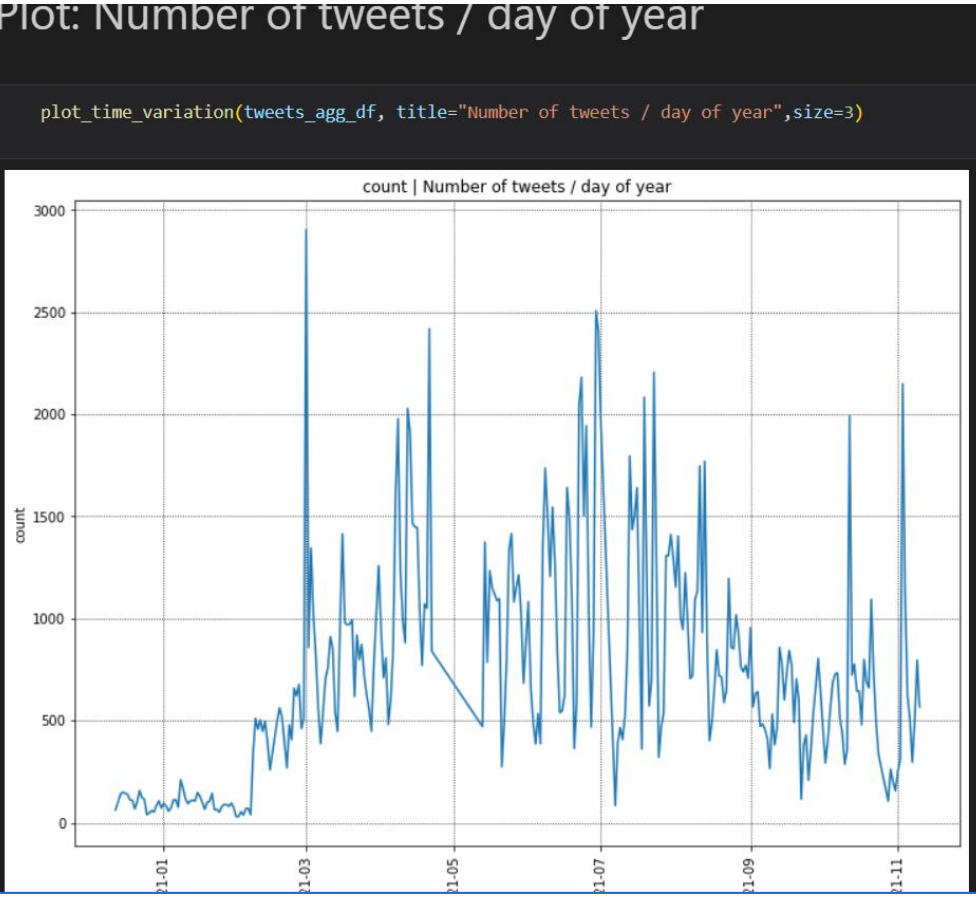
Python

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



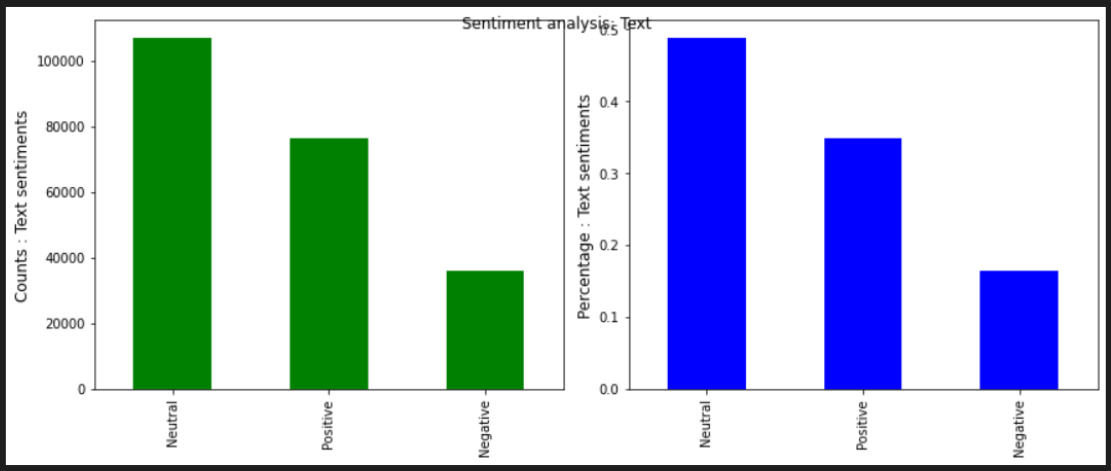
Using plot time variation to plot number of tweets to the day of year

```
plot_time_variation(tweets_agg_df, title="Number of tweets / day of year",size=3)
```



Finding the sentiment score of the tweets and sentiment texts as positive, neutral or negative.

```
tweets_df['sentiment_score'] = tweets_df['text'].apply(lambda x: find_sentiment_score(x))
tweets_df['sentiment'] = tweets_df['text'].apply(lambda x: find_sentiment(x))
plot_sentiment(tweets_df, 'sentiment', 'Text')
```



Getting the learning rate curve and training the sentiment clssifier

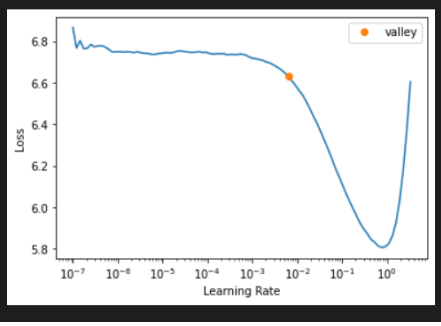
Learning Rate

```
learn.lr_find()
#SuggestedLRs(lr_min=0.04365158379077912, lr_steep=0.019054606556892395
```

Python

```
/usr/local/lib/python3.7/dist-packages/torch/cuda/amp/grad_scaler.py:115: UserWarning: torch.cuda.amp.GradScaler is enabled, but CUDA is not available.  Disabling.
  warnings.warn("torch.cuda.amp.GradScaler is enabled, but CUDA is not available.  Disabling.")


SuggestedLRs(valley=0.0063095735386013985)
```



```
learn.freeze_to(-2)
learn.fit_one_cycle(1, slice(1e-2/(2.6**4),1e-2))
```

Python

```
/usr/local/lib/python3.7/dist-packages/torch/cuda/amp/grad_scaler.py:115: UserWarning: torch.cuda.amp.GradScaler is enabled, but CUDA is not
available.  Disabling.
  warnings.warn("torch.cuda.amp.GradScaler is enabled, but CUDA is not available.  Disabling.")
```

epoch	train_loss	valid_loss	accuracy	time
0	0.775716	0.690291	0.704981	10:27

+ Code + Markdown

```
learn.freeze_to(-3)
learn.fit_one_cycle(1, slice(5e-3/(2.6**4),5e-3))
```

Python

```
/usr/local/lib/python3.7/dist-packages/torch/cuda/amp/grad_scaler.py:115: UserWarning: torch.cuda.amp.GradScaler is enabled, but CUDA is not
available.  Disabling.
  warnings.warn("torch.cuda.amp.GradScaler is enabled, but CUDA is not available.  Disabling.")
```

epoch	train_loss	valid_loss	accuracy	time
0	0.681713	0.609475	0.751277	16:44

```
learn.unfreeze()
learn.fit_one_cycle(3, slice(1e-3/(2.6**4),1e-3))
```

Python

```
/usr/local/lib/python3.7/dist-packages/torch/cuda/amp/grad_scaler.py:115: UserWarning: torch.cuda.amp.GradScaler is enabled, but CUDA is not
available.  Disabling.
  warnings.warn("torch.cuda.amp.GradScaler is enabled, but CUDA is not available.  Disabling.")
```

epoch	train_loss	valid_loss	accuracy	time
0	0.623562	0.601499	0.755587	24:27
1	0.602620	0.593699	0.765645	24:37
2	0.559928	0.594893	0.763570	25:06

Experiments and Results

To perform sentiment analysis, we divided the data into train and test data then performed various classifiers like count vectorizer, Naive bayes, random forest, logistic regression, catboost algorithm, xgboost, SVM and SGD. The final trained model is following.

Stochastic Gradient Descent-SGD Classifier(BINARY CLASSIFICATION)

```
[ ]  from sklearn.linear_model import SGDClassifier

     sgd_clf = SGDClassifier(loss = 'hinge', penalty = 'l2', random_state=0)

     sgd_clf.fit(X_train,y_train)

     sgd_prediction = sgd_clf.predict(X_valid)
     sgd_accuracy = accuracy_score(y_valid,sgd_prediction)
     print("Training accuracy Score    : ",sgd_clf.score(X_train,y_train))
     print("Validation accuracy Score : ",sgd_accuracy )
     print(classification_report(sgd_prediction,y_valid))

     with open('./model_pkl/sgd_binclass.pkl', 'wb') as files:
         pickle.dump(sgd_clf, files)
```

```
Training accuracy Score    :  0.9306430625941652
Validation accuracy Score :  0.9237564772753213
              precision    recall  f1-score   support

           0       0.66      0.84      0.74      5697
           1       0.97      0.94      0.96     38110

    accuracy                           0.92     43807
   macro avg       0.82      0.89      0.85     43807
weighted avg       0.93      0.92      0.93     43807
```

Accuracy Score

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import log_loss

y_pred = sgd_clf.predict(X_valid)

score =accuracy_score(y_valid,y_pred)
print('accuracy is', score)
```

```
accuracy is 0.9237564772753213
```

The accuracy score is .92

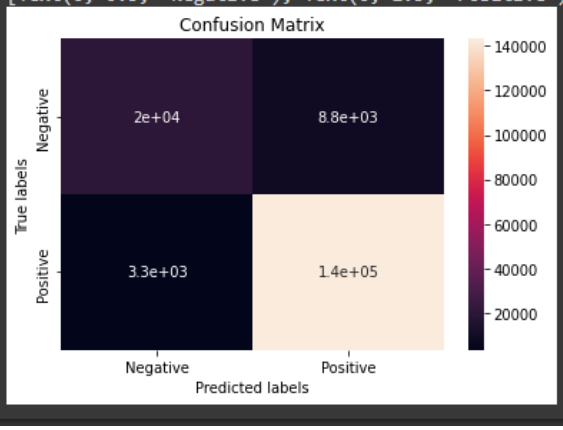
Confusion Matrix

```
labels = ['Negative', 'Positive']
cm = confusion_matrix(y_train, train_class_preds)
print(cm)

ax= plt.subplot()
sns.heatmap(cm, annot=True, ax = ax) #annot=True to annotate cells

# labels, title and ticks
ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('Confusion Matrix')
ax.xaxis.set_ticklabels(labels)
ax.yaxis.set_ticklabels(labels)
```

```
[[ 19879   8836]
 [  3317 143192]]
[Text(0, 0.5, 'Negative'), Text(0, 1.5, 'Positive')]
```

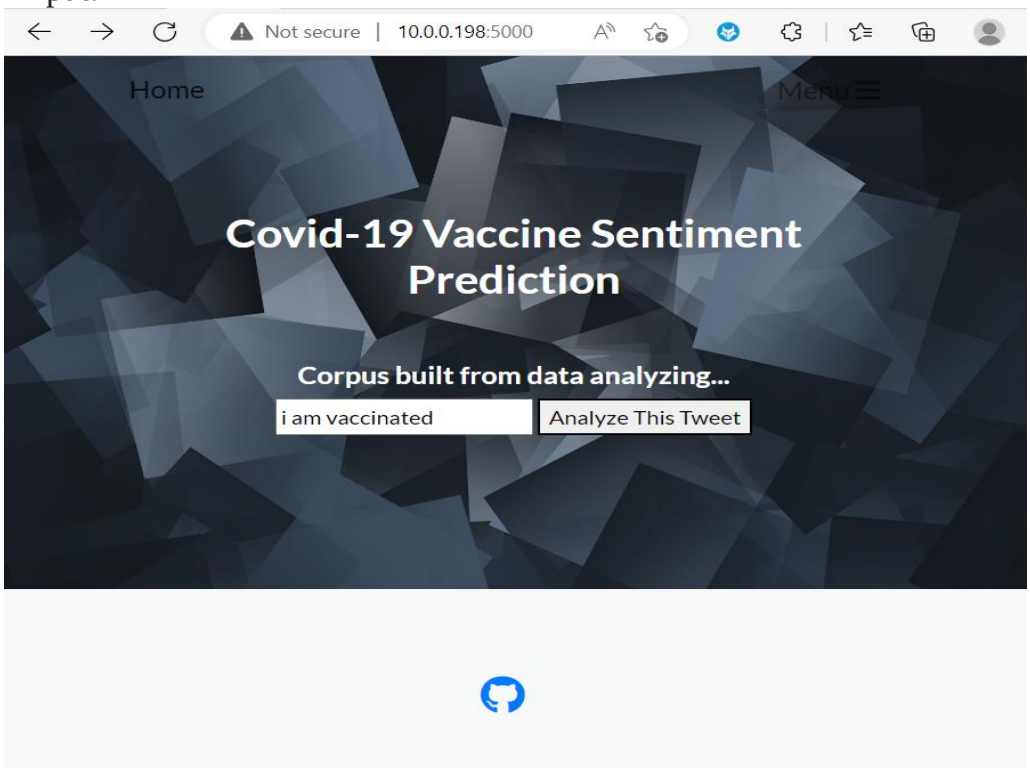


Deployment

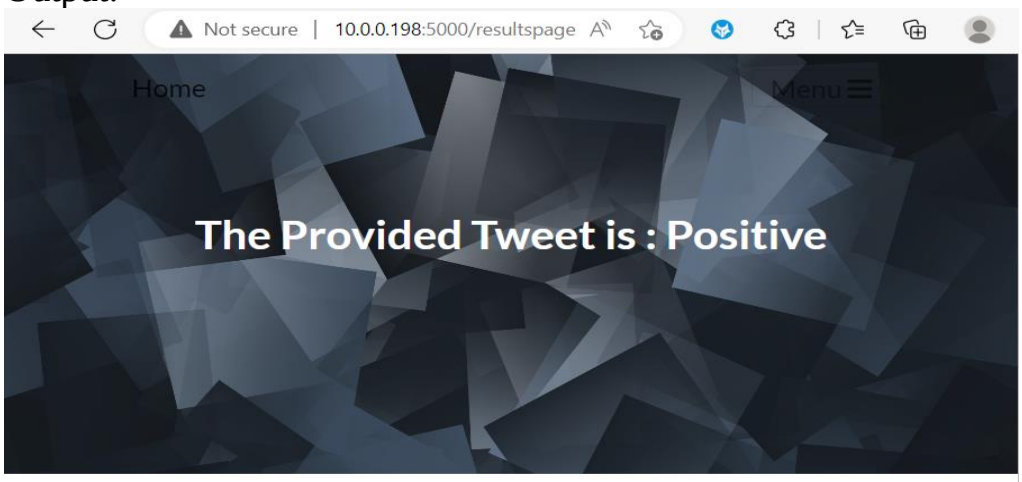
We have made rest API by using flask. And for UI development, we used Python framework. User will provide inputs to our model and in turn we will get the result message either as Positive, Negative or Neutral.

Input values entered by user. Model predicts it as a **Positive**.

Input:

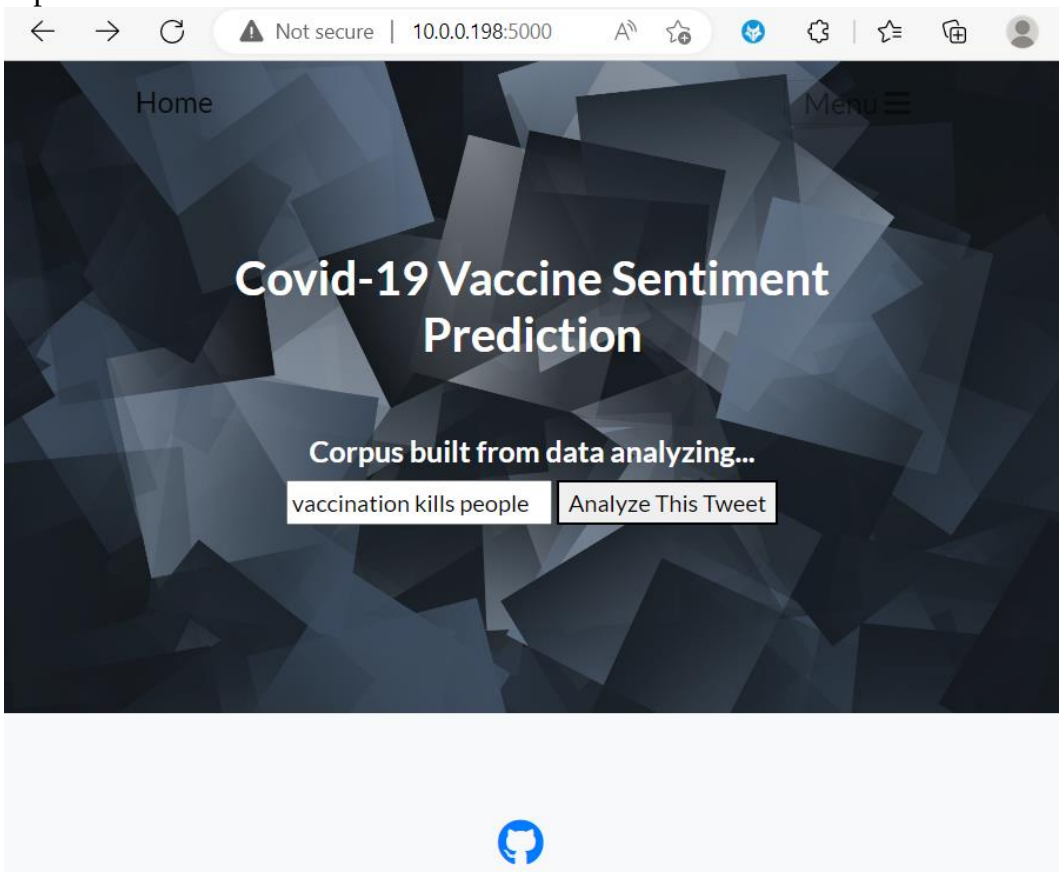


Output:



Input values entered by user, Model predicts it as **Negative**.

Input:



Output:

References

1. https://www.kaggle.com/amartyanambiar/covid-19-vaccine-sentitmental-analysis/notebook
2. https://www.kaggle.com/gpreda/explore-vaccines-tweets - Data-preparation
3. https://towardsdatascience.com/sentiment-analysis-of-covid-19-vaccine-tweets-dc6f41a5e1af