Sentimental analysis using tweets in covid vaccination

downloading and importing modules

```
import nltk
nltk.download('stopwords')
nltk.download('punkt')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
import pandas as pd
import numpy as np
import re
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import style
style.use('ggplot')
from textblob import TextBlob
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
from wordcloud import WordCloud
from sklearn.metrics import accuracy_score,confusion_matrix, ConfusionMatrixDisplay
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

Reading the csv file

```
df = pd.read_csv('/content/vaccination_tweets.csv')
```

Displaying data in csv file

```
df.head()
```

	id	user_name	user_location	user_description	user_created	user_followers	user_friends	user_favou
0	1340539111971516416	Rachel Roh	La Crescenta-Montrose, CA	Aggregator of Asian American news; scanning di...	2009-04-08 17:52:46	405	1692	
1	1338158543359250433	Albert Fong	San Francisco, CA	Marketing dude, tech geek, heavy metal & '80s ...	2009-09-21 15:27:30	834	666	
2	1337858199140118533	eli𝗅𝗍𝖾𝗎 👌	Your Bed	heil, hydra 🖖 ☺	2020-06-25 23:30:28	10	88	
3	1337855739918835717	Charles Adler	Vancouver, BC - Canada	Hosting "CharlesAdlerTonight" Global News Radi...	2008-09-10 11:28:53	49165	3933	
4	1337854064604966912	Citizen News Channel	NaN	Citizen News Channel bringing you an alternati...	2020-04-23 17:58:42	152	580	

Displaying datatype

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11020 entries, 0 to 11019
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   id                11020 non-null  int64 
 1   user_name         11020 non-null  object
 2   user_location     8750 non-null   object
 3   user_description  10341 non-null  object
 4   user_created      11020 non-null  object
 5   user_followers    11020 non-null  int64 
 6   user_friends      11020 non-null  int64 
 7   user_favourites   11020 non-null  int64 
 8   user_verified     11020 non-null  bool  
 9   date              11020 non-null  object
 10  text              11020 non-null  object
 11  hashtags          8438 non-null   object
 12  source            11019 non-null  object
 13  retweets          11020 non-null  int64 
 14  favorites         11020 non-null  int64 
 15  is_retweet        11020 non-null  bool  
dtypes: bool(2), int64(6), object(8)
memory usage: 1.2+ MB
```

Displaying columns names

```
df.columns
```

```
Index(['id', 'user_name', 'user_location', 'user_description', 'user_created',
       'user_followers', 'user_friends', 'user_favourites', 'user_verified',
       'date', 'text', 'hashtags', 'source', 'retweets', 'favorites',
       'is_retweet'],
      dtype='object')
```

Dropping unwanted columns except text column

```
text_df = df.drop(['id', 'user_name', 'user_location', 'user_description', 'user_created',
       'user_followers', 'user_friends', 'user_favourites', 'user_verified',
       'date', 'hashtags', 'source', 'retweets', 'favorites',
       'is_retweet'], axis=1)
```

Displaying the content in text field

```
text_df.head()
```

	text
0	Same folks said daikon paste could treat a cyt...
1	While the world has been on the wrong side of ...
2	#coronavirus #SputnikV #AstraZeneca #PfizerBio...
3	Facts are immutable, Senator, even when you're...
4	Explain to me again why we need a vaccine @Bor...

Displaying Top 5 comments

```
print(text_df['text'].iloc[0],"\n")
print(text_df['text'].iloc[1],"\n")
print(text_df['text'].iloc[2],"\n")
print(text_df['text'].iloc[3],"\n")
print(text_df['text'].iloc[4],"\n")
```

```
Same folks said daikon paste could treat a cytokine storm #PfizerBioNTech https://t.co/xeHhIMg1kF

While the world has been on the wrong side of history this year, hopefully, the biggest vaccination effort we've ev… https://t.co/d

#coronavirus #SputnikV #AstraZeneca #PfizerBioNTech #Moderna #Covid_19 Russian vaccine is created to last 2-4 years… https://t.co/i

Facts are immutable, Senator, even when you're not ethically sturdy enough to acknowledge them. (1) You were born i… https://t.co/j

Explain to me again why we need a vaccine @BorisJohnson @MattHancock #whereareallthesickpeople #PfizerBioNTech… https://t.co/KxbSRo
```

Checking the info of text field

```
text_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11020 entries, 0 to 11019
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   text    11020 non-null  object
dtypes: object(1)
memory usage: 86.2+ KB
```

Preprocessing the text data

```
def data_processing(text):
    text = text.lower()
    text = re.sub(r"https\S+|www\S+https\S+", '',text, flags=re.MULTILINE)
    text = re.sub(r'\@w+|\#','',text)
    text = re.sub(r'[^\w\s]','',text)
    text_tokens = word_tokenize(text)
    filtered_text = [w for w in text_tokens if not w in stop_words]
    return " ".join(filtered_text)
```

Storing the processed data in text_df

```
text_df.text = text_df['text'].apply(data_processing)
```

Dropping the duplicate Texts in our Processed data

```
text_df = text_df.drop_duplicates('text')
```

```
stemmer = PorterStemmer()
def stemming(data):
    text = [stemmer.stem(word) for word in data]
    return data
```

```
text_df['text'] = text_df['text'].apply(lambda x: stemming(x))
```

Showing the Processed data

```
text_df.head()
```

	text
0	folks said daikon paste could treat cytokine s...
1	world wrong side history year hopefully bigges...
2	coronavirus sputnikv astrazeneca pfizerbiontec...
3	facts immutable senator even youre ethically s...
4	explain need vaccine borisjohnson matthancock ...

Showing the Top 5 texts after pre-processing

```
print(text_df['text'].iloc[0],"\n")
print(text_df['text'].iloc[1],"\n")
print(text_df['text'].iloc[2],"\n")
print(text_df['text'].iloc[3],"\n")
print(text_df['text'].iloc[4],"\n")
```

```
folks said daikon paste could treat cytokine storm pfizerbiontech

world wrong side history year hopefully biggest vaccination effort weve ev

coronavirus sputnikv astrazeneca pfizerbiontech moderna covid_19 russian vaccine created last 24 years

facts immutable senator even youre ethically sturdy enough acknowledge 1 born

explain need vaccine borisjohnson matthancock whereareallthesickpeople pfizerbiontech
```

```
text_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10543 entries, 0 to 11019
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   text    10543 non-null  object
dtypes: object(1)
memory usage: 164.7+ KB
```

Checking the Polarity of each texts using Textblob

```
def polarity(text):
    return TextBlob(text).sentiment.polarity
```

Storing the polarity of each comment in text_df by creating a column for Polarity

```
text_df['polarity'] = text_df['text'].apply(polarity)
```

displaying the polarity of top 10 texts using head function

```
text_df.head(10)
```

	text	polarity
0	folks said daikon paste could treat cytokine s...	0.000
1	world wrong side history year hopefully bigges...	-0.500
2	coronavirus sputnikv astrazeneca pfizerbiontec...	0.000
3	facts immutable senator even youre ethically s...	0.100
4	explain need vaccine borisjohnson matthancock ...	0.000
5	anyone useful adviceguidance whether covid vac...	0.400
6	bit sad claim fame success vaccination patriot...	-0.100
7	many bright days 2020 best 1 bidenharris winni...	0.675
8	covid vaccine getting covidvaccine covid19 pfi...	0.000
9	covidvaccine states start getting covid19vacci...	0.000

Based on Polarity labeling the sentiment

```
def sentiment(label):
    if label <0:
        return "Negative"
    elif label ==0:
        return "Neutral"
    elif label>0:
        return "Positive"
```

Storing the Sentiment of each comment in text_df by creating a column for Sentiment

```
text_df['sentiment'] = text_df['polarity'].apply(sentiment)
```

Displaying the text_df

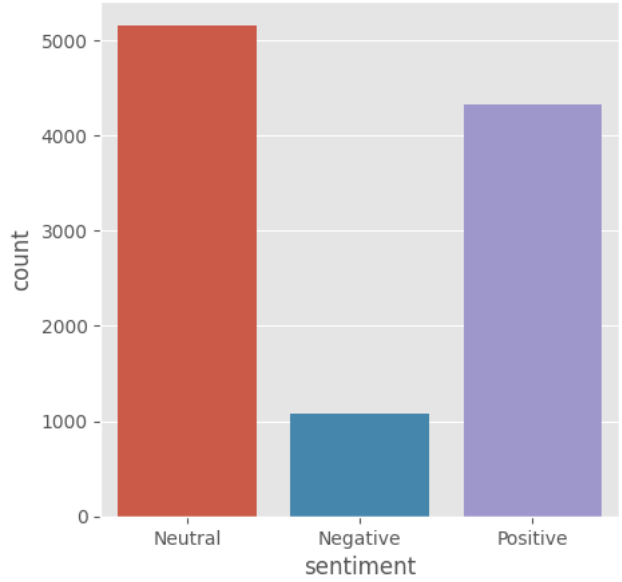
```
text_df.head()
```

	text	polarity	sentiment
0	folks said daikon paste could treat cytokine s...	0.0	Neutral
1	world wrong side history year hopefully bigges...	-0.5	Negative
2	coronavirus sputnikv astrazeneca pfizerbiontec...	0.0	Neutral
3	facts immutable senator even youre ethically s...	0.1	Positive
4	explain need vaccine borisjohnson matthancock ...	0.0	Neutral

Plotting a graph based on the counts of each sentiment in text_df

```
fig = plt.figure(figsize=(5,5))
sns.countplot(x='sentiment', data = text_df)
```

```
<Axes: xlabel='sentiment', ylabel='count'>
```



Diplaying only positive sentiment tweets

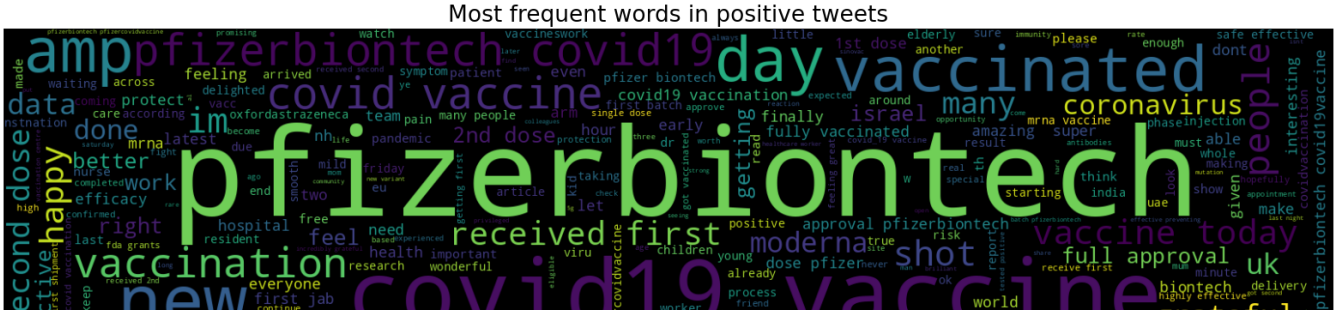
```
pos_tweets = text_df[text_df.sentiment == 'Positive']
pos_tweets = pos_tweets.sort_values(['polarity'], ascending= False)
pos_tweets.head()
```

	text	polarity	sentiment
9317	best way get merrygoround pfizer pfizerbiontec...	1.0	Positive
2340	applying emotion pfizerbiontech based best evi...	1.0	Positive
6295	pfizer jab morning efficient wellorganised tha...	1.0	Positive
5041	get art printed awesome products support redbu...	1.0	Positive
1055	already vaccinated getting vaccine soon plan t...	1.0	Positive

Displaying the most frequent words used in positive tweets and showing wordcloud of words

```
text = ' '.join([word for word in pos_tweets['text']])
plt.figure(figsize=(20,15), facecolor='None')
wordcloud = WordCloud(max_words=500, width=1600, height=800).generate(text)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title('Most frequent words in positive tweets', fontsize=19)
plt.show()
```

Most frequent words in positive tweets



Diplaying only negative sentiment tweets

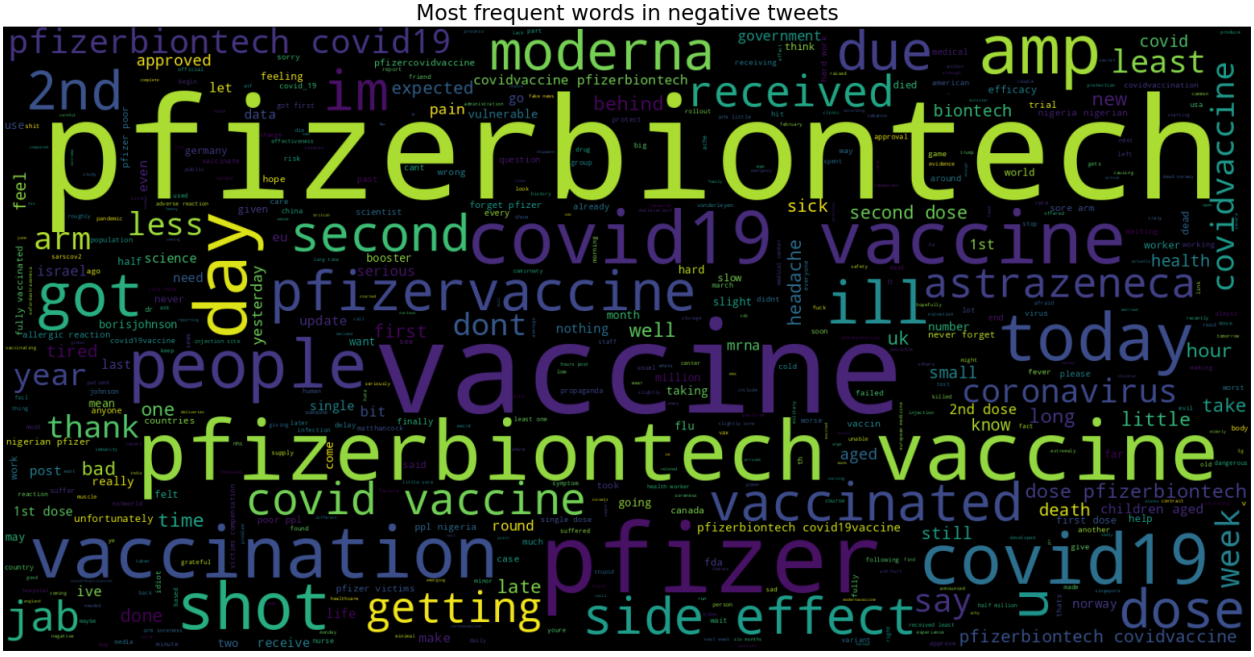
```
neg_tweets = text_df[text_df.sentiment == 'Negative']
neg_tweets = neg_tweets.sort_values(['polarity'], ascending= False)
neg_tweets.head()
```

	text	polarity	sentiment
2912	work skilled nursing facility got first vaccin...	-0.003333	Negative
7256	200321 752308 vaccinations new daily record da...	-0.003409	Negative
2073	ukgovernment cant even vaccinate properly ethi...	-0.004762	Negative
7715	got first dose less waiting time airport vacci...	-0.005556	Negative
7157	nas_k27 second dose due end next month well fa...	-0.006250	Negative

Displaying the most frequent words used in negative tweets and showing wordcloud of words

```
text = ' '.join([word for word in neg_tweets['text']])
plt.figure(figsize=(20,15), facecolor='None')
wordcloud = WordCloud(max_words=500, width=1600, height=800).generate(text)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title('Most frequent words in negative tweets', fontsize=19)
plt.show()
```

Most frequent words in negative tweets



Diplaying only neutral sentiment tweets

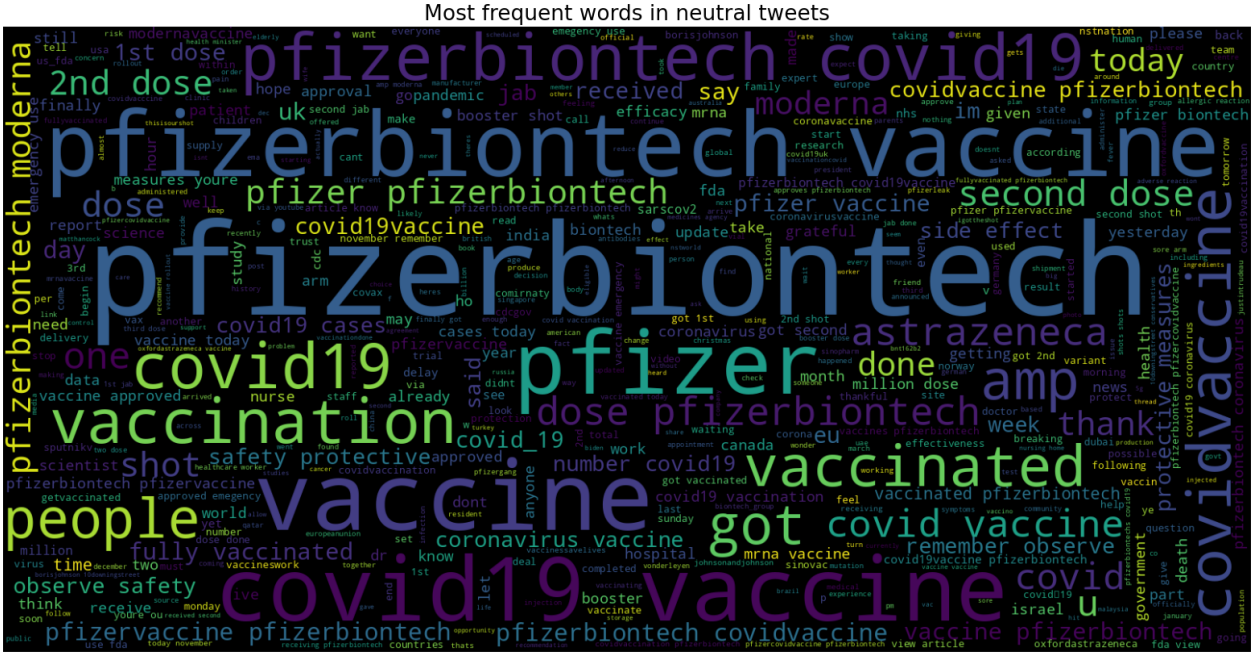
```
neutral_tweets = text_df[text_df.sentiment == 'Neutral']
neutral_tweets = neutral_tweets.sort_values(['polarity'], ascending= False)
neutral_tweets.head()
```

	text	polarity	sentiment
0	folks said daikon paste could treat cytokine s...	0.0	Neutral
7347	anyone else feel like framing vaccine card pfi...	0.0	Neutral
7458	looking forward getting second pfizer shot any...	0.0	Neutral
7454	never thought id running diff vaccine modernav...	0.0	Neutral
7453	john___m dont get choose one person know asked...	0.0	Neutral

Displaying the most frequent words used in tweets neutral and showing wordcloud of words

```
text = ' '.join([word for word in neutral_tweets['text']])
plt.figure(figsize=(20,15), facecolor='None')
wordcloud = WordCloud(max_words=500, width=1600, height=800).generate(text)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title('Most frequent words in neutral tweets', fontsize=19)
plt.show()
```

Most frequent words in neutral tweets



Converting the texts into vector

```
vect = CountVectorizer(ngram_range=(1,2)).fit(text_df['text'])
```

Marking the Independent and dependent labels

```
X = text_df['text']
Y = text_df['sentiment']
X = vect.transform(X)
```

splittig the cleaned data into training and testing the models

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

Printing the shape of training and testing data

```
print("Size of x_train:", (x_train.shape))
print("Size of y_train:", (y_train.shape))
print("Size of x_test:", (x_test.shape))
print("Size of y_test:", (y_test.shape))
```

```
Size of x_train: (8434, 78583)
Size of y_train: (8434,)
Size of x_test: (2109, 78583)
Size of y_test: (2109,)
```

importing warning for removing the warning text in outputs

```
import warnings
warnings.filterwarnings('ignore')
```

Training the data on logistic regression model to find its accuracy

```
logreg = LogisticRegression()
logreg.fit(x_train, y_train)
logreg_pred = logreg.predict(x_test)
logreg_acc = accuracy_score(logreg_pred, y_test)
print("Test accuracy: {:.2f}%".format(logreg_acc*100))
```

```
Test accuracy: 84.64%
```

Confusion matrix

```
style.use('classic')
cm = confusion_matrix(y_test, logreg_pred, labels=logreg.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix = cm, display_labels=logreg.classes_)
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fcfa9548a60>
```

Importing the SVM mdoule , fitting the data and finding the acuuracy

```
from sklearn.svm import LinearSVC
SVCmodel = LinearSVC()
SVCmodel.fit(x_train, y_train)
svc_pred = SVCmodel.predict(x_test)
svc_acc = accuracy_score(svc_pred, y_test)
print("test accuracy: {:.2f}%".format(svc_acc*100))
```

```
test accuracy: 87.34%
```

Confusion matrix for SVM

```
style.use('classic')
cm = confusion_matrix(y_test,svc_pred, labels=SVCmodel.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix = cm, display_labels=SVCmodel.classes_)
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fcfa9577550>
```

Roberta pretrained model for sentimental analysis with the highest accurcy of 88.5%

installing modules and imorting them

```
!pip install torch
!pip install transformers
!pip install pandas
```

```
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (2.0.1+cu118)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch) (3.12.2)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dist-packages (from torch) (4.6.3)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch) (1.11.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch) (3.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch) (3.1.2)
Requirement already satisfied: triton==2.0.0 in /usr/local/lib/python3.10/dist-packages (from torch) (2.0.0)
Requirement already satisfied: cmake in /usr/local/lib/python3.10/dist-packages (from triton==2.0.0->torch) (3.25.2)
Requirement already satisfied: lit in /usr/local/lib/python3.10/dist-packages (from triton==2.0.0->torch) (16.0.6)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch) (2.1.3)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch) (1.3.0)
Collecting transformers
  Downloading transformers-4.30.2-py3-none-any.whl (7.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.2/7.2 MB 64.3 MB/s eta 0:00:00
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers) (3.12.2)
Collecting huggingface-hub<1.0,>=0.14.1 (from transformers)
  Downloading huggingface_hub-0.16.4-py3-none-any.whl (268 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 268.8/268.8 kB 34.4 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (1.22.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (23.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (6.0)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (2022.10.31)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers) (2.27.1)
Collecting tokenizers!=0.11.3,<0.14,>=0.11.1 (from transformers)
  Downloading tokenizers-0.13.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (7.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 137.6 MB/s eta 0:00:00
Collecting safetensors>=0.3.1 (from transformers)
  Downloading safetensors-0.3.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 83.3 MB/s eta 0:00:00
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers) (4.65.0)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.14.1->transformers)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (1.26
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2023.5.
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.4)
Installing collected packages: tokenizers, safetensors, huggingface-hub, transformers
Successfully installed huggingface-hub-0.16.4 safetensors-0.3.1 tokenizers-0.13.3 transformers-4.30.2
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (1.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2022.7.1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.22.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
```

```
from transformers import AutoTokenizer
from transformers import AutoModelForSequenceClassification
from scipy.special import softmax
```

```
MODEL = f"cardiffnlp/twitter-roberta-base-sentiment"
tokenizer = AutoTokenizer.from_pretrained(MODEL)
model = AutoModelForSequenceClassification.from_pretrained(MODEL)
```

```
Downloading (…)lve/main/config.json: 100%  747/747 [00:00<00:00, 26.8kB/s]
Downloading (…)olve/main/vocab.json: 100%  899k/899k [00:00<00:00, 12.0MB/s]
Downloading (…)olve/main/merges.txt: 100%  456k/456k [00:00<00:00, 9.37MB/s]
Downloading (…)cial_tokens_map.json: 100%  150/150 [00:00<00:00, 9.76kB/s]
Downloading pytorch_model.bin: 100%  499M/499M [00:05<00:00, 92.9MB/s]
```

Tesing the user Sentiments

```
example=input("enter the comment:")
# Run for Roberta Model
encoded_text = tokenizer(example, return_tensors='pt')
output = model(**encoded_text)
scores = output[0][0].detach().numpy()
scores = softmax(scores)
scores_dict = {
    'roberta_neg' : scores[0],
    'roberta_neu' : scores[1],
    'roberta_pos' : scores[2]
}
print(scores_dict)
```

```
 enter the comment:i am in good mood
 {'roberta_neg': 0.0014560512, 'roberta_neu': 0.019181935, 'roberta_pos': 0.979362}
```

The End