```
pip install tweepy==4.10.1
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/publ
Collecting tweepy==4.10.1
  Downloading tweepy-4.10.1-py3-none-any.whl (94 kB)
     |████████████████████████████████| 94 kB 2.1 MB/s
Requirement already satisfied: requests-oauthlib<2,>=1.2.0 in /usr/local/lib/python3.7/d
Requirement already satisfied: oauthlib<4,>=3.2.0 in /usr/local/lib/python3.7/dist-packa
Collecting requests<3,>=2.27.0
  Downloading requests-2.28.1-py3-none-any.whl (62 kB)
     |████████████████████████████████| 62 kB 1.3 MB/s
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: charset-normalizer<3,>=2 in /usr/local/lib/python3.7/dist
Installing collected packages: requests, tweepy
  Attempting uninstall: requests
    Found existing installation: requests 2.23.0
    Uninstalling requests-2.23.0:
      Successfully uninstalled requests-2.23.0
  Attempting uninstall: tweepy
    Found existing installation: tweepy 3.10.0
    Uninstalling tweepy-3.10.0:
      Successfully uninstalled tweepy-3.10.0
Successfully installed requests-2.28.1 tweepy-4.10.1
```

```
pip show tweepy
```

```
Name: tweepy
Version: 4.10.1
Summary: Twitter library for Python
Home-page: https://www.tweepy.org/
Author: Joshua Roesslein
Author-email: tweepy@googlegroups.com
License: MIT
Location: /usr/local/lib/python3.7/dist-packages
Requires: requests, oauthlib, requests-oauthlib
Required-by:
```

```
import os
import tweepy as twep
import pandas as pd
```

```
#keys to access the twitter API
consumer_key= 'Y3RJeJFRHnA9QnpZu8z9S0Skb'
consumer_secret= 'mSHz1KdYkTNmYWuxMwX4fMzzGc9qv1OqgyQxWU26pheoGRcLuE'
access_token= '1287803472805367808-EedGVpYxgeXKILni2gt9HvJacgsmxx'
access_token_secret= 'K9jXNXr6i72odGz9XAiUnN5841eIUiDKcrj52RozRO743'
```

```
#Code to access the api and authentication to connect to twitter API
auth = twep.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = twep.API(auth, wait_on_rate_limit=True)


#code to search the Tweets with keyword tesla and add them to a text file
keyword = 'Tesla'
limit=1000
tweets = twep.Cursor(api.search_tweets, q=keyword, tweet_mode='extended').items(limit)
file = open('twitter.txt', 'w', encoding="utf-8")
for tweet in tweets:
    file.write(tweet.full_text+'\n')
file.close()


#code to search the Tweets with keyword #TSLA and add them to a text file
keyword = '#TSLA'
limit=1000
tweets = twep.Cursor(api.search_tweets, q=keyword, tweet_mode='extended').items(limit)
file = open('twitter.txt', 'a', encoding="utf-8")
for tweet in tweets:
    file.write(tweet.full_text+'\n')
file.close()


#code to search the Tweets with keyword yahoofinance and Tesla and add them to a text file
keyword = 'yahoofinance and Tesla'
limit=1000
tweets = twep.Cursor(api.search_tweets, q=keyword, tweet_mode='extended').items(limit)
file = open('twitter.txt', 'a', encoding="utf-8")
for tweet in tweets:
    file.write(tweet.full_text+'\n')
file.close()


#opening file to write the processed text
file = open('processed.txt', 'w', encoding="utf-8")


#preprocessing
import re
with open('twitter.txt','r', encoding="utf-8") as f:
    lines = f.readlines()
f.close()
for line in lines:
    content=' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ", line).spl
    file.write(content+'.'+'\n')
file.close()


#Converting the input file into the list of sentences.
Input = open("processed.txt", "r")
```

```
data = Input.read()
data_to_list = data.split("\n")
Input.close()
```

```
#Adding the input text that needs to be classified to the TEST variable
Test_X=[]
for x in data_to_list:
    Test_X.append(x)
```

```
import nltk
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
True
```

```
#removing the stopwords and peroforming the lemmatization and appending back.
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
nltk.download('stopwords')
stopword = set(stopwords.words('english'))
Test_data=[]
for x in Test_X:
    tokens = word_tokenize(str(x))
    final_tokens = [w for w in tokens if w not in stopword]
    wordLemm = WordNetLemmatizer()
    finalwords=[]
    for w in final_tokens:
        if len(w)>1:
            word = wordLemm.lemmatize(w)
            finalwords.append(word)
    Test_data.append(' '.join(finalwords))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
#Removing duplicate and single word sentences.
Test_X=[]
for x in Test_data:
    if len(x)>10:
        Test_X.append(x)
Test_X = [*set(Test_X)]
```

In the below 2 cells, read the input from 2 different sets that are related to the stock sentiment to train the model.

```
twit = pd.read_csv("all-data.csv", encoding = "latin-1")
Train_Y=twit["Sentiment"]
Train_X=twit["Text"]
```

```
twit = pd.read_csv("stock_data.csv", encoding = "latin-1")
for ind in twit.index:
    if(twit['Sentiment'][ind]==-1):
        twit['Sentiment'][ind]="negative"
    else:
        twit['Sentiment'][ind]="positive"
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_

/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1732: SettingWithCopyWarn
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
  self._setitem_single_block(indexer, value, name)
```

```
Train_X=Train_X.append(twit["Text"])
Train_Y=Train_Y.append(twit["Sentiment"])
```

```
Train_X.shape
```

```
(10637,)
```

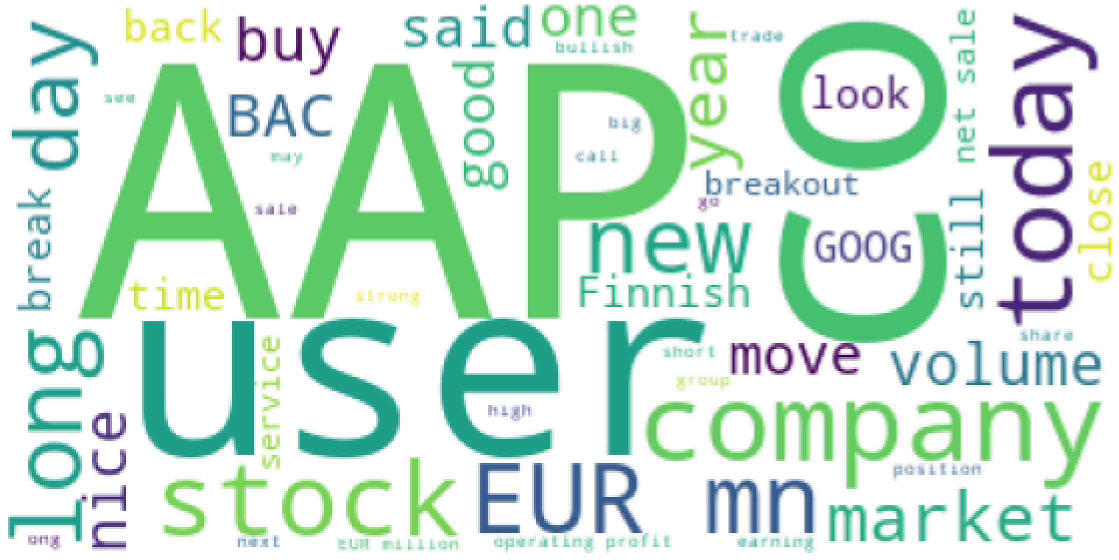
```
#cleaned the train data by removing the stop words and doing the lemmatization
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
nltk.download('stopwords')
stopword = set(stopwords.words('english'))
Train_data=[]
for x in Train_X:
    tokens = word_tokenize(str(x))
    final_tokens = [w for w in tokens if w not in stopword]
    wordLemm = WordNetLemmatizer()
    finalwords=[]
    for w in final_tokens:
        if len(w)>1:
            word = wordLemm.lemmatize(w)
            finalwords.append(word)
    Train_data.append(' '.join(finalwords))
Train_X= Train_data
```

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

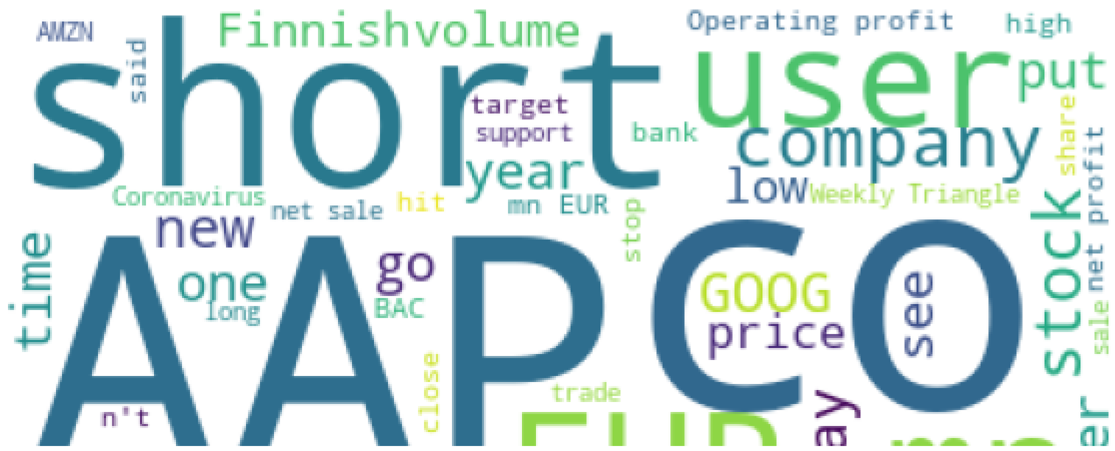
```
#converted the list to Pandas data frame for analysis
df = pd.DataFrame(list(zip(Train_X, Train_Y)),
               columns =['Text', 'Sentiment'])
```

```
#created the wordcloud method to display the words with the sentiment values
from wordcloud import WordCloud
from matplotlib import pyplot as plt

def DisplayWordCloud(input,bcol):
    plt.figure(figsize=(10,10))
    wocl=WordCloud(background_color=bcol,max_words=50, min_word_length=2, contour_width=1, co
    wocl.generate(" ".join(input))
    plt.imshow(wocl)
    plt.axis("off")
```

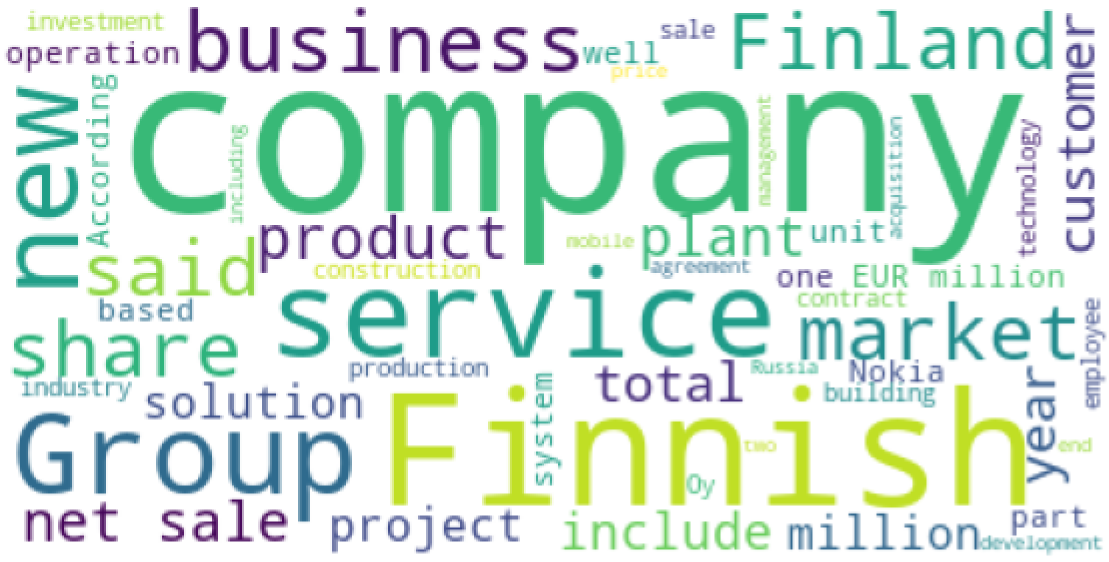
```
DisplayWordCloud(df[df.Sentiment=="positive"].Text, 'white')
```



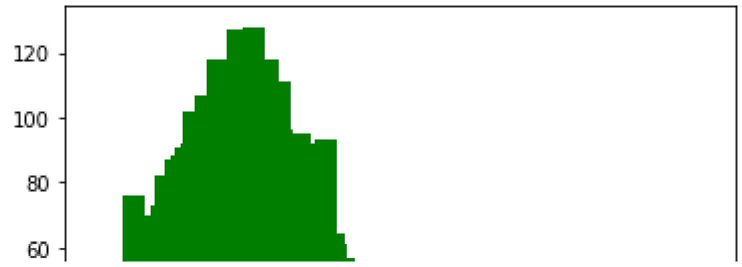
```
DisplayWordCloud(df[df.Sentiment=="negative"].Text, 'white')
```



```
DisplayWordCloud(df[df.Sentiment=="neutral"].Text, 'white')
```

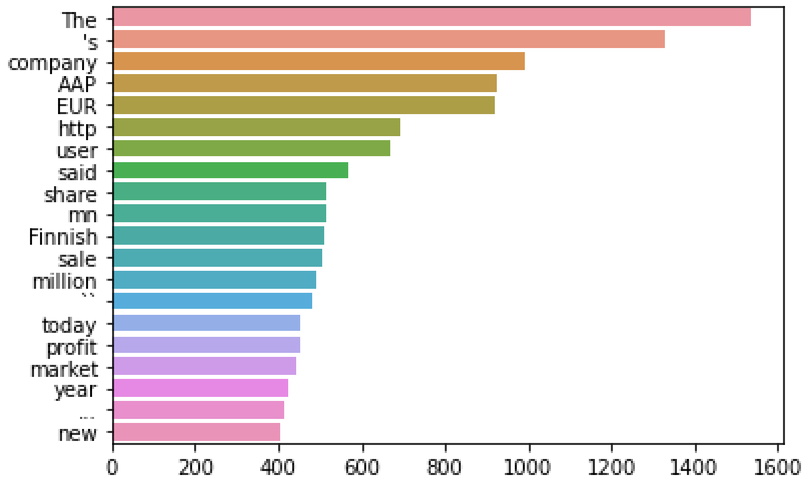


```
#frequency of length of sentences in the Train dataset
from collections import OrderedDict
freq = {}
for line in Train_X:
  l=len(line)
  if (l in freq):
    freq[l] += 1
  else:
    freq[l] = 1
final_dict = OrderedDict(sorted(freq.items()))
plt.bar(final_dict.keys(), final_dict.values(), 10, color='g')
plt.show()
```



```
#most common words in the twitter text
from collections import Counter
import nltk
import seaborn as sns

nltk.download('stopwords')
stop=set(stopwords.words('english'))
Input_str=[]
for line in Train_X:
    word_list= line.split()
    for word in word_list:
      Input_str.append(word)
count=Counter(Input_str)
common=count.most_common()
x, y= [], []
for word,count in common[:20]:
    if (word not in stop):
        x.append(word)
        y.append(count)

sns.barplot(x=y,y=x)
```

```
     [nltk_data] Downloading package stopwords to /root/nltk_data...
     [nltk_data]   Package stopwords is already up-to-date!
     <matplotlib.axes._subplots.AxesSubplot at 0x7f7d751c0890>
```

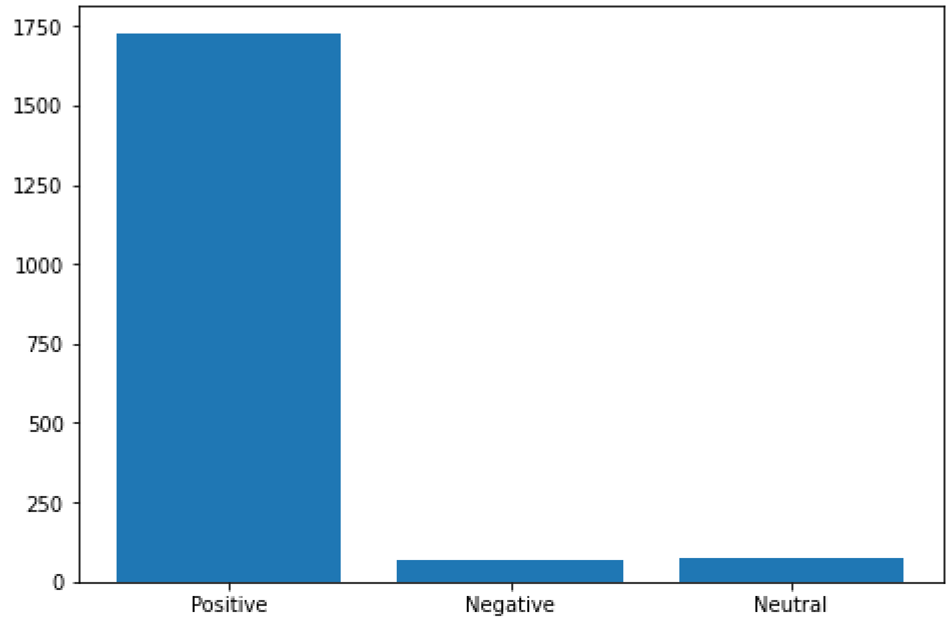


```
#creating a pipeline withe Tf-IDF vector and multinomailNaive bayes classifier as we have pos
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline
```

```
model = make_pipeline(TfidfVectorizer(), MultinomialNB())



#model is trained using FIt method
model.fit(Train_X, Train_Y)
labels = model.predict(Test_X)


#lables for the input are counted based on the sentiment
Final_lables=labels.tolist()
pcount=Final_lables.count("positive")
ncount=Final_lables.count("negative")
necount=Final_lables.count("neutral")
pcount,ncount,necount
```

```
     (1729, 70, 74)
```

```
#plotted a bar chart for the lables that are predicted for the input.
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
Sentiment = ['Positive', 'Negative','Neutral']
Count = [pcount,ncount,necount]
ax.bar(Sentiment,Count)
plt.show()
```


Positive
Negative
Neutral