In [2]:

```
###### Business Understanding
Market sentiment is a qualitative measure of the attitude and mood of investors to financial market
s in general,
and specific sectors or assets in particular. Positive and negative sentiment drive price action,
and also create trading and investment opportunities for active traders and long-term investors ###
#
######

######## Table of content ############

#1. importing stock market data from kaggle
#2. importing python libraries along with NLTK
#3. plotting of text
#4. stopwords finding and wordcloud of  stopwords
#5. Data cleaning using stemming, lemmatization, lower words, upper words, tokenization
#6. data partitioning
#7. Model building using randomforest classifier, naive bayes, SVM, ADAboost, XGboost
#8. Finding Accuracy, precision, recall, through confusion matrix
#9. choose the best model ###########################################################################


import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
%matplotlib inline  ##### %matplotlib inline sets the backend of matplotlib to the 'inline' backen
d: With this backend, the output of plotting commands is displayed inline within frontends like th
e Jupyter notebook, directly below the code cell that produced it. The resulting plots will then a
lso be stored in the notebook document
from sklearn.feature_extraction.text CountVectorizer
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklear.ensemble import RandomForestClassifier
from nltk.stem.porter import PorterStemmer
from nltk import word_tokenize,WordNetLemmatizer
import nltk
import re
nltk.download("wordnet")
```

```
C:\Users\Dipsikha\anaconda3\lib\importlib\_bootstrap.py:219: RuntimeWarning: numpy.ufunc size
changed, may indicate binary incompatibility. Expected 192 from C header, got 216 from PyObject
  return f(*args, **kwds)
C:\Users\Dipsikha\anaconda3\lib\importlib\_bootstrap.py:219: RuntimeWarning: numpy.ufunc size
changed, may indicate binary incompatibility. Expected 192 from C header, got 216 from PyObject
  return f(*args, **kwds)
UsageError: unrecognized arguments: ##### %matplotlib inline sets the backend of matplotlib to the
'inline' backend: With this backend, the output of plotting commands is displayed inline within fr
ontends like the Jupyter notebook, directly below the code cell that produced it. The resulting pl
ots will then also be stored in the notebook document
```

In [3]:

```
data = pd.read_csv("stock_data.csv")
```

In [4]:

```
data
```

Out[4]:

	Text	Sentiment
0	Kickers on my watchlist XIDE TIT SOQ PNK CPW B...	1
1	user: AAP MOVIE. 55% return for the FEA/GEED i...	1

	Text	Sentiment
2	user I'd be afraid to short AMZN - they are lo...	1
3	MNTA Over 12.00	1
4	OI Over 21.37	1
...	...	...
5786	Industry body CII said #discoms are likely to ...	-1
5787	#Gold prices slip below Rs 46,000 as #investor...	-1
5788	Workers at Bajaj Auto have agreed to a 10% wag...	1
5789	#Sharemarket LIVE: Sensex off day’s high, up 6...	1
5790	#Sensex, #Nifty climb off day's highs, still u...	1

5791 rows × 2 columns

In [5]:

```
data.head()
```

Out[5]:

	Text	Sentiment
0	Kickers on my watchlist XIDE TIT SOQ PNK CPW B...	1
1	user: AAP MOVIE. 55% return for the FEA/GEED i...	1
2	user I'd be afraid to short AMZN - they are lo...	1
3	MNTA Over 12.00	1
4	OI Over 21.37	1

In [6]:

```
data.info
```

Out[6]:

```
<bound method DataFrame.info of                                                 Text  Sentiment
0     Kickers on my watchlist XIDE TIT SOQ PNK CPW B...          1
1     user: AAP MOVIE. 55% return for the FEA/GEED i...          1
2     user I'd be afraid to short AMZN - they are lo...          1
3                                   MNTA Over 12.00              1
4                                    OI  Over 21.37              1
...                                                ...        ...
5786  Industry body CII said #discoms are likely to ...         -1
5787  #Gold prices slip below Rs 46,000 as #investor...         -1
5788  Workers at Bajaj Auto have agreed to a 10% wag...          1
5789  #Sharemarket LIVE: Sensex off day’s high, up 6...          1
5790  #Sensex, #Nifty climb off day's highs, still u...          1

[5791 rows x 2 columns]>
```

In [7]:

```
data.describe()
```

Out[7]:

	Sentiment
count	5791.000000
mean	0.272664
std	0.962192
min	-1.000000
25%	-1.000000
50%	1.000000

	Sentiment
75%	1.000000
max	1.000000

In [8]:

```
data.shape
```

Out[8]:

```
(5791, 2)
```

In [9]:

```
########### sentiment count ############

data["Sentiment"].value_counts()
```

Out[9]:

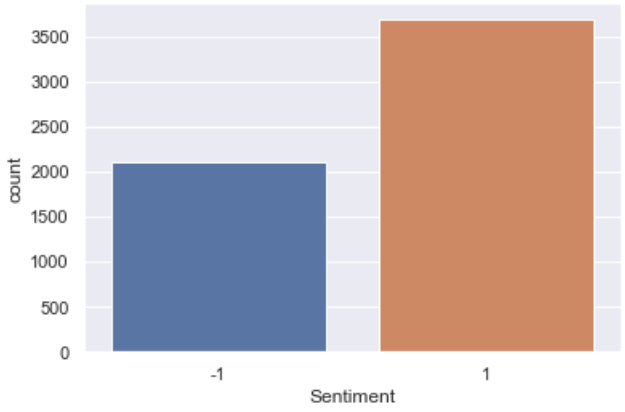
```
 1    3685
-1    2106
Name: Sentiment, dtype: int64
```

In [10]:

```
sns.countplot(data['Sentiment'])
```

Out[10]:

```
<AxesSubplot:xlabel='Sentiment', ylabel='count'>
```



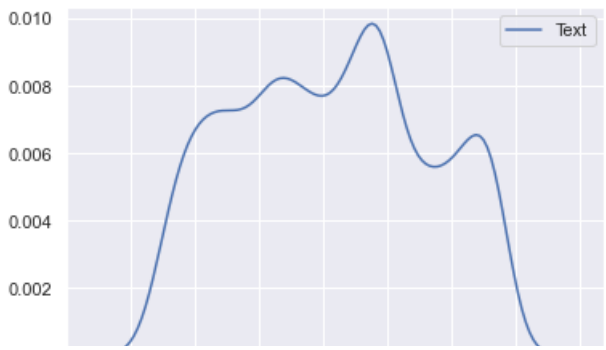
In [11]:

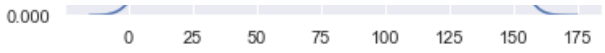
```
###### length of the using KDEplot ###############

sns.kdeplot(data['Text'].str.len())
```

Out[11]:

```
<AxesSubplot:>
```





In [13]:

```
######## checking for stopwords ############

from nltk.corpus import stopwords
stop_words=set(stopwords.words("english"))
print(stop_words)
```

```
{'is', 'such', 'his', 'y', 'into', 'yourself', 'theirs', 'again', 'd', 'too', 'ourselves', 'isn',
're', 'shouldn', 'with', "didn't", 'been', "weren't", 'or', 'off', 'he', 'nor', 'so', "hadn't",
'couldn', "that'll", 'its', 'did', 'where', 'than', 'to', 'wasn', 'more', 'that', 'being',
"isn't", 'our', "mightn't", 'here', 'a', "shouldn't", 'there', 'under', "wasn't", 'below', 'your',
'this', 'through', 'an', 'am', 'as', 'until', 'should', 'are', 'any', 'wouldn', "hasn't",
"haven't", 'what', 't', 'him', 'hasn', 'doesn', "it's", 'why', 'won', 'ma', "needn't", 'himself',
"won't", 'it', 'few', 'now', 'them', 'about', 'against', 'further', 'how', 'my', 'and', 'on',
'once', 'up', 'those', 'the', 'whom', 'only', 'above', 'haven', "shan't", 'between', "aren't", 'mi
ghtn', "don't", 'for', 'you', 'they', 'had', 'of', 'will', 'all', 'not', "should've", 'while', 'no
', 'o', 'yourselves', 'over', 'who', 'in', 'by', 'just', 'hadn', "doesn't", 'their', 'after',
'aren', "mustn't", 'other', 'themselves', 'mustn', 'own', 'needn', 'when', "couldn't", "you've", '
we', 'during', "you'll", 'having', 'which', 'same', 's', "wouldn't", 'have', 'itself', 'these', 'f
rom', 'very', 'but', 'can', 'be', 'shan', 'myself', "you'd", 'don', 'down', 'has', 'ours', 'ain',
'before', "she's", 'does', 'm', 'at', 'because', 've', 'me', 'out', 'each', 'then', 'some',
'doing', 'weren', 'was', 'yours', 'herself', 'll', "you're", 'she', 'if', 'didn', 'i', 'were', 'bo
th', 'do', 'hers', 'most', 'her'}
```

In [14]:

```
word_list=list()
for i in range(len(data)):
    lip = data.Text[i].split()
    for j in lip:
        word_list.append(j)
```

In [15]:

```
from collections import Counter
wordCounter=Counter(word_list)
countedWordDict=dict(wordCounter)
sortedWordDict=sorted(countedWordDict.items(),key=lambda x : x[1],reverse = True)
sortedWordDict[0:20]
```

Out[15]:

```
[('the', 1796),
 ('to', 1668),
 ('a', 1280),
 ('on', 1032),
 ('of', 944),
 ('in', 891),
 ('AAP', 884),
 ('for', 868),
 ('and', 850),
 ('is', 811),
 ('-', 728),
 ('at', 541),
 ('this', 461),
 ('it', 454),
 ('I', 453),
 ('up', 357),
 ('user:', 340),
 ('from', 331),
 ('will', 330),
 ('be', 324)]
```

In [18]:

```
####### making wordcloud ###########

from wordcloud import WordCloud
```

```
wordlist2= " ".join(word_list)
stop_word_cloud=set(stopwords.words("english"))
wordcloud=WordCloud(stopwords=stop_word_cloud,max_words=2000,background_color="white",min_font_size
=3).generate_from_frequencies(countedWordDict)
plt.figure(figsize=(12,10))
plt.axis("off")
plt.imshow(wordcloud)
plt.show()
```

In [20]:

```
######## data processing #########
####### replacing the negative with 0 ##########

data["Sentiment"]=data["Sentiment"].replace(-1,0)
```

In [21]:

```
data["Sentiment"].value_counts()
```

Out[21]:

```
1    3685
0    2106
Name: Sentiment, dtype: int64
```

In [29]:

```
####### NLP processing ##########
from nltk.stem.porter import PorterStemmer
from nltk import word_tokenize,WordNetLemmatizer
ps = PorterStemmer()
lemma = WordNetLemmatizer()
stopwordSet = set(stopwords.words("english"))
```

In [31]:

```
############ cleaning text ###############
import re
text_reviews = list()
for i in range(len(data)):
    text=re.sub('[^a-zA-Z]'," ",data['Text'][i])
    text=text.lower()
    text= word_tokenize(text,language="english")
    text= [lemma.lemmatize(word) for word in text if(word) not in stopwordSet]
    text=" ".join(text)
    text_reviews.append(text)
```

In [34]:

```
######### creating bag of words model ###########
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
cv = CountVectorizer(max_features=1500)
X= cv.fit_transform(text_reviews).toarray()
y= data["Sentiment"]

###### splitting the data #########

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0)
```

In [36]:

```
############ modelling and predicting ############
########## using Logistic Regression Model ##########
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
logreg = LogisticRegression()
logreg.fit(X_train,y_train)
y_pred=logreg.predict(X_test)
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.72      0.64      0.68       417
           1       0.81      0.86      0.84       742

    accuracy                           0.78      1159
   macro avg       0.77      0.75      0.76      1159
weighted avg       0.78      0.78      0.78      1159
```

In [37]:

```
print(confusion_matrix(y_test,y_pred))
```

```
[[267 150]
 [102 640]]
```

In [38]:

```
######### NaiveBayes Multinomial ##########
from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB()
clf.fit(X_train,y_train)
y_pred= clf.predict(X_test)
print(classification_report(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.71      0.70      0.70       417
           1       0.83      0.84      0.83       742

    accuracy                           0.79      1159
   macro avg       0.77      0.77      0.77      1159
weighted avg       0.79      0.79      0.79      1159

[[291 126]
 [120 622]]
```

In [40]:

```
######## apllying randomForestclassifier #########

from sklearn.ensemble import RandomForestClassifier
random_forest=RandomForestClassifier()
random_forest.fit(X_train,y_train)
y_pred=random_forest.predict(X_test)
print(classification_report(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
C:\Users\Dipsikha\anaconda3\lib\importlib\_bootstrap.py:219: RuntimeWarning: numpy.ufunc size
changed, may indicate binary incompatibility. Expected 192 from C header, got 216 from PyObject
  return f(*args, **kwds)
C:\Users\Dipsikha\anaconda3\lib\importlib\_bootstrap.py:219: RuntimeWarning: numpy.ufunc size
changed, may indicate binary incompatibility. Expected 192 from C header, got 216 from PyObject
  return f(*args, **kwds)
```

```
              precision    recall  f1-score   support

           0       0.69      0.73      0.71       417
           1       0.84      0.82      0.83       742

    accuracy                           0.78      1159
   macro avg       0.77      0.77      0.77      1159
weighted avg       0.79      0.78      0.78      1159

[[303 114]
 [137 605]]
```

In [43]:

```
import xgboost as xgb
XGB = xgb.XGBClassifier(random_state = 1)
XGB.fit(X_train,y_train)
y_pred = XGB.predict(X_test)
acc_XGB = round(XGB.score(X_train,y_train)*100,2)
acc_XGB
print(classification_report(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.75      0.52      0.62       417
           1       0.77      0.90      0.83       742

    accuracy                           0.77      1159
   macro avg       0.76      0.71      0.73      1159
weighted avg       0.77      0.77      0.76      1159

[[218 199]
 [ 71 671]]
```

In [44]:

```
# AdaBoost
from sklearn.ensemble import AdaBoostClassifier
Ada = AdaBoostClassifier(random_state=1)
Ada.fit(X_train, y_train)
y_pred = Ada.predict(X_test)
acc_add = round(Ada.score(X_train, y_train) * 100, 2)
acc_add
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.68      0.53      0.59       417
           1       0.76      0.86      0.81       742

    accuracy                           0.74      1159
   macro avg       0.72      0.69      0.70      1159
weighted avg       0.73      0.74      0.73      1159

[[220 197]
 [103 639]]
```

In [47]:

```
# Support Vector Machines
from sklearn.svm import SVC, LinearSVC
```

```
svc = SVC()
svc.fit(X_train, y_train)
y_pred = svc.predict(X_test)
acc_svc = round(svc.score(X_train, y_train) * 100, 2)
acc_svc
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.77      0.59      0.67       417
           1       0.80      0.90      0.85       742

    accuracy                           0.79      1159
   macro avg       0.78      0.75      0.76      1159
weighted avg       0.79      0.79      0.78      1159

[[247 170]
 [ 73 669]]
```

In [ ]:

```
######### The best accuracy we are getting from support vector machine, Naive Bayes algorithm ####
##########
```