```
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.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix, ConfusionMatrixDisplay

df = pd.read_csv('sentiment_analysis.csv')

df.head()
```

```
   Year  Month  Day Time of Tweet  \
0  2018      8   18       morning
1  2018      8   18          noon
2  2017      8   18         night
3  2022      6    8       morning
4  2022      6    8          noon

                                                text sentiment
Platform
0              What a great day!!! Looks like dream.  positive
Twitter
1     I feel sorry, I miss you here in the sea beach  positive
Facebook
2                                      Don't angry me  negative
Facebook
3  We attend in the class just for listening teac...  negative
Facebook
4                   Those who want to go, let them go  negative
Instagram
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 499 entries, 0 to 498
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
```

```
 0   Year           499 non-null    int64
 1   Month          499 non-null    int64
 2   Day            499 non-null    int64
 3   Time of Tweet  499 non-null    object
 4   text           499 non-null    object
 5   sentiment      499 non-null    object
 6   Platform       499 non-null    object
dtypes: int64(3), object(4)
memory usage: 27.4+ KB
```

```
df.isnull().sum()
```

```
Year             0
Month            0
Day              0
Time of Tweet    0
text             0
sentiment        0
Platform         0
dtype: int64
```

```
df.columns
```

```
Index(['Year', 'Month', 'Day', 'Time of Tweet', 'text', 'sentiment',
       'Platform'],
      dtype='object')
```

```
text_df = df.drop(['Year', 'Month', 'Day',], axis=1)
text_df.head()
```

```
  Time of Tweet                                               text
sentiment  \
0       morning              What a great day!!! Looks like dream.
positive
1          noon     I feel sorry, I miss you here in the sea beach
positive
2         night                                        Don't angry me
negative
3       morning  We attend in the class just for listening teac...
negative
4          noon                   Those who want to go, let them go
negative

      Platform
0    Twitter
1    Facebook
2     Facebook
3    Facebook
4   Instagram
```

```
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")
```

```
What a great day!!! Looks like dream.

I feel sorry, I miss you here in the sea beach

Don't angry me

We attend in the class just for listening teachers reading on slide.
Just Nonsence
```

```
text_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 499 entries, 0 to 498
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Time of Tweet  499 non-null    object
 1   text           499 non-null    object
 2   sentiment      499 non-null    object
 3   Platform       499 non-null    object
dtypes: object(4)
memory usage: 15.7+ KB
```

```
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)

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

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))

text_df.head()
```

```
  Time of Tweet                                                  text
sentiment  \
0       morning                          great day looks like dream
positive
1          noon                           feel sorry miss sea beach
positive
2         night                                          dont angry
negative
3       morning  attend class listening teachers reading slide ...
negative
4          noon                                      want go let go
negative

      Platform
0    Twitter
1    Facebook
2     Facebook
3    Facebook
4   Instagram
```

```
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")
```

```
great day looks like dream

feel sorry miss sea beach

dont angry

attend class listening teachers reading slide nonsence

want go let go
```

```
text_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 391 entries, 0 to 498
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Time of Tweet  391 non-null    object
 1   text           391 non-null    object
 2   sentiment      391 non-null    object
 3   Platform       391 non-null    object
dtypes: object(4)
memory usage: 15.3+ KB
```

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

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

text_df.head(10)
```

```
  Time of Tweet                                                  text
sentiment  \
0       morning                          great day looks like dream
positive
1          noon                           feel sorry miss sea beach
positive
2         night                                          dont angry
negative
3       morning  attend class listening teachers reading slide ...
negative
4          noon                                       want go let go
negative
5         night                             night 2 feeling neutral
neutral
6       morning                   2 feedings baby fun smiles coos
positive
7          noon                                          soooo high
neutral
8         night
neutral
9       morning       today first time arrive boat amazing journey
positive

      Platform  polarity
0    Twitter       0.800
1    Facebook     -0.500
2     Facebook    -0.500
3    Facebook      0.000
4   Instagram      0.000
5     Facebook     0.000
6    Facebook      0.300
7   Instagram      0.160
8    Twitter       0.000
9    Facebook      0.425
```

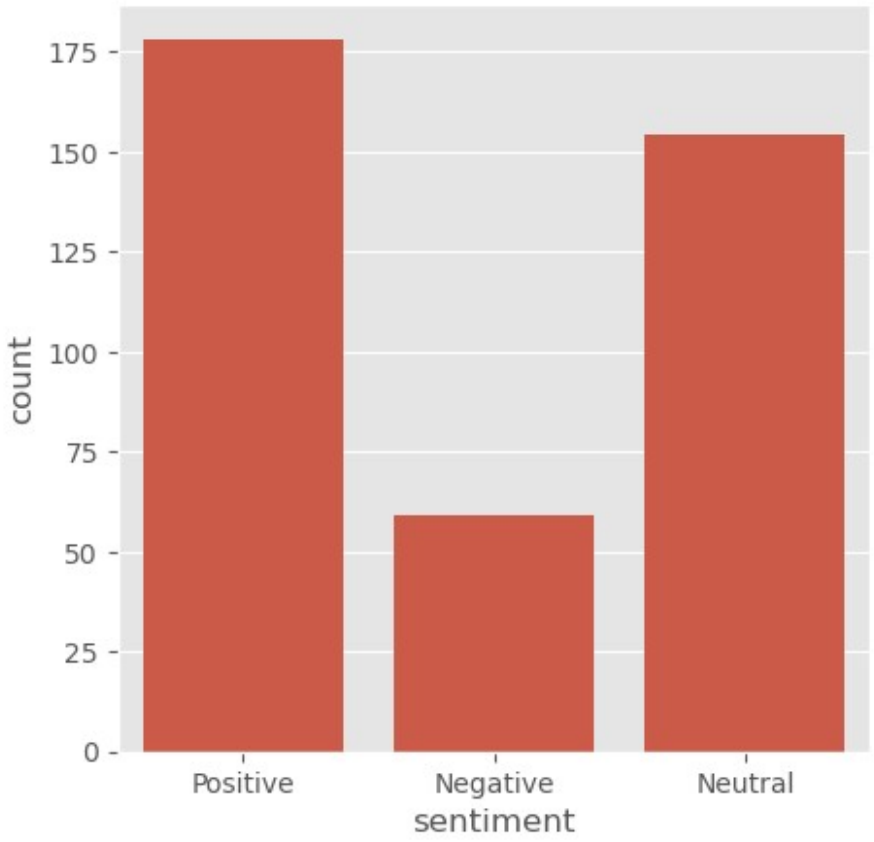
```
def sentiment(label):
    if label <0:
        return "Negative"
    elif label ==0:
        return "Neutral"
    elif label>0:
        return "Positive"
```

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

text_df.head()
```

```
  Time of Tweet                                                  text
sentiment  \
0       morning                          great day looks like dream
Positive
1          noon                           feel sorry miss sea beach
Negative
2         night                                          dont angry
Negative
3       morning  attend class listening teachers reading slide ...
Neutral
4          noon                                       want go let go
Neutral

      Platform  polarity
0    Twitter          0.8
1    Facebook        -0.5
2     Facebook       -0.5
3    Facebook         0.0
4   Instagram         0.0
```

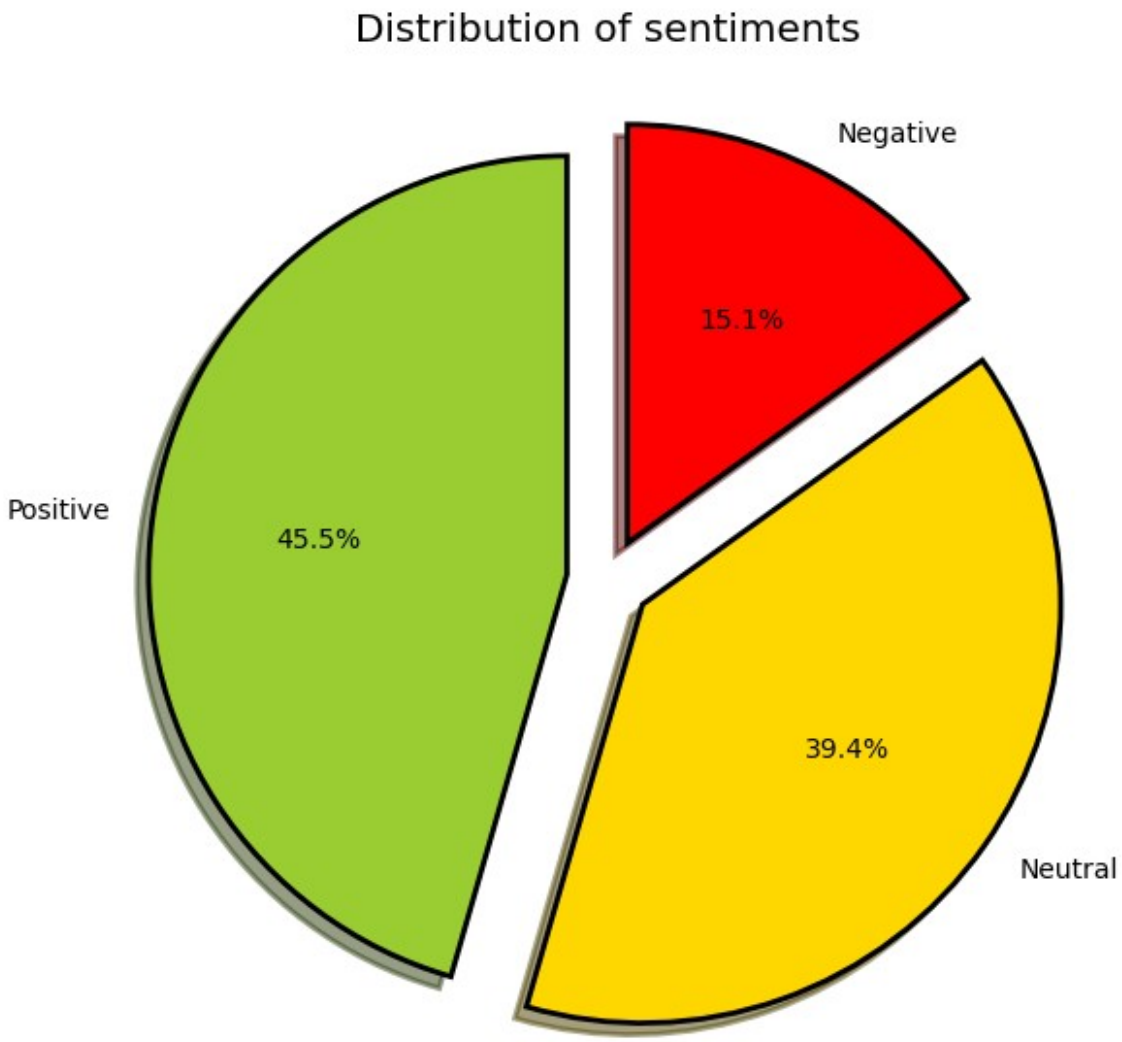
```
fig = plt.figure(figsize=(5,5))
sns.countplot(x='sentiment', data = text_df)
```

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




```
fig = plt.figure(figsize=(7,7))
colors = ("yellowgreen", "gold", "red")
wp = {'linewidth':2, 'edgecolor':"black"}
tags = text_df['sentiment'].value_counts()
explode = (0.1,0.1,0.1)
tags.plot(kind='pie', autopct='%1.1f%%', shadow=True, colors = colors,
          startangle=90, wedgeprops = wp, explode = explode, label='')
plt.title('Distribution of sentiments')
```

```
Text(0.5, 1.0, 'Distribution of sentiments')
```



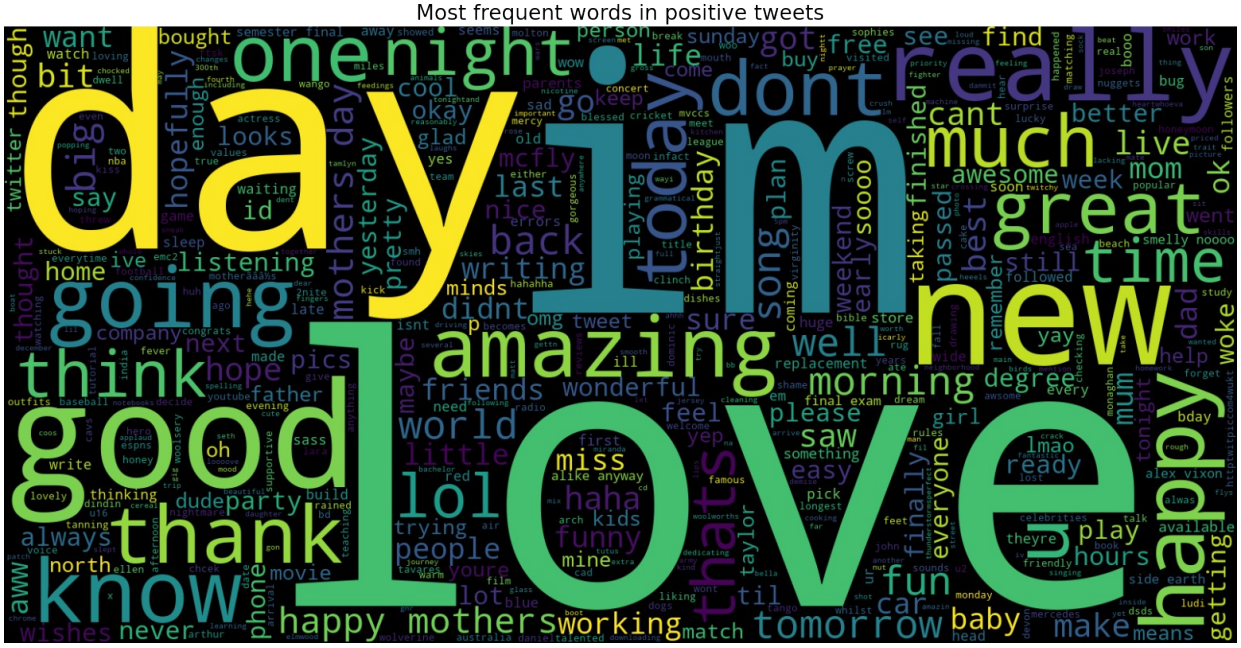

```
pos_tweets = text_df[text_df.sentiment == 'Positive']
pos_tweets = pos_tweets.sort_values(['polarity'], ascending= False)
pos_tweets.head()
```

```
    Time of Tweet
text  \
481          noon  kiss feet people kick anything want morning ev...

434         night      httptwitpiccom4wukt bought ludi rug dogs best

436          noon  loves mum much happy mothers day wonderful mot...

0         morning                           great day looks like dream

31           noon              buy sophies world book im really happy
```

```
    sentiment     Platform  polarity
481  Positive   Instagram        1.0
434  Positive    Facebook        1.0
436  Positive    Twitter         0.9
0    Positive    Twitter         0.8
31   Positive    Twitter         0.8
```

```
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




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

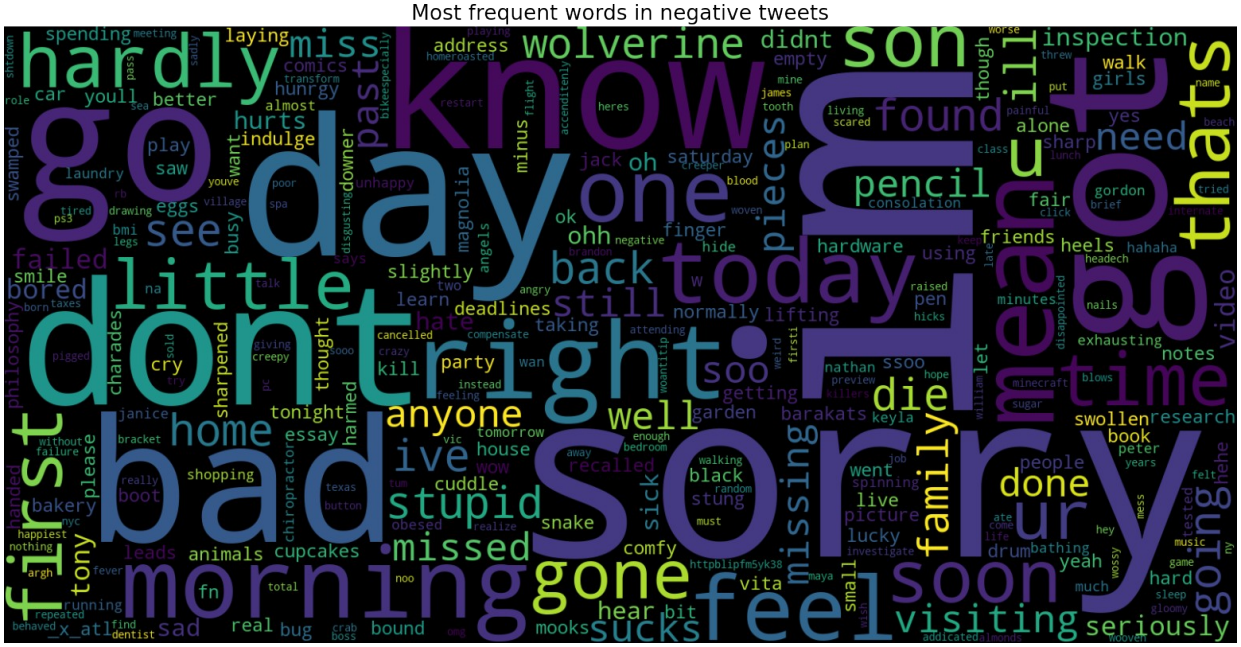
```
    Time of Tweet
text  \
426       morning              im hunrgy right heels kill hardly walk

307          noon  live thats want better bound bad eggs though s...

477       morning  son got stung bug first time little finger sli...

412          noon  _x_atl u mean jack barakats wow u ever gone ho...
```

```
359          night  spending saturday morning taking notes researc...
```

```
    sentiment     Platform  polarity
426  Negative   Instagram  -0.002976
307  Negative     Twitter  -0.021212
477  Negative     Twitter  -0.034722
412  Negative   Instagram  -0.047917
359  Negative   Instagram  -0.050000
```

```
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



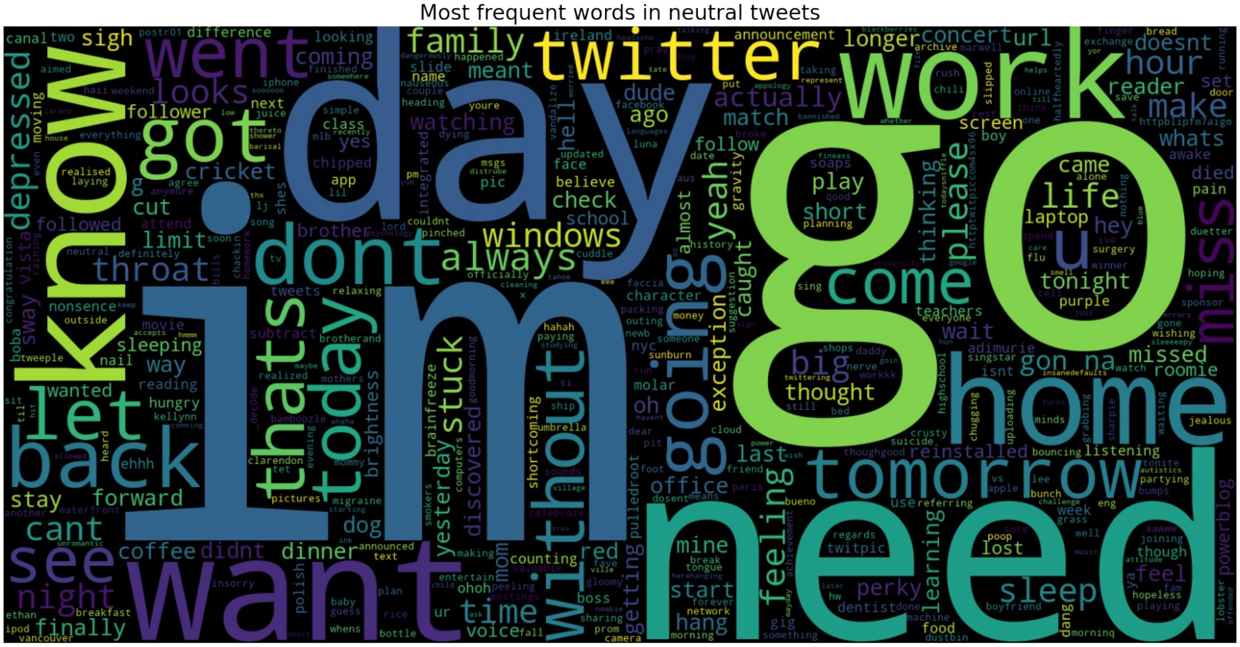

```
neutral_tweets = text_df[text_df.sentiment == 'Neutral']
neutral_tweets = neutral_tweets.sort_values(['polarity'], ascending=
False)
neutral_tweets.head()
```

```
    Time of Tweet
text  \
3          morning  attend class listening teachers reading slide ...

374          night                            cant wait see boy tomorrow
```

```
355          noon  ohoh missed ur tweets im gon na stay awake nig...

357       morning                              boss shes moving nyc

361          noon                 looking forward gig ireland see ya


    sentiment    Platform  polarity
3     Neutral    Facebook       0.0
374   Neutral    Twitter        0.0
355   Neutral    Twitter        0.0
357   Neutral    Facebook       0.0
361   Neutral    Twitter        0.0
```

```python
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




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

feature_names = vect.get_feature_names_out()  # Use
get_feature_names_out() for scikit-learn >= 1.0
```

```
print("Number of features: {}\n".format(len(feature_names)))
print("Feature names (first 20):\n{}".format(feature_names[:20]))
```

```
Number of features: 3314

Feature names (first 20):
['10' '10 minutes' '10 years' '100' '100 brightness' '1130' '1130
wait'
 '140' '140 character' '15' '15 mins' '1750' '1750 go' '19' '19 days'
'21'
 '21 ethan' '21 flys' '25' '25 december']
```

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

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

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: (312, 3314)
Size of y_train: (312,)
Size of x_test: (79, 3314)
Size of y_test: (79,)
```

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

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: 58.23%
```

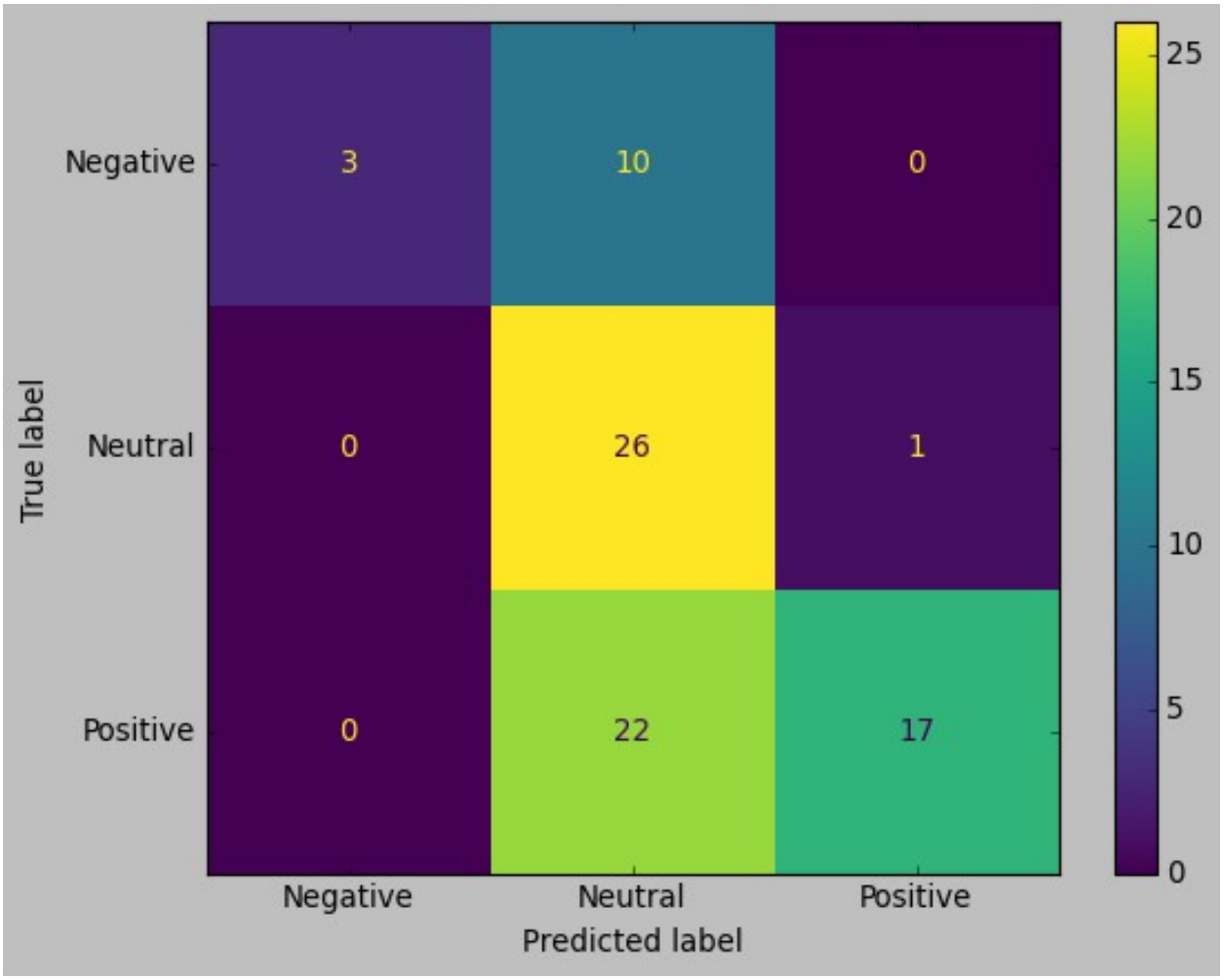
```
print(confusion_matrix(y_test, logreg_pred))
print("\n")
print(classification_report(y_test, logreg_pred))
```

```
[[ 3 10  0]
 [ 0 26  1]
 [ 0 22 17]]


              precision    recall  f1-score   support

    Negative       1.00      0.23      0.38        13
```

```
     Neutral       0.45      0.96      0.61         27
    Positive       0.94      0.44      0.60         39

    accuracy                           0.58         79
   macro avg       0.80      0.54      0.53         79
weighted avg       0.78      0.58      0.57         79
```

```
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
0x2b86a9ebb00>
```




```
from sklearn.model_selection import GridSearchCV
```

```
param_grid={'C':[0.001, 0.01, 0.1, 1, 10]}
grid = GridSearchCV(LogisticRegression(), param_grid)
grid.fit(x_train, y_train)
```

```
GridSearchCV(estimator=LogisticRegression(),
             param_grid={'C': [0.001, 0.01, 0.1, 1, 10]})
```

```
print("Best parameters:", grid.best_params_)
```

```
Best parameters: {'C': 0.1}
```

```
y_pred = grid.predict(x_test)
```

```
logreg_acc = accuracy_score(y_pred, y_test)
print("Test accuracy: {:.2f}%".format(logreg_acc*100))
```

```
Test accuracy: 60.76%
```

```
print(confusion_matrix(y_test, y_pred))
print("\n")
print(classification_report(y_test, y_pred))
```

```
[[ 2 11  0]
 [ 0 26  1]
 [ 0 19 20]]


              precision    recall  f1-score   support

    Negative       1.00      0.15      0.27        13
     Neutral       0.46      0.96      0.63        27
    Positive       0.95      0.51      0.67        39

    accuracy                           0.61        79
   macro avg       0.81      0.54      0.52        79
weighted avg       0.79      0.61      0.59        79
```

```
from sklearn.svm import LinearSVC
```

```
SVCmodel = LinearSVC()
SVCmodel.fit(x_train, y_train)
```

```
LinearSVC()
```

```
svc_pred = SVCmodel.predict(x_test)
svc_acc = accuracy_score(svc_pred, y_test)
print("test accuracy: {:.2f}%".format(svc_acc*100))
```

```
test accuracy: 60.76%
```

```
print(confusion_matrix(y_test, svc_pred))
print("\n")
print(classification_report(y_test, svc_pred))
```

```
[[ 3 10  0]
 [ 0 26  1]
 [ 0 20 19]]


              precision    recall  f1-score   support

    Negative       1.00      0.23      0.38        13
     Neutral       0.46      0.96      0.63        27
    Positive       0.95      0.49      0.64        39

    accuracy                           0.61        79
   macro avg       0.80      0.56      0.55        79
weighted avg       0.79      0.61      0.59        79
```

```
grid = {
    'C':[0.01, 0.1, 1, 10],
    'kernel':["linear","poly","rbf","sigmoid"],
    'degree':[1,3,5,7],
    'gamma':[0.01,1]
}
grid = GridSearchCV(SVCmodel, param_grid)
grid.fit(x_train, y_train)
```

```
GridSearchCV(estimator=LinearSVC(), param_grid={'C': [0.001, 0.01,
0.1, 1, 10]})
```

```
print("Best parameter:", grid.best_params_)
```

```
Best parameter: {'C': 1}
```

```
y_pred = grid.predict(x_test)
```

```
logreg_acc = accuracy_score(y_pred, y_test)
print("Test accuracy: {:.2f}%".format(logreg_acc*100))
```

```
Test accuracy: 60.76%
```

```
print(confusion_matrix(y_test, y_pred))
print("\n")
print(classification_report(y_test, y_pred))
```

```
[[ 3 10  0]
 [ 0 26  1]
 [ 0 20 19]]
```

```
              precision    recall  f1-score   support

    Negative       1.00      0.23      0.38        13
     Neutral       0.46      0.96      0.63        27
    Positive       0.95      0.49      0.64        39

    accuracy                           0.61        79
   macro avg       0.80      0.56      0.55        79
weighted avg       0.79      0.61      0.59        79
```

Additional code to extract data form twitter using twitter api

```
import tweepy #to access the twitter api
import pandas as pd #for basic data operations

# Importing the keys from twitter api
consumerKey = "xxxxxxxxxxxxxxxxxxxx"
consumerSecret = "xxxxxxxxxxxxxxxxxxxx"
accessToken = "xxxxxxxxxxxxxxxxxxxx"
accessTokenSecret = "xxxxxxxxxxxxxxxxxxxx"

# Establish the connection with twitter API
auth = tweepy.OAuthHandler(consumerKey, consumerSecret)
auth.set_access_token(accessToken, accessTokenSecret)
api = tweepy.API(auth)

# Get no of tweets and searched term together
tweets = tweepy.Cursor(api.search, q=searchTerm).items(NoOfTerms)
```