Twitter Sentiment Analysis

Twitter serves as an online social media platform where individuals express their thoughts through tweets. The issue of misuse, particularly for hateful content, has come to light. Twitter aims to address this concern by seeking assistance in developing a robust NLP-based classifier model. The objective is to differentiate negative tweets and prevent their dissemination.The dataset comprises tweet text and corresponding sentiment labels. The training set includes a specific word or phrase (selected_text) extracted from the tweet, representing the provided sentiment.

The goal is to predict the word or phrase within the tweet that best embodies the provided sentiment, encompassing all characters within that span, including commas and spaces.

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import nltk
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
import string
from tqdm import tqdm
from multiprocessing import Pool
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer

from sklearn.svm import SVC
from sklearn.naive_bayes import BernoulliNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, f1_score, \
roc_auc_score, roc_curve, precision_score, recall_score

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

In [2]:
```
# Loading the dataset
df = pd.read_csv('/Users/deepshikha /Downloads/Tweets.csv')
#Let's check the samples of data
df.head()
```

Out[2]:

	textID	text	selected_text	sentiment
0	cb774db0d1	I`d have responded, if I were going	I`d have responded, if I were going	neutral
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	Sooo SAD	negative
2	088c60f138	my boss is bullying me...	bullying me	negative
3	9642c003ef	what interview! leave me alone	leave me alone	negative
4	358bd9e861	Sons of ****, why couldn`t they put them on t...	Sons of ****,	negative

EDA

In [3]:
```
#Let's drop selected text & text id column
df.drop(['selected_text', 'textID'], axis=1, inplace=True)
target = 'sentiment'
df.reset_index(drop=True, inplace=True) #Resetting the index
original_df = df.copy(deep=True)
df.head()
```

Out[3]:

	text	sentiment
0	I`d have responded, if I were going	neutral
1	Sooo SAD I will miss you here in San Diego!!!	negative
2	my boss is bullying me...	negative
3	what interview! leave me alone	negative
4	Sons of ****, why couldn`t they put them on t...	negative

In [4]:
```
#Dimensions of the dataset & information about dataset
print('Dimensions of dataset:', df.shape)
#Checking the dtypes of all the columns
df.info()
```

```
Dimensions of dataset: (27481, 2)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27481 entries, 0 to 27480
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   text       27480 non-null  object
 1   sentiment  27481 non-null  object
dtypes: object(2)
memory usage: 429.5+ KB
```

In [5]:
```
#Descriptive summary of dataset
df.describe()
```

Out[5]:

	text	sentiment
count	27480	27481
unique	27480	3
top	I`d have responded, if I were going	neutral
freq	1	11118

In [6]:
```
#Let's check Null values
df.isnull().sum()
```

Out[6]:
```
text         1
sentiment    0
dtype: int64
```

The dataset has one null row, we can drop it

In [7]:
```
#Dropping the null values
df.dropna(inplace=True)
original_df = df.copy()
#Let's check Duplicates
df.duplicated().sum()
```

Out[7]: 0

In [8]:
```
# Let's get a word count
df['word_count'] = df['text'].apply(lambda x: len(str(x).split(" ")))
df[['text','word_count']].head()
```

Out[8]:

	text	word_count
0	I`d have responded, if I were going	8
1	Sooo SAD I will miss you here in San Diego!!!	11
2	my boss is bullying me...	5
3	what interview! leave me alone	6
4	Sons of ****, why couldn`t they put them on t...	15

In [9]:
```
#Number of Characters- including spaces
df['char_count'] = df['text'].str.len() # this also includes spaces
df[['text','char_count']].head()
```

Out[9]:

	text	char_count
0	I`d have responded, if I were going	36
1	Sooo SAD I will miss you here in San Diego!!!	46
2	my boss is bullying me...	25
3	what interview! leave me alone	31
4	Sons of ****, why couldn`t they put them on t...	75

In [10]:
```
#Average Word Length:
def      avg_word(sentence):
    words = sentence.split()
    return (sum(len(word) for word in words)/len(words))
df['avg_word'] = df['text'].apply(lambda x: avg_word(x))
df[['text','avg_word']].head()
```

Out[10]:

	text	avg_word
0	I`d have responded, if I were going	4.142857
1	Sooo SAD I will miss you here in San Diego!!!	3.600000
2	my boss is bullying me...	4.200000
3	what interview! leave me alone	5.200000
4	Sons of ****, why couldn`t they put them on t...	4.357143

In [11]:
```
#Number of stop Words:
nltk.download('stopwords')
from nltk.corpus import stopwords
stop = stopwords.words('english')

df['stopwords'] = df['text'].apply(lambda x: len([x for x in x.split() if x in stop]))
df[['text','stopwords']].head()
```

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

Out[11]:

	text	stopwords
0	I`d have responded, if I were going	3
1	Sooo SAD I will miss you here in San Diego!!!	4
2	my boss is bullying me...	2
3	what interview! leave me alone	2
4	Sons of ****, why couldn`t they put them on t...	7

In [12]:
```
#Number of special character:
df['hastags'] = df['text'].apply(lambda x: len([x for x in x.split() if x.startswith('@')]))
df[['text','hastags']].head()
```

Out[12]:

	text	hastags
0	I`d have responded, if I were going	0
1	Sooo SAD I will miss you here in San Diego!!!	0
2	my boss is bullying me...	0
3	what interview! leave me alone	0
4	Sons of ****, why couldn`t they put them on t...	0

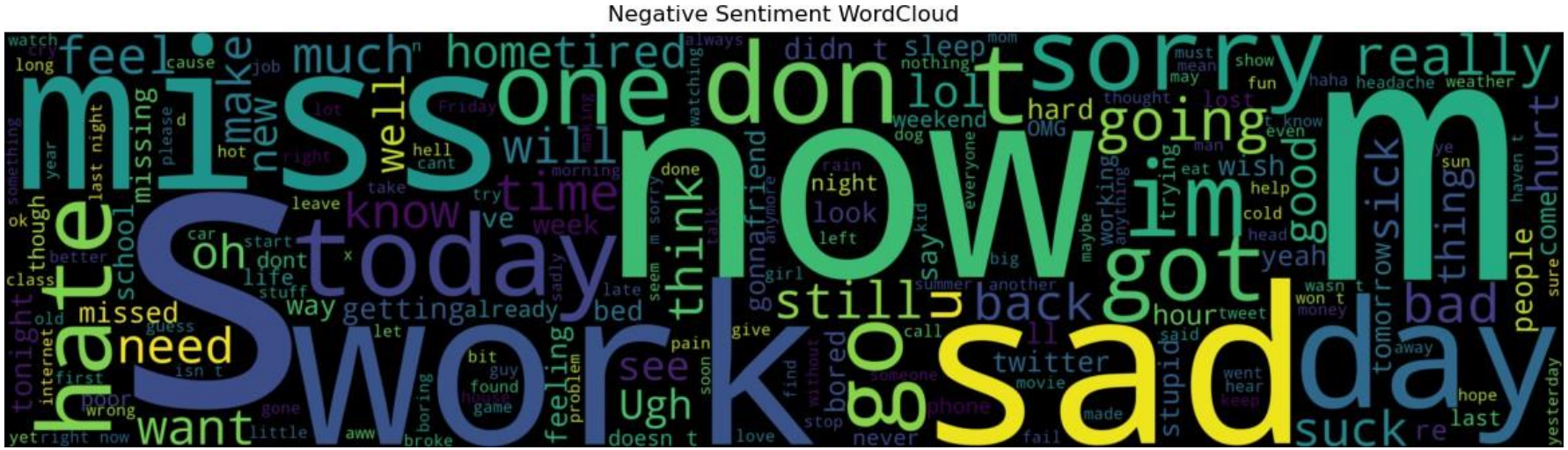
In [13]:
```
#Number of numerics:
df['numerics'] = df['text'].apply(lambda x: len([x for x in x.split() if x.isdigit()]))
df[['text','numerics']].head()
```

Out[13]:

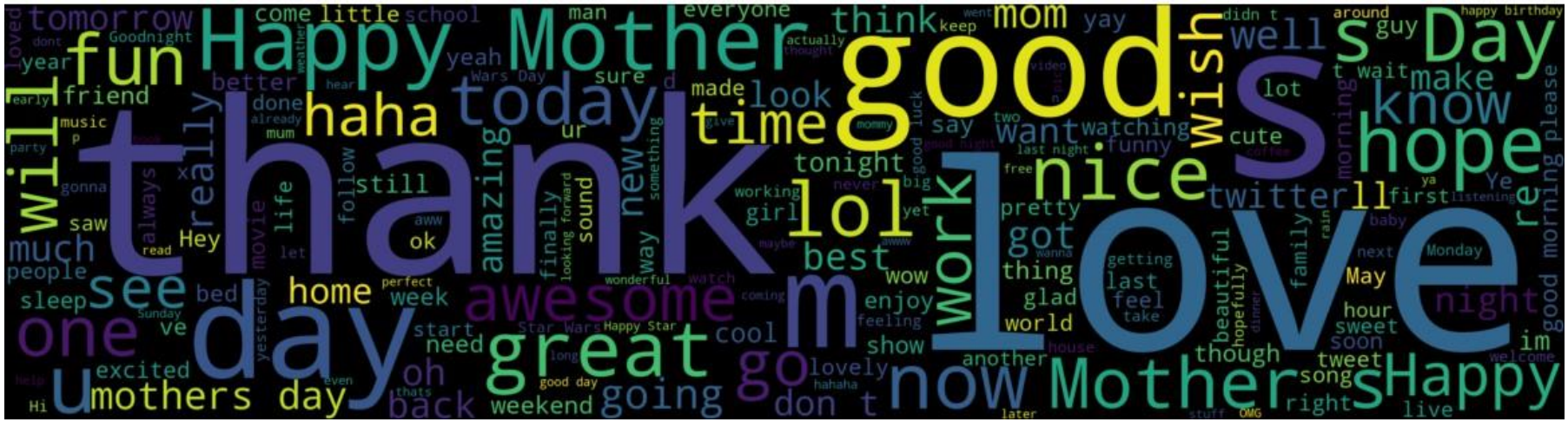
	text	numerics
0	I`d have responded, if I were going	0
1	Sooo SAD I will miss you here in San Diego!!!	0
2	my boss is bullying me...	0
3	what interview! leave me alone	0
4	Sons of ****, why couldn`t they put them on t...	0

Let's visualize the words

In [14]:
```
#!pip install wordcloud
from wordcloud import WordCloud, STOPWORDS
negative_df = df[df['sentiment'] == 'negative']
positive_df = df[df['sentiment'] == 'positive']
neutral_df = df[df['sentiment'] == 'neutral']

# Define a function to generate and display a WordCloud
def generate_wordcloud(data, title):
    words = ' '.join(data['text'])
    cleaned_word = " ".join([word for word in words.split()
                             if 'http' not in word
                                 and not word.startswith('@')
                                 and word != 'RT' ])
    wordcloud = WordCloud(stopwords=STOPWORDS,background_color='black',
                          width=3000, height=800).generate(cleaned_word)
    plt.figure(figsize=(15, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.title(title)
    plt.axis('off')
    plt.show()
# Generate and display WordClouds for each sentiment category
generate_wordcloud(negative_df, 'Negative Sentiment WordCloud')
generate_wordcloud(positive_df, 'Positive Sentiment WordCloud')
generate_wordcloud(neutral_df, 'Neutral Sentiment WordCloud')
```



Positive Sentiment WordCloud



Neutral Sentiment WordCloud

Pre-Processing

In [15]:
```
# Convert text to lowercase
df['text'] = df['text'].apply(lambda x: " ".join(x.lower() for x in x.split()))

# Removal of punctuations
df['text'].str.replace('[^\w\s]','')

#Removal of StopWords
from nltk.corpus import stopwords
stop = stopwords.words('english')
df['text'] = df['text'].apply(lambda x: " ".join(x for x in x.split() if x not in stop))
df['text'].head()
```

Out[15]:
```
0                                  i`d responded, going
1                           sooo sad miss san diego!!!
2                                boss bullying me...
3                             interview! leave alone
4    sons ****, couldn`t put releases already bought
Name: text, dtype: object
```

Common Words Removal

- Let's create a list of 10 frequently occurring words and then decide if we need to remove it or retain it

In [16]:
```
freq = pd.Series(' '.join(df['text']).split()).value_counts()[:30]
freq
```

Out[16]:
```
i`m       2173
day       1481
get       1415
good      1325
like      1303
it`s      1174
go        1162
-         1147
got       1069
going     1062
love      1060
happy      914
work       878
don`t      850
u          848
really     841
one        838
im         824
****       796
back       781
see        765
know       757
can`t      746
time       739
new        725
lol        697
want       695
&          675
still      661
think      656
dtype: int64
```

Let's remove "I'm", '-', '****', '& ' There can be other words too which can be removed, but let's conbtinue with above only

In [17]:
```
freq =["I'm", "-", "****", "&"]
df['text']= df['text'].apply(lambda x: " ".join(x for x in x.split() if x not in freq))
df['text'].head()
```

Out[17]:
```
0                                  i`d responded, going
1                            sooo sad miss san diego!!!
2                                  boss bullying me...
3                               interview! leave alone
4    sons ****, couldn`t put releases already bought
Name: text, dtype: object
```

In [18]:
```
#Rare Words Removal
#This is done as association of these less occurring words with the existing words could be a noise
freq = pd.Series(' '.join(df['text']).split()).value_counts()[-10:]
freq
```

Out[18]:
```
neaaarr                                  1
wer                                      1
sigh.....                                1
@_harrykim                               1
#design                                  1
http://tinyurl.com/dl2upx  (http://tinyurl.com/dl2upx)    1
resources                                1
pours.                                   1
cyalater!!!                              1
((hugs))                                 1
dtype: int64
```

In [19]:
```
#Stemming -refers to the removal of suffices, like "ing", "ly", "s", etc. by a simple rule-based approach¶
from nltk.stem import PorterStemmer
st = PorterStemmer()
df['text'][:5].apply(lambda x: " ".join([st.stem(word) for word in x.split()]))
```

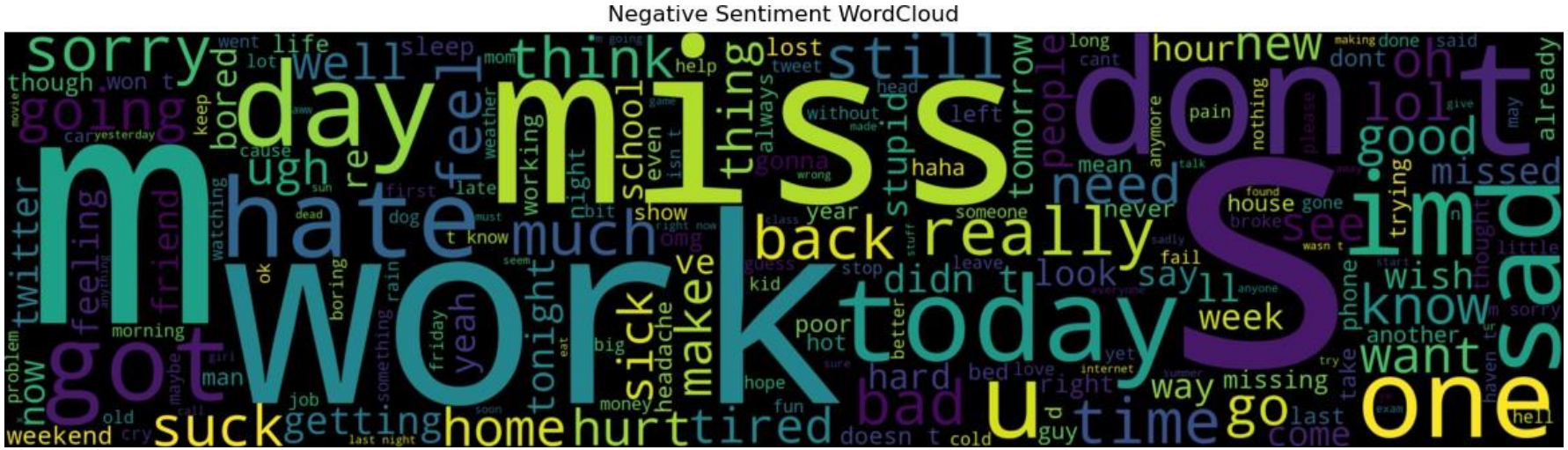
Out[19]:
```
0                                  i`d responded, go
1                            sooo sad miss san diego!!!
2                                  boss bulli me...
3                               interview! leav alon
4    son ****, couldn`t put releas alreadi bought
Name: text, dtype: object
```

In [20]:
```
df[target].value_counts()
```

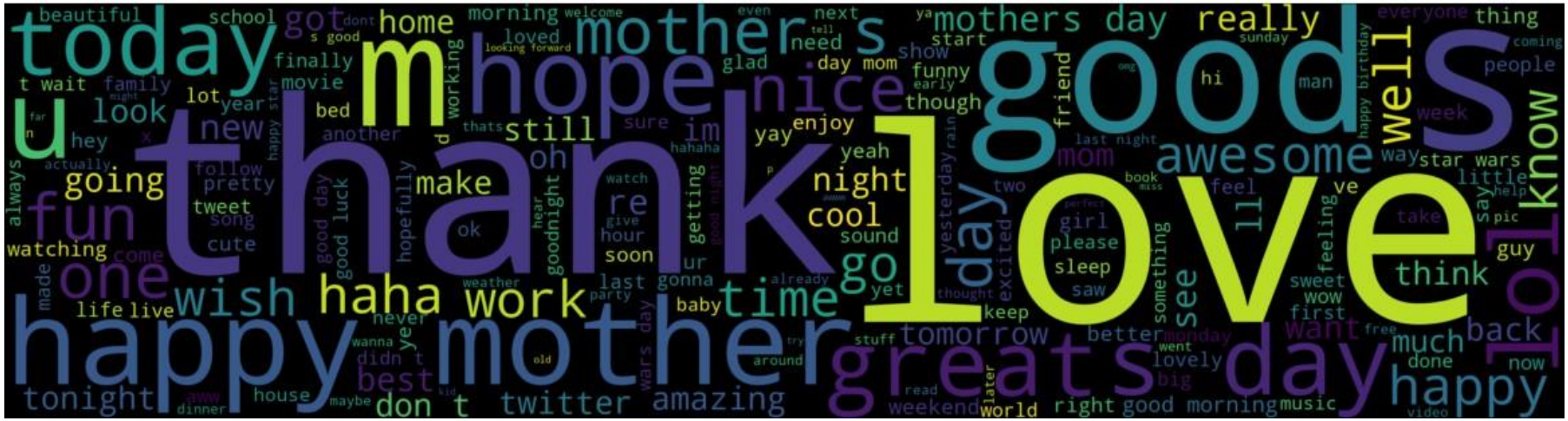
Out[20]:
```
neutral     11117
positive     8582
negative     7781
Name: sentiment, dtype: int64
```

Let's visualize wordcloud post cleaning

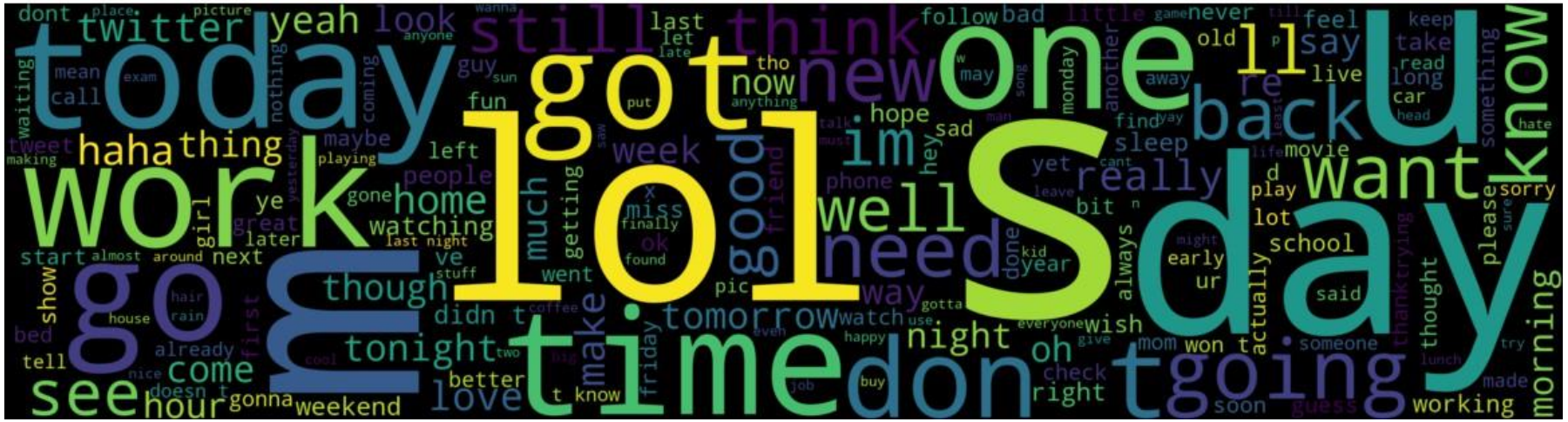
In [21]:
```
from wordcloud import WordCloud, STOPWORDS
negative_df = df[df['sentiment'] == 'negative']
positive_df = df[df['sentiment'] == 'positive']
neutral_df = df[df['sentiment'] == 'neutral']

# Define a function to generate and display a WordCloud
def generate_wordcloud(data, title):
    words = ' '.join(data['text'])
    cleaned_word = " ".join([word for word in words.split()
                             if 'http' not in word
                                 and not word.startswith('@')
                                 and word != 'RT' ])
    wordcloud = WordCloud(stopwords=STOPWORDS,background_color='black',
                          width=3000, height=800).generate(cleaned_word)
    plt.figure(figsize=(15, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.title(title)
    plt.axis('off')
    plt.show()
# Generate and display WordClouds for each sentiment category
generate_wordcloud(negative_df, 'Negative Sentiment WordCloud')
generate_wordcloud(positive_df, 'Positive Sentiment WordCloud')
generate_wordcloud(neutral_df, 'Neutral Sentiment WordCloud')
```



Positive Sentiment WordCloud



Neutral Sentiment WordCloud



Predictive Modeling

In [22]:
```
X = df['text']
y = df['sentiment']
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer ()
X = vectorizer.fit_transform(X).toarray()
vectorizer
```

Out[22]:
```
TfidfVectorizer
TfidfVectorizer()
```

In [23]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
#Let us create first create a table to store the results of various models
results_df = pd.DataFrame(np.zeros((2,5)), columns=['Accuracy', 'Precision','Recall','F1-score','AUC-ROC sco
results_df.index=['Logistic Regression (LR)','Naïve Bayes Classifier (NB)'] #'Decision Tree Classifier (DT)',
results_df
```

Out[23]:

	Accuracy	Precision	Recall	F1-score	AUC-ROC score
Logistic Regression (LR)	0.0	0.0	0.0	0.0	0.0
Naïve Bayes Classifier (NB)	0.0	0.0	0.0	0.0	0.0

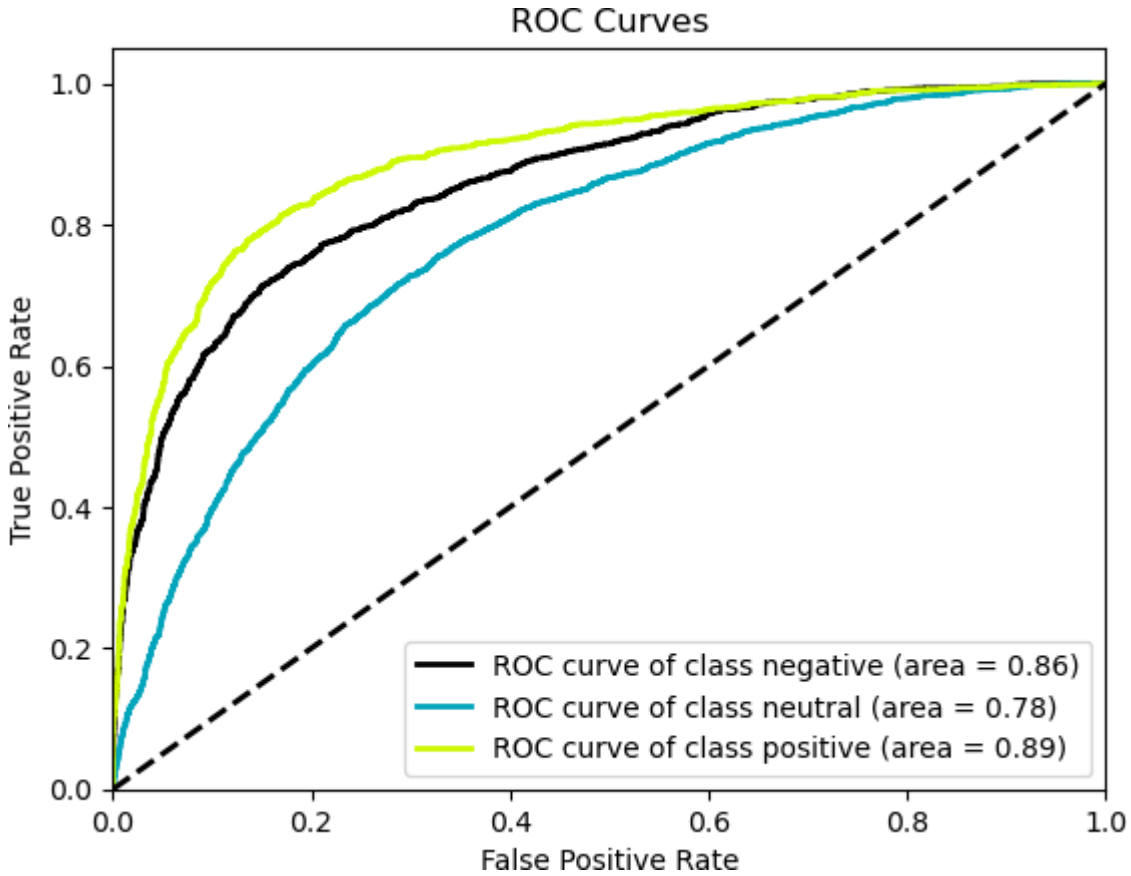
In [24]:
```
tall scikit-plot
efine functions to summarise the Prediction's scores .
itplot.metrics import plot_roc_curve as auc_roc
arn.metrics import accuracy_score, confusion_matrix, classification_report, f1_score, \
core, roc_curve, precision_score, recall_score

cation Summary Function
ification_Summary(pred,pred_prob,i):
ts_df.iloc[i]['Accuracy']=round(accuracy_score(y_test, pred),3)*100
ts_df.iloc[i]['Precision']=round(precision_score(y_test, pred, average='weighted'),3)*100 #, average='weighte
ts_df.iloc[i]['Recall']=round(recall_score(y_test, pred, average='weighted'),3)*100 #, average='weighted'
ts_df.iloc[i]['F1-score']=round(f1_score(y_test, pred, average='weighted'),3)*100 #, average='weighted'
ts_df.iloc[i]['AUC/ROC score']=round(roc_auc_score(y_test, pred_prob, multi_class='ovr'),3)*100 #, multi_clas
('{}{}\033[1m Evaluating {} \033[0m{}{}\n'.format('<'*3,'-'*35,results_df.index[i], '-'*35,'>'*3))
('Accuracy = {}%'.format(round(accuracy_score(y_test, pred),3)*100))
('F1 Score = {}%'.format(round(f1_score(y_test, pred, average='weighted'),3)*100)) #, average='weighted'
('\n \033[1mConfusiton Matrix:\033[0m\n',confusion_matrix(y_test, pred))
('\n\033[1mClassification Report:\033[0m\n',classification_report(y_test, pred))
oc(y_test, pred_prob, curves=['each_class'])
how()
```

In [25]:
```
#Visualising Function
def AUC_ROC_plot(y_test, pred):
    ref = [0 for _ in range(len(y_test))]
    ref_auc = roc_auc_score(y_test, ref)
    lr_auc = roc_auc_score(y_test, pred)

    ns_fpr, ns_tpr, _ = roc_curve(y_test, ref)
    lr_fpr, lr_tpr, _ = roc_curve(y_test, pred)

    plt.plot(ns_fpr, ns_tpr, linestyle='--')
    plt.plot(lr_fpr, lr_tpr, marker='.', label='AUC = {}'.format(round(roc_auc_score(y_test, pred)*100,2)))
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.legend()
    plt.show()
```

Logistics Regression Model

In [26]:
```
# Building Logistic Regression Classifier
from sklearn.linear_model import LogisticRegression

LR_model = LogisticRegression()
LR = LR_model.fit(X_train, y_train)
pred = LR.predict(X_test)
pred_prob = LR.predict_proba(X_test)
Classification_Summary(pred,pred_prob,0)
```

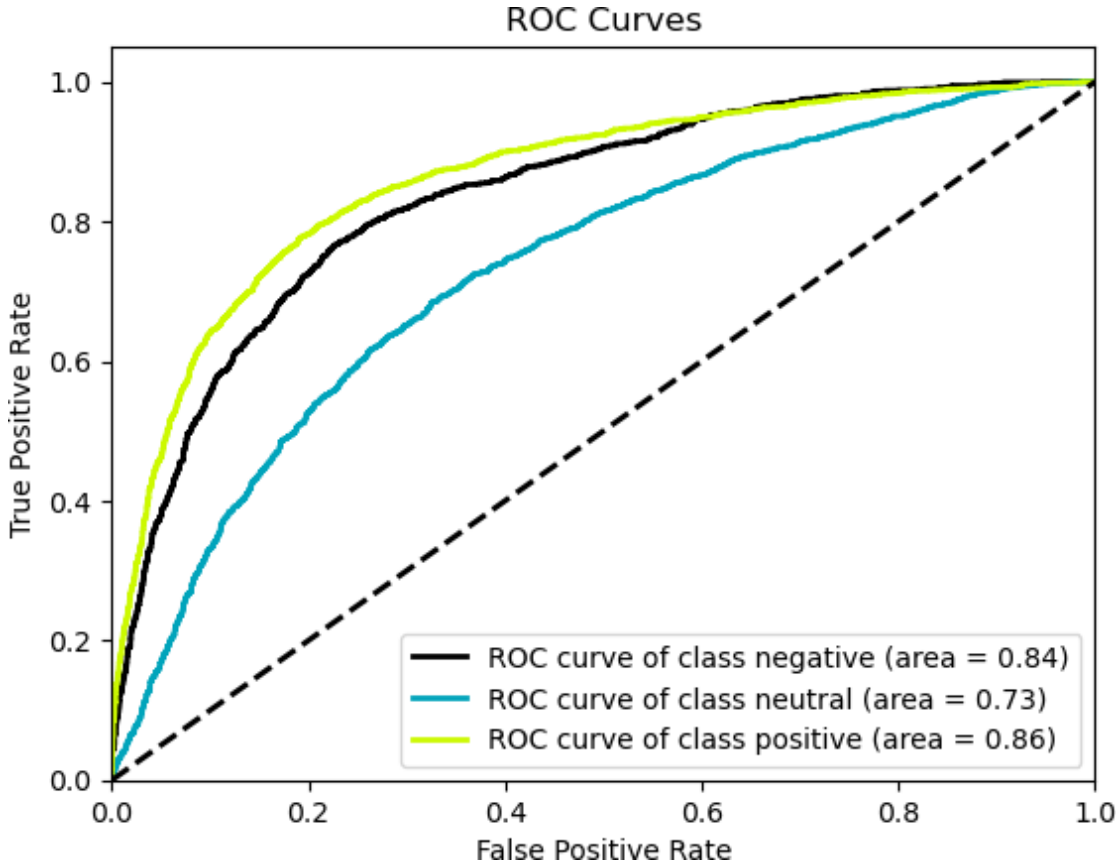
```
<<<---------------------------------- Evaluating Logistic Regression (LR) -------------------------------
--->>>

Accuracy = 68.5%
F1 Score = 68.4%

 Confusiton Matrix:
 [[ 848  589   86]
 [ 257 1775  243]
 [  51  505 1142]]

Classification Report:
              precision    recall  f1-score   support

    negative       0.73      0.56      0.63      1523
     neutral       0.62      0.78      0.69      2275
    positive       0.78      0.67      0.72      1698

    accuracy                           0.69      5496
   macro avg       0.71      0.67      0.68      5496
weighted avg       0.70      0.69      0.68      5496
```



Naive Bayes Classfier

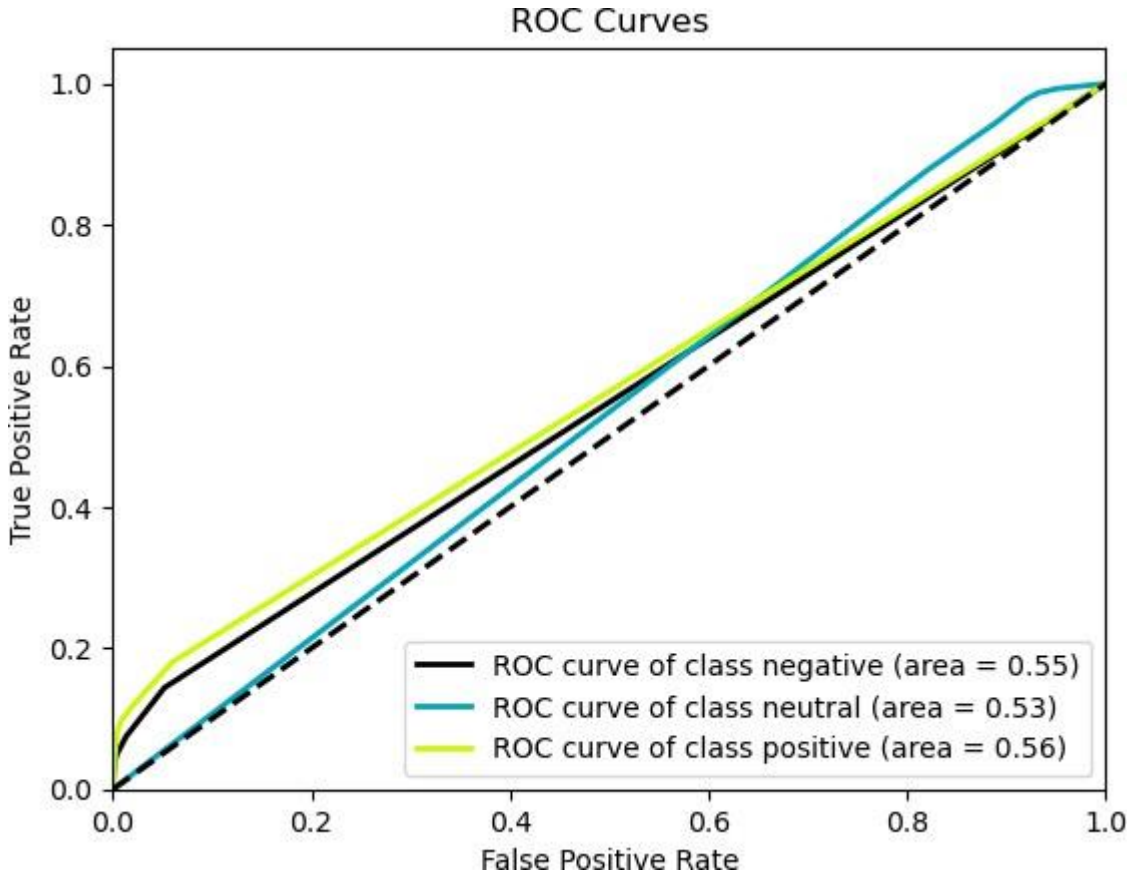
In [27]:
```
#Naive Bayes Classfier
NB_model = BernoulliNB()
NB = NB_model.fit(X_train, y_train)
pred = NB.predict(X_test)
pred_prob = NB.predict_proba(X_test)
Classification_Summary(pred,pred_prob,1)
```

```
<<<---------------------------------- Evaluating Naïve Bayes Classifier (NB) ----------------------------
--------->>>

Accuracy = 63.1%
F1 Score = 62.4%

 Confusiton Matrix:
[[ 648  795   80]
 [ 217 1794  264]
 [  54  620 1024]]

Classification Report:
              precision    recall  f1-score   support

    negative       0.71      0.43      0.53      1523
     neutral       0.56      0.79      0.65      2275
    positive       0.75      0.60      0.67      1698

    accuracy                           0.63      5496
   macro avg       0.67      0.61      0.62      5496
weighted avg       0.66      0.63      0.62      5496
```


ROC Curves
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve of class negative (area = 0.84)
ROC curve of class neutral (area = 0.73)
ROC curve of class positive (area = 0.86)

In [28]:
```
from sklearn.neighbors import KNeighborsClassifier

knn_model = KNeighborsClassifier()
knn = knn_model.fit(X_train, y_train)
pred = knn.predict(X_test)
pred_prob = knn.predict_proba(X_test)
Classification_Summary(pred, pred_prob, 0)
```

```
<<<---------------------------------- Evaluating Logistic Regression (LR) -------------------------------
--->>>

Accuracy = 45.1%
F1 Score = 32.9%

 Confusiton Matrix:
 [[  82 1436    5]
 [  20 2228   27]
 [   4 1528  166]]

Classification Report:
              precision    recall  f1-score   support

    negative       0.77      0.05      0.10      1523
     neutral       0.43      0.98      0.60      2275
    positive       0.84      0.10      0.18      1698

    accuracy                           0.45      5496
   macro avg       0.68      0.38      0.29      5496
weighted avg       0.65      0.45      0.33      5496
```



In [*]:
```
from sklearn.svm import SVC

svm_model = SVC(probability=True)
svm = svm_model.fit(X_train, y_train)
pred = svm.predict(X_test)
pred_prob = svm.predict_proba(X_test)
Classification_Summary(pred, pred_prob, 0)
```

In [*]:
```
from sklearn.ensemble import RandomForestClassifier

rf_model = RandomForestClassifier()
rf = rf_model.fit(X_train, y_train)
pred = rf.predict(X_test)
pred_prob = rf.predict_proba(X_test)
Classification_Summary(pred, pred_prob, 0)
```

In [*]:
```
from sklearn.ensemble import AdaBoostClassifier

ada_model = AdaBoostClassifier()
ada = ada_model.fit(X_train, y_train)
pred = ada.predict(X_test)
pred_prob = ada.predict_proba(X_test)
Classification_Summary(pred, pred_prob, 0)
```

In [*]:
```
from sklearn.ensemble import GradientBoostingClassifier

gb_model = GradientBoostingClassifier()
gb = gb_model.fit(X_train, y_train)
pred = gb.predict(X_test)
pred_prob = gb.predict_proba(X_test)
Classification_Summary(pred, pred_prob, 0)
```

In [*]:
```
from xgboost import XGBClassifier

xgb_model = XGBClassifier()
xgb = xgb_model.fit(X_train, y_train)
pred = xgb.predict(X_test)
pred_prob = xgb.predict_proba(X_test)
Classification_Summary(pred, pred_prob, 0)
```

In [*]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, Dense

# Define the model
rnn_model = Sequential()
rnn_model.add(SimpleRNN(units=50, activation='relu', input_shape=(X_train.shape[1], X_train.shape[2])))
rnn_model.add(Dense(1, activation='sigmoid'))

# Compile the model
rnn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Fit the model
rnn_model.fit(X_train, y_train, epochs=10, batch_size=32)

# Predictions
pred_prob = rnn_model.predict(X_test)
pred = (pred_prob > 0.5).astype(int)

Classification_Summary(pred, pred_prob, 0)
```

In [*]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

# Define the model
lstm_model = Sequential()
lstm_model.add(LSTM(units=50, activation='relu', input_shape=(X_train.shape[1], X_train.shape[2])))
lstm_model.add(Dense(1, activation='sigmoid'))

# Compile the model
lstm_model.compile(optimizer='adam',   loss='binary_crossentropy',   metrics=['accuracy'])

# Fit the model
lstm_model.fit(X_train, y_train, epochs=10, batch_size=32)

# Predictions
pred_prob = lstm_model.predict(X_test)
pred = (pred_prob > 0.5).astype(int)

Classification_Summary(pred, pred_prob, 0)
```

In [ ]:

In [ ]: