project

March 1, 2022

[1]:
```python
# Importing necessary modules.
import re
import string
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer, PorterStemmer
from nltk.probability import FreqDist
from nltk.tokenize import RegexpTokenizer

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from collections import Counter

import nltk
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('words')

import warnings
warnings.filterwarnings("ignore")
plt.rcParams["figure.figsize"] = (10,6)
pd.set_option('display.max_columns', 50)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\AI\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\AI\AppData\Roaming\nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\AI\AppData\Roaming\nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package words to
[nltk_data]     C:\Users\AI\AppData\Roaming\nltk_data…
[nltk_data]   Package words is already up-to-date!
```

0.1 Business Value

There are six different airline companies in this dataset and their customers still complaining about some problems with their services/flights. We are going to analyze and making machine learning project for how airline companies could improve ourselves with our findings.

0.2 Business Problem

In this project, main goal is the predict airline sentiment of flights with machine learning model. This will help airline companies for future work. Depend on customer's review(positive , neutral or negative) airline companies could take action about it.

[125]:
```python
# Import and looking the data.
df = pd.read_csv('Tweets.csv')
df.head()
```

[125]:
```
             tweet_id airline_sentiment  airline_sentiment_confidence  \
0  570306133677760513           neutral                        1.0000
1  570301130888122368          positive                        0.3486
2  570301083672813571           neutral                        0.6837
3  570301031407624196          negative                        1.0000
4  570300817074462722          negative                        1.0000

  negativereason  negativereason_confidence         airline  \
0            NaN                        NaN  Virgin America
1            NaN                     0.0000  Virgin America
2            NaN                        NaN  Virgin America
3     Bad Flight                     0.7033  Virgin America
4     Can't Tell                     1.0000  Virgin America

  airline_sentiment_gold        name negativereason_gold  retweet_count  \
0                    NaN     cairdin                 NaN              0
1                    NaN    jnardino                 NaN              0
2                    NaN  yvonnalynn                 NaN              0
3                    NaN    jnardino                 NaN              0
4                    NaN    jnardino                 NaN              0

                                                text tweet_coord  \
0                @VirginAmerica What @dhepburn said.         NaN
1  @VirginAmerica plus you've added commercials t…         NaN
2  @VirginAmerica I didn't today… Must mean I n…         NaN
3  @VirginAmerica it's really aggressive to blast…         NaN
4  @VirginAmerica and it's a really big bad thing…         NaN

               tweet_created tweet_location               user_timezone
0  2015-02-24 11:35:52 -0800            NaN  Eastern Time (US & Canada)
1  2015-02-24 11:15:59 -0800            NaN  Pacific Time (US & Canada)
2  2015-02-24 11:15:48 -0800      Lets Play  Central Time (US & Canada)
```

```
3  2015-02-24 11:15:36 -0800            NaN  Pacific Time (US & Canada)
4  2015-02-24 11:14:45 -0800            NaN  Pacific Time (US & Canada)
```

[203]:
```
df['tweet_created']
```

[203]:
```
0        2015-02-24 11:35:52 -0800
1        2015-02-24 11:15:59 -0800
2        2015-02-24 11:15:48 -0800
3        2015-02-24 11:15:36 -0800
4        2015-02-24 11:14:45 -0800
                    …
14635    2015-02-22 12:01:01 -0800
14636    2015-02-22 11:59:46 -0800
14637    2015-02-22 11:59:15 -0800
14638    2015-02-22 11:59:02 -0800
14639    2015-02-22 11:58:51 -0800
Name: tweet_created, Length: 14640, dtype: object
```

[126]:
```
# Example of a tweet.
df['text'][10]
```

[126]:
```
'@VirginAmerica did you know that suicide is the second leading cause of death
among teens 10-24'
```

[127]:
```
# Target variable class balance.
df['airline_sentiment'].value_counts()
```

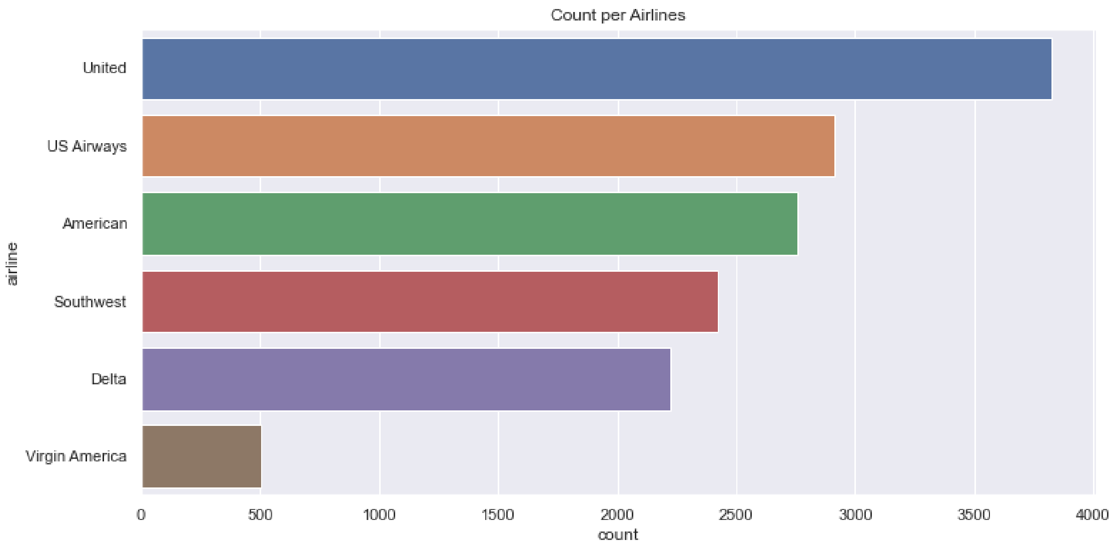
[127]:
```
negative    9178
neutral     3099
positive    2363
Name: airline_sentiment, dtype: int64
```

[128]:
```
# Airline companies balance.
df['airline'].value_counts()
```

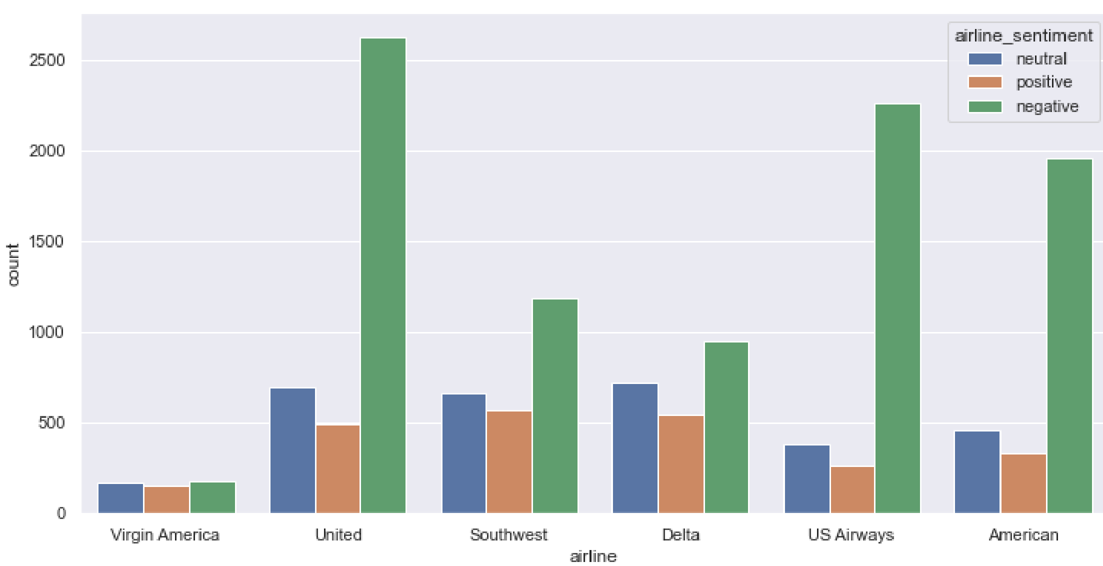
[128]:
```
United            3822
US Airways        2913
American          2759
Southwest         2420
Delta             2222
Virgin America     504
Name: airline, dtype: int64
```

0.3 Data Understanding

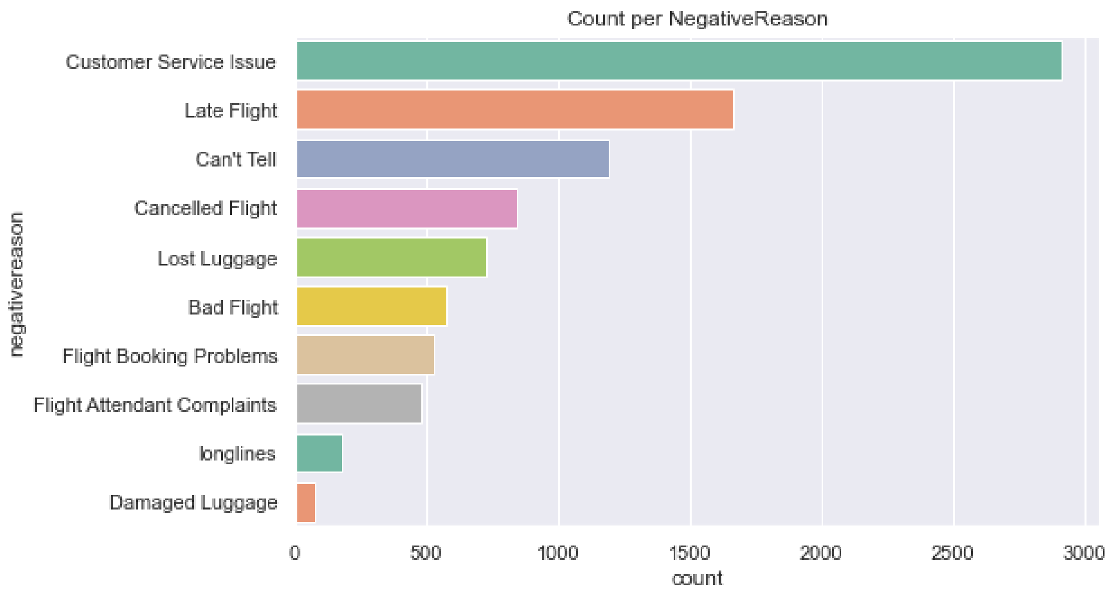
[129]:
```
# Visual of airline companies review counts.
ax = sns.countplot(data = df, y = 'airline',
                   order = df.airline.value_counts().index)
ax.set_title('Count per Airlines',)

plt.show()
```



[130]:
```
#Airline companies sentiment visualization.
sns.countplot(data = df, x ="airline", hue = "airline_sentiment");
sns.set(rc={"figure.figsize":(12, 6)})
```

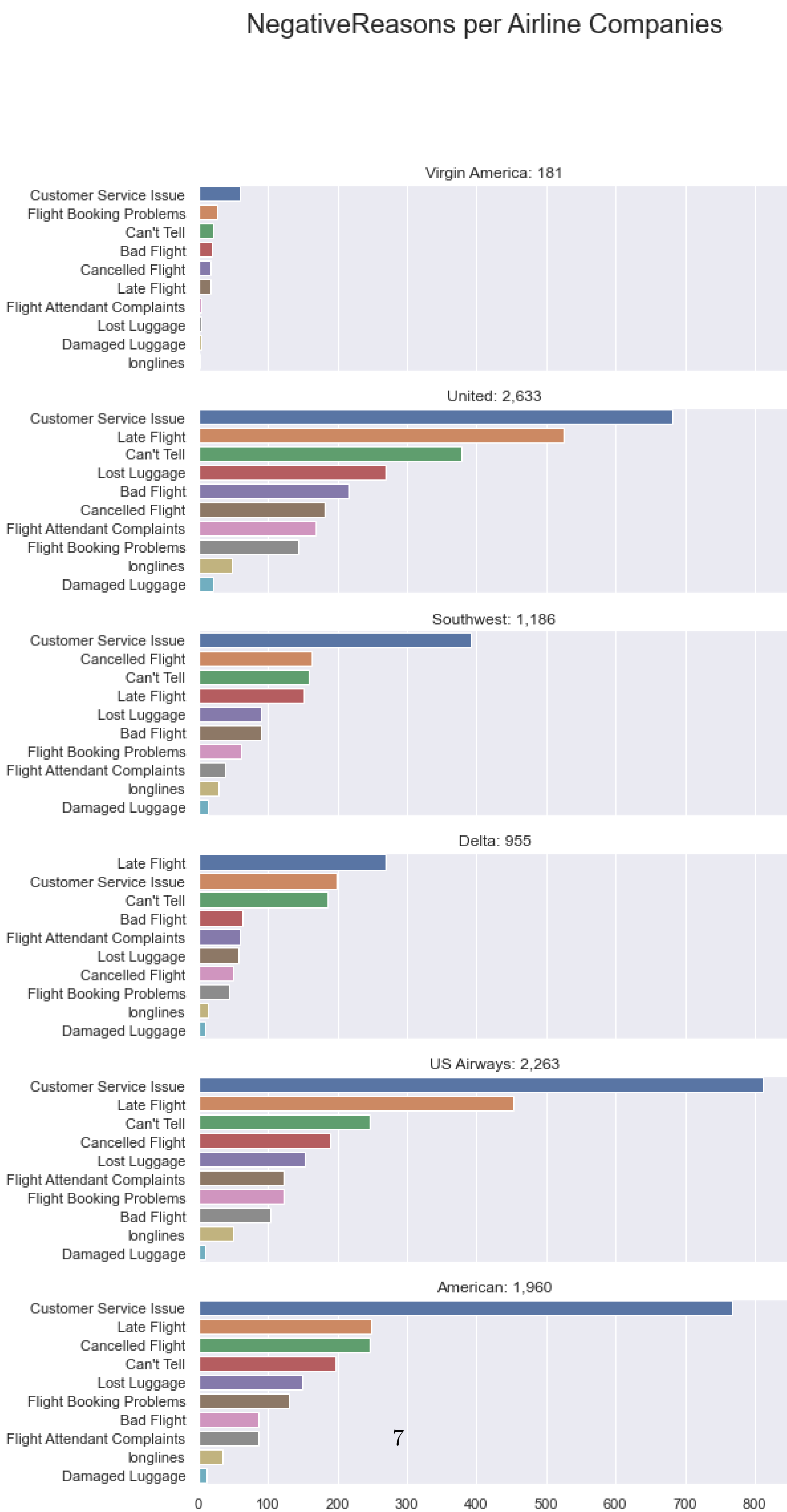


[131]:
```python
# Total negative reasons visual.
plt.figure(figsize=(8,5))
ax = sns.countplot(data = df_neg, y = 'negativereason',
                   palette='Set2',
                   order = df_neg.negativereason.value_counts().index)
ax.set_title('Count per NegativeReason')

plt.show()
```



[202]:
```python
# Negative reasons per airline companies.
fig, axes = plt.subplots(6,1, figsize=(8,18), sharex=True)
axes = axes.flatten()
names = df_neg['airline'].unique()

for name, n in zip(names, axes):
    ax = sns.countplot(data = df_neg[df_neg.airline==name], y =␣
 ↪'negativereason',
                       order = df_neg[df_neg.airline==name].negativereason.
 ↪value_counts().index, ax = n)
    ax.set_title(f"{name}: {format(len(df_neg[df_neg.airline==name]),',')}")
    ax.set_xlabel('')
    ax.set_ylabel('')

plt.suptitle("NegativeReasons per Airline Companies", fontsize = 20)
```

```
plt.show()
```


NegativeReasons per Airline Companies
Virgin America: 181
United: 2,633
Southwest: 1,186
Delta: 955
US Airways: 2,263
American: 1,960
Customer Service Issue
Flight Booking Problems
Can't Tell
Bad Flight
Cancelled Flight
Late Flight
Flight Attendant Complaints
Lost Luggage
Damaged Luggage
longlines
0
100
200
300
400
500
600
700
800

American, US Airways, Southwest: Complaints about customer sevice issue is relatively high.

United : Customer service issue is the most, but customers for this airline experienced late flight more frequently than others. Lost luggage issue happened relatively high.

Delta: Customer service looks not bad, but most of customers experienced late flight.

Virgin America: Mostly about customer service followed by flight booking problem.

0.4 Cleaning

[133]:
```
# Copying data for secure original.
df2 = df.copy()
```

[135]:
```
# Cleaning process from non alphabetic characters.
df2["text"] = df2["text"].str.replace("(@+\w+)", "")
df2["text"].head()
```

[135]:
```
0                                           What  said.
1      plus you've added commercials to the experien…
2      I didn't today… Must mean I need to take an…
3      it's really aggressive to blast obnoxious "en…
4              and it's a really big bad thing about it
Name: text, dtype: object
```

[136]:
```
# Creating variable for english stopwords.
stop_words = stopwords.words('english')
```

[137]:
```
# Creating function for cleaning, tokenize and lemmatization.
def cleaning(data):

    #Tokenize
    text_tokens = word_tokenize(data.replace("'", "").lower())

    #Remove punctuations
    tokens_without_punc = [w for w in text_tokens if w.isalpha()]

    #Removing Stopwords
    tokens_without_sw = [t for t in tokens_without_punc if t not in stop_words]

    #lemma
    text_cleaned = [WordNetLemmatizer().lemmatize(t) for t in tokens_without_sw]

    #joining
    return " ".join(text_cleaned)
```

[138]:
```python
#Applying function to target.
df2["text"] = df2["text"].apply(cleaning)
df2["text"].head()
```

[138]:
```
0                                                 said
1          plus youve added commercial experience tacky
2          didnt today must mean need take another trip
3    really aggressive blast obnoxious entertainmen…
4                                 really big bad thing
Name: text, dtype: object
```

[19]:
```python
" ".join(df2["text"]).split()
```

[19]:
```
['said',
 'plus',
 'youve',
 'added',
 'commercial',
 'experience',
 'tacky',
 'didnt',
 'today',
 'must',
 'mean',
 'need',
 'take',
 'another',
 'trip',
 'really',
 'aggressive',
 'blast',
 'obnoxious',
 'entertainment',
 'guest',
 'face',
 'amp',
 'little',
 'recourse',
 'really',
 'big',
 'bad',
 'thing',
 'seriously',
 'would',
 'pay',
 'flight',
 'seat',
```

```
'didnt',
'playing',
'really',
'bad',
'thing',
'flying',
'va',
'yes',
'nearly',
'every',
'time',
'fly',
'vx',
'ear',
'worm',
'go',
'away',
'really',
'missed',
'prime',
'opportunity',
'men',
'without',
'hat',
'parody',
'http',
'well',
'amazing',
'arrived',
'hour',
'early',
'youre',
'good',
'know',
'suicide',
'second',
'leading',
'cause',
'death',
'among',
'teen',
'lt',
'pretty',
'graphic',
'much',
'better',
'minimal',
```

```
'iconography',
'great',
'deal',
'already',
'thinking',
'trip',
'amp',
'havent',
'even',
'gone',
'trip',
'yet',
'p',
'im',
'flying',
'fabulous',
'seductive',
'sky',
'u',
'take',
'stress',
'away',
'travel',
'http',
'thanks',
'schedule',
'still',
'mia',
'excited',
'first',
'cross',
'country',
'flight',
'lax',
'mco',
'ive',
'heard',
'nothing',
'great',
'thing',
'virgin',
'america',
'flew',
'nyc',
'sfo',
'last',
'week',
```

```
'couldnt',
'fully',
'sit',
'seat',
'due',
'two',
'large',
'gentleman',
'either',
'side',
'help',
'flying',
'know',
'would',
'amazingly',
'awesome',
'please',
'want',
'fly',
'first',
'fare',
'may',
'three',
'time',
'carrier',
'seat',
'available',
'select',
'love',
'graphic',
'http',
'love',
'hipster',
'innovation',
'feel',
'good',
'brand',
'making',
'bos',
'gt',
'la',
'non',
'stop',
'permanently',
'anytime',
'soon',
'guy',
```

```
'messed',
'seating',
'reserved',
'seating',
'friend',
'guy',
'gave',
'seat',
'away',
'want',
'free',
'internet',
'status',
'match',
'program',
'applied',
'three',
'week',
'called',
'emailed',
'response',
'happened',
'ur',
'vegan',
'food',
'option',
'least',
'say',
'ur',
'site',
'know',
'wont',
'able',
'eat',
'anything',
'next',
'hr',
'fail',
'miss',
'dont',
'worry',
'well',
'together',
'soon',
'amazing',
'cant',
'get',
```

```
'cold',
'air',
'vent',
'noair',
'worstflightever',
'roasted',
'sfotobos',
'lax',
'ewr',
'middle',
'seat',
'red',
'eye',
'noob',
'maneuver',
'sendambien',
'andchexmix',
'hi',
'bked',
'cool',
'birthday',
'trip',
'cant',
'add',
'elevate',
'cause',
'entered',
'middle',
'name',
'flight',
'booking',
'problem',
'hour',
'operation',
'club',
'sfo',
'posted',
'online',
'current',
'help',
'left',
'expensive',
'headphone',
'flight',
'iad',
'lax',
'today',
```

```
'seat',
'one',
'answering',
'l',
'amp',
'f',
'number',
'lax',
'awaiting',
'return',
'phone',
'call',
'would',
'prefer',
'use',
'online',
'option',
'great',
'news',
'america',
'could',
'start',
'flight',
'hawaii',
'end',
'year',
'http',
'via',
'nice',
'rt',
'vibe',
'moodlight',
'takeoff',
'touchdown',
'moodlitmonday',
'sciencebehindtheexperience',
'http',
'moodlighting',
'way',
'fly',
'best',
'experience',
'ever',
'cool',
'calming',
'moodlitmonday',
'done',
```

```
'done',
'best',
'airline',
'around',
'hand',
'book',
'flight',
'hawaii',
'chat',
'support',
'working',
'site',
'http',
'view',
'downtown',
'los',
'angeles',
'hollywood',
'sign',
'beyond',
'rain',
'mountain',
'http',
'hey',
'first',
'time',
'flyer',
'next',
'week',
'excited',
'im',
'hard',
'time',
'getting',
'flight',
'added',
'elevate',
'account',
'help',
'plz',
'help',
'win',
'bid',
'upgrade',
'flight',
'lax',
'gt',
```

```
'sea',
'unused',
'ticket',
'moved',
'new',
'city',
'dont',
'fly',
'fly',
'expires',
'travelhelp',
'flight',
'leaving',
'dallas',
'seattle',
'time',
'feb',
'im',
'elevategold',
'good',
'reason',
'rock',
'dream',
'http',
'http',
'wow',
'blew',
'mind',
'last',
'night',
'tribute',
'soundofmusic',
'think',
'agree',
'entertaining',
'flight',
'way',
'supposed',
'take',
'minute',
'ago',
'website',
'still',
'show',
'time',
'flight',
'thanks',
```

```
'julie',
'andrew',
'way',
'though',
'impressive',
'wish',
'flew',
'atlanta',
'soon',
'julie',
'andrew',
'hand',
'flight',
'leaving',
'dallas',
'la',
'february',
'hi',
'im',
'excited',
'gt',
'dal',
'ive',
'trying',
'book',
'since',
'last',
'week',
'amp',
'page',
'never',
'load',
'thx',
'know',
'need',
'spotify',
'stat',
'guiltypleasures',
'im',
'lady',
'gaga',
'amazing',
'carrie',
'new',
'marketing',
'song',
'http',
```

```
'let',
'u',
'know',
'think',
'julie',
'andrew',
'first',
'lady',
'gaga',
'wowd',
'last',
'night',
'carrie',
'meh',
'called',
'week',
'ago',
'adding',
'flight',
'elevate',
'still',
'havent',
'shown',
'help',
'great',
'go',
'carrieunderwood',
'sorry',
'mary',
'martin',
'first',
'love',
'three',
'really',
'cant',
'beat',
'classic',
'flight',
'dal',
'dca',
'tried',
'check',
'could',
'status',
'please',
'heyyyy',
'guyyyys',
```

```
'trying',
'get',
'hour',
'someone',
'call',
'please',
'hi',
'virgin',
'im',
'hold',
'minute',
'earlier',
'flight',
'la',
'nyc',
'tonight',
'earlier',
'congrats',
'winning',
'award',
'best',
'deal',
'airline',
'u',
'http',
'everything',
'fine',
'lost',
'bag',
'need',
'change',
'reservation',
'virgin',
'credit',
'card',
'need',
'modify',
'phone',
'waive',
'change',
'fee',
'online',
'emailed',
'customer',
'service',
'team',
'let',
```

```
'know',
'need',
'tracking',
'number',
'hi',
'booked',
'flight',
'need',
'add',
'baggage',
'airline',
'awesome',
'lax',
'loft',
'need',
'step',
'game',
'dirty',
'table',
'floor',
'http',
'worried',
'great',
'ride',
'new',
'plane',
'great',
'crew',
'airline',
'like',
'awesome',
'flew',
'yall',
'sat',
'morning',
'way',
'correct',
'bill',
'watch',
'best',
'student',
'film',
'country',
'foot',
'http',
'first',
'time',
```

```
'flying',
'different',
'medium',
'bag',
'thanks',
'going',
'customer',
'service',
'anyway',
'speak',
'human',
'asap',
'thank',
'happened',
'doom',
'cant',
'supp',
'biz',
'traveler',
'like',
'customer',
'service',
'like',
'neverflyvirginforbusiness',
'ive',
'applied',
'member',
'inflight',
'crew',
'team',
'im',
'interested',
'flightattendant',
'dreampath',
'youre',
'best',
'whenever',
'begrudgingly',
'use',
'airline',
'im',
'delayed',
'late',
'flight',
'interesting',
'flying',
'cancelled',
```

```
'flight',
'next',
'four',
'flight',
'neverflyvirginforbusiness',
'disappointing',
'experience',
'shared',
'every',
'business',
'traveler',
'meet',
'neverflyvirgin',
'trouble',
'adding',
'flight',
'wife',
'booked',
'elevate',
'account',
'help',
'http',
'cant',
'bring',
'reservation',
'online',
'using',
'flight',
'booking',
'problem',
'code',
'random',
'q',
'whats',
'distribution',
'elevate',
'avatar',
'bet',
'kitty',
'disproportionate',
'share',
'http',
'lt',
'flying',
'va',
'life',
'happens',
```

```
'trying',
'change',
'trip',
'jperhi',
'home',
'page',
'let',
'site',
'back',
'rnp',
'yeah',
'know',
'hi',
'get',
'point',
'elevate',
'account',
'recent',
'flight',
'add',
'flight',
'point',
'account',
'like',
'tv',
'interesting',
'video',
'disappointed',
'cancelled',
'flightled',
'flight',
'flight',
'went',
'jfk',
'saturday',
'landed',
'lax',
'hour',
'late',
'flight',
'bag',
'check',
'business',
'travel',
'friendly',
'nomorevirgin',
'flight',
```

```
'redirected',
'website',
'btw',
'new',
'website',
'isnt',
'great',
'user',
'experience',
'time',
'another',
'redesign',
'cant',
'check',
'add',
'bag',
'website',
'isnt',
'working',
'ive',
'tried',
'desktop',
'mobile',
'http',
'let',
'scanned',
'passenger',
'leave',
'plane',
'told',
'someone',
'remove',
'bag',
'class',
'bin',
'uncomfortable',
'phone',
'number',
'cant',
'find',
'call',
'flight',
'reservation',
'anyone',
'anything',
'today',
'website',
```

```
'useless',
'one',
'answering',
'phone',
'trying',
'add',
'boy',
'prince',
'ressie',
'sf',
'thursday',
'lax',
'http',
'must',
'traveler',
'miss',
'flight',
'late',
'flight',
'check',
'bag',
'missed',
'morning',
'appointment',
'lost',
'business',
'check',
'new',
'music',
'http',
'hows',
'direct',
'flight',
'gt',
'sfo',
'unexpected',
'layover',
'vega',
'fuel',
'yet',
'peep',
'next',
'bought',
'vega',
'flight',
'sneaky',
'late',
```

```
'flight',
'bag',
'check',
'lost',
'business',
'missed',
'flight',
'apt',
'three',
'people',
'flight',
'exp',
'amazing',
'customer',
'service',
'raeann',
'sf',
'shes',
'best',
'customerservice',
'virginamerica',
'flying',
'called',
'service',
'line',
'hung',
'awesome',
'sarcasm',
'site',
'tripping',
'im',
'trying',
'check',
'im',
'getting',
'plain',
'text',
'version',
'reluctant',
'enter',
'card',
'info',
'scheduled',
'sfo',
'dal',
'flight',
'today',
```

```
'changed',
'due',
'weather',
'look',
'like',
'flight',
'still',
'getaway',
'deal',
'may',
'lot',
'cool',
'city',
'http',
'cheapflights',
'farecompare',
'getaway',
'deal',
'may',
'lot',
'cool',
'city',
'http',
'cheapflights',
'farecompare',
'getaway',
'deal',
'may',
'lot',
'cool',
'city',
'http',
'cheapflights',
'farecompare',
'getaway',
'deal',
'may',
'lot',
'cool',
'city',
'http',
'cheapflights',
'farecompare',
'great',
'week',
'come',
'back',
```

```
'phl',
'already',
'need',
'take',
'u',
'horrible',
'cold',
'pleasecomeback',
'http',
'concerned',
'fly',
'plane',
'need',
'delayed',
'due',
'tech',
'stop',
'best',
'airline',
'flown',
'change',
'reservation',
'helpful',
'representative',
'amp',
'comfortable',
'flying',
'experience',
'another',
'rep',
'kicked',
'butt',
'naelah',
'represents',
'team',
'beautifully',
'thank',
'beautiful',
'design',
'right',
'cool',
'still',
'book',
'ticket',
'secure',
'love',
'team',
```

```
 'running',
 'gate',
 'la',
 'tonight',
 'waited',
 'delayed',
 'flight',
 'kept',
 'thing',
 'entertaining',
 'use',
 'another',
 'browser',
 'amp',
 'brand',
 'reputation',
 'built',
 'tech',
 'response',
 'doesnt',
 'compatible',
 'website',
 'flight',
 'flight',
 'booking',
 'problem',
 …]
```

[139]:
```
# Removing all unnecessary columns.
df2 =df2[["airline_sentiment", "text"]]
df2.head()
```

[139]:

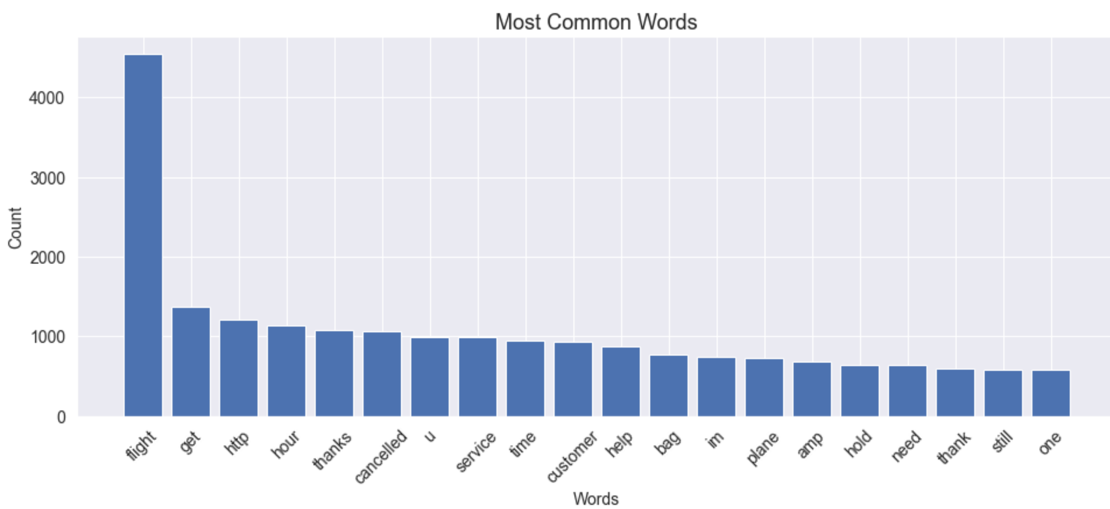
	airline_sentiment	text
0	neutral	said
1	positive	plus youve added commercial experience tacky
2	neutral	didnt today must mean need take another trip
3	negative	really aggressive blast obnoxious entertainmen…
4	negative	really big bad thing

[140]:
```
# Counting most common words.
corpus = " ".join(df2["text"])
tokens_count = Counter(word_tokenize(corpus)).most_common(20)
tokens_count
```

[140]:
```
[('flight', 4544),
 ('get', 1374),
 ('http', 1210),
```

```
('hour', 1138),
('thanks', 1078),
('cancelled', 1056),
('u', 994),
('service', 989),
('time', 946),
('customer', 934),
('help', 869),
('bag', 766),
('im', 743),
('plane', 725),
('amp', 683),
('hold', 642),
('need', 633),
('thank', 602),
('still', 580),
('one', 580)]
```

[167]:
```
# Visaul of most common words.
dic= dict(tokens_count)
fig, ax = plt.subplots(figsize=(16,6))
ax.bar(dic.keys(),dic.values())
ax.set_title('Most Common Words',fontsize=18)
plt.xlabel('Words',fontsize=14)
plt.ylabel('Count',fontsize=14)
ax = plt.gca()
ax.tick_params(labelsize = 14)
plt.xticks(rotation=45)
plt.show()
```



0.5 Train Test Split

[28]:
```
from sklearn.model_selection import train_test_split
```

[168]:
```
# Train test split.
X = df2["text"]
y= df2["airline_sentiment"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪stratify=y, random_state=101)
```

0.6 Count Vectorizer

[30]:
```
from sklearn.feature_extraction.text import CountVectorizer
```

[31]:
```
# Initializing Count Vectorizer.
vectorizer = CountVectorizer()
X_train_count = vectorizer.fit_transform(X_train)
X_test_count = vectorizer.transform(X_test)
```

[32]:
```
# Looking train set into array.
X_train_count.toarray()
```

[32]:
```
array([[0, 0, 0, …, 0, 0, 0],
       [0, 0, 0, …, 0, 0, 0],
       [0, 0, 0, …, 0, 0, 0],
       …,
       [0, 0, 0, …, 0, 0, 0],
       [0, 0, 0, …, 0, 0, 0],
       [1, 0, 0, …, 0, 0, 0]], dtype=int64)
```

[169]:
```
# Look dataframe after process.
pd.DataFrame(X_train_count.toarray(), columns = vectorizer.get_feature_names())
```

[169]:
```
       aa  aaadvantage  aaalwayslate  aadavantage  aadelay  aadv  aadvantage  \
0       0            0             0            0        0     0           0
1       0            0             0            0        0     0           0
2       0            0             0            0        0     0           0
3       0            0             0            0        0     0           0
4       0            0             0            0        0     0           0
…      ..          …             …            …        …     …          …
11707   0            0             0            0        0     0           0
11708   0            0             0            0        0     0           0
11709   0            0             0            0        0     0           0
11710   0            0             0            0        0     0           0
11711   1            0             0            0        0     0           0

       aafail  aaron  aateam  ab  aback  abandoned  abandonment  abassinet  \
0           0      0       0   0      0          0            0          0
```

```
1            0      0       0   0      0          0            0          0
2            0      0       0   0      0          0            0          0
3            0      0       0   0      0          0            0          0
4            0      0       0   0      0          0            0          0
…        …    …     …  ..   …       …         …        …
11707        0      0       0   0      0          0            0          0
11708        0      0       0   0      0          0            0          0
11709        0      0       0   0      0          0            0          0
11710        0      0       0   0      0          0            0          0
11711        0      0       0   0      0          0            0          0

       abbreve  abc  abcletjetbluestreamfeed  abcnews  abducted  ability  \
0            0    0                        0        0         0        0
1            0    0                        0        0         0        0
2            0    0                        0        0         0        0
3            0    0                        0        0         0        0
4            0    0                        0        0         0        0
…          …  …                      …      …       …      …
11707        0    0                        0        0         0        0
11708        0    0                        0        0         0        0
11709        0    0                        0        0         0        0
11710        0    0                        0        0         0        0
11711        0    0                        0        0         0        0

       able  aboard  aboout  abounds  …  yousuck  yout  youth  youve  \
0         0       0       0        0  …        0     0      0      0
1         0       0       0        0  …        0     0      0      0
2         0       0       0        0  …        0     0      0      0
3         0       0       0        0  …        0     0      0      0
4         0       0       0        0  …        0     0      0      0
…     …     …     …      …  …      …   …    …    …
11707     0       0       0        0  …        0     0      0      0
11708     0       0       0        0  …        0     0      0      0
11709     0       0       0        0  …        0     0      0      0
11710     0       0       0        0  …        0     0      0      0
11711     0       0       0        0  …        0     0      0      0

       yponthebeat  yr  ystrdy  yuck  yucki  yuma  yummy  yup  yvonne  yvr  \
0                0   0       0     0      0     0      0    0       0    0
1                0   0       0     0      0     0      0    0       0    0
2                0   0       0     0      0     0      0    0       0    0
3                0   0       0     0      0     0      0    0       0    0
4                0   0       0     0      0     0      0    0       0    0
…             …  ..     …   …    …   …     …  …      …  …
11707            0   0       0     0      0     0      0    0       0    0
11708            0   0       0     0      0     0      0    0       0    0
11709            0   0       0     0      0     0      0    0       0    0
```

```
11710              0    0        0      0       0      0       0     0         0     0
11711              0    0        0      0       0      0       0     0         0     0

       yxe  yyc  yyj  yyz  zabsonre  zero  zfv  zipper  zone  zrh  zurich
0        0    0    0    0         0     0    0       0     0    0       0
1        0    0    0    0         0     0    0       0     0    0       0
2        0    0    0    0         0     0    0       0     0    0       0
3        0    0    0    0         0     0    0       0     0    0       0
4        0    0    0    0         0     0    0       0     0    0       0
…     …   …   …   …       …   …   …     …   …   …     …
11707    0    0    0    0         0     0    0       0     0    0       0
11708    0    0    0    0         0     0    0       0     0    0       0
11709    0    0    0    0         0     0    0       0     0    0       0
11710    0    0    0    0         0     0    0       0     0    0       0
11711    0    0    0    0         0     0    0       0     0    0       0

[11712 rows x 8796 columns]
```

[34]:
```python
from sklearn.metrics import plot_confusion_matrix,classification_report,␣
↪f1_score, recall_score
from sklearn.pipeline import Pipeline
```

[170]:
```python
#Creating function to evaluate our models.
def evaluation(model, X_train, X_test):
    y_pred = model.predict(X_test)
    y_pred_train = model.predict(X_train)
    print("==== Train Set ====")
    print(classification_report(y_train,y_pred_train))
    print("==== Test Set ====")
    print(classification_report(y_test,y_pred))
    plot_confusion_matrix(model,X_test, y_test)
```

0.6.1 Logistic Regression

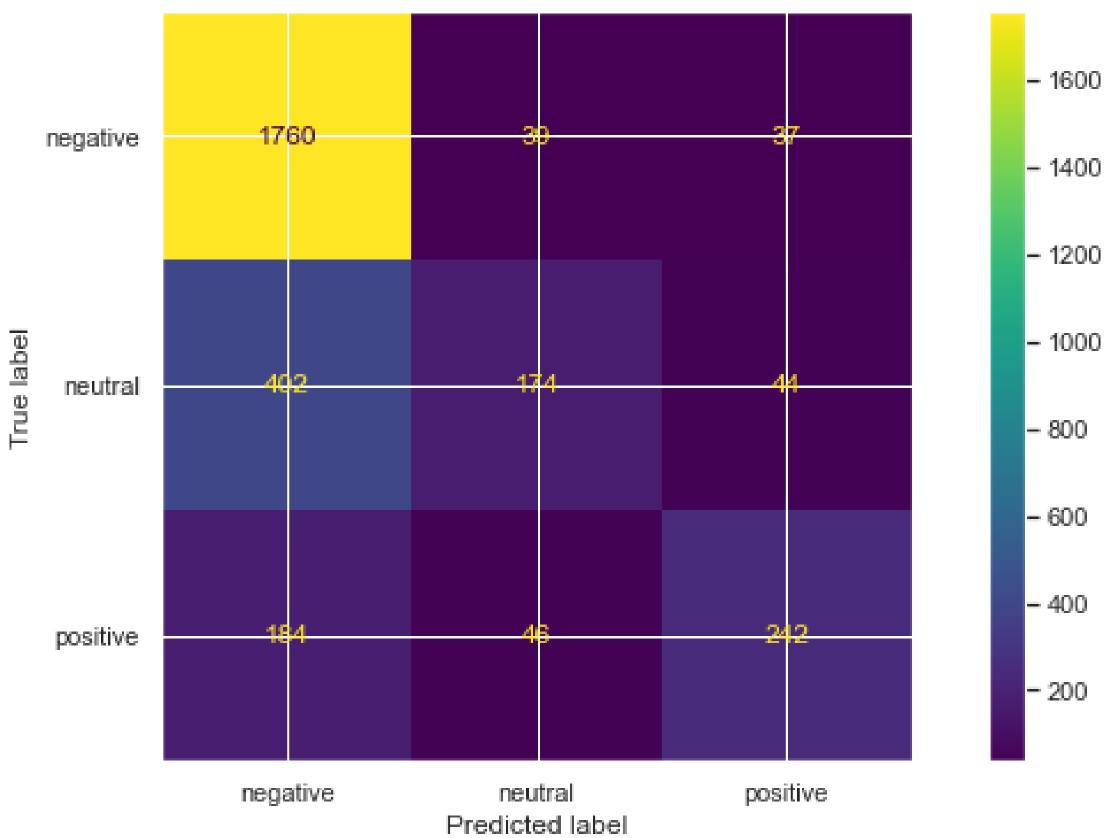
[36]:
```python
# Initiliazing first model.
from sklearn.linear_model import LogisticRegression

log = LogisticRegression(C = 0.02, max_iter=1000)
log.fit(X_train_count,y_train)
```

[36]:
```
LogisticRegression(C=0.02, max_iter=1000)
```

[37]:
```python
print("LOG MODEL")
evaluation(log, X_train_count, X_test_count)
```

```
LOG MODEL
==== Train Set ====
              precision    recall  f1-score   support
```

```
    negative       0.75      0.97      0.85      7342
     neutral       0.76      0.33      0.46      2479
    positive       0.80      0.51      0.62      1891

    accuracy                           0.76     11712
   macro avg       0.77      0.60      0.64     11712
weighted avg       0.76      0.76      0.73     11712

==== Test Set ====
              precision    recall  f1-score   support

    negative       0.75      0.96      0.84      1836
     neutral       0.67      0.28      0.40       620
    positive       0.75      0.51      0.61       472

    accuracy                           0.74      2928
   macro avg       0.72      0.58      0.62      2928
weighted avg       0.73      0.74      0.71      2928
```



0.6.2 Naive Bayes

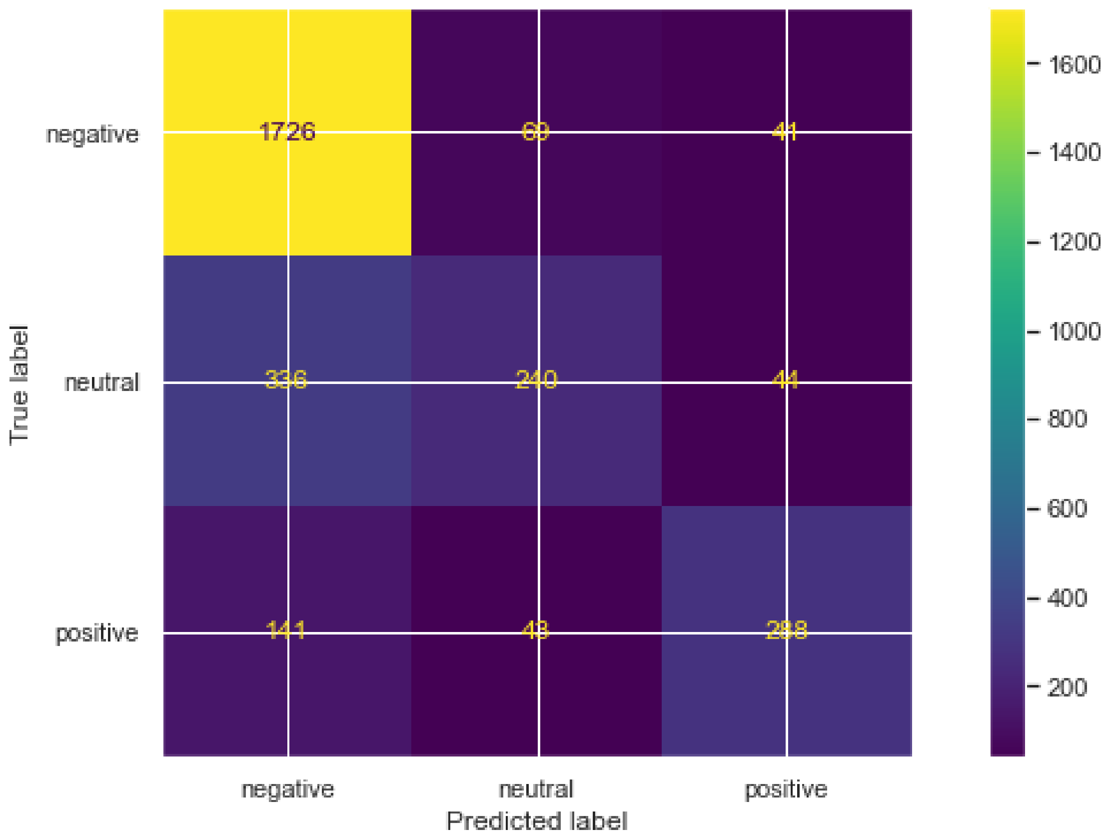
[41]:
```
from sklearn.naive_bayes import MultinomialNB, GaussianNB
```

[42]:
```
#Initiliazing second model.
nb = MultinomialNB()
nb.fit(X_train_count,y_train)
```

[42]:
```
MultinomialNB()
```

[43]:
```
print("NB MODEL")
evaluation(nb, X_train_count, X_test_count)
```

```
NB MODEL
==== Train Set ====
              precision    recall  f1-score   support

    negative       0.84      0.96      0.90      7342
     neutral       0.83      0.58      0.68      2479
    positive       0.86      0.74      0.80      1891

    accuracy                           0.84     11712
   macro avg       0.85      0.76      0.79     11712
weighted avg       0.84      0.84      0.84     11712

==== Test Set ====
              precision    recall  f1-score   support

    negative       0.78      0.94      0.85      1836
     neutral       0.68      0.39      0.49       620
    positive       0.77      0.61      0.68       472

    accuracy                           0.77      2928
   macro avg       0.75      0.65      0.68      2928
weighted avg       0.76      0.77      0.75      2928
```



0.6.3 Ada Boost

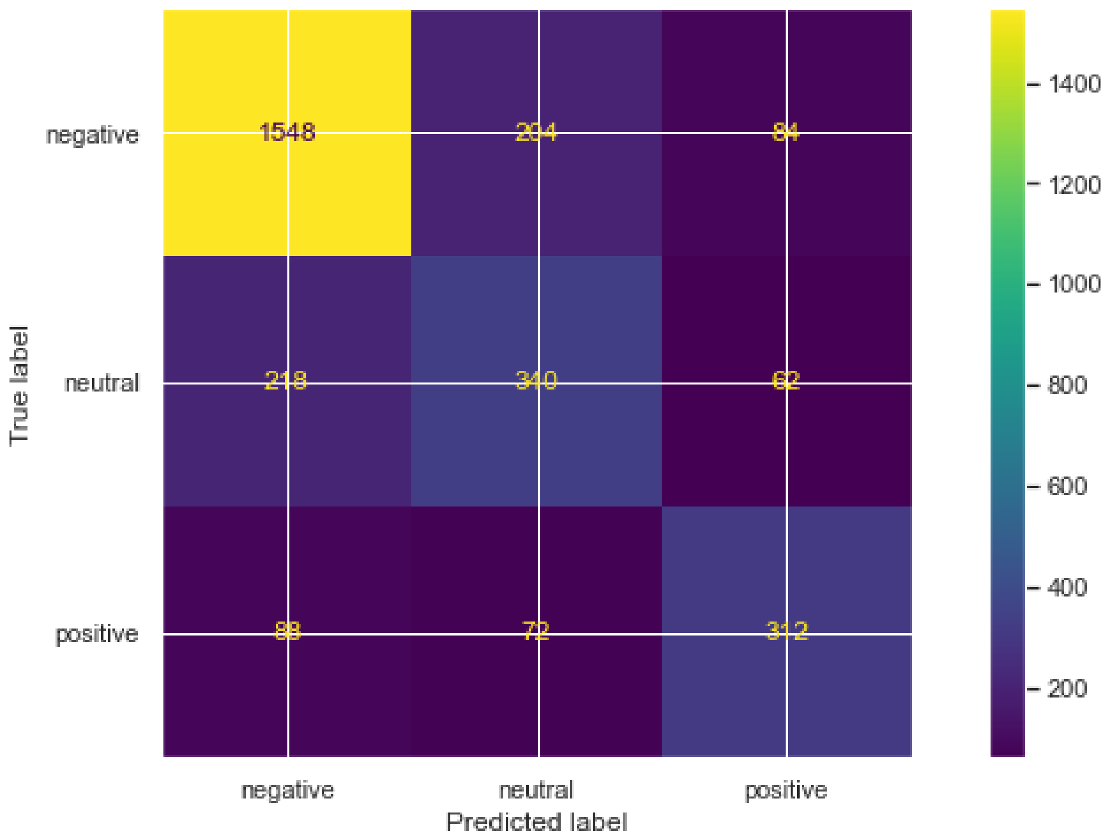
[44]:
```
from sklearn.ensemble import AdaBoostClassifier
#Initiliazing third model.
ada = AdaBoostClassifier(n_estimators= 500, random_state = 42)
ada.fit(X_train_count, y_train)
```

[44]:
```
AdaBoostClassifier(n_estimators=500, random_state=42)
```

[45]:
```
print("Ada MODEL")
evaluation(ada, X_train_count, X_test_count)
```

```
Ada MODEL
==== Train Set ====
              precision    recall  f1-score   support

    negative       0.86      0.88      0.87      7342
     neutral       0.61      0.61      0.61      2479
    positive       0.79      0.72      0.75      1891

    accuracy                           0.80     11712
```

```
   macro avg       0.75      0.73      0.74     11712
weighted avg       0.79      0.80      0.79     11712

==== Test Set ====
              precision    recall  f1-score   support

    negative       0.83      0.84      0.84      1836
     neutral       0.55      0.55      0.55       620
    positive       0.68      0.66      0.67       472

    accuracy                           0.75      2928
   macro avg       0.69      0.68      0.69      2928
weighted avg       0.75      0.75      0.75      2928
```



0.7 TF-IDF

[47]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

[48]:
```
# #Initiliazing TF-IDF.
tf_idf_vectorizer = TfidfVectorizer()
X_train_tf_idf = tf_idf_vectorizer.fit_transform(X_train)
X_test_tf_idf = tf_idf_vectorizer.transform(X_test)
```

[49]:
```
# Looking train set into array.
X_train_tf_idf.toarray()
```

[49]:
```
array([[0.        , 0.        , 0.        , …, 0.        , 0.        ,
         0.        ],
       [0.        , 0.        , 0.        , …, 0.        , 0.        ,
         0.        ],
       [0.        , 0.        , 0.        , …, 0.        , 0.        ,
         0.        ],
       …,
       [0.        , 0.        , 0.        , …, 0.        , 0.        ,
         0.        ],
       [0.        , 0.        , 0.        , …, 0.        , 0.        ,
         0.        ],
       [0.32332003, 0.        , 0.        , …, 0.        , 0.        ,
         0.        ]])
```

[50]:
```
# Look dataframe after process.
pd.DataFrame(X_train_tf_idf.toarray(), columns = tf_idf_vectorizer.
 ↪get_feature_names())
```

[50]:
```
            aa  aaadvantage  aaalwayslate  aadavantage  aadelay  aadv  \
0      0.00000          0.0           0.0          0.0      0.0   0.0
1      0.00000          0.0           0.0          0.0      0.0   0.0
2      0.00000          0.0           0.0          0.0      0.0   0.0
3      0.00000          0.0           0.0          0.0      0.0   0.0
4      0.00000          0.0           0.0          0.0      0.0   0.0
…          …          …            …          …      …   …
11707  0.00000          0.0           0.0          0.0      0.0   0.0
11708  0.00000          0.0           0.0          0.0      0.0   0.0
11709  0.00000          0.0           0.0          0.0      0.0   0.0
11710  0.00000          0.0           0.0          0.0      0.0   0.0
11711  0.32332          0.0           0.0          0.0      0.0   0.0

       aadvantage  aafail  aaron  aateam   ab  aback  abandoned  abandonment  \
0             0.0     0.0    0.0     0.0  0.0    0.0        0.0          0.0
1             0.0     0.0    0.0     0.0  0.0    0.0        0.0          0.0
2             0.0     0.0    0.0     0.0  0.0    0.0        0.0          0.0
3             0.0     0.0    0.0     0.0  0.0    0.0        0.0          0.0
4             0.0     0.0    0.0     0.0  0.0    0.0        0.0          0.0
…             …     …    …     …  …    …        …          …
11707         0.0     0.0    0.0     0.0  0.0    0.0        0.0          0.0
```

```
11708        0.0      0.0     0.0      0.0  0.0     0.0         0.0           0.0
11709        0.0      0.0     0.0      0.0  0.0     0.0         0.0           0.0
11710        0.0      0.0     0.0      0.0  0.0     0.0         0.0           0.0
11711        0.0      0.0     0.0      0.0  0.0     0.0         0.0           0.0

       abassinet  abbreve  abc  abcletjetbluestreamfeed  abcnews  abducted  \
0            0.0      0.0  0.0                      0.0      0.0       0.0
1            0.0      0.0  0.0                      0.0      0.0       0.0
2            0.0      0.0  0.0                      0.0      0.0       0.0
3            0.0      0.0  0.0                      0.0      0.0       0.0
4            0.0      0.0  0.0                      0.0      0.0       0.0
…            …        …   …                      …        …         …
11707        0.0      0.0  0.0                      0.0      0.0       0.0
11708        0.0      0.0  0.0                      0.0      0.0       0.0
11709        0.0      0.0  0.0                      0.0      0.0       0.0
11710        0.0      0.0  0.0                      0.0      0.0       0.0
11711        0.0      0.0  0.0                      0.0      0.0       0.0

       ability  able  aboard  aboout  abounds  …  yousuck  yout  youth  \
0          0.0   0.0     0.0     0.0      0.0  …      0.0   0.0    0.0
1          0.0   0.0     0.0     0.0      0.0  …      0.0   0.0    0.0
2          0.0   0.0     0.0     0.0      0.0  …      0.0   0.0    0.0
3          0.0   0.0     0.0     0.0      0.0  …      0.0   0.0    0.0
4          0.0   0.0     0.0     0.0      0.0  …      0.0   0.0    0.0
…          …   …     …      …       …  …      …    …     …
11707      0.0   0.0     0.0     0.0      0.0  …      0.0   0.0    0.0
11708      0.0   0.0     0.0     0.0      0.0  …      0.0   0.0    0.0
11709      0.0   0.0     0.0     0.0      0.0  …      0.0   0.0    0.0
11710      0.0   0.0     0.0     0.0      0.0  …      0.0   0.0    0.0
11711      0.0   0.0     0.0     0.0      0.0  …      0.0   0.0    0.0

       youve  yponthebeat   yr  ystrdy  yuck  yucki  yuma  yummy  yup  yvonne  \
0        0.0          0.0  0.0     0.0   0.0    0.0   0.0    0.0  0.0     0.0
1        0.0          0.0  0.0     0.0   0.0    0.0   0.0    0.0  0.0     0.0
2        0.0          0.0  0.0     0.0   0.0    0.0   0.0    0.0  0.0     0.0
3        0.0          0.0  0.0     0.0   0.0    0.0   0.0    0.0  0.0     0.0
4        0.0          0.0  0.0     0.0   0.0    0.0   0.0    0.0  0.0     0.0
…      …          …   …    …   …     …   …    …    …   …       …
11707    0.0          0.0  0.0     0.0   0.0    0.0   0.0    0.0  0.0     0.0
11708    0.0          0.0  0.0     0.0   0.0    0.0   0.0    0.0  0.0     0.0
11709    0.0          0.0  0.0     0.0   0.0    0.0   0.0    0.0  0.0     0.0
11710    0.0          0.0  0.0     0.0   0.0    0.0   0.0    0.0  0.0     0.0
11711    0.0          0.0  0.0     0.0   0.0    0.0   0.0    0.0  0.0     0.0

       yvr  yxe  yyc  yyj  yyz  zabsonre  zero  zfv  zipper  zone  zrh  zurich
0      0.0  0.0  0.0  0.0  0.0       0.0   0.0  0.0     0.0   0.0  0.0     0.0
1      0.0  0.0  0.0  0.0  0.0       0.0   0.0  0.0     0.0   0.0  0.0     0.0
```

```
2      0.0  0.0  0.0  0.0  0.0       0.0   0.0  0.0      0.0   0.0  0.0     0.0
3      0.0  0.0  0.0  0.0  0.0       0.0   0.0  0.0      0.0   0.0  0.0     0.0
4      0.0  0.0  0.0  0.0  0.0       0.0   0.0  0.0      0.0   0.0  0.0     0.0
…     …    …    …    …    …          …    …    …    …    …    …      …
11707  0.0  0.0  0.0  0.0  0.0       0.0   0.0  0.0      0.0   0.0  0.0     0.0
11708  0.0  0.0  0.0  0.0  0.0       0.0   0.0  0.0      0.0   0.0  0.0     0.0
11709  0.0  0.0  0.0  0.0  0.0       0.0   0.0  0.0      0.0   0.0  0.0     0.0
11710  0.0  0.0  0.0  0.0  0.0       0.0   0.0  0.0      0.0   0.0  0.0     0.0
11711  0.0  0.0  0.0  0.0  0.0       0.0   0.0  0.0      0.0   0.0  0.0     0.0

[11712 rows x 8796 columns]
```

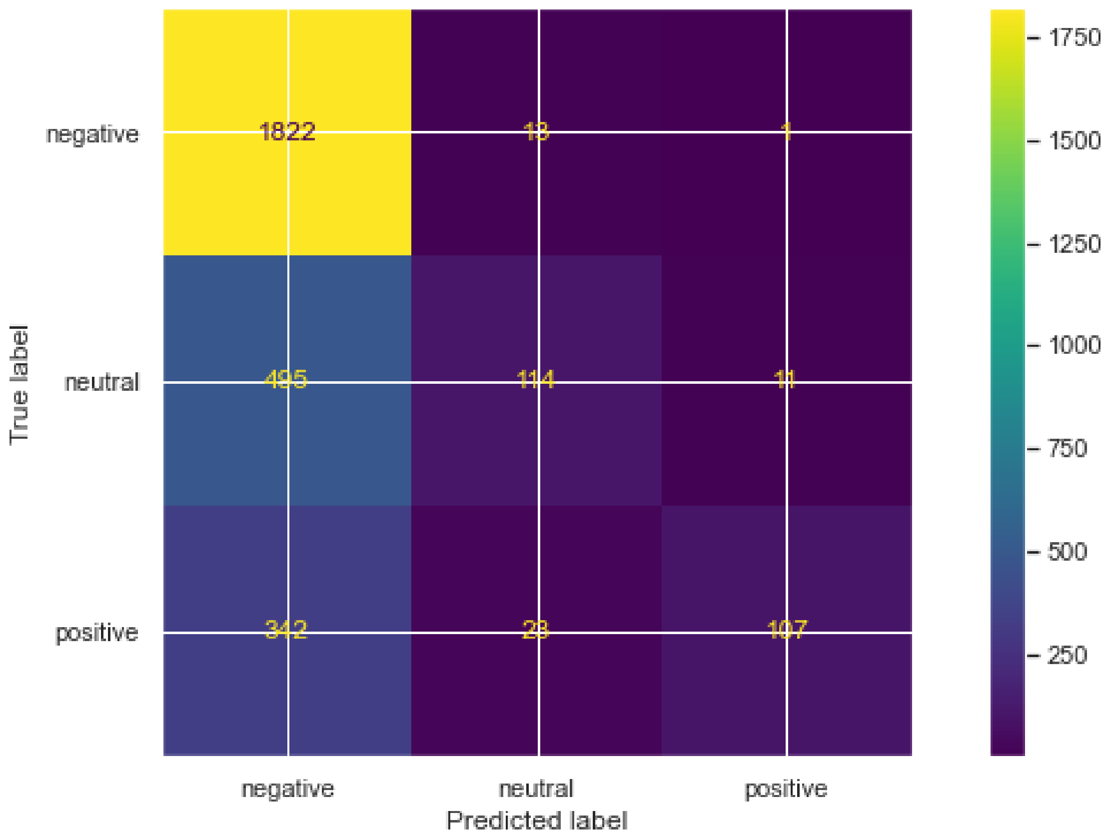
0.7.1 Naive Bayes

[51]:
```python
from sklearn.naive_bayes import MultinomialNB, BernoulliNB
#Initiliazing first model.
nb = MultinomialNB()
nb.fit(X_train_tf_idf,y_train)
```

[51]:
```
MultinomialNB()
```

[52]:
```python
print("NB MODEL")
evaluation(nb, X_train_tf_idf, X_test_tf_idf)
```

```
NB MODEL
==== Train Set ====
              precision    recall  f1-score   support

    negative       0.72      1.00      0.84      7342
     neutral       0.90      0.30      0.46      2479
    positive       0.95      0.39      0.55      1891

    accuracy                           0.75     11712
   macro avg       0.86      0.56      0.61     11712
weighted avg       0.80      0.75      0.71     11712

==== Test Set ====
              precision    recall  f1-score   support

    negative       0.69      0.99      0.81      1836
     neutral       0.76      0.18      0.30       620
    positive       0.90      0.23      0.36       472

    accuracy                           0.70      2928
   macro avg       0.78      0.47      0.49      2928
weighted avg       0.74      0.70      0.63      2928
```



0.7.2 Logistic Regression

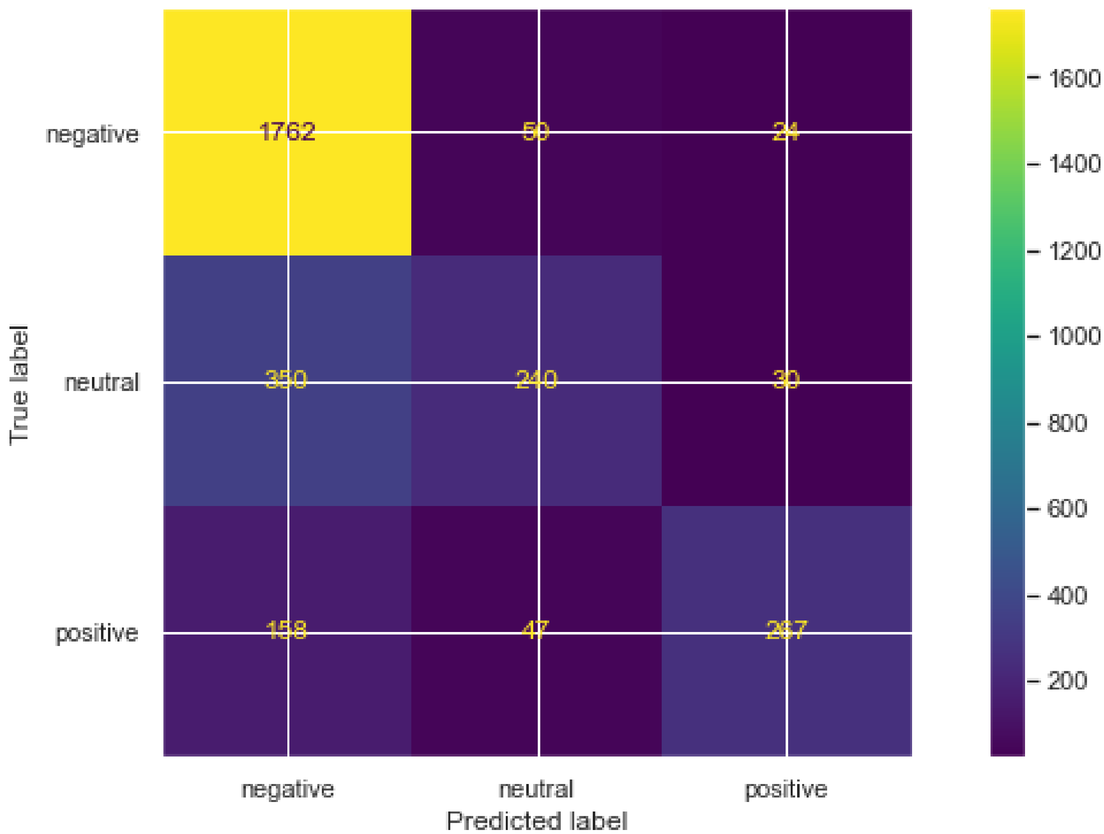
[53]:
```
#Initiliazing second model.
log = LogisticRegression(C=0.4, max_iter=1000)
log.fit(X_train_tf_idf,y_train)
```

[53]:
```
LogisticRegression(C=0.4, max_iter=1000)
```

[54]:
```
print("LOG MODEL")
evaluation(log, X_train_tf_idf, X_test_tf_idf)
```

```
LOG MODEL
==== Train Set ====
              precision    recall  f1-score   support

    negative       0.80      0.98      0.88      7342
     neutral       0.84      0.51      0.63      2479
    positive       0.89      0.59      0.71      1891

    accuracy                           0.82     11712
   macro avg       0.84      0.69      0.74     11712
```

```
weighted avg       0.82      0.82      0.80     11712

==== Test Set ====
              precision    recall  f1-score   support

    negative       0.78      0.96      0.86      1836
     neutral       0.71      0.39      0.50       620
    positive       0.83      0.57      0.67       472

    accuracy                           0.77      2928
   macro avg       0.77      0.64      0.68      2928
weighted avg       0.77      0.77      0.75      2928
```



0.7.3 Random Forest

[55]:
```python
from sklearn.ensemble import RandomForestClassifier
```

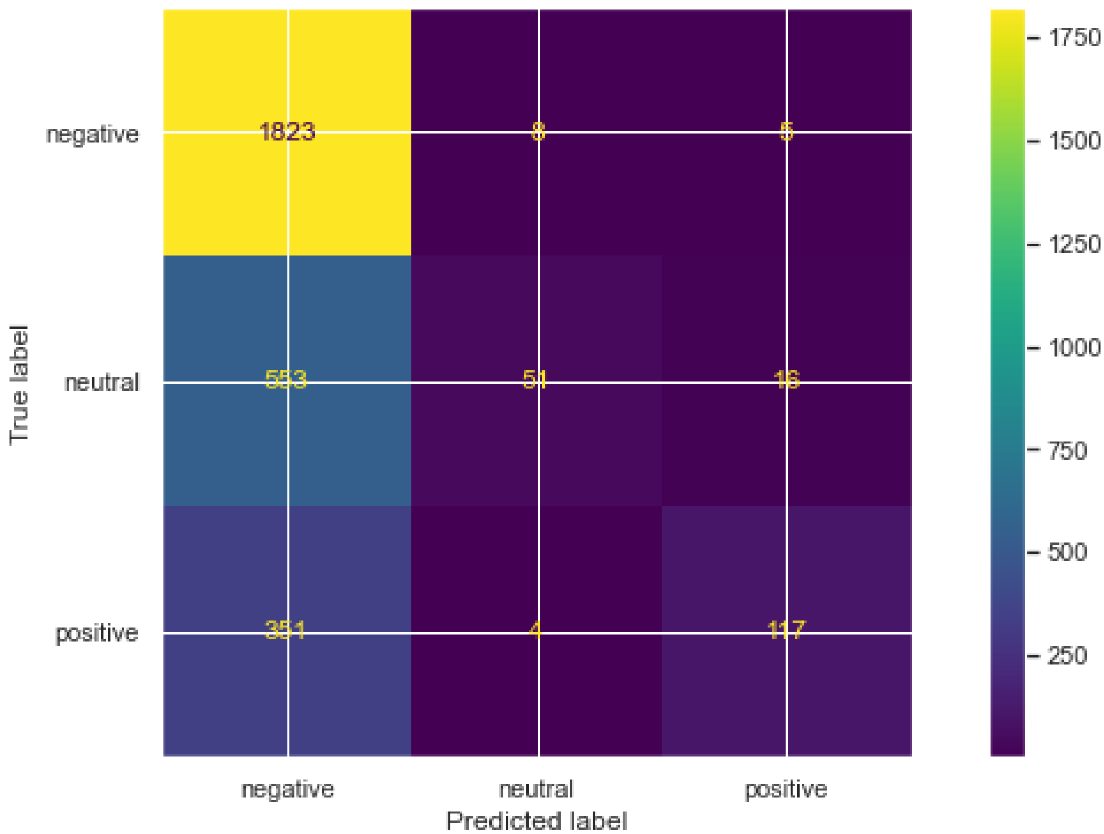
[56]:
```python
#Initiliazing third model.
rf = RandomForestClassifier(100, max_depth=40, random_state = 42, n_jobs = -1)
```

```python
rf.fit(X_train_tf_idf, y_train)
```

[56]: RandomForestClassifier(max_depth=40, n_jobs=-1, random_state=42)

[57]:
```python
print("RF MODEL")
evaluation(rf, X_train_tf_idf, X_test_tf_idf)
```

```
RF MODEL
==== Train Set ====
              precision    recall  f1-score   support

    negative       0.69      1.00      0.82      7342
     neutral       0.98      0.15      0.25      2479
    positive       0.97      0.37      0.53      1891

    accuracy                           0.72     11712
   macro avg       0.88      0.50      0.53     11712
weighted avg       0.80      0.72      0.65     11712

==== Test Set ====
              precision    recall  f1-score   support

    negative       0.67      0.99      0.80      1836
     neutral       0.81      0.08      0.15       620
    positive       0.85      0.25      0.38       472

    accuracy                           0.68      2928
   macro avg       0.78      0.44      0.44      2928
weighted avg       0.73      0.68      0.59      2928
```



Best model is Random Forest until here. Looks like it did pretty well on negative class but need to improve neutral and positive classes too.

[58]:
```python
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.tree import DecisionTreeClassifier
```

0.7.4 Gradient Boosting

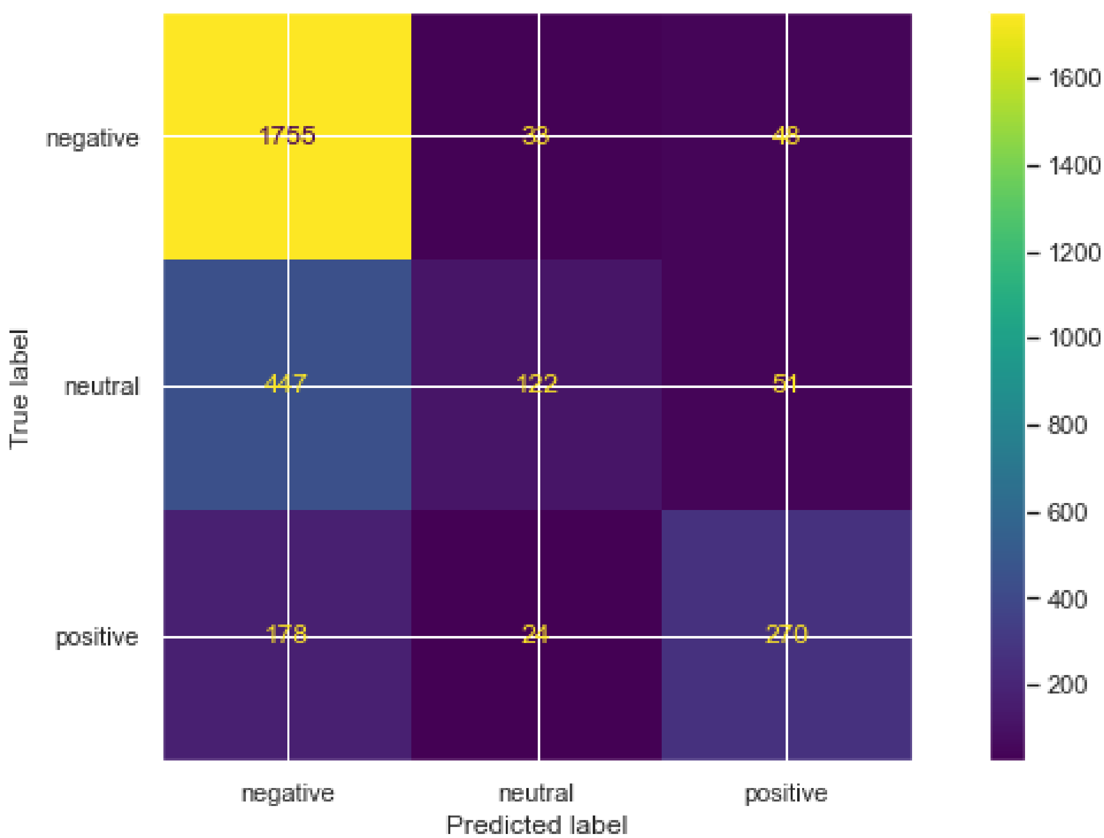
[59]:
```python
#Initiliazing fourth model.
gb = GradientBoostingClassifier()
gb.fit(X_train_tf_idf, y_train)
```

[59]:
```
GradientBoostingClassifier()
```

[60]:
```python
print("RF MODEL")
evaluation(gb, X_train_tf_idf, X_test_tf_idf)
```

```
RF MODEL
==== Train Set ====
              precision    recall  f1-score   support
```

```
    negative       0.74      0.97      0.84      7342
     neutral       0.78      0.24      0.37      2479
    positive       0.79      0.56      0.66      1891

    accuracy                           0.75     11712
   macro avg       0.77      0.59      0.62     11712
weighted avg       0.76      0.75      0.71     11712

==== Test Set ====
              precision    recall  f1-score   support

    negative       0.74      0.96      0.83      1836
     neutral       0.68      0.20      0.31       620
    positive       0.73      0.57      0.64       472

    accuracy                           0.73      2928
   macro avg       0.72      0.57      0.59      2928
weighted avg       0.72      0.73      0.69      2928
```



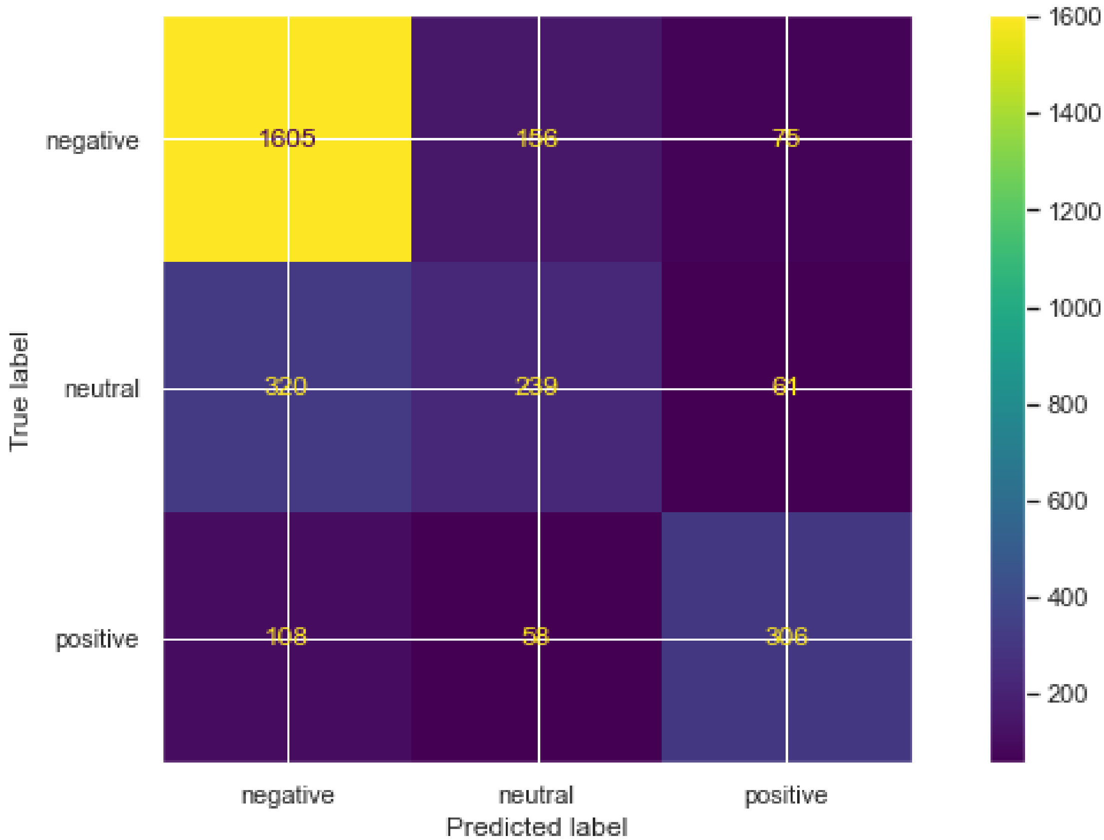
0.7.5 Ada Boost

[61]:
```
#Initiliazing fifth model.
ada = AdaBoostClassifier(n_estimators= 500, random_state = 42)
ada.fit(X_train_tf_idf, y_train)
```

[61]:
```
AdaBoostClassifier(n_estimators=500, random_state=42)
```

[62]:
```
print("Ada MODEL")
evaluation(ada, X_train_tf_idf, X_test_tf_idf)
```

```
Ada MODEL
==== Train Set ====
              precision    recall  f1-score   support

    negative       0.81      0.92      0.86      7342
     neutral       0.63      0.43      0.51      2479
    positive       0.83      0.73      0.78      1891

    accuracy                           0.78     11712
   macro avg       0.76      0.69      0.72     11712
weighted avg       0.77      0.78      0.77     11712

==== Test Set ====
              precision    recall  f1-score   support

    negative       0.79      0.87      0.83      1836
     neutral       0.53      0.39      0.45       620
    positive       0.69      0.65      0.67       472

    accuracy                           0.73      2928
   macro avg       0.67      0.64      0.65      2928
weighted avg       0.72      0.73      0.72      2928
```



0.8 Prediction

[63]:
```
#Importing pipeline.
from sklearn.pipeline import Pipeline
```

[64]:
```
# Initializing pipeline.
pipe = Pipeline([('tfidf',TfidfVectorizer()),('log',LogisticRegression(C=0.4,␣
↪max_iter=1000))])
```

[65]:
```
#Fitting.
pipe.fit(X, y)
```

[65]:
```
Pipeline(steps=[('tfidf', TfidfVectorizer()),
                ('log', LogisticRegression(C=0.4, max_iter=1000))])
```

[66]:
```
#Example prediction.
tweet = "it was not the worst flight i have ever been"
tweet = pd.Series(tweet).apply(cleaning)
pipe.predict(tweet)
```

[66]: array(['negative'], dtype=object)

[67]:
```
#Example prediction.
tweet = "don't enjoy flight"
tweet = pd.Series(tweet).apply(cleaning)
pipe.predict(tweet)
```

[67]: array(['negative'], dtype=object)

[68]:
```
#Example prediction.
tweet = "ok flight"
tweet = pd.Series(tweet).apply(cleaning)
pipe.predict(tweet)
```

[68]: array(['neutral'], dtype=object)

[69]:
```
#Example prediction.
tweet = "doesn't enjoy flight"
tweet = pd.Series(tweet).apply(cleaning)
pipe.predict(tweet)
```

[69]: array(['negative'], dtype=object)

[171]:
```
#Example prediction.(""" WRONG PREDICTION BY MODEL """)
tweet = "liked"
tweet = pd.Series(tweet).apply(cleaning)
pipe.predict(tweet)
```

[171]: array(['negative'], dtype=object)

0.9 Sequential

[70]:
```
#Importing necessary modules.
from tensorflow.keras import layers, models
```

[72]:
```
#Importing necessary modules.
import gensim
from gensim.models import Word2Vec
```

[76]:
```
# Remembering data.
df2
```

[76]:

	airline_sentiment	text
0	neutral	said
1	positive	plus youve added commercial experience tacky
2	neutral	didnt today must mean need take another trip
3	negative	really aggressive blast obnoxious entertainmen…
4	negative	really big bad thing

```
…                …                                              …
14635         positive                 thank got different flight chicago
14636         negative  leaving minute late flight warning communicati…
14637          neutral                      please bring american airline
14638         negative  money change flight dont answer phone suggesti…
14639          neutral  ppl need know many seat next flight plz put u …

[14640 rows x 2 columns]
```

[77]:
```
#Creating target and feature.
target = df2['airline_sentiment']
data = df2['text'].map(word_tokenize).values
```

[80]:
```
# Creating function to tokenize.
def tokenize(d):
    return word_tokenize(d)
```

[81]:
```
# Creating variable for tokenized target variable.
texts_w2v = df2.text.apply(tokenize).to_list()
```

0.9.1 Word2Vec Model

[82]:
```
# Initializing Word2Vec model.
w2v = Word2Vec(sentences = texts_w2v, window = 3,
               vector_size = 100, min_count = 5, workers = 4, sg = 1)
```

[83]:
```
# Looking for tokenized and listed data.
texts_w2v[:5]
```

[83]:
```
[['said'],
 ['plus', 'youve', 'added', 'commercial', 'experience', 'tacky'],
 ['didnt', 'today', 'must', 'mean', 'need', 'take', 'another', 'trip'],
 ['really',
  'aggressive',
  'blast',
  'obnoxious',
  'entertainment',
  'guest',
  'face',
  'amp',
  'little',
  'recourse'],
 ['really', 'big', 'bad', 'thing']]
```

Similar words with the given word examples

[84]:
```
#Looking for similar word with given words.
w2v.wv.most_similar('thank')
```

[84]:
```
[('much', 0.9669926762580872),
 ('quick', 0.9581903219223022),
 ('appreciate', 0.9512640833854675),
 ('amazing', 0.9477537274360657),
 ('twitter', 0.9317639470100403),
 ('care', 0.9286403656005859),
 ('awesome', 0.9187070727348328),
 ('complaint', 0.9129829406738281),
 ('medium', 0.9096567034721375),
 ('tweet', 0.9080104827880859)]
```

[85]:
```
#Looking for similar word with given words.
w2v.wv.most_similar('customerservice')
```

[85]:
```
[('nightmare', 0.994503378868103),
 ('neveragain', 0.9943042397499084),
 ('biggest', 0.9914339184761047),
 ('abysmal', 0.9903689026832581),
 ('horrendous', 0.9900234937667847),
 ('bullshit', 0.9893454313278198),
 ('ashamed', 0.9892138838768005),
 ('loved', 0.9888815879821777),
 ('est', 0.9888523817062378),
 ('learned', 0.9888007044792175)]
```

[86]:
```
#Looking for similar word with given words.
w2v.wv.most_similar('crew')
```

[86]:
```
[('pilot', 0.917129397392273),
 ('ground', 0.8944821357727051),
 ('jfk', 0.8787472248077393),
 ('attendant', 0.8657433986663818),
 ('plane', 0.85694420337677),
 ('staff', 0.8503057360649109),
 ('landing', 0.8486275672912598),
 ('made', 0.848362147808075),
 ('ord', 0.8384466767311096),
 ('san', 0.8360778093338013)]
```

[87]:
```
#Looking for similar word with given words.
w2v.wv.most_similar('delay')
```

[87]:
```
[('delayed', 0.9398424029350281),
 ('ewr', 0.9230515956878662),
 ('stuck', 0.9222317337989807),
 ('sfo', 0.9217750430107117),
 ('maintenance', 0.9214389324188232),
 ('due', 0.9181495904922485),
```

```
 ('mechanical', 0.9120343327522278),
 ('phx', 0.9107290506362915),
 ('phl', 0.9033175706863403),
 ('jfk', 0.9022048115730286)]
```

[88]:
```
#Looking for similar word with given words.
w2v.wv.most_similar('ticket')
```

[88]:
```
[('fee', 0.9149150252342224),
 ('award', 0.9119688868522644),
 ('name', 0.9083679914474487),
 ('credit', 0.902630090713501),
 ('refund', 0.9007839560508728),
 ('use', 0.8909323215484619),
 ('add', 0.8878641128540039),
 ('booked', 0.8862524628639221),
 ('pay', 0.8824276328086853),
 ('companion', 0.8818210363388062)]
```

[89]:
```
# Creating vectors for every text.
def get_avg_vector(sent):
    vector = np.zeros(100)
    total_words = 0
    for word in sent.split():
        if word in w2v.wv.index_to_key:     # don't use .wv.vocab method in␣
 ↪kaggle notebook. instead, use .wv.index_to_key method.
            vector += w2v.wv.word_vec(word)
            total_words += 1
    if total_words > 0:
        return vector / total_words
    else:
        return vector

df2['w2v_vector'] = df2['text'].map(get_avg_vector)
df2[['text', 'w2v_vector']].head(2)
```

[89]:
```
                                           text  \
0                                          said
1  plus youve added commercial experience tacky

                                         w2v_vector
0  [-0.10832026600837708, 0.027967197820544243, -…
1  [-0.07126817880198359, 0.12060708254575729, -0…
```

[90]:
```
#Importing necesarry modules.
from sklearn.svm import SVC
from sklearn.model_selection import StratifiedKFold
```

```
from sklearn.metrics import accuracy_score
```

[91]:
```
# Checking three different models accuracy for improve further.
model_params = {'random_state':42}
model_list = [LogisticRegression(**model_params, solver='liblinear'),
              RandomForestClassifier(**model_params),
#              MultinomialNB(),  # Don't use Naive Bayes since w2v_vector␣
↪contains negative numbers, then it causes an error.
              SVC(**model_params)]
model_name = ['LogisticRegression', 'RandomForest', 'SupportVectorMachine']

skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

for model, model_name in zip(model_list, model_name):
    for n_fold, (trn_idx, vld_idx) in enumerate(skf.split(df2.index, df2.
↪airline_sentiment)):
        X_trn = np.stack(df2.loc[trn_idx, 'w2v_vector'])
        y_trn = df2.loc[trn_idx, 'airline_sentiment']

        X_vld = np.stack(df2.loc[vld_idx, 'w2v_vector'])
        y_vld = df2.loc[vld_idx, 'airline_sentiment']

        model.fit(X_trn, y_trn)
        pred_col = f"{model_name}_w2v_pred"
        df2.loc[vld_idx, pred_col] = model.predict(X_vld)


    print(f"Model: {model_name}, Word2Vec, Accuracy: {accuracy_score(df2.
↪airline_sentiment, df2[pred_col]):.3%}\n")
```

```
Model: LogisticRegression, Word2Vec, Accuracy: 72.097%

Model: RandomForest, Word2Vec, Accuracy: 73.340%

Model: SupportVectorMachine, Word2Vec, Accuracy: 71.407%
```

[98]:
```
#Importing necesarry modules.
from keras.models import Sequential
from keras.layers import Dense, LSTM, Bidirectional,Embedding
from keras.layers import Dropout, Conv1D, MaxPooling1D
from keras.callbacks import EarlyStopping
from keras.layers import Dense, LSTM, Bidirectional,Embedding, Dropout, Conv1D,␣
↪MaxPooling1D
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
```

[99]:
```
#Making function for tokenize and padding.

max_words = 5000
max_len = 100

def tokenize_pad_sequences(text):
    '''
    This function tokenize the input text into sequnences of intergers and then
    pad each sequence to the same length
    '''
    # Text tokenization
    tokenizer = Tokenizer(num_words=max_words, lower=True, split=' ')
    tokenizer.fit_on_texts(text)
    # Transforms text to a sequence of integers
    X = tokenizer.texts_to_sequences(text)
    # Pad sequences to the same length
    X = pad_sequences(X, padding='post', maxlen=max_len)
    # return sequences
    return X, tokenizer

print('Before Tokenization & Padding \n', df2['text'][0],'\n')
X, tokenizer = tokenize_pad_sequences(df['text'])
print('After Tokenization & Padding \n', X[0])
```

```
Before Tokenization & Padding
 said

After Tokenization & Padding
 [ 81  62 226   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0]
```

[100]:
```
#Train test split.
y = pd.get_dummies(df.airline_sentiment)
X_trn, X_tst, y_trn, y_tst = train_test_split(X, y, test_size=0.2,␣
↪random_state=42, stratify=y)
X_trn, X_vld, y_trn, y_vld = train_test_split(X_trn, y_trn, test_size=0.3,␣
↪random_state=42, stratify=y_trn)

print('Train:          ', X_trn.shape, y_trn.shape)
print('Validation Set:', X_vld.shape, y_vld.shape)
print('Test Set:       ', X_tst.shape, y_tst.shape)
```

```
Train:          (8198, 100) (8198, 3)
Validation Set: (3514, 100) (3514, 3)
```

```
Test Set:       (2928, 100) (2928, 3)
```

0.9.2 Sequential Model

[101]:
```
#Creating necessary variables and initializing sequential model. Adding layers.
vocab_size = 5000
embedding_size = 32
epochs=50
max_words = 5000
max_len = 100
batch_size = 64

model= Sequential()
model.add(Embedding(vocab_size, embedding_size, input_length=max_len))
model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2, padding='same'))
model.add(Bidirectional(LSTM(32)))
model.add(Dropout(0.4))
model.add(Dense(3, activation='softmax'))
```

[102]:
```
#Compiling model. Looking into it.
model.compile(loss='categorical_crossentropy', optimizer='adam',␣
 ↪metrics=['accuracy'])
print(model.summary())
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 100, 32)           160000
_________________________________________________________________
conv1d (Conv1D)              (None, 100, 32)           3104
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 50, 32)            0
_________________________________________________________________
bidirectional (Bidirectional (None, 64)                16640
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 3)                 195
=================================================================
Total params: 179,939
Trainable params: 179,939
Non-trainable params: 0
_________________________________________________________________
None
```

[103]:
```
#Trying early stopping and fitting model.
es = EarlyStopping(monitor = 'val_loss', patience=5)
batch_size = 64

history = model.fit(X_trn, y_trn,validation_data=(X_vld,␣
↪y_vld),batch_size=batch_size, epochs=epochs, verbose=1,callbacks = es)
```

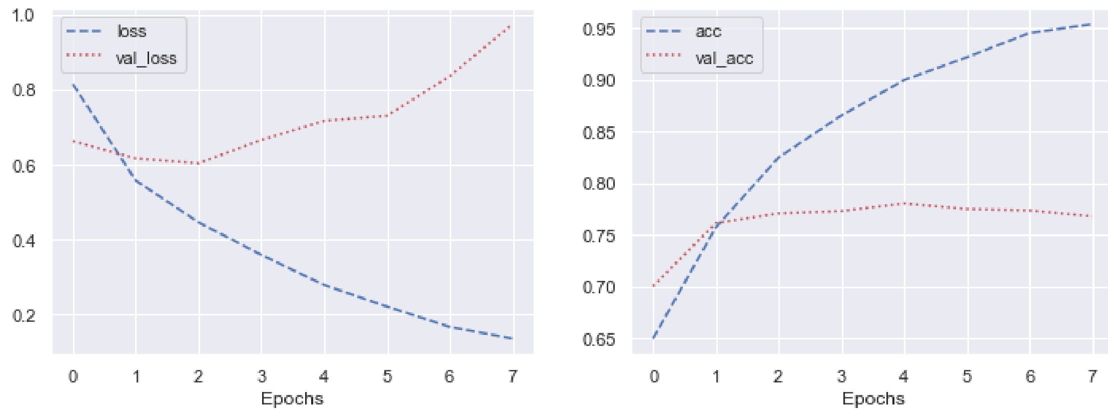
```
Epoch 1/50
129/129 [==============================] - 2s 19ms/step - loss: 0.8140 -
accuracy: 0.6494 - val_loss: 0.6619 - val_accuracy: 0.7001
Epoch 2/50
129/129 [==============================] - 2s 12ms/step - loss: 0.5572 -
accuracy: 0.7570 - val_loss: 0.6161 - val_accuracy: 0.7612
Epoch 3/50
129/129 [==============================] - 2s 12ms/step - loss: 0.4457 -
accuracy: 0.8248 - val_loss: 0.6031 - val_accuracy: 0.7706
Epoch 4/50
129/129 [==============================] - 2s 12ms/step - loss: 0.3590 -
accuracy: 0.8655 - val_loss: 0.6650 - val_accuracy: 0.7729
Epoch 5/50
129/129 [==============================] - 2s 12ms/step - loss: 0.2787 -
accuracy: 0.8999 - val_loss: 0.7160 - val_accuracy: 0.7803
Epoch 6/50
129/129 [==============================] - 2s 12ms/step - loss: 0.2213 -
accuracy: 0.9219 - val_loss: 0.7294 - val_accuracy: 0.7749
Epoch 7/50
129/129 [==============================] - 2s 12ms/step - loss: 0.1672 -
accuracy: 0.9455 - val_loss: 0.8347 - val_accuracy: 0.7735
Epoch 8/50
129/129 [==============================] - 2s 12ms/step - loss: 0.1359 -
accuracy: 0.9540 - val_loss: 0.9739 - val_accuracy: 0.7678
```

[104]:
```
# Evaluate model on the test set
loss, accuracy = model.evaluate(X_tst, y_tst, verbose=0)

# Print metrics
print('Accuracy  : {:.4f}'.format(accuracy))
```

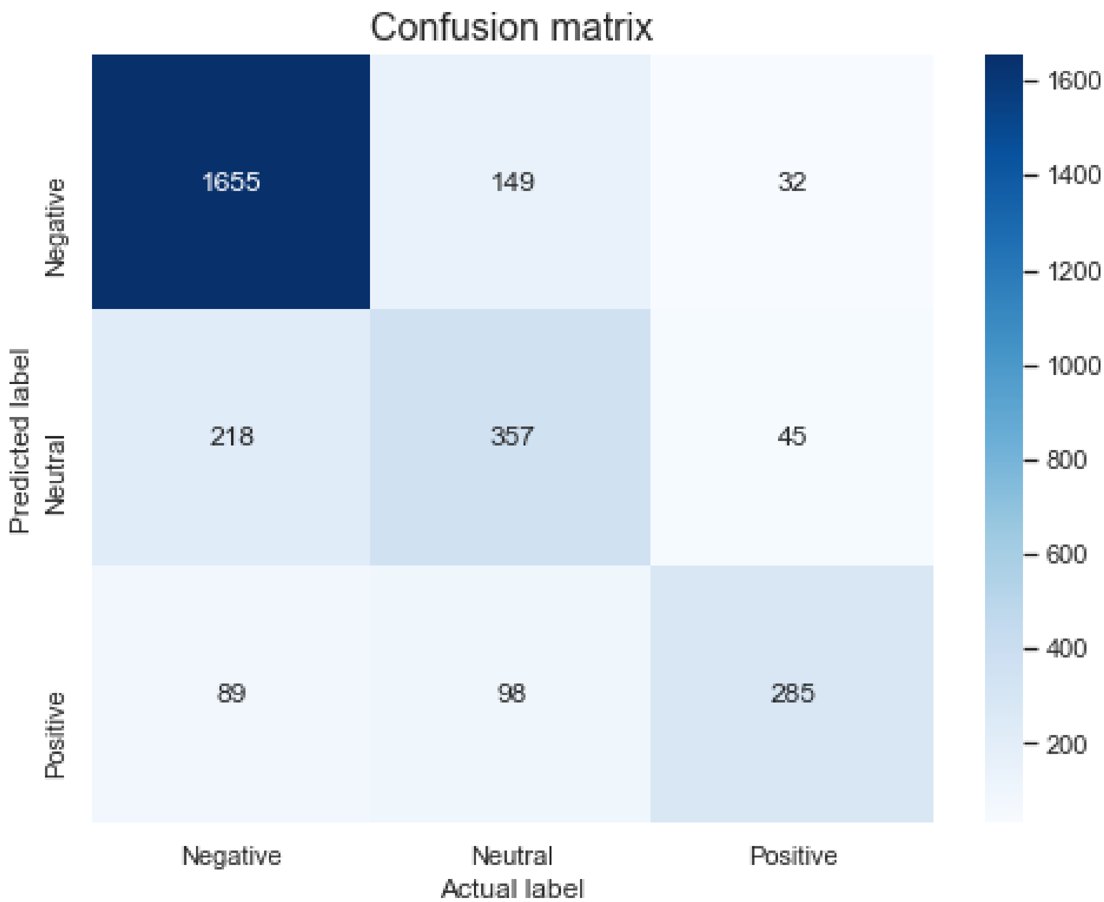
```
Accuracy  : 0.7845
```

[105]:
```
# Visualizing loss and accuracy on sequential model.
plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], 'b--', label = 'loss')
plt.plot(history.history['val_loss'], 'r:', label = 'val_loss')
plt.xlabel('Epochs')
plt.legend()
```

```
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], 'b--', label = 'acc')
plt.plot(history.history['val_accuracy'], 'r:', label = 'val_acc')
plt.xlabel('Epochs')
plt.legend()

plt.show()
```



[106]:
```
from sklearn.metrics import confusion_matrix

# Creating function to see confusion matrix for sequential model.

def plot_confusion_matrix(model, X_test, y_test):
    '''Function to plot confusion matrix for the passed model and the data'''

    sentiment_classes = ['Negative', 'Neutral', 'Positive']
    # use model to do the prediction
    y_pred = model.predict(X_test)
    # compute confusion matrix
    cm = confusion_matrix(np.argmax(np.array(y_test),axis=1), np.argmax(y_pred,␣
 ↪axis=1))
    # plot confusion matrix
    plt.figure(figsize=(8,6))
    sns.heatmap(cm, cmap=plt.cm.Blues, annot=True, fmt='d',
                xticklabels=sentiment_classes,
                yticklabels=sentiment_classes)
    plt.title('Confusion matrix', fontsize=16)
    plt.xlabel('Actual label', fontsize=12)
    plt.ylabel('Predicted label', fontsize=12)

plot_confusion_matrix(model, X_tst, y_tst)
```



0.9.3 Improved(Second) Model

[173]:
```
# Adding negative reasons from original data.
df2['negativereason'] = df['negativereason']
```

[174]:
```
# Filling missing values.
df2['negativereason'].fillna('missing', inplace=True)
```

[175]:
```
# Adding new column to data.
df2['reason_text'] = df2['negativereason'] + ' ' + df2['text']
df2.head()
```

[175]:
```
  airline_sentiment                                              text  \
0           neutral                                              said
1          positive       plus youve added commercial experience tacky
2           neutral       didnt today must mean need take another trip
3          negative  really aggressive blast obnoxious entertainmen…
4          negative                                  really big bad thing
```

```
  negativereason                                        reason_text
0        missing                                       missing said
1        missing  missing plus youve added commercial experience…
2        missing  missing didnt today must mean need take anothe…
3     Bad Flight  Bad Flight really aggressive blast obnoxious e…
4     Can't Tell                    Can't Tell really big bad thing
```

[176]:
```python
# Applying tokenize and padding function on new column.
print('Before Tokenization & Padding \n', df2['reason_text'][0],'\n')
X, tokenizer = tokenize_pad_sequences(df2['reason_text'])
print('After Tokenization & Padding \n', X[0])
```

```
Before Tokenization & Padding
 missing said

After Tokenization & Padding
 [  2 133   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0]
```

[114]:
```python
# Train test split.
X_trn, X_tst, y_trn, y_tst = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42, stratify=y)
X_trn, X_vld, y_trn, y_vld = train_test_split(X_trn, y_trn, test_size=0.3,␣
 ↪random_state=42, stratify=y_trn)

print('Train:          ', X_trn.shape, y_trn.shape)
print('Validation Set:', X_vld.shape, y_vld.shape)
print('Test Set:       ', X_tst.shape, y_tst.shape)
```

```
Train:          (8198, 100) (8198, 3)
Validation Set: (3514, 100) (3514, 3)
Test Set:       (2928, 100) (2928, 3)
```

[116]:
```python
# Initializing another sequential model.
vocab_size = 5000
embedding_size = 32
epochs=50

model= Sequential()
model.add(Embedding(vocab_size, embedding_size, input_length=max_len))
model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Bidirectional(LSTM(32)))
```

```
model.add(Dropout(0.4))
model.add(Dense(3, activation='softmax'))
```

[117]:
```
#Compiling and looking to model.
model.compile(loss='categorical_crossentropy', optimizer='adam',␣
↪metrics=['accuracy'])
print(model.summary())
```

```
Model: "sequential_3"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 100, 32)           160000

_________________________________________________________________
conv1d_2 (Conv1D)            (None, 100, 32)           3104

_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 50, 32)            0

_________________________________________________________________
bidirectional_2 (Bidirection (None, 64)                16640

_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0

_________________________________________________________________
dense_2 (Dense)              (None, 3)                 195
=================================================================
Total params: 179,939
Trainable params: 179,939
Non-trainable params: 0

_________________________________________________________________
None
```

[118]:
```
# Trying early stopping and fitting model.
es = EarlyStopping(monitor = 'val_loss', patience=5)
batch_size = 64

history = model.fit(X_trn, y_trn,
                    validation_data=(X_vld, y_vld),
                    batch_size=batch_size, epochs=epochs, verbose=1,
                    callbacks = [es])
```

```
Epoch 1/50
129/129 [==============================] - 2s 18ms/step - loss: 0.6676 -
accuracy: 0.7079 - val_loss: 0.2751 - val_accuracy: 0.8367
Epoch 2/50
129/129 [==============================] - 2s 12ms/step - loss: 0.2431 -
accuracy: 0.8644 - val_loss: 0.2313 - val_accuracy: 0.8919
Epoch 3/50
129/129 [==============================] - 2s 12ms/step - loss: 0.1382 -
accuracy: 0.9485 - val_loss: 0.1621 - val_accuracy: 0.9291
```

```
Epoch 4/50
129/129 [==============================] - 2s 12ms/step - loss: 0.0883 -
accuracy: 0.9673 - val_loss: 0.1820 - val_accuracy: 0.9266
Epoch 5/50
129/129 [==============================] - 2s 12ms/step - loss: 0.0628 -
accuracy: 0.9783 - val_loss: 0.2081 - val_accuracy: 0.9254
Epoch 6/50
129/129 [==============================] - 2s 12ms/step - loss: 0.0482 -
accuracy: 0.9841 - val_loss: 0.2427 - val_accuracy: 0.9229
Epoch 7/50
129/129 [==============================] - 2s 13ms/step - loss: 0.0375 -
accuracy: 0.9894 - val_loss: 0.2612 - val_accuracy: 0.9212
Epoch 8/50
129/129 [==============================] - 2s 13ms/step - loss: 0.0305 -
accuracy: 0.9918 - val_loss: 0.2775 - val_accuracy: 0.9220
```

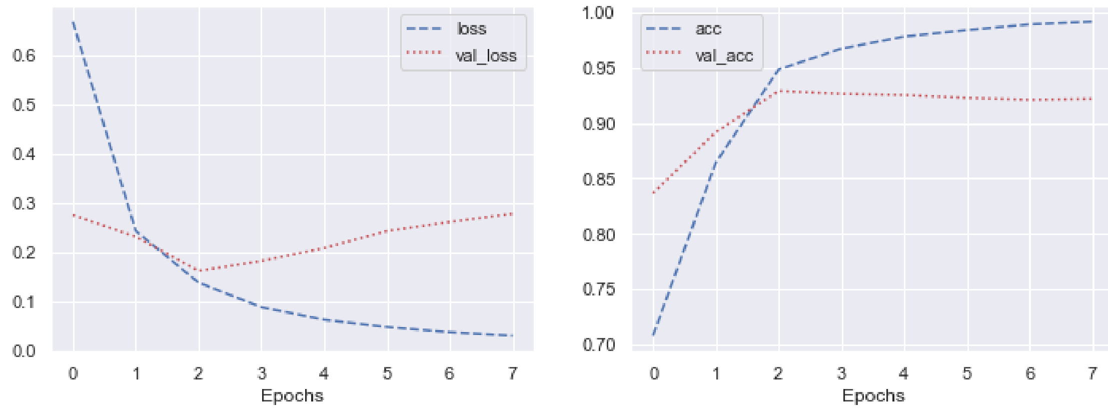
[121]:

```
# Evaluate model on the test set
loss, accuracy = model.evaluate(X_tst, y_tst, verbose=0)

# Print metrics
print('Accuracy  : {:.4f}'.format(accuracy))
```

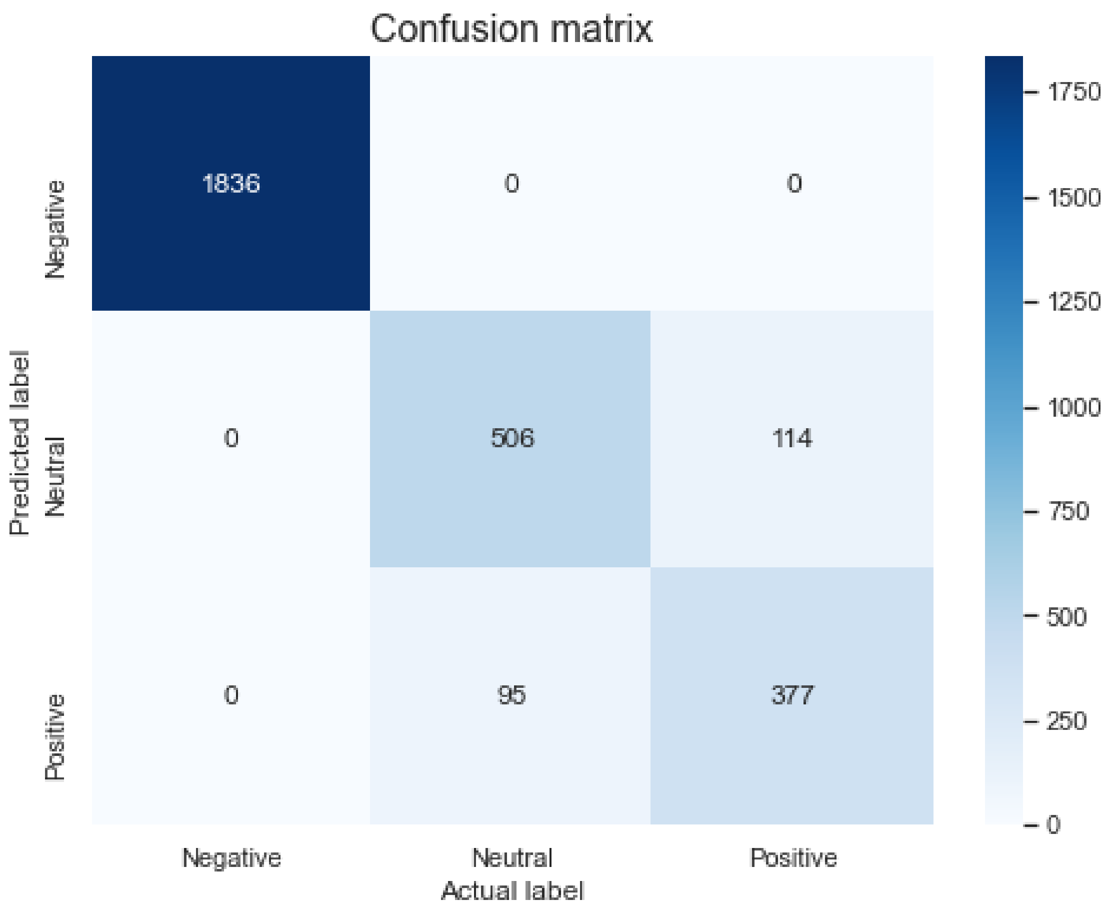
```
Accuracy  : 0.9286
```

[122]:

```
# Visualizing loss and accuracy on sequential model.
plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], 'b--', label = 'loss')
plt.plot(history.history['val_loss'], 'r:', label = 'val_loss')
plt.xlabel('Epochs')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], 'b--', label = 'acc')
plt.plot(history.history['val_accuracy'], 'r:', label = 'val_acc')
plt.xlabel('Epochs')
plt.legend()

plt.show()
```



[177]:
```
# Looking last model confusion matrix.
plot_confusion_matrix(model, X_tst, y_tst)
```



0.10 Conclusion

With final model, prediction accuracy on test set is %92. What this mean is with the **customer review text** my model will predict 92/100 true positive, true neutral or true negative. 8 of 100 tweets is going to be false for true positive, neutral or negatives(as example: 'liked' tweet at prediction section predicted negative which is false negative).

Used accuracy score to evaluation metric because target variables(sentiment) positive,negative and neutral classes imbalanced and all of them equal important for us. We are not focusing just one of them and wanted to accuracy of each of them.

0.11 Future Work

We can work on detail text and improve our model. For example we still get little bit of false positive,neutral or negatives. We would work why is that and what we could do more on our model.

Why liked tweet predicted negative etc.