AHMET16 / **U.S.--Airline-Sentiment_project** Public

Code Issues Pull requests Actions Projects Wiki Security

main

U.S.--Airline-Sentiment_project / U.S.-Airline-Sentiment-Project.ipynb

AHMET16 Add files via upload History

1 contributor

5512 lines (5512 sloc) | 759 KB

In [107…

```
# 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
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier
from xgboost import XGBClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB, BernoulliNB
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix,classification_report,
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE

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.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences

from sklearn.svm import SVC
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import accuracy_score
from tensorflow.keras import layers, models
import gensim
from gensim.models import Word2Vec



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] Downloading package stopwords to
[nltk_data]     /Users/karaoglan/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /Users/karaoglan/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/karaoglan/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package words to /Users/karaoglan/nltk_data...
[nltk_data]   Package words is already up-to-date!
```

Business Value

There are six different airline companies in this dataset; United, US Airways, American, Southwest, Delta and Virgin America. And their customers still complaining about some problems with their services/flights. For an airline company one customer, customer's review, one cancellation flight, one hour or one minute sometimes seconds too much important for a business value. Because of business reputation and business economic status. Instead of other industries economic status is more important at airline industry because this is a transportation company and losing every second for every mile flight. We are going to analyze and making machine learning project for how airline companies could improve ourselves with our findings.

Business Problem

In this project, main goal is the predict airline sentiment of flights with machine learning model. Our problem is customers satisfaction of flights.Some customers not only half satisfied, almost completely not satisfied and have some problems like; customer service issue, late flight, cancellation of flight etc. This problems will make specific airline company to lose money.Since every seconds important for an airline company, we are going to analyze why is that and making machine learning model to prevent at the future. Depend on customer's review(positive , neutral or negative) airline companies could take action about it.

In [108…

```
# Import and looking the data.
df = pd.read_csv('Tweets.csv')
df.head()
```

Out[108…

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason
0	570306133677760513	neutral	1.0000	NaN
1	570301130888122368	positive	0.3486	NaN

2	570301083672813571	neutral	0.6837	NaN
3	570301031407624196	negative	1.0000	Bad Flight
4	570300817074462722	negative	1.0000	Can't Tell

In [109…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14640 entries, 0 to 14639
Data columns (total 15 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   tweet_id                      14640 non-null  int64  
 1   airline_sentiment             14640 non-null  object 
 2   airline_sentiment_confidence  14640 non-null  float64
 3   negativereason                9178 non-null   object 
 4   negativereason_confidence     10522 non-null  float64
 5   airline                       14640 non-null  object 
 6   airline_sentiment_gold        40 non-null     object 
 7   name                          14640 non-null  object 
 8   negativereason_gold           32 non-null     object 
 9   retweet_count                 14640 non-null  int64  
 10  text                          14640 non-null  object 
 11  tweet_coord                   1019 non-null   object 
 12  tweet_created                 14640 non-null  object 
 13  tweet_location                9907 non-null   object 
 14  user_timezone                 9820 non-null   object 
dtypes: float64(2), int64(2), object(11)
memory usage: 1.7+ MB
```

In [110…

```
df['tweet_created']
```

Out[110…

```
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
```

In [111…

```
df['airline_sentiment'].value_counts()
```

Out[111…

```
negative    9178
neutral     3099
positive    2363
```

```
Name: airline_sentiment, dtype: int64
```

In [112…

```
df['airline_sentiment'].value_counts(normalize=True)
```

Out[112…

```
negative    0.626913
neutral     0.211680
positive    0.161407
Name: airline_sentiment, dtype: float64
```

In [113…

```
df['airline'].value_counts()
```

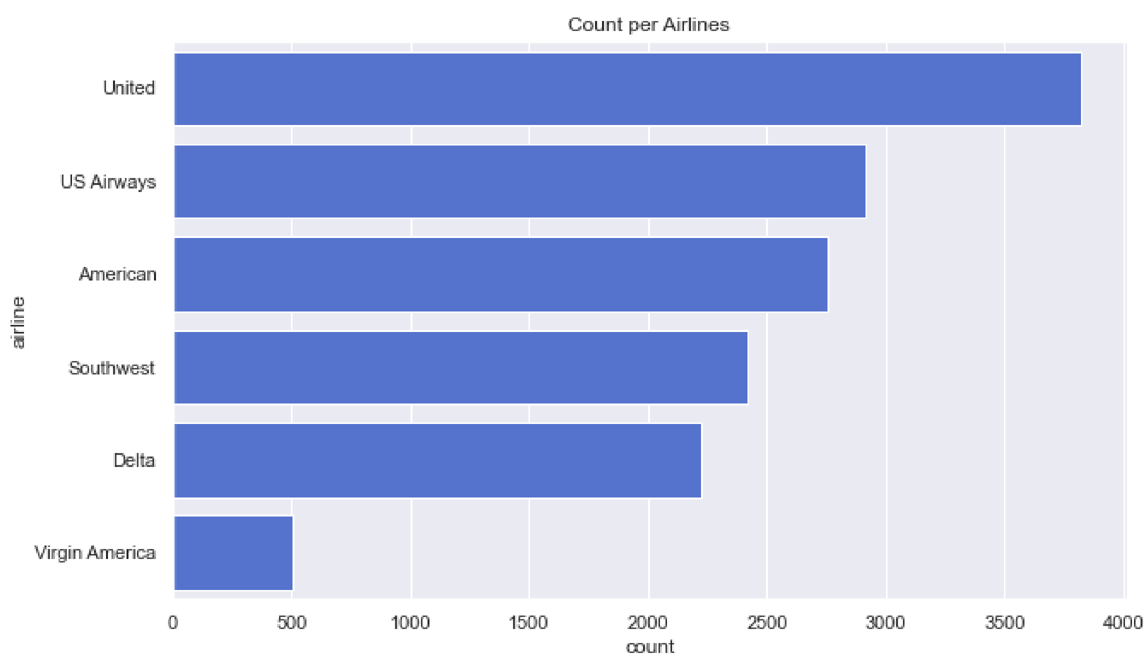
Out[113…

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

Data Understanding

In [114…

```
# Visual of airline companies review counts.
ax = sns.countplot(data = df, y= 'airline', color= 'royalblue',
                  order = df.airline.value_counts().index)
ax.set_title('Count per Airlines')
plt.show()
from matplotlib import pyplot as plt

plt.savefig('Count_per_airlines.png')
```

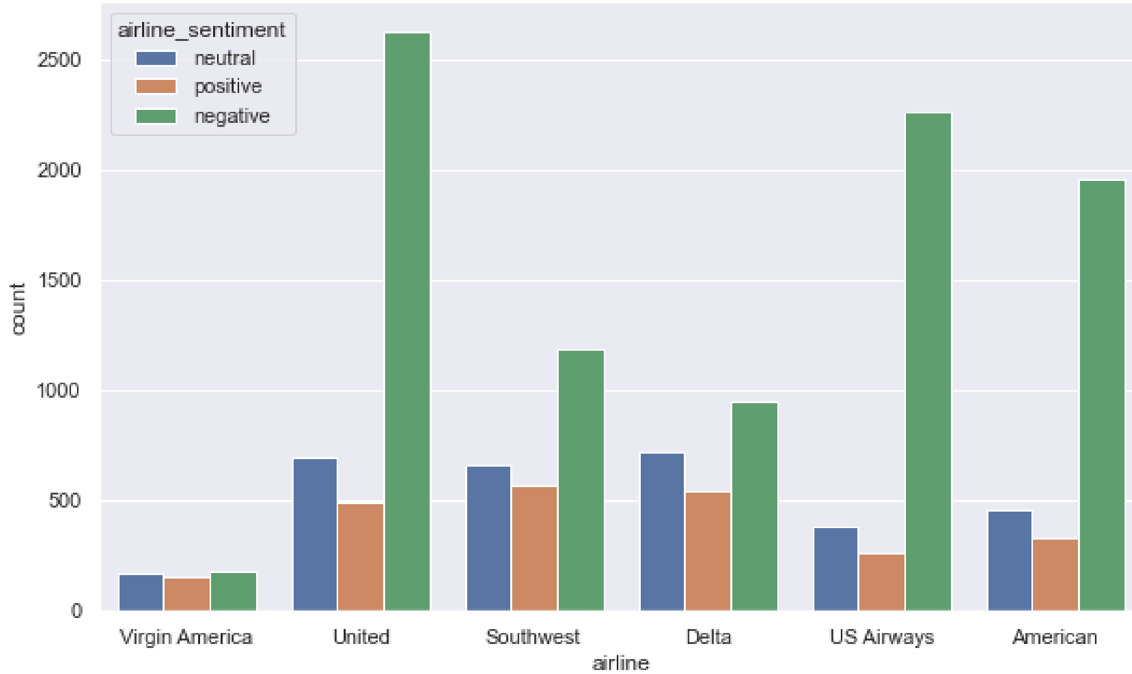



```
<Figure size 720x432 with 0 Axes>
```

In [115…

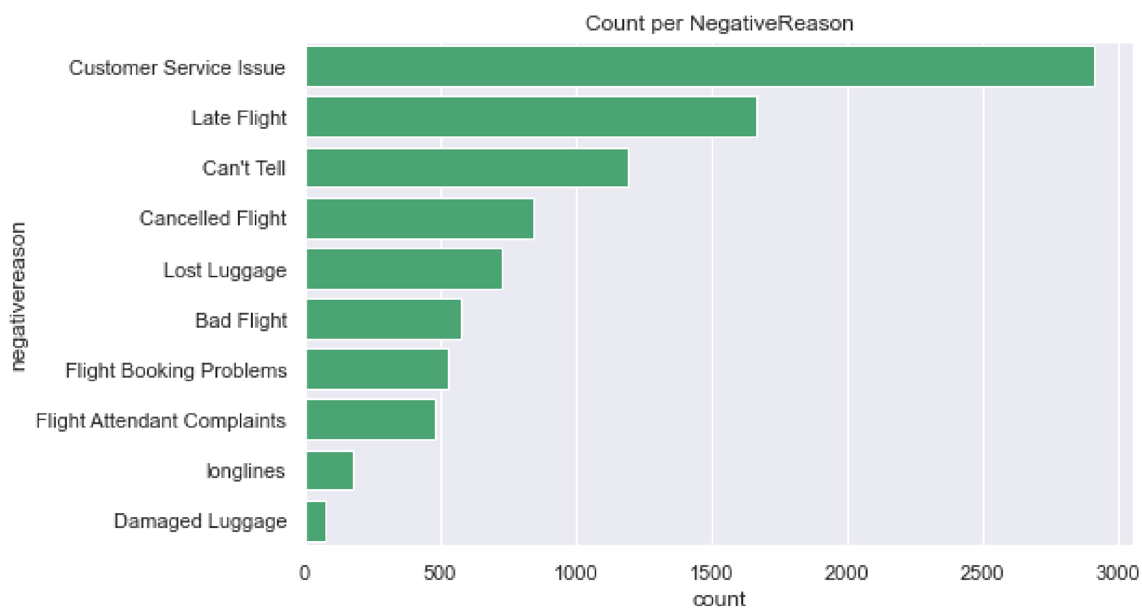
```
#Airline companies sentiment visualization.

sns.countplot(data = df, x = 'airline', hue = "airline_sentiment");
```

```
sns.set(rc={"figure.figsize":(12,6)})

plt.savefig('sentiment_per_airline_companies.png')
```



In [116…

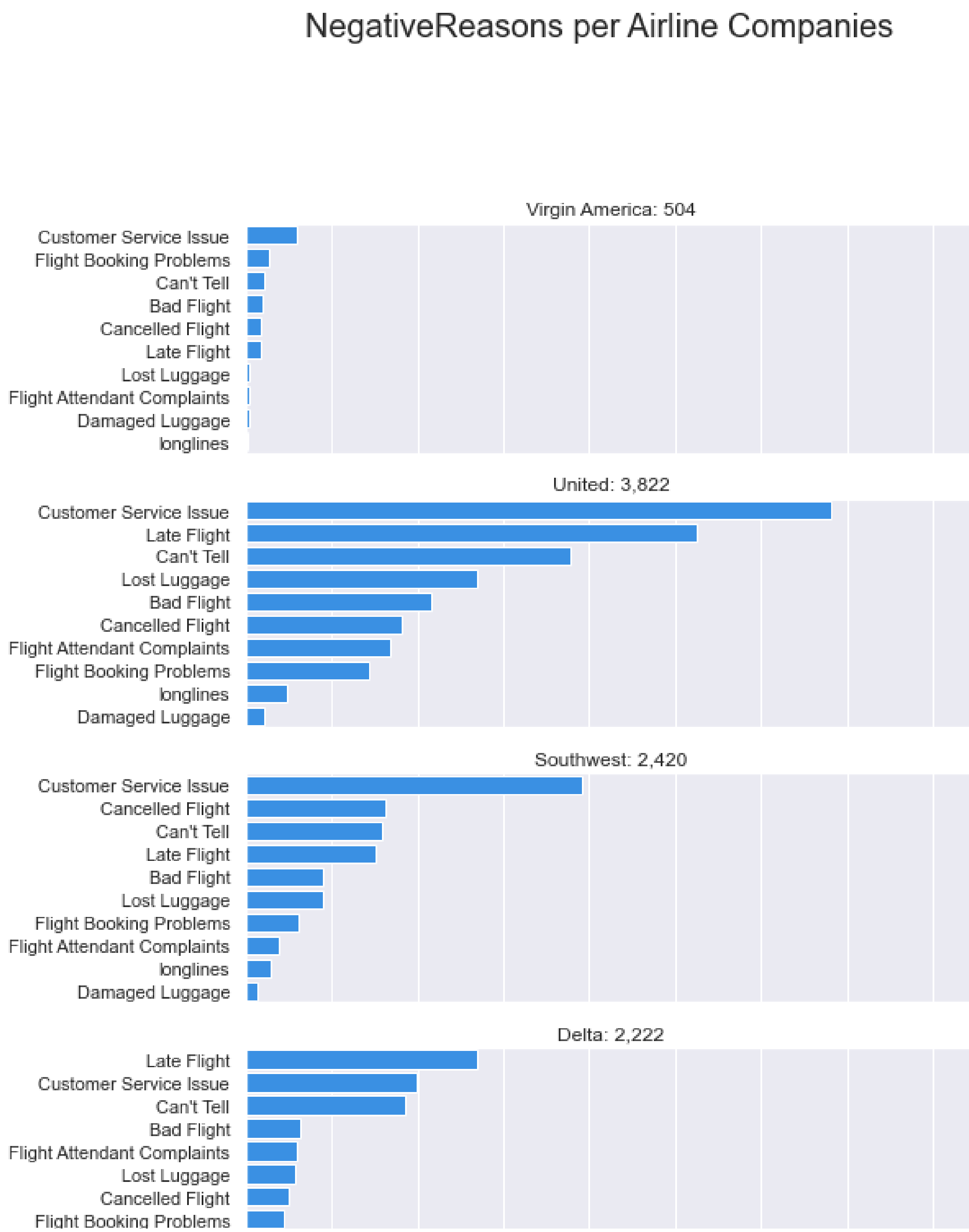
```
# Total negative reasons visual.
plt.figure(figsize=(8,5))
ax = sns.countplot(data = df, y = 'negativereason',
                   color='mediumseagreen',
                   order = df.negativereason.value_counts().index)
ax.set_title('Count per NegativeReason')
plt.savefig('negative_reasons.png')

plt.show()
```

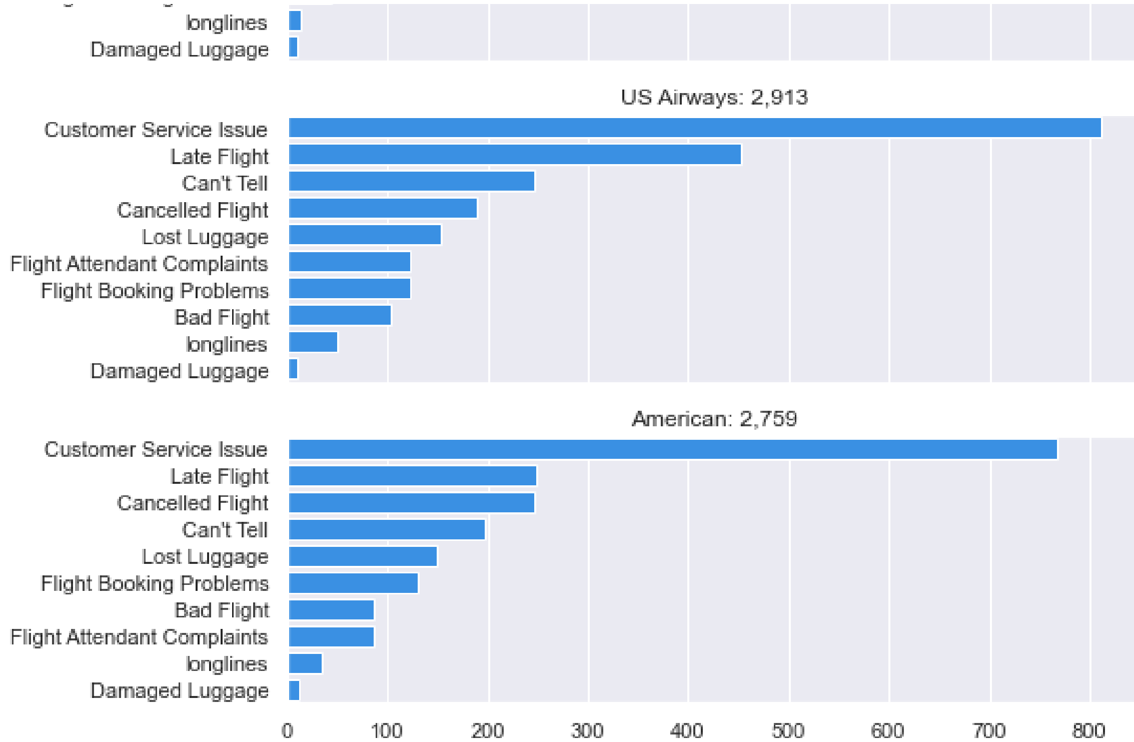


In [117…

```
# Negative reasons per airline companies.
fig, axes = plt.subplots(6,1, figsize=(8,18), sharex=True)
```

```
axes = axes.flatten()
names = df['airline'].unique()

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

plt.suptitle("NegativeReasons per Airline Companies", fontsize = 20)
plt.show()
from matplotlib import pyplot as plt

plt.savefig('Negative Reasons per Airline Companies.png')
```






```
<Figure size 864x432 with 0 Axes>
```

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.

Cleaning

In [118…

```python
# Copying data for secure original.
df2 = df.copy()
```

In [119…

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

Out[119…

```
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
```

In [120…

```
# Creating variable for english stopwords.
stop_words = stopwords.words('english')
```

In [121…

```
# Creating function for cleaning, tokenize and lemmatization.
def cleaning(data):
    """ This function cleans each word from punctuations, lowers each cha
    lemmatization for each word."""

    #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_

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

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

In [122…

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

Out[122…

```
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
```

In [123…

```
" ".join(df2["text"]).split()
```

Out[123…

```
['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'
```

```
'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'
```

```
 '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',
```

```
 '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'
```

```
'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',
```

```
 'use',
 'another',
 'browser',
 'amp',
 'brand',
 'reputation',
 'built',
 'tech',
 'response',
 'doesnt',
 'compatible',
 'website',
 'flight',
 'flight',
 'booking',
 'problem',
 ...]
```

In [124…

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

Out[124…

	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

In [125…

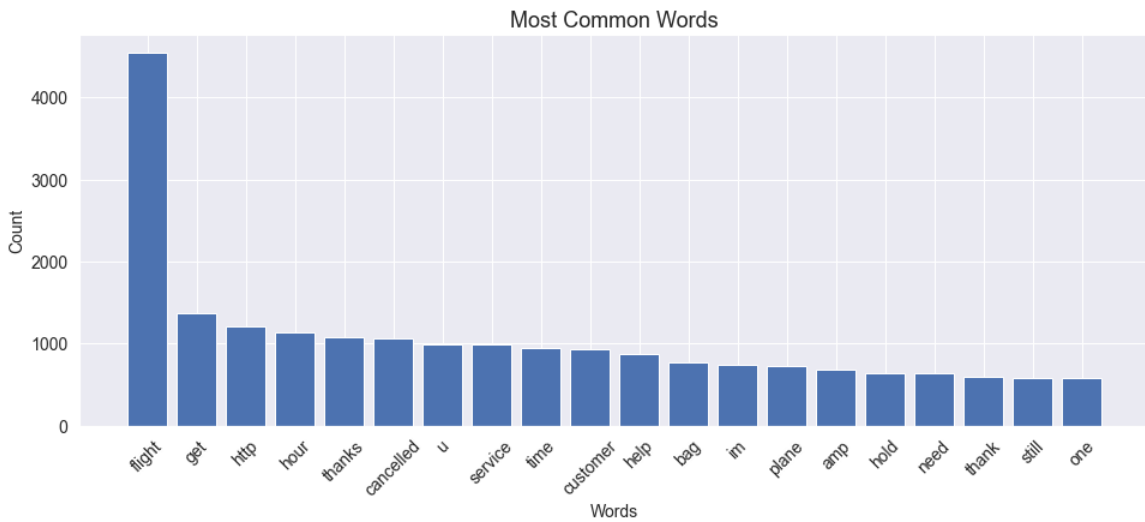
```python
# Counting most common words.
corpus = " ".join(df2["text"])
tokens_count = Counter(word_tokenize(corpus)).most_common(20)
tokens_count
```

Out[125…

```
[('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)]
```

In [126…

```
# 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()
```



In [127…

```
# from sklearn import preprocessing

# # label_encoder object knows how to understand word labels.
# label_encoder = preprocessing.LabelEncoder()

# # Encode labels in column 'species'.
# df['airline_sentiment']= label_encoder.fit_transform(df['airline_sentir

# df['airline_sentiment'].value_counts()
```

Train Test Split

In [128…

```
# Train test split
X = df2["text"]
y = df2["airline_sentiment"]
```

In [129…

```
y.value_counts()
```

Out[129…

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

In [130…

```
tfid = TfidfVectorizer()
X_final = tfid.fit_transform(X)
```

In [131…
```
# Handling imbalanced using SMOTE
smote = SMOTE()
X_sm, y_sm = smote.fit_resample(X_final,y)
```

In [132…
```
X_train, X_test, y_train,y_test = train_test_split(X_sm,y_sm,test_size=0.
```

Count Vectorizer

In [133…
```
# Initializing Count Vectorizer.
c_vec = CountVectorizer()
X_final1= c_vec.fit_transform(X)
```

In [134…
```
# Looking train set into array.
X_final1.toarray()
```

Out[134…
```
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, 0, 0, ..., 0, 0, 0]])
```

In [135…
```
# Look dataframe after process.
pd.DataFrame(X_final1.toarray(), columns = c_vec.get_feature_names())
```

Out[135…

	aa	aaaand	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	(
...	...	...	...	...	...	...	...	..
14635	0	0	0	0	0	0	0	(
14636	0	0	0	0	0	0	0	(
14637	0	0	0	0	0	0	0	(
14638	0	0	0	0	0	0	0	(
14639	0	0	0	0	0	0	0	(

14640 rows × 9861 columns

In [136…
```
#Creating function to evaluate our models.
```

```
def evaluation(model, X_train, X_test):

    """ This function created for visualization and resul to see train ar

    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)
    plt.grid(None)
```

Logistic Regression

In [137…

```
# Initiliazing first model.
log = LogisticRegression(C = 0.02, max_iter=1000)
log.fit(X_train,y_train)
```

Out[137…

```
LogisticRegression(C=0.02, max_iter=1000)
```

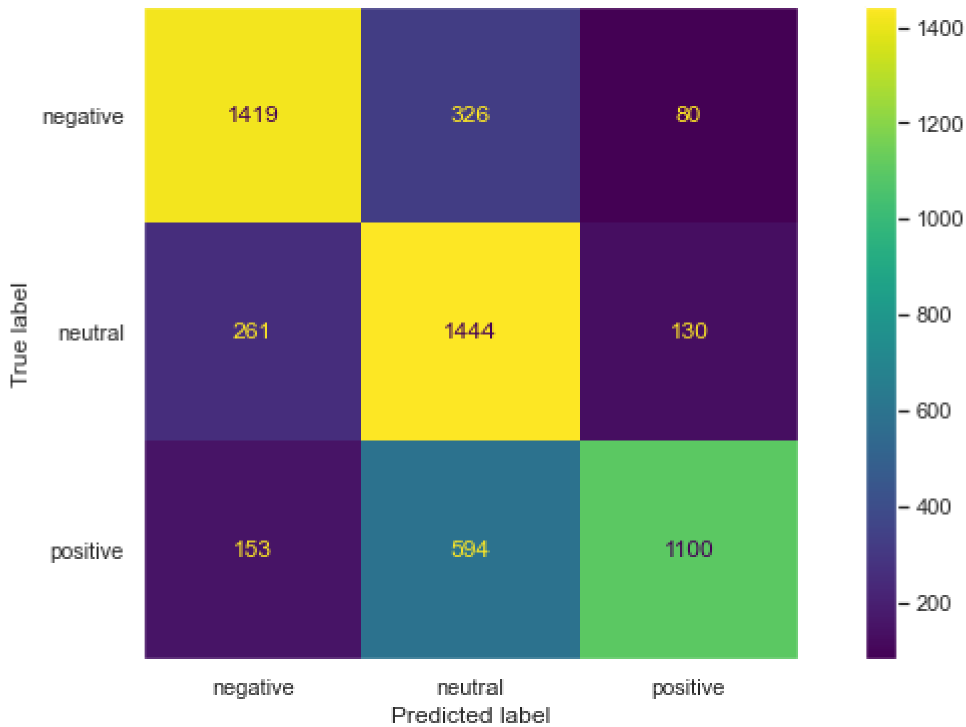
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [138…

```
print("Log Model")
evaluation(log, X_train, X_test)
```

```
Log Model
==== Train Set ====
              precision    recall  f1-score   support

    negative       0.78      0.79      0.78      7353
     neutral       0.62      0.79      0.69      7343
    positive       0.84      0.60      0.70      7331

    accuracy                           0.72     22027
   macro avg       0.75      0.72      0.72     22027
weighted avg       0.75      0.72      0.72     22027

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

    negative       0.77      0.78      0.78      1825
     neutral       0.61      0.79      0.69      1835
    positive       0.84      0.60      0.70      1847

    accuracy                           0.72      5507
   macro avg       0.74      0.72      0.72      5507
```

```
   macro avg       0.74      0.72      0.72      5507
weighted avg       0.74      0.72      0.72      5507
```



Naive Bayes

In [139…

```
#Initiliazing second model.
nb = MultinomialNB()
nb.fit(X_train,y_train)
```

Out[139…

```
MultinomialNB()
```

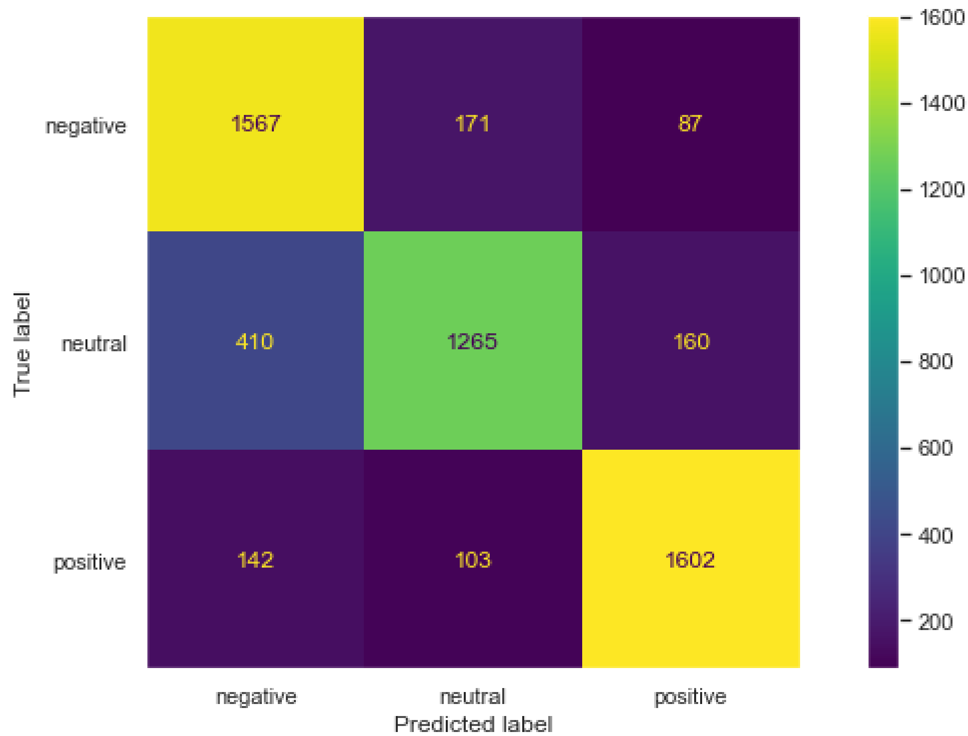
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [140…

```
print("NB Model")
evaluation(nb, X_train, X_test)
```

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

    negative       0.78      0.92      0.84      7353
     neutral       0.88      0.73      0.80      7343
    positive       0.89      0.89      0.89      7331

    accuracy                           0.85     22027
   macro avg       0.85      0.85      0.84     22027
weighted avg       0.85      0.85      0.84     22027

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

```
              precision    recall  f1-score   support

    negative       0.74      0.86      0.79      1825
     neutral       0.82      0.69      0.75      1835
    positive       0.87      0.87      0.87      1847

    accuracy                           0.81      5507
   macro avg       0.81      0.81      0.80      5507
weighted avg       0.81      0.81      0.80      5507
```



Ada Boost

In [141…

```
#Initiliazing third model.
ada = AdaBoostClassifier(n_estimators=500,random_state=42)
ada.fit(X_train,y_train)
```

Out[141…

```
AdaBoostClassifier(n_estimators=500, random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

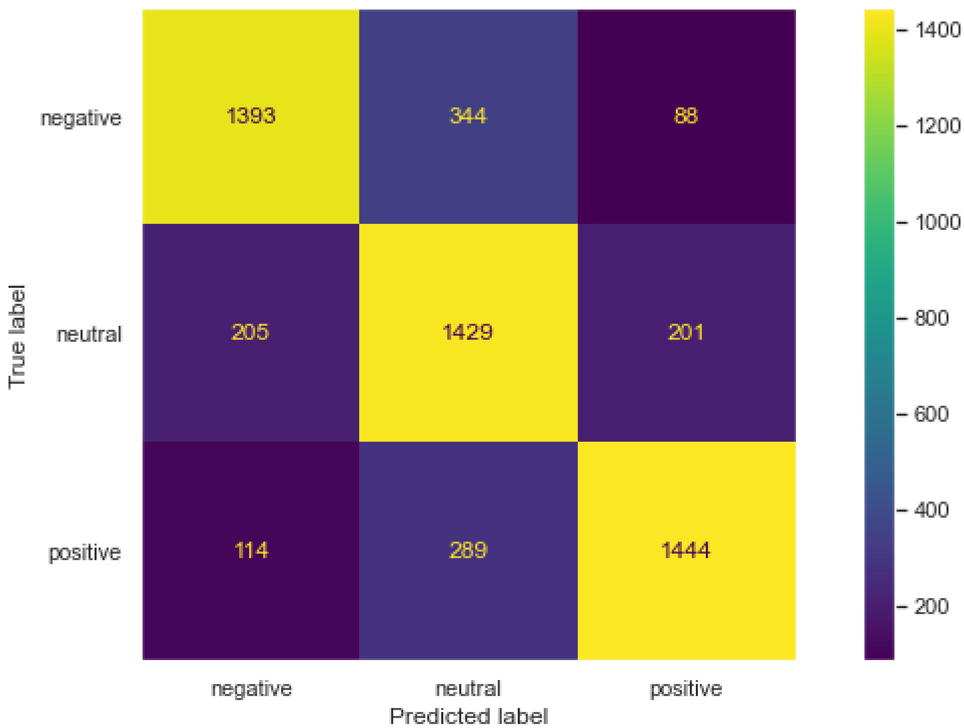
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [142…

```
print("Ada MODEL")
evaluation(ada, X_train, X_test)
```

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

    negative       0.85      0.81      0.83      7353
     neutral       0.73      0.79      0.76      7343
    positive       0.86      0.83      0.84      7331
```

```
    accuracy                           0.81     22027
   macro avg       0.81      0.81      0.81     22027
weighted avg       0.81      0.81      0.81     22027

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

    negative       0.81      0.76      0.79      1825
     neutral       0.69      0.78      0.73      1835
    positive       0.83      0.78      0.81      1847

    accuracy                           0.77      5507
   macro avg       0.78      0.77      0.78      5507
weighted avg       0.78      0.77      0.78      5507
```



TF-IDF

In [143…

```python
# Looking train set into array.
X_final.toarray()
```

Out[143…

```
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., 0., 0., ..., 0., 0., 0.]])
```

In [144…

```python
# Look dataframe after process.
pd.DataFrame(X_final.toarray(),columns = tfid.get_feature_names())
```

Out[144… aa aaaand aaadvantage aaalwayslate aadavantage aadelay aady aadvantag

Out[144…

	aa	aaaand	aaadvantage	aaalwayslate	aadavantage	aadelay	aadv	aadvantag
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.
2	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.
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
...	...	...	...	...	...	...	...	.
14635	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
14636	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
14637	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
14638	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
14639	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.

14640 rows × 9861 columns

Naive Bayes

In [145…

```
#Initiliazing first model.
nb = MultinomialNB()
nb.fit(X_train,y_train)
```

Out[145…

```
MultinomialNB()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [146…

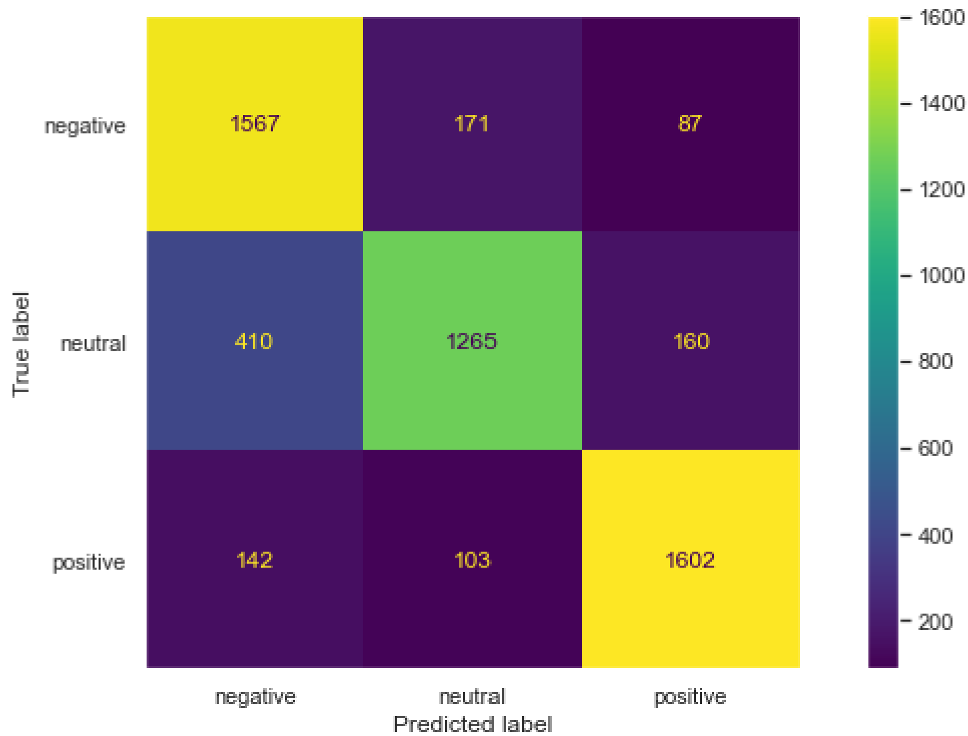
```
# why we did two times there is no difference
```

In [147…

```
print("NB MODEL")
evaluation(nb, X_train, X_test)
```

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

    negative       0.78      0.92      0.84      7353
     neutral       0.88      0.73      0.80      7343
    positive       0.89      0.89      0.89      7331

    accuracy                           0.85     22027
   macro avg       0.85      0.85      0.84     22027
weighted avg       0.85      0.85      0.84     22027

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

```
    negative       0.74      0.86      0.79      1825
     neutral       0.82      0.69      0.75      1835
    positive       0.87      0.87      0.87      1847

    accuracy                           0.81      5507
   macro avg       0.81      0.81      0.80      5507
weighted avg       0.81      0.81      0.80      5507
```



Logistic Regression

In [148…

```
#Initiliazing second model.
log = LogisticRegression(C=0.4, max_iter=1000)
log.fit(X_train,y_train)
```

Out[148… LogisticRegression(C=0.4, max_iter=1000)

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

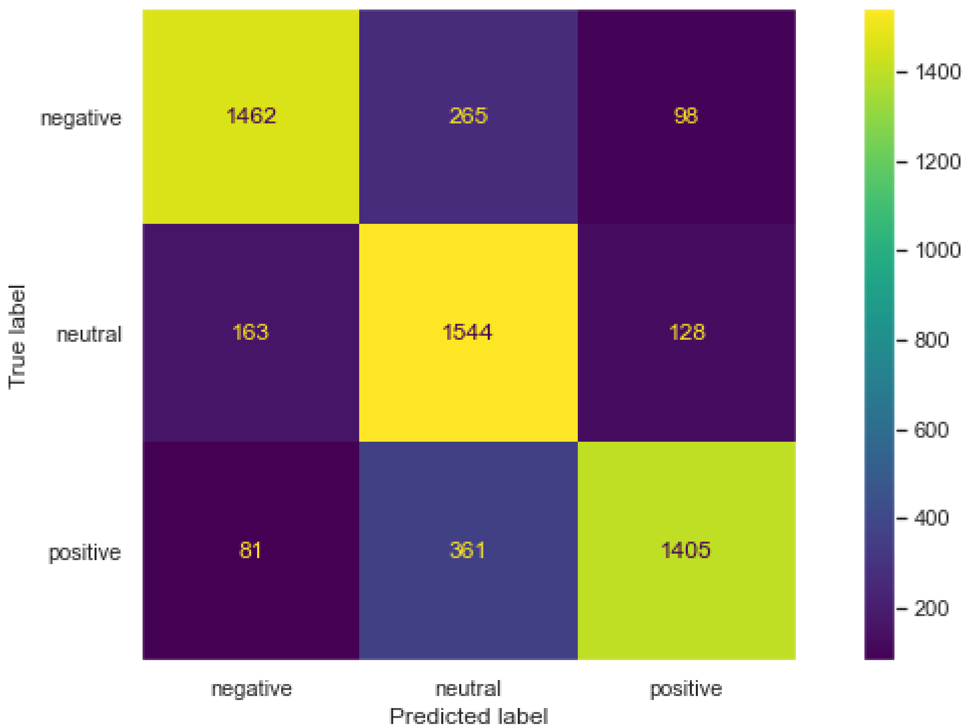
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [149…

```
print("LOG MODEL")
evaluation(log , X_train, X_test)
```

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

    negative       0.88      0.85      0.87      7353
     neutral       0.75      0.86      0.80      7343
    positive       0.89      0.78      0.83      7331
```

```
    accuracy                           0.83     22027
   macro avg       0.84      0.83      0.83     22027
weighted avg       0.84      0.83      0.83     22027

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

    negative       0.86      0.80      0.83      1825
     neutral       0.71      0.84      0.77      1835
    positive       0.86      0.76      0.81      1847

    accuracy                           0.80      5507
   macro avg       0.81      0.80      0.80      5507
weighted avg       0.81      0.80      0.80      5507
```



Random Forest

In [150…

```
#Initiliazing third model
rf = RandomForestClassifier(100, max_depth=40, random_state=42,n_jobs=-1)
rf.fit(X_train,y_train)
```

Out[150…

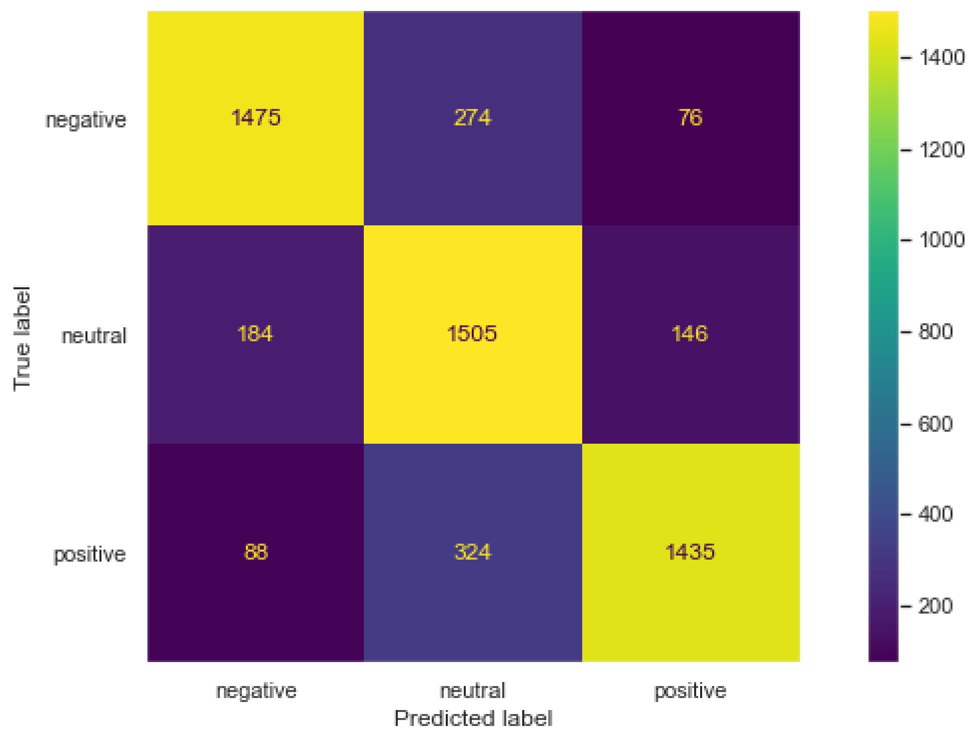
```
RandomForestClassifier(max_depth=40, n_jobs=-1, random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [151…

```
print("RF MODEL")
evaluation(rf, X_train, X_test)
```

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

    negative       0.95      0.86      0.90      7353
     neutral       0.76      0.92      0.83      7343
    positive       0.92      0.82      0.87      7331

    accuracy                           0.86     22027
   macro avg       0.88      0.86      0.87     22027
weighted avg       0.88      0.86      0.87     22027

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

    negative       0.84      0.81      0.83      1825
     neutral       0.72      0.82      0.76      1835
    positive       0.87      0.78      0.82      1847

    accuracy                           0.80      5507
   macro avg       0.81      0.80      0.80      5507
weighted avg       0.81      0.80      0.80      5507
```



Gradient Boosting

In [152…

```
#Initiliazing fourt model
gb =GradientBoostingClassifier()
gb.fit(X_train,y_train)
```

Out[152… GradientBoostingClassifier()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

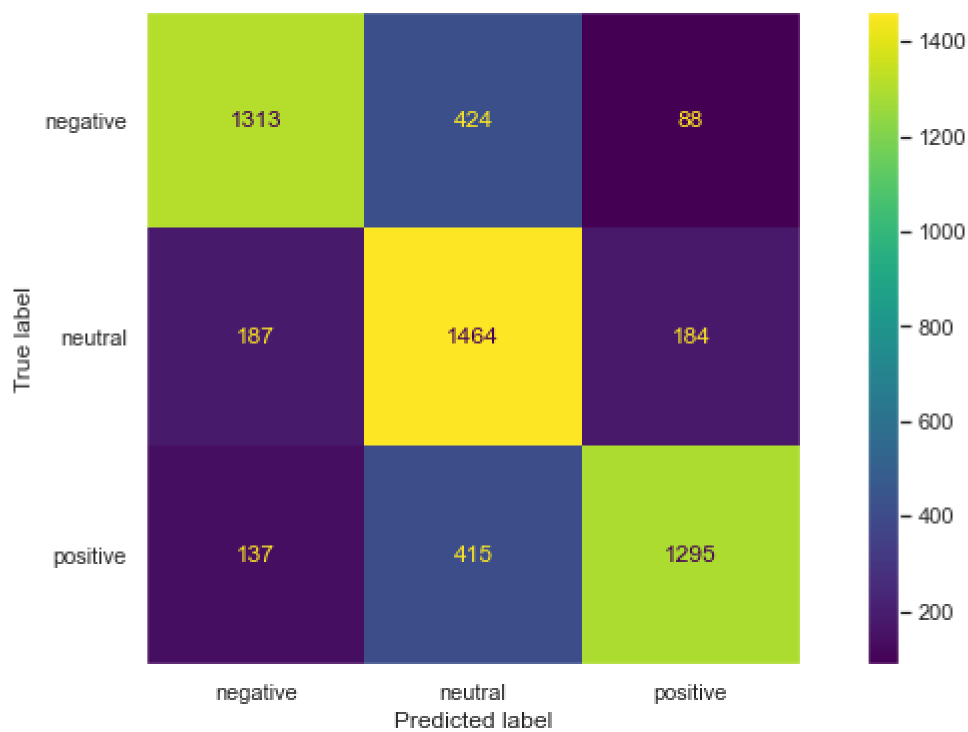
On GitHub, the HTML representation is unable to render, please try loading this

page with nbviewer.org.

In [153…

```
print("GB MODEL")
evaluation(gb, X_train,X_test)
```

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

    negative       0.82      0.73      0.77      7353
     neutral       0.64      0.81      0.72      7343
    positive       0.84      0.71      0.77      7331

    accuracy                           0.75     22027
   macro avg       0.77      0.75      0.75     22027
weighted avg       0.77      0.75      0.75     22027

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

    negative       0.80      0.72      0.76      1825
     neutral       0.64      0.80      0.71      1835
    positive       0.83      0.70      0.76      1847

    accuracy                           0.74      5507
   macro avg       0.75      0.74      0.74      5507
weighted avg       0.75      0.74      0.74      5507
```



Ada Boost

In [154…

```
#Initiliazing fift model
ada =AdaBoostClassifier(n_estimators=500, random_state=42)
```

```
ada.fit(X_train,y_train)
```

Out [154…

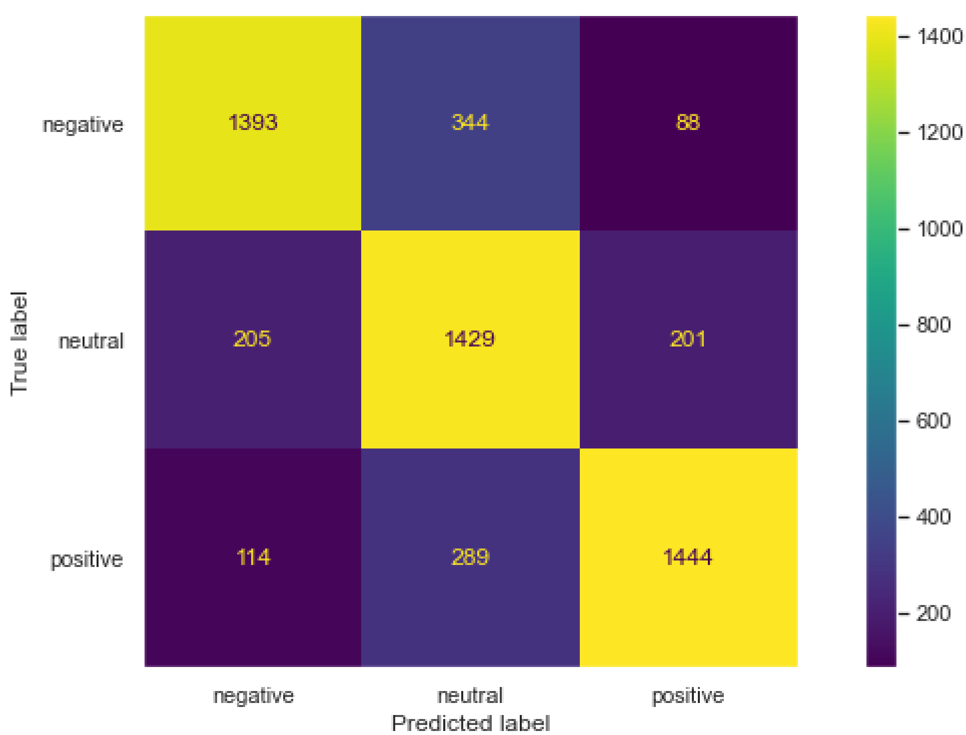
```
AdaBoostClassifier(n_estimators=500, random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [155…

```
print("Ada Model")
evaluation(ada, X_train, X_test)
```

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

    negative       0.85      0.81      0.83      7353
     neutral       0.73      0.79      0.76      7343
    positive       0.86      0.83      0.84      7331

    accuracy                           0.81     22027
   macro avg       0.81      0.81      0.81     22027
weighted avg       0.81      0.81      0.81     22027

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

    negative       0.81      0.76      0.79      1825
     neutral       0.69      0.78      0.73      1835
    positive       0.83      0.78      0.81      1847

    accuracy                           0.77      5507
   macro avg       0.78      0.77      0.78      5507
weighted avg       0.78      0.77      0.78      5507
```



Prediction

In [156…

```
pipe = Pipeline([('tfidf',TfidfVectorizer()),('log',LogisticRegression(C=
```

In [157…

```
pipe.fit(X,y)
```

Out[157…

```
Pipeline(steps=[('tfidf', TfidfVectorizer()),
                ('log', LogisticRegression(C=0.4, max_iter=100
0))])
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [158…

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

Out[158…

```
array(['negative'], dtype=object)
```

In [159…

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

Out[159…

```
array(['negative'], dtype=object)
```

In [160…

```
#example prediction
tweet = "doesn't enjoy flight"
tweet =pd.Series(tweet).apply(cleaning)
pipe.predict(tweet)
```

Out[160…

```
array(['negative'], dtype=object)
```

In [161…

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

Out[161…

```
array(['neutral'], dtype=object)
```

In [162…

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

```
array(['negative'], dtype=object)
```

Out[162… array(['negative'], dtype=object)

In [163…
```
#Example prediction("Wrong prediction by model")
tweet = "liked"
tweet = pd.Series(tweet).apply(cleaning)
pipe.predict(tweet)
```

Out[163… array(['negative'], dtype=object)

Sequential

In [164…
```
# Remembering data
df2
```

Out[164…

	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 × 2 columns

In [165…
```
#Creating target and feature
target = df2["airline_sentiment"]
data = df2['text'].map(word_tokenize).values
```

In [166…
```
# Creating function to tokenize
def tokenize(d):
    return word_tokenize(d)
```

In [167…
```
#Creating variable for tokenized target variable
texts_w2v = df2.text.apply(tokenize).to_list()
```

Word2Vec Model

In [168…

```
# Initialing Word2Vec Model
w2v = Word2Vec(sentences = texts_w2v, window=3,
               vector_size=100, min_count=5, workers=4,sg = 1)
```

In [169…

```
texts_w2v[:5]
```

Out[169…

```
[['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']]
```

In [170…

```
## Similar words with the given word examples
w2v.wv.most_similar('thank')
```

Out[170…

```
[('much', 0.9601995348930359),
 ('quick', 0.9563593864440918),
 ('appreciate', 0.9464036822319031),
 ('tweet', 0.93934166431427),
 ('awesome', 0.9339163303375244),
 ('thanks', 0.9337353706359863),
 ('twitter', 0.9312583804130554),
 ('reply', 0.9298743605613708),
 ('sending', 0.9220132231712341),
 ('detail', 0.9219493269920349)]
```

In [171…

```
#Looking for simillar word with given words.
w2v.wv.most_similar('customerservice')
```

Out[171…

```
[('nightmare', 0.992286205291748),
 ('loved', 0.9903038144111633),
 ('neveragain', 0.9894539713859558),
 ('hanging', 0.9892846941947937),
 ('horrendous', 0.9887707829475403),
 ('biggest', 0.9886414408683777),
 ('learned', 0.9884180426597595),
 ('heard', 0.9883833527565002),
 ('literally', 0.9882695078849792),
 ('abysmal', 0.9879590272903442)]
```

In [172…

```
#Looking for similar word with given words.
w2v.wv.most_similar("crew")
```

Out[172…

```
[('pilot', 0.8965240120887756),
 ('ground', 0.8796437978744507),
 ('attendant', 0.876422107219696),
 ('landing', 0.8552125692367554),
```

```
 ('made', 0.8435713052749634),
 ('air', 0.8376836776733398),
 ('san', 0.8354215025901794),
 ('staff', 0.8324906229972839),
 ('plane', 0.8320526480674744),
 ('ord', 0.8291676640510559)]
```

In [173…

```
#Looking for similar word with given words
w2v.wv.most_similar("delay")
```

Out[173…

```
[('delayed', 0.9477306008338928),
 ('sfo', 0.9112058281898499),
 ('maintenance', 0.9076501131057739),
 ('due', 0.9066833257675171),
 ('stuck', 0.9046614766120911),
 ('mechanical', 0.8995684385299683),
 ('landing', 0.8988839387893677),
 ('phx', 0.8937194347381592),
 ('ewr', 0.8891671299934387),
 ('lax', 0.8885225057601929)]
```

In [174…

```
w2v.wv.most_similar("ticket")
```

Out[174…

```
[('fee', 0.9199082851409912),
 ('award', 0.9181016683578491),
 ('name', 0.912956714630127),
 ('refund', 0.8986184597015381),
 ('bought', 0.8929511904716492),
 ('booked', 0.8923091888427734),
 ('credit', 0.8852400779724121),
 ('add', 0.8833764791488647),
 ('mile', 0.8823671340942383),
 ('buy', 0.8819577097892761)]
```

In [175…

```
# Creating vectors for every text.
def get_avg_vector(sent):
    """
    This function makes vector for every sepcific words in our text data.
    """

    vector = np.zeros(100)
    total_words = 0
    for word in sent.split():
        if word in w2v.wv.index_to_key:
            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)
```

Out[175…

	text	w2v_vector
0	said	[-0.030167607590556145, 0.0658913180232048, -0...

1	plus youve added commercial experience tacky	[-0.0789375588297844, 0.11087281703948974, -0....

In [176…
```
df2['w2v_vector'].values[0].shape
```
Out[176…
```
(100,)
```
In [177…
```
# checking three diffent models accuracy for improve further.
model_params = {"random_state":42}
model_list = [LogisticRegression(**model_params,solver='liblinear'),
              RandomForestClassifier(**model_params),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.
        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.
```
```
Model: LogisticRegression, Word2Vec, Accuracy: 72.131%

Model: RandomForest, Word2Vec, Accuracy: 73.265%

Model: SupportVectorMachine, Word2Vec, Accuracy: 71.496%
```
In [178…
```
#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 ar
    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(df2['text'])
print('After Tokenization & Padding \n', X[0])
```

```
Before Tokenization & Padding
 said

After Tokenization & Padding
 [126   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   0]
```

In [179…

```
#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_
X_trn, X_vld, y_trn, y_vld = train_test_split(X_trn, y_trn, test_size=0.3

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)
```

Sequential Model

In [180…

```
#Creating necessary variables and initializing sequential model. Adding l
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='r
model.add(MaxPooling1D(pool_size=2, padding='same'))
model.add(Bidirectional(LSTM(32)))
model.add(Dropout(0.4))
model.add(Dense(3, activation='softmax'))
```

When to use a Sequential model A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor.

In [181…

```
#Compilling model. Looking into it.

model.compile(loss='categorical_crossentropy' , optimizer="adam",metrics=
print(model.summary())
```

```
Model: "sequential_2"
```

```
_________________________________________________________________
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_3 (Dense)              (None, 3)                 195
=================================================================
Total params: 179,939
Trainable params: 179,939
Non-trainable params: 0
_________________________________________________________________
None
```

In [182…

```python
#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_siz
```

```
Epoch 1/50
129/129 [==============================] - 7s 30ms/step - loss: 0.9421 - accuracy: 0.6307 - val_loss: 0.6840 - val_accuracy: 0.6964
Epoch 2/50
129/129 [==============================] - 3s 24ms/step - loss: 0.6322 - accuracy: 0.7391 - val_loss: 0.5598 - val_accuracy: 0.7783
Epoch 3/50
129/129 [==============================] - 3s 24ms/step - loss: 0.4436 - accuracy: 0.8292 - val_loss: 0.5549 - val_accuracy: 0.7775
Epoch 4/50
129/129 [==============================] - 3s 24ms/step - loss: 0.3255 - accuracy: 0.8823 - val_loss: 0.5952 - val_accuracy: 0.7775
Epoch 5/50
129/129 [==============================] - 3s 24ms/step - loss: 0.2593 - accuracy: 0.9036 - val_loss: 0.6985 - val_accuracy: 0.7689
Epoch 6/50
129/129 [==============================] - 3s 25ms/step - loss: 0.1902 - accuracy: 0.9366 - val_loss: 0.7262 - val_accuracy: 0.7638
Epoch 7/50
129/129 [==============================] - 3s 25ms/step - loss: 0.1552 - accuracy: 0.9455 - val_loss: 0.8082 - val_accuracy: 0.7550
Epoch 8/50
129/129 [==============================] - 3s 25ms/step - loss: 0.1321 - accuracy: 0.9572 - val_loss: 0.9463 - val_accuracy: 0.7575
```

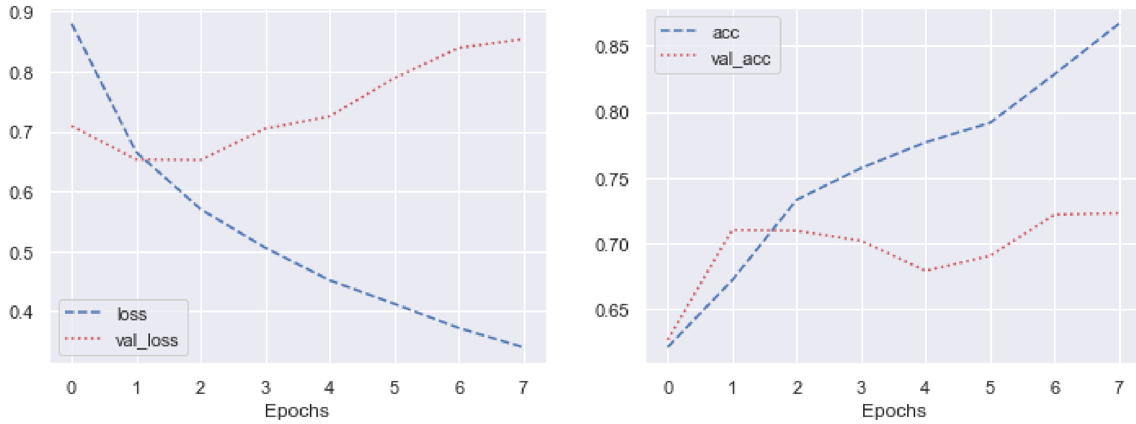
In [183…

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

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

```
Accuracy  : 0.7732
```

In [91]:

```
# 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.savefig('sequential.png')

plt.show()
```



In [185…

```
# 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 dat

    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
    # 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)
plt.savefig('confusion matrix.png')
```

Second Model

In [81]:

```
# Train test split.
X_trn, X_tst, y_trn, y_tst = train_test_split(X, y, test_size=0.2, randon
X_trn, X_vld, y_trn, y_vld = train_test_split(X_trn, y_trn, test_size=0.3

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)
```

In [82]:

```
# 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='r
model.add(MaxPooling1D(pool_size=2))
model.add(Bidirectional(LSTM(32)))
model.add(Dropout(0.4))
model.add(Dense(3, activation='relu'))
model.add(Dense(3, activation='softmax'))
```

In [83]:

```
#Compiling and looking to model.
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=
print(model.summary())
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
```

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 100, 32)           160000
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 100, 32)           3104
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 50, 32)            0
_________________________________________________________________
bidirectional_1 (Bidirection (None, 64)                16640
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 195
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 12
=================================================================
Total params: 179,951
Trainable params: 179,951
Non-trainable params: 0
_________________________________________________________________
None
```

In [84]:

```python
# 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 [==============================] - 8s 33ms/step - loss: 0.9619 -
accuracy: 0.6021 - val_loss: 0.7090 - val_accuracy: 0.6269
Epoch 2/50
129/129 [==============================] - 4s 30ms/step - loss: 0.6658 -
accuracy: 0.6591 - val_loss: 0.6532 - val_accuracy: 0.7103
Epoch 3/50
129/129 [==============================] - 3s 25ms/step - loss: 0.5667 -
accuracy: 0.7334 - val_loss: 0.6524 - val_accuracy: 0.7097
Epoch 4/50
129/129 [==============================] - 3s 26ms/step - loss: 0.5044 -
accuracy: 0.7570 - val_loss: 0.7044 - val_accuracy: 0.7020
Epoch 5/50
129/129 [==============================] - 3s 26ms/step - loss: 0.4550 -
accuracy: 0.7759 - val_loss: 0.7250 - val_accuracy: 0.6793
Epoch 6/50
129/129 [==============================] - 3s 26ms/step - loss: 0.4130 -
accuracy: 0.7896 - val_loss: 0.7887 - val_accuracy: 0.6907
Epoch 7/50
129/129 [==============================] - 3s 24ms/step - loss: 0.3774 -
accuracy: 0.8185 - val_loss: 0.8391 - val_accuracy: 0.7220
Epoch 8/50
129/129 [==============================] - 3s 25ms/step - loss: 0.3292 -
accuracy: 0.8718 - val_loss: 0.8539 - val_accuracy: 0.7231
```

In [85]:

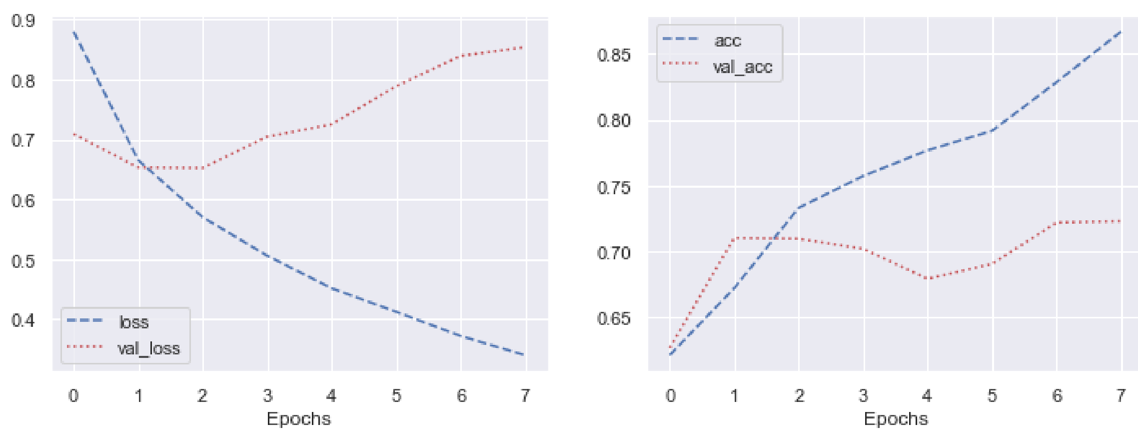
```python
# Evaluate model on the test set
loss, accuracy = model.evaluate(X_tst, y_tst, verbose=0)

# Print metrics
```

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

```
Accuracy  : 0.7432
```

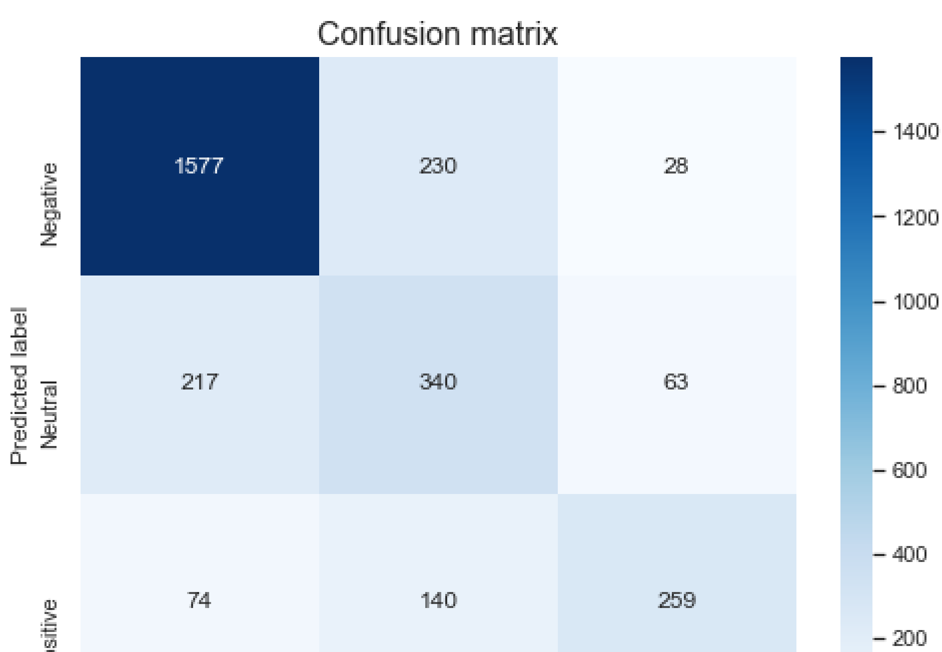
In [86]:

```
# 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()
```



In [87]:

```
# Looking last model confusion matrix.
plot_confusion_matrix(model, X_tst, y_tst)
```



Recommmendations

1.For all the 6 companies should work on customer issue problems.