```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd;
dfTweetOrig=pd.read_csv("/content/drive/MyDrive/YU-ML-Proj-1/Week8/Tweets.csv")
dfTweetOrig.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
```

```
dfTweetOrig.head()
```

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereas
0	570306133677760513	neutral	1.0000	Na
1	570301130888122368	positive	0.3486	Na
2	570301083672813571	neutral	0.6837	Na

Next steps: Generate code with dfTweetOrig View recommended plots

```
dfTweetOrig.shape
```

```
(14640, 15)
```

```
dfTweetSubset=dfTweetOrig[['tweet_id','airline_sentiment','text']]
dfTweetSubset['textLen']=dfTweetOrig['text'].str.len()
```

```
<ipython-input-6-32bdc4832229>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  dfTweetSubset['textLen']=dfTweetOrig['text'].str.len()
```

```
dfTweetSubset.shape
```

```
(14640, 4)
```

```
dfTweetSubset['textLen'].mean()
```

```
103.82206284153006
```

```
dfTweetSubset['textLen'].max()
```

```
186
```

```
dfTweetSubset['textLen'].min()
```

```
12
```

```
dfTweetSubset['textLen']
```

```
0         35
1         72
2         71
3        126
4         55
        ...
14635     63
14636    150
14637     60
14638    135
14639    138
Name: textLen, Length: 14640, dtype: int64
```

% of Tweets based on airline_sentiment Category

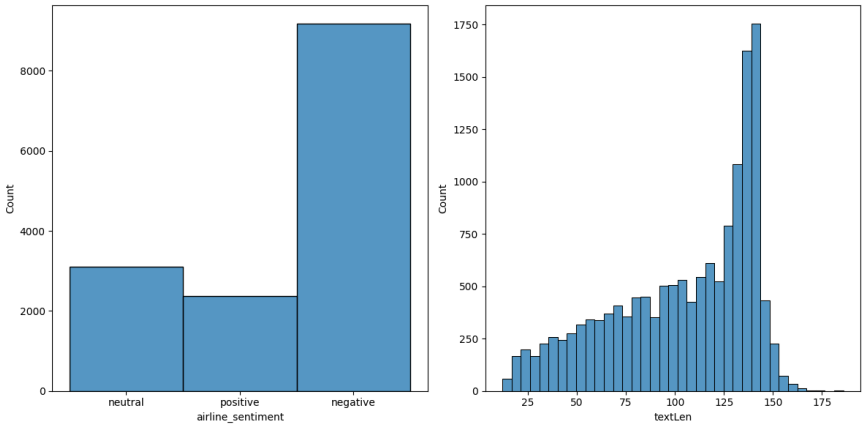
```
df1=dfTweetSubset.groupby('airline_sentiment').count()/14640
df1[['tweet_id']]*100
```

airline_sentiment	tweet_id
negative	62.691257
neutral	21.168033
positive	16.140710

```
from matplotlib import pyplot as plt
plt.rcParams["figure.figsize"] = [12,6]
plt.rcParams["figure.autolayout"] = True
plt.rcParams.update({'font.size': 10})
fig, axes = plt.subplots(1, 2)
import seaborn as sns

sns.histplot(data=dfTweetSubset, x="airline_sentiment", ax=axes[0])
sns.histplot(data=dfTweetSubset, x="textLen", ax=axes[1])
```

```
<Axes: xlabel='textLen', ylabel='Count'>
```




```
dfTweetSubset.isnull().sum()
```

```
tweet_id                      0
airline_sentiment             0
airline_sentiment_gold    14600
text                          0
textLen                       0
dtype: int64
```

Duplicate check & removal

```
dfTweetSubset.shape
```

```
(14640, 5)
```

To count number of dupplicates in the data frame.

```
dfTweetSubset.duplicated().sum()
```

```
127
```

To Count Duplicates and non duplicates

```
dfTweetSubset.duplicated().value_counts()
```

```
False    14513
True       127
dtype: int64
```

Lets remove the duplicates

```
dfTweetSubsetClean1=dfTweetSubset.drop_duplicates()
```

```
dfTweetSubsetClean1.shape
```

```
(14513, 5)
```

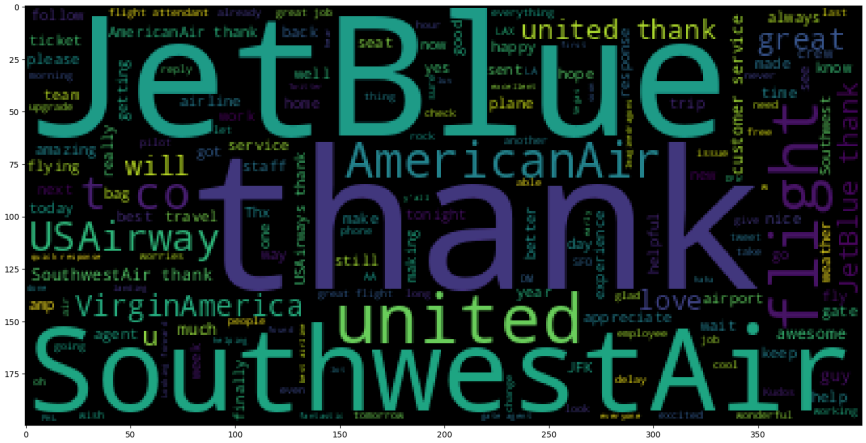
```
dfTweetSubsetClean1['tweet_id'].unique()
```

```
array([570306133677760513, 570301130888122368, 570301083672813571, ...,
       569587242672398336, 569587188687634433, 569587140490866689])
```

```
import pandas as pd
from nltk.corpus import stopwords
import string
import re
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import classification_report
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
import seaborn as sns
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

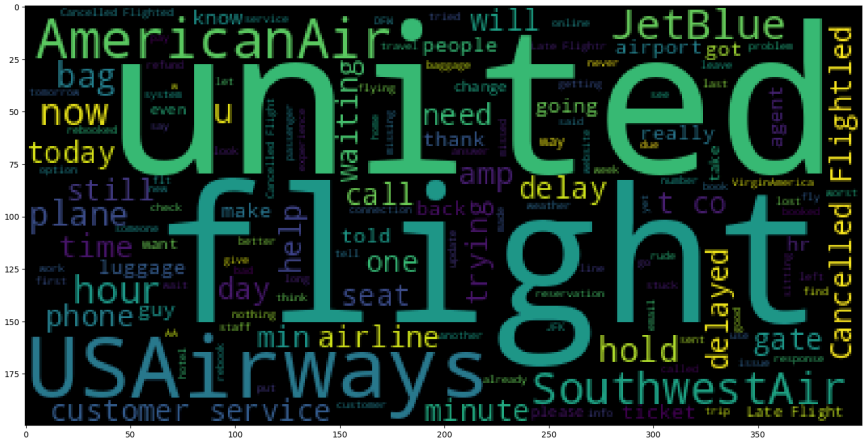
```
positive_tweets = dfTweetSubsetClean1[dfTweetSubset['airline_sentiment'] == "positive"]['text'].tolist()
positive_tweets_string = " ".join(positive_tweets)
plt.figure(figsize=(15,15))
plt.imshow(WordCloud().generate(positive_tweets_string))
```

```
<ipython-input-20-a73d723735a2>:1: UserWarning: Boolean Series key will be reindexed
  positive_tweets = dfTweetSubsetClean1[dfTweetSubset['airline_sentiment'] == "positi
<matplotlib.image.AxesImage at 0x782c7399e320>
```



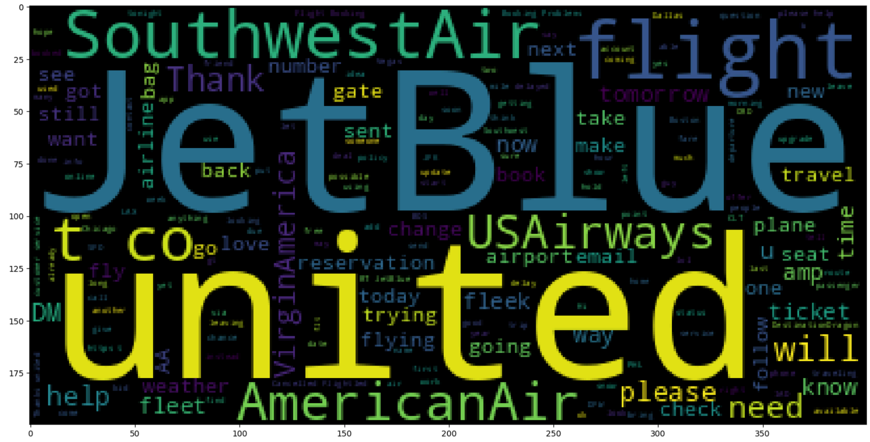

```
negative_tweets = dfTweetSubsetClean1[dfTweetSubset['airline_sentiment'] == "negative"]['text'].tolist()
negative_tweets_string = " ".join(negative_tweets)
plt.figure(figsize=(15,15))
plt.imshow(WordCloud().generate(negative_tweets_string))
```

```
<ipython-input-21-d0b00f92fbd4>:1: UserWarning: Boolean Series key will be reindexed
  negative_tweets = dfTweetSubsetClean1[dfTweetSubset['airline_sentiment'] == "negati
<matplotlib.image.AxesImage at 0x782c7333f400>
```




```
nutral_tweets = dfTweetSubsetClean1[dfTweetSubset['airline_sentiment'] == "neutral"]['text'].tolist()
nutral_tweets_string = " ".join(nutral_tweets)
plt.figure(figsize=(15,15))
plt.imshow(WordCloud().generate(nutral_tweets_string))
```

```
<ipython-input-22-a9830abad1f8>:1: UserWarning: Boolean Series key will be reindexed
  nutral_tweets = dfTweetSubsetClean1[dfTweetSubset['airline_sentiment'] == "neutral"
<matplotlib.image.AxesImage at 0x782c733f3880>
```




```
dfTweetSubsetClean1['text'].head(100)
```

```
0                    @VirginAmerica What @dhepburn said.
1     @VirginAmerica plus you've added commercials t...
2     @VirginAmerica I didn't today... Must mean I n...
3     @VirginAmerica it's really aggressive to blast...
4     @VirginAmerica and it's a really big bad thing...
                            ...
95    @VirginAmerica Is it me, or is your website do...
96    @VirginAmerica I can't check in or add a bag. ...
97    @VirginAmerica - Let 2 scanned in passengers l...
98    @virginamerica What is your phone number. I ca...
99    @VirginAmerica is anyone doing anything there ...
Name: text, Length: 100, dtype: object
```

Performing Text Data Cleaning steps to remove http, puntuations, unicode characters etc.

```
import pandas as pd
import html
import re
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize


pd.set_option('display.max_colwidth', None)
dfTweetSubsetClean1['text']
```

```
0
@VirginAmerica What @dhepburn said.
1                                                                              @VirginAmerica plus you've added commercials
to the experience... tacky.
2                                                                               @VirginAmerica I didn't today... Must mean
I need to take another trip!
3                                   @VirginAmerica it's really aggressive to blast obnoxious "entertainment" in your guests' faces
```

```
&amp;amp; they have little recourse
4                                                                                                    @VirginAmerica and it's a
really big bad thing about it
                                                                                                     ...
14635                                                                                                @AmericanAir thank you we got on a
different flight to Chicago.
14636    @AmericanAir leaving over 20 minutes Late Flight. No warnings or communication until we were 15 minutes Late Flight.
That's called shitty customer svc
14637                                                                                                @AmericanAir Please bring
American Airlines to #BlackBerry10
14638                  @AmericanAir you have my money, you change my flight, and don't answer your phones! Any other suggestions
so I can make my commitment??
14639               @AmericanAir we have 8 ppl so we need 2 know how many seats are on the next flight. Plz put us on standby for
4 people on the next flight?
Name: text, Length: 14513, dtype: object
```

Removing username that starts with @ symbol, http or https:// and any nonalphanumeric characters

```
%time dfTweetSubsetClean1['Tweet_Processed'] = dfTweetSubsetClean1['text'].map(lambda x : ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z
```

```
CPU times: user 188 ms, sys: 0 ns, total: 188 ms
Wall time: 188 ms
<timed exec>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
```

```
dfTweetSubsetClean1[['text','Tweet_Processed']]
```

	text	Tweet_Processed
0	@VirginAmerica What @dhepburn said.	What said
1	@VirginAmerica plus you've added commercials to the experience... tacky.	plus you ve added commercials to the experience tacky
2	@VirginAmerica I didn't today... Must mean I need to take another trip!	I didn t today Must mean I need to take another trip
3	@VirginAmerica it's really aggressive to blast obnoxious "entertainment" in your guests' faces & they have little recourse	it s really aggressive to blast obnoxious entertainment in your guests faces amp they have little recourse
4	@VirginAmerica and it's a really big bad thing about it	and it s a really big bad thing about it
...	...	...
14635	@AmericanAir thank you we got on a different flight to Chicago.	thank you we got on a different flight to Chicago
14636	@AmericanAir leaving over 20 minutes Late Flight. No warnings or communication until we were 15 minutes Late Flight. That's called	leaving over 20 minutes Late Flight No warnings or communication until we were 15 minutes Late Flight That s called shitty

Transforming the text into lower case

```
#Lower Case
%time dfTweetSubsetClean1['Tweet_Processed'] = dfTweetSubsetClean1['Tweet_Processed'].map(lambda x: x.lower())
dfTweetSubsetClean1[['text','Tweet_Processed']]
```

```
CPU times: user 11.6 ms, sys: 0 ns, total: 11.6 ms
Wall time: 15.9 ms
<timed exec>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
```

	text	Tweet_Processed
0	@VirginAmerica What @dhepburn said.	what said
1	@VirginAmerica plus you've added commercials to the experience... tacky.	plus you ve added commercials to the experience tacky

Removing the punchuations

```
#Remove punctuations
%time dfTweetSubsetClean1['Tweet_Processed'] = dfTweetSubsetClean1['Tweet_Processed'].map(lambda x: re.sub(r'[^\w\s]', '', x))
dfTweetSubsetClean1[['text','Tweet_Processed']]
```

```
CPU times: user 34.1 ms, sys: 0 ns, total: 34.1 ms
Wall time: 34.8 ms
<timed exec>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
```

	text	Tweet_Processed
0	@VirginAmerica What @dhepburn said.	what said
1	@VirginAmerica plus you've added commercials to the experience... tacky.	plus you ve added commercials to the experience tacky
2	@VirginAmerica I didn't today... Must mean I need to take another trip!	i didn t today must mean i need to take another trip
3	@VirginAmerica it's really aggressive to blast obnoxious "entertainment" in your guests' faces & they have little recourse	it s really aggressive to blast obnoxious entertainment in your guests faces amp they have little recourse
4	@VirginAmerica and it's a really big bad thing about it	and it s a really big bad thing about it
...	...	...
14635	@AmericanAir thank you we got on a different flight to Chicago.	thank you we got on a different flight to chicago
14636	@AmericanAir leaving over 20 minutes Late Flight. No warnings or communication until we	leaving over 20 minutes late flight no warnings or communication until we were

```
#Remove unicodes
%time dfTweetSubsetClean1['Tweet_Processed'] = dfTweetSubsetClean1['Tweet_Processed'].map(lambda x : re.sub(r'[^\x00-\x7F]+',' ', x))
```

```
CPU times: user 42.9 ms, sys: 0 ns, total: 42.9 ms
Wall time: 44.6 ms
<timed exec>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
```

```
dfTweetSubsetClean1[['text','Tweet_Processed']]
```

	text	Tweet_Processed
0	@VirginAmerica What @dhepburn said.	what said
1	@VirginAmerica plus you've added commercials to the experience... tacky.	plus you ve added commercials to the experience tacky
2	@VirginAmerica I didn't today... Must mean I need to take another trip!	i didn t today must mean i need to take another trip
3	@VirginAmerica it's really aggressive to blast obnoxious "entertainment" in your guests' faces & they have little recourse	it s really aggressive to blast obnoxious entertainment in your guests faces amp they have little recourse
4	@VirginAmerica and it's a really big bad thing about it	and it s a really big bad thing about it
...	...	...

```
from google.colab import drive
drive.mount('/content/drive')
```

```
!pip install nltk
!pip install https://github.com/pandas-profiling/pandas-profiling/archive/master.zip
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pandas_profiling
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Stage-1a: Data Setup

```
import pandas as pd
file_path = 'drive/My Drive/YU-ML-Proj-3/Tweets-modified.csv'
df1 = pd.read_csv(file_path, encoding='latin-1',usecols=[0,1,5,10])
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14640 entries, 0 to 14639
Data columns (total 4 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   tweet_id           14640 non-null  int64 
 1   airline_sentiment  14640 non-null  object
 2   airline            14640 non-null  object
 3   text               14640 non-null  object
dtypes: int64(1), object(3)
memory usage: 457.6+ KB
```

```
df1.dtypes
#print(df1.columns)
```

```
tweet_id              int64
airline_sentiment    object
airline              object
text                 object
dtype: object
```

```
df1.columns=['tweet_id','airline_sentiment', 'airline_type', 'tweet_Original']
```

```
df1.rename(columns={'text ':'tweet_Original'}, inplace=True)
df1.rename(columns={'text ':'tweet_Original'}, inplace=True)
```

```
from random import sample
df1.sample(2, random_state = 42)
#display(df1.head())
display(df1.tail())
```

	tweet_id	airline_sentiment	airline_type	tweet_Original
14635	569587686496825344	positive	American	@AmericanAir thank you we got on a different f...
14636	569587371693355008	negative	American	@AmericanAir leaving over 20 minutes Late Flig...
14637	569587242672398336	positive	American	@AmericanAir Please bring American Airlines to...
14638	569587188687634433	negative	American	@AmericanAir you have my money, you change my ...
14639	569587140490866689	positive	American	@AmericanAir we have 8 ppl so we need 2 know h...

```
len(df1)
df1.count()
```

```
tweet_id            14640
airline_sentiment   14640
airline_type        14640
tweet_Original      14640
dtype: int64
```

```
df1.profile_report()
```

Summarize dataset: 100% 14/14 [00:04<00:00, 2.41it/s, Completed]

Generate report structure: 100% 1/1 [00:05<00:00, 5.09s/it]

Render HTML: 100% 1/1 [00:00<00:00, 1.87it/s]

Overview

Dataset statistics

Number of variables	4
Number of observations	14640
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	147
Duplicate rows (%)	1.0%
Total size in memory	457.6 KiB
Average record size in memory	32.0 B

Variable types

Numeric	1
Categorical	2
Text	1

Alerts

Dataset has 147 (1.0%) duplicate rows — Duplicates

Reproduction

Analysis started	2024-03-14 22:02:38.147599
Analysis finished	2024-03-14 22:02:42.977682
Duration	4.83 seconds

```
# remove duplicated rows
#df1.drop_duplicates(inplace=True)


#df1.dropna(inplace=True)  # This will drop rows with NaN values from your DataFrame


# Identify rows with NaN values
#rows_with_nan = df1[df1.isna().any(axis=1)]

# Display rows with NaN values
#print(rows_with_nan)


#df1.profile_report()


df1 = pd.get_dummies(df1, columns=['airline_sentiment'], drop_first=True)
display(df1.head())
display(df1.tail())
```

	tweet_id	airline_type	tweet_Original	airline_sentiment_positive
0	570301031407624196	Virgin America	@VirginAmerica it's really aggressive to blast...	0
1	570301130888122368	Virgin America	@VirginAmerica plus you've added commercials t...	1
2	570301083672813571	Virgin America	@VirginAmerica I didn't today... Must mean I n...	1
3	570306133677760513	Virgin America	@VirginAmerica What @dhepburn said.	1
4	570300817074462722	Virgin America	@VirginAmerica and it's a really big bad thing...	0

tweet_id	airline_type	tweet_Original	airline_sentiment_positive

```
df2 = df1
df2 = df2[['airline_sentiment_positive','tweet_Original']]
display(df2.head())
display(df2.tail())
```

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

	airline_sentiment_positive	tweet_Original
14635	1	@AmericanAir thank you we got on a different f...
14636	0	@AmericanAir leaving over 20 minutes Late Flig...
14637	1	@AmericanAir Please bring American Airlines to...
14638	0	@AmericanAir you have my money, you change my ...
14639	1	@AmericanAir we have 8 ppl so we need 2 know h...

Stage-1b : Pre-processing on the Text data

```
import re
from bs4 import BeautifulSoup
```

```
# Remove HTTP tags
%time df2['tweet_Processed'] = df2['tweet_Original'].map(lambda x : ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)","
df2.head()
```

```
CPU times: user 201 ms, sys: 3.86 ms, total: 204 ms
Wall time: 210 ms
```

	airline_sentiment_positive	tweet_Original	tweet_Processed
0	0	@VirginAmerica it's really aggressive to blast...	it s really aggressive to blast obnoxious ente...
1	1	@VirginAmerica plus you've added commercials t...	plus you ve added commercials to the experienc...
2	1	@VirginAmerica I didn't today... Must mean I n...	I didn t today Must mean I need to take anothe...
3	1	@VirginAmerica What @dhepburn said.	What said
4	0	@VirginAmerica and it's a really big bad thing...	and it s a really big bad thing about it

```
#Lower Case
%time df2['tweet_Processed'] = df2['tweet_Processed'].map(lambda x: x.lower())
df2.head()
```

```
CPU times: user 6.54 ms, sys: 3.67 ms, total: 10.2 ms
Wall time: 11.6 ms
```

	airline_sentiment_positive	tweet_Original	tweet_Processed
0	0	@VirginAmerica it's really aggressive to blast...	it s really aggressive to blast obnoxious ente...
1	1	@VirginAmerica plus you've added commercials t...	plus you ve added commercials to the experienc...
2	1	@VirginAmerica I didn't today... Must mean I n...	i didn t today must mean i need to take anothe...
3	1	@VirginAmerica What @dhepburn said.	what said
4	0	@VirginAmerica and it's a really big bad thing...	and it s a really big bad thing about it

```
#Remove punctuations-any character that is not a word character or a whitespace character.
%time df2['tweet_Processed'] = df2['tweet_Processed'].map(lambda x: re.sub(r'[^\w\s]', '', x))
df2.head()
```

```
CPU times: user 38 ms, sys: 0 ns, total: 38 ms
Wall time: 41 ms
```

	airline_sentiment_positive	tweet_Original	tweet_Processed
0	0	@VirginAmerica it's really aggressive to blast...	it s really aggressive to blast obnoxious ente...
1	1	@VirginAmerica plus you've added commercials t...	plus you ve added commercials to the experienc...
2	1	@VirginAmerica I didn't today... Must mean I n...	i didn t today must mean i need to take anothe...
3	1	@VirginAmerica What @dhepburn said.	what said
4	0	@VirginAmerica and it's a really big bad thing...	and it s a really big bad thing about it

```
#Remove unicodes
%time df2['tweet_Processed'] = df2['tweet_Processed'].map(lambda x : re.sub(r'[^\x00-\x7F]+',' ', x))
df2.head()
```

```
CPU times: user 42 ms, sys: 0 ns, total: 42 ms
Wall time: 42.8 ms
```

	airline_sentiment_positive	tweet_Original	tweet_Processed
0	0	@VirginAmerica it's really aggressive to blast...	it s really aggressive to blast obnoxious ente...
1	1	@VirginAmerica plus you've added commercials t...	plus you ve added commercials to the experienc...
2	1	@VirginAmerica I didn't today... Must mean I n...	i didn t today must mean i need to take anothe...
3	1	@VirginAmerica What @dhepburn said.	what said
4	0	@VirginAmerica and it's a really big bad thing...	and it s a really big bad thing about it

```
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords


# Remove stopwords
import nltk
nltk.download('stopwords')
stop_words = stopwords.words('english')
%time df2['tweet_Processed'] = df2['tweet_Processed'].map(lambda x : ' '.join([w for w in x.split() if w not in stop_words]))
df2.head()
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
CPU times: user 524 ms, sys: 9.91 ms, total: 534 ms
Wall time: 536 ms
```

	airline_sentiment_positive	tweet_Original	tweet_Processed
0	0	@VirginAmerica it's really aggressive to blast...	really aggressive blast obnoxious entertainmen...
1	1	@VirginAmerica plus you've added commercials t...	plus added commercials experience tacky
2	1	@VirginAmerica I didn't today... Must mean I n...	today must mean need take another trip
3	1	@VirginAmerica What	

```
# Lemmatize the text
nltk.download('wordnet')
lemmer = WordNetLemmatizer()

%time df2['tweet_Processed'] = df2['tweet_Processed'].map(lambda x : ' '.join([lemmer.lemmatize(w) for w in x.split() if w not in stop_w
df2.head()
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
CPU times: user 2.39 s, sys: 39.8 ms, total: 2.43 s
Wall time: 2.44 s
```

	airline_sentiment_positive	tweet_Original	tweet_Processed
0	0	@VirginAmerica it's really aggressive to blast...	really aggressive blast obnoxious entertainmen...
1	1	@VirginAmerica plus you've added commercials t...	plus added commercial experience tacky
2	1	@VirginAmerica I didn't today... Must mean I n...	today must mean need take another trip
3	1	@VirginAmerica What @dhepburn said.	said
4	0	@VirginAmerica and it's a really big bad thing...	really big bad thing

```
#Removing Stop words again after Lemmatize
%time df2['tweet_Processed'] = df2['tweet_Processed'].map(lambda x : ' '.join([w for w in x.split() if w not in stop_words]))
display(df2.head())
display(df2.tail())
```

```
CPU times: user 385 ms, sys: 1.18 ms, total: 387 ms
Wall time: 391 ms
```

	airline_sentiment_positive	tweet_Original	tweet_Processed
0	0	@VirginAmerica it's really aggressive to blast...	really aggressive blast obnoxious entertainmen...
1	1	@VirginAmerica plus you've added commercials t...	plus added commercial experience tacky
2	1	@VirginAmerica I didn't today... Must mean I n...	today must mean need take another trip
3	1	@VirginAmerica What @dhepburn said.	said
4	0	@VirginAmerica and it's a really big bad thing...	really big bad thing

	airline_sentiment_positive	tweet_Original	tweet_Processed
14635	1	@AmericanAir thank you we got on a different f...	thank got different flight chicago
14636	0	@AmericanAir leaving over 20 minutes Late Flig...	leaving 20 minute late flight warning communic...

Stage 2 : Embedding on the processed text data

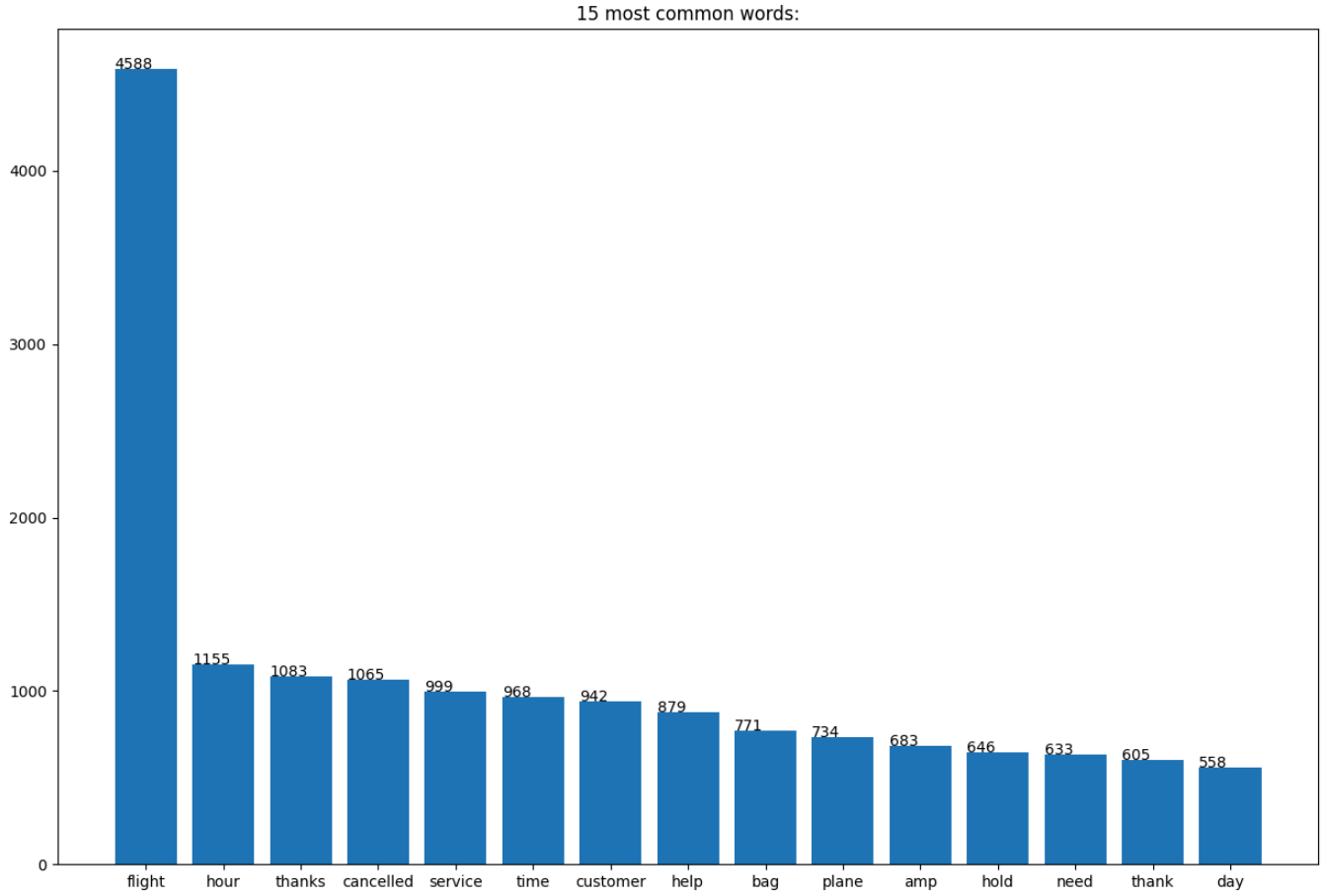
```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import HashingVectorizer
```

2a:EDA

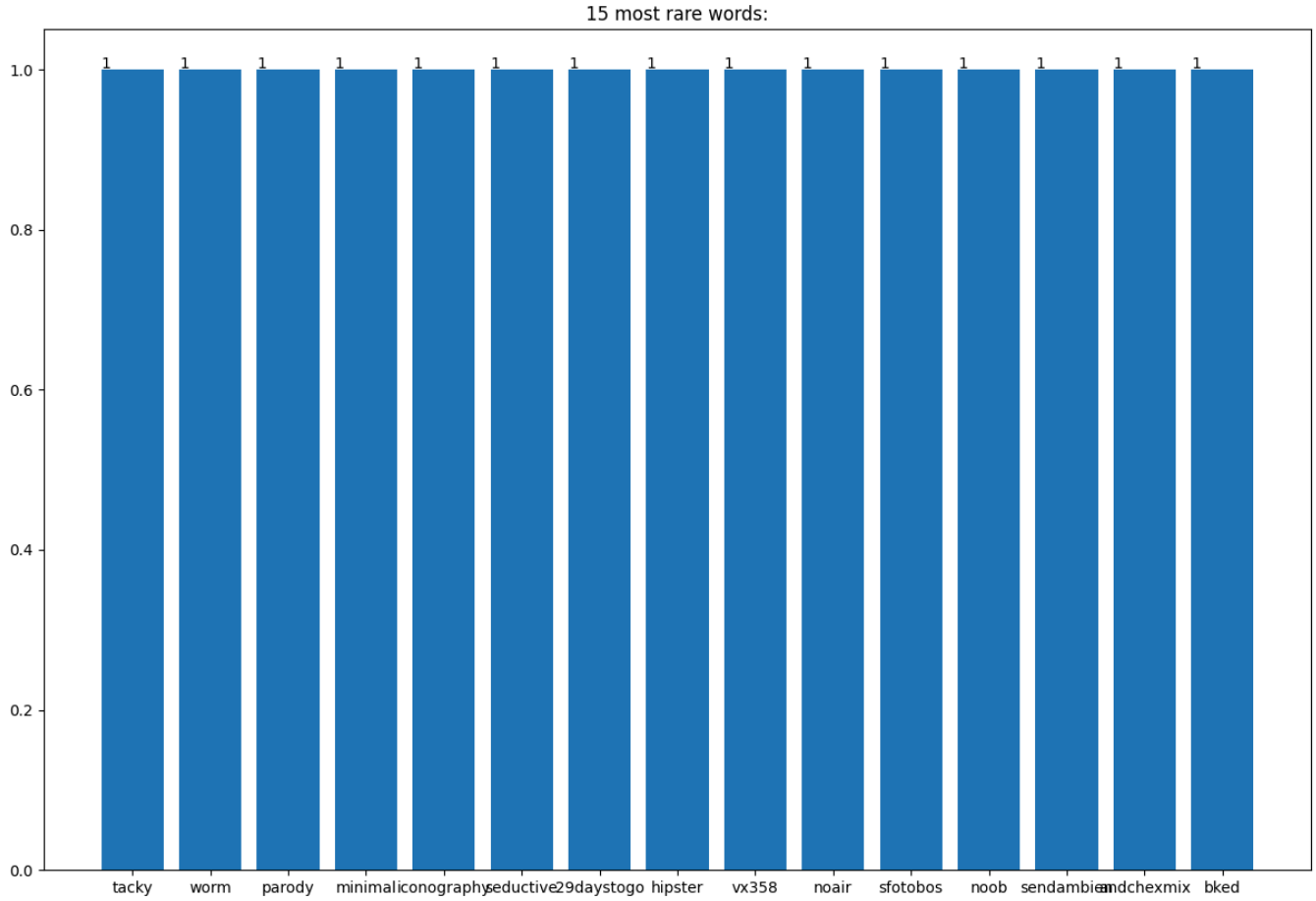
```
#funtion to get 'top N' or 'bottom N' words
def get_n_words(corpus, direction, n):
    vec = CountVectorizer(stop_words = 'english').fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    if direction == "top":
        words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    else:
        words_freq =sorted(words_freq, key = lambda x: x[1], reverse=False)
    return words_freq[:n]
```

```
#10 most common and 10 most rare words
common_words = get_n_words(df2['tweet_Processed'], "top", 15)
rare_words = get_n_words(df2['tweet_Processed'], "bottom", 15)
```

```
common_words = dict(common_words)
names = list(common_words.keys())
values = list(common_words.values())
plt.subplots(figsize = (15,10))
bars = plt.bar(range(len(common_words)),values,tick_label=names)
plt.title('15 most common words:')
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x(), yval + .01, yval)
plt.show()
```




```
rare_words = dict(rare_words)
names = list(rare_words.keys())
values = list(rare_words.values())
plt.subplots(figsize = (15,10))
bars = plt.bar(range(len(rare_words)),values,tick_label=names)
plt.title('15 most rare words:')
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x(), yval + .001, yval)
plt.show()
```




```
# BOW-TF Embedding
from sklearn.feature_extraction.text import CountVectorizer

no_features = 800
tf_vectorizer = CountVectorizer(min_df=.015, max_df=.8, max_features=no_features, ngram_range=(1, 3))

tpl_tf = tf_vectorizer.fit_transform(df2['tweet_Processed'])
display("Bow-TF :", tpl_tf.shape)
df_tf = pd.DataFrame(tpl_tf.toarray(), columns=tf_vectorizer.get_feature_names_out())
display(df_tf.head())
```

```
'Bow-TF :'
(14640, 100)
```

	agent	airline	airport	amp	another	back	bag	baggage	call	cancelled	...	un
0	0	0	0	1	0	0	0	0	0	0	...	
1	0	0	0	0	0	0	0	0	0	0	...	
2	0	0	0	0	1	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	...	

5 rows × 100 columns

```
#Preparing processed and BoW-TF embedded data for Classification
df_tf_m = pd.concat([df2, df_tf], axis = 1)
df_tf_m.drop(columns=['tweet_Original', 'tweet_Processed'], inplace = True)
print(df_tf_m.shape)
display(df_tf_m.head())
display(df_tf_m.tail())
```

```
(14640, 101)
```

	airline_sentiment_positive	agent	airline	airport	amp	another	back	bag	bagg
0	0	0	0	0	1	0	0	0	
1	1	0	0	0	0	0	0	0	
2	1	0	0	0	0	1	0	0	
3	1	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	

5 rows × 101 columns

	airline_sentiment_positive	agent	airline	airport	amp	another	back	bag	
14635	1	0	0	0	0	0	0	0	
14636	0	0	0	0	0	0	0	0	
14637	1	0	1	0	0	0	0	0	
14638	0	0	0	0	0	0	0	0	
14639	1	0	0	0	0	0	0	0	

5 rows × 101 columns

```
# Identify rows with NaN values
#rows_with_nan = df_tf_m[df_tf_m.isna().any(axis=1)]
# Display rows with NaN values
#print(rows_with_nan)


# BoW-TF:IDF Embedding
tfidf_vectorizer = TfidfVectorizer(min_df=.02, max_df=.7, ngram_range=(1,3))

%time tpl_tfidf = tfidf_vectorizer.fit_transform(df2['tweet_Processed'])
display("Bow-TF:IDF :", tpl_tfidf.shape)
df_tfidf = pd.DataFrame(tpl_tfidf.toarray(), columns=tfidf_vectorizer.get_feature_names_out(), index=df2.index)
display(df_tfidf.head())
```

```
CPU times: user 533 ms, sys: 34.7 ms, total: 568 ms
Wall time: 577 ms
'Bow-TF:IDF :'
(14640, 61)
```

	agent	airline	airport	amp	back	bag	call	cancelled	cancelled flightled	change	.
0	0.0	0.0	0.0	0.648372	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	

```
#Preparing processed and BoW-TF:IDF embedded data for Classification
df_tfidf_m = pd.concat([df2, df_tfidf], axis = 1)
df_tfidf_m.drop(columns=['tweet_Original', 'tweet_Processed'], inplace = True)
print(df_tfidf_m.shape)
display(df_tfidf_m.head())
display(df_tfidf_m.tail())
```

```
(14640, 62)
```

	airline_sentiment_positive	agent	airline	airport	amp	back	bag	call	can
0	0	0.0	0.0	0.0	0.648372	0.0	0.0	0.0	
1	1	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	
2	1	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	
3	1	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	
4	0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	

5 rows × 62 columns

	airline_sentiment_positive	agent	airline	airport	amp	back	bag	call	can
14635	1	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	
14636	0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	
14637	1	0.0	0.707557	0.0	0.0	0.0	0.0	0.0	
14638	0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	
14639	1	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	

Stage-3: Model Building

```
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.model_selection import StratifiedKFold, cross_validate, train_test_split, cross_val_score, KFold
from sklearn.preprocessing import StandardScaler, OneHotEncoder, LabelEncoder
from sklearn.metrics import roc_curve, auc, classification_report, confusion_matrix, precision_score, recall_score,  accuracy_score, pre


#function to prepare Confusion Matrix, RoC-AUC curve, and relvant statistics
def clf_report(Y_test, Y_pred, probs):
    print("\n", "Confusion Matrix")
    cm = confusion_matrix(Y_test, Y_pred)
    #print("\n", cm, "\n")
    sns.heatmap(cm, square=True, annot=True, cbar=False, fmt = 'g', cmap='RdBu',
                xticklabels=['positive', 'negative'], yticklabels=['positive', 'negative'])
    plt.xlabel('true label')
    plt.ylabel('predicted label')
    plt.show()
    print("\n", "Classification Report", "\n")
    print(classification_report(Y_test, Y_pred))
    print("Overall Accuracy : ", round(accuracy_score(Y_test, Y_pred) * 100, 2))
    print("Precision Score : ", round(precision_score(Y_test, Y_pred, average='binary') * 100, 2))
    print("Recall Score : ", round(recall_score(Y_test, Y_pred, average='binary') * 100, 2))
    preds = probs[:,1] # this is the probability for 1, column 0 has probability for 0. Prob(0) + Prob(1) = 1
    fpr, tpr, threshold = roc_curve(Y_test, preds)
    roc_auc = auc(fpr, tpr)
    print("AUC : ", round(roc_auc * 100, 2), "\n")
    #display(probs)
    #print("Cutoff Probability : ", preds)
    plt.figure()
    plt.plot(fpr, tpr, label='Best Model on Test Data (area = %0.2f)' % roc_auc)
    plt.plot([0.0, 1.0], [0, 1],'r--')
    plt.xlim([-0.1, 1.1])
    plt.ylim([-0.1, 1.1])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('RoC-AUC on Test Data')
    plt.legend(loc="lower right")
    plt.savefig('Log_ROC')
    plt.show()
    print("-------------------------------------------------------------------------------")
```

```
#function to prepare different Classification models
from imblearn.over_sampling import SMOTE
smote = SMOTE()
def model_dvt(df):
    Y = df['airline_sentiment_positive']
    X = df.drop('airline_sentiment_positive', axis = 1)

    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, train_size = 0.85, random_state = 21)
    print("Before Applying SMOTE")
    print("Train Data Dimensions : ", X_train.shape, Y_train.shape)
    print("Test Data Dimensions : ", X_test.shape)

    X_train_resampled, Y_train_resampled = smote.fit_resample(X_train, Y_train)
    print("After Applying SMOTE")
    print("Train Data Dimensions : ", X_train_resampled.shape, Y_train_resampled.shape)
    print("Test Data Dimensions : ", X_test.shape, Y_test.shape)

    print("\n", 'Random Forest Classifier')
    clf_RF = RandomForestClassifier(n_estimators=500, max_depth=10, random_state=21)
    %time clf_RF.fit(X_train_resampled, Y_train_resampled)
    Y_pred = clf_RF.predict(X_test)
    probs = clf_RF.predict_proba(X_test)
    clf_report(Y_test, Y_pred, probs)

    print("\n", 'AdaBoost Classifier')
    clf_AdB = AdaBoostClassifier(n_estimators=200,random_state=21)
    %time clf_AdB.fit(X_train_resampled, Y_train_resampled)
    Y_pred = clf_AdB.predict(X_test)
    probs = clf_AdB.predict_proba(X_test)
    clf_report(Y_test, Y_pred, probs)

    print("\n", 'Grdient Boosting Classifier')
    clf_GB = GradientBoostingClassifier(n_estimators=200, max_depth=1, random_state=21, learning_rate=1.5)
    %time clf_GB.fit(X_train_resampled, Y_train_resampled)
    Y_pred = clf_GB.predict(X_test)
    probs = clf_GB.predict_proba(X_test)
    clf_report(Y_test, Y_pred, probs)

    print("\n", 'Naive Bayes Classifier')
    #clf = MultinomialNB(alpha = 1.0)
    clf_NB = GaussianNB()
    %time clf_NB.fit(X_train_resampled, Y_train_resampled)
    Y_pred = clf_NB.predict(X_test)
    probs = clf_NB.predict_proba(X_test)
    clf_report(Y_test, Y_pred, probs)

    print("\n", 'Logistic Regression Classifier')
    clf_LR = LogisticRegression(random_state=21)
    %time clf_LR.fit(X_train_resampled, Y_train_resampled)
    Y_pred = clf_LR.predict(X_test)
    probs = clf_LR.predict_proba(X_test)
    clf_report(Y_test, Y_pred, probs)

    print("\n", 'Support Vector Machine Classifier')
    clf_SVM = SVC(probability=True, random_state=21)
    %time clf_SVM.fit(X_train_resampled, Y_train_resampled)
    Y_pred = clf_SVM.predict(X_test)
    probs = clf_SVM.predict_proba(X_test)
    clf_report(Y_test, Y_pred, probs)

    print("\n", 'K-Nearest Neighbors Classifier')
    clf_KNN = KNeighborsClassifier()
    %time clf_KNN.fit(X_train_resampled, Y_train_resampled)
    Y_pred = clf_KNN.predict(X_test)
    probs = clf_KNN.predict_proba(X_test)
    clf_report(Y_test, Y_pred, probs)

    print("\n", 'Decision Tree Classifier')
    clf_DT = DecisionTreeClassifier(random_state=21)
    %time clf_DT.fit(X_train_resampled, Y_train_resampled)
    Y_pred = clf_DT.predict(X_test)
    probs = clf_DT.predict_proba(X_test)
    clf_report(Y_test, Y_pred, probs)

    print("\nLinear Discriminant Analysis")
    clf_LDA = LinearDiscriminantAnalysis()
    %time clf_LDA.fit(X_train_resampled, Y_train_resampled)
    Y_pred = clf_LDA.predict(X_test)
    probs = clf_LDA.predict_proba(X_test)
    clf_report(Y_test, Y_pred, probs)
```

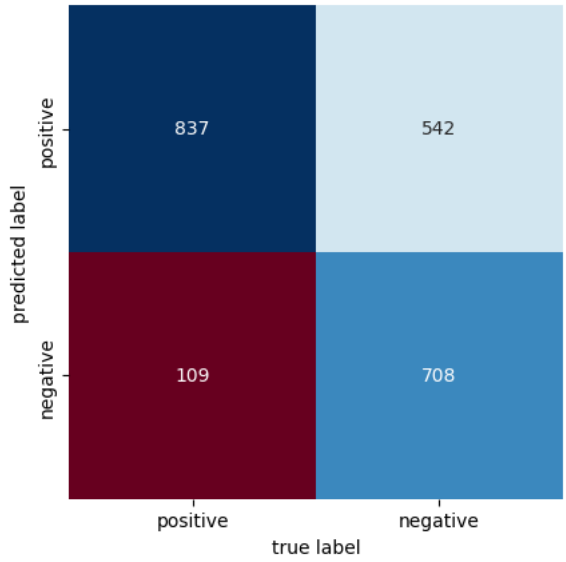
```
#df_tf_m.dropna(inplace=True)


print('Models on Term Frequency - Bag of Words data')
%time model_dvt(df_tf_m)
```

```
Models on Term Frequency - Bag of Words data
Before Applying SMOTE
Train Data Dimensions :  (12444, 100) (12444,)
Test Data Dimensions :  (2196, 100)
After Applying SMOTE
Train Data Dimensions :  (15598, 100) (15598,)
Test Data Dimensions :  (2196, 100) (2196,)

 Random Forest Classifier
CPU times: user 3.97 s, sys: 10.8 ms, total: 3.98 s
Wall time: 3.99 s

 Confusion Matrix
```



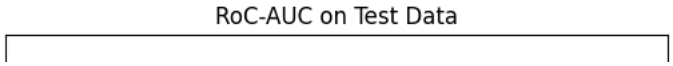

```
 Classification Report

              precision    recall  f1-score   support

           0       0.88      0.61      0.72      1379
           1       0.57      0.87      0.69       817

    accuracy                           0.70      2196
   macro avg       0.73      0.74      0.70      2196
weighted avg       0.77      0.70      0.71      2196

Overall Accuracy :  70.36
Precision Score :  56.64
Recall Score :  86.66
AUC :  82.21
```



4. Hyper-parameter Tuning models that used TF-BoW embedding data

4.1. Grid-Search hyperparameter tuning on AdaBoost Classifier

```
from imblearn.over_sampling import SMOTE
smote = SMOTE()
from sklearn.model_selection import GridSearchCV
Y = df_tf_m['airline_sentiment_positive']
X = df_tf_m.drop('airline_sentiment_positive', axis = 1)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, train_size = 0.85, random_state = 21)
print("Train Data Dimensions : ", X_train.shape)
print("Test Data Dimensions : ", X_test.shape)
X_train_resampled, Y_train_resampled = smote.fit_resample(X_train, Y_train)
```

```
Train Data Dimensions :  (12444, 100)
Test Data Dimensions :  (2196, 100)
```

```
#Creating a grid of hyperparameters
grid_params = {'n_estimators' : [100,200,300],
               'learning_rate' : [1.0, 0.1, 0.05]}

ABC = AdaBoostClassifier()
#Building a 10 fold CV GridSearchCV object
grid_object = GridSearchCV(estimator = ABC, param_grid = grid_params, scoring = 'roc_auc', cv = 10, n_jobs = -1)

#Fitting the grid to the training data
%time grid_object.fit(X_train_resampled, Y_train_resampled)
```

```
CPU times: user 5.94 s, sys: 917 ms, total: 6.86 s
Wall time: 4min 39s
```

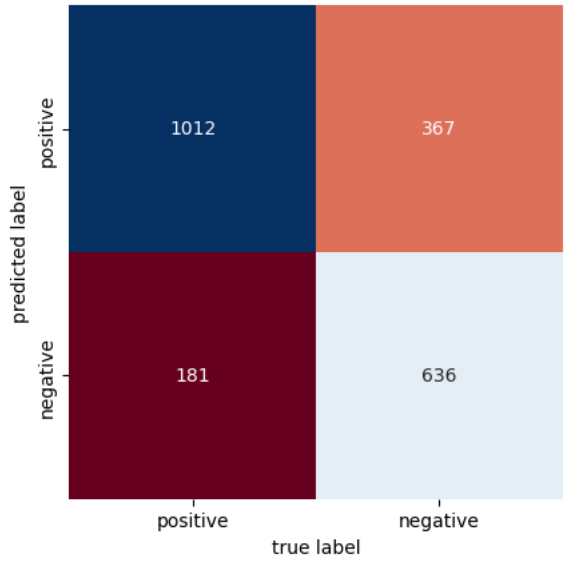

```
#Extracting the best parameters and score
print("Best Parameters : ", grid_object.best_params_)
print("Best_ROC-AUC : ", round(grid_object.best_score_ * 100, 2))
print("Best model : ", grid_object.best_estimator_)

#Applying the tuned parameters back to the model
Y_pred = grid_object.best_estimator_.predict(X_test)
probs = grid_object.best_estimator_.predict_proba(X_test)
clf_report(Y_test, Y_pred, probs)

kfold = KFold(n_splits=10, random_state=25, shuffle=True)
%time results = cross_val_score(grid_object.best_estimator_, X_test, Y_test, cv=kfold)
results = results * 100
results = np.round(results,2)
print("Cross Validation Accuracy : ", round(results.mean(), 2))
print("Cross Validation Accuracy in every fold : ", results)
```

```
Best Parameters :  {'learning_rate': 1.0, 'n_estimators': 100}
Best_ROC-AUC :  84.78
Best model :  AdaBoostClassifier(n_estimators=100)
```

Confusion Matrix



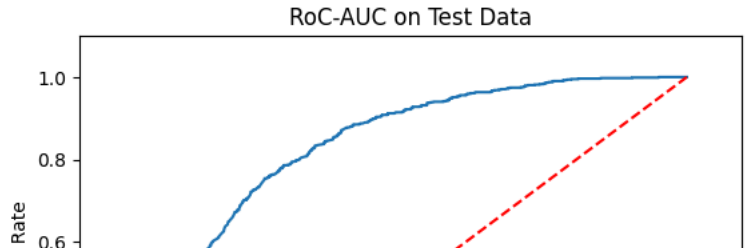
Classification Report

```
              precision    recall  f1-score   support

           0       0.85      0.73      0.79      1379
           1       0.63      0.78      0.70       817

    accuracy                           0.75      2196
   macro avg       0.74      0.76      0.74      2196
weighted avg       0.77      0.75      0.75      2196

Overall Accuracy :  75.05
Precision Score :  63.41
Recall Score :  77.85
AUC :  83.31
```



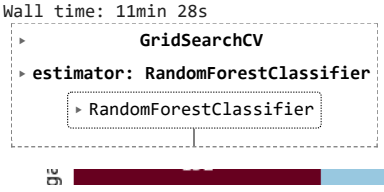
Grid-Search hyperparameter tuning on Random Forest Classifier

```
grid_params = {'n_estimators' : [100,200,300,400,500],
               'max_depth' : [10, 7, 5, 3],
               'criterion' : ['entropy', 'gini']}

RFC = RandomForestClassifier()
grid_object = GridSearchCV(estimator = RFC, param_grid = grid_params, scoring = 'roc_auc', cv = 10, n_jobs = -1)

%time grid_object.fit(X_train_resampled, Y_train_resampled)
```

```
CPU times: user 11.2 s, sys: 1.24 s, total: 12.5 s
Wall time: 11min 28s
```



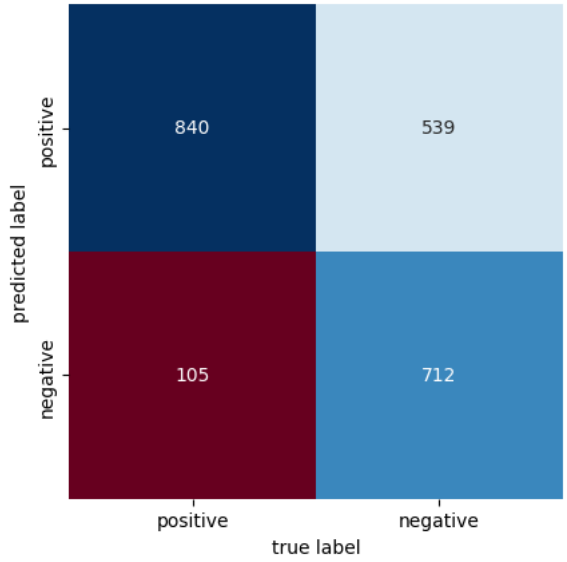
```
# print("Best Parameters : ", grid_object.best_params_)
print("Best_ROC-AUC : ", round(grid_object.best_score_ * 100, 2))
print("Best model : ", grid_object.best_estimator_)

Y_pred = grid_object.best_estimator_.predict(X_test)
probs = grid_object.best_estimator_.predict_proba(X_test)
clf_report(Y_test, Y_pred, probs)

kfold = KFold(n_splits=10, random_state=25, shuffle=True)
%time results = cross_val_score(grid_object.best_estimator_, X_test, Y_test, cv=kfold)
results = results * 100
results = np.round(results,2)
print("Cross Validation Accuracy : ", round(results.mean(), 2))
print("Cross Validation Accuracy in every fold : ", results)
```

```
Best_ROC-AUC :  83.17
Best model :  RandomForestClassifier(criterion='entropy', max_depth=10, n_estimators=400)
```

```
Confusion Matrix
```



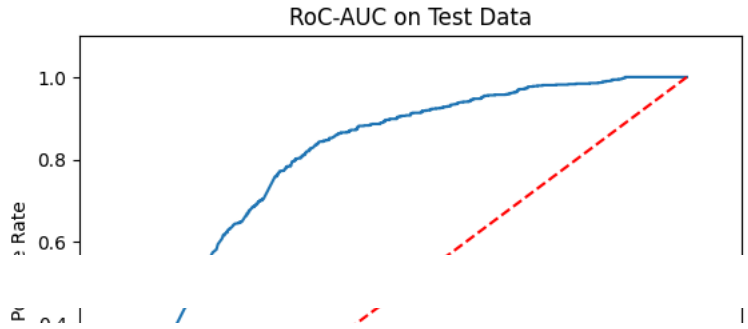

```
Classification Report

              precision    recall  f1-score   support

           0       0.89      0.61      0.72      1379
           1       0.57      0.87      0.69       817

    accuracy                           0.71      2196
   macro avg       0.73      0.74      0.71      2196
weighted avg       0.77      0.71      0.71      2196

Overall Accuracy :  70.67
Precision Score :  56.91
Recall Score :  87.15
AUC :  82.24
```



Grid-Search hyperparameter tuning on GradientBoost

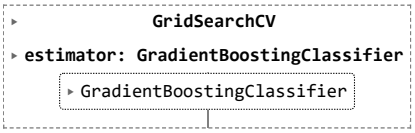
```
grid_params = {
    'n_estimators': [100, 200, 300],
    'learning_rate': [0.1, 0.05, 0.01],
    'max_depth': [3, 5, 7]
}

#Create a Gradient Boosting Classifier object
GBC = GradientBoostingClassifier()

# Build a 10-fold CV GridSearchCV object
grid_object = GridSearchCV(estimator=GBC, param_grid=grid_params, scoring='roc_auc', cv=10, n_jobs=-1)

# Fit the grid to the training data
%time grid_object.fit(X_train_resampled, Y_train_resampled)
```

```
CPU times: user 28.9 s, sys: 2.97 s, total: 31.9 s
Wall time: 29min 53s
```



False Positive Rate

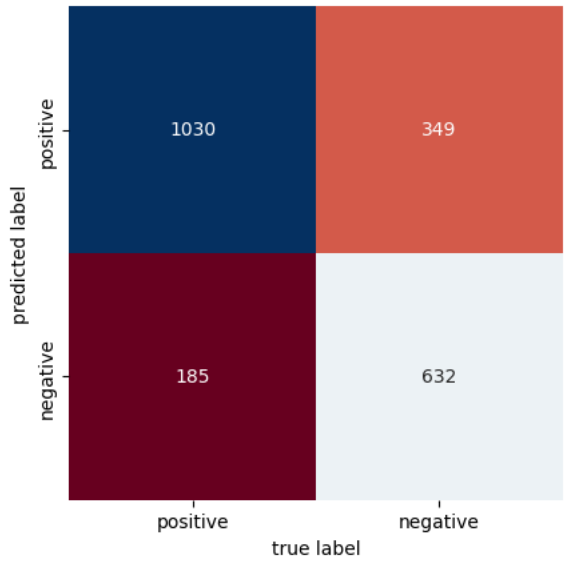
```
# Extract the best parameters and score
print("Best Parameters:", grid_object.best_params_)
print("Best ROC-AUC:", round(grid_object.best_score_ * 100, 2))

# Apply the tuned parameters back to the model
best_model = grid_object.best_estimator_
Y_pred = best_model.predict(X_test)
probs = best_model.predict_proba(X_test)
clf_report(Y_test, Y_pred, probs)

kfold = KFold(n_splits=10, random_state=25, shuffle=True)
%time results = cross_val_score(grid_object.best_estimator_, X_test, Y_test, cv=kfold)
results = results * 100
results = np.round(results,2)
print("Cross Validation Accuracy : ", round(results.mean(), 2))
print("Cross Validation Accuracy in every fold : ", results)
```

```
Best Parameters: {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 300}
Best ROC-AUC: 85.5
```

```
Confusion Matrix
```



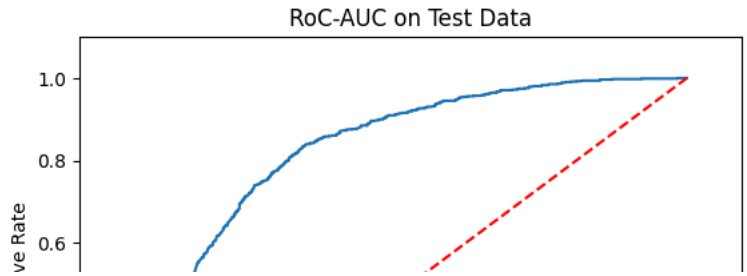

```
Classification Report

              precision    recall  f1-score   support

           0       0.85      0.75      0.79      1379
           1       0.64      0.77      0.70       817

    accuracy                           0.76      2196
   macro avg       0.75      0.76      0.75      2196
weighted avg       0.77      0.76      0.76      2196

Overall Accuracy :  75.68
Precision Score :  64.42
Recall Score :  77.36
AUC :  83.67
```



Grid Search Hyperparameter tunning on Logisitc Regression

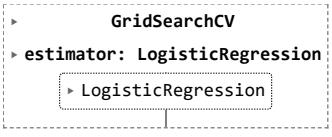
```
# Creating a grid of hyperparameters
grid_params_LR = {'C': [0.001, 0.01, 0.1, 1, 10, 100], 'penalty': ['l1', 'l2']}

# Instantiate Logistic Regression classifier
log_reg = LogisticRegression()

# Building a 10-fold CV GridSearchCV object
grid_object_LR = GridSearchCV(estimator=log_reg, param_grid=grid_params_LR, scoring='roc_auc', cv=10, n_jobs=-1)

# Fitting the grid to the training data
%time grid_object_LR.fit(X_train_resampled, Y_train_resampled)
```

```
CPU times: user 562 ms, sys: 153 ms, total: 715 ms
Wall time: 11 s
```



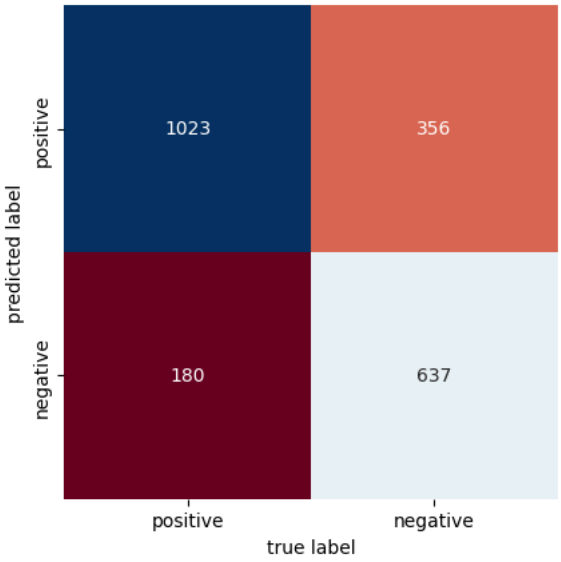
```
#Extracting the best parameters and score
print("Best Parameters : ", grid_object_LR.best_params_)
print("Best_ROC-AUC : ", round(grid_object_LR.best_score_ * 100, 2))
print("Best model : ", grid_object_LR.best_estimator_)

#Applying the tuned parameters back to the model
Y_pred = grid_object_LR.best_estimator_.predict(X_test)
probs = grid_object_LR.best_estimator_.predict_proba(X_test)
clf_report(Y_test, Y_pred, probs)

kfold = KFold(n_splits=10, random_state=25, shuffle=True)
%time results = cross_val_score(grid_object_LR.best_estimator_, X_test, Y_test, cv=kfold)
results = results * 100
results = np.round(results,2)
print("Cross Validation Accuracy : ", round(results.mean(), 2))
print("Cross Validation Accuracy in every fold : ", results)
```

```
Best Parameters :  {'C': 10, 'penalty': 'l2'}
Best_ROC-AUC :  85.18
Best model :  LogisticRegression(C=10)
```

```
Confusion Matrix
```



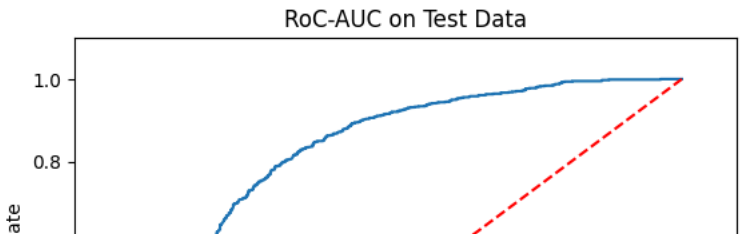

```
Classification Report

              precision    recall  f1-score   support

           0       0.85      0.74      0.79      1379
           1       0.64      0.78      0.70       817

    accuracy                           0.76      2196
   macro avg       0.75      0.76      0.75      2196
weighted avg       0.77      0.76      0.76      2196

Overall Accuracy :  75.59
Precision Score :  64.15
Recall Score :  77.97
AUC :  83.35
```



Grid-Search hyperparameter tuning on Naive Bayes

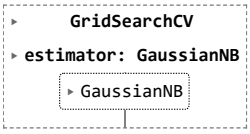
```
# Creating a grid of hyperparameters
grid_params_NB = {}

# Instantiate Gaussian Naive Bayes classifier
NB = GaussianNB()

# Building a 10-fold CV GridSearchCV object
grid_object_NB = GridSearchCV(estimator=NB, param_grid=grid_params_NB, scoring='roc_auc', cv=10, n_jobs=-1)

# Fitting the grid to the training data
%time grid_object_NB.fit(X_train_resampled, Y_train_resampled)
```

```
CPU times: user 80 ms, sys: 11.9 ms, total: 92 ms
Wall time: 761 ms
```




```
weighted avg       0.77      0.74      0.75      2196
```

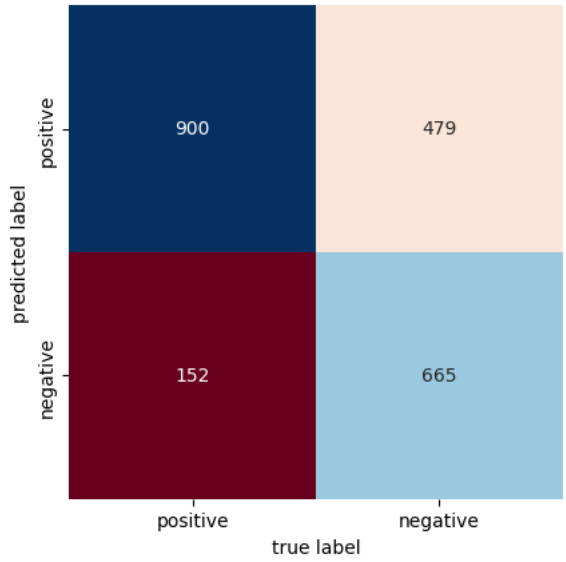
```
#Extracting the best parameters and score
print("Best Parameters : ", grid_object_NB.best_params_)
print("Best_ROC-AUC : ", round(grid_object_NB.best_score_ * 100, 2))
print("Best model : ", grid_object_NB.best_estimator_)

#Applying the tuned parameters back to the model
Y_pred = grid_object_NB.best_estimator_.predict(X_test)
probs = grid_object_NB.best_estimator_.predict_proba(X_test)
clf_report(Y_test, Y_pred, probs)

kfold = KFold(n_splits=10, random_state=25, shuffle=True)
%time results = cross_val_score(grid_object_NB.best_estimator_, X_test, Y_test, cv=kfold)
results = results * 100
results = np.round(results,2)
print("Cross Validation Accuracy : ", round(results.mean(), 2))
print("Cross Validation Accuracy in every fold : ", results)
```

```
Best Parameters :  {}
Best_ROC-AUC :  81.36
Best model :  GaussianNB()
```

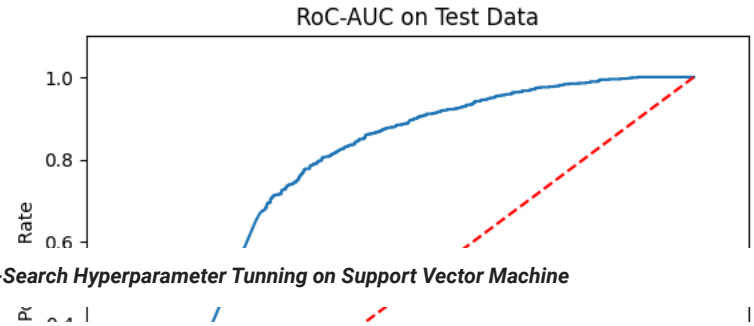
Confusion Matrix



Classification Report

	precision	recall	f1-score	support
0	0.86	0.65	0.74	1379
1	0.58	0.81	0.68	817
accuracy			0.71	2196
macro avg	0.72	0.73	0.71	2196
weighted avg	0.75	0.71	0.72	2196

```
Overall Accuracy :  71.27
Precision Score :  58.13
Recall Score :  81.4
AUC :  78.84
```



Grid-Search Hyperparameter Tunning on Support Vector Machine

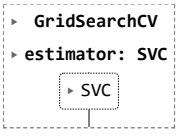
```
# Creating a grid of hyperparameters
grid_params_SVM = {'C': [10], 'gamma': [1], 'kernel': ['rbf']}

# Instantiate Support Vector Machine classifier
svm = SVC(probability=True)

# Building a 10-fold CV GridSearchCV object
grid_object_SVM = GridSearchCV(estimator=svm, param_grid=grid_params_SVM, scoring='roc_auc', cv=10, n_jobs=-1)

# Fitting the grid to the training data
%time grid_object_SVM.fit(X_train_resampled, Y_train_resampled)
```

```
CPU times: user 5min 52s, sys: 5.91 s, total: 5min 58s
Wall time: 46min 7s
```



```
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.svm import SVC
# Instantiate Support Vector Machine classifier with optimized hyperparameters
svm = SVC(C=10, gamma=0.01, kernel='rbf')

def SVM_dvt(X_train_resampled, Y_train_resampled):
  print("\n", 'Support Vector Machine Classifier')
  clf_SVM = SVC(probability=True, C=10, gamma=0.01, kernel='rbf',random_state=21)
  %time clf_SVM.fit(X_train_resampled, Y_train_resampled)
  Y_pred = clf_SVM.predict(X_test)
  probs = clf_SVM.predict_proba(X_test)
  clf_report(Y_test, Y_pred, probs)


#Extracting the best parameters and score
print("Best Parameters : ", grid_object_SVM.best_params_)
print("Best_ROC-AUC : ", round(grid_object_SVM.best_score_ * 100, 2))
print("Best model : ", grid_object_SVM.best_estimator_)

#Applying the tuned parameters back to the model
Y_pred = grid_object_SVM.best_estimator_.predict(X_test)
probs = grid_object_SVM.best_estimator_.predict_proba(X_test)
clf_report(Y_test, Y_pred, probs)

kfold = KFold(n_splits=10, random_state=25, shuffle=True)
%time results = cross_val_score(grid_object_SVM.best_estimator_, X_test, Y_test, cv=kfold)
results = results * 100
results = np.round(results,2)
print("Cross Validation Accuracy : ", round(results.mean(), 2))
print("Cross Validation Accuracy in every fold : ", results)
```

Best Parameters : {'C': 10, 'gamma': 1, 'kernel': 'rbf'}

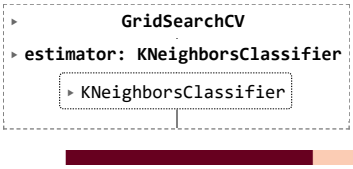
Grid-Seach Hyperparameter Tunning on K-Nearest Neighbour

```
# Creating a grid of hyperparameters
grid_params_KNN = {'n_neighbors': [3, 5, 7], 'weights': ['uniform', 'distance'], 'algorithm': ['auto', 'ball_tree', 'kd_tree']}

# Instantiate K-Nearest Neighbors classifier
knn = KNeighborsClassifier()

# Building a 10-fold CV GridSearchCV object
grid_object_KNN = GridSearchCV(estimator=knn, param_grid=grid_params_KNN, scoring='accuracy', cv=10, n_jobs=-1)

# Fitting the grid to the training data
grid_object_KNN.fit(X_train_resampled, Y_train_resampled)
```




```
#Extracting the best parameters and score
print("Best Parameters : ", grid_object_KNN.best_params_)
print("Best_ROC-AUC : ", round(grid_object_KNN.best_score_ * 100, 2))
print("Best model : ", grid_object_KNN.best_estimator_)

#Applying the tuned parameters back to the model
Y_pred = grid_object_KNN.best_estimator_.predict(X_test)
probs = grid_object_KNN.best_estimator_.predict_proba(X_test)
clf_report(Y_test, Y_pred, probs)

kfold = KFold(n_splits=10, random_state=25, shuffle=True)
%time results = cross_val_score(grid_object_KNN.best_estimator_, X_test, Y_test, cv=kfold)
results = results * 100
results = np.round(results,2)
print("Cross Validation Accuracy : ", round(results.mean(), 2))
print("Cross Validation Accuracy in every fold : ", results)
```

```
Best Parameters :  {'algorithm': 'kd_tree', 'n_neighbors': 7, 'weights': 'distance'}
Best_ROC-AUC :  71.45
Best model :  KNeighborsClassifier(algorithm='kd_tree', n_neighbors=7, weights='distance')

 Confusion Matrix
```

1. Activate GPU and Install Dependencies

```
# Activate GPU for faster training by clicking on 'Runtime' > 'Change runtime type' and then selecting GPU as the Hardware accelerator
# Then check if GPU is available
import torch
torch.cuda.is_available()
```

```
# Install required libraries
!pip install datasets transformers huggingface_hub
!apt-get install git-lfs
```

2. Preprocess data

```
# Load data
from datasets import load_dataset
tweet = load_dataset("jos-ger/tweet-sentiment-airlines")
```

```
# Create a smaller training dataset for faster training times
small_train_dataset = tweet["train"].shuffle(seed=42).select([i for i in list(range(3000))])
small_test_dataset = tweet["test"].shuffle(seed=42).select([i for i in list(range(300))])
print(small_train_dataset[0])
print(small_test_dataset[0])
```

```
# Set DistilBERT tokenizer
from transformers import AutoTokenizer
tokenizer = AutoTokenizer.from_pretrained("distilbert-base-uncased")
```

```
# Prepare the text inputs for the model
def preprocess_function(examples):
    return tokenizer(examples["text"], truncation=True)

tokenized_train = small_train_dataset.map(preprocess_function, batched=True)
tokenized_test = small_test_dataset.map(preprocess_function, batched=True)
```

```
# Use data_collector to convert our samples to PyTorch tensors and concatenate them with the correct amount of padding
from transformers import DataCollatorWithPadding
data_collator = DataCollatorWithPadding(tokenizer=tokenizer)
```

3. Training the model

```
# Define DistilBERT as our base model:
from transformers import AutoModelForSequenceClassification
model = AutoModelForSequenceClassification.from_pretrained("distilbert-base-uncased", num_labels=2)
```

```
# Define the evaluation metrics
import numpy as np
from datasets import load_metric

def compute_metrics(eval_pred):
    load_accuracy = load_metric("accuracy")
    load_f1 = load_metric("f1")

    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    accuracy = load_accuracy.compute(predictions=predictions, references=labels)["accuracy"]
    f1 = load_f1.compute(predictions=predictions, references=labels)["f1"]
    return {"accuracy": accuracy, "f1": f1}
```

```
# Log in to your Hugging Face account
# Get your API token here https://huggingface.co/settings/token
from huggingface_hub import notebook_login

notebook_login()
```

```
!pip install transformers[torch]
```

```
!pip install accelerate -U
```

```
# Define a new Trainer with all the objects we constructed so far
from transformers import TrainingArguments, Trainer

repo_name = "finetuning-sentiment-model-tweet-sentiment-3000-samples"

training_args = TrainingArguments(
    output_dir=repo_name,
    learning_rate=2e-5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=2,
    weight_decay=0.01,
    save_strategy="epoch",
    push_to_hub=True,
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_train,
    eval_dataset=tokenized_test,
    tokenizer=tokenizer,
    data_collator=data_collator,
    compute_metrics=compute_metrics,
)


# Train the model
trainer.train()


# Compute the evaluation metrics
trainer.train()
```

4. Analyzing new data with the model

```
# Upload the model to the Hub
trainer.push_to_hub()


# Run inferences with your new model using Pipeline
from transformers import pipeline

sentiment_model = pipeline(model="jos-ger/finetuning-sentiment-model-3000-samples")

sentiment_model(["Flight 236 was great. Fantastic cabin crew. A+ landing", "You neglected to mention the $200 fee per ticket"])
```