twitter-sentimental-analysis

July 24, 2023

1 Twitter Sentimental Analysis

Twitter Sentiment Analysis is the process of computationally identifying and categorizing tweets expressed in a piece of text, especially in order to determine whether the writer's attitude towards a particular topic, product, etc. is positive, negative, or neutral

1.1 Import the Libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Loading dataset

[2]:
```
df=pd.read_csv(r"C:\Users\RBC\Desktop\NLP Project\Tweeter sentiment␣
↪Analysis\Sentiment.csv")
```

[3]:
```
df.head()
```

[3]:
```
   id              candidate  candidate_confidence relevant_yn  \
0   1  No candidate mentioned                   1.0         yes
1   2           Scott Walker                   1.0         yes
2   3  No candidate mentioned                   1.0         yes
3   4  No candidate mentioned                   1.0         yes
4   5           Donald Trump                   1.0         yes

   relevant_yn_confidence sentiment  sentiment_confidence     subject_matter  \
0                     1.0   Neutral                0.6578  None of the above
1                     1.0  Positive                0.6333  None of the above
2                     1.0   Neutral                0.6629  None of the above
3                     1.0  Positive                1.0000  None of the above
4                     1.0  Positive                0.7045  None of the above

   subject_matter_confidence candidate_gold  … relevant_yn_gold  \
0                     1.0000            NaN  …              NaN
```

```
1                    1.0000            NaN  …              NaN
2                    0.6629            NaN  …              NaN
3                    0.7039            NaN  …              NaN
4                    1.0000            NaN  …              NaN

  retweet_count  sentiment_gold subject_matter_gold  \
0             5            NaN                NaN
1            26            NaN                NaN
2            27            NaN                NaN
3           138            NaN                NaN
4           156            NaN                NaN

                                                text tweet_coord  \
0  RT @NancyLeeGrahn: How did everyone feel about…         NaN
1  RT @ScottWalker: Didn't catch the full #GOPdeb…         NaN
2  RT @TJMShow: No mention of Tamir Rice and the …         NaN
3  RT @RobGeorge: That Carly Fiorina is trending …         NaN
4  RT @DanScavino: #GOPDebate w/ @realDonaldTrump…         NaN

              tweet_created            tweet_id  tweet_location  \
0  2015-08-07 09:54:46 -0700  629697200650592256             NaN
1  2015-08-07 09:54:46 -0700  629697199560069120             NaN
2  2015-08-07 09:54:46 -0700  629697199312482304             NaN
3  2015-08-07 09:54:45 -0700  629697197118861312           Texas
4  2015-08-07 09:54:45 -0700  629697196967903232             NaN

               user_timezone
0                      Quito
1                        NaN
2                        NaN
3  Central Time (US & Canada)
4                    Arizona

[5 rows x 21 columns]
```

[4]:
```
#size
df.shape
```

[4]:
```
(13871, 21)
```

[5]:
```
#information
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13871 entries, 0 to 13870
Data columns (total 21 columns):
 #   Column                      Non-Null Count  Dtype
```

```
---  ------                       --------------  -----
 0   id                           13871 non-null  int64
 1   candidate                    13775 non-null  object
 2   candidate_confidence         13871 non-null  float64
 3   relevant_yn                  13871 non-null  object
 4   relevant_yn_confidence       13871 non-null  float64
 5   sentiment                    13871 non-null  object
 6   sentiment_confidence         13871 non-null  float64
 7   subject_matter               13545 non-null  object
 8   subject_matter_confidence    13871 non-null  float64
 9   candidate_gold               28 non-null     object
 10  name                         13871 non-null  object
 11  relevant_yn_gold             32 non-null     object
 12  retweet_count                13871 non-null  int64
 13  sentiment_gold               15 non-null     object
 14  subject_matter_gold          18 non-null     object
 15  text                         13871 non-null  object
 16  tweet_coord                  21 non-null     object
 17  tweet_created                13871 non-null  object
 18  tweet_id                     13871 non-null  int64
 19  tweet_location               9959 non-null   object
 20  user_timezone                9468 non-null   object
dtypes: float64(4), int64(3), object(14)
memory usage: 2.2+ MB
```

[6]:
```
#checking Null values
df.isnull().sum()
```

[6]:
```
id                           0
candidate                   96
candidate_confidence         0
relevant_yn                  0
relevant_yn_confidence       0
sentiment                    0
sentiment_confidence         0
subject_matter             326
subject_matter_confidence    0
candidate_gold           13843
name                         0
relevant_yn_gold         13839
retweet_count                0
sentiment_gold           13856
subject_matter_gold      13853
text                         0
tweet_coord              13850
tweet_created                0
tweet_id                     0
```

```
tweet_location               3912
user_timezone                4403
dtype: int64
```

Since we are doing twitter sentimental analysis to find sentiment(positive, negative or neutral) from given text so, we will consider only required column for futher processKeeping only the necessary column

[7]:
```
df1=df[['sentiment','text']]
```

[8]:
```
df1.head()
```

[8]:
```
  sentiment                                                text
0   Neutral  RT @NancyLeeGrahn: How did everyone feel about…
1  Positive  RT @ScottWalker: Didn't catch the full #GOPdeb…
2   Neutral  RT @TJMShow: No mention of Tamir Rice and the …
3  Positive  RT @RobGeorge: That Carly Fiorina is trending …
4  Positive  RT @DanScavino: #GOPDebate w/ @realDonaldTrump…
```

[9]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13871 entries, 0 to 13870
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   sentiment  13871 non-null  object
 1   text       13871 non-null  object
dtypes: object(2)
memory usage: 216.9+ KB
```

[10]:
```
df1.describe()
```

[10]:
```
       sentiment                                                text
count      13871                                               13871
unique         3                                               10402
top     Negative  RT @RWSurferGirl: Jeb Bush reminds me of eleva…
freq        8493                                                 161
```

[11]:
```
temp = df.groupby('sentiment').count()['text'].reset_index().
  ↪sort_values(by='text',ascending=False)
temp.style.background_gradient(cmap='Purples')
```

[11]:
```
<pandas.io.formats.style.Styler at 0x1171e2a47f0>
```

[12]:
```
#unique
df1['sentiment'].unique()
```

[12]:
```
array(['Neutral', 'Positive', 'Negative'], dtype=object)
```

[13]:
```
#count value
df1['sentiment'].value_counts()
```

[13]:
```
Negative    8493
Neutral     3142
Positive    2236
Name: sentiment, dtype: int64
```

[14]:
```
df1.sentiment.values
```

[14]:
```
array(['Neutral', 'Positive', 'Neutral', …, 'Positive', 'Negative',
       'Positive'], dtype=object)
```

[15]:
```
df1.text.values
```

[15]:
```
array(['RT @NancyLeeGrahn: How did everyone feel about the Climate Change
question last night? Exactly. #GOPDebate',
       "RT @ScottWalker: Didn't catch the full #GOPdebate last night. Here are
some of Scott's best lines in 90 seconds. #Walker16 http://t.co/ZSfF…",
       'RT @TJMShow: No mention of Tamir Rice and the #GOPDebate was held in
Cleveland? Wow.',
       …,
       'RT @Lrihendry: #TedCruz As President, I will always tell the truth, and
do what I said I would do.  #GOPDebates',
       'RT @JRehling: #GOPDebate Donald Trump says that he doesn\'t have time
for political correctness. How does calling women "fat pigs" save him …',
       'RT @Lrihendry: #TedCruz headed into the Presidential Debates. GO TED!!
\n\n#GOPDebates http://t.co/8S67pz8a4A'],
      dtype=object)
```

[16]:
```
df1['Cleaned_text']=''
```

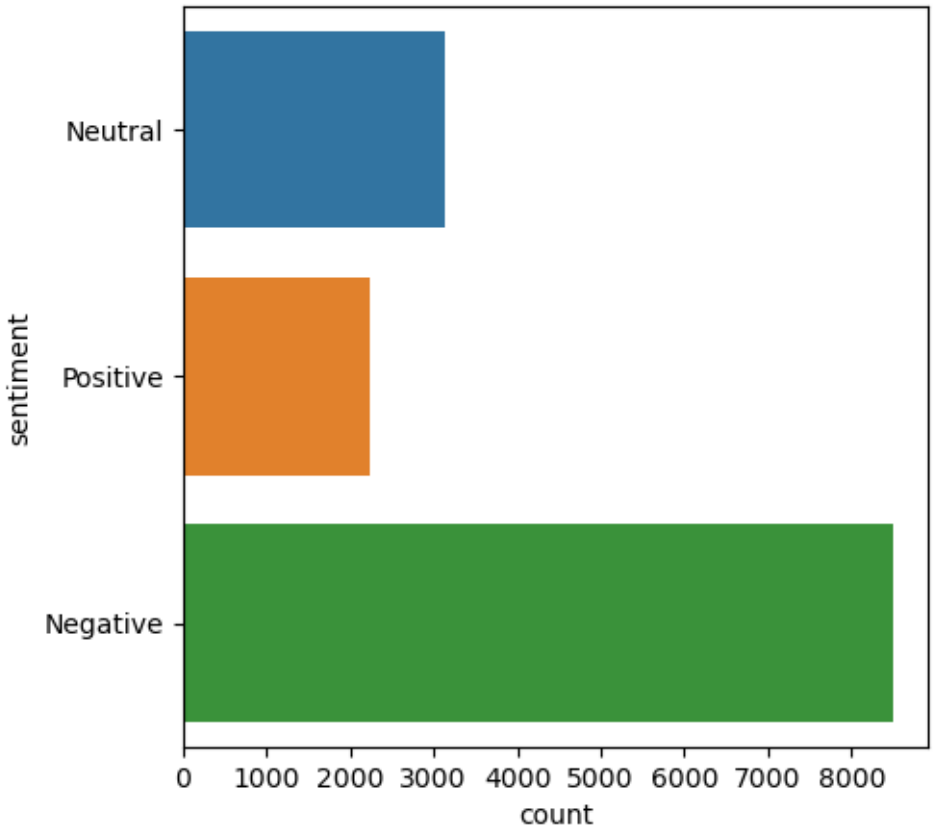
[17]:
```
df1.head()
```

[17]:

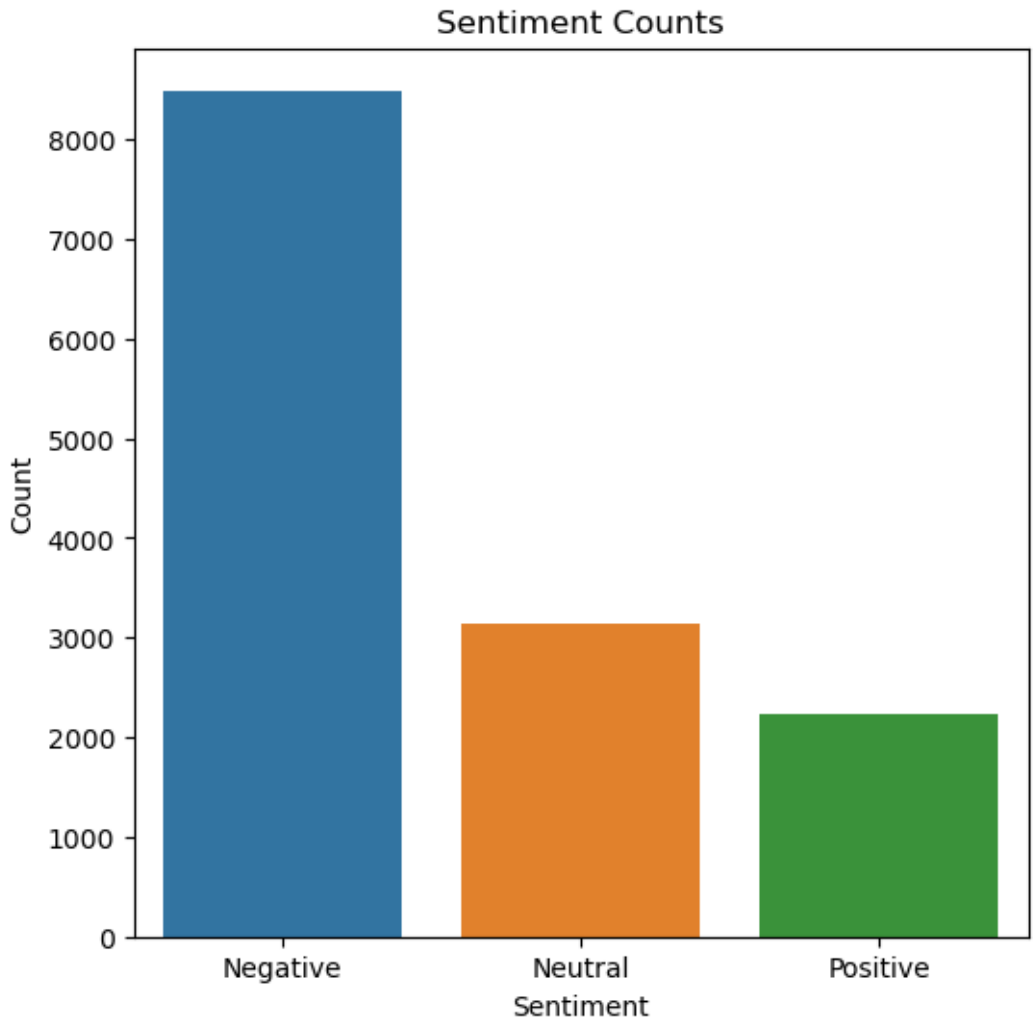
	sentiment	text	Cleaned_text
0	Neutral	RT @NancyLeeGrahn: How did everyone feel about…	
1	Positive	RT @ScottWalker: Didn't catch the full #GOPdeb…	
2	Neutral	RT @TJMShow: No mention of Tamir Rice and the …	
3	Positive	RT @RobGeorge: That Carly Fiorina is trending …	
4	Positive	RT @DanScavino: #GOPDebate w/ @realDonaldTrump…	

1.2 Visualization of dataset

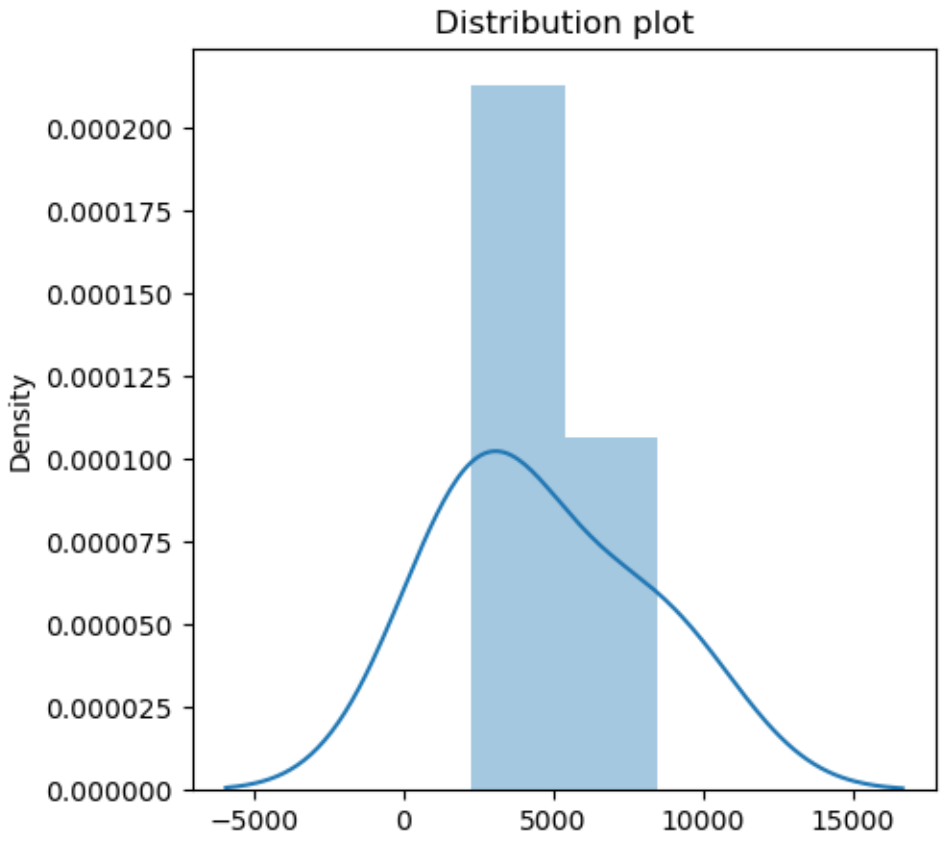
[18]:
```
plt.figure(figsize=(5,5))
sns.countplot(y='sentiment', data=df)
plt.show()
```



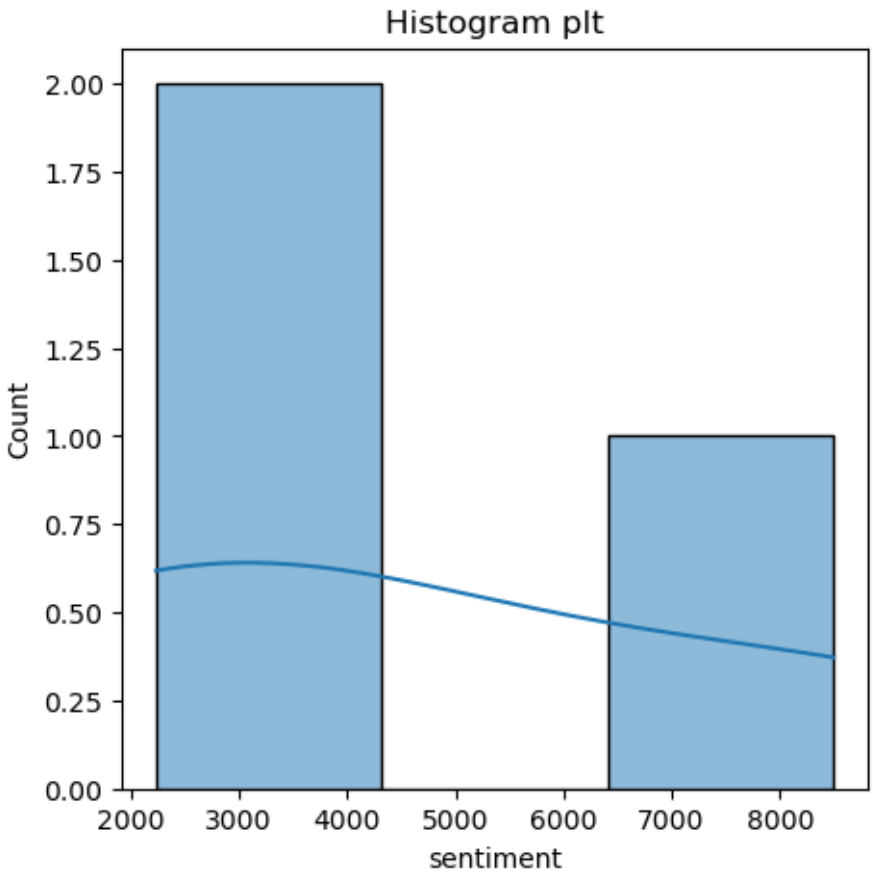
[19]:
```
plt.figure(figsize=(6, 6))
sns.barplot(x=df['sentiment'].value_counts().index, y=df['sentiment'].
 ↪value_counts().values)
plt.title('Sentiment Counts')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.show()
```



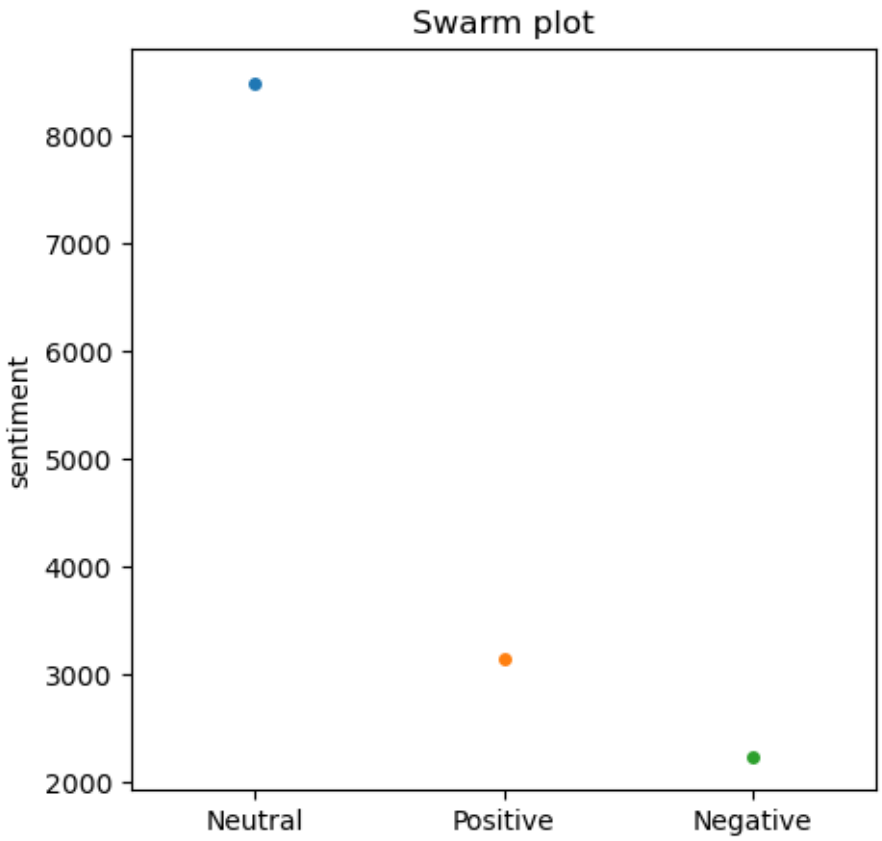
[20]:
```
#Distribution plot
plt.figure(figsize=(5,5))
sns.distplot(x=df['sentiment'].value_counts(),label=df['sentiment'].unique())
plt.title('Distribution plot')
plt.show()
```



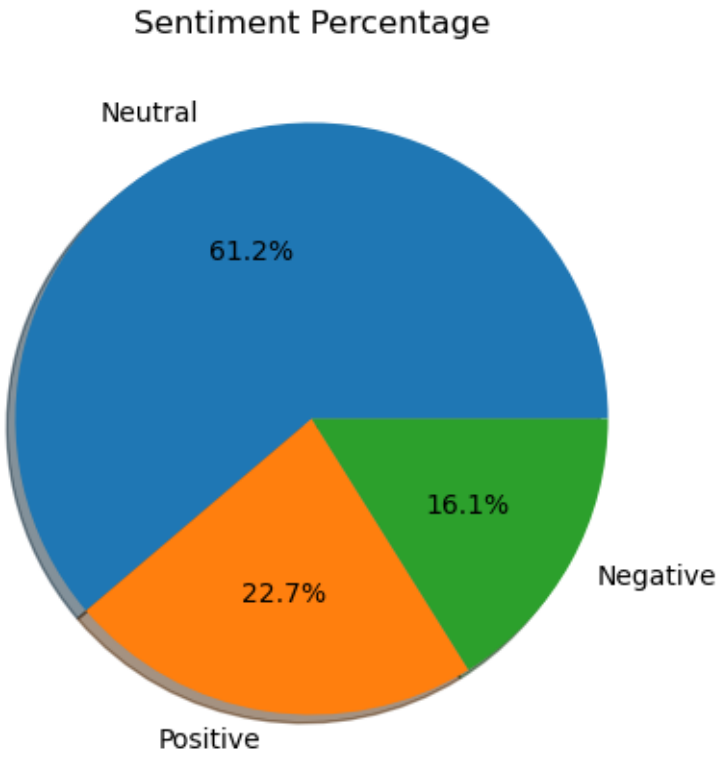
[21]:
```
#histogram plot
plt.figure(figsize=(5,5))
sns.histplot(x=df['sentiment'].value_counts(),label=df['sentiment'].
 ↪unique(),kde=True)
plt.title('Histogram plt')
plt.show()
```



[22]:
```
#swarm plot
plt.figure(figsize=(5,5))
sns.swarmplot(y=df['sentiment'].value_counts(),x=df['sentiment'].
 ↪unique(),data=df)
plt.title('Swarm plot')
plt.show()
```



[23]:
```
#pie chartplot
plt.figure(figsize=(5,5))
plt.pie(df['sentiment'].value_counts(), labels=df['sentiment'].unique(),␣
 ↪autopct='%1.1f%%', shadow=True)
plt.title('Sentiment Percentage')
plt.show()
```



1.2.1 Removing urls,symbols,special words,stopwords,tokenwords

[24]:
```
import re
import nltk
import string
from nltk.corpus import stopwords
stop_words=stopwords.words('english')
stemmer = nltk.SnowballStemmer("english")
def cleanedtext(text):
    text = text.lower()  # Lowercase text
    text = re.sub('https?://\S+|www\.\S+', '', text)  # Remove URLs
    text = re.sub("RT|cc", "", text)  # Remove Twitter elements
    text = re.sub('[^a-zA-Z]', " ", text)  # Remove non-alphabetic characters
    text = re.sub("#\w+", "", text)  # Remove hashtags
    text = re.sub("@\w+", "", text)  # Remove Twitter mentions
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)  # Remove␣
 ↪punctuation
    text = re.sub(r'[^\x00-\x7f]', '', text)  # Remove non-ASCII characters
    text = re.sub('\n', '', text)  # Remove newline characters
    text = re.sub('\w*\d\w*', '', text)  # Remove alphanumeric sequences with␣
 ↪digits
```

```
    text = re.sub('\s+', ' ', text)  # Replace multiple whitespace with a␣
↪single space
    return text
```

[25]:
```
df1['Cleaned_text']=df1.text.apply(lambda x: cleanedtext(x))
```

[26]:
```
df1.head()
```

[26]:
```
  sentiment                                                text  \
0   Neutral  RT @NancyLeeGrahn: How did everyone feel about…
1  Positive  RT @ScottWalker: Didn't catch the full #GOPdeb…
2   Neutral  RT @TJMShow: No mention of Tamir Rice and the …
3  Positive  RT @RobGeorge: That Carly Fiorina is trending …
4  Positive  RT @DanScavino: #GOPDebate w/ @realDonaldTrump…

                                        Cleaned_text
0  rt nancyleegrahn how did everyone feel about t…
1  rt scottwalker didn t catch the full gopdebate…
2  rt tjmshow no mention of tamir rice and the go…
3  rt robgeorge that carly fiorina is trending ho…
4  rt danscavino gopdebate w realdonaldtrump deli…
```

removing stopwords,punctuations

[27]:
```
import re
import nltk
import string
from nltk.corpus import stopwords
from wordcloud import WordCloud
stop_words=stopwords.words('english')
```

[28]:
```
def preprocess(text):
    tokens=[]
    cleaned_word=""
    for word in text.split():
        if word not in stop_words and word not in string.punctuation:
            cleaned_word+=(word+" ")
            tokens.append(word)
            cleaned_word=" ".join(tokens)
    return cleaned_word
```

1.2.2 Most_common Word

[29]:
```
wordfreqdist = nltk.FreqDist(df1['Cleaned_text'])
mostcommon = wordfreqdist.most_common(20)
mostcommon
```

[29]:
```
[('rt rwsurfergirl jeb bush reminds me of elevator music you hear it but you don
t listen gopdebate gopdebates',
  161),
 ('rt rwsurfergirl fox news is obviously trying to influence the makeup of the
republican field gopdebate gopdebates',
  150),
 ('rt rwsurfergirl it is very disappointing that fox news is not conducting a
fair amp balanced debate gopdebate gopdebates',
  142),
 ('rt rwsurfergirl i think cruz and trump need to band together and expose this
set up job and get rid of bush and rubio gopdebate g ',
  140),
 ('rt rwsurfergirl we the american people pick the next president of united
states not fox news gopdebate gopdebates',
  133),
 ('rt rwsurfergirl the candidates don t have to attack realdonaldtrump fox is
doing it for them by stoping him from speaking gopdebate ',
  127),
 ('rt rwsurfergirl is it just me or does anyone else want to punch chris wallace
in the face gopdebate gopdebates ',
  124),
 ('rt rwsurfergirl fox is cherry picking the candidates jeb gets the softball
questions gopdebates gopdebates',
  116),
 ('rt rwsurfergirl why doesn t chris wallace ask the other politicans about
their finances and where their money comes from gopdebate ',
  113),
 ('rt rwsurfergirl so megynkelly posed for adult pictures should we bring that
up gopdebate gopdebates',
  98),
 ('rt rwsurfergirl thanks fox news you re raising realdonaldtrump s ratings
gopdebate gopdebates',
  88),
 ('rt ericstonestreet trump has cam hands gopdebates', 77),
 ('rt rwsurfergirl ask trump a legitimate question look at wallace s face when
trump nails it gopdebate gopdebates',
  74),
 ('rt larryelder trump should have said megyn ask these nine candidates if they
plan to support me when i win the nomination gopdebat ',
  68),
 ('rt lrihendry tedcruz as president i will always tell the truth and do what i
said i would do gopdebates',
  65),
 ('rt rwsurfergirl you would never know realdonaldtrump is the frontrunner from
watching this debate gopdebate gopdebates',
  65),
 ('rt rwsurfergirl i d really like to see how long each candidate was given to
```

```
speak it didn t seem fair amp balanced at all gopdebat ',
  60),
 ('rt rwsurfergirl fox news won t admit who the republican leader is right now i
mean realdonaldtrump only has a double digit lead gopd ',
  59),
 ('rt jamiaw the purpose of the military is to kill people and break things mike
huckabee gopdebates kkkorgop',
  58),
 ('rt donniewahlberg enjoyed the gopdebates and am looking forward to the
democraticdebates next ',
  56)]
```

[30]:
```
from collections import Counter
df1['Cleaned_text'] = df1['Cleaned_text'].apply(lambda x:str(x).split())
top = Counter([item for sublist in df1['Cleaned_text'] for item in sublist])
temp = pd.DataFrame(top.most_common(10))
temp.columns = ['Common_words','count']
temp.style.background_gradient(cmap='Blues')
```

[30]:
```
<pandas.io.formats.style.Styler at 0x11722e8d0a0>
```

[31]:
```
import plotly.express as px
fig = px.bar(temp, x="count", y="Common_words", title='Commmon Words in␣
 ↪Selected Text', orientation='h',
             width=700, height=700,color='Common_words')
fig.show()
```

[32]:
```
fig = px.treemap(temp, path=['Common_words'], values='count',title='Tree of␣
 ↪Most Common Words')
fig.show()
```

[33]:
```
# Removing neutral sentiments
df1 = df1[df1.sentiment != "Neutral"]
```

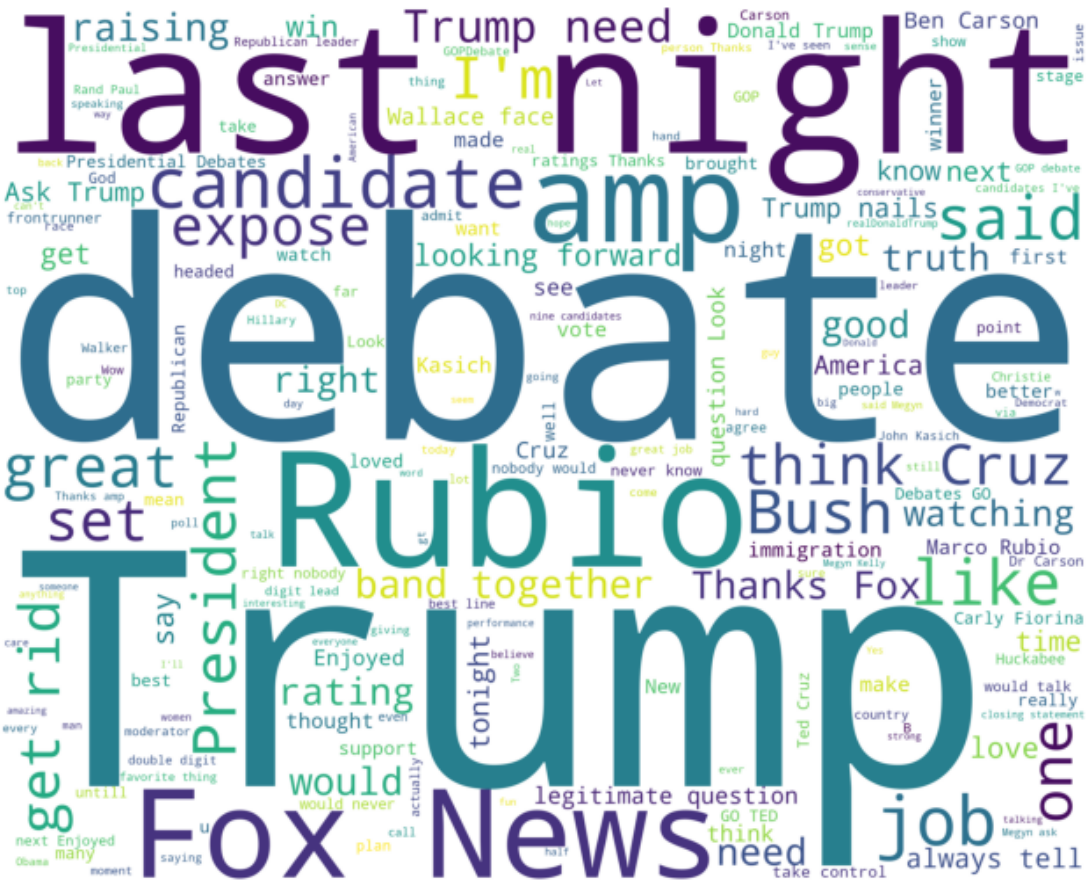
[34]:
```
tweet_pos = df1[ df1['sentiment'] == 'Positive']
tweet_pos = tweet_pos['text']
tweet_neg = df1[ df1['sentiment'] == 'Negative']
tweet_neg = tweet_neg['text']

def wordcloud_draw(df1, color = 'black'):
    words = ' '.join(df1)
    cleaned_word = " ".join([word for word in words.split()
                             if 'http' not in word
                                 and not word.startswith('@')
                                 and not word.startswith('#')
                                 and word != 'RT'
                             ])
```

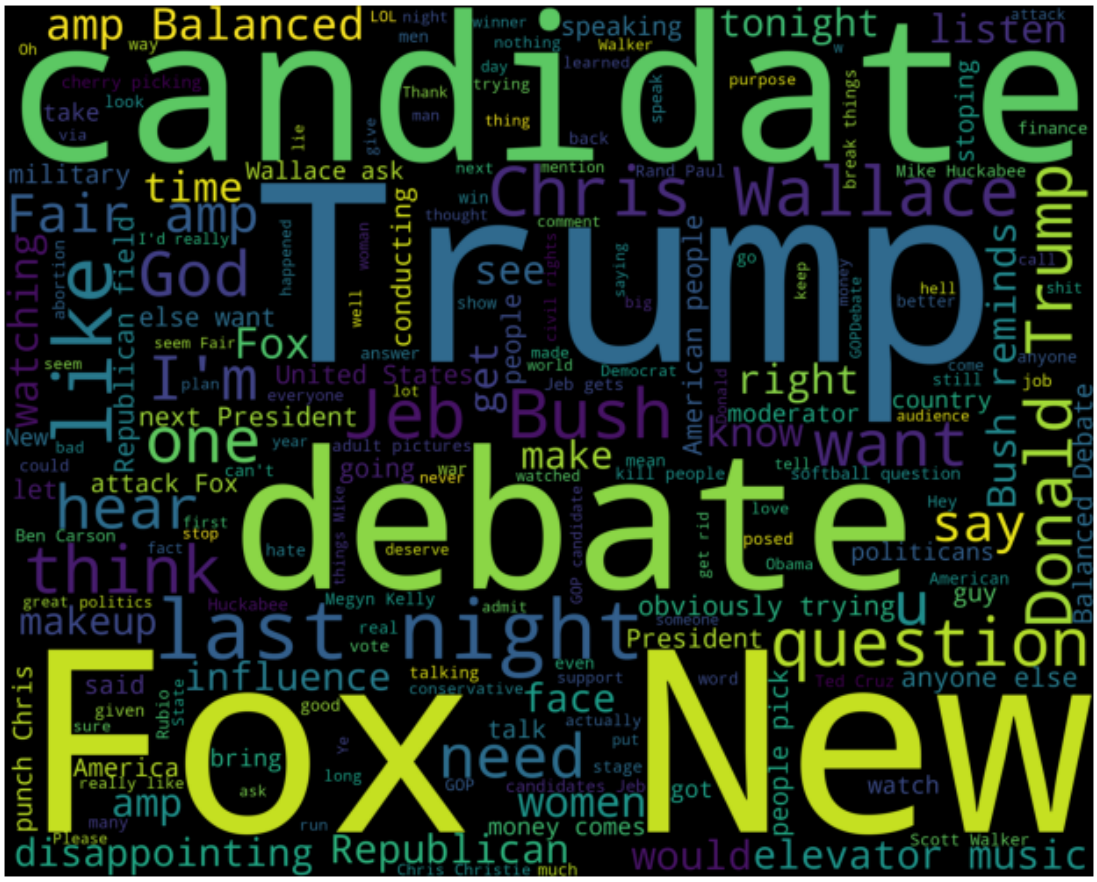
```
    wordcloud = WordCloud(stopwords=stop_words,
                          background_color=color,
                          width=2500,
                          height=2000
                         ).generate(cleaned_word)
    plt.figure(1,figsize=(8, 8))
    plt.imshow(wordcloud)
    plt.axis('off')
    plt.show()

print("Positive words")
wordcloud_draw(tweet_pos,'white')
print("Negative words")
wordcloud_draw(tweet_neg)
```

```
Positive words
```




```
Negative words
```



[35]:
```
df1.head()
```

[35]:

	sentiment	text	\
1	Positive	RT @ScottWalker: Didn't catch the full #GOPdeb…	
3	Positive	RT @RobGeorge: That Carly Fiorina is trending …	
4	Positive	RT @DanScavino: #GOPDebate w/ @realDonaldTrump…	
5	Positive	RT @GregAbbott_TX: @TedCruz: "On my first day …	
6	Negative	RT @warriorwoman91: I liked her and was happy …	

	Cleaned_text
1	[rt, scottwalker, didn, t, catch, the, full, g…
3	[rt, robgeorge, that, carly, fiorina, is, tren…
4	[rt, danscavino, gopdebate, w, realdonaldtrump…
5	[rt, gregabbott, tx, tedcruz, on, my, first, d…
6	[rt, warriorwoman, i, liked, her, and, was, ha…

[36]:
```
df1.sentiment.unique()
```

[36]:
```
array(['Positive', 'Negative'], dtype=object)
```

Spliting into X and Y

Now I will convert these words into categorical values:

[37]:
```
from sklearn.preprocessing import LabelEncoder
var_mod = ['sentiment']
le=LabelEncoder()
```

[38]:
```
for i in var_mod:
    df1[i]=le.fit_transform(df1[i])
```

[ ]:
```

```

1.3 Training Machine Learning Model for Tweet Sentiment

1.4 Train_Test_Split

[39]:
```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from scipy.sparse import hstack
```

[40]:
```
required_text=df1['Cleaned_text'].values
required_target=df1['sentiment'].values
```

[41]:
```
full_text= [' '.join(words) for words in required_text]
word_vectorizer=TfidfVectorizer(sublinear_tf=True,
    stop_words='english',
    max_features=1500)
word_vectorizer.fit(full_text)
wordfeatures=word_vectorizer.transform(full_text)
```

[42]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test =␣
 ↪train_test_split(wordfeatures,required_target,random_state=0, test_size=0.2)
print(X_train.shape)
print(X_test.shape)
```

```
(8583, 1500)
(2146, 1500)
```

[43]:
```
y_train.shape,y_test.shape
```

[43]:
```
((8583,), (2146,))
```

[44]:
```
from sklearn.naive_bayes import MultinomialNB
from sklearn.multiclass import OneVsRestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
```

```
from sklearn.metrics import classification_report
from sklearn import metrics
```

[46]:
```
model_vectorizer= MultinomialNB().fit(X_train, y_train)
prediction=model_vectorizer.predict(X_test)
print(confusion_matrix(y_test,prediction))
print (classification_report(y_test, prediction))
```

```
[[1631   53]
 [ 277  185]]
              precision    recall  f1-score   support

           0       0.85      0.97      0.91      1684
           1       0.78      0.40      0.53       462

    accuracy                           0.85      2146
   macro avg       0.82      0.68      0.72      2146
weighted avg       0.84      0.85      0.83      2146
```

[47]:
```
!pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\programdata\anaconda3\lib\site-
packages (1.7.6)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-
packages (from xgboost) (1.9.1)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-
packages (from xgboost) (1.24.3)
```

[48]:
```
import xgboost as xgb
xgb_model=xgb.XGBClassifier(
        learning_rate=0.1,
        max_depth=7,
        n_estimators=80,
        use_label_encoder=False,
        eval_metric='auc' )
```

[49]:
```
xgb_model_vectorizer = xgb_model.fit(X_train, y_train)
xgb_predictions_vectorizer=xgb_model_vectorizer.predict(X_test)
print(confusion_matrix(y_test,xgb_predictions_vectorizer))
print (classification_report(y_test, xgb_predictions_vectorizer))
```

```
[[1639   45]
 [ 293  169]]
              precision    recall  f1-score   support

           0       0.85      0.97      0.91      1684
           1       0.79      0.37      0.50       462
```

```
    accuracy                           0.84      2146
   macro avg       0.82      0.67      0.70      2146
weighted avg       0.84      0.84      0.82      2146
```

Thank You