In [*]:

```
import pandas as pd
import numpy as np
import string
import spacy
import seaborn as sns
from matplotlib.pyplot import imread
from matplotlib import pyplot as plt
from wordcloud import WordCloud
%matplotlib inline
#import plotly as py
#import cufflinks as cf
#from plotly.offline import iplot
from nltk.corpus import stopwords
from textblob import TextBlob
from textblob import Word
import nltk
#nltk.download('stopwords')
#nltk.download('wordnet')
```

In [*]:

```
'''''''''''
import csv,requests
from bs4 import BeautifulSoup
import emoji


no_of_reviews = input("Enter no of reviews: ")
reviews_num = int(no_of_reviews)
URL = input("Enter URL: ")

header = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTM

#If reviews are less than input reviews this code qwill be usdefull

html = requests.get(URL,headers= header)
soup = BeautifulSoup(html.content,features="html.parser")
#reviews=[soup.find_all("q", {"class": "IRsGHoPm"})[i].span.string for i in range(5)]
avail = [item.get_text(strip=True) for item in soup.select("span._33O9dg0j")]
avail =avail[0].split(" reviews")[0].split(",")
b=""
for d in avail:
    b=b+d
    avail = int(b)

if avail < reviews_num:
    print("Choose URL which has more 5000 reviews ")
    print("Available review(s)", avail)
    exit

from bs4 import BeautifulSoup
q=0
import requests
with open('Reviews.csv','w') as f:
    write = csv.writer(f)
    write.writerow(['REVIEWS'])
    try:
      URL.split()
      x = URL.split("-Reviews-", 1)
      for i in range(5,reviews_num+1,5):
        URL = x[0]+ f"-Reviews-or{i}-"+x[1]
        html = requests.get(URL,headers= header)
        soup = BeautifulSoup(html.content,features="html.parser")
        reviews=[soup.find_all("q", {"class": "IRsGHoPm"})[i].span.string for i in range(5)

        for j in range(len(reviews)):

          reviews[j]=str(reviews[j])

          reviews[j] = emoji.demojize(reviews[j], delimiters=("", ""))

          write.writerow([reviews[j]])
          q+=1
    except Exception as ex:
        template = "An exception of type {0} occurred. Arguments:\n{1!r}"
        message = template.format(type(ex).__name__, ex.args)
        print (message)

# print("User input: ", reviews_num )
# print("Extracted reviews: ", q )
```

```
# print("Omitted reviews:",reviews_num-q)


'''''''''''''''''
```

In [*]:

```
#!pip install nltk
```

In [*]:

```
#!pip install pandas
```

In [*]:

```
#py.offline.init_notebook_mode(connected=True)
#cf.go_offline()
```

In [4]:

```
import pandas as pd
reviews_df = pd.read_csv('Reviews.csv')
print(reviews_df)
print("Total Reviwes Extracted:", len(reviews_df))
```

```
                                                REVIEWS
0     Over all good experience.  My Special thanks t...
1     If you expect Royal Experience then This is no...
2     My room was sea view, though on a lower floor ...
3     We were booked at Taj Tower and was upgraded d...
4     It was just amazing. Went with my family to di...
...                                                 ...
7975  We were in Mumbai for a few days and stayed at...
7976  We stayed at the Taj Mahal for 4 nights.  When...
7977  I have stayed both in the heritage wing and th...
7978  I have stayed at the Taj several times while v...
7979  We had a double deluxe room with a water view ...

[7980 rows x 1 columns]
Total Reviwes Extracted: 7980
```

In [5]:

```
data=reviews_df.rename({'REVIEWS':'reviews'},axis=1 )
data.head()
```

Out[5]:

	reviews
0	Over all good experience. My Special thanks t...
1	If you expect Royal Experience then This is no...
2	My room was sea view, though on a lower floor ...
3	We were booked at Taj Tower and was upgraded d...
4	It was just amazing. Went with my family to di...

In [6]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7980 entries, 0 to 7979
Data columns (total 1 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   reviews  7980 non-null   object
dtypes: object(1)
memory usage: 62.5+ KB
```

In [7]:

```
stop = stopwords.words('english')
#data = data.head(500)
data['stopwords'] = data['reviews'].apply(lambda x: len([x for x in x.split() if x in stop]
#data[['reviews','stopwords']].head(12)
```

In [8]:

```
data['stopwords'].sum()
```

Out[8]:

```
284095
```

In [9]:

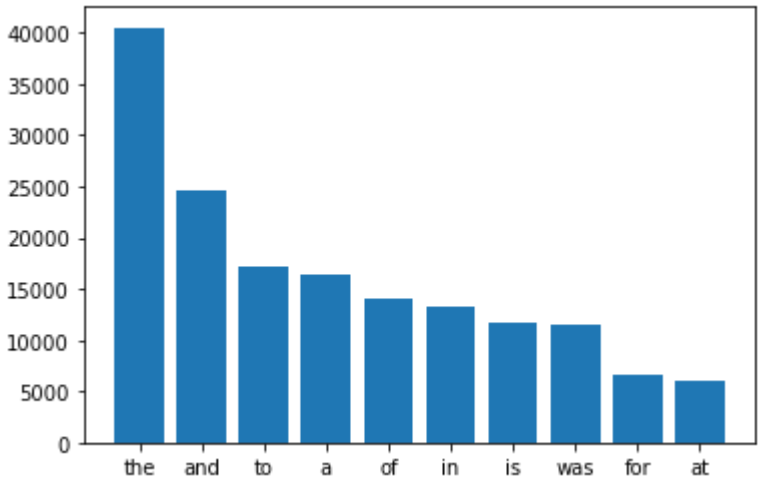
```
corpus=[]
df= data['reviews'].str.split()
df=df.values.tolist()
corpus=[word for x in df for word in x]

from collections import defaultdict
dic=defaultdict(int)
for word in corpus:
    if word in stop:
        dic[word]+=1
```

In [10]:

```
top=sorted(dic.items(), key=lambda x:x[1],reverse=True)[:10]
x,y=zip(*top)
plt.bar(x,y)
```

Out[10]:

```
<BarContainer object of 10 artists>
```



We can evidently see that stopwords such as “the”, and” and “to” dominate in reviews.

In [11]:

```
#Number of Words in single review
data['word_count'] = data['reviews'].apply(lambda x: len(str(x).split(" ")))
#data[['reviews','word_count']].head()
```

In [12]:

```
#Number of characters in single review including spaces
data['char_count'] = data['reviews'].str.len()
#data[['reviews','char_count']].head()
```

In [13]:

```
def get_avg_word_len(x):
    words=x.split()
    word_len=0
    for word in words:
        word_len=word_len+len(word)

    return word_len/len(word)
```

In [14]:

```
data['avg_word_len']=data['reviews'].apply(lambda x:get_avg_word_len(x))
#data[['reviews','avg_word_len']].head()
```

In [15]:

```
data.head()
```

Out[15]:

	reviews	stopwords	word_count	char_count	avg_word_len
0	Over all good experience. My Special thanks t...	36	102	562	230.500000
1	If you expect Royal Experience then This is no...	25	74	413	30.909091
2	My room was sea view, though on a lower floor ...	25	74	413	340.000000
3	We were booked at Taj Tower and was upgraded d...	49	115	625	46.454545
4	It was just amazing. Went with my family to di...	30	66	357	97.333333

In [16]:

```
#replacing special characters with " "
data['reviews'] = data['reviews'].str.replace('[^\w\s\'\"]','')
data['reviews'].head()
```

```
<ipython-input-16-9d7fce53f341>:2: FutureWarning: The default value of regex
will change from True to False in a future version.
  data['reviews'] = data['reviews'].str.replace('[^\w\s\'\"]','')
```

Out[16]:

```
0    Over all good experience  My Special thanks to...
1    If you expect Royal Experience then This is no...
2    My room was sea view though on a lower floor w...
3    We were booked at Taj Tower and was upgraded d...
4    It was just amazing Went with my family to din...
Name: reviews, dtype: object
```

In [17]:

```
#Lowercasing
data['reviews'] = data['reviews'].apply(lambda x: " ".join(x.lower() for x in x.split()))
data['reviews'].head()
```

Out[17]:

```
0    over all good experience my special thanks to ...
1    if you expect royal experience then this is no...
2    my room was sea view though on a lower floor w...
3    we were booked at taj tower and was upgraded d...
4    it was just amazing went with my family to din...
Name: reviews, dtype: object
```

In [18]:

```
#Removing stopwords
sw = stopwords.words('english')
data['reviews'] =data['reviews'].apply(lambda x: " ".join(x for x in x.split() if x not in
data['reviews'].head()
```

Out[18]:

```
0    good experience special thanks narmita taking ...
1    expect royal experience place see crowd 200 pe...
2    room sea view though lower floor requested hig...
3    booked taj tower upgraded due taj inner circle...
4    amazing went family dine indian restaurant amb...
Name: reviews, dtype: object
```

In [19]:

```
#tokenization
#nltk.download('punkt')
```

In [20]:

```
hotel_review=np.array(data['reviews'])
hr=str(hotel_review)
hr
```

Out[20]:

'[\'good experience special thanks narmita taking care room dining food good
work punctual atul taking time assisted internet facility virender welcome a
ssistance staff courteous prompt punctual attending everything need good exp
erience food nice overall pleasant stay would recommend hotel anyone visitin
g mumbai would like stay heart city thank namrata atul virender welcome assi
stance team\'\n \'expect royal experience place see crowd 200 people waiting
check separate checkin palace room ok restaurants crowded poor sitting facil
ities wait 45 minutes order welcome garlands drink called heritage tour aver
age payment better go lake palace udaipur royal experience\'\n \'room sea vi
ew though lower floor requested higher one view great nevertheless architect
somewhat conventional though sea view windows floor ceiling best effect othe
rwise room great room service house keeping perfect expected taj great staff
cheerful well trained definitely coming back\'\n ...\n "stayed heritage wing
new tower heritage wing class bathroom world room massive even check desk up
per class however year stayed new wing carpets shabby bathroom nice home hum
ble house lovely pool smashing pool boys return can\'t afford heritage wing
booked heritage honeymoon given single beds discourteous receptionist unders
tand since fired lovely hotel really need give tower overhaul breakfasts how
ever amazing like brunch"\n \'stayed taj several times visiting bombay wonde
rful location right next gateway india waterfront problem past 2 years hotel
jacked prices almost 300 making poor value compared hotels sprung city hotel
like 5 star hotels bombay follows twotier pricing policy one foreign nationa
ls indian citizens prices foreign nationals 50 100 higher priced dollars usu
ally try get indian citizen rate possible booking travel agent company servi
ce provided good almost everything taj offer overpriced\'\n \'double deluxe
room water view tower total 3 nights one night beginning 2 week trip two nig
hts end unfortunately expectations met expecting anything business travelers
hotel stay first "deluxe" room showed us claimed remodeled dingy couch small
stained large paint cracks walls also odor room like second room showed us b
righter larger bed smelled better even though smoking room first room way to
ld honeymoon could offer us romantic room even extra\']'

In [21]:

```
from nltk.tokenize import word_tokenize
text_tokens = word_tokenize(hr)
print(text_tokens[:500])
```

```
['[', "'good", 'experience', 'special', 'thanks', 'narmita', 'taking', 'car
e', 'room', 'dining', 'food', 'good', 'work', 'punctual', 'atul', 'taking',
'time', 'assisted', 'internet', 'facility', 'virender', 'welcome', 'assistan
ce', 'staff', 'courteous', 'prompt', 'punctual', 'attending', 'everything',
'need', 'good', 'experience', 'food', 'nice', 'overall', 'pleasant', 'stay',
'would', 'recommend', 'hotel', 'anyone', 'visiting', 'mumbai', 'would', 'lik
e', 'stay', 'heart', 'city', 'thank', 'namrata', 'atul', 'virender', 'welcom
e', 'assistance', "team'", "'expect", 'royal', 'experience', 'place', 'see',
'crowd', '200', 'people', 'waiting', 'check', 'separate', 'checkin', 'palac
e', 'room', 'ok', 'restaurants', 'crowded', 'poor', 'sitting', 'facilities',
'wait', '45', 'minutes', 'order', 'welcome', 'garlands', 'drink', 'called',
'heritage', 'tour', 'average', 'payment', 'better', 'go', 'lake', 'palace',
'udaipur', 'royal', "experience'", "'room", 'sea', 'view', 'though', 'lowe
r', 'floor', 'requested', 'higher', 'one', 'view', 'great', 'nevertheless',
'architect', 'somewhat', 'conventional', 'though', 'sea', 'view', 'windows',
'floor', 'ceiling', 'best', 'effect', 'otherwise', 'room', 'great', 'room',
'service', 'house', 'keeping', 'perfect', 'expected', 'taj', 'great', 'staf
f', 'cheerful', 'well', 'trained', 'definitely', 'coming', "back'", '...',
'``', 'stayed', 'heritage', 'wing', 'new', 'tower', 'heritage', 'wing', 'cla
ss', 'bathroom', 'world', 'room', 'massive', 'even', 'check', 'desk', 'uppe
r', 'class', 'however', 'year', 'stayed', 'new', 'wing', 'carpets', 'shabb
y', 'bathroom', 'nice', 'home', 'humble', 'house', 'lovely', 'pool', 'smashi
ng', 'pool', 'boys', 'return', 'ca', "n't", 'afford', 'heritage', 'wing', 'b
ooked', 'heritage', 'honeymoon', 'given', 'single', 'beds', 'discourteous',
'receptionist', 'understand', 'since', 'fired', 'lovely', 'hotel', 'really',
'need', 'give', 'tower', 'overhaul', 'breakfasts', 'however', 'amazing', 'li
ke', 'brunch', "'", "'stayed", 'taj', 'several', 'times', 'visiting', 'bomb
ay', 'wonderful', 'location', 'right', 'next', 'gateway', 'india', 'waterfro
nt', 'problem', 'past', '2', 'years', 'hotel', 'jacked', 'prices', 'almost',
'300', 'making', 'poor', 'value', 'compared', 'hotels', 'sprung', 'city', 'h
otel', 'like', '5', 'star', 'hotels', 'bombay', 'follows', 'twotier', 'prici
ng', 'policy', 'one', 'foreign', 'nationals', 'indian', 'citizens', 'price
s', 'foreign', 'nationals', '50', '100', 'higher', 'priced', 'dollars', 'usu
ally', 'try', 'get', 'indian', 'citizen', 'rate', 'possible', 'booking', 'tr
avel', 'agent', 'company', 'service', 'provided', 'good', 'almost', 'everyth
ing', 'taj', 'offer', "overpriced'", "'double", 'deluxe', 'room', 'water',
'view', 'tower', 'total', '3', 'nights', 'one', 'night', 'beginning', '2',
'week', 'trip', 'two', 'nights', 'end', 'unfortunately', 'expectations', 'me
t', 'expecting', 'anything', 'business', 'travelers', 'hotel', 'stay', 'firs
t', '``', 'deluxe', "'", 'room', 'showed', 'us', 'claimed', 'remodeled', 'd
ingy', 'couch', 'small', 'stained', 'large', 'paint', 'cracks', 'walls', 'al
so', 'odor', 'room', 'like', 'second', 'room', 'showed', 'us', 'brighter',
'larger', 'bed', 'smelled', 'better', 'even', 'though', 'smoking', 'room',
'first', 'room', 'way', 'told', 'honeymoon', 'could', 'offer', 'us', 'romant
ic', 'room', 'even', 'extra', "'", ']']
```

In [22]:

```
#Stemming
'''from nltk.stem import PorterStemmer
st = PorterStemmer()
data['reviews'][:5].apply(lambda x: " ".join([st.stem(word) for word in x.split()]))'''
```

Out[22]:

```
'from nltk.stem import PorterStemmer\nst = PorterStemmer()\ndata[\'reviews
\'][:5].apply(lambda x: " ".join([st.stem(word) for word in x.split()]))'
```

In [23]:

```
#Lemmatization
data['reviews'] = data['reviews'].apply(lambda x: " ".join([Word(word).lemmatize() for word
data['reviews'].head()
```

Out[23]:

```
0    good experience special thanks narmita taking ...
1    expect royal experience place see crowd 200 pe...
2    room sea view though lower floor requested hig...
3    booked taj tower upgraded due taj inner circle...
4    amazing went family dine indian restaurant amb...
Name: reviews, dtype: object
```

In [24]:

```
data.head()
```

Out[24]:

	reviews	stopwords	word_count	char_count	avg_word_len
0	good experience special thanks narmita taking ...	36	102	562	230.500000
1	expect royal experience place see crowd 200 pe...	25	74	413	30.909091
2	room sea view though lower floor requested hig...	25	74	413	340.000000
3	booked taj tower upgraded due taj inner circle...	49	115	625	46.454545
4	amazing went family dine indian restaurant amb...	30	66	357	97.333333

In [25]:

```
#for removing undesirable words with higher frequency
list=("hotel","us","taj","mumbai","india","mahal","would","every","u","made","palace")
data['reviews'] =data['reviews'].apply(lambda x: " ".join(x for x in x.split() if x not in
data['reviews'].head()
```

Out[25]:

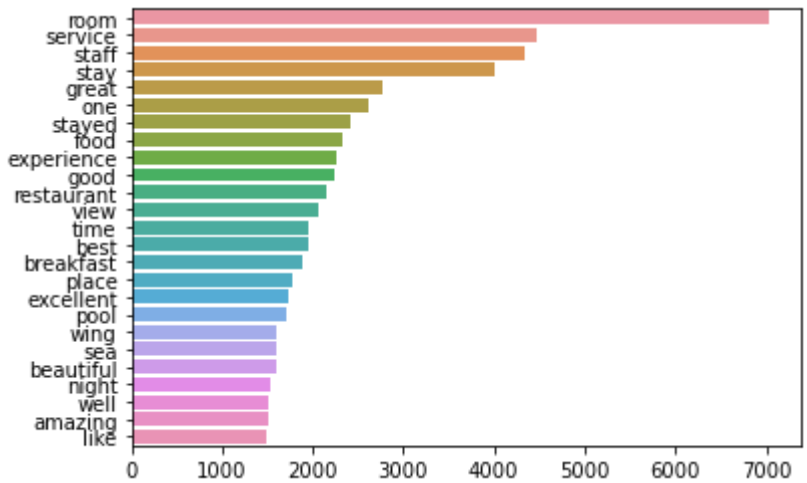
```
0    good experience special thanks narmita taking ...
1    expect royal experience place see crowd 200 pe...
2    room sea view though lower floor requested hig...
3    booked tower upgraded due inner circle member ...
4    amazing went family dine indian restaurant amb...
Name: reviews, dtype: object
```

In [26]:

```
from collections import Counter
def plot_top_non_stopwords_barchart(x):
    stop=set(stopwords.words('english'))

    new= data['reviews'].str.split()
    new=new.values.tolist()
    corpus=[word for i in new for word in i]

    counter=Counter(corpus)
    most=counter.most_common()
    x, y=[], []
    for word,count in most[:25]:
        if (word not in stop):
            x.append(word)
            y.append(count)

    sns.barplot(x=y,y=x)
```

In [27]:

```
plot_top_non_stopwords_barchart(data['reviews'])
```



In [28]:

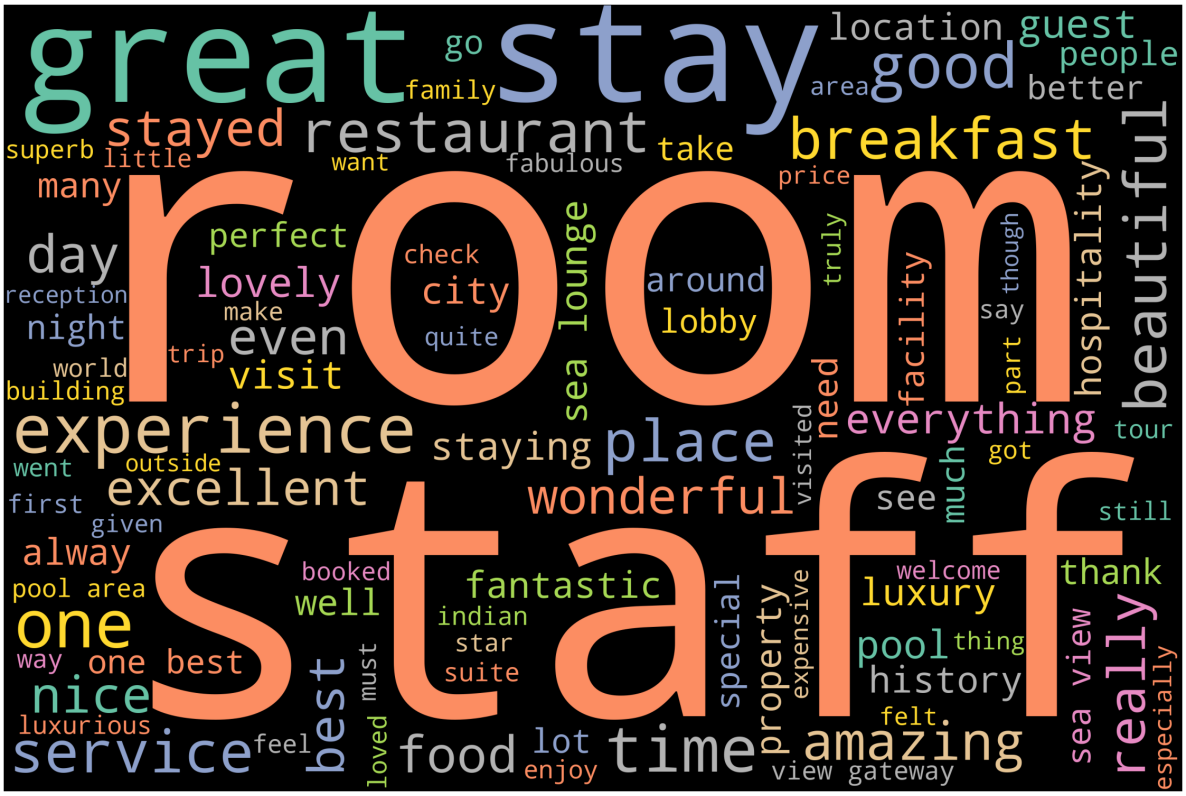
```
# Joining the list into one string/text
text = ' '.join(data['reviews'])
#text
```

In [29]:

```
# Import packages
import matplotlib.pyplot as plt
%matplotlib inline
from wordcloud import WordCloud, STOPWORDS
# Define a function to plot word cloud
def plot_cloud(wordcloud):
    # Set figure size
    plt.figure(figsize=(40, 30))
    # Display image
    plt.imshow(wordcloud)
    # No axis details
    plt.axis("off");
```

In [30]:

```
# Generate wordcloud
stopwords = STOPWORDS
stopwords.add('will')
wordcloud = WordCloud(width = 3000, height = 2000, background_color='black', max_words=100,
# Plot
plot_cloud(wordcloud)
```



In [31]:

```
#sentiment analysis
data['sentiment_polarity'] = data['reviews'].apply(lambda x: TextBlob(x).sentiment.polarity
data[['reviews','sentiment_polarity']].head(10)
```

Out[31]:

	reviews	sentiment_polarity
0	good experience special thanks narmita taking ...	0.562771
1	expect royal experience place see crowd 200 pe...	0.250000
2	room sea view though lower floor requested hig...	0.437013
3	booked tower upgraded due inner circle member ...	-0.018056
4	amazing went family dine indian restaurant amb...	0.308995
5	room service could faster complained assumed s...	-0.083333
6	room image everywhere including website incorr...	0.141000
7	bought sea view room iconic disappoint room fa...	0.472000
8	hotelits come lot historyand experience 1 wond...	0.375000
9	extremely comfortable stay can't say food meal...	0.313095

In [32]:

```
def getAnalysis(score):
  if score < 0:
    return 'Negative'
  elif score == 0:
    return 'Neutral'
  else:
    return 'Positive'
data ['sentiment'] = data  ['sentiment_polarity'].apply(getAnalysis )
data[['reviews','sentiment','sentiment_polarity']].head(20)
```

Out[32]:

	reviews	sentiment	sentiment_polarity
0	good experience special thanks narmita taking ...	Positive	0.562771
1	expect royal experience place see crowd 200 pe...	Positive	0.250000
2	room sea view though lower floor requested hig...	Positive	0.437013
3	booked tower upgraded due inner circle member ...	Negative	-0.018056
4	amazing went family dine indian restaurant amb...	Positive	0.308995
5	room service could faster complained assumed s...	Negative	-0.083333
6	room image everywhere including website incorr...	Positive	0.141000
7	bought sea view room iconic disappoint room fa...	Positive	0.472000
8	hotelits come lot historyand experience 1 wond...	Positive	0.375000
9	extremely comfortable stay can't say food meal...	Positive	0.313095
10	stay nice first really high expectation legacy...	Positive	0.281619
11	heritage property possibly best stayed family ...	Positive	0.480159
12	best hospitality ever truly feel home outside ...	Positive	0.416667
13	begin check experienc smooth staff accommodati...	Positive	0.466667
14	absolutely relaxing thoroughly efficient servi...	Positive	0.432000
15	excellency everything checkin till checkout fr...	Positive	0.433750
16	establishment witnessed empowered customer fac...	Positive	0.275850
17	best property country gave experience lifetime...	Positive	0.498571
18	family stay one night week end 1112jul nice fo...	Positive	0.222656
19	everything perfect except food food quality ma...	Positive	0.258333

In [33]:

```
count=data['sentiment'].value_counts()
count
```

Out[33]:

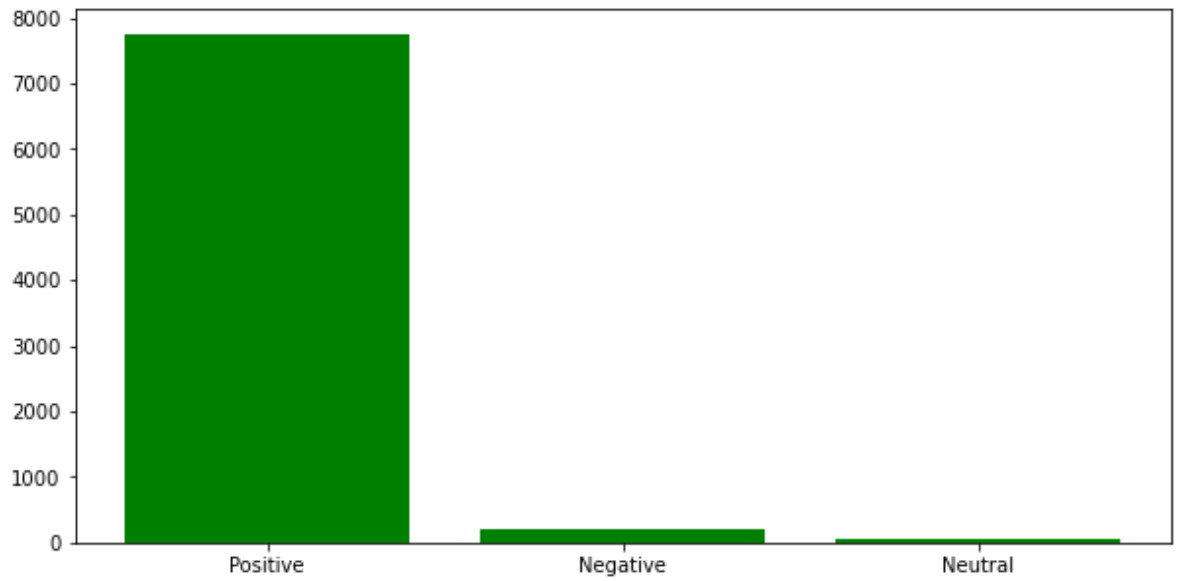
```
Positive    7749
Negative     189
Neutral       42
Name: sentiment, dtype: int64
```

In [34]:

```
import matplotlib.pyplot as plt

fig=plt.figure(figsize=(10,5))

senti=['Positive','Negative','Neutral']
plt.bar(senti,count,color='g')
```

Out[34]:

```
<BarContainer object of 3 artists>
```



Feature Extractiion

count vectoriser tells the frequency of a word.

In [79]:

```
#!pip install sklearn
```

In [35]:

```
# count vectoriser tells the frequency of a word.
from sklearn.feature_extraction.text import CountVectorizer
import numpy as np
vectorizer = CountVectorizer(min_df = 1, max_df = 0.9)
X = vectorizer.fit_transform(data["reviews"])
word_freq_df = pd.DataFrame({'reviews': vectorizer.get_feature_names(), 'occurrences':np.as
word_freq_df['frequency'] = word_freq_df['occurrences']/np.sum(word_freq_df['occurrences'])
#print(word_freq_df.sort('occurrences',ascending = False).head())
```

In [36]:

```
word_freq_df
```

Out[36]:

	reviews	occurrences	frequency
0	001	1	0.000003
1	0030hrs	1	0.000003
2	01	1	0.000003
3	0130	1	0.000003
4	0140am	1	0.000003
...	...	...	...
17459	풀사이드에서	1	0.000003
17460	한잔	1	0.000003
17461	항상	1	0.000003
17462	해도	1	0.000003
17463	호텔에서도	1	0.000003

17464 rows × 3 columns

TFIDF - Term frequency inverse Document Frequencyt

In [37]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(stop_words='english', max_features= 1000, max_df = 0.5, smooth
doc_vec = vectorizer.fit_transform(data["reviews"])
names_features = vectorizer.get_feature_names()
dense = doc_vec.todense()
denselist = dense.tolist()
df = pd.DataFrame(denselist, columns = names_features)
```

In [38]:

```
df
```

Out[38]:

	10	100	12	15	1st	20	2008	24	30	5pm	...	working	world	worth
0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.000000
1	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.000000
2	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.000000
3	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.124222	0.000000
4	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.154454	0.170798
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7975	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.000000
7976	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.098901
7977	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.097560	0.000000
7978	0.0	0.196756	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.000000
7979	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.000000

7980 rows × 1000 columns

Word2Vec

In [39]:

```
#!pip install python-Levenshtein
```

In [40]:

```
import gensim
```

```
C:\Users\cricl\anaconda3\lib\site-packages\gensim\similarities\__init__.py:1
5: UserWarning: The gensim.similarities.levenshtein submodule is disabled, b
ecause the optional Levenshtein package <https://pypi.org/project/python-Lev
enshtein/> is unavailable. Install Levenhstein (e.g. `pip install python-Lev
enshtein`) to suppress this warning.
  warnings.warn(msg)
```

In [41]:

```
#!pip install gensim
```

In [42]:

```
review_text = data.reviews.apply(gensim.utils.simple_preprocess)
```

In [43]:

```
review_text
```

Out[43]:

```
0       [good, experience, special, thanks, narmita, t...
1       [expect, royal, experience, place, see, crowd,...
2       [room, sea, view, though, lower, floor, reques...
3       [booked, tower, upgraded, due, inner, circle, ...
4       [amazing, went, family, dine, indian, restaura...
                              ...
7975    [day, stayed, tower, part, everything, great, ...
7976    [stayed, night, arrived, given, deluxe, room, ...
7977    [stayed, heritage, wing, new, tower, heritage,...
7978    [stayed, several, time, visiting, bombay, wond...
7979    [double, deluxe, room, water, view, tower, tot...
Name: reviews, Length: 7980, dtype: object
```

In [44]:

```
model_w2v = gensim.models.Word2Vec(
    window=10,
    min_count=2,
    workers=4,
)
```

In [45]:

```
model_w2v.build_vocab(review_text, progress_per=1000)
```

In [46]:

```
model_w2v.train(review_text, total_examples=model_w2v.corpus_count, epochs=model_w2v.epochs
```

Out[46]:

```
(1346097, 1570310)
```

In [47]:

```
#model.wv.most_similar("luxury")
```

In [48]:

```
#model.wv.similarity(w1="luxury", w2="expensive")
```

In [49]:

```
#model.wv.similarity(w1="excellent", w2="service")
```

N-gram

In [50]:

```
#Bi-gram
def get_top_n2_words(corpus, n=None):
    vec1 = CountVectorizer(ngram_range=(2,2),  #for tri-gram, put ngram_range=(3,3)
            max_features=2000).fit(corpus)
    bag_of_words = vec1.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in
                  vec1.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1],
                reverse=True)
    return words_freq[:n]
```

In [51]:

```
top2_words = get_top_n2_words(data["reviews"], n=200) #top 200
top2_df = pd.DataFrame(top2_words)
top2_df.columns=["Bi-gram", "Freq"]
top2_df.head()
```

Out[51]:

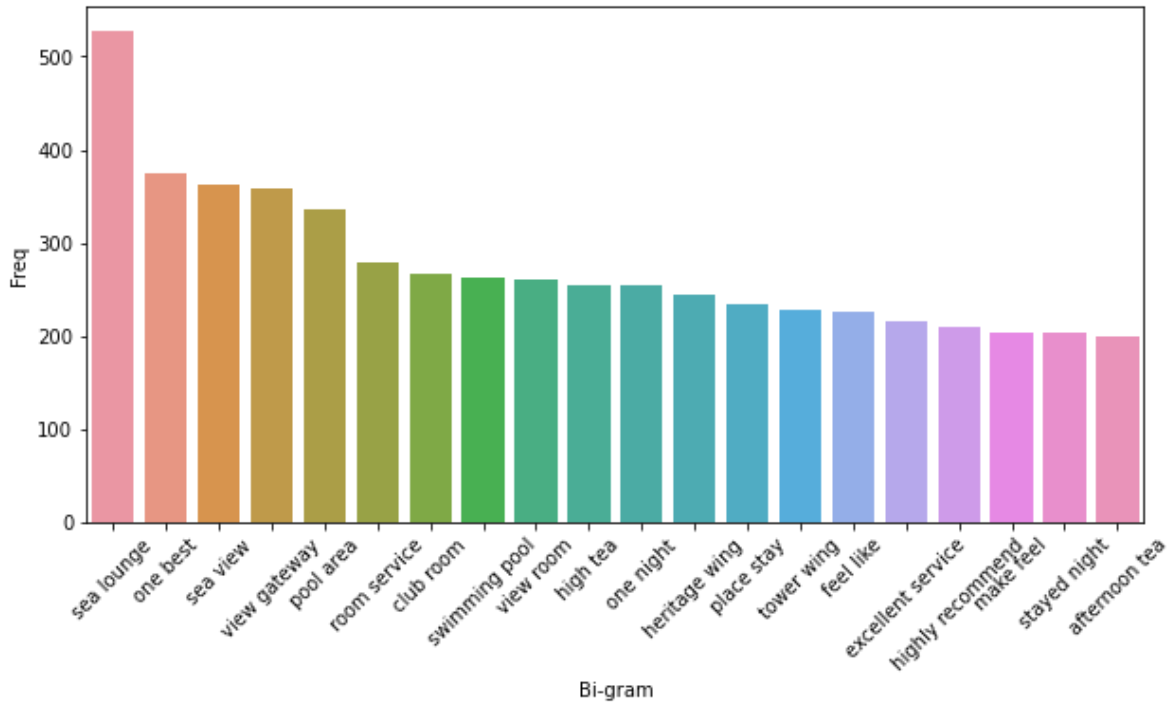
	Bi-gram	Freq
0	sea lounge	528
1	one best	374
2	sea view	363
3	view gateway	358
4	pool area	336

In [52]:

```
#Bi-gram plot
import matplotlib.pyplot as plt
import seaborn as sns
top20_bigram = top2_df.iloc[0:20,:]
fig = plt.figure(figsize = (10, 5))
plot=sns.barplot(x=top20_bigram["Bi-gram"],y=top20_bigram["Freq"])
plot.set_xticklabels(rotation=45,labels = top20_bigram["Bi-gram"])
```

Out[52]:

```
[Text(0, 0, 'sea lounge'),
 Text(1, 0, 'one best'),
 Text(2, 0, 'sea view'),
 Text(3, 0, 'view gateway'),
 Text(4, 0, 'pool area'),
 Text(5, 0, 'room service'),
 Text(6, 0, 'club room'),
 Text(7, 0, 'swimming pool'),
 Text(8, 0, 'view room'),
 Text(9, 0, 'high tea'),
 Text(10, 0, 'one night'),
 Text(11, 0, 'heritage wing'),
 Text(12, 0, 'place stay'),
 Text(13, 0, 'tower wing'),
 Text(14, 0, 'feel like'),
 Text(15, 0, 'excellent service'),
 Text(16, 0, 'highly recommend'),
 Text(17, 0, 'make feel'),
 Text(18, 0, 'stayed night'),
 Text(19, 0, 'afternoon tea')]
```



In [53]:

```
#Tri-gram
def get_top_n3_words(corpus, n=None):
    vec1 = CountVectorizer(ngram_range=(3,3),
           max_features=2000).fit(corpus)
    bag_of_words = vec1.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in
                  vec1.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1],
                reverse=True)
    return words_freq[:n]
```

In [54]:

```
top3_words = get_top_n3_words(data["reviews"], n=200)
top3_df = pd.DataFrame(top3_words)
top3_df.columns=["Tri-gram", "Freq"]
```

In [55]:

```
top3_df
```

Out[55]:

	Tri-gram	Freq
0	breakfast sea lounge	145
1	sea view room	128
2	nothing much trouble	121
3	treated like royalty	83
4	old world charm	77
...	...	...
195	wish could stayed	12
196	upgraded heritage wing	12
197	great pool area	12
198	life time experience	11
199	sea lounge good	11

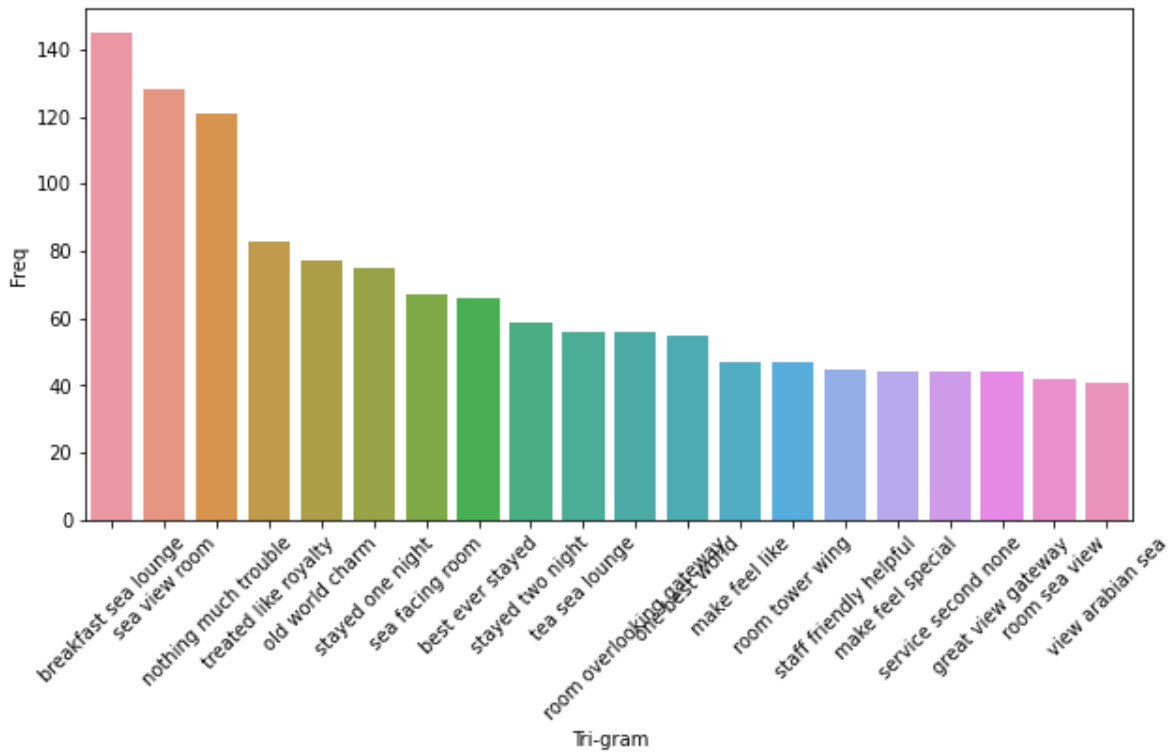
200 rows × 2 columns

In [56]:

```
#Tri-gram plot
import seaborn as sns
top20_trigram = top3_df.iloc[0:20,:]
fig = plt.figure(figsize = (10, 5))
plot=sns.barplot(x=top20_trigram["Tri-gram"],y=top20_trigram["Freq"])
plot.set_xticklabels(rotation=45,labels = top20_trigram["Tri-gram"])
```

Out[56]:

```
[Text(0, 0, 'breakfast sea lounge'),
 Text(1, 0, 'sea view room'),
 Text(2, 0, 'nothing much trouble'),
 Text(3, 0, 'treated like royalty'),
 Text(4, 0, 'old world charm'),
 Text(5, 0, 'stayed one night'),
 Text(6, 0, 'sea facing room'),
 Text(7, 0, 'best ever stayed'),
 Text(8, 0, 'stayed two night'),
 Text(9, 0, 'tea sea lounge'),
 Text(10, 0, 'room overlooking gateway'),
 Text(11, 0, 'one best world'),
 Text(12, 0, 'make feel like'),
 Text(13, 0, 'room tower wing'),
 Text(14, 0, 'staff friendly helpful'),
 Text(15, 0, 'make feel special'),
 Text(16, 0, 'service second none'),
 Text(17, 0, 'great view gateway'),
 Text(18, 0, 'room sea view'),
 Text(19, 0, 'view arabian sea')]
```



Applying naive bayes for classification

In [57]:

```
data.head()
```

Out[57]:

	reviews	stopwords	word_count	char_count	avg_word_len	sentiment_polarity	sentiment
0	good experience special thanks narmita taking ...	36	102	562	230.500000	0.562771	Positive
1	expect royal experience place see crowd 200 pe...	25	74	413	30.909091	0.250000	Positive
2	room sea view though lower floor requested hig...	25	74	413	340.000000	0.437013	Positive
3	booked tower upgraded due inner circle member ...	49	115	625	46.454545	-0.018056	Negative
4	amazing went family dine indian restaurant amb...	30	66	357	97.333333	0.308995	Positive

In [58]:

```
def split_into_words(x):
    return (x.split(" "))
```

In [59]:

```
from sklearn.model_selection import train_test_split

reviews_train,reviews_test = train_test_split(data,test_size=0.3)
```

In [60]:

```
reviews_test.head(10)
```

Out[60]:

	reviews	stopwords	word_count	char_count	avg_word_len	sentiment_polarity	senti
6766	highly recommend staying definitely one 'the b...	24	73	421	49.857143	0.277692	Pos
3917	wife stayed holiday everything spectacular mom...	50	121	657	107.400000	0.500000	Pos
4265	tourist vacation visiting last stop spent thre...	20	49	275	45.400000	0.293333	Pos
4734	ended trip stayed old part true grand dame ser...	71	147	701	42.692308	0.245455	Pos
2960	wonderful stay staff went way ensure little de...	20	40	210	24.428571	0.415278	Pos
2241	far unbelievable world far outweighs experienc...	54	138	733	99.333333	0.219866	Pos
2522	initial noise problem immediately competently ...	21	47	261	71.666667	0.325000	Pos
4240	stayed many year ago honeymoon year later terr...	73	152	786	317.500000	0.300595	Pos
3185	overall great stay second stay time staying ol...	47	112	609	83.000000	0.380303	Pos
7449	staff extremely helpful service oriented found...	11	28	159	16.500000	-0.312500	Neg

In [61]:

```
# Preparing email texts into word count matrix format
reviews_bow = CountVectorizer(analyzer=split_into_words).fit(data.reviews)
```

In [62]:

```
# For all reviews
all_reviews_matrix = reviews_bow.transform(data.reviews)
all_reviews_matrix.shape
```

Out[62]:

```
(7980, 18242)
```

Using TFIDF

In [63]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
matrix = vectorizer.fit_transform(data['reviews'])
```

In [64]:

```
X = matrix
```

In [65]:

```
from sklearn import preprocessing
# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()
# Encode labels in column 'Country'.
Y  = label_encoder.fit_transform(data['sentiment'])
```

In [66]:

```
Y
```

Out[66]:

```
array([2, 2, 2, ..., 2, 2, 2])
```

In [67]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=100)
```

In [68]:

```
Y_test
```

Out[68]:

```
array([2, 2, 2, ..., 2, 2, 2])
```

In [69]:

```
Y_train
```

Out[69]:

```
array([2, 2, 2, ..., 2, 2, 2])
```

In [70]:

```
#!pip install imblearn
```

In [76]:

```
from imblearn.over_sampling import RandomOverSampler
over_sampler = RandomOverSampler()
X_res, y_res = over_sampler.fit_resample(X_train, Y_train)
#print(f"Training target statistics: {Counter(y_res)}")
#print(f"Testing target statistics: {Counter(y_test)}")
print("After Oversampling the shape of X_train:{}".format(X_res.shape))
print("After Oversampling the shape of y_train: {} \n".format(y_res.shape))
```

```
After Oversampling the shape of X_train:(16263, 17464)
After Oversampling the shape of y_train: (16263,)
```

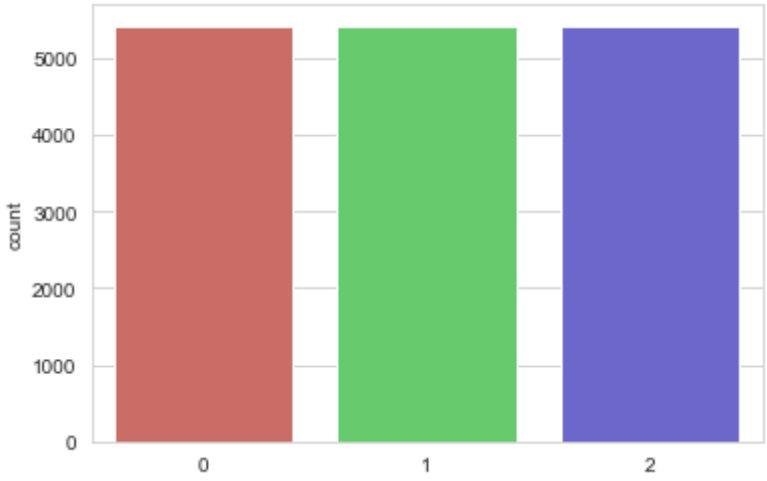
In [77]:

```
import seaborn as sns
sns.set_style("whitegrid")
sns.countplot(y_res, palette = "hls")
```

```
C:\Users\cricl\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[77]:

```
<AxesSubplot:ylabel='count'>
```



In [81]:

```
from imblearn.over_sampling import SMOTE
sm = SMOTE(k_neighbors=1)
X_res, y_res = sm.fit_resample(X_train, Y_train.ravel())
print("After Oversampling the shape of X_train:{}".format(X_res.shape))
print("After Oversampling the shape of y_train: {} \n".format(y_res.shape))
```

```
After Oversampling the shape of X_train:(16263, 17464)
After Oversampling the shape of y_train: (16263,)
```

NAIVE BAYES

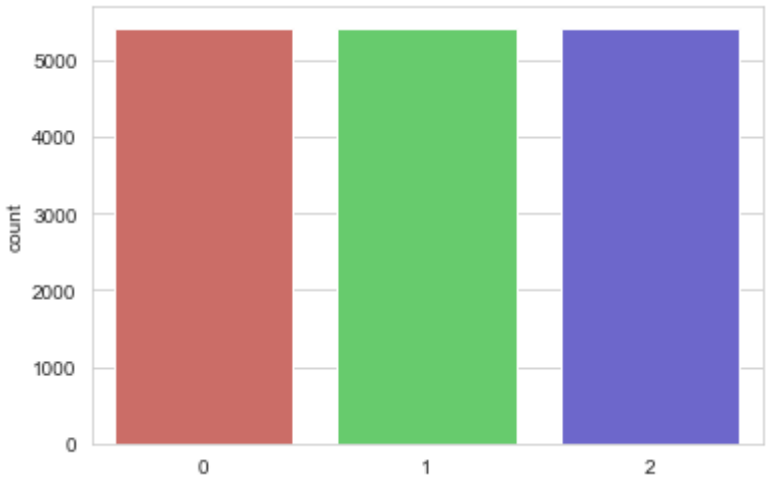
In [82]:

```
import seaborn as sns
sns.set_style("whitegrid")
sns.countplot(y_res, palette = "hls")
```

```
C:\Users\cricl\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[82]:

```
<AxesSubplot:ylabel='count'>
```



In [83]:

```
from sklearn.naive_bayes import MultinomialNB as MB
from sklearn.naive_bayes import GaussianNB as GB

# Multinomial Naive Bayes
classifier_mb = MB()
classifier_mb.fit(X_res,y_res)
```

Out[83]:

```
MultinomialNB()
```

In [84]:

```
train_pred_m = classifier_mb.predict(X_res)
accuracy_train_m = np.mean(train_pred_m==y_res)

test_pred_m = classifier_mb.predict(X_res)
accuracy_test_m = np.mean(test_pred_m==y_res)
```

In [85]:

```
accuracy_train_m
```

Out[85]:

```
0.9961876652524134
```

In [86]:

```
accuracy_test_m
```

Out[86]:

```
0.9961876652524134
```

In [94]:

```
classifier_gb = GB()
classifier_gb.fit(X_res.toarray(),y_res) # we need to convert tfidf into array format which
train_pred_g = classifier_gb.predict(X_res.toarray())
accuracy_train_g = np.mean(train_pred_g==y_res)
test_pred_g = classifier_gb.predict(X_test.toarray())
accuracy_test_g = np.mean(test_pred_g==y_res)
```

```
<ipython-input-94-f92cdbffbd8f>:6: DeprecationWarning: elementwise compariso
n failed; this will raise an error in the future.
  accuracy_test_g = np.mean(test_pred_g==y_res)
```

In [95]:

```
print( accuracy_train_g ,accuracy_test_g)
```

```
0.9976634077353502 0.0
```

In [96]:

```
from sklearn.metrics import classification_report
print(classification_report(Y_test, test_pred_g))
```

```
              precision    recall  f1-score   support

           0       0.03      0.02      0.02        55
           1       0.00      0.00      0.00        11
           2       0.97      0.99      0.98      2328

    accuracy                           0.96      2394
   macro avg       0.33      0.33      0.33      2394
weighted avg       0.95      0.96      0.95      2394
```

KNN CLASSIFICATION

In [97]:

```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report,multilabel_confusion_matrix
```

In [98]:

```
n_neighbors = np.array(range(1,40,2))
param_grid = dict(n_neighbors=n_neighbors)
```

In [99]:

```
model_knn = KNeighborsClassifier()
model_knn.fit(X_res,y_res.ravel())
grid = GridSearchCV(estimator=model_knn, param_grid=param_grid)
grid.fit(X_res, y_res.ravel())
y_pred  = model_knn.predict(X_test)
```

In [100]:

```
#from sklearn.metrics import f1_score
#print("Precision Score : ",precision_score(Y_test, y_pred,pos_label='positive'average='mic
#print("Recall Score : ",recall_score(Y_test, y_pred,pos_label='positive'average='micro'))'
#metrics.f1_score(Y_test, y_pred, labels=np.unique(y_pred),average='micro')''''
```

In [101]:

```
print(classification_report(Y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.03      0.91      0.07        55
           1       0.01      0.55      0.01        11
           2       1.00      0.01      0.02      2328

    accuracy                           0.03      2394
   macro avg       0.35      0.49      0.03      2394
weighted avg       0.97      0.03      0.02      2394
```

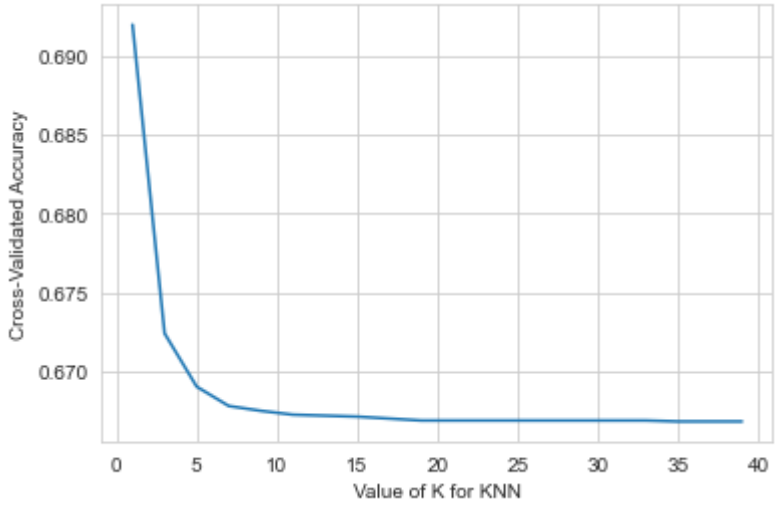
In [102]:

```
print(grid.best_score_)
print(grid.best_params_)
```

```
0.6898481069040632
{'n_neighbors': 1}
```

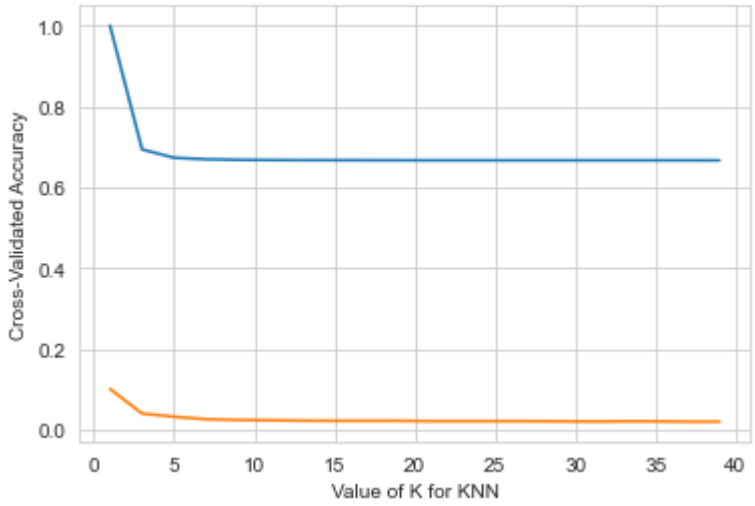
In [103]:

```
import matplotlib.pyplot as plt
%matplotlib inline
# choose k between 1 to 41
k_range = range(1, 40,2)
k_scores = []
# use iteration to caclulator different k in models, then return the average accuracy based
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X_res , y_res.ravel(), cv=10)
    k_scores.append(scores.mean())
# plot to see clearly
plt.plot(k_range, k_scores)
plt.xlabel('Value of K for KNN')
plt.ylabel('Cross-Validated Accuracy')
plt.show()
```



In [104]:

```
import matplotlib.pyplot as plt
%matplotlib inline
# choose k between 1 to 41
k_range = range(1, 40,2)
k_scores_train = []
k_scores_test = []
# use iteration to caclulator different k in models, then return the average accuracy based
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_res,y_res.ravel())
    pred= knn.predict(X_res)
    score = np.mean(y_res.ravel() == pred)
    k_scores_train.append(score)
    pred_test = knn.predict(X_test)
    score_test = np.mean(Y_test == pred_test)
    k_scores_test.append(score_test)
# plot to see clearly
plt.plot(k_range, k_scores_train)
plt.plot(k_range, k_scores_test)
plt.xlabel('Value of K for KNN')
plt.ylabel('Cross-Validated Accuracy')
plt.show()
```



Random Forest Model

In [105]:

```
#Import Random Forest Model
from sklearn.ensemble import RandomForestClassifier

#Create a Gaussian Classifier
clf_RFC=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf_RFC.fit(X_res,y_res.ravel())

y_pred=clf_RFC.predict(X_test)
```

In [106]:

```
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
kfold = KFold(n_splits=10, random_state=7, shuffle=True)
results = cross_val_score(clf_RFC, X_res,y_res.ravel() , cv=kfold)
print(results.mean())
```

```
0.9929905552896955
```

SUPPORT VECTOR MACHINE

In [107]:

```
from sklearn import svm
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
```

In [*]:

```
clf = SVC()
param_grid = [{'kernel':['rbf'],'gamma':[50,5,10,0.5],'C':[15,14,13,12,11,10,0.1,0.001] }]
gsv = GridSearchCV(clf,param_grid,cv=10)
gsv.fit(X_res,y_res)
```

In [*]:

```
gsv.best_params_ , gsv.best_score_
```

In [*]:

```
from sklearn.metrics import accuracy_score, confusion_matrix
clf = SVC(kernel='rbf', C= 15, gamma = 0.5)
clf.fit(X_res,Y_res)
y_pred = clf.predict(X_test)
acc = accuracy_score(y_test, y_pred) * 100
print("Accuracy =", acc)
confusion_matrix(y_test, y_pred)
```

In [*]:

```
clf = SVC()
param_grid = [{'kernel':['poly'],'C':[15,14,13,12,11,10,0.1,0.001] }]
gsv = GridSearchCV(clf,param_grid,cv=10)
gsv.fit(X_train_sm,y_train_sm)
```

In [*]:

```
gsv.best_params_ , gsv.best_score_
```

In [*]:

```
from sklearn.metrics import accuracy_score, confusion_matrix
clf = SVC(C=14, kernel='poly')
clf.fit(X_train_sm,y_train_sm)
y_pred = clf.predict(X_test)
acc = accuracy_score(Y_test, y_pred) * 100
print("Accuracy =", acc)
confusion_matrix(Y_test, y_pred)
```

In [ ]:

```

```

In [ ]:

```

```