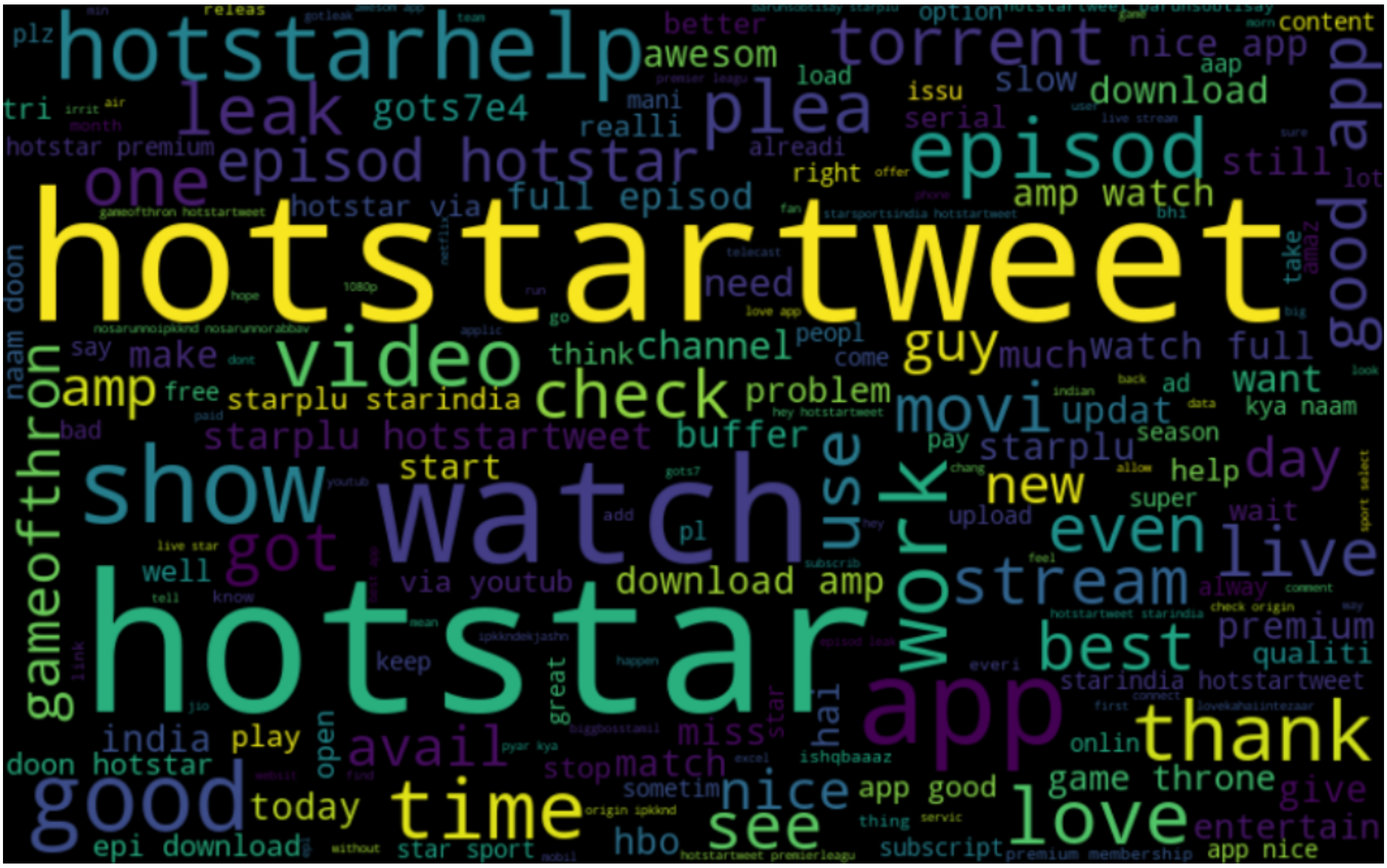
HOTSTAR REVIEWS SENTIMENT ANALYSIS- NLTK & CLASSIFICATION ALGORITHMS



Sentiment Analysis

- Also known as opinion mining or emotion AI is the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information.
- It is used to find out the polarity of the text, which is positive, negative or neutral.

Objective

- The objective is to analyze the sentiment of Hotstar reviews using Natural Language Toolkit(NLTK) alongwith Classification algorithms.

Dataset brief

- The dataset contains Hotstar reviews an Indian subscription video on-demand over-the-top streaming service owned by The Walt Disney Company India. The dataset is divided into ID, Username, Created_Date, Reviews,Lower_Case_Reviews, Sentiment_Manual_BP, Sentiment_Manual, Review_Length, DataSource, Year, Month, Date & Sentiment_Polarity.

Import the libraries

In [1]:

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

Load & read the dataset

In [2]:

```
df=pd.read_csv(r"C:\Users\manme\Documents\Priya\Stats and ML\Dataset\hotstar_reviews.csv")
df.head()
```

Out[2]:

	ID	UserName	Created_Date	Reviews	Lower_Case_Reviews	Sentiment_Manual_BP	Sentime
0	1	NaN	08-10-2017	Hh	hh	Negative	
1	2	NaN	08-11-2017	No	no	Negative	
2	3	asadynwa	08-12-2017	@hotstar_helps during paymnt for premium subsc...	@hotstar_helps during paymnt for premium subsc...	Help	
3	4	jineshroxx	08-11-2017	@hotstartweets I am currently on Jio network a...	@hotstartweets i am currently on jio network a...	Help	
4	5	YaminiSachar	08-05-2017	@hotstartweets the episodes of Sarabhai vs Sar...	@hotstartweets the episodes of sarabhai vs sar...	Help	

Basic info about the dataset

In [3]:

```
df.shape  #check shape
```

Out[3]:

```
(5053, 13)
```

- Dataset has 5053 rows & 13 columns

In [4]:

```
df.columns  #check column names
```

Out[4]:

```
Index(['ID', 'UserName', 'Created_Date', 'Reviews', 'Lower_Case_Reviews',
       'Sentiment_Manual_BP', 'Sentiment_Manual', 'Review_Length',
       'DataSource', 'Year', 'Month', 'Date', 'Sentiment_Polarity'],
      dtype='object')
```

In [5]:

```
df.duplicated().sum()  #check duplicates
```

Out[5]:

```
0
```

In [6]:

```
df.isnull().sum()  #check null values
```

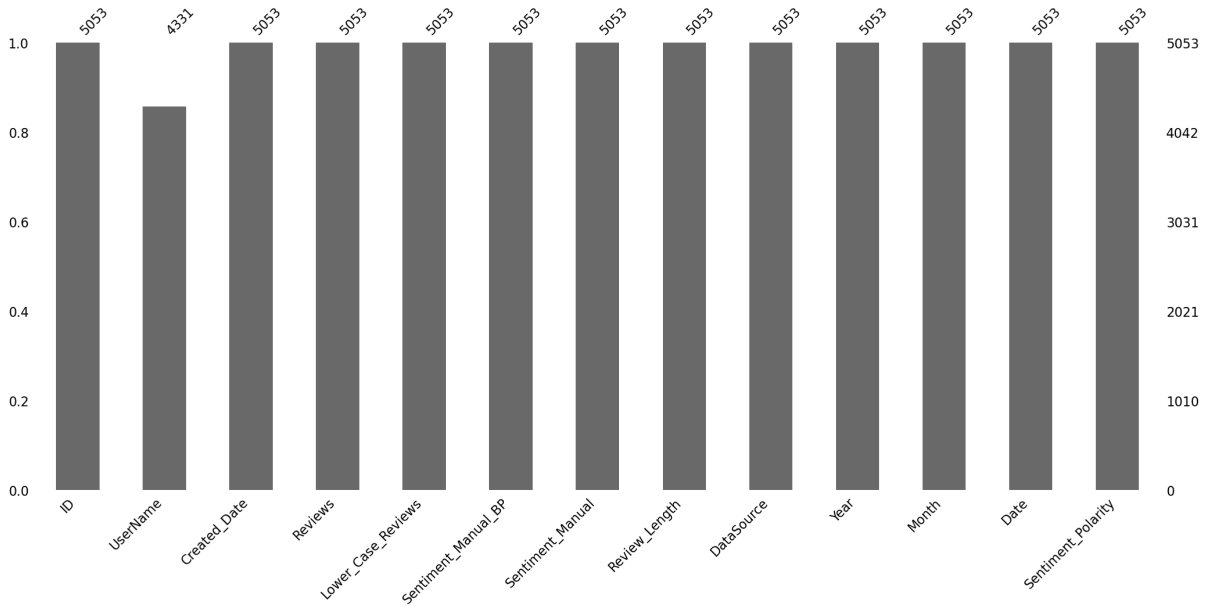
Out[6]:

```
ID                      0
UserName              722
Created_Date            0
Reviews                 0
Lower_Case_Reviews      0
Sentiment_Manual_BP     0
Sentiment_Manual        0
Review_Length           0
DataSource              0
Year                    0
Month                   0
Date                    0
Sentiment_Polarity      0
dtype: int64
```

- Username variable has null values

In [7]:

```
import missingno as msn
msn.bar(df)
plt.show()
```



In [8]:

```
df.info()    #check info
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5053 entries, 0 to 5052
Data columns (total 13 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   ID                   5053 non-null   int64 
 1   UserName             4331 non-null   object
 2   Created_Date         5053 non-null   object
 3   Reviews              5053 non-null   object
 4   Lower_Case_Reviews   5053 non-null   object
 5   Sentiment_Manual_BP  5053 non-null   object
 6   Sentiment_Manual     5053 non-null   object
 7   Review_Length        5053 non-null   int64 
 8   DataSource           5053 non-null   object
 9   Year                 5053 non-null   int64 
 10  Month                5053 non-null   int64 
 11  Date                 5053 non-null   int64 
 12  Sentiment_Polarity   5053 non-null   object
dtypes: int64(5), object(8)
memory usage: 513.3+ KB
```

In [9]:

```
df.describe().T.style.background_gradient(cmap='Blues')     #statistical summary
```

Out[9]:

	count	mean	std	min	25%	50%	7
ID	5053.000000	2527.000000	1458.819786	1.000000	1264.000000	2527.000000	3790.000
Review_Length	5053.000000	73.446665	49.374738	2.000000	26.000000	81.000000	112.000
Year	5053.000000	2017.000000	0.000000	2017.000000	2017.000000	2017.000000	2017.000
Month	5053.000000	8.000000	0.000000	8.000000	8.000000	8.000000	8.000
Date	5053.000000	9.053434	2.490424	4.000000	7.000000	10.000000	11.000

In [10]:

```
df.describe(include='object').T
```

Out[10]:

	count	unique	top	freq
UserName	4331	3303	asianet	56
Created_Date	5053	10	08-11-2017	1056
Reviews	5053	5053	Hh	1
Lower_Case_Reviews	5053	5053	hh	1
Sentiment_Manual_BP	5053	4	Positive	1733
Sentiment_Manual	5053	3	Neutral	1738
DataSource	5053	2	Twitter	2826
Sentiment_Polarity	5053	3	Positive	2513

Exploratory Data Analysis

In [66]:

```
from dataprep.eda import create_report
report = create_report(df, title='Data Report')
report
```

Out[66]:

Data Report Overview Variables ≡ Interactions Correlations

Missing Values

Overview

Dataset Statistics

Number of Variables	16
Number of Rows	5053
Missing Cells	722
Missing Cells (%)	0.9%
Duplicate Rows	0
Duplicate Rows (%)	0.0%
Total Size in Memory	4.1 MB
Average Row Size in Memory	856.8 B
Variable Types	Numerical: 4 Categorical: 12

Dataset Insights

`ID` is uniformly distributed	Uniform
`UserName` has 722 (14.29%) missing values	Missing
`ID` is skewed	Skewed
`Date` is skewed	Skewed
`compound` is skewed	Skewed
`UserName` has a high cardinality: 3303 distinct values	High Cardinality
`Reviews` has a high cardinality: 4642 distinct values	High Cardinality
`Lower_Case_Reviews` has a high cardinality: 5053 distinct values	High Cardinality
`Scores` has a high cardinality: 1281 distinct values	High Cardinality
`Year` has constant value "2017"	Constant

1 2

Variables

List of Frequent words in the dataset

In [12]:

```
from collections import Counter
cnt = Counter()
for text in df["Reviews"].values:
    for word in text.split():
        cnt[word] += 1

cnt.most_common(20)
```

Out[12]:

```
[('@hotstartweets', 1360),
 ('to', 1038),
 ('the', 1034),
 ('on', 1030),
 ('it', 848),
 ('I', 841),
 ('is', 829),
 ('app', 747),
 ('Hotstar', 696),
 ('and', 660),
 ('for', 628),
 ('of', 600),
 ('RT', 599),
 ('a', 589),
 ('in', 523),
 ('hotstar', 517),
 ('s', 451),
 ('you', 415),
 ('watch', 392),
 ('t', 387)]
```

Data Cleaning

1. Lower Casing

In [13]:

```
df['Reviews'] = df['Reviews'].apply(lambda x:x.lower())
```

2. Remove Punctuation marks

In [14]:

```
import string
string.punctuation
```

Out[14]:

```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

In [15]:

```
remove_punc = string.punctuation
def remove_punctuation(text):
    return text.translate(str.maketrans('', '', remove_punc))
df['Reviews'] = df['Reviews'].apply(lambda text: remove_punctuation(text))
```

3. Remove Short words

In [16]:

```
df['Reviews'] = df['Reviews'].apply(lambda x: ' '.join([w for w in x.split() if len(w)>2]))
```

4. Remove Stopwords

- A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query.

In [42]:

```
from nltk.corpus import stopwords
", ".join(stopwords.words('english'))
```

Out[42]:

```
"i, me, my, myself, we, our, ours, ourselves, you, you're, you've, you'll, you'd, y
our, yours, yourself, yourselves, he, him, his, himself, she, she's, her, hers, her
self, it, it's, its, itself, they, them, their, theirs, themselves, what, which, wh
o, whom, this, that, that'll, these, those, am, is, are, was, were, be, been, bein
g, have, has, had, having, do, does, did, doing, a, an, the, and, but, if, or, beca
use, as, until, while, of, at, by, for, with, about, against, between, into, throug
h, during, before, after, above, below, to, from, up, down, in, out, on, off, over,
under, again, further, then, once, here, there, when, where, why, how, all, any, bo
th, each, few, more, most, other, some, such, no, nor, not, only, own, same, so, th
an, too, very, s, t, can, will, just, don, don't, should, should've, now, d, ll, m,
o, re, ve, y, ain, aren, aren't, couldn, couldn't, didn, didn't, doesn, doesn't, ha
dn, hadn't, hasn, hasn't, haven, haven't, isn, isn't, ma, mightn, mightn't, mustn,
mustn't, needn, needn't, shan, shan't, shouldn, shouldn't, wasn, wasn't, weren, wer
en't, won, won't, wouldn, wouldn't"
```

In [18]:

```
stopwords = set(stopwords.words('english'))
def remove_stopwords(text):
    return " ".join([word for word in str(text).split() if word not in stopwords])

df["Reviews"] = df["Reviews"].apply(lambda text: remove_stopwords(text))
```

5. Stemming

- Stemming is the process of producing morphological variants of a root/base word.

In [19]:

```
from nltk.stem.porter import PorterStemmer
stemmer = PorterStemmer()
def stem_words(text):
    return " ".join([stemmer.stem(word) for word in text.split()])

df["Reviews"] = df["Reviews"].apply(lambda text: stem_words(text))
```

6. Lematization

- Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item.

In [20]:

```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
def lemmatize_words(text):
    return " ".join([lemmatizer.lemmatize(word) for word in text.split()])

df["Reviews"] = df["Reviews"].apply(lambda text: lemmatize_words(text))
```

7. Tokenization

- It is the process of tokenizing or splitting a string, text into a list of tokens

In [21]:

```
df['Reviews'] = df['Reviews'].apply(lambda x: x.split())
```

In [22]:

```
df['Reviews'].head()
```

Out[22]:

```
0                                                   []
1                                                   []
2    [hotstarhelp, paymnt, premium, subscript, tran...
3    [hotstartweet, current, jio, network, would, l...
4    [hotstartweet, episod, sarabhai, sarabhai, sea...
Name: Reviews, dtype: object
```

8. Convert list into string

In [23]:

```
def join_back(list_input):
    return " ".join(list_input)
df['Reviews']= df['Reviews'].apply(join_back)
```

In [24]:

```
df['Reviews'].head()
```

Out[24]:

```
0
1
2    hotstarhelp paymnt premium subscript transact ...
3    hotstartweet current jio network would like kn...
4    hotstartweet episod sarabhai sarabhai season d...
Name: Reviews, dtype: object
```

Sentiment Prediction

- We will be using 2 approach - Vader & Supervised machine learning algorithms.

Approach - 1 - VADER

- VADER(Valence Aware Dictionary for Sentiment Reasoning) is an NLTK module that provides sentiment scores based on the words used. It is a rule-based sentiment analyzer in which the terms are generally labeled as per their semantic orientation as either positive or negative. VADER not only tells about the Positivity and Negativity score but also tells us about how positive or negative a sentiment is.

In [25]:

```
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\manme\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

Out[25]:

```
True
```

Classifying Sentiment Scores

- It is a rule-based sentiment analyzer in which the terms are generally labeled as per their semantic orientation to categorise as positive, negative or neutral. Polarity scores method is used to determine the sentiment.

In [26]:

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
sia = SentimentIntensityAnalyzer()
```

In [27]:

```
df['Scores']=df['Reviews'].apply(lambda review : sia.polarity_scores(review) )
```

Compound

- It corresponds to the sum of the valence score of each word in the lexicon and determines the degree of the sentiment rather than the actual value as opposed to the previous ones. Its value is between -1 (most extreme negative sentiment) and +1 (most extreme positive sentiment).

In [28]:

```
df['compound'] = df['Scores'].apply(lambda score_dict : score_dict['compound'])
```

Classifying into Positive, Negative and Neutral

- Positive sentiment where Compound > 0.1
- Negative sentiment where Compound < -0.1
- Neutral sentiment where Compound >=-0.1 & <= 0.1

In [29]:

```
df['comp_score'] = df['compound'].apply(lambda c : 'Positive' if c > 0 else ('Neutral' if c == 0 
df.head()
```

Out[29]:

	ID	UserName	Created_Date	Reviews	Lower_Case_Reviews	Sentiment_Manual_BP	Sentiment_I
0	1	<NA>	08-10-2017		hh	Negative	N
1	2	<NA>	08-11-2017		no	Negative	N
2	3	asadynwa	08-12-2017	hotstarhelp paymnt premium subscript transact ...	@hotstar_helps during paymnt for premium subsc...	Help	N
3	4	jineshroxx	08-11-2017	hotstartweet current jio network would like kn...	@hotstartweets i am currently on jio network a...	Help	N
4	5	YaminiSachar	08-05-2017	hotstartweet episod sarabhai sarabhai season d...	@hotstartweets the episodes of sarabhai vs sar...	Help	N

In [30]:

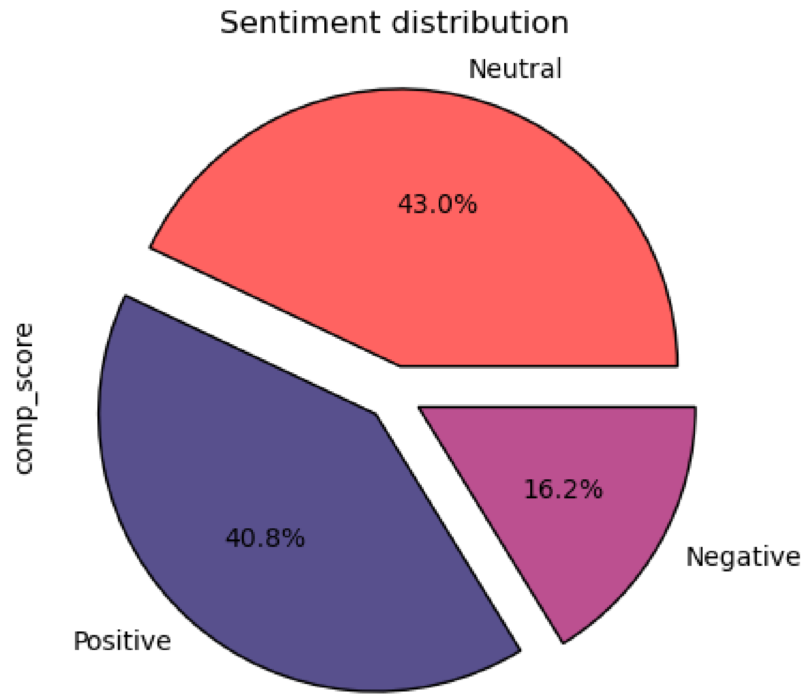
```
df.comp_score.value_counts()
```

Out[30]:

```
Neutral     2171
Positive    2061
Negative     821
Name: comp_score, dtype: int64
```

In [31]:

```
df['comp_score'].value_counts().plot(kind='pie',explode=[0.1,0.1,0.1],autopct='%0.1f%%',
                                    colors=('#ff6361','#58508d','#bc5090'),wedgeprops={'edgecolor':
plt.title('Sentiment distribution')
plt.show()
```



Wordcloud

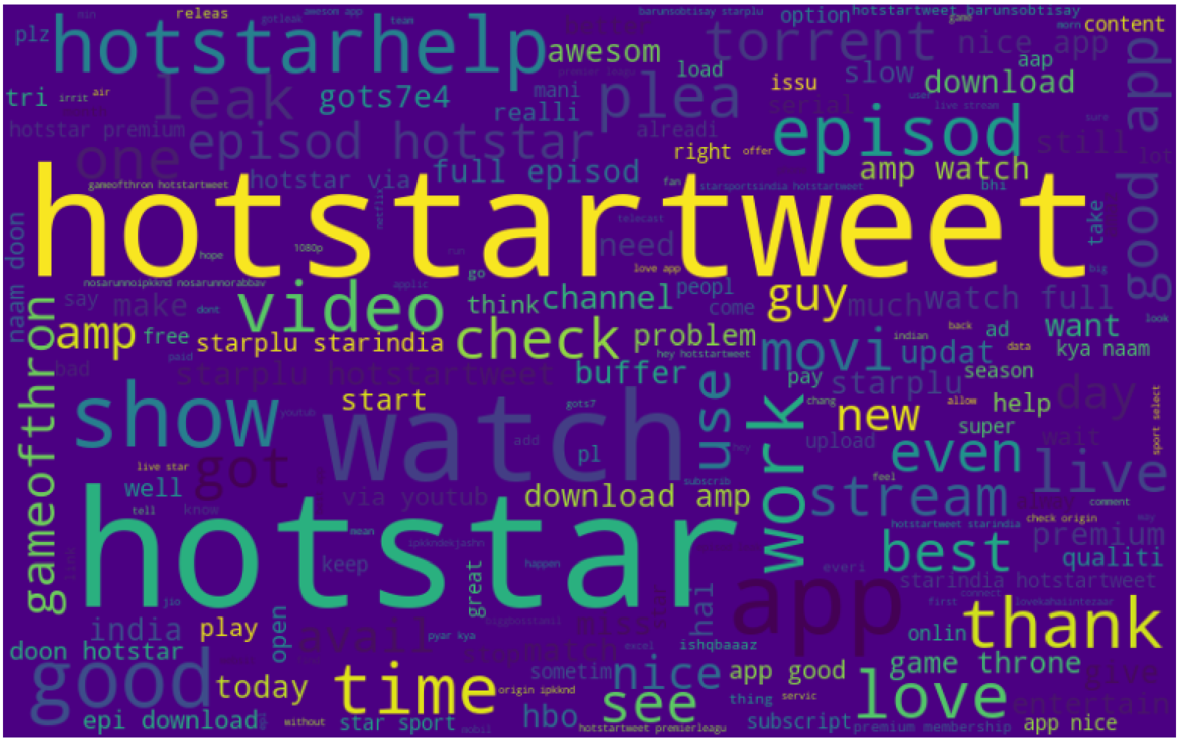
- Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance.

Most Frequent words

In [33]:

```
frequent_words = ' '.join([text for text in df['Reviews']])
from wordcloud import WordCloud
wordcloud = WordCloud(width=800, height=500, random_state=21, max_font_size=110,
                      background_color='indigo',colormap='viridis').generate(frequent_words)

plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```



Positive sentiments Word cloud

In [34]:

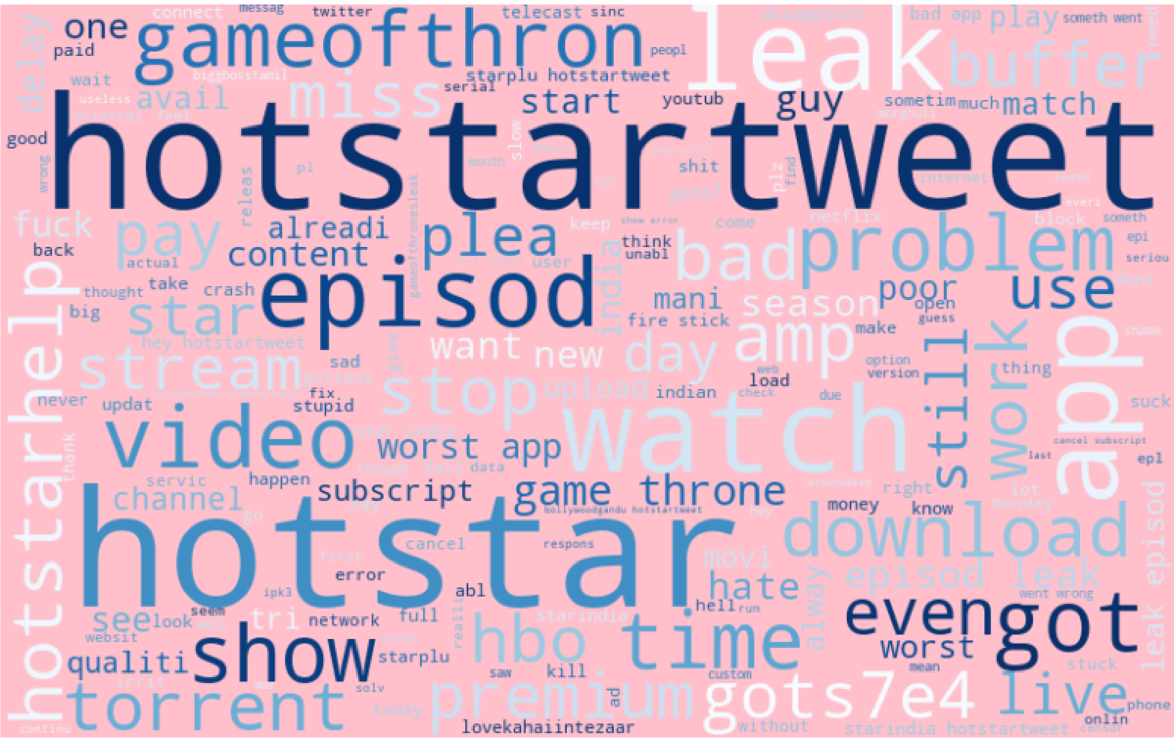
```
positive_sentiments = ' '.join([text for text in df['Reviews'][df.comp_score == 'Positive']])
positive_wordcloud = WordCloud(width=800, height=500, random_state=21, max_font_size=110,
                               background_color='purple',colormap='Set2').generate(positive_senti
plt.figure(figsize=(10, 7))
plt.imshow(positive_wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```

Negative sentiments Word cloud

In [35]:

```
negative_sentiments = ' '.join([text for text in df['Reviews'][df.comp_score == 'Negative']])
negative_wordcloud = WordCloud(width=800, height=500, random_state=21, max_font_size=110,
                               background_color='pink',colormap='Blues').generate(negative_sentim

plt.figure(figsize=(10, 7))
plt.imshow(negative_wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```

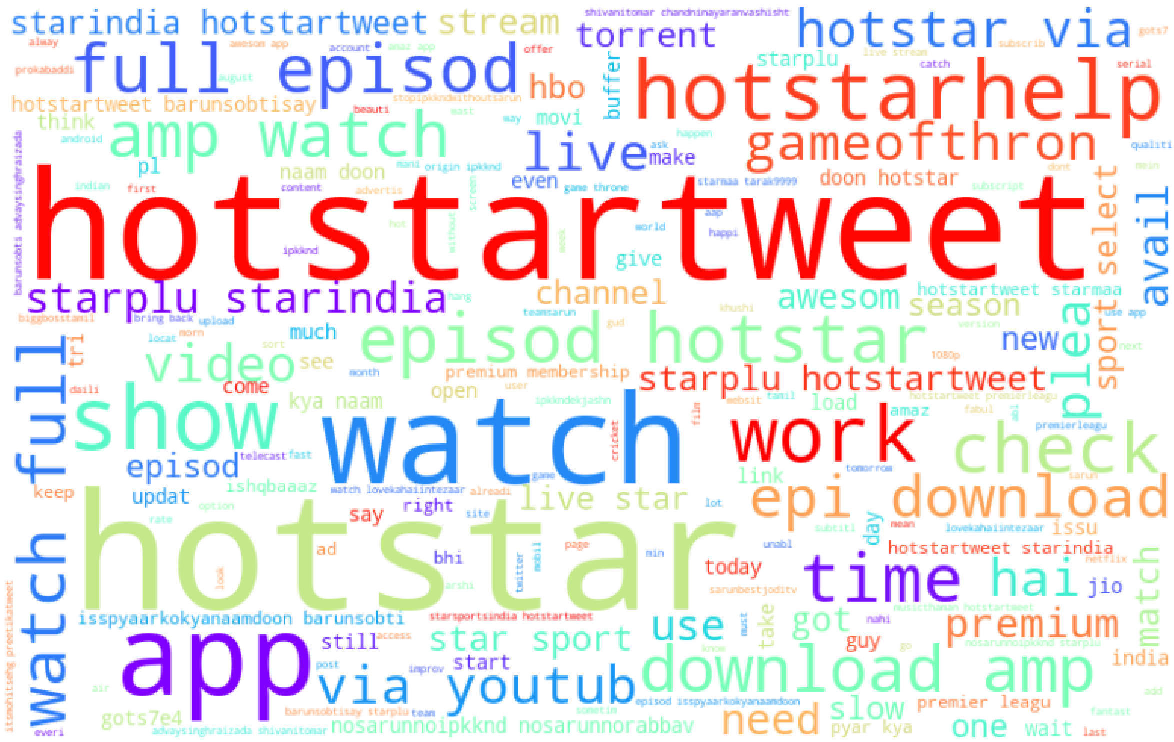


Neutral sentiments Word cloud

In [36]:

```
neutral_sentiments = ' '.join([text for text in df['Reviews'][df.comp_score == 'Neutral']])
neutral_wordcloud = WordCloud(width=800, height=500, random_state=21, max_font_size=110,
                              background_color='white',colormap='rainbow').generate(neutral_senti

plt.figure(figsize=(10, 7))
plt.imshow(neutral_wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```



Evaluation

In [37]:

```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
```

In [38]:

```
print(confusion_matrix(df['Sentiment_Manual'], df['comp_score']))
print("-------------"*5)
print(classification_report(df['Sentiment_Manual'], df['comp_score']))
print("-------------"*5)
print(accuracy_score(df['Sentiment_Manual'], df['comp_score']))
```

```
[[ 542  673  367]
 [ 202 1017  519]
 [  77  481 1175]]
-----------------------------------------------------------------
              precision    recall  f1-score   support

    Negative       0.66      0.34      0.45      1582
     Neutral       0.47      0.59      0.52      1738
    Positive       0.57      0.68      0.62      1733

    accuracy                           0.54      5053
   macro avg       0.57      0.54      0.53      5053
weighted avg       0.56      0.54      0.53      5053

-----------------------------------------------------------------
0.5410647140312685
```

- Vader model is giving accuracy of 54% so to increase accuracy we will build Supervised machine learning models as our next step.

Approach No 2 - Classification algorithm

In [39]:

```
df1=pd.read_csv(r"C:\Users\manme\Documents\Priya\Stats and ML\Dataset\hotstar_reviews.csv")
df1.head()
```

Out[39]:

	ID	UserName	Created_Date	Reviews	Lower_Case_Reviews	Sentiment_Manual_BP	Sentime
0	1	NaN	08-10-2017	Hh	hh	Negative	
1	2	NaN	08-11-2017	No	no	Negative	
2	3	asadynwa	08-12-2017	@hotstar_helps during paymnt for premium subsc...	@hotstar_helps during paymnt for premium subsc...	Help	
3	4	jineshroxx	08-11-2017	@hotstartweets I am currently on Jio network a...	@hotstartweets i am currently on jio network a...	Help	
4	5	YaminiSachar	08-05-2017	@hotstartweets the episodes of Sarabhai vs Sar...	@hotstartweets the episodes of sarabhai vs sar...	Help	

In [40]:

```
df1=df1[['Sentiment_Manual','Reviews']]
df1.head()
```

Out[40]:

	Sentiment_Manual	Reviews
0	Negative	Hh
1	Negative	No
2	Negative	@hotstar_helps during paymnt for premium subsc...
3	Negative	@hotstartweets I am currently on Jio network a...
4	Negative	@hotstartweets the episodes of Sarabhai vs Sar...

Cleaning the Reviews variable

In [43]:

```
from tqdm import tqdm   #to create progress bar for the loops

cleaned_reviews = []

for i in tqdm(df1['Reviews'].values):
    i = re.sub('[^a-zA-Z]',' ',i)
    i = ' '.join(low.lower() for low in i.split() if low.lower() not in stopwords.words('english'
    cleaned_reviews.append(i.strip())
```

```
100%|██████████████████████████████████████████████████|
5053/5053 [00:08<00:00, 571.68it/s]
```

In [44]:

```
cleaned_reviews
```

Out[44]:

```
['hh',
 '',
 'hotstar helps paymnt premium subscription transaction failed twice received re
fund one transaction',
 'hotstartweets currently jio network would like know whether able watch epl tel
ecasted star sports select hd',
 'hotstartweets episodes sarabhai vs sarabhai season downloadable able watch off
line please smthng',
 'hotstartweets able watch latest episode got app allow take screenshot error he
lp resolve asap',
 'please allow rupay maestro payment gateways premium membership mean paytm work
s thru debit cards would great hotstartweets',
 'hotstar helps today epi lovekahaiintezaar nt available available morning showi
ng nt available due expiry',
 'hotstartweets hotstarfraud paid subscription july havent received cashback spe
cified hdfc card',
 'hotstartweets premium accnt hotstar showing tht premium member u pls chk ankus
h gmail com',
 'hotstartweets seeing blank page terms amp conditions hdfc bank cashback offer
hotstar premium membership please help',
 'hotstartweets sir please allow us download videos ur app present option allow
us dwnld mre videos due ltd space',
 'hotstar helps hi pl tab sports homepage isl bundesliga search team name stream
pls look',
 'hotstartweets unable watch star sports select hd live hotstar app even though
premium hotstar membership free trial',
 'hotstartweets great could keep track watch across app web browser expect premi
um account',
 'hotstartweets sir mobile internal disk ltd space ur present option downloading
mks us difficult dwnld videos serial',
 'hotstartweets sir want dwnld episodes mahabharata hindi cant due current offli
ne option ltd space mobile',
 'rt ayesha farha hotstartweets adoringdua get original content every seasons mi
ni series like one',
 'rt jineshroxx hotstartweets currently jio network would like know whether able
watch epl telecasted star',
 'rt supershaheerafc mamtaypatnaik yashapatnaik heard show continued hotstar mon
th removed',
 'rt teamprians lovekahaiintezaar preetikaraoisback starplus hotstartweets post
promo indifferent u r towards',
 'black screen premier league matches jiotv premier league available officialjio
tv jiocare hotstartweets',
 'hotstar helps already premium subscriber subscription ends aug hope able watch
first pl game tonight',
 'hotstartweets existing premium subscribers need get premier league pass incl p
remium membership default',
 'anyone watching arsenal game hotstartweets right give feedback streaming quali
ty arslei hotstar',
```

Feature extraction using TF- IDF

- Feature extraction refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set.

TF- IDF

- TF-IDF stands for Term Frequency Inverse Document Frequency of records. It can be defined as the calculation of how relevant a word in a series or corpus is to a text.

In [45]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer()
x = tfidf.fit_transform(cleaned_reviews).toarray()
pd.DataFrame(x).shape
```

Out[45]:

```
(5053, 7142)
```

In [46]:

```
pd.DataFrame(x).head()
```

Out[46]:

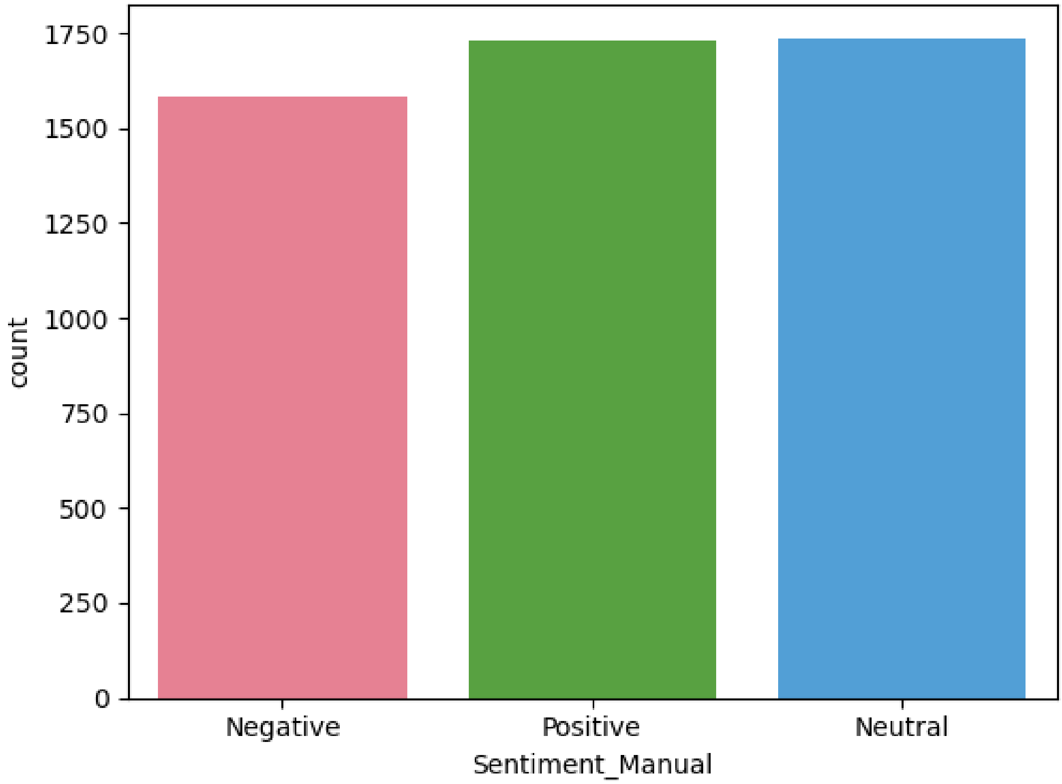
	0	1	2	3	4	5	6	7	8	9	...	7132	7133	7134	7135	7136	7137	7138	7139	7
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

5 rows × 7142 columns

Encoding

In [47]:

```
sns.countplot(x='Sentiment_Manual',data=df1,palette='husl')
plt.show()
```



In [48]:

```
df1['Sentiment_Manual'] = df1['Sentiment_Manual'].astype('category')
df1['Sentiment_Manual'] = df1['Sentiment_Manual'].cat.codes
```

In [49]:

```
df1['Sentiment_Manual'].value_counts()
```

Out[49]:

```
1    1738
2    1733
0    1582
Name: Sentiment_Manual, dtype: int64
```

Split the data into Train & Test

In [50]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, df1['Sentiment_Manual'], test_size=0.25, r
```

Model No 1 - Logistic Regression

In [51]:

```
# Model building
from sklearn.linear_model import LogisticRegression
logit = LogisticRegression(random_state=100)
log=logit.fit(x_train, y_train)
# Predict
y_pred_train_log = logit.predict(x_train)
y_pred_test_log = logit.predict(x_test)
# Evaluate
from sklearn.metrics import confusion_matrix,classification_report, accuracy_score
accuracy_log_test=accuracy_score(y_test,y_pred_test_log)
accuracy_log_train=accuracy_score(y_train,y_pred_train_log)
print('Logistic regression Train accuracy:', accuracy_score(y_train, y_pred_train_log))
print('------'*10)
print('Logistic regression Test accuracy:', accuracy_score(y_test, y_pred_test_log))
```

```
Logistic regression Train accuracy: 0.9242544206914753
------------------------------------------------------------
Logistic regression Test accuracy: 0.7618670886075949
```

In [62]:

```
from sklearn.model_selection import cross_val_score
train_accuracy_log = cross_val_score(log,x_train, y_train, cv=10)
crossval_train_log=train_accuracy_log.mean()
test_accuracy_log = cross_val_score(log,x_test, y_test, cv=10)
crossval_test_log=test_accuracy_log.mean()
print('Logistic regression Train accuracy after Cross validation:', crossval_train_log)
print('-------'*5)
print('Logistic regression Test accuracy after Cross validation:', crossval_test_log)
```

```
Logistic regression Train accuracy after Cross validation: 0.7690657676145805
-----------------------------------
Logistic regression Test accuracy after Cross validation: 0.734939382577178
```

Model No 2 - Random Forest Classifier

In [52]:

```
# Model building
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(n_estimators=200,oob_score=False)
rf.fit(x_train,y_train)
# Predict
y_pred_train_rf=rf.predict(x_train)
y_pred_test_rf=rf.predict(x_test)
# Evaluate
accuracy_rf_test=accuracy_score(y_test,y_pred_test_rf)
accuracy_rf_train=accuracy_score(y_train,y_pred_train_rf)
print('Random Forest - Train accuracy:', accuracy_score(y_train, y_pred_train_rf))
print('-------'*10)
print('Random Forest - Test accuracy:', accuracy_score(y_test, y_pred_test_rf))
```

```
Random Forest - Train accuracy: 0.9968329374505146
----------------------------------------------------------------------
Random Forest - Test accuracy: 0.757120253164557
```

In [65]:

```
train_accuracy_rf = cross_val_score(rf,x_train, y_train, cv=10)
crossval_train_rf=train_accuracy_rf.mean()
test_accuracy_rf = cross_val_score(rf,x_test, y_test, cv=10)
crossval_test_rf=test_accuracy_rf.mean()
print('Random forest after Cross validation Train accuracy:', crossval_train_rf)
print('-------'*10)
print('Random forest  after Cross validation Test accuracy:', crossval_test_rf)
```

```
Random forest after Cross validation Train accuracy: 0.7579860674847482
----------------------------------------------------------------------
Random forest  after Cross validation Test accuracy: 0.7404136982877141
```

Model No 3 - Multinomial Naive Bayes

In [53]:

```
# Model building
from sklearn.naive_bayes import MultinomialNB
mnb = MultinomialNB()
mnb.fit(x_train, y_train)
# Predict the model
y_pred_train_mnb = mnb.predict(x_train)
y_pred_test_mnb = mnb.predict(x_test)
# Evaluate
accuracy_mnb_test=accuracy_score(y_test,y_pred_test_mnb)
accuracy_mnb_train=accuracy_score(y_train,y_pred_train_mnb)
print('Multinomial Naive Bayes -Train accuracy:', accuracy_score(y_train, y_pred_train_mnb))
print('-------'*10)
print('Multinomial Naive Bayes -Test accuracy:', accuracy_score(y_test, y_pred_test_mnb))
```

```
Multinomial Naive Bayes -Train accuracy: 0.9020849828450779
----------------------------------------------------------------------
Multinomial Naive Bayes -Test accuracy: 0.7539556962025317
```

In [63]:

```
train_accuracy_mnb = cross_val_score(mnb,x_train, y_train, cv=10)
crossval_train_mnb=train_accuracy_mnb.mean()
test_accuracy_mnb = cross_val_score(mnb,x_test, y_test, cv=10)
crossval_test_mnb=test_accuracy_mnb.mean()
print('Naives Bayes after Cross validation Train accuracy:', crossval_train_mnb)
print('---------------'*5)
print('Naives Bayes after Cross validation Test accuracy:', crossval_test_mnb)
```

```
Naives Bayes after Cross validation Train accuracy: 0.7460987561251413
---------------------------------------------------------------------------
Naives Bayes after Cross validation Test accuracy: 0.7302274715660543
```

Model No 4 - Support Vector machine

In [54]:

```
# Radial Basis Function Kernel (RBF) - (Defaut SVM) Model building
from sklearn.svm import SVC
svm_rbf = SVC(kernel='rbf')
svm_rbf.fit(x_train, y_train)
#Predict
y_pred_train_rbf = svm_rbf.predict(x_train)
y_pred_test_rbf = svm_rbf.predict(x_test)
#Evaluate
accuracy_rbf_test=accuracy_score(y_test,y_pred_test_rbf)
accuracy_rbf_train=accuracy_score(y_train,y_pred_train_rbf)
print('Rbf - SVM - Train accuracy:', accuracy_score(y_train, y_pred_train_rbf))
print('-------'*10)
print('Rbf - SVM - Test accuracy:', accuracy_score(y_test, y_pred_test_rbf))
```

```
Rbf - SVM - Train accuracy: 0.9778305621536025
----------------------------------------------------------------------
Rbf - SVM - Test accuracy: 0.7729430379746836
```

In [64]:

```
train_accuracy_rbf = cross_val_score(svm_rbf,x_train, y_train, cv=10)
crossval_train_rbf=train_accuracy_rbf.mean()
test_accuracy_rbf = cross_val_score(svm_rbf,x_test, y_test, cv=10)
crossval_test_rbf=test_accuracy_rbf.mean()
print('Rbf- SVM after Cross validation Train accuracy:', crossval_train_rbf)
print('---------------'*5)
print('Rbf- SVM after Cross validation Test accuracy:', crossval_test_rbf)
```

```
Rbf- SVM after Cross validation Train accuracy: 0.7783047842414598
---------------------------------------------------------------------------
Rbf- SVM after Cross validation Test accuracy: 0.7333458317710286
```

Confusion Matrix

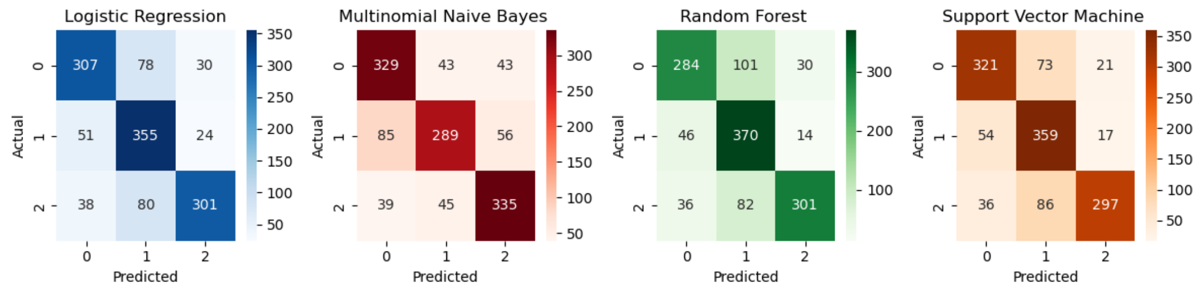
In [78]:

```
plt.figure(figsize=(12,3))
plt.subplot(141)
sns.heatmap(confusion_matrix(y_test,y_pred_test_log),cmap='Blues',annot=True, fmt='g')
plt.title("Logistic Regression")
plt.ylabel('Actual')
plt.xlabel('Predicted')

plt.subplot(142)
sns.heatmap(confusion_matrix(y_test,y_pred_test_mnb),cmap='Reds',annot=True, fmt='g')
plt.title("Multinomial Naive Bayes ")
plt.ylabel('Actual')
plt.xlabel('Predicted')

plt.subplot(143)
sns.heatmap(confusion_matrix(y_test,y_pred_test_rf),cmap='Greens',annot=True, fmt='g')
plt.title("Random Forest ")
plt.ylabel('Actual')
plt.xlabel('Predicted')

plt.subplot(144)
sns.heatmap(confusion_matrix(y_test,y_pred_test_rbf),cmap='Oranges',annot=True, fmt='g')
plt.title("Support Vector Machine ")
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.tight_layout()
plt.show()
```



Combining models in Tabular form

In [57]:

```
Models=['Logistic','Random_forest','MultinomialNB','SVM',]
Trainacc=[accuracy_log_train,accuracy_rf_train,accuracy_mnb_train,accuracy_rbf_train]
Testacc=[accuracy_log_test,accuracy_rf_test,accuracy_mnb_test,accuracy_rbf_test]
```

In [58]:

```
Combined_accuracy=pd.DataFrame({'ModelName':Models,'TrainAccuracy':Trainacc,'TestAccuracy':Testac
print(Combined_accuracy)
```

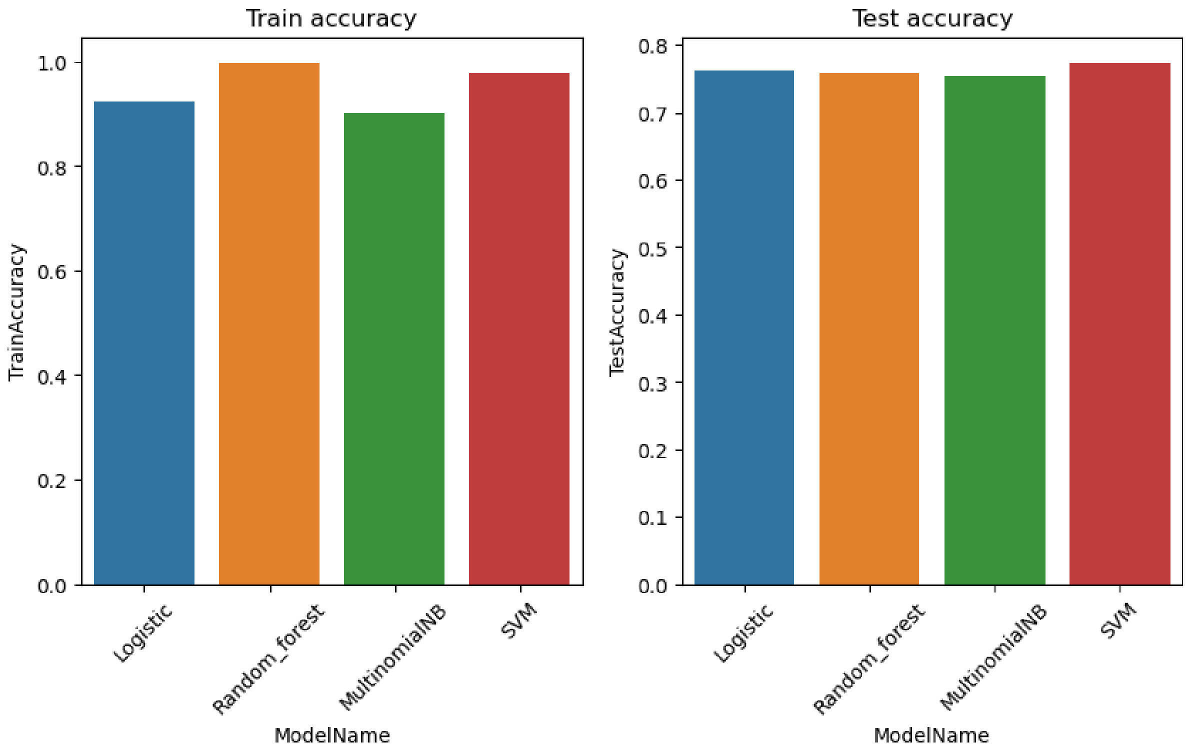
```
       ModelName  TrainAccuracy  TestAccuracy
0       Logistic       0.924254      0.761867
1  Random_forest       0.996833      0.757120
2  MultinomialNB       0.902085      0.753956
3            SVM       0.977831      0.772943
```

Accuracy Visualization

In [61]:

```
plt.figure(figsize=(10,5))
plt.subplot(121)
sns.barplot(x='ModelName',y='TrainAccuracy',data=Combined_accuracy)
plt.xticks(rotation=45)
plt.title('Train accuracy')


plt.subplot(122)
sns.barplot(x='ModelName',y='TestAccuracy',data=Combined_accuracy)
plt.xticks(rotation=45)
plt.title('Test accuracy')
plt.show()
```



Conclusion

- I tried to work upon Hotstar Reviews dataset having 5053 rows & 13 columns .
- The objective was to do Sentiment Analysis and categorize reviews under Positive, Negative & Neutral.
- For this I used 2 approaches, the first one was Vader approach and the second one was Supervised machine learning classification algorithms.
- Text preprocessing was done as the first step of both the approaches.
- Evaluation of Vader model gave an accuracy of 54%.
- To improve accuracy I build Classifier models of Logistic Regression, Multinomial Naive Bayes, Random Forest & Support Vector Machine.
- All the models gave more than 90% Train accuracy but Test accuracy was between 75% to 77% indicating high variance issue.
- To deal with it I tried cross validation of all models which gave Train accuracy between 74 to 77% & Test accuracy of 73-74%.
- After cross validation almost all model's accuracy was almost similar.
- Support Vector Machine had the highest accuracy of 77% & 73% making it the best amongst the rest models to predict Sentiments for this dataset.

In [ ]:

```

```