GROUP NO - 19
SENTIMENT ANALYSIS - AMAZON PRODUCT REVIEWS

PARAM MADAN

INTRODUCTION

- THE PRACTICE OF EVALUATING TEXT TO ASCERTAIN THE EMOTIONAL TONE OF THE AUTHOR'S MESSAGE IS KNOWN AS SENTIMENT ANALYSIS. SENTIMENT ANALYSIS IS THE PROCESS OF EXAMINING CUSTOMER REVIEWS OF PRODUCTS ON AMAZON TO ASCERTAIN WHETHER THEY ARE POSITIVE, NEGATIVE, OR NEUTRAL.
- IN THIS PROJECT, WE WILL EXAMINE AMAZON PRODUCT REVIEWS USING SENTIMENT ANALYSIS TECHNIQUES. MACHINE LEARNING TECHNIQUES WILL BE USED TO DETERMINE THE SENTIMENT THAT EACH REVIEW EXPRESSES. THE OBJECTIVE IS TO OFFER PERCEPTIONS OF CONSUMER PREFERENCES AND OPINIONS ABOUT A CERTAIN PRODUCT, WHICH CAN BE UTILIZED TO SUPPORT DATA-DRIVEN DECISIONS TO ENHANCE THE PRODUCT OR CREATE SUCCESSFUL MARKETING CAMPAIGNS.

DATASET DESCRIPTION

- IT CONSISTS OF MORE THAN 500K REVIEWS.
- IT CONSISTS OF BOTH NUMERICAL AND TEXT DATA.
- THERE ARE 10 COLUMNS AND 568,454 ROWS.

REFERENCE LINK:

- HTTP://WWW.KAGGLE.COM/DATASETS/ARHAMRUMI/AMAZON-PRODUCT-REVIEWS

METHODOLOGY

STEP 1: IMPORTING ALL THE LIBRARIES

STEP 2: AFTER LOADING THE DATA, WE CHECKED FOR THE NULL VALUES AND THEN WE RAN A CODE TO REMOVE THE NULL VALUES FROM OUR DATASET, AND THEN WE RECHECK THE DATASET TO CHECK IF ANY NULL VALUE IS STILL PRESENT OR NO

```
import warnings
warnings.filterwarnings("ignore")
import time
import pandas as pd
import numpy as np
from nltk.corpus import stopwords
from textblob import TextBlob
from textblob import Word
from wordcloud import WordCloud
from wordcloud import STOPWORDS
import string
import nltk
from nltk.tokenize import word_tokenize
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import re
import os
import sys
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
vader=SentimentIntensityAnalyzer()
```

```
#Checking Null values

APR_dataframe.isnull().sum()

]:  Id              0
    Product_ID      0
    User_ID         0
    ProfileName    16
    Help_num        0
    Help_denom      0
    Ratings         0
    Time            0
    Summary        27
    Comments        0
    dtype: int64
```

STEP 3: THEN WE MOVED TO THE STEP OF DATA CLEANING WHEREIN WE CREATED FUNCTIONS TO REMOVE THE PUNCTUATION MARKS AND SIGNS AND SYMBOLS WHICH ARE NOT REQUIRED.

STEP 4: AFTER REMOVING THE PUNCTUATION AND UNNECESSARY SYMBOLS FROM THE COMMENTS WE RAN A FUNCTION TO CHANGE REVIEWS INTO PLAIN TEXT FORMAT

STEP 5: LATER WE CONVERTED THE REVIEWS TO LOWERCASE FOR EASE AND THEN WE CONVERTED THE REVIEWS TO STRING FORMAT.

STEP 6: THEN WE SEPARATED EACH WORD IN THE REVIEWS AND ASSIGNED THEM TO INDIVIDUAL TOKENS

DATA CLEANING PROCESS

#Removing @ and punctuation from the column Summary and Comments

In [6]:
```
def cleancomments (text):
    text = re.sub (r'@[A-Za-z0-9]+', '',text)  #remove @
    text = re.sub ('[^\w\s]', '',text)
    return text
```

In [7]:
```
mainframe['Comments']=mainframe['Comments'].apply(cleancomments)
mainframe['Summary']=mainframe['Summary'].apply(cleancomments)
```

In [8]:
```
stop= stopwords.words('english')
mainframe['Comments']=mainframe['Comments'].apply(lambda x: " ".join(x for x in x.split() if x not in stop))
stop= stopwords.words('english')
mainframe['Summary']=mainframe['Summary'].apply(lambda x: " ".join(x for x in x.split() if x not in stop))
```

#Converting the entire column of summary and comments to lower case

In [9]:
```
mainframe['Comments']=mainframe['Comments'].str.lower()
mainframe['Summary']=mainframe['Summary'].str.lower()
```

#Converting the entire column of summary and comments to String

In [10]:
```
mainframe['Comments']=mainframe['Comments'].astype(str)
mainframe['Summary']=mainframe['Summary'].astype(str)
```

#Creating tokens

```
def CommentsTokenize(Comments):
    tokenize = nltk.word_tokenize(Comments)
    return [g for g in tokenize if g.isalpha()]
```

```
mainframe['Comments_Tokens']=mainframe['Comments'].apply(CommentsTokenize)
```

STEP 7: WE THEN CREATED 2 FUNCTIONS NAMED "SUBJECTIVITY" AND "POLARITY" AND USING THE FEATURES OF TEXTBLOB WE GOT THE POLARITY SCORE WHICH RANGES BETWEEN -1 AND 1, WHERE -1 IS THE MOST NEGATIVE AND 1 IS THE MOST POSITIVE.

STEP 8: THE REVIEWERS HAVE GIVEN NUMBER RATINGS AS WELL TO THEIR REVIEWS SO NOW WE USE THOSE NUMBERS DIRECTLY USE THOSE NUMBERS AND CONSIDER THE REVIEWS AS POSITIVE, NEGATIVE OR NEUTRAL BASED ON THE NUMBERS WHERE IF THE RATING IS 2 OR LESS THAN 2 IT WILL BE CONSIDERED AS NEGATIVE, IF THE NUMBER IS 3 IT WILL BE CONSIDERED AS NEUTRAL AND IF THE RATING IS 3 OR MORE THAN 3 IT WILL BE CONSIDERED AS POSITIVE.

#Calculating polarity and subjectivity for each comment in the Comments column

```
def subjectivity(text):
    return TextBlob(text).sentiment.subjectivity
def polarity(text):
    return TextBlob(text).sentiment.polarity
text_blob['Subjectivity']=text_blob['Comments'].apply(subjectivity)
text_blob['Ploarity']=text_blob['Comments'].apply(polarity)
```

#Based on the polarity score, we are deriving the sentiment of comments into a positive, negative, or neutral value.

```
def polanalysis(score):
    if score<0:
        return 'Negative'
    elif score==0:
        return 'Neutral'
    else:
        return 'Positive'
text_blob['Textblob_Analysis']= text_blob['Ploarity'].apply(polanalysis)
```

#Based on Ratings provided from the user, We are deriving positive, negative, or neutral value.

```
def ratingsanalysis(Ratings):
    if Ratings<3:
        return 'Negative'
    if Ratings==3:
        return 'Neutral'
    if Ratings>3:
        return 'Positive'
text_blob['Ratings_Analysis']=text_blob['Ratings'].apply(ratingsanalysis)
```

STEP 9: NOW WE USE THE VADER SENTIMENT ANALYSIS ON THE SAME REVIEWS, IT DIVIDES THE REVIEWS INTO 4 COLUMNS NEGATIVE, POSITIVE, NEUTRAL AND COMPOUND COLUMN, THE COMPOUND COLUMN GIVES A COMPOUNDED VALUE.

STEP 10: THEN WE EXTRACTED THE PERCENTAGES OF ALL 3 TECHNIQUES THAT IS VADER ANALYSIS, TEXTBLOB ANALYSIS AND ANALYSIS OF THE RATINGS OF THE REVIEW AND WE COMPARED THE PERCENTAGES OF ALL 3 ANALYSIS.

#Now we are using Vader Sentiment Analysis.

```
vader_data=mainframe
```

```
temp_data=[]
for row in vader_data['Comments']:
    ab= vader.polarity_scores(row)
    temp_data.append(ab)
vader_new=pd.DataFrame(temp_data)
```

#Based on the compound value, we are deriving sentiment of comments into a positive, negative, or neutral value.

```
def vaderanalysis(compound):
    if compound<0:
        return 'Negative'
    elif compound==0:
        return 'Neutral'
    else:
        return 'Positive'
vader_new['Vader_Analysis']= vader_new['compound'].apply(vaderanalysis)
```

```
vader_new=vader_new.drop(columns=['neg','neu','pos'])
```

```
new_combined_data= pd.concat([text_blob.reset_index(drop=True), vader_new], axis=1)
```

#We are calculating the percentage of positive, negative, and neutral values in Ratings analysis.

```
(new_combined_data['Ratings_Analysis'].value_counts()/new_combined_data['Ratings_Analysis'].count())*100
```

```
3]: Positive    78.071325
    Negative    14.427413
    Neutral      7.501262
    Name: Ratings_Analysis, dtype: float64
```

#We are calculating the percentage of positive, negative, and neutral values in Vader analysis.

```
(new_combined_data['Vader_Analysis'].value_counts()/new_combined_data['Vader_Analysis'].count())*100
```

```
4]: Positive    90.225910
    Negative     8.385833
    Neutral      1.388256
    Name: Vader_Analysis, dtype: float64
```

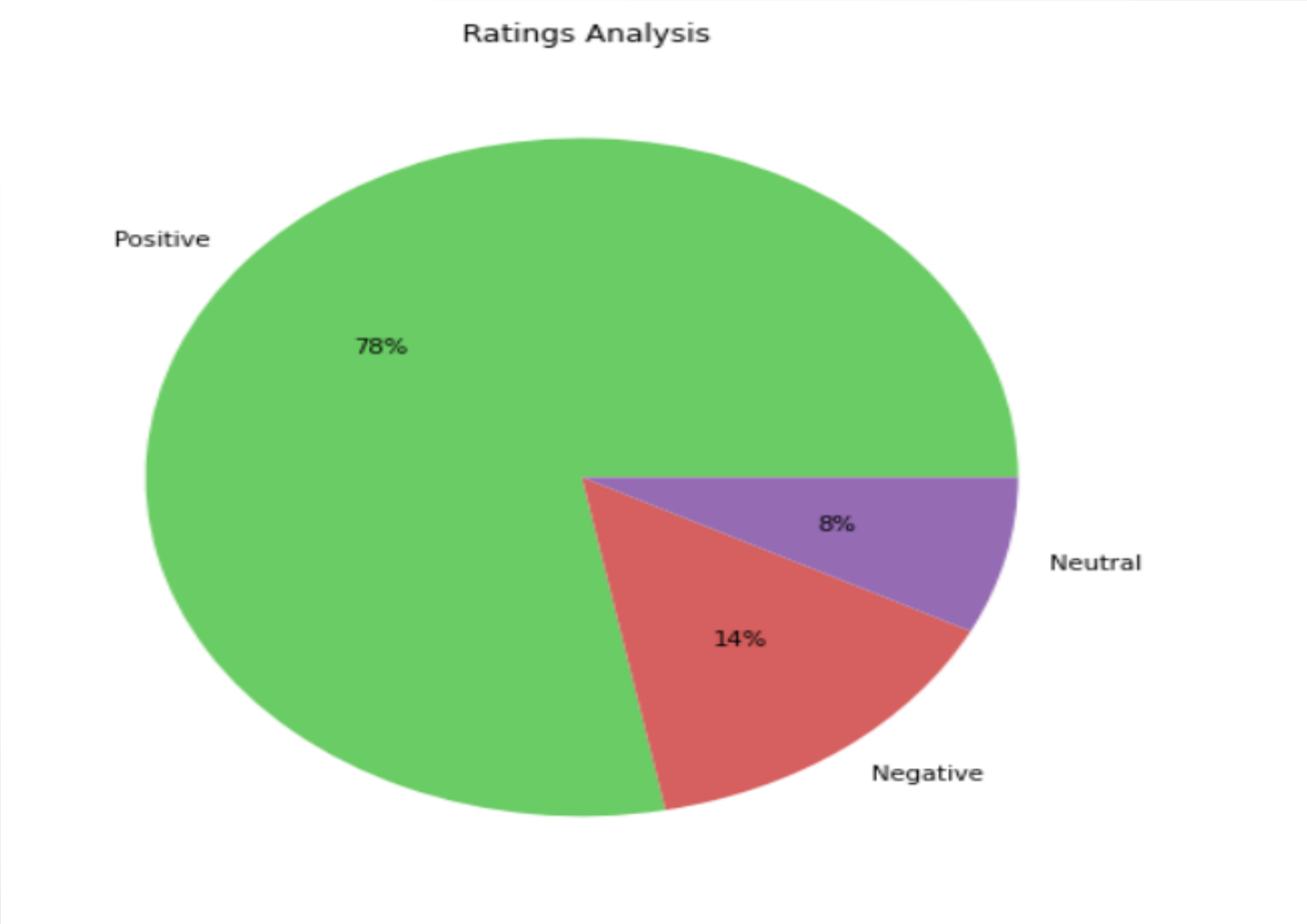
#We are calculating the percentage of positive, negative, and neutral values in textblob analysis.

```
(new_combined_data['Textblob_Analysis'].value_counts()/new_combined_data['Textblob_Analysis'].count())*100
```

```
[22]: Positive    87.316924
      Negative    10.784978
      Neutral      1.898098
      Name: Textblob_Analysis, dtype: float64
```

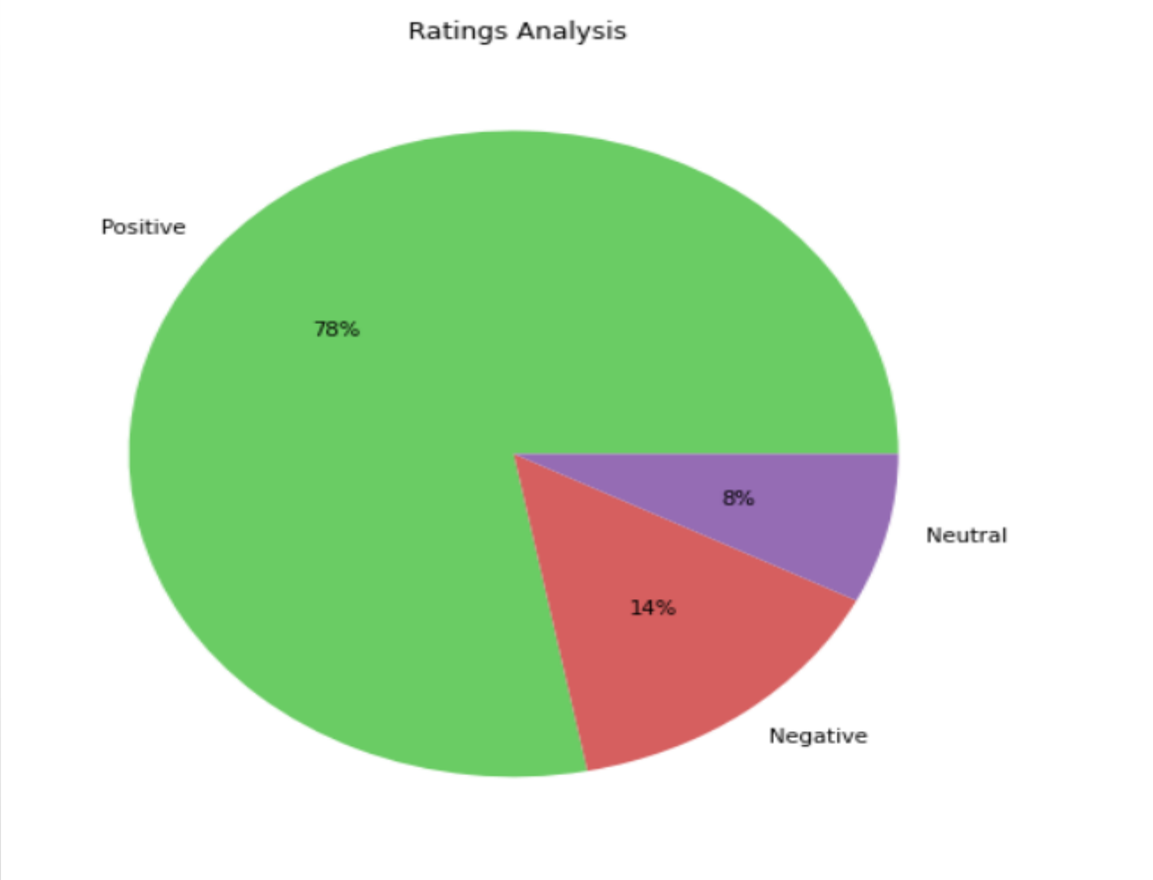
STEP 11: VISUALIZATION – PIE CHART

PIE CHART FOR RATINGS ANALYSIS



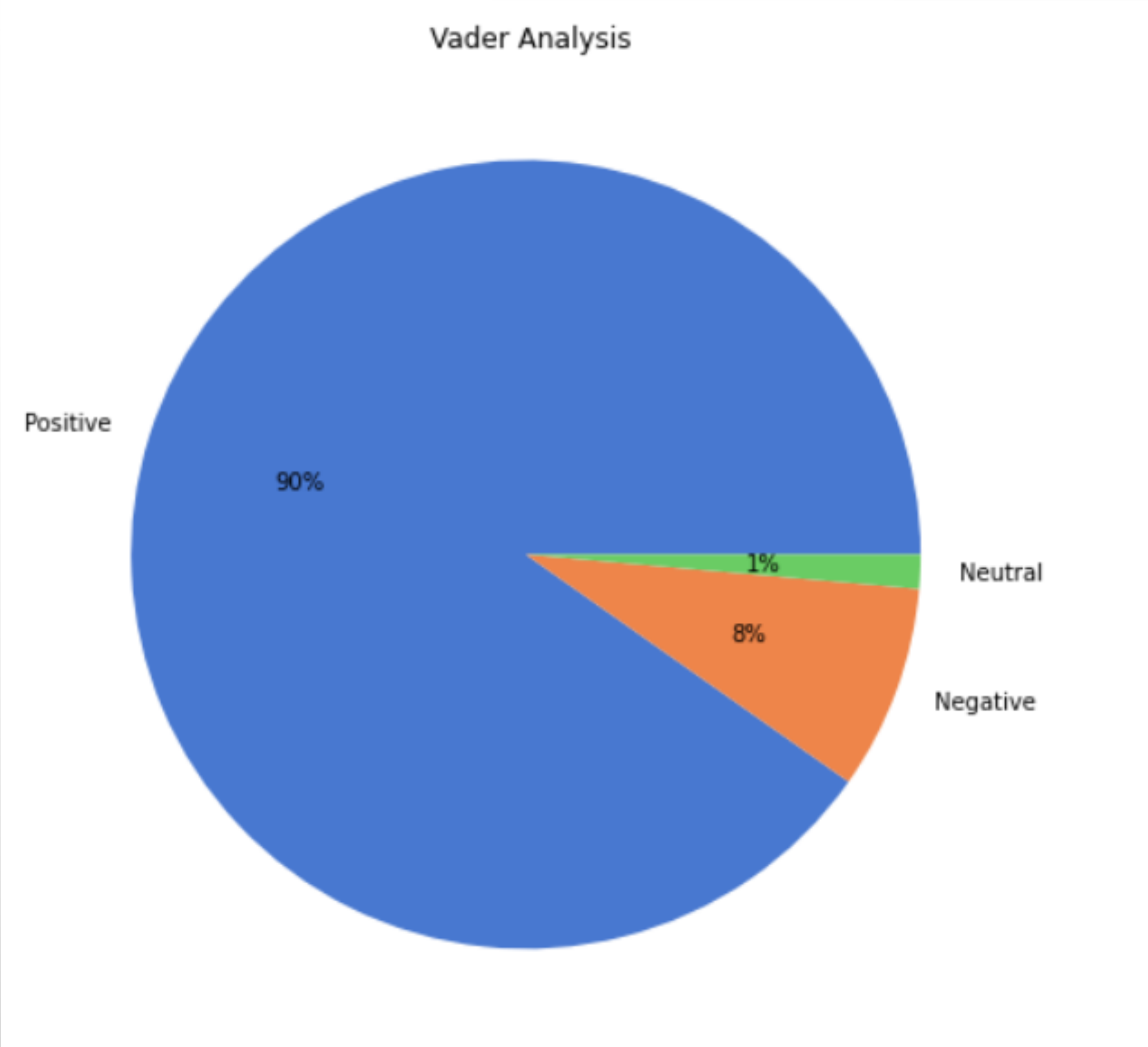
STEP 12:

PIE CHART FOR RATINGS ANALYSIS



STEP 13:

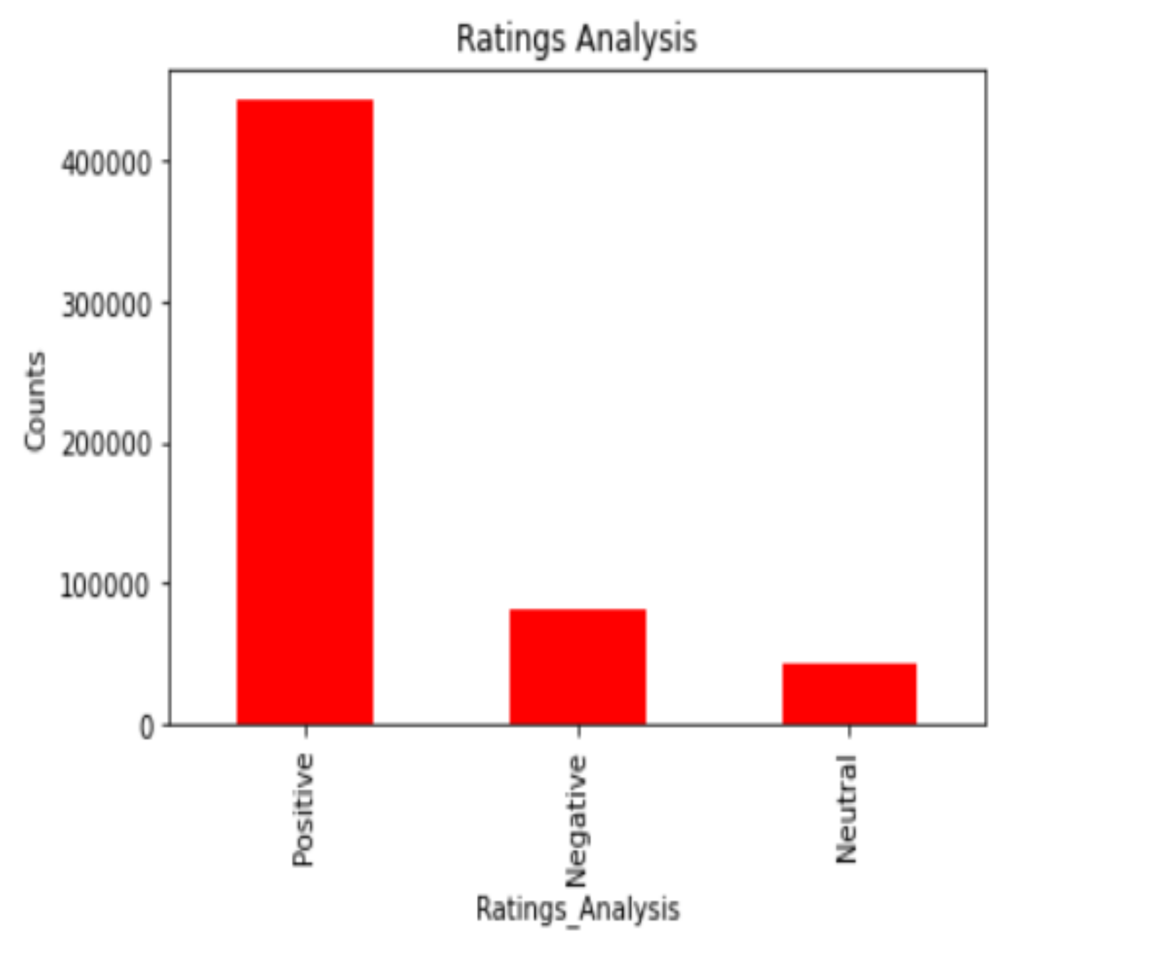
PIE CHART FOR VADER ANALYSIS



VISUALIZATION: BAR GRAPHS

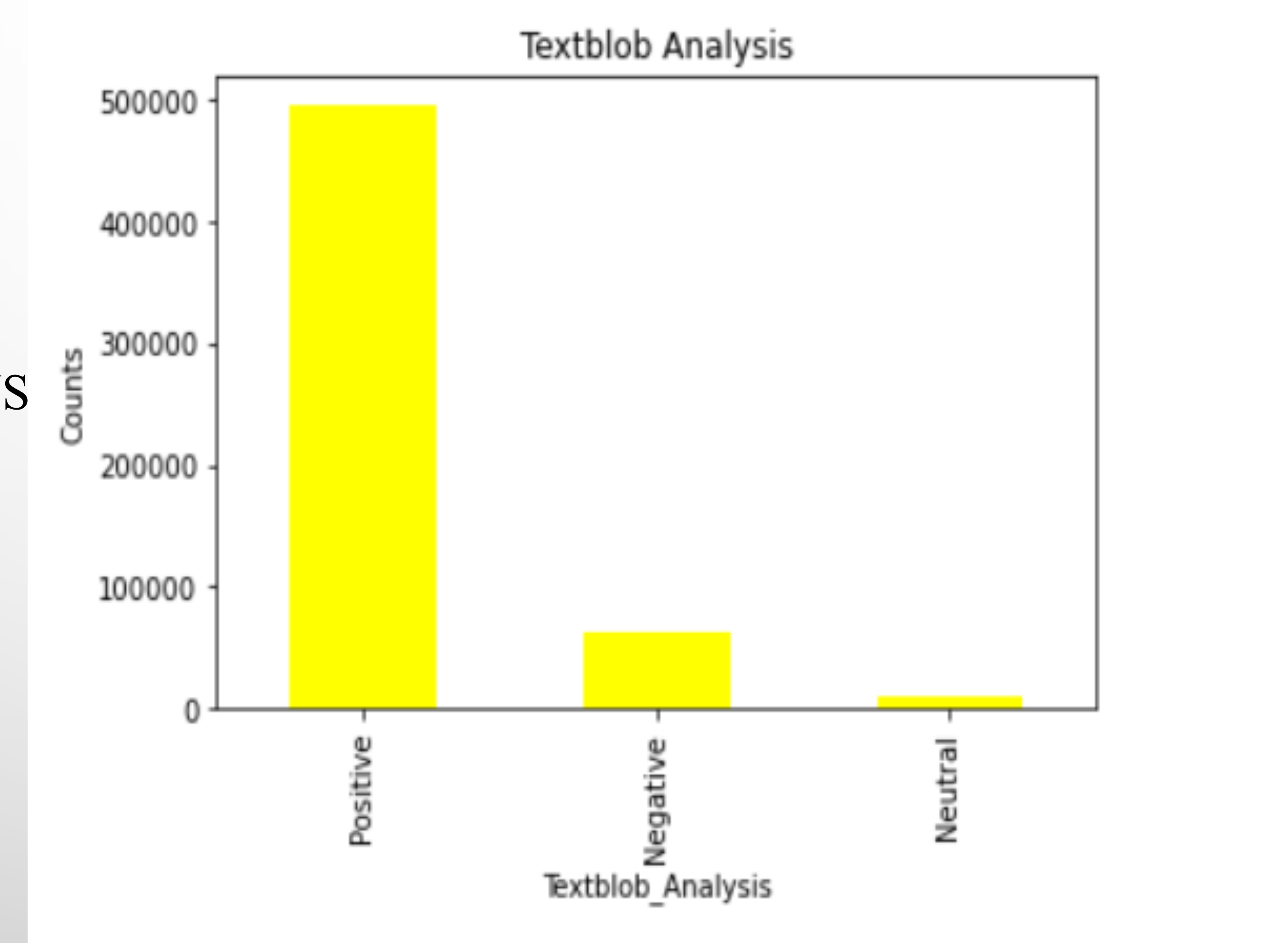
STEP 14:

BAR GRAPH FOR RATINGS ANALYSIS



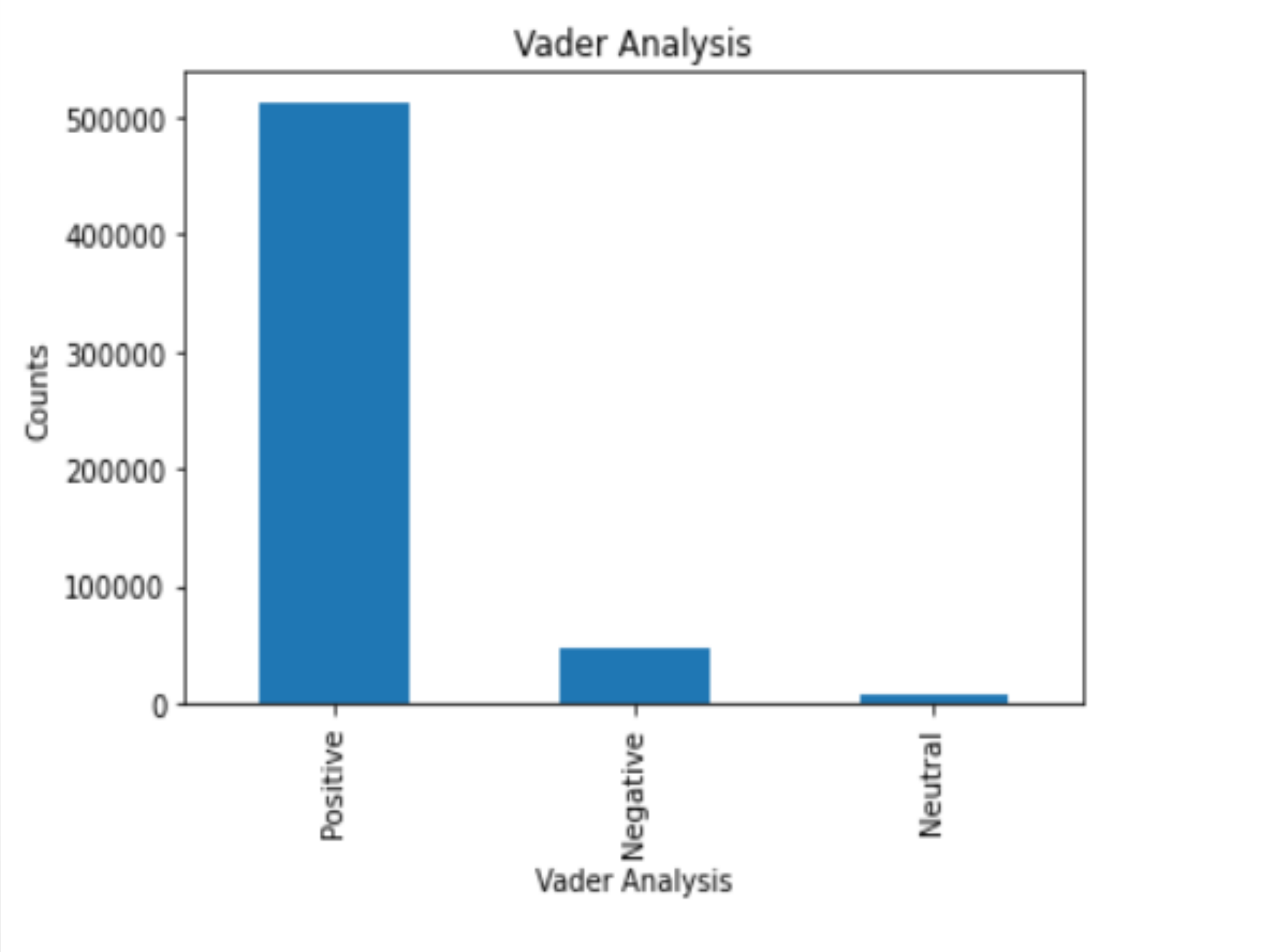
STEP 15:

BAR CHART FOR TEXTBLOB ANALYS



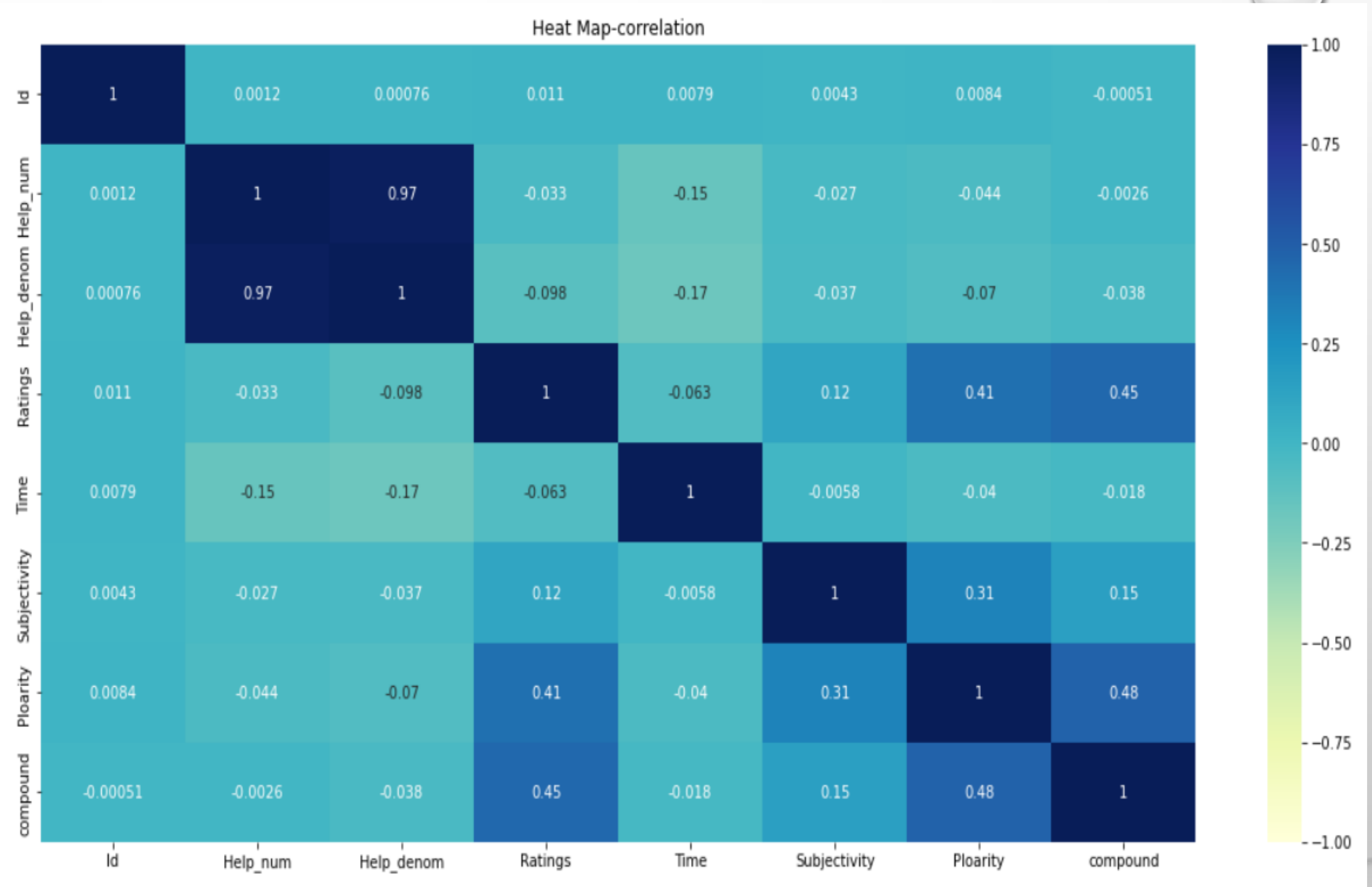
STEP 16:

BAR CHART FOR VADER ANALYSIS



STEP 17:

HEATMAP FOR THE CO-RELATION



STEP 18:

WORDCLOUD FOR GENERIC, POSITIVE AND NEGATIVE REVIEWS

Positive Reviews

Negative Reviews

STEP 19: FIRST, WE REMOVE ALL THE UNNECESSARY COLUMNS WHICH ARE NOT GOING TO BE USED IN THE M.L MODELS.

STEP 20: THEN WE DISCARD THE REVIEWS WHICH ARE NEUTRAL.

STEP 21: THEN WE ASSIGN VALUES 0 AND 1 TO THE NEGATIVE AND POSITIVE REVIEWS RESPECTIVELY.

```
model_data= model_data[model_data['Textblob_Analysis'] != 'Neutral']
model_data= model_data[model_data['Vader_Analysis'] != 'Neutral']
```

```
Textblob_model=model_data
```

```
Textblob_model=Textblob_model.drop(columns=['Comments_Tokens','Ratings_Analysis','Vader_Analysis'])
```

```
def Sentiment(Textblob_Analysis):
    if Textblob_Analysis == 'Negative':
        return 0
    else:
        return 1

Textblob_model['Textblob_Analysis_Sentiment']= Textblob_model['Textblob_Analysis'].apply(Sentiment)
```

STEP 22: WE NOW INSERT THE TEXT FROM THE REVIEWS IN X_TRAIN AND WE INSERT THE SENTIMENT SCORE IN THE Y_TRAIN NOW TO FIT THE DATA IN THE MACHINE LEARNING MODEL WE HAVE TRANSFORMED THE DATA BY USING TF-IDF, USING THIS, EACH WORD IN THE STRING HAS BEEN ASSIGNED A NUMERICAL VALUE SO THAT IT FITS PERFECTLY IN THE MACHINE LEARNING MODEL.

#We are using the TF-IDF ON Comments column and transforming it to numerical values.

```
from joblib import parallel_backend
with parallel_backend('threading', n_jobs=-1):
    tfidf_tb = TfidfVectorizer()
    X_train_tfidf_tb = tfidf_tb.fit_transform(X_train)
```

```
X_test_tfidf_tb = tfidf_tb.transform(X_test.astype('U'))
```

MACHINE LEARNING ALGORITHM APPLICATION

STEP 23: APPLYING XGBOOST ALGORITHM

ML ALGORITHM XGBOOST

```
from xgboost import XGBClassifier
```

```
xgboost_model = XGBClassifier()
xgboost_model.fit(X_train_tfidf_tb, y_train)
xgboost_pred = xgboost_model.predict(X_test_tfidf_tb)
print(xgboost_pred)
```

```
[1 1 1 ... 1 1 1]
```

```
xgboost_accuracy=accuracy_score(y_test,xgboost_pred)
```

```
print("XGBoost Accuracy for Textblob Analysis is:",xgboost_accuracy*100)
```

```
XGBoost Accuracy for Textblob Analysis is: 95.14915576479109
```

ML ALGORITHMS FOR VADER ANALYSIS

```
xgboost_model_v = XGBClassifier()
```

```
xgboost_model_v.fit(X_train_tfidf_v, y_train)
xgboost_pred_v = xgboost_model_v.predict(X_test_tfidf_v)
```

```
xgboost_accuracy_v=accuracy_score(y_test,xgboost_pred_v)
```

```
print("XGBoost Accuracy for Vader Analysis is:",xgboost_accuracy_v*100)
```

```
XGBoost Accuracy for Vader Analysis is: 94.55343805169527
```

STEP 24: APPLYING DECISION TREE ALGORITHM

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier

Decisiontree_model= DecisionTreeClassifier(random_state=34, max_depth=80)

Decisiontree_model.fit(X_train_tfidf_tb,y_train)

56]: DecisionTreeClassifier(max_depth=80, random_state=34)

Decisiontree_pred = Decisiontree_model.predict(X_test_tfidf_tb)

Decisiontree_accuracy=accuracy_score(y_test,Decisiontree_pred)

print("Decision Tree Accuracy for Textblob Analysis is:",Decisiontree_accuracy*100)

Decision Tree Accuracy for Textblob Analysis is: 94.28907699968313
```

Decision Tree

```
Decisiontree_model_v= DecisionTreeClassifier(random_state=34, max_depth=80)

Decisiontree_model_v.fit(X_train_tfidf_v,y_train)

6]: DecisionTreeClassifier(max_depth=80, random_state=34)

Decisiontree_pred_v = Decisiontree_model_v.predict(X_test_tfidf_v)

decisiontree_accuracy_v=accuracy_score(Decisiontree_pred_v,y_test)

print("Decision Tree Accuracy for Vader Analysis is:",decisiontree_accuracy_v*100)

Decision Tree Accuracy for Vader Analysis is: 94.26101127155854
```

STEP 25: APPLYING RANDOM FOREST ALGORITHM

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
```

```
Randomforest_model= RandomForestClassifier(n_estimators=90, max_depth=150)
```

```
Randomforest_model.fit(X_train_tfidf_tb,y_train)
```

```
1]: RandomForestClassifier(max_depth=150, n_estimators=90)
```

```
randomtree_pred=Randomforest_model.predict(X_test_tfidf_tb)
```

```
Randomtree_accuracy=accuracy_score(y_test,randomtree_pred)
```

```
print("Random Tree Accuracy for Textblob Analysis is:",Randomtree_accuracy*100)
```

```
Random Tree Accuracy for Textblob Analysis is: 92.13254266443349
```

Random Forest

```
Randomforest_model_v= RandomForestClassifier(n_estimators=90, max_depth=150)
```

```
Randomforest_model_v.fit(X_train_tfidf_v,y_train)
```

```
1]: RandomForestClassifier(max_depth=150, n_estimators=90)
```

```
Randomforest_pred_v = Randomforest_model_v.predict(X_test_tfidf_v)
```

```
Randomforest_accuracy_v=accuracy_score(Randomforest_pred_v,y_test)
```

```
print("Random Forest Accuracy for Vader Analysis is:",Randomforest_accuracy_v*100)
```

```
Random Forest Accuracy for Vader Analysis is: 93.52224887963423
```

COMPARISONS / CONCLUSION:

XGBOOST - TEXTBLOB ANALYSIS - 95.14%
XGBOOST - VADER ANALYSIS - 94.53%
DECISION TREE - TEXTBLOB ANALYSIS - 94.28%
DECISION TREE - VADER ANALYSIS - 94.26%
RANDOM FOREST - TEXTBLOB ANALYSIS - 92.13%
RANDOM FOREST - VADER ANALYSIS - 93.52%

	ML Models	TB_Accuracy	Vader_Accuracy
0	XGBoost	95.149156	94.553438
1	DecisioN Tree	94.289077	94.261011
2	Random Forest	92.132543	93.522249

REFERENCES

HTTPS://MACHINELEARNINGMASTERY.COM/DEVELOP-FIRST-XGBOOST-MODEL-PYTHON-SCIKIT-LEARN/
HTTPS://YOUTU.BE/ANVRJNLKP0K
HTTPS://YOUTU.BE/TRNPSLOCBV0
HTTPS://YOUTU.BE/ALU_CCXNS-K

THANK YOU