```
 from google.colab import files
upload=files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving Corona_NLP_test.csv to Corona_NLP_test (2).csv

```
import pandas as pd
import numpy as np
#import neattext.functions as nfx
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
True
```

```
test=pd.read_csv("Corona_NLP_test.csv")
test
```

	UserName	ScreenName	Location	TweetAt	OriginalTweet	Sentiment
0	1	44953	NYC	02-03-2020	TRENDING: New Yorkers encounter empty supermar...	Extremely Negative
1	2	44954	Seattle, WA	02-03-2020	When I couldn't find hand sanitizer at Fred Me...	Positive
2	3	44955	NaN	02-03-2020	Find out how you can protect yourself and love...	Extremely Positive
3	4	44956	Chicagoland	02-03-2020	#Panic buying hits #NewYork City as anxious sh...	Negative
4	5	44957	Melbourne, Victoria	03-03-2020	#toiletpaper #dunnypaper #coronavirus #coronav...	Neutral
...	...	...	...	...	...	...
3793	3794	48746	Israel ??	16-03-2020	Meanwhile In A Supermarket in Israel -- People...	Positive

```
##checking the datatypes of the objects
```

```
test.dtypes
```

```
UserName          int64
ScreenName        int64
Location         object
TweetAt          object
OriginalTweet    object
Sentiment        object
dtype: object
```

```
### dropung the username and screen test columns
```

```
test=test.drop(["UserName","ScreenName"],axis=1)
test
```

	Location	TweetAt	OriginalTweet	Sentiment
0	NYC	02-03-2020	TRENDING: New Yorkers encounter empty supermar...	Extremely Negative
1	Seattle, WA	02-03-2020	When I couldn't find hand sanitizer at Fred Me...	Positive
2	NaN	02-03-2020	Find out how you can protect yourself and love...	Extremely Positive
3	Chicagoland	02-03-2020	#Panic buying hits #NewYork City as anxious sh...	Negative
	Melbourne	02-03	#toiletpaper #dunnypaper #coronavirus	

```
##checking the info
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3798 entries, 0 to 3797
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   Location       2964 non-null   object
 1   TweetAt        3798 non-null   object
 2   OriginalTweet  3798 non-null   object
 3   Sentiment      3798 non-null   object
dtypes: object(4)
memory usage: 118.8+ KB
```

```
###checking the null values
```

```
test.isna().sum()
```

```
Location         834
TweetAt            0
OriginalTweet      0
Sentiment          0
dtype: int64
```

```
test.isnull().sum()
```

```
Location         834
TweetAt            0
OriginalTweet      0
Sentiment          0
dtype: int64
```

```
test.drop_duplicates()
test.shape
```

```
(3798, 4)
```

```
test.describe()
```

	Location	TweetAt	OriginalTweet	Sentiment
count	2964	3798	3798	3798
unique	1717	15	3798	5
top	United States	13-03-2020	TRENDING: New Yorkers encounter empty supermar...	Negative
freq	75	1233	1	1041

```
### Checkint the unique location list
```

```
print(test["Location"])
```

```
0                      NYC
1              Seattle, WA
2                      NaN
3             Chicagoland
4       Melbourne, Victoria
               ...
3793             Israel ??
3794        Farmington, NM
```

```
3795          Haverford, PA
3796                   NaN
3797    Arlington, Virginia
Name: Location, Length: 3798, dtype: object
```

```
test.Location.mode()  ## checking highest repaeating state
```

```
0    United States
Name: Location, dtype: object
```

```
test1=test.sort_values("TweetAt")
test1
```

	Location	TweetAt	OriginalTweet	Sentiment
0	NYC	02-03-2020	TRENDING: New Yorkers encounter empty supermar...	Extremely Negative
1	Seattle, WA	02-03-2020	When I couldn't find hand sanitizer at Fred Me...	Positive
2	NaN	02-03-2020	Find out how you can protect yourself and love...	Extremely Positive
3	Chicagoland	02-03-2020	#Panic buying hits #NewYork City as anxious sh...	Negative
4	Melbourne, Victoria	03-03-2020	#toiletpaper #dunnypaper #coronavirus #coronav...	Neutral
...	...	...	...	...
3476	-25.992566, 27.986270	16-03-2020	Amazon delivery infrastructure strained as COV...	Negative
3475	London	16-03-2020	I hear online delivery slots for supermarkets ...	Extremely Positive

```
test1["Location"].mode() ## heighest repeated city
```

```
0    United States
Name: Location, dtype: object
```

```
test[test["Location"]=="United States"]
```

	Location	TweetAt	OriginalTweet	Sentiment
29	United States	07-03-2020	#Coronavirus is "an exposure of all the holes ...	Positive
48	United States	09-03-2020	Global stocks plummeted today due to fear surr...	Extremely Negative
96	United States	10-03-2020	What You Need If Quarantined at Home | #Corona...	Neutral
110	United States	11-03-2020	@realDonaldTrump What can you do about #COVID2...	Negative
112	United States	11-03-2020	Pretty good chance that your waiter/waitress, ...	Extremely Positive
...	...	...	...	...
3322	United States	15-03-2020	When state of emergency is called, schools shu...	Extremely Negative
3328	United States	16-03-2020	@BernardKerik @realDonaldTrump COVID 19 is doi...	Extremely Negative

```
test["Location"].value_counts()
```

```
United States             75
London, England           48
Washington, DC            38
New York, NY              34
Los Angeles, CA           33
                          ..
India, Belgium.            1
```

```
Boston, Montréal, Paris     1
Annapolis, MD               1
here and there              1
Haverford, PA               1
Name: Location, Length: 1717, dtype: int64
```

```
test["Sentiment"].value_counts()
```

```
Negative              1041
Positive               947
Neutral                619
Extremely Positive     599
Extremely Negative     592
Name: Sentiment, dtype: int64
```

```
X1=test[test["Sentiment"]=="Positive"]
X1["Location"].value_counts()
```

```
United States                    22
Washington, DC                   15
London, England                  13
California, USA                   8
Los Angeles, CA                   8
                                 ..
Brighton, England                 1
Orange & Los Angeles Counties     1
Ipswich, England                  1
Here and there. Usually NYC.      1
Israel ??                         1
Name: Location, Length: 540, dtype: int64
```

```
X2=test[test["Sentiment"]=="Negative"]
X2["Location"].value_counts()
```

```
United States                20
New York, NY                 13
London, England              12
London                        9
Canada                        9
                             ..
California ! in Wonderland    1
Bellevue, Washington          1
Port-au-Prince, Haïti         1
Colombo                       1
Farmington, NM                1
Name: Location, Length: 592, dtype: int64
```

```
X3=test[test["Sentiment"]=="Neutral"]
X3["Location"].value_counts()
```

```
Los Angeles, CA    12
London, England    12
United States      11
Toronto             8
United Kingdom      6
                   ..
Geneva              1
Nashville           1
Pacific NW          1
21113               1
Haverford, PA       1
Name: Location, Length: 373, dtype: int64
```

```
X4=test[test["Sentiment"]=="Extremely Positive"]
X4["Location"].value_counts()
```

```
United States          10
London, England         9
San Francisco, CA       8
London                  7
USA                     6
                       ..
Wien, Österreich        1
Jacksonville, FL        1
maryland                1
Austin, TX, USA         1
Arlington, Virginia     1
Name: Location, Length: 363, dtype: int64
```

```
X5=test[test["Sentiment"]=="Extremely Negative"]
X5["Location"].value_counts()
```

```
United States          12
Toronto, Ontario        9
Washington, DC          8
California, USA         7
Canada                  7
                       ..
Delft/Amsterdam, NL     1
Menasha, WI             1
Singapore               1
Hyderabad, India        1
Washington D.C.         1
Name: Location, Length: 353, dtype: int64
```

by obsserving the all sentiment analysing the the top positive rate occurs in united state

```
test["Location"].value_counts().nlargest(30)   ###top 30 locations
```

```
United States              75
London, England            48
Washington, DC             38
New York, NY               34
Los Angeles, CA            33
Toronto, Ontario           29
Canada                     29
California, USA            26
London                     25
Toronto                    21
USA                        20
San Francisco, CA          19
Atlanta, GA                19
United Kingdom             18
Texas, USA                 17
Ireland                    17
New York, USA              16
Los Angeles                16
Chicago, IL                15
India                      15
UK                         14
Ontario, Canada            13
Montréal, Québec           12
New York City              12
Austin, TX                 11
Australia                  11
Washington, D.C.           11
England, United Kingdom    10
NYC                        10
Florida, USA               10
Name: Location, dtype: int64
```

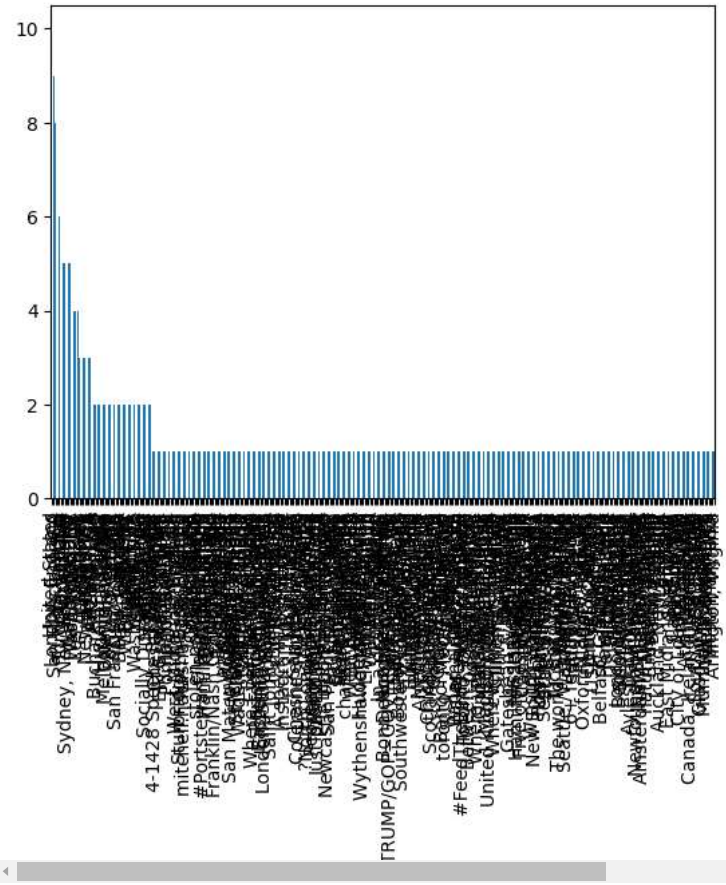
plotting the bar graph to see the top 30 cities

```
test["Location"].value_counts().nlargest(30).plot(kind='bar')
```

```
<Axes: >
```


```
X4["Location"].value_counts().plot(kind="bar")
```

```
<Axes:
>/usr/local/lib/python3.10/dist-packages/IPython/core/events.py:89: UserWarning: Glyph 1
  func(*args, **kwargs)
/usr/local/lib/python3.10/dist-packages/IPython/core/events.py:89: UserWarning: Glyph 14
  func(*args, **kwargs)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Gly
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Gly
  fig.canvas.print_figure(bytes_io, **kw)
```




```
###cleaning  the orginal tweet data    E## text analysis


test["OriginalTweet"]

0       TRENDING: New Yorkers encounter empty supermar...
1       When I couldn't find hand sanitizer at Fred Me...
2       Find out how you can protect yourself and love...
3       #Panic buying hits #NewYork City as anxious sh...
4       #toiletpaper #dunnypaper #coronavirus #coronav...
                              ...
3793    Meanwhile In A Supermarket in Israel -- People...
3794    Did you panic buy a lot of non-perishable item...
3795    Asst Prof of Economics @cconces was on @NBCPhi...
3796    Gov need to do somethings instead of biar je r...
3797    I and @ForestandPaper members are committed to...
Name: OriginalTweet, Length: 3798, dtype: object
```

```
test["OriginalTweet"].unique()
```

```
array(['TRENDING: New Yorkers encounter empty supermarket shelves (pictured, Wegmans in Brooklyn), sold-out online grocers (FoodKick,
MaxDelivery) as #coronavirus-fearing shoppers stock up https://t.co/Gr76pcrLWh https://t.co/ivMKMsqdT1',
       "When I couldn't find hand sanitizer at Fred Meyer, I turned to #Amazon. But $114.97 for a 2 pack of Purell??!!Check out how
#coronavirus concerns are driving up prices. https://t.co/ygbipBflMY",
       'Find out how you can protect yourself and loved ones from #coronavirus. ?',
       ...,
       "Asst Prof of Economics @cconces was on @NBCPhiladelphia talking about her recent research on coronavirus' impact on the
economy. Watch it here (starting at :33): https://t.co/8tfYNoro5l",
       "Gov need to do somethings instead of biar je rakyat assume 'lockdown' ke or even worst. Harini semua supermarket crowded like
hell. Lagi mudah virus tu tersebar ?? #COVID2019",
       'I and @ForestandPaper members are committed to the safety of our employees and our end-users. We are monitoring COVID-19. Rest
assured that tissue manufacturers are continuing to produce and ship products.  https://t.co/qF6hclCAEq https://t.co/xyvbNsFeXA'],
      dtype=object)
```

```
pip install neattext
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: neattext in /usr/local/lib/python3.10/dist-packages (0.1.3)
```

```
import neattext.functions as nfx
```

```
dir(nfx)
```

```
['BTC_ADDRESS_REGEX',
 'CURRENCY_REGEX',
 'CURRENCY_SYMB_REGEX',
 'Counter',
 'DATE_REGEX',
 'EMAIL_REGEX',
 'EMOJI_REGEX',
 'HASTAG_REGEX',
 'MASTERCard_REGEX',
 'MD5_SHA_REGEX',
 'MOST_COMMON_PUNCT_REGEX',
 'NUMBERS_REGEX',
 'PHONE_REGEX',
 'PoBOX_REGEX',
 'SPECIAL_CHARACTERS_REGEX',
 'STOPWORDS',
 'STOPWORDS_de',
 'STOPWORDS_en',
 'STOPWORDS_es',
 'STOPWORDS_fr',
 'STOPWORDS_ru',
 'STOPWORDS_yo',
 'STREET_ADDRESS_REGEX',
 'TextFrame',
 'URL_PATTERN',
 'USER_HANDLES_REGEX',
 'VISACard_REGEX',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__generate_text',
 '__loader__',
 '__name__',
 '__numbers_dict',
 '__package__',
 '__spec__',
 '_lex_richness_herdan',
 '_lex_richness_maas_ttr',
 'clean_text',
 'defaultdict',
 'digit2words',
 'extract_btc_address',
 'extract_currencies',
 'extract_currency_symbols',
 'extract_dates',
 'extract_emails',
 'extract_emojis',
 'extract_hashtags',
 'extract_html_tags',
 'extract_mastercard_addr',
 'extract_md5sha',
 'extract_numbers',
 'extract_pattern',
 'extract_phone_numbers',
 'extract_postoffice_box',
```

```
        'extract_shortwords',
        'extract_special_characters'
```

```
test["OriginalTweet"].loc[2]
```

```
'Find out how you can protect yourself and loved ones from #coronavirus. ?'
```

```
test["OriginalTweet"][656]
```

```
'1. A well-stocked supermarket.\r\r\n2. The same supermarket just 2 minutes later.\r\r
\n\r\r\nShopping is like a military operation at the minute.\r\r\n#sainsburys #Aldi #mo
rrisons #panicbuying #panicbuyinguk #CoronavirusPandemic #Covid 19 #coronavirusinUK htt
```

```
test["OriginalTweet"].shape
```

```
(3798,)
```

```
## removing
 #1.special chars
 #2.urls
 #3.hasatags
```

```
from neattext.functions.functions import remove_hashtags
test["cleaned_tweet"]= test["OriginalTweet"].apply(nfx.remove_hashtags)
```

```
test.head()
```

	Location	TweetAt	OriginalTweet	Sentiment	cleaned_tweet
0	NYC	02-03-2020	TRENDING: New Yorkers encounter empty supermar...	Extremely Negative	TRENDING: New Yorkers encounter empty supermar...
1	Seattle, WA	02-03-2020	When I couldn't find hand sanitizer at Fred Me...	Positive	When I couldn't find hand sanitizer at Fred Me...
2	NaN	02-03-2020	Find out how you can protect yourself and love...	Extremely Positive	Find out how you can protect yourself and love...

```
test["cleaned_tweet"]= test["OriginalTweet"].apply(nfx.remove_numbers)
```

```
test["cleaned_tweet"]= test["cleaned_tweet"].apply(nfx.remove_urls)
```

```
test["cleaned_tweet"]= test["cleaned_tweet"].apply(lambda x: nfx.remove_userhandles(x))
```

```
test["cleaned_tweet"]= test["cleaned_tweet"].apply(nfx.clean_text)
```

```
test["cleaned_tweet"]= test["cleaned_tweet"].apply(nfx.remove_special_characters)
```

```
test["cleaned_tweet"]= test["cleaned_tweet"].apply(nfx.remove_stopwords)
```

```
test["cleaned_tweet"]= test["cleaned_tweet"].apply(nfx.remove_punctuations)
```

```
test["cleaned_tweet"].loc[656]
```

```
'wellstocked supermarket supermarket minutes later shopping like military operation min
ute sainsburys aldi morrisons panicbuying panicbuyinguk coronaviruspandemic covid coron
avirusinuk'
```

```
test.head()
```

	Location	TweetAt	OriginalTweet	Sentiment	cleaned_tweet
			TRENDING: New Yorkers		trending new yorkers

Replacing the extreme positve with posive and alao repalcing with Extrme negitive as negative

```
test["Sentiment"]=test["Sentiment"].str.replace("Extremely Negative","Negative")
test["Sentiment"]=test["Sentiment"].str.replace("Extremely Positive","Positive")
test.head()
```

	Location	TweetAt	OriginalTweet	Sentiment	cleaned_tweet
0	NYC	02-03-2020	TRENDING: New Yorkers encounter empty supermar...	Negative	trending new yorkers encounter supermarket she...
1	Seattle, WA	02-03-2020	When I couldn't find hand sanitizer at Fred Me...	Positive	find hand sanitizer fred meyer turned amazon p...
2	NaN	02-03-2020	Find out how you can protect yourself and love...	Positive	find protect loved ones coronavirus

```
#comparing both original tweet and cleaned tweet
```

```
test[["OriginalTweet","cleaned_tweet"]]
```

	OriginalTweet	cleaned_tweet
0	TRENDING: New Yorkers encounter empty supermar...	trending new yorkers encounter supermarket she...
1	When I couldn't find hand sanitizer at Fred Me...	find hand sanitizer fred meyer turned amazon p...
2	Find out how you can protect yourself and love...	find protect loved ones coronavirus
3	#Panic buying hits #NewYork City as anxious sh...	panic buying hits newyork city anxious shopper...
4	#toiletpaper #dunnypaper #coronavirus #coronav...	toiletpaper dunnypaper coronavirus coronavirus...
...	...	...
3793	Meanwhile In A Supermarket in Israel -- People...	supermarket israel people dance sing stay posi...
3794	Did you panic buy a lot of non-perishable item...	panic buy lot nonperishable items echo needs f...
3795	Asst Prof of Economics @cconces was on	asst prof economics talking recent research

sentimental analysis, for sentimental analysis text blob is used to find sentiment of the sentence it is positive or negative

```
from textblob import TextBlob
```

```
from textblob.en import Sentiment
def get_sentiment(test):
  blob= TextBlob(test)
  polarity_score=blob.sentiment.polarity
  subjectivity_score=blob.sentiment.subjectivity
  if polarity_score>0:
    sentiment_score="positive"
  elif polarity_score==0:
    sentiment_score="neatrul"
  elif polarity_score<0:
    sentiment_score="negative"
  else:
    entiment_score="neatrul"
  result={"polarity":polarity_score,
          "subjectivity":subjectivity_score,
          "sentiment":sentiment_score}
  return result
```

```
get_sentiment(test["cleaned_tweet"].loc[344])
```

```
{'polarity': -0.016666666666666666,
 'subjectivity': 0.375,
 'sentiment': 'negative'}
```

```
get_sentiment(test["cleaned_tweet"].loc[78])
```

```
{'polarity': 0.01666666666666669,
 'subjectivity': 0.6833333333333333,
 'sentiment': 'positive'}
```

```
get_sentiment(test["cleaned_tweet"].loc[98])
```

```
{'polarity': 0.0, 'subjectivity': 0.0, 'sentiment': 'neatrul'}
```

```
test["cleaned_tweet"].loc[98]
```

```
'spiking prices state emergency crime report coronavirus covid'
```

```
test["sentiment_resullts"]=test["cleaned_tweet"].apply(get_sentiment)  ##storing  the all sentiment analysis is one table
```

```
test["sentiment_resullts"].value_counts()
```

```
{'polarity': 0.0, 'subjectivity': 0.0, 'sentiment': 'neatrul'}                                  882
{'polarity': -0.03333333333333333, 'subjectivity': 0.0, 'sentiment': 'negative'}                 84
{'polarity': 0.0, 'subjectivity': 0.1, 'sentiment': 'neatrul'}                                   49
{'polarity': 0.5, 'subjectivity': 0.5, 'sentiment': 'positive'}                                  36
{'polarity': -0.5, 'subjectivity': 1.0, 'sentiment': 'negative'}                                 33
                                                                                               ...
{'polarity': -0.08333333333333334, 'subjectivity': 0.9166666666666667, 'sentiment': 'negative'}    1
{'polarity': 0.5833333333333333, 'subjectivity': 0.5166666666666667, 'sentiment': 'positive'}     1
{'polarity': 0.27777777777777773, 'subjectivity': 0.3666666666666667, 'sentiment': 'positive'}    1
{'polarity': 0.3475, 'subjectivity': 0.7816666666666666, 'sentiment': 'positive'}                 1
{'polarity': 0.125, 'subjectivity': 0.35, 'sentiment': 'positive'}                               1
Name: sentiment_resullts, Length: 1727, dtype: int64
```

```
###creating a data dramme for the sentiment analysis
```

```
pd.json_normalize(test["sentiment_resullts"])
```

	polarity	subjectivity	sentiment
0	0.136364	0.454545	positive
1	0.000000	0.000000	neatrul
2	0.700000	0.800000	positive
3	0.075000	1.000000	positive
4	-0.033333	0.000000	negative
...	...	...	...
3793	0.227273	0.545455	positive
3794	0.500000	0.888889	positive
3795	0.000000	0.175000	neatrul
3796	-1.000000	1.000000	negative
3797	0.000000	0.000000	neatrul

3798 rows × 3 columns

```
## adding this data frame into main datarame
test=test.join(pd.json_normalize(test["sentiment_resullts"]))
```

```
test.head()
```

	Location	TweetAt	OriginalTweet	Sentiment	cleaned_tweet	sentiment_resullts	p
0	NYC	02-03-2020	TRENDING: New Yorkers encounter empty supermar...	Negative	trending new yorkers encounter supermarket she...	{'polarity': 0.13636363636363635, 'subjectivit...	
1	Seattle, WA	02-03-2020	When I couldn't find hand sanitizer at Fred Me...	Positive	find hand sanitizer fred meyer turned amazon p...	{'polarity': 0.0, 'subjectivity': 0.0, 'sentim...	
			Find out how		find protect	{'polarity': 0.7	

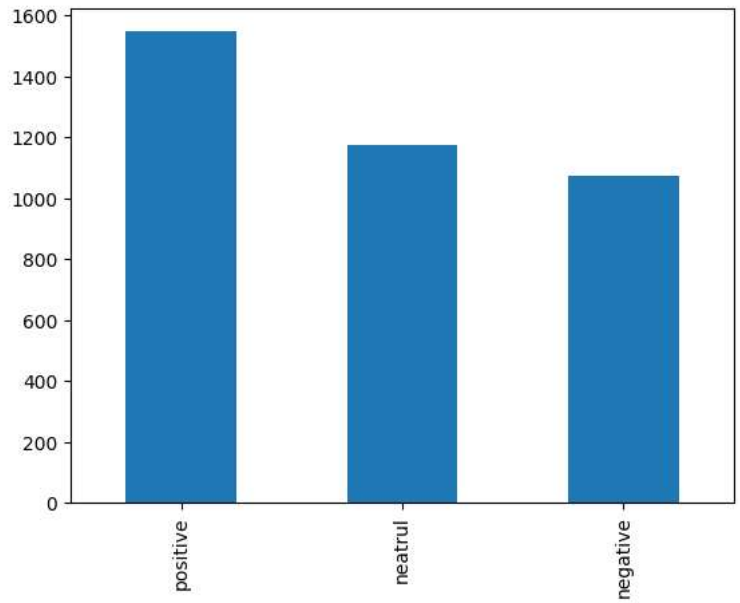
```
test["sentiment"].value_counts()
```

```
positive    1547
neatrul     1176
negative    1075
Name: sentiment, dtype: int64
```

plotting a bar graph between the sentiments(positive,negative,neatrul

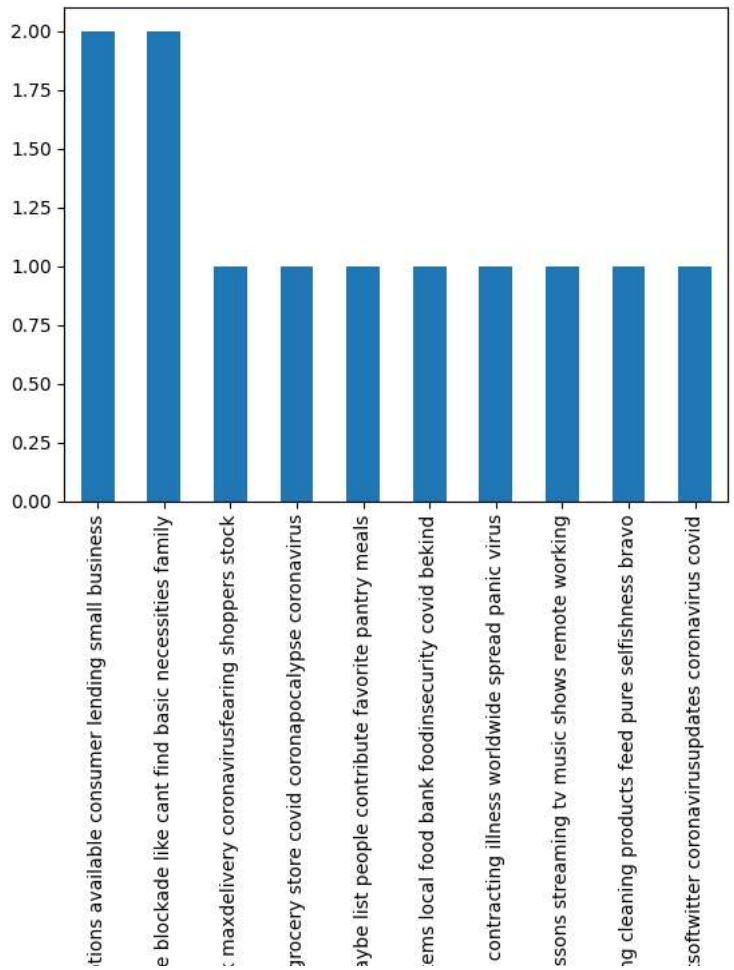
```
test["sentiment"].value_counts().plot(kind="bar")
```

```
<Axes: >
```

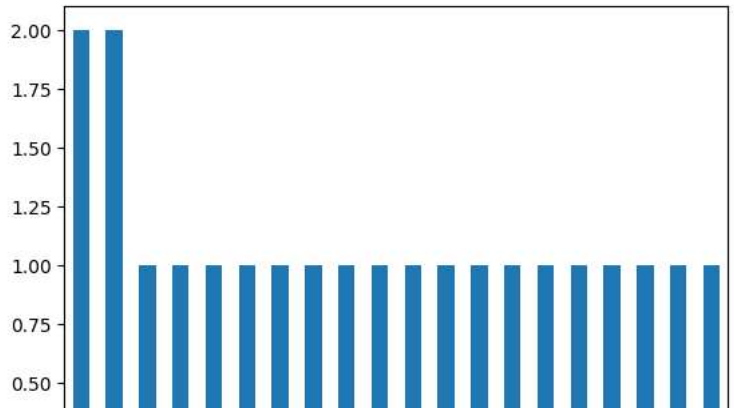


accesing the ony positive tweets, accessing onlu negative tweets, accessing only neatrul tweets

```
positive_tweets=test[test["sentiment"]=="positive"]["cleaned_tweet"].value_counts().nlargest(10).plot(kind="bar")
#negative_tweets=test[test["sentiment"]=="negative"]["cleaned_tweet"].value_counts().nlargest(10).plot(kind="bar")
#neatrul_tweets=test[test["sentiment"]=="neatrul"]["cleaned_tweet"].value_counts().nlargest(10).plot(kind="bar")
```



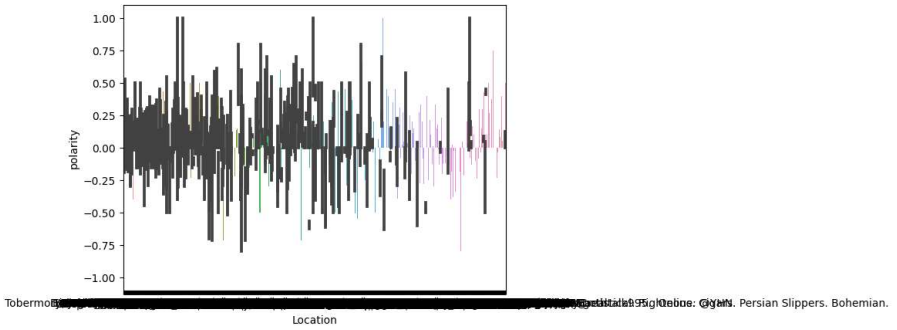

```
negative_tweets=test[test["sentiment"]=="negative"]["cleaned_tweet"].value_counts().nlargest(20).plot(kind="bar")
```




```
neatrul_tweets=test[test["sentiment"]=="neatrul"]["cleaned_tweet"].value_counts().nlargest(10).plot(kind="bar")
```

```
## ploting a bar graph between locating and based on their polarity score

sns.barplot(x="Location",y="polarity",data=test)
```

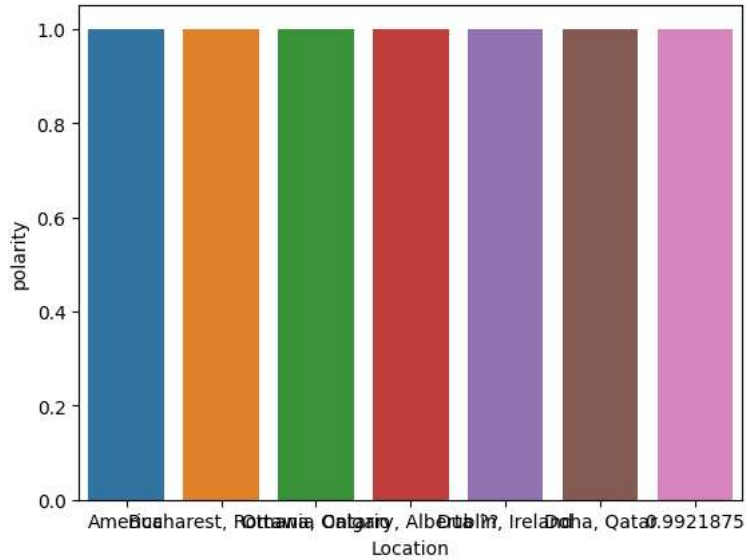
```
<Axes: xlabel='Location',
ylabel='polarity'>/usr/local/lib/python3.10/dist-packages/IPython/core/events.py:89: Use
  func(*args, **kwargs)
/usr/local/lib/python3.10/dist-packages/IPython/core/events.py:89: UserWarning: Glyph 14
  func(*args, **kwargs)
/usr/local/lib/python3.10/dist-packages/IPython/core/events.py:89: UserWarning: Glyph 14
  func(*args, **kwargs)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Gly
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Gly
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Gly
  fig.canvas.print_figure(bytes_io, **kw)
```



##plotting a graph to find top 10 positivc cities

```
sns.barplot(x="Location",y="polarity",data=test.nlargest(10,"polarity"))
```

```
<Axes: xlabel='Location', ylabel='polarity'>
```



#plottig a grpah to find top 10 negitive cities

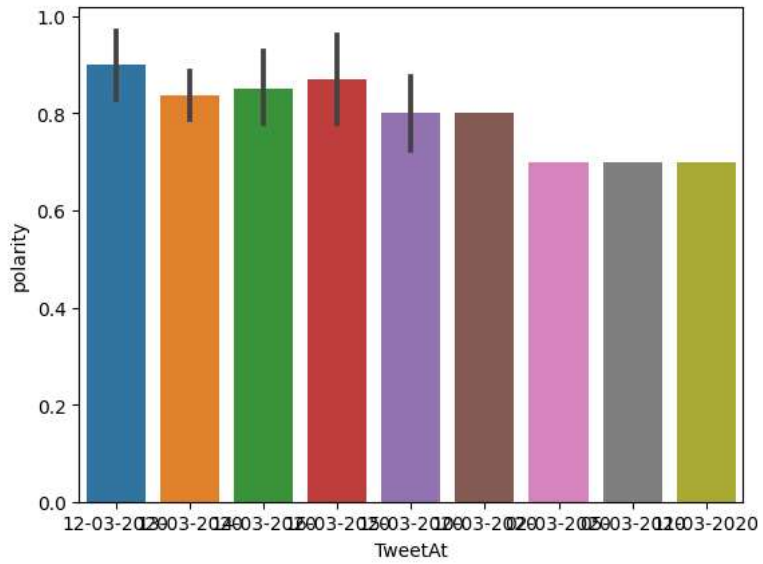
```
sns.barplot(x="Location",y="polarity",data=test[test["sentiment"]=="negative"].nlargest(10,"polarity"))
```

```
<Axes: xlabel='Location', ylabel='polarity'>
```


```
##plotting a graph to find top 50 dates which are get positve rate
```

```
sns.barplot(x="TweetAt",y="polarity",data=test[test["sentiment"]=="positive"].nlargest(50,"polarity"))
```

```
<Axes: xlabel='TweetAt', ylabel='polarity'>
```




```
## top 50 negative dates
```

```
sns.barplot(x="TweetAt",y="polarity",data=test[test["sentiment"]=="negative"].nlargest(50,"polarity"))
```

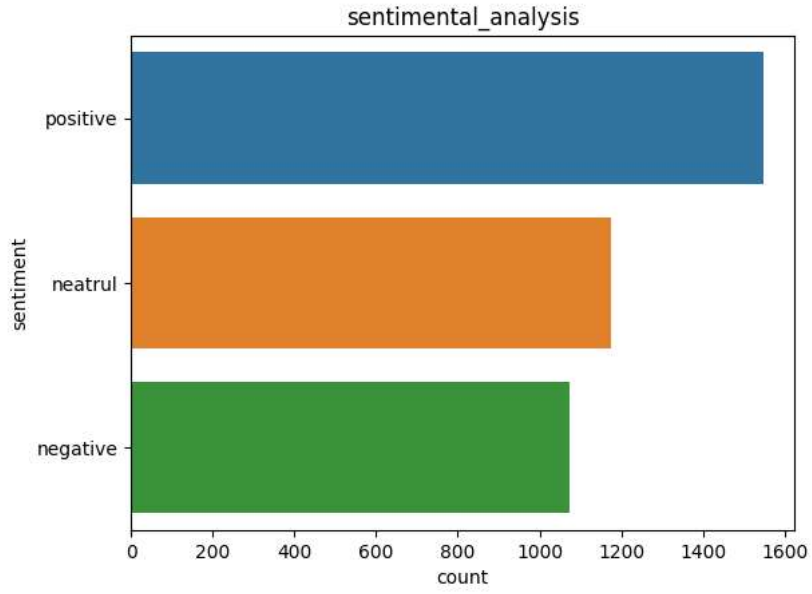
```
<Axes: xlabel='TweetAt', ylabel='polarity'>
```

```
total=test["cleaned_tweet"]
```

```
test["cleaned_tweet"][0]
```

```
'trending new yorkers encounter supermarket shelves pictured wegmans brooklyn soldout o
nline grocers foodkick maxdelivery coronavirusfearing shoppers stock'
```

```
sns.countplot(data=test,y="sentiment")
plt.title("sentimental_analysis")
plt.show()
```




```
##feature extraction
```

```
totaltext=" ".join(test["cleaned_tweet"])
totaltext
```

```
'trending new yorkers encounter supermarket shelves pictured wegmans brooklyn soldout o
nline grocers foodkick maxdelivery coronavirusfearing shoppers stock find hand sanitize
r fred meyer turned amazon pack purellcheck coronavirus concerns driving prices find pr
otect loved ones coronavirus panic buying hits newyork city anxious shoppers stock food
ampmedical supplies healthcare worker bigapple st confirmed coronavirus patient bloombe
rg staged event qanon qanon qanon election cdc toiletpaper dunnypaper coronavirus coron
avirusaustralia coronavirusupdate covid news corvid newsmelb dunnypapergate costco week
buying baby milk powder buying toilet paper remember time paid gallon regular gas los a
```

```
## tokenization
```

```
import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize
total_tokens=word_tokenize(totaltext)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
print(len(total_tokens))
```

```
65043
```

```
###normalzation  .. it used to make all  the words into lower
```

```
lower_words=[x.lower() for x in total_tokens]
print(lower_words)
```

```
['trending', 'new', 'yorkers', 'encounter', 'supermarket', 'shelves', 'pictured', 'wegmans', 'brooklyn', 'soldout', 'online', 'grocers',
```

```
finaltext=" ".join(lower_words) ## joining the all lower words into one paragraph for lemmatization
finaltext
```

```
'trending new yorkers encounter supermarket shelves pictured wegmans brooklyn soldout o
nline grocers foodkick maxdelivery coronavirusfearing shoppers stock find hand sanitize
r fred meyer turned amazon pack purellcheck coronavirus concerns driving prices find pr
otect loved ones coronavirus panic buying hits newyork city anxious shoppers stock food
ampmedical supplies healthcare worker bigapple st confirmed coronavirus patient bloombe
rg staged event qanon qanon qanon election cdc toiletpaper dunnypaper coronavirus coron
avirusaustralia coronavirusupdate covid news corvid newsmelb dunnypapergate costco week
buying baby milk powder buying toilet paper remember time paid gallon regular gas los a
```

```
##lemmatization
import nltk
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
True
```

```
from nltk.stem import WordNetLemmatizer
lemmatizer= WordNetLemmatizer()
```

```
lemmas=[]
for i in finaltext.split():
  lemmas.append(lemmatizer.lemmatize(i))
print(lemmas)
```

```
['trending', 'new', 'yorkers', 'encounter', 'supermarket', 'shelf', 'pictured', 'wegmans', 'brooklyn', 'soldout', 'online', 'grocer', 'f
```

```
len(lemmas)
```

```
65043
```

```
  ##count vectorizer is used to find and tell each word repeats how many times in the total data sets
```

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer= CountVectorizer()
words=vectorizer.fit_transform(lemmas)
words[0:40]
```

```
<40x9899 sparse matrix of type '<class 'numpy.int64'>'
        with 40 stored elements in Compressed Sparse Row format>
```

```
vectorizer.get_feature_names_out()[50:100]  ## it is printing 50 to 100 words
```

```
array(['achieved', 'acknowledging', 'acme', 'acmodels', 'acorn', 'act',
       'acting', 'action', 'actionhunger', 'activated', 'active',
       'activist', 'activity', 'acton', 'actor', 'actual', 'actually',
       'actwisenotstupid', 'acute', 'ad', 'ada', 'adapt', 'add', 'added',
       'addiction', 'adding', 'addition', 'additional', 'address',
       'addressing', 'adelaide', 'adequate', 'adequately', 'adesso',
       'adgs', 'adhere', 'adik', 'adimo', 'adjourns', 'adjust',
       'adjusted', 'adjusting', 'admin', 'administration', 'admit',
       'adobe', 'adopt', 'adoption', 'adult', 'advance'], dtype=object)
```

```
### it wiil tells us the how many times each word is repeated in the tweets
```

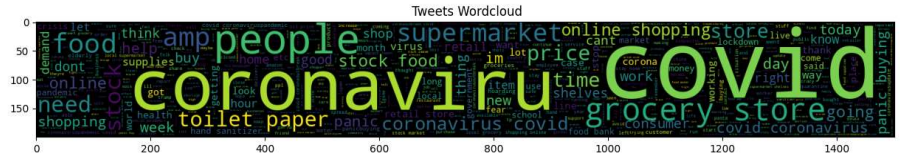
```
pd.DataFrame.from_records([vectorizer.vocabulary_]).T.sort_values(0,ascending=True).head(50)
```

	0
aadya	0
aadyasitara	1
aamiin	2
aapl	3
abajam	4
abandon	5
abandoning	6
abceyewitness	7
abeg	8
abid	9
abiding	10
abiity	11
ability	12
able	13
abomination	14
abroad	15
absolute	16
absolutely	17
absurd	18
absurdity	19
abt	20
abtas	21
abundance	22
abuse	23

```
##making word cloud
```

```
%matplotlib inline
from wordcloud import WordCloud
```

```
plt.figure(figsize=(15,50))
wordcloud=WordCloud(max_words=5000,width=1500,background_color="black").generate(" ".join(lower_words))
plt.imshow(wordcloud)
plt.title(" Tweets Wordcloud")
#plt.axix("off")
plt.show()
```




```
## converting the tfdf matrix for model evaluation
```

```
## intiailzing the x and y variable
```

```
x=test["cleaned_tweet"]
y=test["sentiment"]


###labe emcoding the target vaaraible(y)



from sklearn.preprocessing import LabelEncoder
LE=LabelEncoder()
y_Le=LE.fit_transform(y)
y_Le=pd.DataFrame(y_Le)
y_Le.head()
```

	0
0	2
1	0
2	2
3	2
4	1

```
from sklearn.feature_extraction.text import TfidfVectorizer
matrix=TfidfVectorizer(ngram_range=(1,2),max_features=2500)
x_tf=matrix.fit_transform(x.values.tolist()).toarray()


x_tf=pd.DataFrame(x_tf)
x_tf.head()
```

	0	1	2	3	4	5	6	7	8	9	...	2490	2491	2492	2493	2494	2495	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	
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	
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	
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	
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	

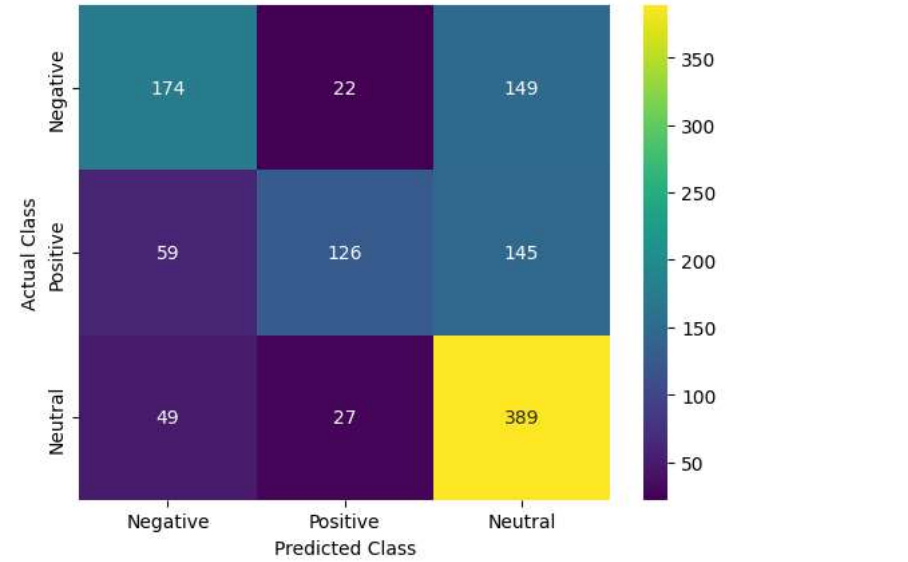
5 rows × 2500 columns

```
## data is splitting into train and split




from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_tf,y_Le,test_size=0.3)


##model building


from sklearn.naive_bayes import MultinomialNB, BernoulliNB
from sklearn.ensemble import RandomForestClassifier
import xgboost as xgb
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, roc_auc_score, accuracy_score


def model_train(model, x_train, x_test, y_train, y_test):
    model.fit(x_train,y_train)
    y_pred_train = model.predict(x_train)
    y_pred_test = model.predict(x_test)

    print("--------------------Training Performance----------------------")
    print(accuracy_score(y_train,y_pred_train))
    print(classification_report(y_train,y_pred_train))
    print("------------------------------------------------------------")
```

```
    print("--------------------Testing Performance----------------------")
    print(accuracy_score(y_test,y_pred_test))
    print(classification_report(y_test,y_pred_test))

    sns.heatmap(confusion_matrix(y_test, y_pred_test),cmap='viridis',annot=True,fmt='.4g',
            xticklabels=['Negative','Positive', 'Neutral'],yticklabels=['Negative','Positive', 'Neutral'])
    plt.xlabel('Predicted Class')
    plt.ylabel('Actual Class')
    plt.show()
```

```
##multinomianb classifer
modelNB = MultinomialNB()
model_train(modelNB, x_train, x_test, y_train, y_test)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversior
  y = column_or_1d(y, warn=True)
--------------------Training Performance----------------------
0.846124905944319
              precision    recall  f1-score   support

           0       0.90      0.81      0.85       831
           1       0.95      0.72      0.81       745
           2       0.78      0.96      0.86      1082

    accuracy                           0.85      2658
   macro avg       0.87      0.83      0.84      2658
weighted avg       0.86      0.85      0.84      2658

------------------------------------------------------------
--------------------Testing Performance----------------------
0.6043859649122807
              precision    recall  f1-score   support

           0       0.62      0.50      0.56       345
           1       0.72      0.38      0.50       330
           2       0.57      0.84      0.68       465

    accuracy                           0.60      1140
   macro avg       0.64      0.57      0.58      1140
weighted avg       0.63      0.60      0.59      1140
```




```
###logestic regression
```

```
modelLr=LogisticRegression()
model_train(modelLr,x_train,x_test,y_train,y_test)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversior
  y = column_or_1d(y, warn=True)
--------------------Training Performance---------------------
0.9221218961625283
              precision    recall  f1-score   support

           0       0.92      0.94      0.93       831
           1       0.95      0.85      0.90       745
           2       0.91      0.96      0.93      1082

    accuracy                           0.92      2658
   macro avg       0.93      0.92      0.92      2658
weighted avg       0.92      0.92      0.92      2658

------------------------------------------------------------
--------------------Testing Performance----------------------
0.6429824561403509
              precision    recall  f1-score   support

           0       0.61      0.65      0.63       345
           1       0.68      0.48      0.56       330
           2       0.65      0.75      0.70       465

    accuracy                           0.64      1140
   macro avg       0.65      0.63      0.63      1140
weighted avg       0.65      0.64      0.64      1140
```

svc CLassifier

```
models=SVC()
model_train(models,x_train,x_test,y_train,y_test)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversior
  y = column_or_1d(y, warn=True)
--------------------Training Performance---------------------
0.991346877351392
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       831
           1       0.99      0.99      0.99       745
           2       0.99      0.99      0.99      1082

    accuracy                           0.99      2658
   macro avg       0.99      0.99      0.99      2658
weighted avg       0.99      0.99      0.99      2658

------------------------------------------------------------
--------------------Testing Performance----------------------
0.6219298245614036
              precision    recall  f1-score   support

           0       0.60      0.60      0.60       345
           1       0.73      0.41      0.53       330
           2       0.60      0.78      0.68       465
```

```
#random forest



modelrf=RandomForestClassifier()
#model_train(modelrf,x_train,x_test,y_train,y_test)
```

```
model_train(modelrf,x_train,x_test,y_train,y_test)
```

```
      <ipython-input-287-c8b3cdf06f2d>:2: DataConversionWarning: A column-vector y was passed
        model.fit(x_train,y_train)
      -------------------Training Performance--------------------
```

```
##LDA


modellda=LDA()
model-train(modellda,x_train,x_test,y_train,y_test)
```

```
      ---------------------------------------------------------------------------
      NameError                                 Traceback (most recent call last)
      <ipython-input-302-884adbb1de94> in <cell line: 2>()
            1 modellda=LDA()
      ----> 2 model-train(modellda,x_train,x_test,y_train,y_test)

      NameError: name 'model' is not defined
```

SEARCH STACK OVERFLOW

```
           1       0.71      0.51      0.59       330


    accuracy                           0.64      1140
   macro avg       0.66      0.65      0.64      1140
weighted avg       0.67      0.64      0.64      1140
```