```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import nltk
from nltk.stem import WordNetLemmatizer
import re  #regular expression(to rmove special characers)
df=pd.read_csv("/content/twitter_validation.csv",encoding="ISO-8859-1",header=None)  #to read the emojis
df.columns=['id','location','target','text']
df
```

	id	location	target	text
0	3364	Facebook	Irrelevant	I mentioned on Facebook that I was struggling ...
1	352	Amazon	Neutral	BBC News - Amazon boss Jeff Bezos rejects clai...
2	8312	Microsoft	Negative	@Microsoft Why do I pay for WORD when it funct...
3	4371	CS-GO	Negative	CSGO matchmaking is so full of closet hacking,...
4	4433	Google	Neutral	Now the President is slapping Americans in the...
...	...	...	...	...
995	4891	GrandTheftAuto(GTA)	Irrelevant	âï¸ Toronto is the arts and culture capital...
996	4359	CS-GO	Irrelevant	tHIS IS ACTUALLY A GOOD MOVE TOT BRING MORE VI...
997	2652	Borderlands	Positive	Today sucked so itâs time to drink wine n pl...
998	8069	Microsoft	Positive	Bought a fraction of Microsoft today. Small wins.
999	6960	johnson&johnson	Neutral	Johnson & Johnson to stop selling talc baby po...

1000 rows × 4 columns

```python
df.head()
```

	id	location	target	text
0	3364	Facebook	Irrelevant	I mentioned on Facebook that I was struggling ...
1	352	Amazon	Neutral	BBC News - Amazon boss Jeff Bezos rejects clai...
2	8312	Microsoft	Negative	@Microsoft Why do I pay for WORD when it funct...
3	4371	CS-GO	Negative	CSGO matchmaking is so full of closet hacking,...
4	4433	Google	Neutral	Now the President is slapping Americans in the...

```python
df.tail()
```

	id	location	target	text
995	4891	GrandTheftAuto(GTA)	Irrelevant	âï¸ Toronto is the arts and culture capital...
996	4359	CS-GO	Irrelevant	tHIS IS ACTUALLY A GOOD MOVE TOT BRING MORE VI...
997	2652	Borderlands	Positive	Today sucked so itâs time to drink wine n pl...
998	8069	Microsoft	Positive	Bought a fraction of Microsoft today. Small wins.
999	6960	johnson&johnson	Neutral	Johnson & Johnson to stop selling talc baby po...

✓ 0s completed at 9:44 PM

```
df.columns
```

```
Index(['id', 'location', 'target', 'text'], dtype='object')
```

```
df.shape
```

```
(1000, 4)
```

```
df.isna().sum()
```

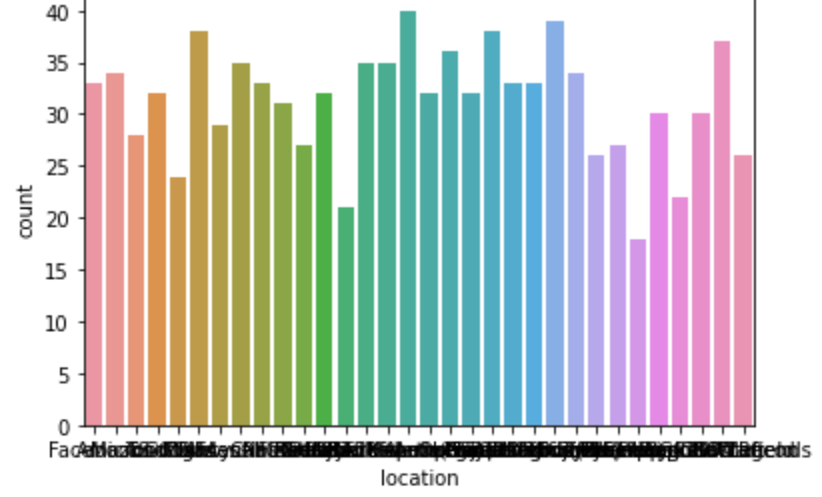
```
id          0
location    0
target      0
text        0
dtype: int64
```

```
df['location'].value_counts()
```

```
RedDeadRedemption(RDR)                 40
johnson&johnson                        39
FIFA                                   38
PlayerUnknownsBattlegrounds(PUBG)      38
LeagueOfLegends                        37
ApexLegends                            36
TomClancysRainbowSix                   35
Nvidia                                 35
GrandTheftAuto(GTA)                    35
Amazon                                 34
Fortnite                               34
Facebook                               33
PlayStation5(PS5)                      33
AssassinsCreed                         33
Borderlands                            33
Overwatch                              32
Hearthstone                            32
Verizon                                32
CS-GO                                  32
CallOfDuty                             31
Cyberpunk2077                          30
WorldOfCraft                           30
MaddenNFL                              29
Microsoft                              28
Dota2                                  27
CallOfDutyBlackopsColdWar              27
Xbox(Xseries)                          26
Battlefield                            26
Google                                 24
TomClancysGhostRecon                   22
NBA2K                                  21
HomeDepot                              18
Name: location, dtype: int64
```

```
import seaborn as sns
sns.countplot(x='location',data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f637a367e80>
```

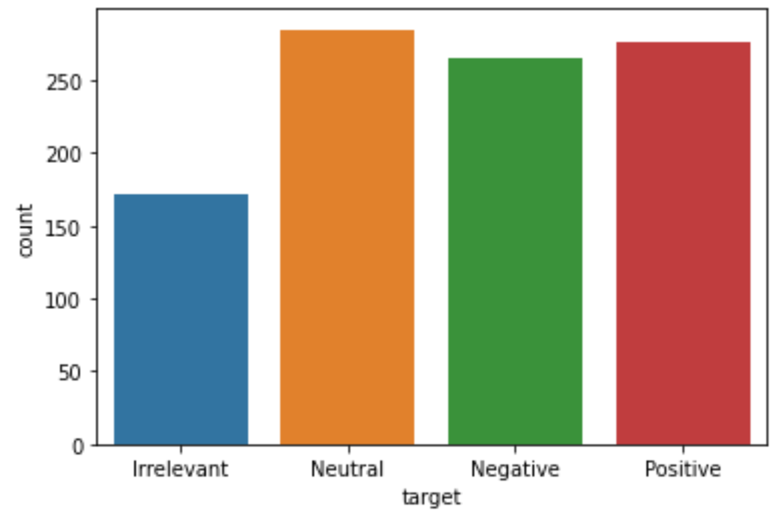


```
df['target'].value_counts()
```

```
Neutral       285
Positive      277
Negative      266
Irrelevant    172
Name: target, dtype: int64
```

```
sns.countplot(x='target',data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f637a34c1f0>
```



```
# drop irrelevent in target
df.drop(df.index[(df['target']=='Irrelevant')],axis=0,inplace=True)
df
# reset the index value
df.reset_index(drop=True,inplace=True)
df
# drop id and location
df.drop(['id','location'],axis=1,inplace=True)
df
```

	target	text
0	Neutral	BBC News - Amazon boss Jeff Bezos rejects clai...
1	Negative	@Microsoft Why do I pay for WORD when it funct...
2	Negative	CSGO matchmaking is so full of closet hacking,...
3	Neutral	Now the President is slapping Americans in the...

4	Negative	Hi @EAHelp Iâve had Madeleine McCann in my c...
...	...	...
823	Negative	Please explain how this is possible! How can t...
824	Positive	Good on Sony. As much as I want to see the new...
825	Positive	Today sucked so itâs time to drink wine n pl...
826	Positive	Bought a fraction of Microsoft today. Small wins.
827	Neutral	Johnson & Johnson to stop selling talc baby po...

828 rows × 2 columns

```
# Target=====>positive (+1)     negative (-1)      neutral (0)
# here we use map function
df['target']=df['target'].map({'Positive':1,'Negative':-1,'Neutral':0})
df
```

	target	text
0	0	BBC News - Amazon boss Jeff Bezos rejects clai...
1	-1	@Microsoft Why do I pay for WORD when it funct...
2	-1	CSGO matchmaking is so full of closet hacking,...
3	0	Now the President is slapping Americans in the...
4	-1	Hi @EAHelp Iâve had Madeleine McCann in my c...
...	...	...
823	-1	Please explain how this is possible! How can t...
824	1	Good on Sony. As much as I want to see the new...
825	1	Today sucked so itâs time to drink wine n pl...
826	1	Bought a fraction of Microsoft today. Small wins.
827	0	Johnson & Johnson to stop selling talc baby po...

828 rows × 2 columns

```
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
True
```

```
# assign text into variable
tweets=df.text
tweets
```

```
0      BBC News - Amazon boss Jeff Bezos rejects clai...
1      @Microsoft Why do I pay for WORD when it funct...
2      CSGO matchmaking is so full of closet hacking,...
3      Now the President is slapping Americans in the...
4      Hi @EAHelp Iâve had Madeleine McCann in my c...
                             ...
823    Please explain how this is possible! How can t...
824    Good on Sony. As much as I want to see the new...
825    Today sucked so itâs time to drink wine n pl...
826    Bought a fraction of Microsoft today. Small wins.
827    Johnson & Johnson to stop selling talc baby po...
Name: text, Length: 828, dtype: object
```

```
# tokenisation
from nltk import TweetTokenizer
tk=TweetTokenizer()
tweets=tweets.apply(lambda x:tk.tokenize(x)).apply(lambda x:" ".join(x))
tweets
```

```
0      BBC News - Amazon boss Jeff Bezos rejects clai...
1      @Microsoft Why do I pay for WORD when it funct...
2      CSGO matchmaking is so full of closet hacking ...
3      Now the President is slapping Americans in the...
4      Hi @EAHelp Iâ   ve had Madeleine McCann in m...
                             ...
823    Please explain how this is possible ! How can ...
824    Good on Sony . As much as I want to see the ne...
825    Today sucked so itâ   s time to drink wine n...
826    Bought a fraction of Microsoft today . Small w...
827    Johnson & Johnson to stop selling talc baby po...
Name: text, Length: 828, dtype: object
```

```
# remove the special characters
tweets=tweets.str.replace('[^a-zA-Z0-9]+',' ')  #  ^ denote except
tweets
```

```
<ipython-input-16-bc435e4ca2fe>:2: FutureWarning: The default value of regex will change from True to False in a future version.
  tweets=tweets.str.replace('[^a-zA-Z0-9]+',' ')  #  ^ denote except
0      BBC News Amazon boss Jeff Bezos rejects claims...
1       Microsoft Why do I pay for WORD when it funct...
2      CSGO matchmaking is so full of closet hacking ...
3      Now the President is slapping Americans in the...
4      Hi EAHelp I ve had Madeleine McCann in my cell...
                             ...
823    Please explain how this is possible How can th...
824    Good on Sony As much as I want to see the new ...
825    Today sucked so it s time to drink wine n play...
826     Bought a fraction of Microsoft today Small wins
827    Johnson Johnson to stop selling talc baby powd...
Name: text, Length: 828, dtype: object
```

```
# collect character length above 3
from nltk.tokenize import word_tokenize
tweets=tweets.apply(lambda x:' '.join([w for w in word_tokenize(x) if len(w)>=3]))
```

```
tweets
```

```
0      BBC News Amazon boss Jeff Bezos rejects claims...
1      Microsoft Why pay for WORD when functions poor...
2      CSGO matchmaking full closet hacking truly awf...
3      Now the President slapping Americans the face ...
4      EAHelp had Madeleine McCann cellar for the pas...
                             ...                        
823    Please explain how this possible How can they ...
824    Good Sony much want see the new PS5 what going...
825    Today sucked time drink wine play borderlands ...
826           Bought fraction Microsoft today Small wins
827    Johnson Johnson stop selling talc baby powder ...
Name: text, Length: 828, dtype: object
```

```
# stemming
# snowball_Stemmer
from nltk import SnowballStemmer
from nltk.tokenize import word_tokenize
stemmer=SnowballStemmer('english')
tweets=tweets.apply(lambda x:[stemmer.stem(i.lower()) for i in tk.tokenize(x)]).apply(lambda x:' '.join(x))
tweets
```

```
0      bbc news amazon boss jeff bezo reject claim co...
1      microsoft whi pay for word when function poor ...
2           csgo matchmak full closet hack truli aw game
3      now the presid slap american the face that rea...
4      eahelp had madelein mccann cellar for the past...
                             ...                        
823    pleas explain how this possibl how can they le...
824    good soni much want see the new ps5 what go ri...
825    today suck time drink wine play borderland unt...
826            bought fraction microsoft today small win
827    johnson johnson stop sell talc babi powder and...
Name: text, Length: 828, dtype: object
```

```
#remove the stopwords
from nltk.corpus import stopwords
stop=stopwords.words('english')
tweets=tweets.apply(lambda x:[i for i in tk.tokenize(x) if i not in stop]).apply(lambda x:' '.join(x))
tweets
```

```
0      bbc news amazon boss jeff bezo reject claim co...
1      microsoft whi pay word function poor samsungus...
2           csgo matchmak full closet hack truli aw game
3      presid slap american face realli commit unlaw ...
4      eahelp madelein mccann cellar past year littl ...
                             ...                        
823    pleas explain possibl let compani overcharg sc...
824    good soni much want see new ps5 go right much ...
825    today suck time drink wine play borderland sun...
826            bought fraction microsoft today small win
827    johnson johnson stop sell talc babi powder can...
Name: text, Length: 828, dtype: object
```

```
# vectorisation
from sklearn.feature_extraction.text import TfidfVectorizer
vec=TfidfVectorizer()
train_data=vec.fit_transform(tweets)
print(train_data)
```

```
  (0, 691)      0.2608257828483461
  (0, 1004)     0.2608257828483461
  (0, 1130)     0.23509805002803952
  (0, 1996)     0.13277165480466424
  (0, 309)      0.22681557001542715
  (0, 860)      0.17354914655342313
  (0, 807)      0.21432663830218204
  (0, 2761)     0.2608257828483461
  (0, 568)      0.2608257828483461
  (0, 1833)     0.24577602391989378
  (0, 633)      0.22681557001542715
  (0, 376)      0.1515362387424402
  (0, 2287)     0.38864111655856126
  (0, 538)      0.49155204783978756
  (1, 797)      0.4055823664694651
  (1, 2891)     0.4055823664694651
  (1, 2558)     0.3821800909185634
  (1, 1405)     0.4055823664694651
  (1, 3679)     0.36557591217188057
  (1, 2462)     0.3126902562590763
  (1, 3639)     0.26216072802580975
  (1, 2155)     0.24555654927912696
  (2, 1427)     0.1689251539717079
  (2, 486)      0.36574263611909275
  (2, 3432)     0.36574263611909275
  :     :
  (825, 3393)   0.3395996844494919
  (825, 3383)   0.2560582225152134
  (825, 631)    0.22981061112100945
  (825, 997)    0.2315686698425631
  (825, 3373)   0.21750175079084832
  (825, 3226)   0.2904718522758868
  (825, 2527)   0.17148706662740873
  (826, 1381)   0.5079831062080814
  (826, 3070)   0.47867226429410115
  (826, 636)    0.4174215841659411
  (826, 3650)   0.353278941165688
  (826, 3383)   0.34523850330234374
  (826, 2155)   0.3075542453642147
  (827, 195)    0.3283693467320579
  (827, 1132)   0.3283693467320579
  (827, 2808)   0.3283693467320579
  (827, 143)    0.3283693467320579
  (827, 712)    0.2770320970909926
  (827, 2946)   0.2635889502019104
  (827, 3274)   0.2635889502019104
  (827, 3193)   0.2311987519368367
  (827, 1686)   0.15963411936668057
  (827, 2576)   0.24870786898500463
  (827, 506)    0.23743856420618148
  (827, 1854)   0.3947412386878786
```

```
train_data.shape
# 828 sentence and 3783 uniquie value
```

```
(828, 3783)
```

```
y=df['target'].values
y
```

```
array([ 0, -1, -1,  0, -1,  1,  1,  1, -1,  1,  1, -1,  0, -1,  1,  1, -1,
        1, -1, -1,  0, -1,  0,  0, -1, -1,  1,  1, -1,  1, -1,  0,  0,  1,
            0,  1,  0,  0,  0,  1,  0, -1, -1, -1,  0,  1, -1, -1,  1,  1,  1,
            1,  1, -1, -1,  1,  1, -1,  0, -1,  0, -1,  1, -1, -1,  1,  1,  1,
            0,  0,  0,  1,  1,  0,  1,  0, -1, -1,  0,  0, -1,  1, -1, -1, -1,
            0,  1,  0, -1,  1,  1,  0,  1,  0,  1, -1,  0,  0,  0, -1,  0, -1,
            0,  0,  1,  1,  0, -1, -1,  1, -1,  0, -1,  1,  0, -1,  0,  1,  0,
            1,  1,  0,  0,  0,  0,  1,  0,  1,  1, -1,  0,  0,  0,  0, -1,  0,
            1, -1,  0, -1,  0, -1, -1, -1,  1,  1,  1,  0,  0,  1,  0,  0,  0,
            1,  0, -1, -1,  0,  1,  1,  0,  1,  1,  0,  0, -1, -1, -1, -1,  1,
            0,  0,  1,  1,  1,  1, -1,  1,  1,  0, -1, -1, -1,  1,  1, -1, -1,
            1,  1, -1,  1,  1, -1,  1,  0, -1,  0,  0,  1, -1,  1,  1,  0,  1,
           -1, -1,  1,  1,  1,  1,  0,  0,  1, -1,  0,  1,  0, -1,  0,  0, -1,
            1,  1, -1,  0,  1,  0, -1,  0, -1,  1,  1, -1, -1, -1,  1, -1,  0,
            1,  0,  0, -1,  1, -1,  1, -1,  0,  0,  1, -1,  0, -1,  1, -1,  1,
            1,  1,  1,  1,  1, -1, -1,  1, -1,  0,  0,  0,  1,  0,  1, -1,  0,
            0,  0,  0, -1,  1, -1, -1,  1,  1,  0,  0, -1, -1, -1,  0,  1,  0,
           -1,  1,  0, -1, -1, -1,  1,  0,  0, -1,  1,  1,  0,  1,  0,  0,  1,
            1, -1,  0,  1, -1,  0, -1, -1,  1,  1,  1,  1,  0, -1,  0,  1,  0,
            1, -1, -1, -1,  1,  0,  1, -1,  0, -1,  1,  1,  1,  1,  0,  0,  0,
           -1,  1,  1,  0, -1,  1,  0, -1, -1, -1, -1, -1,  0,  0,  0,  1,  1,
           -1, -1,  0, -1,  0,  0, -1,  1, -1,  1,  1,  1,  0,  1,  0,  0, -1,
            1,  0,  0,  0,  0,  0,  0,  0,  0, -1, -1,  1,  1,  0, -1, -1,  1,
            1, -1,  1,  1,  1,  1,  1,  0, -1,  1,  0,  0,  1,  1,  1,  1,  0,
           -1, -1, -1, -1,  0,  1, -1, -1,  1,  1,  0,  0, -1, -1,  1,  0, -1,
           -1, -1,  0,  0,  1, -1, -1, -1,  0,  0,  0, -1, -1,  1, -1,  0, -1,
            0,  1, -1,  0,  1,  1, -1,  0,  0,  1, -1, -1,  0,  0, -1,  1, -1,
            0, -1, -1, -1,  1, -1,  1, -1,  1, -1, -1,  0, -1,  0, -1,  1, -1,
            0, -1, -1,  0,  0,  1, -1,  1,  0,  0,  0,  0, -1,  0,  0,  0, -1,
           -1,  0,  1,  0,  0, -1,  0,  1,  0,  0,  0,  0,  0,  1,  0,  1,  1,
            1,  0, -1,  1,  0,  0, -1,  1,  0,  0, -1,  0, -1,  0,  1, -1,  1,
           -1, -1,  0,  0,  0,  0,  1,  1,  1, -1, -1,  0,  1,  0,  0, -1,  1,
            1,  0,  1, -1, -1,  0,  1, -1,  1, -1,  0,  1,  1,  0,  0,  0,  1,
            0, -1,  0,  0, -1,  1, -1,  0,  1,  1,  1,  1,  0, -1,  0,  1,  1,
            1,  1,  1, -1,  0,  1,  0,  0, -1, -1, -1,  0,  1,  0, -1,  1,  1,
            1,  0,  1, -1,  0, -1,  0, -1,  0,  0,  1, -1,  1,  1,  0, -1,  0,
           -1, -1, -1, -1,  1,  1,  1,  1,  0, -1, -1,  1, -1, -1,  0,  0,  1,
            0, -1,  0,  1, -1,  0,  1, -1,  0,  0,  1, -1,  0, -1,  1,  1,  0,
            1,  0,  1, -1,  0,  0,  0,  1,  0,  0, -1,  1,  0, -1, -1,  0,  0,
            1, -1, -1, -1, -1,  1,  0,  0,  1,  0, -1,  1,  1, -1,  1,  1,  0,
           -1,  0,  1,  1, -1, -1, -1,  1, -1,  0, -1,  0,  0,  1,  1, -1,  0,
            1, -1, -1, -1, -1, -1, -1, -1, -1,  0, -1,  0,  0,  0,  1,  0,  0,
            0, -1,  0,  1,  0, -1, -1,  1,  0,  1,  0,  1,  0, -1,  1,  1,  1,
            1, -1, -1,  1,  0,  0,  0,  0,  0,  0, -1, -1, -1, -1,  1, -1,  0,
            1,  0, -1,  1,  1, -1,  1,  0,  0,  1, -1,  0, -1,  0,  1,  1,  0,
           -1,  1, -1, -1,  0, -1,  0, -1,  1,  0, -1, -1,  1,  1, -1,  0, -1,
            0,  0,  0,  0,  0,  0,  1,  0,  1,  1,  1, -1,  0,  1,  0,  1,  0,
            1,  0,  1,  0, -1, -1,  1,  1,  1,  1,  0, -1,  1,  1, -1, -1, -1,
            0,  1,  0,  1,  1,  0,  1, -1,  1,  1,  1,  0])
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(train_data,y,test_size=0.30,random_state=42)
```

```
x_test
```

```
    <249x3783 sparse matrix of type '<class 'numpy.float64'>'
            with 3255 stored elements in Compressed Sparse Row format>
```

```
y_train
```

```
    array([ 1,  1, -1, -1,  0, -1,  0,  1,  1,  0, -1,  0, -1, -1,  1,  0, -1,
            1, -1, -1,  1,  0,  1, -1, -1,  0,  0,  1, -1,  1, -1,  0,  0, -1,
           -1, -1, -1,  0,  0,  1, -1,  0,  0, -1,  1,  1,  1, -1,  0,  1, -1,
           -1,  1,  0,  1, -1, -1,  1,  1, -1,  1,  0,  1,  1,  0,  1,  0,  0,
       -1,  1,  0,  1, -1, -1, -1, -1, -1, -1, -1,  0, -1,  1, -1,  0,  1,
        0,  1,  1,  0,  1, -1,  1,  0, -1,  1, -1, -1,  0,  0, -1,  0,  1,
       -1, -1,  1, -1,  0,  1,  1,  0,  1,  0, -1,  1,  1,  0,  0,  0,  0,
        1, -1,  1,  1,  1,  1,  0,  1,  0, -1,  0,  0,  1,  0, -1, -1, -1,
       -1,  1,  1,  1, -1,  1,  0,  1,  1,  1,  1,  0,  0, -1, -1,  0,  0,
        0, -1,  0,  0,  0,  1,  1,  0, -1, -1,  0,  0,  0, -1, -1, -1, -1,
       -1, -1,  0,  0, -1, -1,  0,  1, -1, -1,  1, -1,  0,  0, -1, -1, -1,
        0,  0, -1,  0,  0,  1,  0, -1, -1, -1,  0,  1,  1,  1,  1,  1,  1,
        0,  1, -1,  1, -1, -1, -1,  0, -1,  1,  1, -1,  1, -1,  0,  0, -1,
        1,  0, -1,  1,  1,  0,  1, -1, -1, -1,  1,  0,  0, -1,  0,  0,  0,
        0,  1,  1, -1,  1,  1,  0,  1,  0, -1, -1,  1,  1,  1,  1,  1,  1,
        0,  1,  0,  0,  1, -1,  0,  1, -1,  1, -1,  0,  0,  1,  0,  1,  0,
        1, -1,  1,  1,  0,  1,  0, -1,  0,  1,  0,  0,  1,  0, -1,  0,  1,
        1,  0, -1,  1, -1,  0,  1,  1, -1,  1, -1,  0,  0, -1,  0,  0,  1,
        0,  0,  1,  1,  0,  0,  0, -1,  0,  0, -1,  0, -1,  0,  1, -1,  0,
        1,  0,  1,  1,  0, -1, -1,  0, -1, -1,  0, -1,  1, -1, -1,  1,  0,
       -1,  0,  0,  0,  1, -1,  0,  1,  0,  1,  0, -1,  1, -1, -1,  0,  0,
       -1,  1,  0,  1, -1,  1,  0,  1,  0,  1,  0, -1,  1, -1,  0,  1,  1,
        1,  0,  0, -1,  0, -1,  1,  0,  1, -1,  1,  1,  1, -1,  0, -1, -1,
        1,  1, -1, -1,  0,  1, -1, -1, -1,  1,  0,  1,  0,  0, -1,  0,  0,
        0,  0, -1,  0,  1,  0, -1,  1,  1,  0,  0,  0,  1,  1,  0,  1,  1,
        1,  1,  0,  0,  0,  0,  1, -1, -1,  0,  0, -1, -1, -1, -1,  1, -1,
        1, -1, -1, -1, -1, -1, -1,  1,  1,  0,  0,  1,  0, -1,  0,  1, -1,
        1,  0,  1,  0,  0,  0,  0,  1,  1, -1, -1,  0,  1, -1,  0,  1,  1,
       -1, -1,  1, -1,  0, -1,  0,  1,  0,  1,  0, -1,  0,  1, -1, -1,  0,
       -1,  0,  1,  1,  1,  1,  1,  1,  0, -1,  0,  1, -1, -1, -1,  0,  0,
       -1, -1,  0,  1,  0,  0,  1,  0,  0,  0,  1, -1, -1,  1, -1,  0,  0,
       -1,  0,  1,  0, -1,  0, -1,  1, -1, -1, -1,  0, -1, -1,  0,  1,  0,
        1,  1,  0,  1, -1, -1,  0,  0,  1,  1,  1,  1,  0,  0,  0,  0,  0,
        0,  1, -1,  0,  0, -1, -1,  0, -1,  0, -1,  0, -1,  1,  0, -1,  0,
        0])
```

```
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
svm_model=SVC()
nb_model=MultinomialNB()
rf_model=RandomForestClassifier()
de_model=DecisionTreeClassifier()
lstmodel=[svm_model,nb_model,rf_model,de_model]
```

```
from sklearn.metrics import confusion_matrix,classification_report
for i in lstmodel:
  print(i)
  i.fit(x_train,y_train)
  y_pred=i.predict(x_test)
  print("******")
  print(classification_report(y_test,y_pred))
  print("******")
```

```
SVC()
******
              precision    recall  f1-score   support

          -1       0.67      0.52      0.59        79
           0       0.46      0.76      0.57        79
           1       0.76      0.48      0.59        91

    accuracy                           0.58       249
   macro avg       0.63      0.59      0.58       249
```

```
weighted avg       0.64      0.58      0.58       249

******
MultinomialNB()
******
              precision    recall  f1-score   support

          -1       0.60      0.54      0.57        79
           0       0.47      0.54      0.50        79
           1       0.64      0.59      0.61        91

    accuracy                           0.56       249
   macro avg       0.57      0.56      0.56       249
weighted avg       0.57      0.56      0.56       249

******
RandomForestClassifier()
******
              precision    recall  f1-score   support

          -1       0.51      0.73      0.60        79
           0       0.53      0.53      0.53        79
           1       0.71      0.43      0.53        91

    accuracy                           0.56       249
   macro avg       0.58      0.56      0.55       249
weighted avg       0.59      0.56      0.55       249

******
DecisionTreeClassifier()
******
              precision    recall  f1-score   support

          -1       0.47      0.56      0.51        79
           0       0.47      0.48      0.48        79
           1       0.58      0.48      0.53        91

    accuracy                           0.51       249
   macro avg       0.51      0.51      0.51       249
weighted avg       0.51      0.51      0.51       249

******
```