SRI RAMACHANDRA
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Category - I Deemed to be University) Porur, Chennai
SRI RAMACHANDRA FACULTY OF ENGINEERING AND TECHNOLOGY

CSE Code – CSE 380

Course Title – Natural Language Processing

2020-2021

Project Report

TWITTER SENTIMENT ANALYSIS

By,

Santosh Prasad D

E0119050

Abstract

Sentiment analysis deals with identifying and classifying opinions or sentiments expressed in source text. Social media is generating a vast amount of sentiment rich data in the form of tweets, status updates, blog posts etc. Sentiment analysis of this user generated data is very useful in knowing the opinion of the crowd. Twitter sentiment analysis is difficult compared to general sentiment analysis due to the presence of slang words and misspellings. The maximum limit of characters that are allowed in Twitter is 140. Knowledge base approach and Machine learning approach are the two strategies used for analyzing sentiments from the text. In this paper, we try to analyze the twitter posts about electronic products like mobiles, laptops etc. using Machine Learning approach. By doing sentiment analysis in a specific domain, it is possible to identify the effect of domain information in sentiment classification. We present a new feature vector for classifying the tweets as positive, negative and extract peoples' opinion about products.

Introduction used

We have chosen to work with twitter since we feel it is a better approximation of public sentiment as opposed to conventional internet articles and web blogs. The reason is that the amount of relevant data is much larger for twitter, as compared to traditional blogging sites. Moreover the response on twitter is more prompt and also more general (since the number of users who tweet is substantially more than those who write web blogs on a daily basis). Sentiment analysis of public is highly critical in macro-scale socioeconomic phenomena like predicting the stock market rate of a particular firm. This could be done by analyzing overall public sentiment towards that firm with respect to time and using economics tools for finding the correlation between public sentiment and the firm's stock market value. Firms can also estimate how well their product is responding in the market, which areas of the market is it having a favorable response and in which a negative response (since twitter allows us to download stream of geo-tagged tweets for particular locations. If firms can get this information they can analyze the reasons behind geographically differentiated response, and so they can market

their product in a more optimized manner by looking for appropriate solutions like creating suitable market segments. Predicting the results of popular political elections and polls is also an emerging application to sentiment analysis. One such study was conducted by Tumasjan et al. in Germany for predicting the outcome of federal elections in which concluded that twitter is a good reflection of offline sentiment

Objectives of the Project

1. Tweets Preprocessing and Cleaning
2. Train test split of data
3. Extracting Features from reviews
4. Model Building: Sentiment Analysis

Question 1 & 2: Write a python code to read the list of files from a directory and store it in a data frame. Extract each record from the data frame and perform the following operation, Find the vector representation of the whole record.

Q1, Q2

```
[ ]  import numpy as np
     import pandas as pd
     import re
     import string
     from wordcloud import WordCloud
     from nltk.tokenize import sent_tokenize, word_tokenize
     from nltk.corpus import stopwords
     from nltk.stem import WordNetLemmatizer, PorterStemmer
     from nltk import pos_tag, ne_chunk
     from nltk.chunk import tree2conlltags

     import seaborn as sns
     import matplotlib.pyplot as plt
     from collections import Counter

     import nltk
     nltk.download('stopwords')
     nltk.download('punkt')
     nltk.download('wordnet')
     nltk.download('averaged_perceptron_tagger')
     nltk.download('maxent_ne_chunker')
     nltk.download('words')

     import warnings
     warnings.filterwarnings("ignore")
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
```

```
columns = ["Id","Entity","Target","Text"]
data = pd.read_csv("/content/twitter_training.csv",
                   names=columns,header=None)
```

```
data1=data.head(5000)
data2=data.tail(5000)
```

```
df = data1[["Text","Target"]]
```

```
df.head()
```

	Text	Target
0	im getting on borderlands and i will murder yo...	Positive
1	I am coming to the borders and I will kill you...	Positive
2	im getting on borderlands and i will kill you ...	Positive
3	im coming on borderlands and i will murder you...	Positive
4	im getting on borderlands 2 and i will murder ...	Positive

```
df.shape
```

```
(5000, 2)
```

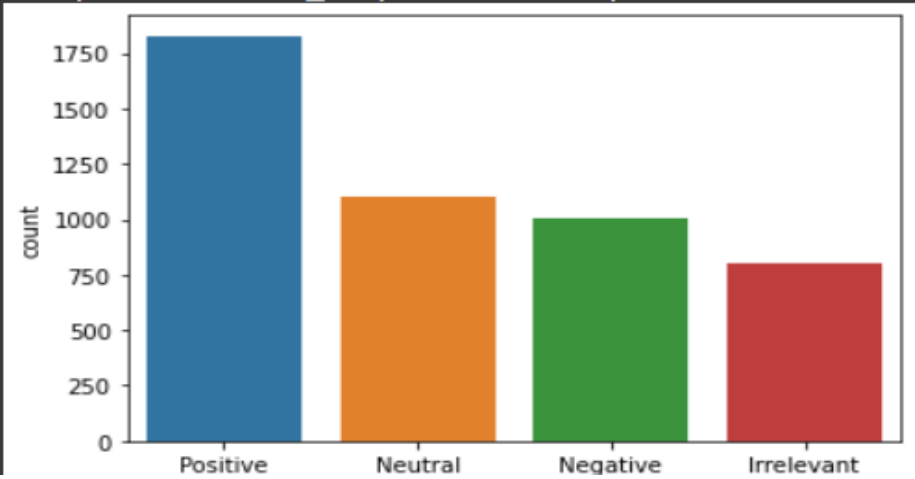
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Text    4952 non-null   object
 1   Target  5000 non-null   object
dtypes: object(2)
memory usage: 78.2+ KB
```

```
df= df.drop_duplicates()
```

```
sns.countplot(x="Target",data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8ad261dc10>
```



```
sentiment = []

for i in df["Target"]:
    if i == "Positive":
        sentiment.append(1)
    elif (i == "Irrelevant") or (i == "Neutral"):
        sentiment.append(0)
    else:
        sentiment.append(-1)
df["Sentiment"] = sentiment
```

```
df.head()
```

	Text	Target	Sentiment
0	im getting on borderlands and i will murder yo...	Positive	1
1	I am coming to the borders and I will kill you...	Positive	1
2	im getting on borderlands and i will kill you ...	Positive	1
3	im coming on borderlands and i will murder you...	Positive	1
4	im getting on borderlands 2 and i will murder ...	Positive	1

```
stop_words = set(stopwords.words("english"))
```

Text Cleaner

```
[ ]  df["Text"] = df["Text"].str.replace("\d","")
```

```
[ ]  def cleaner(data):
         # Tokens
         tokens = word_tokenize(str(data).replace("'", "").lower())

         # Remove Puncs
         without_punc = [w for w in tokens if w.isalpha()]

         # Stopwords
         without_sw = [t for t in without_punc if t not in stop_words]

         # Lemmatize
         text_len = [WordNetLemmatizer().lemmatize(t) for t in without_sw]
         # Stem
         text_cleaned = [PorterStemmer().stem(w) for w in text_len]

         return " ".join(text_cleaned)
```

```
[ ]  df["Text"] = df["Text"].apply(cleaner)
     df["Text"].head()
```

```
0     im get borderland murder
1             come border kill
2       im get borderland kill
3    im come borderland murder
4     im get borderland murder
Name: Text, dtype: object
```

```
[ ]  df["Text"]=df["Text"].str.replace("im","")
```

```
[ ] df["Text"]=df["Text"].str.replace("im","")
    df["Text"].head()
```

```
0      get borderland murder
1           come border kill
2        get borderland kill
3     come borderland murder
4      get borderland murder
Name: Text, dtype: object
```

Rare Words

```
rare_words = pd.Series(" ".join(df["Text"]).split()).value_counts()
rare_words
```

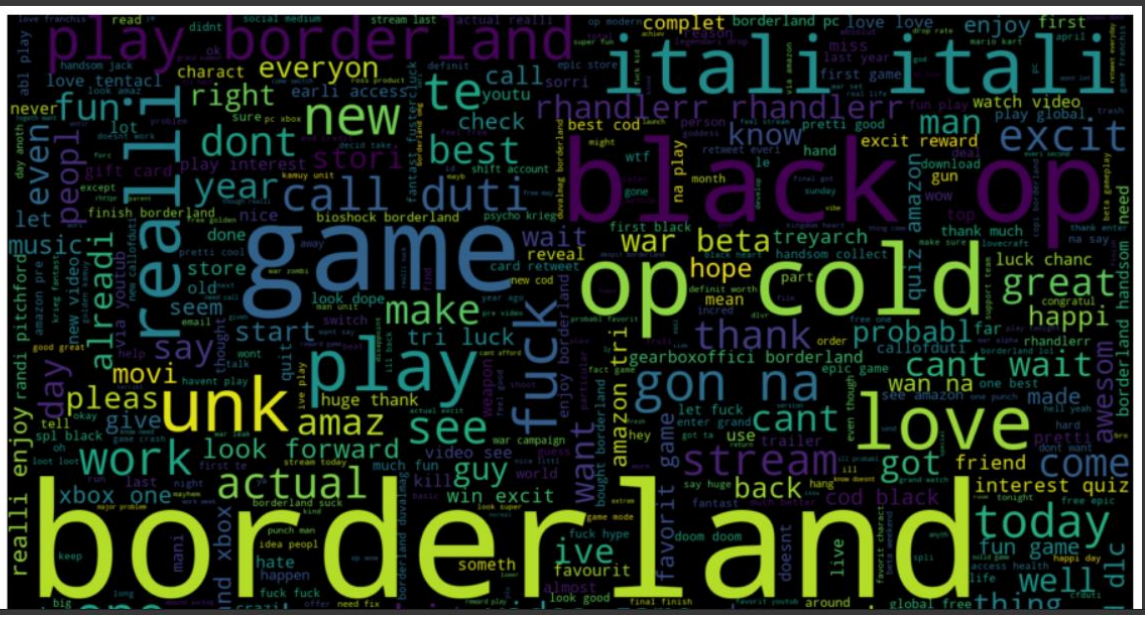
```
borderland    1509
game           810
play           575
black          511
war            497
              ...
strongli         1
five             1
del              1
financi          1
geynuqumv        1
Length: 4022, dtype: int64
```

```
[ ] rare_words = rare_words[rare_words <= 2]
```

```
[ ] df["Text"] = df["Text"].apply(lambda x: " ".join([i for i in x.split() if i not in rare_words.index]))
```

Word Cloud

```
plt.figure(figsize=(16,12))
wordcloud = WordCloud(background_color="black",max_words=500, width=1500, height=1000).generate(' '.join(df['Text']))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



Train test split

```
[ ] from sklearn.model_selection import train_test_split
    from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
    from sklearn.naive_bayes import MultinomialNB, BernoulliNB
    from sklearn.ensemble import RandomForestClassifier
```

```
[ ] X = df["Text"]
    y = df["Sentiment"]
```

```
[ ] X_train,X_test,y_train,y_test = train_test_split(X,y, test_size = 0.30,random_state= 42)
```

Count Vectorizer

```
[ ] from sklearn.feature_extraction.text import CountVectorizer
```

```
[ ] vt = CountVectorizer(analyzer="word")
    X_train_count = vt.fit_transform(X_train)
    X_test_count = vt.transform(X_test)
```

```
    print(X_train_count.toarray())
    X_train_count
```

```
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
```

MultinomialNB

```
[ ] nb_model = MultinomialNB()
    nb_model.fit(X_train_count,y_train)
```

```
MultinomialNB()
```

```
[ ] nb_pred = nb_model.predict(X_test_count)
    nb_train_pred = nb_model.predict(X_train_count)
```

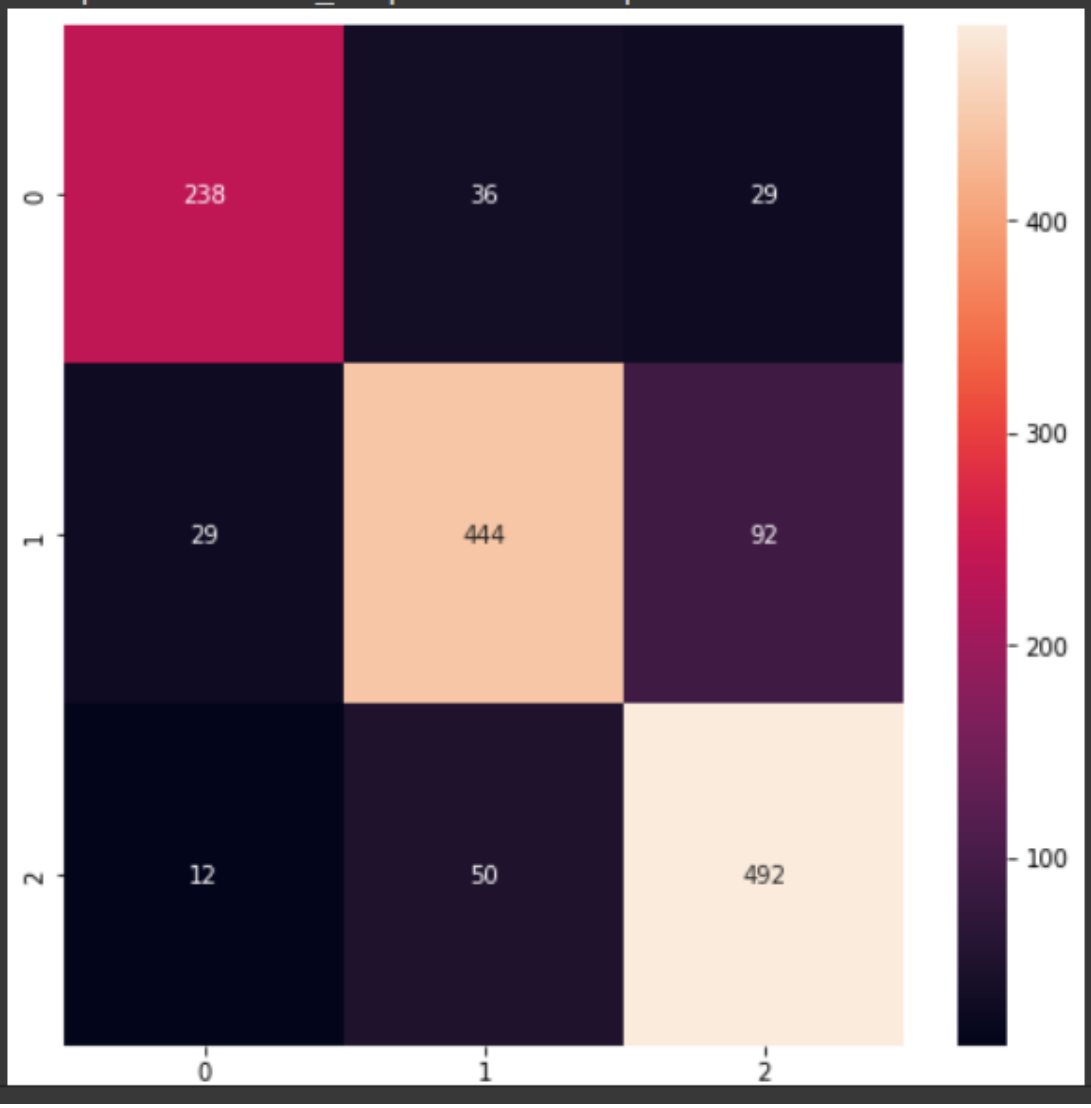
```
print("X Test")
print(classification_report(y_test,nb_pred))
print("X Train")
print(classification_report(y_train,nb_train_pred))

plt.figure(figsize=(8,8))
sns.heatmap(confusion_matrix(y_test,nb_pred),annot = True,fmt = "d")
```

```
X Test
              precision    recall  f1-score   support

          -1       0.85      0.79      0.82       303
           0       0.84      0.79      0.81       565
           1       0.80      0.89      0.84       554

    accuracy                           0.83      1422
   macro avg       0.83      0.82      0.82      1422
weighted avg       0.83      0.83      0.82      1422
```

X Train

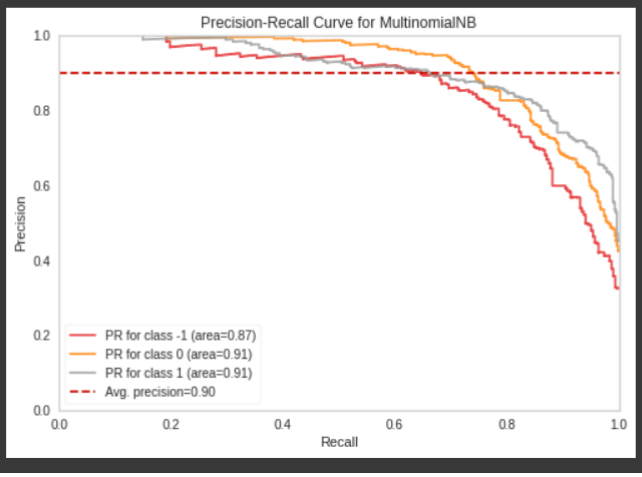
	precision	recall	f1-score	support
-1	0.90	0.79	0.84	701
0	0.87	0.87	0.87	1341
1	0.86	0.91	0.88	1274
accuracy			0.87	3316
macro avg	0.88	0.86	0.87	3316
weighted avg	0.87	0.87	0.87	3316

<matplotlib.axes._subplots.AxesSubplot at 0x7f8acbe48c50>



```
[ ] from yellowbrick.classifier import PrecisionRecallCurve
```

```
viz = PrecisionRecallCurve(MultinomialNB(),
                           classes=nb_model.classes_,   # label ların isimleri ile gözükmesi için yoksa 0,1,2 gözükür
                           per_class=True,  # bütün class lar grafikte gözüksün diye True yaptık.
                           cmap="Set1")
viz.fit(X_train_count,y_train)
viz.score(X_test_count, y_test)  # test datası üzerinden skorları alıyoruz
viz.show();
```



Question 3: Apply any machine learning or deep learning model to perform the required NLP task

Q3

Random Forest

```
[ ] from sklearn.model_selection import GridSearchCV
```

```
[ ] rf_params = {"max_depth": [2,4,6,10,12],
                 "max_features": [2,5,7],
                 "n_estimators":[12,15,16,]}
    rf = RandomForestClassifier()
```

```
[ ] rf_model_cv = GridSearchCV(rf,rf_params,cv = 5,n_jobs = -1)
```

```
[ ] rf_model_cv.fit(X_train_count,y_train)
```

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(), n_jobs=-1,
             param_grid={'max_depth': [2, 4, 6, 10, 12],
                         'max_features': [2, 5, 7],
                         'n_estimators': [12, 15, 16]})
```

```
rf_model_cv.best_params_
```

```
{'max_depth': 12, 'max_features': 7, 'n_estimators': 15}
```

+ Code + Text

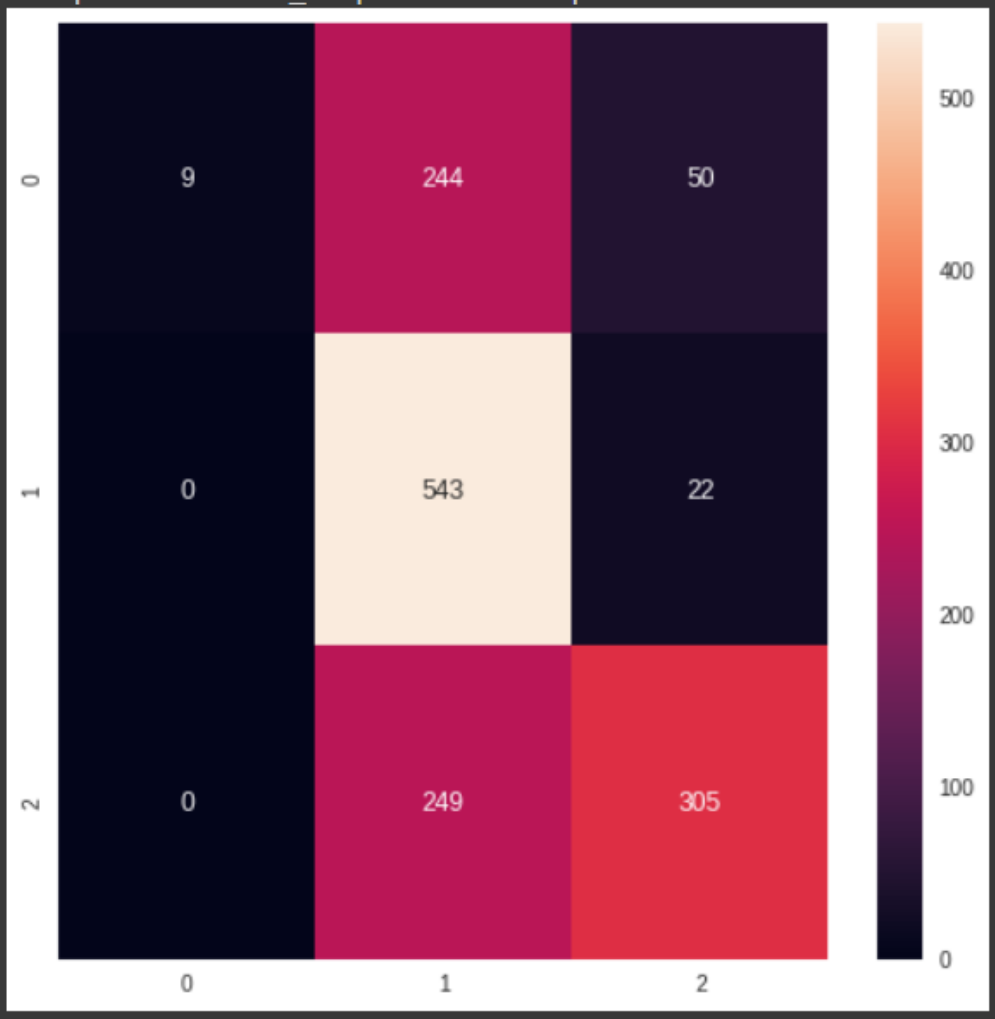
```
[ ] rf_tuned = RandomForestClassifier(max_depth = 12,
                                      max_features = 7,
                                      min_samples_split = 2).fit(X_train_count,y_train)
```

```
[ ] rf_pred = rf_tuned.predict(X_test_count)
    rf_train_pred = rf_tuned.predict(X_train_count)
```

```
    print("X Test")
    print(classification_report(y_test,rf_pred))
    print("X Train")
    print(classification_report(y_train,rf_train_pred))

    plt.figure(figsize=(8,8))
    sns.heatmap(confusion_matrix(y_test,rf_pred),annot = True,fmt = "d")
```

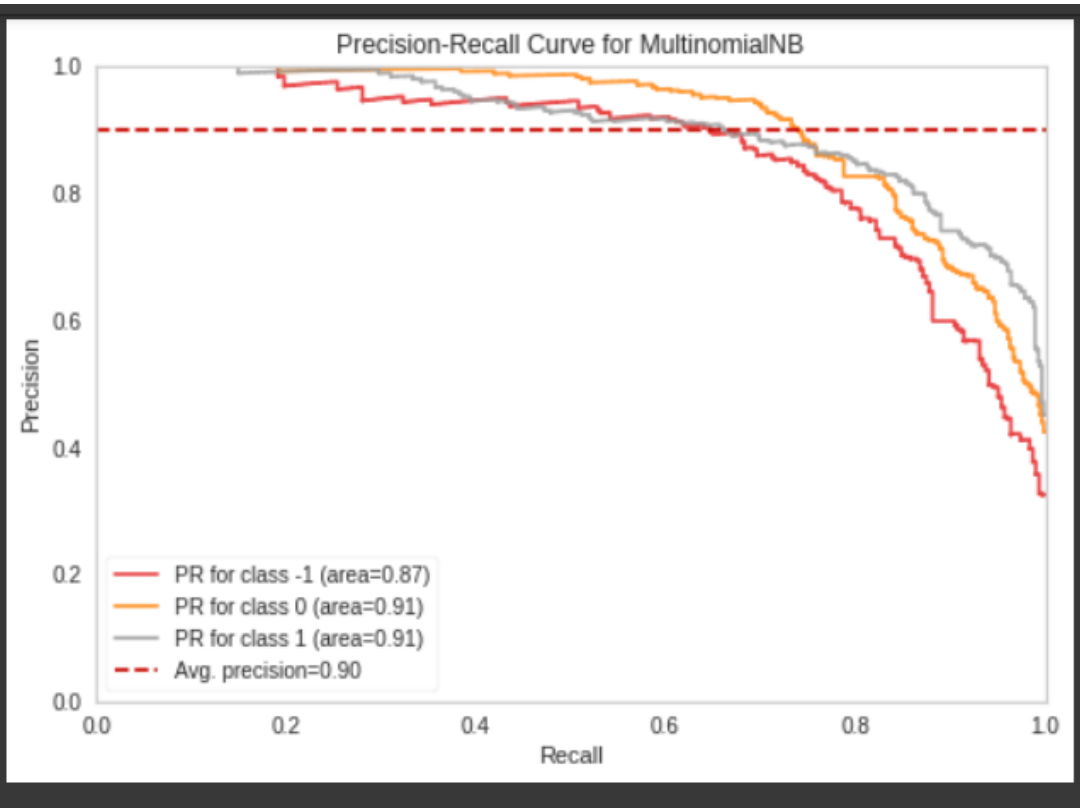
```
X Test
              precision    recall  f1-score   support

          -1       1.00      0.03      0.06       303
           0       0.52      0.96      0.68       565
           1       0.81      0.55      0.66       554

    accuracy                           0.60      1422
   macro avg       0.78      0.51      0.46      1422
weighted avg       0.74      0.60      0.54      1422

X Train
              precision    recall  f1-score   support

          -1       1.00      0.05      0.10       701
           0       0.56      0.99      0.71      1341
           1       0.91      0.64      0.75      1274

    accuracy                           0.66      3316
   macro avg       0.82      0.56      0.52      3316
weighted avg       0.79      0.66      0.60      3316
```

```
[ ] <matplotlib.axes._subplots.AxesSubplot at 0x7f8ad08ebe90>
```




```
viz = PrecisionRecallCurve(MultinomialNB(),
                           classes=rf_tuned.classes_,
                           per_class=True,
                           cmap="Set1")
viz.fit(X_train_count,y_train)
viz.score(X_test_count, y_test)
viz.show();
```

[ ]


Precision-Recall Curve for MultinomialNB
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
PR for class -1 (area=0.87)
PR for class 0 (area=0.91)
PR for class 1 (area=0.91)
Avg. precision=0.90

Question 4: Extend the classification/ prediction/ NLP task with a new dataset. The new dataset should be related to the first dataset that you have considered.

Q4

Second Half

```
[ ] df = data2[["Text","Target"]]
```

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

	Text	Target
5000	Amazon UK launches Sherlock Holmes Advent Cale...	Neutral
5001	Amazon UK launches the Sherlock Family Advent ...	Neutral
5002	Amazon UK launches their the following Sherloc...	Neutral
5003	Amazon UK launches the Sherlock Holmes 5 Calen...	Neutral
5004	Amazon Web Services re/Start program helps peo...	Neutral

```
[ ] df.shape
```

```
(5000, 2)
```

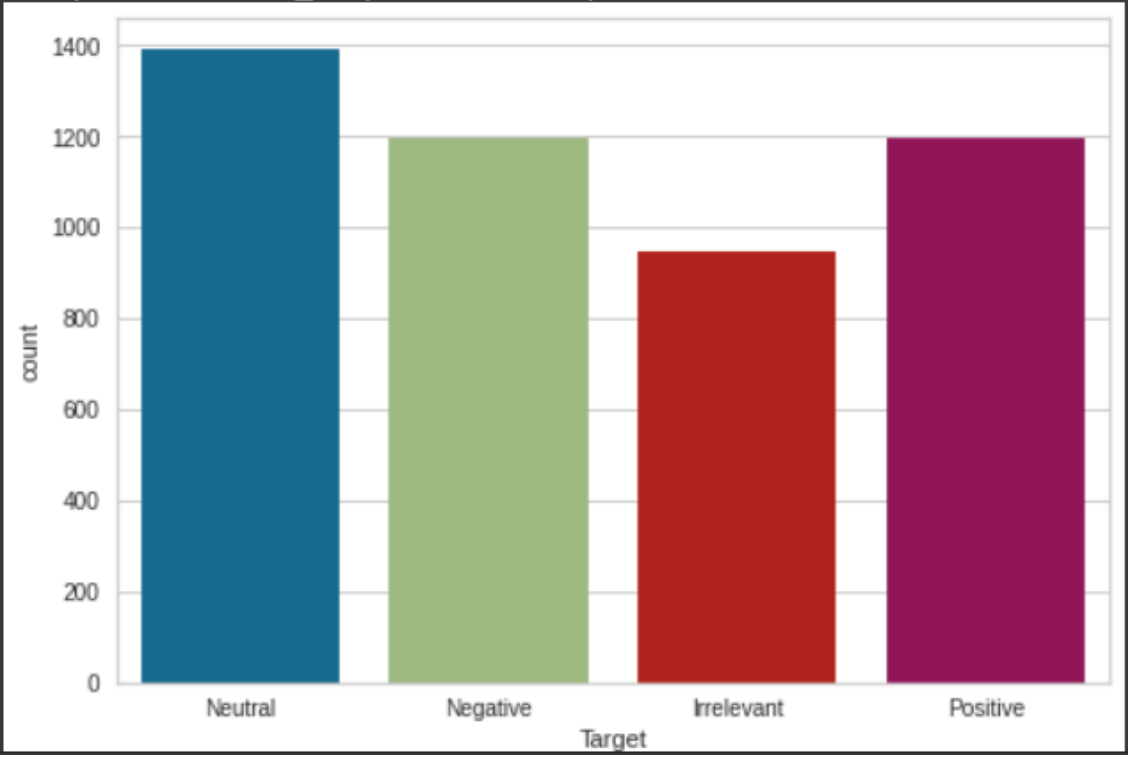
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 5000 to 9999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Text    4936 non-null   object
 1   Target  5000 non-null   object
dtypes: object(2)
memory usage: 78.3+ KB
```

```
df= df.drop_duplicates()
```

```
sns.countplot(x="Target",data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8acbf81490>
```



```
sentiment = []

for i in df["Target"]:
    if i == "Positive":
        sentiment.append(1)
    elif (i == "Irrelevant") or (i == "Neutral"):
        sentiment.append(0)
    else:
        sentiment.append(-1)
df["Sentiment"] = sentiment
```

```
df.head()
```

	Text	Target	Sentiment
5000	Amazon UK launches Sherlock Holmes Advent Cale...	Neutral	0
5001	Amazon UK launches the Sherlock Family Advent ...	Neutral	0
5002	Amazon UK launches their the following Sherloc...	Neutral	0
5003	Amazon UK launches the Sherlock Holmes 5 Calen...	Neutral	0
5004	Amazon Web Services re/Start program helps peo...	Neutral	0

```
stop_words = set(stopwords.words("english"))
```

Text Cleaner

```
[ ]  df["Text"] = df["Text"].str.replace("\d","")
```

```
[ ]  def cleaner(data):
         # Tokens
         tokens = word_tokenize(str(data).replace("'", "").lower())

         # Remove Puncs
         without_punc = [w for w in tokens if w.isalpha()]

         # Stopwords
         without_sw = [t for t in without_punc if t not in stop_words]

         # Lemmatize
         text_len = [WordNetLemmatizer().lemmatize(t) for t in without_sw]
         # Stem
         text_cleaned = [PorterStemmer().stem(w) for w in text_len]

         return " ".join(text_cleaned)
```

```
[ ]  df["Text"] = df["Text"].apply(cleaner)
     df["Text"].head()
```

```
5000    amazon uk launch sherlock holm advent calendar...
5001    amazon uk launch sherlock famili advent calend...
5002    amazon uk launch follow sherlock sherlock holm...
5003    amazon uk launch sherlock holm calendar last y...
5004    amazon web servic program help peopl train tec...
Name: Text, dtype: object
```

```
[ ]  df["Text"]=df["Text"].str.replace("im","")
     df["Text"].head()
```

```
5000    amazon uk launch sherlock holm advent calendar...
5001    amazon uk launch sherlock famili advent calend...
5002    amazon uk launch follow sherlock sherlock holm...
5003    amazon uk launch sherlock holm calendar last y...
5004    amazon web servic program help peopl train tec...
Name: Text, dtype: object
```

Rare Words

```
rare_words = pd.Series(" ".join(df["Text"]).split()).value_counts()
rare_words
```

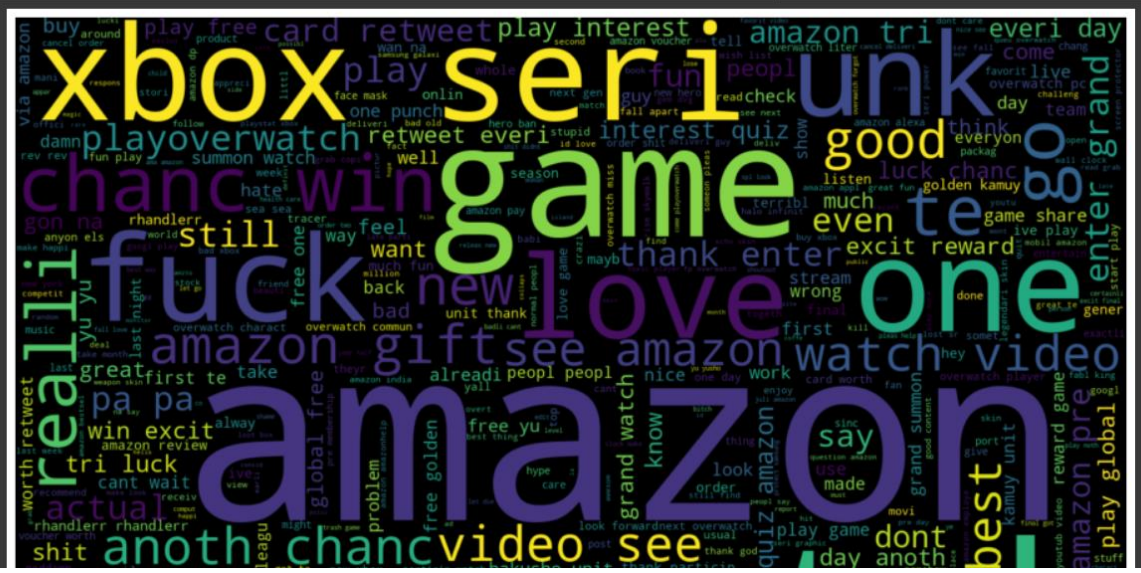
```
amazon       1509
overwatch    1161
play          714
game          595
get           421
             ...
gk              3
rdea            3
qyjn            3
microphon       3
xgzll           3
Length: 2892, dtype: int64
```

```
[ ]  rare_words = rare_words[rare_words <= 2]
```

```
[ ]  df["Text"] = df["Text"].apply(lambda x: " ".join([i for i in x.split() if i not in rare_words.index]))
```

Word Cloud

```
plt.figure(figsize=(16,12))
wordcloud = WordCloud(background_color="black",max_words=500, width=1500, height=1000).generate(' '.join(df['Text']))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



Train test split

```
[ ] from sklearn.model_selection import train_test_split
    from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
    from sklearn.naive_bayes import MultinomialNB, BernoulliNB
    from sklearn.ensemble import RandomForestClassifier
```

```
    X = df["Text"]
    y = df["Sentiment"]
```

```
[ ] X_train,X_test,y_train,y_test = train_test_split(X,y, test_size = 0.30,random_state= 42)
```

Count Vectorizer

```
[ ] from sklearn.feature_extraction.text import CountVectorizer
```

```
[ ] vt = CountVectorizer(analyzer="word")
    X_train_count = vt.fit_transform(X_train)
    X_test_count = vt.transform(X_test)
```

```
[ ] print(X_train_count.toarray())
    X_train_count
```

MultinomialNB

```
[ ]  nb_model = MultinomialNB()
     nb_model.fit(X_train_count,y_train)
```

```
MultinomialNB()
```

```
[ ]  nb_pred = nb_model.predict(X_test_count)
     nb_train_pred = nb_model.predict(X_train_count)
```

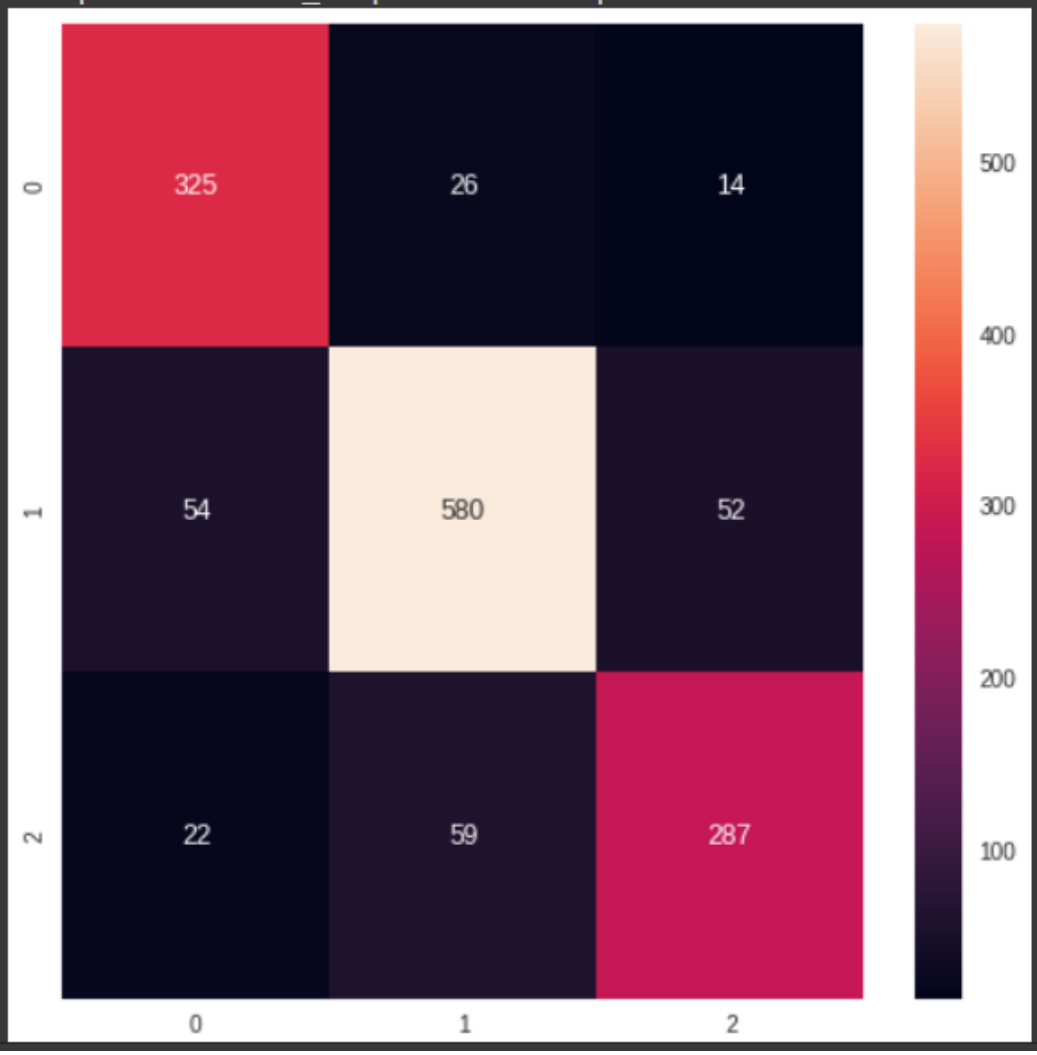
```
     print("X Test")
     print(classification_report(y_test,nb_pred))
     print("X Train")
     print(classification_report(y_train,nb_train_pred))

     plt.figure(figsize=(8,8))
     sns.heatmap(confusion_matrix(y_test,nb_pred),annot = True,fmt = "d")
```

```
X Test
              precision    recall  f1-score   support

          -1       0.81      0.89      0.85       365
           0       0.87      0.85      0.86       686
           1       0.81      0.78      0.80       368

    accuracy                           0.84      1419
   macro avg       0.83      0.84      0.83      1419
weighted avg       0.84      0.84      0.84      1419
```

X Train

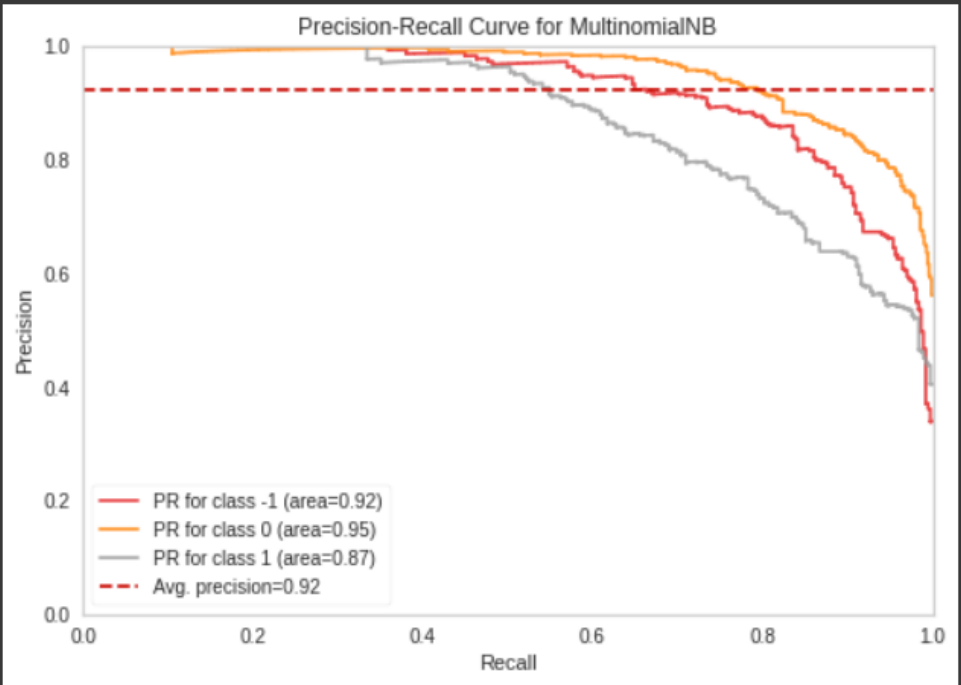
	precision	recall	f1-score	support
-1	0.91	0.91	0.91	832
0	0.92	0.90	0.91	1652
1	0.84	0.86	0.85	827
accuracy			0.89	3311
macro avg	0.89	0.89	0.89	3311
weighted avg	0.89	0.89	0.89	3311

<matplotlib.axes._subplots.AxesSubplot at 0x7f8acbea1090>



```
from yellowbrick.classifier import PrecisionRecallCurve
```

```
viz = PrecisionRecallCurve(MultinomialNB(),
                           classes=nb_model.classes_,   # label ların isimleri il
                           per_class=True,  # bütün class lar grafikte gözüksün d
                           cmap="Set1")
viz.fit(X_train_count,y_train)
viz.score(X_test_count, y_test)  # test datası üzerinden skorları alıyoruz
viz.show();
```



Random Forest

```
from sklearn.model_selection import GridSearchCV
```

```
rf_params = {"max_depth": [2,4,6,10,12],
             "max_features": [2,5,7],
             "n_estimators":[12,15,16,]}
rf = RandomForestClassifier()
```

```
rf_model_cv = GridSearchCV(rf,rf_params,cv = 5,n_jobs = -1)
```

```
rf_model_cv.fit(X_train_count,y_train)
```

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(), n_jobs=-1,
             param_grid={'max_depth': [2, 4, 6, 10, 12],
                         'max_features': [2, 5, 7],
                         'n_estimators': [12, 15, 16]})
```

```
rf_model_cv.best_params_
```

```
{'max_depth': 12, 'max_features': 7, 'n_estimators': 15}
```

```
rf_tuned = RandomForestClassifier(max_depth = 12,
                                  max_features = 7,
                                  min_samples_split = 2).fit(X_train_count,y_train)
```

```
rf_pred = rf_tuned.predict(X_test_count)
rf_train_pred = rf_tuned.predict(X_train_count)
```

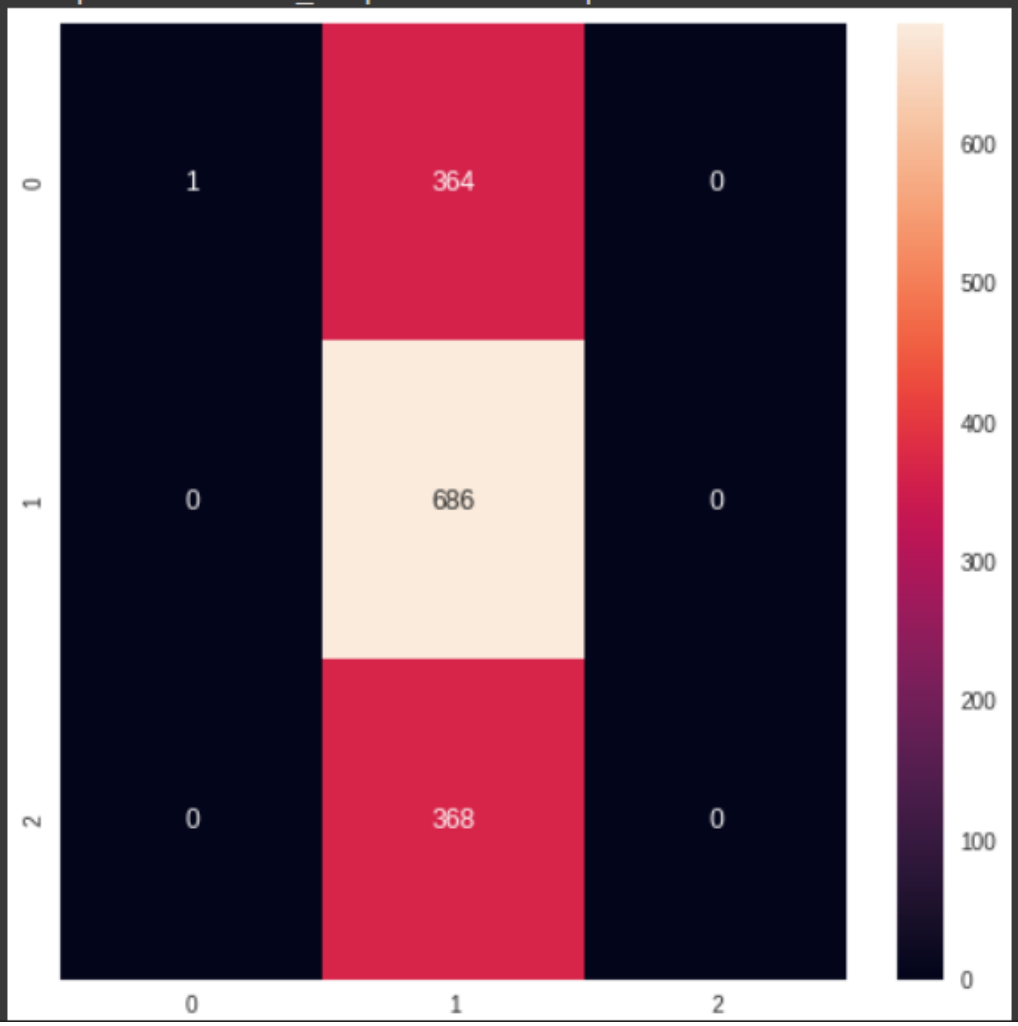
```
print("X Test")
print(classification_report(y_test,rf_pred))
print("X Train")
print(classification_report(y_train,rf_train_pred))

plt.figure(figsize=(8,8))
sns.heatmap(confusion_matrix(y_test,rf_pred),annot = True,fmt = "d")
```

```
X Test
              precision    recall  f1-score   support

          -1       1.00      0.00      0.01       365
           0       0.48      1.00      0.65       686
           1       0.00      0.00      0.00       368

    accuracy                           0.48      1419
   macro avg       0.49      0.33      0.22      1419
weighted avg       0.49      0.48      0.32      1419
```

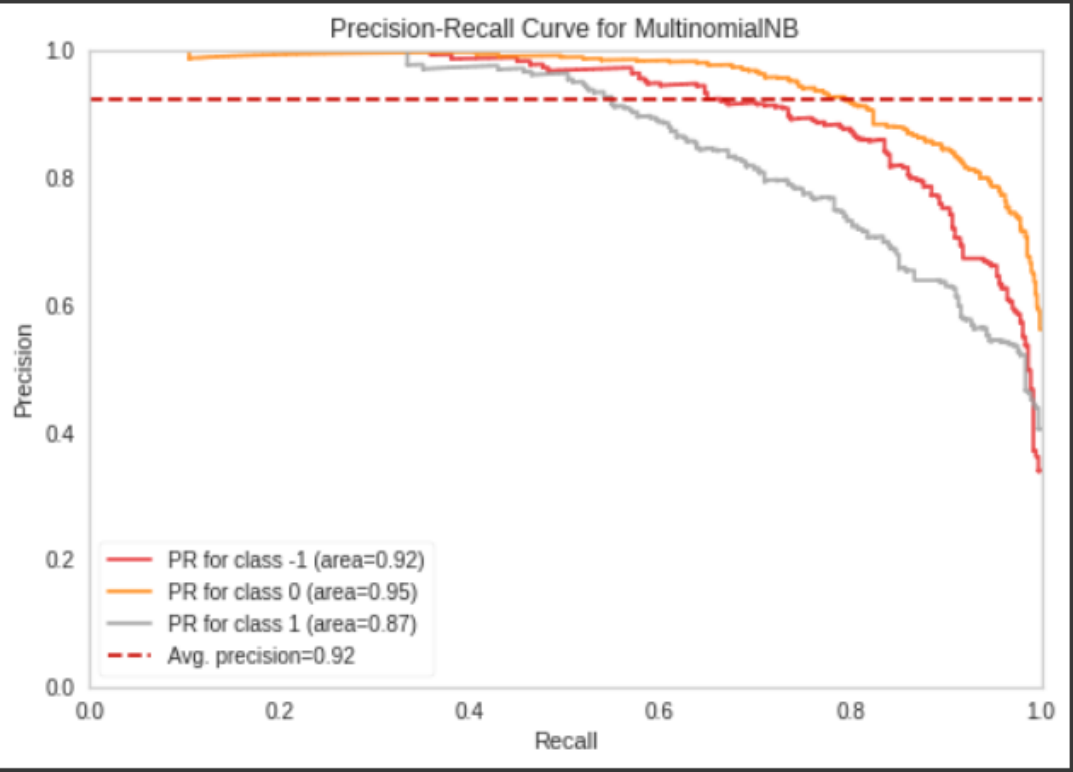
X Train

	precision	recall	f1-score	support
-1	1.00	0.01	0.03	832
0	0.50	1.00	0.67	1652
1	1.00	0.00	0.00	827
accuracy			0.50	3311
macro avg	0.83	0.34	0.23	3311
weighted avg	0.75	0.50	0.34	3311

<matplotlib.axes._subplots.AxesSubplot at 0x7f8acbea0150>

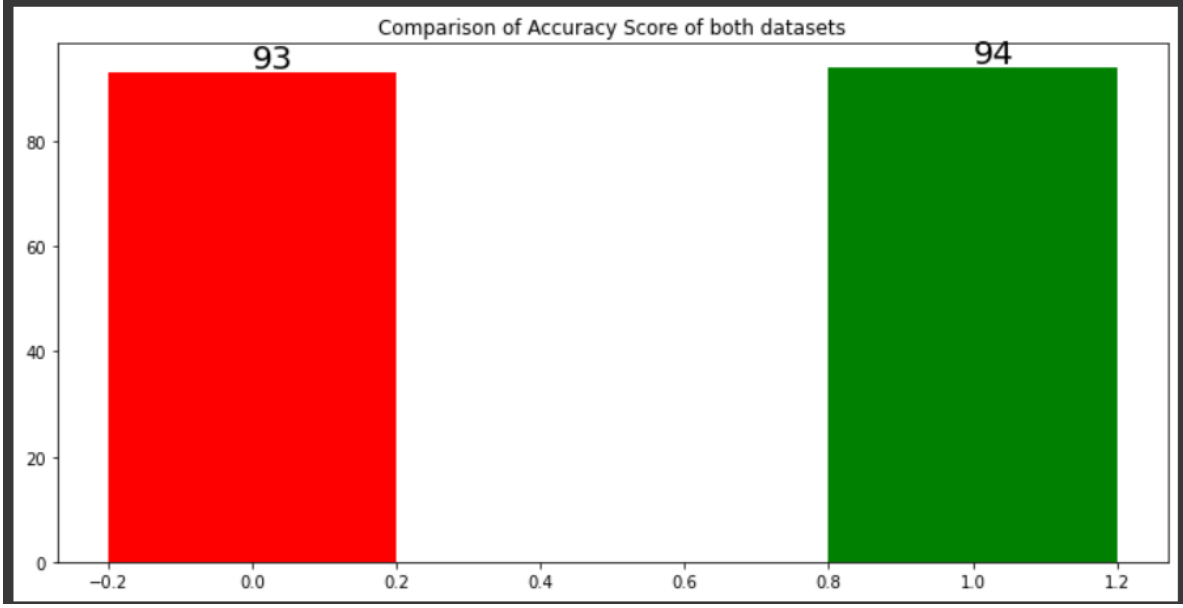


```
viz = PrecisionRecallCurve(MultinomialNB(),
                           classes=rf_tuned.classes_,
                           per_class=True,
                           cmap="Set1")
viz.fit(X_train_count,y_train)
viz.score(X_test_count, y_test)
viz.show();
```



Question 5: Perform a bar chart visualization to show the difference in performance between the two datasets

```
import numpy as np
import matplotlib.pyplot as plt
N=2
menMeans=(93,94)
ind=np.arange(N)
c=["red","green"]
fig, ax= plt.subplots(figsize=(10,5))
ax.bar(ind,menMeans,width=0.4,color=c)
for index,data in enumerate(menMeans):
    plt.text(x=index,y=data+1, s=f"{data}", fontdict=dict(fontsize=20))
plt.tight_layout()
plt.title("Comparison of Accuracy Score of both datasets")
plt.show()
```



Conclusion:

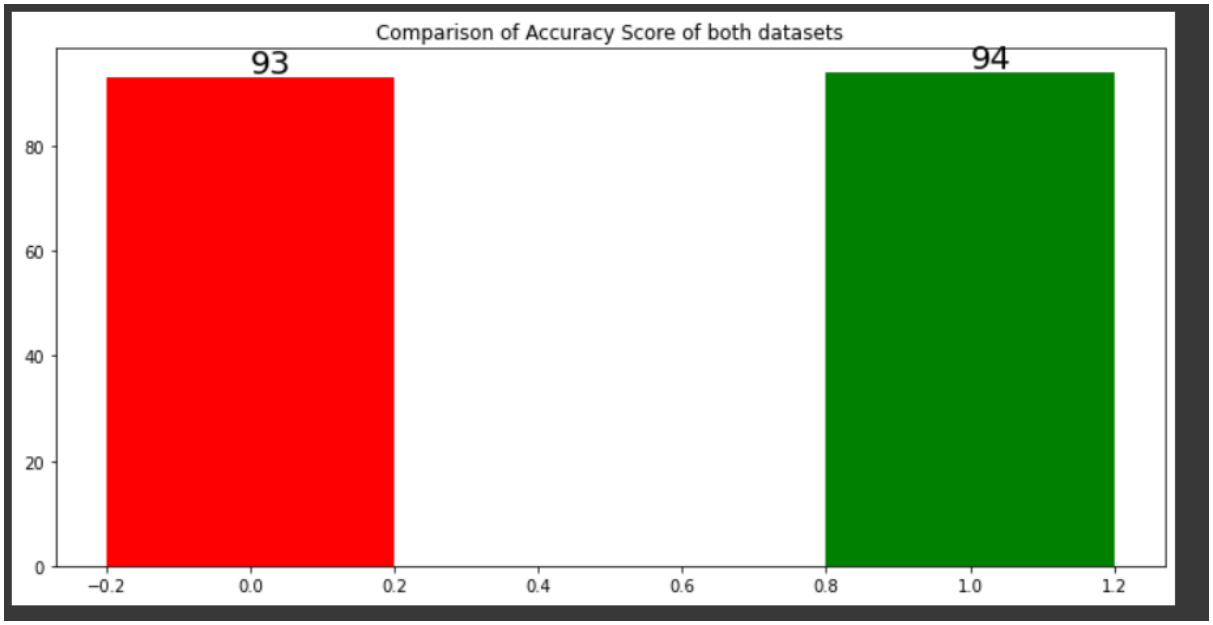
The task of sentiment analysis, especially in the domain of micro-blogging, is still in the developing stage and far from complete. So, we propose a couple of ideas which we feel are worth exploring in the future and may result in further improved performance. Last but not the least, we can attempt to model human confidence in our system. For example, if we have 5 human labeler's labelling each tweet, we can plot the tweet in the 2-dimensional objectivity / subjectivity and positivity / negativity plane while differentiating between tweets in which all 5 labels agree, only 4 agree, only 3 agree or no majority vote is reached. We could develop our custom cost function for coming up with optimized class boundaries such that highest weightage is given to those tweets in which all 5 labels agree and as the number of agreements start decreasing, so do the weights assigned. In this way the effects of human confidence can be visualized in sentiment analysis.

Results and Discussion:

Accuracy and other performance metrics comparison

Two datasets have been used based on the topic of "Twitter Sentiment Analysis". For each dataset, the required pre-processing, splitting of dependent and independent features, train and test split, vectorization have been completed and prepared for model building process.

For each dataset, a machine learning model has been built, fit and the metrics have been compared.



The accuracy score of Logistic Regression model on dataset 1 is 93%, whereas the accuracy score of Logistic Regression model on dataset 2 is comparatively higher, i.e., 94%.

For Dataset 1:

X Test

	precision	recall	f1-score	support
-1	0.85	0.79	0.82	303
0	0.84	0.79	0.81	565
1	0.80	0.89	0.84	554
accuracy			0.83	1422
macro avg	0.83	0.82	0.82	1422
weighted avg	0.83	0.83	0.82	1422

X Train

	precision	recall	f1-score	support
-1	0.90	0.79	0.84	701
0	0.87	0.87	0.87	1341
1	0.86	0.91	0.88	1274
accuracy			0.87	3316
macro avg	0.88	0.86	0.87	3316
weighted avg	0.87	0.87	0.87	3316

For Dataset 2:

X Test

	precision	recall	f1-score	support
-1	0.81	0.89	0.85	365
0	0.87	0.85	0.86	686
1	0.81	0.78	0.80	368
accuracy			0.84	1419
macro avg	0.83	0.84	0.83	1419
weighted avg	0.84	0.84	0.84	1419

X Train

	precision	recall	f1-score	support
-1	0.91	0.91	0.91	832
0	0.92	0.90	0.91	1652
1	0.84	0.86	0.85	827
accuracy			0.89	3311
macro avg	0.89	0.89	0.89	3311
weighted avg	0.89	0.89	0.89	3311

Accuracy scores of X test and X train is higher in dataset 2 when compared with dataset 1 and hence dataset 2 gives better accuracy results.

From the above graphs, reports and metrics evaluated, it is evident that the Logistic Regression model performed better and gave a higher accuracy when trained on dataset 2 with comparison to dataset 1.