RATINGS PREDICTION -NPL CLASSIFICATION PROJECT

Nowadays, a massive amount of reviews is available online. Besides offering a valuable source of information, these informational contents generated by users, also called User Generated Contents (UGC) strongly impact the purchase decision of customers. As a matter of fact, a recent survey (Hinckley, 2015) revealed that 67.7% of consumers are effectively influenced by online reviews when making their purchase decisions. More precisely, 54.7% recognized that these reviews were either fairly, very or absolutely important in their purchase decision making. Relying on online reviews has thus become a second nature for consumers.

In their research process, consumers want to find useful information as quickly as possible. However, searching and comparing text reviews can be frustrating for users as they feel submerged with information (Ganu, Elhada & Marian, 2009). Indeed, the massive amount of text reviews as well as its unstructured text format prevent the user from choosing a product with ease. The star-rating, i.e. stars from 1 to 5 on Amazon, rather than its text content gives a quick overview of the product quality. This numerical information is the number one factor used in an early phase by consumers to compare products before making their purchase decision.

Step 1:Importing Libraries :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

import nltk
import string
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import nltk
from nltk.stem import PorterStemmer, WordNetLemmatizer
from wordcloud import WordCloud
from sklearn.feature_extraction.text import TfidfVectorizer
import xgboost
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,
confusion_matrix,
classification_report,roc_curve,roc_auc_score,auc
from sklearn.model_selection import train_test_split
from sklearn.metrics import
accuracy_score,classification_report,confusion_matrix,f1_sco
re
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import
cross_val_score,GridSearchCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import
RandomForestClassifier,AdaBoostClassifier,GradientBoostingCl
assifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier

import joblib

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

Step 2: Importing Dataset :

```
data = pd.read_csv("/content/sample_data/Review_data.csv")


data.head()
```

	Unnamed: 0	Number of ratings	Full Review
0	0	5	Good phone at this price point\nCamera is by f...
1	1	5	Awesome Purchase...\nFirstly, thank you Flipkart...
2	2	5	Very nice product... i am fully satisfied than...
3	3	5	Apple iPhone 12 mini is incredible, peaceful t...
4	4	5	Switched from android.Truecaller not working f...

```
data.shape
```

```
(21367, 3)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21367 entries, 0 to 21366
Data columns (total 3 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Unnamed: 0         21367 non-null  int64 
 1   Number of ratings  21367 non-null  int64 
 2   Full Review        21367 non-null  object
dtypes: int64(2), object(1)
memory usage: 500.9+ KB
```

```
data.isnull().sum()
```

```
Unnamed: 0           0
Number of ratings    0
Full Review          0
dtype: int64
```

Step 3: Problem Statement and Background

In machine learning, classification is used to classify a new observation into a specific set/category based on a training set of data containing observations whose category is known in advance. The most common example is “spam” or “non-spam” classes for emails. In E-Commerce,

classifier algorithms can be used to classify sentiments of review based on words. The specific words in the language are categorized in advance for their positive or negative sentiments.

Classification is an instance of supervised learning. Training set has correctly identified observations. Classifier algorithms are used to create cluster/sets from the uncategorized unsupervised data based on similarity and/or distance from the training data set.

Step 4: Studying data & identifying hidden patterns

Datasets for the 6 different categories were merged in a single data-frame, with below dimensions:

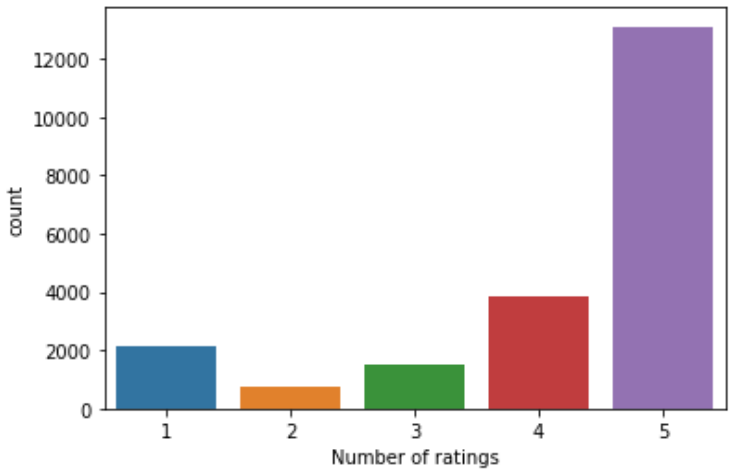
Total records = 21367

Total attributes = 2

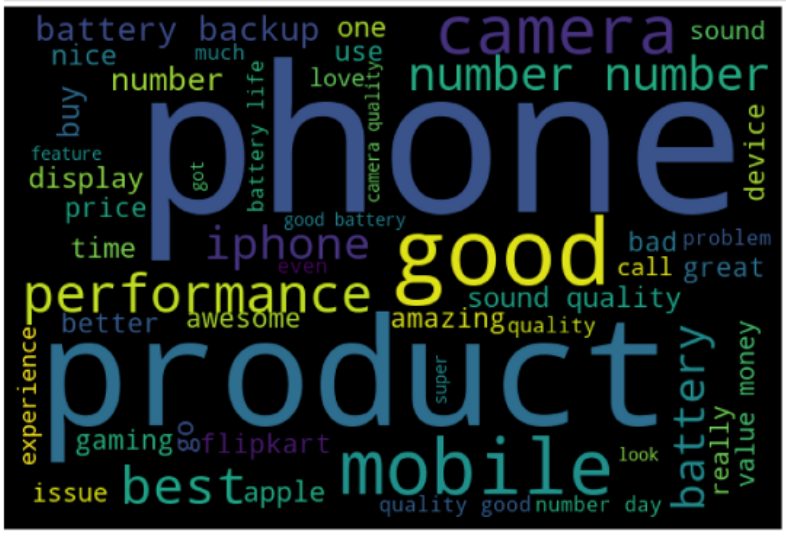
The features of interest were found to be ‘reviewText’, ‘summary’ and ‘rating’ as these were most useful and relevant for model building,prediction as well as abstraction. In the ‘reviewText’ field, mixed bag of short and long reviews were found, some of which produced no or little information about the product. In the exploratory analysis, reviewText attribute was explored using the word-cloud visualization for each category.Figure 2 shows word frequencies of tokens in the train set of Office Products category.

```
#checking the values counts

sns.countplot(x='Number of ratings',data=data)
```




```
# checking the most frequent words
words = data['Full Review']
words_cloud = WordCloud(width=600, height=400,
background_color = 'black', max_words=50).generate(''.
join(words))
plt.figure(figsize=(10,8), facecolor='k')
plt.imshow(words_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```



This ***visualisation shows*** *the number of frequent words.*

STEP 5: Data Preprocessing

1. removal of special character ***marks:*** There are many numeric and special characters which we have to remove from dataset.

```
#replacing email address with email
data['Full Review'] =data['Full
Review'].str.replace(r'^.+@[^\.].*\.[a-z]
{2,}$','emailaddress')

# Replace URLs with 'webaddress'
data['Full Review'] = data['Full
Review'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]
{2,3}(/\S*)?$','webaddress')

# Replace money symbols with 'moneysymb' (£ can by typed
with ALT key + 156)
data['Full Review'] = data['Full
Review'].str.replace(r'£|\$', 'dollers')

# Replace 10 digit phone numbers (formats include
paranthesis, spaces, no spaces, dashes) with 'phonenumber'
data['Full Review'] = data['Full Review'].str.replace(r'^\(?
[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$','phonenumber')

# Replace numbers with 'number'
data['Full Review'] = data['Full Review'].str.replace(r'\d+
(\.\d+)?', 'number')

# Remove punctuation
data['Full Review'] = data['Full
Review'].str.replace(r'[^\w\d\s]', ' ')

# Replace whitespace between terms with a single space
data['Full Review'] = data['Full
Review'].str.replace(r'\s+', ' ')

# Remove leading and trailing whitespace
data['Full Review'] = data['Full
Review'].str.replace(r'^\s+|\s+?$', '')
```

2. Preparation for removal of punctuation marks & Updating the list of stop words

I imported the string library comprising all punctuation characters and appended the numeric digits to it, as those were required to be removed too.

Stemming is the process of converting inflected/derived words to their word stem or the root form. Basically, a large number of similar origin words are converted to the same word. E.g. words like “stems”, “stemmer”, “stemming”, “stemmed” are based on “stem”. This helps in achieving the training process with a better accuracy.

Lemmatising is the process of grouping together the inflected forms of a word so they can be analyzed as a single item. This is quite similar to stemming in its working but not exactly same. Lemmatising depends on correctly identifying the intended part of speech and meaning of a word in a sentence, as well as within the larger context surrounding that sentence, such as neighboring sentences or even an entire document. I used the ***word-net library*** in nltk for this purpose. Stemmer and Lemmatizer were imported from nltk.

3. Applying Tf-id Vectorizer :

Tf-id *Vectorizer is used for converting a string of words into a matrix of words. Column headers have the words themselves and the cell values signify the frequency of occurrence of the word.*

```
#  Convert text into vectors using TF-IDF
tf_vec = TfidfVectorizer(max_features = 20000,
stop_words='english')
X_features = tf_vec.fit_transform(data['Full Review'])
X = X_features
```

STEP 6: Model Building :

```
y = data['Number of ratings']


x_train,x_test,y_train,y_test=train_test_split(
X,y,random_state=56,test_size=.30)
```

```
#RandomForestClassifier
RF = RandomForestClassifier()

RF.fit(x_train, y_train)
y_pred_train = RF.predict(x_train)
print('Training accuracy is
{}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = RF.predict(x_test)
print('Test accuracy is
{}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))


# MultinomialNBClassifiear
MN = MultinomialNB()

MN.fit(x_train, y_train)
y_pred_train = MN.predict(x_train)
print('Training accuracy is
{}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = MN.predict(x_test)
print('Test accuracy is
{}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))


Training accuracy is 0.8827895159133459
Test accuracy is 0.6967711745437529
[[ 452   16   17    5  144]
 [ 118   34    3    4   73]
 [  80    7   63   28  302]
 [  29    3   15  130  964]
 [  42    5   13   76 3788]]
              precision    recall  f1-score   support

           1       0.63      0.71      0.67       634
           2       0.52      0.15      0.23       232
           3       0.57      0.13      0.21       480
           4       0.53      0.11      0.19      1141
           5       0.72      0.97      0.82      3924

    accuracy                           0.70      6411
   macro avg       0.59      0.41      0.42      6411
weighted avg       0.66      0.70      0.63      6411
```

```
# GradientBoostingClassifier
GBC = GradientBoostingClassifier()

GBC.fit(x_train, y_train)
y_pred_train = GBC.predict(x_train)
print('Training accuracy is
{}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = GBC.predict(x_test)
print('Test accuracy is
{}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))


Training accuracy is 0.8827895159133459
Test accuracy is 0.6493526750896896
[[ 401   50   47   23  113]
 [  99   44   18   13   58]
 [  73   18   96   44  249]
 [  41    9   58  213  820]
 [  60   25   78  352 3409]]
              precision    recall  f1-score   support

           1       0.59      0.63      0.61       634
           2       0.30      0.19      0.23       232
           3       0.32      0.20      0.25       480
           4       0.33      0.19      0.24      1141
           5       0.73      0.87      0.80      3924

    accuracy                           0.65      6411
   macro avg       0.46      0.42      0.43      6411
weighted avg       0.60      0.65      0.62      6411
```

```
# MultinomialNBClassifiear
MN = MultinomialNB()

MN.fit(x_train, y_train)
y_pred_train = MN.predict(x_train)
print('Training accuracy is
{}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = MN.predict(x_test)
print('Test accuracy is
{}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.6715699384862263
Test accuracy is 0.6526282951177663
[[ 273    0    0    5  356]
 [  71    0    1    1  159]
 [  45    0    0    5  430]
 [  17    0    0   13 1111]
 [  16    0    0   10 3898]]
              precision    recall  f1-score   support

           1       0.65      0.43      0.52       634
           2       0.00      0.00      0.00       232
           3       0.00      0.00      0.00       480
           4       0.38      0.01      0.02      1141
           5       0.65      0.99      0.79      3924

    accuracy                           0.65      6411
   macro avg       0.34      0.29      0.27      6411
weighted avg       0.53      0.65      0.54      6411
```

```
# GradientBoostingClassifier
GBC = GradientBoostingClassifier()

GBC.fit(x_train, y_train)
y_pred_train = GBC.predict(x_train)
print('Training accuracy is
{}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = GBC.predict(x_test)
print('Test accuracy is
{}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 7056699652313453
Test accuracy is 6696303228825456
[[ 331   16   11    9  267]
 [  87   14    3    7  121]
 [  61    5   27   20  367]
 [  13    1   19   60 1048]
 [  30    3    6   24 3861]]
              precision    recall  f1-score   support

           1       0.63      0.52      0.57       634
           2       0.36      0.06      0.10       232
           3       0.41      0.06      0.10       480
           4       0.50      0.05      0.10      1141
           5       0.68      0.98      0.81      3924

    accuracy                           0.67      6411
   macro avg       0.52      0.34      0.34      6411
weighted avg       0.61      0.67      0.58      6411
```

```
from sklearn.svm import SVC
# Support vactor classifier
svc = SVC()

svc.fit(x_train, y_train)
y_pred_train = svc.predict(x_train)
print('Training accuracy is
{}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = svc.predict(x_test)
print('Test accuracy is
{}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))


Training accuracy is 0.7931933672104841
Test accuracy is 0.6920917173607861
[[ 461    2   19    2  150]
 [ 126   13    6    2   85]
 [  95    0   50   10  325]
 [  31    0   23   64 1023]
 [  41    0   12   22 3849]]
              precision    recall  f1-score   support

           1       0.61      0.73      0.66       634
           2       0.87      0.06      0.11       232
           3       0.45      0.10      0.17       480
           4       0.64      0.06      0.10      1141
           5       0.71      0.98      0.82      3924

    accuracy                           0.69      6411
   macro avg       0.66      0.38      0.37      6411
weighted avg       0.67      0.69      0.60      6411
```

STEP 7: Conclution :

To conclude, In this project we have performed various techniques like Data analysis which has proven way for any organizations and enterprises to gain the information they need to make better decisions, serve their customers, and increase productivity and revenue. The benefits of data analysis are almost too numerous to count, and some of the most rewarding benefits include getting the right information for business, getting more value out of it. And it creates more effective marketing campaigns, gaining a better understanding of customers, and so on. In addition, Stemming is the process of converting

inflected/derived words to their word stem or the root form. Basically, a large number of similar origin words are converted to the same word. E.g. words like “stems”, “stemmer”, “stemming”, “stemmed” are based on “stem”. This helps in achieving the training process with a better accuracy.

Lemmatising is the process of grouping together the inflected forms of a word so they can be analyzed as a single item. This is quite similar to stemming in its working but not exactly same. Lemmatising depends on correctly identifying the intended part of speech and meaning of a word in a sentence, as well as within the larger context surrounding that sentence, such as neighboring sentences or even an entire document. I used the word-net library in nltk for this purpose. Stemmer and Lemmatizer were imported from nltk.

Tfidf Vectorizer is used for converting a string of words into a matrix of words. Column headers have the words themselves and the cell values signify the frequency of occurrence of the word.Finally I reached the core part of the project, where I could start building the classifier. I had use 4 major algorithm . Where, our random forest model giving the good accuracy hence selected it as a final predictor.