FLIP ROBO

Ratings Prediction

Submitted by:

SHARUKH ANSARI

ACKNOWLEDGMENT

I’m extremely grateful to SME Khushboo Garg who give me an opportunity to work on this project. I’d also like to show my gratitude to live Data trained support who help me with this project and guided me how to select a best model in case of imbalanced dataset.

For this particular task, I referred the following websites and articles when stuck:

- https://towardsdatascience.com/a-common-mistake-to-avoid-when-encoding-ordinal-features-79e402796ab4

- https://stackoverflow.com/questions/43590489/gridsearchcv-random-forest-regressor-tuning-best-params

INTRODUCTION

Business Problem Framing

Need to predict the ratings (1-5) of various products based on the reviews written by customers based on data scrapped from e-commerce sites.

Conceptual Background of the Domain Problem Ratings Prediction

We have a client who has a website where people write different reviews for technical products. Now they are adding a new feature to their website i.e. The reviewer will have to add stars(rating) as well with the review. The rating is out 5 stars and it only has 5 options available 1 star, 2 stars, 3 stars, 4 stars, 5 stars. Now they want to predict ratings for the reviews which were written in the past and they don't have a rating. So, we have to build an application which can predict the rating by seeing the review.

Data Collection Phase –

We have to scrape at least 20000 rows of data. You can scrape more data as well, it's up to you. More the data better the model. In this section you need to scrape the reviews of different laptops,Phones, Headphones, smart watches, Professional Cameras, Printers, monitors, Home theater, router from different e-commerce websites. Basically, we need these columns-

1) reviews of the product. 2) rating of the product. You can fetch other data as well, if you think data can be useful or can help in the project.

For data collection I have used selenium automation and scrap the data from flipkart and amazon.

Review Of Literature:

In today's era most of the people like to purchase products Online from ease of their home after viewing products rating and Reviews. So it is very important for any online ecommerce to provide rating and reviews of the product.

Motivation behind undertaking of this problem:

As the online purchasing product is becoming trend so it is also important for any ecommerce to predict the rating of the product based on the reviews written by purchaser. So that anyone who is about to purchase that product knows the exact rating of that product. So here we are going to build a model which can understand the context of the reviews and based on that review it can predict the rating of the reviews.

Data Source and their format:

First, we need to scrape ecommerce websites for reviews and ratings of that respective reviews.

```python
# To Store the product Urls
urls = []
# Fetching urls of the products
for i in key:
    time.sleep(1)
    try:
        search_box.send_keys(Keys.CONTROL+"a")# Selecting all the text present inside the search box
        search_box.send_keys(Keys.DELETE)# Deleting it
        search_box.send_keys(i)# Sending keys for searching
        search_btn.click()# Clicking on search button
    except StaleElementReferenceException:
        driver.find_element_by_xpath('//input[@class="_3704LK"]').send_keys(Keys.CONTROL+"a")
        driver.find_element_by_xpath('//input[@class="_3704LK"]').send_keys(Keys.DELETE)
        driver.find_element_by_xpath('//div[@class="col-12-12 _2oO9oE"]//button').click()

    time.sleep(2)
    for i in driver.find_elements_by_xpath('//a[@class="_1fQZEK" or @class="s1Q9rs"]'):# Extracting product url
        urls.append(i.get_attribute('href'))
```

```python
Rating = [] # Creating list to store data
review_s = []
full_r = []
for i in urls:
    driver.get(i)
    time.sleep(2)

    # Clicking on all reviews
    try:
        all_review =driver.find_element_by_xpath("//div[@class='_3UAT2v _16PBlm']//span")
        all_review.click()
        time.sleep(2)
    except:
        continue
    # Defining loop to navigate first 10 pages
    for pages in range(0,10):

        # Fecthing rating
        try:
            for i in driver.find_elements_by_xpath("//div[@class='col _2wzgFH K0kLPL']/div[1]"):
                if i.text is None:
                    Rating.append("---")
                else:
                    Rating.append(i.text.split("\n")[0])

        except StaleElementReferenceException:
            pass


        # Short Reviews
        try:
            for i in driver.find_elements_by_xpath("//div[@class='col _2wzgFH K0kLPL']/div[1]"):
                if i.text is None:
                    review_s.append("---")
                else:
                    review_s.append(i.text.split("\n"))
        except StaleElementReferenceException:
            pass

        # Fetching full reviews
        try:
            for i in driver.find_elements_by_xpath('//div[@class="t-ZTKy"]'):
                if i.text is None:
                    full_r.append("---")
                else:
                    full_r.append(i.text)
        except StaleElementReferenceException:
            pass


        try:
            button=driver.find_element_by_xpath("//*[contains(text(), 'Next')]")
            driver.execute_script("arguments[0].click();", button)
            time.sleep(2)

        except:
            break
```

```python
# To Store the product Urls
amzn_urls = []
for i in key:
    time.sleep(1)
    try:
        driver.find_element_by_id('twotabsearchtextbox').clear()
        driver.find_element_by_id('twotabsearchtextbox').send_keys(i)
        time.sleep(1)
        driver.find_element_by_xpath('//span[@class="nav-search-submit-text nav-sprite nav-progressive-attribute"]').click()
        time.sleep(2)
        for j in driver.find_elements_by_xpath('//h2[@class="a-size-mini a-spacing-none a-color-base s-line-clamp-2"]//a'):
            amzn_urls.append(j.get_attribute('href'))
    except NoSuchElementException:
        driver.refresh()
        continue
```

```python
for i in amzn_urls:
    driver.get(i)
    time.sleep(2)

    try:
        all_review =driver.find_element_by_xpath('//div[@id="reviews-medley-footer"]//a')
        all_review.click()
        time.sleep(2)
    except:
        continue

    try:
            driver.find_element_by_xpath('//a[@data-hook="cr-translate-these-reviews-link"]').click()
    except:
        pass



    try:
        for i in driver.find_elements_by_xpath('//*[@class="a-section celwidget"]/div[2]/a[1]'):
            if i.text is " ":
                Rating.append("---")
            else:
                Rating.append(i.get_attribute('title').split(".")[0])

    except StaleElementReferenceException:
        pass



    try:
        for i in driver.find_elements_by_xpath('//a[@data-hook="review-title"]'):
            if i.text is " ":
                review_s.append("---")
            else:
                review_s.append(i.text.split("\n"))
    except StaleElementReferenceException:
        pass

    try:
        for i in driver.find_elements_by_xpath('//span[contains(@data-hook,"review-body")]'):
            if i.text is " ":
                full_r.append("---")
            else:
                full_r.append(i.text)
    except NoSuchElementException:
        for i in range(0,len(review_s)):
            full_r.append("--")
```

I have scrapped the data from Flipkart and Amazon. Because it help to avoid overfitting of the model.

Sample dataset:

	Ratings	Full Reviews	Short Reviews
0	5	Awesome Watch , More better than I ve expected...	[5, Excellent]
1	5	It's charging time Is not good If you will use...	[5, Awesome]
2	5	Extremely value for money	[5, Terrific]
3	5	Nice product	[5, Best in the market!]
4	3	Super	[3, Fair]

I have fetched the short reviews and full Reviews as well. So that **in future we can also build a model which can predict the rating of the comment based on short reviews as well.**

Mathematical modelling and Data Cleanings:

We need to find and replace stop words, special characters and patterns from the datasets. We drop the stop words from the dataset because it doesn't carry any information from the dataset. Removing punctuation, and special character is very important because they did not carry any information.

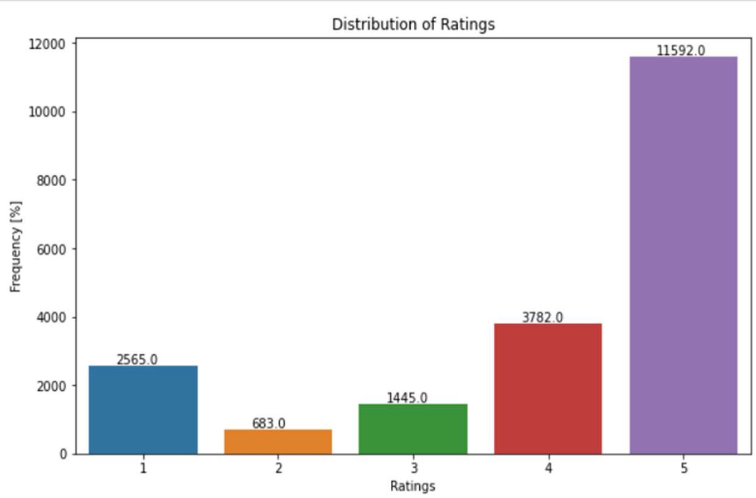
1. Firstly, we need to make sure that the data is converted to one case so I have converted all the comments to lower case.

2. Regex Expression: - A regular expression is a sequence of characters that specifies a search pattern. Usually, such patterns are used by string-searching algorithms for "find" or "find and replace" operations on strings, or for input validation. We have used to replace punctuation, and other special character which we need to remove from the dataset.

3. We have seen the distribution of data before and after cleaning and we have noticed that after cleaning the dataset the length of the comment reduced.

Checking Target column distribution:

```
plt.figure(figsize=(10,6))
ax = sns.countplot(x="Ratings", data=df)
plt.title('Distribution of Ratings')
plt.xlabel('Ratings')
plt.ylabel('Frequency [%]')

for p in ax.patches:
        ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.2, p.get_height()+70))
```



Pre-Processing Steps:

Data Cleaning

```
# Cheking very first full reviews
df["Full Reviews"][0]
```

```
"Awesome Watch , More better than I ve expected, I m loving it 😀 it's look like my apple smartwatch , it's Amazing Experience flippant, its an outstanding product 😀 👌 with a low price , I think in this price range this is the best nd usefull thing flipkart ,Too good 👍 And too many Applications and things in this watch ⌚ just 👏 👏 👏 👏"
```

```
df["Full Reviews"][0]
```

```
"Awesome Watch , More better than I ve expected, I m loving it 😀 it's look like my apple smartwatch , it's Amazing Experience flippant, its an outstanding product 😀 👌 with a low price , I think in this price range this is the best nd usefull thing flipkart ,Too good 👍 And too many Applications and things in this watch ⌚ just 👏 👏 👏 👏"
```

```
# Cheking very first short reviews
df["Short Reviews"][0]
```

```
"['5', 'Excellent']"
```

```
df["Full Reviews"] = df["Full Reviews"].apply(lambda x: str(x))
df["Short Reviews"] = df["Short Reviews"].apply(lambda x: str(x))
```

```
# Create function to remove emoji and emoticons
def emoji(string):
    emoji_pattern = re.compile("["
                           u"\U0001F600-\U0001F64F"  # emoticons
                           u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                           u"\U0001F680-\U0001F6FF"  # transport & map symbols
                           u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                           u"\U00002702-\U000027B0"
                           u"\U000024C2-\U0001F251"
                           "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r'', string)
```

```
# Function To remove url, html text, Numeric character
def remove_URL(text):
    url = re.compile(r"https?://\S+|www\.\S+")
    return url.sub(r"",text)

def remove_html(text):
    html = re.compile(r"<.*?>")
    return html.sub(r"",text)


def removing_NumericCharacters(string):
    regex = "[0-9]"
    return (re.sub(regex, "", string))
```

```
# Applying above mention function on the dataset using map
df["Full Reviews"] = df["Full Reviews"].map(lambda x: remove_URL(x))
df["Full Reviews"] = df["Full Reviews"].map(lambda x: remove_html(x))
df["Full Reviews"] = df["Full Reviews"].map(lambda x: emoji(x))


df["Short Reviews"] = df["Short Reviews"].map(lambda x: remove_URL(x))
df["Short Reviews"] = df["Short Reviews"].map(lambda x: remove_html(x))
df["Short Reviews"] = df["Short Reviews"].map(lambda x: emoji(x))
df["Short Reviews"] = df["Short Reviews"].map(lambda x: removing_NumericCharacters(x))
```

```
# Performing further cleaning
lemmatizer = nltk.stem.WordNetLemmatizer()

def cleaning(df,stop_words):
    #Converting to lower case
    df["Full Reviews"] = df["Full Reviews"].str.lower()

    # Removing stopwords
    df["Full Reviews"] = df["Full Reviews"].apply(lambda x: " ".join(x for x in x.split() if x not in stop_words))

    # Lemmatization
    df["Full Reviews"] = df["Full Reviews"].apply(lambda x:" ".join([lemmatizer.lemmatize(x) for x in x.split()]))

    # Removing Punchuation
    df["Full Reviews"] = df["Full Reviews"].str.replace(r'[^\w\d\s]',"")

    # Removing Extra Whitespaces using split and join method
    df["Full Reviews"] = df["Full Reviews"].apply(lambda x:" ".join(x for x in x.split()))

    # Replacing Email address
    df["Full Reviews"]= df["Full Reviews"].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$',"emailaddress")

    # Replacing \n
    df["Full Reviews"] = df["Full Reviews"].replace("\n"," ")

    return df

# I am appending few stopwords which people uses while writting any comment
stop_words = set(stopwords.words("english")+["aww","hmm","cant","dont","u","ur","4","d","e","im","hey","yo","ja"])
df = cleaning(df,stop_words)
```

Spelling Correction:

While some users write reviews they made some spelling mistakes and this can decrease our model accuracy.

```
# Checking Spelling and replacing it with the correct word using spellchecker
from spellchecker import SpellChecker

spell = SpellChecker()

df["Full Reviews"] = df["Full Reviews"].apply(lambda x:" ".join([spell.correction(x) for x in x.split()]))
```

For further EDA I have categorized the rating into good or bad. The rating 3 and above 3 is consider as good whereas the rating below 3 is consider as poor.

Extracting features from comments:

TF-IDF :- **TF-IDF** stands for Term Frequency Inverse Document Frequency of records. It can be defined as the calculation of how relevant a word in a series or corpus is to a text. The meaning increases proportionally to the number of times in the text a word appears but is compensated by the word frequency in the corpus (data-set). It is used to convert text into vectors because the machine learning model only understand the vectors.

```
# Converting text into vectors
tf_vec = TfidfVectorizer()
features = tf_vec.fit_transform(df["Full Reviews"])
```

Hardware and Software Requirement:

- Minimum core i5 or higher
- Minimum 8gb of ram
- Sufficient disk space Software and tools required

Software Requirement

- Python is widely used in scientific and numeric computing
- SciPy is a collection of packages for mathematics, science, and engineering.
- Pandas is a data analysis and modelling library.

Modules or library required for project data analysis and visualization

- Pandas for data analysis and import
- NumPy to perform Mathematical operation.
- Nltk to perform text processing
- Seaborn and matplotlib to data visualization
- Scikit-learn: All the models, metrics and feature selection etc are present inside of that module. We import from this library according to our need

Modeling

Models used: **Multinomial Naive Bayes, Bagging Classifier,Bernouli classifier, Random Forest classifier and CatBoostClassifier**. Rating classification done using traditional machine learning techniques.

Reason for choosing Multinomial Naive Bayes, Bagging Classifier, Bernouli classifier, Random Forest classifier and CatBoost Classifier: All are good at imbalanced dataset and handling large number of features in the dataset.

Splitting Training and Testing Data: Splitting the data into training and test datasets, where training data contains 70 percent and test data contains 25 percent.

Applying model: I trained the model for all the model without tuning hyperparameters as I got results with default parameter settings.

MultinomialNB

```
mnb = MultinomialNB()
mnb.fit(x_train,y_train)
```

```
MultinomialNB()
```

```
y_pred = mnb.predict(x_test)
```

```
print("Accuracy Score is:",accuracy_score(y_test,y_pred),"\n","=-"*60,"\n",
      "Classification report :\n",classification_report(y_test,y_pred),
      "\n","=-"*60,"\n","Cross Validation Score :",cross_val_score(mnb,x,y,cv=5).mean())
```

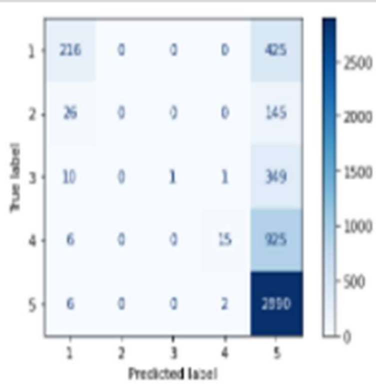
```
Accuracy Score is: 0.6222842336057405
 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
 Classification report :
               precision    recall  f1-score   support

           1       0.82      0.34      0.48       641
           2       0.00      0.00      0.00       171
           3       1.00      0.00      0.01       361
           4       0.83      0.02      0.03       946
           5       0.61      1.00      0.76      2898

    accuracy                           0.62      5017
   macro avg       0.65      0.27      0.25      5017
weighted avg       0.69      0.62      0.50      5017

 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
 Cross Validation Score : 0.6178808756941038
```

```
a.append(accuracy_score(y_test,y_pred))
c.append(cross_val_score(mnb,x,y,cv=5).mean())
f.append(f1_score(y_pred,y_test,average='macro'))
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[ 216,    0,    0,    0,  425],
       [  26,    0,    0,    0,  145],
       [  10,    0,    1,    1,  349],
       [   6,    0,    0,   15,  925],
       [   6,    0,    0,    2, 2890]], dtype=int64)
```

```
disp = plot_confusion_matrix(mnb, x_test, y_test,
                             cmap=plt.cm.Blues)
plt.show()
```



Bagging Classifier

```
bag_clf = BaggingClassifier()

bag_clf.fit(x_train,y_train)
```

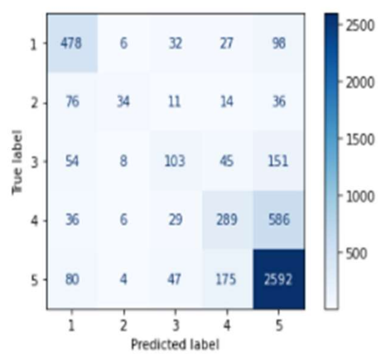
```
BaggingClassifier()
```

```
y_pred = bag_clf.predict(x_test)
```

```
print("Accuracy Score is:",accuracy_score(y_test,y_pred),"\n","=-"*60,"\n",
      "Classification report :\n",classification_report(y_test,y_pred),"\n","=-"*60,
      "\n","Cross Validation Score :",cross_val_score(bag_clf,x,y,cv=5).mean())
```

```
Accuracy Score is: 0.6968307753637633
 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
 Classification report :
               precision    recall  f1-score   support

           1       0.66      0.75      0.70       641
           2       0.59      0.20      0.30       171
           3       0.46      0.29      0.35       361
           4       0.53      0.31      0.39       946
           5       0.75      0.89      0.81      2898

    accuracy                           0.70      5017
   macro avg       0.60      0.49      0.51      5017
weighted avg       0.67      0.70      0.67      5017

 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
 Cross Validation Score : 0.6293915973882094
```

```
disp = plot_confusion_matrix(bag_clf, x_test, y_test,
                             cmap=plt.cm.Blues)
plt.show()
```



Bernouli Naive Bayes Classifier

```
bernouli = BernoulliNB()
bernouli.fit(x_train,y_train)
```

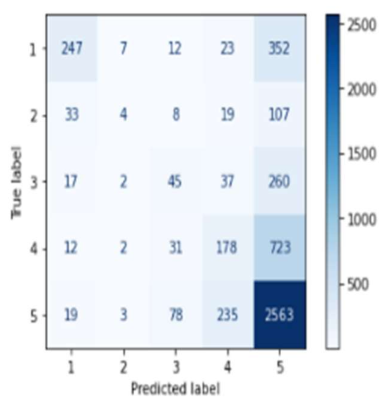
```
BernoulliNB()
```

```
y_pred = bernouli.predict(x_test)
```

```
print("Accuracy Score is:",accuracy_score(y_test,y_pred),"\n","=-"*60,"\n",
      "Classification report :\n",classification_report(y_test,y_pred),
      "\n","=-"*60,"\n","Cross Validation Score :",cross_val_score(bernouli,x,y,cv=5).mean())
```

```
Accuracy Score is: 0.6053418377516444
 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
 Classification report :
               precision    recall  f1-score   support

           1       0.75      0.39      0.51       641
           2       0.22      0.02      0.04       171
           3       0.26      0.12      0.17       361
           4       0.36      0.19      0.25       946
           5       0.64      0.88      0.74      2898

    accuracy                           0.61      5017
   macro avg       0.45      0.32      0.34      5017
weighted avg       0.56      0.61      0.55      5017

 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
 Cross Validation Score : 0.5786114783158024
```

```
disp = plot_confusion_matrix(bernouli, x_test, y_test,
                             cmap=plt.cm.Blues)
plt.show()
```



RandomForestClassifier:

```
rf = RandomForestClassifier()
rf.fit(x_train,y_train)
```

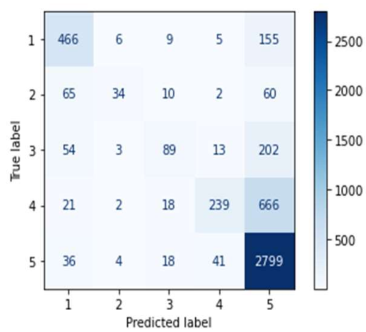
```
RandomForestClassifier()
```

```
y_pred = rf.predict(x_test)
```

```
print("Accuracy Score is:",accuracy_score(y_test,y_pred),"\n","=-"*60,"\n",
      "Classification report :\n",classification_report(y_test,y_pred),
      "\n","=-"*60,"\n","Cross Validation Score :",cross_val_score(rf,x,y,cv=5).mean())
```

```
Accuracy Score is: 0.7229419972094877
 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
 Classification report :
               precision    recall  f1-score   support

           1       0.73      0.73      0.73       641
           2       0.69      0.20      0.31       171
           3       0.62      0.25      0.35       361
           4       0.80      0.25      0.38       946
           5       0.72      0.97      0.83      2898

    accuracy                           0.72      5017
   macro avg       0.71      0.48      0.52      5017
weighted avg       0.73      0.72      0.68      5017

 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
 Cross Validation Score : 0.6537600705033008
```

```
disp = plot_confusion_matrix(rf, x_test, y_test,
                             cmap=plt.cm.Blues)
plt.show()
```



CatBoostClassifier

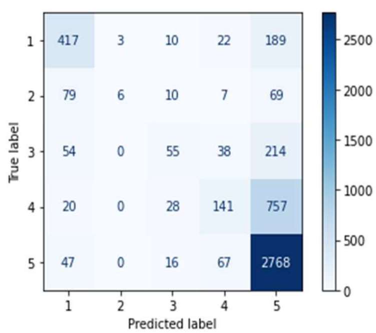
```
from catboost import CatBoostClassifier

classifier = CatBoostClassifier(loss_function='MultiClass',depth=10,iterations=100)
classifier.fit(x_train,y_train)
```

```
Learning rate set to 0.5
0:      learn: 1.2037200        total: 6.06s    remaining: 9m 59s
1:      learn: 1.0927844        total: 12.7s    remaining: 10m 20s
2:      learn: 1.0540795        total: 19.6s    remaining: 10m 34s
3:      learn: 1.0147070        total: 26.3s    remaining: 10m 30s
4:      learn: 0.9913735        total: 32.5s    remaining: 10m 17s
5:      learn: 0.9748378        total: 40s      remaining: 10m 26s
6:      learn: 0.9688873        total: 47.4s    remaining: 10m 29s
7:      learn: 0.9562239        total: 54.6s    remaining: 10m 28s
8:      learn: 0.9499192        total: 1m 2s    remaining: 10m 27s
```

```python
print("Accuracy Score is:",accuracy_score(y_test,y_pred),"\n","=-"*60,"\n",
      "Classification report :\n",classification_report(y_test,y_pred),
      "\n","=-"*60,"\n","Cross Validation Score :",cv)
```

```
Accuracy Score is: 0.6751046442096871
 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
 Classification report :
               precision    recall  f1-score   support

           1       0.68      0.65      0.66       641
           2       0.67      0.04      0.07       171
           3       0.46      0.15      0.23       361
           4       0.51      0.15      0.23       946
           5       0.69      0.96      0.80      2898

    accuracy                           0.68      5017
   macro avg       0.60      0.39      0.40      5017
weighted avg       0.64      0.68      0.61      5017

 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
 Cross Validation Score : 0.6510685439238271
```

```python
disp = plot_confusion_matrix(classifier, x_test, y_test,
                                 cmap=plt.cm.Blues)
plt.show()
```



Final Model after hyperparameter Tuning:

```python
rf = RandomForestClassifier(random_state=0,n_estimators=200,max_depth=None,criterion='entropy')
```

```python
rf.fit(x_train,y_train)
```

```
RandomForestClassifier(criterion='entropy', n_estimators=200, random_state=0)
```

```python
y_pred = rf.predict(x_test)
```

```python
accuracy_score(y_test,y_pred)
```

```
0.725134542555312
```

```python
f1_score(y_test,y_pred,average='weighted')
```

```
0.6787631478844812
```

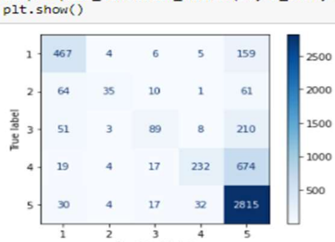
```python
print("\n",classification_report(y_test,y_pred))
```

```
               precision    recall  f1-score   support

           1       0.74      0.73      0.73       641
           2       0.70      0.20      0.32       171
           3       0.64      0.25      0.36       361
           4       0.83      0.25      0.38       946
           5       0.72      0.97      0.83      2898

    accuracy                           0.73      5017
   macro avg       0.73      0.48      0.52      5017
weighted avg       0.74      0.73      0.68      5017
```

```python
print("Cross Validation Score :",cross_val_score(rf,x,y,cv=5).mean())
```

```
Cross Validation Score : 0.6539095473343919
```

```python
disp = plot_confusion_matrix(rf, x_test, y_test,cmap=plt.cm.Blues)
plt.show()
```



Visualization :-

Word cloud of Full Reviews of Rating 3 and Above

```
# Generating WordCloud for rating 3 and above
from wordcloud import WordCloud
def word_cloud(text,color):
    good = df["Full Reviews"][df["Classification"]==text]
    wordcloud = WordCloud(width=400,height=300,background_color='white',max_words=40).generate(" ".join(good))
    plt.figure(figsize=(6,8),facecolor=color)
    plt.imshow(wordcloud)
    plt.axis('off')
    plt.tight_layout(pad=0)
    return plt.show()

word_cloud("Good","g")
```

For rating 3 and above we can see that words like nice product, good, quality is frequently.

```
# Generating WordCloud for rating below 3
word_cloud("Bad","r")
```

From above word cloud for rating below 3. we can see that there is word like working, product, Bad this kind of word used Frequently.

Short Review WordCloud:

If we just take a look at the Short reviews of the customer for the rating 3 and above we can see that they use the words like excellent, wonderful, good, worth, penny etc frequently.

```
Srt_wordcloud("Bad","r")
```

If we just take a look at the Short reviews of the customer for the rating below 3 we can see that they use the words like worst experience, waste money, good, poor, unsatisfactory etc frequently. We can also build a model which can take short reviews as input and generate output for us

Conclusions:

	Model	Accuracy score	Cross Validation Score	F1 score
0	MultnomialNB	0.622284	0.617881	0.511313
1	Bagging classifier	0.696831	0.628644	0.856538
2	Bernouli Naive bayes	0.605342	0.578611	0.597253
3	RandomForest Classifier	0.722942	0.654258	0.672709
4	CatBoostClassifier	0.675105	0.651069	0.660222

As we know when that when the dataset is imbalanced at that time we need to consider **F1 Score** to choose the Best Model. **I have calculate weighted F1 Score for all the model because it take class imbalance into account**. Here that model is **Bagging Classifier**. So i am gonna tune that RandomForestClassifier with best parameter using hyperparameter Tunning.**

- We have selected the best model on the basis of F1 score.
- As we can see above that Bagging Classifier giving the best F1 score.
- We have built which can classify the ratings of a given comments

Scope of Improvements:

- As my data was limited, I did not go for under-sampling, but it was needed as not all classes have equal percentage of data for model training. Ideally, I could have reduced the number of data points having Rating = 5, as that contributes to about 35-40% of the data.

- Need to fetch more data of other rating other than 5.