FLIP ROBO

Ratings Prediction
Project Report

Submitted By:

Vishaldeep Agrawal

ACKNOWLEDGMENT

I would like to express my deepest gratitude to my SME (Subject Matter Expert) Shubham Yadav as well as Flip Robo Technologies who gave me the opportunity to do this project on Rating Prediction, which also helped me in doing lots of research wherein I came to know about so many new things especially the Natural Language Processing and Natural Language Toolkit parts.

Also, I have utilized a few external resources that helped me to complete this project. I ensured that I learn from the samples and modify things according to my project requirement. All the external resources that were used in creating this project are listed below:

1) https://www.google.com/

2) https://www.youtube.com/

3) https://scikit-learn.org/stable/user_guide.html

4) https://github.com/

5) https://www.kaggle.com/

6) https://medium.com/

7) https://towardsdatascience.com/

8) https://www.analyticsvidhya.com/

INTRODUCTION

Business Problem Framing

The rise in E-commerce has brought a significant rise in the importance of customer reviews. There are hundreds of review sites online and massive amounts of reviews for every product. Customers have changed their way of shopping and according to a recent survey, 70 percent of customers say that they use rating filters to filter out low rated items in their searches.

The ability to successfully decide whether a review will be helpful to other customers and thus give the product more exposure is vital to companies that support these reviews, companies like Google, Amazon and Yelp!.

There are two main methods to approach this problem. The first one is based on review text content analysis and uses the principles of natural language process (the NLP method). This method lacks the insights that can be drawn from the relationship between costumers and items. The second one is based on recommender systems, specifically on collaborative filtering, and focuses on the reviewer's point of view.

Conceptual background of domain problem

Rating prediction is a well-known recommendation task aiming to predict a user's rating for those items which were not rated yet by her. Predictions are computed from users' explicit feedback, i.e., their ratings provided on some items in the past. Another type of feedback are user reviews provided on items which implicitly express users' opinions on items. Recent studies indicate that opinions inferred from users' reviews on items are strong predictors of user's implicit feedback or even ratings and thus, should be utilized in computation. As far as we know, all the recent works on recommendation techniques utilizing opinions inferred from users' reviews are either focused on the item recommendation task or use only the opinion information, completely leaving users' ratings out of consideration. The approach proposed in this paper is filling this gap, providing a simple, personalized and scalable rating prediction framework utilizing both ratings provided by users and opinions inferred from their reviews.

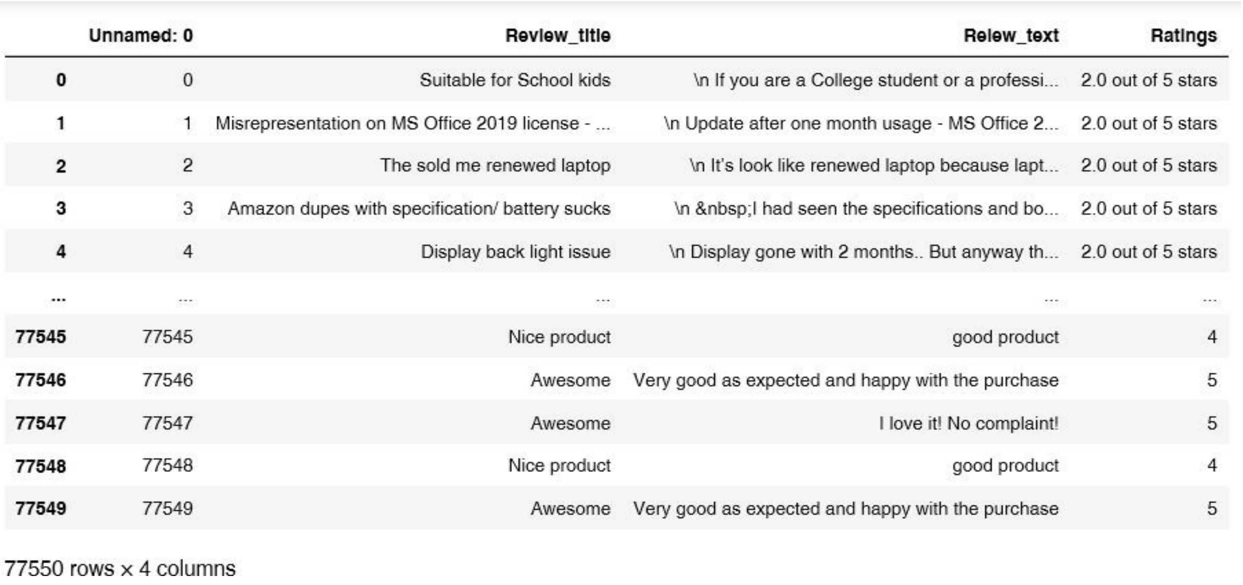
Experimental results provided on dataset containing user ratings and reviews from the real-world Amazon and Flipkart Product Review Data show the effectiveness of the proposed framework.

Analytical Problem Framing

– Mathematical/Analytical modeling of the problem

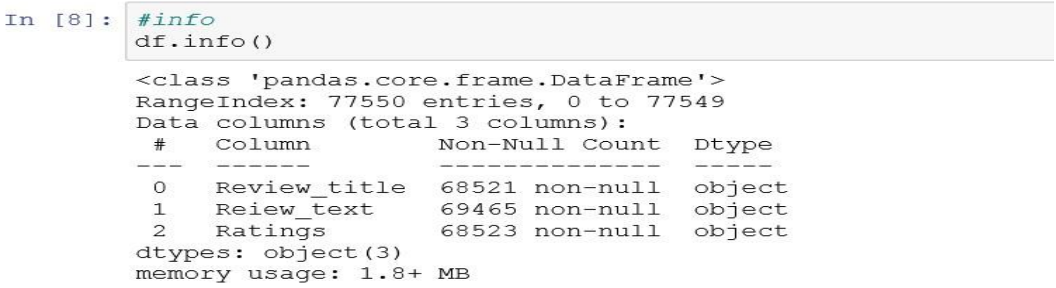
As per the client’s requirement for this rating prediction project I have scraped reviews and ratings from well-known e-commerce sites. This is then saved into .csv format. Also, I have shared the script for web scraping into the github repository.

Then loaded this data into a data frame and did some of the important natural language processing steps and gone through several EDA steps to analyze the data. After all the necessary steps I have built a NLP ML model to predict the ratings.



	Unnamed: 0	Review_title	Reiew_text	Ratings
0	0	Suitable for School kids	\n If you are a College student or a professi...	2.0 out of 5 stars
1	1	Misrepresentation on MS Office 2019 license - ...	\n Update after one month usage - MS Office 2...	2.0 out of 5 stars
2	2	The sold me renewed laptop	\n It's look like renewed laptop because lapt...	2.0 out of 5 stars
3	3	Amazon dupes with specification/ battery sucks	\n I had seen the specifications and bo...	2.0 out of 5 stars
4	4	Display back light issue	\n Display gone with 2 months.. But anyway th...	2.0 out of 5 stars
...	...	...	...	...
77545	77545	Nice product	good product	4
77546	77546	Awesome	Very good as expected and happy with the purchase	5
77547	77547	Awesome	I love it! No complaint!	5
77548	77548	Nice product	good product	4
77549	77549	Awesome	Very good as expected and happy with the purchase	5

77550 rows × 4 columns



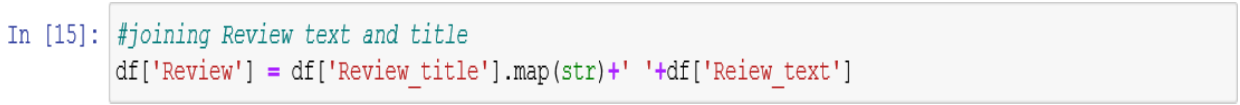
```
In [8]: #info
        df.info()

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 77550 entries, 0 to 77549
        Data columns (total 3 columns):
         #   Column        Non-Null Count  Dtype
        ---  ------        --------------  -----
         0   Review_title  68521 non-null  object
         1   Reiew_text    69465 non-null  object
         2   Ratings       68523 non-null  object
        dtypes: object(3)
        memory usage: 1.8+ MB
```

Looking at above both figures we can see that our data set contains 77550 different rows and 4 columns among which I have removed unwanted column (Unnamed:0). And for this project Ratings is our target column. There are some missing values in our dataset which have been removed from the dataset.

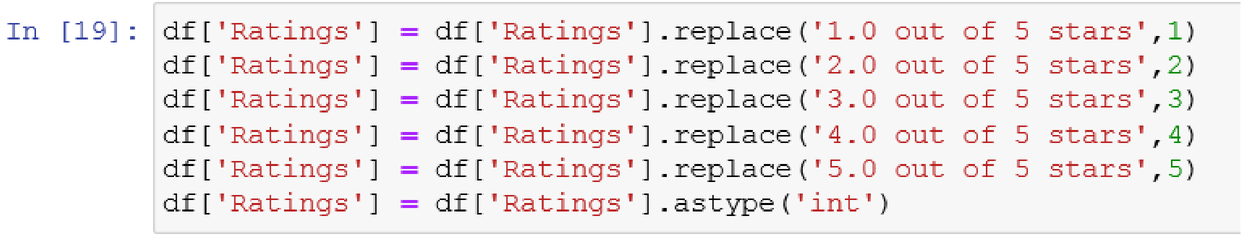
Data Processing:

At first, I have joined both columns Review_title and Review_text into a new column as Review.



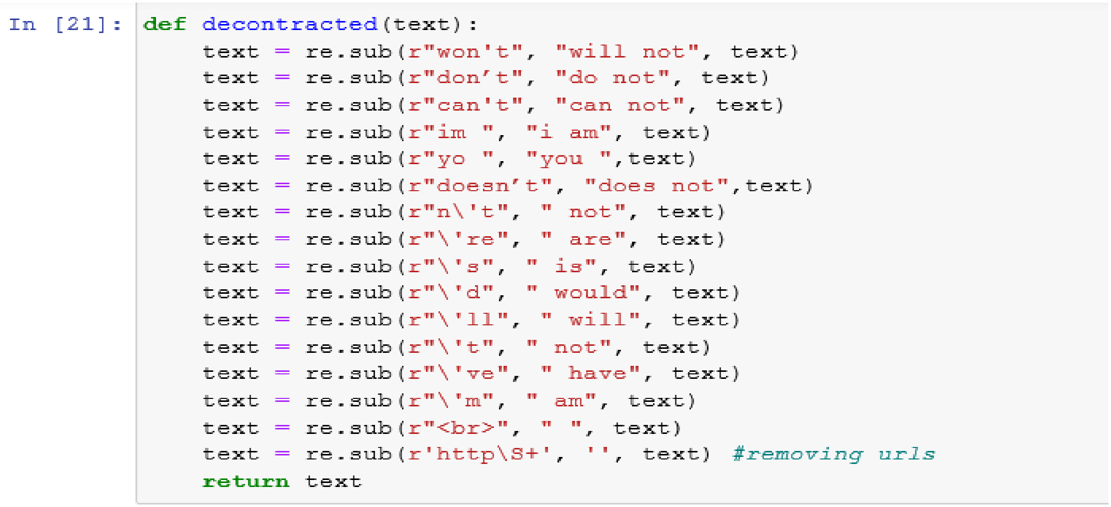
```
In [15]: #joining Review text and title
         df['Review'] = df['Review_title'].map(str)+' '+df['Reiew_text']
```

Then all the entries from Ratings columns have been converted to respective integer values

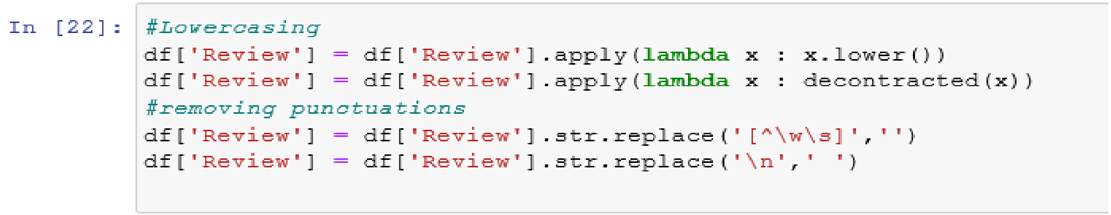


```
In [19]: df['Ratings'] = df['Ratings'].replace('1.0 out of 5 stars',1)
         df['Ratings'] = df['Ratings'].replace('2.0 out of 5 stars',2)
         df['Ratings'] = df['Ratings'].replace('3.0 out of 5 stars',3)
         df['Ratings'] = df['Ratings'].replace('4.0 out of 5 stars',4)
         df['Ratings'] = df['Ratings'].replace('5.0 out of 5 stars',5)
         df['Ratings'] = df['Ratings'].astype('int')
```

Text processing



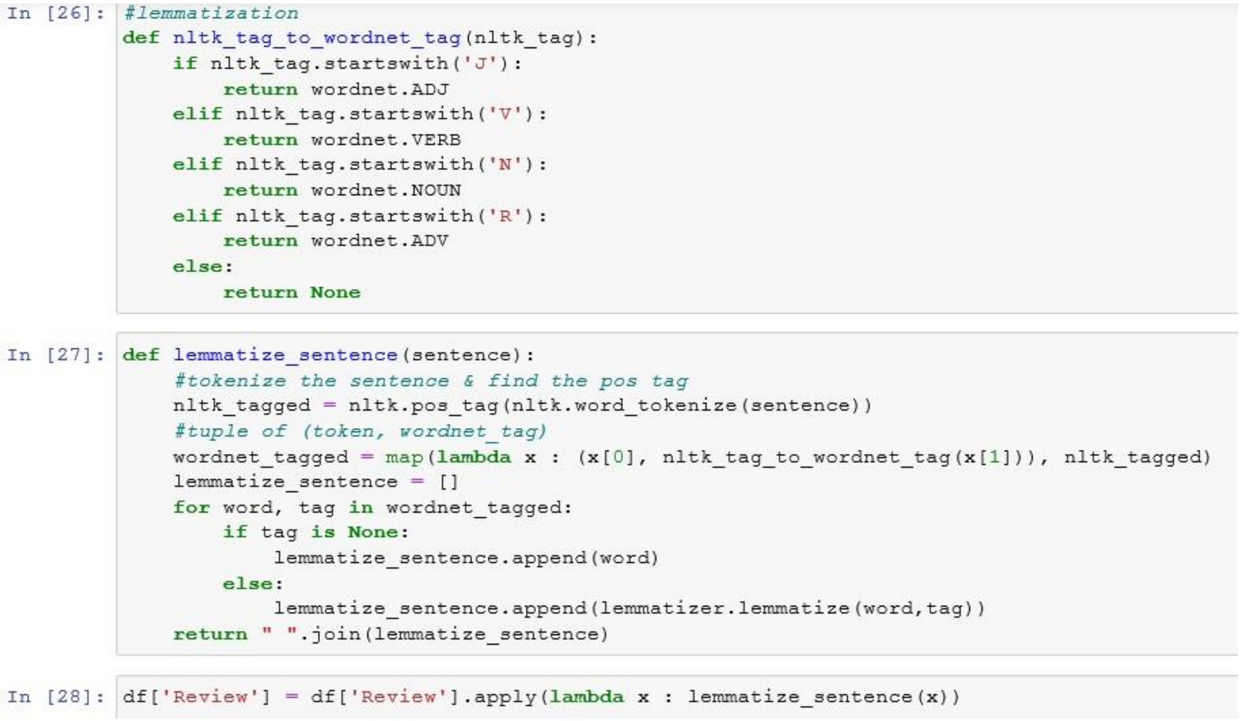
```
In [21]: def decontracted(text):
             text = re.sub(r"won't", "will not", text)
             text = re.sub(r"don’t", "do not", text)
             text = re.sub(r"can't", "can not", text)
             text = re.sub(r"im ", "i am", text)
             text = re.sub(r"yo ", "you ",text)
             text = re.sub(r"doesn’t", "does not",text)
             text = re.sub(r"n\'t", " not", text)
             text = re.sub(r"\'re", " are", text)
             text = re.sub(r"\'s", " is", text)
             text = re.sub(r"\'d", " would", text)
             text = re.sub(r"\'ll", " will", text)
             text = re.sub(r"\'t", " not", text)
             text = re.sub(r"\'ve", " have", text)
             text = re.sub(r"\'m", " am", text)
             text = re.sub(r"<br>", " ", text)
             text = re.sub(r'http\S+', '', text) #removing urls
             return text
```



```
In [22]: #Lowercasing
         df['Review'] = df['Review'].apply(lambda x : x.lower())
         df['Review'] = df['Review'].apply(lambda x : decontracted(x))
         #removing punctuations
         df['Review'] = df['Review'].str.replace('[^\w\s]','')
         df['Review'] = df['Review'].str.replace('\n',' ')
```

For text processing I have defined a function to replace some of the words with proper words. All text is converted to lowercase and removed different punctuations from the text of Review column.

Lemmatization: Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item. Lemmatization is similar to stemming but it brings context to the words.



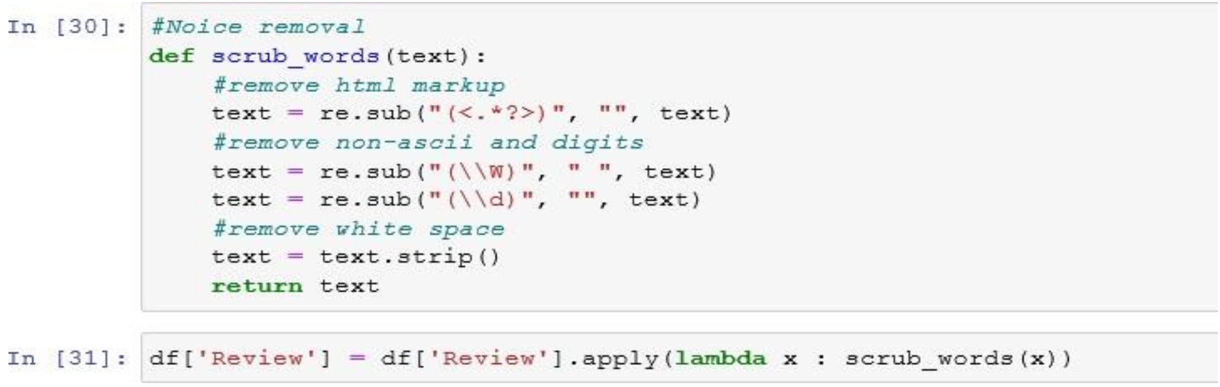
```
In [26]: #lemmatization
         def nltk_tag_to_wordnet_tag(nltk_tag):
             if nltk_tag.startswith('J'):
                 return wordnet.ADJ
             elif nltk_tag.startswith('V'):
                 return wordnet.VERB
             elif nltk_tag.startswith('N'):
                 return wordnet.NOUN
             elif nltk_tag.startswith('R'):
                 return wordnet.ADV
             else:
                 return None

In [27]: def lemmatize_sentence(sentence):
             #tokenize the sentence & find the pos tag
             nltk_tagged = nltk.pos_tag(nltk.word_tokenize(sentence))
             #tuple of (token, wordnet_tag)
             wordnet_tagged = map(lambda x : (x[0], nltk_tag_to_wordnet_tag(x[1])), nltk_tagged)
             lemmatize_sentence = []
             for word, tag in wordnet_tagged:
                 if tag is None:
                     lemmatize_sentence.append(word)
                 else:
                     lemmatize_sentence.append(lemmatizer.lemmatize(word,tag))
             return " ".join(lemmatize_sentence)

In [28]: df['Review'] = df['Review'].apply(lambda x : lemmatize_sentence(x))
```

For lemmatizing the text, I have defined these two functions first will give the wordnet tag for the nltk_tagged word then with respect to this wordnet tag lemmatization of each word is done.

Text Normalization – Standardization



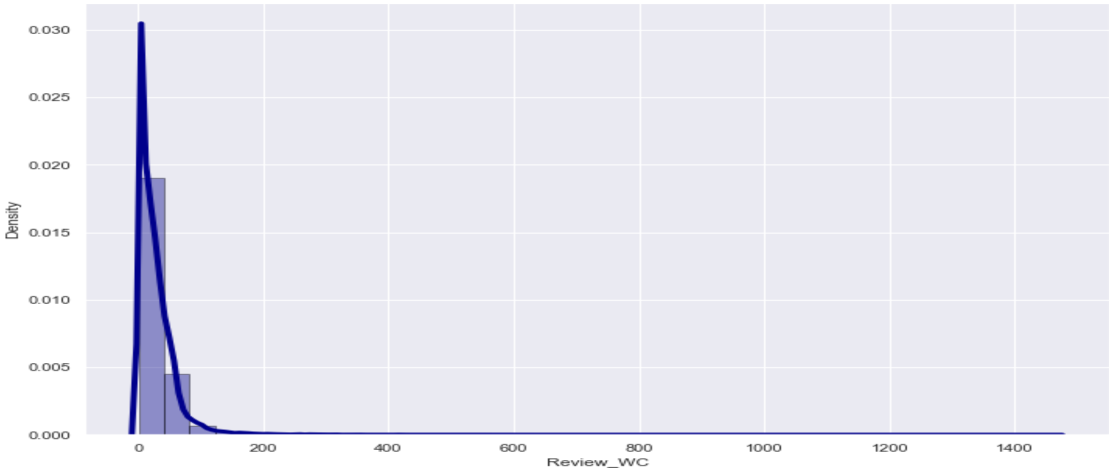
```
In [30]: #Noice removal
         def scrub_words(text):
             #remove html markup
             text = re.sub("(<.*?>)", "", text)
             #remove non-ascii and digits
             text = re.sub("(\\W)", " ", text)
             text = re.sub("(\\d)", "", text)
             #remove white space
             text = text.strip()
             return text

In [31]: df['Review'] = df['Review'].apply(lambda x : scrub_words(x))
```

Finally for standardizing our test and removing numbers from it I have defined a function as scrub_words as shown in above code and applied to the review column.

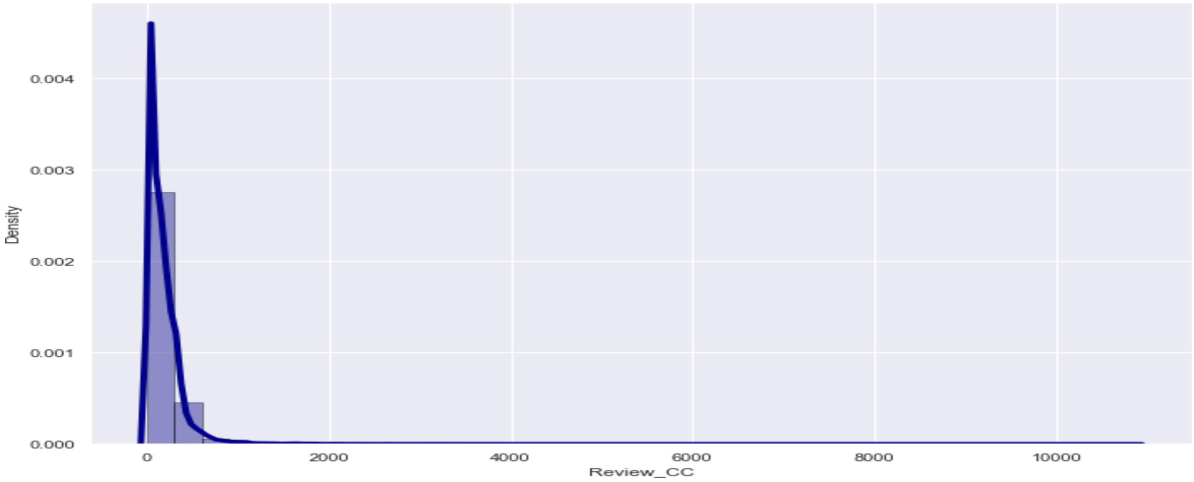
Exploratory Data Analysis:

Word_count of review



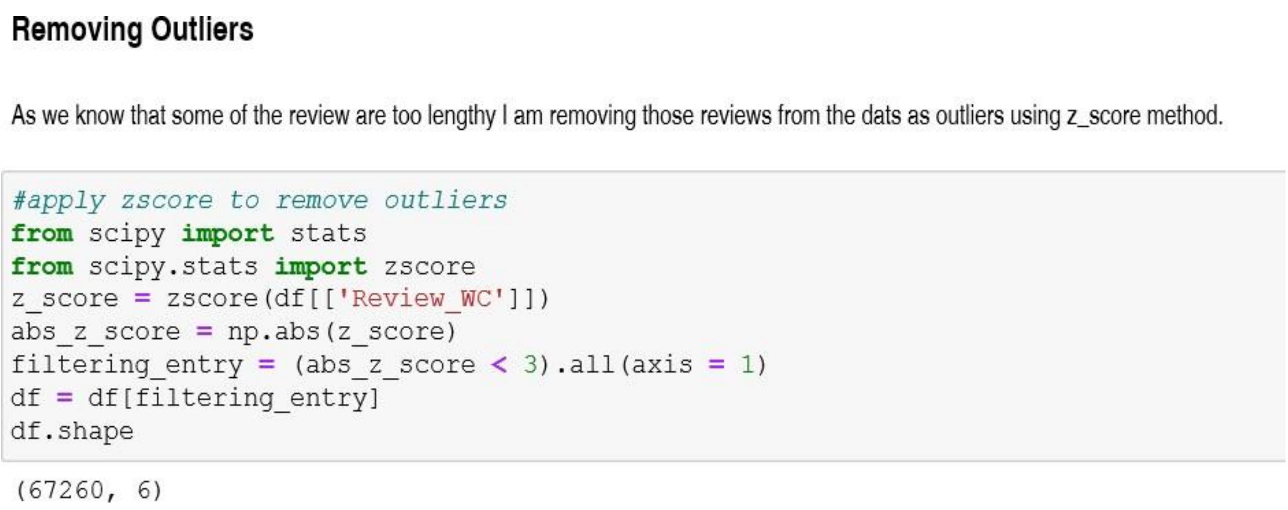
Above figure shows the number of words from each review text. Looking at this histogram we can conclude that most of the review text is in the range of 0 to 200 of words. Rest reviews can be considered as outliers in our data.

Character count of review



The plot for character count is almost similar to the plot of word count. We can see that most of the reviews are in the range of 0 to 1500 numbers of characters.

Looking at these plots I have decided to remove the data with too long reviews by considering them as outliers.



Removing Outliers

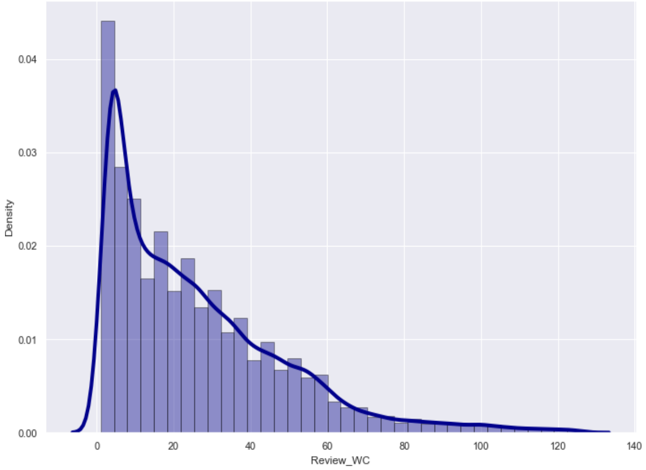
As we know that some of the review are too lengthy I am removing those reviews from the dats as outliers using z_score method.

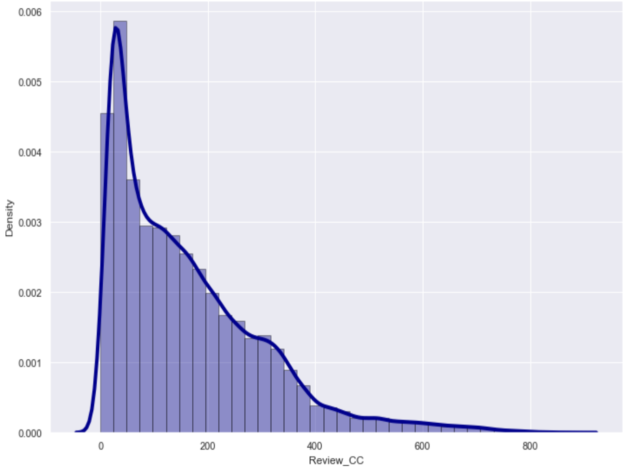
```
#apply zscore to remove outliers
from scipy import stats
from scipy.stats import zscore
z_score = zscore(df[['Review_WC']])
abs_z_score = np.abs(z_score)
filtering_entry = (abs_z_score < 3).all(axis = 1)
df = df[filtering_entry]
df.shape
```

```
(67260, 6)
```

And by removing these outliers I am not loosing much of the data so it is good to remove those entries for better results.

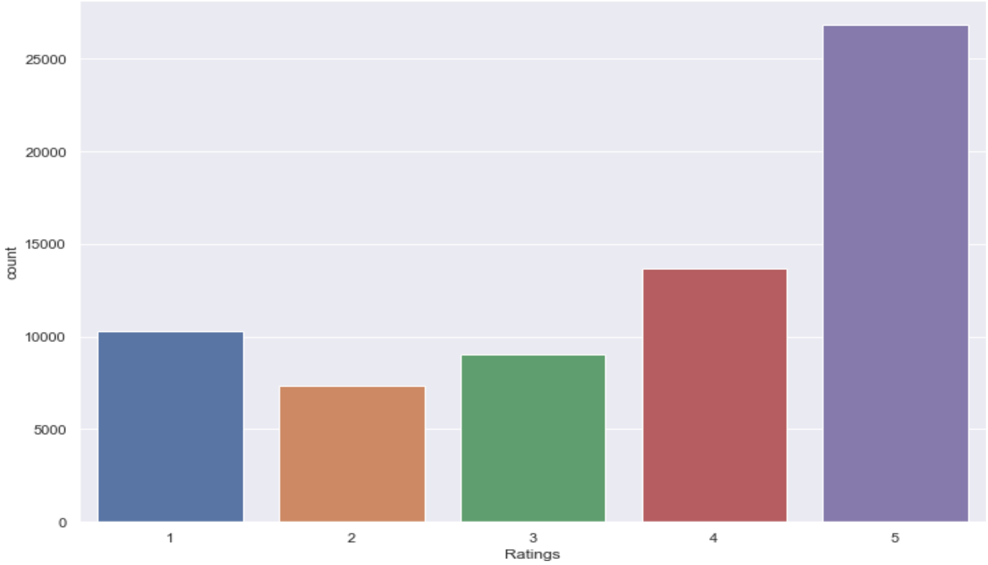
Plotting histograms for word count and character counts again after removing outliers



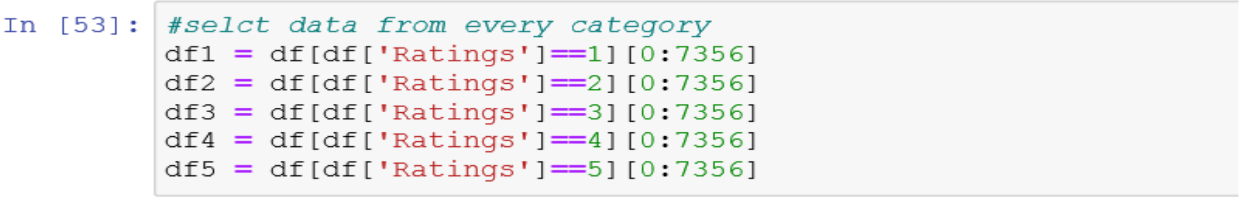


After plotting histograms for word counts and character counts after removing outliers we can see now we are with good range of number of words and characters.

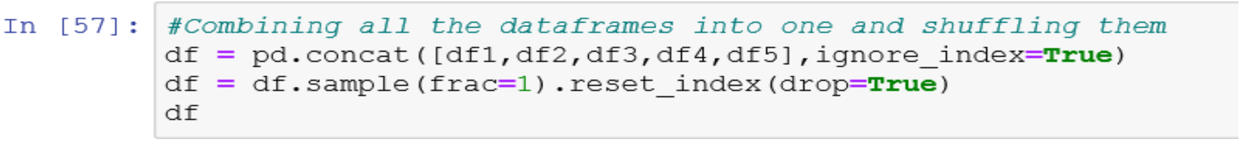
Ratings (Target Variable):



The above figure is representing count-plot for our target variable that is “Ratings”. Looking at this plot we can say that there are more numbers of reviews rated as 5 stars than others. And the reviews which are rated as 2 stars are very less in numbers when compared to others. This will cause the problem of imbalance for our model. So I have decided to select equal number of reviews from each class. I have observed that there are 7356 numbers of reviews rated as 2 stars which are least among all. So I am selecting 7356 numbers of reviews from each class as input for our model to eliminate the problem of imbalance from our data set.

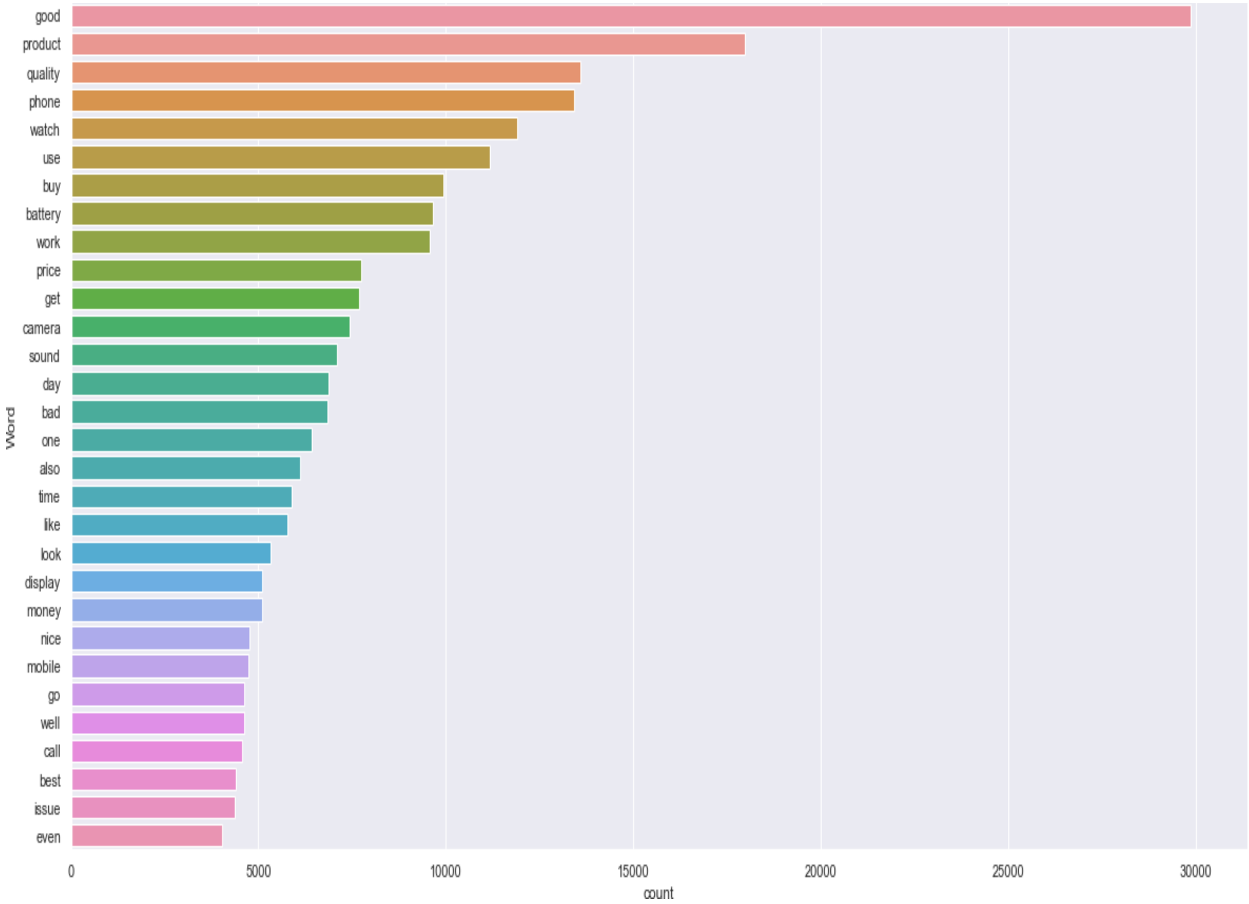


```
In [53]: #selct data from every category
         df1 = df[df['Ratings']==1][0:7356]
         df2 = df[df['Ratings']==2][0:7356]
         df3 = df[df['Ratings']==3][0:7356]
         df4 = df[df['Ratings']==4][0:7356]
         df5 = df[df['Ratings']==5][0:7356]
```



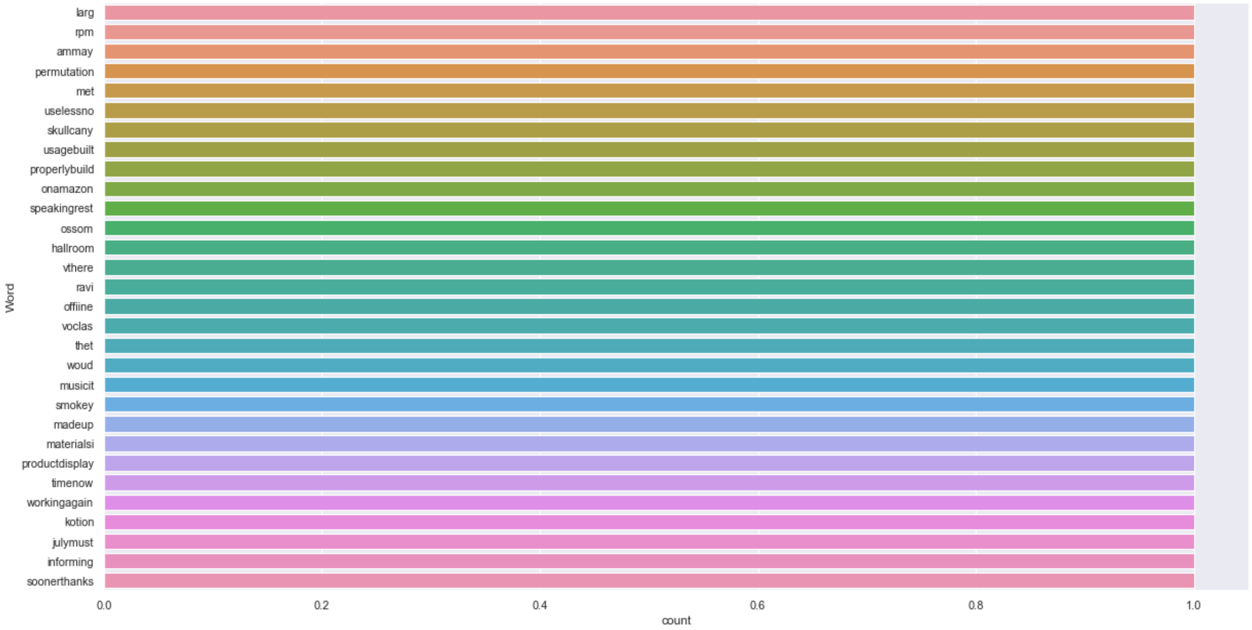
```
In [57]: #Combining all the dataframes into one and shuffling them
         df = pd.concat([df1,df2,df3,df4,df5],ignore_index=True)
         df = df.sample(frac=1).reset_index(drop=True)
         df
```

Top 30 most frequently occurring words:



The above bar plot is showing top 30 most frequently occurring words in our reviews. We can see the words like ‘good’, ‘product’, ‘quality’ etc. are occurring more frequently.

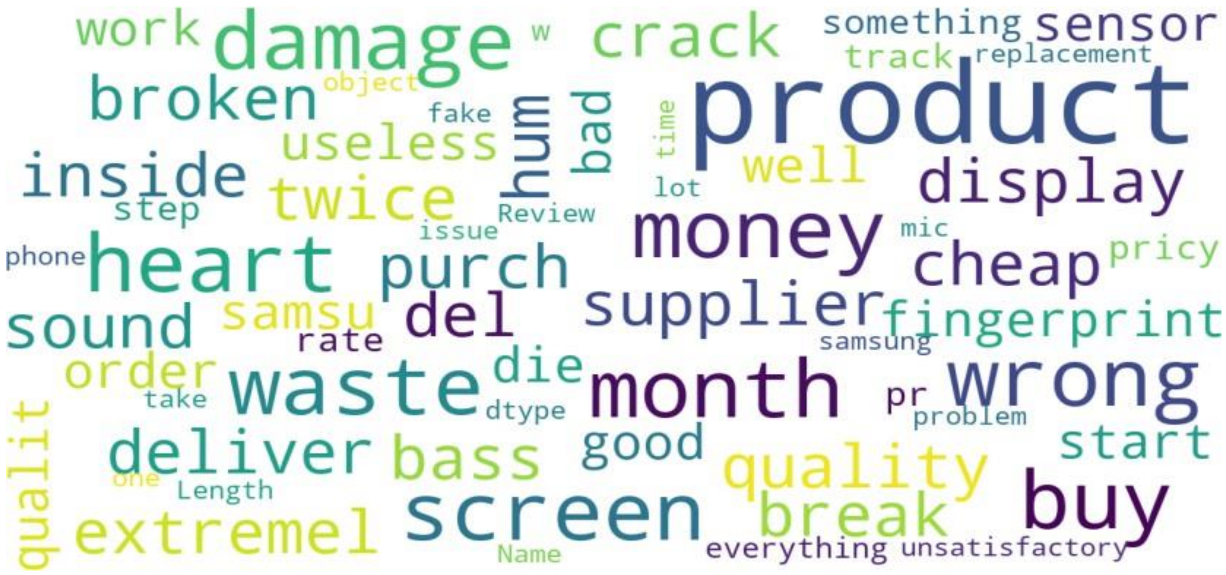
Top 30 Rarely occurring words:



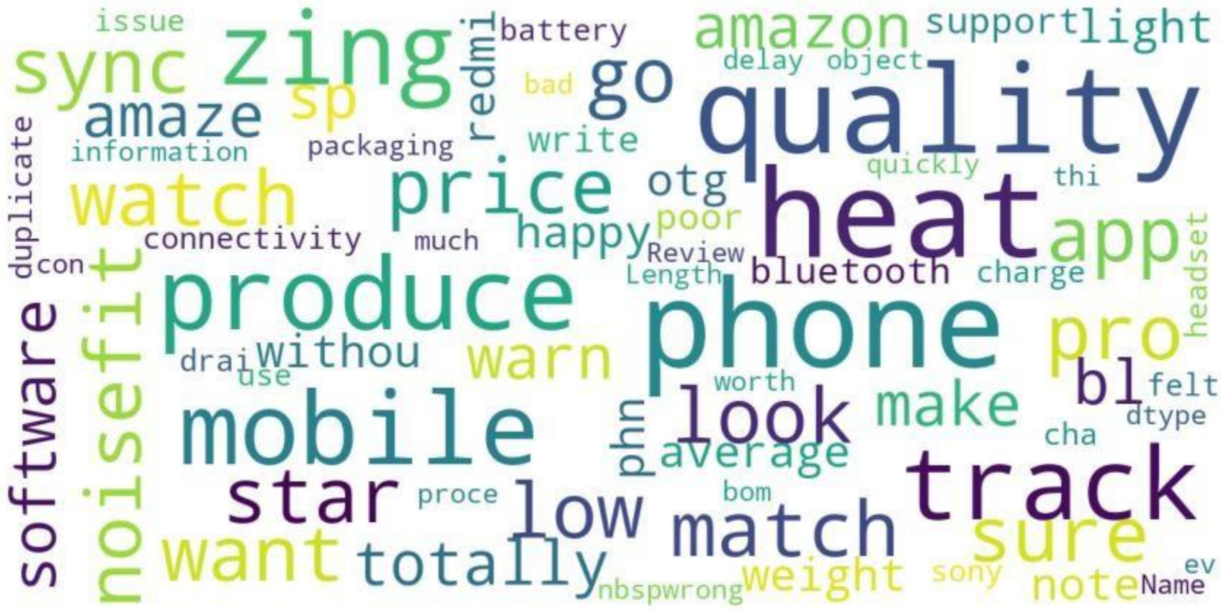
Above figure is representing bar plot for top 30 rarely occurring words. Many of which are spelled incorrectly that's why these are occurring only once.

Now using word cloud, I have visualized the frequently occurring words with respect to particular rating:

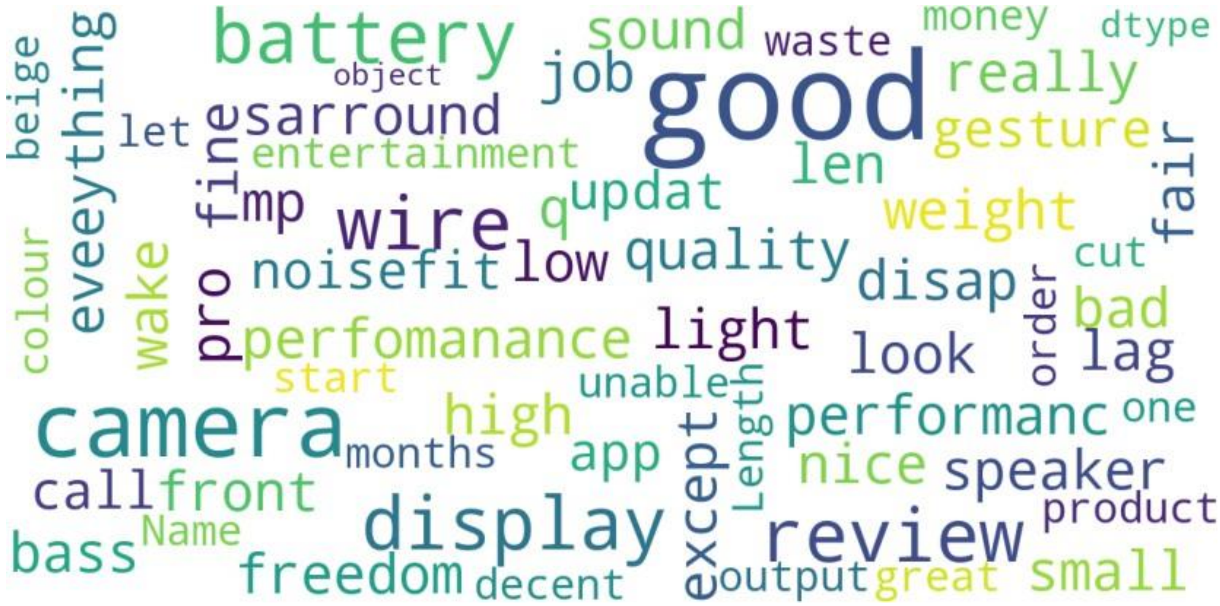
Words for rating = 1:



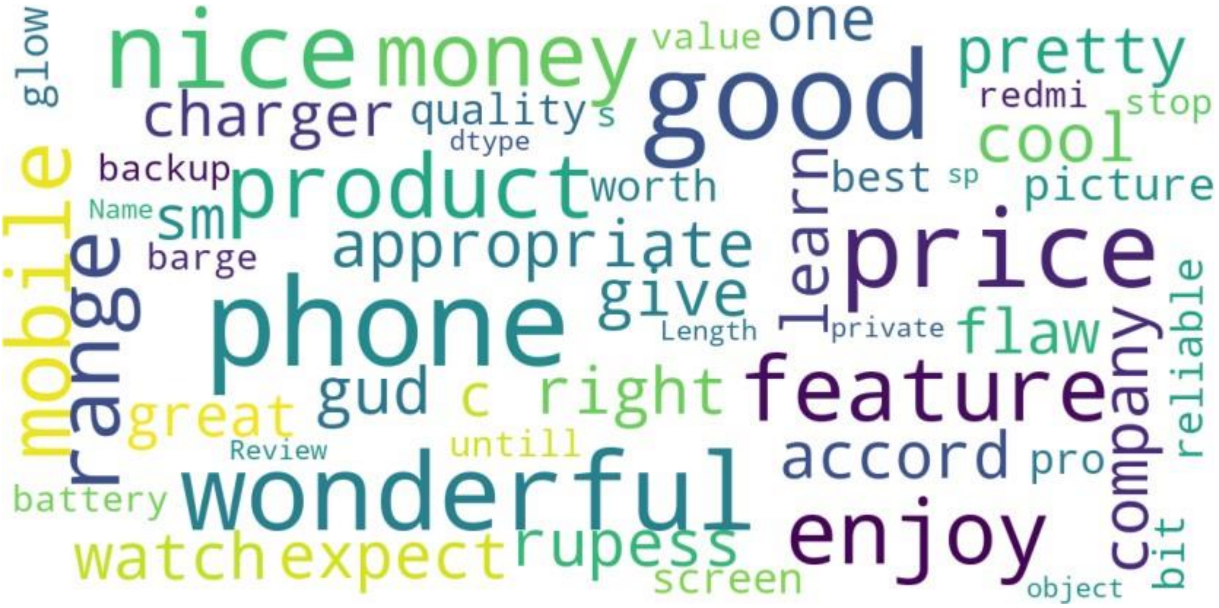
Words for rating = 2:



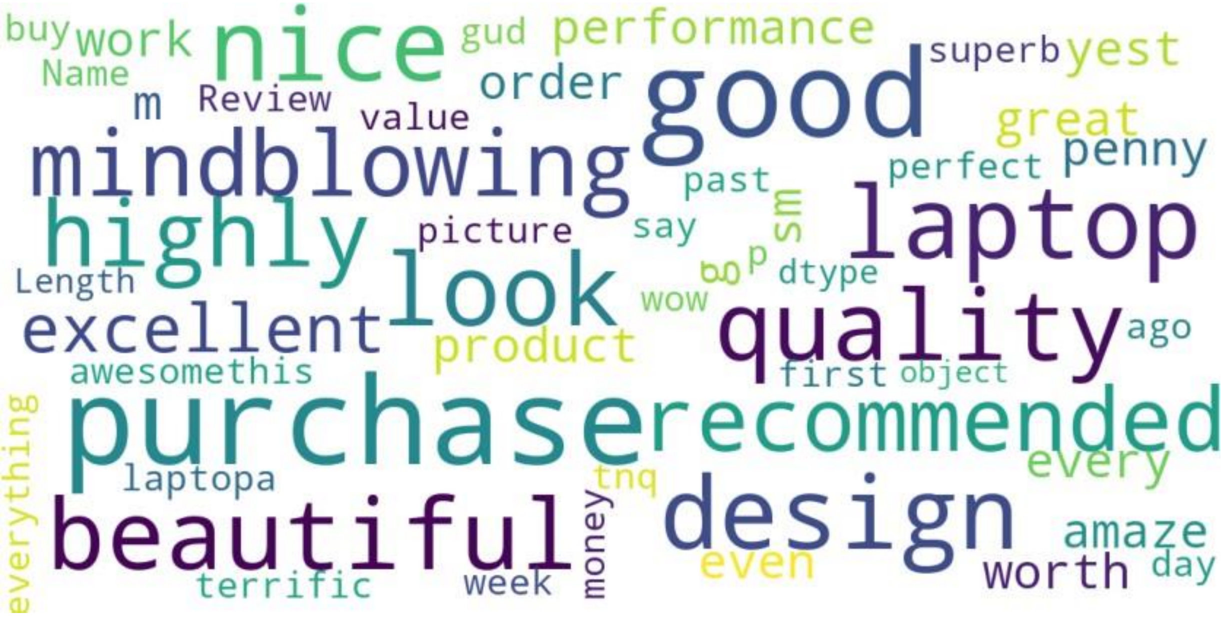
Words for rating = 3:



Words for rating = 4:



Words for rating = 5:



Model Development and Evaluation

As for this project we are going to predict the ratings based on the reviews given by customers this will be a classification task. For this purpose, I have collected data from amazon and flipkart.

Going through various NLP steps and analyzing the data using different EDA steps I have built several models using **Tfidf vectorizer.** Among all the different algorithms i have used LinearSVC is giving highest accuracy. Other algorithms like LGBMClassifier, XGBClassifier and RandomForestClassifier are also giving good accuracies. Considering all f1_scores, recall and precision for different classes and cross validation score I can say the LinearSVC is giving better performance than others. So, I am selecting it as best suitable algorithm for our final model.
I have used following algorithms and evaluated them

- RandomForestClassifier
- LinearSVC
- LogisticRegression
- MultinomialNB
- XGBClassifier
- BernoulliNB
- LightGBMClassifier
- SGDClassifier

- From all of these above models LinearSVC was giving me good performance.

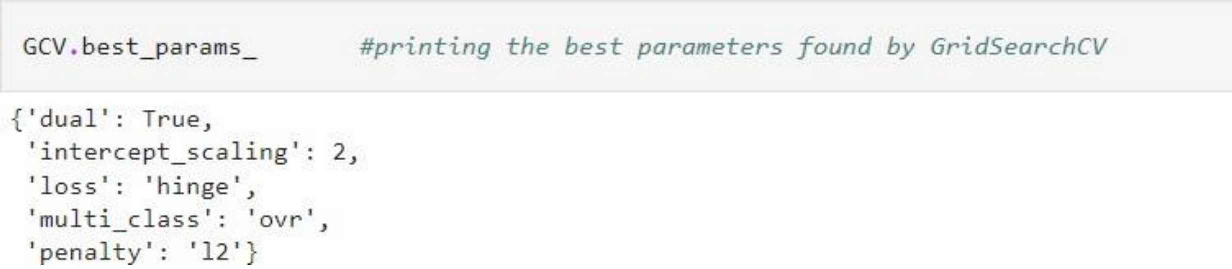
Key Metrics for success in solving problem under consideration

I have used the following metrics for evaluation:

- As this is classification problem, I am using accuracy score here.
- In this case I have checked for the confusion matrix which will give clear idea about true and false predictions.
- I have checked for classification report which gives overall performance metric of any algorithm with all f1_scores, precisions and recall scores.
- And Cross-validation score for checking the model performance for different folds.

Hyperparameter Tuning

I have done hyperparameter tuning for LinearSVC for the parameters like ‘penalty’, ‘loss’, ‘multi_class’, ‘intercept_scaling’, ‘dual’.



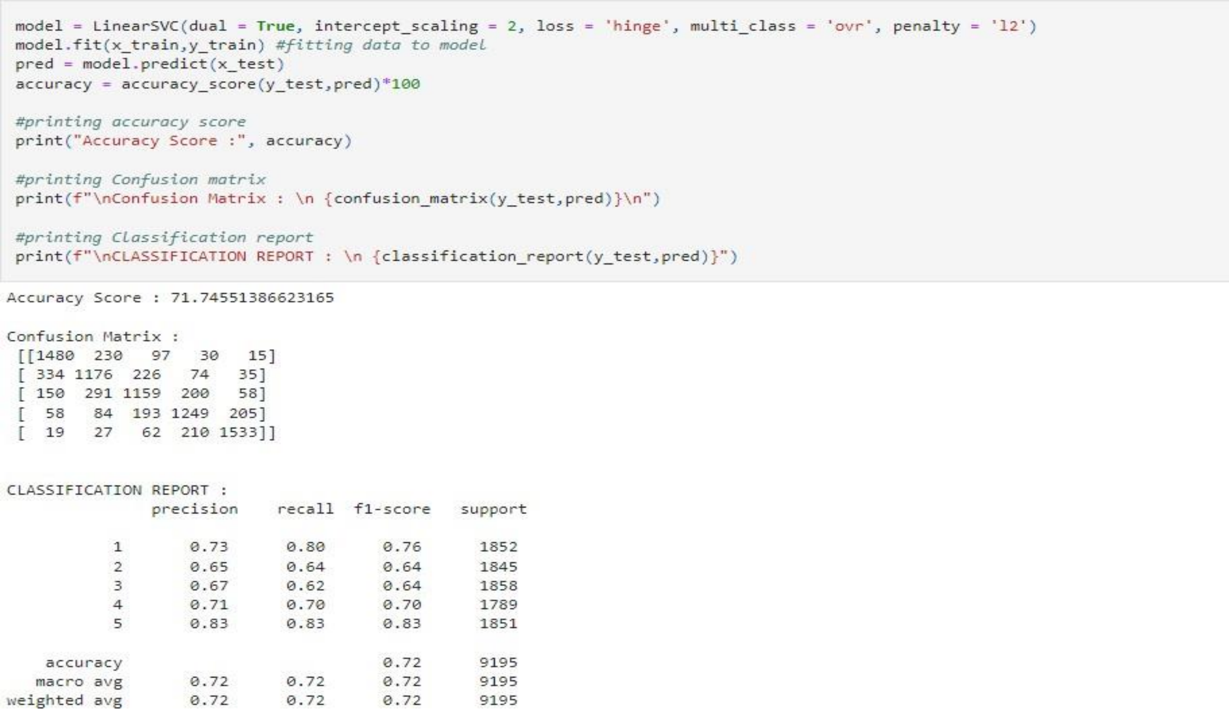
```
GCV.best_params_       #printing the best parameters found by GridSearchCV
```

```
{'dual': True,
 'intercept_scaling': 2,
 'loss': 'hinge',
 'multi_class': 'ovr',
 'penalty': 'l2'}
```

And after doing hyper-parameter tuning, I got above parameters as best suitable parameters for our final model.

I have tested my final model using these parameters and got better results compared to earlier results for my final model.

Final Model:



```
model = LinearSVC(dual = True, intercept_scaling = 2, loss = 'hinge', multi_class = 'ovr', penalty = 'l2')
model.fit(x_train,y_train) #fitting data to model
pred = model.predict(x_test)
accuracy = accuracy_score(y_test,pred)*100

#printing accuracy score
print("Accuracy Score :", accuracy)

#printing Confusion matrix
print(f"\nConfusion Matrix : \n {confusion_matrix(y_test,pred)}\n")

#printing Classification report
print(f"\nCLASSIFICATION REPORT : \n {classification_report(y_test,pred)}")
```

```
Accuracy Score : 71.74551386623165

Confusion Matrix :
 [[1480  230   97   30   15]
 [ 334 1176  226   74   35]
 [ 150  291 1159  200   58]
 [  58   84  193 1249  205]
 [  19   27   62  210 1533]]


CLASSIFICATION REPORT :
               precision    recall  f1-score   support

           1       0.73      0.80      0.76      1852
           2       0.65      0.64      0.64      1845
           3       0.67      0.62      0.64      1858
           4       0.71      0.70      0.70      1789
           5       0.83      0.83      0.83      1851

    accuracy                           0.72      9195
   macro avg       0.72      0.72      0.72      9195
weighted avg       0.72      0.72      0.72      9195
```

Great; after doing hyperparameter tuning we have got improved accuracy score for our final model.

Conclusion:

Key findings of the study

In this project I have collected data of reviews and ratings for different products from amazon.in and flipkart.com. Then I have done different text processing for reviews column and chose equal number of text from each rating class to eliminate problem of imbalance. By doing different EDA steps I have analyzed the text. We have checked frequently occurring words in our data as well as rarely occurring words. After all these steps I have built function to train and test different algorithms and using various evaluation metrics I have selected LinearSVC for our final model.

Finally by doing hyperparameter tuning we got optimum parameters for our final model. And finally we got improved accuracy score for our final model.

Limitations of this work and scope for the future work

As we know the content of text in reviews is totally depends on the reviewer and they may rate differently which is totally depends on that particular person. So, it is difficult to predict ratings based on the reviews with higher accuracies.

Still, we can improve our accuracy by fetching more data and by doing extensive hyperparameter tuning.