RATINGS PREDICTION

Submitted by:

Jyuthika Mankar

ACKNOWLEDGMENT

I am thankful to Flip Robo Technologies, situated in Bengaluru for giving me this opportunity to work as an Intern in their company. The experience which I am accumulating with the kind of projects Flip Robo Technologies is providing is nurturing. “Ratings Prediction” is one of the projects given to me for Web Scrapping, Machine Learning and Data Analysis. The dataset has been scrapped from an ecommerce website and working on that data has been proved to be a learning experience in every way. I would like to thank my SME Ms Sapna Verma for being considerate always to the interns and resolving the issues whenever they arose regarding the current project.

INTRODUCTION

- Business Problem Framing

  One of the clients 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. 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.

- Conceptual Background of the 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 customers. 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 project 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.

- Review of Literature
  - What is rating? Rating is a classification or ranking of someone or something based on a comparative assessment of their

quality, standard or overall performance. This project is more about exploration, feature engineering and classification that can be done on this data. Since we scrape huge amount of data that includes five stars rating, we can do better data exploration and derive some interesting features using the available columns. We can categorize the ratings as: 1, 2, 3, 4 and 5 stars respectively.

- The goal of this project is to build an application which can predict the rating by seeing the review. In the long term, this would allow people to better explain and review their purchase with each other in this increasingly digital world.
- All the external resources that were used in creating this project are listed below:
  1)https://www.google.com/
  2) https://scikit-learn.org/stable/user_guide.html
  3) https://github.com/
  4) https://www.kaggle.com/
  7) https://medium.com/
  8) https://towardsdatascience.com/
  9) https://www.nltk.org/api/nltk.html
  10) https://www.nltk.org/data.html

- Motivation for the Problem Undertaken
  - Every day we come across various products in our lives, on the digital medium we swipe across hundreds of product choices under one category. It will be tedious for the customer to make selection. Here comes 'reviews' where customers who have already got that product leave a rating after using them and brief their experience by giving reviews. As we know ratings can be easily sorted and judged whether a product is good or bad. But when it comes to sentence reviews, we need to read through every line to make sure the review conveys a positive or negative sense. In the era of artificial intelligence, things like that have got easy with the Natural Language Processing (NLP) technology. Therefore, it is important to minimize the number

of false positives our model produces, to encourage all constructive conversation.

- Our model also provides beneficence for the platform hosts as it replaces the need to manually moderate discussions, saving time and resources. Employing a machine learning model to predict ratings promotes easier way to distinguish between products qualities, costs and many other features. Many product reviews are not accompanied by a scale rating system, consisting only of a textual evaluation. In this case, it becomes daunting and time-consuming to compare different products in order to eventually make a choice between them. Therefore, models able to predict the user rating from the text review are critically important. Getting an overall sense of a textual review could in turn improve consumer experience.

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 file. 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 analyse the data. After all the necessary steps I have built an NLP ML model to predict the ratings. In our scrapped dataset our target variable column "Ratings" is a categorical variable i.e., it can be classified as 1, 2, 3, 4 and 5 stars. Therefore, we will be handling this modelling problem as a multiclass classification project.

  1. Data was cleaned by removing punctuations, white spaces, and special characters.
  2. Using the stopwords package, all stop words were removed along with some other values which occurred often but did not have any meaning.

3. Tf-idf vectoriser was used to convert text to vectors and weights were allocated for each text.
4. Metrics like accuracy, and f1 score was used to evaluate the model's efficiency.
5. K-Fold Cross Validation Score was used to identify if the model was overfitting or underfitting.

- Data Sources and their formats

  Reviews and ratings for various technical products were collected from an ecommerce website. The scrapped data script was written in Python programming language using the Selenium package. The reviews were stored in csv as well as format. I scrapped reviews and ratings of 9 products from Flipkart I got 21444 ratings and reviews.

```
# Creating a dataframe
data = list(zip(title,review_text,ratings))
df = pd.DataFrame(data, columns = ["Review_title","Review_text","Ratings"])
df
```

	Review_title	Review_text	Ratings
0	Fabulous!	After one month of usage for office work and m...	5
1	Worth every penny	Laptop is worth the price. It is fast and smoo...	5
2	Great product	I got it for 54k\nFor this price\nBuild Qualit...	5
3	Great product	One of the best laptop in given prize segment....	5
4	Awesome	Value for money just go for it✨	5
...	...	...	...
21439	Brilliant	Very nice	5
21440	Excellent	Excellent Resolution\nGood working\nVery very ...	5
21441	Brilliant	Really good monitor. Very effective and useful...	5
21442	Good choice	Actually its not a 24 inch display its just 21...	4
21443	Super!	Nice products	5

21444 rows × 3 columns

- Data Preprocessing Done
  1. First, (after importing the needed libraires) I loaded the .csv file wherein the data was stored
  2. Data information and shape was found out. The data set had all objective columns except the Ratings column which was integer.

The dataset had 21444 rows and 4 columns namely Unnamed:0, Review_title, Review_text and Ratings.

```
#DataSet Information:
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21444 entries, 0 to 21443
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Unnamed: 0    21444 non-null  int64 
 1   Review_title  21444 non-null  object
 2   Review_text   21444 non-null  object
 3   Ratings       21444 non-null  int64 
dtypes: int64(2), object(2)
memory usage: 670.2+ KB
```

```
df.shape
```

```
(21444, 4)
```

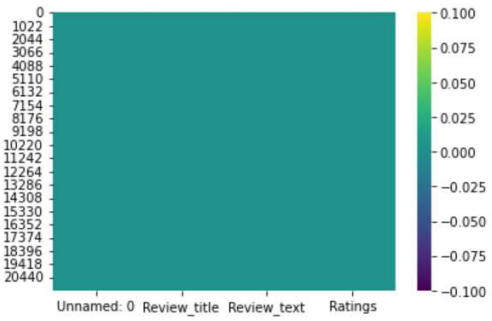
3. Null values were checked and it was seen that there were no null values in the dataset.

```
df.isnull().sum()
```

```
Unnamed: 0      0
Review_title    0
Review_text     0
Ratings         0
dtype: int64
```

```
sns.heatmap(df.isnull(),cmap='viridis')
```

```
<AxesSubplot:>
```



4. The column “Unnamed:0” was dropped as it played no significance in the ML.
5. Data was cleaned by removing punctuations, white spaces, and special characters.

```
def clean_text(text):
    # lower text
    text = text.lower()
    text = re.sub("[^\w\s]", " ", text)
    # tokenize text and remove puncutation
    text = [word.strip(string.punctuation) for word in text.split(" ")]
    # remove words that contain numbers
    text = [word for word in text if not any(c.isdigit() for c in word)]
    #Remove leading and trailing whitespace
    #text=re.sub("[^\s+|\s+?$]"," ",text)
    # remove stop words
    stop = set(stopwords.words('english') + ['u', 'ü', 'ur', '4', '2', 'im', 'dont', 'doin', 'ure']
    text = [x for x in text if x not in stop]
    # remove empty tokens
    text = [t for t in text if len(t) > 0]
    # pos tag text
    pos_tags = pos_tag(text)
    # lemmatize text
    text = [WordNetLemmatizer().lemmatize(t[0], get_wordnet_pos(t[1])) for t in pos_tags]
    #text=stemmer.stem(text)
    # remove words with only two letter
    text = [t for t in text if len(t) > 2]
    # join all
    text = " ".join(text)
    return(text)
```

6. Using the stopwords package, all stop words were removed along with some other values which occurred often but did not have any meaning.
7. Lemmatization was used on all the texts or reviews.
8. Tf-idf vectoriser was used to convert text to vectors and weights were allocated for each text.

- Data Inputs- Logic- Output Relationships
  1. The data was cleaned and all stop words were removed.
  2. Word count and character count iof the cleaned review data was found out.

```
# Creating column for word counts in the review text
df['Clean_Review_WC'] = df['Clean_Review'].apply(lambda x: len(str(x).split(' ')))
df[['Clean_Review_WC', 'Clean_Review']].head(10)
```

	Clean_Review_WC	Clean_Review
0	39	fabulous one month usage office work multimedi...
1	19	worth every penny laptop worth price fast smoo...
2	38	great product get price\nbuild quality build s...
3	16	great product one best laptop give prize segme...
4	3	awesome value money
5	9	good quality product nice love good laptop sim...
6	7	wonderful good product time get stuck good
7	5	classy product nice product must
8	14	value money best laptop one thing battery give...
9	3	great product good

  3. EDA was performed by plotting the Word count, character count and the rating column.

4. Word clouds that helped to identify frequently occurring words for each rating was created.
5. The review column was converted into vectors and was used as an input.
6. Various classifiers like Support Vector Classifier, Bernoulli NB, Multinomial NB, Decision Tree Classifier, Random Forest Classifier, Bagging Classifier, Extra Trees Classifier and SGD Classifier were used to train the model.
7. Accuracy and F1 scores of all the classifiers were compared and at the end, SGD Classifier was predicting the ratings with an average accuracy of about 95% after hyperpara-tuning.

- Hardware and Software Requirements and Tools Used

Hardware: 8GB Ram, Core-i5,10th Gen

Software: Following libraries were used:

1.Pandas: To read the csv file, to convert the data into dataframe, for description and data type of data, to save the final output.

2. Matplotlib: To plot the graphs

3. Seaborn: To plot the graphs

4. Stopwords: To remove the stopwords from our corpus.

5.Word Cloud: To create a word cloud representing the highest frequency of texts in the corpus.

6.WordNetLemmatizer: To lemmatize the text data.

7. Outlier removal: Outliers were removed from the cleaned reviews.

8. TfidfVectoriser: To convert text into vectors.

9. train_test_split: To split dataset into training and testing dataset.

10. Various Classifiers: Classification models used to train our data.

11. Confusion_matrix: To evaluate the accuracy of a classification.

12. Classification_report: A report is presented with accuracy scores, f1scores, recall and precision scores as well

13. Joblib: To save the model.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  Reviews and ratings from e-commerce websites for various products were scrapped and stored in a .csv file. The data cleaning was performed on the dataset – checking null values, stopwords, punctuations and other non-essential characters were removed. The text data was lemmatized. Data was analysed and visualized using word clouds for each rating, to identify common words that appear in a review for a particular rating. The text data was then vectorised and given as input to various classification models. K-Fold Cross Validation Score was used to identify if the model was overfit or underfit. Performance metrics like Accuracy, F1 Score, confusion matrix was used to analyse the performance of the model. Finally, using the best model, predicted on the test data and saved the model for future use.

- Testing of Identified Approaches (Algorithms)

  The algorithms used for the training and testing were:

  - Support Vector Classifier
  - Bernoulli NB
  - Multinomial NB
  - Decision Tree Classifier
  - Random Forest Classifier
  - Bagging Classifier
  - Extra Trees Classifier
  - SGD Classifier

- Run and Evaluate selected models

```
# Defining the Classification Machine Learning Algorithms
LR = LogisticRegression(solver='lbfgs')
SVC = LinearSVC()
BNB = BernoulliNB()
MNB=MultinomialNB()
DTC=DecisionTreeClassifier()
RFC=RandomForestClassifier()
BGC=BaggingClassifier()
ETC=ExtraTreesClassifier()
SGD = SGDClassifier()
```

```
#Putting machine learning Models in a list so that it can be used for further evaluation in loop.
models=[]
models.append(('Logistic Regression()',LR))
models.append(('Linear SVC()',SVC))
models.append(('BernoulliNB()',BNB))
models.append(('MultinomialNB()',MNB))
models.append(('DecisionTreeClassifier',DTC))
models.append(('RandomForestClassifier',RFC))
models.append(('BaggingClassifier',BGC))
models.append(('ExtraTreesClassifier',ETC))
models.append(('SGDClassifier()',SGD))
```

Accuracy score at the best random state were found

```
#Function which will find best Random State and then calculate Maximum Accuracy Score corresponding to it and print accuracy score in one go.
def max_acc_score(clf,x,y):
    max_acc_score=0
    final_r_state=0
    for r_state in range(42,100):
        x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.30,random_state=r_state,stratify=y)
        clf.fit(x_train,y_train)
        y_pred=clf.predict(x_test)
        acc_score=accuracy_score(y_test,y_pred)
        if acc_score > max_acc_score:
            max_acc_score=acc_score
            final_r_state=r_state
    print('Max Accuracy Score corresponding to Random State ', final_r_state, 'is:', max_acc_score)
    print('\n')
    return final_r_state
```

```
#Lists to store model name, Learning score, Accuracy score, cross_val_score, Auc Roc score .
Model=[]
Score=[]
Acc_score=[]
CVScore=[]

#For Loop to Calculate Accuracy Score, Cross Val Score, Classification Report, Confusion Matrix

for name,model in models:
    print('----------------------',name,'----------------------------')
    Model.append(name)
    print(model)
#Now here I am calling a function which will calculate the max accuracy score for each model and return best random state.
    r_state=max_acc_score(model,x,y)
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=r_state,stratify=y)
    model.fit(x_train,y_train)
#Learning Score
    score=model.score(x_train,y_train)
    print('Learning Score : ',score)
    Score.append(score*100)
    y_pred=model.predict(x_test)
    acc_score=accuracy_score(y_test,y_pred)
    print('Accuracy Score : ',acc_score)
    Acc_score.append(acc_score*100)
#Final CV Score
    cv_score=cross_val_score(model,x,y,cv=10,scoring='accuracy').mean()
    print('Cross Val Score : ', cv_score)
    CVScore.append(cv_score*100)
#Classification report
    print('Classification Report:\n',classification_report(y_test,y_pred))
    print('Confusion Matrix:\n',confusion_matrix(y_test,y_pred))
    print('\n')
```

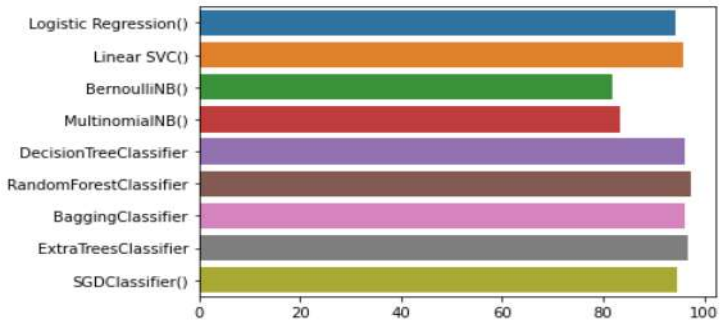
These Algorithms showed the best accuracy score.

```
---------------------- RandomForestClassifier ----------------------------
RandomForestClassifier()
Max Accuracy Score corresponding to Random State  87 is: 0.9737668865849827


Learning Score :  0.9940756698532381
Accuracy Score :  0.9728243795161797
Cross Val Score :  0.9444392082940624
Classification Report:
              precision    recall  f1-score   support

           1       0.98      1.00      0.99       634
           2       1.00      0.88      0.93       145
           3       0.98      0.91      0.95       370
           4       0.98      0.91      0.94      1250
           5       0.97      1.00      0.98      3967

    accuracy                           0.97      6366
   macro avg       0.98      0.94      0.96      6366
weighted avg       0.97      0.97      0.97      6366

Confusion Matrix:
 [[ 632    0    1    0    1]
 [  12  127    0    1    5]
 [   3    0  338   14   15]
 [   0    0    4 1140  106]
 [   0    0    2    9 3956]]

---------------------- ExtraTreesClassifier ----------------------------
ExtraTreesClassifier()
Max Accuracy Score corresponding to Random State  73 is: 0.9684260131950989


Learning Score :  0.9940083479197522
Accuracy Score :  0.9676405906377631
Cross Val Score :  0.9363807728557964
Classification Report:
              precision    recall  f1-score   support

           1       0.97      0.99      0.98       634
           2       1.00      0.86      0.93       145
           3       0.98      0.90      0.94       370
           4       0.98      0.89      0.94      1250
           5       0.96      1.00      0.98      3967

    accuracy                           0.97      6366
   macro avg       0.98      0.93      0.95      6366
weighted avg       0.97      0.97      0.97      6366

Confusion Matrix:
 [[ 627    0    3    1    3]
 [   8  125    1    3    8]
 [   6    0  333   14   17]
 [   3    0    3 1118  126]
 [   5    0    0    5 3957]]



---------------------- SGDClassifier() ----------------------------
SGDClassifier()
Max Accuracy Score corresponding to Random State  53 is: 0.9481621112158342


Learning Score :  0.9620304295139357
Accuracy Score :  0.9478479421928998
Cross Val Score :  0.924410933081998
Classification Report:
              precision    recall  f1-score   support

           1       0.97      0.99      0.98       634
           2       0.97      0.83      0.90       145
           3       0.93      0.84      0.89       370
           4       0.93      0.83      0.88      1250
           5       0.95      0.99      0.97      3967

    accuracy                           0.95      6366
   macro avg       0.95      0.90      0.92      6366
weighted avg       0.95      0.95      0.95      6366

Confusion Matrix:
 [[ 628    0    3    1    2]
 [   8  121    4    8    4]
 [   7    1  312   37   13]
 [   4    2   10 1042  192]
 [   0    1    6   29 3931]]
```

	Model	Learning Score	Accuracy Score	Cross Val Score
0	Logistic Regression()	95.563485	94.470625	92.525919
1	Linear SVC()	97.549482	95.915803	93.119698
2	BernoulliNB()	83.930254	81.950990	79.095193
3	MultinomialNB()	83.795611	83.443292	81.456173
4	DecisionTreeClassifier	99.400835	96.025762	91.606975
5	RandomForestClassifier	99.407567	97.282438	94.443921
6	BaggingClassifier	99.185405	96.135721	92.049953
7	ExtraTreesClassifier	99.400835	96.764059	93.638077
8	SGDClassifier()	96.203043	94.784794	92.441093

```
#visualisation of Accuracy Score
sns.barplot(y=Model,x=Acc_score)
```

```
<AxesSubplot:>
```



- Key Metrics for success in solving problem under consideration

  After getting the accuracy score and CV score, the best models were chosen and best parameters for these models were found by hyper paratuning.

  1. Random Forest
  2. Extra Trees
  3. SGD

  After hyper paratuning the Random Forest classifier, it gave the accuracy score of 86%.

```
rfc=RandomForestClassifier()
parameters={'n_estimators':[100,300,500],'max_depth':[15, 25, 30],'min_samples_leaf': [1,3,5], 'min_samples_split': [1,5,8]}
clf=GridSearchCV(rfc,parameters,cv=5)
clf.fit(x_train,y_train)
clf.best_params_
```

```
{'max_depth': 30,
 'min_samples_leaf': 1,
 'min_samples_split': 5,
 'n_estimators': 100}
```

```
#Applying the parameters we got after hyper parameter tuning
rfc=RandomForestClassifier(n_estimators=100,max_depth=30,min_samples_leaf=1, min_samples_split=5)
rfc.fit(x_train,y_train)
predrf=rfc.predict(x_test)
print(accuracy_score(y_test,predrf))
print(confusion_matrix(y_test,predrf))
print(classification_report(y_test,predrf))
```

```
0.8572101790763431
[[ 560    0    1    0   73]
 [  12   59    0    1   73]
 [   5    0  160    4  201]
 [   4    0    0  712  534]
 [   0    0    0    1 3966]]
              precision    recall  f1-score   support

           1       0.96      0.88      0.92       634
           2       1.00      0.41      0.58       145
           3       0.99      0.43      0.60       370
           4       0.99      0.57      0.72      1250
           5       0.82      1.00      0.90      3967

    accuracy                           0.86      6366
   macro avg       0.95      0.66      0.75      6366
weighted avg       0.88      0.86      0.84      6366
```

Extra Trees Classifier gave the accuracy score of 94% after hyper paratuning.

```
#Applying the parameters we got after hyper parameter tuning
etc=ExtraTreesClassifier(n_estimators=500,max_depth=15,max_features='sqrt',class_weight='balanced_subsample')
etc.fit(x_train,y_train)
predetc=etc.predict(x_test)
print(accuracy_score(y_test,predetc))
print(confusion_matrix(y_test,predetc))
print(classification_report(y_test,predetc))
```

```
0.9359095193213949
[[ 629    1    1    0    3]
 [  10  128    5    2    0]
 [   5    6  327   29    3]
 [  12    9   27 1125   77]
 [   5   10   13  190 3749]]
              precision    recall  f1-score   support

           1       0.95      0.99      0.97       634
           2       0.83      0.88      0.86       145
           3       0.88      0.88      0.88       370
           4       0.84      0.90      0.87      1250
           5       0.98      0.95      0.96      3967

    accuracy                           0.94      6366
   macro avg       0.89      0.92      0.91      6366
weighted avg       0.94      0.94      0.94      6366
```

SGD Classifier gives 95% accuracy after hyper paratuning.

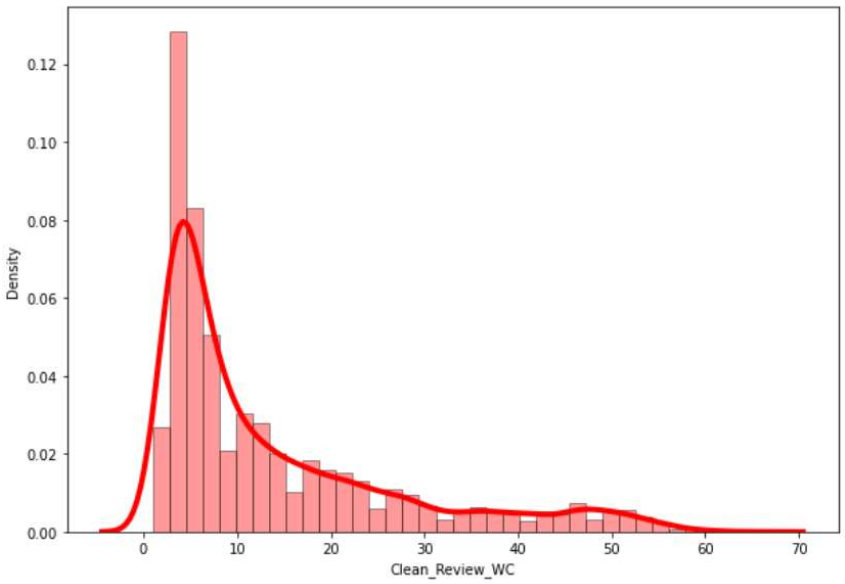
```
#Applying the parameters we got after hyper parameter tuning
sgd=SGDClassifier(loss='modified_huber',alpha=0.0001,fit_intercept=False, penalty='elasticnet')
sgd.fit(x_train,y_train)
predsgd=sgd.predict(x_test)
print(accuracy_score(y_test,predsgd))
print(confusion_matrix(y_test,predsgd))
print(classification_report(y_test,predsgd))
```

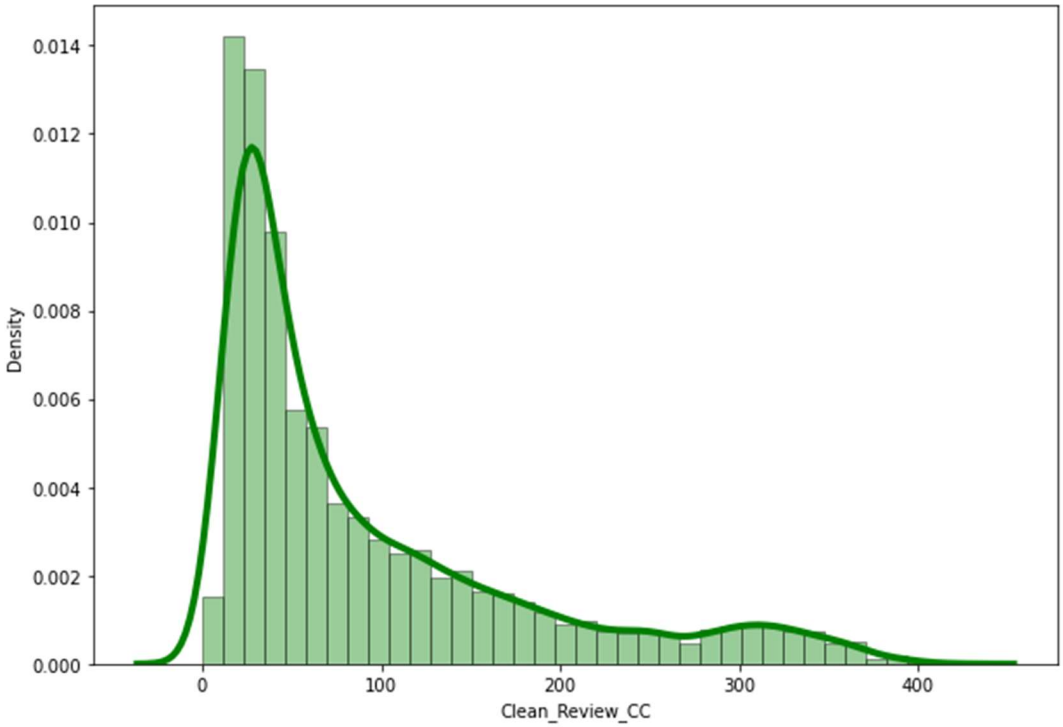
```
0.9544454916745209
[[ 630    0    2    1    1]
 [   4  127    6    8    0]
 [   4    0  324   29   13]
 [   2    1    8 1100  139]
 [   0    1    7   64 3895]]
              precision    recall  f1-score   support

           1       0.98      0.99      0.99       634
           2       0.98      0.88      0.93       145
           3       0.93      0.88      0.90       370
           4       0.92      0.88      0.90      1250
           5       0.96      0.98      0.97      3967

    accuracy                           0.95      6366
   macro avg       0.96      0.92      0.94      6366
weighted avg       0.95      0.95      0.95      6366
```

- Visualizations

  The visualization process involved plotting the graphs for the Word counts, character counts, ratings and making a word cloud.

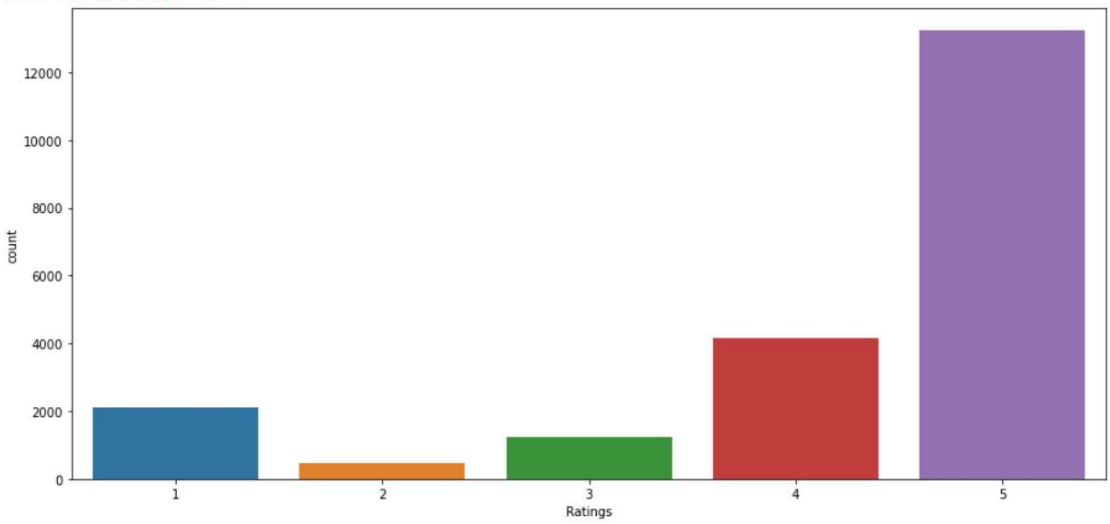
```
# Density plot and histogram of all word count
plt.figure(figsize=(10,7))
sns.distplot(df['Clean_Review_WC'], hist = True, kde = True,
             bins = int(180/5), color = 'red',
             hist_kws = {'edgecolor':'black'},
             kde_kws = {'linewidth':4})
plt.show()
```

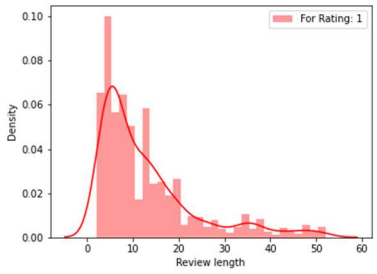


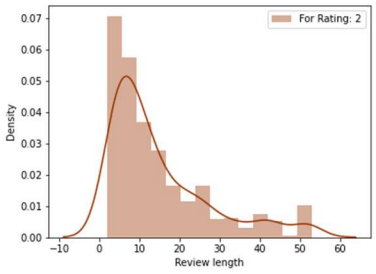


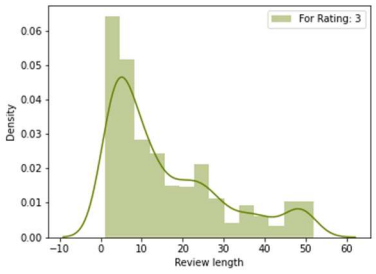

```
# Checking the count of target column values
plt.figure(figsize=(15,7))
sns.countplot(df['Ratings'])
print(df.Ratings.value_counts())
plt.show()
```

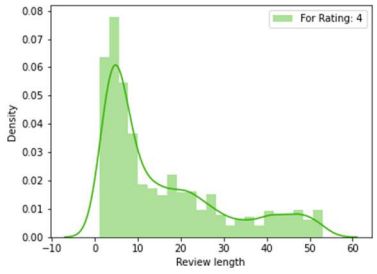
```
5    13224
4     4167
1     2112
3     1233
2      484
Name: Ratings, dtype: int64
```

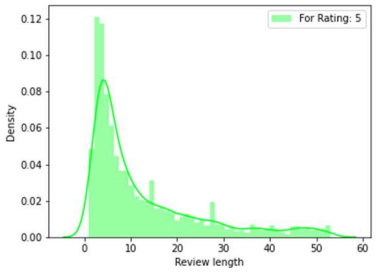


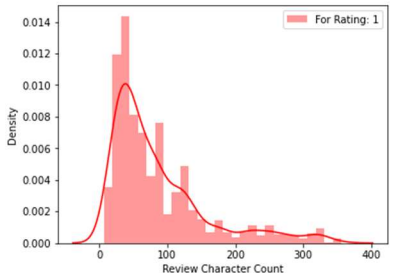


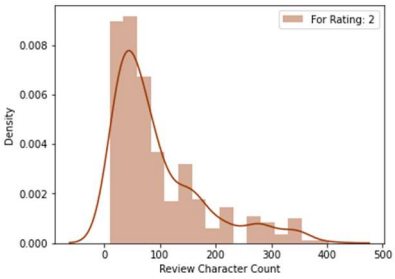


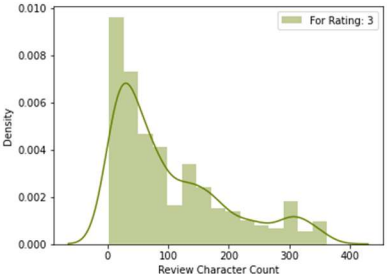


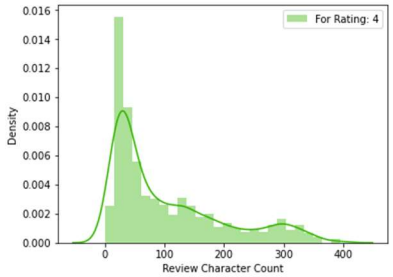


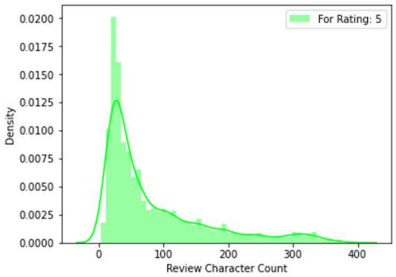






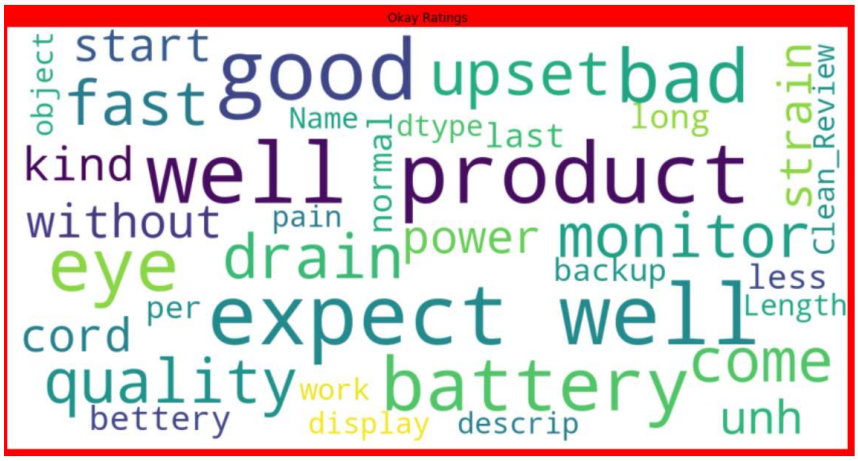







```
# word cloud for lowest rating
Display_wordcloud(df['clean_Review'][df['Ratings']==1],"Lowest")
```

```
# word cloud for 2nd lowest rating
Display_wordcloud(df['Clean_Review'][df['Ratings']==2],"Okay")
```




```
# word cloud for average rating
Display_wordcloud(df['Clean_Review'][df['Ratings']==3],"Average")
```


```
# word cloud for good rating
Display_wordcloud(df['Clean_Review'][df['Ratings']==4],"Good")
```

```
# word cloud for good rating
Display_wordcloud(df['Clean_Review'][df['Ratings']==5],"Best")
```

- Interpretation of the Results

  Using distplot the following observations were made:

  1. The data consists of 5-6 as the most occurring word count, while least occurring word counts were somewhere between 50-55.
  2. The data consists of 20-30 as the most occurring character count, while least were somewhere 370-380.
  3. Rating 5 was highest in count with above 1200 count. Least count was for the rating 2, where 5 was excellent and 1 was worst.
  4. Moving further with the removal and replacement of certain terms (like punctuations, extra spaces, numbers, money symbols, smiley etc) as well as removal of stop words, it was evident that the length of review text decreases by a large amount. This was also depicted by using distribution plots and histograms.
  5. With the help of word cloud, it was found that rating 1 which was the labelled as ‘Lowest’ consists of words like bad, product, work, expectation, quality etc.
  6. Rating 2, which was labelled as Okay consists of words like expect, well, good, product etc.
  7. Rating 3 which was labelled as Average consists of word like good, product, except, Okay, decent etc.
  8. Rating 4 which was labelled as Good consists of word like good, love, wonderful, value, nice etc.

9. Rating 5 which was labelled as Best consists of words like excellent, recommend, great, highly, super, brilliant, worth etc.

CONCLUSION

- Key Findings and Conclusions of the Study
  - After all the process used in Natural Language Processing has been applied on our dataset, our model is able to predict ratings with an accuracy of 95% with SGD classifier.
  - The model was saved in the pkl file.

```
import joblib
joblib.dump(sgd, "Rating_Prediction.pkl")
```

```
['Rating_Prediction.pkl']
```

Loading the model

```
Model = joblib.load("Rating_Prediction.pkl")
```

```
a = np.array(y_test)
predicted = np.array(Model.predict(x_test))
df_final = pd.DataFrame({"Original":a,"Predicted":predicted},index=range(len(a)))
df_final
```

	Original	Predicted
0	1	1
1	5	5
2	5	5
3	5	5
4	2	2
...	...	...
6361	5	5
6362	5	5
6363	3	3
6364	5	5
6365	2	2

6366 rows × 2 columns

- ## Learning Outcomes of the Study in respect of Data Science

  In this project we were able to learn various Natural Language Processing techniques like lemmatization, stemming, removal of Stop Words, etc. This project has demonstrated the importance of sampling effectively, modelling and predicting data. Through different powerful tools of visualization, we were able to analyse and interpret different hidden insights about the data.
  The few challenges while working on this project are:
  1. Imbalanced Dataset. 2. Lots of Text data.

  We converted text data into vectors with the help of TfidfVectorizer.

- ## Limitations of this work and Scope for Future Work

  The future of sentiment analysis is going to continue to dig deeper, far past the surface of the number of likes, comments and shares, and aim to reach, and truly understand, the significance of social media interactions and what they tell us about the consumers behind the screens. This forecast also predicts broader applications for sentiment analysis – brands will continue to leverage this tool, and so will individuals in the public eye, governments, non-profits, education centres and many other domain organizations. Sentiment analysis is getting better because social media is increasingly more emotive and expressive. A short while ago, Facebook introduced "Reactions," which allows its users to not just 'Like' content, but attach an emoticon, whether it be a heart, a shocked face, angry face, etc. To the average social media user, this is a fun, seemingly silly feature that gives him or her a little more freedom with their responses. But, to anyone looking to leverage social media data for sentiment analysis, this provides an entirely new layer of data that wasn't available before. Every time the major social media platforms update themselves and add more features, the data behind those interactions gets broader and deeper. Negative reviews can carry as much weight as positive ones. One study found that 82% of those

who read online reviews specifically seek out negative reviews. Research indicates that users spend five times as long on sites when interacting with negative reviews, with an 85% increase in conversion rate. Customers like to see lots of reviews. A single review with a few positive words makes up an opinion, but a few dozen that say the same thing make a consensus. The more reviews, the better, and one study found that consumers want to see at least 40 reviews to justify trusting an average star rating. However, a few reviews are still better than no reviews.