FLIP ROBO

Project Predicting Ratings for Reviews

Submitted by:

Aditi Gupta

ACKNOWLEDGMENT

Gratitude takes three forms-"A feeling from heart, an expression in words and a giving in return". We take this opportunity to express our feelings.

I express my gracious gratitude to our project guide **Khushboo Garg**, SME of Flip Robo Technologies, for his valuable guidance and assistance. I am very thankful for her encouragement and inspiration that made project successful.

I express my gracious gratitude to our Trainer **Dr. Deepika Sharma**, Training Head of DataTrained - Data Analytics, Data Science Online Training, Bengaluru, for her valuable guidance and assistance throughout the PG Program. I am very thankful for her encouragement and inspiration that made project successful.

I would like to thank **Shankar**, In House Data Scientist of DataTrained – Data Analytics, Data Science Online Training, Bengaluru for his Profound guidance throughout the training.

My special thanks to all the Instructors and Subject Matter Experts of DataTrained – Data Analytics, Data Science Online Training, Bengaluru, Who helped me during the live sessions and during doubt clearing Scenarios from which I received a lots of suggestions that improved the quality of the work.

I express my deep sense of gratitude to my family for their moral support and understanding without which the completion of my project would not have been perceivable.

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

  We should have knowledge on how to scrape the data from the web. The web scraping script should be able to scrape data from all kinds of ecommerce websites. Natural Language Processing and its various techniques will help us to analyse the reviews and build a model that can predict the ratings.

- Motivation for the Problem Undertaken

  I would like to build an efficient model for the client who can be reliable on the model to predict ratings for any kinds of reviews fed to the data, such that it will be useful for the client to add the new feature hassle free.

Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem

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 various ecommerce websites like Flipkart and Amazon. The scrapper was written in Python programming language using the Selenium package.

The reviews were stored in csv format. I scrapped reviews and ratings of 9 products for 20 pages so from amazon I got 26308 ratings and reviews and from flipkart I got 27251 ratings and reviews in total which I saved in 2 different csvs.

```
# storing amazon rating and reviews
df1=pd.read_csv("amazon_review_rating.csv")
df1
```

	Review	Rating
0	i went with a refurbished item on this amd im ...	5.0 out of 5 stars
1	Nodes on wifi network randomly lose internet b...	2.0 out of 5 stars
2	This model has the power of the DIR-879 with s...	5.0 out of 5 stars
3	Works great when it's working. First to use th...	3.0 out of 5 stars
4	I replaced my router from Spectrum which had a...	5.0 out of 5 stars
...	...	...
26303	Just got this...so time will tell.. This is a ...	5.0 out of 5 stars
26304	Worked and was so light for a while it slowed ...	3.0 out of 5 stars
26305	I've only had the laptop for a month but I'm v...	5.0 out of 5 stars
26306	Small and compact. Just what I was looking for...	5.0 out of 5 stars
26307	Computer was deliver in a timely manner and we...	5.0 out of 5 stars

26308 rows × 2 columns

```
# storing flipkart rating and reviews
df2=pd.read_csv("flipkart_review_rating.csv")
df2
```

	Review	Rating
0	An affordable beast ! Pros: 1. Incredible perf...	5
1	To be honest Pro's 1) RGB keyboard 2)144Hzs wi...	4
2	Best laptop in this price segment.. battery is...	5
3	This laptop is a beast, and a steal for your m...	5
4	The Laptop is a masterpiece with stunnig desig...	4
...	...	...
27246	Although I'm a geek, I always had issues with ...	4
27247	I really did bought this product just seeing t...	1
27248	Cisco linkys brands is know for its networking...	5
27249	This is an excellent router, easy to install, ...	5
27250	Seamless set up. Works very well. Does not han...	5

27251 rows × 2 columns

- Data Preprocessing Done

1. First, I loaded the 2 csv in which I stored the data I scrapped then removed all nan values. Nan values are present only in amazon dataset.

2. I noticed that ratings datatypes are different in both amazon has ratings in object type as "4.0 out of 5" and in flipkart it is like "4"…so I have done some data cleaning by removing the "out of 5" from amazon csv and converting that column into int format.

3. In order to have a balanced dataset, reviews with equal count of ratings were chosen from the two datasets and made a new dataset called df which contains 20420 total reviews and ratings with 4084 count of each.

```
df = pd.DataFrame()

amazon_rating5 = df1[df1['Rating'] == 5][:2042]
amazon_rating4 = df1[df1['Rating'] == 4][:2042]
amazon_rating3 = df1[df1['Rating'] == 3][:1907]
amazon_rating2 = df1[df1['Rating'] == 2][:1450]
amazon_rating1 = df1[df1['Rating'] == 1][:2042]
flipkart_rating5 = df2[df2['Rating'] == 5][:2042]
flipkart_rating4 = df2[df2['Rating'] == 4][:2042]
flipkart_rating3 = df2[df2['Rating'] == 3][:2177]
flipkart_rating2 = df2[df2['Rating'] == 2][:2634]
flipkart_rating1 = df2[df2['Rating'] == 1][:2042]

df = df.append(flipkart_rating5,ignore_index=True)
df = df.append(amazon_rating5,ignore_index=True)
df = df.append(flipkart_rating4,ignore_index=True)
df = df.append(amazon_rating4,ignore_index=True)
df = df.append(flipkart_rating3,ignore_index=True)
df = df.append(amazon_rating3,ignore_index=True)
df = df.append(flipkart_rating2,ignore_index=True)
df = df.append(amazon_rating2,ignore_index=True)
df = df.append(amazon_rating1,ignore_index=True)
df = df.append(flipkart_rating1,ignore_index=True)

df
```

	Review	Rating
0	An affordable beast ! Pros: 1. Incredible perf...	5
1	Best laptop in this price segment.. battery is...	5
2	This laptop is a beast, and a steal for your m...	5
3	Good laptop but customer care folks are real d...	5
4	So i wanted a decent Gaming Laptop with Good s...	5
...	...	...
20415	Firstly it is not 1200 mbps its 867 mbps. Seco...	1
20416	Doesn't working 3 decos on even 2 floors in 15...	1
20417	I have used lot of Routers over the last 6-7 y...	1
20418	DLink sucks and the 3 year warranty is a sham....	1
20419	I got this reading all the reviews, my belkin ...	1

20420 rows × 2 columns

4. Data was cleaned by removing punctuations, white spaces, and special characters.
5. Using the stopwords package, all stop words were removed along with some other values which occurred often but did not have any meaning.
6. Lemmatization was used on all the texts or reviews.
7. 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. EDA was performed by creating word clouds that helped to identify frequently occurring words for each ratings.
3. The review column was converted into vectors and was used as an input.
4. Various classifiers like Multinomial Naive Bayes, Random Forest Classifier, SVC, Decision Tree Classifier, and KNeighbors Classifier were used to train the model.
5. 4. Accuracy and F1 scores of all the classifiers were compared and at the end, ExtraTreesClassifier was predicting the ratings with an average accuracy of about 63%.

- ## Hardware and Software Requirements and Tools Used

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

  Software : Following librarieswere 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. WordCloud: To create a word cloud representing the highest frequency of texts in the corpus.

  6. WordNetLemmatizer : To lemmatize the text data

  7. TfidfVectoriser : To convert text into vectors.

  8. train_test_split : To split dataset into training and testing dataset.
  9. KNeighbourClassifier, Support Vector Classifier, Decision Tree Classifier,Random Forest Classifier, Multinomial NB : Classification models used to train our data.

  10. Confusion_matrix : To evaluate the accuracy of a classification.
  11. Classification_report : A report is presented with accuracy scores, f1 scores, recall and precision scores as well

  12. Joblib : To save the model

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  Firstly, reviews and ratings from various e-commerce websites for various products were scrapped and stored in a csv file. Later that data was imported and a new dataset was created, with the count of each rating to be similar in order to balance the dataset.

  The data cleaning was performed on the dataset - all missing values were handled, 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 were 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 : Multinomial NB , Decision Tree Classifier, Random Forest Classifier, Bagging Classifier, Extra Trees Classifier and XGB Classifier.

- Run and Evaluate selected models

```
# Importing useful libraries for model training

from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier

# Ensemble Techniques...

from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from xgboost import XGBClassifier

# Model selection libraries...
from sklearn.model_selection import cross_val_score, cross_val_predict, train_test_split
from sklearn.model_selection import GridSearchCV

# Importing some metrics we can use to evaluate our model performance....
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.metrics import roc_auc_score, roc_curve, auc
from sklearn.metrics import precision_score, recall_score, f1_score

# Creating instances for different Classifiers

MNB=MultinomialNB()
DTC=DecisionTreeClassifier()
RFC=RandomForestClassifier()
BGC=BaggingClassifier()
ETC=ExtraTreesClassifier()
XGB=XGBClassifier()
```

```
#     Putting Scikit-Learn machine learning Models in a list so that it can be used for further evaluation in loop.
models=[]
models.append(('MultinomialNB()',MNB))
models.append(('DecisionTreeClassifier',DTC))
models.append(('RandomForestClassifier',RFC))
models.append(('BaggingClassifier',BGC))
models.append(('ExtraTreesClassifier',ETC))
models.append(('XGBClassifier',XGB))
```

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

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

for name,model in models:
    print('****************************',name,'******************************')
    print('\n')
    Model.append(name)
    print(model)
    print('\n')

     #         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)
#................Finding Cross_val_score..................
    cv_score=cross_val_score(model,x,y,cv=10,scoring='accuracy').mean()
    print('Cross Val Score : ', cv_score)
    cvs.append(cv_score*100)


#................Classification Report...........................
    print('Classification Report:\n',classification_report(y_test,y_pred))
    print('\n')

    print('Confusion Matrix:\n',confusion_matrix(y_test,y_pred))
    print('\n')
```

```
************************** MultinomialNB() ****************************


MultinomialNB()


Max Accuracy Score corresponding to Random State  59 is: 0.5848841005550114


Learning Score :  0.63187351336225
Accuracy Score :  0.5848841005550114
Cross Val Score :  0.4634671890303624
Classification Report:
               precision    recall  f1-score   support

           1       0.61      0.72      0.66      1225
           2       0.69      0.51      0.59      1226
           3       0.46      0.43      0.44      1225
           4       0.56      0.47      0.51      1225
           5       0.61      0.79      0.69      1225

    accuracy                           0.58      6126
   macro avg       0.59      0.58      0.58      6126
weighted avg       0.59      0.58      0.58      6126



Confusion Matrix:
 [[882 122 123  59  39]
 [325 627 153  81  40]
 [193  99 531 197 205]
 [ 28  50 245 580 322]
 [ 17  10 115 120 963]]
```

```
************************** DecisionTreeClassifier ****************************


DecisionTreeClassifier()


Max Accuracy Score corresponding to Random State  43 is: 0.5971269996735227


Learning Score :  0.8957604589338184
Accuracy Score :  0.5956578517793013
Cross Val Score :  0.4343290891283056
Classification Report:
               precision    recall  f1-score   support

           1       0.61      0.66      0.63      1225
           2       0.61      0.51      0.56      1225
           3       0.46      0.51      0.48      1225
           4       0.62      0.58      0.60      1225
           5       0.70      0.72      0.71      1226

    accuracy                           0.60      6126
   macro avg       0.60      0.60      0.60      6126
weighted avg       0.60      0.60      0.60      6126



Confusion Matrix:
 [[806 179 167  48  25]
 [268 628 194  80  55]
 [160 146 621 170 128]
 [ 50  49 236 712 178]
 [ 31  34 141 138 882]]
```

```
*************************** RandomForestClassifier *****************************


RandomForestClassifier()


Max Accuracy Score corresponding to Random State  87 is: 0.6524649036891936


Learning Score :  0.8955505806632154
Accuracy Score :  0.6495266079007509
Cross Val Score :  0.4909892262487757
Classification Report:
              precision    recall  f1-score   support

           1       0.64      0.71      0.67      1225
           2       0.72      0.54      0.62      1225
           3       0.51      0.55      0.53      1225
           4       0.69      0.63      0.66      1226
           5       0.72      0.81      0.76      1225

    accuracy                           0.65      6126
   macro avg       0.65      0.65      0.65      6126
weighted avg       0.65      0.65      0.65      6126



Confusion Matrix:
 [[870 130 148  45  32]
 [296 664 182  49  34]
 [160  92 674 152 147]
 [ 21  33 212 777 183]
 [ 12   9 100 110 994]]
```

```
*************************** BaggingClassifier *****************************


BaggingClassifier()


Max Accuracy Score corresponding to Random State  99 is: 0.618837740777016


Learning Score :  0.8841471946271162
Accuracy Score :  0.6147567744041789
Cross Val Score :  0.4671890303623898
Classification Report:
              precision    recall  f1-score   support

           1       0.61      0.68      0.64      1225
           2       0.66      0.52      0.58      1226
           3       0.48      0.55      0.51      1225
           4       0.65      0.57      0.60      1225
           5       0.71      0.76      0.73      1225

    accuracy                           0.61      6126
   macro avg       0.62      0.61      0.61      6126
weighted avg       0.62      0.61      0.61      6126



Confusion Matrix:
 [[830 159 166  43  27]
 [276 643 205  63  39]
 [178 106 670 152 119]
 [ 38  50 243 697 197]
 [ 28  23 123 125 926]]
```

```
************************** ExtraTreesClassifier ****************************


ExtraTreesClassifier()


Max Accuracy Score corresponding to Random State  44 is: 0.6593209271955599


Learning Score :  0.8943612704631314
Accuracy Score :  0.6604635977799543
Cross Val Score :  0.4929970617042116
Classification Report:
               precision    recall  f1-score   support

           1       0.65      0.76      0.70      1225
           2       0.72      0.58      0.64      1225
           3       0.52      0.56      0.54      1225
           4       0.72      0.60      0.65      1225
           5       0.72      0.81      0.76      1226

    accuracy                           0.66      6126
   macro avg       0.67      0.66      0.66      6126
weighted avg       0.67      0.66      0.66      6126



Confusion Matrix:
 [[930 126 119  20  30]
 [275 710 167  47  26]
 [191 103 684 128 119]
 [ 23  40 212 735 215]
 [ 17   7 125  90 987]]
```

```
************************** XGBClassifier ****************************


XGBClassifier(base_score=None, booster=None, colsample_bylevel=None,
              colsample_bynode=None, colsample_bytree=None, gamma=None,
              gpu_id=None, importance_type='gain', interaction_constraints=None,
              learning_rate=None, max_delta_step=None, max_depth=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,
              random_state=None, reg_alpha=None, reg_lambda=None,
              scale_pos_weight=None, subsample=None, tree_method=None,
              validate_parameters=None, verbosity=None)
Max Accuracy Score corresponding to Random State  81 is: 0.6335292197192295

Cross Val Score :  0.48540646425073464
Classification Report:
               precision    recall  f1-score   support

           1       0.66      0.74      0.70      1225
           2       0.62      0.57      0.59      1225
           3       0.51      0.51      0.51      1225
           4       0.68      0.57      0.62      1226
           5       0.70      0.78      0.74      1225

    accuracy                           0.63      6126
   macro avg       0.63      0.63      0.63      6126
weighted avg       0.63      0.63      0.63      6126



Confusion Matrix:
 [[908 167 100  32  18]
 [289 697 171  48  20]
 [156 175 629 133 132]
 [ 19  70 214 694 229]
 [ 10  19 131 112 953]]
```

- ## Hyperparameter Tuning

  From the above results we have seen that we are getting highest accuracy score in Random Forest Classifier and Extra trees classifier now we will use Gridsearch CV in these two to find the best one.

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

```
#Performing Hyperparameter tuning on RandomForestClassifier

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': 500}
```

```
#Applying the parameters we got after hyper parameter tuning
rfc=RandomForestClassifier(n_estimators=500,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))
```

```
0.6243878550440745
[[913  79 167  29  37]
 [306 598 248  39  34]
 [179  57 753  82 154]
 [ 32  32 318 600 244]
 [ 19   5 192  48 961]]
              precision    recall  f1-score   support

           1       0.63      0.75      0.68      1225
           2       0.78      0.49      0.60      1225
           3       0.45      0.61      0.52      1225
           4       0.75      0.49      0.59      1226
           5       0.67      0.78      0.72      1225

    accuracy                           0.62      6126
   macro avg       0.66      0.62      0.62      6126
weighted avg       0.66      0.62      0.62      6126
```

```
#Performing Hyperparameter tuning on ExtraTreesClassifier

etc=ExtraTreesClassifier()
parameters={'n_estimators':[100,300,500,800,1200],'max_depth':[3, 5, 8, 15, 25, 30]}
clf=GridSearchCV(etc,parameters,cv=5)
clf.fit(x_train,y_train)
clf.best_params_
```

```
{'max_depth': 30, 'n_estimators': 1200}
```

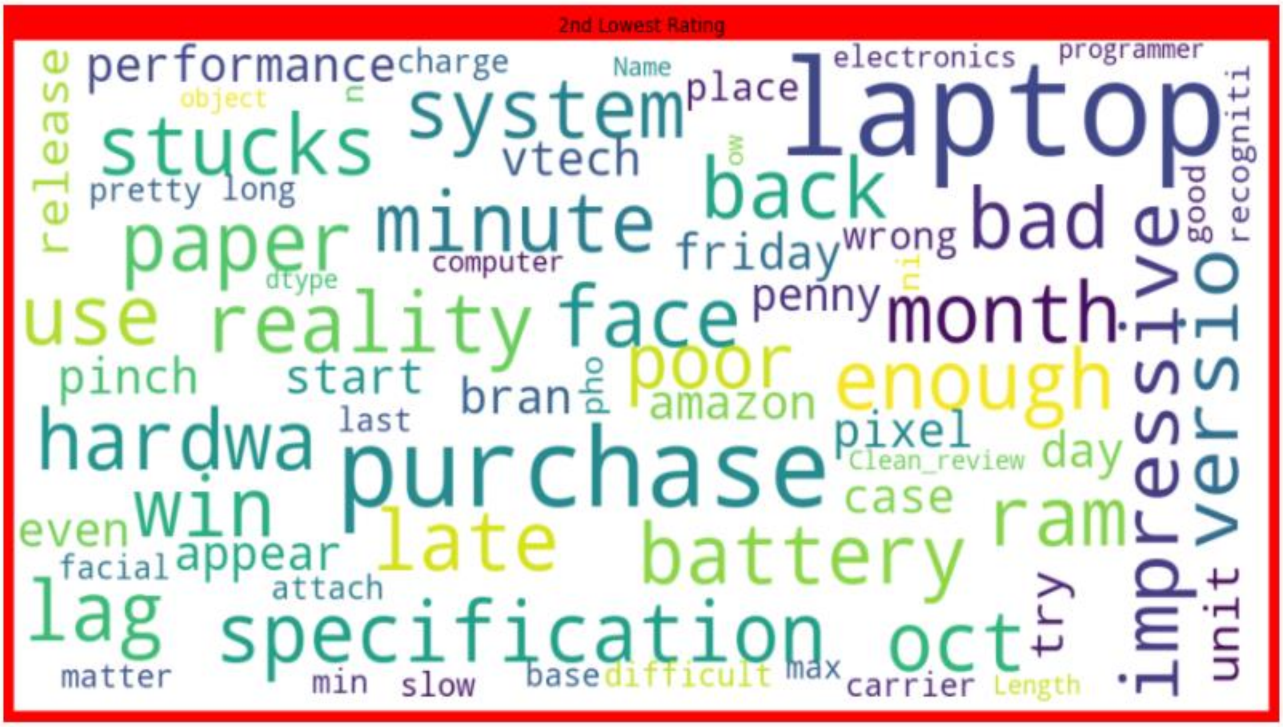
```
#Applying the parameters we got after hyper parameter tuning
etc=RandomForestClassifier(n_estimators=1200,max_depth=30)
etc.fit(x_train,y_train)
predetc=etc.predict(x_test)
print(accuracy_score(y_test,predetc))
print(confusion_matrix(y_test,predetc))
print(classification_report(y_test,predetc))
```

```
0.6312438785504407
[[936  71 166  25  27]
 [310 604 241  39  31]
 [178  57 754  87 149]
 [ 33  38 314 615 226]
 [ 20   7 196  44 958]]
              precision    recall  f1-score   support

           1       0.63      0.76      0.69      1225
           2       0.78      0.49      0.60      1225
           3       0.45      0.62      0.52      1225
           4       0.76      0.50      0.60      1226
           5       0.69      0.78      0.73      1225

    accuracy                           0.63      6126
   macro avg       0.66      0.63      0.63      6126
weighted avg       0.66      0.63      0.63      6126
```

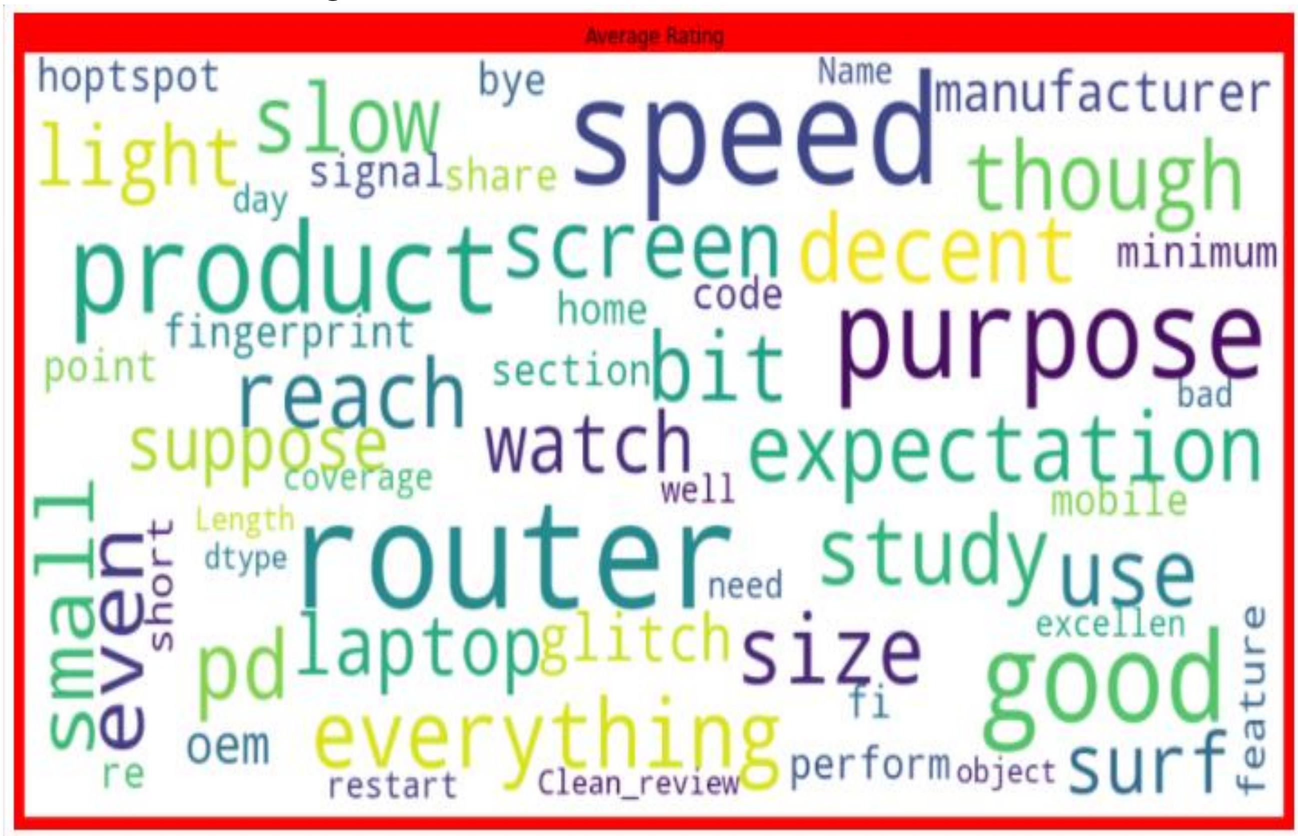
- Visualizations

Word Cloud for Rating 1

Word Cloud for Rating 2



Word Cloud for Rating 3



Word Cloud for Rating 4

Word Cloud for Rating 5

- # Interpretation of Result

After applying results of hyperparameter tuning we have choosen ExtraTreesClassifier as our final model.

```
etc=RandomForestClassifier(n_estimators=1200,max_depth=30)
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))
```

```
0.6302644466209598
[[921  72 176  25  31]
 [305 596 255  38  31]
 [176  53 772  79 145]
 [ 33  36 313 616 228]
 [ 21   6 198  44 956]]
              precision    recall  f1-score   support

           1       0.63      0.75      0.69      1225
           2       0.78      0.49      0.60      1225
           3       0.45      0.63      0.53      1225
           4       0.77      0.50      0.61      1226
           5       0.69      0.78      0.73      1225

    accuracy                           0.63      6126
   macro avg       0.66      0.63      0.63      6126
weighted avg       0.66      0.63      0.63      6126
```

```
# cross validation

from sklearn.model_selection import cross_val_score

scores=cross_val_score(etc,x,y,cv=10)
print(scores)
print(scores.mean(),scores.std())
```

```
[0.33790402 0.47110676 0.47110676 0.53085211 0.68658178 0.59353575
 0.47306562 0.52693438 0.61851126 0.58325171]
0.5292850146914789 0.09290256478794608
```

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 63% with Extra trees classifier.

- Learning Outcomes of the Study in respect of Data Science

  For any given dataset, the EDA process is extremely important as well as beneficial in order to build an effective model. Visualizations help us to analyze the data patterns, outliers, and various information of the events that occurred. It will also help in the data cleaning process. Data cleaning and manipulation is the next big step which will bring out the best in the data.

  While working on this project, initially web scraping was a challenge. Reviews and ratings were being duplicated or null strings would be appended. I read more articles on the internet, understood the problem carefully and tried various ways to bring out the best method and executed this project.

- Limitations of this work and Scope for Future Work

  With web scraping, more data can be retrieved and can be used to train the model. More data will build the model better.. Currently, despite having retrieved more than 20K data from the e-commerce websites, balancing the data with the same count of ratings was a major objective but had to leave out many reviews.

THANKS