RATING PREDICTION PROJECT

Submitted by:

Yatendra Jha

ACKNOWLEDGMENT

I would like to thank Flip Robo Technologies for providing me with the opportunity to work on this project from which I have learned a lot. I am also grateful to Miss Khushboo Garg for her constant guidance and support.

Some of the reference sources are as follows:

- Internet
- Coding Ninjas
- Medium.com
- Analytics Vidhya
- StackOverflow

TABLE OF CONTENTS

ACKNOWLEDGMENT 2
INTRODUCTION 1
BUSINESS PROBLEM FRAMING 1
CONCEPTUAL BACKGROUND OF THE DOMAIN PROBLEM 1
REVIEW OF LITERATURE 1
MOTIVATION FOR THE PROBLEM UNDERTAKEN 2
ANALYTICAL PROBLEM FRAMING 2
MATHEMATICAL/ ANALYTICAL MODELING OF THE PROBLEM 2
DATA SOURCES AND THEIR FORMATS 4
DATA PREPROCESSING DONE 5
DATA INPUTS- LOGIC- OUTPUT RELATIONSHIPS 6
HARDWARE AND SOFTWARE REQUIREMENTS AND TOOLS USED 6
MODEL/S DEVELOPMENT AND EVALUATION 8
IDENTIFICATION OF POSSIBLE PROBLEM-SOLVING APPROACHES (METHODS) 8
TESTING OF IDENTIFIED APPROACHES (ALGORITHMS) 8
RUN AND EVALUATE SELECTED MODELS 9
KEY METRICS FOR SUCCESS IN SOLVING PROBLEM UNDE CONSIDERATION 10
VISUALIZATION 10
KEY METRICS FOR SUCCESS IN SOLVING PROBLEM UNDER CONSIDERATION 18
CONCLUSION 18
KEY FINDINGS AND CONCLUSIONS OF THE STUDY 18
LEARNING OUTCOMES OF THE STUDY IN RESPECT OF DATA SCIENCE 18
LIMITATIONS OF THIS WORK AND SCOPE FOR FUTURE WORK 19

INTRODUCTION

BUSINESS PROBLEM FRAMING

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 rating. So we, we have to build an application which can predict the rating by seeing the review.

CONCEPTUAL BACKGROUND OF THE DOMAIN PROBLEM

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

REVIEW OF LITERATURE

The rapid development of Web 2.0 and e-commerce has led to a proliferation in the number of online user reviews. Online reviews contain a wealth of sentiment information that is important for many decision-making processes, such as personal consumption decisions, commodity quality monitoring, and social opinion mining. Mining the sentiment and opinions that are contained in online reviews has become an important topic in natural language processing, machine learning, and Web mining.

MOTIVATION FOR THE PROBLEM UNDERTAKEN

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.

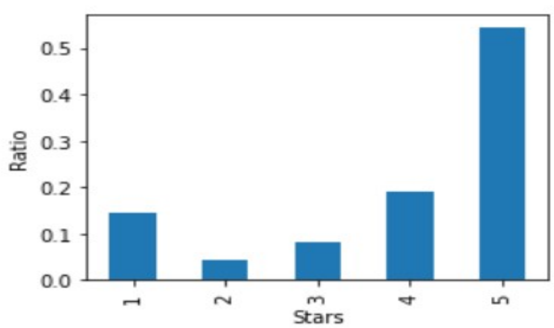
ANALYTICAL PROBLEM FRAMING

MATHEMATICAL/ ANALYTICAL MODELING OF THE PROBLEM

- There are in total 39369 rows and 2 columns of ratings and reviews are in our dataset post web scraping from FLIPKART.

We found the occurrence of ratings ratio as shown below:

```
In [7]: plot_labels(Rating, "stars")
```



We can observe that the dataset is imbalanced.

```
In [5]: print('Rating counts','\n',Rating.Ratings.value_counts())

Rating counts
 5     21394
4     7461
1     5654
3     3186
2     1674
Name: Ratings, dtype: int64
```

Observation:

Maximum, 21394 number of ratings present is of 5 star and minimum, 1674 is of 2 star.

- Maximum 21394 numbers of ratings present is of 5 star and minimum 1674 is of 2 star.
- We then create two more columns length and clean_leangth on the basis of the lengths of the text before and after cleanining for our analysis purpose.

In [8]:
```
Rating['length']=Rating.Full_review.str.len()
Rating.head()
```

Out[8]:

	Ratings	Full_review	length
0	5	Its an absolute beast if u know what are the n...	500
1	5	This is the best laptop in this range.I reciev...	500
2	5	Good product as used of now.... Everything is ...	271
3	5	AWESOME LAPTOP. It supports many high spec gam...	96
4	4	For that price... it's exceptionally good. Pla...	342

Here we create another column length based on the length of reviews.

In [12]:
```
#convert text to lowercase
Rating['Full_review']=Rating['Full_review'].str.lower()
```

In [13]:
```
Rating['Full_review']=Rating['Full_review'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$', 'emailaddress')

Rating['Full_review']=Rating['Full_review'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$', 'webaddress')

Rating['Full_review']=Rating['Full_review'].str.replace(r'£|\$', 'dollers')

Rating['Full_review']=Rating['Full_review'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$', 'phonenumber')

Rating['Full_review']=Rating['Full_review'].str.replace(r'\d+(\.\d+)?', 'numbr')
```

In [14]:
```
#remove punctuation
Rating['Full_review']=Rating['Full_review'].str.replace(r'[^\w\d\s]', ' ')

#replace whitespace between terms with a single space
Rating['Full_review']=Rating['Full_review'].str.replace(r'\s+', ' ')

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

In [15]:
```
Rating.head()
```

Out[15]:

	Ratings	Full_review	length
0	5	its an absolute beast if u know what are the n...	500
1	5	this is the best laptop in this range i reciev...	500
2	5	good product as used of now everything is good...	271
3	5	awesome laptop it supports many high spec game...	96
4	4	for that price it s exceptionally good played ...	342

In [16]:
```
#Remove stopwords
import string
import nltk
from nltk.corpus import stopwords

stop_words = set(stopwords.words('english') + ['u', 'ur', '4', '2', 'im', 'dont', 'doin', 'ure'])

Rating['Full_review'] = Rating['Full_review'].apply(lambda x: ' '.join(term for term in x.split() if term not in stop_words))
```

In [17]:
```
Rating['clean_length'] = Rating.Full_review.str.len()
```

```
In [18]: Rating.head()
```

Out[18]:

	Ratings	Full_review	length	clean_length
0	5	absolute beast know necessary steps follow com...	500	294
1	5	best laptop range recieved late delivery due b...	500	337
2	5	good product used everything good also ssd slo...	271	150
3	5	awesome laptop supports many high spec games l...	96	84
4	4	price exceptionally good played far cry numbr ...	342	254

```
In [19]: print('original Review length', Rating.length.sum())
         print('clean Review length', Rating.clean_length.sum())

original Review length 2383903
clean Review length 1692896
```

DATA SOURCES AND THEIR FORMATS

The variable features of this problem statement are as follows:-

- **Ratings:** It is the Label column, which includes ratings in the form of integers from 1 to 5.
- **Full_review:** It contains text data on the basis of which we have to build a model to predict ratings.

Dataset description

- Data is scrapped from the FLIPKART for various items like Laptop, Headphones, Routers, Mobile Phones, Smart Watches, Professional Camera, Printers, Home Theater, Monitors etc.

```
In [124]: Reviews
```

Out[124]:

	Ratings	Full_review
0	5	Its an absolute beast if u know what are the n...
1	5	This is the best laptop in this range.I reciev...
2	5	Good product as used of now.... Everything is ...
3	5	AWESOME LAPTOP. It supports many high spec gam...
4	4	For that price... it's exceptionally good. Pla...
...	...	...
39364	5	This device completely covers my bunglow.. Goo...
39365	5	a great product from tenda,....i literally kee...
39366	3	Good router bt Flipkart fluctuates it's price....
39367	4	Too much gd...lokking nice and fit..range is a...
39368	5	Really great product, delivery was awesome. Al...

39369 rows × 2 columns

Identification of possible problem-solving approaches (methods)

After collecting the data, we need to build a machine learning model. Before model buildings we do all data preprocessing steps involving NLP. Try different models with different hyper parameters and select the best model.

a) Data Cleaning
b) Exploratory Data Analysis
c) Data Preprocessing
d) Model Building
e) Model Evaluation
f) Selecting the best model

DATA PREPROCESSING DONE

We first looked for the null values present in the dataset. We noticed that there were no null values present in our dataset. Then we performed text processing. Data usually comes from a variety of sources and often in different formats. For this reason transforming your raw data is essential. However, this is not a simple process, as text data often contains redundant and repetitive words. This means that processing the text data is the first step in our solution. The fundamental steps involved in text pre-processing are, cleaning the raw data tokenizing the cleaned data.

Some of the steps are as follows:-

Cleaning the Raw Data

This phase involves the deletion of words or characters that do not add value to the meaning of the text. Some of the standard cleaning steps are listed below:

- Lowering case
- Removal of special characters
- Removal of stopwords
- Removal of hyperlinks
- Removal of numbers

- Removal of whitespaces

Lowering Case

Lowering the case of text is essential for the following reasons: The words, 'TEXT', 'Text', 'text' all add the same value to a sentence lowering the case of all the words is very helpful for reducing the dimensions by decreasing the size of the vocabulary.

Removal of special characters

This is another text processing technique that will help to treat words like 'hurray' and 'hurray!' in the same way.

Removal of stop words

Stopwords are commonly occurring words in a language like 'the', 'a', and so on. Most of the time they can be removed from the text because they don't provide valuable information.

Set of assumptions related to the problem under consideration

By looking into the target variable label we assumed that it was a Multiclass classification type of problem.

We observed that dataset was imbalance so we will have to balance the dataset for better outcome.

DATA INPUTS- LOGIC- OUTPUT RELATIONSHIPS

For this data's input and output logic, we will analyse words frequency for each label, so that we can get the most frequent words that were used in different features.

HARDWARE AND SOFTWARE REQUIREMENTS AND TOOLS USED

HARDWARE:

SOFTWARE:

Jupyter Notebook (Anaconda 3) – Python 3.7.6

Microsoft Excel 2010

LIBRARIES:

- Pandas: To read the Data file in form of data.
- Matplotlib: This library is typically used to plot the figures for better visualisation of data.
- Seaborn: A advanced version of Matplotlib
- Scikit Learn: This is the most important library for Machine Learning since it contains various Machine Learning Algorithms which are used in this project. Scikit Learn also contains Preprocessing library which is used in data preprocessing. Apart from this, it contains a very useful joblib library for serialization purpose using which the final model has been saved in this project.
- NLTK: Natural language took kit is one of the most used libraries for building NLP projects.
- Through pandas library we loaded our csv file ‘messages’ into dataframe and performed data manipulation and analysis. With the help of numpy we worked with arrays.

- With the help of matplotlib and seaborn we did plot various graphs and figures and done data visualization.
- With wordcloud we got sense of loud words present in the dataset. Through tfidf vectorizer we converted text into vectors.
- Through smote technique we handled the imbalanced dataset.
- Through Gridsearchcv we tried to find the best parameters of random forest classifier.
- Through joblib we saved our model in csv format.

MODEL/S DEVELOPMENT AND EVALUATION

IDENTIFICATION OF POSSIBLE PROBLEM-SOLVING APPROACHES (METHODS)

- Preprocessing involved the following steps:-
- Removing Punctuations and other special characters
- Removing Stop Words
- Stemming and Lemmatising Applying
- tfidf Vectorizer
- Splitting dataset into Training and Testing

TESTING OF IDENTIFIED APPROACHES (ALGORITHMS)

The algorithms we used for the training and testing are as follows:-

- Decision tree classifier
- Kneighbors classifier
- MultinomialNB
- Random forest classifier
- Adaboost classifier
- Gradient boosting classifier
- Bagging classifier
- Extra trees classifier

RUN AND EVALUATE SELECTED MODELS

In [36]:
```
#Importing all the model library

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

#Importing Boosting models
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier

#Importing error metrics
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,roc_curve,auc
from sklearn.model_selection import GridSearchCV,cross_val_score
```

In [37]:
```
KNN=KNeighborsClassifier(n_neighbors=6)
DT=DecisionTreeClassifier(random_state=6)
XGB=XGBClassifier()
RF=RandomForestClassifier()
ADA=AdaBoostClassifier()
MNB=MultinomialNB()
GBC=GradientBoostingClassifier()
BC=BaggingClassifier()
ETC=ExtraTreesClassifier()
```

In [38]:
```
models= []
models.append(('KNeighborsClassifier', KNN))
models.append(('DecisionTreeClassifier', DT))
models.append(('XGBClassifier', XGB))
models.append(('RandomForestClassifier', RF))
models.append(('AdaBoostClassifier', ADA))
models.append(('MultinomialNB', MNB))
models.append(('GradientBoostingClassifier', GBC))
models.append(('BaggingClassifier', BC))
models.append(('ExtraTreesClassifier', ETC))
```

In [40]:
```
result = pd.DataFrame({'Model': Model, 'Accuracy_score': score,'Cross_val_score': cvs})
result
```

Out[40]:

	Model	Accuracy_score	Cross_val_score
0	KNeighborsClassifier	32.778766	54.034738
1	DecisionTreeClassifier	52.298705	56.897601
2	XGBClassifier	52.819406	63.336622
3	RandomForestClassifier	55.613411	63.138502
4	AdaBoostClassifier	42.684785	61.995473
5	MultinomialNB	49.453899	62.328237
6	GradientBoostingClassifier	47.129794	61.959915
7	BaggingClassifier	52.908306	59.966022
8	ExtraTreesClassifier	55.930912	62.356178

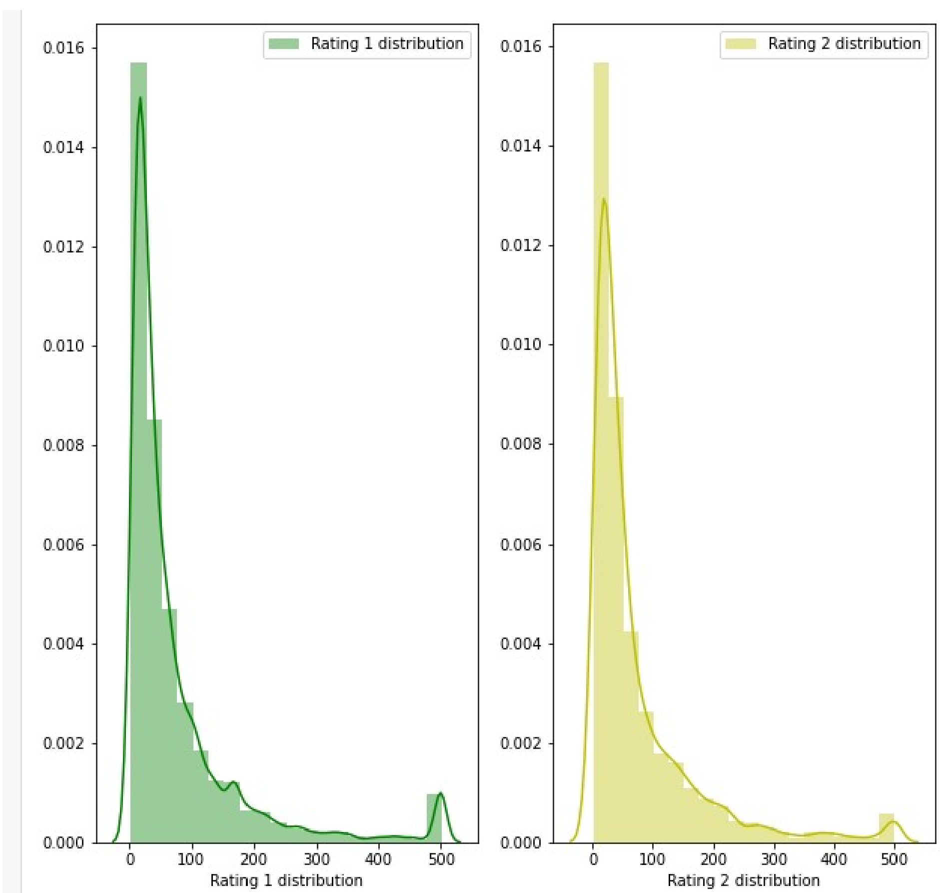
KEY METRICS FOR SUCCESS IN SOLVING PROBLEM UNDE CONSIDERATION

On the basis of accuracy and confusion matrix we save Random forest classifier as our final model.

VISUALIZATION

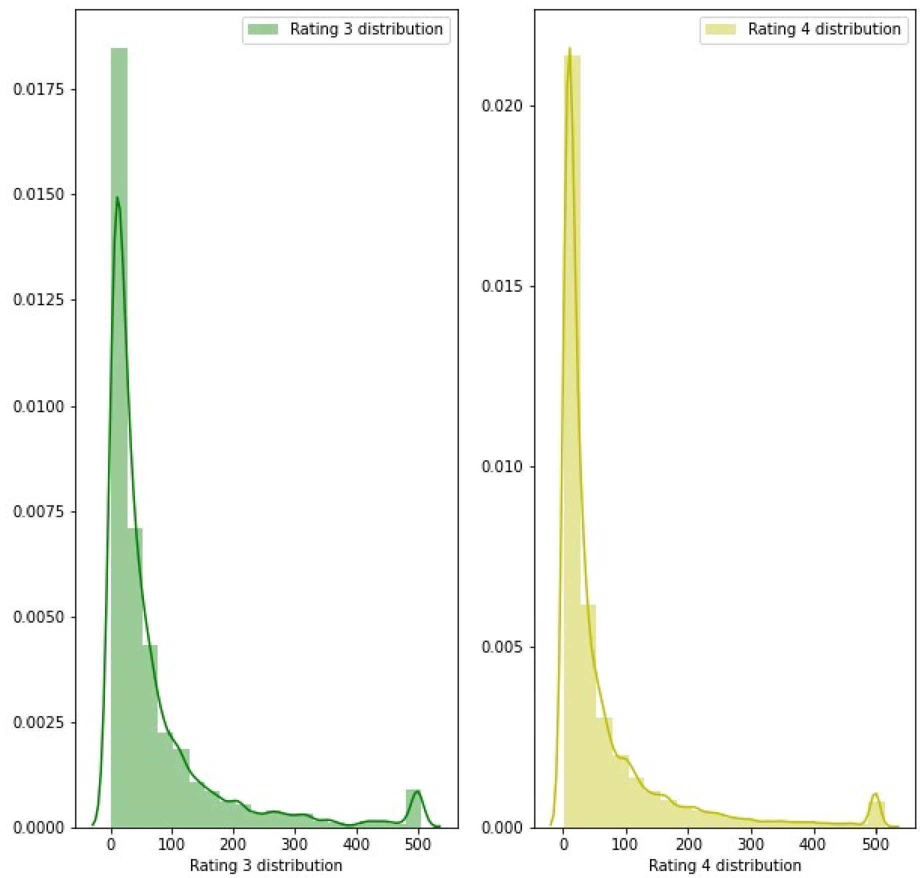
Rating 1 and Rating 2 distribution before cleaning the reviews:

```
In [20]: #message distribution before cleaning

f,ax = plt.subplots(1,2,figsize=(10,10))

sns.distplot(Rating[Rating['Ratings']==1]['length'],bins=20,ax=ax[0],label='Rating 1 distribution',color='g')
ax[0].set_xlabel('Rating 1 distribution')
ax[0].legend()

sns.distplot(Rating[Rating['Ratings']==2]['length'],bins=20,ax=ax[1],label='Rating 2 distribution',color='y')
ax[1].set_xlabel('Rating 2 distribution')
ax[1].legend()

plt.show()
```



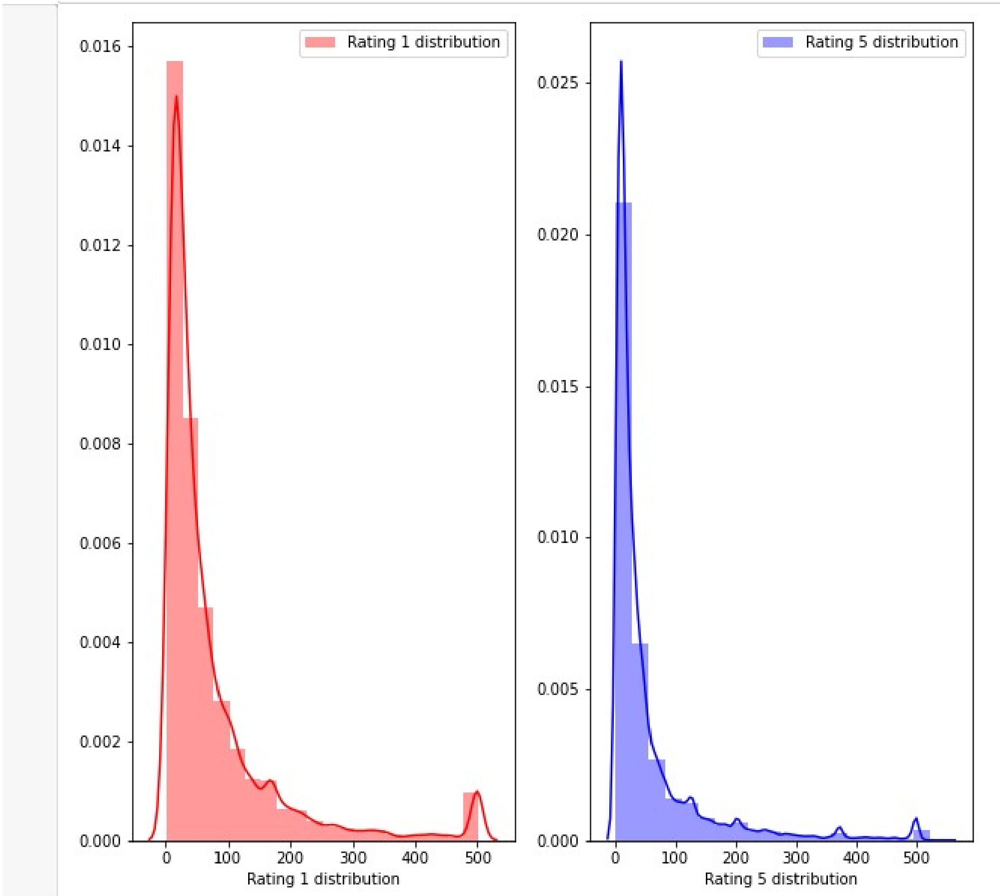
Rating 3 and and Rating 4 distribution before cleaning the reviews:

```
In [21]: f,ax = plt.subplots(1,2,figsize=(10,10))

sns.distplot(Rating[Rating['Ratings']==3]['length'],bins=20,ax=ax[0],label='Rating 3 distribution',color='g')
ax[0].set_xlabel('Rating 3 distribution')
ax[0].legend()

sns.distplot(Rating[Rating['Ratings']==4]['length'],bins=20,ax=ax[1],label='Rating 4 distribution',color='y')
ax[1].set_xlabel('Rating 4 distribution')
ax[1].legend()

plt.show()
```



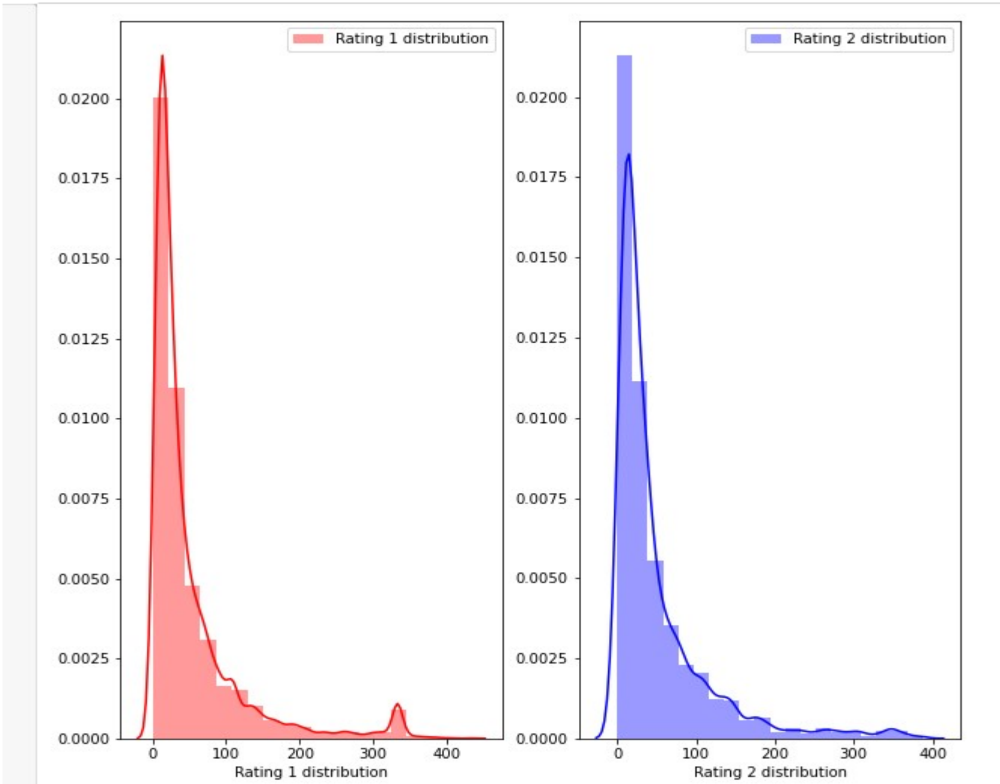
Rating 1 and Rating 5 distribution before cleaning reviews:

```
In [22]: f,ax = plt.subplots(1,2,figsize=(10,10))

sns.distplot(Rating[Rating['Ratings']==1]['length'],bins=20,ax=ax[0],label='Rating 1 distribution',color='r')
ax[0].set_xlabel('Rating 1 distribution')
ax[0].legend()

sns.distplot(Rating[Rating['Ratings']==5]['length'],bins=20,ax=ax[1],label='Rating 5 distribution',color='b')
ax[1].set_xlabel('Rating 5 distribution')
ax[1].legend()

plt.show()
```

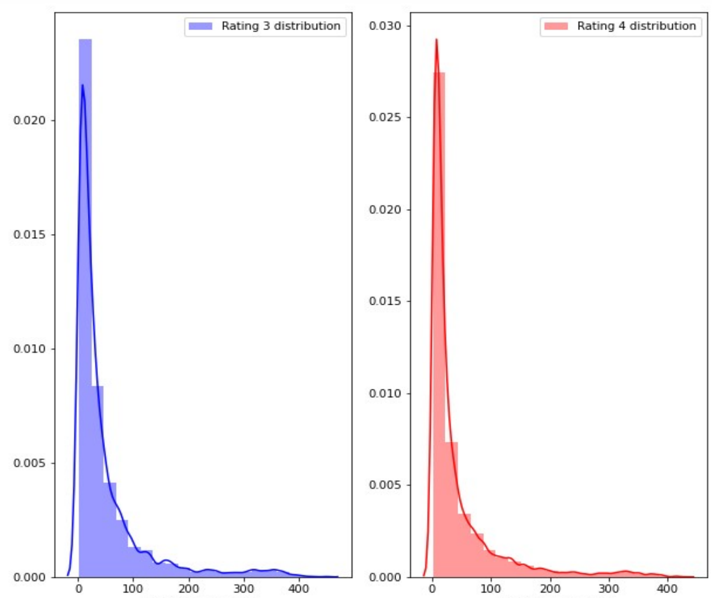


Rating 1 and Rating 2 distribution after cleaning the reviews:

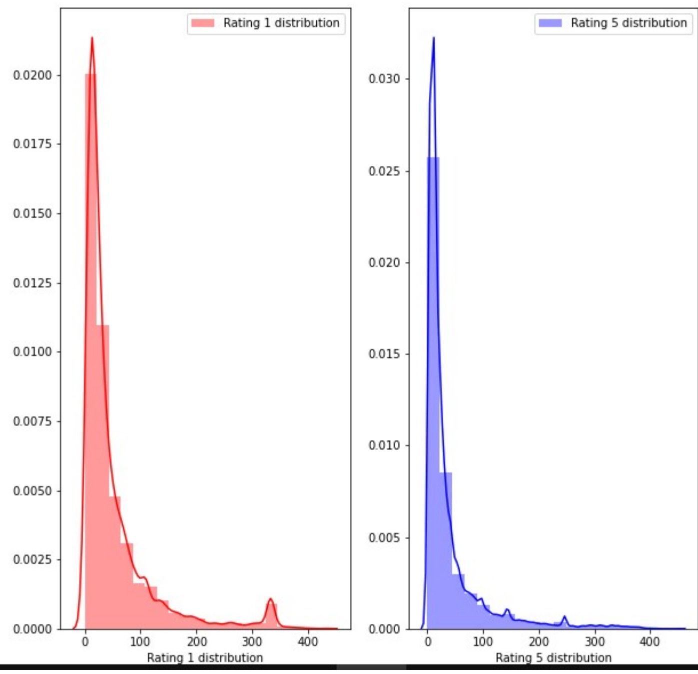
```
In [23]: #message distribution after cleaning

         f,ax = plt.subplots(1,2,figsize=(10,10))

         sns.distplot(Rating[Rating['Ratings']==1]['clean_length'],bins=20,ax=ax[0],label='Rating 1 distribution',color='r')
         ax[0].set_xlabel('Rating 1 distribution')
         ax[0].legend()

         sns.distplot(Rating[Rating['Ratings']==2]['clean_length'],bins=20,ax=ax[1],label='Rating 2 distribution',color='b')
         ax[1].set_xlabel('Rating 2 distribution')
         ax[1].legend()

         plt.show()
```




```
sns.distplot(Rating[Rating['Ratings']==3]['clean_length'],bins=20,ax=ax[0],label='Rating 3 distribution',color='b')
ax[0].set_xlabel('Rating 3 distribution')
ax[0].legend()

sns.distplot(Rating[Rating['Ratings']==4]['clean_length'],bins=20,ax=ax[1],label='Rating 4 distribution',color='r')
ax[1].set_xlabel('Rating 4 distribution')
ax[1].legend()

plt.show()
```



In [25]:
```
f,ax = plt.subplots(1,2,figsize=(10,10))

sns.distplot(Rating[Rating['Ratings']==1]['clean_length'],bins=20,ax=ax[0],label='Rating 1 distribution',color='r')
ax[0].set_xlabel('Rating 1 distribution')
ax[0].legend()

sns.distplot(Rating[Rating['Ratings']==5]['clean_length'],bins=20,ax=ax[1],label='Rating 5 distribution',color='b')
ax[1].set_xlabel('Rating 5 distribution')
ax[1].legend()

plt.show()
```

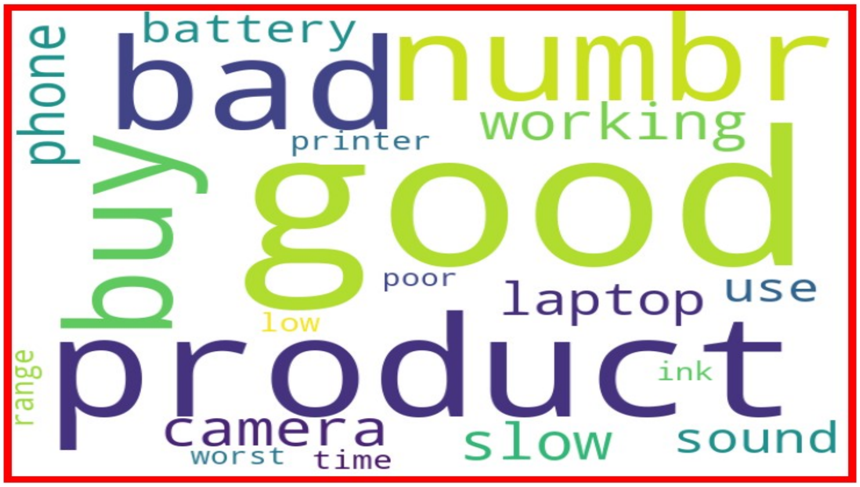


Getting sense of review Loud words in Rating 1:

In [26]:
```
#getting sense of review Loud words in Rating 1
from wordcloud import WordCloud

Rating1=Rating['Full_review'][Rating['Ratings']==1]

spam_cloud = WordCloud(width=700,height=500,background_color='white',max_words=20).generate(' '.join(Rating1))

plt.figure(figsize=(10,8),facecolor='r')
plt.imshow(spam_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

Getting sense of review Loud words in Rating 2:

```
In [27]: #getting sense of review Loud words in Rating 2

Rating2=Rating['Full_review'][Rating['Ratings']==2]

spam_cloud = WordCloud(width=700,height=500,background_color='white',max_words=20).generate(' '.join(Rating2))

plt.figure(figsize=(10,8),facecolor='r')
plt.imshow(spam_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

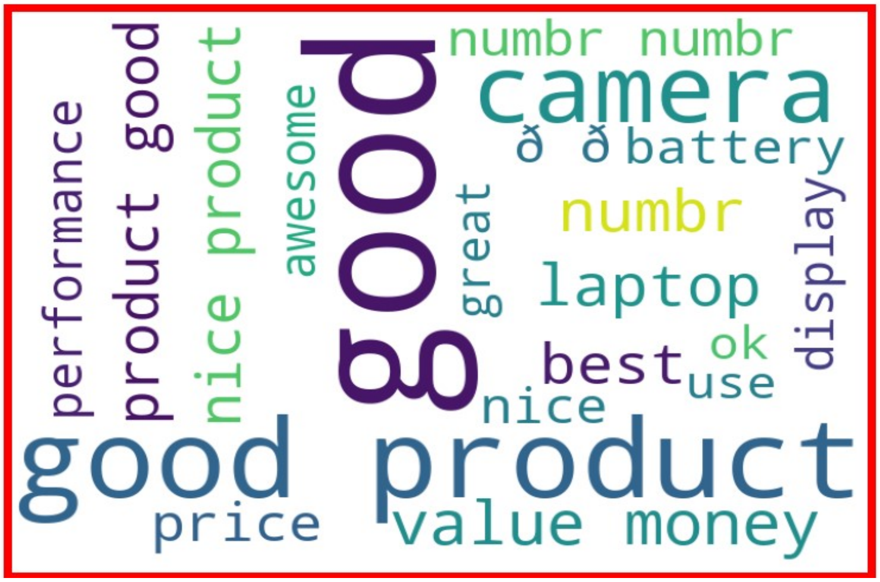


Getting sense of review Loud words in Rating 3:

```
In [28]: #getting sense of review Loud words in Rating 3

Rating3=Rating['Full_review'][Rating['Ratings']==3]

spam_cloud = WordCloud(width=700,height=500,background_color='white',max_words=20).generate(' '.join(Rating3))

plt.figure(figsize=(10,8),facecolor='r')
plt.imshow(spam_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

Getting sense of review Loud words in Rating 4:

```
In [29]: #getting sense of review Loud words in Rating 4

Rating4=Rating['Full_review'][Rating['Ratings']==4]

spam_cloud = WordCloud(width=700,height=500,background_color='white',max_words=20).generate(' '.join(Rating4))

plt.figure(figsize=(10,8),facecolor='r')
plt.imshow(spam_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```



Getting sense of review Loud words in Rating 5:

```
In [30]: #getting sense of review Loud words in Rating 5

Rating5=Rating['Full_review'][Rating['Ratings']==5]

spam_cloud = WordCloud(width=700,height=500,background_color='white',max_words=20).generate(' '.join(Rating5))

plt.figure(figsize=(10,8),facecolor='r')
plt.imshow(spam_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

FINAL MODEL

```
In [40]: result = pd.DataFrame({'Model': Model, 'Accuracy_score': score,'Cross_val_score': cvs})
         result
```

Out[40]:

	Model	Accuracy_score	Cross_val_score
0	KNeighborsClassifier	32.778766	54.034738
1	DecisionTreeClassifier	52.298705	56.897601
2	XGBClassifier	52.819406	63.336622
3	RandomForestClassifier	55.613411	63.138502
4	AdaBoostClassifier	42.684785	61.995473
5	MultinomialNB	49.453899	62.328237
6	GradientBoostingClassifier	47.129794	61.959915
7	BaggingClassifier	52.908306	59.966022
8	ExtraTreesClassifier	55.930912	62.356178

Using gridsearch cv to find the best parameters in random forest

```
In [41]: from sklearn.model_selection import GridSearchCV

         parameters={'max_depth': [80, 90, 100], 'min_samples_leaf': [3, 4, 5], 'min_samples_split': [8, 10, 12], 'n_estimators': [100, 2(
         rfc=RandomForestClassifier()

         clf=GridSearchCV(rfc,parameters,cv=5,n_jobs=-1)
         clf.fit(x_train_ns,y_train_ns)
         print(clf.best_params_)

         {'max_depth': 100, 'min_samples_leaf': 3, 'min_samples_split': 8, 'n_estimators': 300}
```

```
In [42]: #RandomForesetClassifier with best parameters

         rfc=RandomForestClassifier(max_depth=100, min_samples_leaf=3, min_samples_split=8, n_estimators=1000)
         rfc.fit(x_train_ns,y_train_ns)
         rfc.score(x_train_ns,y_train_ns)
         predrfc=rfc.predict(x_test)
         print(accuracy_score(y_test,predrfc))
         print(confusion_matrix(y_test,predrfc))
         print(classification_report(y_test,predrfc))
```

```
0.5486410972821946
[[ 839   96   52   51   75]
 [ 153   62   49   37   26]
 [ 111   70  190  189  121]
 [  64   37  177  675  519]
 [ 100   56  224 1347 2554]]
              precision    recall  f1-score   support

           1       0.66      0.75      0.71      1113
           2       0.19      0.19      0.19       327
           3       0.27      0.28      0.28       681
           4       0.29      0.46      0.36      1472
           5       0.78      0.60      0.67      4281

    accuracy                           0.55      7874
   macro avg       0.44      0.46      0.44      7874
weighted avg       0.60      0.55      0.57      7874
```

KEY METRICS FOR SUCCESS IN SOLVING PROBLEM UNDER CONSIDERATION

- When it comes to the evaluation of a data science model's performance, sometimes accuracy may not be the best indicator.
- Some problems that we are solving in real life might have a very imbalanced class and using accuracy might not give us enough confidence to understand the algorithm's performance.
- In the Rating Prediction problem that we are trying to solve, the data is balanced. So accuracy score nearly tells the right predictions. So the problem of overfitting in this problem is nearly not to occur. So here, we are using an accuracy score to find a better model.

CONCLUSION

KEY FINDINGS AND CONCLUSIONS OF THE STUDY

In this project we have tried to detect the Ratings in commercial websites on a scale of 1 to 5 on the basis of the reviews given by the users. We made use of natural language processing and machine learning algorithms in order to do so. We interpreted that Random forest classifier model is giving us best results.

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 Stopwords.
This project has demonstrated the importance of sampling effectively, modelling and predicting data.
Through different powerful tools of visualization we were able to analyses and interpret different hidden insights about the data.
The few challenges while working on this project are:-

- Imbalanced dataset
- Lots of text data

The dataset was highly imbalanced so we balanced the dataset using smote technique. We converted text data into vectors with the help of tfidf vectorizer.

LIMITATIONS OF THIS WORK AND SCOPE FOR FUTURE WORK

While we couldn't reach out goal of maximum accuracy in Ratings prediction project, we did end up creating a system that can with some improvement and deep learning algorithms get very close to that goal. As with any project there is room for improvement here. The very nature of this project allows for multiple algorithms to be integrated together as modules and their results can be combined to increase the accuracy of the final result. This model can further be improved with the addition of more algorithms into it. However, the output of these algorithms needs to be in the same format as the others. Once that condition is satisfied, the modules are easy to add as done in the code. This provides a great degree of modularity and versatility to the project.


thank you