RATINGS PREDICTION PROJECT

Submitted by:

Rajashri Sadafule Darveshi

ACKNOWLEDGMENT

I would like to express my special thanks of gratitude to the sources Medium, Towards Data Science, Stack Overflow, Data Trained, Fliprobo,which helped me to accomplish this project.

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 number 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

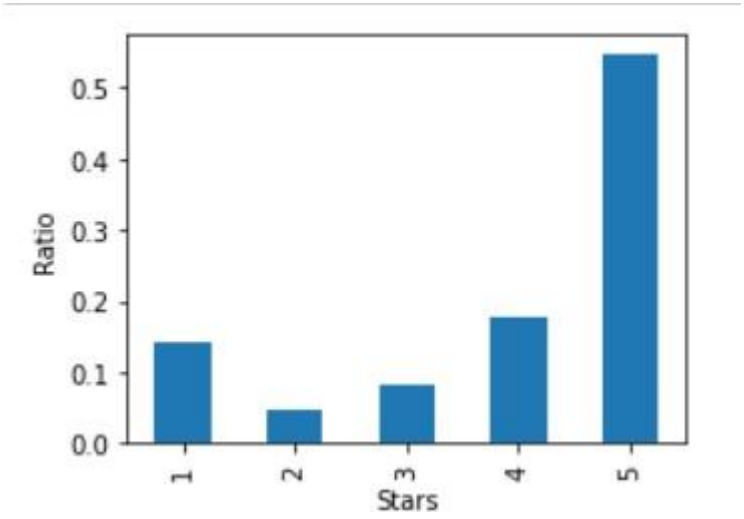
Many product reviews are not accompanied by a scale rating system, consisting only of a textual evaluation. In this case, it becomes daunting and time-consuming to compare different products to eventually make a choice between them. Therefore, models able to predict the user rating from the text review are critically important. Getting an overall sense of a textual review could in turn improve consumer experience.

Analytical Problem Framing

Mathematical/ Analytical Modelling of the Problem

There are in total 40573 rows and 2 columns of ratings and reviews are present in our dataset.

We found the occurrence of ratings ratio as shown below,



We can observe that our dataset is quite imbalanced.

```
Rating counts
 5    22169
4     7219
1     5839
3     3376
2     1970
Name: Ratings, dtype: int64
```

Maximum, 22169 number of ratings present is of 5 star and minimum, 1970 is of 2 stars.

We then create two more columns length and clean_leangth based on the lengths of the text before and after cleaning for our analysis purpose.

	Ratings	Full_review	length	clean_length
0	5	best laptop range recieved late delivery due b...	500	337
1	5	good product used everything good also ssd slo...	271	150
2	5	awesome laptop supports many high spec games l...	96	84
3	4	price exceptionally good played far cry numbr ...	342	254
4	4	ram upgrade must useable ram numbrgb ryzen num...	502	393

Data Sources and their formats

The variable features of this problem statement are,

- **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.

Data Pre-processing 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.

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 stop words
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

Stop words 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.

Hardware and Software Requirements and Tools Used

This project was done on laptop with i5 processor with quad cores and eight threads with 8gb of ram and latest GeForce GTX 1650 GPU on Anaconda, Jupiter notebook.

The tools, libraries, and packages we used for accomplishing this project are pandas, NumPy, matplotlib, seaborn, word cloud, tiff vectorizer, smote, Grid search, joblib.

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 word cloud 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)

Pre-processing 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

Multinominal Random forest classifier

Ad boost classifier

Gradient boosting classifier

Bagging classifier

Extra trees classifier

Run and evaluate selected models

The algorithms we used are shown in fig,

```
#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
```

The results observed over different evaluation metrics are shown in fig,

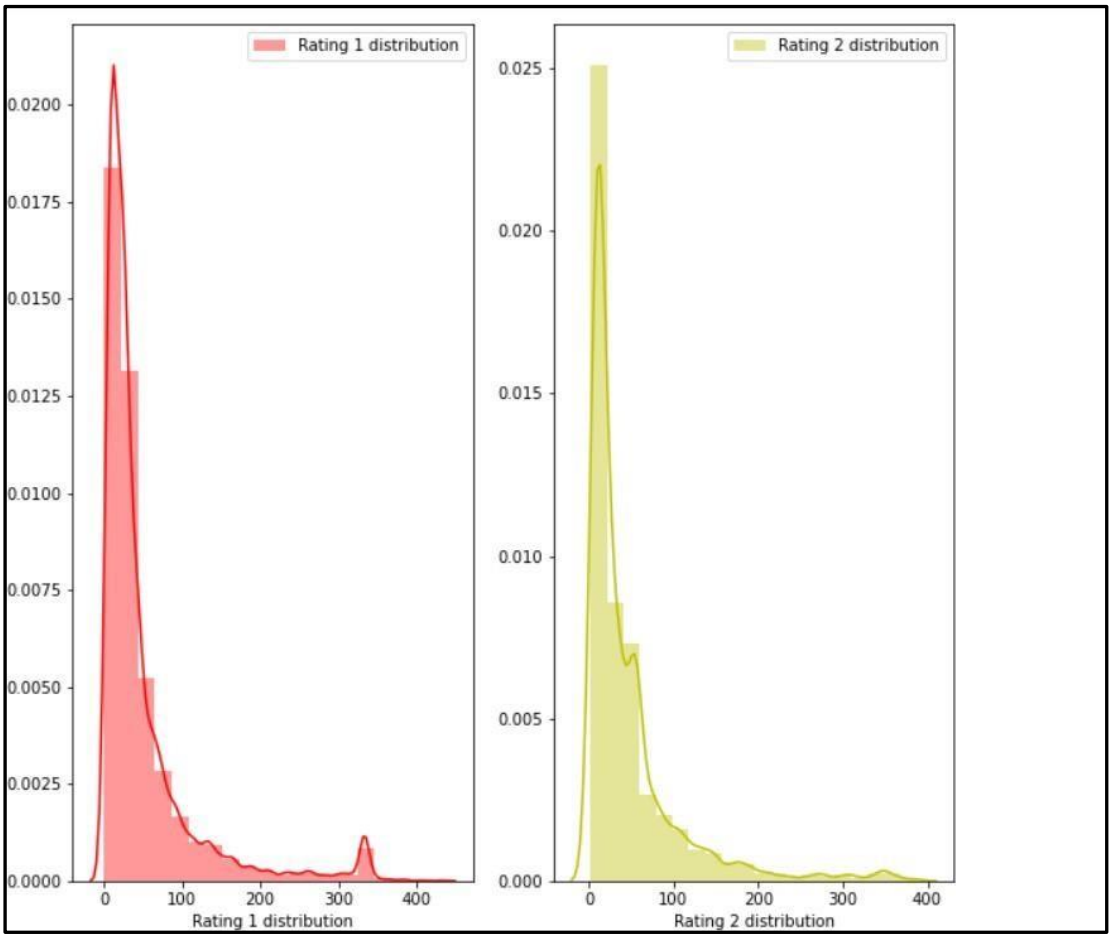
	Model	Accuracy_score	Cross_val_score
0	KNeighborsClassifier	42.772643	57.831708
1	DecisionTreeClassifier	56.759088	57.940008
2	XGBClassifier	58.311768	63.480543
3	RandomForestClassifier	60.973506	63.870033
4	AdaBoostClassifier	51.608133	62.243298
5	MultinomialNB	54.799754	62.524411
6	GradientBoostingClassifier	56.487985	63.231706
7	BaggingClassifier	57.966728	61.402893
8	ExtraTreesClassifier	60.862600	63.751793

Key Metrics for success in solving problem under consideration

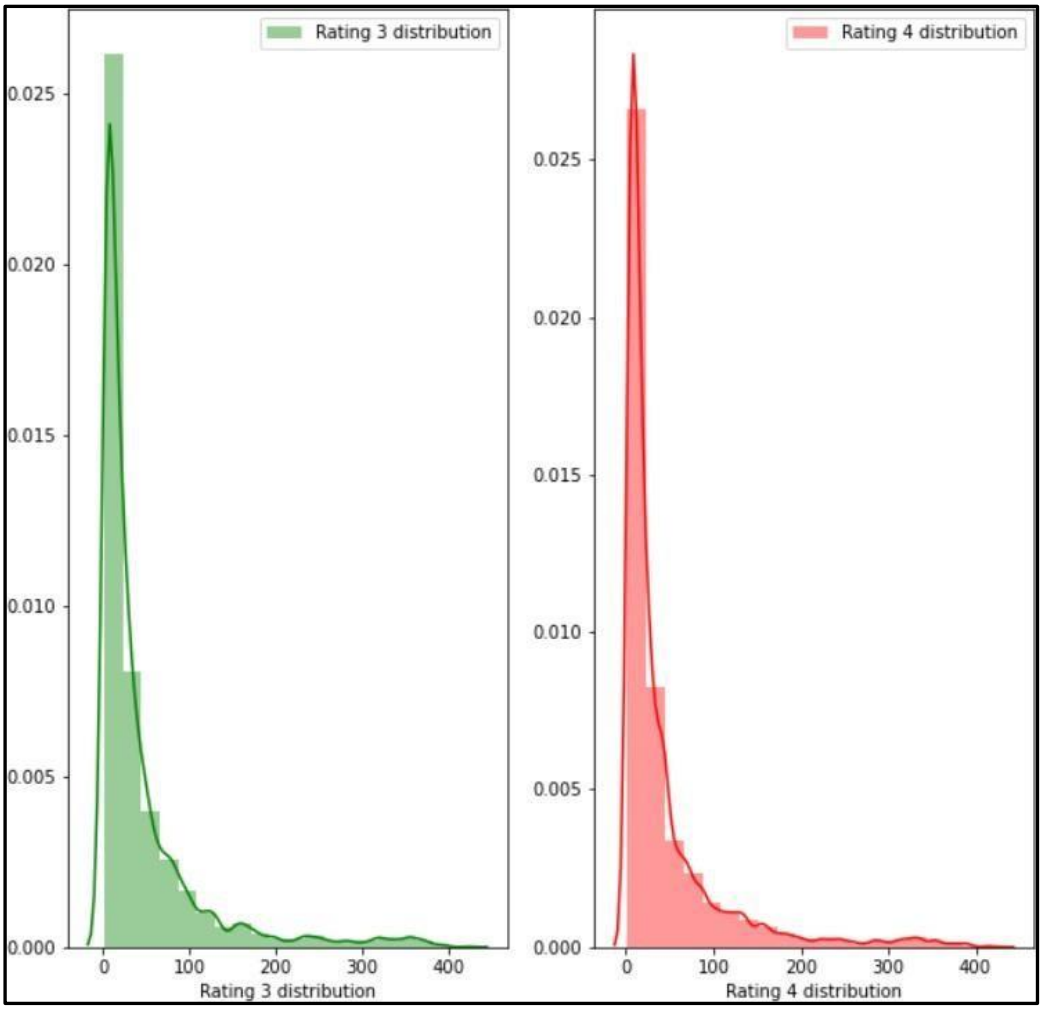
On the basis of accuracy and confusion matrix we save Random Forest classifier as our final model.

Visualizations

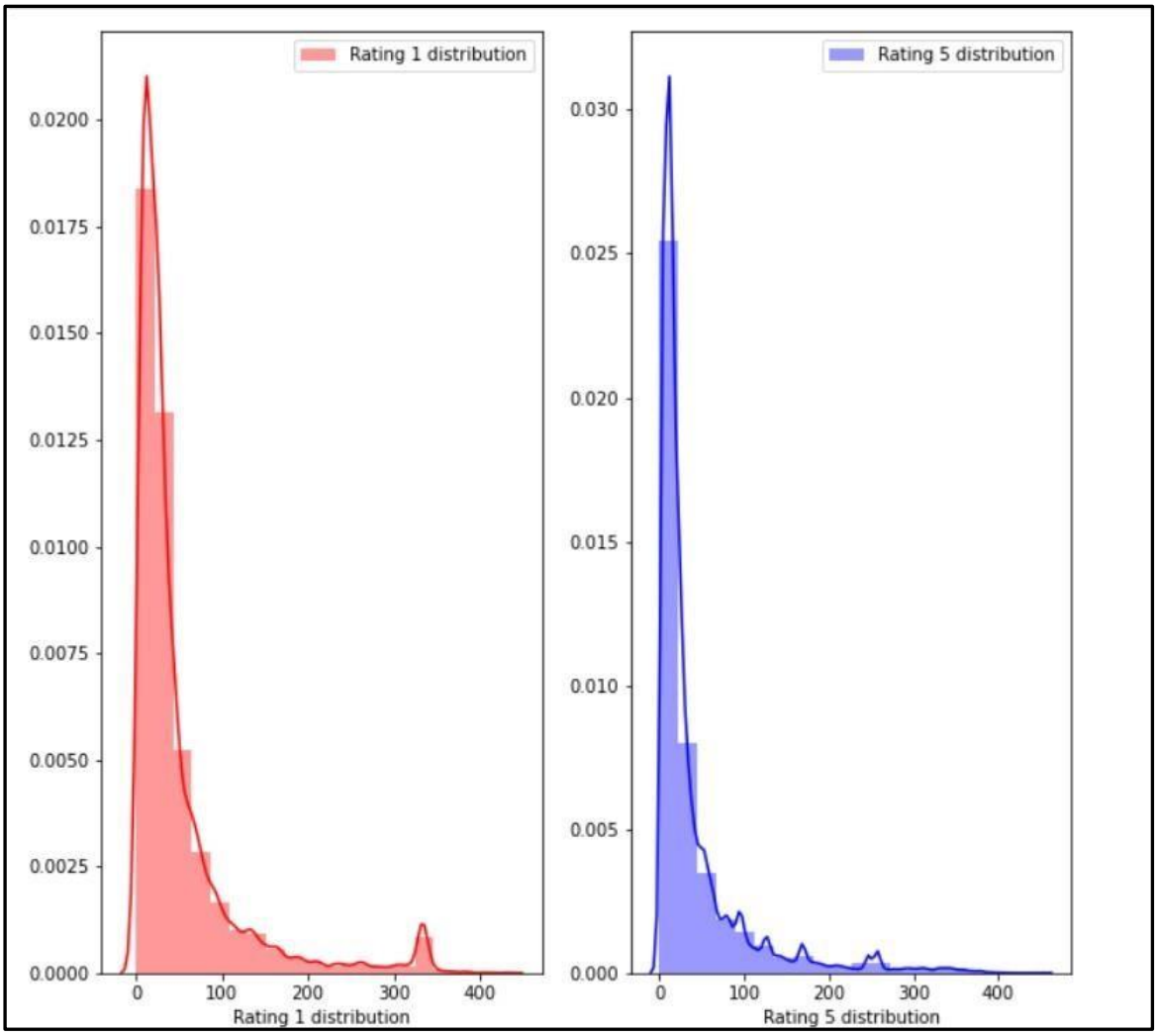
Rating 1 and Rating 2 distribution after cleaning the reviews:



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



Rating 1 and Rating 5 distribution after cleaning reviews:



getting sense of review Loud words in Rating 1:

getting sense of review Loud words in Rating 2:

getting sense of review Loud words in Rating 3:

getting sense of review Loud words in Rating 4:

getting sense of review Loud words in Rating 5:

Comparision of all Algorithm

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

```
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))
```

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

	Model	Accuracy_score	Cross_val_score
0	KNeighborsClassifier	42.772643	57.831708
1	DecisionTreeClassifier	56.759088	57.940008
2	XGBClassifier	58.311768	63.480543
3	RandomForestClassifier	60.973506	63.870033
4	AdaBoostClassifier	51.608133	62.243298
5	MultinomialNB	54.799754	62.524411
6	GradientBoostingClassifier	56.487985	63.231706
7	BaggingClassifier	57.966728	61.402893
8	ExtraTreesClassifier	60.862600	63.751793

Interpretation of the Results

We interpreted that Random Forest classifier model was giving us the best results with the accuracy score of 60.97 and comparatively better f1-score so we saved it as our final model.

```
#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))
```

```
0.5897720271102895
[[ 869  135   75   44   86]
 [ 156  125   62   28   41]
 [  86   47  205  209  131]
 [  62   32  187  640  517]
 [ 107   53  200 1071 2947]]
              precision    recall  f1-score   support

           1       0.68      0.72      0.70      1209
           2       0.32      0.30      0.31       412
           3       0.28      0.30      0.29       678
           4       0.32      0.45      0.37      1438
           5       0.79      0.67      0.73      4378

    accuracy                           0.59      8115
   macro avg       0.48      0.49      0.48      8115
weighted avg       0.62      0.59      0.60      8115
```

fig. Confusion matrix of Random forest classifier

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 based on the reviews given by the users. We made use of natural language processing and machine learning algorithms 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

Through 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 analyse and interpret different hidden insights about the data.

The few challenges while working on this project where: -

- 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 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