Rating Prediction Project

Submitted by:

Shama Tanweer

Internship-12

ACKNOWLEDGMENT

I would like to express my sincere gratitude to FlipRobo Technologies for supporting me throughout the internship and giving me the opportunity to explore the depth of Data Science by providing multiple projects like this.

Separately, I would like to thank:

- SME khushboo Garg
- Data Trained Team

I have taken help from following sites whenever stuck:

- TF-IDF Vectorizer scikit-learn. Deep understanding TfidfVectorizer by… | by Mukesh Chaudhary | Medium
- https://stackoverflow.com/questions/66097701/how-can-i-fix-this-warning-in-xgboost

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 wriiten 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 GeneratedContents (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

- **Motivation for the Problem Undertaken**

The exposure to real world data and the opportunity to deploy my skills in solving a real time problem has been the primary objective. 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. The main motivation is to build a prototype models able to predict the user rating from the text review. Getting an overall sense of a textual review could in turn improve consumer experience

Analytical Problem Framing

Data Collection Phase

We have to scrape at least 20000 rows of data.More the data better the model .In this section we scrape the reviews of different laptops, Phones, Headphones, smart watches, Professional Cameras, Printers, monitors, Home theater, router from amazon and flipkart e-commerce websites. Basically, we need these columns: 1) reviews of the product. 2) rating of the product.we try to fetch an equal number of reviews for each rating. It will balance our data set.

• Mathematical/ Analytical Modeling of the Problem

- Final Dataset after has a shape of (99779, 4).
- 'Unnamed: 0','Unnamed: 0.1' are dropped as it is of no use.
- Dataset was highly imbalanced with most number of ‘5’ ratings and least number of ‘2’ ratings
- Here we will use NLP techniques like word tokenization, lemmatization and tfidf vectorizer then those processed data will be used to create best

model using various classification based supervised machine learning algorithms like , Multinomial NB, Random Forest Classifier etc

- Dataset contains few null value which were removed.

Data Sources and their formats

- The data is obtained by scrapping data from e-commerce website like Flipkart and Amazon. The scrapper was written in Python programming language using the Selenium package.
- The dataframe was saved in csv format.
- The sample data for reference is shown below.

Data Acquisition

```
df1=pd.read_csv('amazon_flipkart_reviews.csv')
df1
```

	Unnamed: 0	Unnamed: 0.1	reviews	ratings
0	0	0.0	Disclaimer: I am an I.T. network person at a l...	5
1	1	1.0	I bought the NETGEAR AX6600 to upgrade my old ...	1
2	2	2.0	I bought this to upgrade my older nighthawk r8...	1
3	3	3.0	This router replaced a great R7800 router whic...	5
4	4	4.0	Router #1 I received from the pre-order had a ...	1
...	...	...	...	...
99774	99774	NaN	Overall Mobile is excellent...to our childrens...	5
99775	99775	NaN	Good smartphone for medium usage.	5
99776	99776	NaN	It's batter to buy stock android over ads and ...	5
99777	99777	NaN	Nice phone at given price..	5
99778	99778	NaN	Nice mobile.. purchased for my mother	5

99779 rows × 4 columns

DataSet description

There are 4 columns in the dataset provided:
The description of each of the column is given below:

- Unnamed:0 & unnamed:0.1 is serial number
- reviews: it contains reviews given by customer in the past after shopping.
- rating: The rating is out 5 stars and it only has 5 options available 1 star, 2 stars, 3 stars, 4 stars, 5 stars.

• Identification of possible problem-solving approaches (methods)

Used the following process for problem solving
1. Data Preprocessing
2. Building word dictionary
3. Feature extraction
4. Training classifiers
5. Testing
6. Performance evaluation using multiple metrics

Data Pre-processing

Data usually comes from a variety of source & is often inconsistent,inaccurate.Data preprocessing helps to enhance the quality of data and make it ready for various ML model.We have applied various methods for data preprocessing methods in this project .

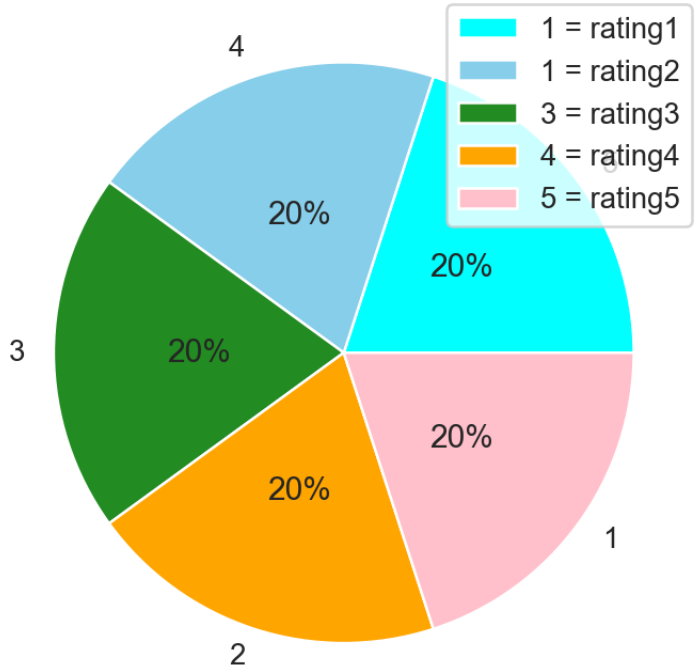
- First we check shape by using(df1.shape)
- Then checked datatype of various features & found that all features are of object type

Dropping features

```
# Let's drop Unnamed: 0  & unnamed:0.1 from dataset as it does not seem important
df1.drop(['Unnamed: 0','unnamed:0.1'],axis=1,inplace=True)
```

Checking distribution of all rating

```
lb=df['ratings'].value_counts().index.tolist()
val=df['ratings'].value_counts().values.tolist()
exp=(0.025,0)
clr=('cyan','skyblue',‘forestgreen','orange','pink)
plt.figure(figsize=(10,6),dpi=140)
sns.set_context('talk',font_scale=0.4)
sns.set(style='whitegrid')
plt.pie(x=val,explode=exp,labels=lb,colors=clr,autopct='%2.0f%%',pctdis
tance=0.5, shadow=True,radius=0.9)
plt.legend(["1 = rating1",'1 = rating2','3 = rating3','4 = rating4','5
= rating5'])
plt.show()
```



Previously the data was imbalanced with highest number of ‘5’ rating.We made it equally ditributed by dropping rows

- Cleaning the raw data-It involves deletion of words or special characters that do not add meaning to the text.
  - Important cleaning steps are:
    1. Lowering case
    2. Handling of special characters
    3. Removal of stopwords
    4. Handling of hyperlinks

5. Removing leading and trailing white space
6. Replacing urls with web address
7. Converted words to most suitable base form by using lemmatization

```
# function to filter using POS tagging. This will be called inside the
below function
def get_pos(pos_tag):
    if pos_tag.startswith('J'):
        return wordnet.ADJ
    elif pos_tag.startswith('N'):
        return wordnet.NOUN
    elif pos_tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN


# Function for data cleaning.
def Processed_data(Review):
    # Replace email addresses with 'email'
    Review =re.sub(r'^.+@[^\.].*\.[a-z]{2,}$',' ', Review)

    # Replace 10 digit phone numbers (formats include paranthesis, spac
es, no spaces, dashes) with 'phonenumber'
    Review =re.sub(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$',' ', Re
view)

    # getting only words(i.e removing all the special characters)
    Review = re.sub(r'[^\w]', ' ', Review)

    # getting only words(i.e removing all the" _ ")
    Review = re.sub(r'[\_]', ' ', Review)

    # getting rid of unwanted characters(i.e remove all the single char
acters left)
    Review =re.sub(r'\s+[a-zA-Z]\s+', ' ', Review)
    # Removing extra whitespaces
    Review =re.sub(r'\s+', ' ', Review, flags=re.I)

    #converting all the letters of the review into lowercase
    Review = Review.lower()

    # splitting every words from the sentences
    Review = Review.split()

    # iterating through each words and checking if they are stopwords o
r not,
    Review =[word for word in Review if not word in set(STOPWORDS)]

    # remove empty tokens
    Review = [text for text in Review if len(text) > 0]

    # getting pos tag text
    pos_tags = pos_tag(Review)

    # considering words having length more than 3only
    Review = [text for text in Review if len(text) > 3]
```

```
    # performing lemmatization operation and passing the word in get_pos function to get filtered using POS
    Review = [(WordNetLemmatizer().lemmatize(text[0], get_pos(text[1])))for text in pos_tags]


    Review = ' '.join(Review)
    return Review
```

For Data pre-processing we did some data cleaning, where we used wordNet lemmatizer to clean the words and removed special characters using Regexp Tokenizer and filter the words by removing stop words and then used lemmatizers and joined and return the filtered words.
Used TFIDF vectorizer to convert those text into vectors, and split the data and into test and train and trained various Machine learning algorithms.

Adding additional attribute :

To compare the length of reviews before preprocessing and after preprocessing an addition column was added:

```
#New columns for length of reviews
df['length_reviews'] = df.reviews.str.len()
```

```
#Adding New columns for processed reviews
df['clean_reviews']=df['reviews'].apply(Processed_data)
```

```
#New columns for length of reviews after preprocessing

df['clean_reviews_length']=df.clean_reviews.str.len()
```

```
print ('Origian Length', df.length_reviews.sum())
print ('Clean Length', df.clean_reviews_length.sum())
print('Total Reduction = ',df['length_reviews'].sum()-df['clean_reviews_length'].sum())
```

```
Origian Length 12847468
Clean Length 6451443
Total Reduction =  6396025
```

After executing all these steps it was found that all the words & special characters were removed from the dataset which were of no use and consuming memory

- ## Data Inputs- Logic- Output Relationships

The input data was raw and object type, this data was transformed to the format fit for a machine learning model , using various steps of preprocessing described above. The logic is

simply converting all important words of the input to a vector form and then using it to train the model.

Hardware and Software Requirements and Tools Used:

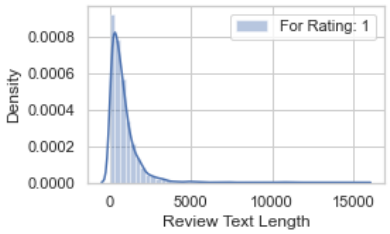
- Framework-Annconda
- IDE-Jupyter –Notebook
- Coding Language-Python
- Hardware used: system memory 8GB,
- Processor: core i3

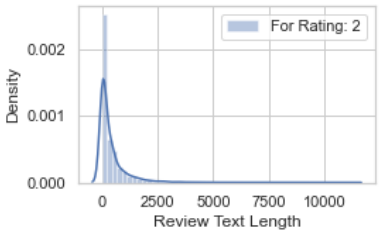
Libraries used: Below are the python library used in this project. •

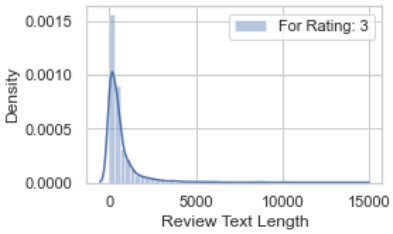
- Pandas: To read the Data file in form of data.
- Matplotlib: This library is typically used to plot the figures for better visualisation of data.
- Seaborn: A advanced version of Matplotlib
- Scikit Learn: This is the most important library for Machine Learning since it contains various Machine Learning Algorithms which are used in this project. Scikit Learn also contains Preprocessing library which is used in data preprocessing. Apart from this it contains very useful joblib library for serialization purpose using which final model have been saved in this project.
- NLTK: Natural language took kit is one of the most used library for building NLP projects.

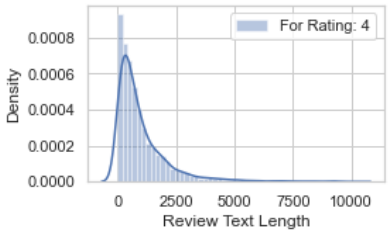
Then we have plotted graph to show distribution of word count before cleaning and after cleaning

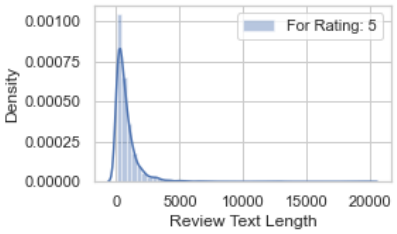
Before cleaning:



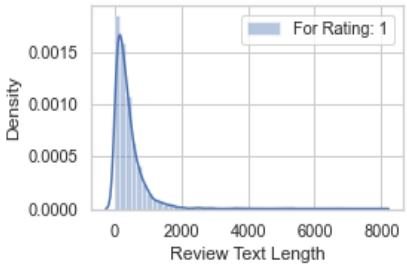


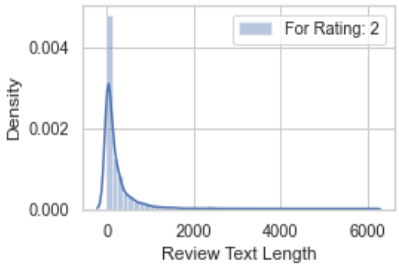


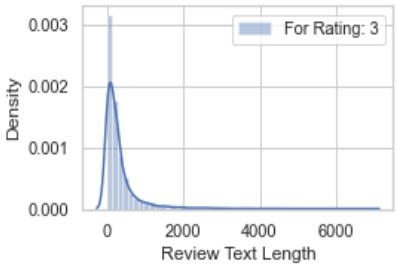


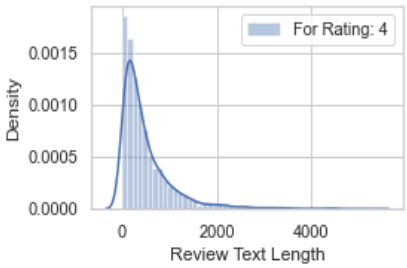


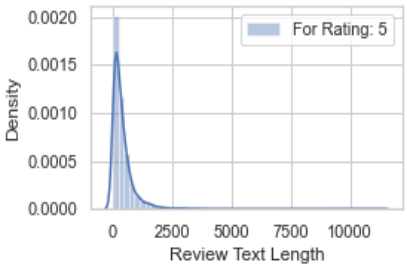
After cleaning









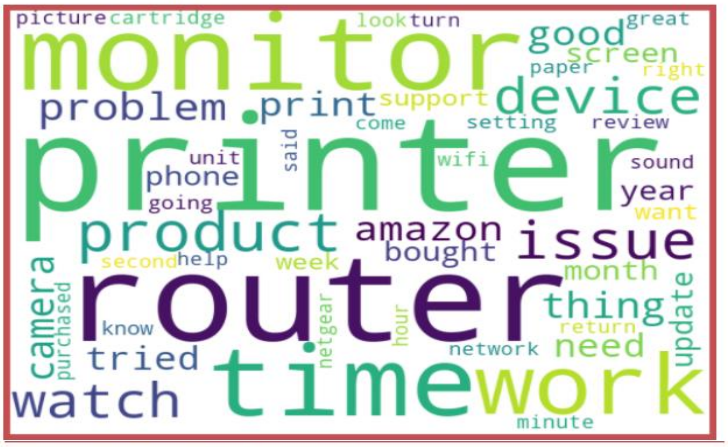


To get better view of words contained in reviews , a word dictionary (wordcloud) was made showing the words highly occurred in all reviews according to rating given by customer .

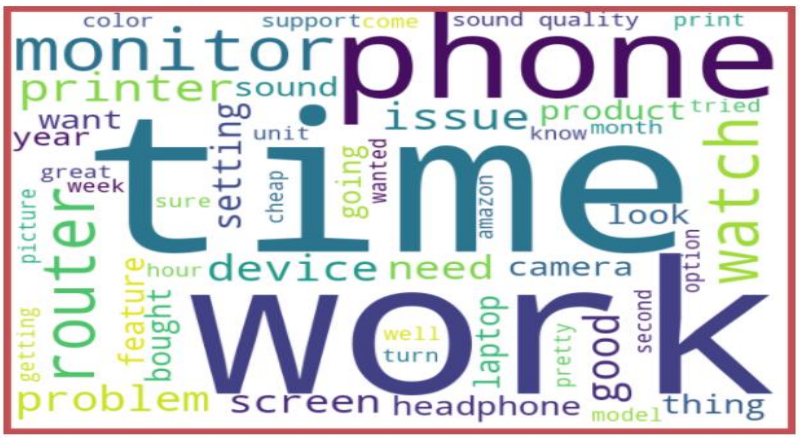
```
#Getting sense of loud words in all reviews accord
ing to rating

for i in [1,2,3,4,5]:
    print('******************Rating',i,'************************')

    reviews = df['clean_reviews'][df['ratings']==i]

    reviews_cloud = WordCloud(width=700,height=500,background_color='wh
ite',max_words=50).generate(' '.join(reviews))

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

```
plt.show()
```

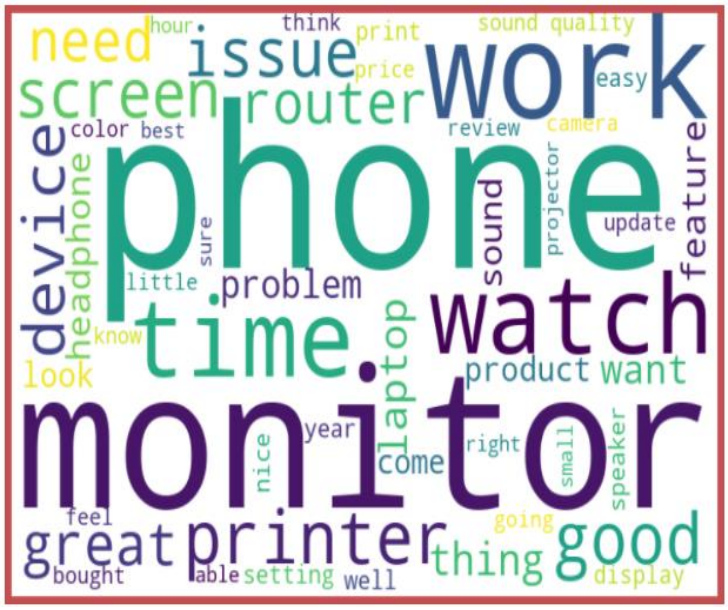
**********Rating 1 **********



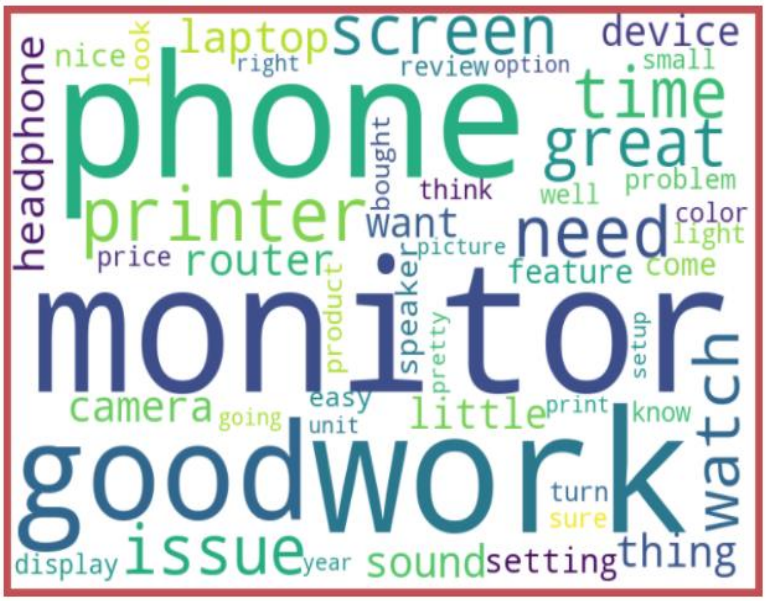
**********Rating 2 **********



**********Rating 3 **********



**********Rating 4 **********



**********Rating 5 **********

➢ State the set of assumptions (if any) related to the problem under consideration

Only assumptions which were taken related to the problem was that we dropped the **unnamed:0 & unnamed:0.1** as it was not important

Model/s Development and Evaluation

➢ problem-solving approaches:

Understanding the problem is the first crucial steps in solving any problem. From the dataset it can be concluded that it's a multiclass classification problem containing 5 classes . To choose the best model I decided to run my preprocessed dataset on 5 classification algorithms

Training Classifier:

We converted all the text into vectors , using TF-IDF. Then we have split features and label.

```
# 1. Convert text into vectors using TF-IDF
# 3. Split feature and label
tf_vec = TfidfVectorizer()
features = tf_vec.fit_transform(df['clean_reviews'])

X = features
y = df['ratings']
X.shape
```

```
(17743, 25040)
```

Testing of Identified Approaches (Algorithms)

- **Below are the algorithms used for this problem:**

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

# Ensemble Techniques...

from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import AdaBoostClassifier
```

Run and Evaluate selected models

In my approach I have first prepared a method which gives all necessary classification metrics of an algorithm like classification metrics , accuracy_score, confusion matrix.

```
# Finding best Random State and then calculate Maximum Accuracy Score
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=.3
0,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
```

```
Model=[]
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,r
andom_state=r_state,stratify=y)
    model.fit(x_train,y_train)
#..............Accuracy Score...........

    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='roc_auc').mean()
    print('Cross Val Score : ', cv_score)
    cvs.append(cv_score*100)


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

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

	Model	Accuracy_score	Cross_val_score
0	MultinomialNB()	62.295698	50.168912
1	DecisionTreeClassifier	55.495022	44.048240
2	RandomForestClassifier	65.865114	54.069695
3	AdaBoostClassifier	51.643810	41.601696
4	XGBClassifier	64.888221	51.448494

Among all the results we got from all the algorithm used using the for loop above, RandomForest was working best since it was giving highest accuracy score of 66% .Below was the results I got for RandomForest

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


RandomForestClassifier()


Max Accuracy Score corresponding to Random State  85 is: 0.6678564719143341


Learning Score :  0.9878421900161031
Accuracy Score :  0.6586511365771182
Cross Val Score :  0.5406969528558043
Classification Report:
              precision    recall  f1-score   support

           1       0.62      0.78      0.69      1065
           2       0.78      0.62      0.69      1065
           3       0.70      0.47      0.56      1065
           4       0.57      0.58      0.58      1064
           5       0.66      0.84      0.74      1064

    accuracy                           0.66      5323
   macro avg       0.67      0.66      0.65      5323
weighted avg       0.67      0.66      0.65      5323
```

I then decided to tune the parameters of RandomForest to see if there exist more suitable combination of precision and Recall.

```
{'max_depth': 17, 'max_features': 'auto', 'n_estimators': 2000}
0.5659420289855073
```

```
#Using the best parameters obtained
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.30,random_state=85,stratify=y)
rfc=RandomForestClassifier(max_depth=17,max_features='auto',n_estimators=2000)
rfc.fit(x_train,y_train)
rfc_pred=rfc.predict(x_test)
print("Accuracy score: ",accuracy_score(y_test,rfc_pred)*100)
print('Cross validation score: ',cross_val_score(rfc,X,y,cv=3,scoring='accuracy').mean()*100)
print('Classification report: \n')
print(classification_report(y_test,rfc_pred))
print('Confusion matrix: \n')
print(confusion_matrix(y_test,rfc_pred))
```

```
Accuracy score:  59.233514935186925
Cross validation score:  40.5632159954368
Classification report:

              precision    recall  f1-score   support

           1       0.62      0.68      0.65      1065
           2       0.48      0.70      0.57      1065
           3       0.69      0.34      0.45      1065
           4       0.60      0.49      0.54      1064
           5       0.66      0.76      0.70      1064

    accuracy                           0.59      5323
   macro avg       0.61      0.59      0.58      5323
weighted avg       0.61      0.59      0.58      5323

Confusion matrix:

[[722 201  11  43  88]
 [145 747  46  85  42]
 [147 299 360 172  87]
 [120 151  76 520 197]
 [ 37 147  25  51 804]]
```

Since it didn’t show any improvement in result even after hyperparameter tuning. Hence, we will be choosing theRandomForestClassifier Model with default values that we got earlier.

Final Model

```
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.30,random_state=85,stratify=y)
RFC=RandomForestClassifier()
RFC.fit(x_train,y_train)
RFC_pred=RFC.predict(x_test)
print("Accuracy score: ",accuracy_score(y_test,RFC_pred)*100)
print('Cross validation score: ',cross_val_score(RFC,X,y,cv=3,scoring='accuracy').mean()*100)
print('Classification report: \n')
print(classification_report(y_test,RFC_pred))
print('Confusion matrix: \n')
print(confusion_matrix(y_test,RFC_pred))
```

```
Accuracy score:  65.95904565094871
Cross validation score:  43.696927492614385
Classification report:

              precision    recall  f1-score   support

           1       0.61      0.79      0.69      1065
           2       0.80      0.62      0.70      1065
           3       0.70      0.48      0.57      1065
           4       0.57      0.57      0.57      1064
           5       0.67      0.84      0.75      1064

    accuracy                           0.66      5323
   macro avg       0.67      0.66      0.65      5323
weighted avg       0.67      0.66      0.65      5323

Confusion matrix:

[[843  47  20  63  92]
 [172 659  86 104  44]
 [168  85 507 208  97]
 [151  20  79 607 207]
 [ 43  16  28  82 895]]
```

Saving the best model

```
# Printing predicted values
pred_value=pd.DataFrame(data=y_test)
pred_value['Predicted values']=RFC_pred
pred_value
```

```

```

```
# Saving the best model.
import joblib
```

```
joblib.dump(RFC,'rating_Predict.pkl')
```

```
['rating_Predict.pkl']
```

```
# Saving the Predicted values in csv file
pred_value.to_csv('rating_Prediction.csv')
```

- **Key Metrics for success in solving problem under consideration**

Key metrics used for finalising the model was accuracy_score, cross_val_score. Since in case of RandomForestClassifier it was givingus maximum score among all other models and it's performing well. It's cross_val_score is also satisfactory and it shows that our model is neither underfitting/overfitting.

CONCLUSION

➢ Key Findings and Conclusions of the Study

Dataset was quite imbalanced with very high number 5 ratings in comparison to other ratings. Raw data contained many unwanted text. .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 65% with Random forest classifier.

➢ Learning Outcomes of the Study in respect of Data Science

I found visualisation a very useful technique to infer insights from dataset especially in case of large datasets.
Use of TF-IDF was very helpful in text to numeric conversion

➢ Limitations of this work and Scope for Future Work

Although I have performed all necessary steps of data cleaning I was able to get 65% of accuracy . 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. So more data will definitely help in achieving high accuracy

Thankyou