Rating Prediction Project

Submitted by:

Ambika Saraf

ACKNOWLEDGMENT

I have taken efforts in this project however it would not have been completed without guidance from other people. I warmly acknowledge the invaluable supervision and an inspired guidance by our SME Mr. Keshav Bansal, FlipRobo Technology.

I would also like to express my sincere thanks to Data trained Education and FlipRobo Technology for giving me an opportunity to work on this project.

I also want to express my gratitude towards my friends and family who have patiently extended all sorts of help for accomplishing this.

I am grateful to one and all who are directly or indirectly involved in successful completion of this project.

INTRODUCTION

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

Data Collection Phase

You have to scrape at least 20000 rows of data. You can scrape more data as well, it's up to you. more the data better the model

In this section you need to scrape the reviews of different laptops, Phones, Headphones, smart watches, Professional Cameras, Printers, Monitors, Home theater, Router from different e-commerce websites.

Basically, we need these columns-

1) Reviews of the product.
2) Rating of the product.

I have fetched data from two websites so that our model does not overfit.

Model Building Phase

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

Follow the complete life cycle of data science. Include all the steps like-

1. Data Cleaning
2. Exploratory Data Analysis
3. Data Preprocessing
4. Model Building
5. Model Evaluation
6. Selecting the best model

ANALYTICAL FRAMING

The project begins with data collecting phase. We have scraped data from “Amazon.in” and “Flipkart.com” of various technical products like Laptop, phone, headphone and camera. The target variable here is rating provided against each review. The processing of reviews are done using NLP techniques. Our goal is to build a classification model which classifies reviews in 5 categories i.e. ratings 5 star, 4 star, 3 star, 2 star or 1 star.

In our scraped data the target variable “Ratings” is categorical variable and has 5 classes whereas, “Reviews” is feature containing text data.

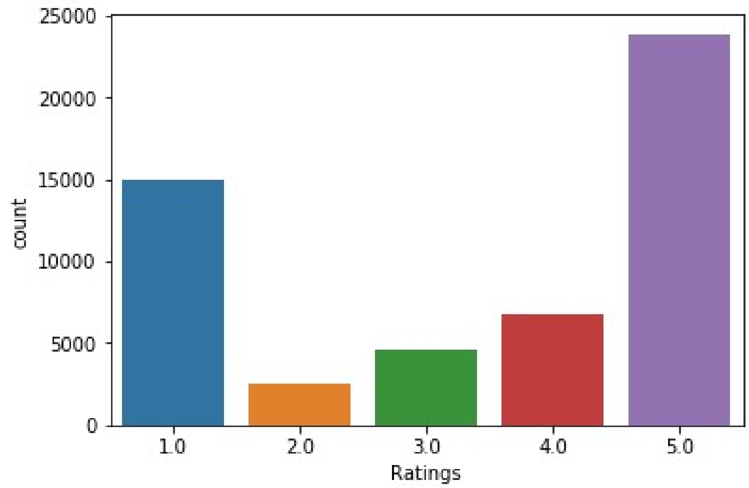
DATA SOURCES AND THEIR FORMATS

- We collected the data from difference e-commerce websites like www.Amazon.in and www.Flipkart.com . The data is scrapped using Web scraping technique and the framework used is Selenium.
- We scrapped nearly 52000 of data.
- We have created separate data frames for each product and combined all the data frames into a single data frame in the end.
- Next we have saved it into a csv file.
- Data fetched looks like as shown below:

	Unnamed: 0	Unnamed: 0.1	Ratings	Reviews
0	0	0	1.0	Laptop getting hang very much and very much sl...
1	1	1	3.0	The package i has received was well protected...
2	2	2	3.0	It looks the screen size is small . Its not lo...
3	3	3	1.0	HiThis is a true feedback, requesting you to n...
4	4	4	3.0	How to install a new SSD? The SSD does not sho...

Distribution of ratings:

5.0 :   23857
1.0 : 14995
4.0 :   6786
3.0 :   4551
2.0 :   2556



Hardware and Software Requirements and Tools Used

- Laptop with stable internet connection (Project done in jupyter notebook)
- scikit-learn
- TfidVectorizer
- nltk
- matplotlib
- pandas
- numpy

DATA PRE-PROCESSING

We will first drop unnecessary columns and since our data does not have any null values we will start with NLP techniques.

- Removing stopwords

```
## Removing Stopwords
import nltk.corpus
nltk.download('stopwords')
from nltk.corpus import stopwords
stop=stopwords.words('english')
df['Reviews']=df["Reviews"].apply(lambda x:' '.join([word for word in x.split() if word not in (stop)]))
```

- Next I have defined a function “clean_text” in which I have removed e-mail addresses, phone numbers, urls, numbers, punctuations, single white spaces and trailing white spaces. Below is the code given:

```
# defining function to clean text
def clean_text(df, text):

    #Converting all messages to lowercase
    df[text] = df[text].str.lower()

    #Replace email addresses with ' '
    df[text] = df[text].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$',' ')

    #Replace URLs with ' '
    df[text] = df[text].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$',' ')

    #Replace 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with ' '
    df[text] = df[text].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$',' ')

    #Replace numbers with 'numbr'
    df[text] = df[text].str.replace(r'\d+(\.\d+)?', ' ')

    #Remove punctuation
    df[text] = df[text].str.replace(r'[^\w\d\s]', ' ')

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

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

- Next step is lemmatization of text, Lemmatization means mapping words to its stem i.e. base word by removing suffixes or prefixes.

```
def word_lemmatizer(text):
    result=[]
    text = text.split()
    lem_text=[WordNetLemmatizer().lemmatize(i,pos='v') for i in text]
    for token in lem_text:
        if len(token)>=3:
            result.append(token)
    text=' '.join(result)
    return text

df["clean_review"]=df["Reviews"].apply(lambda x: word_lemmatizer(x))
df.head()
```

- We know the list of products of which we have scraped reviews so we will remove those words as they are not much contributing in prediction of rating but there occurrence is too many times.

- After the above steps we can observe the difference between length of text in reviews columns before and after cleaning.

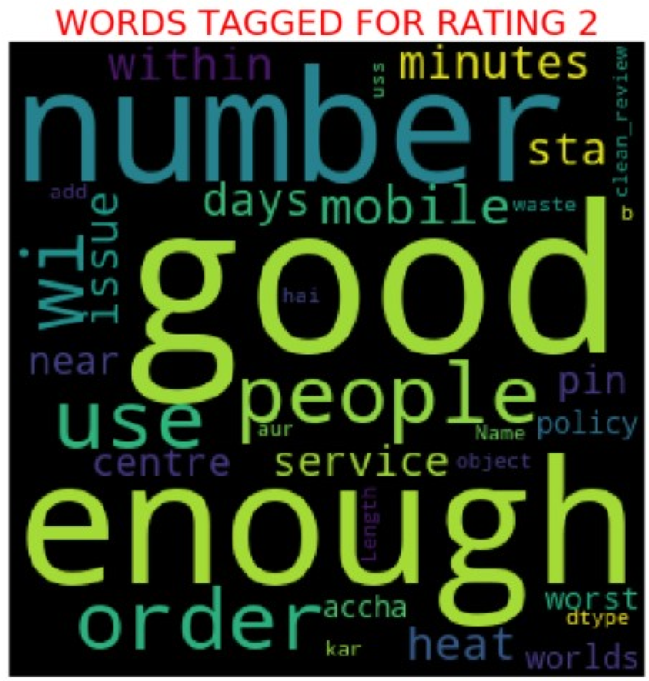
	Ratings	Reviews	review_length	clean_review	clean_text_length
0	1.0	laptop getting hang much much slow start this ...	98	get hang much much slow start this happen wi...	64
1	3.0	the package received well protected damage occ...	415	the package receive well protect damage occur ...	242
2	3.0	it looks screen size small its look inch can o...	80	it look screen size small its look inch can on...	52
3	1.0	hithis true feedback requesting buy laptop her...	429	hithis true feedback request buy here i orde...	265
4	3.0	how install new ssd the ssd show boot menu it ...	204	how install new ssd the ssd show boot menu it ...	146

PLOTTING WORD CLOUDS

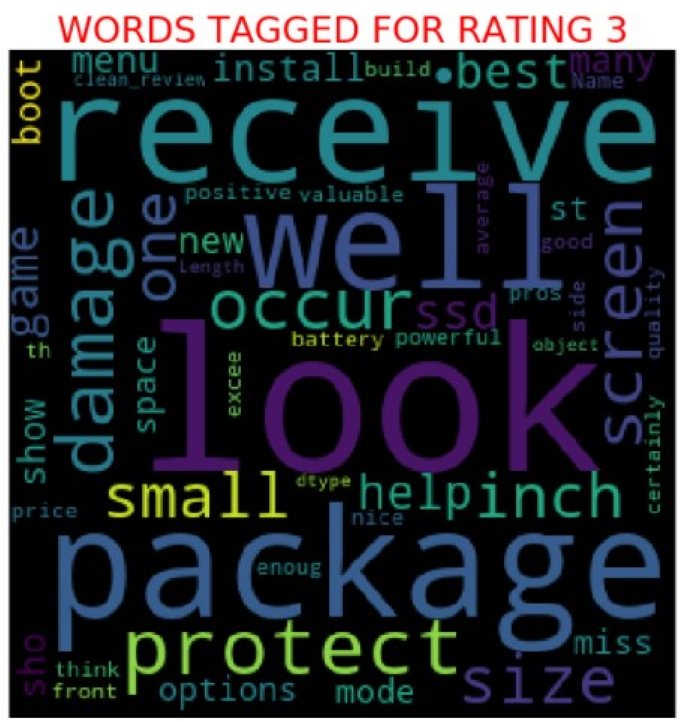
Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance.

For Rating 1

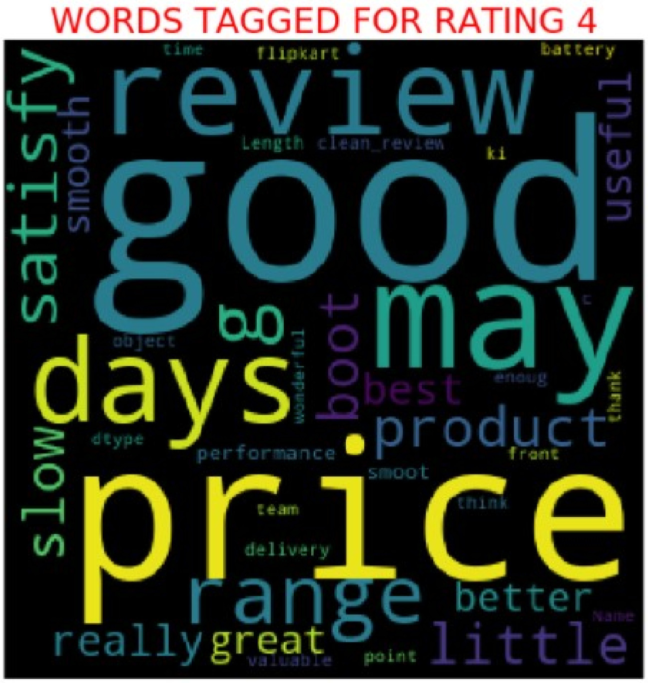
For Rating 2



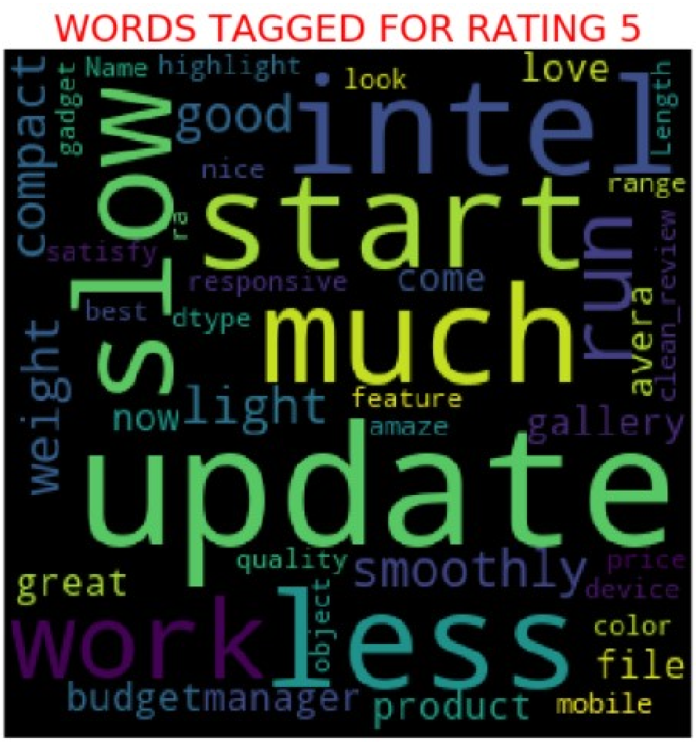
For Rating 3



For Rating 4



For Rating 5



Preparing Data for Model

Now we will extract features using TfidVectorizer (Term Frequency Inverse Document Frequency).

```
def Tf_idf_train(text):
    tfid = TfidfVectorizer(min_df=3,smooth_idf=False)
    return tfid.fit_transform(text)
x=Tf_idf_train(df['clean_review'])

print("Shape of x: ",x.shape)

y = df['Ratings'].values
print("Shape of y: ",y.shape)
```

```
Shape of x:  (52745, 3959)
Shape of y:  (52745,)
```

MODEL BUILDING AND EVALUATION

Algorithms used are:

- Logistic Regression
- Decision Tree Classifier
- KNeighbors Classifier
- Random Forest Classifier
- Gradient Boosting Classifier
- AdaBoostClassifier

Logistic Regression

```
*** Logistic Regression ***

accuracy_score:  0.778418139076363

cross_val_score:  0.46635700066357

Classification report:
              precision    recall  f1-score   support

         1.0       0.76      0.81      0.78      3803
         2.0       0.70      0.68      0.69       630
         3.0       0.77      0.63      0.69      1096
         4.0       0.69      0.54      0.60      1682
         5.0       0.82      0.87      0.84      5976

    accuracy                           0.78     13187
   macro avg       0.75      0.70      0.72     13187
weighted avg       0.78      0.78      0.77     13187
```

```
Confusion matrix:

[[3069   56   71  112  495]
 [  58  431   11   68   62]
 [ 161   23  686   63  163]
 [ 274   31   63  903  411]
 [ 497   77   64  162 5176]]
```

DecisionTree

```
*** DecisionTreeClassifier ***

accuracy_score:  0.7748540229013422

cross_val_score:  0.4303156697317281

Classification report:
              precision    recall  f1-score   support

         1.0       0.75      0.80      0.78      3803
         2.0       0.67      0.73      0.70       630
         3.0       0.71      0.70      0.71      1096
         4.0       0.66      0.61      0.63      1682
         5.0       0.85      0.82      0.83      5976

    accuracy                           0.77     13187
   macro avg       0.73      0.73      0.73     13187
weighted avg       0.78      0.77      0.77     13187
```

```
Confusion matrix:

[[3057   56   95  141  454]
 [  58  463   11   68   30]
 [ 153   23  771   56   93]
 [ 254   31   82 1030  285]
 [ 549  121  132  277 4897]]
```

KNeighbors

```
*** KNeighborsClassifier ***

accuracy_score:  0.7615833775688178

cross_val_score:  0.432325338894682

Classification report:

              precision    recall  f1-score   support

         1.0       0.74      0.78      0.76      3803
         2.0       0.70      0.67      0.68       630
         3.0       0.72      0.66      0.69      1096
         4.0       0.62      0.63      0.62      1682
         5.0       0.83      0.82      0.82      5976

    accuracy                           0.76     13187
   macro avg       0.72      0.71      0.72     13187
weighted avg       0.76      0.76      0.76     13187
```

```
Confusion matrix:

[[2964   48   79  196  516]
 [  72  423    2   59   74]
 [ 170   31  725   55  115]
 [ 238   38   71 1056  279]
 [ 573   68  123  337 4875]]
```

Random Forest

```
*** RandomForestClassifier ***

accuracy_score:  0.7743990293470843

cross_val_score:  0.48197933453407904

Classification report:

              precision    recall  f1-score   support

         1.0       0.76      0.79      0.77      3803
         2.0       0.66      0.73      0.70       630
         3.0       0.71      0.70      0.70      1096
         4.0       0.65      0.62      0.63      1682
         5.0       0.85      0.83      0.84      5976

    accuracy                           0.77     13187
   macro avg       0.73      0.73      0.73     13187
weighted avg       0.78      0.77      0.77     13187
```

```
Confusion matrix:

[[3008   64   95  155  481]
 [  50  462   11   68   39]
 [ 151   23  763   58  101]
 [ 254   31   80 1039  278]
 [ 512  116  121  287 4940]]
```

AdaBoost

```
*** AdaBoostClassifier ***

accuracy_score:  0.5506938651702434

cross_val_score:  0.4820741302493127

Classification report:
              precision    recall  f1-score   support

         1.0       0.60      0.43      0.50      3803
         2.0       0.50      0.12      0.19       630
         3.0       0.54      0.13      0.21      1096
         4.0       0.34      0.04      0.08      1682
         5.0       0.54      0.89      0.67      5976

    accuracy                           0.55     13187
   macro avg       0.51      0.32      0.33     13187
weighted avg       0.53      0.55      0.49     13187
```

```
Confusion matrix:

[[1652    0   10   29 2112]
 [ 213   75   27   57  258]
 [ 174    0  139   40  743]
 [ 185    0   43   74 1380]
 [ 525   74   39   16 5322]]
```

Gradient Boosting

```
*** GradientBoostingClassifier ***

accuracy_score:  0.7612042162736028

cross_val_score:  0.5091098682339559

Classification report:
              precision    recall  f1-score   support

         1.0       0.75      0.79      0.77      3803
         2.0       0.73      0.66      0.69       630
         3.0       0.85      0.52      0.64      1096
         4.0       0.80      0.44      0.56      1682
         5.0       0.76      0.89      0.82      5976

    accuracy                           0.76     13187
   macro avg       0.78      0.66      0.70     13187
weighted avg       0.77      0.76      0.75     13187
```

```
Confusion matrix:

[[3018   44   24   53  664]
 [  57  413    1   33  126]
 [ 187   23  568   19  299]
 [ 257   39   52  732  602]
 [ 519   45   23   82 5307]]
```

Choosing Best Model

After running the loop we get a dataframe showing each model and scores obtained.

	Model	Accuracy_score	Cross_val_score
0	Logistic Regression	77.758398	46.123803
1	DecisionTreeClassifier	77.492986	41.162195
2	KNeighborsClassifier	76.272086	42.369893
3	RandomForestClassifier	77.530902	48.243435
4	AdaBoostClassifier	54.159399	50.459759
5	GradientBoostingClassifier	76.014256	53.043890

Looking the various metrics we conclude the gradient boosting and random forest perform better compared to other models. So we will tune theses two models and then finalize the more efficient one.

HYPER-PARAMETRIC TUNING

We have used Grid Search CV to find best parameters and use those parameters in building model.

Gradient Boosting Classification

```
gb= GradientBoostingClassifier()
params={'loss':['deviance'],'n_estimators':[10,15], 'criterion':['mse'],
       'max_depth':[5,7], 'min_samples_split':[4,6],
       'min_samples_leaf':[2,3]}
grd=GridSearchCV(gb,param_grid=params)
grd.fit(x_train,y_train)
print('best params=>',grd.best_params_)
```

```
best params=> {'criterion': 'mse', 'loss': 'deviance', 'max_depth': 7, 'min_samples_leaf': 2, 'min_samples_split': 4, 'n_estima
tors': 15}
```

```
gb= grd.best_estimator_
gb.fit(x_train,y_train)
y_pred=gb.predict(x_test)
print("Gradient Boosting Classification: Accuracy = ",accuracy_score(y_test,y_pred))
print("\n Confusion Matrix= ",confusion_matrix(y_test,y_pred))
print("\n Classification Report= ",classification_report(y_test,y_pred))
```

```
Gradient Boosting Classification: Accuracy =  0.7219231060893304
```

```
Confusion Matrix=  [[2624    0   24   16 1139]
 [  25  259    1    6  339]
 [ 149    0  516   11  420]
 [ 247   21   32  613  769]
 [ 378   32   17   41 5508]]

Classification Report=               precision    recall  f1-score   support

         1.0       0.77      0.69      0.73      3803
         2.0       0.83      0.41      0.55       630
         3.0       0.87      0.47      0.61      1096
         4.0       0.89      0.36      0.52      1682
         5.0       0.67      0.92      0.78      5976

    accuracy                           0.72     13187
   macro avg       0.81      0.57      0.64     13187
weighted avg       0.75      0.72      0.71     13187
```

Random Forest Classification

```
rmf= RandomForestClassifier()
params={'n_estimators':[13,15], 'criterion':['entropy'],
        'max_depth':[10], 'min_samples_split':[10,11],
        'min_samples_leaf':[5,6]}
grd_r=GridSearchCV(rmf,param_grid=params)
grd_r.fit(x_train,y_train)
print('best params=>',grd_r.best_params_)
```

```
best params=> {'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 5, 'min_samples_split': 10, 'n_estimators': 15}
```

```
rmf= grd_r.best_estimator_
rmf.fit(x_train,y_train)
y_pred=rmf.predict(x_test)
print("Random Forest Classification: Accuracy = ",accuracy_score(y_test,y_pred))
print("\n Confusion Matrix= ",confusion_matrix(y_test,y_pred))
print("\n Classification Report= ",classification_report(y_test,y_pred))
```

```
Random Forest Classification: Accuracy =  0.5877000075832259


 Confusion Matrix=  [[1395    0    4    0 2404]
 [  67   59    1    9  494]
 [  91    0  299    0  706]
 [  79    0   32  163 1408]
 [ 134    0    6    2 5834]]

 Classification Report=               precision    recall  f1-score   support

         1.0       0.79      0.37      0.50      3803
         2.0       1.00      0.09      0.17       630
         3.0       0.87      0.27      0.42      1096
         4.0       0.94      0.10      0.18      1682
         5.0       0.54      0.98      0.69      5976

    accuracy                           0.59     13187
   macro avg       0.83      0.36      0.39     13187
weighted avg       0.71      0.59      0.52     13187
```

After applying hyper-parameter tuning we can see that Gradient Boosting classifier gives an accuracy of 72.19% and Random Forest Classifier gives an accuracy of 58.77%. Therefore we will finalize Gradient Boosting Model as our final model and save it using pickle for future use.

CONCLUSIONS

KEY FINDINGS AND CONCLUSIONS OF THE STUDY

First, we collected the reviews and ratings data from different e-commerce websites like Amazon and Flipkart and it was done by using Webscraping. The framework used for webscraping was Selenium, which has an advantage of automating our process of collecting data
We collected almost 52000 of data which contained the ratings from 1.0 to 5.0 and their reviews. Then we combined it into a single dataframe and saved it to a csv file.

We have used NLP to pre-process and clean data following steps were performed:

- Removing punctuations, e-mails, numbers, white spaces, etc.
- Removing stopwords
- Lemmatizing and removing words with length less than 3

Next we have vectorized text column using Tfidf Vectorizer and then separated data into train and test. After separating our train and test data, we have used different algorithms to build a model.
After looking at various metrics and parametric tuning we concluded gradient boosting model with accuracy 72.19% as our best model and saved it using pickle.

LEARNING OUTCOMES OF THE STUDY IN RESPECT OF DATA SCIENCE

After the completion of this project, we got an insight of how to collect data, pre-processing the data, analyzing the data and building a model. We have used NLP techniques to clean data. Also it helped me in exploring different algorithms and metrics to get the best output.

LIMITATIONS OF THIS WORK AND SCOPE FOR FUTURE WORK

More time consumption during hyperparameter tuning for both models, as the data was large and less number of parameters were used during tuning. This project is done with limited resources and can be made more efficient in future.