FLIP ROBO

Project Name

FLIGHT PRICE PREDICTION

PROJECT REPORT

Submitted by:

Priyanka Saikia

ACKNOWLEDGMENT

I would like to express my deep sense of gratitude to my SME (Subject Matter Expert) **Mr. Shubham Yadav** as well as **Flip Robo Technologies** who gave me the golden opportunity to do this data analysis project on **Flight Price Prediction Project** , which also helped me in doing lots of research and I came to know about so many new things.
I have put in my all efforts while doing this project.

All the external resources that were used in creating this project are listed below:

1) https://www.google.com/
2) https://www.youtube.com/
3) https://github.com/
4) https://www.kaggle.com/
5) https://towardsdatascience.com/
6) https://www.analyticsvidhya.com/

Priyanka Saikia

INTRODUCTION

- ## Business Problem Framing:

The airline industry is considered as one of the most sophisticated industry in using complex pricing strategies. Nowadays, ticket prices can vary dynamically and significantly for the same flight, even for nearby seats. The ticket price of a specific flight can change up to 7 times a day. Customers are seeking to get the lowest price for their ticket, while airline companies are trying to keep their overall revenue as high as possible and maximize their profit. However, mismatches between available seats and passenger demand usually leads to either the customer paying more or the airlines company losing revenue. Airlines companies are generally equipped with advanced tools and capabilities that enable them to control the pricing process. However, customers are also becoming more strategic with the development of various online tools to compare prices across various airline companies. In addition, competition between airlines makes the task of determining optimal pricing is hard for everyone.

Anyone who has booked a flight ticket knows how unexpectedly the prices vary. The cheapest available ticket on a given flight gets more and less expensive over time. This usually happens as an attempt to maximize revenue based on:

1.Time of purchase patterns (making sure last-minute purchases are expensive)
2. Keeping the flight as full as they want it (raising prices on a flight which is filling up in order to reduce sales and hold back inventory for those expensive last-minute expensive purchases)

- ## Conceptual Background of the Domain Problem:

Airline companies use complex algorithms to calculate flight prices given various conditions present at that particular time. These methods take financial, marketing, and various social factors into account to predict flight prices. Nowadays, the number of people using flights has increased significantly. It is difficult for airlines to maintain prices since prices change dynamically due to different conditions. That’s why we will try to use machine learning to solve this problem. This can help airlines by predicting what prices they can maintain. It can also help customers to predict future flight prices and plan their journey accordingly.

- ## Review of Literature

As per the requirement of client, we have scraped data from online sites like yatra.com and based on that data we have done analysis like based on which feature of the data, prices are changing and checked the relationship of flight price with all the feature like which flight he should choose.

- ## Motivation for the Problem Undertaken

I have worked on this on the bases of client requirements and followed all the steps till model deployment.

Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem

As per the client's requirement for this flight price prediction project I have scraped data from yatra.com website. This is then saved into Excel format file. I have shared the script for web scraping into the GitHub repository.

In our scraped dataset our target variable column "Price" is a continuous variable. Therefore, we will be handling this modelling problem as a Regression problem

Then loaded this data into a dataframe. For checking datatypes and null values pandas.DataFrame.info() method has been used. I have gone through several EDA steps to analyse the data. After all the necessary steps I have built an ML model to predict the flight prices.

- ## Data Sources and their formats

This project is done in three parts:
- Data Collection
- Data Analysis
- Model Building

1. Data Collection

You have to scrape at least 1500 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 have to scrape the data of flights from different websites (yatra.com, skyscanner.com, official websites of airlines, etc). The number of columns for data doesn't have limit, it's up to you and your creativity. Generally, these columns are airline name, date of journey, source, destination, route, departure time, arrival time, duration, total stops and the target variable price. You can make changes to it, you can add or you can remove some columns, it completely depends on the website from which you are fetching the data.

2. Data Analysis

After cleaning the data, you have to do some analysis on the data.
Do airfares change frequently? Do they move in small increments or in large jumps? Do they tend to go up or down over time? What is the best time to buy so that the consumer can save the most by taking the least risk? Does price increase as we get near to departure date? Is Indigo cheaper than Jet Airways? Are morning flights expensive?

3. Model Building

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

Following the complete life cycle of data science. Including all the steps like-

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

- ## Data Sources and their formats

The dataset is in the form of Excel file and consists of 9 columns (8 features and 1 label) with 3365 rows as explained below:

1) Airline: Name of the airline
2) Dep_time: Time of departure of flight from the source location
3) Arrival_Time: Time of arrival at destination
4) Duration: Total time of the journey from source to destination
5) Source: City name from where the flight is departing
6) Destination: Name of the city where flight is arriving
7) Meal_availability : Information about meal fare
8) Total_stops: Number of stops during the journey
9) Price : Flight fare

Loading the dataset

```
df = pd.read_excel("Flight_Price_Dataset.xlsx")
df
```

	Unnamed: 0	Airline	Dep_Time	Arrival_Time	Duration	Source	Destination	Meal_Availability	Total_Stops	Price
0	0	Go First	08:00	10:10	2h 10m	New Delhi	Mumbai	eCash 250	Non Stop	5,954
1	1	Go First	14:20	16:35	2h 15m	New Delhi	Mumbai	eCash 250	Non Stop	5,954
2	2	Go First	21:00	23:15	2h 15m	New Delhi	Mumbai	eCash 250	Non Stop	5,954
3	3	Go First	16:45	21:25	4h 40m	New Delhi	Mumbai	eCash 250	1 Stop	5,954
4	4	Go First	12:35	18:10	5h 35m	New Delhi	Mumbai	eCash 250	1 Stop	5,954
...	...	...	...	...	...	...	...	...	...	...
3360	3360	Air India	11:05	20:30	9h 25m	Lucknow	Goa	Free Meal	1 Stop	12,084
3361	3361	Air India	08:50	14:55	6h 05m	Lucknow	Goa	eCash 250	1 Stop	12,818
3362	3362	Air India	08:55	22:25	13h 30m	Lucknow	Goa	Free Meal	3 Stop(s)	13,137
3363	3363	Air India	08:55	06:25	21h 30m	Lucknow	Goa	Free Meal	2 Stop(s)	13,137
3364	3364	Air India	20:15	20:30	24h 15m	Lucknow	Goa	Free Meal	2 Stop(s)	13,137

3365 rows × 10 columns

- ## Data Pre-processing Done:

The following pre-processing pipeline is required to be performed before building the classification model prediction:

1. Load dataset:

```
data = pd.read_csv('Rating_Review_Data.csv',index_col=0)
data
```

	Review_title	Review_text	Ratings
0	Suitable for School kids	\n If you are a College student or a professi...	2.0 out of 5 stars
1	Misrepresentation on MS Office 2019 license - ...	\n Update after one month usage - MS Office 2...	2.0 out of 5 stars
2	The sold me renewed laptop	\n It's look like renewed laptop because lapt...	2.0 out of 5 stars
3	Amazon dupes with specification/ battery sucks	\n I had seen the specifications and bo...	2.0 out of 5 stars
4	Display back light issue	\n Display gone with 2 months.. But anyway th...	2.0 out of 5 stars
...	...	...	...
77545	Nice product	good product	4
77546	Awesome	Very good as expected and happy with the purchase	5
77547	Awesome	I love it! No complaint!	5
77548	Nice product	good product	4
77549	Awesome	Very good as expected and happy with the purchase	5

77550 rows × 3 columns

2. Merging feature Review_title and Review_text to review_text
3. Dropping columns Review_title and Review_text.
4. Treating Null Values by dropping null value rows using pandas dropna() method.
5. Converting review_text to lower-case.
6. Removing punctuations, leading whitespaces, trailing whitespaces and replace money symbols with 'dollars', numbers with 'numbr', white space between terms with single space.
7. Removing Stop Words
8. Converting Text into Vectors using TfidfVectorizer
9. Building the predictive model.
10. Loading the serialized Model
11. Predicting Output by using the final model.

Checked the ratings column and it had 10 values instead of 5 so had to clean it through and ensure that our target label was updated as a numeric datatype instead of the object datatype value. Made sure that the string entries were replaced properly.

```python
#incorporating the string object datatype values with numeric star values
df['rating'] = df['rating'].replace('1.0 out of 5 stars',1)
df['rating'] = df['rating'].replace('2.0 out of 5 stars',2)
df['rating'] = df['rating'].replace('3.0 out of 5 stars',3)
df['rating'] = df['rating'].replace('4.0 out of 5 stars',4)
df['rating'] = df['rating'].replace('5.0 out of 5 stars',5)
df['rating'] = df['rating'].astype('int')
df['rating'].unique()
```

```
array([2, 3, 1, 5, 4])
```

```python
df.head()
```

	rating	review_text
0	2	Suitable for School kids \n If you are a Coll...
1	2	Misrepresentation on MS Office 2019 license - ...
2	2	The sold me renewed laptop \n It's look like ...
3	2	Amazon dupes with specification/ battery sucks...
4	2	Display back light issue \n Display gone with...

- ## Data Inputs- Logic- Output Relationships

Input	Logic (algorithm)	Output
Review_text (object)	MultinomialNB SGDClassifier KNeighborsClassifier DecisionTreeClassifier	rating

There is 1 input variable needs to be provided to the logic to get the output i.e. rating. Logic highlighted in green i.e. SGDClassifier is the best performing algorithm among all other logics on this dataset.

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

By looking into the target variable/label, we assumed that it was a multiclass classification type of problem. Also, we observed that our dataset was imbalance so we will have to balance the dataset for better prediction accuracy outcome. The 5-star rating system allows respondents to rank their feedback on a 5-point scale from 1 to 5.

The more stars that are selected, the more positively your customer is responding to the purchased products. People tend to overlook businesses with a less than four star rating or lower. Usually, when people are researching a company, the goal is to find one with the highest overall score and best reviews. Having a five-star rating means a lot for your reputation score and acquiring new customers.

- ## Hardware and Software Requirements and Tools Used

➢ Hardware Used:
RAM: 8 GB
CPU: AMD A8 Quad Core 2.2 Ghz
GPU: AMD Redon R5 Graphics

➢ Software Tools used:
Programming language: Python 3.0
Distribution: Anaconda Navigator
Browser-based language shell: Jupyter Notebook

➢ Libraries/Packages Used:

- Pandas
- Numpy
- Matplotlib
- Seaborn
- Scipy.stats
- Sklearn
- NLTK
- wordcloud

Model/s Development and Evaluation

- ## Identification of possible problem-solving approaches (methods)

We checked through the entire training dataset for any kind of missing values information

```
#checking null values
df.isnull().sum()
```

```
rating          9027
review_text     8085
dtype: int64
```

We dropped all the null values from our dataframe using the using pandas dropna() method.

Then we went ahead and took a look at the dataset information. Using the info method, we are able to inspect the datatype information. We have a total of 2 columns and both have object datatype.

Code:

```
#checking general information
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 77550 entries, 0 to 77549
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   rating        68523 non-null  object
 1   review_text   69465 non-null  object
dtypes: object(2)
memory usage: 1.8+ MB
```

- ## Testing of Identified Approaches (Algorithms)

The list of all the algorithms used for the training and testing classification model are listed below:

1. MultinomialNB

2. SGDClassifier

3. KNeighborsClassifier

4. DecisionTreeClassifier

From all of these above models, SGDClassifier gave me good performance.

- ## Run and Evaluate selected models

To perform training and testing operation(s) following functions has been defined for which codes are as follows:

```python
#function to get best random state
def get_best_random_state(model,X,Y,t_size=0.25,rs_range=range(1,301,50)):
    best_rstate = 0
    best_accuracy_score = 0
    random_state_message = "\r"

    for i in tqdm.tqdm(rs_range,desc=f"Best_Random_State => {model}"):
        X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=t_size,random_state=i)
        model.fit(X_train, Y_train)
        y_pred = model.predict(X_test)
        a_score = accuracy_score(Y_test,y_pred)

        if a_score > best_accuracy_score:
            best_accuracy_score = a_score
            best_rstate = i

        random_state_message += f"[{i}: {round(a_score*100,2)}]<--->"
        sys.stdout.write(random_state_message)

    sys.stdout.write(f"\n\nBest Random State: {best_rstate} found with Accuracy: {best_accuracy_score}")
    return best_rstate, best_accuracy_score



#function to get best cv score
def get_best_cv(model,X_train,Y_train,parameters,cv_range=range(5,25,5)):
    best_cv_score = 0
    best_cv = 0

    cv_message = "\r"
    for i in tqdm.tqdm(cv_range,desc=f"Best_CV => {model}"):
        gscv = GridSearchCV(model,parameters)
        gscv.fit(X_train,Y_train)

        cv_score = cross_val_score(gscv.best_estimator_,X_train,Y_train,cv=i).mean()

        if cv_score > best_cv_score:
            best_cv_score = cv_score
            best_cv = i

        cv_message += f"[{i}:{round(cv_score*100,2)}]<--->"
        sys.stdout.write(cv_message)

    sys.stdout.write(f"\n\nBest CV: {best_cv} found with Cross Val Score: {best_cv_score}")

    return best_cv, best_cv_score


#function to build models
def build_models(models,X,Y,t_size=0.25,rs_range=range(1,301,50),cv_range=range(5,25,5)):
    for i in tqdm.tqdm(models,desc="Building Models"):
        sys.stdout.write("\n=============================================================================================\n")
        sys.stdout.write(f"Current Model in Progress: {i} ")
        sys.stdout.write("\n=============================================================================================\n")

        #start time
        start_time = timeit.default_timer()

        #Find the best random state
        best_random_state, best_accuracy_score = get_best_random_state(models[i]['name'],X,Y,t_size,rs_range)
        sys.stdout.write("\n")

        #Spliting train and test data using train_test_split method with best random state value
        X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=t_size,random_state=best_random_state)
```

```
        #Find the best CV
        best_cv, best_cv_score = get_best_cv(models[i]['name'],X_train,Y_train,models[i]['parameters'],cv_range)
        sys.stdout.write("\n\nBuilding Model...")

        #Training the model using best CV
        gscv = GridSearchCV(models[i]['name'],models[i]['parameters'],cv=best_cv)
        gscv.fit(X_train,Y_train)

        #Testing model
        y_pred = gscv.best_estimator_.predict(X_test)

        #Recording model performance
        model_accuracy_score = accuracy_score(Y_test,y_pred)
        model_confusion_matrix = confusion_matrix(Y_test,y_pred)
        model_classification_report = classification_report(Y_test,y_pred)

        #end time
        end_time = timeit.default_timer()
        sys.stdout.write(f"Completed in [{end_time-start_time} sec.]")

        #storing model specifications
        models[i]['initial_accuracy_score'] = best_accuracy_score
        models[i]['best_random_state'] = best_random_state
        models[i]['x_train'] = X_train
        models[i]['x_test'] = X_test
        models[i]['y_train'] = Y_train
        models[i]['y_test'] = Y_test
        models[i]['best_cv'] = best_cv
        models[i]['best_cv_score'] = best_cv_score
        models[i]['gscv'] = gscv
        models[i]['y_predict'] = y_pred
        models[i]['final_accuracy'] = model_accuracy_score
        models[i]['confusion_matrix'] = model_confusion_matrix
        models[i]['classification_report'] = model_classification_report
        models[i]['build_time'] = f"{end_time - start_time} (in sec.)"

        sys.stdout.write("\n==========================================================================================\n\n\n")

    return models


#function to display model performance
def display_performance(models):
    model_names = []
    model_initial_score = []
    model_cross_val_score = []
    model_final_score = []
    model_build_time = []
    for i in models:
        model_names.append(i)
        model_initial_score.append(models[i]['initial_accuracy_score'])
        model_cross_val_score.append(models[i]['best_cv_score'])
        model_final_score.append(models[i]['final_accuracy'])
        model_build_time.append(models[i]['build_time'])

    model_df = pd.DataFrame({
        "Name": model_names,
        "Initial Score": model_initial_score,
        "Cross Val Score": model_cross_val_score,
        "Final Score": model_final_score,
        "Build Time": model_build_time,
    })

    model_df['Difference (Final Score - Cross Val Score)'] = model_df['Final Score'] - model_df['Cross Val Score']
    display(model_df)
```

```
for i in models:
    print("=====================================================")
    print(f"for model: {i}")
    print("=====================================================")
    print("CLASSIFICATION REPORT")
    print(models[i]['classification_report'])
    print("CONFUSION MATRIX")
    print(models[i]['confusion_matrix'])
    print("=====================================================\n\n")

return
```

Displaying the models performance:

```
#displaying model performances
display_performance(models)
```

	Name	Initial Score	Cross Val Score	Final Score	Build Time	Difference (Final Score - Cross Val Score)
0	MultinomialNB	0.618981	0.603717	0.618981	5.434509373000083 (in sec.)	0.015264
1	SGDClassifier	0.630456	0.648257	0.650938	121.64814187600007 (in sec.)	0.002681
2	KNeighborsClassifier	0.552386	0.641573	0.646863	1872.012561575 (in sec.)	0.005291
3	DecisionTreeClassifier	0.657158	0.635711	0.664450	4730.549403769 (in sec.)	0.028740

```
=====================================================
for model: MultinomialNB
=====================================================
CLASSIFICATION REPORT
              precision    recall  f1-score   support

           1       0.68      0.73      0.70      1862
           2       0.55      0.48      0.51      1845
           3       0.49      0.58      0.53      1793
           4       0.62      0.59      0.60      1958
           5       0.78      0.70      0.74      1867

    accuracy                           0.62      9325
   macro avg       0.62      0.62      0.62      9325
weighted avg       0.62      0.62      0.62      9325

CONFUSION MATRIX
[[1366  309  149   27   11]
 [ 424  894  411   85   31]
 [ 158  300 1040  235   60]
 [  51   91  391 1161  264]
 [  24   33  121  378 1311]]
=====================================================



=====================================================
```

```
for model: SGDClassifier
============================================================
CLASSIFICATION REPORT
              precision    recall  f1-score   support

           1       0.69      0.74      0.72      1886
           2       0.56      0.56      0.56      1843
           3       0.59      0.53      0.56      1850
           4       0.62      0.63      0.63      1843
           5       0.79      0.78      0.78      1903

    accuracy                           0.65      9325
   macro avg       0.65      0.65      0.65      9325
weighted avg       0.65      0.65      0.65      9325

CONFUSION MATRIX
[[1401  315  111   44   15]
 [ 393 1036  263  107   44]
 [ 164  346  987  278   75]
 [  42  128  241 1168  264]
 [  19   37   74  295 1478]]
============================================================


============================================================
for model: KNeighborsClassifier
============================================================
CLASSIFICATION REPORT
              precision    recall  f1-score   support

           1       0.66      0.72      0.69      1870
           2       0.68      0.55      0.61      1901
           3       0.53      0.64      0.58      1877
           4       0.67      0.56      0.61      1808
           5       0.72      0.76      0.74      1869

    accuracy                           0.65      9325
   macro avg       0.65      0.65      0.65      9325
weighted avg       0.65      0.65      0.65      9325

CONFUSION MATRIX
[[1351  176  218   58   67]
 [ 330 1050  360   92   69]
 [ 198  189 1201  170  119]
 [ 103   77  317 1018  293]
 [  74   50  153  180 1412]]
```

==

==
for model: DecisionTreeClassifier
==
CLASSIFICATION REPORT

	precision	recall	f1-score	support
1	0.72	0.72	0.72	1851
2	0.61	0.61	0.61	1895
3	0.60	0.60	0.60	1871
4	0.64	0.66	0.65	1870
5	0.76	0.74	0.75	1838
accuracy			0.66	9325
macro avg	0.67	0.66	0.67	9325
weighted avg	0.67	0.66	0.66	9325

CONFUSION MATRIX

```
[[1332  278  144   59   38]
 [ 299 1156  265  130   45]
 [ 141  266 1126  231  107]
 [  44  128  238 1231  229]
 [  37   76  109  265 1351]]
```

==

Observation: From the above model performance comparison, it is clear that **SGDClassifier** performs well with **accuracy_score of 65.09%** and **lowest difference between accuracy_score and cross_val_score**. Hence, we are proceeding with **SGDClassifier** as the final model.

Saving the best model:

```
import joblib
#selecting best model
best_model = models['SGDClassifier']

#saving model
joblib.dump(best_model['gscv'].best_estimator_,open('Rating_Review.obj','wb'))
```

- ## Key Metrics for success in solving problem under consideration:

To find out best performing model, the following metrices are used:

1. Accuracy Score: It is used to check the model performance score between 0.0 to 1.0

2. Confusion Matrix: A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.

3. Cross Validation: Cross-validation helps to find out the over fitting and under fitting of the model. In the cross validation the model is made to run on different subsets of the dataset which will get multiple measures of the model. If we take 5 folds, the data will be divided into 5 pieces where each part being 20% of full dataset. While running the Cross validation the 1st part (20%) of the 5 parts will be kept out as a holdout set for validation and everything else is used for training data. This way we will get the first estimate of the model quality of the dataset.

In the similar way further iterations are made for the second 20% of the dataset is held as a holdout set and remaining 4 parts are used for training data during process. This way we will get the second estimate of the model quality of the dataset. These steps are repeated during the cross-validation process to get the remaining estimate of the model quality.

4. Classification Report: A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False.

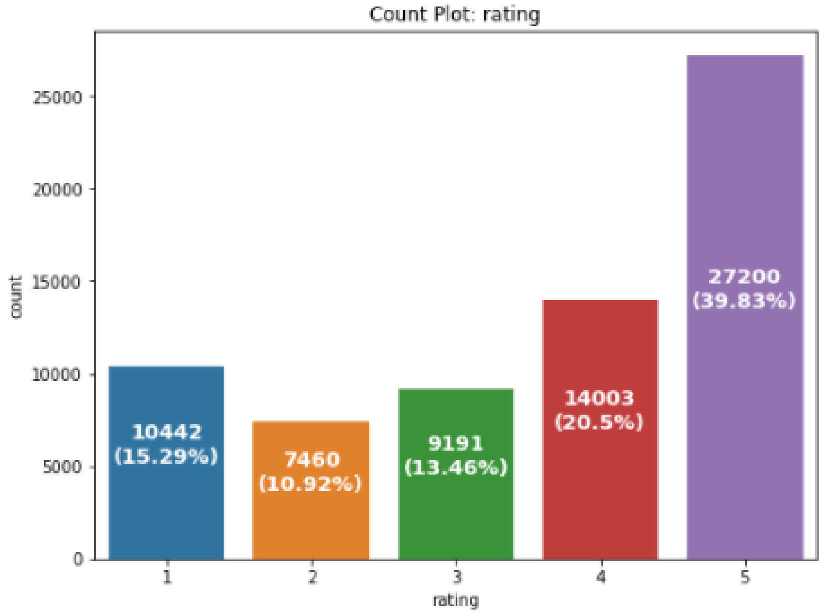
5. Hyperparameter Tuning: There is a list of different machine learning models. They all are different in some way or the other, but what makes them different is nothing but input parameters for the model. These input parameters are named as Hyperparameters. These hyperparameters will define the architecture of the model, and the best part about these is that you get a choice to select these for your model. You must select from a specific list of hyperparameters for a given model as it varies from model to model. We are not aware of optimal values for hyperparameters which would generate the best model output. So, what we tell the model is to explore and select the optimal model architecture automatically. This selection procedure for hyperparameter is known as Hyperparameter Tuning. We can do tuning by using GridSearchCV. GridSearchCV is a function that comes in Scikit-learn (or SK-learn) model selection package. An important point here to note is that we need to have Scikit-learn library installed on the computer. This function helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end we can select the best parameters from the listed hyperparameters.

- ## Visualizations:

  To better understand the data, the following types of visualizations have been used.

➢ Univariate analysis is the simplest form of data analysis where the data being analyzed contains only one variable. In this project, distribution plot, count plot, box plot and bar plot has been used.

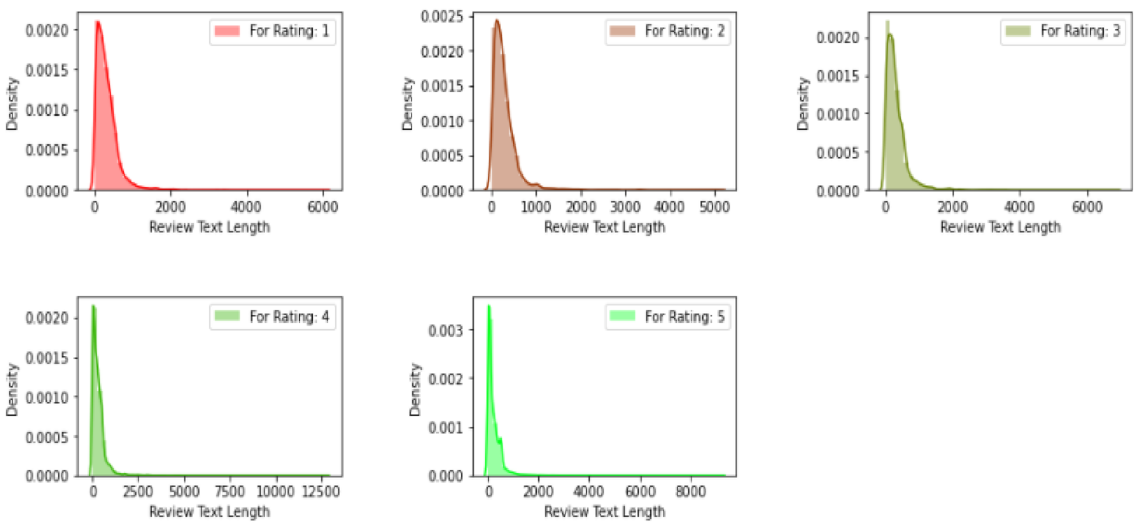
Count Plot:



Observations:

- In the above plot we can see how initially we had an imbalanced class concern that we later rectified by manually choosing the same number of records for each and every class and ensuring that the dataset get balanced multiclass label variable.
- There are records available for all ratings i.e., from 1 to 5.
- There are highest number of 5 star ratings followed by 4 star ratings present in the dataset.
- We can see a high 1 star rating as well compared to 2 and 3 star rating reviews.

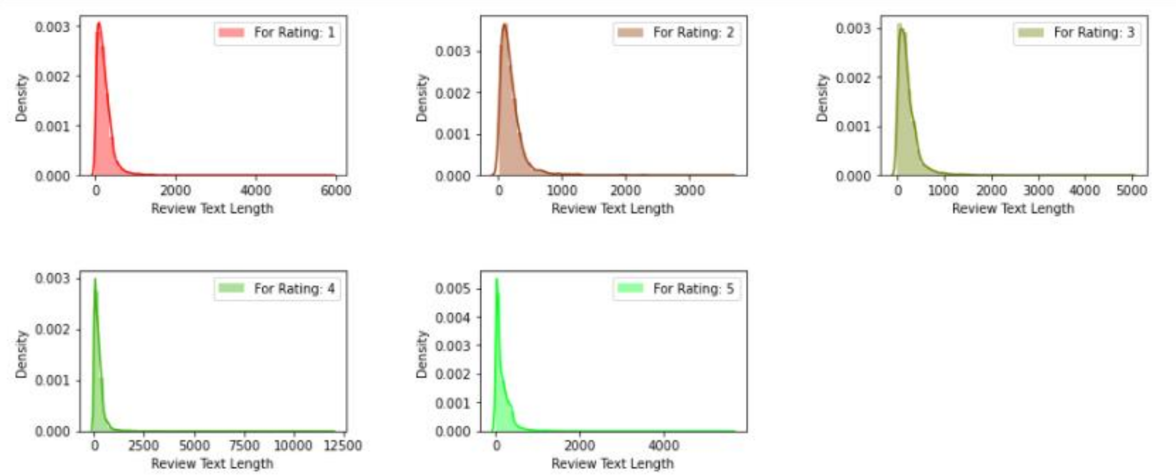
Distribution Plot: Review text length Distribution Plot before Data Cleaning



Observations:

- Rating 1, 3, 5 has almost similar review text length.
- Rating 4 has the highest review text length.
- Rating 2 has the lowest review text length.

Review text length Distribution Plot after Data Cleaning:



Observation: Review text length reduced by almost 1000 characters for Rating 1 to 5.

Displaying with WordCloud:

Observations:

- **for Rating: 1**
  It is mostly consists of words like phone, headphone, waste, money, slow, worst, issue etc.
- **for Rating: 2**
  It is mostly consists of words like phone, good, watch, product, problem, issue, bad etc.
- **for Rating: 3**
  It is mostly consists of words like phone, good, watch, problem, sound, feature, quality etc.
- **for Rating: 4**
  It is mostly consists of words like phone, good, value, money, nice product, battery backup, great etc.
- **for Rating: 5**
  It is mostly consists of words like value, money, must buy, great, perfect, better, terrific purchase, mind blowing etc.

- ## Interpretation of the Results

  - Starting with univariate analysis, with the help of countplot, it was found that the data consists of highest number of 5 star rating followed by 4 star ratings. Moving further with the removal and replacement of certain terms (like, punctuations, extra spaces, numbers, money symbols) as well as removal of stop words, it was evident that the length of review text decreases by a large amount. This was also depicted by using distribution plot.

  - With the help of wordcloud, it was found that the rating 1 consists of words like phone, headphone, waste, money, slow, worst, issue etc, rating 2 consists of words like problem, issue, bad, etc, rating 3 consists of words like problem, sound, quality etc, rating 4 consists of word like good, value, money, nice, etc. and rating 5 consists of words like must buy, great, perfect, mind blowing etc.

CONCLUSION

- ## Key Findings and Conclusions of the Study

  From the model performance comparison, it is clear that SGDClassifier performs well with accuracy_score of 65.09% and lowest difference between accuracy_score and cross_val_score, hence proceeding with SGDClassifier as our final model.

- ## Learning Outcomes of the Study in respect of Data Science

  During the data analysis, Review_text feature contains null values which have been dropped, but these values can also be replaced with some other values which might impact the model performance either in positive or negative way. As of now, I am finishing this project with my current approach which gives the **final accuracy score of 65.09% and cross_val_score: 64.82%** and this can be further improved by training with more specific data.

- ## Limitations of this work and Scope for Future Work

  Current model is limited to technical product rating(s) and reviews data but this can further be improved for other sectors of ecommerce rating(s) prediction by training the model accordingly. The overall score can also be improved further by training the model with more specific data.

  As we know the content of text in reviews is totally depends on the reviewer and they may rate differently which is totally depends on that particular person. So it is difficult to predict ratings based on the reviews with higher accuracies. Still we can improve our accuracy by fetching more data and by doing extensive hyper-parameter tuning.