FLIP ROBO

PROJECT REPORT ON:

"Rating Prediction Project"

SUBMITTED BY:

Mr. Chris Chhotai

ACKNOWLEDGMENT

I would like to express my heartfelt gratitude to Ms. Swati Mahaseth, my SME (Subject Matter Expert), as well as Flip Robo Technologies, for allowing me to work on this project on flight price prediction and for assisting me in conducting extensive research, which allowed me to learn a lot of new things, particularly in terms of data collection.

A huge thanks to my academic team "Datatrained" who are the reason behind what I am today. Last but not least my parents who have been my backbone in every step of my life. And also thank you for many other persons who has helped me directly or indirectly to complete the project.

Contents:

1.INTRODUCTION

1.1 Business Problem Framing:

Rating prediction is a well-known recommendation task that attempts to forecast a user's rating for goods that she has not yet evaluated. Users' explicit input, i.e., their past ratings on some goods, is used to calculate predictions. User reviews, which implicitly indicate consumers' thoughts on things, are another sort of feedback. Recent research suggests that inferred opinions from user evaluations on items are strong predictors of implicit feedback or even ratings from users, and hence should be used in computation.

Customer reviews have become increasingly important as E-commerce has grown in popularity. There are hundreds of review sites online, and each product has a large number of reviews. Buyers' buying habits have changed, and 70 percent of customers claim they use rating filters to filter out low-rated items in their searches, according to a recent survey. Companies that encourage reviews, such as Google, Amazon, and Yelp!, rely on their ability to determine if a review will be beneficial to other customers and so increase product exposure. There are two major approaches to this problem. The first is based on content analysis of review texts and employs natural language processing concepts (the NLP method). This strategy lacks the information that can be gained from the customer-item relationship. The second is based on recommender systems, notably collaborative filtering, and is concerned with the reviewer's perspective.

We have a client who runs a website where users may leave reviews for various technological products. They are now adding a new element to their website, requiring reviewers to include stars (ratings) with their reviews. The rating is out of five stars, and there are only five alternatives available. 1 star, 2 stars, 3 stars, 4 stars, and 5 stars are the different types of stars. They now seek to forecast ratings for reviews that were written in the past but did not receive a rating. As a result, we need to create an application that can anticipate the rating based on the review.

1.2 Conceptual Background of the Domain Problem

In today's e-commerce applications, such as targeted advertising, tailored marketing, and information retrieval, recommendation systems are critical components. In recent years, the value of contextual data has prompted the creation of personalised recommendations based on the contextual data available to consumers. Review-based recommendation, as opposed to traditional algorithms that rely solely on a user's rating history, should present consumers with more relevant results. We present a review-based recommendation system that mines user evaluations for contextual information. The features gained by analysing textual reviews utilising methods established in the Natural Language Processing (NLP) and information retrieval disciplines to compute a utility function over a given item are the focus of the proposed approach. An item's utility is a metric that indicates how much it is preferred in the context of the user. The context inference in our system is modelled as a resemblance between the user's and item's review histories. We used our technology to mine contextual data from customer reviews of technological products and apply it to provide review-based rating prediction as an example application. The anticipated ratings can provide item-based recommendations, which should display in the product page's suggested items list. In compared to typical prediction approaches, our evaluations reveal that our methodology can assist provide better prediction rating ratings.

To our knowledge, all current efforts on recommendation approaches based on inferred opinions from user reviews are either focused on the item recommendation task or employ only the opinion information, fully disregarding user ratings. The approach suggested in this research addresses this gap by providing a simple, tailored, and scalable rating prediction framework that incorporates both user ratings and inferred opinions from their reviews. The suggested framework's usefulness is demonstrated by experimental results on a dataset containing user ratings and reviews from real-world Amazon and Flipkart Product Review Data.

1.3 Review of Literature

In real life, people's decisions are frequently influenced by the actions or recommendations of their friends. The use of social data has been extensively researched. Based on probabilistic matrix factorization, Yang et al. introduce the concept of "Trust Circles" in social networks. Individual choice, according to

Jiang et al., is another crucial component. Some websites do not always provide structured information, and all these methods do not take advantage of the user's unstructured data, such as reviews and explicit social network data, making it difficult to make a reasonable prediction for each user. The phrase "sentiment factor" is utilised to improve social recommendation in this case.

The growth of Web 2.0 and e-commerce has resulted in an increase in the quantity of online user reviews. Many decision-making processes, such as personal consumption decisions, commodity quality monitoring, and social opinion mining, rely on sentiment information found in online reviews. In natural language processing, machine learning, and Web mining, mining the sentiment and opinions included in online reviews has become a hot topic.

1.4 Motivation for the Problem Undertaken

FlipRobo first offered me the project as part of their internship programme. The primary goal has been to gain exposure to real-world data and the ability to apply my talents to a real-time situation. Many product reviews lack a scale rating mechanism, instead relying solely on a textual evaluation. In this circumstance, comparing several items in order to make a decision between them becomes difficult and time-consuming. As a result, algorithms that can predict the user rating based on the text review are vital. Getting a feel of a textual evaluation as a whole could improve the consumer experience. However, I was drawn to this project because it is in a relatively new field of study. We have a lot of options, but fewer solid solutions. The main aim is to create a prototype of an online hate and abuse review classifier that can be used to classify hate and positive comments so that they may be controlled and corrected based on the reviewer's preferences.

2.Analytical Problem Framing

2.1 Mathematical/ Analytical Modelling of the Problem

In this case, the ratings can be 1, 2, 3, 4, or 5, with 1 representing the most likely ness of the product to the client and 5 representing the least likely ness of the product to the customer. As a result, it's evident that this is a multi-classification problem, and I'll have to employ all of the classification methods to build the model. One type of supervised learning algorithm that we would use is classification. Only categorisation will be done here. Because the dataset only has one feature, filtering the words is required to avoid overfitting. To find the regularisation parameter, we would first eliminate email, phone numbers, web addresses, spaces, and stop words from the classification section of the project. We also used TFID to translate the tokens from the train documents into vectors so that machine can do further processing. While building the model, I utilised all of the classification algorithms, then tuned and saved the best model.

2.2 Data Sources and their formats

The data set contains nearly 33,294 samples with 2 features. Since **Ratings** is my target column and it is a categorical column with 5 categories, so this problem is a **Multi Classification Problem**. The Ratings can be 1, 2, 3, 4 or 5, which represents the likely ness of the product to the customer. The data set includes:

- Full review: Title and Text content of the Review.
- Ratings: Ratings out of 5 stars.

This project is more about exploration, feature engineering and classification that can be done on this data. Since the data set is huge and includes multiclassification of ratings, we can do good amount of data exploration and derive some interesting features using the Review column available.

We need to build a model that can predict Ratings of the reviewer.

2.3 Data Pre-processing Done

The process of turning raw data into a legible format for use by Machine Learning models is known as data pre-processing. Data pre-processing is a key stage in Machine Learning since the quality of data and the relevant information that can be gleaned from it has a direct impact on our model's capacity to learn. As a result, it's critical that we pre-process our data before feeding it into our model.

I have used following pre-processing steps:

- ✓ Importing necessary libraries and loading dataset as a data frame.
- ✓ Checked some statistical information like shape, number of unique values present, info, null values, value counts etc.
- ✓ Checked for null values and I replaced those null values using imputation method. And removed Unnamed: 0.
- ✓ Done text pre-processing techniques like Removing Punctuations and other special characters, Splitting the comments into individual words, Removing Stop Words, Stemming and Lemmatization.
- ✓ After getting a cleaned data used TF-IDF vectorizer. It'll help to transform the text data to feature vector which can be used as input in our 6 modelling. It is a common algorithm to transform text into numbers. It measures the originality of a word by comparing the frequency of appearance of a word in a document with the number of documents the words appear in. Mathematically, TF-IDF = TF(t*d)*IDF(t,d)

2.4 Data Inputs- Logic- Output Relationships

The dataset consists of 2 features with a label. The features are independent, and label is dependent as our label varies the values(text) of our independent variables changes.

- I checked the distribution of skewness using dist plots and used count plots to check the counts available in each column as a part of univariate analysis.
- Got to know the frequently occurring and rare occurring word with the help of count plot.
- And was able to see the words in the Review text with reference to their ratings using word cloud.

2.5 Hardware & Software Requirements & Tools Used

While taking up the project we should be familiar with the Hardware and software required for the successful completion of the project. Here we need the following hardware and software.

Hardware required	Software/s required
Processor: core i7 RAM: 16 GB ROM/SSD: 512 GB	Distribution: Anaconda Navigator Programming language: Python Browser based language shell: Jupyter Notebook

Libraries required:-

Importing Necessary Libraries:

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
from collections import defaultdict, Counter
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')

from nltk import FreqDist
from nltk.tokenize import word_tokenize
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import wordnet
from nltk import FreqDist
```

✓ To run the program and to build the model we need some basic libraries as follows:

In [35]:
```
#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 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
```

- ✓ **import pandas as pd**: **pandas** is a popular Python-based data analysis toolkit which can be imported using import pandas as pd. It presents a diverse range of utilities, ranging from parsing multiple file formats to converting an entire data table into a numpy matrix array. This makes pandas a trusted ally in data science and machine learning.
- ✓ **import numpy as np**: NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.
- ✓ **import seaborn as sns:** Seaborn is a data visualization library built on top of matplotlib and closely integrated with pandas data structures in Python. Visualization is the central part of Seaborn which helps in exploration and understanding of data.
- ✓ **Import matplotlib.pyplot as plt:** matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

With this sufficient library we can go ahead with our model building.

3.Data Analysis and Visualization

3.1 Identification of possible problem-solving approaches (methods)

Using the TF-IDF vectorizer, I transformed text into feature vectors and separated our feature and labels. In addition, before feeding the input data into the machine learning models, I made sure that it was cleaned and scaled. Simply making the Reviews more relevant so that we have less words to process and may achieve more accuracy.

3.2 Testing of Identified Approaches (Algorithms)

In this nlp based project we need to predict Ratings which is a multiclassification problem. I have converted the text into vectors using TFIDF vectorizer and separated our feature and labels then build the model using OneVsRestClassifier. Among all the algorithms which I have used for this purpose I have chosen SGDClassifier as best suitable algorithm for our final model as it is performing well compared to other algorithms while evaluating with different metrics I have used following algorithms and evaluated them

- KNeighbors Classifier
- DecisionTree Classifier
- RandomForest Classifier
- AdaBoost Classifier
- MultinomialNB
- GradientBoosting Classifier
- Bagging Classifier
- ExtraTrees Classifier

From all of these above models RandomForest Classifier was giving me good performance with less difference in accuracy score and cv score.

3.3 Key Metrics for success in solving problem under consideration

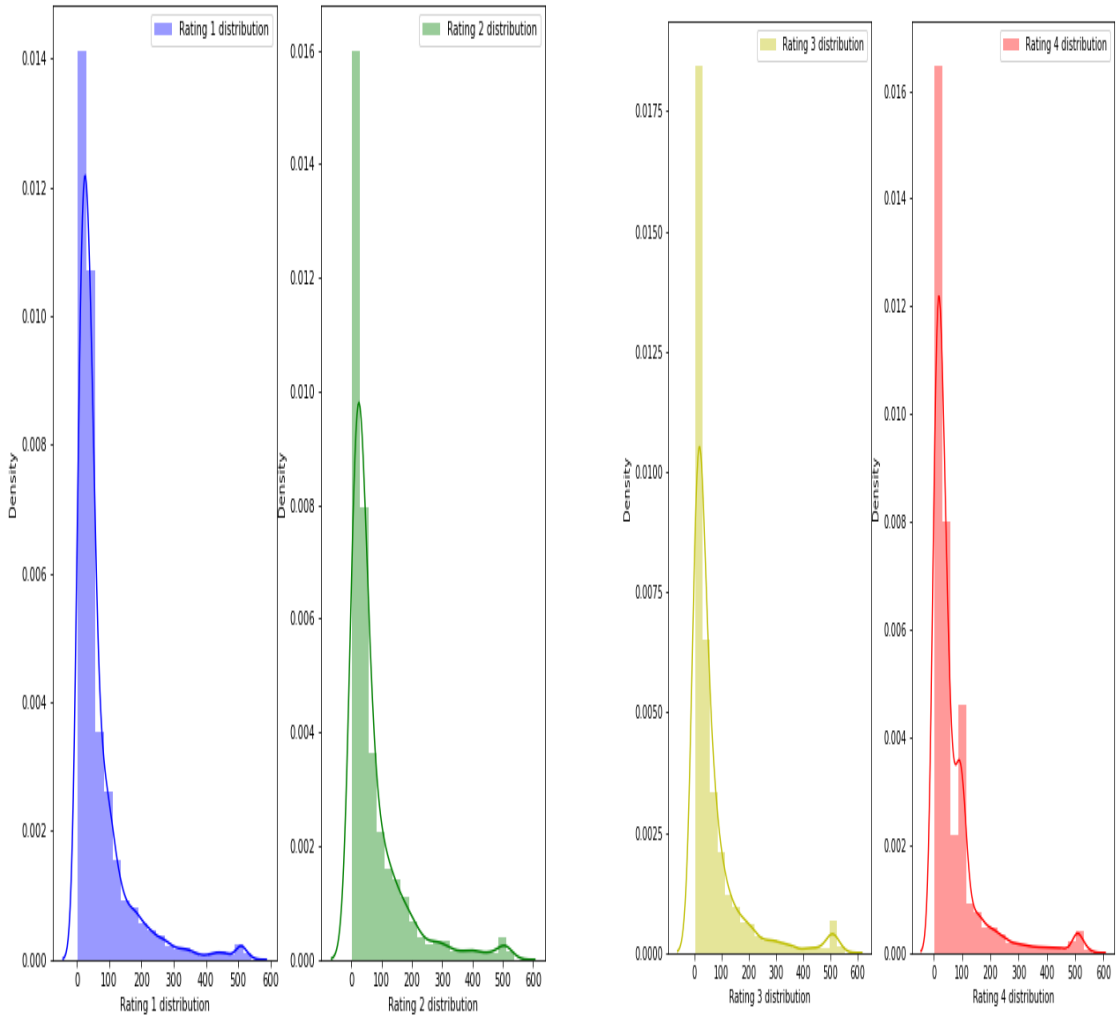
I have used the following metrics for evaluation:

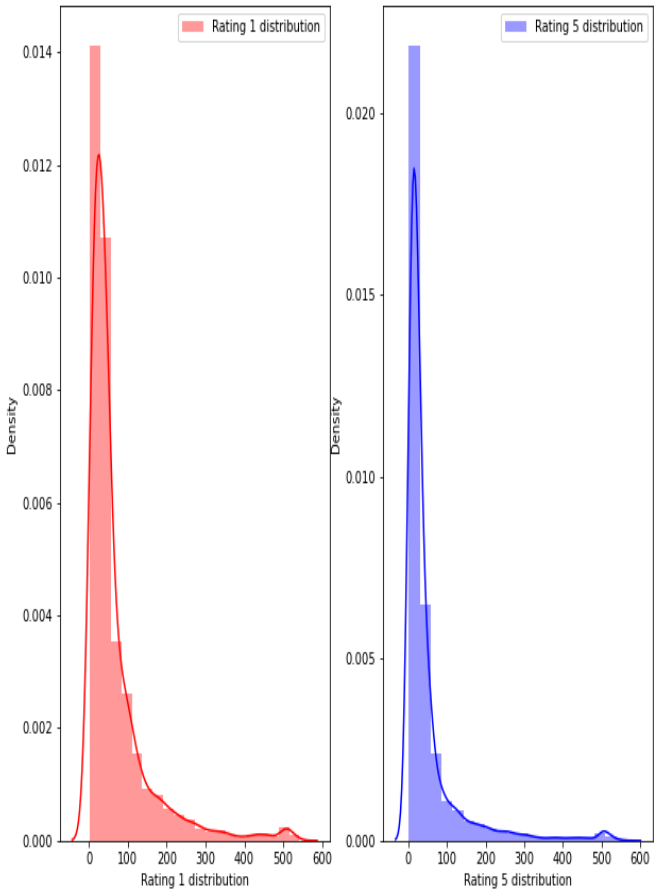
- I have used f1_score, precision_score, recall_score, multilabel_confusion_matrix and hamming loss all these evaluation metrics to select best suitable algorithm for our final model.
- **Precision** can be seen as a measure of quality,higher precision means that an algorithm returns more relevant results than irrelevant ones.
- **Recall** is used as a measure of quantity and high recall means that an algorithm returns most of the relevant results.

- **Accuracy score** is used when the True Positives and True negatives are more important.Accuracy can be used when the class distribution is similar.
- **F1-score** is used when the False Negatives and False Positives are crucial. While F1-score is a better metric when there are imbalanced classes.

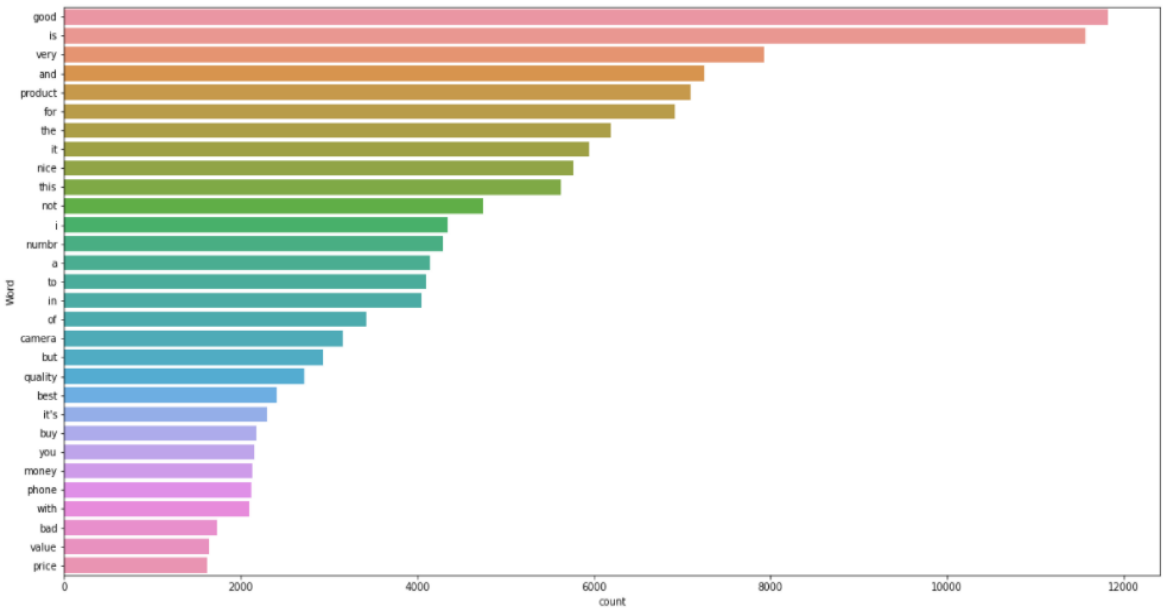
3.4 Visualizations

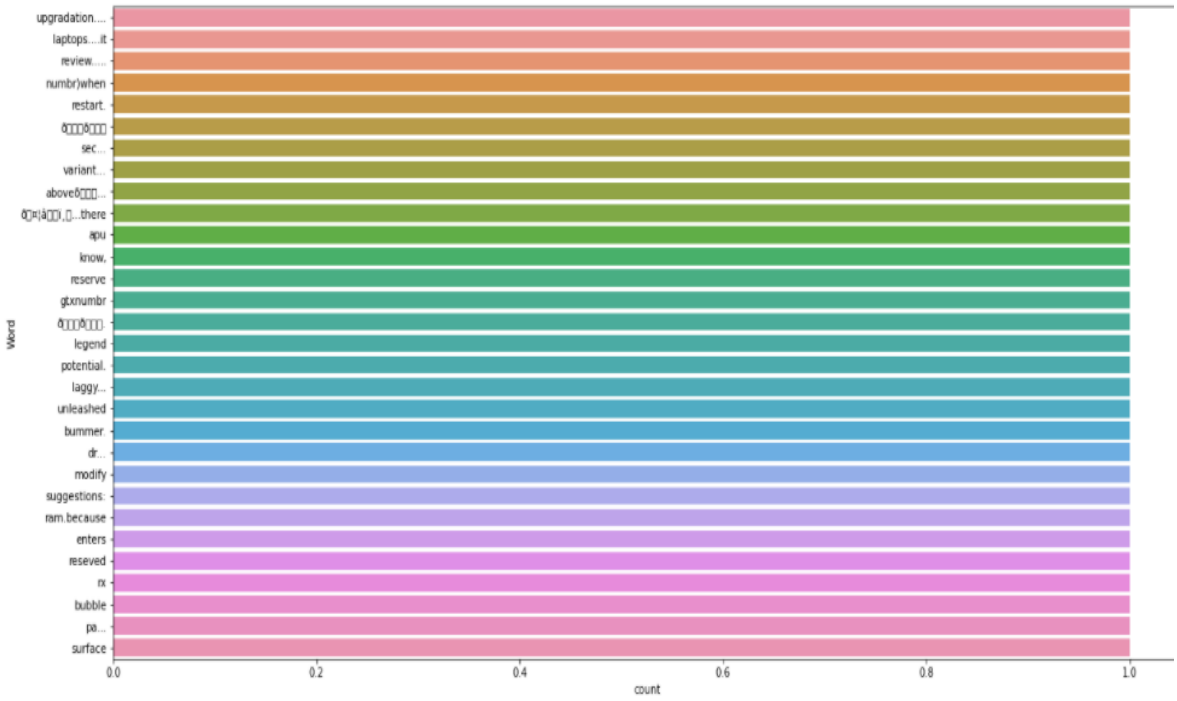
i)Comparing two ratings using hist plot:





Top 30 most frequently occuring and rarely occuring words:





- ✓ By seeing the above plot we can see that Good, product, quality.....are occurring frequently. And the second plot shows rarely occuring words.

3.5Run and Evaluate selected models

1.Model Building:

I have used 6 classification algorithms. First, I have created 6 different classification algorithms and are appended in the variable models. Followed by TFIDF vectorization and data balancing. Then, ran a for loop which contained the accuracy of the models along with different evaluation metrics.

In [36]:
```
# Defining the algorithms
KNN=KNeighborsClassifier(n_neighbors=6)
DT=DecisionTreeClassifier(random_state=6)

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

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 [37]:
```
Model= []
score= []
cvs=[]
for name,model in models:
    print('****************************************',name,'**********************************************************')
    print('\n')
    Model.append(name)
    model.fit(x_train,y_train)
    print(model)
    pre=model.predict(x_test)
    print('\n')
    AS=accuracy_score(y_test,pre)
    print('Accuracy_score = ',AS)
    score.append(AS*100)
    print('\n')
    sc= cross_val_score(model, x, y, cv=10, scoring='accuracy').mean()
    print('Cross_Val_Score = ',sc)
    cvs.append(sc*100)
    print('\n')
    print('classification_report\n',classification_report(y_test,pre))
    print('\n')
    cm=confusion_matrix(y_test,pre)
    print(cm)
    print('\n')
    plt.figure(figsize=(10,40))
    plt.subplot(911)
    plt.title(name)
    print(sns.heatmap(cm,annot=True))
    print('\n\n')
```

```
****************************************** KNeighborsClassifier ************************************************************

KNeighborsClassifier(n_neighbors=6)

Accuracy_score =  0.5720078089803273

Cross_Val_Score =  0.597724780953979

classification_report
              precision    recall  f1-score   support

           1       0.68      0.66      0.67      1078
           2       0.11      0.09      0.10       270
           3       0.23      0.10      0.14       489
           4       0.33      0.52      0.40      1316
           5       0.76      0.67      0.71      3506

    accuracy                           0.57      6659
   macro avg       0.42      0.41      0.40      6659
weighted avg       0.60      0.57      0.58      6659


[[ 713   97   27  174   67]
 [ 125   25   23   74   23]
 [  89   39   47  179  135]
 [  33   30   55  679  519]
 [  82   43   54  982 2345]]

AxesSubplot(0.125,0.808774;0.62x0.0712264)



****************************************** DecisionTreeClassifier ************************************************************

DecisionTreeClassifier(random_state=6)

Accuracy_score =  0.6512989938429193

Cross_Val_Score =  0.6413365573443676

classification_report
              precision    recall  f1-score   support

           1       0.69      0.71      0.70      1078
           2       0.14      0.09      0.11       270
           3       0.22      0.17      0.19       489
           4       0.48      0.37      0.42      1316
           5       0.75      0.85      0.79      3506

    accuracy                           0.65      6659
   macro avg       0.46      0.44      0.44      6659
weighted avg       0.62      0.65      0.63      6659


[[ 762   76   92   51   97]
 [ 136   24   49   31   30]
 [  85   35   84  100  185]
 [  45    8   79  491  693]
 [  79   29   79  343 2976]]
```

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

RandomForestClassifier()

Accuracy_score =  0.7032587475596936

Cross_Val_Score =  0.6954911114178162

classification_report
              precision    recall  f1-score   support

           1       0.71      0.86      0.78      1078
           2       0.21      0.03      0.05       270
           3       0.35      0.10      0.16       489
           4       0.67      0.30      0.42      1316
           5       0.72      0.94      0.82      3506

    accuracy                           0.70      6659
   macro avg       0.53      0.45      0.44      6659
weighted avg       0.66      0.70      0.65      6659


[[ 927   14   18   18  101]
 [ 175    8   23   13   51]
 [ 111   12   51   41  274]
 [  39    1   30  399  847]
 [  57    3   24  124 3298]]

AxesSubplot(0.125,0.808774;0.62x0.0712264)

****************************************** AdaBoostClassifier ****************************************************************

AdaBoostClassifier()

Accuracy_score =  0.6862892326175102

Cross_Val_Score =  0.6727260122844383

classification_report
              precision    recall  f1-score   support

           1       0.69      0.76      0.73      1078
           2       0.25      0.00      0.01       270
           3       0.30      0.10      0.15       489
           4       0.68      0.27      0.39      1316
           5       0.70      0.95      0.81      3506

    accuracy                           0.69      6659
   macro avg       0.53      0.42      0.41      6659
weighted avg       0.65      0.69      0.63      6659


[[ 820    1   14   32  211]
 [ 156    1   17   22   74]
 [ 102    1   49   44  293]
 [  44    1   51  353  867]
 [  62    0   31   66 3347]]

AxesSubplot(0.125,0.808774;0.62x0.0712264)
```

```
****************************************** MultinomialNB ************************************************************

MultinomialNB()

Accuracy_score =  0.6690193722781198

Cross_Val_Score =  0.6649490824558412

classification_report
              precision    recall  f1-score   support

           1       0.70      0.77      0.73      1078
           2       0.00      0.00      0.00       270
           3       0.40      0.01      0.02       489
           4       0.73      0.16      0.26      1316
           5       0.66      0.97      0.79      3506

    accuracy                           0.67      6659
   macro avg       0.50      0.38      0.36      6659
weighted avg       0.63      0.67      0.58      6659


[[ 827    0    1   13  237]
 [ 155    0    3   12  100]
 [ 108    0    6   18  357]
 [  40    0    3  204 1069]
 [  54    0    2   32 3418]]

AxesSubplot(0.125,0.808774;0.62x0.0712264)


****************************************** GradientBoostingClassifier ************************************************************

GradientBoostingClassifier()

Accuracy_score =  0.699204084697402

Cross_Val_Score =  0.6906556721936716

classification_report
              precision    recall  f1-score   support

           1       0.71      0.78      0.74      1078
           2       0.17      0.01      0.03       270
           3       0.39      0.08      0.14       489
           4       0.75      0.29      0.42      1316
           5       0.70      0.97      0.81      3506

    accuracy                           0.70      6659
   macro avg       0.54      0.43      0.43      6659
weighted avg       0.67      0.70      0.64      6659


[[ 841    9   10   17  201]
 [ 155    4   20   12   79]
 [ 104    7   41   45  292]
 [  35    1   20  382  878]
 [  47    3   13   55 3388]]

AxesSubplot(0.125,0.808774;0.62x0.0712264)
```

```
****************************************** BaggingClassifier ************************************************************

BaggingClassifier()

Accuracy_score =  0.67742904339991

Cross_Val_Score =  0.6743466777080475

classification_report
              precision    recall  f1-score   support

           1       0.67      0.80      0.73      1078
           2       0.12      0.03      0.05       270
           3       0.30      0.16      0.21       489
           4       0.54      0.35      0.43      1316
           5       0.74      0.88      0.80      3506

    accuracy                           0.68      6659
   macro avg       0.47      0.45      0.44      6659
weighted avg       0.63      0.68      0.64      6659


[[ 866   38   40   24  110]
 [ 172    9   30   17   42]
 [ 114   17   77   79  202]
 [  55    3   61  467  730]
 [  83    5   53  273 3092]]

AxesSubplot(0.125,0.808774;0.62x0.0712264)

****************************************** ExtraTreesClassifier ************************************************************

ExtraTreesClassifier()

Accuracy_score =  0.7019071932722631

Cross_Val_Score =  0.6926677022471555

classification_report
              precision    recall  f1-score   support

           1       0.71      0.85      0.77      1078
           2       0.21      0.03      0.05       270
           3       0.37      0.12      0.18       489
           4       0.68      0.30      0.41      1316
           5       0.72      0.94      0.82      3506

    accuracy                           0.70      6659
   macro avg       0.54      0.45      0.45      6659
weighted avg       0.66      0.70      0.65      6659


[[ 917   16   19   15  111]
 [ 171    8   26    9   56]
 [ 120   13   60   39  257]
 [  32    1   28  389  866]
 [  59    1   28  118 3300]]

AxesSubplot(0.125,0.808774;0.62x0.0712264)
```

2.Hyper Parameter Tunning:

I have done hyperparameter tuning for RandomForest Classifier for the parameters like ‘max_depth’, ‘min_samples_leaf’, ‘min_samples_split’ and ‘n_estimators’.

HyperParameter Tuning:

In [39]:
```
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,y)
print(clf.best_params_)
```

```
{'max_depth': 100, 'min_samples_leaf': 3, 'min_samples_split': 12, 'n_estimators': 1000}
```

In [40]:
```
#RandomForesetClassifier with best parameters

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

```
0.6939480402462832
[[ 859    0    0    1  218]
 [ 167    0    0    1  102]
 [ 113    0    1    3  372]
 [  39    0    0  306  971]
 [  49    0    0    2 3455]]
              precision    recall  f1-score   support

           1       0.70      0.80      0.75      1078
           2       0.00      0.00      0.00       270
           3       1.00      0.00      0.00       489
           4       0.98      0.23      0.38      1316
           5       0.68      0.99      0.80      3506

    accuracy                           0.69      6659
   macro avg       0.67      0.40      0.39      6659
weighted avg       0.74      0.69      0.62      6659
```

- ✓ And after doing hyperparameter tuning I got above parameters as best suitable parameters for our final model.
- ✓ I have trained my final model using these parameters and it was unable to increase the accuracy of the model.
- ✓ After training and building our final model I saved this model into .pkl file.

3. Saving the model and Predictions:

- I have saved my best model using .pkl as follows.

Model Saving:

In [43]:
```
#saving our model

import joblib
joblib.dump(RF,'Ratings_prediction.pkl')
```

Out[43]:
```
['Ratings_prediction.pkl']
```

4.CONCLUSION

4.1 Key Findings and Conclusions of the Study

- ✓ In this project I have collected data of reviews and ratings for different products from amazon.in and flipkart.com.
- ✓ We attempted to detect Ratings in commercial websites based on user reviews on a scale of 1 to 5 on a scale of 1 to 5. We employed natural language processing and machine learning approaches to do this.
- ✓ Then, in order to eliminate the problem of imbalance, I performed different text processing for the reviews column and chose an equal number of texts from each rating class. I analysed the text using different EDA methods.
- ✓ We have checked frequently occurring words in our data as well as rarely occurring words.
- ✓ After completing all of these procedures, I created a function to train and test multiple algorithms, and I chose Random Forest Classifier as our final model based on numerous assessment criteria.
- ✓ Finally, by doing hyperparameter tuning we got optimum parameters for our final model.

4.2 Learning Outcomes of the Study in respect of Data Science

I used flipkart.com to scrape the reviews and ratings of several technical gadgets. Because of advancements in computing technology, it is now possible to evaluate social data that could not previously be gathered, processed, or analysed. In property research, new analytical approaches (NLP) of machine learning can be applied. The power of visualisation has aided us in comprehending data through graphical representation, and it has helped me understand what data is attempting to communicate. To remove unrealistic values, punctuations, urls, data cleansing is one of the most crucial tasks. This research is an exploratory attempt to compare the results of six machine learning methods in calculating Rating.

To summarise, the use of NLP in rating classification is still in its infancy. We hope that our work has taken a tiny step forward in terms of giving methodological and empirical contributions to crediting institutes, as well as proposing an alternate approach to rating valuation.

4.3 Limitations of this work and Scope for Future Work

As we all know, the content of words in reviews is entirely up to the reviewer, and they may evaluate things differently depending on who they are. As a result, predicting ratings based on more accurate evaluations is challenging. We can still increase our precision by obtaining more data and performing intensive hyperparameter adjustment.

While we were unable to achieve our aim of maximum accuracy in the Ratings prediction project, we were able to develop a system that, with little tweaking and deep learning algorithms, can come quite close. There is always space for improvement in any endeavour. Because of the nature of this project, multiple algorithms can be merged as modules and their findings mixed to improve the accuracy of the final output. More algorithms can be added to this model to improve it even further. These algorithms' output, however, must be in the same format as the others. The modules are simple to add once that criterion is met, as seen in the code. The project gains a lot of modularity and versatility as a result of this.