FLIP ROBO

Ratings Prediction Report

Submitted by –

Vikram Purohit

ACKNOWLEDGMENT

I express my sincere gratitude to Flip Robo Technologies for giving me the opportunity to work on this project on Ratings Prediction using machine learning algorithms and NLTK suite.

I acknowledge my indebtedness to the author of the paper titled: “Review-Based Rating Prediction” for providing me with invaluable knowledge and insights into the importance of contextual information of user sentiments in determining the rating of products, the role of natural language processing tools and techniques in identifying the user sentiments towards various products based on their reviews and ratings and in helping build models to predict user ratings based on the input reviews.

INTRODUCTION

Business Problem Framing

A website has a forum for writing technical reviews of products and consists of repository of 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. An application to predict the rating by seeing the review is required to be built.

Therefore, a predictive model to accurately predict a user's rating based on input review is required to be made.

Conceptual Background of the Domain Problem

Predictive modelling, Classification algorithms are some of the machine learning techniques used along with the various libraries of the NLTK suite for Classification of comments. Using NLTK tools, the frequencies of malignant words occurring in textual data were estimated and given appropriate weightage, whilst filtering out words, and other noise which do not have any impact on the semantics of the comments and reducing the words to their base lemmas for efficient processing and accurate classification of the comments.

Review of Literature

A Research paper titled: “Review-Based Rating Prediction” by Tal Hadad was reviewed and studied to gain insights into the importance of contextual information of user sentiments in determining the rating of products, the role of natural language processing tools and techniques in identifying the user sentiments towards various products based on their reviews and ratings It is learnt that Contextual information about a user’s opinion of a product can be explicit or implicit and can be inferred in different ways such as user score ratings and textual reviews. A user may express in his review(s), their satisfaction / dissatisfaction with a product, based on its quality , features performance, and monetary worth and they may then give the product a rating score based on their opinion of it. These reviews have contextual data based on users’ experiences with the products and their opinions of them. The user ratings have a strong correlation with the contextual data carried in their reviews. Thus comparing the similarity in the reviews with the similarity in the scores based on those reviews can be a basis for predicting user ratings based on the context, inference and semantics of their reviews.

Motivation for the Problem Undertaken

Ratings are an important metric in e-commerce application to determine a product's quality, consumer demand, worth and profitability. The sentiment of a user towards a product is reflected in their rating score and their review of the product. This helps determine how the product is perceived by the consumers and in turn gives an idea about the acceptance of the product by the consumers. There is a strong positive correlation between the rating of a product and its consumer demand. Therefore, it is necessary to build a predictive model which can, with good accuracy predict what rating a user might give a particular product based on the user review. This helps understand user sentiment towards a product and determine the product's worth and acceptance by consumers.

Analytical Problem Framing

Mathematical/ Analytical Modelling of the Problem

Various Classification analysis techniques were used to build predictive models to understand the relationships that exist between user review and the corresponding user rating.

The user reviews are collected, processed and normalised. Based on the context of the reviews on various items, with similar ratings, prediction of the rating for a given review can be made based on similar reviews which already have corresponding ratings.

In order to predict ratings for user revies, models such as Logistic regression, Random Forest Classifier Boost Classifier, Extreme Gradient Boost Classifier, Multinomial Naïve Bayes Classifier, Complement Naïve Bayes Classifier and Passive Aggressive Classifier were used.

Data Sources and their formats.

The Dataset was compiled by scraping User review and rating Data for various products from https://amazon.in and https://www.flipkart.com/

The data was converted into a Pandas Dataframe under various Comment and Ratings columns and saved as a .csv file.

Dataset Description

	Unnamed: 0	Comment	Rating
0	0	Just unboxed the phone. Seems to be a better o...	5.0
1	1	This is perfect phone, good speed, decent spec...	5.0
2	2	A very satisfying experience with the device s...	5.0
3	3	It's a real monster. Everything is awesome fro...	5.0
4	4	I am using this phone for the last one week an...	5.0
...	...	...	...
22167	22167	Switching from a windows laptop to a MacBook w...	5.0
22168	22168	Superb I am happy to have this but only one th...	5.0
22169	22169	top class, must buy	5.0
22170	22170	Value for money Best combination of 8gb ram 51...	4.0
22171	22171	I got it at 60k 1) Superb performance. 2)Feel ...	5.0

22172 rows × 3 columns

The columns are:

- Comment: User review of a product.
- Rating: Corresponding user rating score for a User review

Data Preprocessing Done

• Rows with null values were removed.

• Columns: Unnamed: 0(just a series of numbers) was dropped since it doesn't contribute to building a good model for predicting the target variable values.

• The train and test dataset contents were then converted into lowercase.

• Punctuations, unnecessary characters etc were removed, currency symbols, phone numbers, web urls, email addresses etc were replaced with single words

• Tokens that contributed nothing to semantics of the messages were removed as Stop words. Finally retained tokens were lemmatized using WordNetLemmatizer().

• The string lengths of original comments and the cleaned comments were then compared.

Data Inputs- Logic- Output Relationships

The comment tokens so vectorised using TfidVectorizer are input and the corresponding rating is predicted based on their context as output by classification models.

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

The comment content made available in Dataset is assumed to be written in English Language in the standard Greco-Roman script. This is so that the Stopword package and WordNetLemmatizer can be effectively used.

Hardware and Software Requirements and Tools Used Hardware Used:

• Processor: AMD Ryzen 9 5900HX(8 Cores 16 Logical Processors)

• Physical Memory: 16.0GB (3200MHz)

• GPU: Nvidia RTX 3060 (192 bits), 6GB DDR6 VRAM, 3840 CUDA cores.

Software Used:

• Windows 10 Operating System

• Anaconda Package and Environment Manager: Anaconda is a distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows and provides a host of tools and environment for conducting Data Analytical and Scientific works. Anaconda provides all the necessary Python packages and libraries for Machine learning projects.

• Jupyter Notebook: The Jupyter Notebook is an open-source web application that allows data scientists to create and share documents that integrate live code, equations, computational output, visualizations, and other multimedia resources, along with explanatory text in a single document.

• Python3: It is open source, interpreted, high level language and provides great approach for object-oriented programming. It is one of the best languages used for Data Analytics And Data science projects/application. Python provides numerous libraries to deal with mathematics, statistics and scientific function.

• **Python Libraries used:**

o Pandas: For carrying out Data Analysis, Data Manipulation, Data Cleaning etc

o Numpy: For performing a variety of operations on the datasets.

o matplotlib.pyplot, Seaborn: For visualizing Data and various relationships between Feature and Label Columns

o imblearn.over_sampling: To employ SMOTE technique for balancing out the classes. Statsmodels: For performing statistical analysis

o sklearn for Modelling Machine learning algorithms, Data Encoding, Evaluation metrics, Data Transformation,Data Scaling, Component analysis,Feature selection etc.

o re, string: To perform regex operations

o Wordcloud: For Data Visualization

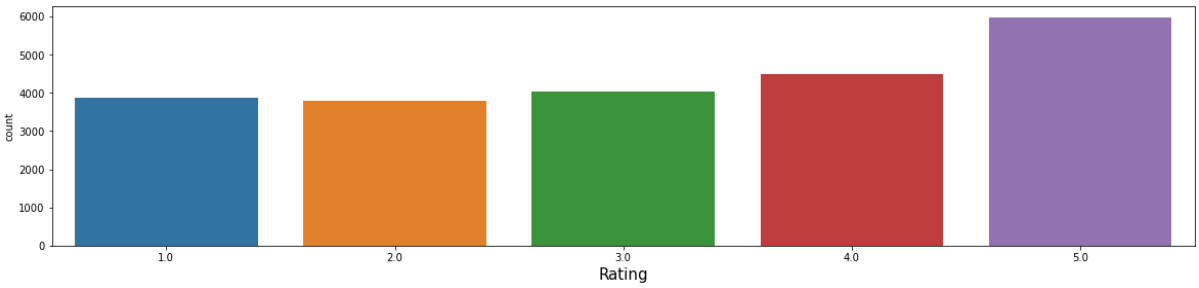
o NLTK: To use various Natural Language Processing Tools.

Exploratory Data Analysis

Visualizations

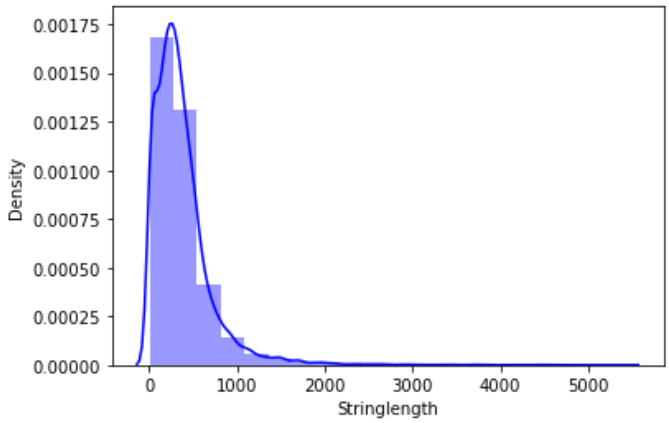
Barplots **,** Countplots , Distplots ,WordClouds were used to visualise the data of all the columns and their relationships with Target variable.

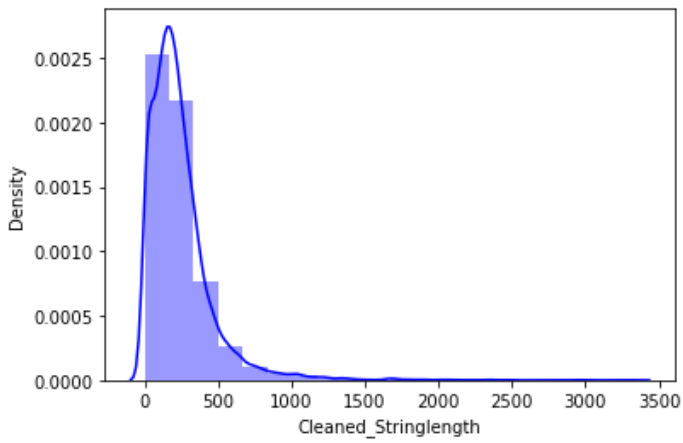
Analyzing the Target Variable



The rating classes 1.0-4.0 are fairly balanced, the 5.0 class represents the highest number of reviews.

Unprocessed vs Cleaned string lengths





Above graphs show that the string length of comments was drastically brought down after processing.

Word Clouds of the most frequent words used in reviews corresponding to various Rating Scores

From the graphs above the following observations are made:

• Reviews corresponding to 5.0 rating frequently carry words like: ‘great’,’best’,’perfect’,’better,’good’ etc indicating very high customer satisfaction and high quality product.

• Reviews corresponding to 4.0 rating frequently carry words like: ‘good’,’better’,’nice’,’value money’,’decent’, ‘quality’,’awesome’ etc indicating high customer satisfaction and good quality product.

• Reviews corresponding to 3.0 rating frequently carry words like: ‘good’,’well’,’purchased,’bad quality’,’issue’ etc indicating customer dissatisfaction and average to below average product quality.

• Reviews corresponding to 2.0 rating frequently carry words like: ‘problem’, ‘replacement,’stopped working’,’worst experience’,’quality’ etc indicating high customer dissatisfaction and below average product quality.

• Reviews corresponding to 1.0 rating frequently carry words like: ‘stopped’,’working’,’cheap’,’return’,’issue’,’wase money’,’poor quality’,’customer care’,’bad’,’used’,’worst’, ‘poor build quality’ etc indicate very high customer dissatisfaction and poor quality product.

Top 10 words and their corresponding Ratings, along with their counts

TOP 10 Words and Ratings,with their counts

	Five Star Words	Four Star Words	Three Star Words	Two Star Words	One Star Words
0	(num, 10271)	(num, 9947)	(num, 7194)	(num, 4721)	(num, 5455)
1	(good, 4675)	(good, 5121)	(good, 3534)	(quality, 2300)	(product, 2836)
2	(quality, 3978)	(quality, 3658)	(quality, 3123)	(good, 2047)	(phone, 2236)
3	(sound, 2935)	(sound, 2603)	(sound, 2229)	(phone, 1622)	(quality, 2086)
4	(earphone, 2858)	(phone, 2344)	(phone, 1986)	(sound, 1585)	(earphone, 1755)
5	(phone, 2688)	(earphone, 2093)	(earphone, 1675)	(product, 1500)	(one, 1525)
6	(price, 2183)	(bass, 2042)	(bass, 1457)	(earphone, 1356)	(working, 1504)
7	(bass, 2066)	(price, 1834)	(one, 1333)	(one, 1106)	(good, 1336)
8	(product, 1988)	(one, 1572)	(product, 1324)	(month, 989)	(sound, 1304)
9	(one, 1769)	(battery, 1547)	(camera, 1116)	(working, 906)	(month, 1282)

Model/s Development and Evaluation

The model algorithms used were as follows:

• Logistic Regression: It is a classification algorithm used to find the probability of event success and event failure. It is used when the dependent variable is binary(0/1, True/False, Yes/No) in nature. It supports categorizing data into discrete classes by studying the relationship from a given set of labelled data. It learns a linear relationship from the given dataset and then introduces a nonlinearity in the form of the Sigmoid function. It not only provides a measure of how appropriate a predictor(coefficient size)is, but also its direction of association (positive or negative).

• Multinomial Naïve Bayes Classifier: Multinomial Naive Bayes algorithm is a probabilistic learning method that is mostly used in Natural Language Processing (NLP). The algorithm is based on the Bayes theorem. It calculates the probability of each tag for a given sample and then gives the tag with the highest probability as output.

• XGBClassifier: XGBoost uses decision trees as base learners; combining many weak learners to make a strong learner. As a result it is referred to as an ensemble learning method since it uses the output of many models in the final prediction. It uses the power of parallel processing and supports regularization.

• RandomForestClassifier: A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. A random forest produces good predictions that can be understood easily. It reduces overfitting and can handle large datasets efficiently. The random forest algorithm provides a higher level of accuracy in predicting outcomes over the decision tree algorithm. Complement Naïve Bayes Classifier: Complement Naive Bayes is somewhat an adaptation of the standard Multinomial Naive Bayes algorithm. Complement Naive Bayes is particularly suited to work with imbalanced datasets. In complement Naive Bayes, instead of calculating the probability of an item belonging to a certain class, we calculate the probability of the item belonging to all the classes.

• Passive Aggressive Classifier: Passive-Aggressive algorithms do not require a learning rate and are called so

because if the prediction is correct, keep the model and do not make any changes. i.e., the data in the example is not enough to cause any changes in the model. If the prediction is incorrect, make changes to the model. i.e., some change to the model may correct it.

• AdaBoost Classifier: The basis of this algorithm is the Boosting main core: give more weight to the misclassified observations. the meta-learner adapts based upon the results of the weak classifiers, giving more weight to the misclassified observations of the last weak learner. The individual learners can be weak, but as long as the performance of each weak learner is better than random guessing, the final model can converge to a strong learner (a learner not influenced by outliers and with a great generalization power, in order to have strong performances on unknown data)

Balancing out the classes using SMOTE technique

SMOTE was used to balance the classes in the Label column.

```
from imblearn.over_sampling import SMOTE as sm

smt_x,smt_y = sm().fit_resample(X,y)
```

Train-Test Split

Best Random State was found to be 35

```
from sklearn.ensemble import RandomForestClassifier
maxAcc = 0
maxRS=0
for i in range(0,100):
    x_train,x_test,y_train,y_test = train_test_split(smt_x,smt_y,test_size = .30, random_state = i)
    modRF =  RandomForestClassifier()
    modRF.fit(x_train,y_train)
    pred = modRF.predict(x_test)
    acc  = accuracy_score(y_test,pred)
    if acc>maxAcc:
        maxAcc=acc
        maxRS=i
print(f"Best Accuracy is: {maxAcc} on random_state: {maxRS}")
```

```
Best Accuracy is: 0.7462319973205315 on random_state: 35
```

```
x_train,x_test,y_train,y_test = train_test_split(smt_x,smt_y,test_size = .30,random_state = 35)
```

Classification Model Building

Training The Models

```
RFC.fit(x_train,y_train)
XGBC.fit(x_train,y_train)
adbc.fit(x_train,y_train)
LOGR.fit(x_train,y_train)
MNB.fit(x_train,y_train)
CNB.fit(x_train,y_train)
```

```
pc.fit(x_train,y_train)
```

All Models have been trained.

Analyzing Accuracy of The Models

Classification Report consisting of Precision,Recall, Support and F1- score were the metrics used to evaluate the Model Performance. Precision is defined as the ratio of true positives to the sum of true and false positives. Recall is defined as the ratio of true positives to the sum of true positives and false negatives.The F1 is the weighted harmonic mean of precision and recall. The closer the value of the F1 score is to 1.0, the better the expected performance of the model is. Support is the number of actual occurrences of the class in the dataset. It doesn't vary between models; it just diagnoses the performance evaluation process.

AdaBoost Classifier Model Accuracy

```
adbcpred = adbc.predict(x_test)
accu = classification_report(y_test,adbcpred)
```

```
conf_matrx = confusion_matrix(y_test,adbcpred)
conf_matrx
```

```
array([[1126,  428,  124,   50,   62],
       [ 546,  617,  318,  194,  122],
       [ 204,  462,  492,  368,  232],
       [  88,  209,  298,  726,  493],
       [  78,  141,  136,  513,  930]], dtype=int64)
```

```
print(accu)
```

	precision	recall	f1-score	support
1.0	0.55	0.63	0.59	1790
2.0	0.33	0.34	0.34	1797
3.0	0.36	0.28	0.31	1758
4.0	0.39	0.40	0.40	1814
5.0	0.51	0.52	0.51	1798
accuracy			0.43	8957
macro avg	0.43	0.43	0.43	8957
weighted avg	0.43	0.43	0.43	8957

Complement Naive Bayes Model Accuracy

```
CNBpred = CNB.predict(x_test)
accu = classification_report(y_test,CNBpred)
```

```
conf_matrx = confusion_matrix(y_test,CNBpred)
```

```
conf_matrx
```

```
array([[1510,  115,   74,   45,   46],
       [ 552,  881,  179,  105,   80],
       [ 280,  162,  832,  216,  268],
       [ 152,   86,  151,  877,  548],
       [ 142,   51,   81,  233, 1291]], dtype=int64)
```

```
print(accu)
```

	precision	recall	f1-score	support
1.0	0.57	0.84	0.68	1790
2.0	0.68	0.49	0.57	1797
3.0	0.63	0.47	0.54	1758
4.0	0.59	0.48	0.53	1814
5.0	0.58	0.72	0.64	1798
accuracy			0.60	8957
macro avg	0.61	0.60	0.59	8957
weighted avg	0.61	0.60	0.59	8957

Logistic Regression Model Accuracy

```
LOGRpred = LOGR.predict(x_test)
accu = classification_report(y_test,LOGRpred)

```

```
conf_matrx = confusion_matrix(y_test,LOGRpred)
conf_matrx
```

```
array([[1402,  224,   93,   47,   24],
       [ 384, 1052,  223,   88,   50],
       [ 158,  254,  929,  217,  200],
       [  77,  117,  246,  885,  489],
       [  82,   60,  123,  348, 1185]], dtype=int64)
```

```
print(accu)
```

	precision	recall	f1-score	support
1.0	0.67	0.78	0.72	1790
2.0	0.62	0.59	0.60	1797
3.0	0.58	0.53	0.55	1758
4.0	0.56	0.49	0.52	1814
5.0	0.61	0.66	0.63	1798
accuracy			0.61	8957
macro avg	0.61	0.61	0.61	8957
weighted avg	0.61	0.61	0.61	8957

Passive Aggressive Classifier Model Accuracy

```
pcpred = pc.predict(x_test)
accu = classification_report(y_test,pcpred)
```

```
conf_matrx = confusion_matrix(y_test,pcpred)
```

```
conf_matrx
```

```
array([[1454,  140,   91,   74,   31],
       [ 130, 1370,  136,  101,   60],
       [  70,  143, 1227,  190,  128],
       [  65,   94,  212, 1155,  288],
       [  58,  109,  189,  407, 1035]], dtype=int64)
```

```
print(accu)
```

```
              precision    recall  f1-score   support

         1.0       0.82      0.81      0.82      1790
         2.0       0.74      0.76      0.75      1797
         3.0       0.66      0.70      0.68      1758
         4.0       0.60      0.64      0.62      1814
         5.0       0.67      0.58      0.62      1798

    accuracy                           0.70      8957
   macro avg       0.70      0.70      0.70      8957
weighted avg       0.70      0.70      0.70      8957
```

Random Forest Classifier Model Accuracy

```
RFCpred = RFC.predict(x_test)
accu = classification_report(y_test,RFCpred)
```

```
conf_matrx = confusion_matrix(y_test,RFCpred)
conf_matrx
```

```
array([[1585,  103,   44,   29,   29],
       [ 171, 1375,  113,   62,   76],
       [ 101,  118, 1263,  136,  140],
       [  51,   56,  157, 1212,  338],
       [  58,   53,  102,  295, 1290]], dtype=int64)
```

```
print(accu)
```

```
              precision    recall  f1-score   support

         1.0       0.81      0.89      0.84      1790
         2.0       0.81      0.77      0.79      1797
         3.0       0.75      0.72      0.73      1758
         4.0       0.70      0.67      0.68      1814
         5.0       0.69      0.72      0.70      1798

    accuracy                           0.75      8957
   macro avg       0.75      0.75      0.75      8957
weighted avg       0.75      0.75      0.75      8957
```

XGB Classifier Model Accuracy

```
XGBCpred = XGBC.predict(x_test)
accu = classification_report(y_test,XGBCpred)
```

```
conf_matrx = confusion_matrix(y_test,XGBCpred)
conf_matrx
```

```
array([[1438,  202,   77,   28,   45],
       [ 304, 1133,  192,   94,   74],
       [ 146,  252,  981,  197,  182],
       [  49,  105,  208,  998,  454],
       [  61,   80,   98,  264, 1295]], dtype=int64)
```

```
print(accu)
```

```
              precision    recall  f1-score   support

         1.0       0.72      0.80      0.76      1790
         2.0       0.64      0.63      0.63      1797
         3.0       0.63      0.56      0.59      1758
         4.0       0.63      0.55      0.59      1814
         5.0       0.63      0.72      0.67      1798

    accuracy                           0.65      8957
   macro avg       0.65      0.65      0.65      8957
weighted avg       0.65      0.65      0.65      8957
```

Model Cross Validation

Cross validation is a technique for assessing how the statistical analysis generalises to an independent data set.It is a technique for evaluating machine learning models by training several models on subsets of the available input data and evaluating them on the complementary subset of the data. Using cross-validation, there are high chances that we can detect over-fitting with ease. Model Cross Validation scores were then obtained for assessing how the statistical analysis generalises to an independent data set. The models were evaluated by training several models on subsets of the available input data and evaluating them on the complementary subset of the data.

XGB Classifier

```
print(cvs(XGBC,smt_x,smt_y,cv=5).mean())
```

Adaboost Classifier

```
print(cvs(adbc,smt_x,smt_y,cv=5).mean())
```

```
0.3980907720649807
```

Logistic Regression

```
print(cvs(LOGR,smt_x,smt_y,cv=5).mean())
```

```
0.5696198291743426
```

Random Forest Classifier

```
print(cvs(RFC,smt_x,smt_y,cv=5).mean())
```

```
0.7293585664042874
```

Multinomial Naive Bayes

```
print(cvs(MNB,smt_x,smt_y,cv=5).mean())
```

```
0.5515324066320549
```

Complement Naive Bayes

```
print(cvs(CNB,smt_x,smt_y,cv=5).mean())
```

```
0.5719979902863842
```

Passive Aggressive Classifier

```
print(cvs(pc,smt_x,smt_y,cv=5).mean())
```

```
0.6885613800033494
```

ROC AUC Scores

The score is used to summarize the trade-off between the true positive rate and false positive rate for a predictive model using different probability thresholds. The AUC value lies between 0.5 to 1 where 0.5 denotes a bad classifier and 1 denotes an excellent classifier.

Logistic Regression

```
roc_auc_score(y_test,lr_prob,multi_class='ovo')
```

```
0.856201546227344
```

Multinomial Naive Bayes

```
roc_auc_score(y_test,mnb_prob,multi_class='ovo')
```

```
0.8490587707773576
```

Complement Naive Bayes ¶

```
roc_auc_score(y_test,cnb_prob,multi_class='ovo')
```

```
0.858399172298617
```

Adaboost Classifier

```
roc_auc_score(y_test,adbc_prob,multi_class='ovo')
```

```
0.745466211649427
```

XGB Classifier

```
roc_auc_score(y_test,xgbc_prob,multi_class='ovo')
```

```
0.888636918334397
```

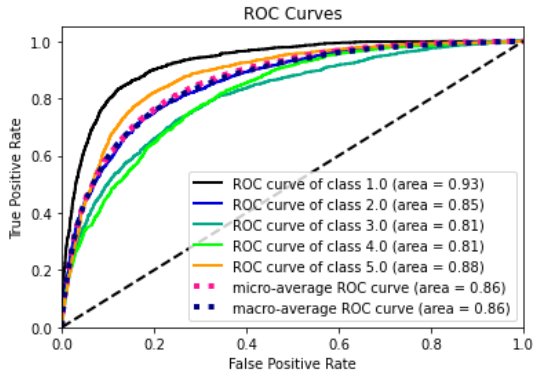
Random Forest Classifier

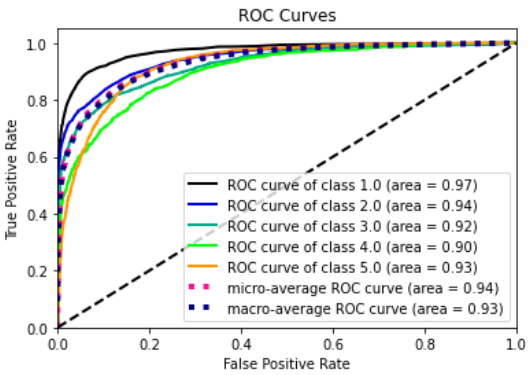
```
roc_auc_score(y_test,rf_prob,multi_class='ovo')
```

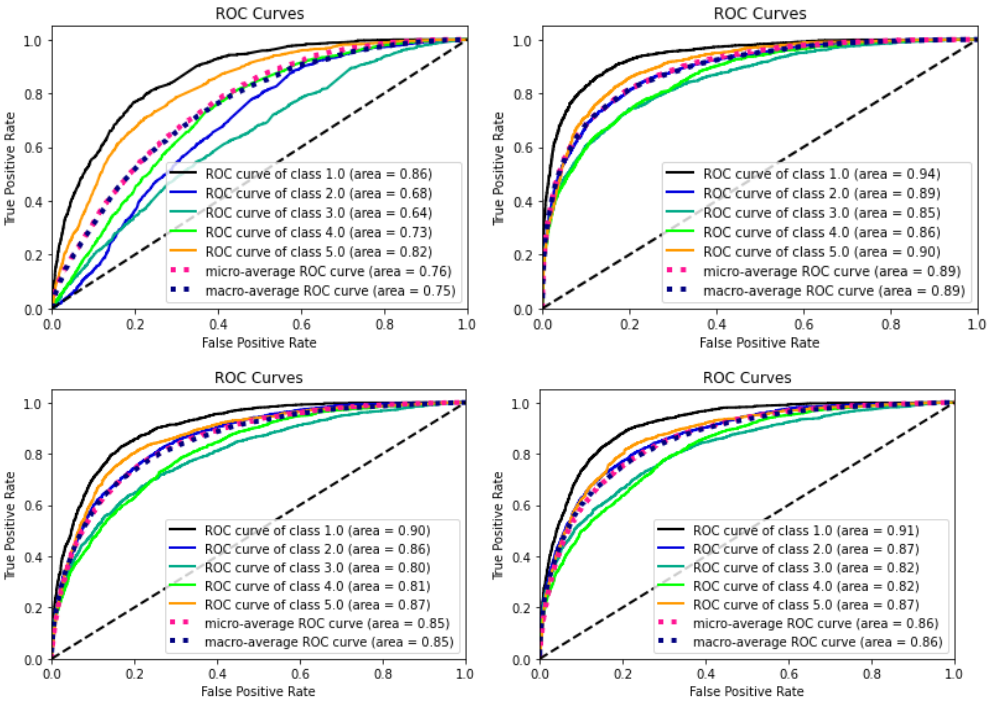
```
0.9331741740327718
```

ROC AUC Curves

The AUC-ROC curve helps us visualize how well our machine learning classifier is performing. ROC curves are appropriate when the observations are balanced between each class.







Interpretation of the Results

Based on comparing the above graphs, roc_auc_scores,Precision, Recall, Accuracy Scores with Cross validation scores, it is determined that Random Forest Classifier is the best models for the dataset.

Hyper Parameter Tuning

GridSearchCV was used for Hyper Parameter Tuning of the Random Forest Classifier model.

Hyper Parameter Tuning

```
parameter = {'n_estimators':[400,500,600],'max_depth': [80,90,95],'min_samples_leaf':[2,5,30],'min_samples_split':[1,2,5],'c
```

```
GridCV = GridSearchCV(RandomForestClassifier(),parameter,cv=5,n_jobs = -1,verbose = 1)
```

```
<IPython.core.display.Javascript object>
```

```
GridCV.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 486 candidates, totalling 2430 fits
```

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(), n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [80, 90, 95],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'min_samples_leaf': [2, 5, 30],
                         'min_samples_split': [1, 2, 5],
                         'n_estimators': [400, 500, 600]},
             verbose=1)
```

```
GridCV.best_params_
```

```
{'criterion': 'gini',
 'max_depth': 95,
 'max_features': 'auto',
 'min_samples_leaf': 2,
 'min_samples_split': 2,
 'n_estimators': 500}
```

```
Best_mod = RandomForestClassifier(n_estimators = 500,criterion = 'gini', max_depth= 95, max_features = 'auto',min_samples_le
Best_mod.fit(x_train,y_train)
rfpred = Best_mod.predict(x_test)
acc = accuracy_score(y_test,rfpred)
print(acc*100)
conf_matrx = confusion_matrix(y_test,rfpred)
conf_matrx
```

```
70.5593390644189
```

```
array([[1558,  105,   51,   20,   56],
       [ 288, 1211,  127,   48,  123],
       [ 150,  161, 1099,  112,  236],
       [  68,   73,  144, 1052,  477],
       [  86,   38,   54,  220, 1400]], dtype=int64)
```

Based on the input parameter values and after fitting the train datasets The Random Forest Classifier model was further tuned based on the parameter values yielded from GridsearchCV. The Random Forest Regressor model displayed an accuracy of 70.59%.

CONCLUSION

Key Findings and Conclusions of the Study and Learning Outcomes with respect to Data Science

The final model performed with 70.59% accuracy, Recall score of 0.81 for ratings 1.0 and 2.0, 0.75 for 3.0 rating score, 0.70 for 4.0 rating score and 0.69 for 5.0 rating score, which means that the model is optimized better to predict ratings for bad reviews and average reviews.

Learning Outcomes of the Study in respect of Data Science

Data cleaning was a very important step in removing null values from the dataset.

Visualising data helped identify class composition of label column and the most frequently occurring words in reviews corresponding to each of the rating scores.

The various data pre-processing and feature engineering steps in the project lent cognizance to various efficient methods for processing textual data. The NLTK suite is very useful in pre-processing text-based data and building classification models.

Limitations of this work and Scope for Future Work

A small dataset to work with posed a challenge in building highly accurate models. By training the models on more diverse data sets, longer comments, and a more balanced dataset, more accurate and efficient classification models can be built.