Rating Prediction Based on Reviews

Submitted by:

Bramee Venkatesan

ACKNOWLEDGMENT

Thanks for giving me the opportunity to work in Flip Robo Technologies as Intern and would like to express my gratitude to Data Trained Institute as well for trained me in Data Science Domain.

This helps me to do my projects well and understand the concepts.

Resources Referred – Google, GitHub, Blogs for conceptual referring

Links – Medium.com, towardsdatascience.com, machinelearningmastery.com

INTRODUCTION

- Business Problem Framing

  A client who has a website where people write different reviews for technical products. Now they are adding a new feature to their website i.e. The reviewer will have to add stars(rating) as well with the review.

  The rating is out 5 stars, and it only has 5 options available 1 star, 2 stars, 3 stars, 4 stars, 5 stars.

  Now they want to predict ratings for the reviews which were written in the past and they don't have a rating.

  So, we must build an application which can predict the rating by seeing the review.

- Motivation for the Problem Undertaken

  This will help the customers to understand the product before buy from the website.

  The rating system is easy for the customers to understand about the product than going over all the reviews and it will help the client to bring modifications to the product.

Analytical Problem Framing

- Mathematical/ Analytical Modelling of the Problem
  Target variable is rating which is in ordinal format .so the problem is Multi class classification.

- Data Sources and their formats
  Dataset is extracted by multiple websites through Scraping, and it has 20184 rows and 2 columns.

```
#Loading the dataset,

rr = pd.read_excel("ratingdata.xlsx")
rr
```

	Reviews	rating
0	Somewat not satisfied ..but ok on	1
1	Very comfortable	1
2	Not comfortable , I wanted something soft but ...	1
3	I got my black jeans today. I was really confu...	1
4	Ok ok	1
...	...	...
20179	It is Matt lipstick also creamy. Beautiful nud...	4
20180	Too good and worth it's name LIKE A BOSS	4
20181	Nice natural colour. Can be used like a base o...	4
20182	Good one, but almost 80% lipstick get transferred	4
20183	Super creamy lipstick. Applies like butter. Ve...	4

20184 rows × 2 columns

- Data Pre-processing Done

Data has null values in reviews column, and it has been removed.

As our input is review which has regular expressions, punctuations, and emoji, need to treat it using lemmatization, stemming word count vectorizer.

Target variable is im-balanced and need to balanced using resampling technique.

```
stop_words = stopwords.words("English") # stop words defining

lemmatizer = WordNetLemmatizer() #defining Lemmatizer
```

```
rr['Reviews'] = rr['Reviews'].replace('\n','') # replacing \n values

def clean_Reviews(text):

    lower_text = text.lower()  # converting to lower case

    text = re.sub(r'[0-9]'," ", text)  # removing numbers

    #text = re.sub('W*dw*','',text)   # removing words and digits combination

    text = re.sub(r'[^\w\s]', ' ', text)  # removing punctuations
    text = re.sub(r'\_',' ',text)


   ## clean_words = re.sub(r'[^\x00-\x7f]',r'', text) #Removing all the non-ascii characters

    text = " ".join(text.split()) #Removing the unwanted white spaces

    tokenized_text = word_tokenize(text) #Splitting data into words

    #Removing remaining tokens that are not alphabetic, Removing stop words and Lemmatizing the text
    removed_stop_text = [lemmatizer.lemmatize(word) for word in tokenized_text if word not in stop_words if word.isalpha()]

    return " ".join(removed_stop_text)
```

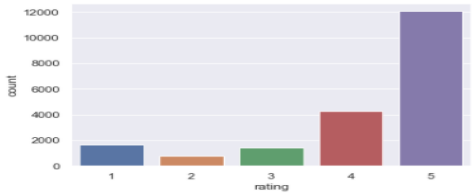
```
rr['Reviews'] = rr['Reviews'].apply(clean_Reviews)
```

```
tf_vec = TfidfVectorizer(max_features = 10000, stop_words='english') #
```

```
#Let's Separate the input and output variables represented by X and y
X = tf_vec.fit_transform(rr['Reviews'])
y = rr['rating']
```

```
sns.countplot(y)
```

```
<AxesSubplot:xlabel='rating', ylabel='count'>
```

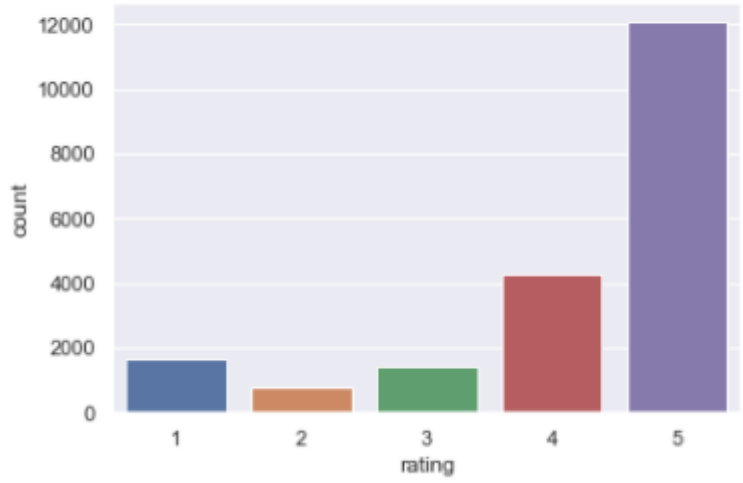



```
# using re-sample techinique to imbalance the balanced class,

from sklearn.utils import resample
from imblearn.over_sampling import SMOTE

sm = SMOTE()
x_over,y_over = sm.fit_resample(X,y)
```

- Data Inputs- Logic- Output Relationships

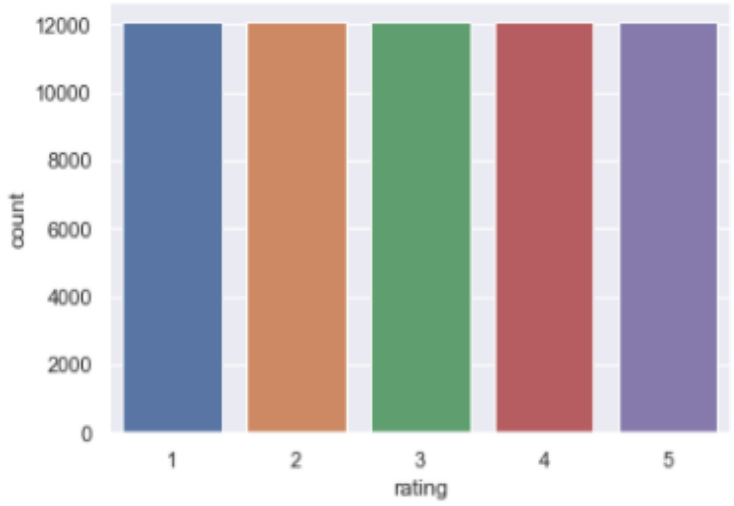
We can see that we have more 5 rating in our dataset, and it is imbalanced.

```
sns.set_theme()
sns.countplot(rr['rating'])
plt.show()
```

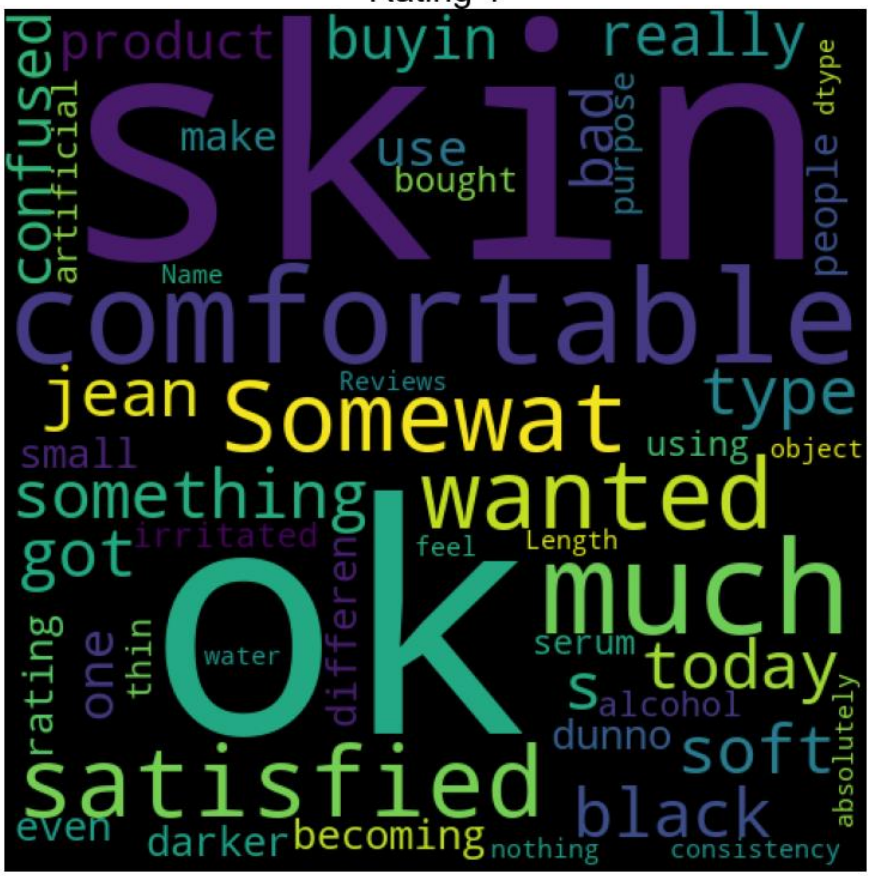



```
sns.countplot(y_over)
```

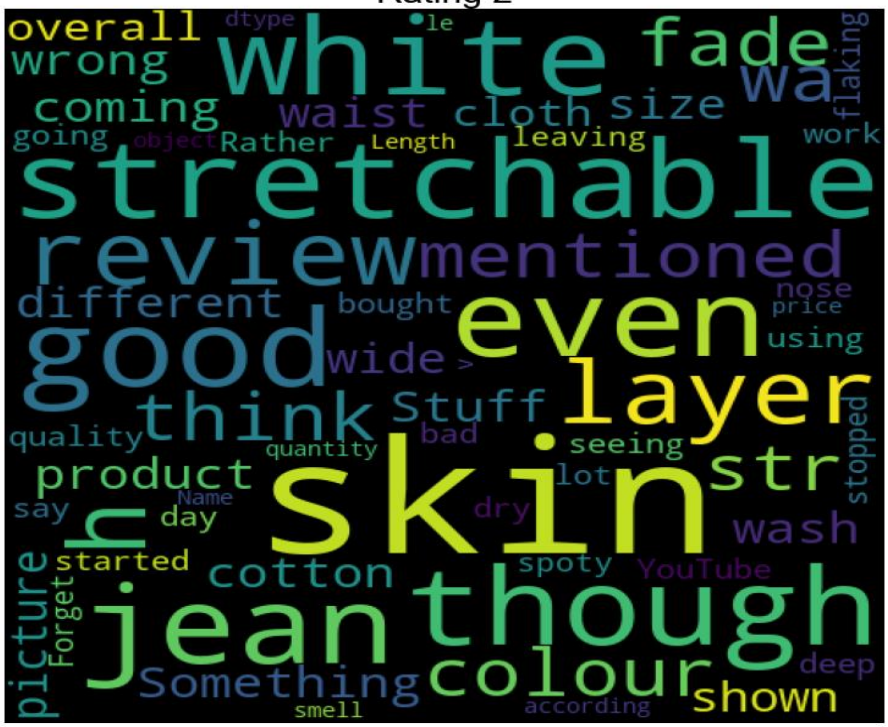
```
<AxesSubplot:xlabel='rating', ylabel='count'>
```



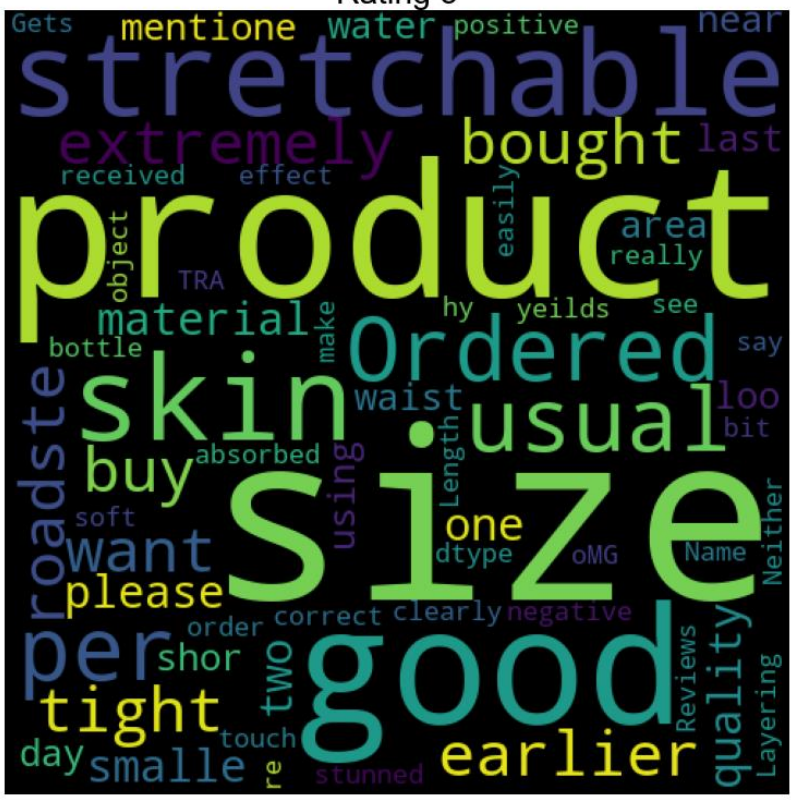
Rating 1



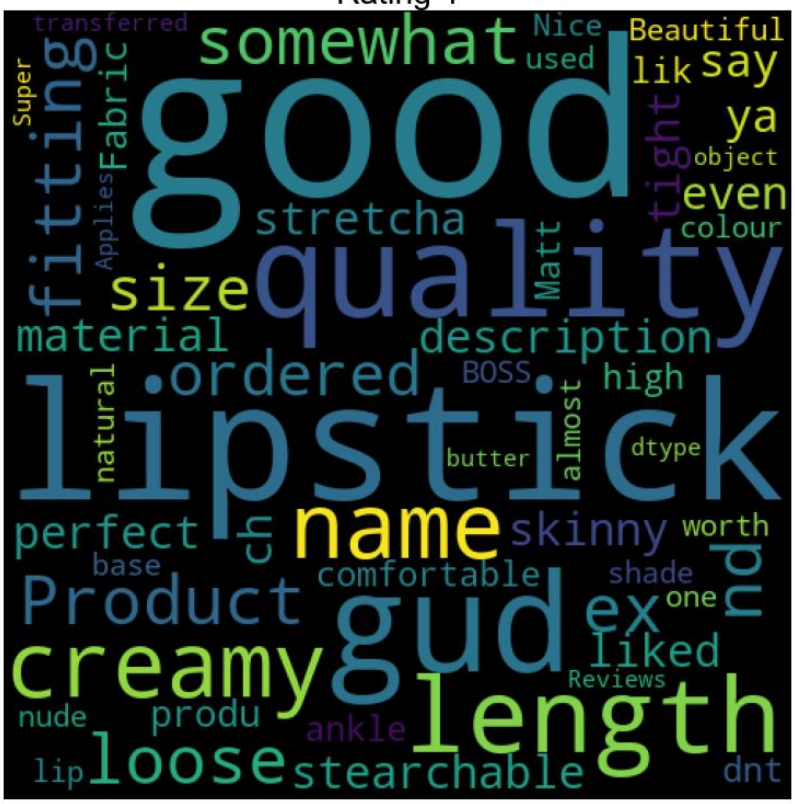
Rating 2



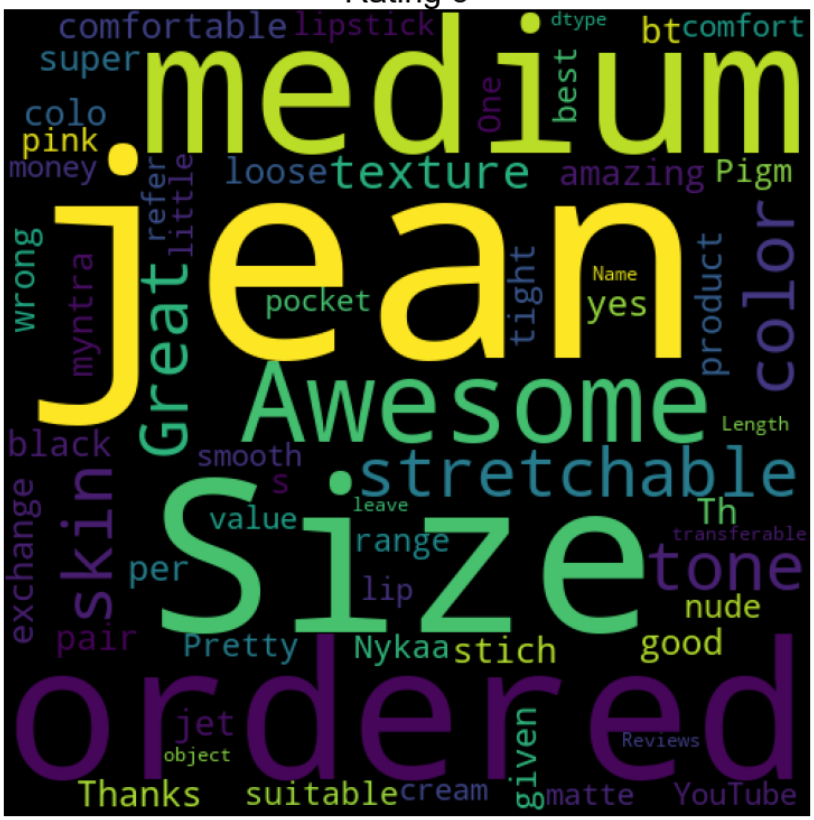
Rating 3



Rating 4



Rating 5



The frequent words are categorized based on rating 1 to rating 5 in the above image.

- Hardware and Software Requirements and Tools Used

  Model training was done on Jupiter Notebook. Kernel Version is Python3. Hardware -- > Intel 8GB RAM, i5 processor. The above libraries and packages used in this project for building a model.

```
: #importing required libraries

  import pandas as pd
  import numpy as np
  from sklearn.model_selection import train_test_split, cross_val_score, RandomizedSearchCV
  from sklearn.metrics import f1_score,accuracy_score,classification_report,confusion_matrix,roc_curve,roc_auc_score
  import seaborn as sns
  import matplotlib.pyplot as plt

  import warnings
  warnings.filterwarnings("ignore")
```

```
: import re
  import nltk
  import string
  from nltk.corpus import stopwords
  from wordcloud import WordCloud
  from nltk.tokenize import word_tokenize
  from nltk.stem import WordNetLemmatizer
  from sklearn.feature_extraction.text import TfidfVectorizer
```

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)
  As the target variable variables is multi class classification problem and it is imbalanced.

  Using resampling techniques SMOTE, we are going to balance the target variables.
- Testing of Identified Approaches (Algorithms)
  - Logistic Regression
  - Passive Aggressive Classifier
  - Decision Tree Classifier
  - Random Forest Classifier
- Run and evaluate selected models

```
# Logistic Regression

from sklearn.linear_model import LogisticRegression

lor = LogisticRegression()
lor.fit(x_train,y_train)
y_pred = lor.predict(x_test)
scr_lor = cross_val_score(lor,x_over,y_over,cv=5)


print("F1 score \n", f1_score(y_test,y_pred, average = 'micro'))
print("CV Score :", scr_lor.mean())
print("-----------------------------------------------------------\n")
print("Classification Report \n", classification_report(y_test,y_pred))
print("-----------------------------------------------------------\n")
print("Confusion Matrix \n", confusion_matrix(y_test,y_pred))
```

```
F1 score
 0.6899693531019631
CV Score : 0.640255114718794
-----------------------------------------------------------

Classification Report
              precision    recall  f1-score   support

           1       0.77      0.82      0.80      2402
           2       0.76      0.78      0.77      2484
           3       0.63      0.66      0.64      2348
           4       0.59      0.58      0.58      2389
           5       0.69      0.61      0.64      2450

    accuracy                           0.69     12073
   macro avg       0.69      0.69      0.69     12073
weighted avg       0.69      0.69      0.69     12073

-----------------------------------------------------------

Confusion Matrix
 [[1972  212  154   40   24]
 [ 282 1933  186   38   45]
 [ 146  206 1561  332  103]
 [  63   89  355 1377  505]
 [  83   95  237  548 1487]]
```

```python
#passive Aggressive Classifier

from sklearn.linear_model import PassiveAggressiveClassifier

pac = PassiveAggressiveClassifier()
pac.fit(x_train,y_train)
y_pred = pac.predict(x_test)
scr_pac = cross_val_score(pac,x_over,y_over,cv=5)


print("F1 score \n", f1_score(y_test,y_pred,average = 'micro'))
print("CV Score :", scr_pac.mean())
print("-----------------------------------------------------------\n")
print("Classification Report \n", classification_report(y_test,y_pred))
print("-----------------------------------------------------------\n")
print("Confusion Matrix \n", confusion_matrix(y_test,y_pred))
```

```
F1 score
 0.7368508241530689
CV Score : 0.7004886937795081
-----------------------------------------------------------

Classification Report
              precision    recall  f1-score   support

           1       0.88      0.86      0.87      2402
           2       0.75      0.90      0.82      2484
           3       0.73      0.73      0.73      2348
           4       0.64      0.61      0.62      2389
           5       0.68      0.58      0.63      2450

    accuracy                           0.74     12073
   macro avg       0.73      0.74      0.73     12073
weighted avg       0.73      0.74      0.73     12073

-----------------------------------------------------------

Confusion Matrix
 [[2069  173  102   32   26]
 [  89 2230  115   27   23]
 [  47  234 1713  209  145]
 [  62  165  224 1460  478]
 [  88  176  205  557 1424]]
```

```python
# Decision tree

from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier()
dt.fit(x_train,y_train)
y_pred = dt.predict(x_test)
scr_dt = cross_val_score(dt,x_over,y_over,cv=5)


print("F1 score \n", f1_score(y_test,y_pred, average = 'micro'))
print("CV Score :", scr_dt.mean())
print("-----------------------------------------------------------\n")
print("Classification Report \n", classification_report(y_test,y_pred))
print("-----------------------------------------------------------\n")
print("Confusion Matrix \n", confusion_matrix(y_test,y_pred))
```

```
F1 score
 0.7547419862503106
CV Score : 0.7278058477594632
-----------------------------------------------------------

Classification Report
              precision    recall  f1-score   support

           1       0.85      0.86      0.86      2402
           2       0.86      0.88      0.87      2484
           3       0.75      0.80      0.77      2348
           4       0.62      0.66      0.64      2389
           5       0.68      0.58      0.62      2450

    accuracy                           0.75     12073
   macro avg       0.75      0.75      0.75     12073
weighted avg       0.75      0.75      0.75     12073

-----------------------------------------------------------

Confusion Matrix
 [[2074  107   86   67   68]
 [ 106 2182  105   47   44]
 [  80   89 1875  196  108]
 [  63   75  226 1572  453]
 [ 104   75  217  645 1409]]
```

```
# Random forest

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
y_pred = rfc.predict(x_test)
scr_rfc = cross_val_score(rfc,x_over,y_over,cv=5)

print("F1 score \n", f1_score(y_test,y_pred, average = 'micro'))
print("CV Score :", scr_rfc.mean())
print("-----------------------------------------------------------\n")
print("Classification Report \n", classification_report(y_test,y_pred))
print("-----------------------------------------------------------\n")
print("Confusion Matrix \n", confusion_matrix(y_test,y_pred))
```

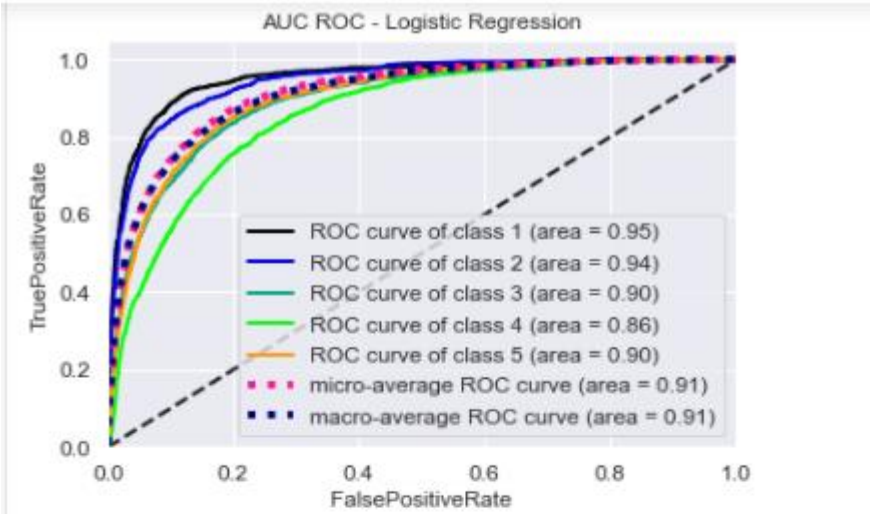
```
F1 score
 0.8757558187691544
CV Score : 0.8284436345564483
-----------------------------------------------------------

Classification Report
              precision    recall  f1-score   support

           1       0.94      0.96      0.95      2402
           2       0.96      0.96      0.96      2484
           3       0.87      0.91      0.89      2348
           4       0.77      0.80      0.79      2389
           5       0.83      0.74      0.78      2450

    accuracy                           0.88     12073
   macro avg       0.87      0.88      0.87     12073
weighted avg       0.88      0.88      0.87     12073

-----------------------------------------------------------

Confusion Matrix
 [[2310   13   29   27   23]
 [  49 2375   44    9    7]
 [  22   32 2146  118   30]
 [  17   20  121 1920  311]
 [  64   28  118  418 1822]]
```

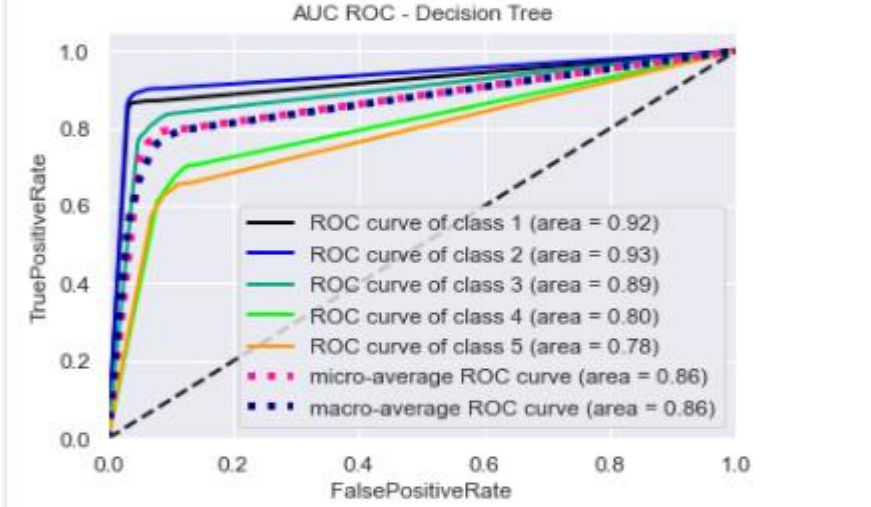
- Key Metrics for success in solving problem under consideration

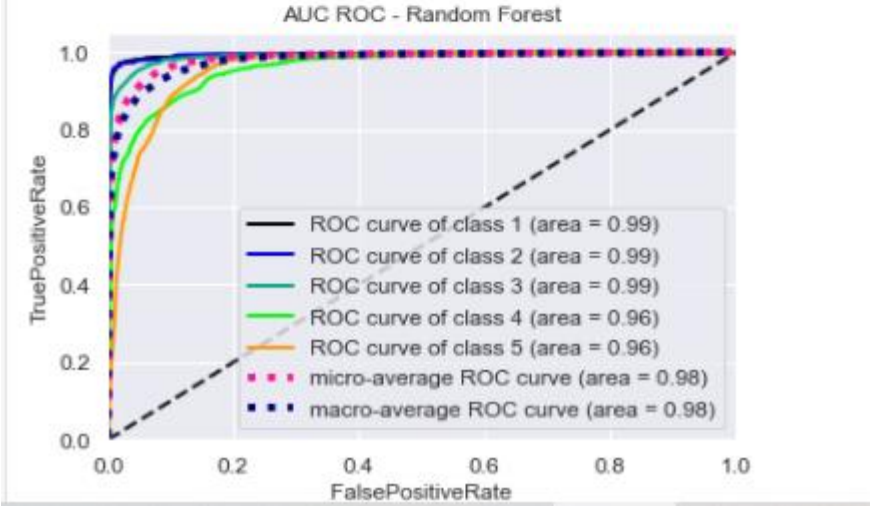
  Key Metrices used were the F1 Score as the classes are imbalanced, Cross validation Score and AUC & ROC Curve.

  Classification report will show the detailed report of accuracy, precision, re-call and support which we used.

  Confusion Matrix also used to show the positive and negative.


AUC ROC - Logistic Regression
TruePositiveRate
FalsePositiveRate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve of class 1 (area = 0.95)
ROC curve of class 2 (area = 0.94)
ROC curve of class 3 (area = 0.90)
ROC curve of class 4 (area = 0.86)
ROC curve of class 5 (area = 0.90)
micro-average ROC curve (area = 0.91)
macro-average ROC curve (area = 0.91)


AUC ROC - Decision Tree
TruePositiveRate
FalsePositiveRate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve of class 1 (area = 0.92)
ROC curve of class 2 (area = 0.93)
ROC curve of class 3 (area = 0.89)
ROC curve of class 4 (area = 0.80)
ROC curve of class 5 (area = 0.78)
micro-average ROC curve (area = 0.86)
macro-average ROC curve (area = 0.86)


AUC ROC - Random Forest
TruePositiveRate
FalsePositiveRate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve of class 1 (area = 0.99)
ROC curve of class 2 (area = 0.99)
ROC curve of class 3 (area = 0.99)
ROC curve of class 4 (area = 0.96)
ROC curve of class 5 (area = 0.96)
micro-average ROC curve (area = 0.98)
macro-average ROC curve (area = 0.98)

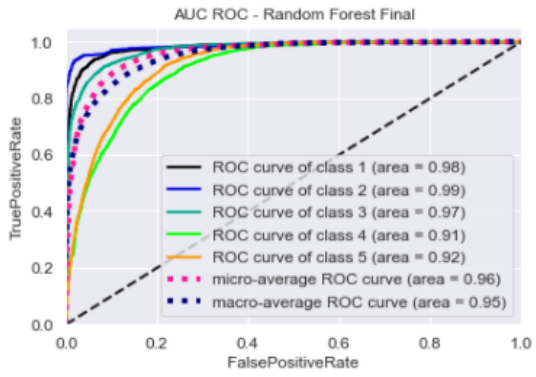
- Interpretation of the Results

  Random Forest is the best model and applied the same model to increase the hyper parameter tuning.

```
final = RandomForestClassifier(n_estimators=100 , criterion = 'entropy', max_depth = 128,
                               min_samples_leaf =2,min_samples_split =3)

final.fit(x_train,y_train)
pred = final.predict(x_test)

print("F1 score \n", f1_score(y_test,pred, average='micro'))
print("Accuracy score \n", accuracy_score(y_test,pred))
print("-----------------------------------------------------------\n")
print("Classification Report \n", classification_report(y_test,pred))
print("-----------------------------------------------------------\n")
print("Confusion Matrix \n",confusion_matrix(y_test,pred))
```

```
F1 score
 0.802865899113725
Accuracy score
 0.8028658991137249
-----------------------------------------------------------

Classification Report
              precision    recall  f1-score   support

           1       0.86      0.93      0.90      2402
           2       0.94      0.91      0.92      2484
           3       0.80      0.85      0.82      2348
           4       0.70      0.60      0.65      2389
           5       0.70      0.73      0.71      2450

    accuracy                           0.80     12073
   macro avg       0.80      0.80      0.80     12073
weighted avg       0.80      0.80      0.80     12073

-----------------------------------------------------------

Confusion Matrix
 [[2235   24   60   45   38]
 [ 128 2249   69   23   15]
 [  66   32 1988  164   98]
 [  77   39  216 1431  626]
 [  81   49  159  371 1790]]
```

```
#final model roc curve

probas4 = final.predict_proba(x_test)
skplt.metrics.plot_roc(y_test,probas4)

plt.xlabel("FalsePositiveRate")
plt.ylabel("TruePositiveRate")
plt.title('AUC ROC - Random Forest Final')
plt.show()
```



CONCLUSION

- Key Findings and Conclusions of the Study

  Rating has been predicted through the given reviews and here the data collection has been done from various websites. The rating has been classified as 1 as very poor and 5 has excellent that will help the client to figure out rating based on reviews.

- Limitations of this work and Scope for Future Work

  The project is different than the earlier one as it is NLP with ML Project and here the data needs to be collected by self from different e-commerce websites.

  I didn’t include Boosting, SVC algorithm as it is consuming more time.