RATING PROJECT

Submitted by:

Anusha K

INTRODUCTION

One of the Client who has a website where people write different reviews for many technical products. Now they are adding a new feature to their website i.e., the reviewer will have to add stars(rating) as well as the review. The rating is out of 5 starts. Now the task is to predict the rating by seeing the review which were written in the past.

Once the rating is applied to the existing reviews, when they promote their products, they receive more specific reviews which will help them to enhance the customer experience.

And because of that, they will have more granular control on how to handle their customers i.e., they can target the customers who are not satisfied with the product and come up with a business plan.

Analytical Problem Framing

- Mathematical/ Analytical Modelling of the Problem

  The rating and the review data for this project was collected from the different websites like Amazon, Flipkart etc to train the model and to predict the rating of a particular review.

  This will be a classification problem as we have 5 ratings as categorical values in the Target variable.

- Data Sources and their formats

  We have collected the data from different websites using the selenium web-scraping technology. The reviews and the rating were scraped from the internet for the electronic products such as Laptops, Phones, Headphones, Smart Watches, Professional cameras, Printer, Monitors, Home-theatres, and Routers.

  The important fields would be the

  - Rating of the product
  - Review of the product

  Rating is of integer type and the Review is text data.

- Data Pre-processing Done

  Even though the data was scraped from internet, there are Null values is the dataset and the entries are removed as they do not have any weight with just 1 attribute.

  We know the Review is of text data, this needs to be carefully treated to obtain the final output.

  In the text pre-processing, each review was converted to lowercase, all the punctuations are removed from the review, Word tokenization has been done on the review, now the data is in the form of list of words.

From this list of tokenized words, stopwords has been removed. Numbers or the alphanumeric values are replaced or removed by appropriate values.

On this data, Lemmatization has been applied, this will retain the real meaning of the words. We also have emojis in our review data. This needs to be removed. Now the data is ready for the feature extraction.

Here we will use CountVectorizer transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text.

- Data Inputs- Logic- Output Relationships

  Review is the input data and rating are the output variable. If the input variable has positive comments the rating will be higher. If the review is negative, the rating is at lower side.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  The above problem is rating classification problem. Initially the data was collected from various websites and followed by data analysis. As this input data is text, we need to clean the data and then building model and selecting the best model.

- Testing of Identified Approaches (Algorithms)

  In this classification problem we have considered below set of machine learning algorithms

  - GaussianNB,
  - MultinomialNB
  - DecisionTreeClassifier
  - GradientBoostingClassifier
  - RandomForestClassifier
  - SVC
  - KNeighborsClassifier

- Run and Evaluate selected models

  The highest accuracy was obtained from the RandomForestClassifier. Below is the code and the result snippet from each of the algorithms:

```
gnb=GaussianNB()
dtc=DecisionTreeClassifier()
gbc=GradientBoostingClassifier()
rfc=RandomForestClassifier()
svc=SVC()
knc=KNeighborsClassifier()
mnb=MultinomialNB()

model = [gnb, dtc, gbc, rfc, svc, knc, mnb]

for m in model:
    m.fit(x_train, y_train)
    y_pred=m.predict(x_test)
    acc=accuracy_score(y_test, y_pred)

    print("Accuracy_Score of ",m, "is", acc)
    print("Confusion_Matrix :", confusion_matrix(y_test, y_pred))
    print("Classification_Report :", classification_report(y_test, y_pred))
    print('\n')
    print("**********************************************************************")
    print("**********************************************************************")
```

Results:

```
Accuracy_Score of  GaussianNB() is 0.243123089747152
Confusion_Matrix : [[255 461 166  41  21]
 [ 35 123  35  14   9]
 [ 25 145 109  35  21]
 [ 63 234 115 170  56]
 [112 652 250 234 218]]
Classification_Report :               precision    recall  f1-score   support

           1       0.52      0.27      0.36       944
           2       0.08      0.57      0.13       216
           3       0.16      0.33      0.22       335
           4       0.34      0.27      0.30       638
           5       0.67      0.15      0.24      1466

    accuracy                           0.24      3599
   macro avg       0.35      0.32      0.25      3599
weighted avg       0.49      0.24      0.27      3599
```

```
**************************************************************************
**************************************************************************
Accuracy_Score of  DecisionTreeClassifier() is 0.612948041122534
Confusion_Matrix : [[ 688   53   56   58   89]
 [  74   82   14   14   32]
 [  79   16  134   40   66]
 [  67   19   38  261  253]
 [ 100   27   61  237 1041]]
Classification_Report :               precision    recall  f1-score   support

           1       0.68      0.73      0.70       944
           2       0.42      0.38      0.40       216
           3       0.44      0.40      0.42       335
           4       0.43      0.41      0.42       638
           5       0.70      0.71      0.71      1466

    accuracy                           0.61      3599
   macro avg       0.53      0.53      0.53      3599
weighted avg       0.61      0.61      0.61      3599
```

```
**************************************************************************
**************************************************************************
Accuracy_Score of  GradientBoostingClassifier() is 0.6096137816060017
Confusion_Matrix : [[ 722    5    9   41  167]
 [ 114   11    2   33   56]
 [ 103    4   33   54  141]
 [  61    1    6  125  445]
 [  74    1    8   80 1303]]
Classification_Report :               precision    recall  f1-score   support

           1       0.67      0.76      0.72       944
           2       0.50      0.05      0.09       216
           3       0.57      0.10      0.17       335
           4       0.38      0.20      0.26       638
           5       0.62      0.89      0.73      1466

    accuracy                           0.61      3599
   macro avg       0.55      0.40      0.39      3599
weighted avg       0.58      0.61      0.55      3599
```

```
**************************************************************************
**************************************************************************
Accuracy_Score of  RandomForestClassifier() is 0.7129758266185051
Confusion_Matrix : [[ 857    0    2    7   78]
 [ 100   64    2    5   45]
 [ 109    1  103   15  107]
 [  70    0    8  194  366]
 [  71    1    5   41 1348]]
Classification_Report :               precision    recall  f1-score   support

           1       0.71      0.91      0.80       944
           2       0.97      0.30      0.45       216
           3       0.86      0.31      0.45       335
           4       0.74      0.30      0.43       638
           5       0.69      0.92      0.79      1466

    accuracy                           0.71      3599
   macro avg       0.79      0.55      0.59      3599
weighted avg       0.74      0.71      0.68      3599
```

```
************************************************************************
************************************************************************
Accuracy_Score of  SVC() is 0.6838010558488469
Confusion_Matrix : [[ 821    0   11   13   99]
 [ 113   36    8   11   48]
 [ 109    0   77   22  127]
 [  53    0   11  150  424]
 [  51    1    8   29 1377]]
Classification_Report :               precision    recall  f1-score   support

           1       0.72      0.87      0.79       944
           2       0.97      0.17      0.28       216
           3       0.67      0.23      0.34       335
           4       0.67      0.24      0.35       638
           5       0.66      0.94      0.78      1466

    accuracy                           0.68      3599
   macro avg       0.74      0.49      0.51      3599
weighted avg       0.70      0.68      0.63      3599
```

```
************************************************************************
************************************************************************
Accuracy_Score of  KNeighborsClassifier() is 0.5501528202278411
Confusion_Matrix : [[ 689   21   23   49  162]
 [ 114   25    7   18   52]
 [ 129    9   58   33  106]
 [ 125   10   36  133  334]
 [ 146   26   48  171 1075]]
Classification_Report :               precision    recall  f1-score   support

           1       0.57      0.73      0.64       944
           2       0.27      0.12      0.16       216
           3       0.34      0.17      0.23       335
           4       0.33      0.21      0.26       638
           5       0.62      0.73      0.67      1466

    accuracy                           0.55      3599
   macro avg       0.43      0.39      0.39      3599
weighted avg       0.51      0.55      0.52      3599
```

```
************************************************************************
************************************************************************
Accuracy_Score of  MultinomialNB() is 0.6323978883023061
Confusion_Matrix : [[ 756   42   51   47   48]
 [  79   62   28   21   26]
 [  64   24   99   72   76]
 [  61   12   46  203  316]
 [  84   17   33  176 1156]]
Classification_Report :               precision    recall  f1-score   support

           1       0.72      0.80      0.76       944
           2       0.39      0.29      0.33       216
           3       0.39      0.30      0.33       335
           4       0.39      0.32      0.35       638
           5       0.71      0.79      0.75      1466

    accuracy                           0.63      3599
   macro avg       0.52      0.50      0.51      3599
weighted avg       0.61      0.63      0.62      3599
```

Then with highest accuracy, hyperparameter tuning, and cross validation has been performed to obtain the best accuracy.

```
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
```

```
ran_class=RandomForestClassifier()
param={'max_features' : ['sqrt','auto', 'log2'],
       'n_estimators' : [100, 500, 1000],
       'min_samples_leaf' : [1, 5, 10, 15],
       'min_samples_split' : [5, 10, 15, 20],
       'random_state': list(range(0,50)),
       'bootstrap': [True, False],
       'criterion' : ['entropy','gini']
      }

cv=KFold(n_splits=10,shuffle=False)
```

```
RCV=RandomizedSearchCV(estimator=ran_class, param_distributions=param, cv=cv, verbose=2)
RCV.fit(x_train, y_train)
```

Result:

```
Confusion_Matrix :
 [[ 850    0    2    8   84]
 [ 101   64    1    6   44]
 [  98    1  108   24  104]
 [  69    1    4  189  375]
 [  72    2    2   43 1347]]
Accuracy_Score : 0.7107529869408169
Classification_Report :
               precision    recall  f1-score   support

           1       0.71      0.90      0.80       944
           2       0.94      0.30      0.45       216
           3       0.92      0.32      0.48       335
           4       0.70      0.30      0.42       638
           5       0.69      0.92      0.79      1466

    accuracy                           0.71      3599
   macro avg       0.79      0.55      0.59      3599
weighted avg       0.73      0.71      0.68      3599
```

CONCLUSION

As per the model performance, RandomForestClassifier is found to be the best algorithm with hyper parameter and cross validation the accuracy score comes up to 71%.