NAME OF THE PROJECT

“Ratings Prediction”

Submitted by:

Ashish Kumar Samal

ACKNOWLEDGMENT

It is a matter of great pleasure to express my profound feeling of reference to all the people who helped and supported me during the project. I would like to convey my sincere thanks to FLIPROBO TECHNOLOGIES AND DATATRAINED EDUCATION for constantly helping me with the valuable inputs during the project duration. Their inspiring guidance and everlasting enthusiasm have been valuable assets during the tenure of my project.

ASHISH KUMAR SAMAL

INTRODUCTION

- ## Business Problem Framing

  We have 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 have to build an application which can predict the rating by seeing the review.

- ## Conceptual Background of the Domain Problem

  Natural language processing (**NLP**) is a subfield of linguistics, computer science, and Artificial Intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyse large amounts of natural language data. The goal is a computer capable of "understanding" the contents of documents, including the contextual nuances of the language within them. The technology can then accurately extract information and insights contained in the documents as well as categorize and organize the documents themselves.

- ## Motivation for the Problem Undertaken

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

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  Describe the mathematical, statistical and analytics modelling done during this project along with the proper justification.

- Data Sources and their formats
  The reviews on different products were scrapped from different e-commerce websites.Mostly the data comprised of reviews given by a product user on the portal and the ratings they have given on basis of their satisfaction level with the product.
- Data Preprocessing Done
  At first we had to clean the ratings column and included only the number of ratings,removing other strings which is not useful.Then we need to perform cleaning of reviews text by help of NLTK.For that we conerted the strings to lower and removed the punctuations and all the stopwords and then lemmatization was done in which the words will be brought back to their root word.

- Data Inputs- Logic- Output Relationships
  After the data is cleansed.We need to change the word into their vectorial representation by using TFIDF vectorizer. It is the Term Frequency-Inverse Document Frequency model which is also a bag-of-words model. It is different from the regular corpus because it down weights the tokens i.e. words appearing frequently across documents.
- Hardware and Software Requirements and Tools Used
  Libraries like pandas,numpy,regex,string.Puntuations were used for data cleaning.
  And further nltk was used to remove stopwords and word_tokenize is used for tokenizing the strings.
  WordNetLemmatizer is used for lemmatization of the token words.
  TfidfVectorizer is used to convert the tokens/words into their vectorial representation.
  And Scikit Learn is used for model building and then model is hypertuned.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

-Data Cleansing

-Visualization and EDA

-Train Test Split

-Model Building and evaluation

-Cross Validation

-Hyperparameter tuning of best model

-Saving the best model

- Testing of Identified Approaches (Algorithms)
  Random Forest Classifier
  Logistic Regression
  Random Forest Classifier
  K Neighbors Classifier
  Naïve Bayes Classifier
- Run and Evaluate selected models

```
from sklearn.ensemble import RandomForestClassifier
model=RandomForestClassifier()
model.fit(features_train,labels_train)
model_predictions=model.predict(features_test)
print('Accuracy',accuracy_score(labels_test,model_predictions))
print(classification_report(labels_test,model_predictions))
```

```
Accuracy 0.7135996251610636
              precision    recall  f1-score   support

           1       0.68      0.81      0.74      2579
           2       0.69      0.47      0.56      1852
           3       0.70      0.51      0.59      2332
           4       0.69      0.53      0.60      3471
           5       0.74      0.91      0.81      6840

    accuracy                           0.71     17074
   macro avg       0.70      0.65      0.66     17074
weighted avg       0.71      0.71      0.70     17074
```

```
from sklearn.linear_model import LogisticRegression
model3=LogisticRegression()
model3.fit(features_train,labels_train)
model3_predictions=model3.predict(features_test)
print('Accuracy',accuracy_score(labels_test,model3_predictions))
print(classification_report(labels_test,model3_predictions))
```

```
Accuracy 0.5526026795519438
              precision    recall  f1-score   support

           1       0.55      0.72      0.62      2058
           2       0.35      0.18      0.24      1541
           3       0.36      0.25      0.29      1877
           4       0.42      0.25      0.31      2773
           5       0.64      0.86      0.73      5410

    accuracy                           0.55     13659
   macro avg       0.46      0.45      0.44     13659
weighted avg       0.51      0.55      0.52     13659
```

```
from sklearn.tree import DecisionTreeClassifier
model2=DecisionTreeClassifier()
model2.fit(features_train,labels_train)
model2_predictions=model2.predict(features_test)
print('Accuracy2',accuracy_score(labels_test,model2_predictions))
print(classification_report(labels_test,model2_predictions))
```

```
Accuracy2 0.678014495936745
              precision    recall  f1-score   support

           1       0.68      0.69      0.69      2058
           2       0.58      0.53      0.55      1541
           3       0.58      0.50      0.54      1877
           4       0.63      0.56      0.59      2773
           5       0.75      0.84      0.79      5410

    accuracy                           0.68     13659
   macro avg       0.64      0.62      0.63     13659
weighted avg       0.67      0.68      0.67     13659
```

```
from sklearn.neighbors import KNeighborsClassifier
model4=KNeighborsClassifier()
model4.fit(features_train,labels_train)
model4_predictions=model4.predict(features_test)
print('Accuracy4',accuracy_score(labels_test,model4_predictions))
print(classification_report(labels_test,model4_predictions))
```

```
Accuracy4 0.535031847133758
              precision    recall  f1-score   support

           1       0.51      0.56      0.53      2058
           2       0.36      0.33      0.34      1541
           3       0.44      0.30      0.35      1877
           4       0.48      0.33      0.39      2773
           5       0.61      0.77      0.68      5410

    accuracy                           0.54     13659
   macro avg       0.48      0.46      0.46     13659
weighted avg       0.52      0.54      0.52     13659
```

```
from sklearn.naive_bayes import MultinomialNB
model5=MultinomialNB()
model5.fit(features_train,labels_train)
model5_predictions=model5.predict(features_test)
print('Accuracy4',accuracy_score(labels_test,model5_predictions))
print(classification_report(labels_test,model5_predictions))
```

```
Accuracy4 0.5326890694779999
              precision    recall  f1-score   support

           1       0.54      0.69      0.61      2058
           2       0.37      0.07      0.12      1541
           3       0.37      0.19      0.25      1877
           4       0.40      0.18      0.24      2773
           5       0.57      0.91      0.70      5410

    accuracy                           0.53     13659
   macro avg       0.45      0.41      0.38     13659
weighted avg       0.48      0.53      0.47     13659
```

- Key Metrics for success in solving problem under consideration

```
parameter={'n_estimators':[100,200,350],'max_depth':range(5,35,5),
          'criterion':['gini','entropy'],
           'min_samples_split':[2,5,10,15,100]}
GCV2=GridSearchCV(model,parameter,cv=2,verbose=2,scoring='accuracy',n_jobs=-1)
GCV2.fit(features_train,labels_train)
GCV2.best_params_
```

```
Fitting 2 folds for each of 150 candidates, totalling 300 fits

{'criterion': 'entropy',
 'max_depth': 25,
 'min_samples_split': 2,
 'n_estimators': 200}
```

```
final_mod=RandomForestClassifier(criterion='entropy',max_depth=25,min_samples_split=2, n_estimators=200)
final_mod.fit(features_train,labels_train)
pred=final_mod.predict(features_test)
acc1=accuracy_score(labels_test,pred)
print(acc1)
print(classification_report(labels_test,pred))
```

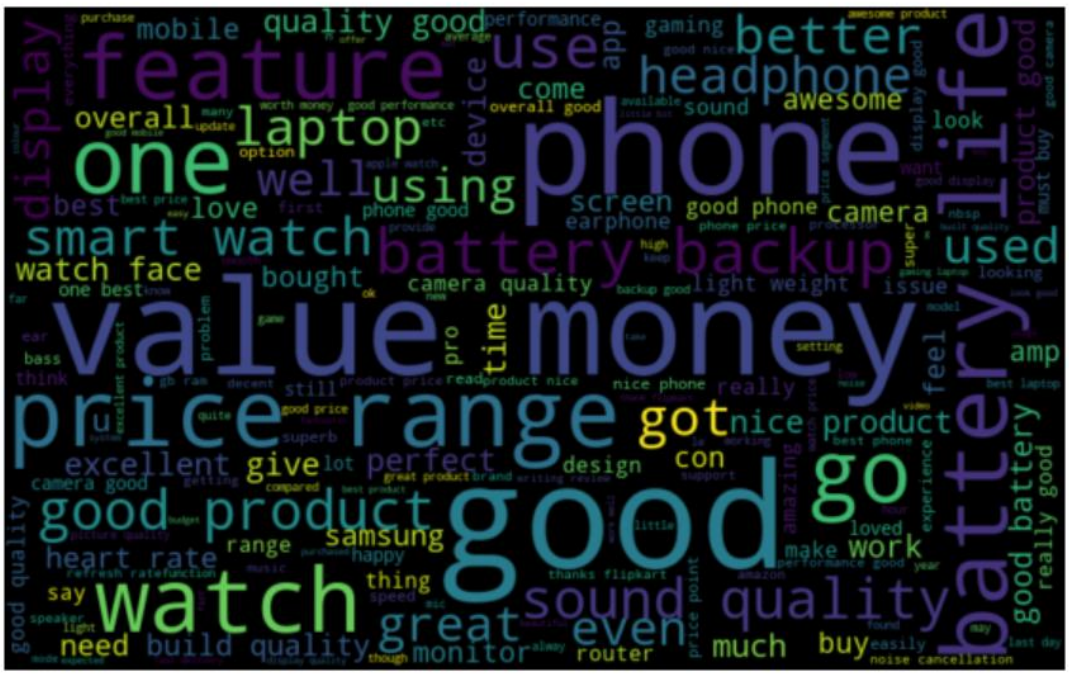
```
0.6252782007731053
              precision    recall  f1-score   support

           1       0.63      0.75      0.68      2579
           2       0.90      0.23      0.37      1852
           3       0.81      0.27      0.41      2332
           4       0.78      0.30      0.44      3471
           5       0.58      0.97      0.73      6840
```

- Visualizations

```
subset=df[df.Ratings=='1']
text=subset.Review_text123.values
words=' '.join(text)
create_wordcloud(words)
```


```
subset=df[df.Ratings=='5']
text=subset.Review_text123.values
words=' '.join(text)
create_wordcloud(words)
```



- Interpretation of the Results

  After trying different models we found out Random Forest Classifier is performing well and hence hypertuned the model to find the accuracy of model as 0.62.

CONCLUSION

- Key Findings and Conclusions of the Study
  The model is performing with an accuracy score of 0.62.

- Learning Outcomes of the Study in respect of Data Science
  Use of wordcloud and NLTK for NLP preprocessing.

- Limitations of this work and Scope for Future Work

  The only limitation of the work which can get better with the balancing of all labels which can be done by taking more datas of deficit labels into consideration.