Rating Prediction

Submitted by:

C S Manjunath Reddy

ACKNOWLEDGMENT

I would like to thank FlipRobo for giving me this opportunity. The DataTrained institute classes helped me to solve this problem.

The language used for web scrapping is Selenium and Pandas. For the project, the language used Python with Pandas, NumPy, Sklearn.

I referred to data trained provided classes of NLP to treat the text data and Fliprobo provided classes of web scrapping to scrape the reviews and ratings from different websites. The data was collected from Amazon and Flipkart websites for training the ML model for rating prediction.

INTRODUCTION

- Business Problem Framing

  The client has a website where their customers write reviews for the technical products. Now the client wants to add a rating (stars) feature. The ratings are 1 star,2 stars,3 stars,4 stars,5 stars. The client wants to predict the rating for the reviews written in past and they don't have ratings. So our task is to scrap reviews with ratings for the technical products from different websites and build an ML model to predict the past reviews from the client website.

- Conceptual Background of the Domain Problem

  The domain-related concepts are web scrapping, Data science, NLP, Machine Learning.

- Review of Literature

  I researched different technical products from amazon and Flipkart and collected the ratings and reviews as data sources to train the model. Once I scrapped the data I used it to train the model with the help of pandas and Sickit-learn.

- Motivation for the Problem Undertaken

  Client wants to predict the ratings for the reviews provided by their customers in past which don’t have ratings. So, I collected the data (reviews and ratings) provided by customers from different websites for technical products to train Ml model and to predict ratings for the reviews written on client website.

Analytical Problem Framing

- Data Sources and their formats

  The data needed is reviews and ratings, the data is collected from amazon and Flipkart websites, provided by customers for different technical products.

- Data Preprocessing Done

  I used NLP techniques for data preprocessing. Regex (Regular expressions) techniques are used to replace the numbers into text, remove the spaces, changing the text into lower case. Used the Stopwords technique to remove stopwords from the data. The actual length of the data before preprocessing is 2754264 and clean_data length is 1929013.

- Hardware and Software Requirements and Tools Used

  Selenium : used to web scrapping to get reviews with ratings from multiple websites for multiple technical products.

  Pandas : used to create data frames.

  NLP-Regular Expression : used to clean the text data.

  NLP-Tf-idf : To change the text data into numbers.

  Scikit-Learn : Used Ml algorithms to train the data and evaluate best model.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  Collected the data by using selenium webdriver.

  Created the data frame for the scraped data.

  Used the data to train the modem and to predict the rating for the reviews provided by customers.

  Saved the best model for future usage.

- Testing of Identified Approaches (Algorithms)

  Navie_bayes.

  KNN (K-Nearest Neighbour).

  SVC (Support Vector Classifier).

  Logistic Regression.

- Run and Evaluate selected models

  Importing libraries:

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import warnings
warnings.filterwarnings('ignore')
```

  Creating instantiate:

```
Mb=MultinomialNB()
svc=SVC()
knn=KNeighborsClassifier()
lr=LogisticRegression()
```

Logistic Regression:

```
lr.fit(x_train,y_train)
pred=lr.predict(x_test)
print(accuracy_score(y_test,pred))
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
0.537248028045574
[[1387  183  162  102   49]
 [ 437  425  479  144   41]
 [ 294  294  857  309  158]
 [ 146   57  181  971  565]
 [ 129   15   63  416 1264]]
              precision    recall  f1-score   support

         1.0       0.58      0.74      0.65      1883
         2.0       0.44      0.28      0.34      1526
         3.0       0.49      0.45      0.47      1912
         4.0       0.50      0.51      0.50      1920
         5.0       0.61      0.67      0.64      1887

    accuracy                           0.54      9128
   macro avg       0.52      0.53      0.52      9128
weighted avg       0.53      0.54      0.53      9128
```

#KNN

```
knn.fit(x_train,y_train)
pred=knn.predict(x_test)
print(accuracy_score(y_test,pred))
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
0.362182296231376
[[828 364 133 489  69]
 [404 432 196 446  48]
 [365 451 343 616 137]
 [239 241 160 848 432]
 [142 150 107 633 855]]
              precision    recall  f1-score   support

         1.0       0.42      0.44      0.43      1883
         2.0       0.26      0.28      0.27      1526
         3.0       0.37      0.18      0.24      1912
         4.0       0.28      0.44      0.34      1920
         5.0       0.55      0.45      0.50      1887

    accuracy                           0.36      9128
   macro avg       0.38      0.36      0.36      9128
weighted avg       0.38      0.36      0.36      9128
```

SVC

```
svc.fit(x_train,y_train)
pred=svc.predict(x_test)
print(accuracy_score(y_test,pred))
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
0.55488606485539
[[1419  151  169  100   44]
 [ 394  491  480  119   42]
 [ 304  236  897  308  167]
 [ 144   42  183  923  628]
 [ 135    9   58  350 1335]]
              precision    recall  f1-score   support

         1.0       0.59      0.75      0.66      1883
         2.0       0.53      0.32      0.40      1526
         3.0       0.50      0.47      0.48      1912
         4.0       0.51      0.48      0.50      1920
         5.0       0.60      0.71      0.65      1887

    accuracy                           0.55      9128
   macro avg       0.55      0.55      0.54      9128
weighted avg       0.55      0.55      0.54      9128
```

Naive_bayes:

```
Mb.fit(x_train,y_train)
pred=Mb.predict(x_test)
print(accuracy_score(y_test,pred))
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
0.5287028921998247
[[1388  101  220   68  106]
 [ 450  203  649  130   94]
 [ 305  114 1007  213  273]
 [ 158   29  216  692  825]
 [ 114    2   64  171 1536]]
              precision    recall  f1-score   support

         1.0       0.57      0.74      0.65      1883
         2.0       0.45      0.13      0.21      1526
         3.0       0.47      0.53      0.50      1912
         4.0       0.54      0.36      0.43      1920
         5.0       0.54      0.81      0.65      1887

    accuracy                           0.53      9128
   macro avg       0.52      0.51      0.49      9128
weighted avg       0.52      0.53      0.50      9128
```

- Key Metrics for success in solving problem under consideration

  Key Metrics are Confusion Matrix and Classification report .
  Confusion matrix shows True positive and false positive rate.
  Classification report shows F1-score(which shows how good the model trained).

CONCLUSION

- Key Findings and Conclusions of the Study

  Key Findings – Find how to clean the text data.

  Trained the data with multiple algorithms and checked SVC (support vector classifier) is the better algorithm with better f1 score and accuracy.

- Learning Outcomes of the Study in respect of Data Science

  I used Navie-byes, KNN, Logistic Regression, SVC algorithms to train the data. I used NLP techniques (Regex, bag of words) to clean and change the text data into numerical format. When I trained the data with the algorithms it shows SVC working better with better F1 score when compared with other algorithms.

- Limitations of this work and Scope for Future Work

  We can improve the result by scrapping still more data and training the model with better accuracy.