RATING PREDICTION

Submitted by:

Sinchana

Acknowledgement

I would like to thank all of the contributions made on the sites which has helped me to understand the real world problem statement. Towards completion of the project I have taken help of github contributors. Thanks to all of them and I have learnt on the topic NLP sessions given by the institute while doing implementation

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.

As a Data Scientist, we have to apply our analytical skills to give findings and conclusions in detailed data analysis written in Jupyter notebook.

Conceptual Background of the Domain Problem

Created model can be used to predict ratings of reviews, it might be a good tool for online shopping sites such as Flipkart.com, Amazon.in etc and manufacturer companies who might look into their product ratings and reviews so that they can make their investment according to demand of customers , which might help them to save time and earn more profits.

Review of Literature

Now a days people buy more products from online sites, So by using this model, rating prediction and positive or negative review prediction of a product is easy and less time consuming.

Motivation for the Problem Undertaken

Data Science help us to make predictions at areas like health sectors, auto industry, education, media etc. For our project we decided to implement Prediction of Rating of reviews. We have to create a model which will help online sites and manufacturer of a product if they have to modify their product or not according to customers requirement.

Analytical Problem Framing

Mathematical/ Analytical Modelling of the Problem

We would perform one type of supervised learning algorithms: classification. While it seems more reasonable to perform classification since we have 5 types of Rating i.e., 1, 2, 3, 4, 5.

The prediction will base on the dataset which is scrapped from Flipkart and Amazon technical product. Scrapping dataset contains reviews and ratings of different products.

Data Sources and their formats

We scraped more than 30000 rows of data. We can scrape more data as well. More the data better the model.

In this section we need to scrape the reviews of different laptops, Phones, Headphones, smart watches, Professional Cameras, Printers, Monitors, Home theatre, Router from different e-commerce websites.

Basically, we need these columns

1) reviews of the product.

2) rating of the product.

We fetched other data as well. Our columns are –

1. Rating
2. Review
3. Long Review

```
import selenium
import pandas as pd
from selenium import webdriver
from selenium.common.exceptions import NoSuchElementException
import time
```

```
#importing chromedriver
driver = webdriver.Chrome(r"C:\\chromedriver.exe")
```

```
Review = []
Rating = []
Long_Review = []
```

```
#2273/20 +1
for i in range(1,501):
    URL = "https://www.amazon.in/TP-Link-TL-WR845N-300Mbps-Wireless-N-Router/product-reviews/B01HGCLUH6/
    driver.get(URL)
    time.sleep(2)
    Rat = driver.find_elements_by_xpath("//div[@data-hook='review']/div/div/div[2]/a[1]")
    Rev = driver.find_elements_by_xpath("//div[@data-hook='review']/div/div/div[2]/a[2]/span")
    LRev = driver.find_elements_by_xpath("//div[@data-hook='review']/div/div/div[4]/span/span")
    for i in range(0,len(Rat)):
        Rate = Rat[i].get_attribute("title");
        Rating.append(Rate.replace('.0 out of 5 stars',"").strip())
        try:
```

Data Pre-processing Done

1. First we have to remove all other words and symbols except alphates (a-z and A-Z).
2. We have to make all the words lower.
3. We have to split words from different sentences.
4. Then we have made a list named stop list which contains all the repetition of words whose impact is negligible while predicting the output.
5. We have to remove all the punctuations that present inside the sentence.
6. After removing all these unnecessary noises, we have converted words to their base form by using WordnetLemmatizer.
7. Many unnecessary words are removed.

Hardware and Software Requirements and Tools Used

The system requirements are description of features and functionalities of the target system. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from client's point of view. System requirements are all of the requirements at the system level that describe the functions which the system as a whole should fulfil to satisfy the stakeholder needs and requirements, and is expressed in an appropriate combination of textual statements, views, and non-functional requirements; the latter expressing the levels of safety, security, reliability, etc., that will be necessary.

Hardware requirements: -

1. Processor — core i5 and above
2. RAM — 8 GB or above
3. SSD — 250GB or above

Software requirements: -

Anaconda

Libraries: -

from sklearn.model_selection import train_test_split,cross_val_score

Train_test_split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data. With this function, you don't need to divide the dataset manually. By default, Sklearn train_test_split will make random partitions for the two subsets.

The algorithm is trained and tested K times, each time a new set is used as testing set while remaining sets are used for training. Finally, the result of the K-Fold Cross-Validation is the average of the results obtained on each set.

from sklearn.neighbors import KNeighborsClassifier

K Nearest Neighbor(KNN) is a very simple, easy to understand, versatile and one of the topmost machine learning algorithms. KNN used in the variety of applications such as finance, healthcare, political science, handwriting detection, image recognition and video recognition

from sklearn.linear_model import LogisticRegression

The library sklearn can be used to perform logistic regression in a few lines as shown using the LogisticRegression class. It also supports multiple features. It requires the input values to be in a specific format hence they have been reshaped before training using the fit method.

from sklearn.tree import DecisionTreeClassifier

Decision Tree is a white box type of ML algorithm. It shares internal decision-making logic, which is not available in the black box type of algorithms such as Neural Network. Its training time is faster compared to the neural network algorithm. The time complexity of decision trees is a function of the number of records and number of attributes in the given data. The decision tree is a distribution-free or non-parametric method, which does not depend upon probability distribution assumptions. Decision trees can handle high dimensional data with good accuracy.

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

These are the approaches to solve this problem

1. Understand business problem
2. Scrapping data from flipkart.com using selenium
3. Data analysis
4. Data Cleaning
5. Data pre-processing
6. Model Building
7. Model Evaluation
8. Selecting the best model

Testing of Identified Approaches (Algorithms)

Listing down all the algorithms used for the training and testing.

- KNN = KNeighborsClassifier()
- LR = LogisticRegression()
- BNB = BernoulliNB()
- SVC = SVC()
- DT = DecisionTreeClassifier()
- RF = RandomForestClassifier()

I applied all these algorithms in the dataset.

Run and Evaluate selected models

```
********************************************************************************
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
accuracy score of -> LogisticRegression()
0.6165386676192177
[[2358  268  230   43   83]
 [ 589  775  222   72   74]
 [ 282  149  908   93  413]
 [ 126  103  182  776  811]
 [ 104   36  206  208 2087]]
              precision    recall  f1-score   support

           1       0.68      0.79      0.73      2982
           2       0.58      0.45      0.51      1732
           3       0.52      0.49      0.51      1845
           4       0.65      0.39      0.49      1998
           5       0.60      0.79      0.68      2641

    accuracy                           0.62     11198
   macro avg       0.61      0.58      0.58     11198
weighted avg       0.62      0.62      0.60     11198

[0.57749498 0.61795044 0.55204287 0.56784997 0.46275456]
0.555618560068458
Difference between Accuracy score and cross validatio score is -  0.06092010755075972
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
********************************************************************************
********************************************************************************
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
accuracy score of -> KNeighborsClassifier()
0.5320592963029113
[[2000  468  342   76   96]
 [ 507  739  294  112   80]
 [ 227  454  674  268  222]
 [ 117  196  242 1031  412]
 [ 109  172  182  664 1514]]
              precision    recall  f1-score   support

           1       0.68      0.67      0.67      2982
           2       0.36      0.43      0.39      1732
           3       0.39      0.37      0.38      1845
           4       0.48      0.52      0.50      1998
           5       0.65      0.57      0.61      2641

    accuracy                           0.53     11198
   macro avg       0.51      0.51      0.51     11198
weighted avg       0.54      0.53      0.53     11198

[0.49068989 0.56758205 0.48600134 0.49792364 0.3892015 ]
0.4862796838982059
Difference between Accuracy score and cross validatio score is -  0.04577961240470535
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
********************************************************************************
```

```
********************************************************************************
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
accuracy score of -> BernoulliNB()
0.5843007679942847
[[2380  160  213   38  191]
 [ 627  638  227   89  151]
 [ 292  120  609   86  738]
 [ 118  104   80  584 1112]
 [ 107   27   52  123 2332]]
              precision    recall  f1-score   support

           1       0.68      0.80      0.73      2982
           2       0.61      0.37      0.46      1732
           3       0.52      0.33      0.40      1845
           4       0.63      0.29      0.40      1998
           5       0.52      0.88      0.65      2641

    accuracy                           0.58     11198
   macro avg       0.59      0.53      0.53     11198
weighted avg       0.59      0.58      0.56     11198

[0.52525117 0.60843938 0.52726055 0.56101808 0.46449625]
0.537293087642295
Difference between Accuracy score and cross validatio score is -  0.047007680351989656
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
********************************************************************************
********************************************************************************
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
accuracy score of -> SVC()
0.6326129666011788
[[2491  243  153   29   66]
 [ 660  777  175   69   51]
 [ 330  158  877   89  391]
 [ 128   99  137  832  802]
 [ 128   23  163  220 2107]]
              precision    recall  f1-score   support

           1       0.67      0.84      0.74      2982
           2       0.60      0.45      0.51      1732
           3       0.58      0.48      0.52      1845
           4       0.67      0.42      0.51      1998
           5       0.62      0.80      0.70      2641

    accuracy                           0.63     11198
   macro avg       0.63      0.59      0.60     11198
weighted avg       0.63      0.63      0.62     11198

[0.5699933  0.62880107 0.58312123 0.58044206 0.47119507]
0.5667105477580621
Difference between Accuracy score and cross validatio score is -  0.06590241884311676
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
********************************************************************************
```

```
********************************************************************************
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
accuracy score of -> DecisionTreeClassifier()
0.6129666011787819
[[2279  303  265   53   82]
 [ 579  728  254  108   63]
 [ 220  176  948  114  387]
 [  74   94  183  859  788]
 [  57   39  213  282 2050]]
              precision    recall  f1-score   support

           1       0.71      0.76      0.74      2982
           2       0.54      0.42      0.47      1732
           3       0.51      0.51      0.51      1845
           4       0.61      0.43      0.50      1998
           5       0.61      0.78      0.68      2641

    accuracy                           0.61     11198
   macro avg       0.60      0.58      0.58     11198
weighted avg       0.61      0.61      0.60     11198

[0.53824514 0.62505023 0.58499665 0.56851976 0.44105038]
0.5515724326959179
Difference between Accuracy score and cross validatio score is -  0.06139416848286394
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
********************************************************************************
********************************************************************************
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
accuracy score of -> RandomForestClassifier()
0.6268083586354706
[[2357  259  246   44   76]
 [ 608  741  229   87   67]
 [ 231  175  946  102  391]
 [  75   87  172  881  783]
 [  61   27  201  258 2094]]
              precision    recall  f1-score   support

           1       0.71      0.79      0.75      2982
           2       0.57      0.43      0.49      1732
           3       0.53      0.51      0.52      1845
           4       0.64      0.44      0.52      1998
           5       0.61      0.79      0.69      2641

    accuracy                           0.63     11198
   macro avg       0.61      0.59      0.59     11198
weighted avg       0.62      0.63      0.62     11198

[0.56584059 0.63536504 0.58995311 0.57990623 0.45632369]
0.5654777313780852
Difference between Accuracy score and cross validatio score is -  0.06133062725738536
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
********************************************************************************
```

```
********************************************************************************
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
accuracy score of -> KNeighborsClassifier(n_neighbors=3)
0.5286658331844972
[[2180  199  501   34   68]
 [ 643  588  398   59   44]
 [ 299  190 1226   60   70]
 [ 130  130  868  540  330]
 [ 133   51  929  142 1386]]
              precision    recall  f1-score   support

           1       0.64      0.73      0.68      2982
           2       0.51      0.34      0.41      1732
           3       0.31      0.66      0.43      1845
           4       0.65      0.27      0.38      1998
           5       0.73      0.52      0.61      2641

    accuracy                           0.53     11198
   macro avg       0.57      0.51      0.50     11198
weighted avg       0.59      0.53      0.53     11198

[0.53369056 0.50408573 0.4158071  0.39410583 0.36254019]
0.44204588185379573
Difference between Accuracy score and cross validatio score is -  0.0866199513307015
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
********************************************************************************
```

Hyperparameter Tuning ¶

```python
from sklearn.model_selection import GridSearchCV

# defining parameter range
param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf']}

grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3)

# fitting the model for grid search
grid.fit(x_train, y_train)
```

```
# print best parameter after tuning
print(grid.best_params_)

# print how our model looks after hyper-parameter tuning
print(grid.best_estimator_)
```

```
{'C': 1, 'gamma': 1, 'kernel': 'rbf'}
SVC(C=1, gamma=1)
```

```
grid_predictions = grid.predict(x_test)
# print classification report
print(classification_report(y_test, grid_predictions))
```

```
              precision    recall  f1-score   support

           1       0.67      0.84      0.74      2982
           2       0.60      0.45      0.51      1732
           3       0.58      0.47      0.52      1845
           4       0.68      0.41      0.51      1998
           5       0.61      0.80      0.69      2641

    accuracy                           0.63     11198
   macro avg       0.63      0.59      0.60     11198
weighted avg       0.63      0.63      0.62     11198
```

Saving the Model

```
import joblib
joblib.dump(grid.best_estimator_,"Rating.obj")
SVR_from_joblib=joblib.load("Rating.obj")
Predicted = SVR_from_joblib.predict(x_test)
```

Key Metrics for success in solving problem under consideration

Precision: can be seen as a measure of quality, **higher precision** means that an algorithm returns more relevant results than irrelevant ones.

Recall is used as a measure of quantity and high recall means that an algorithm returns most of the relevant results.

Accuracy score is used when the True Positives and True negatives are more important. **Accuracy** can be used when the class distribution is similar.

F1-score is used when the False Negatives and False Positives are crucial. While F1-score is a better metric when there are imbalanced classes.

Cross_val_score :- To run **cross-validation** on multiple metrics and also to return train **scores**, fit times and **score** times. Get predictions from each split of **cross-validation** for diagnostic purposes. Make a scorer from a performance metric or loss function.

AUC_ROC _score :- ROC curve. It is a plot of the false positive rate (x-axis) versus the true positive rate (y-axis) for a number of different candidate threshold values between 0.0 and 1.0

Visualizations

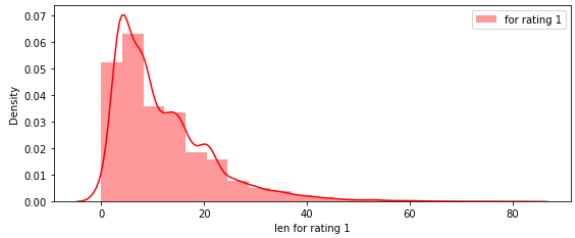
One Rating –

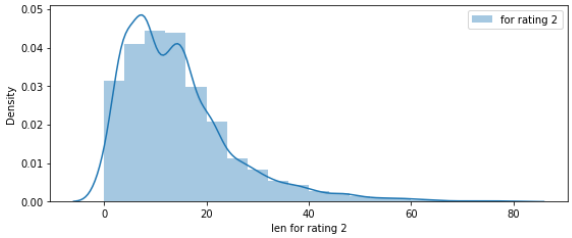
Two Rating –

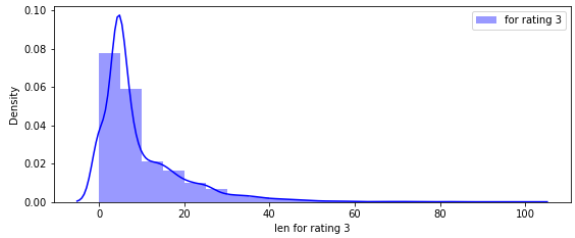
Three Rating –

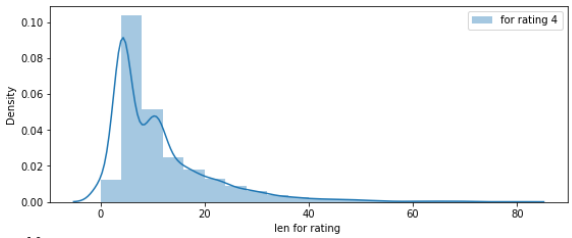
Four Rating –

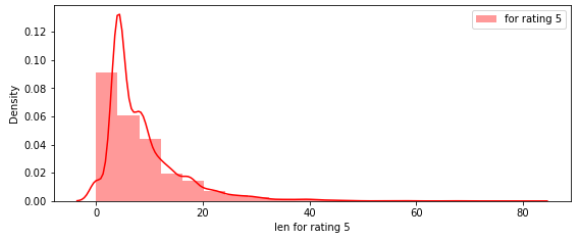
Five Rating –

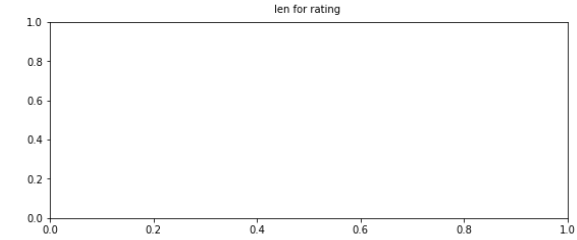












Interpretation of the Results

Comparing recall, precision, f1score of all models, we choose SVC is our best model. It gives accuracy score of approximately 63.261%.

Conclusion

Key Findings and Conclusions of the Study

First, we scrap Reviews and Ratings data of different technical products from flipkart.com using selenium web driver. After scrapping, we save this dataset in a csv file named "flipkartreviews.csv".

Then we pre-process data by cleaning duplicates, noise and unnecessary words which are not helpful for this project.

Then we convert text into vectors by using tfidfvectorizer as we know our ml model understands only integers.

We choose our best random_state. Then we build our model.

Though our dataset is imbalanced, we clearly see that f1 score of SVC is good compared to all other algorithms.

So, we check croos_val_score of this model to check if it is overfitting or not.

We confirm and conclude that this is the best model. We performed hyperparameter tuning by using GridSearchCV.

We save the model by using joblib.

Learning Outcomes of the Study in respect of Data Science

The power of visualization is helpful for the understanding of data into the graphical representation its help me to understand that what data is trying to say, Data cleaning is one of the most important steps to remove missing value or null value.
Various algorithms I used in this dataset and to get out best result and save that model. The best algorithm is SVC.

Limitations of this work and Scope for Future Work

As we know data is increasing in every second in our day today life. So more the data better the model. If we make this dataset for sentiment analysis, we choose ratings 3 or more as our threshold for being helpful reviews or good or positive reviews and below 3 we choose reviews is not helpful or bad reviews or negative reviews. For example: two people give their reviews on a product as 'a nice product. But their ratings are '4' and '5' respectively. This is where the model fails to predict whether to choose rating '4' or'5' Due to increase in data in our daily basics, this model can be used to predict ratings of reviews. It might be a good tool for online shopping sites and manufacturer companies who may predict their customers ratings so that they can make their investment according to the demand of customers, which might help them to save time and earn more profits.