Rating Prediction

Submitted by:

R. Rajesh kannan

ACKNOWLEDGEMENT

Reference: pandas.pydata.org, seaborn.pydata.org, matplotlib.org

Resource: stackoverflow.com,geeksforgeeks.org

Data Sources are from different e-commerce website i.e(amazon,flipkart)

Other Resources are Project Use case.

Introduction

Business Problem Framing:

Data Sources are webscrap from Indian e-commerce like amazon and flipkart .

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

Problem: 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:

It is similar to sentiment analysis because the input data are in text format with help of NLP and train the required model to predict rating of reviews.

Motivation for the Problem Undertaken:

Aim of this project is to predict rating for the review data which get from e-commerce. To get rating for respective reviews, we have to preprocessing the review text by NLP using NLTK and train the required model we can predict the rating. These reviews are based on customer satisfaction on product.

Analytical Problem Framing

Mathematical/ Analytical Modeling of the Problem:

Here all the data are categorical data and some ordinal data, so there are no require of mathematical and statistical modeling.

For analytical purpose I used visualization to analysis the data. Here I used matplotlib.pyplot, seaborn , wordcloud and simple plot function in pandas.

Data Sources and their formats:

Data Sources: From Indian E-commerce

Formats:

```
In [8]: print(df1.columns)
        print(df2.columns)
        print(df1.dtypes)
        print(df2.dtypes)
```

```
Index(['Unnamed: 0', 'Catogery', 'Summery', 'Review', 'Rating'], dtype='object')
Index(['Unnamed: 0', 'Catogery', 'Summery', 'Review', 'Rating'], dtype='object')
Unnamed: 0     int64
Catogery      object
Summery       object
Review        object
Rating         int32
dtype: object
Unnamed: 0     int64
Catogery      object
Summery       object
Review        object
Rating         int64
dtype: object
```

Every data type is object data type except the Rating.

Necessary:

Basically, we need these columns-
1) reviews of the product.
2) rating of the product

Here all the categorical, integer and ordinal data are necessary, we can treat that necessary variable as our requirement. If the data have more than 60% is null then we have to drop that, if there replacement value for null value in description then treat by replacement, but here there is no presence of null value.

Data Preprocessing:

There is no presence of lot of null values in the given data set.

```
In [14]: df.isnull().sum()

Out[14]: Catogery     0
         Summery      1
         Review      14
         Rating       0
         dtype: int64
```

While scraping the date in one go without apply exception there may some missing data.

Categorical data will not contain outliers and also there will be no presence of skewed in data here all values are in text format with respective integer data.

Data Inputs and Output Logic Relationships:

There is presence of output features in scraped dataset.

we can predict it by certain models. Output data will be predict by classification model i.e here we have to predict Rating .

Visualization techniques like uni-variant visualization can also perform to see relationship of input features and also to visualize its % of occupation in total data.

To find:

Apply preprocessing skills to get the text data clean by remove the junks like punctuation, stopwords, numbers etc. and the train and predict the model.

Hardware and Software Requirements and Tools Used:

Hardware: i5 processor, 8 GB RAM.

Software: OS(windows),

Tools: Jupiter Notebook or Py charm

Libraries: numpy, pandas, matplotlib, seaborn, sklearn NLTK

Packages: Pyplot, metrics, model_selection, and respective model packages.

Analysis by Visualizations:

Here I divide data into three different category (category, Rating, Review).

Pre-Processing Text Data

Apply preprocessing skills to get the text data clean by remove the junks like punctuation, stopwords, numbers etc. and the train and predict the model.

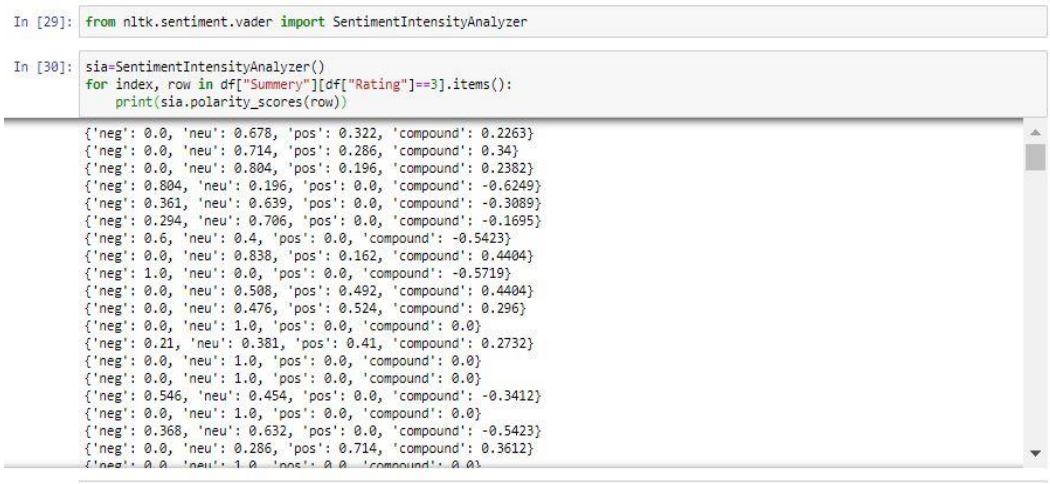
With the help of NLTK we can do this process to remove junk values

```
In [24]: df["Review"]=df["Review"].str.lower()
         #make tokenize
         df["Review"]=df.apply(lambda x:word_tokenize(x["Review"]),axis=1)
         #to remove numeric terms
         df["Review"]=df["Review"].apply(lambda x:[word for word in x if word.isalpha()])
         #to lemmatize each word
         df["Review"]=df["Review"].apply(lambda x:[WordNetLemmatizer().lemmatize(word) for word in x ])
         #to remove stopwords
         df["Review"]=df["Review"].apply(lambda x:[word for word in x if word not in stopwords.words("english")])
         df["Review"]=df["Review"].apply(lambda x: str(" ".join(x)))

In [25]: len(df["Review"])

Out[25]: 20125

In [26]: df["clear_length"]=df["Review"].str.len()

In [27]: df["length"].sum()

Out[27]: 4913782

In [28]: df["clear_length"].sum()

Out[28]: 3072869
```

Sentiment Analyzer

Sentiment Analyzer is done by apply Sentiment Intensity Analyzer on pre processed text data.

Polarity score which shows that negative, neutral and positive score based on that score the data will consider it is positive or negative.



```
In [29]: from nltk.sentiment.vader import SentimentIntensityAnalyzer

In [30]: sia=SentimentIntensityAnalyzer()
         for index, row in df["Summery"][df["Rating"]==3].items():
             print(sia.polarity_scores(row))

{'neg': 0.0, 'neu': 0.678, 'pos': 0.322, 'compound': 0.2263}
{'neg': 0.0, 'neu': 0.714, 'pos': 0.286, 'compound': 0.34}
{'neg': 0.0, 'neu': 0.804, 'pos': 0.196, 'compound': 0.2382}
{'neg': 0.804, 'neu': 0.196, 'pos': 0.0, 'compound': -0.6249}
{'neg': 0.361, 'neu': 0.639, 'pos': 0.0, 'compound': -0.3089}
{'neg': 0.294, 'neu': 0.706, 'pos': 0.0, 'compound': -0.1695}
{'neg': 0.6, 'neu': 0.4, 'pos': 0.0, 'compound': -0.5423}
{'neg': 0.0, 'neu': 0.838, 'pos': 0.162, 'compound': 0.4404}
{'neg': 1.0, 'neu': 0.0, 'pos': 0.0, 'compound': -0.5719}
{'neg': 0.0, 'neu': 0.508, 'pos': 0.492, 'compound': 0.4404}
{'neg': 0.0, 'neu': 0.476, 'pos': 0.524, 'compound': 0.296}
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
{'neg': 0.21, 'neu': 0.381, 'pos': 0.41, 'compound': 0.2732}
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
{'neg': 0.546, 'neu': 0.454, 'pos': 0.0, 'compound': -0.3412}
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
{'neg': 0.368, 'neu': 0.632, 'pos': 0.0, 'compound': -0.5423}
{'neg': 0.0, 'neu': 0.286, 'pos': 0.714, 'compound': 0.3612}
```

Text to vector:

We can not able to input data as text format so we have to convert it to vector for text data for make it as input.

```
In [31]: from sklearn.feature_extraction.text import TfidfVectorizer
```

```
In [32]: vect=TfidfVectorizer(ngram_range=(1,3))
         text=vect.fit_transform(df["Review"])
```

Models Development and Evaluation

Identification of Possible Problem solving approaches:

Model Selection: In Superwised there are classification and regression models. But here output variable have only 5 unique values, so it will come under classification model.

Testing of identification Approaches:

To find the best random state we have to run the for loop by iterating some range of numbers in a model Train Test Split. Random state with high accuracy will be considers as best.

After the selection the best random state we have to find best model by getting good accuracy score, through train and test the data.

Run and Evaluate selected models:

Import the required model through respective packages in sklearn library,

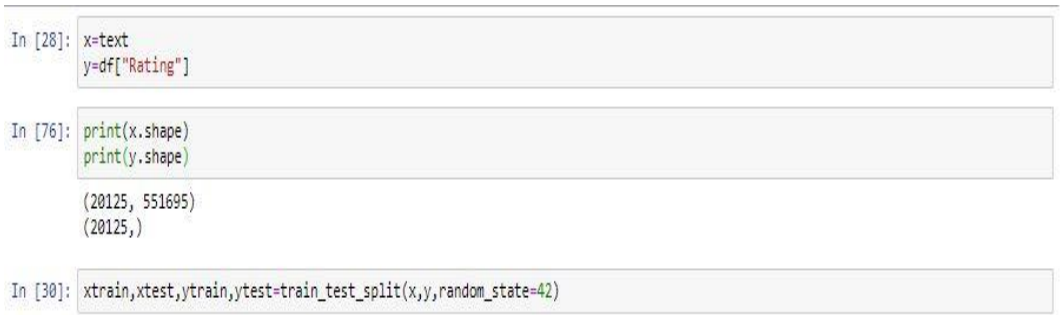
Due to presence of text data as input we have to covert text to vector then to numeric as for model understanding

```
from sklearn.model_selection import train_test_split,cross_val_score,cross_val_predict,GridSearchCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

Have to split the data as input train and test, output train and test.

Here train input is Review, train output are Rating.

After the selection the best random state we have to find best model by getting good accuracy score, through train and test the data.



Firstly have to fit the model,

Secondly have to train the input and output data,

Then have to predict certain test input data,

Finally have to check the accuracy score by compare output test data with predicted output data.

Make the required models in the list.

Not only by this process can find the best model because there may occur over fitting due to presence of high bias and high variance in it. So we have to do Cross Validation on it for right result.

```
In [35]: model=[nb,rf,dt,kn]
         for i in model:
             print(i,cross_val_score(i,x,y,cv=4).mean())

MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True) 0.5048010098822922
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False) 0.5156398824561323
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best') 0.47971975463362837
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform') 0.39694698956773183
```

So these models in the list have to iterate through cross validation.

From above MultinomialNB is consider as a good model when compare with other.

Hyper Parameter Tuning: For to improve the accuracy score for selected best model or all model, we have to apply Grid Search or Randomized search, by apply all the required parameters of respective model inside Grid Search we can get better accuracy.

```
In [66]: para={'alpha': np.linspace(0.5, 1.5, 6),'fit_prior': [True, False],}
         gv=GridSearchCV(MultinomialNB(),para,cv=4)
         gv.fit(xtrain,ytrain)

Out[66]: GridSearchCV(cv=4, error_score='raise-deprecating',
                      estimator=MultinomialNB(alpha=1.0, class_prior=None,
                                              fit_prior=True),
                      iid='warn', n_jobs=None,
                      param_grid={'alpha': array([0.5, 0.7, 0.9, 1.1, 1.3, 1.5]),
                                  'fit_prior': [True, False]},
                      pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
                      scoring=None, verbose=0)

In [67]: gv.best_score_

Out[67]: 0.634731332405751

In [68]: gv.best_estimator_

Out[68]: MultinomialNB(alpha=0.5, class_prior=None, fit_prior=False)
```

- MultinomialNB gives a better result at 63%.
- And also came to know the best parameters .

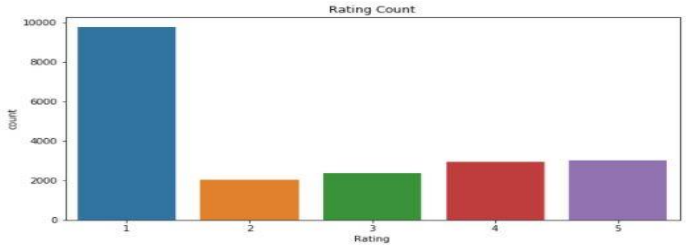
Predict the review data for test input by fit train data in MultinomialNB model.

Visualization on user details:

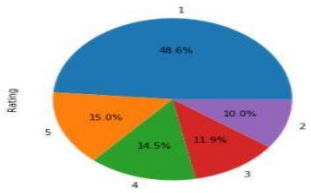
Uni variant visualization:

Count of Rating:

```
In [19]: plt.figure(figsize=(10,5))
         sns.countplot(df["Rating"])
         plt.title("Rating Count")
         plt.show()
```




```
In [61]: plt.figure(figsize=(10,5))
         df["Rating"].value_counts().plot.pie(autopct="%.1f%%")
         plt.show()
```

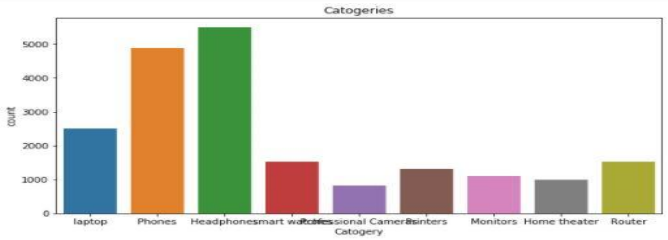


It shows number of count the rating in data set.

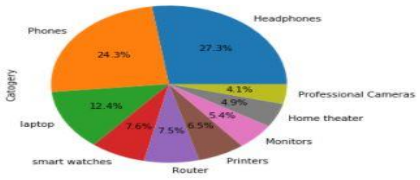
The pie graph shows the percentage for rating count.

Count of Product:

```
In [87]: plt.figure(figsize=(10,5))
         sns.countplot(df["Catogery"])
         plt.title("Catogeries")
         plt.show()
```




```
In [86]: plt.figure(figsize=(10,5))
         df["Catogery"].value_counts().plot.pie(autopct="%.1f%%")
         plt.show()
```



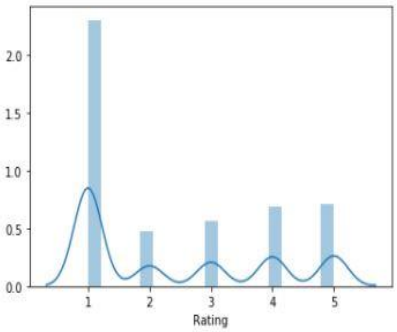
It shows number of count the the given category of product occur in data set.

The pie graph shows the percentage for different product count.

Distribution of Rating:

It shows good distribution but little imbalance in data.

```
In [18]: sns.distplot(df["Rating"])
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x246ec0d75c8>
```



CONCLUSION

Key finding: Analysis in review after clean the data and predict Rating by train the model.

Inferences: From the report it concluded that there are no wrong data. By clean it and prediction was lead to get good model.

Observations:

1. For rating features mostly customer select agree.
2. For customer details features(city-Delhi, browser-Google chrome, device-smartphone, net-Mobile Internet) plays major role in online shopping.
3. For website rating features Amazon.in and flipkart.com are liked by most of the customers.

Learning Outcomes of the study in respect of Data Science

- I learned by visualize also can get important text and also find how to clean text information.
- Learned to analyse and predict in text features.

Limitations and Future work:

Limitations: Not able to scrape the data in balanced way for example if you are fetching 10000 reviews then all ratings 1,2,3,4,5 should be 2000.

Steps to follow further: Here I clean all the data by NLTK technique but I did not apply any sentiment analysis on review to get better prediction.