Ratings Prediction Project

Submitted by:

CHILUVERI SAIPRAKASH

ACKNOWLEDGMENT

I wish to express my sincere gratitude to **DataTrained** Academy and **FlipRobo Technologies** who gave me the opportunity to do the **Ratings Prediction Project**. It helped me to do a lot of research and I have grasped many new things.

I have put good effort in this project. However it would not have been possible without the kind support and help of many individuals. I would like to extend my sincere thanks to all of them. I would also thankful to the blogs/articles through online platforms which gave me a lot of information in finishing this project within the limited time.

INTRODUCTION

Business Problem Framing:

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:

Our objective is to create a Classification model that predicts the rating for a specific review.

Motivation for the Problem Undertaken:

Every problem begins with ideas that are further developed and inspired to address a variety of situations and circumstances. Learning the theoretical background for data science or machine learning are often a frightening experience, because it involves multiple fields of mathematics and an extended list of online resources. By proper practical research and practice I can become better in this field. These suggestions are derived from my mentors/SME's and my own experience in the beginner projects.

Analytical Problem Framing

Mathematical/ Analytical Modelling of the Problem:

- Initially we scrape the data from different websites of the different products and store them in csv files
- We check the basic information of the dataset that is its shape and its information using pandas library.
- Then we check the unique information of our data columns.
- After that we check for null values(-) and replace – values with No rating in the dataset.
- We also create new columns to check the length of data before and after cleaning the Reviews feature to check the distribution of our data.

- Before making the model we convert our text into vectors so for that we use technique known as TF-IDF Vectorizer.
- Then, we pass the vectors into the model and make the predictions.

Data Sources and their formats:

In this project the sample data is obtained by scraping different E-commerce websites for different products. The scraped data is stored in a csv (comma separated values) format. The dataset contains 87402 rows and 2 columns. We import the libraries and load the dataset as shown in below figure.

```
1 import pandas as pd
2 import warnings
3 warnings.filterwarnings('ignore')
```

```
1 df = pd.read_csv("Final.csv")
2 df
```

	Rating	Reviews
0	5.0	Very Good sound clarity with bass, build quali...
1	4.0	Built quality is very good. Bass is a bit too ...
2	5.0	They are just awesome . no less then JBL. voca...
3	5.0	nice product with beautiful design and fast de...
4	4.0	Got it by the next day of order! Excellent ser...
...	...	...
87397	5.0	Five Stars
87398	5.0	Value for Money
87399	3.0	Poor
87400	5.0	It does not go above 100Mbps
87401	2.0	I am no satisfied

87402 rows × 2 columns

Data Description:

There are 2 columns in the dataset. The description of each of the column is given below:

Rating: Rating of a particular product given by a customer.

Reviews: The Review of a product given by a customer.

Data Preprocessing Done:

By checking the information of the dataset, we get a clear idea of the dataset which includes null values, data types, count etc

Data Cleaning:

- A new column is framed named "length" which specifies the length of Reviews column.
- Removing HTML tags
- Removing special characters
- Converting Reviews column to lowercase
- Removing stopwords
- Using Snowball Stemmer for Stemming
- We create new column (length of cleaned data) after removing puncuations, stopwords from Reviews column to check how much data is cleaned.
- Used a library preprocess_kgptalkie and cleaned.

```
import preprocess_kgptalkie as ps
import re
```

```
def clean(x):
    x = str(x).lower().replace('\\', '').replace('_', ' ')
    x = ps.cont_exp(x)
    x = ps.remove_emails(x)
    x = ps.remove_urls(x)
    x = ps.remove_html_tags(x)
    x = ps.remove_accented_chars(x)
    x = ps.remove_special_chars(x)
    x = re.sub("(.)\\1{2,}", "\\1", x)
    return x
```

```
df['Reviews'] = df['Reviews'].apply(lambda x: clean(x))
df.head()
```

	Rating	Reviews	length
0	5.0	very good sound clarity with bass build qualit...	142.0
1	4.0	built quality is very good bass is a bit too h...	222.0
2	5.0	they are just awesome no less then jbl vocals ...	171.0
3	5.0	nice product with beautiful design and fast de...	86.0
4	4.0	got it by the next day of order excellent serv...	468.0

State the set of assumptions (if any) related to the problem under consideration:

The column Ratings has '-' values. So, we replaced them with "No Rating" and considered only rows with values such as 1, 2,3,4,5.

Hardware and Software Requirements and Tools Used:

Processor Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz
Installed RAM 4.00 GB
System type 64-bit operating system, x64-based processor
Edition Windows 10 Pro
Software: The complete project is done using Jupyter Notebook.

Initially we load the basic libraries such as pandas, seaborn and matplotlib
NumPy: NumPy is the library used for scientific computing in python. It is also termed as numerical python.
Pandas: Pandas is mainly used for data analysis. It allows importing data from various file formats.

sklearn.**preprocessing**: The sklearn.preprocessing package provides several common utility functions and transformer classes to change raw feature vectors into a representation that is more suitable for the downstream estimators. In general, learning algorithms benefit from standardization of the dataset.

NLTK: NLTK or Natural Language Toolkit is a Python package that you can use for NLP. A lot of the data that you could be analyzing is unstructured data and contains human-readable text.

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

The whole problem-solving approach includes the following steps:

- **Problem Framing:** It includes understanding the problem that is whether the problem is of regression or classification. The present project is of Classification type.

⇨ **Data Understanding:** Data understanding means having an intimate grasp of both the distributions of variables and the relationships between variables. It also includes data visualization.

⇨ **Data Cleaning:** The process of identifying and repairing issues with the data is termed as data cleaning. Statistical methods are used for data cleaning.

⇨ **Data Selection:** The process of reducing the scope of data to those elements that are most useful for making predictions is called data selection.

⇨ **Data Preparation:** It includes the data to identify the features to be selected and removed. Before passing the data into the model, we should convert all the categorical data into numerical. So, we use TF-IDF Vectorizer and convert them to vectors.

⇨ **Model Evaluation:** Model evaluation consists of identifying input and output variables and splitting the dataset into train and test datasets. The output variable is “label” and Reviews is input variable. In this project, we have split the data into 80:20 train and test respectively.

⇨ **Model Configuration:** Hyper parameter tuning the models will get the best fit parameters of each and every model. In this project we use GridSearchCV for knowing the best fit parameters. This can be seen detail in the further sections with a snapshot of it.

⇨ **Model Selection:** The process of selecting one method as the solution is called model selection. It includes the classification model performance metrics.

```
from nltk.corpus import stopwords  #For stopwords
from nltk.tokenize import word_tokenize

def remove_stopwords(text):
    stop_words = set(stopwords.words('english') + ['u', 'ü', 'ur', '4', '2', 'im', 'dont', 'doin', 'ure'])
    words = word_tokenize(text)
    return [w for w in words if w not in stop_words]

df.Reviews = df.Reviews.apply(remove_stopwords)
df.Reviews.head()
```

```
0    [good, sound, clarity, bass, build, quality, g...
1    [built, quality, good, bass, bit, high, listen...
2    [awesome, less, jbl, vocals, balanceclear, sou...
3    [nice, product, beautiful, design, fast, deliv...
4    [got, next, day, order, excellent, service, fl...
Name: Reviews, dtype: object
```

```
# Stemming using Snowball
from nltk.stem import SnowballStemmer

def stem_text(text):
    snowball = SnowballStemmer('english')
    return " ".join([snowball.stem(w) for w in text])

df.Reviews = df.Reviews.apply(stem_text)
df.Reviews.head()
```

```
good sound clariti bass build qualiti good wir...
built qualiti good bass bit high listen longer...
awesom less jbl vocal balanceclear sound even ...
nice product beauti design fast deliveri thank...
got next day order excel servic flipkart earph...
```

Since, Machine learning model takes only numerical we convert the Reviews column using Tfidfvectorizer to convert them to vectors as shown in below figure.

```
# Importing the library and converting it into vectors
from sklearn.feature_extraction.text import TfidfVectorizer
# Using TfidfVectorizer to deal the frequent words
tf_vec = TfidfVectorizer(max_features=25000, ngram_range=(1,5), analyzer='char')

#Seperating into input and output variables
x = tf_vec.fit_transform(df['Reviews'])
y = df['Rating']
```

Testing of Identified Approaches (Algorithms):

The Machine Learning Algorithms used in this project for training and testing to predict the loan defaulters are namely:

- LinearSVC
- MultinomialNB Classifier
- Passive Aggressive Classifier
- Random Forest Classifier

Run and Evaluate selected models:

We have split the train dataset into train and test in 80:20 ratios and imported the machine learning algorithms and their performance metrics from sklearn library.

```
#Breaking our input and target variable into train and test data
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

```
# Importing all the model library
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.ensemble import RandomForestClassifier
# Importing performance metrics
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

Passed the best fit parameters to the specified models.

```
# Passing the best fit parameters
svc = LinearSVC(C=20, class_weight='balanced')
mnb = MultinomialNB(alpha = 1.0,fit_prior = True)
pac = PassiveAggressiveClassifier(C = 0.8, n_iter_no_change = 4)
rfc = RandomForestClassifier(n_estimators = 100,min_samples_split = 2)
```

```
# Evaluating the models using to find their performance metrics
def evaluate_metrics(model):
    model.fit(x_train,y_train)
    model.score(x_train,y_train)
    pred=model.predict(x_test)
    print('Accuracy score of',model,'is:')
    print(accuracy_score(y_test,pred))
    print(confusion_matrix(y_test,pred))
    print(classification_report(y_test,pred))
```

```
1 evaluate_metrics(svc)
```

```
Accuracy score of LinearSVC(C=20, class_weight='balanced', dual=True, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0) is:
0.9877009324409358
[[    2     3    10     1     0]
 [    1     7     6     0     1]
 [    7    12   142    36     6]
 [    1    18    54  5544     5]
 [    0    19    29     6 11571]]
              precision    recall  f1-score   support

         1.0       0.18      0.12      0.15        16
         2.0       0.12      0.47      0.19        15
         3.0       0.59      0.70      0.64       203
         4.0       0.99      0.99      0.99      5622
         5.0       1.00      1.00      1.00     11625

    accuracy                           0.99     17481
   macro avg       0.58      0.65      0.59     17481
weighted avg       0.99      0.99      0.99     17481
```

```
1 evaluate_metrics(mnb)
```

```
Accuracy score of MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True) is:
0.9835821749327842
[[    0     0    12     0     4]
 [    0     0     8     0     7]
 [    0     0   143    27    33]
 [    0     0   128  5454    40]
 [    0     0    23     5 11597]]
              precision    recall  f1-score   support

         1.0       0.00      0.00      0.00        16
         2.0       0.00      0.00      0.00        15
         3.0       0.46      0.70      0.55       203
         4.0       0.99      0.97      0.98      5622
         5.0       0.99      1.00      1.00     11625

    accuracy                           0.98     17481
   macro avg       0.49      0.53      0.51     17481
weighted avg       0.99      0.98      0.98     17481
```

```
1 evaluate_metrics(pac)
```

```
Accuracy score of PassiveAggressiveClassifier(C=0.8, average=False, class_weight=None,
                           early_stopping=False, fit_intercept=True,
                           loss='hinge', max_iter=1000, n_iter_no_change=4,
                           n_jobs=None, random_state=None, shuffle=True,
                           tol=0.001, validation_fraction=0.1, verbose=0,
                           warm_start=False) is:
0.9873004976831989
[[    1     0    13     2     0]
 [    0     1    11     2     1]
 [    3     8   122    60    10]
 [    0     0    55  5563     4]
 [    0     0    42    11 11572]]
              precision    recall  f1-score   support

         1.0       0.25      0.06      0.10        16
         2.0       0.11      0.07      0.08        15
         3.0       0.50      0.60      0.55       203
         4.0       0.99      0.99      0.99      5622
         5.0       1.00      1.00      1.00     11625

    accuracy                           0.99     17481
   macro avg       0.57      0.54      0.54     17481
weighted avg       0.99      0.99      0.99     17481
```

```
1 evaluate_metrics(rfc)
```

```
curacy score of RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=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=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False) is:
9883301870602368
    1     0    10     3     2]
    0     1     4     4     6]
    2     6   117    70     8]
    0     0    44  5568    10]
    0     0    21    14 11590]]
              precision    recall  f1-score   support

         1.0       0.33      0.06      0.11        16
         2.0       0.14      0.07      0.09        15
         3.0       0.60      0.58      0.59       203
         4.0       0.98      0.99      0.99      5622
         5.0       1.00      1.00      1.00     11625

    accuracy                           0.99     17481
   macro avg       0.61      0.54      0.55     17481
ighted avg       0.99      0.99      0.99     17481
```

From all the above models, ***RandomForestClassifier is performing good So,we opt it as final model.***

Key Metrics for success in solving problem under consideration:

The entire Classification model's performance is found by Accuracy score, Confusion matrix and Classification Report (includes f1 score, recall, and precision). As the scraped data is having less number of ratings for 1, 2 star the performance metrics are low for them.

Interpretation of the Results:

Observations from pre-processing:

- There are some '-' values in the dataset. So, we replaced them with "No Rating".
- All the text data is converted to vectors using TFIDF-Vectorizer.

Observations from modelling:

- Target variable is selected as "Rating"

- Split the dataset to 80:20 for train and test respectively.
- All the data obtained from above steps given to model and observed the predictions.
- Random Forest Classifier has the better performance metrics.

CONCLUSION

Key Findings and Conclusions of the Study

Learning Outcomes of the Study in respect of Data Science:

- Scraping the data from E-Commerce websites made easy to make our own dataset.
- All the Data pre-processing steps made the problem easier to clean the data.
- Four Algorithms are used, in which "**Random Forest Classifier**" has the best classification metrics.

Limitations of this work and Scope for Future Work:

Limitations:

We tried to scrape the equal amount of ratings. So it couldn't make the dataset balanced.

Conclusion:

Rating Prediction made simpler by passing the model with Reviews. It gave us better understand the customer behavior for a particular product.