Rating Prediction Project

Submitted by:

Nahid Anjum Gouri

ACKNOWLEDGMENT

The project entitled “RATINGS PREDICTION PROJECT” is done by me during my internship with Flip Robo Technologies. I am grateful to Data Trained and Flip Robo Technologies for their guidance during this project. Other reference websites used to complete this project are:

1. Stackoverflow.com
2. Towardsdatascience.com
3. Medium.com

INTRODUCTION

Conceptual Background of the Domain Problem

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.

Problem Statement

Ratings Prediction: We have to predict ratings of all the products based on the reviews given by customers which will be pre-processed and trained.

Analytical Problem Framing

Data Collection Phase

- We have to scrape at least 20000 rows of data. We can scrape more data as well, it's up to us. More the data better the model In this section you need to scrape the reviews of different laptops, Phones, Headphones, smart watches, Professional Cameras, Printers, monitors, Home theater, router from different e-commerce websites. Basically, we need these columns: 1) reviews of the product. 2) rating of the product. You can fetch other data as well, if you think data can be useful or can help in the project. It completely depends on your imagination or assumption. Hint: – Try fetching data from different websites. If data is from different websites, it will help our model to remove the effect of over fitting. - Try to fetch an equal number of reviews for each rating, for example if you are fetching 10000 reviews then all ratings 1,2,3,4,5 should be 2000. It will balance our data set. - Convert all the ratings to their round number, as there are only 5 options for rating i.e., 1, 2, 3, 4, 5. If a rating is 4.5 convert it 5.

Model Building Phase

- After collecting the data, you need to build a machine learning model. Before model building, do all data pre-processing steps involving NLP. Try different models with different hyper parameters and select the best model. Follow the complete life cycle of data science. Include all the steps like
  1. Data Cleaning
  2. Exploratory Data Analysis
  3. Data Pre-processing
  4. Model Building
  5. Model Evaluation
  6. Selecting the best model

- Firstly I have collected data i.e, I did webscraping of a website Flipcart and Amazon then collected all the reviews and ratings of products that we wanted to scrape such as : ['laptops', 'Phones', 'Headphones', 'smart

watches', 'Professional Cameras', 'Printers', 'monitors', 'Home theater', 'router'].

Finally I created a dataframe and stored the ratings and reviews in it and also saved it in csv format. Sample data is as shown below:

1. Amazon Scrapping

```
7]: df=pd.DataFrame({})
    df['Ratings']=rating
    df['Reviews']=review
    df
```

7]:

	Ratings	Reviews
0	3.0 out of 5 stars	\n\n\n\n\n\n\n\n\n \n Your browser does not ...
1	1.0 out of 5 stars	\n\n\n\n\n\n\n\n\n\n \n \n \n Laptop get...
2	3.0 out of 5 stars	\n\n\n\n\n\n\n\n\n\n \n \n \n It looks t...
3	3.0 out of 5 stars	\n\n\n\n\n\n\n\n\n\n \n \n \n How to ins...
4	1.0 out of 5 stars	\n\n\n\n\n\n\n\n\n\n \n \n \n HiThis is ...
...	...	...
11597	4.0 out of 5 stars	Pros:-\n1. The Watch has great battery life ca...
11598	5.0 out of 5 stars	I have been using this watch since 14 days. Ba...
11599	4.0 out of 5 stars	PROS : GOOD BATTERY LIFE 🔋\nAmazing touch feed...
11600	4.0 out of 5 stars	This was my first smartwatch, and I was very e...
11601	4.0 out of 5 stars	Watch is good after the update please try to g...

11602 rows × 2 columns

```
8]: df.to_csv('Amazon_ratings_final.csv')
```

2. Flipcart Dataframe:

	Unnamed: 0	Ratings	Reviews
0	0	4	Product is good but the fan is running continu...
1	1	5	It's just amazing! Let me get the details here...
2	2	5	Very good It's a beast at the price of 40k Lov...
3	3	5	It's amazing! Best is this category. I just bo...
4	4	4	Thanks Flipkart for fast delivery in our lockd...
...	...	...	...
17295	17295	5	This is varrast product
17296	17296	1	Worst product no sound quality and not connect...
17297	17297	5	Super 👌
17298	17298	1	Very bad very very bad remote is not working bad
17299	17299	4	Nice

58535 rows × 3 columns

```
]: Rating_Final['Ratings'].value_counts()
```

```
]: 5    33910
   4    11686
   1     6682
   3     4093
   2     2164
   Name: Ratings, dtype: int64
```

As we can see that we have 11602 reviews from Amazon and 58535 reviews from Flipcart. We can also see that there is a huge difference in number of ratings. We have large number of ‘5’ ratings but less number of ‘2’ and ‘1’ ratings. If we had used data as it was it would have been an imbalanced model. So I have put all data together and used equal number of ratings. I have made a new dataframe with 4740 rows of each of the ratings.

First I have imported all required libraries and uploaded all data in csv format. As we can see below:

```
# Let's import the required Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import string
import re
from collections import Counter
from gensim import corpora
from gensim.utils import simple_preprocess
from gensim.parsing.preprocessing import STOPWORDS
from sklearn.feature_extraction.text import TfidfVectorizer


import nltk
from nltk.corpus import wordnet, stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer, SnowballStemmer

from nltk import pos_tag

import warnings
warnings.filterwarnings('ignore')
```

Then imported both csv files:

```
amazon=pd.read_csv("Amazon_ratings_final.csv")
amazon
```

```
flipkart=pd.read_csv("flipkart_ratings_final.csv")
flipkart
```

Then I have combined both dataframe :

```
#joining both data frames to create a single dataframe
ds=pd.concat([amazon,flipkart])
ds['Ratings']=(ds['Ratings']).astype(int)
```

```
ds
```

	Ratings	Reviews
0	3	Your browser does not support HTML5 video....
1	1	Laptop getting hang very much and ve...
2	3	It looks the screen size is small . ...
3	3	How to install a new SSD? The SSD do...
4	1	HiThis is a true feedback, requestin...
...	...	...
60500	5	This is varrast product
60501	1	Worst product no sound quality and not connect...
60502	5	Super 👌
60503	1	Very bad very very bad remote is not working bad
60504	4	Nice

72107 rows × 2 columns

As we can see in above image that there are 72107 rows . Now I have checked data for null value and value count:

```
ds.isnull().sum()
```

```
Ratings    0
Reviews    0
dtype: int64
```

```
ds["Ratings"].value_counts()
```

```
5    39257
4    13557
1     9480
3     5073
2     4740
Name: Ratings, dtype: int64
```

Then for avoiding imbalanced data problem I have rearranged the data frame and took only 4740 rows of each of the ratings. So this how I ended up having 23700 rows of ratings.

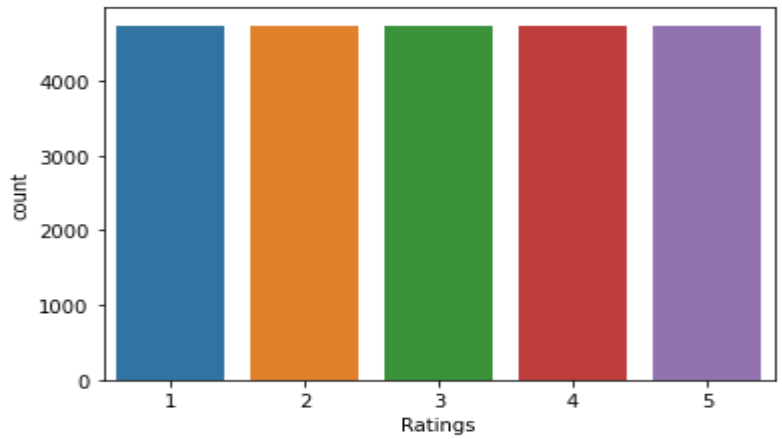
```
rating5 = ds[ds['Ratings'] == 5][:4740]
rating4 = ds[ds['Ratings'] == 4][:4740]
rating3 = ds[ds['Ratings'] == 3][:4740]
rating2 = ds[ds['Ratings'] == 2][:4740]
rating1 = ds[ds['Ratings'] == 1][:4740]

ds_Final=pd.concat([rating5,rating4,rating3,rating2,rating1],axis=0,ignore_index=True)
ds_Final
```

	Ratings	Reviews
0	5	Hp laptop with intel less than 40k.A...
1	5	A really good laptop for online clas...
2	5	Good laptop for day to day work .. i...
3	5	Likes:Best laptop under 50k budget.L...
4	5	I bought this exchanging my Acer Asp...
...	...	...
23695	1	It's very awful experience with the brand like...
23696	1	Sound Quality is Good , Battery Back is good ,...
23697	1	Received the product in good shape. Very good ...
23698	1	Bad product didn't work Requests not accepted...
23699	1	It is very bad

23700 rows × 2 columns

We can see that I now have 23700 rows of rating with each rating in equal number. As we can see in the countplot.

```
#visualizing the Rating column
sns.countplot(ds_Final['Ratings'])
plt.show()
```



Data Pre-Processing

- In this we will be performing data cleaning such as removing html tags, special characters, converting everything to lowercase, replace email addresses with 'email', URLs with 'webaddress', money symbols with 'moneysymb', 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenumber', numbers with 'numbr', whitespace between terms with a single space, removing leading and trailing whitespace and punctuations. We also removed Stopwords.

```
# 1. Remove HTML tags
#Regex rule : '<.*?>'

def clean(text):
    cleaned = re.compile(r'<.*?>')
    return re.sub(cleaned,'',text) # substring replace with ''(space)

ds_Final.Reviews = ds_Final.Reviews.apply(clean)
ds_Final.Reviews
```

```
0                   Hp laptop with intel less than 40k.A...
1                   A really good laptop for online clas...
2                   Good laptop for day to day work .. i...
3                   Likes:Best laptop under 50k budget.L...
4                   I bought this exchanging my Acer Asp...
                              ...
23695    It's very awful experience with the brand like...
23696    Sound Quality is Good , Battery Back is good ,...
23697    Received the product in good shape. Very good ...
23698    Bad product didn't work  Requests not accepted...
23699                                        It is very bad
Name: Reviews, Length: 23700, dtype: object
```

```
# 3. Convert everything to lowercase
def to_lower(text):
    return text.lower()

ds_Final.Reviews = ds_Final.Reviews.apply(to_lower)
ds_Final.Reviews
```

```
0                   hp laptop with intel less than 40k a...
1                   a really good laptop for online clas...
2                   good laptop for day to day work    i...
3                   likes best laptop under 50k budget l...
4                   i bought this exchanging my acer asp...
                              ...
23695    it s very awful experience with the brand like...
23696    sound quality is good   battery back is good  ...
23697    received the product in good shape  very good ...
23698    bad product didn t work  requests not accepted...
23699                                        it is very bad
Name: Reviews, Length: 23700, dtype: object
```

```
# Replace email addresses with 'email'
ds_Final['Reviews'] = ds_Final['Reviews'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$',
                                  'emailaddress')

    # Replace URLs with 'webaddress'
ds_Final['Reviews'] = ds_Final['Reviews'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$',
                                  'webaddress')

    # Replace money symbols with 'moneysymb' (£ can by typed with ALT key + 156)
ds_Final['Reviews'] = ds_Final['Reviews'].str.replace(r'£|\$', 'dollers')

    # Replace 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenumber'
ds_Final['Reviews'] = ds_Final['Reviews'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$',
                                  'phonenumber')


    # Replace numbers with 'numbr'
ds_Final['Reviews'] = ds_Final['Reviews'].str.replace(r'\d+(\.\d+)?', 'numbr')

    # Remove punctuation
ds_Final['Reviews'] = ds_Final['Reviews'].str.replace(r'[^\w\d\s]', ' ')

    # Replace whitespace between terms with a single space
ds_Final['Reviews'] = ds_Final['Reviews'].str.replace(r'\s+', ' ')

    # Remove leading and trailing whitespace
ds_Final['Reviews'] = ds_Final['Reviews'].str.replace(r'^\s+|\s+?$', '')
```

- Then we performed stemming using Snowball Stemmer and WordNetLemmatizer. Then we further pre-process the text and save the final processed data in processed_review.

```
from nltk.stem import SnowballStemmer, WordNetLemmatizer
stemmer = SnowballStemmer("english")
import gensim
def lemmatize_stemming(text):
    return stemmer.stem(WordNetLemmatizer().lemmatize(text,pos='v'))
```

```
#Tokenize and Lemmatize
def preprocess(text):
    result=[]
    for token in text:
        if len(token)>=3:
            result.append(lemmatize_stemming(token))

    return result
```

```
# Processing review with above Function
processed_review = []

for doc in ds_Final.Reviews:
    processed_review.append(preprocess(doc))

print(len(processed_review))
processed_review[:3]
```

- To get sense of loud words in Review corresponding to each rating i., 1, 2, 3, 4 and 5 we have used wordcloud and visualized it as follows:

```
Display_wordcloud(ds_Final['Reviews'][ds_Final['Ratings']==1],"RATING-1")
```


```
Display_wordcloud(ds_Final['Reviews'][ds_Final['Ratings']==2],"RATING-2")
```

Model/s Development and Evaluation

Review column have been converted to tokens using TFidfVectorizer. Then using train_test_split we split the data into training and testing dataset. All the required libraries have been imported for building model:

```
from sklearn.feature_extraction.text import TfidfVectorizer

tf_vec = TfidfVectorizer()
features = tf_vec.fit_transform(ds_Final['Reviews']) #Using

X = features
y = ds_Final['Ratings']

print("X.shape = ",X.shape)
print("y.shape = ",y.shape)
```

```
X.shape =  (23700, 14981)
y.shape =  (23700,)
```

```
# Importing libraries for model training

from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier



from sklearn.model_selection import cross_val_score, cross_val_predict, train_test_split
from sklearn.model_selection import GridSearchCV


# Importing evaluation metrics for model performance....
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.metrics import roc_auc_score, roc_curve, auc
from sklearn.metrics import precision_score, recall_score, f1_score
```

```
#splitting the data into training and testing
x_train,x_test,y_train,y_test=train_test_split(X,y,random_state=42,test_size=0.30,stratify=y)
```

We have used following algorithms such as: RandomForestClassifier, AdaBoostClassifier, MultinomialNB, XGBClassifier, DecisionTreeClassifier and KNeighborsClassifier:

```
# List of Models
models=[]

models.append(('MultinomialNB()',MNB))
models.append(('DecisionTreeClassifier',DT))
models.append(('KNeighborsClassifier',KNN))
models.append(('RandomForestClassifier',RFC))
models.append(('AdaBoostClassifier',ADA))
models.append(('XGBClassifier', XGB))
```

We have formed a loop where all the algorithms will be used one by one and their corresponding accuracy_score, cross_val_score and classification report will be evaluated.

```
for name,model in models:
    print('**************',name,'***************')
    print('\n')
    Model.append(name)
    model.fit(x_train,y_train)
    print(model)
    pre=model.predict(x_test)
    print('\n')
    AS=accuracy_score(y_test,pre)
    print('Accuracy_score=',AS)
    score.append(AS*100)
    print('\n')
    sc=cross_val_score(model,X,y,cv=5,scoring='accuracy').mean()
    print('Cross_Val_Score=',sc)
    cvs.append(sc*100)
    print('\n')
    print('classification report\n',classification_report(y_test,pre))
    print('\n')
    cm=confusion_matrix(y_test,pre)
    print(cm)
    print('\n')
    plt.figure(figsize=(10,40))
    plt.subplot(911)
    plt.title(name)
    print(sns.heatmap(cm,annot=True))
    #plt.subplot(912)
    #plt.title(name)
    print('\n\n')
```

Key metrics used for finalising the model was accuracy_score, cross_val_score. Since in case of RandomForestClassifier it was giving us maximum score among all other models and it's performing well. It's cross_val_score is also satisfactory and it shows that our model is neither underfitting/overfitting.

```
result=pd.DataFrame({'Model': Model,'Score': score,'Cross_Val_Score':cvs})
result
```

	Model	Score	Cross_Val_Score
0	MultinomialNB()	56.019691	39.316456
1	DecisionTreeClassifier	64.683544	45.565401
2	KNeighborsClassifier	37.313643	28.172996
3	RandomForestClassifier	70.239100	51.054852
4	AdaBoostClassifier	50.703235	41.805907
5	XGBClassifier	66.568214	48.046414

Since RandomForestClassifier was giving us maximum score we applied hyper parameter tuning on it to further improve the result using GridSearchCV.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
rf=RandomForestClassifier()
parameters={'n_estimators':[50,100,150,200,250,300,350,400,450,500],'max_depth':[5, 8, 15, 25, 30]}
clf=GridSearchCV(rf,parameters,cv=5)
clf.fit(X,y)
clf.best_params_
```

```
{'max_depth': 30, 'n_estimators': 450}
```

```
#Applying the parameters we got after hyper parameter tuning
rf=RandomForestClassifier(n_estimators=450,max_depth=30)
rf.fit(x_train,y_train)
predrf=rf.predict(x_test)
print(accuracy_score(y_test,predrf))
print(confusion_matrix(y_test,predrf))
print(classification_report(y_test,predrf))
```

```
0.6526019690576652
[[ 912  310   58   54   88]
 [ 173  955  160   77   57]
 [  89  369  706  127  131]
 [  51   97  157  817  300]
 [  52   41   13   66 1250]]
              precision    recall  f1-score   support

           1       0.71      0.64      0.68      1422
           2       0.54      0.67      0.60      1422
           3       0.65      0.50      0.56      1422
           4       0.72      0.57      0.64      1422
           5       0.68      0.88      0.77      1422
```

Since it didn't show any improvement in result even after hyperparameter tuning. Hence, we will be choosing the RandomForestClassifier Model with default values that we gotearlier.

We will further use that model to predict our test data set as follows:

```
test_data=pd.DataFrame(data=y_test,)
test_data['Predicted values']=Predrf

test_data.to_csv('Ratings_Predict.csv')

test_data
```

	Ratings	Predicted values
10735	3	3
9775	3	5
20653	1	1
6730	4	2
15906	2	2
...	...	...
19149	1	1
23506	1	3
10472	3	2
20559	1	1
8566	4	4

7110 rows × 2 columns

Then we have saved predicted values as pickle file:

```
# Creating Pickle File
import joblib
joblib.dump(RF,'Ratings_Prediction_dataset.pkl')
```

```
['Ratings_Prediction_dataset.pkl']
```

Conclusion

- We got 70% accuracy score with RandomaforestClassifier.
- We had to go through multiple source of data in order to balanced dataset.
- We can use this model for predicting customers inclination for future use.