Product Rating Prediction Project

Submitted by:

Mekhala Misra

ACKNOWLEDGMENT

I took help from following websites:

1)Geek for geeks

2)Pandas documentation

3)researchgate.net

INTRODUCTION

- ## Business Problem Framing

  Depending on customers reviews we have to predict the rating that they could have given to that product. It is a NLP problem. So here we are working on this project in 2 phases:

  1. Data Collection: I collected reviews and rating data from amazon and flipkart for mobile, laptops and smart watches.
  2. Analysing the data and building the model.

Analytical Problem Framing

- ## Mathematical/ Analytical Modelling of the Problem

  This is a NLP problem as we have to analyse the reviews of customers and depending on that we have to predict ratings given by them.

- ## Data Sources and their formats

  Data set is collected reviews and rating data from amazon and flipkart for mobile, laptops and smart watches.

```
In [14]:  1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5474 entries, 0 to 5473
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Review  5467 non-null   object 
 1   Rating  5467 non-null   float64
dtypes: float64(1), object(1)
memory usage: 85.7+ KB
```

Following are the columns present and their respective data types:

```
In [14]:
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5474 entries, 0 to 5473
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Review  5467 non-null   object
 1   Rating  5467 non-null   float64
dtypes: float64(1), object(1)
memory usage: 85.7+ KB
```

Here is the glimpse of data:

Out[5]:

	Unnamed: 0	Review	Rating
0	0	Nice product	4.0
1	1	Good choice	4.0
2	2	Classy product	5.0
3	3	Moderate	5.0
4	4	Super!	4.0

- # Data Pre-processing

- The dataset had no null values
- Feature Engineering-We have to fetch comment words from the reviews so in order to do that we first converted entire reviews feature to lower case then replaced all the phone numbers, email ids, URL's, any sort of number and currency by phno,emailed,link,numbr and currency respectively.

In [11]:
```
#converting all the reviews to lower case so thats its easy to analyse them.
df_up['Review']=df_up['Review'].str.lower()
```

In [12]:
```
#Replacing email address,links, phone numbers, any sort of numbers and currency as they are not a review
df_up['Review'] = df_up['Review'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$','emailid')
df_up['Review'] = df_up['Review'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$','link')
df_up['Review'] = df_up['Review'].str.replace(r'£|\$', 'currency')
df_up['Review'] = df_up['Review'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$','phno')
df_up['Review'] = df_up['Review'].str.replace(r'\d+(\.\d+)?', 'numbr')
```

- Further we removed all kinds of punctuations from the feature Review.

In [67]:
```
#Removing punctuations
df_up['Review'] = df_up['Review'].apply(lambda x: ' '.join(
    term for term in str(x).split() if term not in string.punctuation))
```

- In order to fetch just the abusive words we also have to remove all kind of stop words.

In [68]:
```
# Removing stop words
sw = set(stopwords.words('english') + ['u', 'ur', '4', '2', 'im', 'dont', 'doin', 'ure'])
df_up['Review'] = df_up['Review'].apply(lambda x: ' '.join(
    term for term in str(x).split() if term not in sw))
```

- In order to further analyse the Review using morphological analysis of the words called lemmatization.

In [69]:
```
lm=WordNetLemmatizer()
df_up['Review'] = df_up['Review'].apply(lambda x: ' '.join(
 lm.lemmatize(t) for t in str(x).split()))
```

- Then finally we converted the Review into its vector form Term Frequency and Inverse Document Frequency method

In [73]:
```
# Convert text into vectors using TFIDF

tfidf = TfidfVectorizer(max_features = 10000, stop_words='english')
x = tfidf.fit_transform(df_up['Review'].apply(lambda x: np.str_(x)))
```

- We split the data into training set and testing set using train test split method.
- On this train and test data we applied various models: logistic regression, decision tree classifier, random forest classifier , Knn classifier.
- The best performing model is random forest as it has least difference between cv score and test accuracy and recall, f1-score is the best among all. Since the model accuracy is very low so we performed hyperparameter tuning on it but there is no significant change in accuracy.
- So, we saved our previous random forest classifier model

Hardware and Software Requirements and Tools Used

We imported following packages:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import LabelEncoder
from statsmodels.stats.outliers_influence import variance_inflation_factor
from scipy.stats import zscore
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.preprocessing import power_transform
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn import metrics
from sklearn.metrics import roc_curve,auc,classification_report
from sklearn.feature_extraction.text import TfidfVectorizer
import warnings
warnings.filterwarnings('ignore')
from nltk.stem import WordNetLemmatizer
import nltk
from nltk.corpus import  stopwords
import string
```

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

- Since this is a classification problem so I will use all the classification algorithms such as:
  Decision Tree classifier
  Random Forest classifier
  Knn classifier
  Logistic regressor

- Testing of Identified Approaches (Algorithms)

- We applied Logistic regression, Decision Tree classifier,random forest classifier, knn classifier on the clean data

- Depending on the model accuracy, confusion matrix, auc-roc curve and classification report we opted random forest classifier.

- Since random forest classifier is giving best accuracy already so I did not apply hyper parameter tuning on dataset as the dataset is very large and time consuming and scope of accuracy improvement is not much high. So, we saved our previous random forest classifier model.

- Run and Evaluate selected models

1) Decision Tree Classifier

We applied this algorithm and found the train accuracy to be 36.4% and test accuracy to be 36.6%.

In [97]:
```
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier()
dt.fit(x_train,y_train)
dt_pred_train=dt.predict(x_train)
dt_pred_test=dt.predict(x_test)
dt_acc_train=round(accuracy_score(y_train,dt_pred_train)*100,1)
dt_acc_test=round(accuracy_score(y_test,dt_pred_test)*100,1)
print("acc train",dt_acc_train)
print("acc test",dt_acc_test)
```

```
acc train 36.4
acc test 36.6
```

In [98]:
```
# CV Score
cv_score_best_dt=cross_val_score(dt,x,y,cv=4).mean()*100
print("cross validation score is-",cv_score_best)
print("accuracy score for decision tree model is-",dt_acc_test)
```

```
cross validation score is- 36.42223454394233
accuracy score for decision tree model is- 36.6
```

The test accuracy for decision tree regression is36.6% and its cv score is 36.4% thus making us sure that the model is not overfitted, although decision tree regressor is prone to overfitting.

We also tested this model on other metrics- classification report:

In [92]:
```
#classification report
print(classification_report(y_test, dt_pred_test))
```

```
              precision    recall  f1-score   support

         2.0       0.00      0.00      0.00       737
         3.0       0.00      0.00      0.00       688
         4.0       0.25      0.89      0.39       713
         5.0       0.25      0.10      0.15       748

    accuracy                           0.25      2886
   macro avg       0.13      0.25      0.13      2886
weighted avg       0.13      0.25      0.13      2886
```

2) Random Forest Classifier

We applied this algorithm and found the train accuracy to be 36.4% and test accuracy to be 36.6%.

In [94]:
```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=10,test_size=0.20)
rf.fit(x_train,y_train)
pred_train=rf.predict(x_train)
pred_test=rf.predict(x_test)
rf_train_acc=round(accuracy_score(y_train,pred_train)*100,1)
rf_test_acc=round(accuracy_score(y_test,pred_test)*100,1)
print("\nTrain Accuracy- ",rf_train_acc)
print("\nTest Accuracy- ",rf_test_acc)
```

```
Train Accuracy-  36.4

Test Accuracy-  36.6
```

In [95]:
```
cv_score_best_rf=cross_val_score(rf,x,y,cv=11).mean()*100
print("cross validation score is-",cv_score_best_rf)
print("accuracy score for Knn classifier model is-",rf_test_acc)
```

```
cross validation score is- 36.43986116114542
accuracy score for Knn classifier model is- 36.6
```

The test accuracy for random forest classifier is 36.4% and its cv score is 36.6% thus making us sure that the model is not overfitted.

We also tested this model on other metrics- classification report:

In [96]:
```
#classification report
print(classification_report(y_test, pred_test))
```

```
              precision    recall  f1-score   support

         2.0       0.00      0.00      0.00       726
         3.0       0.28      1.00      0.44       717
         4.0       0.00      0.00      0.00       746
         5.0       1.00      0.49      0.65       697

    accuracy                           0.37      2886
   macro avg       0.32      0.37      0.27      2886
weighted avg       0.31      0.37      0.27      2886
```

3) Knn classifier

We applied this algorithm and found the train accuracy to be 36.6% and test accuracy to be 35.6%.

In [88]:
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=1,test_size=0.20)
knn.fit(x_train,y_train)
pred_train=knn.predict(x_train)
pred_test=knn.predict(x_test)
knn_train_acc=round(accuracy_score(y_train,pred_train)*100,1)
knn_test_acc=round(accuracy_score(y_test,pred_test)*100,1)
print("\nTrain Accuracy- ",knn_train_acc)
print("\nTest Accuracy- ",knn_test_acc)
```

```
Train Accuracy-  36.6

Test Accuracy-  35.6
```

In [89]:
```
cv_score_best_knn=cross_val_score(knn,x,y,cv=11).mean()*100
print("cross validation score is-",cv_score_best_knn)
print("accuracy score for Knn classifier model is-",knn_test_acc)
```

```
cross validation score is- 36.37056541019955
accuracy score for Knn classifier model is- 35.6
```

The test accuracy for Knn regression is 35.6% and its cv score is 36.3% thus making us sure that the model is not overfitted.

We also tested this model on other metrics- classification report:

In [93]:
```
#classification report
print(classification_report(y_test, pred_test))
```

```
              precision    recall  f1-score   support

         2.0       0.00      0.00      0.00       737
         3.0       0.27      1.00      0.43       688
         4.0       0.00      0.00      0.00       713
         5.0       1.00      0.45      0.62       748

    accuracy                           0.36      2886
   macro avg       0.32      0.36      0.26      2886
weighted avg       0.32      0.36      0.26      2886
```

4) Logistic Regressor

We applied this algorithm and found the train accuracy to be 36.6% and test accuracy to be 36.0%.

In [99]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=9,test_size=0.20)
lr.fit(x_train,y_train)
pred_train=lr.predict(x_train)
pred_test=lr.predict(x_test)
train_accuracy=round(accuracy_score(y_train,pred_train)*100,1)
test_accuracy=round(accuracy_score(y_test,pred_test)*100,1)
print("\ntrain accuracy-",train_accuracy)
print("\ntest accuracy-",test_accuracy)
```

```
train accuracy- 36.6

test accuracy- 36.0
```

In [100]:
```
# CV Score
cv_score_best=cross_val_score(lr,x,y,cv=4).mean()*100
print("cross validation score is-",cv_score_best)
print("accuracy score for logistic regression model is-",test_accuracy)
```

```
cross validation score is- 36.42223454394233
accuracy score for logistic regression model is- 36.0
```

The test accuracy for Logistic regressor regression is 36.0% and its cv score is 36.4% thus making us sure that the model is not overfitted.

We also tested this model on other metrics- classification report:

In [83]:
```
#classification report
print(classification_report(y_test, pred_test))
```

```
              precision    recall  f1-score   support

         2.0       0.00      0.00      0.00       747
         3.0       0.00      0.00      0.00       739
         4.0       0.28      1.00      0.44       729
         5.0       1.00      0.46      0.63       671

    accuracy                           0.36      2886
   macro avg       0.32      0.37      0.27      2886
weighted avg       0.30      0.36      0.26      2886
```

The best performing model is random forest Regressor as its test accuracy and CV score both are almost same. So we will finalize this model.
We further performed hyperparameter tuning on it but there was no significant change in accuracy.