FLIP ROBO

RATING PREDICTION

Submitted by:

ARPITA MISHRA

Introduction

Present project aimed at building a model to predict star rating based on the review. Around 20,000 reviews and ratings of different laptops, phones, smart watches, cameras were scrapped using selenium as a tool. These review data were used to train and build a model.

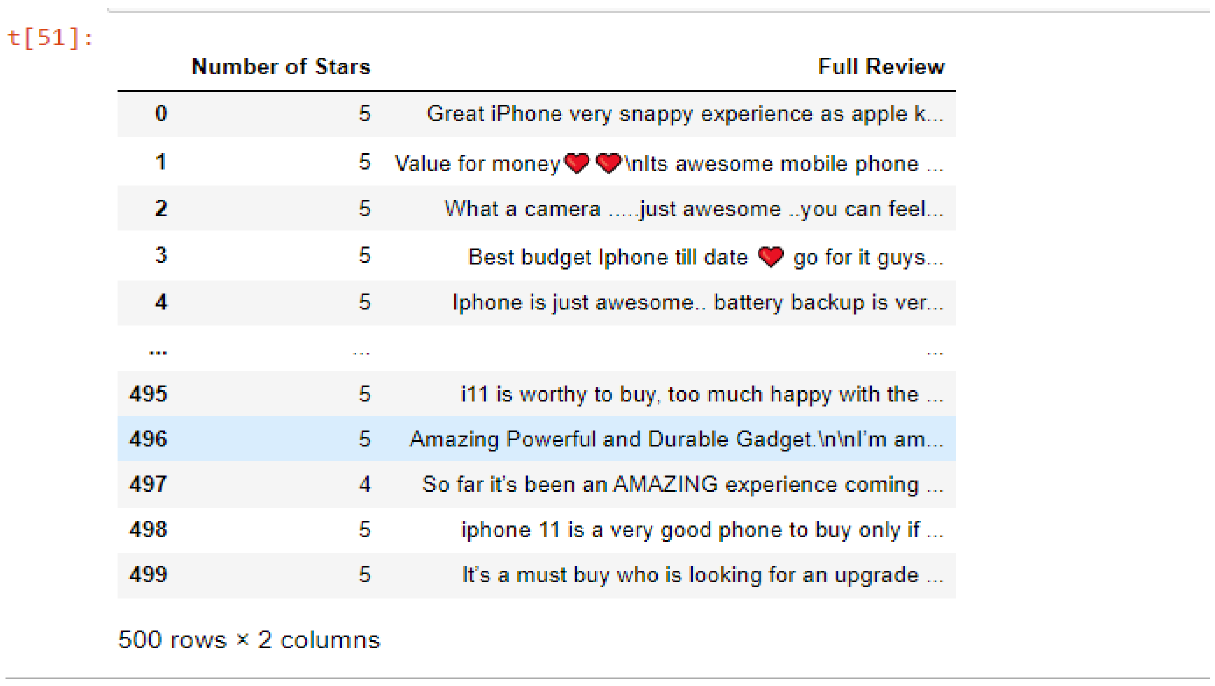
Data Collection(Scraping data)

Using Selenium web driver, review and ratings were fetched for different products from e-commerce sites. These data were saved in the form of csv file.

import all the required libraries

```
import pandas as pd
import selenium
from selenium import webdriver
import time
fromselenium.common.exceptionsimport
StaleElementReferenceException, NoSuchElementException

driver=webdriver.Chrome("chromedriver.exe")
time.sleep(3)

url = "https://www.flipkart.com/apple-iphone-11-black-64-gb-includes-
earpods-power-adapter/product-
reviews/itm0f37c2240b217?pid=MOBFKCTSVZAXUHGR&lid=LSTM
OBFKCTSVZAXUHGREPBFGI&marketplace=FLIPKART"

driver.get(url)
time.sleep(2)
#creating empty list
urls=[]
stars=[]
complete_review=[]
time.sleep(5)
```

```
#Taking 50 pages into consideration using for loop

for i in range(50):


url=driver.find_element_by_xpath("//a[@class='_1LKTO3']").get_attribute('
href')
driver.get(url)

#for scrapping the number of stars
  for j in driver.find_elements_by_xpath("//div[@class='_3LWZlK _1BLPMq']"):
      stars.append(j.text)


#for scrapping the complete review

 for k in driver.find_elements_by_xpath("//div[@class='t-ZTKy']/div/div"):

             complete_review.append(k.text)


#Combining all the lists into a single dataframe

      df1=pd.DataFrame({'Number of Stars': stars,'Full Review':
      complete_review})
      df1
```



t[51]:

	Number of Stars	Full Review
0	5	Great iPhone very snappy experience as apple k...
1	5	Value for money❤❤\nIts awesome mobile phone ...
2	5	What a camera .....just awesome ..you can feel...
3	5	Best budget Iphone till date ❤ go for it guys...
4	5	Iphone is just awesome.. battery backup is ver...
...	...	...
495	5	i11 is worthy to buy, too much happy with the ...
496	5	Amazing Powerful and Durable Gadget.\n\nI'm am...
497	4	So far it's been an AMAZING experience coming ...
498	5	iphone 11 is a very good phone to buy only if ...
499	5	It's a must buy who is looking for an upgrade ...

500 rows × 2 columns

Similarly, ratings and review were fetched for multiple products to get around 20,000 data. These data were saved in csv file and will be used for model building

Model Building

For model building, all the necessary libraries like Pandas, numpy and matplot lib were imported on jupyter notebook. The data set was loaded as df by using **pd.read_csv.** The shape of the dataset was found to be 20370 rows and 3 columns. The columns included the review and their respective ratings of several electronic products.

To see if the dataset was balanced or not value_counts() was used which showed that the dataset was not imbalanced.

- df['Number of Stars'].value_counts()

```
5    11920
4     4124
1     2227
3     1744
2      355
Name: Number of Stars, dtype: int64
```

Data PreProcessing

The data collected from the e-commerce websites were not clean. So the data had to be preprocessed before it could be used to train a model. All the text in the data were converted in lower case, punctuations, special characters and digits were removed.

```
import re
import string

def text_clean_1(text):
    text=text.lower()
    text=re.sub('\[.*?#]','',text)
    text=re.sub('[%s]'%re.escape(string.punctuation),'',text)
    text=re.sub('\w*\d\w*','',text)
    return text
```

```
cleaned1=lambda x:text_clean_1(x)
df['cleaned_FullReview']=pd.DataFrame(df['Full
Review'].apply(cleaned1))
```

Beside this, all the emojis present in the data were also removed.

```
def text_clean_2(text):
    text=re.sub('\n','',text)
    text=re.sub(emoji.get_emoji_regexp(),r'',text)
    return text
cleaned2=lambda x:text_clean_2(x)

df['cleaned_review_final']=pd.DataFrame(df['cleaned_FullReview'].apply
(cleaned2))
```

Algorithm

The data were split into x and y using train-test-split. The review was used as independent variable while the rating was used as dependent variable(label). TFIDF vectorizer was imported for word embedding and Linear SVC was used to build the model. A pipeline was created which included firstly vectorization followed by the use of classifier.

The model was trained by model.fit using train data set. The trained model was used to predict the rating from test data. The predicted data along with 'y_test' data was used to find accuracy score, confusion matrix and classification report. Following code were used:

```
from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from sklearn.svm import LinearSVC

from sklearn.pipeline import Pipeline
```

```
tvec=TfidfVectorizer()
clf2=LinearSVC()
model=Pipeline([('vectorizer',tvec),('classifier',clf2)])
model.fit(x_train,y_train)
pred=model.predict(x_test)
print(confusion_matrix(pred,y_test))
print(accuracy_score(pred,y_test))
```

Result

The accuracy score was found to be 97.8 percent. The model was tested on a new review and was found to have a good accuracy.

Classification report :

```
              precision    recall  f1-score   support

           1       0.98      0.99      0.99       444
           2       1.00      1.00      1.00        59
           3       0.93      1.00      0.96       340
           4       0.93      1.00      0.96       755
           5       1.00      0.97      0.98      2476

    accuracy                           0.98      4074
   macro avg       0.97      0.99      0.98      4074
weighted avg       0.98      0.98      0.98      4074
```

Confusion matrix:

```
[[ 441    0    3    0    0]
 [   0   59    0    0    0]
 [   0    0  340    0    0]
 [   0    0    0  755    0]
 [   8    0   23   55 2390]]
```

Testing the model on new review

```
example= ['pathetic product']

result= model.predict(example)

print(result)
```

Saving the model

The model was saved using joblib.

```
import joblib

filename='ratingmodel.pkl'

joblib.dump(model,'ratingmodel.pkl')
```

Summary and conclusion

Thus, in the present project, review and rating of several electronic product were scrapped to be used to build a model that can be used to predict ratings based on the review. A linear SVC model was build which exhibited the accuracy of almost 98 percent. This model will be now used to predict ratings on a new review.