CUSTOMER RATINGS PREDICTION

Submitted by:

Aaron D’souza

ACKNOWLEDGMENT

References:

Paper refereed: https://towardsdatascience.com/review-rating-prediction-a-combined-approach-538c617c495c

Link (For concepts): https://www.udemy.com/

Link (TfIdf intuition): https://youtu.be/D2V1okCEsiE

Link (github reference): https://github.com/anujvyas/Natural-Language-Processing-Projects/blob/master/Sentiment%20Analysis%20-%20Restaurant%20Reviews/Sentiment%20Analysis%20of%20Restaurant%20Reviews.ipynb

Link (Kaggle reference): https://www.kaggle.com/lakshmi25npathi/sentiment-analysis-of-imdb-movie-reviews

INTRODUCTION

- Business Problem Framing

  The rise in E — commerce, has brought a significant rise in the importance of customer reviews.

  There are hundreds of review sites online and massive amounts of reviews for every product.

  Customers have changed their way of shopping and according to a recent survey, 70 percent of customers say that they use rating filters to filter out low rated items in their searches.

  The ability to successfully decide whether a review will be helpful to other customers and thus give the product more exposure is vital to companies that support these reviews, companies like Google, Amazon and Yelp!.

- ## Conceptual Background of the Domain Problem

In the present era, most of the data on the internet is in the form of raw text. These gold mines of data are invaluable since it contains lots of underlying information which can be extracted using natural language processing or text analytics techniques.

The data from these text-based documents disclose users' sentiments and opinions about a particular subject.

In this project, customer reviews from Amazon.com, Flipkart.com, and Paytm mall are pre-processed, analysed using our proposed framework, and how these textual reviews justify the star ratings is studied. Features derived from textual reviews are used to predict its corresponding star ratings.

To accomplish it, the prediction problem is transformed into a multi-class classification task to classify reviews to one of the five classes corresponding to its star rating.

- ## Review of Literature

In recent times, several opinion mining and sentiment analysis studies of reviews have been conducted.

SVM and Naive Bayes classifiers are trained to classify the movie ratings as either "high" or "low" based on its reviews.

Various linguistic features are extracted from the textual reviews and feature selection is performed using TF-IDF and information gain. The results found that the SVM classifier modelled using features selected based on information gain was most accurate.

However, the model lacks granularity as it cannot distinguish between "bad" (with 2 star rating) and "worst" (with 1 star rating) reviews.

- Motivation for the Problem Undertaken

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.

As an aspiring Data Scientist my task is to preprocess the text data and use the features derived from textual reviews to predict it's corresponding star ratings.

Analytical Problem Framing

- ## Mathematical/ Analytical Modelling of the Problem

The goal is to build a model that will extract information from text-based data and predict the corresponding customer ratings.

In terms of model building we have two columns "Review" column and "Rating" column. There are 5 values of rating ranging from (1 to 5).

Models used for extracting information from text based reviews:

- **Bags of words model**
- **Term Frequency-Inverse Document Frequency model (TF-IDF)**

Classification model used:

- **Naive Bayes**
- **Logistic Regression.**

- ## Data Sources and their formats

The sample data was scrapped from three e-commerce websites Amazon.com Flipkart.com and Paytm mall.

The data was scrapped using selenium web-driver and than saved in .CSV (comma-separated values) format.

A total of three data-types in the dataset Int and Object.

The data set has a total of 37 columns:

Review : Customer reviews for the products

Rating : ratings on customer reviews

	Review	Rating
0	look ok, but my heart monitor sensors stopped ...	3
1	Sudden drop in price after I order,\nAdapter I...	1
2	good	5
3	LAN ports are 10/100 Mbps so technically you a...	2
4	This phone is fantastic just wow it's as good ...	5

Fig.1Dataframe Sample

- Data Preprocessing Done

  Dropping Null values:

```
In [13]: df.isnull().sum()
Out[13]: Review    14
         Rating     0
         dtype: int64
```

```
In [14]: # There are only 14 missing values in 20000 so we can drop them
         df.dropna(inplace=True)
```

```
In [15]: df.isnull().sum()
Out[15]: Review    0
         Rating    0
         dtype: int64
```

Fig.2 Drop null values

Checking for any blank spaces in the dataset:

```
## Lets check for any kind of blank space
blanks = []
for i,rev,rat in df.itertuples():
    if type(rev) == "str":

        if rev.isspace():

            blanks.append(i)
```

fig.3 Drop blank spaces

Checking for any “-” symbols in the dataset:

```
In [19]:
## Lets try to remove the "-" symbol
df[df["Review"] == "-"]
```

Out[19]:

	Review	Rating
10	-	1
112	-	3
393	-	4
468	-	2
663	-	5

fig.4 dropping “-” symbol

Creating a function to remove and replace “-” symbols with corresponding texts:

In [22]:
```
def impute_Reviewes(cols):

    Review = cols[0]
    Rating = cols[1]

    if pd.isnull(Review):

        if Rating == 1:
            return "Very Bad Product"
        elif Rating == 2:
            return "It's a decent product not good enough"
        elif Rating == 3:
            return "Average product"
        elif Rating == 4:
            return "Very Good Product"
        elif Rating == 5:
            return "Best Product ever"

    else:
        return Review
```

fig.5 function to replace “-” symbol

Replacing all the email addresses, web addresses money symbols and phone numbers:

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

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

```
# Replace money symbols with 'moneysymb' (£ can by typed with ALT key + 156)
df['Review'] = df['Review'].str.replace(r'£|\$₹', 'moneysymbol')
```

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

fig.6 text preprocessing

Replacing all the Numbers , punctuations and white spaces from text:

```
# Replace numbers with 'number'
df['Review'] = df['Review'].str.replace(r'\d+(\.\d+)?', 'number')
```

```
# Remove punctuation
df['Review'] = df['Review'].str.replace(r'[^\w\d\s]', ' ')
```

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

fig.7 text preprocessing

Removing all the stop words from the text:

```
# create a variable name stop word

stop_words = set(stopwords.words('english') + my_words)
```

```
# Removing Stop words
df["Review"] = df["Review"].apply(lambda x: ' '.join(term for term in x.split() if term not in sto
```

```
df.head()
```

	Review	Rating	Length
0	look ok heart monitor sensors stopped working ...	3	88
1	sudden drop price order adapter looks duplicate	1	60
2	good	5	4
3	lan ports number number mbps technically alway...	2	383
4	phone fantastic wow good expected battery almo...	5	250

fig.8 stop word removal

Removing all the emojis from the text:

In [86]:
```
# We need to remove the emojes from review texts
def deEmojify(text):

    regrex_pattern = re.compile(pattern = "["
        u"\U0001F600-\U0001F64F"  # emoticons
        u"\U0001F300-\U0001F5FF"  # symbols & pictographs
        u"\U0001F680-\U0001F6FF"  # transport & map symbols
        u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                           "]+", flags = re.UNICODE)
    return regrex_pattern.sub(r'',text)
```

In [87]:
```
df["Review"] = df["Review"].apply(deEmojify)
```

fig.9 emojis removal

Removing all the foreign languages from the text:

```
# We need to remove all the Foreign languages from the reviewes
df = df[df['Review'].map(lambda x: x.isascii())]
```

```
df.head()
```

	Review	Rating
0	look ok heart monitor sensors stopped working ...	3
1	sudden drop price order adapter looks duplicate	1
2	good	5
3	lan ports number number mbps technically alway...	2
4	phone fantastic wow good expected battery almo...	5

fig.10 foreign language removal

Using porter steamer on individual words:

```
#Stemming the text
def simple_stemmer(text):

    ps=nltk.porter.PorterStemmer()

    text= ' '.join([ps.stem(word) for word in text.split()])

    return text
```

```
df['Review'] = df['Review'].apply(simple_stemmer)
```

fig.11 applying porter steamer

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

In this project my goal was to check the researchers' thesis. It was not to find the best model for the problem. I will try to prove that combining formerly known data about each user's similarity to other users, with the sentiment analysis of the review text itself, will help us improve the model prediction of what rating the user's review will get.

- Hardware and Software Requirements and Tools Used

Hardware used:

OS: Windows 10 Home Single Language 64 bit

Ram: 8 GB

Processor: Intel I5

Software used:

Jupyter Notebook

Model/s Development and Evaluation

- Testing of Identified Approaches (Algorithms)

Models used for extracting information from text based reviews:

- **Bags of words model:**

  The bag-of-words model is a simplifying representation used in natural language processing and information retrieval (IR). In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. The bag-of-words model has also been used for computer vision.

  The bag-of-words model is commonly used in methods of document classification where the (frequency of) occurrence of each word is used as a feature for training a classifier.

Document	the	cat	sat	in	hat	with
the cat sat	1	1	1	0	0	0
the cat sat in the hat	2	1	1	1	1	0
the cat with the hat	2	1	0	0	1	1

fig.12 BOW model example

- **Term Frequency-Inverse Document Frequency model (TF-IDF) :**

  In information retrieval, tf–idf, TF*IDF, or TFIDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling.

  The tf–idf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$$\begin{aligned} tf_{i,j} &= \text{number of occurrences of } i \text{ in } j \\ df_i &= \text{number of documents containing } i \\ N &= \text{total number of documents} \end{aligned}$$

fig.13 Tfidf formula

Classification model used:

- **Naive Bayes:**

  A naive Bayes classifier is an algorithm that uses Bayes' theorem to classify objects. Naive Bayes classifiers assume strong, or naive, independence between attributes of data points. Popular uses of naive Bayes classifiers include spam filters, text analysis and medical diagnosis. These classifiers are widely used for machine learning because they are simple to implement.

- **Logistic Regression:**

  Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

- Run and Evaluate selected models

  Models used for extracting information from text based reviews:

  Bags of words model

Applying count vectorizer

```
from sklearn.feature_extraction.text import CountVectorizer
```

```
vectorizer = CountVectorizer()
```

```
reviews_countvectorizer = vectorizer.fit_transform(df['Review'])
```

```
#print(vectorizer.get_feature_names())
```

```
print(reviews_countvectorizer.toarray())
```

```
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

fig.14 Applying BOW model

Term Frequency-Inverse Document Frequency model (TF-IDF)

Applying TFIDF

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
tfidf = TfidfVectorizer()
```

```
reviews_tv = tfidf.fit_transform(df['Review'])
```

```
#print(tfidf.get_feature_names())
```

```
print(reviews_tv.toarray())
```

```
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
```

fig.15 Applying Tfidf model

Classification model used:

Naive Bayes

Naive Bayes

```
[118]: from sklearn.naive_bayes import MultinomialNB

[119]: NB_classifier = MultinomialNB()

[120]: NB_classifier.fit(X_train,y_train)
t[120]: MultinomialNB()

[121]: NB_classifier.score(X_train,y_train)
t[121]: 0.6378122308354867
```

fig.16 Training model on MultinomialNB

Training accuracy: 63 %
Validation accuracy: 51 %

Logistic Regression.

Logistic Regression

```
In [127]:  1  from sklearn.linear_model import LogisticRegression

In [128]:  1  Lr = LogisticRegression()

In [129]:  1  Lr.fit(X_train,y_train)

Out[129]:  LogisticRegression()

In [245]:  1  Lr.score(X_train,y_train)

Out[245]:  0.6571202985931668
```

fig.17 Training model on Logistic regression

Training accuracy: 66%
Validation accuracy: 57%

- Key Metrics for success in solving problem under consideration

Accuracy score for Multinomial Classifier: 51 %

Classification report for Multinomial Classifier:

```
In [125]:  1  print(classification_report(y_test,predict_NB))

              precision    recall  f1-score   support

           1       0.57      0.71      0.64      1188
           2       0.54      0.36      0.43      1144
           3       0.44      0.35      0.39      1144
           4       0.42      0.47      0.45      1209
           5       0.56      0.64      0.60      1287

    accuracy                           0.51      5972
   macro avg       0.51      0.51      0.50      5972
weighted avg       0.51      0.51      0.50      5972
```

fig.18 Classification report for Multinomial Classifier

Confusion Matrix for Multinomial Classifier:

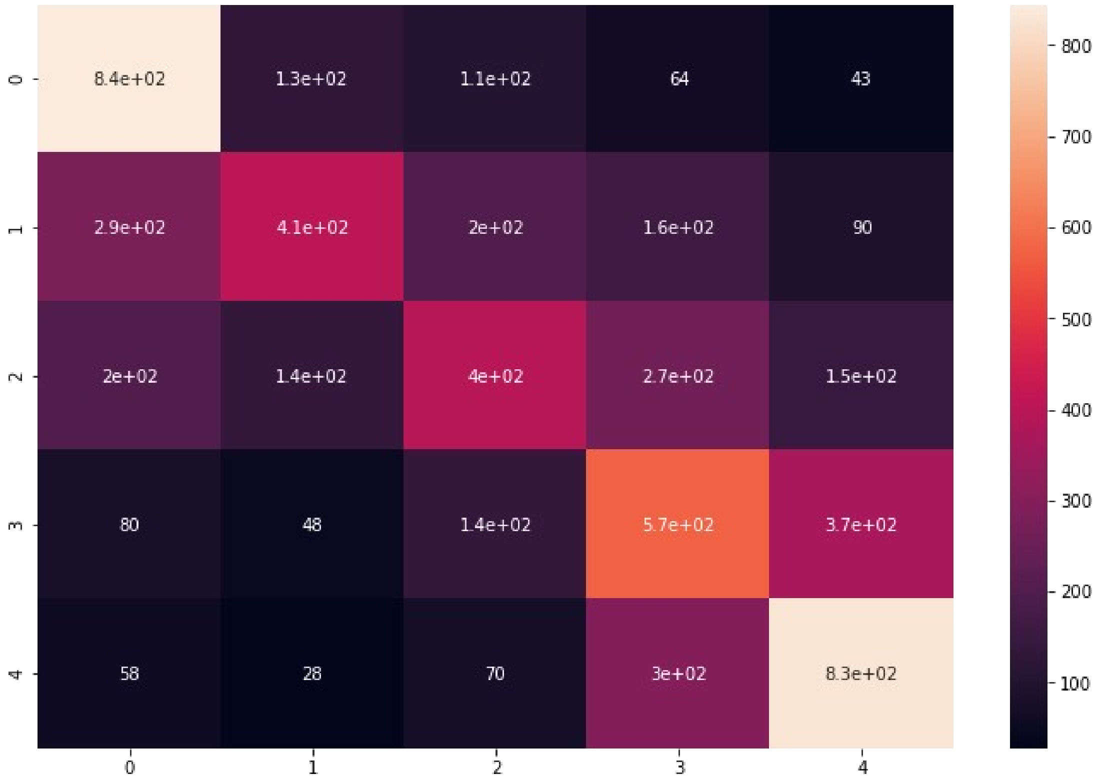

fig.19 Confusion matrix for Multinomial Classifier

Accuracy score for Logistic Regression: 57 %

Classification report for Logistic Regression:

```
In [133]:  1  print(classification_report(y_test,predict_LR))

              precision    recall  f1-score   support

           1       0.69      0.71      0.70      1188
           2       0.50      0.51      0.51      1144
           3       0.45      0.49      0.47      1144
           4       0.56      0.45      0.50      1209
           5       0.63      0.67      0.65      1287

    accuracy                           0.57      5972
   macro avg       0.57      0.57      0.56      5972
weighted avg       0.57      0.57      0.57      5972
```

fig.20 Classification Report for Logistic regression

Confusion Matrix for Logistic Regression:

fig.21 Confusion matrix for Logistic regression

Building an Artificial Neural network :

```
In [177]:
# Building the model
model = tf.keras.models.Sequential()

model.add(tf.keras.layers.Dense(units=500,activation='relu',input_shape = (14890, )))

model.add(tf.keras.layers.Dense(units=500,activation='relu'))

model.add(tf.keras.layers.Dense(units=500,activation='relu'))

model.add(tf.keras.layers.Dense(units=400,activation='relu'))

model.add(tf.keras.layers.Dense(units=200,activation='relu'))

model.add(tf.keras.layers.Dense(units=6,activation='softmax'))
```

fig.22 ANN

Model Summary:

```
In [178]:  model.summary()

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_6 (Dense)              (None, 500)               7445500
_________________________________________________________________
dense_7 (Dense)              (None, 500)               250500
_________________________________________________________________
dense_8 (Dense)              (None, 500)               250500
_________________________________________________________________
dense_9 (Dense)              (None, 400)               200400
_________________________________________________________________
dense_10 (Dense)             (None, 200)               80200
_________________________________________________________________
dense_11 (Dense)             (None, 6)                 1206
=================================================================
Total params: 8,228,306
Trainable params: 8,228,306
Non-trainable params: 0
```

fig.23 Model Summary

Compiling the model:

```
In [183]:  history = model.fit(X_train, y_train, batch_size= 64,epochs= 30,validation_split=0.2)

Epoch 1/30
175/175 [==============================] - 10s 58ms/step - loss: 1.2986 - accuracy: 0.4249 - val_los
s: 1.1426 - val_accuracy: 0.5235
Epoch 2/30
175/175 [==============================] - 10s 58ms/step - loss: 0.7780 - accuracy: 0.7007 - val_los
s: 1.1535 - val_accuracy: 0.5676
Epoch 3/30
175/175 [==============================] - 11s 61ms/step - loss: 0.3590 - accuracy: 0.8690 - val_los
s: 1.5206 - val_accuracy: 0.5590
Epoch 4/30
175/175 [==============================] - 11s 62ms/step - loss: 0.1895 - accuracy: 0.9317 - val_los
s: 1.6685 - val_accuracy: 0.5773
Epoch 5/30
175/175 [==============================] - 11s 63ms/step - loss: 0.1137 - accuracy: 0.9564 - val_los
s: 2.1848 - val_accuracy: 0.5741
Epoch 6/30
175/175 [==============================] - 11s 60ms/step - loss: 0.0903 - accuracy: 0.9631 - val_los
```

fig.24 Training Ann

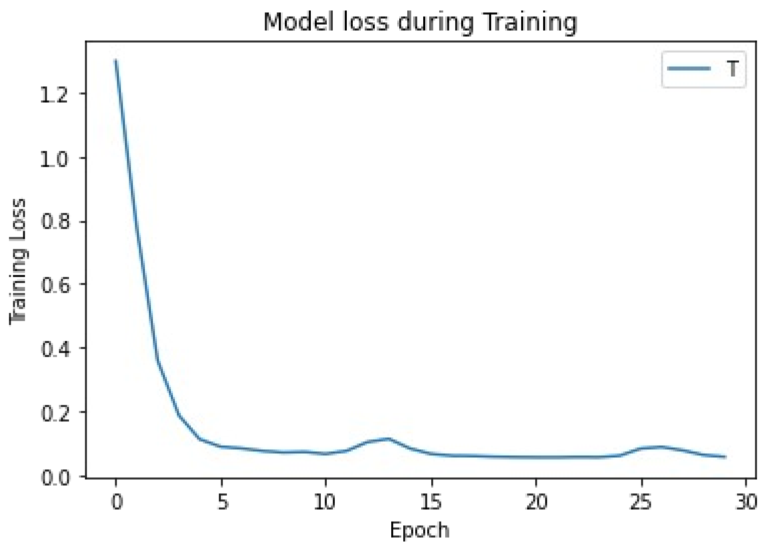

fig.25 Training loss vs Epoch

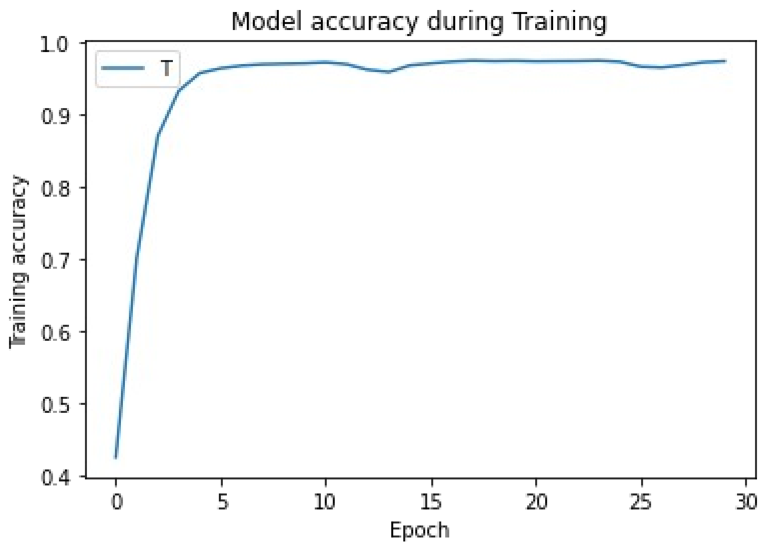

fig.26 Training accuracy vs Epoch

ANN model accuracy: 57 %

ANN model Classification report:

```
:   1  print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
1	0.70	0.68	0.69	1188
2	0.55	0.52	0.53	1144
3	0.49	0.46	0.47	1144
4	0.52	0.56	0.54	1209
5	0.60	0.63	0.62	1287
accuracy			0.57	5972
macro avg	0.57	0.57	0.57	5972
weighted avg	0.57	0.57	0.57	5972

```
:   1  print(accuracy_score(y_test,y_pred))
```

```
0.5713328868050904
```

fig.27 ANN accuracy and report

ANN model Confusion matrix:

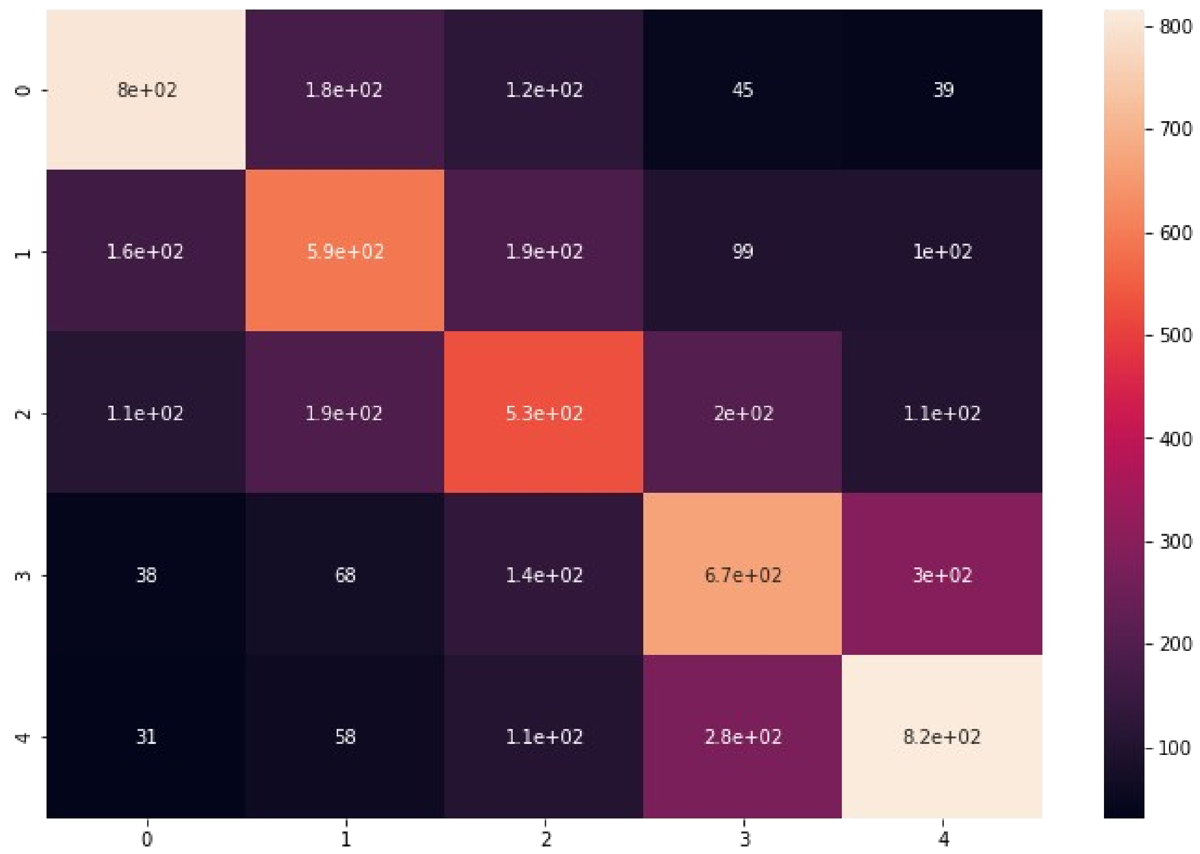

fig.28 ANN confusion matrix.

- Visualizations

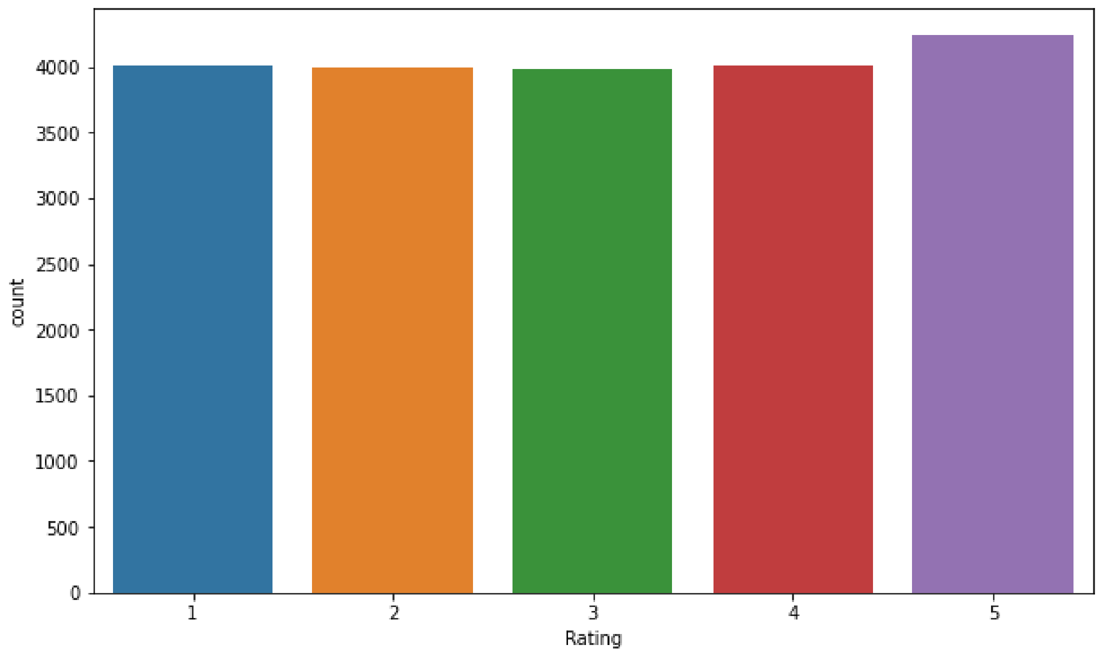

fig.29 Rating count

The fig.27 shows that the dependent variable Rating is balanced.

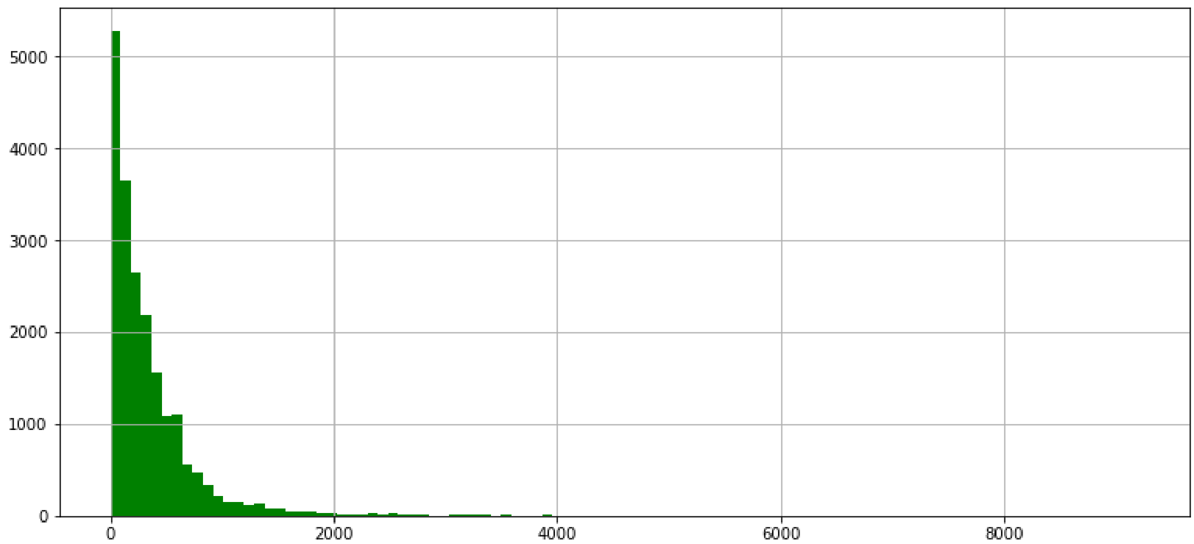

fig.30 review length

The fig.28 shows that Most of the reviews are very short but, there are also some very long reviews with word lengths up to 4000.

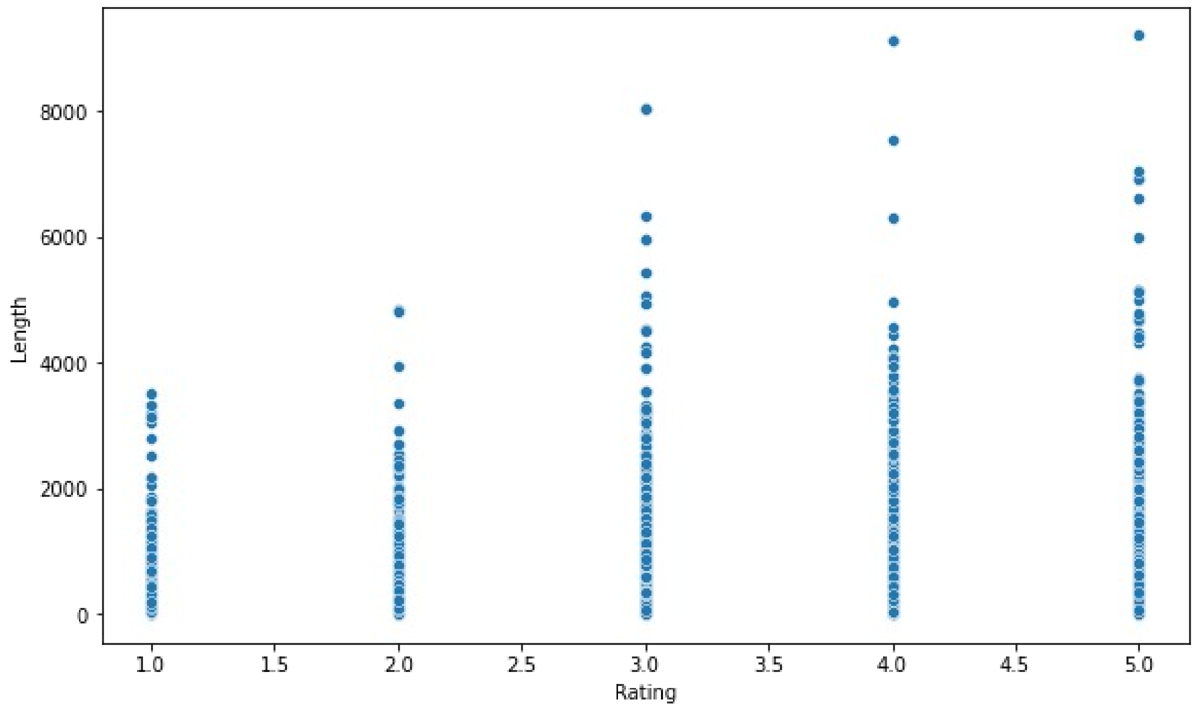

fig.31 rating vs length

From fig.29 plot we can clearly observe that, High review's are having longer message length.

fig.32 rating 5 star

From fig.30 plot we can clearly observe the word cloud for 5-star customer ratings and there are a lot of positive words in there.

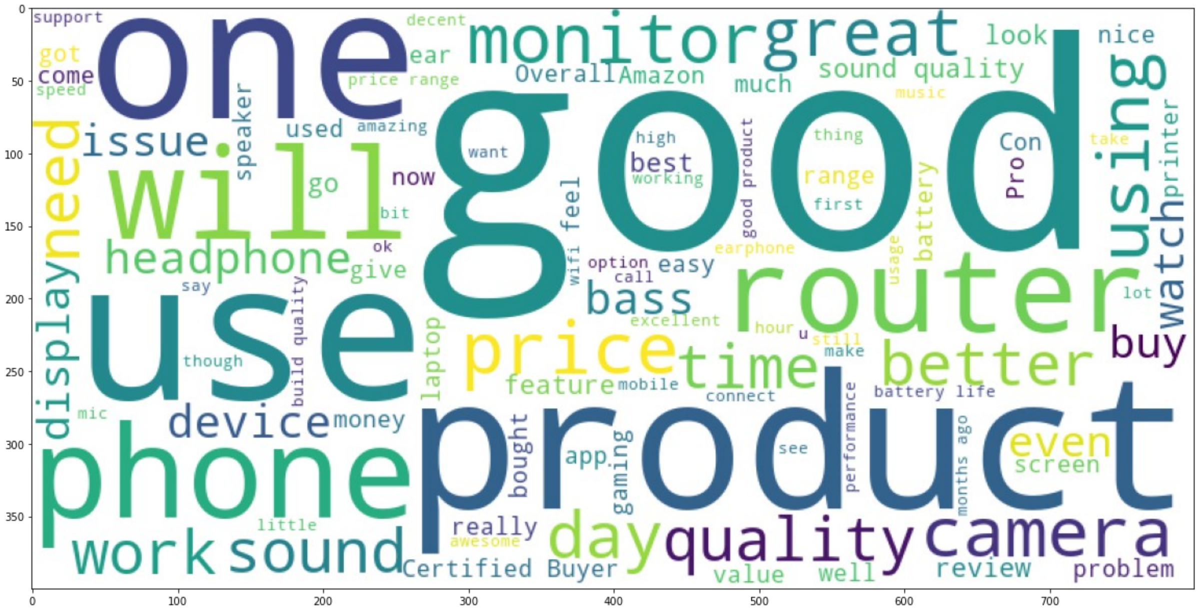

fig.33 rating 4 star

From fig.31 plot we can clearly observe the word cloud for 4 star ratings

fig.34 rating 3 star

From fig.32 plot we can clearly observe the word cloud for 3 star ratings.

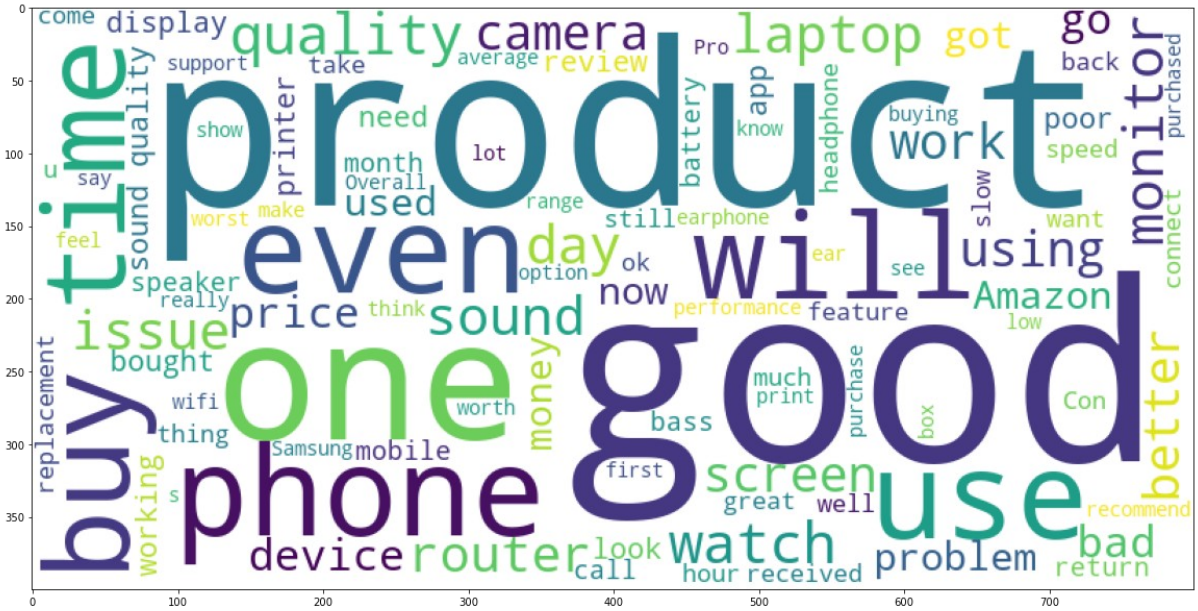

fig.35 rating 2 star

From fig.33 plot we can clearly observe the word cloud for 2 star ratings.

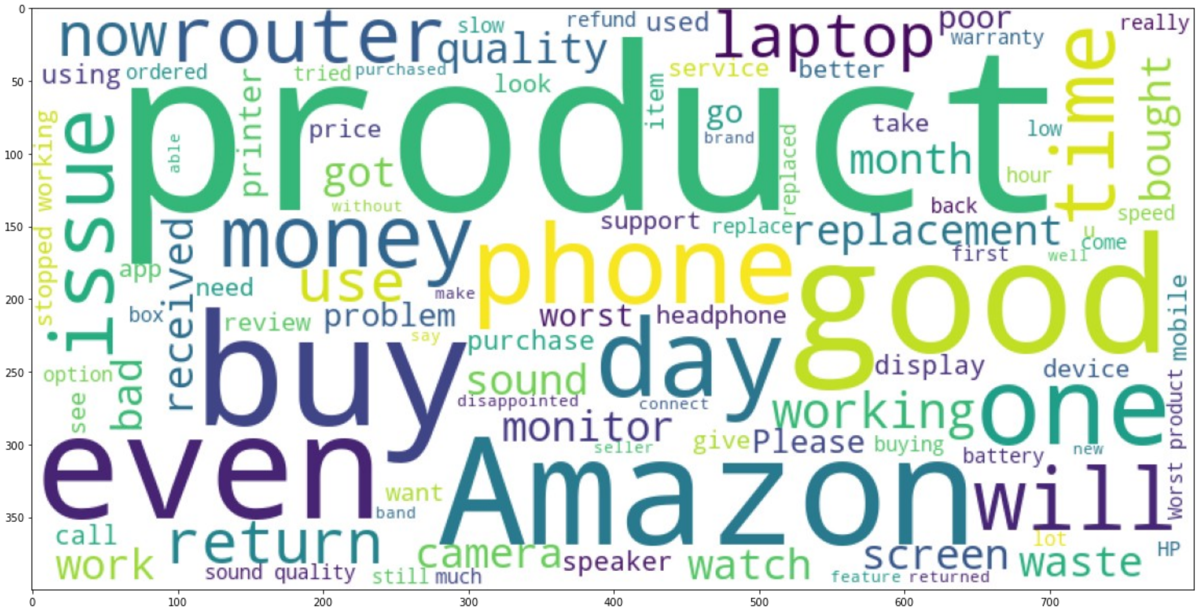

fig.36 rating 1 star

From fig.34 plot we can clearly observe the word cloud for 1 star ratings.

CONCLUSION

- Key Findings and Conclusions of the Study

  From the results obtained, we find that classifiers models are able to predict ratings of the reviews using various features derived from its textual content with a decent accuracy.

  This suggests a strong correlation between the customer reviews (in-text property) and ratings (out-of text property) we considered for our analysis.

  Hence, it answers the question: Sentiments of the text reviews do affect its corresponding ratings. It has been observed that among the features used for classification, polarity of the review and length of the review are more influential and highly correlated with its rating.

- Learning Outcomes of the Study in respect of Data Science
  - **Data Scrapping using Selenium web-driver**
  - **Building an Artificial Neural Network**
  - **Building a web app using Flask and uploading the web app on Heroku**

- Limitations of this work and Scope for Future Work

  Can improve the accuracy and will try to reduce the over-fitting.