FLIP ROBO

REVIEW RATING PREDICATION USING NLP

Submitted by: Priya Patidar
Internship Batch: 30
SME : Mohd. Kashif sir

ACKNOWLEDGMENT

I would like to express my sincere thanks and gratitude to my SME Mr. Mohd. Kashif Sir as well as the “FlipRobo Technologies” team for letting me work on the “Used Car Price Prediction” project. Their suggestions and directions have helped me in the completion of this project successfully. This project also helped me in doing lots of research wherein I came to know about so many new things.

Introduction Business

Problem Framing

Our customer has a website where users can post various product reviews for technical items. The reviewer will now be required to include stars (ratings) along with the review on their website, which is a new feature they are currently adding. There are only 5 alternatives available, and the ranking is out of 5. 1, 2, 3, 4, and 5 stars, respectively. They are attempting to forecast ratings for past reviews that have not yet received one. Therefore, we must create a programme that can gauge the rating from the review.

Conceptual Background of the Domain Problem

In general, shoppers utilize two straightforward heuristics to determine whether to make a final purchase of a product: ratings and pricing. The total star ratings of the product reviews, however, frequently do not accurately reflect the polarity of the opinions. Due of the possibility of varying customer ratings for a given review, rating prediction becomes a challenging topic. For instance, one person might give a product a 5-star rating and rate it as nice, but another user might write the same comment and only give it a 3-star rating. Additionally, reviews could include anecdotal data, which is not informative and makes forecasting more difficult.

Users may select different ways to express their sentiments. For instance, some users might use the word "good" to describe a merely passable product, while others might use it to describe a top-notch one. In addition to user bias, there is product bias. For example, the opinion word "long" can express a "positive" feeling for a cell phone's battery life but a "negative" feeling for a camera's focus time. We may use different opinion words to review different products, or even the same opinion word to express different sentiment polarities for different products. For the purpose of forecasting review ratings, it is crucial to take into account both the relationships between the review authors and the target products.

Review of Literature

According to the Lackermair, Kailer and Kanmaz (2013), product reviews and ratings represent an important source of information for consumers and are helpful tools in order to support their buying decisions [6]. They also found out that consumers are willing to compare both positive and negative reviews when searching for a specific product. The authors argue that customers need compact and concise information about the products. Therefore, consumers first need to pre-select the potential products matching their requirements. With this aim in mind, consumers use the star ratings as an indicator for selecting products. Later, when a limited number of potentials products have been chosen, reading the associated text review will reveal more details about the products and therefore help consumers making a final decision.

It becomes daunting and time-consuming to compare different products in order to eventually make a choice between them. Therefore, models able to predict the user rating from the text review are critically important (Baccianella, Esuli & Sebastiani, 2009) [7].

Pang, Lee and Vaithyanathan (2002) [9] approach this predictive task as an opinion mining problem enabling to automatically distinguish between positive and negative reviews. In order to determine the reviews polarity, the authors use text classification techniques by training and testing binary classifiers on movie reviews containing 36.6% of negative reviews and 63.4% of positive reviews. On the top of that, they also try to identify appropriate features to enhance the performance of the classifiers.

Dave, Lawrence, and Pennock (2003) [10] also deal with the issue of class imbalance with a majority of positive reviews and show similar results. SVM outperforms Naïve Bayes with an accuracy greater than 85% and the implementation of part-of-speech as well as stemming is also ineffective. However, this work demonstrates that bigrams turn out to be more successful at capturing context than unigrams in the specific situation of their datasets, despite earlier research having produced better results with unigrams.

to capture the weights of such characteristics by minimising the mean square error.

Motivation for the Problem Undertaken

My first project from Flip Robo Technologies under the internship programme was the project. The main drivers behind this were the chance to apply my skill set to a real-world problem and the exposure to data from the actual world.

The data needed for this project must be scraped from an e-commerce site and cleaned up. Its associated star ratings are predicted using features collected from textual evaluations. To do this, the prediction issue is turned into a task requiring multi-class classification, where reviews are assigned to one of five categories based on their star rating. Gaining a general understanding of a text review may enhance the user experience. However, the reason I decided to do this project was because it is relatively a new field of research.

Analytical Problem Framing

Mathematical / Analytical Modelling of the Problem

$$tf - idf_{t,d} = tf_{t,d} \cdot idf_t$$

where:

- $tf_{t,d} = \frac{n_{t,d}}{\sum_k n_{k,d}}$ with $n_{t,d}$ the number of term t contained in a document d, and $\sum_k n_{k,d}$ the total number of terms k in the document d
- $idf_t = \log \frac{N}{df_t}$ with N the total number of documents and df_t the number of documents containing the term t

In order to apply text classification, the unstructured format of text has to be converted into a structured format for the simple reason that it is much easier for computer to deal with umbers than text. This is mainly achieved by projecting the textual contents into Vector Space Model, where text data is converted into vectors of numbers.

Documents are frequently handled like a Bag-of-Words (BoW) in the field of text categorization, which means that each word is distinct from the other words that are present in the text. They are scrutinized without consideration for grammar or word order. In this model, the classifier is trained using the term-frequency (the frequency with which each word occurs) as a feature. However, the use of the word frequency suggests that all concepts are given equal weight. The word frequency, as its name implies, does nothing more than weight each term according to how frequently it occurs; it does not take the discriminatory potential of terms into consideration. Each word is given a term frequency inverse document frequency in order to handle this issue and penalise words that are used excessively (tf-idf) score which is defined above:

Data Sources and their formats

Data is collected from Amazon.using selenium and saved in CSV file. Around 46866 Reviews are collected for this project.

```
In [21]: print('No. of Rows :',data.shape[0])
         print('No. of Columns :',data.shape[1])
         pd.set_option('display.max_columns',None) # # This will enable us to see truncated columns
         data.head()

No. of Rows : 46866
No. of Columns : 2
```

Out[21]:

	Rattings	Review
0	1	Please don't purchase any Lenovo product.4 mon...
1	1	Please don't purchase any Lenovo product.4 mon...
2	1	Please don't purchase any Lenovo product.4 mon...
3	2	I usually research a lot before buying a lapto...
4	5	I am a software engineer, working with Reactjs...

This is multi-classification problem and Rating is our target feature class to be predicated in this project. There are five different categories in feature target i.e., The rating is out 5 stars and it only has 5 options available 1 star, 2 stars, 3 stars, 4 stars, 5 stars.

```
In [22]: data.info() #Checking the datatype of all the columns present

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 46866 entries, 0 to 46865
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Rattings  46866 non-null  int64
 1   Review    46863 non-null  object
dtypes: int64(1), object(1)
memory usage: 732.4+ KB
```

There are some missing values in product review. The datatype of Product review is object while datatypes of Ratings is int.

Data Pre-processing

The dataset is large and it may contain some data error. In order to reach clean, error free data some data cleaning & data pre-processing performed data.

Missing Value Imputation:

Missing value in product reviews are replace with 'Review Not Available'.

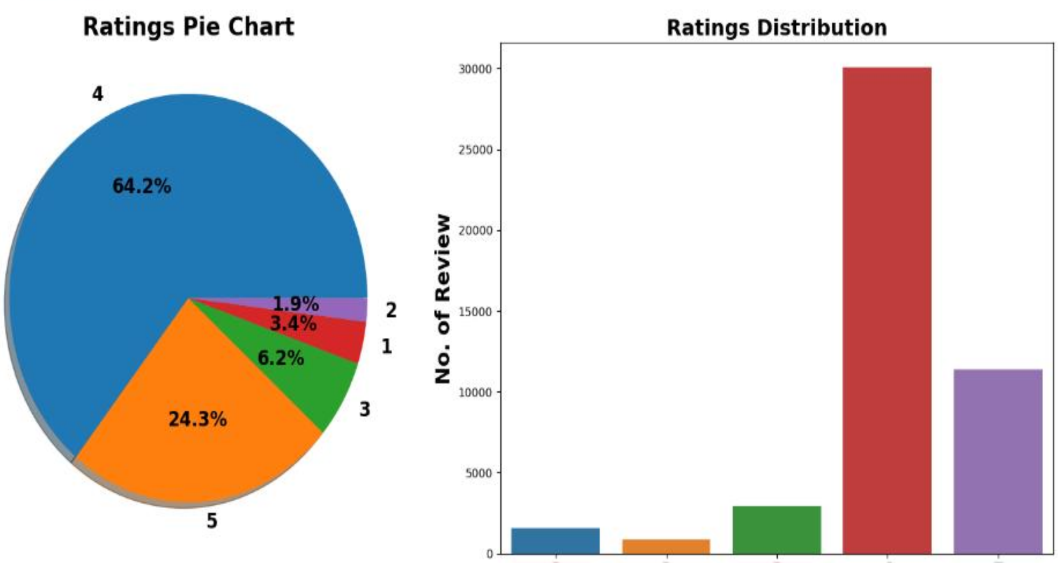
In [23]:

```
plt.figure(figsize=(15,7))
sns.heatmap(data.isnull(),cmap='Pastel2')
plt.show()
data.isnull().sum()
```

We will replace missing value in Review with 'Review Not Available'

In [25]:

```
# Pie & Count plot of Ratings
plt.rcParams["figure.autolayout"] = True
f,ax=plt.subplots(1,2,figsize=(15,7))
data['Rattings'].value_counts().plot.pie(autopct='%2.1f%%',
                                textprops ={'fontsize':16 ,'fontweight' :'bold'}, ax=ax[0],shadow=True)
ax[0].set_title('Ratings Pie Chart', fontsize=20,fontweight ='bold')
ax[0].set_ylabel('')
sns.countplot('Rattings',data=data,ax=ax[1])
ax[1].set_title('Ratings Distribution',fontsize=18,fontweight ='bold')
ax[1].set_xlabel("Ratings",fontsize=18,fontweight ='bold')
ax[1].set_ylabel("No. of Review",fontsize=18,fontweight ='bold')
plt.xticks(fontsize=18,fontweight ='bold')
plt.tight_layout()
plt.show()
print('Value Counts of Rating:')
data['Rattings'].value_counts()
```



Data is pre-processed using the following techniques:

Convert the text to lowercase
Remove the punctuations, digits and special characters
Tokenize the text, filter out the adjectives used in the review and create a new column in data frame
Remove the stop words
Stemming and
Lemmatising
Applying Text Vectorization to convert text into numeric

Removing Stop Words

Stemming and Lemmatising

Applying Count Vectoriser

```
In [27]:
#Importing required libraries
import re
import string
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import SnowballStemmer, WordNetLemmatizer
from wordcloud import WordCloud
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\pranay\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\pranay\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\pranay\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

```
In [28]:
from sklearn.feature_extraction.text import TfidfVectorizer,CountVectorizer
```

Applying Regular expression for text extraction.

```
In [29]:
def clean_text(data, data_column_name):

    #Converting all messages to lowercase
    data[data_column_name] = data[data_column_name].str.lower()

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

In [30]:
```python
#Calling the class
clean_text(data, 'Review')
data['Review'].tail(3)
```

Out[30]:
```
46863    originally using deco e numbr route fully sati...
46864    hialways problem getting strong enough wifi ro...
46865     price negative rest product reliable hassle free
Name: Review, dtype: object
```

Data Tokenization using RegexpTokenizer

In [31]:
```python
#Tokenizing the data using RegexpTokenizer
from nltk.tokenize import RegexpTokenizer
tokenizer=RegexpTokenizer(r'\w+')
data['Review'] = data['Review'].apply(lambda x: tokenizer.tokenize(x.lower()))
data.head()
```

Out[31]:

	Ratings	Review
0	1	[please, purchase, lenovo, product, numbr, mon...
1	1	[please, purchase, lenovo, product, numbr, mon...
2	1	[please, purchase, lenovo, product, numbr, mon...
3	2	[usually, research, lot, buying, laptop, hesit...
4	5	[software, engineer, working, reactjs, bought,...

Stemming & Lemmatization

In [32]:
```python
# Lemmatizing and then Stemming with Snowball to get root words and further reducing characters
stemmer = SnowballStemmer("english")
import gensim
def lemmatize_stemming(text):
    return stemmer.stem(WordNetLemmatizer().lemmatize(text,pos='v'))
```

In [33]:
```python
#Tokenize and Lemmatize
def preprocess(text):
    result=[]
    for token in text:
        if len(token)>=3:
            result.append(lemmatize_stemming(token))
    return result
```

In [34]:
```python
#Processing review with above Function
processed_review = []

for doc in data.Review:
    processed_review.append(preprocess(doc))

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

```
46866
```

Out[34]:
```
[['pleas',
  'purchas',
  'lenovo',
  'product',
  'numbr',
  'month',
  'ago',
  'purchas',
  'lenovo',
  'ideapad',
  'flex',
  'display',
  'issu',
  'complaint',
  'issu',
```

Data Inputs- Logic- Output Relationships

The dataset consists of 2 features with a label. The features are independent and label is dependent as our label varies the values (text) of our independent variable's changes. Using word cloud, we can see most occurring word for different categories.

Hardware & Software Requirements with Tool Used

Hardware Used -

- Processor — Intel i3 processor with 2.4GHZ
- RAM— 4GB
- GPU — 2GB AMD Radeon Graphics card Software utilised -
- Anaconda – Jupyter Notebook
- Selenium – Web scraping
- Google Colab – for Hyper parameter tuning
- Libraries Used – General library for data wrangling & visualsation

Models Development & Evaluation

Identification Of Possible Problem-Solving Approaches (Methods)

First part of problem solving is to scrap data from amazon which we already done. Second is performing text mining operation to convert textual review in ML algorithm useable form. Third part of problem building machine learning model to predict rating on review. This problem can be solve using classification-based machine learning algorithm like logistics regression. Further Hyperparameter tuning performed to build more accurate model out of best model.

Testing of Identified Approaches (Algorithms)

The different classification algorithm used in this project to build ML model are as below:

Random Forest classifier
Decision Tree classifier
Logistics Regression
AdaBoost Classifier
Gradient Boosting
Classifier

Key Metrics for Success in Solving Problem Under Consideration

Precision can be seen as a measure of quality; higher precision means that an algorithm returns more relevant results than irrelevant ones.
Recall is used as a measure of quantity and high recall means that an algorithm returns most of the relevant results.

Accuracy score is used when the True Positives and True negatives are more important. Accuracy can be used when the class distribution is similar. F1-score is used when the False Negatives and False Positives are crucial. While F1-score is a better metric when there are imbalanced classes.

Cross validation Score: To run cross-validation on multiple metrics and also to return train scores, fit times and score times. Get predictions from each split of cross-validation for diagnostic purposes. Make a scorer from a performance metric or loss function.

Run And Evaluate Selected Models

- Logistics Regression

```
In [55]:
# Logistics Regression
# Creating train_test_split using best random_state
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=55, test_size=.3)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=55, test_size=.3)
log_reg=LogisticRegression()
log_reg.fit(X_train,Y_train)
y_pred=log_reg.predict(X_test)
print('\033[1m'+'Logistics Regression Evaluation'+'\033[0m')
print('\n')
print('\033[1m'+'Accuracy Score of Logistics Regression :'+'\033[0m', accuracy_score(Y_test, y_pred))
print('\n')
print('\033[1m'+'Confusion matrix of Logistics Regression :'+'\033[0m \n',confusion_matrix(Y_test, y_pred))
print('\n')
print('\033[1m'+'classification Report of Logistics Regression'+'\033[0m \n',classification_report(Y_test, y_pred))
```

```
Logistics Regression Evaluation


Accuracy Score of Logistics Regression : 0.9440256045519203


Confusion matrix of Logistics Regression :
 [[ 429    1    2   24    5]
 [   0  227    1   36    3]
 [   0    0  755  117   15]
 [   3    0    4 8973   85]
 [   0    0    1  490 2889]]


classification Report of Logistics Regression
               precision    recall  f1-score   support

           1       0.99      0.93      0.96       461
           2       1.00      0.85      0.92       267
           3       0.99      0.85      0.92       887
           4       0.93      0.99      0.96      9065
           5       0.96      0.85      0.91      3380

    accuracy                           0.94     14060
   macro avg       0.97      0.90      0.93     14060
weighted avg       0.95      0.94      0.94     14060
```

Train-test split is used to split data into training data & testing data. Further best random state is investigated through loop.

```
In [44]:
# Finding best Random state
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix,classification_report,f1_score
maxAccu=0
maxRS=0
for i in range(50,100):
    X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.3, random_state=i)
    log_reg=LogisticRegression()
    log_reg.fit(X_train,Y_train)
    y_pred=log_reg.predict(X_test)
    acc=accuracy_score(Y_test,y_pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print('Best accuracy is', maxAccu ,'on Random_state', maxRS)
```

```
Best accuracy is 0.9440256045519203 on Random_state 55
```

- Decision Tree Classifier

Decision Tree Classifier model is built and evaluation matrix is shown as below:

Decision Tree Classifier

```
In [57]:  dt=DecisionTreeClassifier()
          dt.fit(X_train,Y_train)
          y_pred=dt.predict(X_test)
          print('\033[1m'+'Decision Tree Classifier Evaluation'+'\033[0m')
          print('\n')
          print('\033[1m'+'Accuracy Score of Decision Tree Classifier :'+'\033[0m', accuracy_score(Y_test, y_pred))
          print('\n')
          print('\033[1m'+'Confusion matrix of Decision Tree Classifier :'+'\033[0m \n',confusion_matrix(Y_test, y_pred))
          print('\n')
          print('\033[1m'+'classification Report of Decision Tree Classifier'+'\033[0m \n',classification_report(Y_test, y_pred))
```

```
Decision Tree Classifier Evaluation


Accuracy Score of Decision Tree Classifier : 0.9692745376955904


Confusion matrix of Decision Tree Classifier :
 [[ 448    0    3    8    2]
 [    1  256    1    7    2]
 [    1    1  844   38    3]
 [    6    2    4 9020   33]
 [    1    1    3  315 3060]]


classification Report of Decision Tree Classifier
               precision    recall  f1-score   support

           1       0.98      0.97      0.98       461
           2       0.98      0.96      0.97       267
           3       0.99      0.95      0.97       887
           4       0.96      1.00      0.98      9065
           5       0.99      0.91      0.94      3380

    accuracy                           0.97     14060
   macro avg       0.98      0.96      0.97     14060
weighted avg       0.97      0.97      0.97     14060
```

- Random Forest Classifier

Random Forest Classifier

```
In [49]:  rf=RandomForestClassifier()
          rf.fit(X_train,Y_train)
          y_pred=rf.predict(X_test)
          print('\033[1m'+'Random Forest Classifier'+'\033[0m')
          print('\n')
          print('\033[1m'+'Accuracy Score of Random Forest Classifier :'+'\033[0m', accuracy_score(Y_test, y_pred))
          print('\n')
          print('\033[1m'+'Confusion matrix of Random Forest Classifier :'+'\033[0m \n',confusion_matrix(Y_test, y_pred))
          print('\n')
          print('\033[1m'+'classification Report of Random Forest Classifier'+'\033[0m \n',classification_report(Y_test, y_pred))
```

```
Random Forest Classifier


Accuracy Score of Random Forest Classifier : 0.966145092460882


Confusion matrix of Random Forest Classifier :
 [[ 447    1    0   15    1]
 [    0  259    0    6    0]
 [    0    0  782   68    4]
 [    4    0    6 9023   18]
 [    0    0    0  353 3073]]


classification Report of Random Forest Classifier
               precision    recall  f1-score   support

           1       0.99      0.96      0.98       464
           2       1.00      0.98      0.99       265
           3       0.99      0.92      0.95       854
           4       0.95      1.00      0.97      9051
           5       0.99      0.90      0.94      3426

    accuracy                           0.97     14060
   macro avg       0.99      0.95      0.97     14060
weighted avg       0.97      0.97      0.97     14060
```

- Ada Boost Classifier

AdaBoost Classifier

In [60]:
```
ad=AdaBoostClassifier()
ad.fit(X_train,Y_train)
y_pred=ad.predict(X_test)
print('\033[1m'+'AdaBoost Classifier Evaluation'+'\033[0m')
print('\n')
print('\033[1m'+'Accuracy Score of AdaBoost Classifier :'+'\033[0m', accuracy_score(Y_test, y_pred))
print('\n')
print('\033[1m'+'Confusion matrix of AdaBoost Classifier :'+'\033[0m \n',confusion_matrix(Y_test, y_pred))
print('\n')
print('\033[1m'+'classification Report of AdaBoost Classifier'+'\033[0m \n',classification_report(Y_test, y_pred))
```

```
AdaBoost Classifier Evaluation


Accuracy Score of AdaBoost Classifier : 0.6862731152204836


Confusion matrix of AdaBoost Classifier :
 [[ 191    0    9  237   24]
 [    2  109    4  113   39]
 [    9    0   21  835   22]
 [   15    8   52 8776  214]
 [    8    0   11 2809  552]]


classification Report of AdaBoost Classifier
               precision    recall  f1-score   support

           1       0.85      0.41      0.56       461
           2       0.93      0.41      0.57       267
           3       0.22      0.02      0.04       887
           4       0.69      0.97      0.80      9065
           5       0.65      0.16      0.26      3380

    accuracy                           0.69     14060
   macro avg       0.67      0.40      0.45     14060
weighted avg       0.66      0.69      0.61     14060
```

- Gradient Boosting Classifier

Gradient Boosting Classifier

In [65]:
```
gbc=GradientBoostingClassifier()
gbc.fit(X_train,Y_train)
y_pred=gbc.predict(X_test)
print('\033[1m'+'Gradient Boosting Classifier Evaluation'+'\033[0m')
print('\n')
print('\033[1m'+'Accuracy Score of Gradient Boosting Classifier :'+'\033[0m', accuracy_score(Y_test, y_pred))
print('\n')
print('\033[1m'+'Confusion matrix of Gradient Boosting Classifier :'+'\033[0m \n',confusion_matrix(Y_test, y_pred))
print('\n')
print('\033[1m'+'classification Report of Gradient Boosting Classifier'+'\033[0m \n',classification_report(Y_test, y_pred))
```

```
Gradient Boosting Classifier Evaluation


Accuracy Score of Gradient Boosting Classifier : 0.9151493598862019


Confusion matrix of Gradient Boosting Classifier :
 [[ 415    0    0   44    2]
 [   0  251    0   14    2]
 [   0    0  610  258   19]
 [   3    2    0 9023   37]
 [   0    0    1  811 2568]]


classification Report of Gradient Boosting Classifier
              precision    recall  f1-score   support

           1       0.99      0.90      0.94       461
           2       0.99      0.94      0.97       267
           3       1.00      0.69      0.81       887
           4       0.89      1.00      0.94      9065
           5       0.98      0.76      0.85      3380

    accuracy                           0.92     14060
   macro avg       0.97      0.86      0.90     14060
weighted avg       0.92      0.92      0.91     14060
```

5-fold Cross validation performed over all model. We can see that Random Forest Classifier gives us good Accuracy and maximum f1 score along with best Cross-validation score. Hyperparameter tuning is applied over Random Forest model and used it as final model.

Hyper Parameter Tuning : GridSearchCV

In [63]:
```python
from sklearn.model_selection import GridSearchCV
parameter = { 'max_features': ['auto', 'log2'],
             'criterion':['gini','entropy'],
             'n_estimators': [75,100,150]}
GCV = GridSearchCV(RandomForestClassifier(),parameter,verbose=10)
GCV.fit(X_train,Y_train)
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
[CV 1/5; 1/12] START criterion=gini, max_features=auto, n_estimators=75.........
[CV 1/5; 1/12] END criterion=gini, max_features=auto, n_estimators=75;, score=0.968 total time=  25.6s
[CV 2/5; 1/12] START criterion=gini, max_features=auto, n_estimators=75.........
[CV 2/5; 1/12] END criterion=gini, max_features=auto, n_estimators=75;, score=0.967 total time=  26.4s
[CV 3/5; 1/12] START criterion=gini, max_features=auto, n_estimators=75.........
[CV 3/5; 1/12] END criterion=gini, max_features=auto, n_estimators=75;, score=0.964 total time=  27.6s
[CV 4/5; 1/12] START criterion=gini, max_features=auto, n_estimators=75.........
[CV 4/5; 1/12] END criterion=gini, max_features=auto, n_estimators=75;, score=0.968 total time=  26.3s
[CV 5/5; 1/12] START criterion=gini, max_features=auto, n_estimators=75.........
[CV 5/5; 1/12] END criterion=gini, max_features=auto, n_estimators=75;, score=0.971 total time=  25.8s
[CV 1/5; 2/12] START criterion=gini, max_features=auto, n_estimators=100........
[CV 1/5; 2/12] END criterion=gini, max_features=auto, n_estimators=100;, score=0.968 total time=  35.2s
[CV 2/5; 2/12] START criterion=gini, max_features=auto, n_estimators=100........
[CV 2/5; 2/12] END criterion=gini, max_features=auto, n_estimators=100;, score=0.966 total time=  35.4s
[CV 3/5; 2/12] START criterion=gini, max_features=auto, n_estimators=100........
[CV 3/5; 2/12] END criterion=gini, max_features=auto, n_estimators=100;, score=0.965 total time=  37.9s
[CV 4/5; 2/12] START criterion=gini, max_features=auto, n_estimators=100........
[CV 4/5; 2/12] END criterion=gini, max_features=auto, n_estimators=100;, score=0.968 total time=  35.7s
[CV 5/5; 2/12] START criterion=gini, max_features=auto, n_estimators=100........
[CV 5/5; 2/12] END criterion=gini, max_features=auto, n_estimators=100;, score=0.971 total time=  35.6s
[CV 1/5; 3/12] START criterion=gini, max_features=auto, n_estimators=150........
[CV 1/5; 3/12] END criterion=gini, max_features=auto, n_estimators=150;, score=0.968 total time=  55.2s
[CV 2/5; 3/12] START criterion=gini, max_features=auto, n_estimators=150........
[CV 2/5; 3/12] END criterion=gini, max_features=auto, n_estimators=150;, score=0.966 total time=  58.3s
[CV 3/5; 3/12] START criterion=gini, max_features=auto, n_estimators=150........
[CV 3/5; 3/12] END criterion=gini, max_features=auto, n_estimators=150;, score=0.965 total time=  54.5s
[CV 4/5; 3/12] START criterion=gini, max_features=auto, n_estimators=150........
[CV 4/5; 3/12] END criterion=gini, max_features=auto, n_estimators=150;, score=0.968 total time= 1.0min
[CV 5/5; 3/12] START criterion=gini, max_features=auto, n_estimators=150........
[CV 5/5; 3/12] END criterion=gini, max_features=auto, n_estimators=150;, score=0.970 total time=  54.9s
[CV 1/5; 4/12] START criterion=gini, max_features=log2, n_estimators=75.........
[CV 1/5; 4/12] END criterion=gini, max_features=log2, n_estimators=75;, score=0.968 total time=  27.2s
```

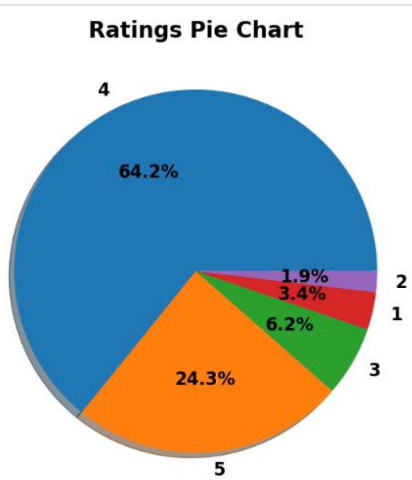
Final model is built using best parameter in hyper parameters tuning. The corresponding evaluation matrix shown below:

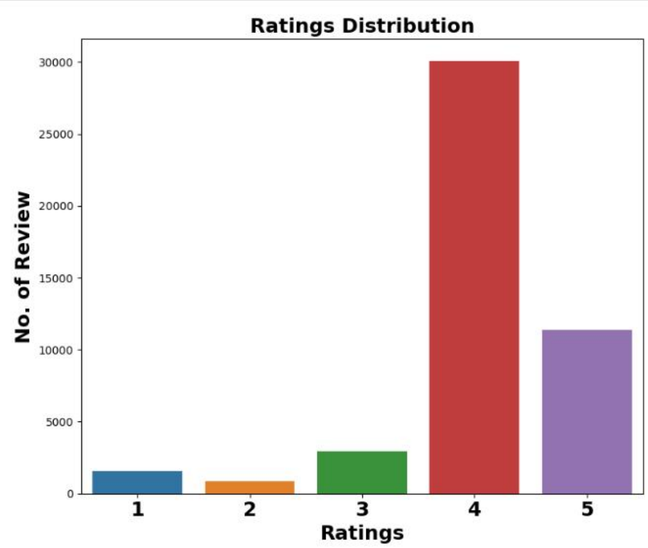
Final Model

```
In [66]:  Final_mod = RandomForestClassifier(criterion='gini',n_estimators= 150,max_features='log2')
          Final_mod.fit(X_train,Y_train)
          y_pred=Final_mod.predict(X_test)
          print('\033[1m'+'Final Random Forest Classifier Model'+'\033[0m')
          print('\033[1m'+'Accuracy Score :'+'\033[0m\n', accuracy_score(Y_test, y_pred))
          print('\n')
          print('\033[1m'+'Confusion matrix of Random Forest Classifier :'+'\033[0m \n',confusion_matrix(Y_test, y_pred))
          print('\n')
          print('\033[1m'+'Classification Report of Random Forest Classifier'+'\033[0m \n',classification_report(Y_test, y_pred))
```

```
Final Random Forest Classifier Model
Accuracy Score :
 0.9699857752489331


Confusion matrix of Random Forest Classifier :
 [[ 443    0    0   18    0]
 [   0  256    0   11    0]
 [   0    0  842   45    0]
 [   3    0    2 9051    9]
 [   0    0    0  334 3046]]


Classification Report of Random Forest Classifier
               precision    recall  f1-score   support

           1       0.99      0.96      0.98       461
           2       1.00      0.96      0.98       267
           3       1.00      0.95      0.97       887
           4       0.96      1.00      0.98      9065
           5       1.00      0.90      0.95      3380

    accuracy                           0.97     14060
   macro avg       0.99      0.95      0.97     14060
weighted avg       0.97      0.97      0.97     14060
```

Visualizations



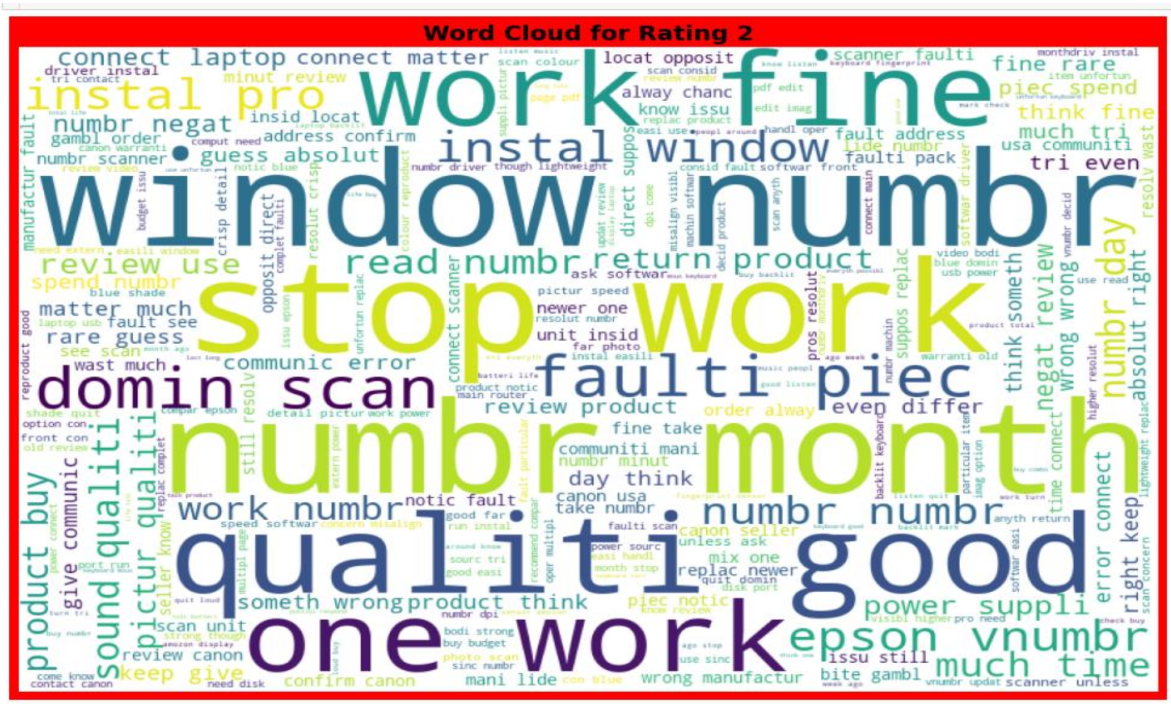


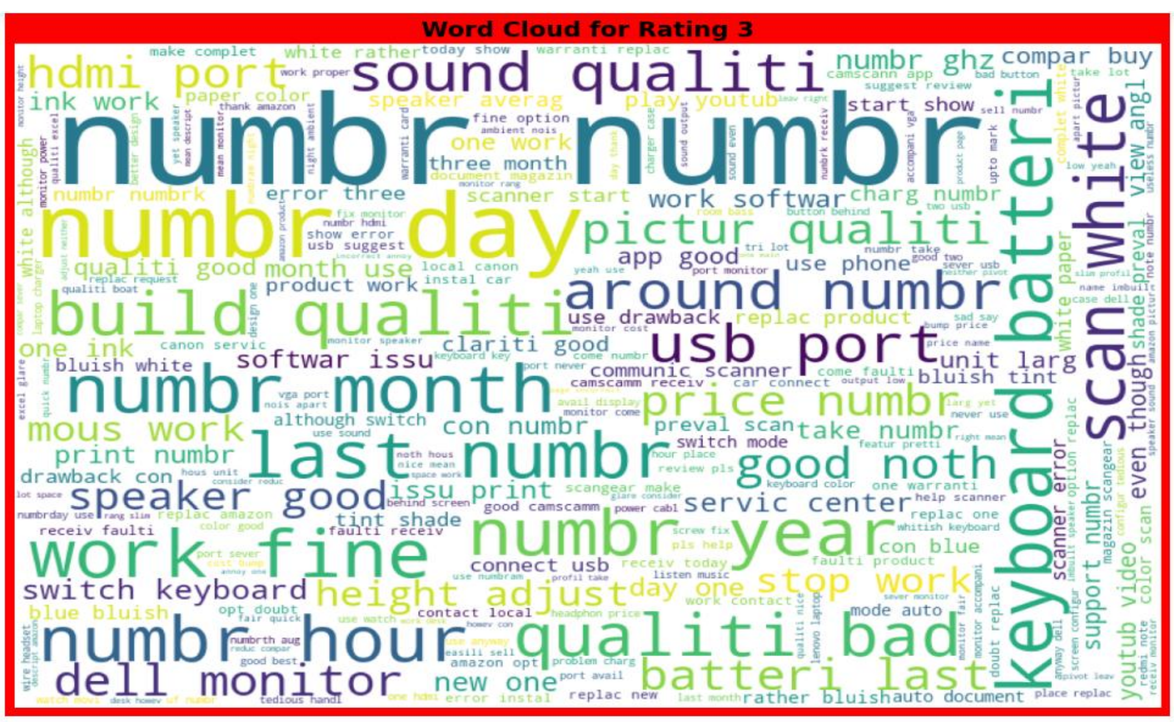
Comment:

1. Around 64% customer given 4- star rating followed by 24% customer given 5-star rating.
2. Average Rating is 4.04

Word Cloud:

Word Cloud is a visualization technique for text data wherein each word is picturized with its importance in the context or its frequency.
The more commonly the term appears within the text being analysed, the larger the word appears in the image generated.The enlarged texts are the greatest number of words used here and small texts are the smaller number of words used






Word Cloud for Rating 4
numbr numbr
camera qualiti
batteri life
nice product
good product
product good
qualiti good
price rang
valu money
numbr day
good batteri
phone good
overal good
good phone
build qualiti
batteri backup
numbr month
numbr camera
sound qualiti
finger print
good nice
good good
pretti good
watch face
home use
nice good
work fine
good printer
print qualiti
camera good
good valu
product nice
look good
numbr year
littl bite
around numbr
use phone
fast charg
realli good
numbr star
phone price
refresh rate
android numbr
good numbr
perform good
front camera
pictur qualiti
heat issu
good mobil
call featur
easi use
numbr hour
work well
even though
batteri drain
numbr batteri

Word Cloud for Rating 5
numbr numbr
print qualiti
photo print
usb cabl
numbr year
numbr month
sound qualiti
batteri life
qualiti good
good product
pictur qualiti
paper tray
product good
easi use
easi setup
black white
work good
use numbr
wifi numbr
price rang
valu money
around numbr
numbr dpi
numbr ghz
build qualiti
excel product
connect scanner
copi scan
numbr minut
first instal
instal driver
canon lide
instal softwar
usb port
nice product
work well
numbr day
rat numbr
within numbr
scan numbr
numbr second
almost numbr
numbr hour
qualiti product
good qualiti
laptop desktop
numbr paper
work smooth
wifi work
printer featur
heavi duti
ink bottl
use printer
realli good
absolut perfect
satisfi product
great product
super fast
one best
lide numbr
need instal

Conclusion

Key Findings and Conclusion of the Study

Final Model

```
In [66]:  Final_mod = RandomForestClassifier(criterion='gini',n_estimators= 150,max_features='log2')
          Final_mod.fit(X_train,Y_train)
          y_pred=Final_mod.predict(X_test)
          print('\033[1m'+'Final Random Forest Classifier Model'+'\033[0m')
          print('\033[1m'+'Accuracy Score :'+'\033[0m\n', accuracy_score(Y_test, y_pred))
          print('\n')
          print('\033[1m'+'Confusion matrix of Random Forest Classifier :'+'\033[0m \n',confusion_matrix(Y_test, y_pred))
          print('\n')
          print('\033[1m'+'Classification Report of Random Forest Classifier'+'\033[0m \n',classification_report(Y_test, y_pred))

Final Random Forest Classifier Model
Accuracy Score :
 0.9699857752489331
```

Final Model is giving us Accuracy score of 96.99%

Learning Outcomes of Data Science

Hands on chance to enhance my web scraping skillset.
In this project we were able to learn various Natural language processing techniques like lemmatization, stemming, removal of Stop words.
This project has demonstrated the importance of sampling effectively, modelling and predicting data.

Limitations of this work and Scope for the Future

More input features can be scrap to build predication model.
There is scope for application of advanced deep learning NLP tool to enhanced text mining operation which eventually help in building more accurate model with good cross validation score.

References use in this project:

- SCIKIT Learn Library Documentation
- Blogs from towards datascience, Analytics Vidya, Medium
- Andrew Ng Notes on Machine Learning (GitHub)
- Data Science Projects with Python Second Edition by Packt
- Hands on Machine learning with scikit learn and tensor flow by Aurelien Geron
- Lackermair, G., Kailer, D. & Kanmaz, K. (2013). Importance of online product reviews from a consumer's perspective. Horizon Research Publishing, 1-5. doi:
- 10.13189/aeb.2013.010101
- Baccianella, S., Esuli, A. & Sebastiani, F. (2009). Multi-facet rating of product reviews. Proceedings of the 31st European Conference on Information Retrieval (ECIR), 461- 472