“A PROJECT REPORT ON REVIEW RATING PREDICTION’

SUBMITTED BY

HIMAJA IJJADA

Caution - Please view the presentation as a slide show for better experience

TOPICS TO BE DISCUSSED

- Introduction
- Problem statement
- Conceptual Background of the Domain Problem
- Motivation for the Problem Undertaken
- Mathematical/ Analytical Modeling of the Problem
- Data Sources and their formats
- Data Preprocessing Done
- Software Used
- Visualizations
- Model/s Development and evaluation
- Interpretation of the Results
- Key Findings and Conclusions of the Study

Introduction

- Rating prediction is a well-known recommendation task aiming to predict a user's rating for those items which were not rated yet by her. Predictions are computed from users' explicit feedback, i.e. their ratings provided on some items in the past. Another type of feedback are user reviews provided on items which implicitly express users' opinions on items. Recent studies indicate that opinions inferred from users' reviews on items are strong predictors of user's implicit feedback or even ratings and thus, should be utilized in computation.

- 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!. There are two main methods to approach this problem. The first one is based on review text content analysis and uses the principles of natural language process (the NLP method). This method lacks the insights that can be drawn from the relationship between costumers and items. The second one is based on recommender systems, specifically on collaborative filtering, and focuses on the reviewer's point of view.

Problem statement

- 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 rating. So we, we have to build an application which can predict the rating by seeing the review.

Conceptual Background of the Domain Problem

- Recommendation systems are an important units in today's e-commerce applications, such as targeted advertising, personalized marketing and information retrieval. In recent years, the importance of contextual information has motivated generation of personalized recommendations according to the available contextual information of users. Compared to the traditional systems which mainly utilize user's rating history, review-based recommendation hopefully provide more relevant results to users. We introduce a review-based recommendation approach that obtains contextual information by mining user reviews. The proposed approach relate to features obtained by analyzing textual reviews using methods developed in Natural Language Processing (NLP) and information retrieval discipline to compute a utility function over a given item.
- An item utility is a measure that shows how much it is preferred according to user's current context. In our system, the context inference is modelled as similarity between the user's reviews history and the item reviews history. As an example application, we used our method to mine contextual data from customer's reviews of technical products and use it to produce review-based rating prediction. The predicted ratings can generate recommendations that are item-based and should appear at the recommended items list in the product page. Our evaluations (surprisingly) suggest that our system can help produce better prediction rating scores in comparison to the standard prediction methods.
- As far as we know, all the recent works on recommendation techniques utilizing opinions inferred from user's reviews are either focused on the item recommendation task or use only the opinion information, completely leaving user's ratings out of consideration. The approach proposed in this report is filling this gap, providing a simple, personalized and scalable rating prediction framework utilizing both ratings provided by users and opinions inferred from their reviews. Experimental results provided on dataset containing user ratings and reviews from the real world Amazon and Flipkart Product Review Data show the effectiveness of the proposed framework.

Motivation for the Problem Undertaken

- The project was provided to me by FlipRobo as a part of the internship program. The exposure to real world data and the opportunity to deploy my skillset in solving a real time problem has been the primary objective. Many product reviews are not accompanied by a scale rating system, consisting only of a textual evaluation. In this case, 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. Getting an overall sense of a textual review could in turn improve consumer experience. However, the motivation for taking this project was that it is relatively a new field of research. Here we have many options but less concrete solutions. The main motivation is to build a prototype of online hate and abuse review classifier which can used to classify hate and good comments so that it can be controlled and corrected according to the reviewer’s choice.

Mathematical/ Analytical Modeling of the Problem

- In this problem the Ratings can be 1, 2, 3, 4 or 5, which represents the likely ness of the product to the customer. So clearly it is a multi-classification problem and I have to use all classification algorithms while building the model. We would perform one type of supervised learning algorithms: Classification. Here, we will only perform classification. Since there only 1 feature in the dataset, filtering the words is needed to prevent overfitting. In order to determine the regularization parameter, throughout the project in classification part, we would first remove email, phone number, web address, spaces and stops words etc. In order to further improve our models, we also performed TFID in order to convert the tokens from the train documents into vectors so that machine can do further processing. I have used all the classification algorithms while building model then tuned the best model and saved the best model.

- I will need to build multiple classification machine learning models. Before model building will need to perform all data pre-processing steps involving NLP. After trying different classification models with different hyper parameters then will select the best model out of it. Will need to follow the complete life cycle of data science that includes steps like -
    1. Data Cleaning
    2. Exploratory Data Analysis
    3. Data Pre-processing
    4. Model Building
    5. Model Evaluation
    6. Selecting the best model

- Finally, we compared the results of proposed and baseline features with other machine learning algorithms. Findings of the comparison indicate the significance of the proposed features in review rating prediction.

Data Analysis steps :

Import Libraries → Import Dataset → Data Pre Processing

↓

Ordinal encoding ← Visualization ← Finding null values

Data Sources and their formats

```python
#Reading csv file
df = pd.read_csv("ratings and reviews.csv")
df
```

	Unnamed	Unnamed: 0	Comment	Rating
0	0	0	I was little confused after ordering it as few...	5.0
1	1	1	I have over the years developed good trust in...	5.0
2	2	2	Happy to say that it's a wonderful Washing mac...	5.0
3	3	3	I buy on sep 19 but its not working taking wat...	5.0
4	4	4	Affordable and awesome go for it	5.0
...	...	...	...	...
28568	12949	12949	Not usable for professionals.	1
28569	12950	12950	(Honest opinion)\nGreat Product I would say.....	5
28570	12951	12951	Mouse disconnects automatically,in 3 minutes idle	2
28571	12952	12952	Touch pad turns off automatically after few se...	2
28572	12953	12953	Keyboard is very small and cannot be accessibl...	1

28573 rows × 4 columns

```python
# Dropping unnecessary column
df.drop(columns = 'Unnamed: 0', inplace = True)
df.drop(columns = 'Unnamed', inplace = True)
df
```

	Comment	Rating
0	I was little confused after ordering it as few...	5.0
1	I have over the years developed good trust in...	5.0
2	Happy to say that it's a wonderful Washing mac...	5.0
3	I buy on sep 19 but its not working taking wat...	5.0
4	Affordable and awesome go for it	5.0
...	...	...
28568	Not usable for professionals.	1
28569	(Honest opinion)\nGreat Product I would say.....	5
28570	Mouse disconnects automatically,in 3 minutes idle	2
28571	Touch pad turns off automatically after few se...	2
28572	Keyboard is very small and cannot be accessibl...	1

28573 rows × 2 columns

The data set contains nearly 28573 samples with 2 features. Since **Rating** is my target column and it is a categorical column with 5 categories, this problem is a **Multi Classification Problem**. The Ratings can be 1, 2, 3, 4 or 5, which represents the likeliness of the product to the customer. The data set includes:

- Comment: Text Content of the Review.
- Rating: Ratings out of 5 stars.

This project is more about exploration, feature engineering and classification that can be done on this data. Since the data set is huge and includes multi classification of ratings, we can do good amount of data exploration and derive some interesting features using the Review column available. We need to build a model that can predict Ratings of the review.

Data Preprocessing Done

Importing all necessary libraries and packages

```
# Preprocessing
import numpy as np
import pandas as pd

# Visualization
import seaborn as sns
import matplotlib.pyplot as plt
import os
import scipy as stats

# To remove outliers
from scipy.stats import zscore

#importing nltk libraries
import nltk
from nltk.corpus import stopwords
import re
import string
from nltk import FreqDist
from nltk.tokenize import word_tokenize
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import wordnet
from nltk import FreqDist

# Evaluation Metrics
from sklearn import metrics
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.metrics import roc_curve,accuracy_score,roc_auc_score,hamming_loss, log_loss

# Warning
import warnings
%matplotlib inline
warnings.filterwarnings('ignore')
```

We have imported all the necessary libraries/packages.

Checking the shape of the dataset

```
# Checking the shape of the dataset
print("There are {} Rows and {} Columns in the dataset".format(df.shape[0], df.shape[1]))
```

```
There are 28573 Rows and 2 Columns in the dataset
```

So there are 28573 rows and 2 columns in the dataset.

Checking the column names in the dataset

```
# Checking the column names in the dataset
print("Columns present in the dataset are:\n",df.columns)
```

```
Columns present in the dataset are:
 Index(['Comment', 'Rating'], dtype='object')
```

So above 2 are the column names in the dataset.

Checking the info of the dataset

```
# Let's check the info of the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28573 entries, 0 to 28572
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   Comment  28428 non-null  object
 1   Rating   28573 non-null  object
dtypes: object(2)
memory usage: 446.6+ KB
```

By observing the info we can say that there are some null values in the dataset and all the columns are of object data type which means all the entries are string entries.

Checking the unique value count of target column

```
# Checking the unique value count of target column
df['Rating'].unique()
```

```
array(['5.0', '4.0', '3.0', '-', '2.0', '1.0', '5', '4', '1', '2', '3'],
      dtype=object)
```

Looking the above entries in target column we can observe that

- There are some blank spaces in the column which need to be addressed
- There are string entrie which we shall replace with their respective rating values(stars).

Converting to numeric datatype and checking for null values

```
df['Rating'] = pd.to_numeric(df['Rating'], errors='coerce')
```

```
# Checking for null values
print("Null values in the dataset: \n", df.isnull().sum())
```

```
Null values in the dataset:
 Comment    145
Rating     168
dtype: int64
```

So we have a minimal amount of nan values in the Comment column of the dataset. Apart from the star ratings there is another column which has nan values. Let's view the percentage of the nan values in the both the Rating and Comment columns.

Checking the percentage of missing / nan values in the columns

```
percent_missing1 = df. isnull(). sum() * 100 / len(df)
percent_missing1
```

```
Comment    0.507472
Rating     0.587968
dtype: float64
```

There are approximately 0.5% null values in both the columns. We can use imputation methods to fill these nan values but this could impact the model, so these rows with null values can be dropped from the dataset

Dropping the rows with null values

```
#Dropping the rows with null values
df = df.dropna()
df
```

	Comment	Rating
0	I was little confused after ordering it as few...	5.0
1	I have over the years developed good trust in...	5.0
2	Happy to say that it's a wonderful Washing mac...	5.0
3	I buy on sep 19 but its not working taking wat...	5.0
4	Affordable and awesome go for it	5.0
...	...	...
28568	Not usable for professionals.	1.0
28569	(Honest opinion)\nGreat Product I would say.....	5.0
28570	Mouse disconnects automatically,in 3 minutes idle	2.0
28571	Touch pad turns off automatically after few se...	2.0
28572	Keyboard is very small and cannot be accessibl...	1.0

28260 rows × 2 columns

Now after dropping the null values we have 28260 rows and 2 columns in the dataset

Checking for null values again

```
#Checking for null values again
df.isnull().sum()
```

```
Comment    0
Rating     0
dtype: int64
```

```
# Let's visualize the null values clearly in dataset using heat map
sns.set(rc={'figure.figsize':(12,8)})
sns.heatmap(data=df.isnull())
plt.xticks(rotation=90, fontsize=10)
plt.yticks(rotation=0, fontsize=10)
plt.show()
```

Now we can observe that there are no null values in the dataset

Checking the unique value count of target column and converting the values in the column to integer datatype

```
# Checking the unique value count of target column
df['Rating'].unique()
```

```
array([5., 4., 3., 2., 1.])
```

```
#Converting the values in the column to integer datatype
df['Rating'] = df['Rating'].astype('int')
```

```
# Checking the unique value count of target column again
df['Rating'].unique()
```

```
array([5, 4, 3, 2, 1])
```

Now we can see that the entries in the rating column are in integers

Let's have a look into our Review column and see first 2 entries how the data looks:

```
# Checking data of first row in Review column
df['Comment'][0]
```

```
'I was little confused after ordering it as few reviews regarding this WM was negative because they r
ecieved damaged products.  Trust me it was greatly handled by Amazon and delivery guys in my case and
its working very fine, no complain!  shown me open box as policy after checking product they delivere
d it on 2nd floor.  Thanks amazon, highly satisfied with this order.  Regarding washing and other fea
tures it working fine and about noise i want to say come on guys its a machine it will make little no
ise as its spinning, but its not irritating, for me its a silent, its like little more than phone on
vibration mode.  Blindly go for it, if its handled by good logistics than you wont regret regarding p
roduct.'
```

```
# Checking data of second row in Review column
df['Comment'][1]
```

```
" I have over the years developed good trust in Amazon basic's products. Washing machine, microwave,
refrigerators, laptops, any thing they make and sell is reliable, good quality, excellent service , z
ero delivery issues, no problems with installations! Now I find it easier and more reliable than walk
ing into a store nearby . Amazon Basics gives better products and service at much lower price. Great
products great experience!!👍👌"
```

By observing the Reviews we can say that there are many words, numbers, as well as punctuations which are not important for our predictions. So we need to do good text processing.

Text Processing:

```
#Here I am defining a function to replace some of the contracted words to their full form and removing
def decontracted(text):
    text = re.sub(r"won't", "will not", text)
    text = re.sub(r"don’t", "do not", text)
    text = re.sub(r"can't", "can not", text)
    text = re.sub(r"im ", "i am", text)
    text = re.sub(r"yo ", "you ",text)
    text = re.sub(r"doesn’t", "does not",text)
    text = re.sub(r"n\'t", " not", text)
    text = re.sub(r"\'re", " are", text)
    text = re.sub(r"\'s", " is", text)
    text = re.sub(r"\'d", " would", text)
    text = re.sub(r"\'ll", " will", text)
    text = re.sub(r"\'t", " not", text)
    text = re.sub(r"\'ve", " have", text)
    text = re.sub(r"\'m", " am", text)
    text = re.sub(r"<br>", " ", text)
    text = re.sub(r'http\S+', '', text) #removing urls
    return text
```

Changing all words to Lowercase

```python
# Changing all words to there Lowercase
df['Comment'] = df['Comment'].apply(lambda x : x.lower())

df['Comment'] = df['Comment'].apply(lambda x : decontracted(x))

# Removing punctuations
df['Comment'] = df['Comment'].str.replace('[^\w\s]','')
df['Comment'] = df['Comment'].str.replace('\n',' ')
```

Let's have a look into our text again:

```python
# Checking data of first row in Comment column again
df['Comment'][0]
```

```
'i was little confused after ordering it as few reviews regarding this wm was negative because they r
ecieved damaged products  trust me it was greatly handled by amazon and delivery guys in my case and
its working very fine no complain  shown me open box as policy after checking product they delivered
it on 2nd floor  thanks amazon highly satisfied with this order  regarding washing and other features
it working fine and about noise i want to say come on guys its a machine it will make little noise as
its spinning but its not irritating for me its a silent its like little more than phone on vibration
mode  blindly go for it if its handled by good logistics than you wont regret regarding product'
```

```python
# Checking data of second row in Comment column again
df['Comment'][1]
```

```
' i have over the years developed good trust in amazon basic is products washing machine microwave re
frigerators laptops any thing they make and sell is reliable good quality excellent service  zero del
ivery issues no problems with installations now i find it easier and more reliable than walking into
a store nearby  amazon basics gives better products and service at much lower price great products gr
eat experience'
```

Now the data looks far better than previous. And we have successfully removed punctuations and unwanted text from our text and lowercased all the text data.

Removing Stop Words:

```python
# Removing stopwords
stop = stopwords.words('english')
df['Comment'] = df['Comment'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop
```

```python
# Checking the text data again
df['Comment'][0]
```

```
'little confused ordering reviews regarding wm negative recieved damaged products trust greatly handl
ed amazon delivery guys case working fine complain shown open box policy checking product delivered 2
nd floor thanks amazon highly satisfied order regarding washing features working fine noise want say
come guys machine make little noise spinning irritating silent like little phone vibration mode blind
ly go handled good logistics wont regret regarding product'
```

```python
# Checking the text data again
df['Comment'][1]
```

```
'years developed good trust amazon basic products washing machine microwave refrigerators laptops thi
ng make sell reliable good quality excellent service zero delivery issues problems installations find
easier reliable walking store nearby amazon basics gives better products service much lower price gre
at products great experience'
```

Now we have removed all stop words from the text data.

Lemmatization:

```python
#Initialising Lemmatizer
lemmatizer = nltk.stem.WordNetLemmatizer()
```

```python
#Downloading all required content from following link
nltk.download()
```

```
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml

True
```

```python
#Defining functiom to convert nltk tag to wordnet tags
def nltk_tag_to_wordnet_tag(nltk_tag):
    if nltk_tag.startswith('J'):
        return wordnet.ADJ
    elif nltk_tag.startswith('V'):
        return wordnet.VERB
    elif nltk_tag.startswith('N'):
        return wordnet.NOUN
    elif nltk_tag.startswith('R'):
        return wordnet.ADV
    else:
        return None
```

```python
#defining function to lemmatize our text
def lemmatize_sentence(sentence):
    #tokenize the sentence & find the pos tag
    nltk_tagged = nltk.pos_tag(nltk.word_tokenize(sentence))
    #tuple of (token, wordnet_tag)
    wordnet_tagged = map(lambda x : (x[0], nltk_tag_to_wordnet_tag(x[1])), nltk_tagged)
    lemmatize_sentence = []
    for word, tag in wordnet_tagged:
        if tag is None:
            lemmatize_sentence.append(word)
        else:
            lemmatize_sentence.append(lemmatizer.lemmatize(word,tag))
    return " ".join(lemmatize_sentence)
```

```python
df['Comment'] = df['Comment'].apply(lambda x : lemmatize_sentence(x))
```

```python
# Checking the text data again
df['Comment'][0]
```

```
'little confused order review regard wm negative recieved damage product trust greatly handle amazon
delivery guy case work fine complain show open box policy check product deliver 2nd floor thanks amaz
on highly satisfied order regard wash feature work fine noise want say come guy machine make little n
oise spin irritate silent like little phone vibration mode blindly go handle good logistics wont regr
et regard product'
```

```python
# Checking the text data again
df['Comment'][1]
```

```
'year develop good trust amazon basic product wash machine microwave refrigerator laptops thing make
sell reliable good quality excellent service zero delivery issue problem installation find easy relia
ble walk store nearby amazon basic give good product service much low price great product great exper
ience'
```

So now we have removed the inflectional endings and left out with the base or dictionary form of a word.

Text Normalization - Standardization:

```
#Noice removal
def scrub_words(text):
    #remove html markup
    text = re.sub("(<.*?>)", "", text)
    #remove non-ascii and digits
    text = re.sub("(\\W)", " ", text)
    text = re.sub("(\\d)", "", text)
    #remove white space
    text = text.strip()
    return text
```

```
df['Comment'] = df['Comment'].apply(lambda x : scrub_words(x))
```

```
# Checking the text data again
df['Comment'][0]
```

```
'little confused order review regard wm negative recieved damage product trust greatly handle amazon
delivery guy case work fine complain show open box policy check product deliver nd floor thanks amazo
n highly satisfied order regard wash feature work fine noise want say come guy machine make little no
ise spin irritate silent like little phone vibration mode blindly go handle good logistics wont regre
t regard product'
```

```
# Checking the text data again
df['Comment'][1]
```

```
'year develop good trust amazon basic product wash machine microwave refrigerator laptops thing make
sell reliable good quality excellent service zero delivery issue problem installation find easy relia
ble walk store nearby amazon basic give good product service much low price great product great exper
ience'
```

Finally I have defined a function scrub_words for removing the noise from the text. It will remove any html markups, digits and white spaces from the text.
Now we did all the text-processing steps and got required input for our model. We will get into Visualization part now.

Removing Outliers:

As we know that some of the review are too lengthy, so i have to treat them as outliers and remove them using z_score method.

```
#Checking the shape of the dataset
df.shape
```

```
(28260, 4)
```

```
# Applying zscore to remove outliers
from scipy import stats
from scipy.stats import zscore
z_score = zscore(df[['Comment_WordCount']])
abs_z_score = np.abs(z_score)
filtering_entry = (abs_z_score < 3).all(axis = 1)
df = df[filtering_entry]
df.shape
```

```
(27790, 4)
```

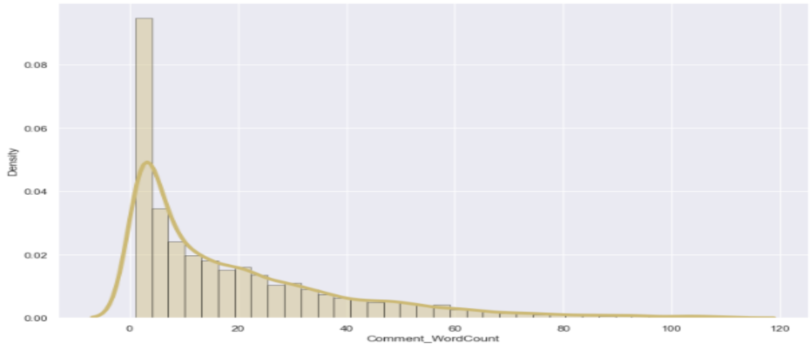
```
outliers_percentage = (28260 - 27790) * 100 / 28260
outliers_percentage
```

```
1.6631280962491153
```

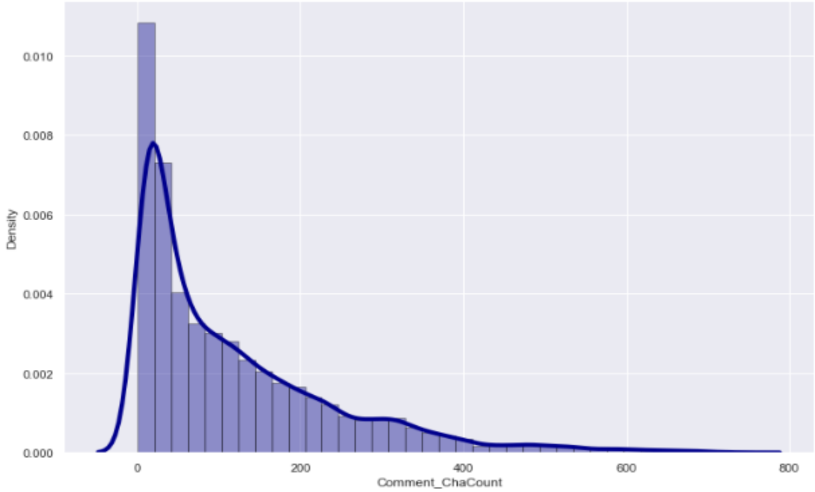
Great by removing the outliers we are losing 1.66% of data which is very less and it is in acceptable range.

Plotting histograms for word count and character counts again after removing outliers:

```
# density plot and histogram of Review word count
sns.distplot(df['Comment_WordCount'], hist = True, kde = True,
             bins = int(180/5), color = 'y',
             hist_kws = {'edgecolor':'black'},
             kde_kws = {'linewidth':4})
plt.show()
```




```
# density plot and histogram of all character count
sns.distplot(df['Comment_ChaCount'], hist = True, kde = True,
             bins = int(180/5), color = 'darkblue',
             hist_kws = {'edgecolor':'black'},
             kde_kws = {'linewidth':4})
plt.show()
```

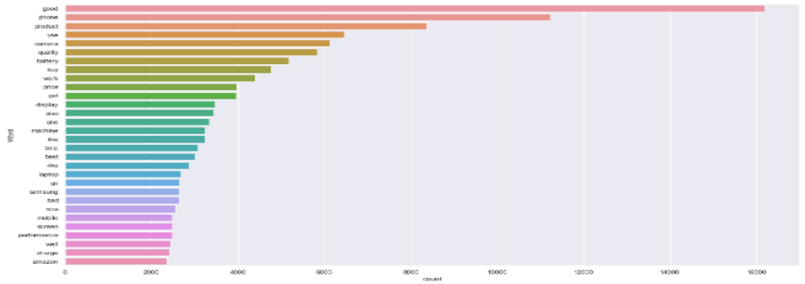


After plotting histograms for word counts and character counts and after removing outliers we can see we are left out with good range of number of words and characters.

iii) Top 30 most frequently occuring words:

```python
#function to plot most frequent terms
def freq_words(x, terms = 30):
    all_words = ' '.join([text for text in x])
    all_words = all_words.split()
    fdist = FreqDist(all_words)
    words_df = pd.DataFrame({'word':list(fdist.keys()),
                             'count':list(fdist.values())})
    #selecting top 30 most freq words
    d = words_df.nlargest(columns = 'count', n = terms)
    plt.figure(figsize = (20,10))
    ax = sns.barplot(data = d, x='count', y='word')
    ax.set(ylabel = 'Word')
    plt.show()
```

```python
freq_words(df['Comment'])
```

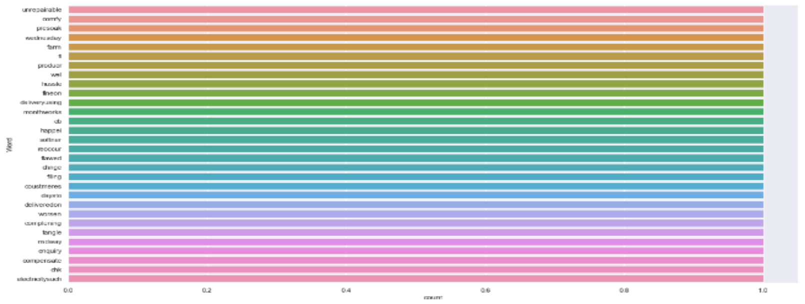


By seeing the above plot we can see that Good, phone, product, use......are occurring frequently.

iv) Top 30 Rare words:

```python
#function to plot least frequent terms
def rare_words(x, terms = 30):
    all_words = ' '.join([text for text in x])
    all_words = all_words.split()
    fdist = FreqDist(all_words)
    words_df = pd.DataFrame({'word':list(fdist.keys()),
                             'count':list(fdist.values())})
    #selecting top 30 most freq words
    d = words_df.nsmallest(columns = 'count', n = terms)
    plt.figure(figsize = (20,10))
    ax = sns.barplot(data = d, x='count', y='word')
    ax.set(ylabel = 'Word')
    plt.show()
```

```python
rare_words(df['Comment'])
```



Above list of words are have rare occurrence in Review.

Word cloud:

```python
from wordcloud import WordCloud, STOPWORDS
stopwords = set(STOPWORDS)
def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
                    background_color='white',
                    stopwords = stopwords,
                    max_words = 500,
                    max_font_size = 40,
                    scale = 3,
                    random_state = 1).generate(str(data))
    fig = plt.figure(1, figsize=(15,15))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)
    plt.imshow(wordcloud)
    plt.show()
```

```python
#Let's plot the loud words with Rating 1
from wordcloud import WordCloud

df1=df['Comment'][df['Rating']==1]

spam_cloud = WordCloud(width=700,height=500,background_color='white',stopwords = stopwords,max_words =

plt.figure(figsize=(10,8),facecolor='r')
plt.imshow(spam_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

Rating 1:

Rating 2:

```python
#Let's plot the loud words with Rating 2
from wordcloud import WordCloud

df2=df['Comment'][df['Rating']==2]

spam_cloud = WordCloud(width=700,height=500,background_color='white',stopwords = stopwords,max_words =

plt.figure(figsize=(10,8),facecolor='b')
plt.imshow(spam_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

Rating 3:

```python
#Let's plot the loud words with Rating 3
from wordcloud import WordCloud

df3=df['Comment'][df['Rating']==3]

spam_cloud = WordCloud(width=700,height=500,background_color='white',stopwords = stopwords,max_words =

plt.figure(figsize=(10,8),facecolor='g')
plt.imshow(spam_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

Rating 4:

```python
#Let's plot the loud words with Rating 4
from wordcloud import WordCloud

df4=df['Comment'][df['Rating']==4]

spam_cloud = WordCloud(width=700,height=500,background_color='white',stopwords = stopwords,max_words =

plt.figure(figsize=(10,8),facecolor='y')
plt.imshow(spam_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
```

Rating 5:

```python
#Let's plot the loud words with Rating 5
from wordcloud import WordCloud

df5=df['Comment'][df['Rating']==5]

spam_cloud = WordCloud(width=700,height=500,background_color='white',stopwords = stopwords,max_words =

plt.figure(figsize=(10,8),facecolor='blue')
plt.imshow(spam_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
```

Assumptions

- From the above plots we can clearly see the words which are indication of Reviewer's opinion on products.

- Here most frequent words used for each Rating is displayed in the word cloud.

Software Used

- Distribution: Anaconda Navigator
- Programming language: Python
- Browser based language shell: Jupyter Notebook
- Word cloud: For visual display of text data
- Libraries/Packages specifically being used - Pandas, NumPy, matplotlib, seaborn, scikit-learn, pandas-profiling, missingno, NLTK

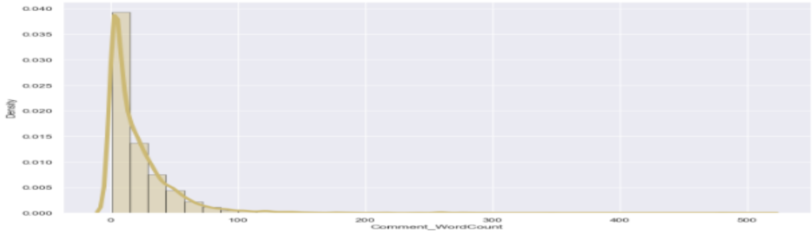
Visualizations

i) Word Counts:

```
# Creating column for word counts in the text
df['Comment_WordCount'] = df['Comment'].apply(lambda x: len(str(x).split(' ')))
df[['Comment_WordCount','Comment']].head()
```

	Comment_WordCount	Comment
0	65	little confused order review regard wm negativ...
1	44	year develop good trust amazon basic product w...
2	115	happy say wonderful washing machine quality su...
3	18	buy sep work take water wash drum move litera...
4	3	affordable awesome go

```
# density plot and histogram of Review word count
sns.distplot(df['Comment_WordCount'], hist = True, kde = True,
             bins = int(180/5), color = 'y',
             hist_kws = {'edgecolor':'black'},
             kde_kws = {'linewidth':4})
plt.show()
```



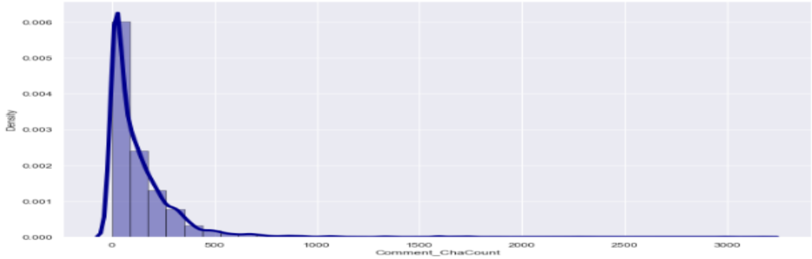
By observing the histogram we can clearly see that most of our text is having the number of words in the range of 0 to 100 but some of the reviews are too lengthy which may act like outliers in our data.

ii) Character count:

```
# Creating column for character counts in the text
df['Comment_ChaCount'] = df['Comment'].str.len()
df[['Comment_ChaCount','Comment']].head()
```

	Comment_ChaCount	Comment
0	419	little confused order review regard wm negativ...
1	307	year develop good trust amazon basic product w...
2	668	happy say wonderful washing machine quality su...
3	96	buy sep work take water wash drum move litera...
4	21	affordable awesome go

```
# density plot and histogram of all character count
sns.distplot(df['Comment_ChaCount'], hist = True, kde = True,
             bins = int(180/5), color = 'darkblue',
             hist_kws = {'edgecolor':'black'},
             kde_kws = {'linewidth':4})
plt.show()
```



Above plot represents histogram for character count of Review text, which is quite similar to the histogram of word count.

Model/s Development and evaluation

Identification of possible problem-solving approaches (methods)

Checking the value counts of Rating column

```
#Checking the value counts of Rating column
df.Rating.value_counts()
```

```
5    11009
4     5785
1     4272
3     3625
2     3099
Name: Rating, dtype: int64
```

Converting text data into vectors using Tfidf Vectorizer:

```
#using the n_gram tfidf vectorizer(Word vectors)
from sklearn.feature_extraction.text import TfidfVectorizer
word_vectorizer = TfidfVectorizer(
                                 sublinear_tf = True,
                                 strip_accents = 'unicode',
                                 analyzer = 'word',
                                 token_pattern = r'\w{1,}',
                                 stop_words = 'english',
                                 ngram_range = (1,3),
                                 max_features = 100000)
word_vectorizer.fit(x)
train_word_features = word_vectorizer.transform(x)
```

```
#Character vectorizer
char_vectorizer = TfidfVectorizer(
                                 sublinear_tf = True,
                                 strip_accents = 'unicode',
                                 analyzer = 'char',
                                 stop_words = 'english',
                                 ngram_range = (2,6),
                                 max_features = 50000)
char_vectorizer.fit(x)
train_char_features = char_vectorizer.transform(x)
```

```
#Combining both word vectors and character vectors as input for our model
from scipy.sparse import hstack
train_features = hstack([train_char_features,train_word_features])
```

I have converted text into feature vectors using TF-IDF vectorizer and separated our feature and labels. Also, before building the model, I made sure that the input data is cleaned and scaled before it was fed into the machine learning models. Just making the Reviews more appropriate so that we'll get less word to process and get more accuracy. Removed extra spaces, converted email address into email keyword, and phone number etc. Tried to make Reviews small and more appropriate as much as possible.

Testing of Identified Approaches (Algorithms)

Model Building and Evaluation:

```
# Separating feature and label
x = df['Comment']
y = df['Rating']
```

Splitting the data into train and test:

```
# Splitting train and test data
seed = 1
x_train, x_test, y_train, y_test = train_test_split(train_features, y, test_size = 0.25, random_state =
```

Data Balancing:

```
#Lets check the shapes of traning and test data
print("x_train", x_train.shape)
print("x_test", x_test.shape)
print("y_train", y_train.shape)
print("y_test", y_test.shape)
```

```
x_train (20842, 150000)
x_test (6948, 150000)
y_train (20842,)
y_test (6948,)
```

Now let's do oversmapling in order to make data balanced.

```
#Checking the value counts of Rating column
y.value_counts()
```

```
5    11009
4     5785
1     4272
3     3625
2     3099
Name: Rating, dtype: int64
```

```
#Checking the number of classes before fit
from collections import Counter
print("The number of classes before fit{}".format(Counter(y_train)))
```

```
The number of classes before fitCounter({5: 8286, 4: 4342, 1: 3211, 3: 2688, 2: 2315})
```

So we have maximum count 8286 for 5 rating which may hamper the model accuracy. Hence we shall balance the data using SMOTE

Oversample and plot imbalanced dataset with SMOTE

```
# Oversample and plot imbalanced dataset with SMOTE
from sklearn.datasets import make_classification
from imblearn.over_sampling import SMOTE

# transforming the dataset
os=SMOTE(sampling_strategy = {1: 37633, 2: 37633, 3: 37633, 4: 37633, 5: 37633})
x_train_ns,y_train_ns=os.fit_resample(x_train,y_train)

print("The number of classes before fit{}".format(Counter(y_train)))
print("The number of classes after fit {}".format(Counter(y_train_ns)))
```

```
The number of classes before fitCounter({5: 8286, 4: 4342, 1: 3211, 3: 2688, 2: 2315})
The number of classes after fit Counter({5: 37633, 1: 37633, 2: 37633, 4: 37633, 3: 37633})
```

So now we have successfully balanced the data. Let's proceed with model building.

In this nlp based project we need to predict Ratings which is a multiclassification problem. I have converted the text into vectors using TFIDF vectorizer and separated our feature and labels then build the model using One Vs Rest Classifier. Among all the algorithms which I have used for this purpose I have chosen SGDClassifier as best suitable algorithm for our final model as it is performing well compared to other algorithms while evaluating with different metrics I have used following algorithms and evaluated them

- LinearSVC
- LogisticRegression
- RandomForestClassifier
- DecisionTreeClassifier
- XGBClassifier
- SGDClassifier

```
# Importing libraries for ML Algorithms
from xgboost import XGBClassifier
from sklearn.svm import LinearSVC
from lightgbm import LGBMClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import GradientBoostingClassifier, AdaBoostClassifier
from sklearn.naive_bayes import MultinomialNB, GaussianNB,BernoulliNB
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV
```

```
# defining the algorithms
rf = RandomForestClassifier()
DTC = DecisionTreeClassifier()
svc = LinearSVC()
lr = LogisticRegression(solver='lbfgs')
mnb = MultinomialNB()
bnb = BernoulliNB()
xgb = XGBClassifier(verbosity=0)
lgb = LGBMClassifier()
sgd = SGDClassifier()
```

Run and Evaluate selected models

```
#creating a function to train and test the model with evaluation
def BuiltModel(model):
    print('*'*30+model.__class__.__name__+'*'*30)
    model.fit(x_train_ns,y_train_ns)
    y_pred = model.predict(x_train_ns)
    pred = model.predict(x_test)

    accuracy = accuracy_score(y_test,pred)*100

    print(f"Accuracy Score:", accuracy)
    print("-" * 50)

    #confusion matrix & classification report

    print(f"CLASSIFICATION REPORT : \n {classification_report(y_test,pred)}")
    print(f"Confusion Matrix : \n {confusion_matrix(y_test,pred)}\n")
```

```
******************************LogisticRegression******************************
Accuracy Score: 82.74323546344272
--------------------------------------------------
CLASSIFICATION REPORT :
              precision    recall  f1-score   support

           1       0.89      0.91      0.90      1061
           2       0.88      0.89      0.88       784
           3       0.75      0.80      0.77       937
           4       0.73      0.72      0.72      1443
           5       0.87      0.84      0.86      2723

    accuracy                           0.83      6948
   macro avg       0.82      0.83      0.83      6948
weighted avg       0.83      0.83      0.83      6948

Confusion Matrix :
 [[ 969   28   37   17   10]
 [  36  695   31   15    7]
 [  36   33  749   80   39]
 [  29   23   76 1043  272]
 [  21   14  112  283 2293]]

******************************LinearSVC******************************
Accuracy Score: 83.7651122625216
--------------------------------------------------
CLASSIFICATION REPORT :
              precision    recall  f1-score   support

           1       0.90      0.92      0.91      1061
           2       0.90      0.89      0.90       784
           3       0.77      0.80      0.79       937
           4       0.71      0.77      0.74      1443
           5       0.89      0.84      0.86      2723

    accuracy                           0.84      6948
   macro avg       0.84      0.84      0.84      6948
weighted avg       0.84      0.84      0.84      6948

Confusion Matrix :
 [[ 972   20   30   23   16]
 [  33  697   29   18    7]
 [  34   28  750   85   40]
 [  20   20   69 1113  221]
[   18     8    91   318  2288]]
```

```
******************************DecisionTreeClassifier****************
Accuracy Score: 81.72135866436385
--------------------------------------------------
CLASSIFICATION REPORT :
               precision    recall  f1-score   support

           1       0.86      0.88      0.87      1061
           2       0.87      0.90      0.88       784
           3       0.77      0.79      0.78       937
           4       0.71      0.73      0.72      1443
           5       0.86      0.83      0.84      2723

    accuracy                           0.82      6948
   macro avg       0.81      0.82      0.82      6948
weighted avg       0.82      0.82      0.82      6948

Confusion Matrix :
 [[ 935   26   25   35   40]
 [  34  702   13   21   14]
 [  41   19  743   82   52]
 [  34   23   78 1049  259]
 [  48   38  107  281 2249]]
```

```
******************************RandomForestClassifier****************
Accuracy Score: 85.03166378814048
--------------------------------------------------
CLASSIFICATION REPORT :
               precision    recall  f1-score   support

           1       0.88      0.93      0.90      1061
           2       0.96      0.89      0.92       784
           3       0.83      0.79      0.81       937
           4       0.78      0.72      0.75      1443
           5       0.85      0.90      0.87      2723

    accuracy                           0.85      6948
   macro avg       0.86      0.84      0.85      6948
weighted avg       0.85      0.85      0.85      6948

Confusion Matrix :
 [[ 984   14   21   16   26]
 [  42  696   15   18   13]
 [  44    7  740   69   77]
 [  25    7   52 1041  318]
 [  19    2   68  187 2447]]
```

```
******************************SGDClassifier*************************
Accuracy Score: 79.74956822107082
--------------------------------------------------
CLASSIFICATION REPORT :
               precision    recall  f1-score   support

           1       0.86      0.91      0.88      1061
           2       0.86      0.87      0.87       784
           3       0.70      0.75      0.72       937
           4       0.73      0.62      0.67      1443
           5       0.82      0.84      0.83      2723

    accuracy                           0.80      6948
   macro avg       0.79      0.80      0.79      6948
weighted avg       0.80      0.80      0.80      6948

Confusion Matrix :
 [[ 967   26   38   14   16]
 [  43  685   29    8   19]
 [  56   40  700   68   73]
 [  33   30   96  899  385]
 [  31   17  144  241 2290]]
```

```
******************************XGBClassifier*************************
Accuracy Score: 80.15256188831317
--------------------------------------------------
CLASSIFICATION REPORT :
               precision    recall  f1-score   support

           1       0.84      0.92      0.87      1061
           2       0.87      0.86      0.86       784
           3       0.73      0.75      0.74       937
           4       0.70      0.68      0.69      1443
           5       0.85      0.82      0.83      2723

    accuracy                           0.80      6948
   macro avg       0.80      0.81      0.80      6948
weighted avg       0.80      0.80      0.80      6948

Confusion Matrix :
 [[ 971   34   23   12   21]
 [  54  673   28   20    9]
 [  56   30  704   91   56]
 [  36   20   86  981  320]
 [  45   18  123  297 2240]]
```

Cross validation score:

```
# Defning function cross_val to find cv score of models
def cross_val(model):
    print('*'*30+model.__class__.__name__+'*'*30)
    scores = cross_val_score(model,train_features,y, cv = 3).mean()*100
    print("Cross validation score :", scores)
```

```
for model in [lr,svc,DTC,sgd,rf,xgb]:
    cross_val(model)
```

```
******************************LogisticRegression******************************
Cross validation score : 51.130103153351946
******************************LinearSVC******************************
Cross validation score : 51.25244789705995
******************************DecisionTreeClassifier******************************
Cross validation score : 42.72795150862678
******************************SGDClassifier******************************
Cross validation score : 51.85695160502307
******************************RandomForestClassifier******************************
Cross validation score : 50.36010982651092
******************************XGBClassifier******************************
Cross validation score : 50.2054142924186
```

All our algorithms are giving approximately 50% accuracy as cross validation scores due to less number of the features and data imbalance. Among these algorithms I am selecting SGD Classifier as best fitting algorithm for our final model as it is giving least difference between accuracy and cv score.

Interpretation of the Results

From all of these above models SGDClassifier is giving me better performance with less difference between the accuracy score and cv score.

HyperParameter Tuning:

```
# Let's selects different parameters for tuning
grid_params = {
                'penalty':['l2','l1','elasticnet'],
                'loss':['hinge','squared_hinge'],
                'n_jobs':[-1,1]
                }VE E
```

```
# Training the model with the given parameters using GridSearchCV
GCV =  GridSearchCV(sgd, grid_params, cv = 3, verbose=10)
GCV.fit(x_train_ns,y_train_ns)
```

```
# Printing the best parameters found by GridSearchCV
GCV.best_params_
```

```
{'loss': 'squared_hinge', 'n_jobs': 1, 'penalty': 'l1'}
```

Final Model:

```
# Training our final model with above best parameters
model = SGDClassifier(loss = 'squared_hinge', n_jobs = 1, penalty = 'l1')
model.fit(x_train_ns,y_train_ns) #fitting data to model
pred = model.predict(x_test)
accuracy = accuracy_score(y_test,pred)*100

# Printing accuracy score
print("Accuracy Score :", accuracy)

# Printing Confusion matrix
print(f"\nConfusion Matrix : \n {confusion_matrix(y_test,pred)}\n")

# Printing Classification report
print(f"\nCLASSIFICATION REPORT : \n {classification_report(y_test,pred)}")
```

```
Accuracy Score : 77.93609671848014

Confusion Matrix :
 [[ 914   41   38   34   34]
 [  45  674   28   10   27]
 [  41   41  677   88   90]
 [  28   25   68  981  341]
 [  45   30   98  381 2169]]


CLASSIFICATION REPORT :
               precision    recall  f1-score   support

           1       0.85      0.86      0.86      1061
           2       0.83      0.86      0.85       784
           3       0.74      0.72      0.73       937
           4       0.66      0.68      0.67      1443
           5       0.82      0.80      0.81      2723

    accuracy                           0.78      6948
   macro avg       0.78      0.78      0.78      6948
weighted avg       0.78      0.78      0.78      6948
```

After hyperparameter tuning we are unable to improve our model accuracy due to less number of features and the data imbalance.

Model Saving:

```
import joblib
joblib.dump(model,"Rating_Prediction.pkl")
```

```
['Rating_Prediction.pkl']
```

Finally I have saved the model into .pkl file.

Key Findings and Conclusions of the Study

- In this project I have collected data of reviews and ratings for different products from amazon.in and flipkart.com.
- we have tried to detect the Ratings in commercial websites on a scale of 1 to 5 on the basis of the reviews given by the users. We made use of natural language processing and machine learning algorithms in order to do so.
- Then I have done different text processing for reviews column and chose equal number of text from each rating class to eliminate problem of imbalance. By doing different EDA steps I have analyzed the text.
- We have checked frequently occurring words in our data as well as rarely occurring words.
- After all these steps I have built function to train and test different algorithms and using various evaluation metrics I have selected SGD Classifier for our final model.
- Finally by doing hyperparameter tuning we got optimum parameters for our final model.

Learning Outcomes of the Study in respect of Data Science

- I have scrapped the reviews and ratings of different technical products from flipkart.com and amazon.in websites. Improvement in computing technology has made it possible to examine social information that cannot previously be captured, processed and analyzed. New analytical techniques (NLP) of machine learning can be used in property research. The power of visualization has helped us in understanding the data by graphical representation it has made me to understand what data is trying to say. Data cleaning is one of the most important steps to remove unrealistic values, punctuations, URLs, email address and stop words. This study is an exploratory attempt to use 6 machine learning algorithms in estimating Rating, and then compare their results.
- To conclude, the application of NLP in rating classification is still at an early stage. We hope this study has moved a small step ahead in providing some methodological and empirical contributions to crediting institutes, and presenting an alternative approach to the valuation of Ratings.
- 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. Through different powerful tools of visualization we were able to analyses and interpret different hidden insights about the data. The few challenges while working on this project are:-
    - Imbalanced dataset
    - Lots of text data

Thank you