Sentiment Analysis of Web Scraped Product Reviews

A Mini-project report Submitted to the Department of Computer Applications, Bharathiar University, in the partial fulfilment of the requirements for the Award of degree of

MASTER OF SCIENCE IN DATA ANALYTICS

Submitted by

ASPIN C
(18CSEG007)

Under the guidance of

Dr. R. RAJESWARI,
Associate Professor

DEPARTMENT OF COMPUTER APPLICATIONS

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

BHARATHIAR UNIVERSITY

COIMBATORE-641 046

TAMIL NADU

NOVEMBER-2019

DECLARATION

DECLARATION

I hereby declare that this mini-project work titled, "**Sentiment Analysis of Web Scraped Product Reviews**" submitted to Department of Computer Applications, Bharathiar University, in partial fulfilment of the requirements for the award of the degree of **Master of Science in Data Analytics,** is a record of original work done by me, under the supervision and guidance of **Dr. R. RAJESWARI,** Associate professor, Department of Computer Applications, Bharathiar University, and that this project work has not formed the basis for the award of any Degree /Diploma /Associateship /Fellowship or similar title to any candidate of any University.

Place : Coimbatore	Signature of the Candidate
Date :	(ASPIN C)

Countersigned by

Project Guide

CERTIFICATE

CERTIFICATE

This is certify that, this mini-project work entitled, "**Sentiment Analysis of Web Scraped Product Reviews**" submitted to Bharathiar University, in partial fulfilment of the requirements for the award of the degree of **Master of Science in Data Analytics,** is a record of original work done by **ASPIN C (18CSEG007)**, during his period of study in the Department of Computer Applications, Bharathiar University, Coimbatore, under my supervision and guidance and that this project work has not formed the basis for the award of any Degree /Diploma /Associateship /Fellowship or similar title to any candidate of any University.

Place : Coimbatore
Date :

Project Guide   Head of the Department

Submitted for the Project VIVA-VOCE Examination held on ___________________

Internal Examiner   External Examiner

TABLE OF CONTENTS

TABLE OF CONTENTS

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

I express my respectful thanks to our Professor & Head of the Department, **Dr. T. DEVI, M.C.A., M.Phil., Ph.D. (UK),** Department of Computer Applications, Bharathiar University, for permitting me to carry out my mini project work in "**Sentiment Analysis of Web Scraped Product Reviews**"

I express heart-felt gratitude to my project guide **Dr.R.RAJESWARI,** Associate professor, Department of Computer Applications, Bharathiar University, for her valuable support during research work.

I would like to heartily thank all respected faculty members, laboratory staff of Department of Computer Applications, Bharathiar University, Coimbatore- 641046.

I thank my parents for their encouragement and moral support. I would also thank **MonkeyLearn API** to complete this project without any complexity.

I thank everybody whose names are mentioned and not mentioned here, who has directly or indirectly contributed in bringing out this project successfully.

I. INTRODUCTION

As there is large volume of data in the world, individuals are proposed to develop a framework that can distinguish and classify data based on opinions. Sentiment analysis (opinion mining) can be considered as the one of the major application of Natural Language Processing (NLP). We are looking forward to sentiment in product reviews using MonkeyLearn API[7]. We have used web scraping method to retrieve product reviews from Flipkart website which is used as dataset in this experiment. Finally, we compare the sentiment reviews, so that decision making will be lot easier for the user regarding purchasing of the product.

Web Scraping means extracting information from websites by parsing the HTML of the webpage[8]. Parsing a HTML webpage is really easy in Python. You can get the relevant information you need using just a few lines of code in Python. As Flipkart is a famous one the product reviews from this website is considered for scrapping.

Text communication is one of the most popular forms of day today conversion. We chat, message, tweet, share status, email, write blogs, share opinion and feedback in our daily routine. All of these activities are generating text in a significant amount, which is unstructured in nature. In this area of the online marketplace and social media, it is essential to analyse vast quantities of data, to understand people's opinion.

NLP enables the computer to interact with humans in a natural manner. It helps the computer to understand the human language and derive meaning from it. NLP is applicable in several problematic from speech recognition, language translation, classifying documents to information extraction[6].

Opinion mining also known as sentiment analysis technique implements various algorithms to analyse the corpus of data and make sense out of it. This technique helps to identify the orientation of a sentence thereby recognising the element of positivity or negativity in it. Automated opinion mining can be implemented through a machine learning based approach. Opinion mining uses natural language processing to extract the subjective information from the data (in this case its customer reviews[2]).

i. Objectives

There is an increase in demand for e-commerce with people preferring online purchasing of goods and products. The e-commerce websites are loaded with large volume of data. Also, social media helps a great deal in sharing of this information. This has greatly influenced consumer habits all over the world. Due to the vivid reviews provided by the customers, there is a feedback environment being developed for helping customers buy the right product and guiding companies to enhance the features of product suiting consumer's demand[3]. The only disadvantage of availability of this huge volume of data is its diversity and its structural non-uniformness.

The customer finds it difficult to precisely find the review for a particular feature of a product that they intend to buy. Also, there is a mixture of positive and negative reviews thereby making it difficult for customer to find a cogent response. Also these reviews suffer from spammed reviews from unauthenticated users. So to avoid this confusion and make this review system more transparent and user friendly we propose a technique to extract feature based opinion from a diverse pool of reviews, segregate it with respect to the aspects of the product and further classifying it into positive and negative reviews using machine learning based approach. The objectives of this work are to:

- Perform web scraping of product reviews from Flipkart[8].
- Perform sentiment analysis of the web scrapped product reviews[1].

II. BACKGROUND

i. Scraped Product Reviews

Scraping Customer Reviews from Flipkart[8] can be useful for

- Getting complete review details that you can’t get with python.
- Monitoring customer opinion on products that you sell or manufacture using Data Analysis.
- Create Flipkart Review Datasets for Educational Purposes and Research.

Requests, is one of the packages in **Python** that made the language interesting. Requests is based on Python’s urllib2 module. Requests gets the web page for you, but you need to parse the HTML from the page to retrieve the data. That is done by **BeautifulSoup**. Some use lxml to parse the HTML, but in this work BeautifulSoup is used.

ii. Data Pre-processing

The first step should be cleaning the data in order to obtain better features. We will achieve this by doing some of the basic pre-processing steps on our data[6]. Some text pre-processing steps are,

- Lower casing
- Punctuation removal
- Stop words removal
- Frequent words removal
- Rare words removal
- Spelling correction

iii. Sentence Extraction

Once you have identified, extracted, and cleansed the content needed for your use case, the next step is to have an understanding of that content. In many use cases, the content with the most important information is written down in a natural language (such as English, German, Spanish, Chinese, etc.) and not conveniently tagged. To extract information from this content you will need to rely on some levels of text mining, text extraction, or possibly full-up natural language processing (NLP) techniques[5].

- Tokenization
- Stemming
- Lemmatization

iv. Classification

Text classification[4] is the process of assigning predefined tags or categories to unstructured text. It's considered one of the most useful Natural Language Processing (NLP) techniques because it's so versatile and can organize, structure and categorize pretty much anything to deliver meaningful data and solve problems.

CLASSIFICATION ALGORITHMS

The classification[6] step usually involves a statistical model like Naïve Bayes, Logistic Regression, Support Vector Machines, or Neural Networks.

- Naïve Bayes:

  A family of probabilistic algorithms that uses Bayes Theorem to predict the category of a text.

- Linear Regression:

  A very well-known algorithm in statistics used to predict some value (Y) given a set of features (X).

- Support Vector Machines:

  A non-probabilistic model which uses a representation of text examples as points in a multidimensional space. These examples are mapped so that the examples of the different categories (sentiments) belong to distinct regions of that space.. Then, new texts are mapped onto that same space and predicted to belong to a category based on which region they fall into.

- Deep Learning:

  A diverse set of algorithms that attempts to imitate how the human brain works by employing artificial neural networks to process data.

III. PROPOSED WORK

i. Block Diagram:

The block diagram of the proposed system is given in figure 1.

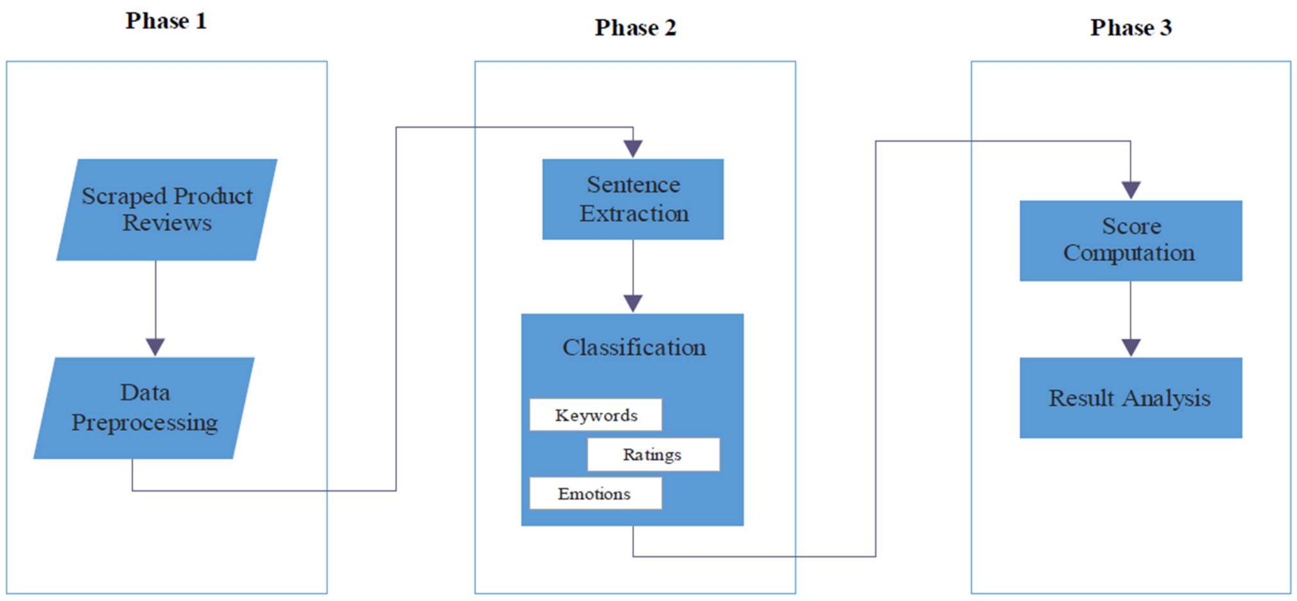


Figure.1 Block diagram of the proposed system

ii. Methodology:

a) Scraped Product Reviews

Requests, is one of the packages in **Python** that made the language interesting. Requests is based on Python's urllib2 module. Requests gets the web page for you, but you need to parse the HTML from the page to retrieve the data. That is done by **BeautifulSoup**. Some use lxml to parse the HTML, but in this work BeautifulSoup is used.

b) Data Pre-processing

- Punctuation removal

  As it doesn't add any extra information while treating text data.
- Stop words removal

  Stop words (or commonly occurring words) should be removed from the text data.

- Frequent words removal

  We just removed commonly occurring words in a general sense.
- Rare words removal

  We removed the most common words, this time let's remove rarely occurring words from the text. Because they're so rare.

c) Sentence Extraction

To extract information from this content you will need to rely on some levels of text mining, text extraction, or possibly full-up natural language processing (NLP) techniques.

- Tokenization

  It will make each words separate in a sentence.
- Stemming

  Stemming refers to the removal of suffices by a simple rule-based approach.
- Lemmatization

  It makes use of the vocabulary and does a morphological analysis to obtain the root word.

d) Classification

Below, we're going to focus the most common text classification task of sentiment analysis,

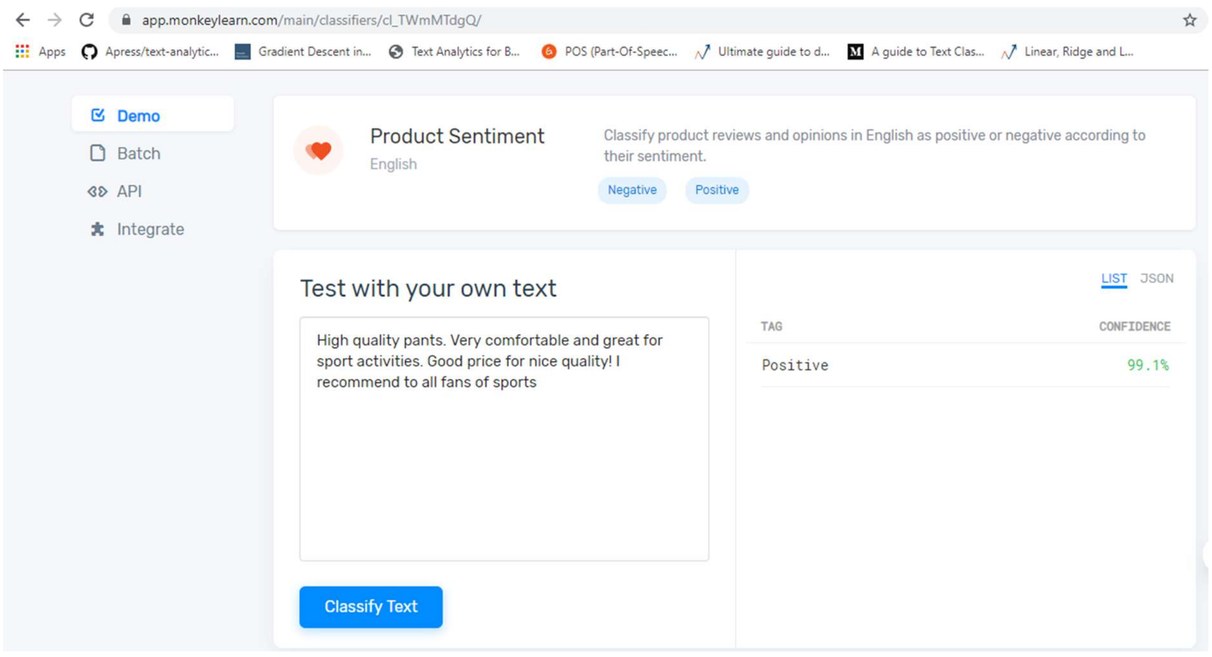
- Product Sentiment Analysis[3]

This machine learning tool can provide insights by automatically analysing product reviews and separating them into tags: Positive, Neutral and Negative. By using sentiment analysis to structure product reviews, you can,

- Understand what your customers like and dislike about your product.
- Compare your product reviews with those of your competitors.
- Get the latest product insights in real-time, 24/7.
- Save hundreds of hours of manual data processing.

e) Score Computation

We can automate product review analysis with machine learning. We can build our own text analysis model without needing to know how to code or have experience in machine learning. From figure.2, we can see MonkeyLearn API[7] will gives a tag of sentiment and its confidence level in percentage.



Classify product reviews and opinions in English as positive or negative according to their sentiment.

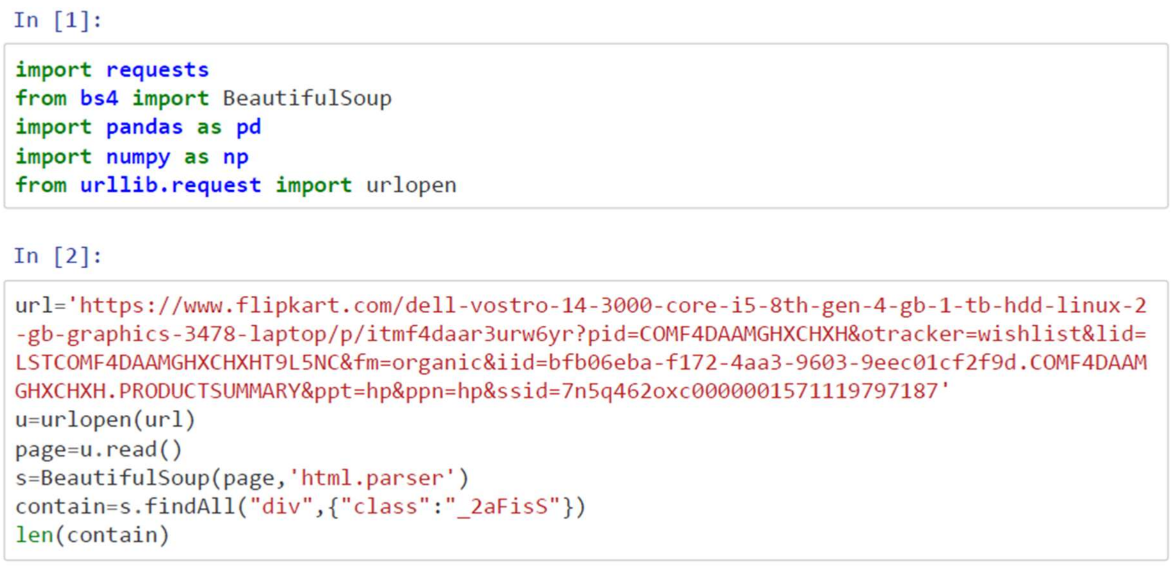
High quality pants. Very comfortable and great for sport activities. Good price for nice quality! I recommend to all fans of sports

TAG	CONFIDENCE
Positive	99.1%

Figure.2 Score computation using Monkeylearn.

IV. IMPLEMENTATION

i. Coding And Explanation:



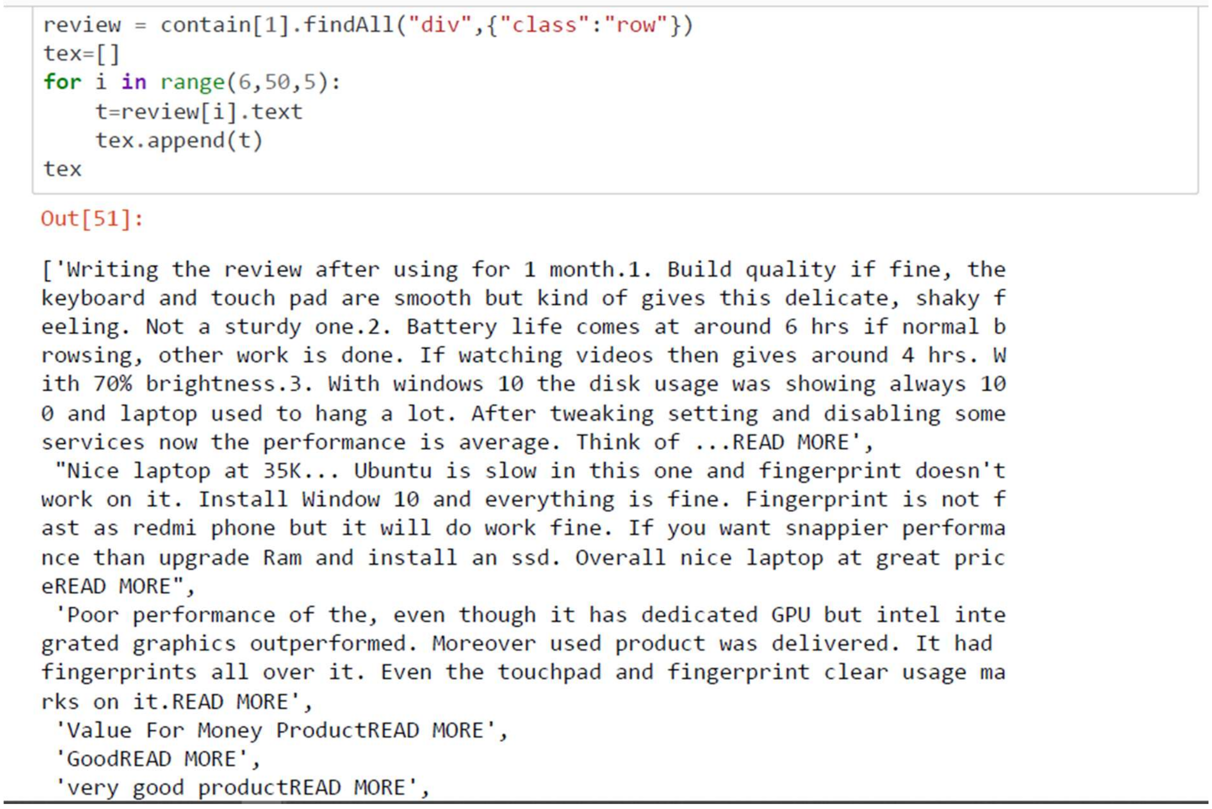
```
In [1]:

import requests
from bs4 import BeautifulSoup
import pandas as pd
import numpy as np
from urllib.request import urlopen
```

```
In [2]:

url='https://www.flipkart.com/dell-vostro-14-3000-core-i5-8th-gen-4-gb-1-tb-hdd-linux-2
-gb-graphics-3478-laptop/p/itmf4daar3urw6yr?pid=COMF4DAAMGHXCHXH&otracker=wishlist&lid=
LSTCOMF4DAAMGHXCHXHT9L5NC&fm=organic&iid=bfb06eba-f172-4aa3-9603-9eec01cf2f9d.COMF4DAAM
GHXCHXH.PRODUCTSUMMARY&ppt=hp&ppn=hp&ssid=7n5q462oxc0000001571119797187'
u=urlopen(url)
page=u.read()
s=BeautifulSoup(page,'html.parser')
contain=s.findAll("div",{"class":"_2aFisS"})
len(contain)
```

This is the coding for importing required libraries for web scraping and basic operations in database. **Requests**, is one of the packages in **Python** that made the language interesting. Requests is based on Python’s urllib2 module. Requests gets the web page for you, but you need to parse the HTML from the page to retrieve the data. That is done by **BeautifulSoup**.

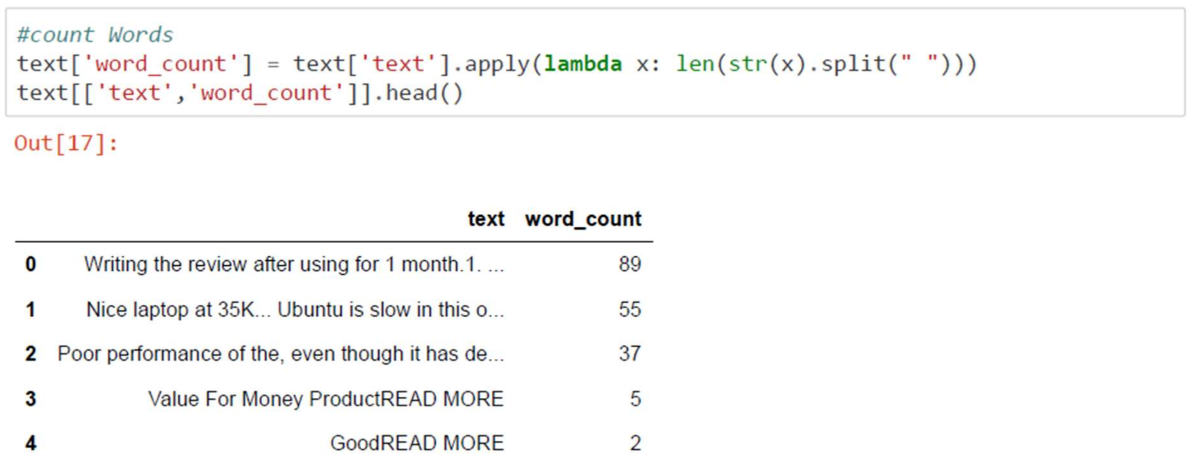


```
review = contain[1].findAll("div",{"class":"row"})
tex=[]
for i in range(6,50,5):
    t=review[i].text
    tex.append(t)
tex
```

```
Out[51]:

['Writing the review after using for 1 month.1. Build quality if fine, the
keyboard and touch pad are smooth but kind of gives this delicate, shaky f
eeling. Not a sturdy one.2. Battery life comes at around 6 hrs if normal b
rowsing, other work is done. If watching videos then gives around 4 hrs. W
ith 70% brightness.3. With windows 10 the disk usage was showing always 10
0 and laptop used to hang a lot. After tweaking setting and disabling some
services now the performance is average. Think of ...READ MORE',
 "Nice laptop at 35K... Ubuntu is slow in this one and fingerprint doesn't
work on it. Install Window 10 and everything is fine. Fingerprint is not f
ast as redmi phone but it will do work fine. If you want snappier performa
nce than upgrade Ram and install an ssd. Overall nice laptop at great pric
eREAD MORE",
 'Poor performance of the, even though it has dedicated GPU but intel inte
grated graphics outperformed. Moreover used product was delivered. It had
fingerprints all over it. Even the touchpad and fingerprint clear usage ma
rks on it.READ MORE',
 'Value For Money ProductREAD MORE',
 'GoodREAD MORE',
 'very good productREAD MORE',
```

The above code is used to scrap a product review for a particular product of **DELL Vostro 3478 14" Laptop** in **FLIPKART**[8]. Output is showing the product reviews by the customers.

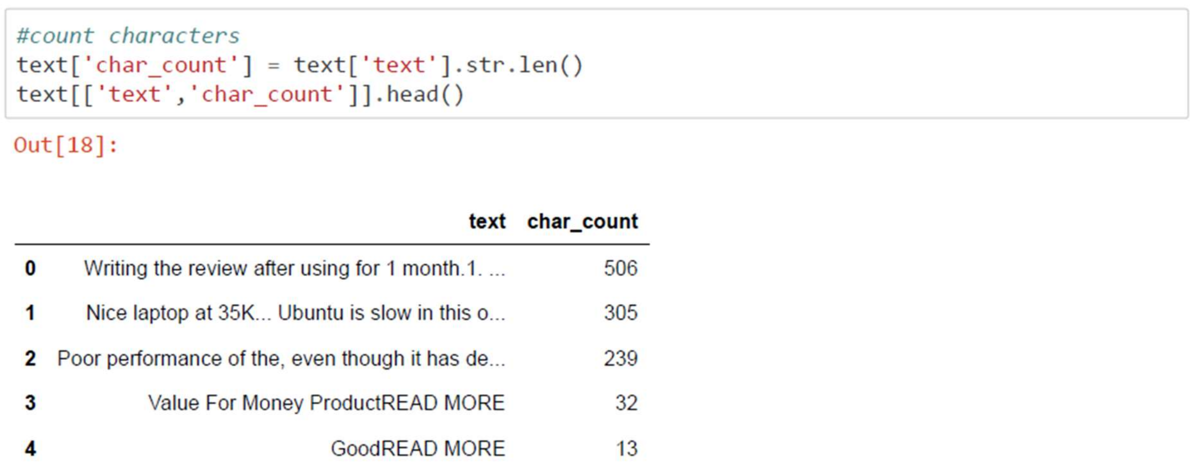


```
#count Words
text['word_count'] = text['text'].apply(lambda x: len(str(x).split(" ")))
text[['text','word_count']].head()
```

```
Out[17]:
```

	text	word_count
0	Writing the review after using for 1 month.1. ...	89
1	Nice laptop at 35K... Ubuntu is slow in this o...	55
2	Poor performance of the, even though it has de...	37
3	Value For Money ProductREAD MORE	5
4	GoodREAD MORE	2

One of the most basic features we can extract is the number of words in each review. The basic intuition behind this is that generally, the negative sentiments contain a lesser amount of words than the positive ones.

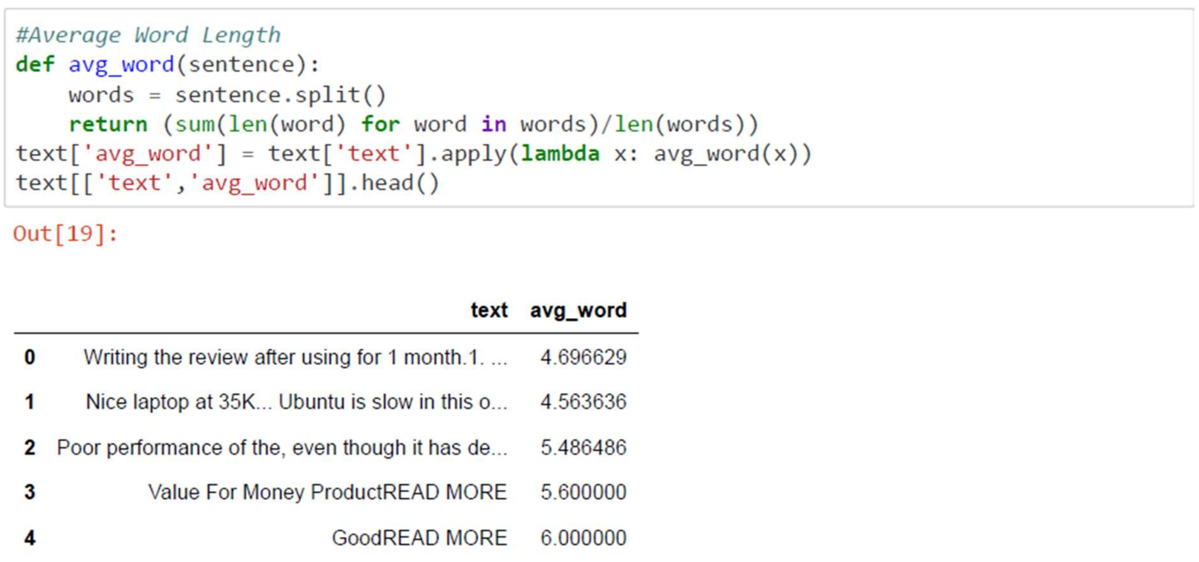


```
#count characters
text['char_count'] = text['text'].str.len()
text[['text','char_count']].head()
```

```
Out[18]:
```

	text	char_count
0	Writing the review after using for 1 month.1. ...	506
1	Nice laptop at 35K... Ubuntu is slow in this o...	305
2	Poor performance of the, even though it has de...	239
3	Value For Money ProductREAD MORE	32
4	GoodREAD MORE	13

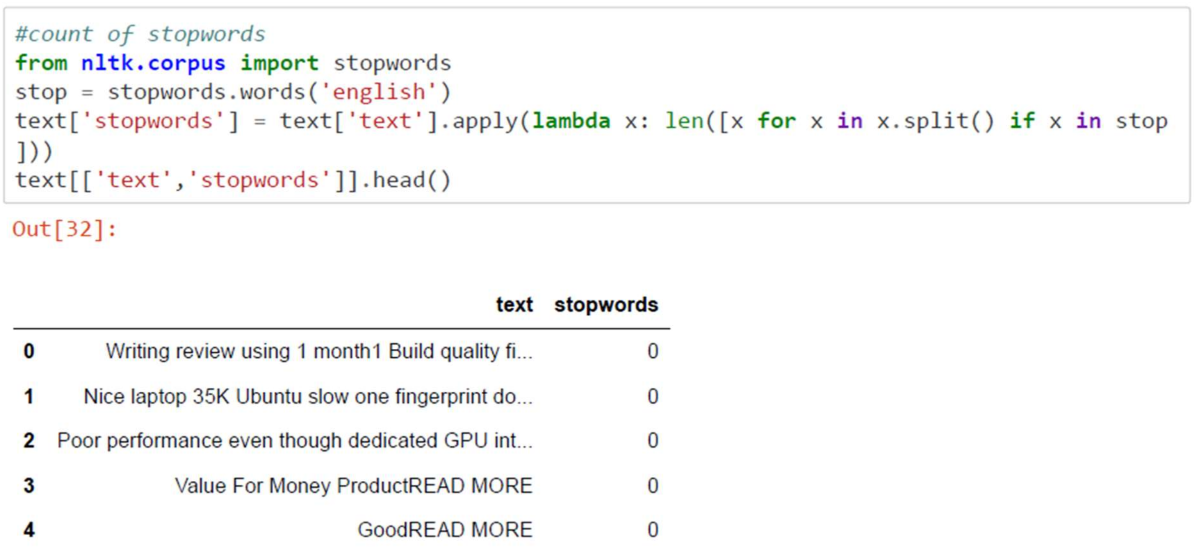
This feature is also based on the previous feature intuition. Here, we calculate the number of characters in each review. This is done by calculating the length of the review.


```
#Average Word Length
def avg_word(sentence):
    words = sentence.split()
    return (sum(len(word) for word in words)/len(words))
text['avg_word'] = text['text'].apply(lambda x: avg_word(x))
text[['text','avg_word']].head()
```

```
Out[19]:

                                                 text   avg_word
0       Writing the review after using for 1 month.1. ...   4.696629
1        Nice laptop at 35K... Ubuntu is slow in this o...   4.563636
2  Poor performance of the, even though it has de...   5.486486
3                   Value For Money ProductREAD MORE   5.600000
4                                        GoodREAD MORE   6.000000
```

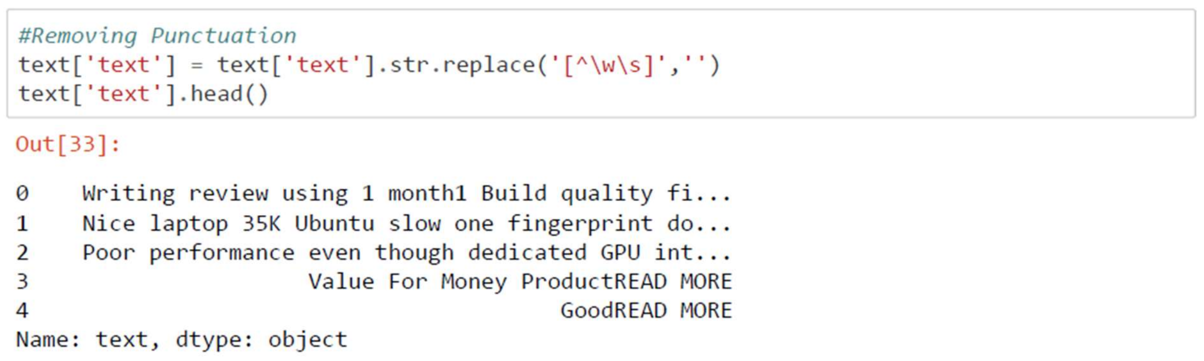
We will also extract another feature which will calculate the average word length of each review. This can also potentially help us in improving our model. Here, we simply take the sum of the length of all the words and divide it by the total length of the review:


```
#count of stopwords
from nltk.corpus import stopwords
stop = stopwords.words('english')
text['stopwords'] = text['text'].apply(lambda x: len([x for x in x.split() if x in stop
]))
text[['text','stopwords']].head()
```

```
Out[32]:

                                                 text   stopwords
0         Writing review using 1 month1 Build quality fi...           0
1        Nice laptop 35K Ubuntu slow one fingerprint do...           0
2  Poor performance even though dedicated GPU int...           0
3                   Value For Money ProductREAD MORE           0
4                                        GoodREAD MORE           0
```

Generally, while solving an NLP problem, the first thing we do is to remove the stopwords. But sometimes calculating the number of stopwords can also give us some extra information which we might have been losing before. Here, we have imported stopwords from NLTK, which is a basic NLP library in python.

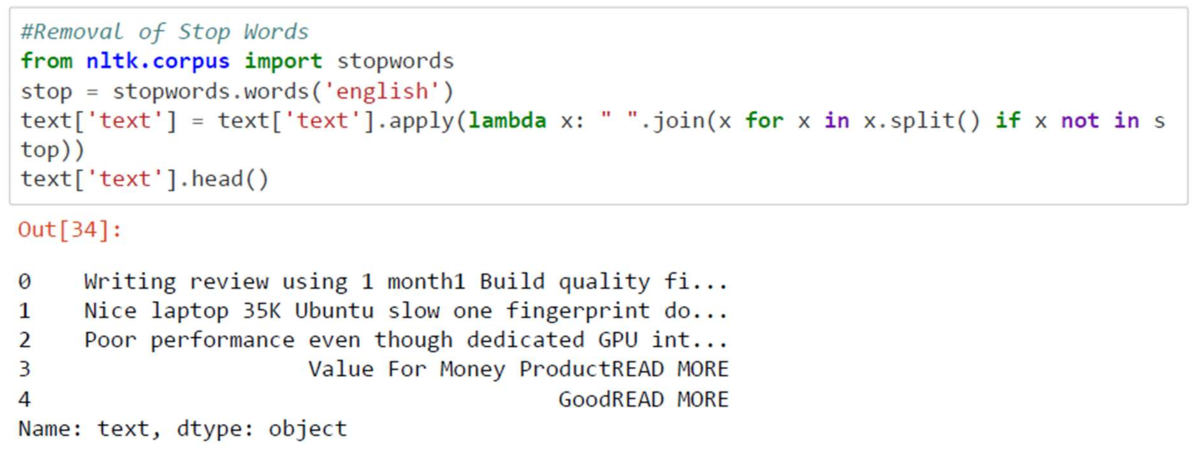


```
#Removing Punctuation
text['text'] = text['text'].str.replace('[^\w\s]','')
text['text'].head()
```

```
Out[33]:

0    Writing review using 1 month1 Build quality fi...
1    Nice laptop 35K Ubuntu slow one fingerprint do...
2    Poor performance even though dedicated GPU int...
3                     Value For Money ProductREAD MORE
4                                        GoodREAD MORE
Name: text, dtype: object
```

The next step is to remove punctuation, as it doesn't add any extra information while treating text data. Therefore removing all instances of it will help us reduce the size of the training data.



```
#Removal of Stop Words
from nltk.corpus import stopwords
stop = stopwords.words('english')
text['text'] = text['text'].apply(lambda x: " ".join(x for x in x.split() if x not in s
top))
text['text'].head()
```

```
Out[34]:

0    Writing review using 1 month1 Build quality fi...
1    Nice laptop 35K Ubuntu slow one fingerprint do...
2    Poor performance even though dedicated GPU int...
3                     Value For Money ProductREAD MORE
4                                        GoodREAD MORE
Name: text, dtype: object
```

As we discussed earlier, stop words (or commonly occurring words) should be removed from the text data. For this purpose, we can either create a list of stopwords ourselves or we can use predefined libraries.

```
# tokenization and find the frequency

from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
t=[]
for i in range(len(text['text'])):
    tokenized_text=sent_tokenize(text['text'][i])
    tokenized_word=word_tokenize(text['text'][i])
    t =t+tokenized_word
#print(t)
freq = pd.Series(' '.join(text['text']).split()).value_counts()[:10]
print(freq)
fdist = FreqDist(t)
print(fdist)
```

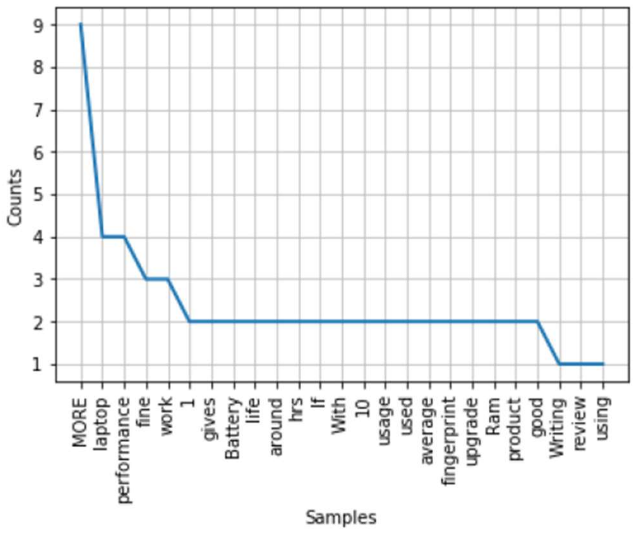
```
MORE           9
laptop         4
performance    4
work           3
fine           3
used           2
fingerprint    2
usage          2
```

In statistics, a frequency distribution is a list, table or graph that displays the frequency of various outcomes in a sample. Each entry in the table contains the frequency or count of the occurrences of values within a particular group or interval.

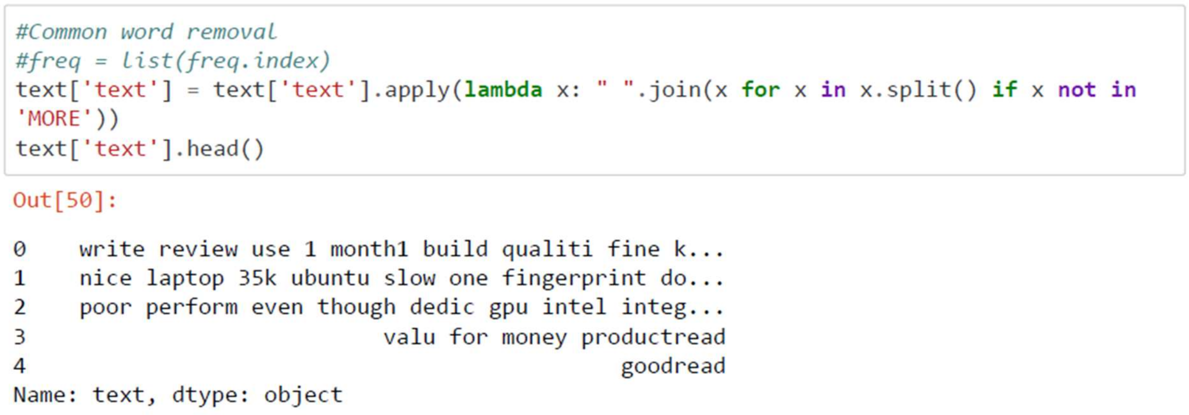
```
#plotting the frequency of words
import matplotlib.pyplot as plt

fdist.plot(25,cumulative=False)

plt.show()
```



Frequency is a built-in function which comes under the statistical category. The frequency distribution can be defined as a list of data or graph that gives the frequency of various outcomes.

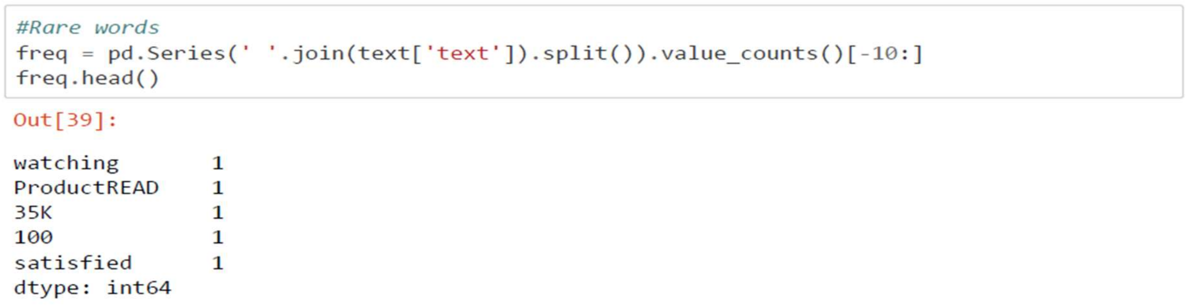


```
#Common word removal
#freq = list(freq.index)
text['text'] = text['text'].apply(lambda x: " ".join(x for x in x.split() if x not in
'MORE'))
text['text'].head()
```

```
Out[50]:

0    write review use 1 month1 build qualiti fine k...
1    nice laptop 35k ubuntu slow one fingerprint do...
2    poor perform even though dedic gpu intel integ...
3                               valu for money productread
4                                                 goodread
Name: text, dtype: object
```

Previously, we just removed commonly occurring words in a general sense. We can also remove commonly occurring words from our text data first, let's check the 10 most frequently occurring words in our text data then take call to remove or retain.

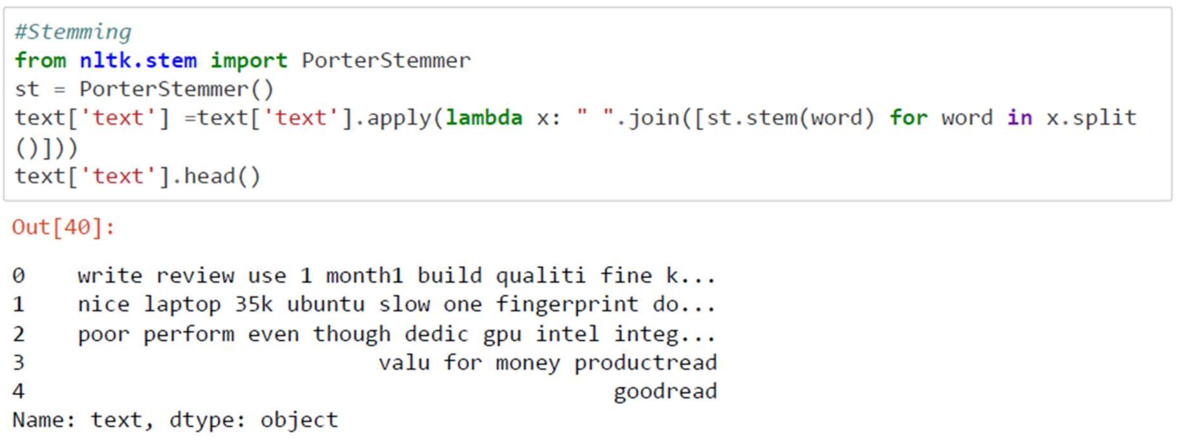


```
#Rare words
freq = pd.Series(' '.join(text['text']).split()).value_counts()[-10:]
freq.head()
```

```
Out[39]:

watching       1
ProductREAD    1
35K            1
100            1
satisfied      1
dtype: int64
```

Similarly, just as we removed the most common words, this time let's remove rarely occurring words from the text. Because they're so rare, the association between them and other words is dominated by noise. You can replace rare words with a more general form and then this will have higher counts

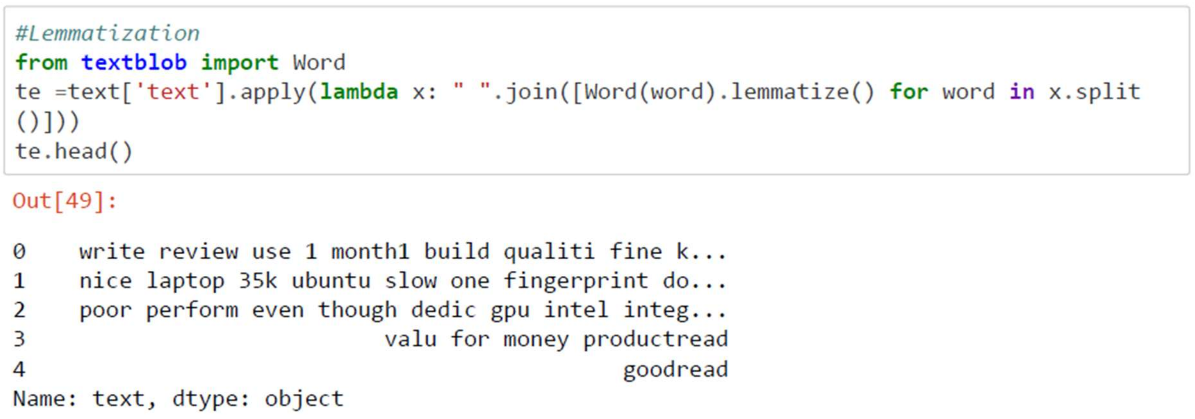


```
#Stemming
from nltk.stem import PorterStemmer
st = PorterStemmer()
text['text'] =text['text'].apply(lambda x: " ".join([st.stem(word) for word in x.split
()]))
text['text'].head()
```

```
Out[40]:

0    write review use 1 month1 build qualiti fine k...
1    nice laptop 35k ubuntu slow one fingerprint do...
2    poor perform even though dedic gpu intel integ...
3                            valu for money productread
4                                              goodread
Name: text, dtype: object
```

Stemming refers to the removal of suffices, like “ing”, “ly”, “s”, etc. by a simple rule-based approach. For this purpose, we will use PorterStemmer from the NLTK library.

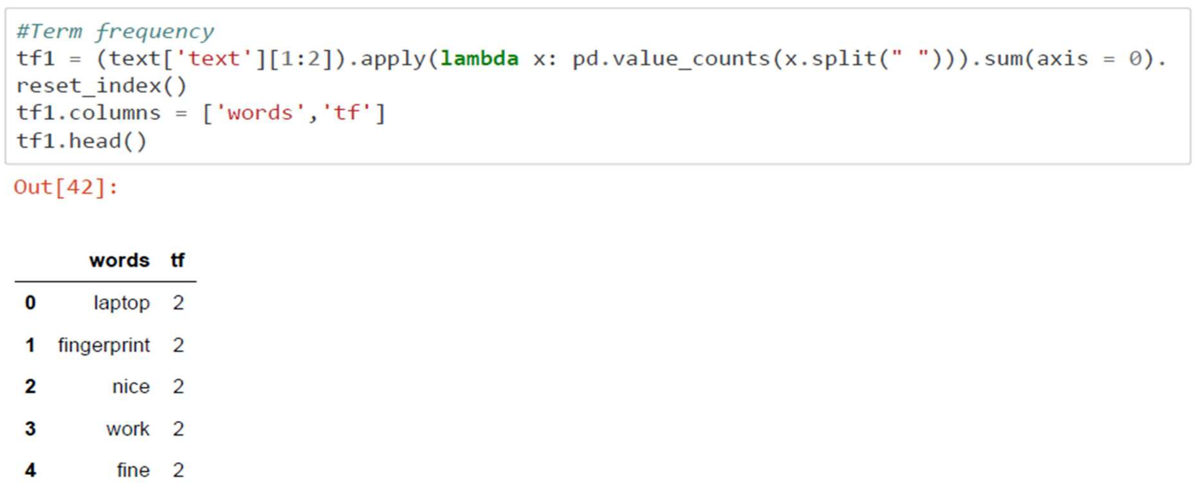


```
#Lemmatization
from textblob import Word
te =text['text'].apply(lambda x: " ".join([Word(word).lemmatize() for word in x.split
()]))
te.head()
```

```
Out[49]:

0    write review use 1 month1 build qualiti fine k...
1    nice laptop 35k ubuntu slow one fingerprint do...
2    poor perform even though dedic gpu intel integ...
3                            valu for money productread
4                                              goodread
Name: text, dtype: object
```

Lemmatization is a more effective option than stemming because it converts the word into its root word, rather than just stripping the suffices. It makes use of the vocabulary and does a morphological analysis to obtain the root word. Therefore, we usually prefer using lemmatization over stemming.



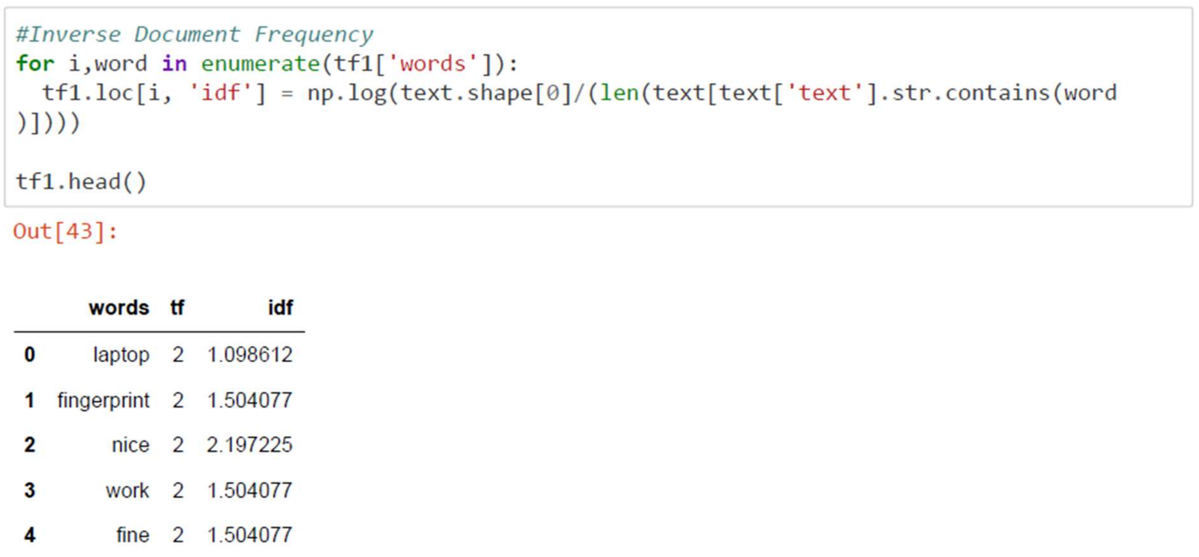
```
#Term frequency
tf1 = (text['text'][1:2]).apply(lambda x: pd.value_counts(x.split(" "))).sum(axis = 0).
reset_index()
tf1.columns = ['words','tf']
tf1.head()
```

```
Out[42]:

         words  tf
0       laptop   2
1  fingerprint   2
2         nice   2
3         work   2
4         fine   2
```

Term frequency is simply the ratio of the count of a word present in a sentence, to the length of the sentence. Therefore, we can generalize term frequency as:

TF = (Number of times term T appears in the particular row) / (number of terms in that row)



```
#Inverse Document Frequency
for i,word in enumerate(tf1['words']):
  tf1.loc[i, 'idf'] = np.log(text.shape[0]/(len(text[text['text'].str.contains(word
)])))

tf1.head()
```

```
Out[43]:

         words  tf       idf
0       laptop   2  1.098612
1  fingerprint   2  1.504077
2         nice   2  2.197225
3         work   2  1.504077
4         fine   2  1.504077
```

The intuition behind inverse document frequency (IDF) is that a word is not of much use to us if it's appearing in all the documents. Therefore, the IDF of each word is the log of the ratio of the total number of rows to the number of rows in which that word is present.

IDF = log(N/n),

Where, N is the total number of rows and n is the number of rows in which the word was present.

```
#Term Frequency - Inverse Document Frequency (TF-IDF)
tf1['tf-idf'] = tf1['tf'] * tf1['idf']
tf1.head()
```

```
Out[44]:
```

	words	tf	idf	tf-idf
0	laptop	2	1.098612	2.197225
1	fingerprint	2	1.504077	3.008155
2	nice	2	2.197225	4.394449
3	work	2	1.504077	3.008155
4	fine	2	1.504077	3.008155

We can see that the TF-IDF has penalized words like ‘don’t’, ‘can’t’, and ‘use’ because they are commonly occurring words. However, it has given a high weight to “disappointed” since that will be very useful in determining the sentiment of the review.

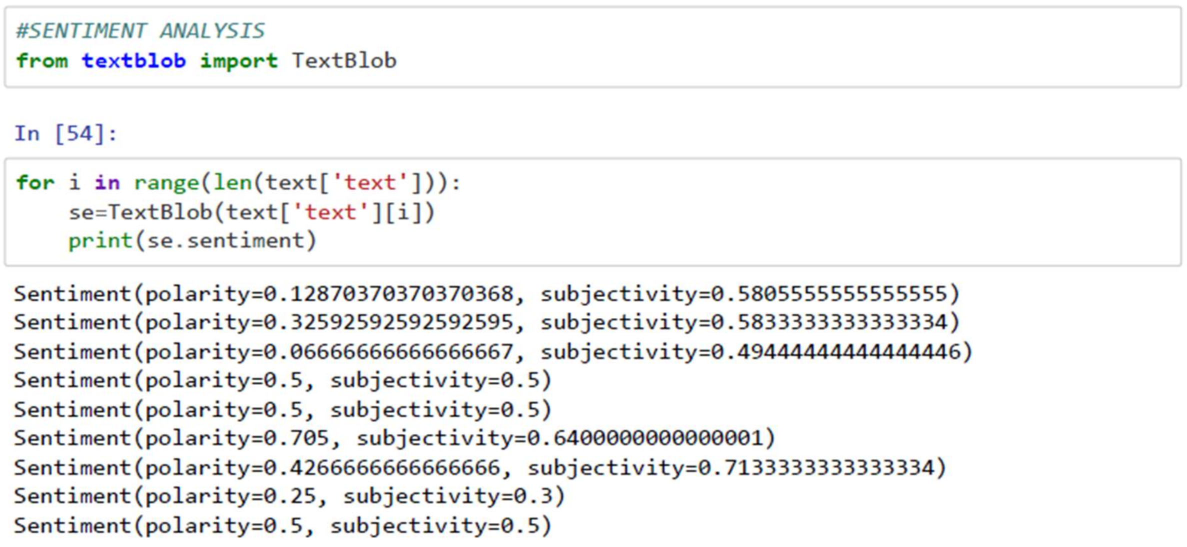
```
#WORD CLOUD
from wordcloud import WordCloud
comment_words = ' '
for words in t:
    comment_words = comment_words + words + ' '

wordcloud = WordCloud(width = 600, height = 600,
                background_color='white' ,
                stopwords = stop,
                min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (5, 5), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
```

normal
Window
used
month
fine
Install
life
feeling
product
videos
one2
Ram
around
laptop
pad
hrs
browsing
keyboard
Nice
review
brightness
Ubuntu
sturdy
fingerprint
delicate
comes
showing
work
gives
usage
good
touch
shaky
quality
Build
Battery
done
average
upgrade
setting
kind
watching
Writing
smooth
performance
ProductREAD
even
using
services

A tag cloud (word cloud, or weighted list in visual design) is a novelty visual representation of text data, typically used to depict keyword metadata (tags) on websites, or to visualize free form text. Tags are usually single words, and the importance of each tag is shown with font size or colour. This format is useful for quickly perceiving the most prominent terms to determine its relative prominence.



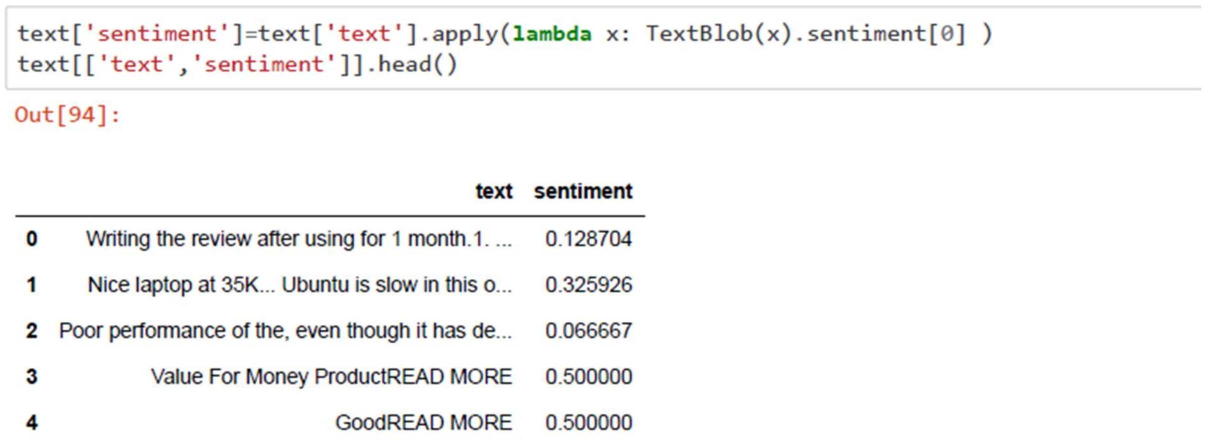
```
#SENTIMENT ANALYSIS
from textblob import TextBlob
```

In [54]:

```
for i in range(len(text['text'])):
    se=TextBlob(text['text'][i])
    print(se.sentiment)
```

```
Sentiment(polarity=0.12870370370370368, subjectivity=0.5805555555555555)
Sentiment(polarity=0.32592592592592595, subjectivity=0.5833333333333334)
Sentiment(polarity=0.06666666666666667, subjectivity=0.49444444444444446)
Sentiment(polarity=0.5, subjectivity=0.5)
Sentiment(polarity=0.5, subjectivity=0.5)
Sentiment(polarity=0.705, subjectivity=0.6400000000000001)
Sentiment(polarity=0.42666666666666666, subjectivity=0.7133333333333334)
Sentiment(polarity=0.25, subjectivity=0.3)
Sentiment(polarity=0.5, subjectivity=0.5)
```

Our problem was to detect the sentiment of the review. So, before applying any ML/DL models (which can have a separate feature detecting the sentiment using the textblob library), let’s check the sentiment of the reviews.

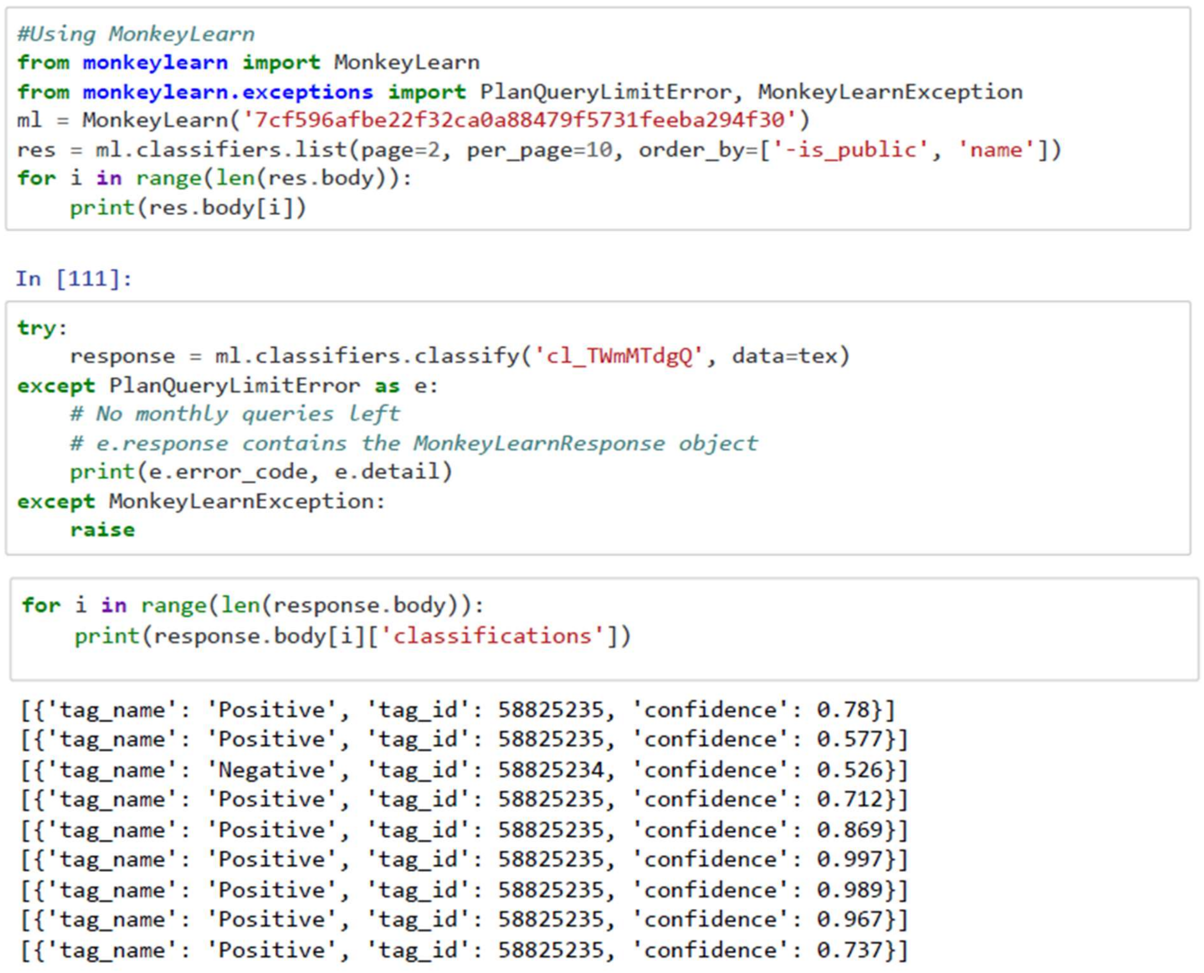


```
text['sentiment']=text['text'].apply(lambda x: TextBlob(x).sentiment[0] )
text[['text','sentiment']].head()
```

Out[94]:

	text	sentiment
0	Writing the review after using for 1 month.1. ...	0.128704
1	Nice laptop at 35K... Ubuntu is slow in this o...	0.325926
2	Poor performance of the, even though it has de...	0.066667
3	Value For Money ProductREAD MORE	0.500000
4	GoodREAD MORE	0.500000

Above, you can see that it returns a tuple representing polarity and subjectivity of each review. Here, we only extract polarity as it indicates the sentiment as value nearer to 1 means a positive sentiment and values nearer to -1 means a negative sentiment. This can also work as a feature for building a machine learning model.



```
#Using MonkeyLearn
from monkeylearn import MonkeyLearn
from monkeylearn.exceptions import PlanQueryLimitError, MonkeyLearnException
ml = MonkeyLearn('7cf596afbe22f32ca0a88479f5731feeba294f30')
res = ml.classifiers.list(page=2, per_page=10, order_by=['-is_public', 'name'])
for i in range(len(res.body)):
    print(res.body[i])
```

In [111]:

```
try:
    response = ml.classifiers.classify('cl_TWmMTdgQ', data=tex)
except PlanQueryLimitError as e:
    # No monthly queries left
    # e.response contains the MonkeyLearnResponse object
    print(e.error_code, e.detail)
except MonkeyLearnException:
    raise
```

```
for i in range(len(response.body)):
    print(response.body[i]['classifications'])
```

```
[{'tag_name': 'Positive', 'tag_id': 58825235, 'confidence': 0.78}]
[{'tag_name': 'Positive', 'tag_id': 58825235, 'confidence': 0.577}]
[{'tag_name': 'Negative', 'tag_id': 58825234, 'confidence': 0.526}]
[{'tag_name': 'Positive', 'tag_id': 58825235, 'confidence': 0.712}]
[{'tag_name': 'Positive', 'tag_id': 58825235, 'confidence': 0.869}]
[{'tag_name': 'Positive', 'tag_id': 58825235, 'confidence': 0.997}]
[{'tag_name': 'Positive', 'tag_id': 58825235, 'confidence': 0.989}]
[{'tag_name': 'Positive', 'tag_id': 58825235, 'confidence': 0.967}]
[{'tag_name': 'Positive', 'tag_id': 58825235, 'confidence': 0.737}]
```

In this section using MonkeyLearn package we can analyse our data of product reviews. For that first we want to sign up for free in MonkeyLearn webpage[7] using our E-mail id. By that we can get Your_API_Key. Using that we can easily classify our text data for product sentiment analysis.

By using that code and API we get the output of sentiment_tag and confidence level.

V. EXPERIMENTS AND RESULTS

In this experiment, we show web-scrap review of DELL Vostro 3478 Laptop from Flipkart website[8]. Product reviews were extracted using web scraping method which was implemented in Python 3(Jupyter Notebook) using BeautifulSoup library, which was implemented in Windows 10 with an Intel® Core™ i5-8th Gen. Processor (@3.00 GHz) and 4 GB RAM.

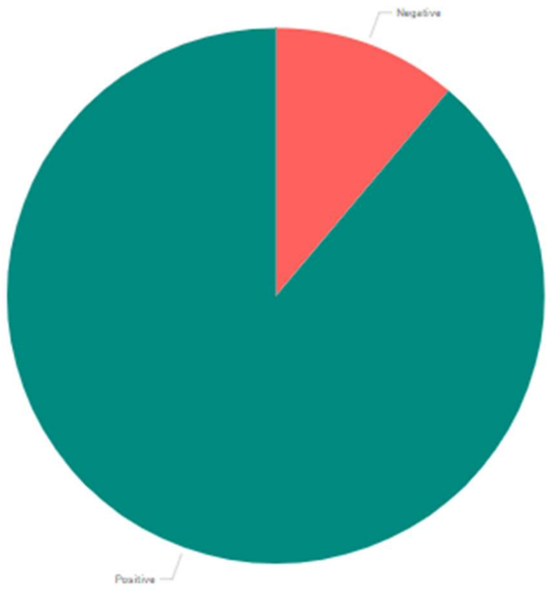


Figure. 3 Pie chart on reviews

In Figure.3, Pie chart on reviews, it shows a pie chart of the review data which is the customer review of a Dell Vostro 3478 laptop on Flipkart page. By this chat, we can conclude that for that particular product most of the customers giving positive review.

Nearly 90% of customers raise the review as positive. By the negative reviews we can propose an ideas for further manufacturing for that laptop company.

VI. CONCLUSION AND FUTURE WORK

In this experiment, we have proposed scrap review data from the Flipkart website, and makes classification based on customer reviews on particular product of DELL Vostro 3478 Laptop. In this project, we compare product reviews by MonkeyLearn API, so that decision making will be a lot easier for the user regarding purchasing of the product.

Additionally, depending on time constraints, we also plan on introducing a portal for laptop manufacturers to analyse the public perception of their product over time and identify key areas where their product can be improved.

For future work, we can implement this project on Google’s TensorFlow, which is an open source framework for machine learning. And also I have an idea to make a website for sentiment analysis by this code without MonkeyLearn API (for that we should have real time structured data for training) for text, audio data.

REFERENCES

Research Papers:

1. Mahek Merchant, Ricky Parmar (2016). Sentiment Analysis of Web Scraped Product Reviews using Hadoop. Volume 4 Issue VIII, IC Value: 13.98 ISSN: 2321-9653.
2. Pratiksha Ashiwal, S.R.Tandan, Priyanka Tripathi, Rohit Miri (2016). Web Information Retrieval Using Python and BeautifulSoup. Volume 4 Issue VI, IC Value: 13.98 ISSN: 2321-9653.
3. Devendra Kamalapurkar, Ninad Bagwe, R. Harikrishnan, Salil Shahane, Mrs. Manisha Gahirwal (2017). SENTIMENT ANALYSIS OF PRODUCT REVIEWS. ISSN: 2277-9655.

URLs:

4. https://www.datacamp.com/community/tutorials/text-analytics-beginners-nltk
5. https://www.analyticsvidhya.com/blog/2018/02/the-different-methods-deal-text-data-predictive-python/
6. https://medium.com/@bedigunjit/simple-guide-to-text-classification-nlp-using-svm-and-naive-bayes-with-python-421db3a72d34
7. https://monkeylearn.com/api/v3/?python#create-classifier

Scraped product review webpage:

8. https://www.flipkart.com/dell-vostro-14-3000-core-i5-8th-gen-4-gb-1-tb-hdd-linux-2-gb-graphics-3478-laptop/p/itmf4daar3urw6yr?pid=COMF4DAAMGHXCHXH&lid=LSTCOMF4DAAMGHXCHXHH24PRM