NAME OF THE PROJECT

Product Review Rating Using NLP

Submitted by:
Arshi Maitra

Acknowledgement:

This project has been completed with the help of training documents and live classes recordings from Data Trained Education. Few help on coding have also been taken from few data science websites like Toward Data Science, Geek for Geeks, Stack Overflow. All data related to product rating has been taken from FlipKart under Headphones, Camera, Speakers, Mobiles, Iphone and TV. I would also like to extend my gratitude to FlipRobo team who have given me this wonderful opportunity opportunity to work on this amazing dataset.

INTRODUCTION

Machine Learning (ML) is a field of Artificial Intelligence where data-driven algorithms learn patterns by getting exposed to relevant data. ML has gained massive importance in the field of Natural Language Processing (NLP), i.e. interpreting human language. In this article, we will focus on the use of both ML and NLP in predicting the ratings of user reviews.

We have a client who has a website where people write different reviews for technical products. Now they are adding a new feature to their website i.e. The reviewer will have to add stars(rating) as well with the review. The rating is out 5 stars and it only has 5 options available 1 star, 2 stars, 3 stars, 4 stars, 5 stars. Now they want to predict ratings for the reviews which were written in the past and they don't have a rating.

Therefore the main goal of this project is to determine the following:

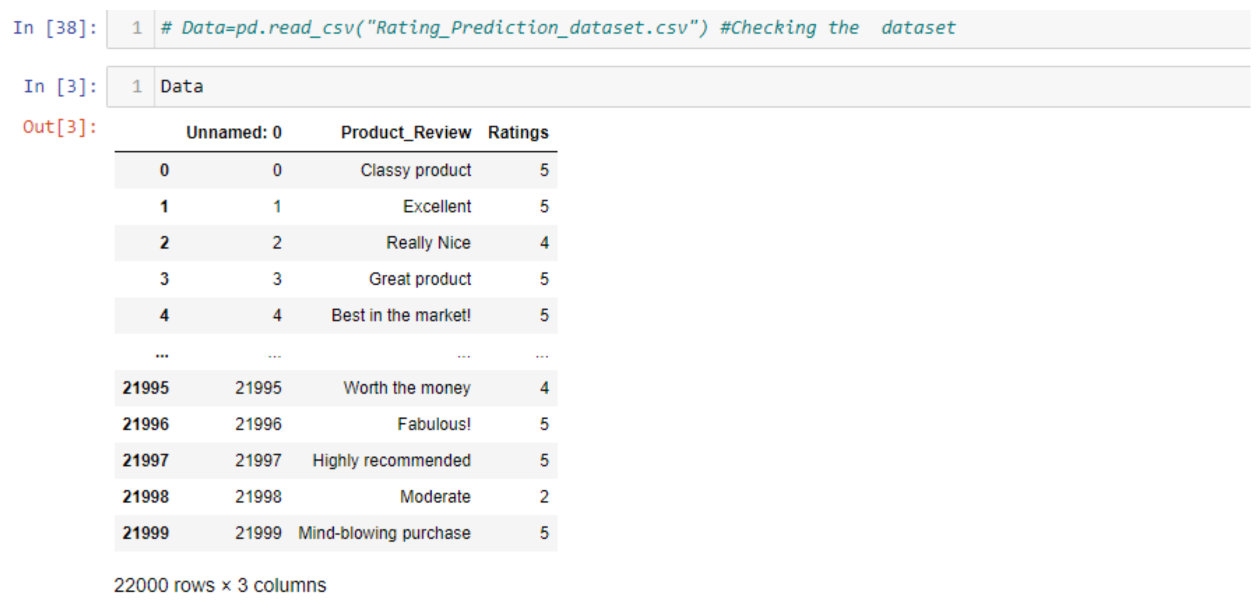
- To predict the rating of the product by looking at their review

The following data has been taken from flipkart.com. Here the ratings of various electronic products have been scraped around 22,000. We will be trying to analyze the pattern of words as well as the ratings prediction from the scraped dataset.

Analytical Problem Framing

Web Scraping

Before importing the necessary libraries for data analysis and prediction, the data has been first scrapped from the website www.flipkart.com by using the Selenium method from Web Scraping. The data is saved in CSV which is later on used for analysis:



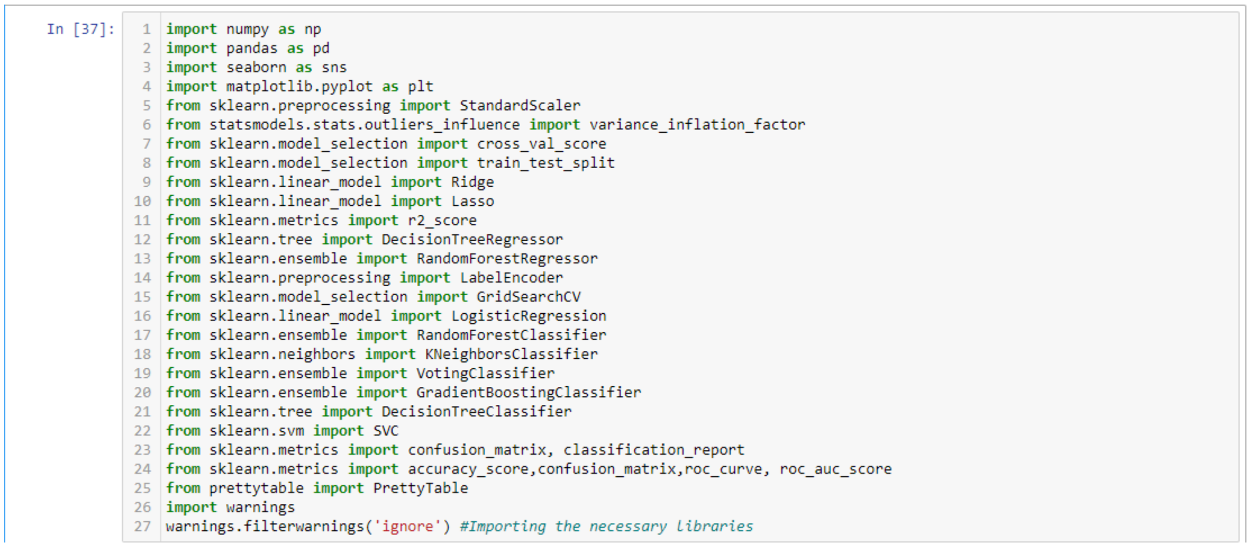
```
In [38]:  # Data=pd.read_csv("Rating_Prediction_dataset.csv") #Checking the  dataset

In [3]:  Data
Out[3]:
```

	Unnamed: 0	Product_Review	Ratings
0	0	Classy product	5
1	1	Excellent	5
2	2	Really Nice	4
3	3	Great product	5
4	4	Best in the market!	5
...	...	...	...
21995	21995	Worth the money	4
21996	21996	Fabulous!	5
21997	21997	Highly recommended	5
21998	21998	Moderate	2
21999	21999	Mind-blowing purchase	5

22000 rows × 3 columns

Importing Necessary Libraries

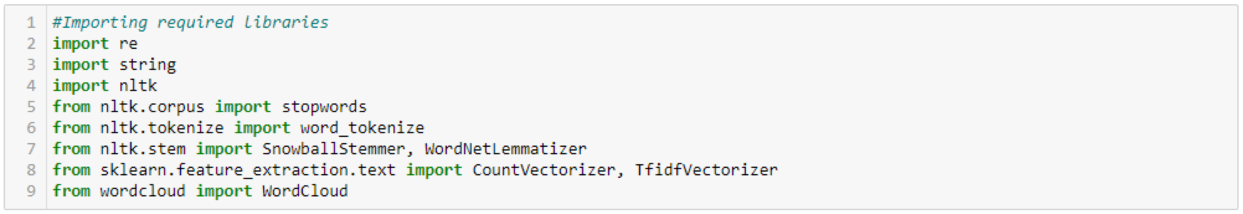
ML:



In [37]:

```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.metrics import r2_score
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import accuracy_score,confusion_matrix,roc_curve, roc_auc_score
from prettytable import PrettyTable
import warnings
warnings.filterwarnings('ignore') #Importing the necessary libraries
```

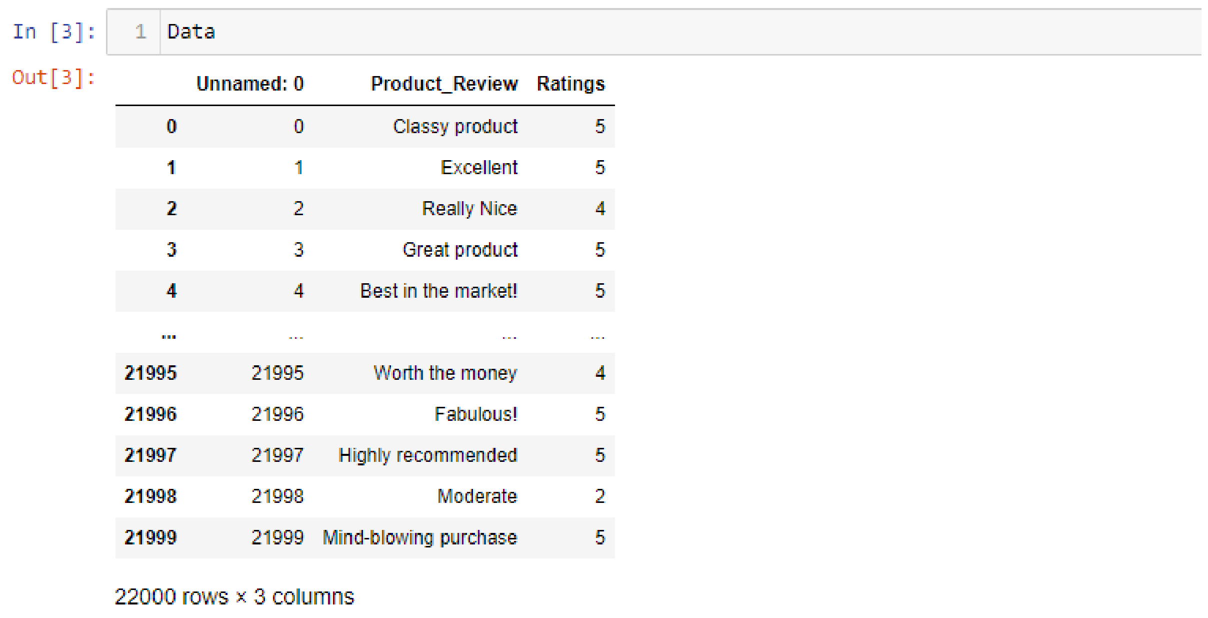
NLP:



```python
#Importing required libraries
import re
import string
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import SnowballStemmer, WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from wordcloud import WordCloud
```

Data preprocessing/Data Cleaning:

Loading the dataset



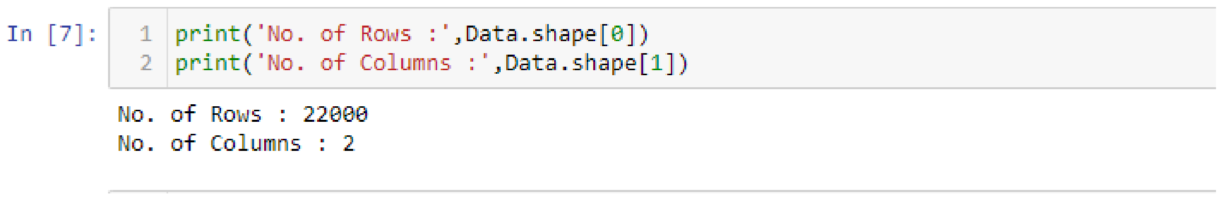
```
In [3]: Data
```

Out[3]:

	Unnamed: 0	Product_Review	Ratings
0	0	Classy product	5
1	1	Excellent	5
2	2	Really Nice	4
3	3	Great product	5
4	4	Best in the market!	5
...	...	...	...
21995	21995	Worth the money	4
21996	21996	Fabulous!	5
21997	21997	Highly recommended	5
21998	21998	Moderate	2
21999	21999	Mind-blowing purchase	5

22000 rows × 3 columns

Checking the number of rows and columns



```
In [7]: print('No. of Rows :',Data.shape[0])
        print('No. of Columns :',Data.shape[1])
```

```
No. of Rows : 22000
No. of Columns : 2
```

Checking the null values from the data:

```
In [8]:  Data.isnull().sum() # Checking the null values
```

```
Out[8]: Product_Review    0
        Ratings           0
        dtype: int64
```

Dropping the unnecessary column:

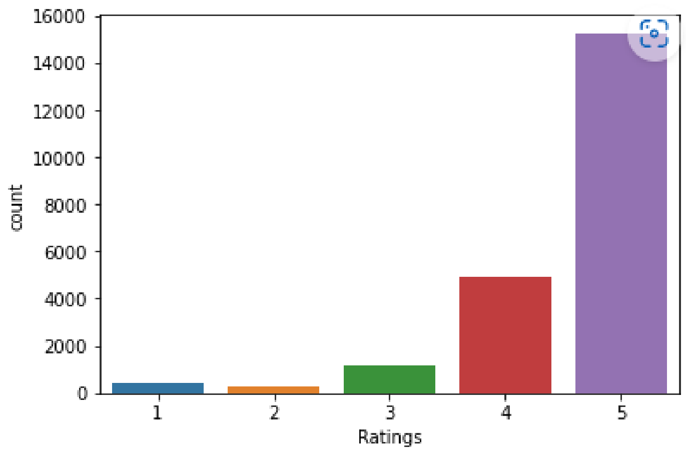
```
# Dropping unnecssary index column Unnamed:0
Data.drop('Unnamed: 0',axis=1,inplace=True)
```

Visualization:

Checking Rating Distribution:

```
sns.countplot(Data['Ratings']) # Checking the distribution of ratings
```

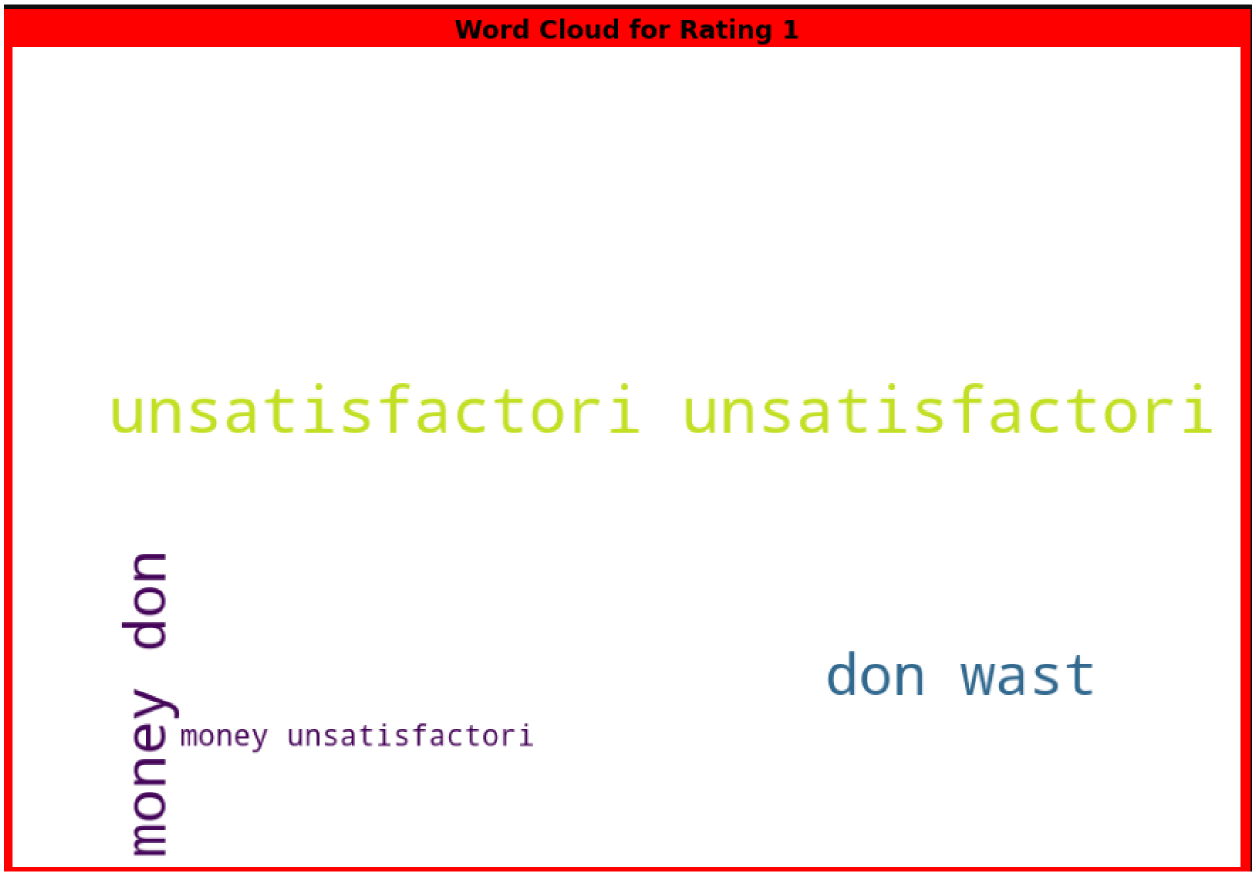
```
<AxesSubplot:xlabel='Ratings', ylabel='count'>
```

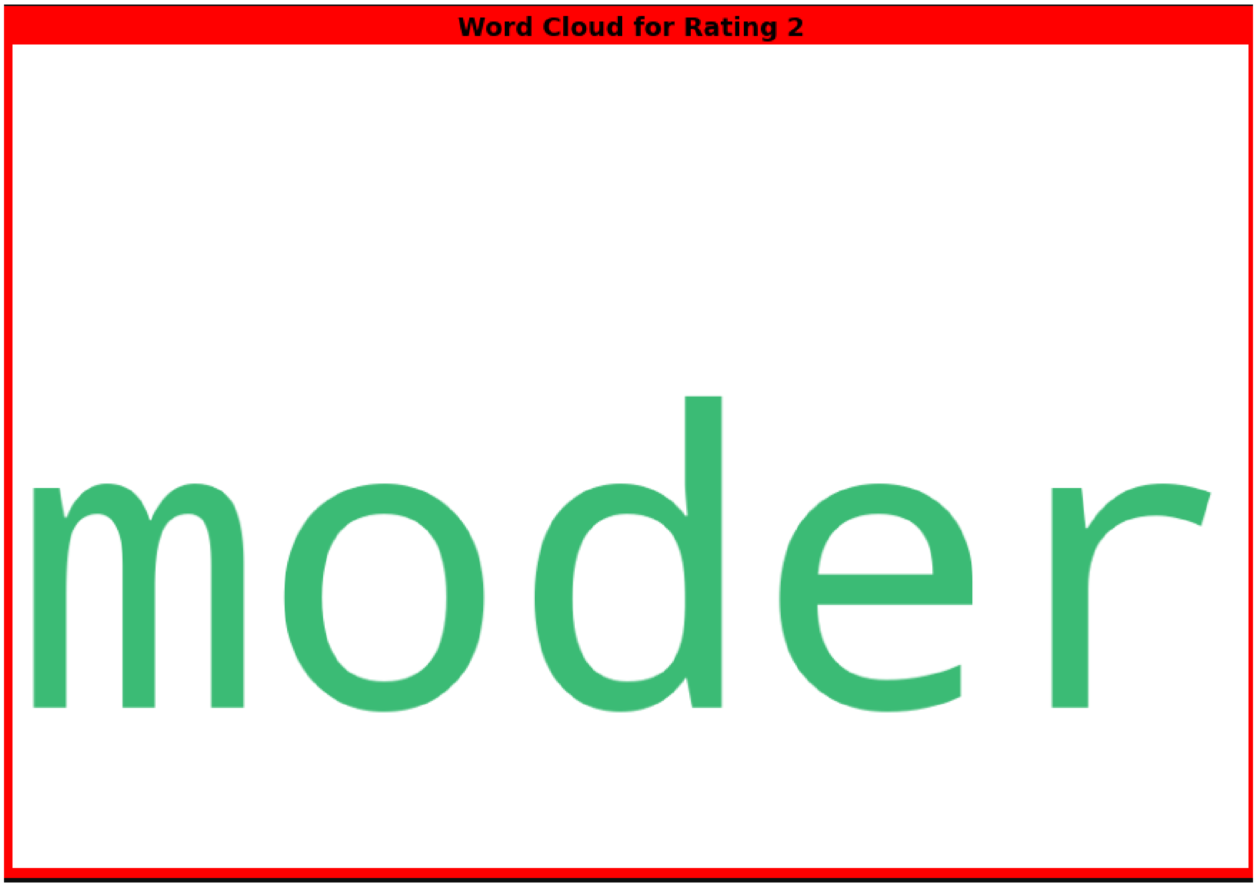


Inference: The highest number of rating is 5 while rating 2 is the lowest.

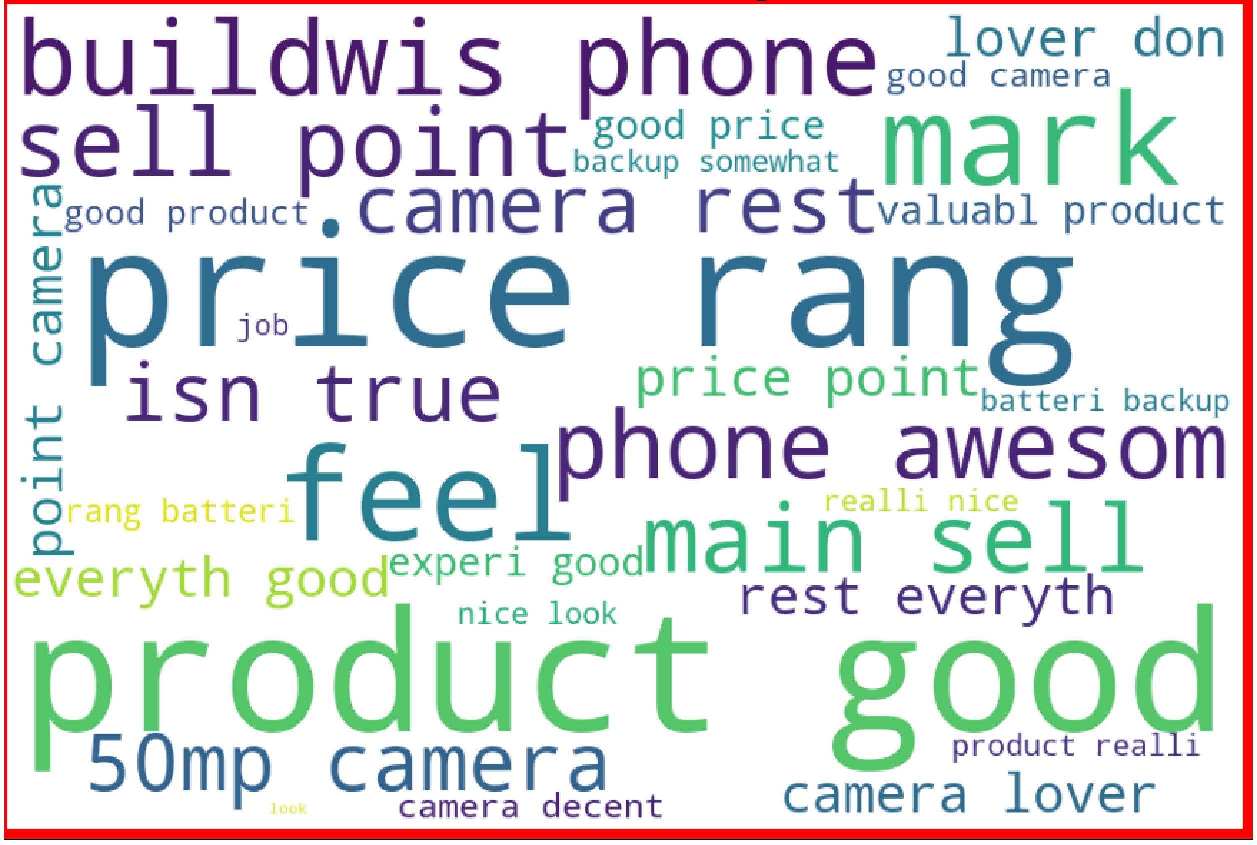
WordCloud:

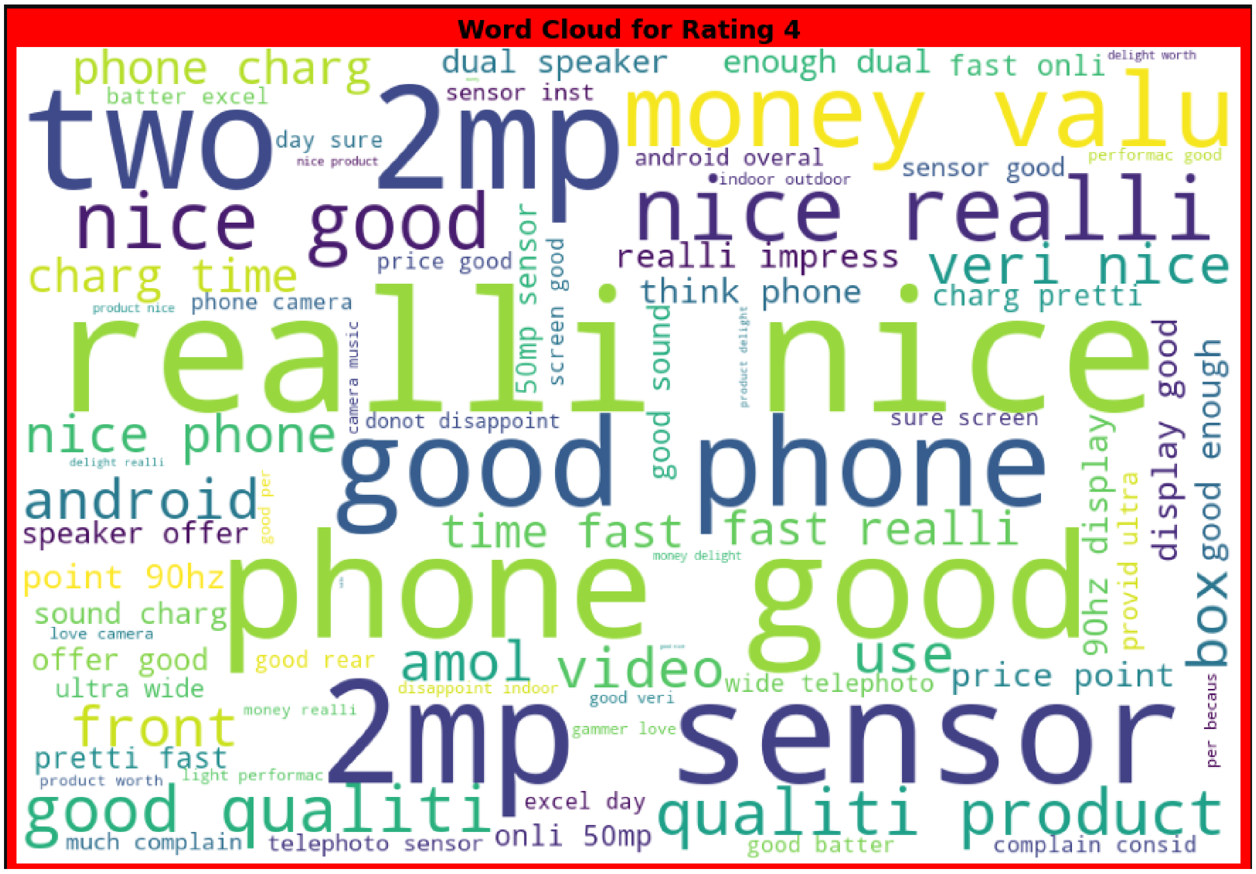
Word clouds or tag clouds are graphical representations of word frequency that give greater prominence to words that appear more frequently in a source text. The larger the word in the visual the more common the word was in the document



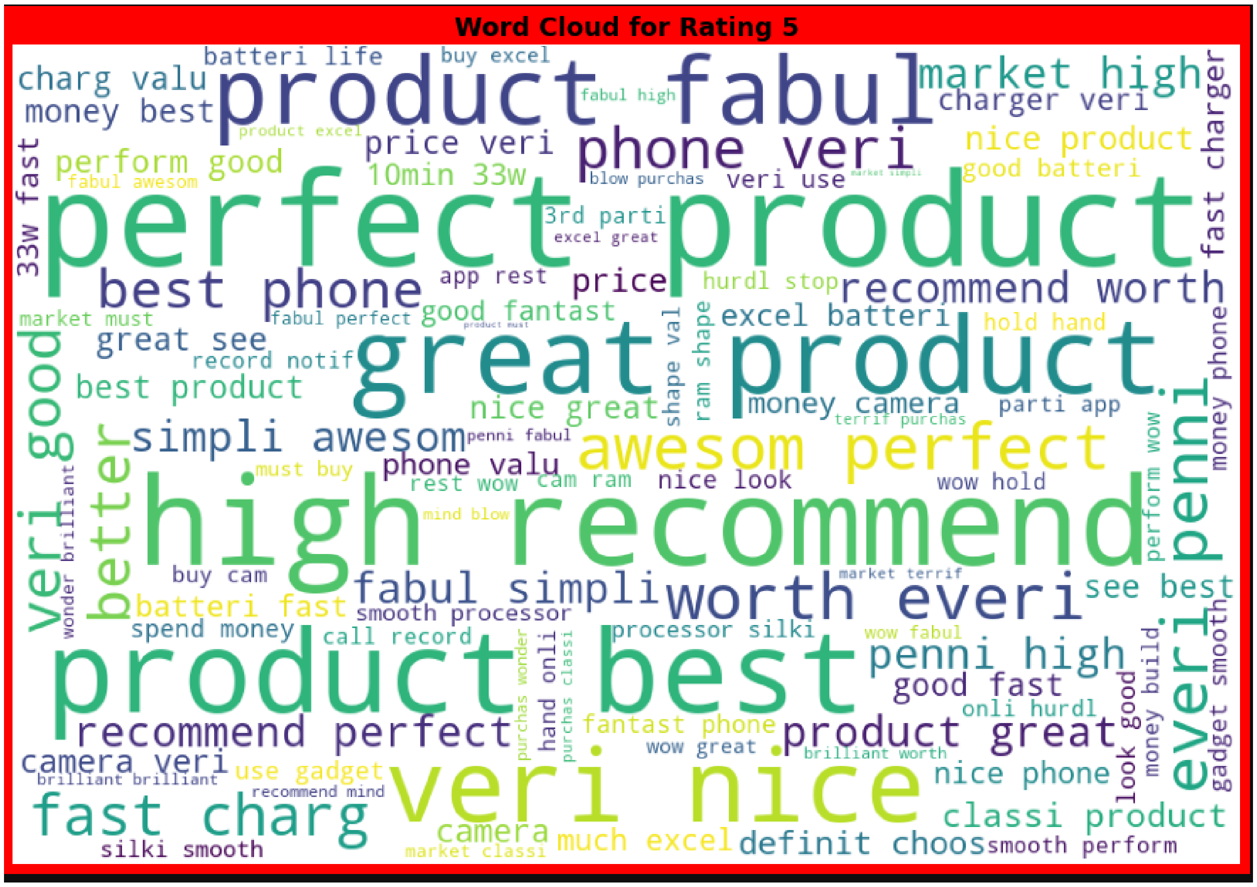

Word Cloud for Rating 2
moder

Word Cloud for Rating 3




Word Cloud for Rating 4

Word Cloud for Rating 5

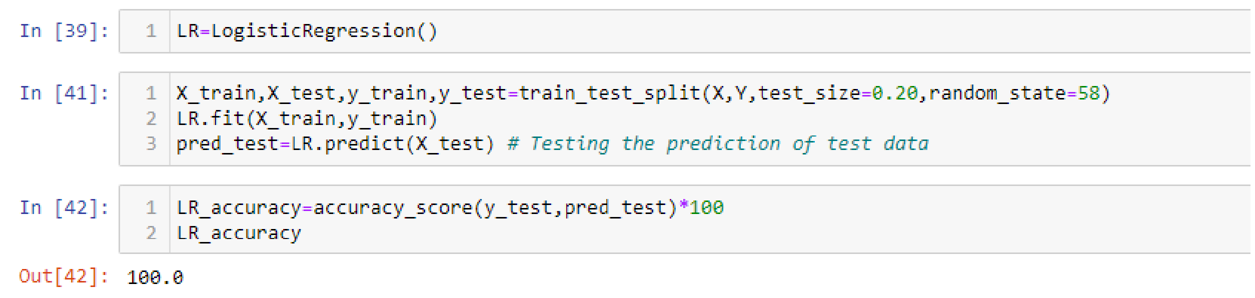


Model/s Development and Evaluation

(Logistic Regression, Decision Tree Classifier, Random Forest Classifier)

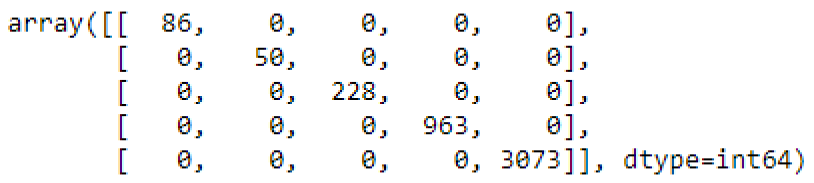
Logistic Regression:

Training and Testing the model



```
In [39]:  LR=LogisticRegression()

In [41]:  X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=0.20,random_state=58)
          LR.fit(X_train,y_train)
          pred_test=LR.predict(X_test) # Testing the prediction of test data

In [42]:  LR_accuracy=accuracy_score(y_test,pred_test)*100
          LR_accuracy

Out[42]:  100.0
```

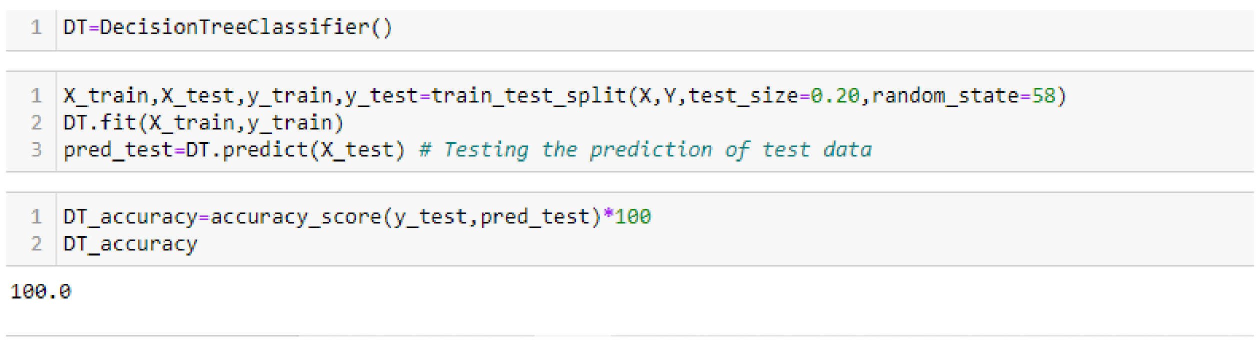
Confusion Matrix:



```
array([[  86,    0,    0,    0,    0],
       [   0,   50,    0,    0,    0],
       [   0,    0,  228,    0,    0],
       [   0,    0,    0,  963,    0],
       [   0,    0,    0,    0, 3073]], dtype=int64)
```

Decision Tree Classifier

Training and Testing the model



```
DT=DecisionTreeClassifier()

X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=0.20,random_state=58)
DT.fit(X_train,y_train)
pred_test=DT.predict(X_test) # Testing the prediction of test data

DT_accuracy=accuracy_score(y_test,pred_test)*100
DT_accuracy

100.0
```

Confusion Matrix:

```
array([[  86,    0,    0,    0,    0],
       [   0,   50,    0,    0,    0],
       [   0,    0,  228,    0,    0],
       [   0,    0,    0,  963,    0],
       [   0,    0,    0,    0, 3073]], dtype=int64)
```

Random Forest Classifier

Training and Testing the model

```
RF=RandomForestClassifier()
```

```
X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=0.20,random_state=58)
RF.fit(X_train,y_train)
pred_test=RF.predict(X_test) # Testing the prediction of test data
```

```
RF_accuracy=accuracy_score(y_test,pred_test)*100
RF_accuracy
```

```
100.0
```

Confusion Matrix:

```
array([[  86,    0,    0,    0,    0],
       [   0,   50,    0,    0,    0],
       [   0,    0,  228,    0,    0],
       [   0,    0,    0,  963,    0],
       [   0,    0,    0,    0, 3073]], dtype=int64)
```

Hyperparameter Tuning:

As all the scores came down to 100%, hence the hypertuning parameter is not performed. Random Forest Classifier is chosen since It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

Conclusion

Learning Outcomes:

1. Hands on chance to enhance web scraping skills

2. In this project, I was able to learn various NLP techniques like Stemming, Lemmatization and removal of stop words.

3. This project has stressed the importance of sampling, modeling and predicting data.

Limitations:

1. More input features could be scraped.
2. There is scope for application of advance deep learning NLP tool.

Thank You!