



## Rating Predictions

Submitted by:

Aniruddha Sawant

### **ACKNOWLEDGMENT**

Aniruddha Sawant ("I") acknowledge that I have collected data from 'Flipkart.com' and used the data and files provided by Flip Robo ("Company") namely 'Problem-Statement' to build the predictive model and have used 'sample documentation' for guidance and to write report.

# INTRODUCTION

- **Background of the Domain Problem: -**

Now a days consumers buy products with the help of ratings and reviews where both the aspects contribute equally. But it is difficult for a company who were previously not letting his consumers to write reviews and now they wanted to provide ratings also.

- **Business Problem: -**

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

- **Review of Literature: -**

There is a database where reviews of technical products are mentioned and we have label called ratings where we need to predict the ratings of the product based on its reviews. But before using reviews to predict the outcome we have collected the data, cleaned the data, transform the data into structured format and run many EDAs to make findings.

Have used total 7 machine learning models to train the data however have chosen MultinomialNB Model since its accuracy of train data and test data is high and close to each other i.e., 67.72% and 63.17% respectively.

- **Motivation for the Problem Undertaken: -**

We are required to model the ratings of reviews with the available independent variable. This model will then be used by client to rate the previously added comments or reviews. They can ultimately help consumers to get more information about the product by providing ratings. Further, the model will be a good way for the management to understand the review and rating dynamics of a new market.

# Analytical Problem Framing

## • Mathematical/ Analytical Modelling of the Problem

1. Identifying sources of the data
2. Data collection
3. Data structuring
4. Analysing the data
5. Cleaning and processing the data
6. Writing down findings and observations
7. Using different models to train the data
8. Selecting best fitted model for predictions
9. Hyper tuning the selected model
10. Predicting outcome for test data

## • Data Sources and their formats

We have used flipkart.com website to collect the data to build the model. We have used selenium to crawl the data. We have created the data set named RatingData.csv by crawling the data.

(Note: - Have crawled more than 30,000 of data but have to balance the data hence have kept only 20,000 of data with ultimate balance between ratings.)


### 1. Data source sample: - ([Sample link](#))

3★ Just okay

Review After Using 5 Days:

1. SD 750 works very well and give a very good performance
2. Fingerprint Sensor is very quick and responsive
3. Camera is not up to the mark. Many other devices in the same range perform ver
4. Charing adapter not provided in the box. Gives a bad impression. Although the d
5. Samsung Keyboard's haptic feedback is awesome. I have not tried any other key
6. Knox security present
7. ...

[READ MORE](#)

Vivek Kumar Singh  Certified Buyer, Saran District 6 months ago

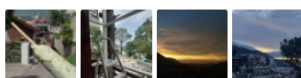
5★ Must buy!

This phone is a good choice

It is my first samsung phone and I'll say I am satisfied enough till date .

It's camera quality is good , the colour contrast is good and videos are stable but n

It's battery backup is real good lasts 2 days with 80-85% charge and Charging is als  
I'd say go with this phone !



Arjun Verma  Certified Buyer, Solan 1 month ago

## 2. Dataset sample: -

Below dataset has 1 feature i.e Reviews and 1 label i.e Rating. Dataset was being cleaned using Python and excel.

Samples: -

Cleaned data sample:-

In [6]: 1 df

Out[6]:

|       | Rating | Reviews   |
|-------|--------|---|
| 0     | 1.0    | Looks wise ok,but performance and speed wise z... |
| 1     | 1.0    | Don't buy mobile phone from Flipkart              |
| 2     | 1.0    | Camera is very poor and set is not smooth enough  |
| 3     | 1.0    | Samsung very full slow weest phone money weest... |
| 4     | 1.0    | After purchasing the phone and within a week, ... |
| ...   | ...    | ...   |
| 20087 | 5.0    | Printer is good with good quality printing and... |
| 20088 | 5.0    | Good Product                                      |
| 20089 | 5.0    | Best printer                                      |
| 20090 | 5.0    | Good,, thanks flipkart                            |
| 20091 | 5.0    | Very good brother dcp 220 photo printer photo ... |

20092 rows × 2 columns

- **Data Pre-processing**

1. Clean and organized the data.
2. Checked the data type of each column.
3. Changed the Float data type to Integer.
4. Checked whether the data has any Null Values.
5. Checked whether the data is categorical data or continuous data.
6. There are many categorical columns which has same output with different variable so used replace function to clean the data.
7. Checked the Distribution of data.
8. Processed the data and encoded reviews column to build model.

- **Data Inputs**

1. Reviews – Reviews of the products.
2. Rating – ratings to the products.

## • Hardware and Software Requirements and Tools Used

### 1. Libraries and packages used

- import numpy as np – For Numpy work
- import pandas as pd – To work on DataFrame
- import seaborn as sns – Plotting Graphs
- import matplotlib.pyplot as plt - Plotting Graphs
- import pickle – To save the Model
- from sklearn.model\_selection import train\_test\_split – To split the data into train and test.
- from sklearn.model\_selection import GridSearchCV – to search for hyperparameter
- from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix – accuracy metrics
- from sklearn import metrics.
- import warnings, warnings.filterwarnings('ignore') – To ignore unwanted Warnings

### 2. Machine Learning models used

- from sklearn.ensemble import RandomForestClassifier
- from sklearn.linear\_model import LogisticRegression
- from sklearn.ensemble import GradientBoostingClassifier
- from sklearn.tree import DecisionTreeClassifier
- from sklearn.neighbors import KNeighborsClassifier
- from sklearn.svm import SVC
- from sklearn.naive\_bayes import MultinomialNB

### 3. NPL Libraries

- import nltk
- import string
- from nltk.corpus import stopwords
- from nltk.tokenize import word\_tokenize
- from nltk.stem import PorterStemmer, WordNetLemmatizer
- from wordcloud import WordCloud
- from sklearn.feature\_extraction.text import TfidfVectorizer

### 4. Hardware used – 11th Gen Intel(R) Core (TM) i3-1115G4 @ 3.00GHz 3.00 GHz with 8.00 GB RAM and Windows 11

### 5. Software used – Anaconda and Jupyter Notebook to build the model and Excel to clean and structured the data.

# Model/s Development and Evaluation

- **Identification of possible problem**

1. The data was not structured and organized hence cleaned the data using various data cleaning and pre-processing techniques.
2. Converted a collection of raw documents to a matrix.

- **Testing of Identified Approaches**

These are the algorithms which have been used to train and test data.

1. RandomForestClassifier
2. LogisticRegression
3. GradientBoostingClassifier
4. DecisionTreeClassifier
5. KNeighborsClassifier
6. SVC
7. MultinomialNB



- Run and evaluate selected models

## 1. LogisticRegression: -

```
1 x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.25,random_state = 141)
2 log = LogisticRegression()
3 log.fit(x_train,y_train)
4
5 print_score(log,x_train,x_test,y_train,y_test, train=True)
6 print_score(log,x_train,x_test,y_train,y_test, train=False)
```

=====Train Result=====

Accuracy Score: 71.63%

=====Test Result=====

Accuracy Score: 68.07%

Test Classification Report

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 1            | 0.73      | 0.84   | 0.78     | 1073    |
| 2            | 0.63      | 0.44   | 0.52     | 606     |
| 3            | 0.59      | 0.67   | 0.63     | 1036    |
| 4            | 0.68      | 0.53   | 0.60     | 991     |
| 5            | 0.72      | 0.79   | 0.76     | 1317    |
| accuracy     |           |        | 0.68     | 5023    |
| macro avg    | 0.67      | 0.65   | 0.66     | 5023    |
| weighted avg | 0.68      | 0.68   | 0.67     | 5023    |

## 2. DecisionTreeClassifier: -

```
1 x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.25,random_state = 193)
2 dtc = DecisionTreeClassifier()
3 dtc.fit(x_train,y_train)
4
5 print_score(dtc,x_train,x_test,y_train,y_test, train=True)
6 print_score(dtc,x_train,x_test,y_train,y_test, train=False)
```

=====Train Result=====

Accuracy Score: 85.36%

=====Test Result=====

Accuracy Score: 72.19%

Test Classification Report

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 1            | 0.77      | 0.81   | 0.79     | 1034    |
| 2            | 0.69      | 0.61   | 0.65     | 620     |
| 3            | 0.66      | 0.78   | 0.72     | 1055    |
| 4            | 0.73      | 0.60   | 0.65     | 1050    |
| 5            | 0.75      | 0.77   | 0.76     | 1264    |
| accuracy     |           |        | 0.72     | 5023    |
| macro avg    | 0.72      | 0.71   | 0.71     | 5023    |
| weighted avg | 0.72      | 0.72   | 0.72     | 5023    |

### 3. MultinomialNB: -

```
1 x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.25,random_state = 2)
2 multiNB = MultinomialNB()
3 multiNB.fit(x_train,y_train)
4
5 print_score(multiNB,x_train,x_test,y_train,y_test, train=True)
6 print_score(multiNB,x_train,x_test,y_train,y_test, train=False)
```

=====Train Result=====

Accuracy Score: 65.21%

=====Test Result=====

Accuracy Score: 63.53%

Test Classification Report

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 1            | 0.70      | 0.86   | 0.77     | 1083    |
| 2            | 0.84      | 0.16   | 0.26     | 543     |
| 3            | 0.54      | 0.63   | 0.58     | 1033    |
| 4            | 0.71      | 0.38   | 0.50     | 1031    |
| 5            | 0.62      | 0.85   | 0.71     | 1333    |
| accuracy     |           |        | 0.64     | 5023    |
| macro avg    | 0.68      | 0.58   | 0.57     | 5023    |
| weighted avg | 0.66      | 0.64   | 0.61     | 5023    |

### 4. RandomForestClassifier: -

```
1 x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.25,random_state = 69)
2 rfc = RandomForestClassifier()
3 rfc.fit(x_train,y_train)
4
5 print_score(rfc,x_train,x_test,y_train,y_test, train=True)
6 print_score(rfc,x_train,x_test,y_train,y_test, train=False)
```

=====Train Result=====

Accuracy Score: 85.35%

=====Test Result=====

Accuracy Score: 74.24%

Test Classification Report

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 1            | 0.80      | 0.87   | 0.83     | 1059    |
| 2            | 0.82      | 0.60   | 0.69     | 569     |
| 3            | 0.66      | 0.80   | 0.72     | 1045    |
| 4            | 0.74      | 0.56   | 0.64     | 1014    |
| 5            | 0.75      | 0.80   | 0.77     | 1336    |
| accuracy     |           |        | 0.74     | 5023    |
| macro avg    | 0.75      | 0.73   | 0.73     | 5023    |
| weighted avg | 0.75      | 0.74   | 0.74     | 5023    |

### 5. GradientBoostingClassifier: -

```
1 x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.25,random_state = 51)
2 gbdt = GradientBoostingClassifier()
3 gbdt.fit(x_train,y_train)
4
5 print_score(gbdt,x_train,x_test,y_train,y_test, train=True)
6 print_score(gbdt,x_train,x_test,y_train,y_test, train=False)
```

=====Train Result=====

Accuracy Score: 66.20%

=====Test Result=====

Accuracy Score: 64.24%

Test Classification Report

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 1            | 0.73      | 0.79   | 0.76     | 1062    |
| 2            | 0.56      | 0.40   | 0.47     | 587     |
| 3            | 0.55      | 0.63   | 0.59     | 1030    |
| 4            | 0.63      | 0.51   | 0.56     | 1025    |
| 5            | 0.68      | 0.75   | 0.71     | 1319    |
| accuracy     |           |        | 0.64     | 5023    |
| macro avg    | 0.63      | 0.61   | 0.62     | 5023    |
| weighted avg | 0.64      | 0.64   | 0.64     | 5023    |

## 6. KNeighborsClassifier: -

```
1 x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.25,random_state = 23)
2 knn = KNeighborsClassifier()
3 knn.fit(x_train,y_train)
4
5 print_score(knn,x_train,x_test,y_train,y_test, train=True)
6 print_score(knn,x_train,x_test,y_train,y_test, train=False)
```

=====Train Result=====

Accuracy Score: 69.26%

=====Test Result=====

Accuracy Score: 60.98%

Test Classification Report

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 1            | 0.72      | 0.72   | 0.72     | 1077    |
| 2            | 0.47      | 0.44   | 0.46     | 588     |
| 3            | 0.51      | 0.70   | 0.59     | 989     |
| 4            | 0.61      | 0.50   | 0.55     | 1026    |
| 5            | 0.69      | 0.62   | 0.65     | 1343    |
| accuracy     |           |        | 0.61     | 5023    |
| macro avg    | 0.60      | 0.59   | 0.59     | 5023    |
| weighted avg | 0.62      | 0.61   | 0.61     | 5023    |

## 7. Support Vector Classifier: -

```
1 x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.25,random_state = 51)
2 svc = SVC()
3 svc.fit(x_train,y_train)
4
5 print_score(svc,x_train,x_test,y_train,y_test, train=True)
6 print_score(svc,x_train,x_test,y_train,y_test, train=False)
```

=====Train Result=====

Accuracy Score: 81.23%

=====Test Result=====

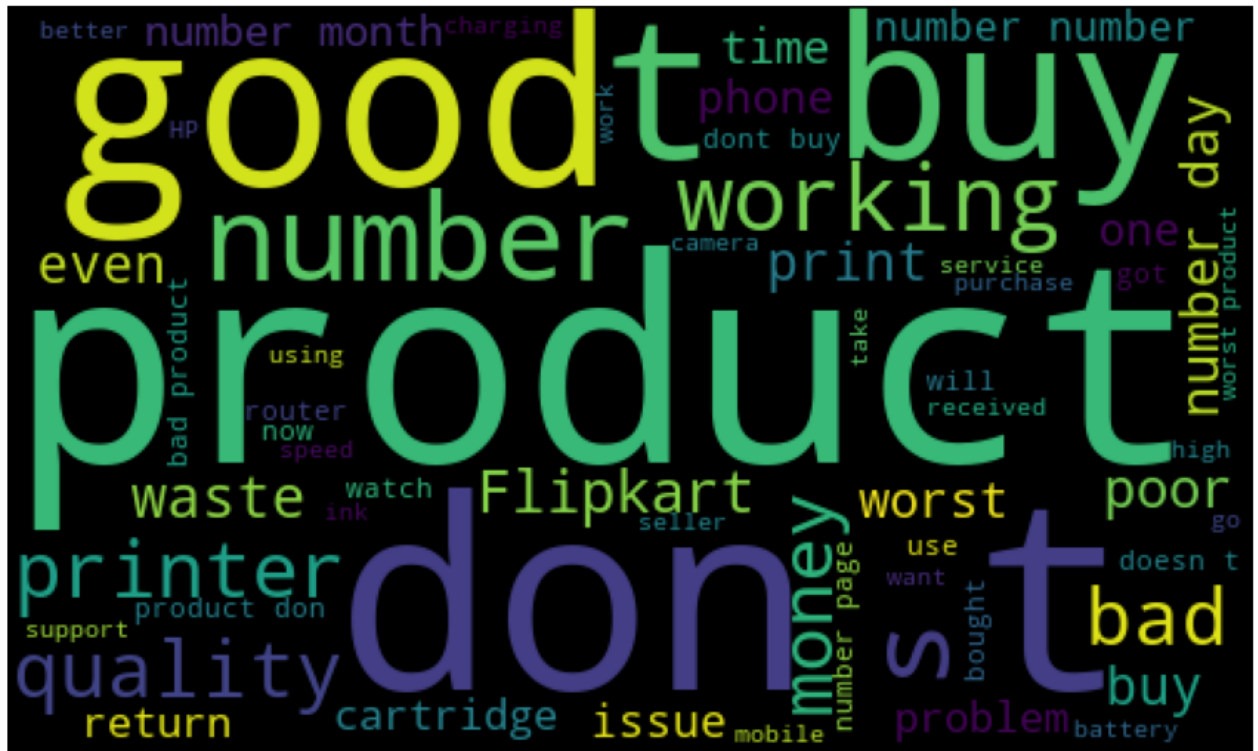
Accuracy Score: 73.46%

Test Classification Report

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 1            | 0.78      | 0.89   | 0.83     | 1062    |
| 2            | 0.80      | 0.53   | 0.64     | 587     |
| 3            | 0.64      | 0.76   | 0.69     | 1030    |
| 4            | 0.79      | 0.56   | 0.66     | 1025    |
| 5            | 0.73      | 0.82   | 0.77     | 1319    |
| accuracy     |           |        | 0.73     | 5023    |
| macro avg    | 0.75      | 0.71   | 0.72     | 5023    |
| weighted avg | 0.74      | 0.73   | 0.73     | 5023    |

- **Visualizations**

## 1. WordCloud: -



- **Interpretation of the Results**

Below is the list of highly influencing words to predict Ratings.

1. Buy
2. Don't
3. Waste
4. Working
5. Quality
6. Issue
7. Money
8. Bad
9. Good
10. Number

## CONCLUSION

- **Key Findings and Conclusions of the Study**

1. Selecting MultinomialNB model since the Accuracy score i.e., 67.72 % and test scores i.e., 63.17 % are greater and close to each other.
2. Precision and recall is 63%.
3. Result after Hyper Parameter Tuning
  - Previous Test Accuracy Score 63.53% and New Test Accuracy score 63.17%
  - Previous Train Accuracy Score 65.21% and New Train Accuracy score 67.72%

- **Learning Outcomes of the Study in respect of Data Science**

1. Data Cleaning helps to convert unorganized and unstructured data into structured data which will be used to make findings.
2. Data visualization helps understand and analyse the data.
3. Model building helps to predict outcomes, in this case MultinomialNB model fits perfect for this dataset.

- **Limitations of this work and Scope for Future Work**

1. In many cases there are words which are used in many ratings like below words are used in Rating 3,4 and 5 [Good quality]
2. It is necessary to keep an eye on new and updated data to further train the model and make decision as per new data.