FLIP ROBO

# Rating Prediction

Submitted by:

Lahari S.K

## ACKNOWLEDGMENT

I would like to thank each and every one in completion of this project.

## INTRODUCTION

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 flipkart.

Domain knowledge NLP,Machine Learning and Neural Networks can be advantageous for predicting the ratings. One can make use of nlp for the reprocessing of the data and has to choose between Machine Learning or the Neural Networks for building the model and the evaluation.

**Importance of reviews and ratings in business**

- Ratings and reviews impact everything from sales to SEO.
- Ratings and reviews help shoppers make informed, confident purchases.
- Reviews improve your brand’s trustworthiness.
- Reviews contain insights about products, processes, and purchasers.
- Better Understanding of Customers & Improve Customer Service.
- Credibility & Social Proof.
- Fight with experience to save margins.
- Allow Consumers to Have a Voice and Create Customer Loyalty.
- Improve Rankings
- Consumers are Doing your Marketing for Business.
- Reviews Generate More Reviews

## Analytical Problem Framing

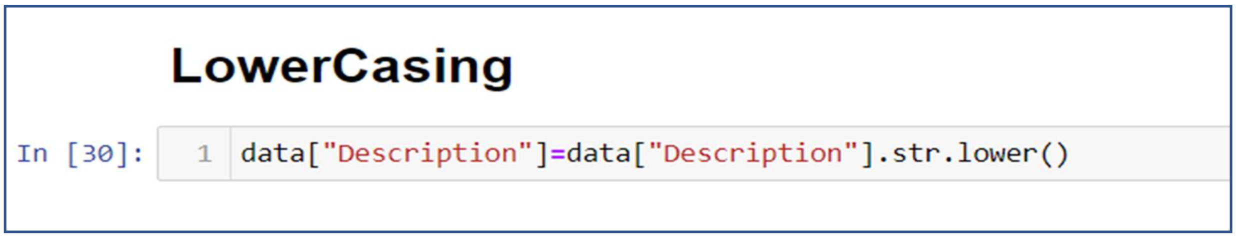
Data is collected from www.flipkat.com through web scraping.The names of the products from which reviews, description and ratings are scrapped are as follows

- Laptops
- Mobiles
- Headphones
- Camera
- Printer
- Router
- Home Theatre
- Monitor
- SmartWatches

Information was scrapped using selenium. In and around 94309 rows were scrapped and saved as separate files in the form of csv.

## Data Preprocessing Done

1. **LowerCase:**
   Converting the texts into lower case so that uniformity is maintained.

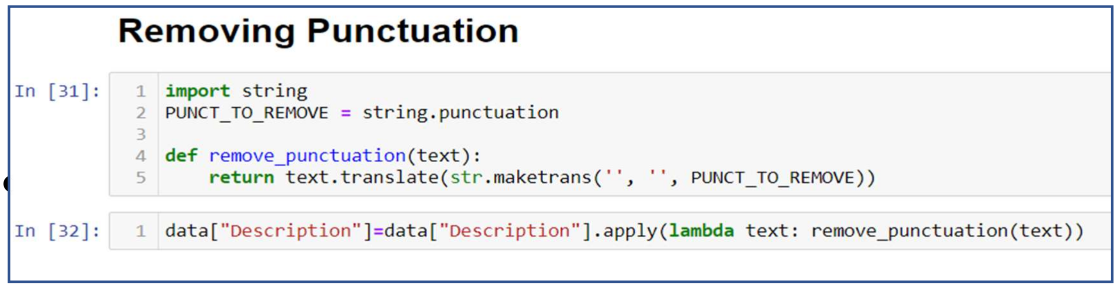


**LowerCasing**

```
In [30]:  data["Description"]=data["Description"].str.lower()
```

2.**Removing punctuation:**

Punctuations just adds on to the length of the text so they are removed using

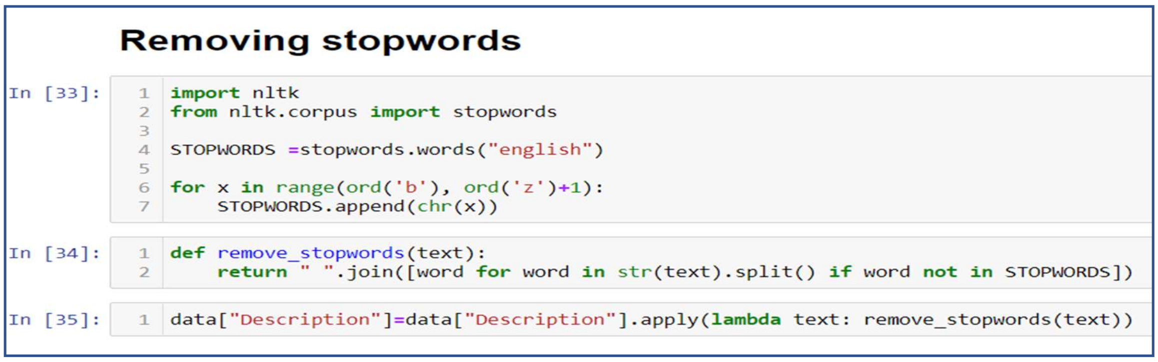


**Removing Punctuation**

```
In [31]:  import string
          PUNCT_TO_REMOVE = string.punctuation

          def remove_punctuation(text):
              return text.translate(str.maketrans('', '', PUNCT_TO_REMOVE))

In [32]:  data["Description"]=data["Description"].apply(lambda text: remove_punctuation(text))
```

3.**Removing Stopwords:**

Stopwords are removed .



## Removing stopwords

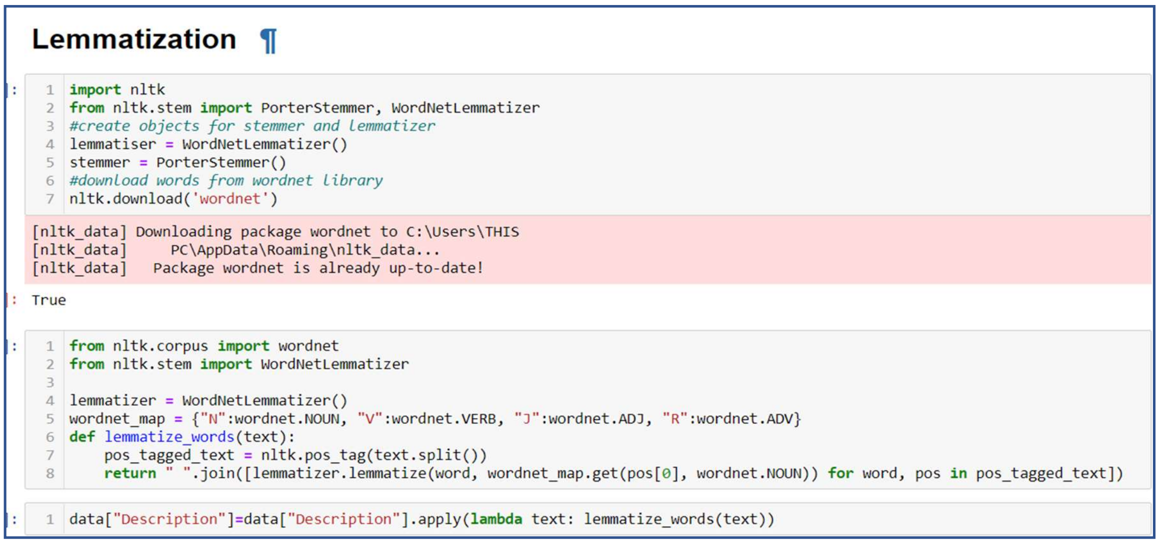
```
In [33]:
import nltk
from nltk.corpus import stopwords

STOPWORDS =stopwords.words("english")

for x in range(ord('b'), ord('z')+1):
    STOPWORDS.append(chr(x))
```

```
In [34]:
def remove_stopwords(text):
    return " ".join([word for word in str(text).split() if word not in STOPWORDS])
```

```
In [35]:
data["Description"]=data["Description"].apply(lambda text: remove_stopwords(text))
```

4.**Lemmatization**:

Wordnetlemmatizer and porter stemmers are used to  bring the text to the base level.



## Lemmatization ¶

```
]:
import nltk
from nltk.stem import PorterStemmer, WordNetLemmatizer
#create objects for stemmer and lemmatizer
lemmatiser = WordNetLemmatizer()
stemmer = PorterStemmer()
#download words from wordnet library
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to C:\Users\THIS
[nltk_data]     PC\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

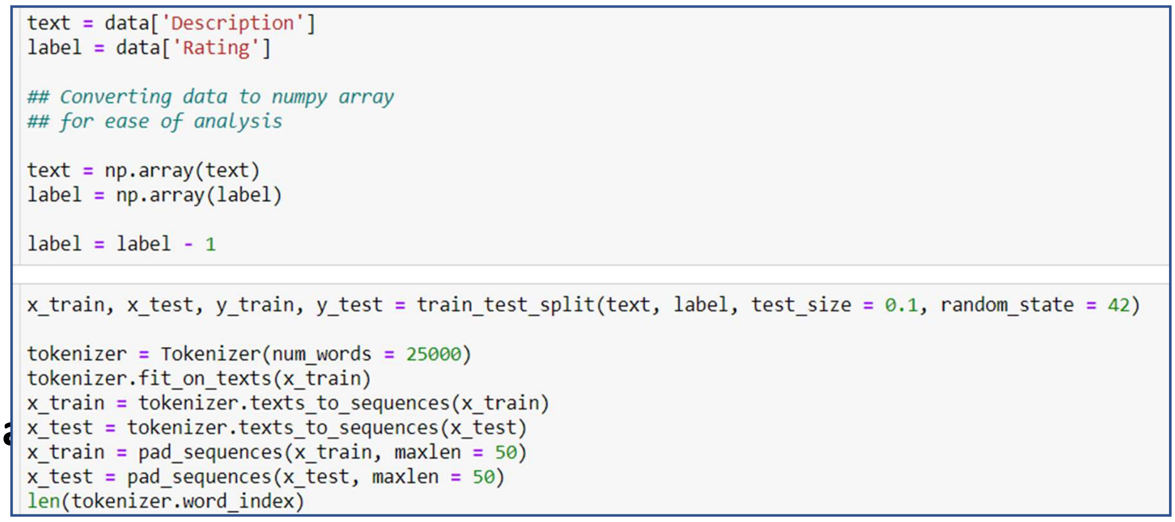
```
]: True
```

```
]:
from nltk.corpus import wordnet
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
wordnet_map = {"N":wordnet.NOUN, "V":wordnet.VERB, "J":wordnet.ADJ, "R":wordnet.ADV}
def lemmatize_words(text):
    pos_tagged_text = nltk.pos_tag(text.split())
    return " ".join([lemmatizer.lemmatize(word, wordnet_map.get(pos[0], wordnet.NOUN)) for word, pos in pos_tagged_text])
```

```
]:
data["Description"]=data["Description"].apply(lambda text: lemmatize_words(text))
```

5.**Tokenization**:

The texts are converted into vectors.



```
text = data['Description']
label = data['Rating']

## Converting data to numpy array
## for ease of analysis

text = np.array(text)
label = np.array(label)

label = label - 1
```

```
x_train, x_test, y_train, y_test = train_test_split(text, label, test_size = 0.1, random_state = 42)

tokenizer = Tokenizer(num_words = 25000)
tokenizer.fit_on_texts(x_train)
x_train = tokenizer.texts_to_sequences(x_train)
x_test = tokenizer.texts_to_sequences(x_test)
x_train = pad_sequences(x_train, maxlen = 50)
x_test = pad_sequences(x_test, maxlen = 50)
len(tokenizer.word_index)
```

**Ha**

**Hardware Requirements**

1. Windows 10
2. 64 bit Operating System
3. 8GB RAM
4. 5 GB free disk space

**Software Requirements**

1. import selenium
2. import pandas as pd
3. from selenium import webdriver
4. from selenium.common.exceptions import NoSuchElementException
5. import time
6. import numpy as np
7. import pandas as pd
8. import seaborn as sns
9. import warnings
10. warnings.filterwarnings('ignore')
11. from wordcloud import WordCloud
12. import matplotlib.pyplot as plt
13. import nltk
14. from nltk.corpus import stopwords
15. from collections import Counter
16. from nltk.stem import PorterStemmer, WordNetLemmatizer
17. from nltk.corpus import wordnet
18. from nltk.stem import WordNetLemmatizer
19. import tensorflow as tf
20. import numpy as np
21. from tensorflow.keras.preprocessing.text import Tokenizer
22. from tensorflow.keras.preprocessing.sequence import pad_sequences
23. from sklearn.model_selection import train_test_split

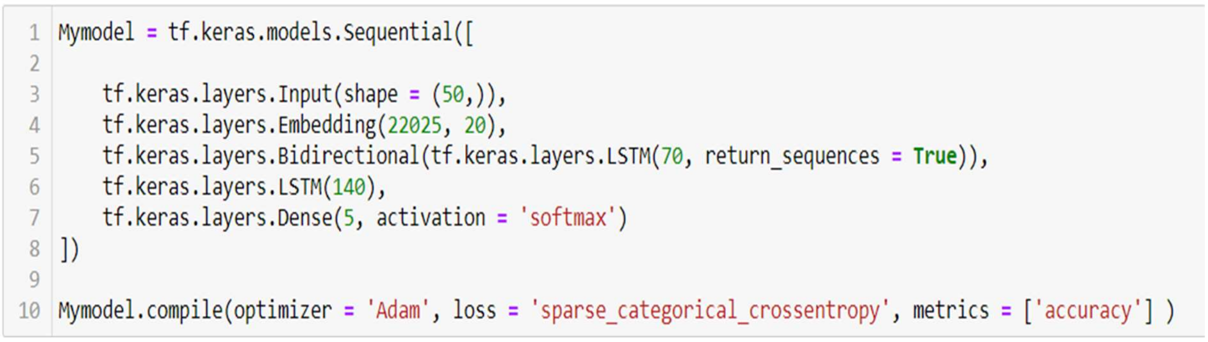
# Model/s Development and Evaluation

## Identification of possible problem-solving approaches

**1.Machine Learning**: Text classification is a machine learning technique that automatically assigns tags or categories to text. Using natural language processing (NLP), text classifiers can analyse and sort text by sentiment, topic, and customer intent – faster and more accurately than humans.

**2.Neual Network**: Deep Learning can be better on text classification that simpler ML techniques, but only on very large datasets and well designed/tuned models. The Bidirectional wrapper is used with a LSTM layer, this propagates the input forwards and backwards through the LSTM layer and then concatenates the outputs. This helps LSTM to learn long term dependencies. We then fit it to a dense neural network to do classification

## Testing of Identified Approaches (Algorithms)



```
Mymodel = tf.keras.models.Sequential([

    tf.keras.layers.Input(shape = (50,)),
    tf.keras.layers.Embedding(22025, 20),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(70, return_sequences = True)),
    tf.keras.layers.LSTM(140),
    tf.keras.layers.Dense(5, activation = 'softmax')
])

Mymodel.compile(optimizer = 'Adam', loss = 'sparse_categorical_crossentropy', metrics = ['accuracy'] )
```

**1.tf.keras.models.Sequential:** A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor.

**2.tf.keras.layers.Input(shape=()):** Allow us to build a Keras model just by knowing the inputs mentioning its shape.

**3.tf.keras.layers.Embedding():** The model will take as input an integer matrix of size (batch, input_length), and the largest integer (i.e. word index) in the input.

**4.tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(()):** Bidirectional wrapper for RNNs.

**5.tf.keras.layers.LSTM :** Long Short-Term Memory layer.

**6.tf.keras.layers.Dense():** Dense implements the operation: output = activation(dot(input, kernel) + bias) where activation is the element-wise activation function passed as the activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if use_bias is True). These are all attributes of Dense.

**7.activation:softmax :** The softmax function is used as the activation function in the output layer of neural network models that predict a multinomial probability distribution.

## Key Metrics for success in solving problem under consideration

**1.optimizer:** Adam is a replacement optimization algorithm for stochastic gradient descent for training deep learning models. Adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems**.**

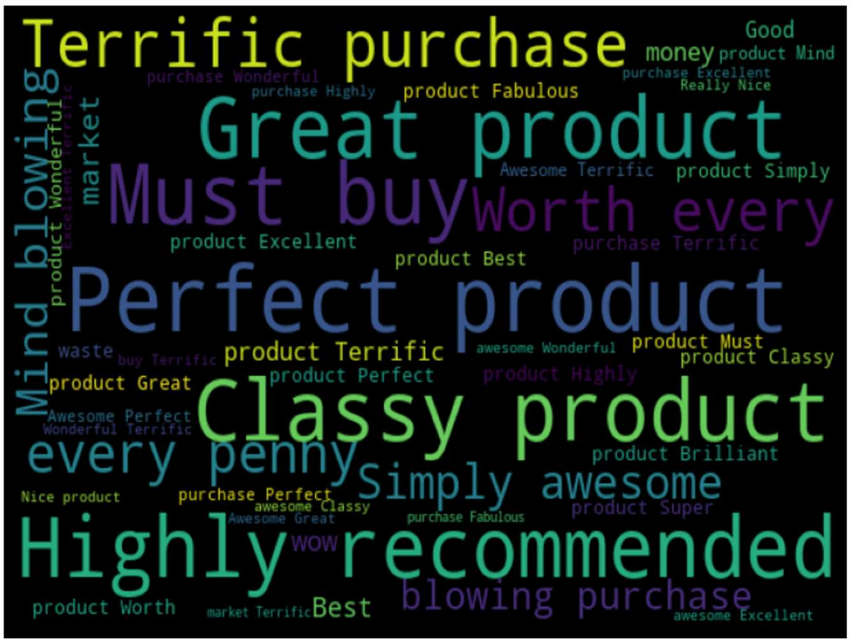
**2.loss :** Produces a category index of the most likely matching category**.**

**3.metric:** This metric creates two local variables, total and count that are used to compute the frequency with which y_pred matches y_true. This frequency is ultimately returned as binary accuracy: an idempotent operation that simply divides total by count**.**

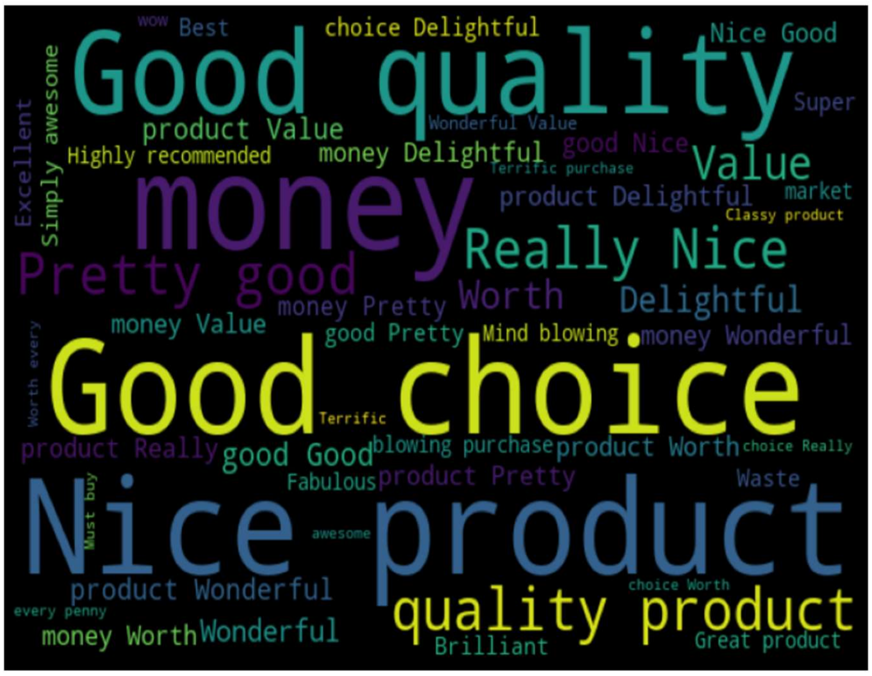
**The model is having accuracy of 0.65 with 1.48 loss.**

## Visualizations

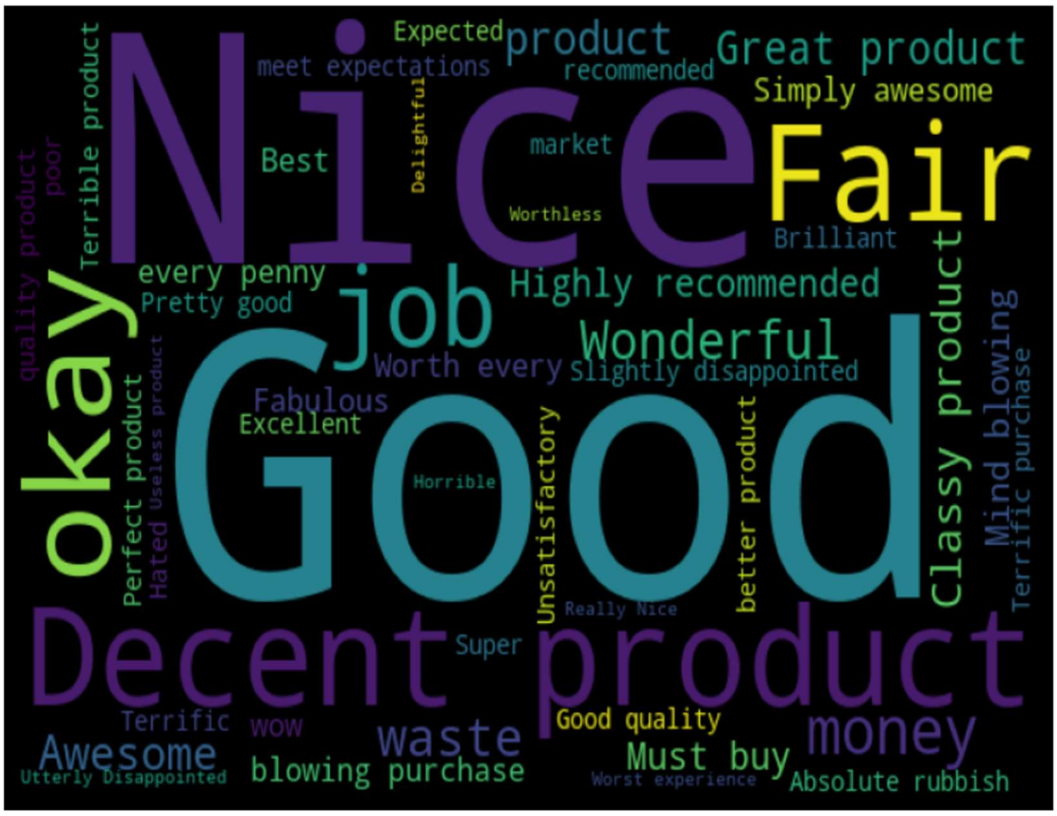
### 1.Word Cloud : Rating5



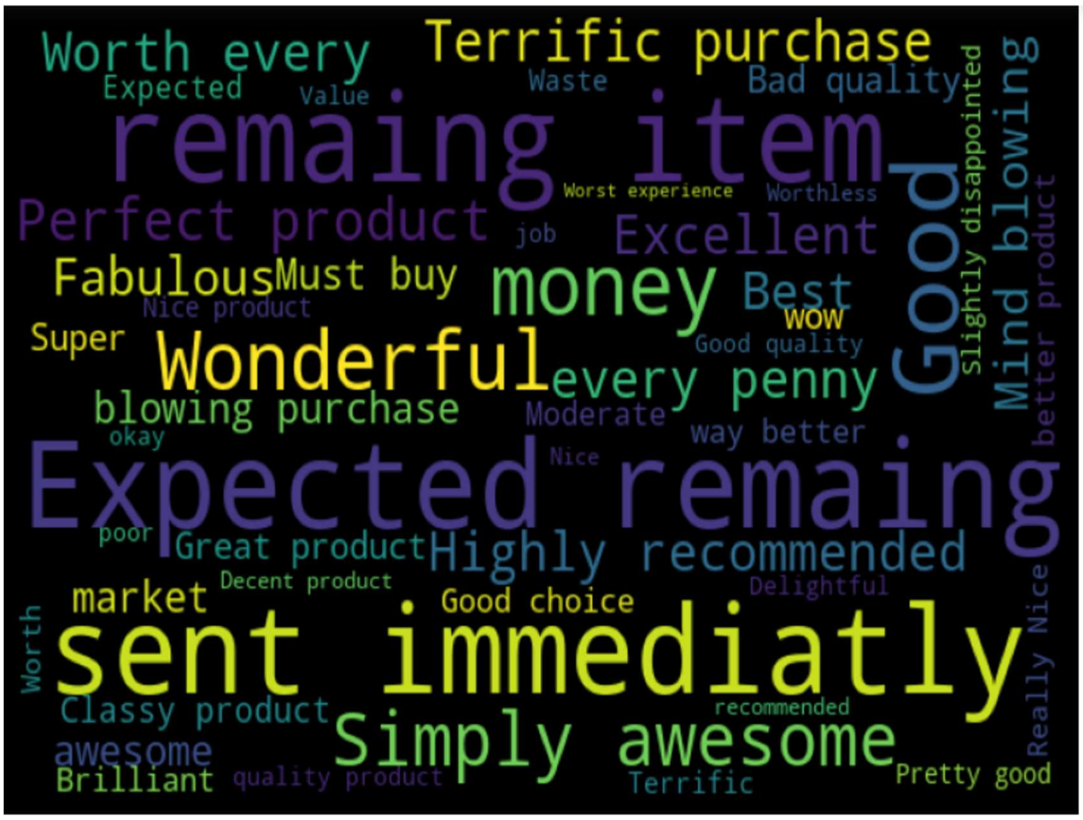
### 2.Word Cloud: Rating 4



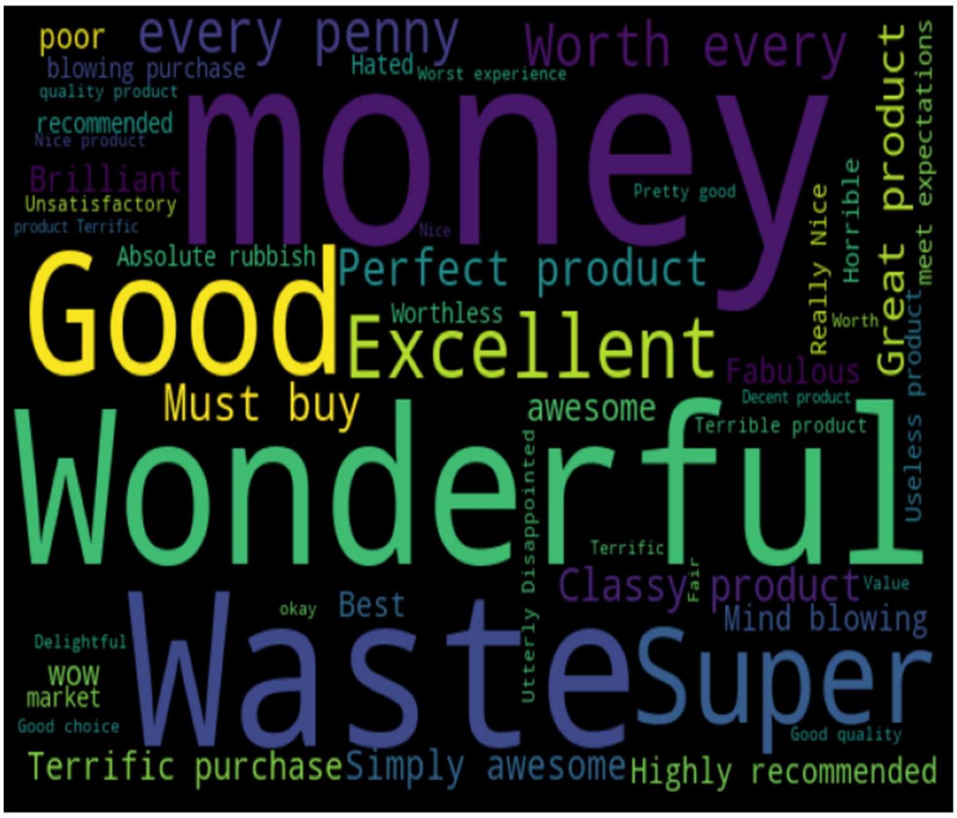
**3.Word Cloud 3:**



**2.Word Cloud2:**



**1.WordCloud1:**



## CONCLUSION

The model can be made better by converting the emojis and emoticons to texts.Some other fields can be included which will determine the rating.