Amazon Fine Food Reviews Analysis using Big data techniques

Contents

Executive Summary:

Amazon is one of the biggest sellers of fine foods over its website and a consumer plays a major role for this industry and pleasing him would help you capture the market. A customer's behavior could also help us in understanding the necessity of the products and its future trends. Food reviews are one of the major sources for the Data Scientists in the companies to make conclusions over to what are trends in the present market. The goal of this project is to capture the feedback of the customer in terms of being positive or negative, this could help a vendor refine their products specifically for the requirement of a customer and make necessary changes to their product catalog. These reviews can be categorized and can help us in deciding how a single change can improve the revenue which it could generate. There are multiple ways for achieving our intended goal. Sentiment and predictive analysis can help us in the best possible way. Machine learning algorithms could always be implemented into the e-commerce industry and can make a huge impact when did right. While purchasing any item, we see many reviews for each product and it takes a lot of time going through all the reviews. So, our analysis will save user's time and will give high-level feedback about that product.

Statement of Scope:

The dataset reflects the daily volume of reviews that have been recorded for the Amazon fine food products by its customers. Our focus is to depict a typical sentiment analysis-based model. The model takes a collection of customer reviews as input variables, process them using data pre-processing steps and then perform a sentiment classification if the review of the product is a positive or a negative one. Below are the details of each of these evaluating factors:

1. Clean reviews to study customer sentiments.

2. Prepare the cleaned data for the Machine Learning models

3. Create a Bag of Words model and predict sentiment of customer reviews.

Project Objectives:

The objective of this project is to study sentiment analysis of the customer reviews on Amazon fine food products. Based on the study we predict sentiment of the customer reviews and help Amazon rank their products for better user experience. This review analysis is useful for the companies and stakeholders who do their business with Amazon by selling their products. More the users are satisfied by third-party products that are sold on Amazon, more would the companies and stakeholders' who sell those products reputation and consequently, sales would be increased. If the company's products do not do well initially, our sentiment analysis can help those companies assess their products performance and come up with better products.

Target and Predictor Variables:

We use the text reviews to perform customer sentiment analysis and use it to predict customer reviews.

Target Variables: Score

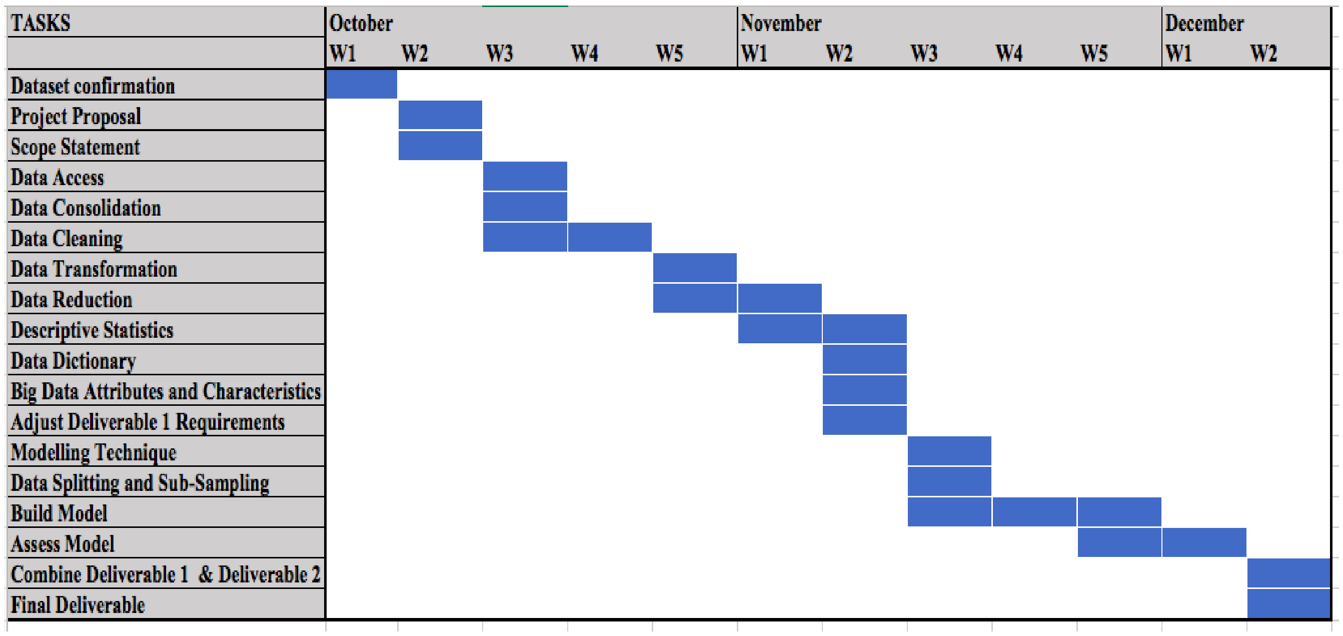
Score – Rating is given by users from 1 to 5. These ratings would later be transformed into positive and negative reviews.

Predictor Variables: Text

Text – Reviews gave by customers for the fine food products.

Project Schedule:

The tentative project schedule and resource allocation are as shown below.



TASKS	October					November					December	
	W1	W2	W3	W4	W5	W1	W2	W3	W4	W5	W1	W2
Dataset confirmation	🟦											
Project Proposal		🟦										
Scope Statement		🟦										
Data Access			🟦									
Data Consolidation			🟦									
Data Cleaning			🟦	🟦								
Data Transformation					🟦							
Data Reduction					🟦	🟦						
Descriptive Statistics						🟦	🟦					
Data Dictionary							🟦					
Big Data Attributes and Characteristics							🟦					
Adjust Deliverable 1 Requirements							🟦					
Modelling Technique								🟦				
Data Splitting and Sub-Sampling								🟦				
Build Model								🟦	🟦	🟦		
Assess Model										🟦	🟦	
Combine Deliverable 1 & Deliverable 2												🟦
Final Deliverable												🟦

Figure 1: Gantt Chart 1

Work Breakdown Structure:

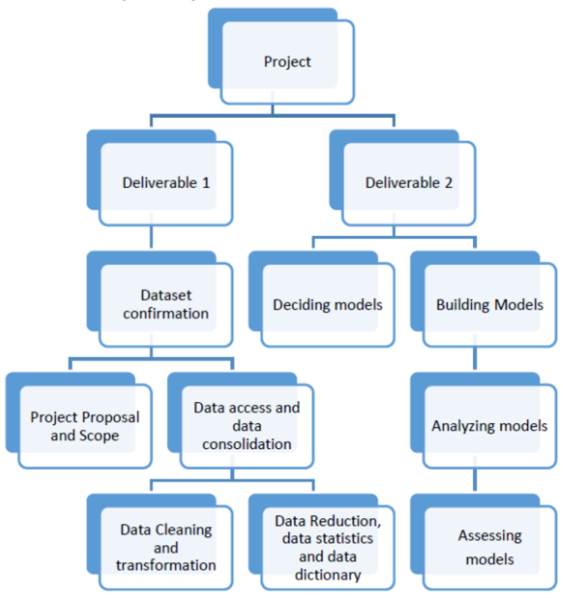

Project
Deliverable 1
Deliverable 2
Dataset confirmation
Deciding models
Building Models
Project Proposal and Scope
Data access and data consolidation
Analyzing models
Data Cleaning and transformation
Data Reduction, data statistics and data dictionary
Assessing models

Figure 2: WBS

Resource Allocation:

Task Number	Tasks	Number of Resources and Name of the resource
1	Dataset confirmation	4 (Team)
2	Project Proposal	2(Mrunali, Bhavana)
3	Scope Statement	2 (Chakri, Sai)
4	Data Access	1 (Mrunali)
5	Data Consolidation	1 (Bhavana)
6	Data Cleaning	4 (Team)
7	Data Transformation	1 (Sai)
8	Data Reduction	1 (Chakri)
9	Descriptive Statistics	4 (Team)
10	Data Dictionary	1 (Bhavana)
11	Big Data Attributes and Characteristics	4 (Team)
12	Adjust Deliverable 1 Requirements	1 (Mrunali)
13	Modelling Technique	4 (Team)
14	Data Splitting and Sub-Sampling	4 (Team)
15	Building a Model	4 (Team)
16	Assess Model	4 (Team)
17	Combine Deliverable 1 & Deliverable 2	1 (Chakri)
18	Final Deliverable	4 (Team)

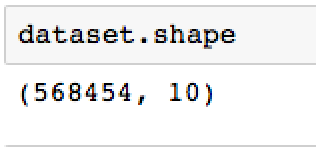
Figure 3: Resource Allocation

Data Preparation:

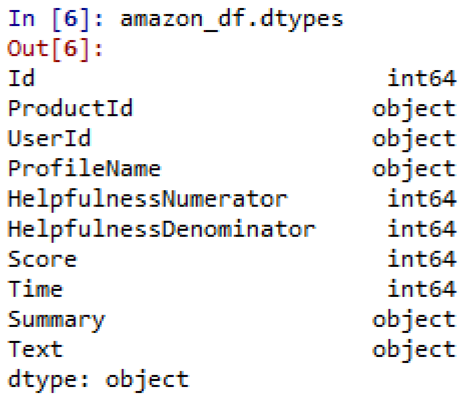
Data Access:

We have obtained the dataset from Kaggle. The data file we obtained is in CSV format. The original dataset consists of 568K records and 10 Variables. This dataset consists of customer reviews of fine foods from October 1999 to October 2012. It encompasses details such as product and user information, ratings, and text reviews. It also includes reviews from all other Amazon categories. Data includes the below characteristics:

- Reviews dated from October 1999 to October 2012
- A total of 568,454 user reviews
- 256,059 users
- 74,258 fine food products
- 260 users with more than 50 reviews



```
dataset.shape
(568454, 10)
```



```
In [6]: amazon_df.dtypes
Out[6]:
Id                          int64
ProductId                  object
UserId                     object
ProfileName                object
HelpfulnessNumerator        int64
HelpfulnessDenominator      int64
Score                       int64
Time                        int64
Summary                    object
Text                       object
dtype: object
```

Figure 4: Dataset Count and Datatypes

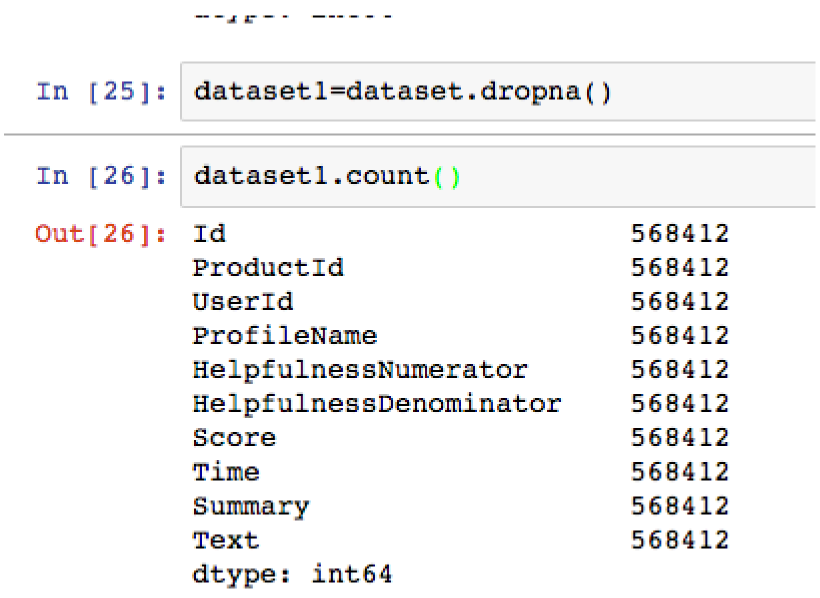
Data Consolidation:

The data we obtained from the Kaggle website was clean. All data is available in a single CSV file, so we did not have to consolidate anything with respect to the Data consolidation section.

Data Cleaning:

Columns with missing values:

The data initially had missing values in some of the columns. The below screenshot shows the dropping of all the null values in data, the count was reduced to 568412 from 568454.



```
In [25]: dataset1=dataset.dropna()

In [26]: dataset1.count()

Out[26]: Id                         568412
         ProductId                  568412
         UserId                     568412
         ProfileName                568412
         HelpfulnessNumerator       568412
         HelpfulnessDenominator     568412
         Score                      568412
         Time                       568412
         Summary                    568412
         Text                       568412
         dtype: int64
```

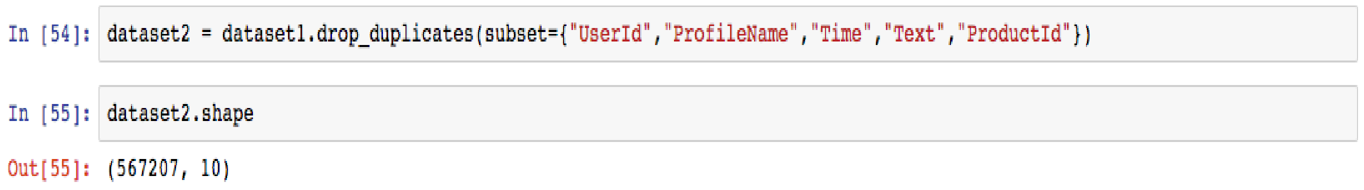
Figure 5: Count after dropping missing values

Elimination of erroneous data:

We analysed and found that our dataset does not contain any erroneous data, so we don’t need to perform any task related to the elimination of erroneous data.

Duplicate records:

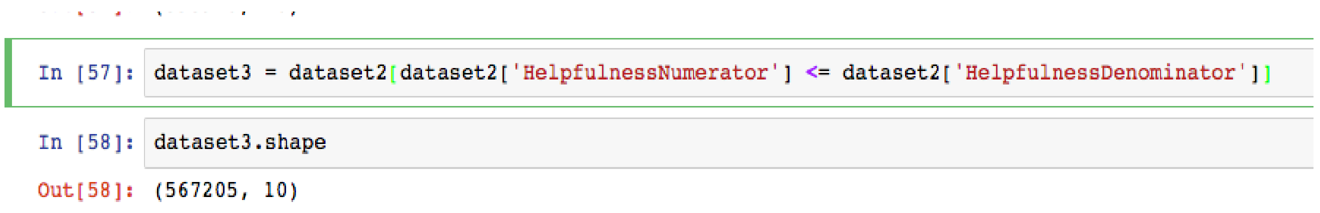
Duplication task is related to removing duplicate rows. Here, it is necessary to remove duplicates in order to get unbiased results. We are checking duplicates based on the combination of ‘UserId’, ‘ProfileName’, ‘Time’, ‘Text’, ‘ProductId’ columns. If the combination of these values is giving the same records, then we are removing those records. Below screenshot shows the count of records after deleting duplicate records. There are many duplicate records and the new count after elimination of those records is 567207.



```
In [54]: dataset2 = dataset1.drop_duplicates(subset={"UserId","ProfileName","Time","Text","ProductId"})

In [55]: dataset2.shape
Out[55]: (567207, 10)
```

Figure 6: Count after removing duplicate records

‘HelpfulnessNumerator’ column shows the count of several users who found the review helpful whereas ‘HelpfulnessDenominator’ shows the total count of the number of users who found the reviews both helpful and not helpful. Logically, ‘HelpfulnessNumerator’ should always be less than or equal to ‘HelpfulnessDenominator’. So, we checked this condition and found 2 records which have ‘HelpfulnessNumerator’ greater than ‘HelpfulnessDenominator’ and removed those records. Hence the final record count is 567205.

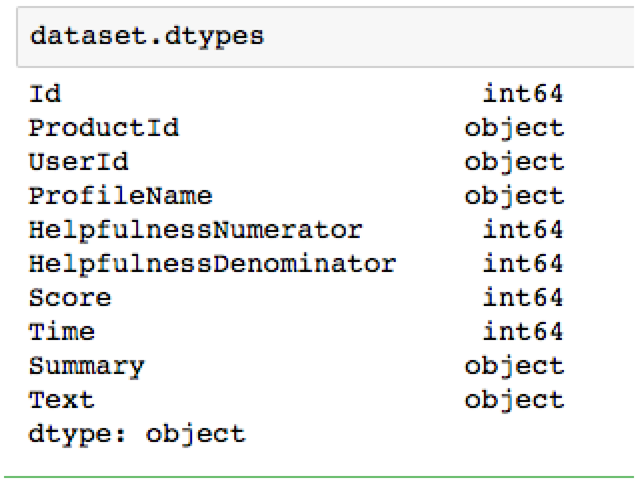


```
In [57]: dataset3 = dataset2[dataset2['HelpfulnessNumerator'] <= dataset2['HelpfulnessDenominator']]

In [58]: dataset3.shape
Out[58]: (567205, 10)
```

Figure 7: Count after cleaning erroneous values

Adjustments to data types:

The ‘Time’ column has datatype as int64. Ideally, it was expected to be in ‘DateTime’ format. We would be getting rid of this column as it will not be used for our analysis. But other than that, all datatypes are appropriate according to the column values, so we don’t need to perform any data type adjustments.

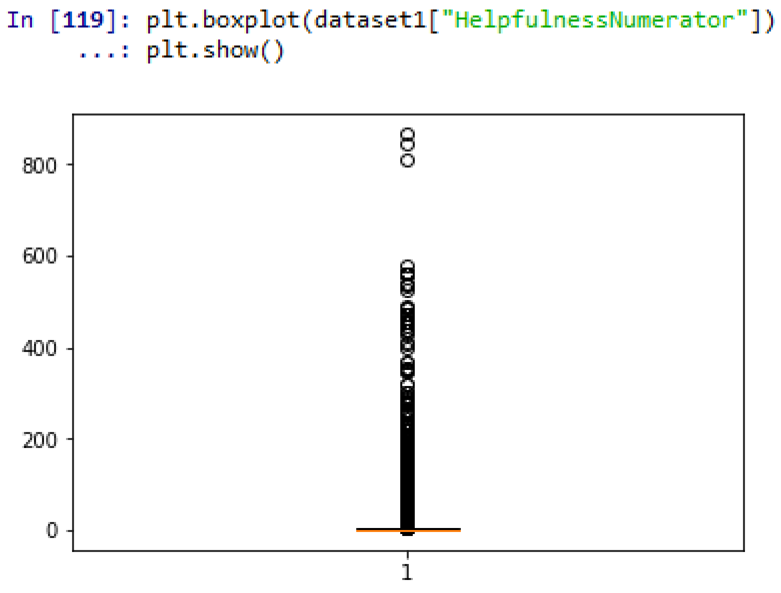


```
dataset.dtypes

Id                         int64
ProductId                 object
UserId                    object
ProfileName               object
HelpfulnessNumerator       int64
HelpfulnessDenominator     int64
Score                      int64
Time                       int64
Summary                   object
Text                      object
dtype: object
```

Figure 8: Datatype Adjustments

Outliers:

Columns ‘HelpfulnessNumerator’ and ‘HelpfulnessDenominator’ have continuous values. The following graph shows outliers for both the columns. We can see that there are many outliers for both the columns.



```
In [119]: plt.boxplot(dataset1["HelpfulnessNumerator"])
     ...: plt.show()
```

Figure 9: Outliers – Helpfulness Numerator

```
In [121]: plt.boxplot(dataset1["HelpfulnessDenominator"])
     ...: plt.show()
```

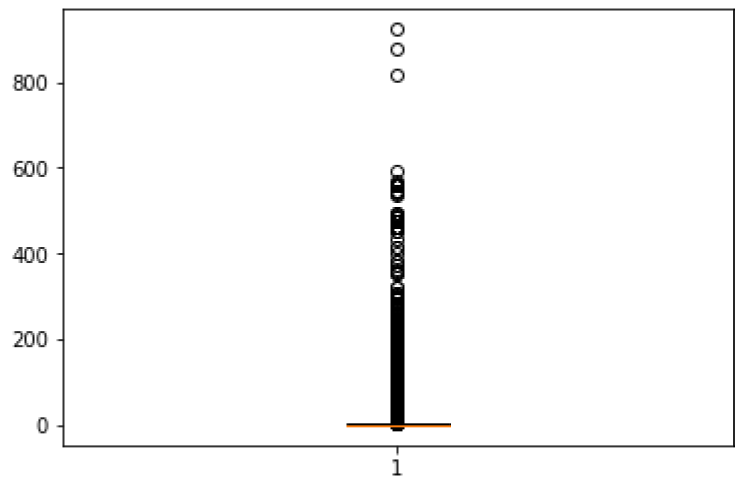


Figure 10: Outliers – Helpfulness Denominator

We would not be removing the outliers because they are simply upvotes to each specific product.

Data Cleaning for Textual data:

As we are working with user reviews of the Amazon fine foods, it requires a lot of cleaning and pre-processing of the data. The user reviews are highly unstructured and noisy in nature. The user reviews include typos, wrong grammar, stop words, expressions, etc.

Below steps are taken to clean our data:

1. Tokenization: Tokenization is a step which splits longer strings of text into smaller pieces or tokens. Larger chunks of text can be tokenized into sentences, sentences can be tokenized into words, etc. Further processing is generally performed after a piece of text has been appropriately tokenized.

2. Stemming: Stemming is the process of eliminating affixes (suffixes, prefixes, infixes, circumfixes) from a word in order to obtain a word stem. Stemming is not a simple text

manipulation. They rely on a detailed and nuanced understanding of grammatical rules and norms. An example of this application is converting the word ‘loved’ or ‘loving’ to love.

3. Removal of Stop words: Stop words are those words which are filtered out before further processing of text, since these words contribute little to overall meaning, given that they are generally the most common words in a language. For instance, "the," "and," and "a," while all required words in a passage, don't generally contribute greatly to one's understanding of content.

4. Other steps:

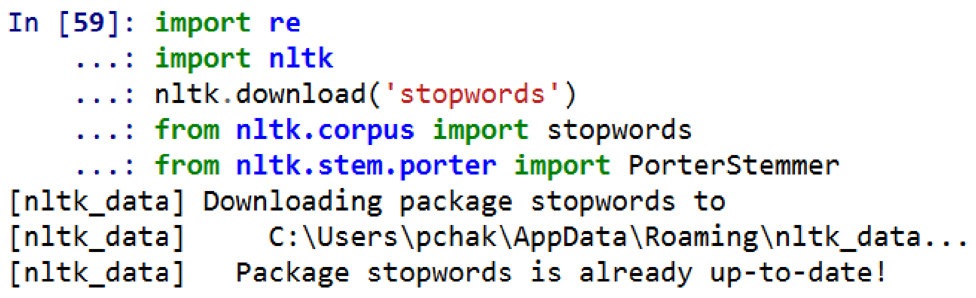
a) Set all characters to lowercase

b) Remove numbers.

c) Remove punctuation (this is a part of tokenization, but still worth keeping in mind at this stage, even as confirmation).

d) Strip whitespace (also generally part of tokenization)

We import the libraries re (stands for “Regular expression”), nltk (stands for “Natural language Tool Kit”), and packages ‘stopwords’, ‘PorterStemmer’. These components play a vital role in the text pre-processing steps. They are imported into our code and are given in the screenshot below.



```
In [59]: import re
    ...: import nltk
    ...: nltk.download('stopwords')
    ...: from nltk.corpus import stopwords
    ...: from nltk.stem.porter import PorterStemmer
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\pchak\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

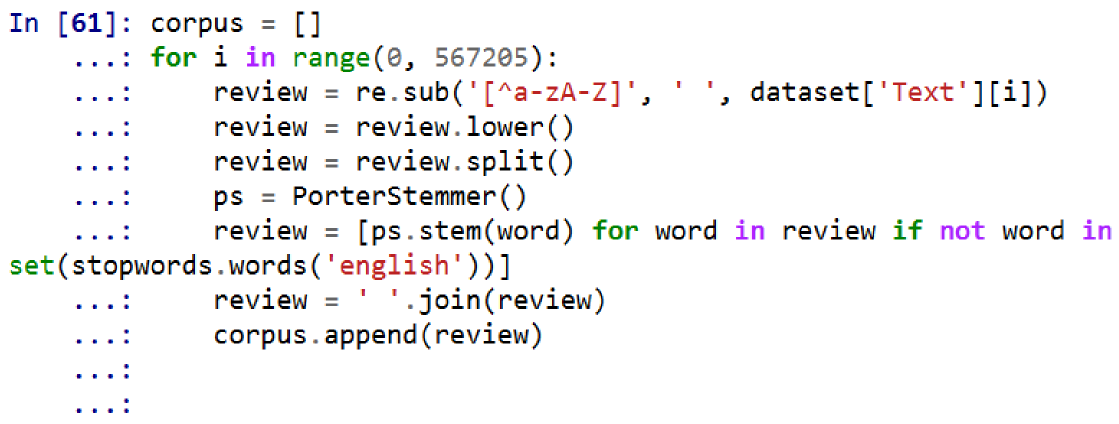
Figure 11: Import NLTK libraries and package

We clean all the text reviews in a single go by running a for loop through the entire ‘Text’ column. The steps inside the for-loop code are explained as below: -

1. We filter the review to take values only between ‘a-z’ and ‘A-Z’, this eliminates the numbers from the review.

2. We convert the text to lower cases, split them into individual words and apply the stemming process to review.

3. The usage of stop words package removes all the stop words from the review and finally we join them using the join function.

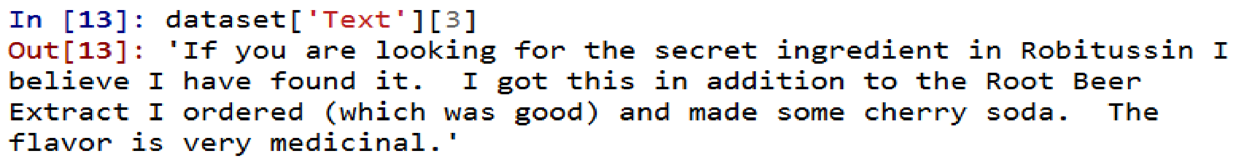


```
In [61]: corpus = []
    ...: for i in range(0, 567205):
    ...:     review = re.sub('[^a-zA-Z]', ' ', dataset['Text'][i])
    ...:     review = review.lower()
    ...:     review = review.split()
    ...:     ps = PorterStemmer()
    ...:     review = [ps.stem(word) for word in review if not word in
set(stopwords.words('english'))]
    ...:     review = ' '.join(review)
    ...:     corpus.append(review)
    ...:
    ...:
```

Figure 12: Text review cleaning code

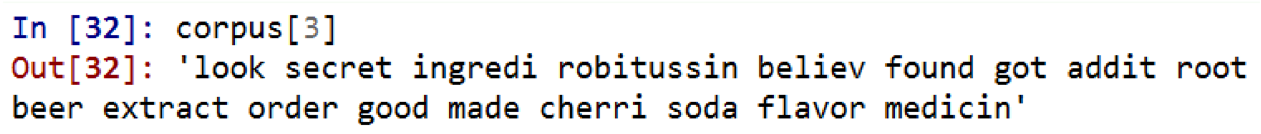
4. All the reviews are appended into corpus array. This array would later be used in our analysis to perform the bag of words model.

The below two screenshots shows the difference between the uncleaned text given as a feedback by the customer and cleaned text after the above cleaning steps have been performed.



```
In [13]: dataset['Text'][3]
Out[13]: 'If you are looking for the secret ingredient in Robitussin I
believe I have found it.  I got this in addition to the Root Beer
Extract I ordered (which was good) and made some cherry soda.  The
flavor is very medicinal.'
```

Figure 13: Text review before cleaning



```
In [32]: corpus[3]
Out[32]: 'look secret ingredi robitussin believ found got addit root
beer extract order good made cherri soda flavor medicin'
```

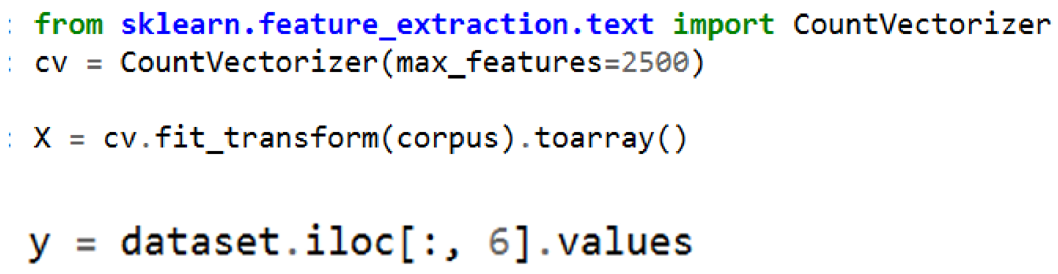
Figure 14: Text review after cleaning

Bag of words model:

For a given list of reviews, we extract only the unigram words (i.e., terms) to create an unordered list of words. This does comprise of POS tag, syntax, semantics, position, bigrams, or trigrams. Only the unigram words themselves are present, making for a bunch of words to represent the document. Thus, this bunch of words is called *Bag-of-words*.

Count Vectorizer:

Given the bag of words that you extracted from the reviews, you create a feature vector for the text reviews, where each feature is a word (term) and the feature value is a term weight. The term weight might be a binary value (with 1 indicating that the term occurred in the review, and 0 indicating that it did not). The result of this will be very large vectors. We get very accurate counts of the word content of our text data. However, we used the max features argument that takes in the count of the top 2500 words. The code is given as shown below:



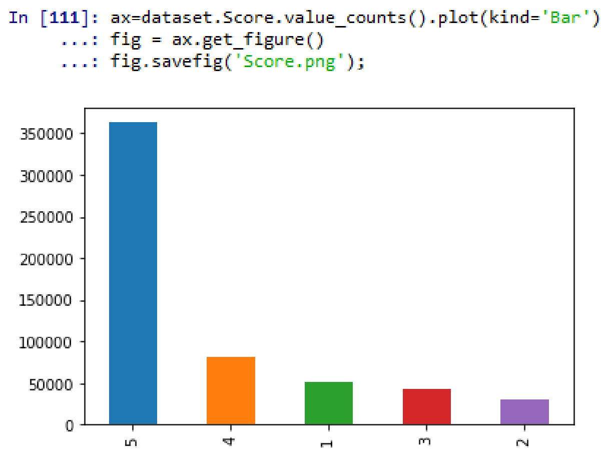
```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=2500)

X = cv.fit_transform(corpus).toarray()

y = dataset.iloc[:, 6].values
```

Figure 15: Bag of words code

We even create the X and Y vectors which are nothing but the predictor and target variable to be predicted respectively. These vectors are later being split into training and test data for predictive modeling.

Data Transformation:

We observe a five-star rating system for the 'Score' column. The 'Score' column has more reviews with ratings 5, this can lead to unbalanced classes.



```
In [111]: ax=dataset.Score.value_counts().plot(kind='Bar')
     ...: fig = ax.get_figure()
     ...: fig.savefig('Score.png');
```

Figure 16: 'Score' before transformation

We transform this column such that a positive review is considered as 1 and negative review is considered as 0. A positive review is aggregation of ratings 4 and 5 whereas the remaining ratings are aggregated as negative reviews. Below bar chart shows the rating after data transformation.

```
In [13]: ax=amazon_df1.Score.value_counts().plot(kind='bar')
    ...: fig = ax.get_figure()
    ...: fig.savefig('Score.png');
```

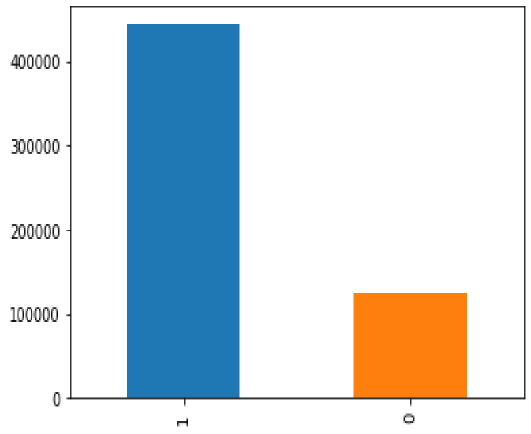


Figure 17: 'Score' after transformation

Data Reduction:

We will not be using the 'Time' column of our dataset for analysis as it is not significant for sentiment analysis and in addition to that it is available in the Unix time format. Hence, we would be getting rid of this column.

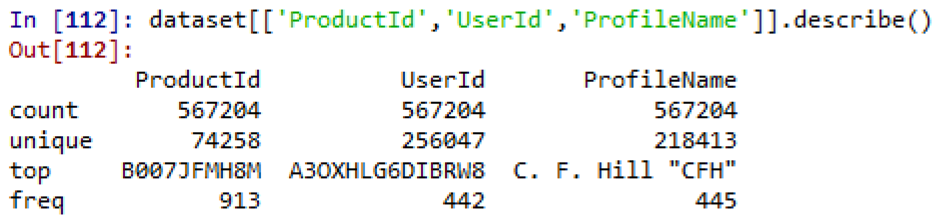
Descriptive Statistics:

It is always advised to study the variables in the data frame before performing any analysis on that. Here we have both numeric and object variables and we find the summary of them separately.

Categorical variables:

After cleaning the dataset, we have a total of 567204 rows. Following observations can be made from the categorical values:

- There are 74258 various products that have been reviewed by 256047 unique users.



```
In [112]: dataset[['ProductId','UserId','ProfileName']].describe()
Out[112]:
         ProductId          UserId       ProfileName
count       567204          567204            567204
unique       74258          256047            218413
top     B007JFMH8M  A3OXHLG6DIBRW8  C. F. Hill "CFH"
freq           913             442               445
```

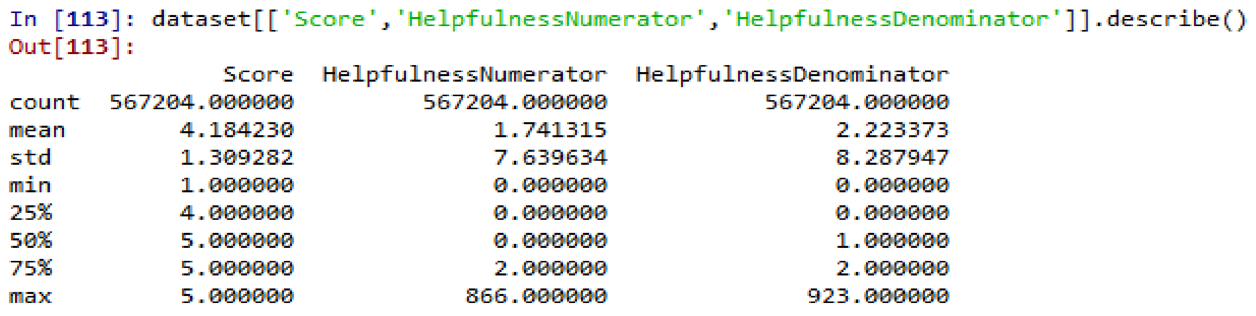
Figure 18: Descriptive Statistics-Categorical

- The most repeated product is B007JFMH8M, the user is A30XHLG6DIBRW8 and profile name is C. F. Hill “CFH”.
- One product has been reviewed a maximum of 913 times, one user has reviewed 442 times and one profile name has reviewed 445 products.

Numerical variables:

After cleaning the dataset, we have a total of 567204 rows. Following observations can be made from the numerical values:

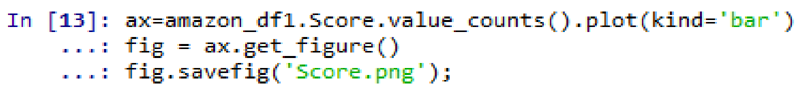
- The mean of the ‘Score’ is 4.18, for ‘HelpfulnessNumerator’ it is 1.74 and for ‘HelpfulnessDenominator’ it is 2.22.



```
In [113]: dataset[['Score','HelpfulnessNumerator','HelpfulnessDenominator']].describe()
Out[113]:
               Score  HelpfulnessNumerator  HelpfulnessDenominator
count  567204.000000         567204.000000           567204.000000
mean        4.184230              1.741315                2.223373
std         1.309282              7.639634                8.287947
min         1.000000              0.000000                0.000000
25%         4.000000              0.000000                0.000000
50%         5.000000              0.000000                1.000000
75%         5.000000              2.000000                2.000000
max         5.000000            866.000000              923.000000
```

Figure 19: Descriptive Statistics-Numerical

- The minimum and maximum 'Score' are 1 and 5 respectively, whereas for 'HelpfulnessNumerator' and 'HelpfulnessDenominator' minimum is 0 and maximum is 866 and 923 respectively.
- The 'Score' column is transformed and hence the new minimum and maximum is 0 and 1 respectively.



```
In [13]: ax=amazon_df1.Score.value_counts().plot(kind='bar')
    ...: fig = ax.get_figure()
    ...: fig.savefig('Score.png');
```

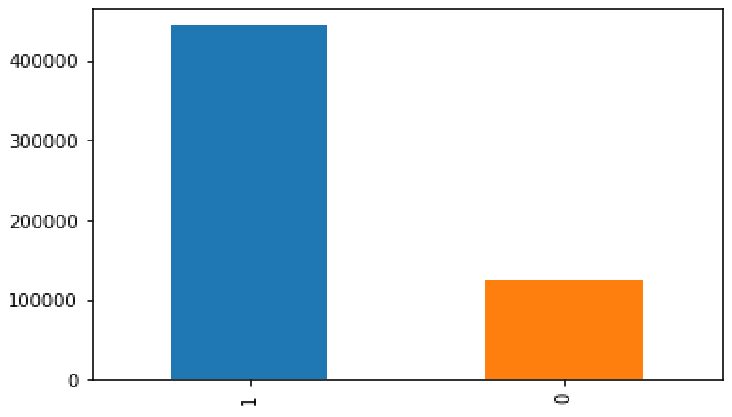


Figure 20: Descriptive Statistics - 'Score' bar chart

Above graph shows the positive and negative rating. A positive rating is considered as 1 and negative is considered as 0.

QQ plot:

```
In [127]: stats.probplot(dataset1['HelpfulnessDenominator'], dist="norm", plot=pylab)
Out[127]:
((array([-4.71273974, -4.52877704, -4.42919216, ...,  4.42919216,
          4.52877704,  4.71273974]),
  array([  0,   0,   0, ..., 815, 878, 923], dtype=int64)),
 (3.677485079029565, 2.2233728958187879, 0.44371133384971662))
```

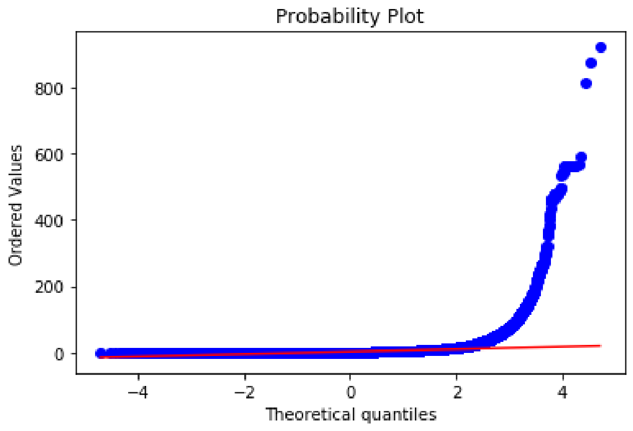


Figure 21: QQ plot – Helpfulness Denominator

```
In [124]: stats.probplot(dataset1['HelpfulnessNumerator'], dist="norm", plot=pylab)
Out[124]:
((array([-4.71273974, -4.52877704, -4.42919216, ...,  4.42919216,
          4.52877704,  4.71273974]),
  array([  0,   0,   0, ..., 808, 844, 866], dtype=int64)),
 (3.0408109416756894, 1.7413152939683072, 0.39802782279471233))
```

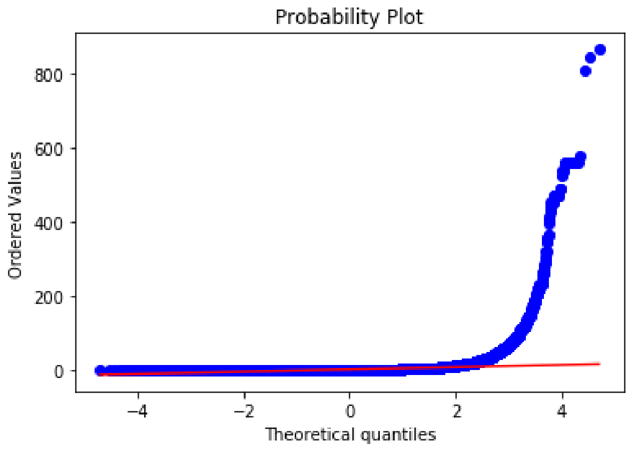


Figure 22: QQ plot – Helpfulness Numerator

QQ plots are not normally distributed a few of the reference points are nicely represented on the reference line whereas the other values are going out of reference line.

Data Dictionary:

Attribute Name	Description	Data Type	Source
Id	Row Id	Int	https://www.kaggle.com/snap/amazon-fine-food-reviews
ProductId	Unique identifier for the product	Object	https://www.kaggle.com/snap/amazon-fine-food-reviews
UserId	Unique identifier for the user	Object	https://www.kaggle.com/snap/amazon-fine-food-reviews
ProfileName	Profile name of the user	Object	https://www.kaggle.com/snap/amazon-fine-food-reviews
HelpfulnessNumerator	Number of users who found the review helpful	Int	https://www.kaggle.com/snap/amazon-fine-food-reviews
HelpfulnessDenominator	Number of users who indicated whether they found the review helpful	Int	https://www.kaggle.com/snap/amazon-fine-food-reviews
Score	Rating between 1 and 5	Int	https://www.kaggle.com/snap/amazon-fine-food-reviews
Time	Timestamp for the review	Int	https://www.kaggle.com/snap/amazon-fine-food-reviews

Summary	Brief summary of the review	Object	https://www.kaggle.com/snap/amazon-fine-food-reviews
Text	Text of the review	Object	https://www.kaggle.com/snap/amazon-fine-food-reviews

Figure 23: Data Dictionary

Big Data Attributes and Characteristics:

The Term "BIG DATA" can be differentiated from the regular data by the source of the data and the way they are collected. There is a certain parameter by which we can differentiate the data from the big data. Having a huge number of records will not certainly make data the "Big Data". These parameters are known as four Vs of the Big data. They are Volume, Variety, Velocity, and Veracity. Our Dataset consists of the reviews about the products given by the customers in reference to their usage and the ratings they have provided. Our data set of reviews and a summary of the comments have both volume and velocity. In terms of the variety, they are different types of data and multiple sets of values and multiple set of columns which are of different data types. There are food and non-food items in our dataset, which gets a huge set of items in its bracket. The reviews we are dealing with have been recorded and captured over time, there is a time stamp included which explains the generation of data over time. The data frame also satisfies the term veracity, even though the data has many numbers of columns and there are large amounts of data. The data is not uncertain. The data is reliable and complete and consistent

Volume:

The volume refers to the volume of the data in terms of size. The dataset we have been working with is about 270 MB which is a large amount when in terms of the size. There are about 5,68,454 records and they all are particular to each user and the review written by each of them. These unique features of each of the data records make it them volume rich and great for big data analysis. The Volume is certainly referring as a character when the data is at rest.

Velocity:

The Velocity refers to how fast the data is being streamed or what is the span of the data being collected. Here velocity refers to the time span between the comments on every product. The data in our dataset is being collected over time. The data we are dealing with also has time stamp included as one of the columns which explain the data being collected over time and explains the detail of the data which has been recorded over time.

Variety:

Variety refers to the different types of data we are been dealing with in our analysis. There are many types of data that are present in the given review's dataset. These types range from the integers to the continuous values. Textual reviews contain words from multiple languages, emotions, punctuations. So tremendous text cleaning process is required in order to make this data useful.

Modeling Techniques:

There are many machine learning algorithms which are readily available and are being used by the data scientists. No algorithm is perfect and can be used in all the scenarios given. Each of the techniques have their own merits and demerits and are specific to the analysis we are about to perform. It majorly depends on the target variable we are trying to predict. Here the target variable is to predict whether a review would be a good one or a bad one. Since we are expecting the model to give an output which would be in terms of a category of a positive or negative response, we could choose any of the classification algorithms. There are many classification algorithms available, but we have chosen Decision tree and Random forest learner.

Decision Tree:

Decision trees are very good with categorical variables. They work on the principle of dividing and conquer. They can easily and efficiently decide on the decision boundaries and can segregate the variables into the respective categories and our models must take in a lot of categorical variable. Decision trees make the split basing their influence in predicting the end variable. This gives an extra advantage that we can know about the most important variable without consulting any extra parameters unlike we do for any other data mining techniques. Also, the decision trees are very easy to interpret. Once drawn they can be interpreted by visually seeing the flow of the data over the nodes.

Random Forest:

Random forest learner is also one of the best in business when it comes to predicting a given categorical variable. They work on the principle of building multiple trees for predicting and builds a single tree which is the more optimized and accurate. These types of trees would help you in facing any type of a scenario, for instance, if we are planning on to add in a few more data to the existing data then there is a chance that these models may have come across such situation already. This helps our prediction to be more accurate and more stable. They do well when we are dealing with categorical variable, they could be one of our best bets. The more the number of trees the improved the accuracy would be for these types of models. Also, these trees help us deal with multiple features which may be interim correlated. These trees also reduce the variance which is not in the case of the regular other trees.

Model Goals:

Decision Tree:

The decision tree is identified as the simplest model as it is very easy to interpret. The aim is to build a model that predicts the sentiment of the target variable "Text" as either a positive review or a negative review. To encapsulate the training data in the smallest possible tree is a goal of the decision tree. Decision tree provides simple rules to split the node at the top of the tree, that is a root node. Various branches of different length are formed. Minimize the tree size is a logical rule so that decision can be made faster. Hence, we can easily interpret the effect of predictors "Text" on the target variable "Score".

Random Forest:

Random forest is also popular model like the decision tree. This model is a good choice when the relationship between dependent and independent variable is non-linear. It provides ordered list of important variables and a majority tree is selected. Random forest runtimes are quite fast, so the performance of the overall model can be improved by this tree. Random forest combines tree with the notion of an ensemble. Prediction of a target variable "score" would become easy with this model.

Data Specific techniques:

Decision tree algorithm is a classification and regression algorithm used for predictive modeling of both discrete and continuous attributes. The data we are dealing with has an expected output of a categorical variable, to a review to be categorized into a positive or a negative one. All our predicting variables are also categorical. Decision trees and Random forests would both excel in this type of analysis. Decision trees with an ease in their interpretation while working with categorical variables and providing an important measure, Random forest building multiple trees and proving a more accurate and stable prediction. These two types would provide some very good and accurate analysis.

Data Splitting and Subsampling:

Our Target variable is “Score” with 567205 records and distribution of this is as follows:

Score Type	Count
0	124250
1	442955

Figure 24: ‘Score’ distribution

Count of score 1 is very high as compared to 0 that is because there are more positive comments as compared to number of negative comments. A decision tree and random forest models can fit into the training dataset. The most important variables within the dataset are the top nodes on which tree split is done. We use data splitting generally to have a robust predictive model. Data is split usually into 2 portions namely Training data and Test data. Sometimes even validation data is also considered. We also must split the data in a way that it should prevent overfitting and underfitting. If our model is overfitting, then it generally unable to provide accurate predictions and if our model is under fitted then it would miss trends and data. Hence, data splitting is one of the important aspects. Cross-validation is also one of the techniques that can be used to prevent this. We have split the data in the 80-20 ratio for the following reasons:

- 80% data is used as training data and 20% as testing data. This implies that our random forest model is trained sufficiently and is a good contender for the robust predictor model.
- Since there are greater than half a million records, this split would be better for our model.
- If we would have taken 70-30 or 60-40, our model would have missed certain data trends and points. That would have made this model as under fit and could not be used for identifying sentiment analysis and hence 80-20 would be a good fit for our models.

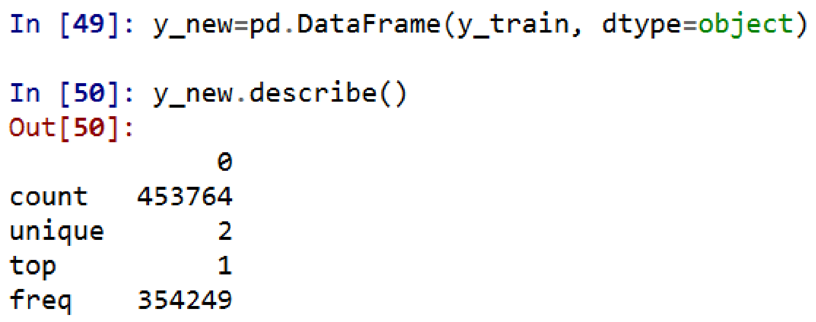
Comparison of variables between test and training data:

Numerical Variables Comparison:

For our models, we do not have any numerical data fed into the variable. All our variables are categorical. These categorical variables have been converted to dummy variables. It would not be of much significance if we study the descriptive statistics of the dummy variables. Hence there is no comparison for numerical variables for training and test data.

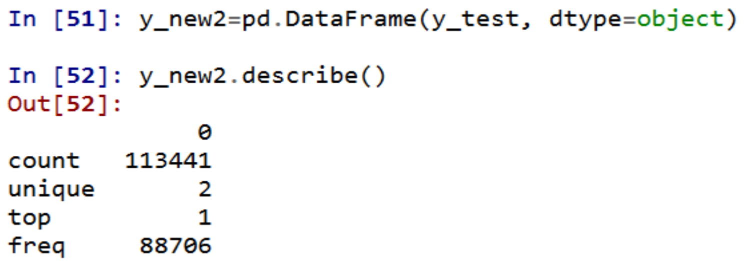
Categorical Variables Comparison:

Score variable ranged from 1 to 5. We combined score ranging from 1 to 3 as score 0 and score 4 and 5 is considered as score 1. Primary reason for score value as 3 is categorized into 1 as generally rating as 3 is considered as neutral score, which is neither good nor bad. The below summary of the categorical variables shows similarities in both the training and test data set, i.e., the frequent observation is score. The percentage of the occurrence of the score ‘1’ observation is higher in both training and test dataset and is approximately 78% distributed in both the datasets.



```
In [49]: y_new=pd.DataFrame(y_train, dtype=object)

In [50]: y_new.describe()
Out[50]:
                0
count   453764
unique       2
top          1
freq    354249
```

Figure 25: Training Dataset



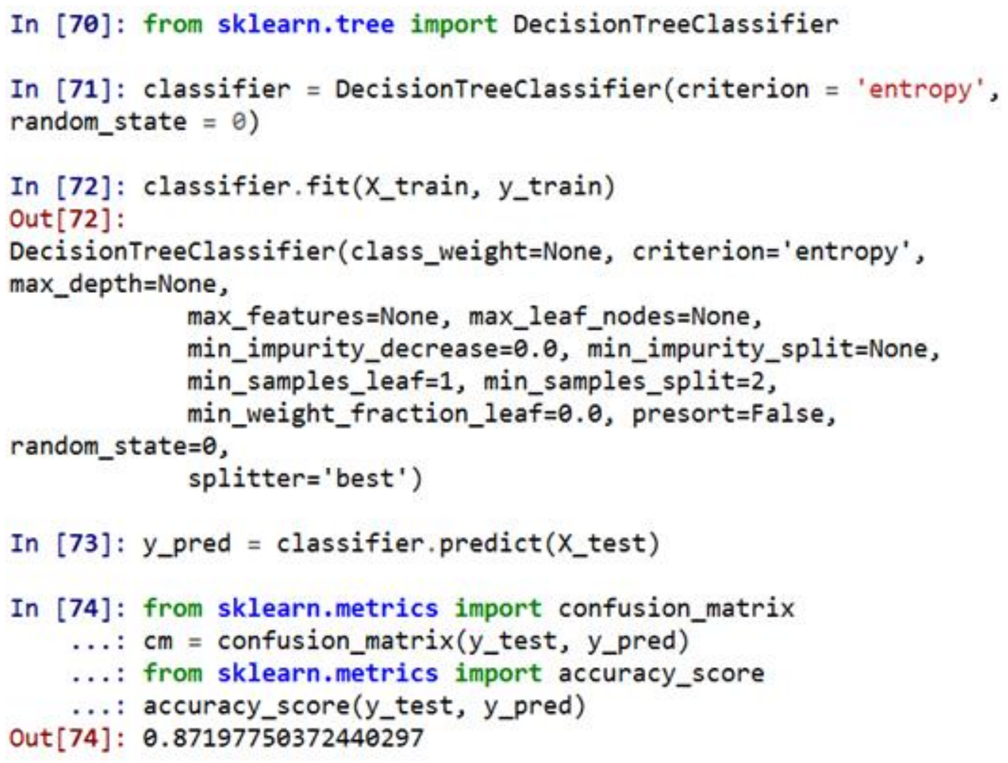
```
In [51]: y_new2=pd.DataFrame(y_test, dtype=object)

In [52]: y_new2.describe()
Out[52]:
                0
count   113441
unique       2
top          1
freq     88706
```

Figure 26: Testing Dataset

Model Building:

Decision Tree Model:

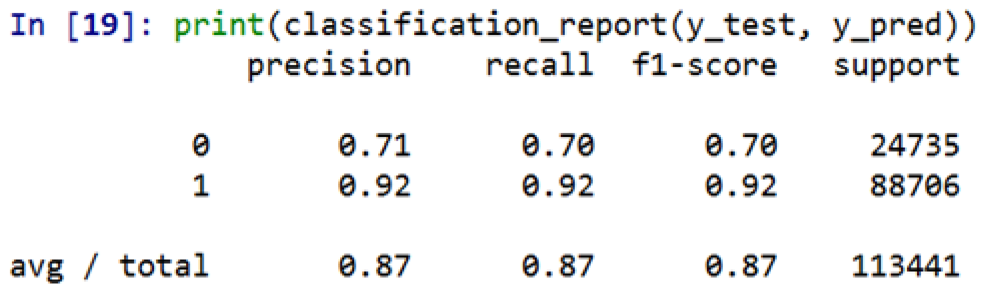
When building a model that has categorical variables and numerical variables in a decision tree, CART (Classification and Regression trees) procedure is usually chosen. This is because the parent node is divided into two child nodes, one that satisfies the parent condition and the other node is when the parent condition is not satisfied. The child nodes are again divided in the same way. The screenshot below shows the code used to generate the decision tree.



```
In [70]: from sklearn.tree import DecisionTreeClassifier

In [71]: classifier = DecisionTreeClassifier(criterion = 'entropy',
random_state = 0)

In [72]: classifier.fit(X_train, y_train)
Out[72]:
DecisionTreeClassifier(class_weight=None, criterion='entropy',
max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False,
random_state=0,
            splitter='best')

In [73]: y_pred = classifier.predict(X_test)

In [74]: from sklearn.metrics import confusion_matrix
    ...: cm = confusion_matrix(y_test, y_pred)
    ...: from sklearn.metrics import accuracy_score
    ...: accuracy_score(y_test, y_pred)
Out[74]: 0.8719775037244029
```

Figure 27: Decision tree model code

Our model has reviews in the form of categorical variables, only the top 2500 features have been used to predict the sentiment of the reviews. The Decision tree model gives 87.19% accuracy which simply suggests that our model was able to able to correctly predict the sentiment of a customer in 87.19% cases. Below screenshot shows us the classification report of how the Decision model had performed.



```
In [19]: print(classification_report(y_test, y_pred))
             precision    recall  f1-score   support

          0       0.71      0.70      0.70     24735
          1       0.92      0.92      0.92     88706

avg / total       0.87      0.87      0.87    113441
```

Figure 28: Decision tree classification report

Results and Interpretation:

From the accuracy results and classification report, we see our decision model has performed pretty good. The precision, recall, and f1-score for a positive review is 92% and for the negative review, it is about 70%.

Precision:

Precision is the ratio of correctly predicted positive sentiments to the total predicted positive sentiments. The question that this metric answer is of all reviews that labelled as positive, how many are positive? High precision relates to the low false positive rate. We have got 71% precision for negative review which is decent and 92% for a positive review which is pretty good.

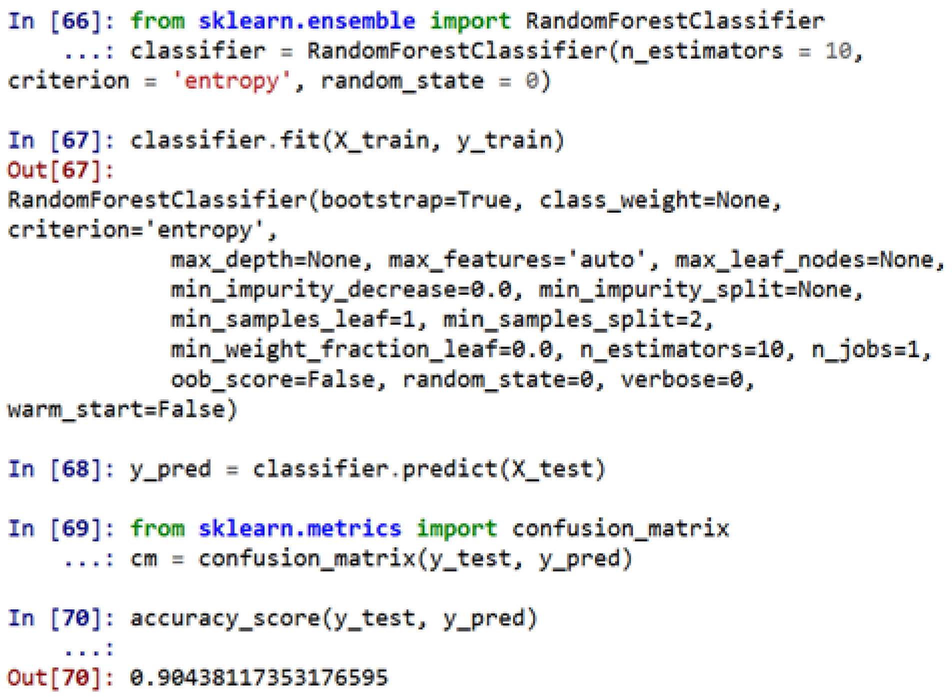
Recall:

Recall is the ratio of correctly predicted positive sentiments to the all sentiments in actual class - Positive. The question recall answers is: Of all the reviews that are truly positive, how many did we label? We have got recall of 70% for negative reviews which is decent for this model as it’s above 0.5 and for positive reviews, it is 92% which is pretty good.

F1-Score:

F1-Score is simply the weighted average of Precision and Recall, we get 92% for a positive review and 70% for a negative review.

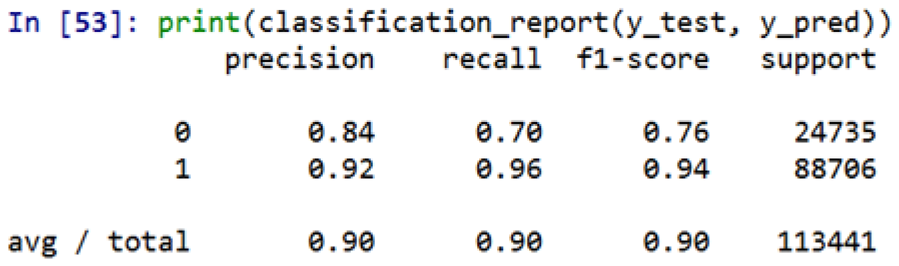
Random Forest Model:



```
In [66]: from sklearn.ensemble import RandomForestClassifier
    ...: classifier = RandomForestClassifier(n_estimators = 10,
criterion = 'entropy', random_state = 0)

In [67]: classifier.fit(X_train, y_train)
Out[67]:
RandomForestClassifier(bootstrap=True, class_weight=None,
criterion='entropy',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
            oob_score=False, random_state=0, verbose=0,
warm_start=False)

In [68]: y_pred = classifier.predict(X_test)

In [69]: from sklearn.metrics import confusion_matrix
    ...: cm = confusion_matrix(y_test, y_pred)

In [70]: accuracy_score(y_test, y_pred)
    ...:
Out[70]: 0.90438117353176595
```

Figure 29: Random forest model code

After decision tree, we ran Random Forest model and found 90.43 % accuracy which is more than decision tree accuracy. This high accuracy suggests that our Random forest model was able to able to correctly predict the sentiment of a customer in 90.43% of the cases. Below screenshot shows us the classification report of how the Random forest model had performed.



```
In [53]: print(classification_report(y_test, y_pred))
             precision    recall  f1-score   support

          0       0.84      0.70      0.76     24735
          1       0.92      0.96      0.94     88706

avg / total       0.90      0.90      0.90    113441
```

Figure 30: Random forest classification report

Results and Interpretation:

From the accuracy results and classification report, we even see our Random forest model has performed pretty good. It performed slightly better than the Decision tree model. The precision and recall for a positive review are 92% and 96% respectively and for the negative review, it is about 84% and 70% respectively.

Precision:

Precision is the ratio of correctly predicted positive sentiments to the total predicted positive sentiments. The question that this metric answer is of all reviews that labelled as positive, how many are positive? High precision relates to the low false positive rate. We have got 84% precision for negative review which is good and 92% for a positive review which is pretty good.

Recall:

Recall is the ratio of correctly predicted positive sentiments to the all sentiments in actual class - Positive. The question recall answers is: Of all the reviews that are truly positive, how many did we label? We have got recall of 70% for negative reviews which is decent for this model as it's above 0.5 and for positive reviews, it is 96% which is excellent.

F1-Score:

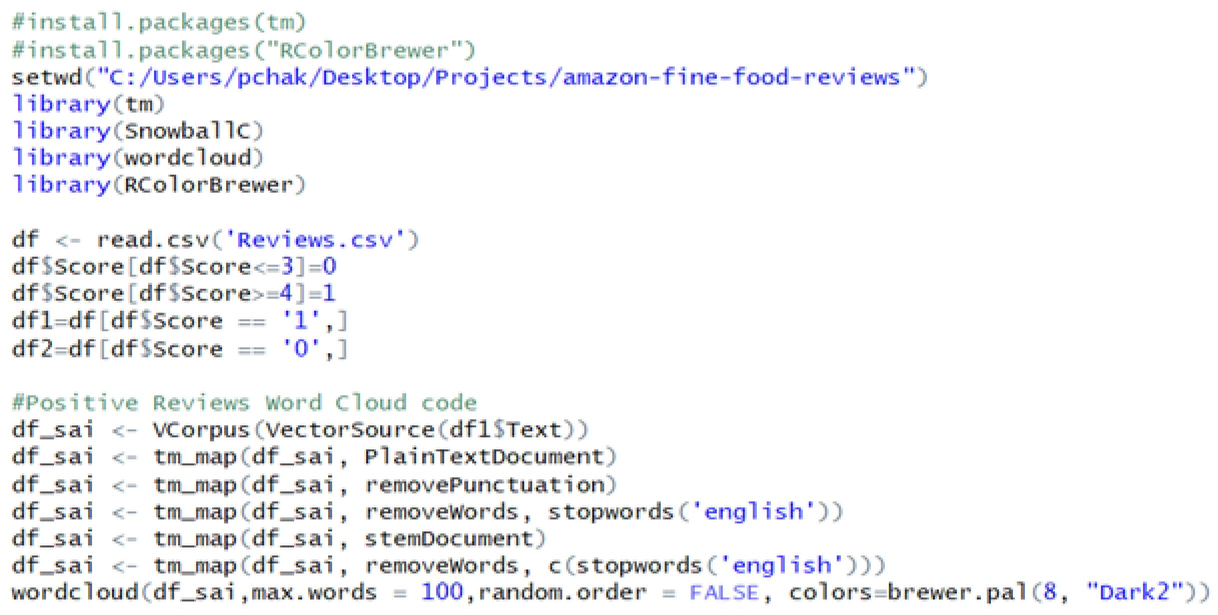
F1-Score is simply the weighted average of Precision and Recall, we get 94% for a positive review and 76% for a negative review.

Word Cloud:

We generated Word Cloud of both positive and negative reviews in R language. The cloud gives a basic idea on the words that are being majorly used on a positive or a negative review. It highlights those important words in bigger size and gives Amazon a general idea on how its products are doing, improve on those suggestions and target right customers for right suggestions of products. This type of analysis can also be extended to individual products so that the company of the product can understand its customer sentiment and make their products better. But since we had limited number of reviews on individual products, we just limited our scope to overall positive and negative reviews only.

Positive Reviews:

The code for the positive reviews in R language and the Word Cloud is shown in next two screenshots below.



```
#install.packages(tm)
#install.packages("RColorBrewer")
setwd("C:/Users/pchak/Desktop/Projects/amazon-fine-food-reviews")
library(tm)
library(SnowballC)
library(wordcloud)
library(RColorBrewer)

df <- read.csv('Reviews.csv')
df$Score[df$Score<=3]=0
df$Score[df$Score>=4]=1
df1=df[df$Score == '1',]
df2=df[df$Score == '0',]

#Positive Reviews Word Cloud code
df_sai <- VCorpus(VectorSource(df1$Text))
df_sai <- tm_map(df_sai, PlainTextDocument)
df_sai <- tm_map(df_sai, removePunctuation)
df_sai <- tm_map(df_sai, removeWords, stopwords('english'))
df_sai <- tm_map(df_sai, stemDocument)
df_sai <- tm_map(df_sai, removeWords, c(stopwords('english')))
wordcloud(df_sai,max.words = 100,random.order = FALSE, colors=brewer.pal(8, "Dark2"))
```

Figure 31: Positive review code

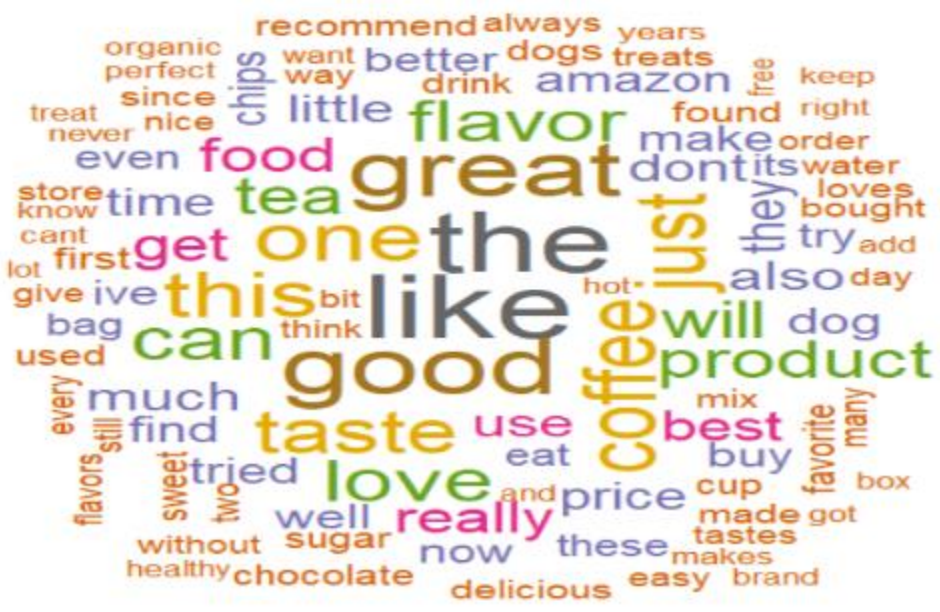
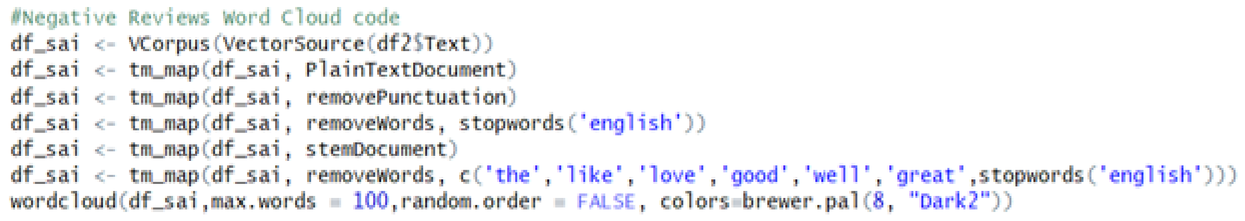


Figure 32: Positive review word cloud

Negative Reviews:

The code for the negative reviews in R language and the Word Cloud is shown in next two screenshots below.



```
#Negative Reviews Word Cloud code
df_sai <- VCorpus(VectorSource(df2$Text))
df_sai <- tm_map(df_sai, PlainTextDocument)
df_sai <- tm_map(df_sai, removePunctuation)
df_sai <- tm_map(df_sai, removeWords, stopwords('english'))
df_sai <- tm_map(df_sai, stemDocument)
df_sai <- tm_map(df_sai, removeWords, c('the','like','love','good','well','great',stopwords('english')))
wordcloud(df_sai,max.words = 100,random.order = FALSE, colors=brewer.pal(8, "Dark2"))
```

Figure 33: Negative review code

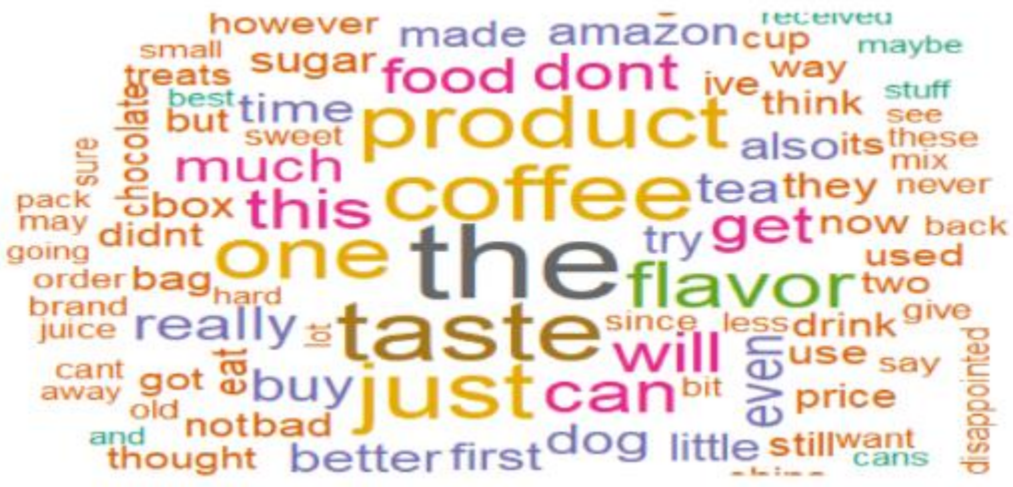


Figure 34: Negative review word cloud

Model Assessments:

Decision Tree Model:

Strengths:

- The accuracy of the decision tree is high, so we can say that the prediction accuracy of target variable is high.
- Decision tree is very easy to understand and interpret. Also, the visual display is good. So, a person not having much analytics knowledge can also understand and interpret the model.
- Linearity is not required in the decision tree execution. Hence, we can expect the correct result for the nonlinear model as well.
- Data preparation efforts can be reduced by using a decision tree.
- Missing values do not affect the decision tree splitting and result.
- Outliers does not make any difference in decision tree analysis.

- Decision tree can handle continuous and discrete target variables efficiently.

Weakness:

- Prediction of continuous target variables is a challenging task.
- Decision trees have dynamic behaviour, so it might create issues.
- In case of large and complex decision tree, analysis and interpretation could be difficult. Knowledge of tree pruning is required in this case.
- The relationship between target variable and predictors might change. Because any minor change in data can affect result so it could affect the analysis.

Random Forest Model:

Strengths:

- Random forest handles missing and unbalanced data very well. Also, it has methods for balancing error in case of unbalanced datasets.
- Random forest runtimes are fast.
- Even on small dataset random forest can achieve good accuracy so computational cost of training is low.
- Random forest gives more accuracy. Also, predictions, results are more accurate. It is one of the most accurate algorithms available.
- Random forests visual display is good, so they are easy to understand and interpret.
- Random forest can handle thousands of input variable without variable deletion.

Weakness:

- Random forest can face overfitting problem when used for regression. Because they cannot predict beyond the training data range.

- Random forests are biased in case of categorical variables with different number of levels. Therefore, the variable importance score is not reliable.
- Random forest error increases as the inter-tree correlation increases. So, trees must be uncorrelated as possible.
- We need to select number of trees.

Word Cloud:

Strengths:

- Positive and negative reviews can be studied using this analysis. This could help us to identify the opinion of the writer.

Weakness:

- Recognizing things like sarcasm, jokes, and exaggerations would be difficult.

Conclusions:

Decision Tree Model:

Accuracy:

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. Decision tree model has total of 87.19% accuracy.

ROC Curve:

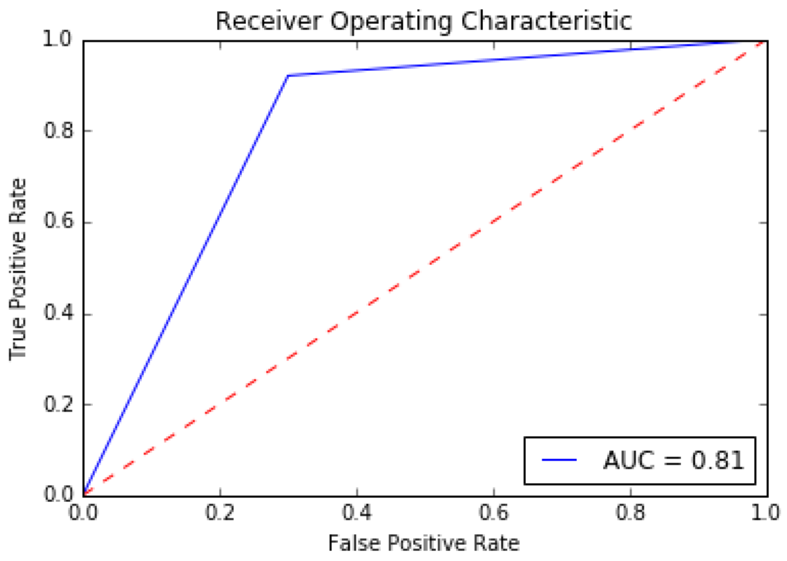


Figure 35: Decision tree ROC

ROC or Receiving operating characteristic is used to test the performance of a binary classifier. It depicts how good the classifiers explain the true positives and at what level it shows the false positives. ROC defines the rate at which a prediction is performed correctly. ROC gives perfect results when the classifier is binary. Hence, we have used ROC curve to study the truth of the prediction results. Here we find the area under the curve for Decision tree model to be 0.81 which is decent.

Random Forest Model:

Accuracy:

Accuracy is the measure how well the model correlates an outcome with the attributes in the data that has been provided. Random forest model has total of 90.43% accuracy.

ROC Curve:

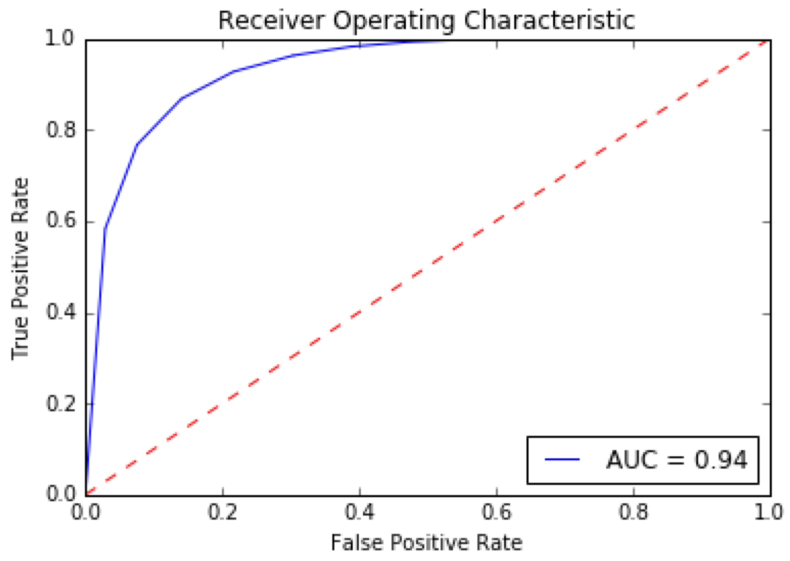


Figure 36: Random forest ROC

We have plotted ROC curve to study the truth of the prediction results. Here we find the area under the curve for Random forest model to be 0.94 which is very good. ROC curve is away from diagonal and near to the axis, so it looks a good fit.

Both the models, Random forest and Decision tree agreed with each other in predicting most of the correct sentiments. Our accuracy for the random forest model is 90.43% whereas for decision tree accuracy is 87.19%. Although we have got good accuracy for both models, Random forest model performed better in terms of accuracy and in other metrics such as recall, precision, and AUC for ROC curve. Hence, we can say the two models complemented each other well in determining customer sentiments.

The purpose of this project was to develop a predictive model which would help to understand the helpfulness of the reviews. We have successfully achieved our goal by developing models which correctly identifies whether the review is positive or negative with high accuracy. Our results not only provide the score analysis but also provide the visualized word cloud. Visualized word cloud will save a lot of time going through all the reviews and will give high-level feedback about that product. This will help business owners to understand the future trends and customer behaviour to the granular level. We can implement this study in other product categories of Amazon and for other industries as well.

References:

https://www.kaggle.com/snap/amazon-fine-food-reviews

https://www.kaggle.com/shashanksai/text-preprocessing-using-python#

https://www.kaggle.com/poonaml/text-classification-using-spacy#

https://www.kdnuggets.com/2017/12/general-approach-preprocessing-text-data.html

http://blog.citizennet.com/blog/2012/11/10/random-forests-ensembles-and-performance-metrics

http://blog.citizennet.com/blog/2012/11/10/random-forests-ensembles-and-performance-metrics