HarvardX PH125.9x - Capstone CYO - Building Models To Predict Happy versus Unhappy On Amazon Fine Foods Reviews

Aditya Wresniyandaka

March 10, 2019

Executive Summary

Customer reviews are a very important and integral part of the success of online commerce sites like Amazon, Whole Foods, etc. Customers rate their experiences with the products and services offered, and are encouraged to write comments as part of the review. The project reported in this document analyzed the Amazon Fine Food Reviews from October 1999-2012 originally posted at the Stanford Network Analysis Project (SNAP) web site (https://snap.stanford.edu/data/web-FineFoods.html). There are 568,454 rows from 256,059 users for 74,258 products. The original format is 4.7 million rows in one column. The dataset posted in Kaggle has been formated in a tabular format so we will use this dataset. (https://www.kaggle.com/snap/amazon-fine-food-reviews). Two models (Naive Bayes with 10-fold Cross Validation and Random Forests) are built to predict customer review classification, i.e. whether they are Happy or Unhappy. The finaly accuracy from each model is reported as 0.795 for Naive Bayes, and 0.855 for Random Forests which provides 7% improvement over Naive Bayes.

Analysis Methodology and Approach

The focus of this analysis is on the Text reviews and their corresponding scores (in the scale of 1 to 5 from lowest to highest rating). We group the scores to 1 & 2 as Unhappy, and 4 & 5 as Happy. We remove Score 3 as this is Neutral. We also examine any imbalance in the Score distribution (it is highly likely that the reviews are skewed toward one end of the range). To address the limits on the computer resources available, we take 50% of the dataset to build the models.

Data retrieval

The dataset from Kaggle is packaged as a ZIP file. Inside the ZIP file, the data is available in two formats - SQLite database and CSV. For an additional learning experience, we choose the SQLite database. This will be useful as in real-life environments data is stored in a database system (data warehouse, data lake, etc.). We load the database into the R environment and pull the fields needed for the analysis using an SQL statement. These fields we are interested in are Id, Text and Score.

Exploratory Data Analysis (EDA), data cleansing and shaping

Various exploratory data analysis activities are done further to understand the data. The most important aspect is to see whether the scores (1 to 5) are evenly distributed across the dataset. It turns out that this is not the case. The data is skewed toward the highest score 5. To address this imbalance, we use the down sampling technique so all scores are distributed evenly. As stated earlier, since the amount of data is very large and the machine resources are limited, we sample 50% of the data without replacement. This would ensure that no rows are repeated in the sample.

As part of the EDA, we also look at the top 20 words from both the Happy and Unhappy sides. This process is done in an iterative fashion so we can exclude words that don't add value for the analysis. For example words like 'coffee', 'tea', 'chocolate' are removed from the set. Words like 'happy', 'best', or 'disappoint' would be more meaningful in the analysis.

Prior to building the model, the list of words are converted into a corpus (in linguistics, corpus or the plural form corpora is a large and structured set of texts). The corpus will go through a series of data cleansing like removing punctuations, numbers, white spaces, English stop words, and words that are not meaningful in the analysis as previously stated.

Once the corpus is processed, we convert it into a Document Term Matrix (DTM). Each document (or row) represents a review texts. It has words and the DTM lays out the words in a matrix with the occurance of each word in the document as the value in the matrix. Depending on the number of reviews/documents/rows and the length of the review, the DTM can be very large. Hence the reason we only take 50% of the dataset to avoid crashing the machine. Further, we select words that show up at least 5 times in the matrix.

As the final step before building the model, we split the matrix into 80% for training and 20% for testing or validation.

Building the model and evaluating/calculating the accuracy

We use the caret package as the framework for building our models. It is a robust and flexible framework because it can run various algorithms with the single method train() call. We can also easily examine the model produced by caret to see what variables get the high scores of importance in the model. The validation is performed against the test dataset using the predict() method. The confusion matrix in caret generates not only a matrix of True Positive/False Positive/True Negative/False Negative, but it also reports the accuracy, kappa value, sensitivity, etc. The final summary lists the accuracy comparison between two algorithms: Naive Bayes with 10-fold Cross Validation using the e1071 package and Random Forests using the ranger package.

Note: To run the R script, a computer with 16GB or more RAM is recommended.

Exploratory data analysis

Read and examine the Reviews table

```
## [1] "Table:  Reviews"
```

```
## [1] "List of columns (with no actual rows pulled from the database):"
```

```
## 'data.frame':    0 obs. of  10 variables:
##  $ Id                    : int
##  $ ProductId             : chr
##  $ UserId                : chr
##  $ ProfileName           : chr
##  $ HelpfulnessNumerator  : int
##  $ HelpfulnessDenominator: int
##  $ Score                 : int
##  $ Time                  : int
##  $ Summary               : chr
##  $ Text                  : chr
```

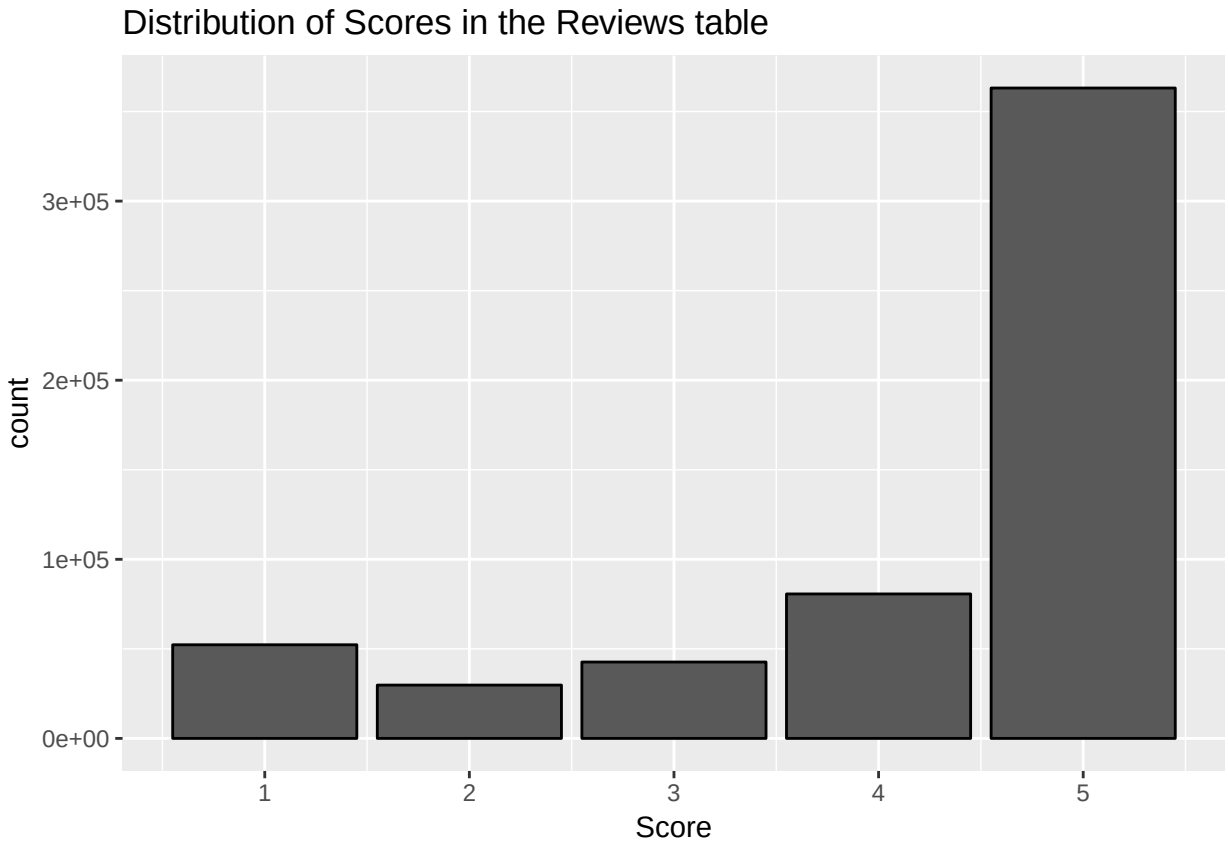
```
## [1] "Let's get the fields needed from the table."
```

```
## [1] "Check the rows and columns of the Reviews table"
```

```
## [1] "Number of rows: 568454"
```

```
## [1] "Number of columns: 5"
```

Distribution of Scores in the Reviews table




```
## [1] "The distribution of Happy vs Unhappy"
```

```
##
##   Happy Unhappy
##  443777   82037
```

```
## [1] "The distribution and proportion of Happy vs Unhappy after down sampling"
```

```
##
##   Happy Unhappy
##   82037   82037
```

```
##
##   Happy Unhappy
##     0.5     0.5
```

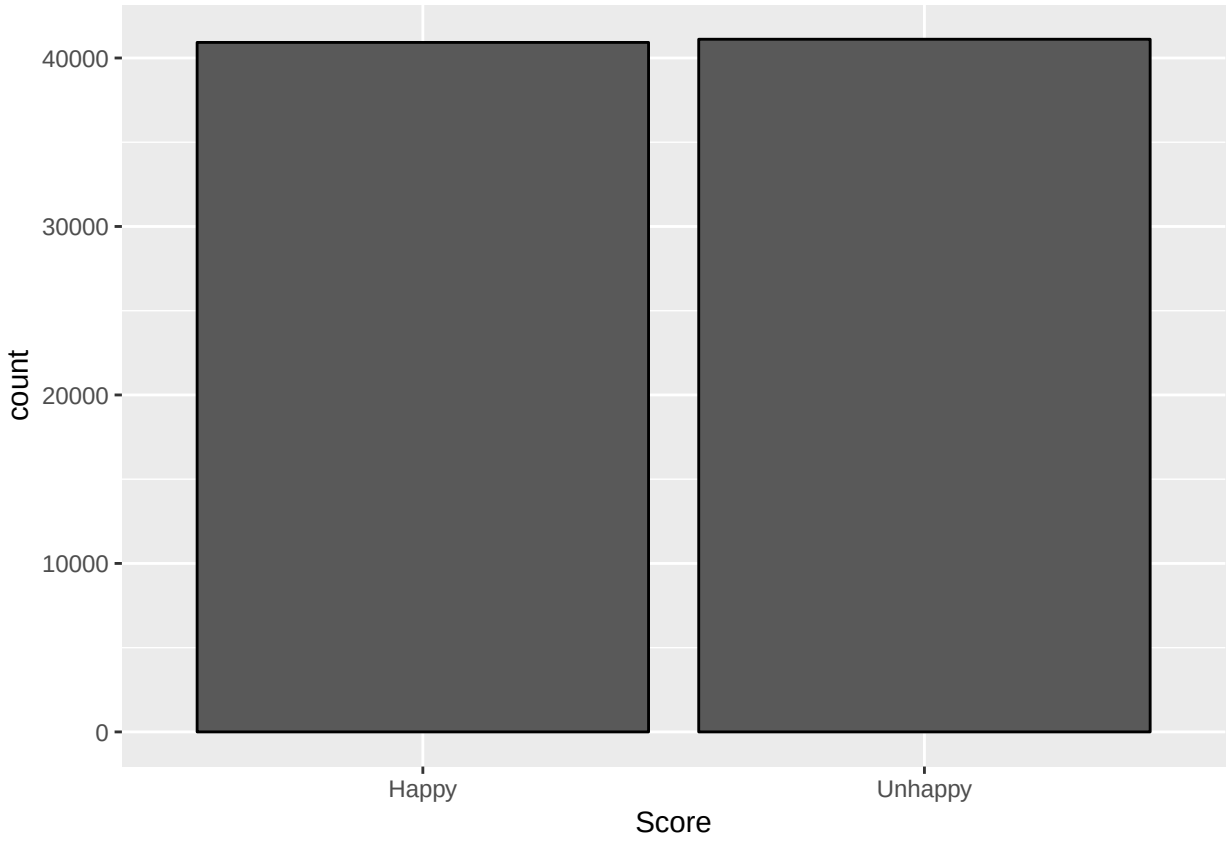
```
## [1] "Because of the large compute power (CPU and RAM) requirements,"
```

```
## [1] "we will sample 50% of the rows in the dataset."
```

```
## [1] "The final distribution and proportion of Happy vs Unhappy after reducing"
```

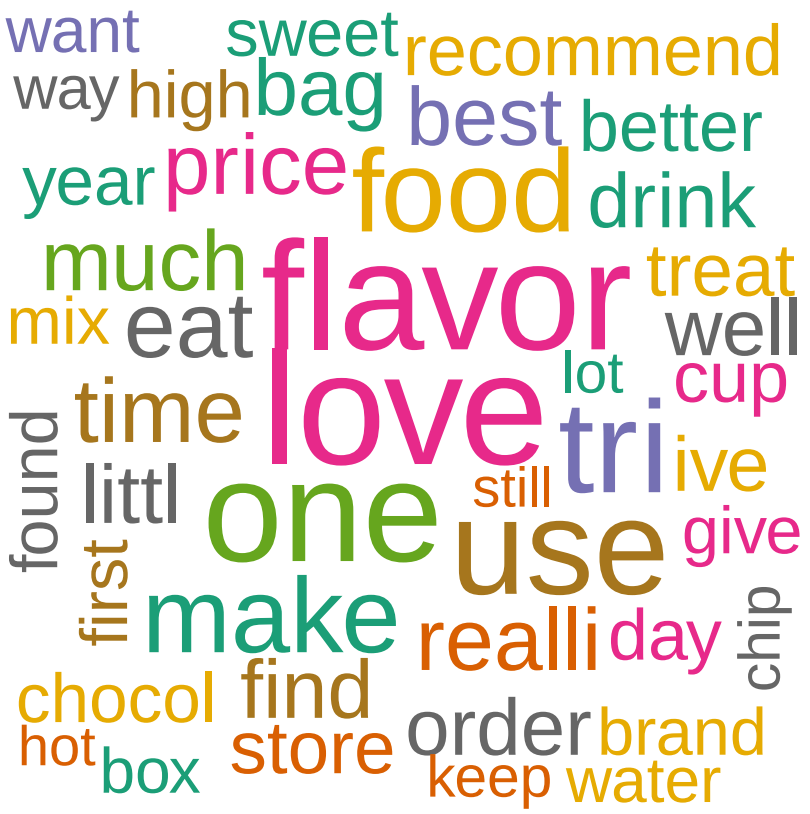
```
## [1] "the number of rows to 50% of the dataset:"
```

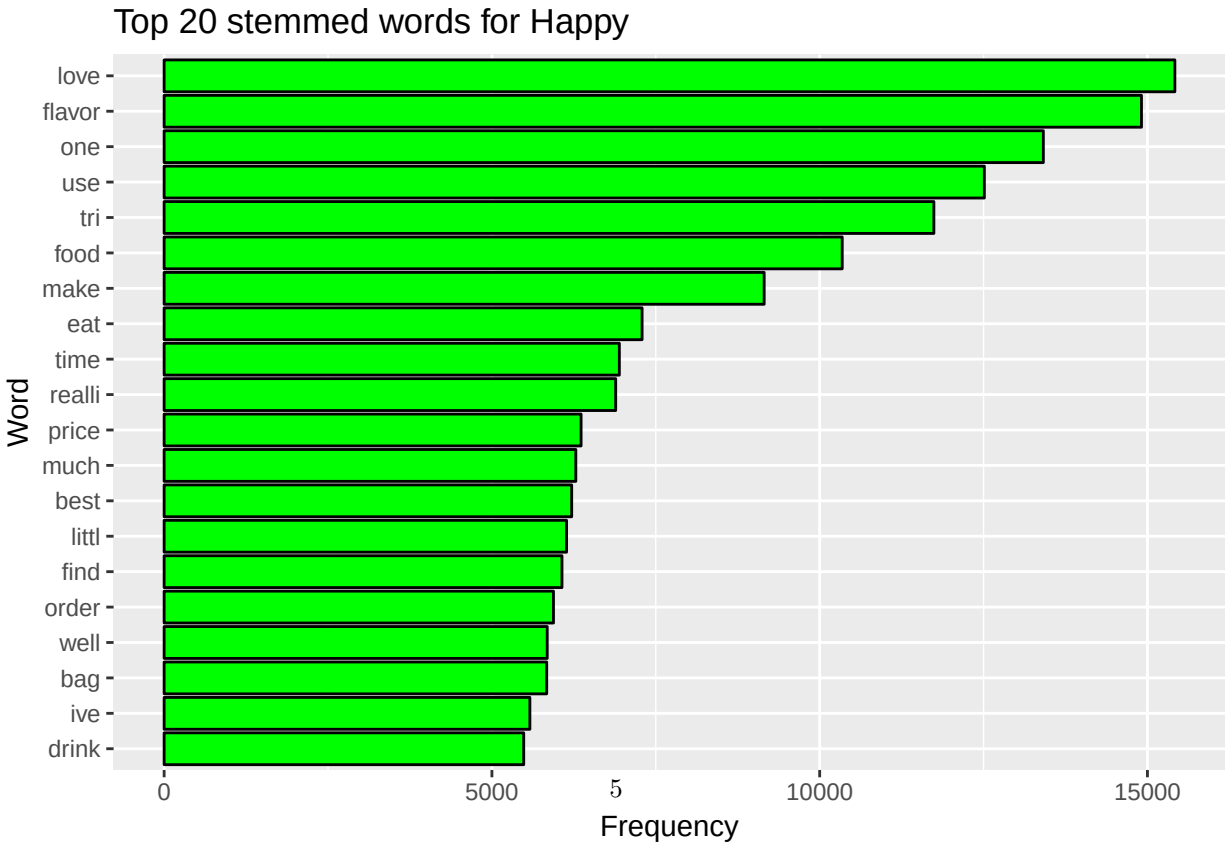
```
##
##     Happy   Unhappy
## 0.4988481 0.5011519
```


40000
30000
20000
10000
0
count
Happy
Unhappy
Score

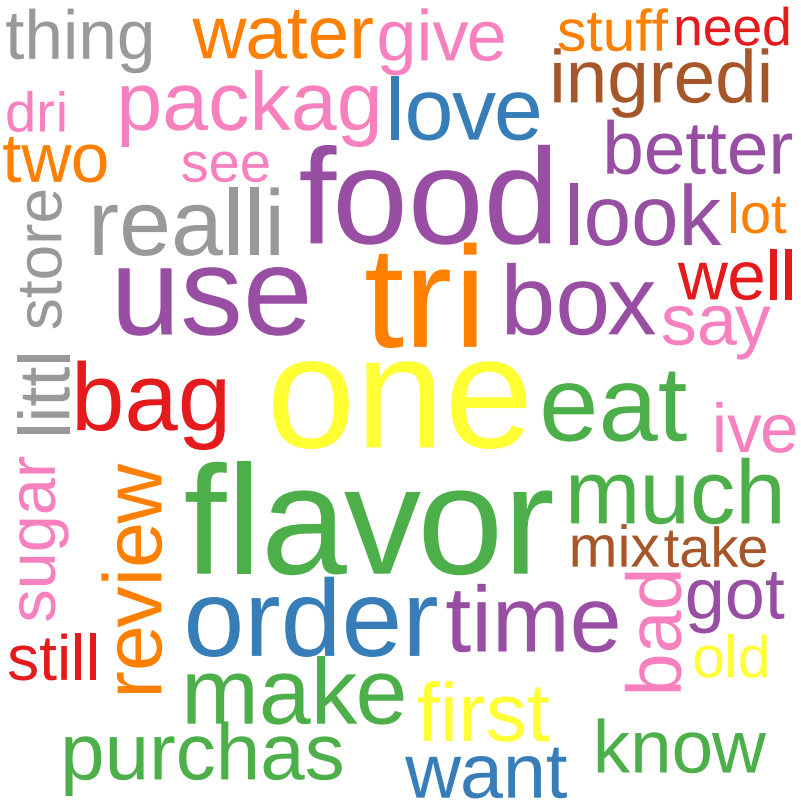
Create the corpora, and generate the word clouds and list of words for each classification

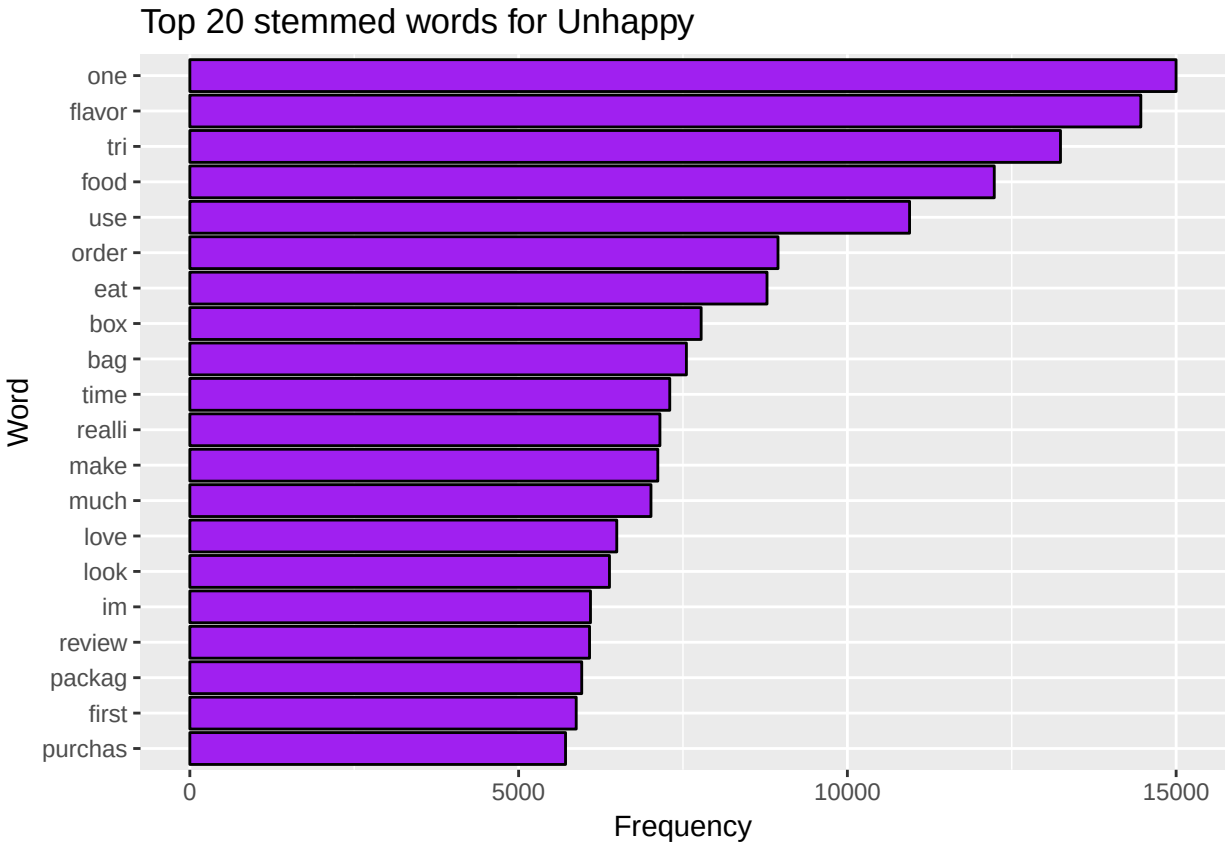
Happy - Word Cloud and Top 20 Words



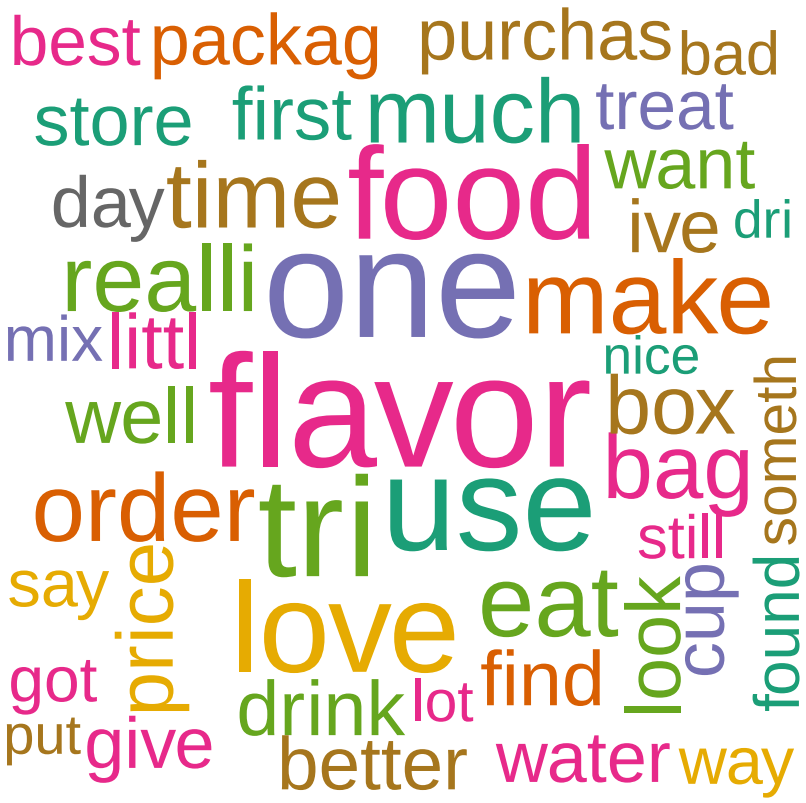


Unhappy - Word Cloud and Top 20 Words





Wordcloud for 100 words from the combination of both Happy and Unhappy



We can see from the word clouds and word frequency charts, even after removing words that we feel are not meaningful in the classification, there is still no clear distinction between words in the Happy versus Unhappy group. Words like 'price' or 'flavor' can go both ways. We could continue with adding words into the list of remove words and re-run the steps, but at this time we will leave it as is and move to creating the models.

Building and testing the algorithms

We convert the corpus to a Document Term Matrix, remove sparse terms, select words that meet the minimum frequency, and then split into 80% training and 20% test.

Naive Bayes with 10-fold Cross Validation Model

```
## [1] "Check the dimension of our Document Term Matrix (DTM):"
## [1] 82037 62431
## [1] "Check the dimension after removing sparse terms:"
## [1] 82037   650
## [1] "Notice the number of terms have reduced."
## [1] "Number of rows in the train dataset: 65629"
## [1] "Number of rows in the test dataset: 16408"
## [1] "The Confusion Matrix against the test data set using Naive Bayes:"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Happy Unhappy
##    Happy    6920    2106
##    Unhappy  1265    6117
##
##                Accuracy : 0.7946
##                  95% CI : (0.7883, 0.8007)
##     No Information Rate : 0.5012
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5892
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8454
##             Specificity : 0.7439
##          Pos Pred Value : 0.7667
##          Neg Pred Value : 0.8286
##              Prevalence : 0.4988
##          Detection Rate : 0.4217
##    Detection Prevalence : 0.5501
##       Balanced Accuracy : 0.7947
##
##        'Positive' Class : Happy
##
```

Variable Importance

We will examine the variables that are the most important in building this model.

```
## ROC curve variable importance
##
##   only 20 most important variables shown (out of 650)
##
##            Importance
## love           100.00
## disappoint      56.16
## best            49.66
## bad             44.12
## review          38.16
## delici          37.17
## money           37.07
## thought         36.53
## favorit         33.05
## perfect         32.59
## wast            28.52
## return          28.40
## box             28.31
## receiv          26.84
## away            26.82
## look            26.55
## price           26.50
## snack           26.36
## nice            25.80
```

```
## packag          24.55
```

Random Forests Model

```
## [1] "The Confusion Matrix against the test data set using Random Forests:"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Happy Unhappy
##    Happy    6937    1129
##    Unhappy  1248    7094
##
##                Accuracy : 0.8551
##                  95% CI : (0.8497, 0.8605)
##     No Information Rate : 0.5012
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.7103
##  Mcnemar's Test P-Value : 0.01551
##
##             Sensitivity : 0.8475
##             Specificity : 0.8627
##          Pos Pred Value : 0.8600
##          Neg Pred Value : 0.8504
##              Prevalence : 0.4988
##          Detection Rate : 0.4228
##    Detection Prevalence : 0.4916
##       Balanced Accuracy : 0.8551
##
##        'Positive' Class : Happy
##
```

Accuracy Summary/Results

The accuracy values from the two algorithms are listed in the following table.

Method	Accuracy
Naive Bayes against Test dataset	**0.7945514**
Random Forests against Test dataset	**0.8551316**

Conclusion

From the comparison of the accuracy values produced by the Naive Bayes with 10-fold Cross Validation versus the Random Forests algorithm, the latter produces a higher accuracy, which is **0.855 (86%)** (about 7% higher than the first model). Further study could be done to try to improve the accuracy by including all rows from the original dataset. This would require a machine capable of processing a large amount of dataset beyond the 16GB RAM can handle. Another option is to adjust the tuning grid parameters in the caret package.