Ragunath Sivakumaran

Background to the task

I conducted this task while simultaneously executing simulations for my Masters' thesis. So at many instances, I was hampered by computational speed and time constraint. Throughout the report, I have indicated wherever I felt I could have done more tasks if I were to perform the same task in a different scenario. Also the report was compiled to demonstrate how I have proceeded with the task and not as a formal report written to the client. I have used R for text mining and Matlab for training a classifier. The code written has been embedded in the Appendix.

Introduction to the Dataset

Amazon Fine Food Review comprises 568,454 food reviews left till October 2012 by Amazon users. The dataset contains 10 features such as ID, Product ID, …, Summary, Text. A complete description of the data can be found at https://www.kaggle.com/snap/amazon-fine-food-reviews . Of the 10 features, the following two features are important to this analyses are

- **Summary -** a brief summary of the review and
- **Text -** the text of the review

For the competition, I tried to build a predictor of the Summary feature on the basis of the Text feature. It is to be noted that both the features are texts. Also, <u>I tried to make a single predictor combining both the text features, but this is not yet completed</u>.

Both the features (Summary and Text of the review) are texts in general and written in natural language which might be decisive as there may be a lot of variations which have to be accounted for. After the initial review of the two features, it was observed that the summary as described comprised a brief overview of the review text and in fact it had the same words being repeated in the text for many observations and the words quite relevant. For example, the review text "Great taffy at a great price…. If you're a taffy lover, this is a deal" had a summary "Great taffy". This indicates there definitely is some correlation between the summary and the review text, and it has not been chosen automatically.

However, for the sake of predicting the summary, predicting actual words would be quite cumbersome and hence the summary should be given some meaningful tag. The words in the summary can be split into many words and can be assigned a meaningful tag or the summary as a whole can be tagged. The former can be made by giving the summary a general tag and a specific tag which identifies something more specific than the general tag, thus making it a multi-level classification problem. The latter is a generalization and a singly type of tag can be used to represent the summary. For the sake of simplicity, the latter method, i.e. having a tag (or a category) to represent the type of summary is chosen instead of a multilevel summary. The next challenging concern would be to create these tags automatically as no other feature can be used except the two bulleted above.

Creating Summary Categories

As the summary is given in natural language instead of a category, it requires pre-processing so that a common minimum generalization could be achieved. This is done in order to create a meaningful knowledge base that is similar and to avoid unnecessary variations that do not add any meaning to the summary. For this task, I have used the package "tm" in R for text mining. First, all the summaries were converted into lower case alphabets so that there won't be any differences between the same words. Next the punctuation marks and numbers were removed from the text as they are less meaningful compared to the words for this analysis. Then the common stop words such as is/was/may were removed as they are necessary for sentence construction, but for analysis they add very little value. Finally, the words were stemmed so that words which represent different tenses such as running/run could be taken as the same. After this, the words which occur at least twenty times were inspected as all the pre-processing steps were conducted by inbuilt functions and there are chances that they might miss certain obvious aspects. Once, the frequent words used for the summary have been inspected, I decided that quite a few important words were required to be changed. The word awesome was replaced with the word amazing (both were in the summary) as they represent the same connotation. Likewise, yummy with yum, cats with cat, dogs with dog, excellent with fantastic, loves with love, tasting and tasty with taste, and treats with treat were all replaced. Once this was done, I generated a word cloud to visually inspect if there are any obvious words that are still left to be pre-processed. The generated word cloud is give in Figure 1.

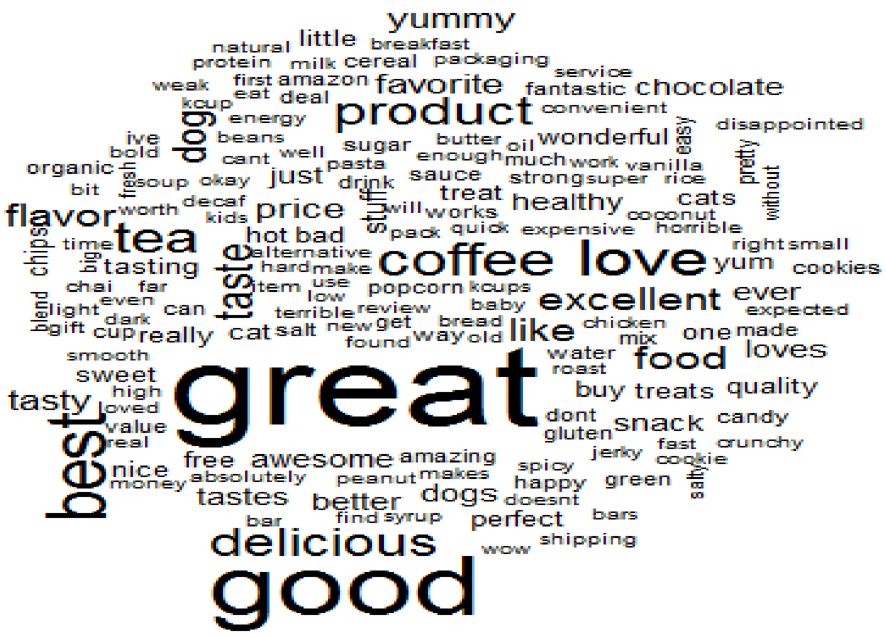

Figure 1. Word cloud of the summary feature after pre-processing the summary

As can be observed from figure 1, the words mostly represented some meaningful information which could be useful in categorizing the reviews. I created the Document Term Matrix (DTM) which represented the frequency of the words uttered for each observation (review). This DTM was observed to be very sparse and any algorithm to categorize the summary wouldn't be effective in such a sparse matrix, so I decided to remove those words. I removed words that are sparser than 99.5% which resulted in a total of 62 words. I wanted to remove words that are sparser than 99.9% that resulted in 172 words for each review which by initial inspection appeared to have a very good representation. Although I wanted to retain 172 words, the operation to categorize ran out of memory owing to a large dataset and hence I retained only 62 words.

Back to the task of categorizing the reviews, there are no targets given and hence I resorted to unsupervised learning for categorizing the review. I chose clustering as the unsupervised task based on which I will categorize. I would have liked to check both hierarchical clustering and k-means clustering, but for a huge record like the current task, hierarchical clustering requires a lot of time and enormous memory for the program which is not feasible for the current task. For this reason, I decided to use k-means clustering. The challenge in k-means clustering is to select the number of clusters whereas for hierarchical clustering, I could have chosen the cluster based on the vertical distances. So I selected k-means clustering and I iterated from 1 to 25 clusters so that I could check for the within cluster variance represented at each case. A scree plot representing different clusters on the x-axis and the within cluster variance on the y-axis pertaining to the iteration is given in figure 2.

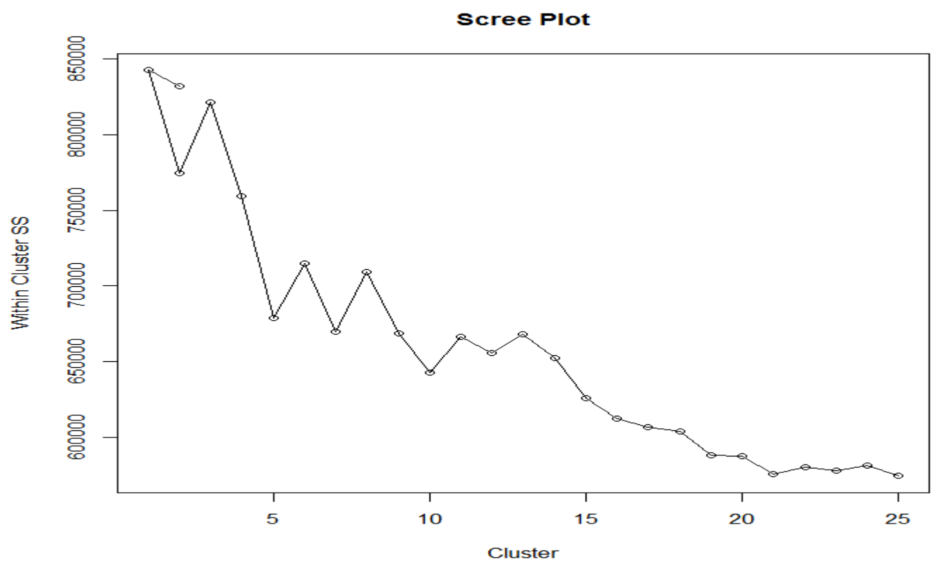

Figure 2. Cluster iteration and within cluster variance

As can be observed from the figure, an elbow (representing less change in variance for every cluster after) could be spotted for clusters 5, 10 and 20. I chose 5 clusters as I believe this represents a good variation to categorize the summary. Selecting 10 or 20 clusters might improve the clustering efficiency, but might turn out be an overkill for the prediction.

5 clusters represented 25% of the within cluster variance when compared to the total variance. Although this accounts for a meagre amount, I still proceeded with this owing to simplicity. A quick look at the exploration gave the following results for the clusters.

Number of Observations	Cluster
35829	1
19769	2
396199	3
47785	4
68872	5

Table 1. Cluster Details

From table 3, it is evident that most variables are assigned to cluster 3. Although this leads to a strong suspicion that the clustering might need some improvement, I have proceeded with it as other unsupervised learning methods require more memory and computational power which I could not afford at this moment. I planned to come back to this once I remove multicollinearity with the data to test if a sample of the cluster has a good discriminative power. So I proceeded with creating a document term matrix for the predictors.

Creating Term Frequencies for the Text Feature

As the text feature is the predictor for the summary feature, I created the frequencies of words through a document term matrix by means of the same procedures I followed for the summary feature in the previous section. The only difference is here I used sparsity of 95%, i.e. only terms which are sparser than 95% were included in the document term matrix. The document term matrix thus created had **101 predictors**.

Data for Analyses

From this step onwards, I have worked only with the clusters and document term frequencies. So, **the response/independent variable for analyses would be a multinomial classification comprised of levels 1, 2, 3, 4 and 5**. The clusters correspond to categories of summary. The **predictor/independent variables would be the frequency of words used in the review text for each review**. With regard to the predictors, only those words which are sparser than 95% when tokenized are considered for the analyses. There were **101 predictors** each indicating their frequency of occurrence in each cluster.

Data Division

The dataset comprises 568,454 examples and 101 predictors. I randomly divided the data into training set comprising 70% of the examples, validation set comprising 15% of the examples and test comprising the remaining 15% of the examples. In many cases like Cross validation and Ensemble methods, the validation was performed along with the training and so I have combined the training and the validation sets (total-85% of the examples) for these cases.

Multicollinearity and Data Reduction

Now that I have grouped the summary feature into a meaningful classification of clusters and I have created a document term matrix of predictors, I have to check for multicollinearity among the data. The words used might be correlated with each other and hence it will be a good choice to check for multicollinearity. A simple correlation coefficient among the predictors revealed there is less correlation among the predictors. The correlation coefficient is very simplistic to measure multicollinearity for a large dataset like this as the dataset is very sparse and the correlation coefficient would definitely be very less. Hence I decided to use GLMNet with cross validation to check if the data could be reduced. GLMNet if regularized will reduce the dimension even if there is no multicollinearity (which is undesirable) and a careful approach is required at this stage so that important predictors are not lost due to over regularization. This will lead to loss of vital information which might affect the quality of the prediction to be made.

Using GLMNet at this moment seemed the best option as the primary task is not data reduction, but to develop a predictive model (I have explained about the steps taken for data reduction in the next section). Nevertheless, any correlation among the predictors should be avoided as the training methods which are going to be used might assume independence of the predictor variables.

One of the best ways to avoid false detection of multicollinearity in data reduction methods such as Elastic Net, GLMNet, etc. is to use cross validation and check the Mean Square Error (MSE) for the validation set. An important aspect to be concerned while resorting to GLMNet is the choice of regularization value. An over regularized model could cause loss of valuable information; the error acquired would be propagated all throughout the analysis and hence this parameter should be carefully considered. The regularization parameter should be chosen in accordance with the MSE of the validation set during cross validation. Conducting cross validation on a huge dataset with more than half a million records is computationally intensive as well as time consuming and hence I wanted to experiment with a small sample of the large dataset. The data had to sampled such that the sample could also be used for tuning the parameters for learners and not just for multicollinearity.

Samples Created

Actual Representative Sample

I wanted to create a sample that mirrors the same proportion of clusters in the actual data. Hence I created a sample which I will refer as "Actual Representative Sample" which comprises 4727 examples from cluster 1, 2608 examples from cluster 2, 52273 examples from cluster 3, 6304 examples from cluster 4, 9086 examples from cluster 5. In total the actual representative sample contains 75000 examples and the proportion of examples belonging to each cluster is same as that of the original data. This could prevent the user-bias being introduced by presenting a balanced sample while the actual data is unbalanced.

Equal Representative Sample

After careful consideration, I decided to pick 15000 samples from each cluster pertaining to the training set and the remaining form the sample test set, i.e. 75000 examples in total for the sample set out of which 15000 would belong to each cluster. This will be referred to as "equal representative sample" in this task. This way the learners could be given equal number of examples to learn while tuning the parameters.

The equal representative sample is balanced whereas the original data is unbalanced (refer Table 1). Although the equal representative sample might tune the learners well, the tuned values might give erratic predictions owing to the unbalanced nature of the actual data. <u>It is to be noted that the samples are chosen just for tuning the parameters or for exploration, and not for any training for which the results will be concluded. Although this may not be the best way, I decided on it owing to its simplicity.</u>

Data Reduction with GLMNet

The GLMNet algorithm uses the parametrized unconstrained formulation of Elastic Net whose cost function comprises the least squares objective and a regularization term. As the clusters correspond to multiple classes from 1 to 5, the GLMNet formulation for multinomial logistic regression is appropriate for this task. The GLMNet task would be to minimize the cost function so as to select the set of coefficients (by eliminating multi collinearity) representing each class. I decided to go with α=1 option for GLMNet which corresponds to the Generalized LASSO formulation as my intention is to detect for multicollinearity and LASSO makes all but one of the correlated coefficients 0 (which is easy to spot). The interpretation of multicollinearity pertaining to a Gaussian response for LASSO is straight forward; in case of multicollinearity all except one of the coefficients corresponding to the correlated variables will be set to 0, i.e. the non-zero coefficients at the end of the minimization exercise correspond to predictors which are not correlated. In the case of multinomial logit, the coefficients are generated for each class (the classification level) and for each predictor, i.e. the generated coefficients correspond to each class and a predictor pair. So the total number of coefficients to be dealt with is $k \times p$ (if k corresponds to the number of classes and p corresponds to the number of predictors). In this task, the GLMNet formulation will return 510 coefficients (5 per predictor). The only way to check for multi collinearity is to check if all the 5 coefficients corresponding to a predictor are simultaneously set to 0 in which case the predictor would be rendered meaningless.

For the purpose of trying out different regularization terms an auto-grid which is available in the GLMNet package is used. The equal representative sample and the actual representative sample was fitted with the regularized multinomial logistic regression. I used 10-fold Cross Validation technique in which the data to be fitted is divided into ten parts and nine parts would be fitted during each iteration. The fitted model would be validated with the left out part which is called validation set. This will be continued till all the different parts are validated once and the MSE for the validation set would be averaged. This process would be repeated for all the values in the regularization grid.

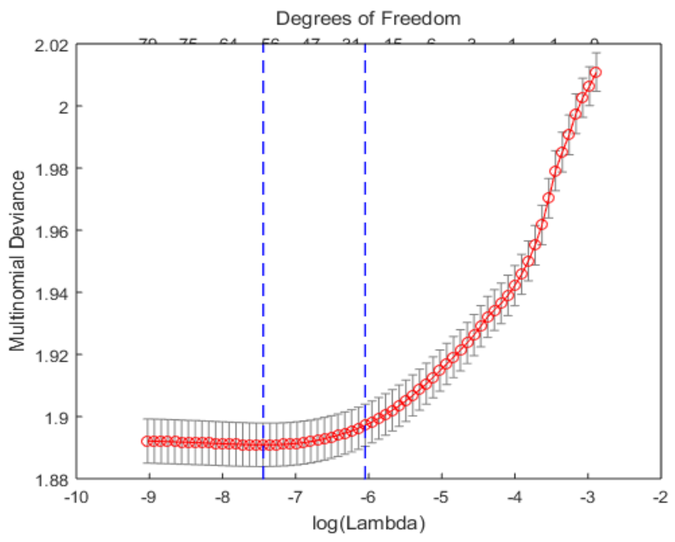

Figure 3. Regularization Grid Vs Deviance

When I observed the averaged MSE values pertaining to the models fitted for the regularization grid, some of the regularized models had lower MSE than relatively low-regularized values. It appeared that the predictors are not independent, i.e. it appeared that the data might have multicollinearity. The regularization parameters against the deviance is shown for the validation set corresponding to the equal representative sample in figure 3. The x-axis comprises the regularization grid shown in log scale and the y-axis corresponds to the deviance.

Since both the samples showed signs of multicollinearity, I fitted the regularized logistic regression model with the entire training (and validation) data by using the Cross Validation scheme mentioned before. Instead of 10-fold Cross Validation, I used 5-fold Cross validation to save time as the difference in averaging out would be very minimal. Both the equal representative sample and the actual representative sample had the least MSE for a regularization value of 0.0002 after the cross validation. So I trained a multinomial logistic regression for the entire training data with a regularization value of 0.0002. Surprisingly none of the predictors had coefficients set to 0 simultaneously for all the classes although both the samples pointed to the contrary. At this stage, I didn’t have any evidence to show multicollinearity among the predictors. At this stage I wanted to select a regularization grid myself with larger variability and another grid of α varying for 0 to 1, i.e. form a grid of α varying from Generalized Ridge Regression to Generalized LASSO, but the data reduction stage took a lot of time and I stopped checking for multicollinearity owing to time constraint. The results might vary if I take a different grid, but I assumed independence among predictors owing to lack of evidence. Hence I will use the same 101 predictors created initially.

Classification

The next important step is to classify the examples by using the clusters as dependent/response variables and the features as the independent variables/predictors. Considering the size of the training set, dimension of the data, time constraint and computing power with my laptop there are two classification techniques that I would consider for this task. They are Naive Bayes Classification Technique and Ensemble of Classification Trees to be built using Random Forest algorithm. I have explained these two techniques further.

Naive Bayes Classification

The Naive Bayes algorithm is one of the simplest algorithms implementing a classifier. The reason that I chose to implement this first is because of its simplicity. It is very fast in convergence and might give a good idea of what to expect in case if it fails. The Naive Bayes algorithm can be explained as follows: For each feature that can be termed as X_i, i=1, 2, …, p and each response class that can be termed as Y_j, j=1,2,3,4 and 5, the conditional probability $P(X_i | Y_j)$ and the unconditional probability $P(Y_j)$ are determined. When a new prediction is to be made the probabilities $P(Y_j | X_i)$ are computed according to Bayes theorem by making use of the probabilities determined already. In other terms, Naïve Bayes estimator computes the posterior probabilities by means of likelihood P(X | Y) and prior P(Y). It is simple and thus converges faster.

One of the fundamental assumptions for the Naïve Bayes Classifier is that it assumes the predictors to be independent. In the current task, I didn't find any evidence of multicollinearity and so I confirm this assumption for the predictors in this task. One of the advantages of Naïve Bayes classifier is that it is not required to have a huge training set and this will be very handy for the two samples I have created.

I used the equal representative sample and the actual representative sample to train a Naïve Bayes model one each for both the samples and tested on the test sets corresponding to each sample. The classification accuracy was 26.7% for the former and 26.2% for latter. This indicates a poor training of the classifier. I conducted a 5-fold cross validation for both the models trained and averaged the classification accuracy produced for validation set at each fold. It still didn't improve the results. It was quite clear that Naïve Bayes doesn't have the predictive ability for this task.

An important aspect to be considered for the poor learning ability is that the set of independent variables are very sparse. For a multinomial response and a sparse dataset, it might turn out that the likelihood of an example belonging to a particular classification is 0 (owing to the nature of likelihood calculation) which is a very strong statement. In other words, just because a few words are not present in the example it cannot be said that it will not fall in a particular classification and hence this would have proved to be quite decisive for the failure.

Another important factor to be considered is the independence assumption which I had confirmed earlier. Although there was no sufficient evidence to prove multicollinearity, if the parameters were properly tuned, the results could have been better. But as I stated in the previous section, owing to the time and dataset constraint I couldn't conduct much beyond the basic analyses. Although this assumption might have been violated, I don't think this alone caused such a poor prediction performance. The last point of concern would be to specify individual distributions to the predictors which might have improved the predictive ability.

The Naïve Bayes classifier could be thought of having a linear separator and text document frequency terms have a very high chance of being not linearly separable and this issue will be addressed by the other method which follows. Laplace correction to the Naïve Bayes Classifier could be applied, but again the prediction is very poor that it is highly doubtful that Laplace correction would increase the classifier accuracy.

Random Forest Classification

As the Naive Bayes algorithm had a poor prediction capability, I resorted to use Ensemble methods. An ensemble is a set of predictive models each predicting the same target variable from the same input variables. One of the main advantages of Ensemble methods is that a group of weak learners together can be combined to form a strong learner and as observed from the Naive Bayes prediction, the learning ability has been very weak. The weak learning ability can be attributed to the fact that the data to be trained and predicted is very sparse and a single learner would always misclassify the data. Instead a group of learners acting on the data and each producing its own classification so that the best of the chosen ones could be used and such a learner would minimize the variation although they might introduce some bias. Hence, I chose Classification Trees to be used in this task and Random Forest algorithm to build the Classification Trees. As known, Classification Trees represent an input domain to an output domain with nodes of the tree being annotated with a test. In Random Forest algorithm, for the test to be annotated with each node a set of randomly chosen attributes are considered and the attribute which has maximum information gain is filled from top to bottom. The number of attributes to be considered in this task was $\sqrt{101}$ which is the default value.

The Random Forest algorithm makes uses of Bootstrap method, i.e. a random sample of a particular size with replacement is taken to build a classification tree and the random sample is called bootstrap sample. Once the bootstrap sample is chosen, then a certain number of random samples (as explained in the previous paragraph) without replacement of the predictors/features are chosen. The Classification Tree is built with a chosen combination of bootstrap samples and predictors. Similarly, many Classification Trees are built and the output is chosen by a majority vote over the set of trees. The samples left out for each bootstrap sample is called out-of-bag samples and these will be very useful in choosing the right set of parameters to be applied.

As the size of training set is very high and building trees with a huge data set is time consuming as well as computationally intensive, I chose the equal representative sample and the actual representative sample described before so that I could train the data on the samples to select the parameters. The important parameters (some parameters are not listed) to be chosen while building a decision tree with Random Forest algorithm are

- To select the number of trees to be built
- To select the number of bootstrap samples
- To select the fraction of predictors to be used in each tree
- Pruning Level – the depth of each tree to be considered

I would have liked to tune the parameters by forming a grid for each parameter and selecting the best parameter by building the models for all the values in the grid. This would have been very time consuming although the results would be very reliable, but I do not have the time and machine with computational power to do this. Hence, I decided to select the number of trees and bootstrap samples with a simple search, and the rest with the default values.

For selecting the size of the bootstrap sample, I varied the fraction of sample in bag to the total size for values in the interval [0.66, 0.75, 0.85]. This is a coarse interval and doesn't cover a wide range of variability, but owing to the time constraint, I selected this small interval so that I could choose the best value. For each value in the interval, I built a decision tree pertaining to the equal representative sample and actual representative sample and checked for accuracy in the test set. The results for 0.66, 0.75 and 0.85 were 74.33%, 74.59% and 74.72% for the equal representative sample and 74.28%, 74.55% and 74.65% for the actual representative sample. Although there isn't a huge difference, I decided to go with the simple test and hence I selected 0.85 to be the fraction of bootstrap sample for the training set. The results were more encouraging than the previous classifier used for this task.

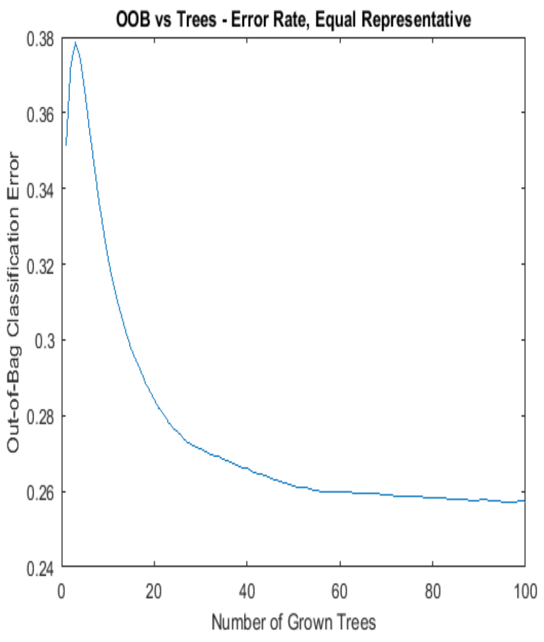

Figure 4. Error Rate Equal Representative Sample

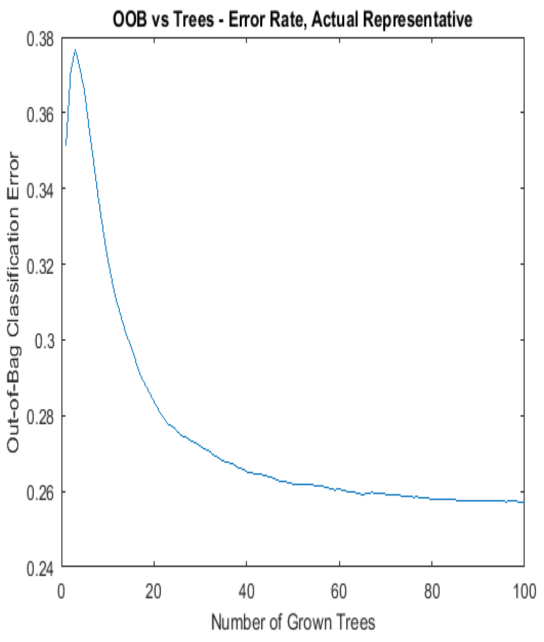

Figure 5. Error Rate Actual Representative Sample

From the equal representative sample and the actual representative sample, I chose 100 trees to be built and I observed the out-of-bag classification error for all the trees built. The out-of-bag classification error for different trees are shown in figure 4 and figure 5, where the number of trees are shown along the x-axis and the out-of-bag classification error are shown along the y-axis. As can be observed from the figures, the reduction in out-of-bag classification error for more than 40 trees is very less and hence 40 would be an appropriate number of trees to be built for the training set.

So the parameters chosen for building the classification trees are:

- 85% for the bootstrap sample.
- 40 trees, each representing different bootstrap samples (with duplication among the samples).
- The rest of the parameters such as the fraction of predictors to be chosen for each tree, are set to the default value. The fraction of predictors to be used in each classification tree by default is the square root of the number of feature which in this task would be √101. Also, I didn't choose any pruning as pruning in general would affect the prediction capability unless chosen carefully.

The training (and validation) set comprised of 483186 examples chosen at random and the test set comprised 85268 examples not used for training. It is to be noted that the while training the classification trees, the in-bag and out-of-bag samples are similar to the training and the validation set sand hence the training set with 483186 examples can be seen as combination of the training set and the validation set but owing to the nature of building the trees, they are combined.

Once the trees were built with the chosen parameters, they were tested with the test set and classification predicted from this process is compared with the actual classification from the test set. **It was observed that 80.11% of the classifications in the test set were identified correctly.** The Classification Trees had a good accuracy on the test set and thus validates some of my assumptions stated earlier. Accuracy is not the only measure that could grade the classifier. I wanted to create an ROC curve for each classification level with Sensitivity and specificity levels so as to have a better view, but owing to time constraint I haven't done that here.

An important aspect to be validated for the trained classification trees is by computing the Mean Classification Margin. For each observation, the margin is defined as the difference between the score for the true class and the maximal score for other classes predicted by this tree (in this case the response in the test set). The cumulative classification margin uses the scores averaged over all trees and the mean cumulative classification margin is the cumulative margin averaged over all observations, i.e. every new element in the returned array represents the cumulative margin obtained by including a new tree in the ensemble.

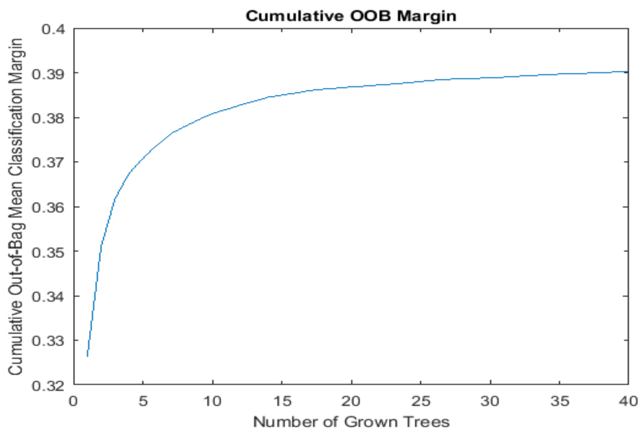

Figure 4. Cumulative OOB Margin for the trees

The cumulative out-of-bag mean classification margin for the test set is plotted in figure 6. In the figure, the x-axis contains the trees built and the y-axis contains the cumulative out-of-bag mean classification margin. The gradual increase in the Mean Classification Margin validates the earlier accuracy, i.e. the training was successful. As the accuracy in the test set and the gradual increase in the mean classification margin for the same, I would conclude the Ensemble method built a good learner for this task.

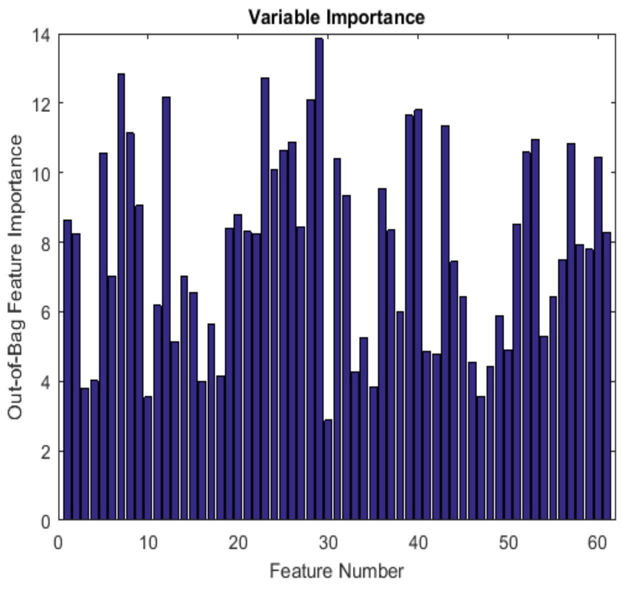

Figure 5. OOB Feature Importance

For every tree built, each feature is permuted among the training data and the out-of-bag error is computed again on this changed data set. If there is increase in the error, then it indicates that the variable is important. This measure is computed for every tree, then averaged over the entire ensemble and divided by the standard deviation over the entire ensemble. The measure can be termed as out-of-bag feature importance. The out-of-bag feature importance for all the predictors are given in figure 7. The figure contains x-axis indicating the feature number and y-axis indicating the out-of-bag feature importance. It is to be noted that a large number of features have high out-of-bag feature importance and hence we can conclude the predictors represented a good collection. Some of the important features (out-of-bag feature importance > 7) as indicated by the model are "**also, always, amazon, bag, better, brand, buy, can, cup, day, doesnt, drink, enough, every, favorite, find, flavors, food, found, get, makes, one, right, since, small, still, store, stuff, tea, time, used, water, way, well, will, without"**

Improvement Scope

Although the final model has been described, the following points convey some aspects of improvement which I would have accounted in the analysis if I would have had more time and better computational power.

- The first aspect I would have liked to improve is clustering. I would have tried with more clusters and certainly I would have tried hierarchical clustering for the data.
- I would have tried exploring the possibility of multilevel clustering, i.e. having more than one level to combine the data.
- The next aspect is choosing the correct grid while using the GLMNet. I would have liked to train the models for the grid values I would have selected instead of the value chosen by the software and another grid for α varying from 0 to 1. The best parameter for regularization would have been the one that gave the least MSE on the validation set for the combination of the two grids.
- I would have liked to conduct a test for distribution pertaining to some of the important features and would have used it while fitting the Naïve Bayes classification. I would have also tried Laplace correction for Naïve Bayes setting.
- I would have liked to make a finer grid for choosing the bootstrap fraction.
- I would have liked to select the following parameters from a grid
    - Number of parameters to be considered while building a classification tree.
    - Consider pruning the tree.
- I would have used ROC for multiclass (one vs all) for each level. That would have given a better idea about the classifier.

I believe some of these improvements will improve the accuracy of prediction. Thank you!

Appendix

Matlab code – Analysis and predictor building

Click the attachment below to open the Matlab code _ pin shaped attachment

R Code – Text Mining

Click the attachment below to open the R code pin shaped attachment