

**SCHOOL OF BIOLOGICAL SCIENCES**

BS3033: Data Science for Biologists

Group Project Report

Team 3

Soh Kai Jin U1740760K

Justin Chia Cheng Ze U1740466J

Chong Shu Chen U1740627B

Lim Rui Jia, Hannah U1640334B

Contents

[1. Introduction 3](#_Toc25104121)

[2. Methods 4](#_Toc25104122)

[2.1 Exploratory Data Analysis 4](#_Toc25104123)

[2.2 Data Pre-Processing 6](#_Toc25104124)

[3. Results 7](#_Toc25104125)

[3.1 Ingredient Word Cloud 7](#_Toc25104126)

[3.2 Data Partition 9](#_Toc25104127)

[3.3 Data Cleaning of Ingredient Names 9](#_Toc25104128)

[3.4 Decision Tree Model 10](#_Toc25104129)

[3.5 Random Forest Model 12](#_Toc25104130)

[3.6 R Shiny Dashboard: cuisineR 14](#_Toc25104131)

[4. Discussion 16](#_Toc25104132)

[4.1 Predictive Model Performance 16](#_Toc25104133)

[4.2 Strengths and limitations of decision tree 19](#_Toc25104134)

[4.3 Limitations 22](#_Toc25104135)

[4.4 Uses 22](#_Toc25104136)

[5. Future Directions 23](#_Toc25104137)

[6. References 24](#_Toc25104138)

# 1. Introduction

Ingredients form the crux of every dish. A single ingredient could be involved in many different recipes belonging to different cuisines, or it could be intimately tied to a specific national cuisine. In this report, a cuisine is defined as a set of dishes originating from a particular country. For instance, dishes from Korea would be classified under Korean cuisine, Chinese cuisine from China and so forth. Among the multitude of recipes being added online onto public domain food websites, many of them are variants of pre-existing recipes. These online recipes not only provide fundamental information such as the ingredients and type of cuisine, but could also reveal interesting patterns of common ingredients. Manual classification of each new incoming recipe accurately is taxing and prone to error.

Therefore, the main aim of this study is to implement and analyze current classification models to identify and select the best model that can predict the cuisine with a high accuracy given an input of ingredients. In application, our cuisine classifier could possibly be implemented to complement online grocery shopping websites, recommending shoppers cuisines based on ingredients they are about to buy or bought.

Successful classifying of the type of cuisine from a list of ingredients would be very useful in complementing the existing method of food pairing. In the conventional food pairing approach, different foods are paired together using the aroma profile of each food. Food pairing is further supported by the food pairing hypothesis which states that food sharing flavor compounds were more likely to taste well when tasted together. [1] In our report, we are not pairing ingredients, but rather intend to better classify the type of cuisine given the recipe containing the ingredients. These recipes could potentially be grouped together according to the type of cuisine after successful classification. Hence, individuals who displayed a preference for a particular Italian cuisine would be shown recipes for other Italian cuisines.

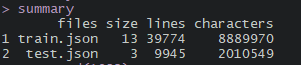
Text mining is defined as "the discovery by computer of new, previously unknown, information by automatically extracting information from different written resources". [2] While text mining can be applied to many domains, such as natural language processing (NLP), many issues arise when analyzing a huge volume of unstructured data. To avoid this, during pre-processing, various rules and regulations are needed to standardize the text in order to make subsequent manipulation and analysis efficient. [3]

In this project, we also aim to demonstrate the extensive applications and usefulness of text mining with a machine learner. Our team similarly had to tackle the issue of unstructured data commonly encountered in text mining and NLP. In essence, this report serves to explore our thought process and rationale during the construction of the cuisine predictor. All coding was done in R language using the RStudio interface.

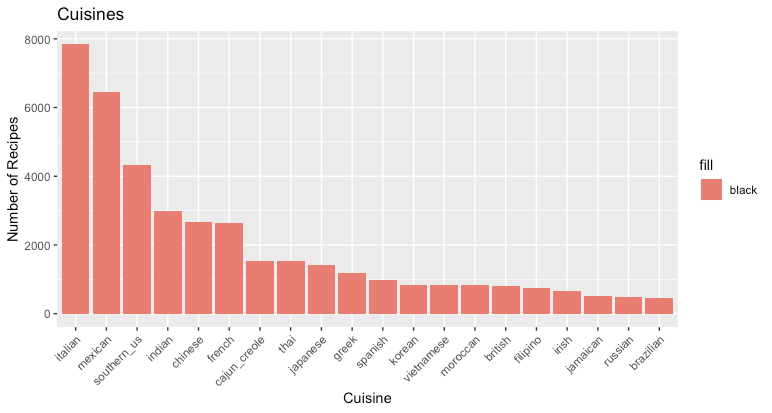
# 2. Methods

## 2.1 Exploratory Data Analysis

The dataset “train.json” was downloaded from Kaggle.com [4] to serve as both training and testing data sets to develop the machine learning model used in natural language processing (NLP). The dataset is structured such that recipes appear as lists, with ingredients and their respective cuisine being members of that list. A preliminary analysis identified the file sizes (megabyte, MB), the number of lines and characters in each dataset. This preliminary analysis showed that the dataset is a large data set. The total number of recipes within the dataset is 39774.



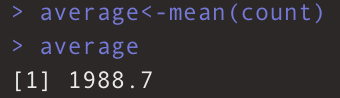
*Figure 1. Summary of train.json file.*



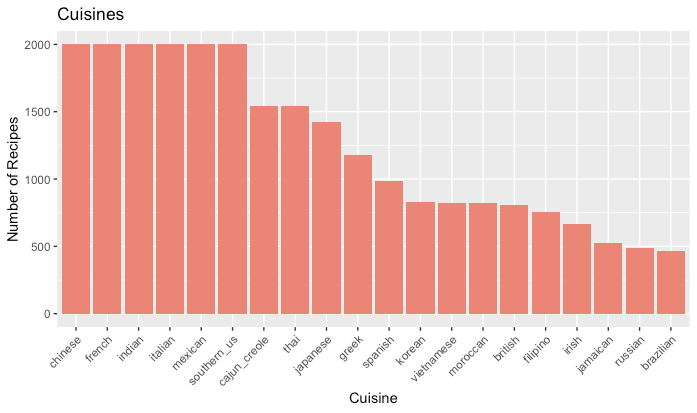
*Figure 2. Histogram of the number of recipes of each type of cuisine from the dataset train.json. Note that Brazilian, Russian and Jamaican are the lowest represented cuisines, while Italian cuisine is the highest represented.*

In order to further visualize the underlying distribution of the data set, the number of recipes of each type of cuisine is plotted in the form of a histogram as seen in *Figure 2* above. The number of recipes inputted for training would have implications on the training of the model later. It is understood that a cuisine with a larger number of recipes available for training would be predicted more accurately when validated using the testing data set.

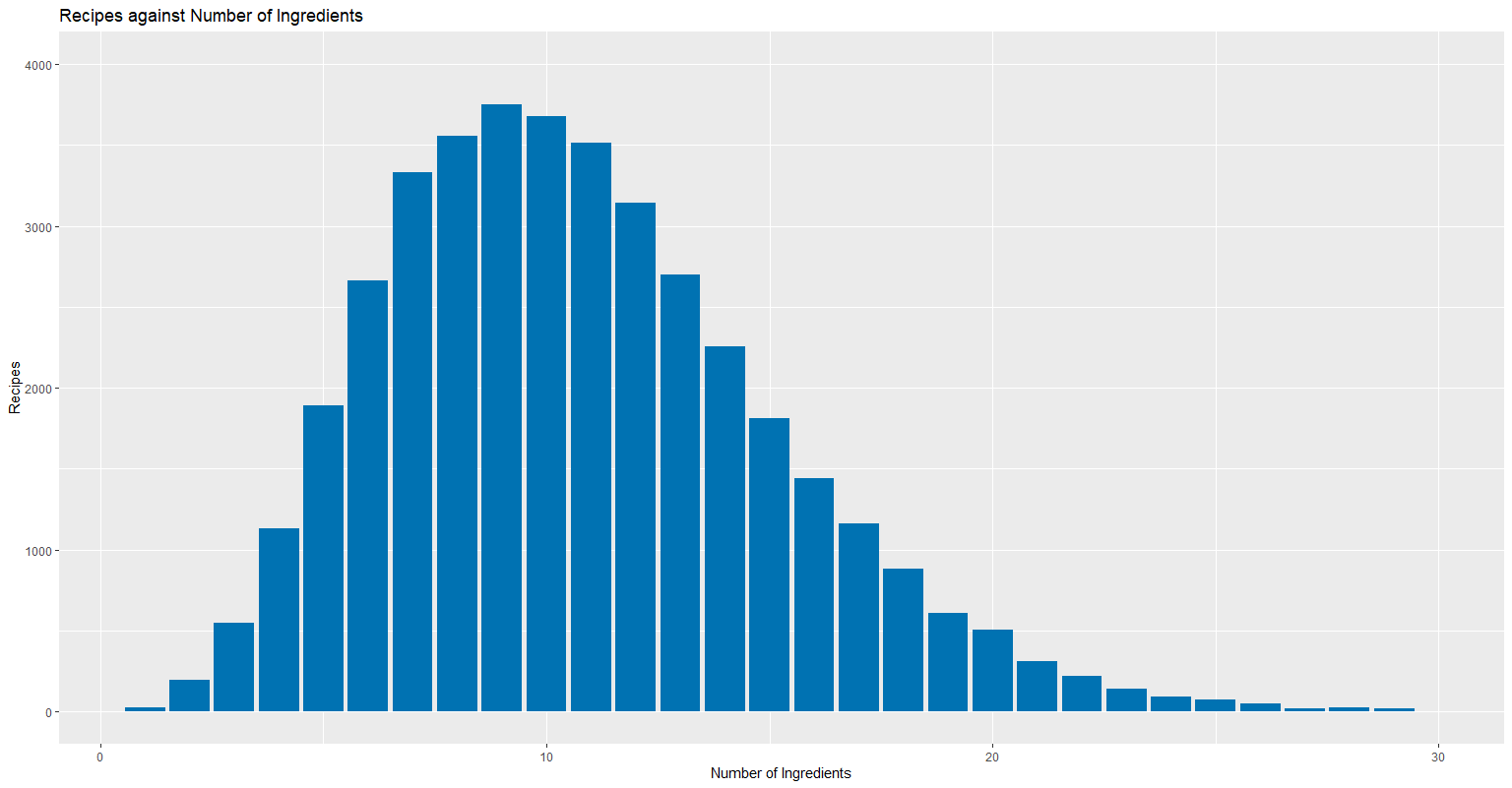
A common issue encountered in machine learning is that of class imbalance. The current dataset train.json from Kaggle.com has an unequal representation of cuisines. As shown by the histogram in *Figure 2*, we can observe that Italian cuisine has the greatest number of recipes, followed by Mexican and Southern US cuisines. In contrast, Brazilian, Russian and Jamaican are the lowest represented cuisines. Since our main aim is to generate a classifier that can predict all classes equally well rather than favoring a few classes that have an exceptionally big proportion of training samples, the dataset has to be adjusted to represent all cuisines in the dataset equally. To tackle this, we chose to undersample the abundant classes according to the mean number of recipes per cuisine (*Figure 3*). After rounding off the mean, abundant classes with samples sizes greater than 2,000 would now be sampled at 2,000 each. The resulting adjusted dataset is shown by the histogram in *Figure* 4.



*Figure 3. Showing the calculated mean number of recipes per cuisine in train.json.*

****

*Figure 4. Histogram showing the number of recipes of each cuisine from the train.json dataset. The top 6 cuisines (Chinese, French, Indian, Italian, Mexican, Southern\_US) were undersampled to balance the different classes.*



*Figure 5. Histogram showing the number of recipes with their corresponding number of ingredients. This histogram was generated with the original train.json dataset.*

*Figure 5* above shows a histogram demonstrating the number of recipes against the number of ingredients in each respective recipe in the original train.json dataset. Recipes with about 9 to 10 ingredients constitute the highest number of recipes in the dataset.

## 2.2 Data Pre-Processing

Next, several preprocessing steps were performed. In R, using the *TM* (text mining) package, a corpus is created from the lists of ingredients within the dataset. A corpus is essentially a way to store a list of words in a format for NLP. Subsequently, there are several important functions within the *tm* package for cleaning and preprocessing data. Firstly, *tolower()* was used to convert all the text to lowercase. Secondly, *removePunctuation()* was used to remove all punctuation contained within the text. Thirdly, *removeNumbers()* removed numbers from the text. Fourthly, we also applied the function *stripWhiteSpace()* to remove extra spaces and empty tabs between consecutive words in the text. Finally, the function *stemDocument()* is used to segment words into their common base. For example, all three words “grating”, “grate”, “grated” would be considered as “grat” after the application of the function.

After the preprocessing step was successfully completed, a document term matrix (DTM) was created. A DTM is a matrix describing the frequency of terms that occur in a collection of documents. In this case, each recipe (documents) would act as the row of the matrix while the column of the matrix would indicate the ingredients in each recipe (terms).

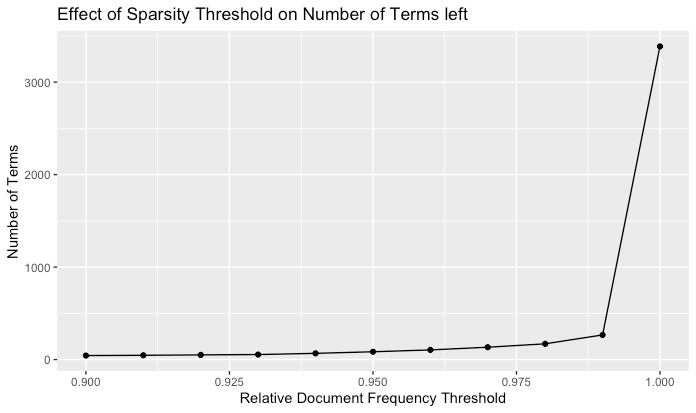
# 3. Results

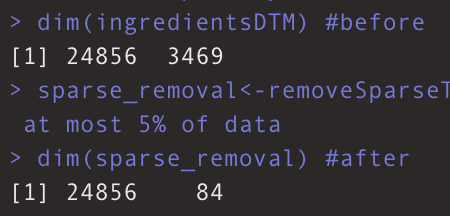
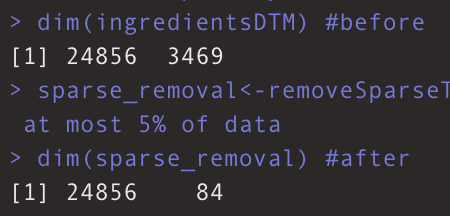
## 3.1 Ingredient Word Cloud



*Figure 6. Word cloud generated from the train.json data set after performing the stemDocument() function. Higher frequency words are depicted in the center with bigger word sizes while lower frequency words are positioned at the periphery with smaller word sizes.*

“Pepper”, “salt” and “oil” can be seen as the most frequently occurring words in all the recipes of the training data set. Some ingredients appear as truncated forms of their names, such as “chees” instead of “cheese”. This is due to *stemDocument* reducing the ingredients name to its root, or minimal letters which can be used to identify that ingredient.





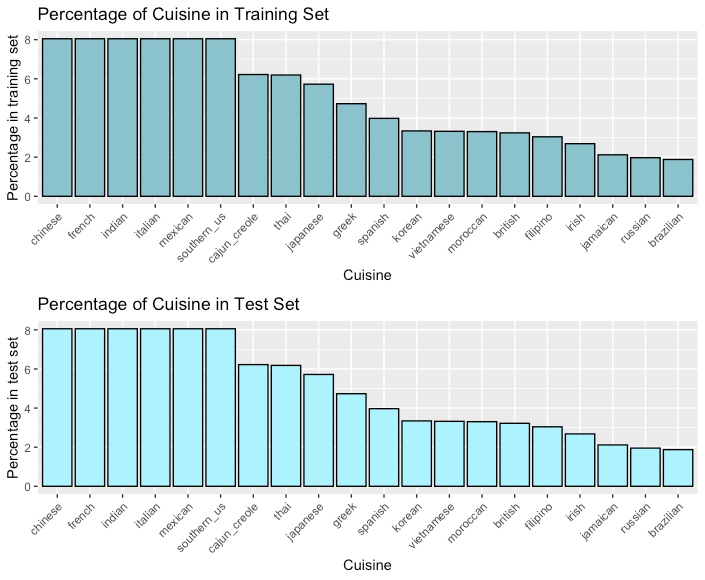


*Figure 7. Showing the dimensions of the dataset before and after removal of sparse terms*

The dimension of ingredientsDTM (Document Term Matrix) without sparsity removal consists of 24856 recipes and 3469 ingredients. We decided to remove sparse words for the machine learning model to minimize variation and hence enable the model to generalize better and prevent overfitting. Sparse words which were removed are defined as terms that occur in less than 5% of the total recipes. Following removal of sparse words, only 84 ingredients were left. This is characteristic of feature selection, which only selects features (ingredients) with a strong relationship to the output variable (type of cuisine).

## 3.2 Data Partition

Before applying any machine learning model to the dataset, we split the dataset into separate training and testing sets. The testing set is important for evaluating the machine learning model to ensure it is not overfitted after training the model with the training set. We create a data partition, allocating 80% of the data to training set and 20% to testing set. To ensure that the data is split according to different cuisines evenly, we compared the following figures next to each other.



*Figure 8: Bar chart of the percentage of each cuisine in both training set (top) and test set (bottom).*

From *Figure 8,* it can be easily concluded that the dataset has been split evenly using createDataPartition. Similar percentages of each cuisine is observed in both training and test set. Hence, there is no bias in cuisine distribution between training and test set.

## 3.3 Data Cleaning of Ingredient Names

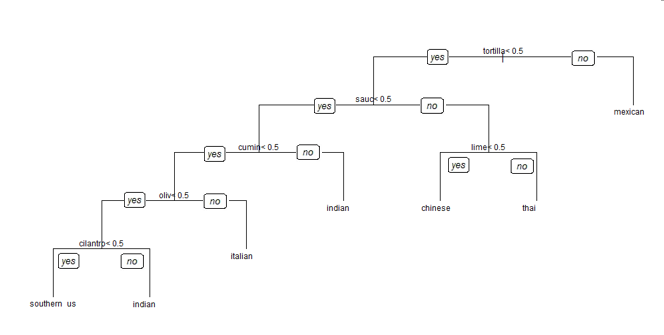
Subsequently, the testing and training sets used for building the models were cleaned, with adjectives removed from the ingredient list. In addition, root words that have been stemmed with the stemDocument function were replaced with their relevant ingredients. This step is crucial for the user to only select only ingredients in the dropdown list in the R shiny dashboard, as well as identify each ingredient easily. As a result of data cleaning, the number of ingredients used for the prediction decreased from 84 to 57.

## 3.4 Decision Tree Model

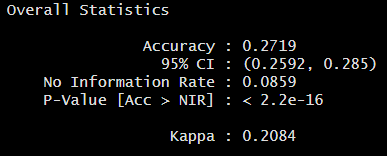
A classification and regression tree model (CART) was created as part of model building for machine learning. This decision tree model is a top-down approach starting from the root which will then partitioned into branches. Each root serves as an input variable and according to specifications, two different outputs will be generated. Each branching is performed with the aim of lowering the Gini index.

The Gini index gives us the probability of incorrectly classifying a randomly chosen variable. Therefore, an ideal Gini index would need to be as low as possible. The degree of Gini index ranges between 0 and 1. A Gini index of 0 indicates that all samples belong to the same class or there only one class exists. On the other hand, a Gini index of 1 would indicate that all the samples were randomly distributed across all different classes. At each node, the splitting would be performed by taking the features (in this case the ingredient) with the lowest Gini index. As a result, the Gini index would be lowered to the computed level after the split.

In our report, the Gini index before performing the decision tree would be very close to one since the samples are randomly distributed across all 20 different classes. After the decision tree is performed, the Gini index should ideally be as close to 0 as possible, provided that the decision tree is a good model. A properly trained decision tree model would hence be important and valuable for our project.



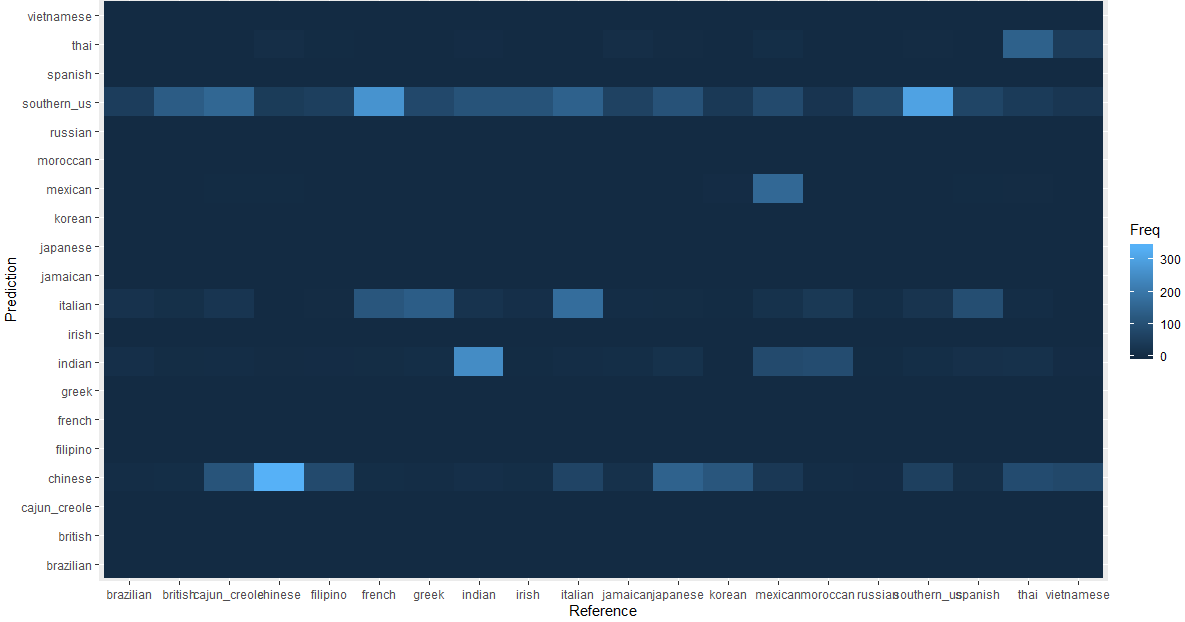
*Figure 9. CART decision tree schematic for classifying the type of cuisines according to the percentage of certain ingredients within recipes of a particular cuisine.*



*Figure 10. Overall statistics of the decision tree model.*

As seen in *Figure 10* above, the decision tree has only a relatively modest accuracy score of 0.2719. While it is not entirely useless, its performance could hardly be considered very useful.

Next, a multiclass confusion matrix was generated. The columns in confusion matrix refer to the actual type of cuisine a recipe belongs to (hence serving as the reference), while the rows refer to the prediction output by the machine learning model. From the plot in *Figure 11,* we can observe that several cuisines have errors in their predictions from the CART model.

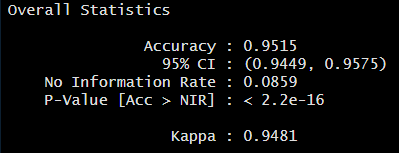


*Figure 11. A multiclass confusion matrix comparing predictions provided by the model in the y-axis against actual type of cuisine shown in the x-axis.*

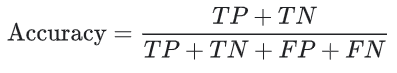
From the confusion matrix in *Figure 9* above, we can observe that the performance of the decision tree is not satisfactory when predicting the correct type of cuisine. The error of prediction is most apparent in the case of southern US cuisine, with the algorithm erroneously predicting southern US cuisine for almost all the different cuisines. This likely shows that southern US cuisine utilizes ingredients that are shared by other cuisines. For cuisines that only have relatively few recipes for training (Brazilian, Russian etc), it is not possible to make a clear distinction between successful and failed predictions of the correct type of cuisine. The frequency of success and failure is likely to be similar between cuisines with relatively few recipes for training. It is also interesting to note that the model is capable of predicting Mexican very successfully. This is probably explained by the usage of unique ingredients in Mexican cuisine that are rarely used in other cuisines.

## 3.5 Random Forest Model

The random forest classifier is essentially an improvement of the decision tree model that we used earlier. In random forest, a large number of relatively uncorrelated decision tree models were used. Each decision tree in the random forest would only choose a subset of the training data randomly for training, contrasting with the CART model which would deal with the training data in its entirety. In addition, in a random forest algorithm, not all 20 variables (types of cuisines) will be used for training a decision tree. When less variables or parameters are used for training, this would result in lower variance and would theoretically boost prediction accuracy. Once the random forest is constructed, the result of the classification of a random forest is the majority vote from among all trees in a random forest. [5] In other words, the predictions done by the random forest model is an average of the predictions of all decision trees generated in the random forest. In this report, we used the function *randomForest()* for the random forest classifier and a total of 100 trees were generated.



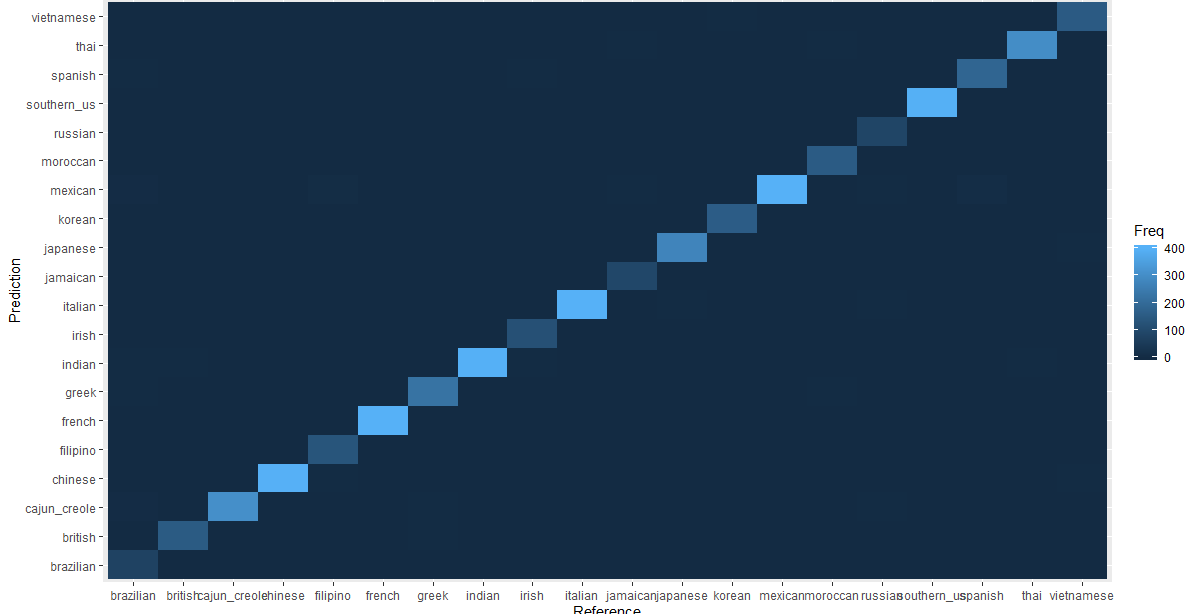
*Figure 12. Statistics of the random forest model.*



*Figure 13. Formula for calculating accuracy.*

Based from the accuracy score shown in *Figure 12* above, we can conclude that random forest model has a significantly higher accuracy score (0.9515) than when using a single decision tree earlier.

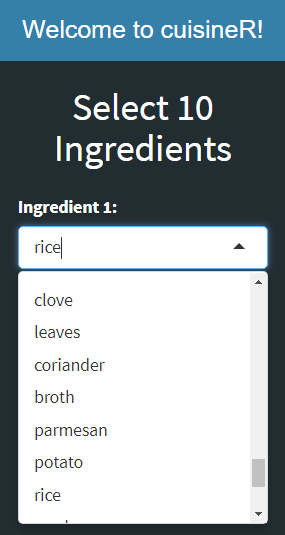
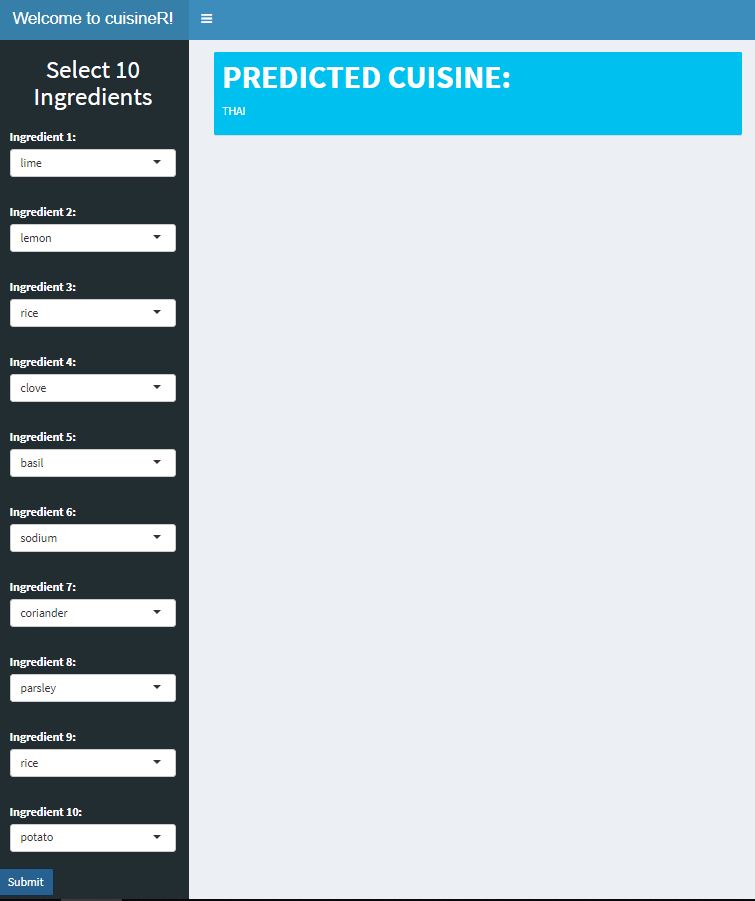
A multiclass confusion matrix was plotted to visualize the performance of the Random Forest Model.

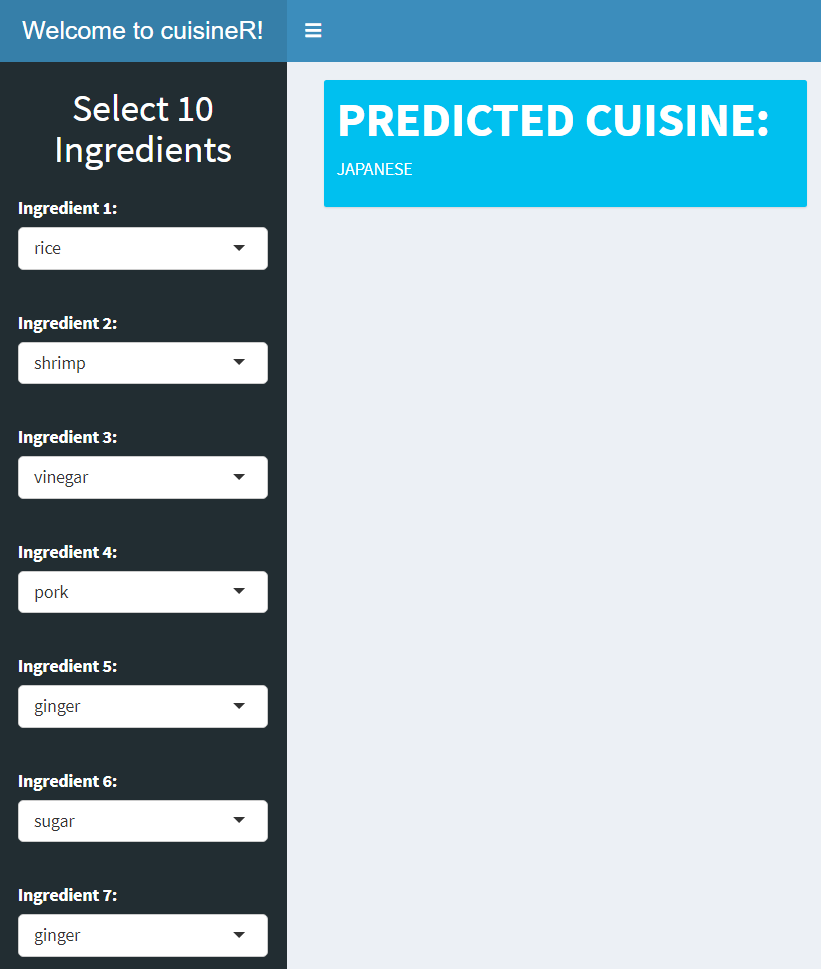


*Figure 14. The multiclass confusion matrix comparing the prediction of the cuisine generated by the random forest algorithm in the y-axis against the actual cuisine as the reference in the x-axis.*

From the confusion matrix above, we can conclude that the prediction capability of the random forest algorithm is indeed impressive as it displays a clear trend of prediction where the predictions are at a highest frequency when matched against the correct reference.

## 3.6 R Shiny Dashboard: cuisineR



**

*Figure 15. cuisineR: A R Shiny dashboard with the implementation of our Random Forest model. Top left: Dropdown selection input list of ingredients. Top right: Overall interface inclusive of ingredient selection and display of predicted cuisine. Bottom: A closer look of the interface after input of ingredients.*

As a direct application of our machine learning model into a real-life scenario, we developed the R shiny dashboard *cuisineR*. Apart from the reasons listed earlier in the introduction, one of the main purposes of cuisineR is to bolster the making of decisions for online grocery shoppers. It could be implemented as an additional function to an online grocery shopping webpage, which will predict the likely cuisine of an online shopper, thereby inspiring the customer to cook up a suitable cuisine.

We first conceptualized a basic dashboard where 10 different ingredients can be input through a drop-down list. With reference to *Figure 5*, the mode number of ingredients for all the recipes in the dataset was about 10. It was this reason that we set 10 as the number of different ingredients to be input. The 10 ingredients will then be read through the ui into the server and added into a dataframe. This dataframe will then be fitted against our Random Forest model to derive a prediction of the most likely cuisine.

A function was first built to input reactive values from the 10 ingredients selected by the user into a binary dataframe that contains all ‘0’ values by default. When an ingredient is input into the function, the ‘0’ values are replaced by ‘1’ values. The resultant data frame will be using to predict the cuisine with our Random Forest model. Subsequently, the function was adapted to register reactive variables input from the ui’s dropdown selection in the server. The resultant output of the function was then rendered into text output in the ui of the R Shiny web application. The output of the R shiny app is the predicted cuisine.

We named the web application *cuisineR* after the French word “cuisiner”, meaning “to cook”.

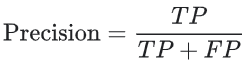
# 4. Discussion

## 4.1 Predictive Model Performance

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Mean Sensitivity** | **Mean Specificity** | **Mean**  **Precision** | **Mean F1 Score** | **Balanced Accuracy** |
| CART | 0.1550927 | 0.9650929 | NA | NA | 0.5600928 |
| Random Forest | 0.9502109 | 0.9986956 | 0.979237 | 0.9638679 | 0.9744533 |

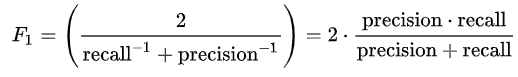
*Figure 16. Table showing the mean metrics of all the cuisines for the CART and Random Forest models.*

As observed in *Figure 16* above, the CART model was a weak classifier for the classification of cuisine. This is evidenced by the low mean sensitivity of 0.155 despite the high mean specificity of 0.965, suggesting a low true positive rate in the model built. As such, subsequent predictions of the CART model are not reliable and irreproducible. In our Random Forest model, the mean sensitivity (0.950) and mean specificity (0.999) indicate that the algorithm produced predictions that are true to the actual cuisines in our dataset.



*Figure 17. Formula for calculating precision.*

For the CART model, the calculated mean precision resulted in an NA. This could be because there were no true positives detected for some cuisines. For the Random Forest model, the precision is 0.979, which is a measure of how many cases are true positive out of the total detected positive space. Precision and Recall (mean sensitivity) must both be considered in order to decrease or increase the classification threshold, especially if other models are made; such as logistic regression. As both precision cannot be calculated and recall is low, we can say that the CART model has a poor performance in classification of cuisines in our problem statement.



*Figure 18. Formula for calculating the F1 score.*

The F1 score of the CART model calculated above resulted in an NA. In a similar fashion as the calculated mean precision above, this could be because there were no true positives detected in some cuisines when calculating the precision, resulting in an NA value for the F1 score. For the Random Forest model, the F1 score is high at 0.964. The F1 is also known as the harmonic mean between precision and recall. A score closer to 1 will constitute a high performance. The F1 score is used to calculate the balance between precision and recall, taking both into consideration when determining the performance of the model. Our Random Forest model had a high F1 score, meaning that it has high accuracy, taking into account both the precision and recall in its classification algorithm.



*Figure 19. Formula for calculating the Balanced Accuracy*

The above formula shows the calculation of balanced accuracy (bACC). The bACC metric is suggested to use to measure accuracy of the total performance of a model, taking into consideration both the positive and negative classes. bACC essentially involves the normalization of any true positive and true negative predictions by the number of positive and negative samples, which are then summed up and divide by two. The bACC metric is suggested for usage on a dataset imbalanced dataset such as ours as evidenced in *Figure 2*, which shows the uneven distribution of the number of recipes for each cuisine.

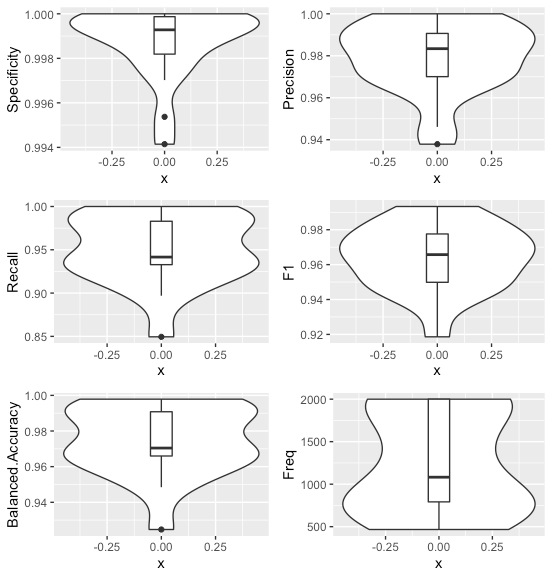


*Figure 20. Formula for calculating the F1 score*

The false discovery rate (FDR) of the model can then be calculated from the precision or positive predictive value as shown in the above formula (*Figure 20*). The false discovery rate is a metric which conceptualizes the rate that any features called significant are truly null. [6]

An FDR of less than 0.05 means that among all features determined as significant, less than 5% of these are truly null. Similar to how alpha is determined as the threshold for the p-value, the threshold is also set for the q-value to control the FDR. The q-value is the analog for the p-value. [7] A q-value of 0.05 means that 5% of significant results will be false positives. This is an important metric to consider if the q-value is more than 0.05. It is important to look at the FDR because if a significance test is permuted 1000 times, it is expected to obtain 50 false positives by random chance alone. The metric was determined in this case to check if subsequent corrections need to be done via the Benjamini-Hochberg procedure to control the FDR using q-values. [7]

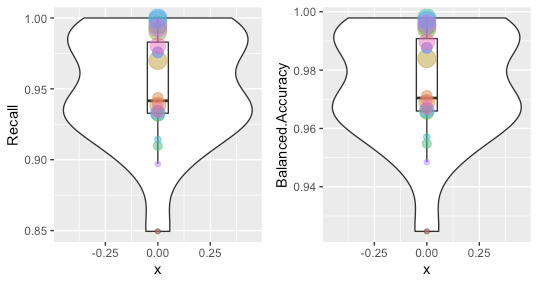
In our Random Forest model, the mean FDR = 1 - 0.979237 = 0.020763, where PPV is also known as precision. This means that the FDR for the model is acceptable and the features determined by the model are robust and considered significant.

****

*Figure 21. Graphs showing distribution of performance metrics parameters (Specificity, Precision, Recall, F1, Balanced Accuracy) of the cuisines predicted using random forest model and Frequency distribution of cuisines.*

*Figure 21* shows that the distribution of various performance indicators is within a rather narrow range across cuisines, mostly skewed towards 1. Hence the predictive model will perform quite well across the various cuisine types. Here, we focus on precision instead of recall, though both have a reasonably narrow value range. Precision is used to evaluate our machine learning model as we aim to minimize false cuisine predictions (false positives). The recall is not as important because the sheer number of recipes per cuisine indicates that it is not as important to try to capture as many cases as possible in a particular cuisine. When examining the precision for each type of cuisine, its value is within a rather narrow range of 1.0 to 0.94, which implies that a very high portion of predictions are true positives across all the cuisines. Hence, cuisines predicted based on the ingredients list are highly reliable.

Also, it is observed that recall and balanced accuracy show 2 separate peaks in the distribution across cuisines. This could be due to class imbalance, which is reflected in the graph showing frequency of cuisines (*Figure 21: Freq)*. Plot points showing frequency were added to demonstrate this, as shown in *Figure 22 below*.



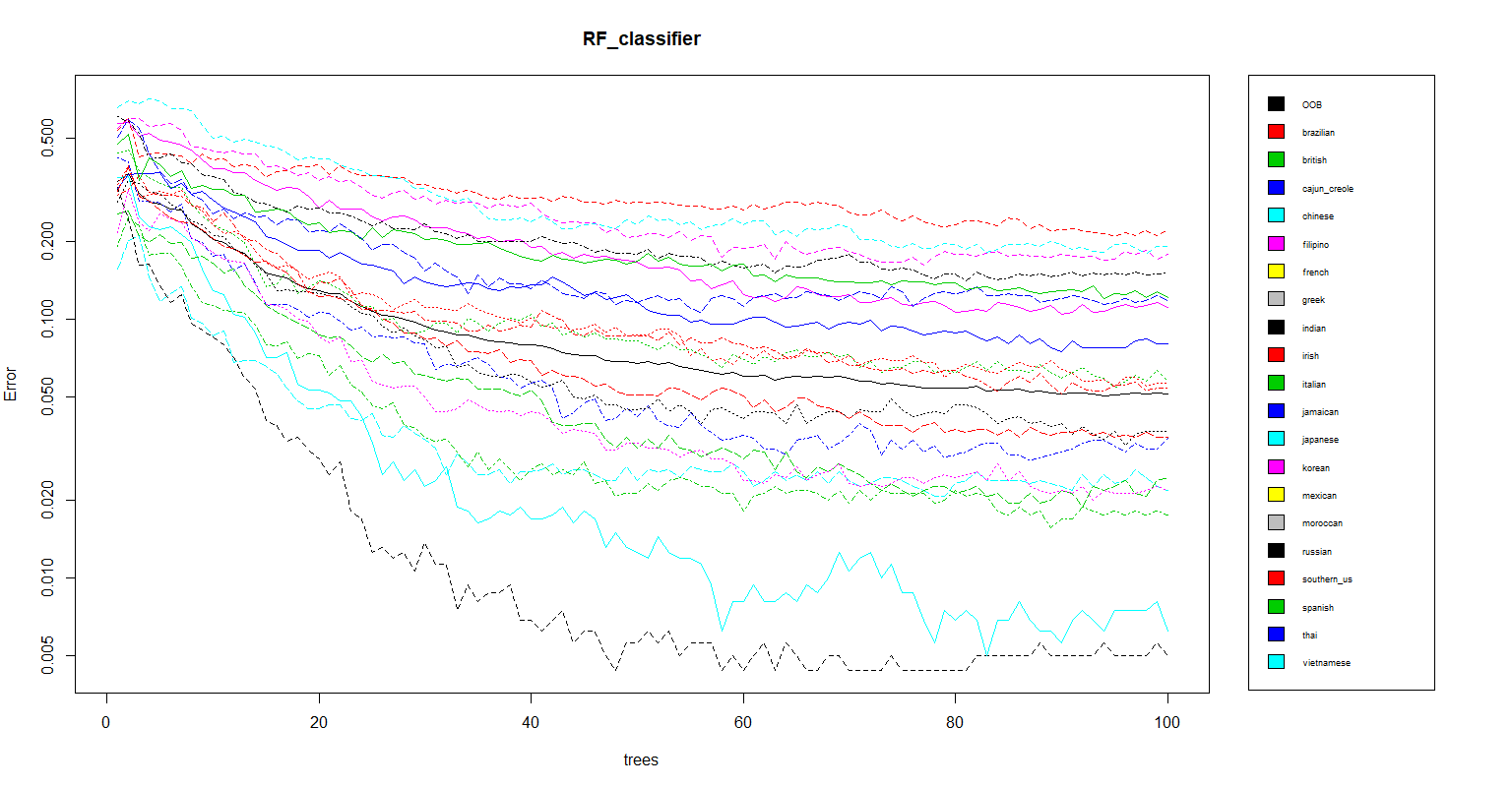
*Figure 22. Graphs showing cuisine frequencies (size of geom\_point) of each cuisine type (colour of geom\_point) within Recall and Balanced Accuracy distribution.*

After adding plot points showing the class size of each cuisine onto both distributions, there is an observed association of the distribution of cuisine frequency with the bimodal distribution of both sensitivity and balanced accuracy. Therefore, it can be said that the bimodal distribution of both indicators is due to bias from the class imbalance within the dataset.

## 4.2 Strengths and limitations of decision tree

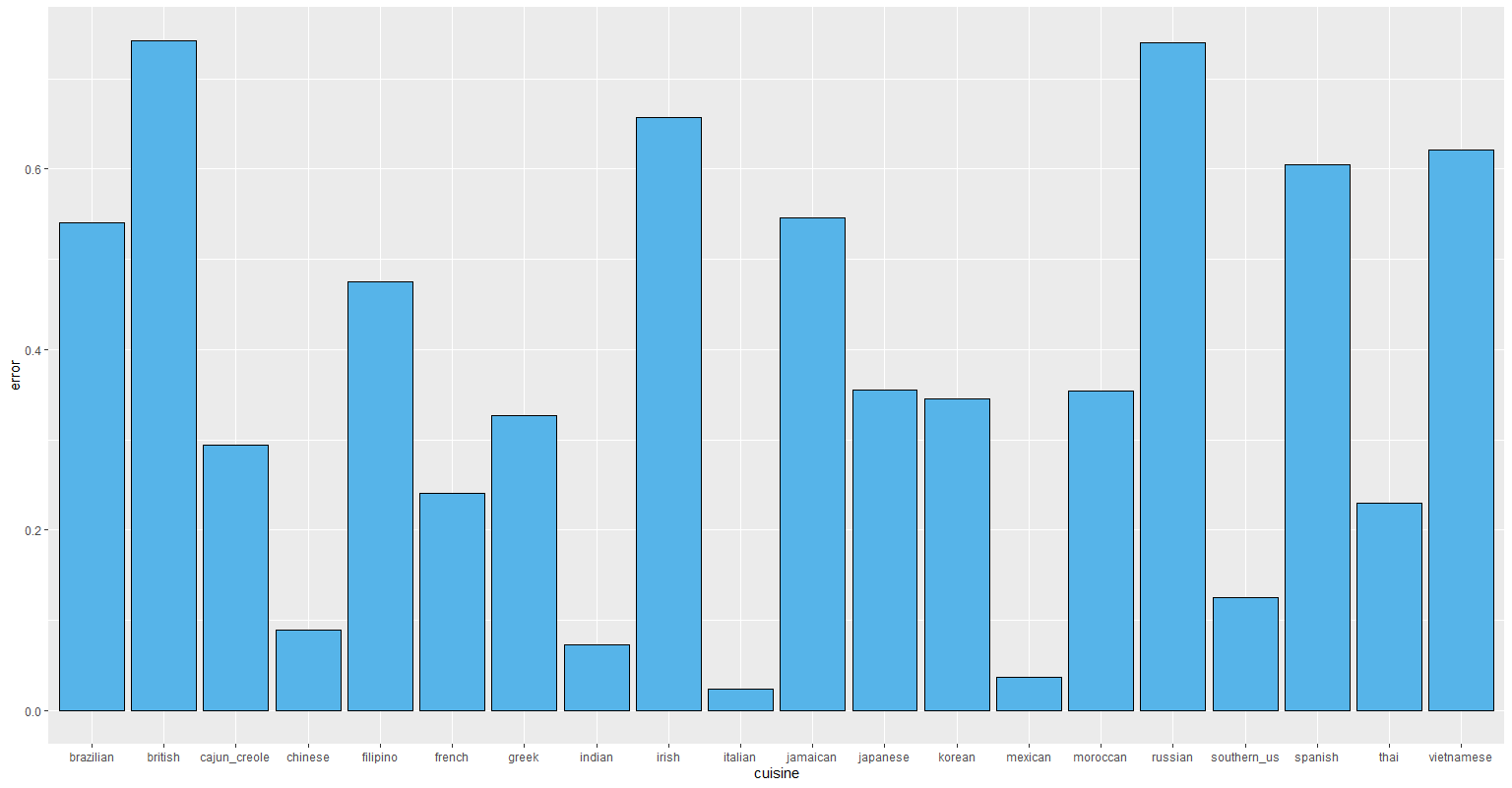
One of the strengths of decision tree as a machine learning model is that it is relatively easy to be interpreted and understood based on the classification decisions. Furthermore, decision tree algorithm is also very suitable for dealing with categorical data that we are using. However, a decision tree could also run the risk of being overly branched, hence committing an overfitting problem, while incurring a potential loss in accuracy. In other words, it is locally optimized as it classifies the training data perfectly, but it may not be a globally optimal model as there is no guarantee that the decision tree could work well on other data set.

One way to counter this problem would be to perform pruning. Pruning is the act of removing branches that contain low-importance features, hence enabling the decision tree to generalize and predict better. Another way is to implement random forest classifier to counter the low accuracy problem of CART model of decision tree. Hence, we decided to proceed with our analysis using a random forest classifier.  We shall discuss more about random forest in the following section.

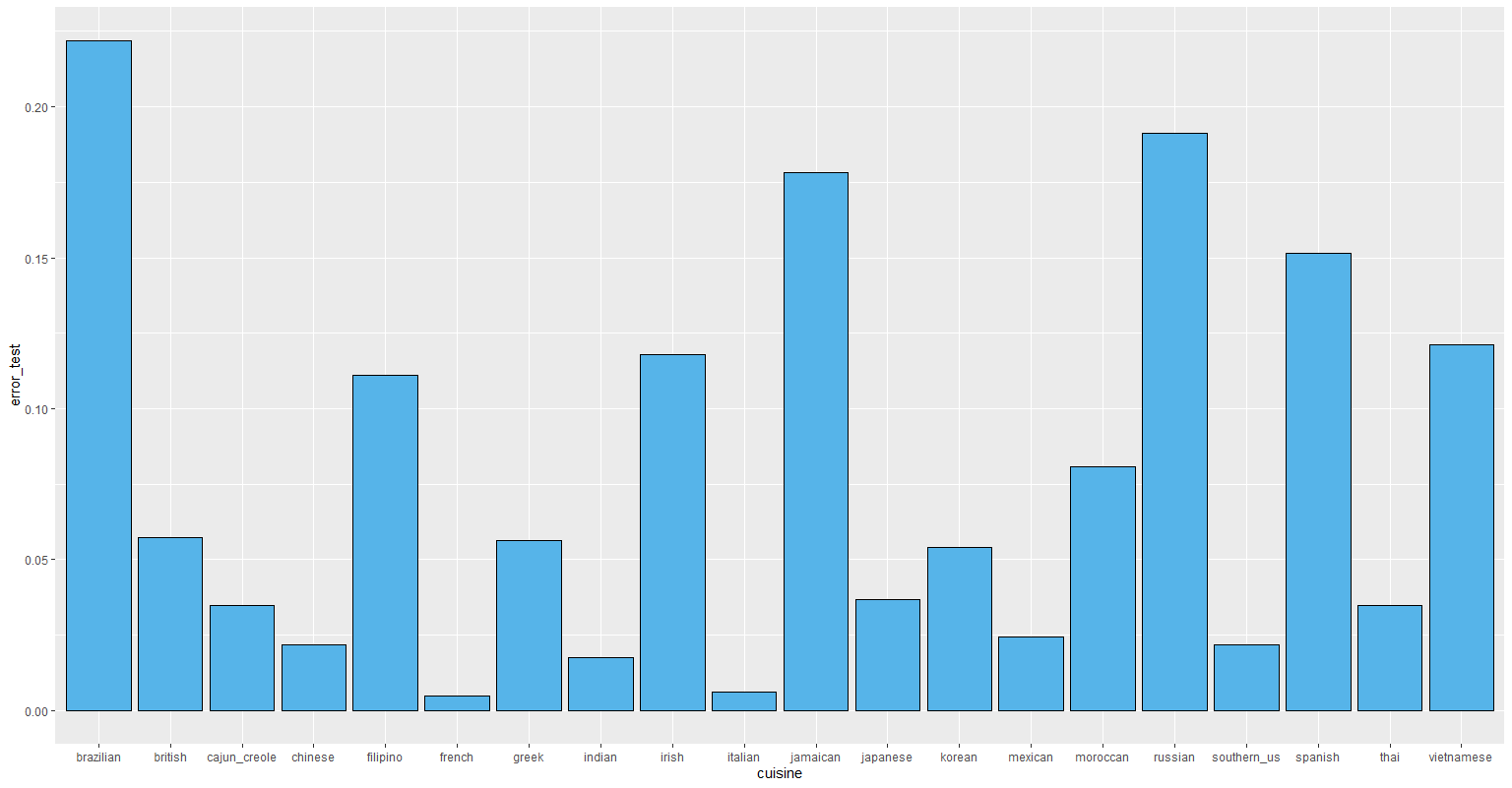
**

*Figure 23. Graph of error against the number of trees grown in our Random Forest model. Each colored line represents a cuisine. The black line represents the entire sample.*

As observed in *Figure 23* above, in our Random Forest model, as the number of trees grown in the Random Forest model increases, the error for each cuisine decreases. This is because as each tree is grown on an independent bootstrap sample to the maximum depth and a prediction is subsequently made by aggregating the predictions or voting for a prediction made by the single trees.



*Figure 24. Out-of-bag estimate of error rate of each cuisine using the original unbalanced dataset shown in figure 2.*

**

*Figure 25. Out-of-bag estimate of error rate of each cuisine using the balanced dataset shown in figure 4. Note that the y-axis scale for figure 24 is different from that of figure 25.*

From *Figure 25*, we can observe that Italian cuisine has the lowest estimate of error among all cuisines in the dataset. This is possibly due to the large representation of Italian cuisine in the dataset as observed in *Figure 2*. Similarly, the Russian cuisine has a high estimate of error possibly due to the smaller representation in the dataset. In contrast, despite having a smaller cuisine size, Brazilian and Jamaican cuisine have lower estimates of error as compared to British cuisine. This could be dependent on how distinct each cuisine is. Also, having a higher number of ingredients that could lead to recipes having stronger features and hence a lower error rate.

After balancing the dataset by undersampling of the top 3 abundant classes (Italian, Mexican, Southern\_US), we observed an overall improvement in error rate across the cuisines as seen in both *Figure 23*, and between *Figures 24* and *25*. Where the highest error rate in the unbalanced dataset was observed to be around 0.7, the highest error rate observed in the balanced dataset 0.225. Vast improvements were seen for cuisines such as British, Brazilian and French.

## 4.3 Limitations

To minimize noise from the dataset, our group removed low frequency words using the removeSparseTerms function.

However, simply removing sparse terms could have potentially removed rare ingredients from the dataset. In addition, this removal of sparse terms could have eroded the true meaning of the ingredients. While “carrots” alone still constitute an ingredient, it does not demonstrate the true nature of the carrots being “diced”, which in practicality, will not be useful for classification of real datasets.

As predictions made by our model is made based on extrapolations of the provided data, using a dataset from a single source could result in bias. Hence our model may not be generalizable.

## 4.4 Uses

We anticipate that our cuisine predictor would be able to help automate classification of recipes on food or restaurant websites according to their most probably cuisine, reducing manpower and human judgement needed to manually classify new recipes. It could also serve as a teaching tool to inspire others to be flexible with their perception of ingredients, showing them how a set of ingredients could be used for several other cuisines.

As mentioned in the introduction and under the R shiny section, a possible way to implement this idea is to incorporate the cuisine classifier onto online grocery shopping websites. Recipes in the predicted cuisine with ingredients similar to the shopper’s cart could also be recommended to the shopper. This would serve to similarly inspire shoppers who would want to explore different styles of cooking. Also, it can be useful for building food recommendation by finding the similarities between cuisines, since similarities arise mainly because of the ingredients.

# 5. Future Directions

As mentioned earlier, due to the strong class imbalance in the original dataset, our group chose to undersample recipes from cuisines which were larger than the mean. However, there are two pitfalls of this method. The first reason the cost of unused data. The second reason involves the manner which cuisines are undersampled. Undersampling cuisines leaves the differences between undersampled cuisines up to chance. The newly undersampled cuisine may not be representative of the original distribution, with this issue being exacerbated at small sample sizes. Our group acknowledges that certain cuisines could have naturally more or fewer recipes than others due to cultural differences, making the search for more recipes difficult. An alternative would be to attempt at other methods such as synthetic minority over-representation technique (SMOTE) or weighting. Furthermore, the model could further be independently evaluated by testing against actual recipes, outside of the Kaggle dataset, with known cuisines.

Earlier on, we chose 10 different ingredients to be the input of the R shiny application as 10 is the mode number of ingredients for all recipes in the dataset. (*Figure 5*) Additional verification should be done to select the ideal number of ingredients to be inputted to get the best prediction accuracy. One way to perform this is to look at the reproducibility of how well a certain cuisine can be predicted. This could be done by taking different numbers of ingredients and by plotting the prediction accuracy for the quantity of ingredients used.

In the R shiny interface, ingredients initially were listed in their truncated form rather than their actual names. While the cuisine predictor can be run on this interface, it is difficult to identify drop down ingredients in its truncated form. In an earlier version of the R shiny application, an ingredient just listed as “black” could possibly represent several different ingredients, for instance “black beans” or “black pepper”. In order to subvert this issue, root words from the *stemDocument()* function were also manually corrected to actual full ingredient words. The final list of 57 ingredients was used to update the R Shiny dashboard. (*Figure 13*). However, manually replacing each ingredient in this manner is time consuming and impractical when dealing with large-scale data. A possible improvement for our project would be to explore specifying certain words as bigrams. In this manner, words that are associated together, such as “olive” and “oil”, would be combined and treated as a single unit “olive oil”. A possible difficulty would be that two single words would also treated as a bigram.

Lastly, the R shiny web application can be further developed to have more features that show more information regarding the predicted cuisine. A recommendation would be to include a basic description, examples of some recipes, and a recipe recommendation tool based on the predicted cuisine. Furthermore, the interface which ingredients are selected is cumbersome as the user would have to scroll through the entire list of ingredients. As a cosmetic improvement, a function could be incorporated such that ingredients selected through a checkbox group, where multiple ingredients can be selected from a list of checkboxes.

# 6. References

[1] Ahn Y-Y, Ahnert SE, Bagrow JP, Barabási A-L. Flavor network and the principles of food pairing. Scientific Reports. 2011;1(1). doi:10.1038/srep00196

[2] Fan W, Wallace L, Rich S, Zhang Z. Tapping the power of text mining. Communications of the ACM. 2006;49(9):76–82. doi:10.1145/1151030.1151032

[3] Talib R, Kashif M, Ayesha S, Fatima F. Text Mining: Techniques, Applications and Issues. International Journal of Advanced Computer Science and Applications. 2016;7(11). doi:10.14569/ijacsa.2016.071153

[4] <https://www.kaggle.com/kaggle/recipe-ingredients-dataset>

[5] Natingga D. Data science algorithms in a week: top 7 algorithms for scientific computing, data analysis, and machine learning. Birmingham: Packt; 2018.

[6] <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4448847/>

Colquhoun D. An investigation of the false discovery rate and the misinterpretation of p -values. Royal Society Open Science. 2014;1(3):140216. doi:10.1098/rsos.140216

[7] False Discovery Rate. Search the website. [accessed 2019 Nov 13]. <https://www.mailman.columbia.edu/research/population-health-methods/false-discovery-rate>