**Capstone Project Milestone Report:**

*Categorizing Cuisines based on Ingredients*

Contents

[1 Introduction 3](#_Toc458628241)

[2 Data Set 3](#_Toc458628242)

[3 Data Wrangling 4](#_Toc458628243)

[4 Exploratory Data Analysis 4](#_Toc458628244)

[5 REGRESSION Models 6](#_Toc458628245)

[5.1 Results Discussion: 7](#_Toc458628246)

[6 Future Work 7](#_Toc458628247)

[7 Acknowledgements 8](#_Toc458628248)

[8 Addendum R- code 8](#_Toc458628249)

# Introduction

When we go for lunch or dinner we would sense aromas of different dishes. It always intrigued me how change in ingredients would affect the aroma of a dish. Especially coming from India, I have observed a small variation in ingredients can change the aromas drastically.

I have chosen this interesting project from Kaggle which is also on similar lines to predict cuisines based on given set of the ingredients.

# Data Set

The data set used in this study is from “What's Cooking?” Kaggle Data competition <https://www.kaggle.com/c/whats-cooking/data>

The data set shows a list of id, cuisine and ingredients. The data set is available in json format. The dependent variable is cuisine. The independent variable is ingredients. Train data set is used for creating model. Test data is used to checking the accuracy of the model.

{  
 "**id**": 24717,  
 "**cuisine**": "indian",  
 "**ingredients**": [  
 "tumeric",  
 "vegetable stock",  
 "tomatoes",  
 "garam masala",  
 "naan",  
 "red lentils",  
 "red chili peppers",  
 "onions",  
 "spinach",  
 "sweet potatoes"  
 ]  
 },

# Data Wrangling

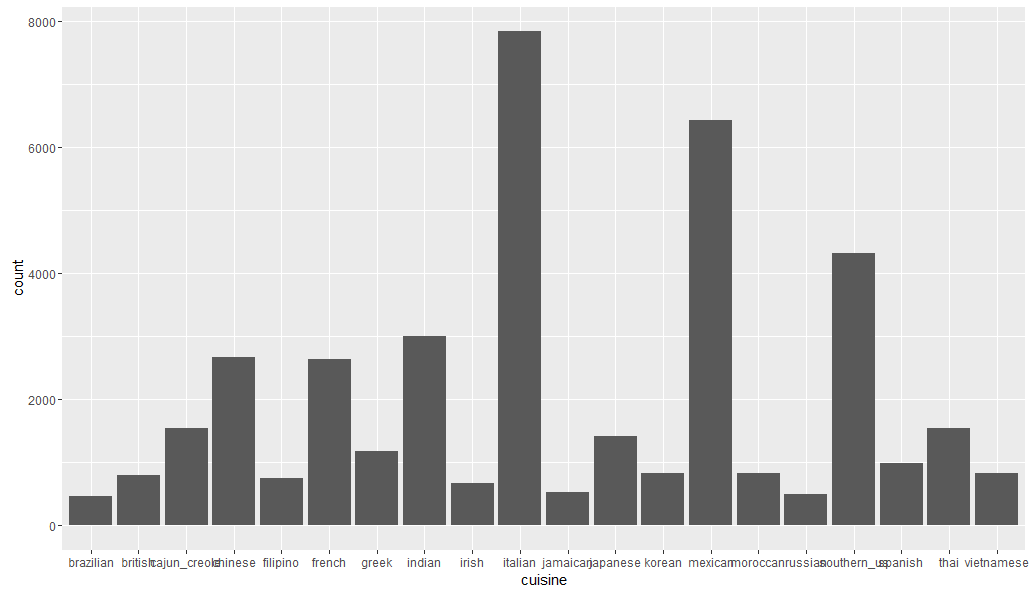
Initial analysis of the ingredients show that the data doesn’t look clean. Some of the issues found are

* Same word but some in lower cases and some in upper cases.
* Numerical values
* Punctuations, Brackets
* Same word but in -es, -s forms.
* White spaces.

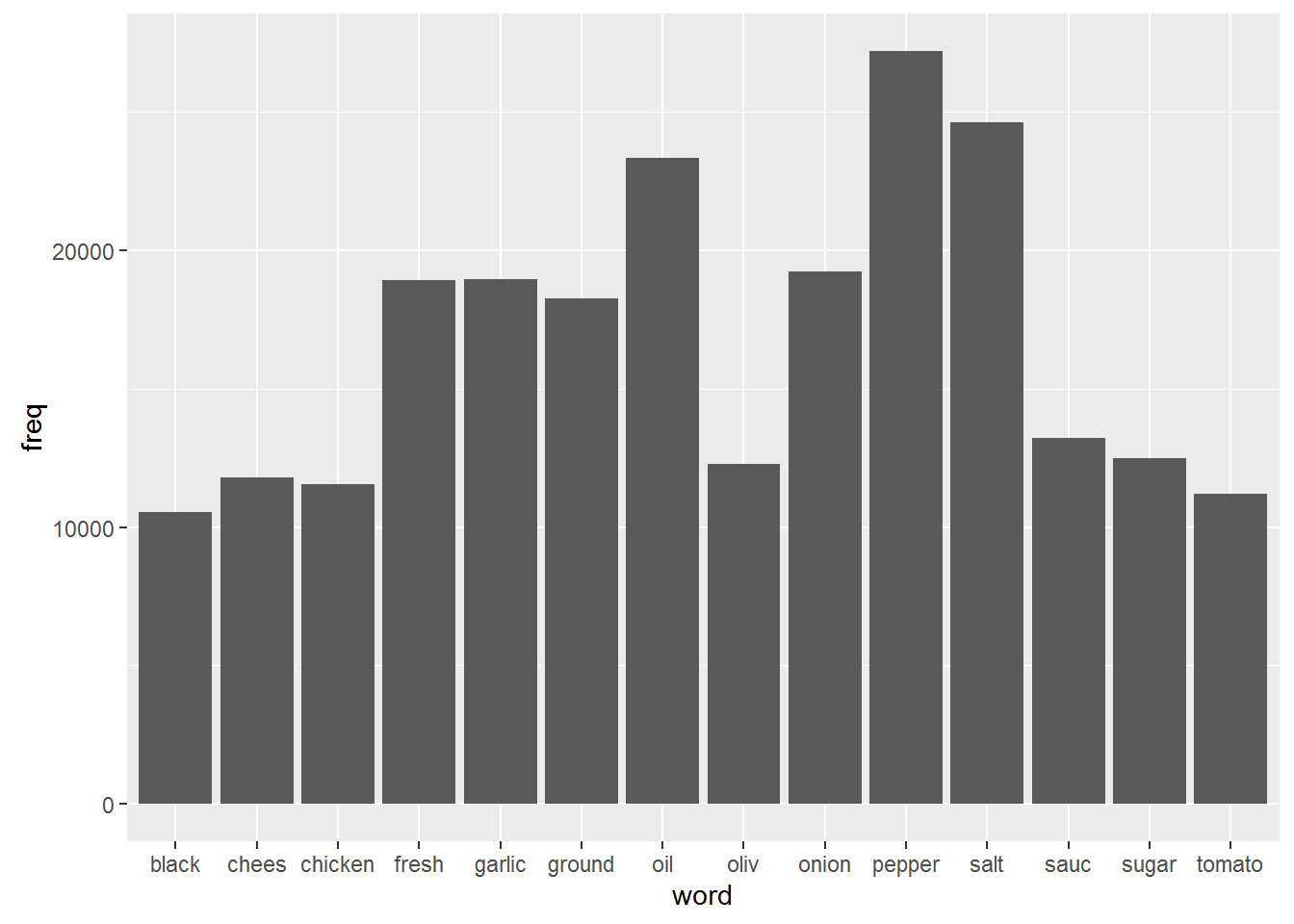
Initial idea was to convert each ingredient into a row instead of the list by using unnest function and use clean the issues found above. Then use convert rows into variables using the dummyvar function. But it turned out to be not practical given 39000 different cuisines and 2702 unique ingredients. So ended up using Corpus function in tm package to clean the data and create Document matrix having each variable as column and values as 1 or 0. 1 represents that particular ingredient is present in that cuisine and 0 represents that it’s not present.

# Exploratory Data Analysis

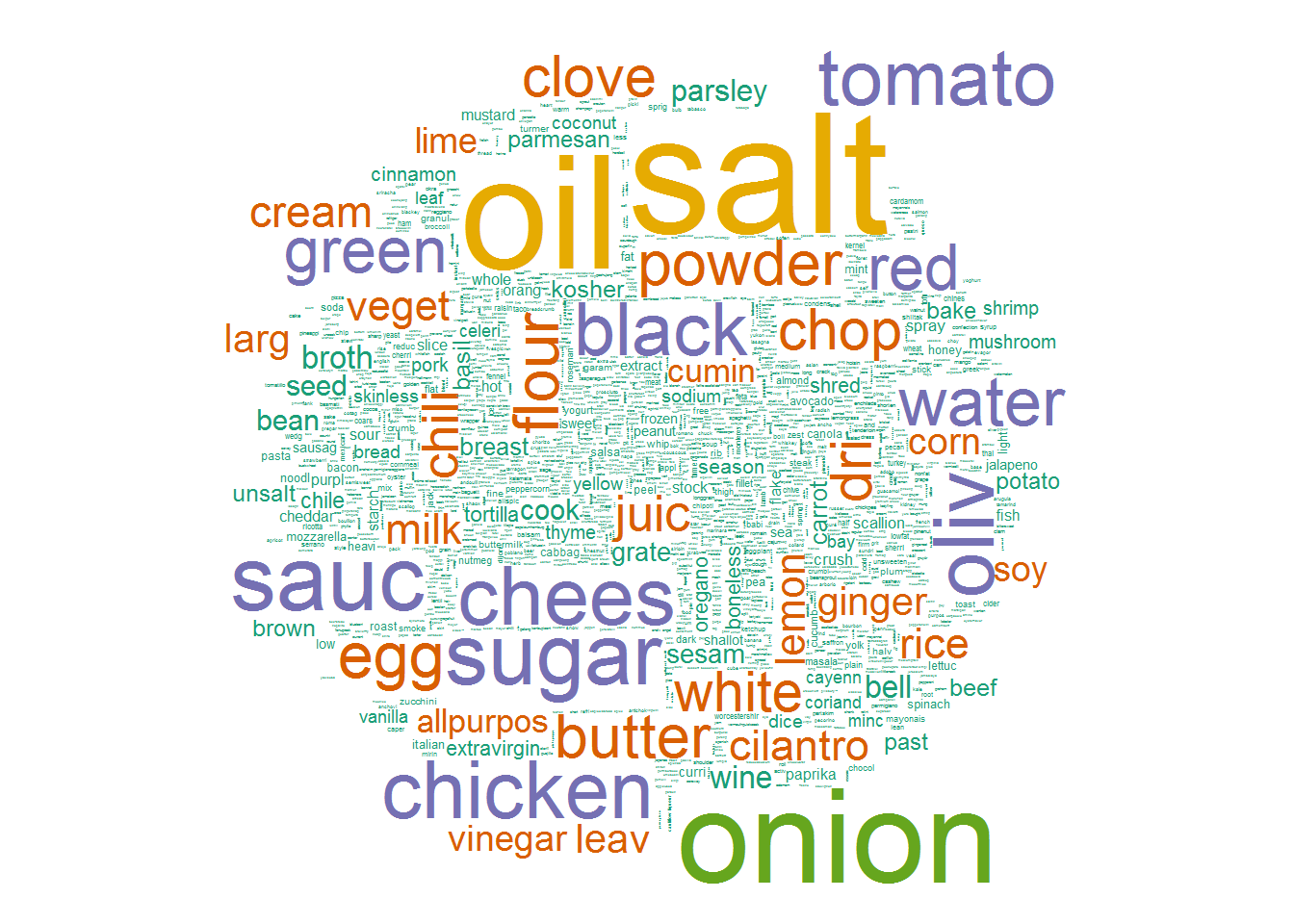
Plotting the frequency of cuisines, we can observe that the Italian is highest in number followed by Mexican and southern us



Plotting the ingredients whose frequency is greater than 10,000, we can observe that pepper, salt, oil are the most used ingredients



Word cloud: Plotting the most used words.

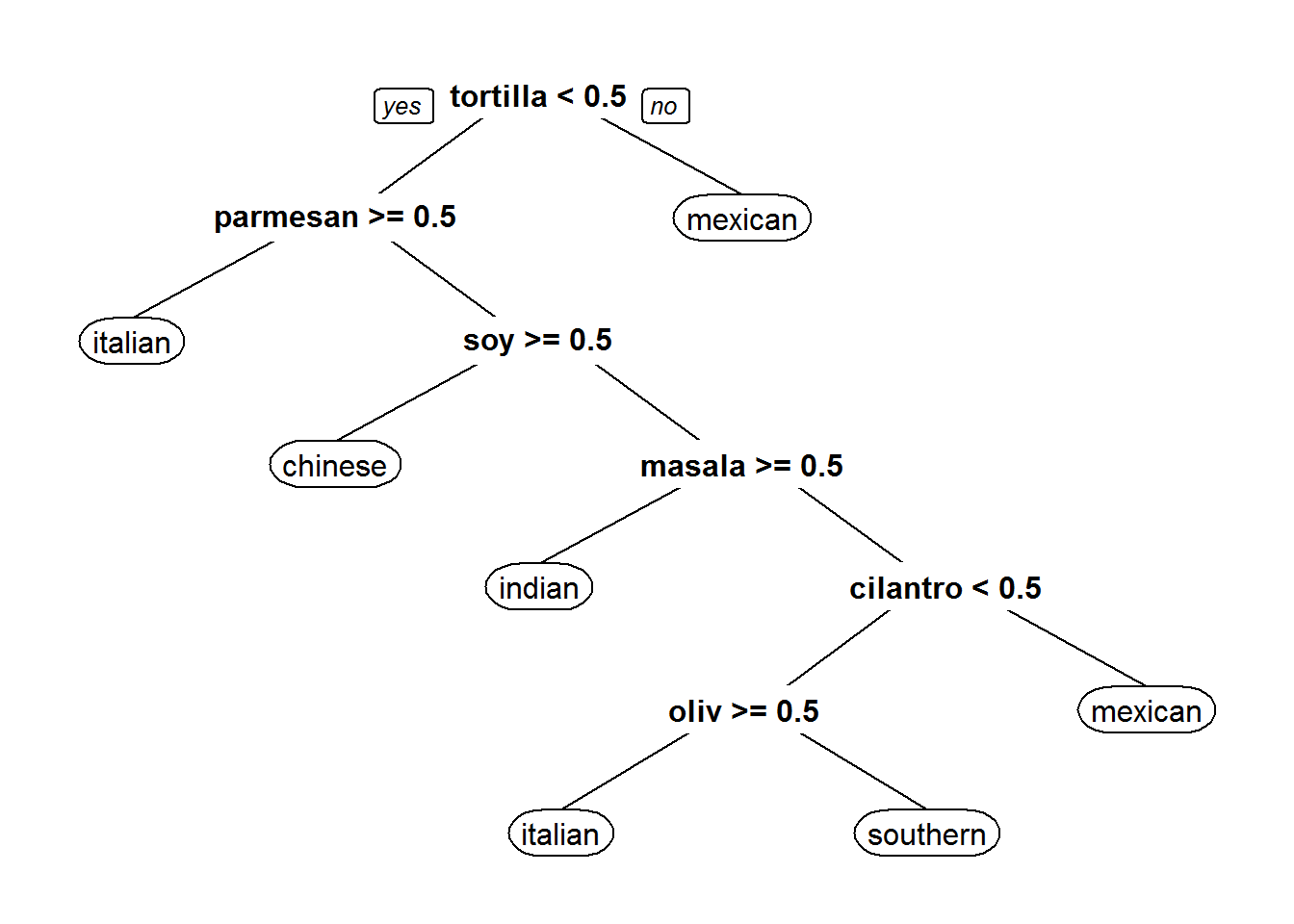


# REGRESSION Models

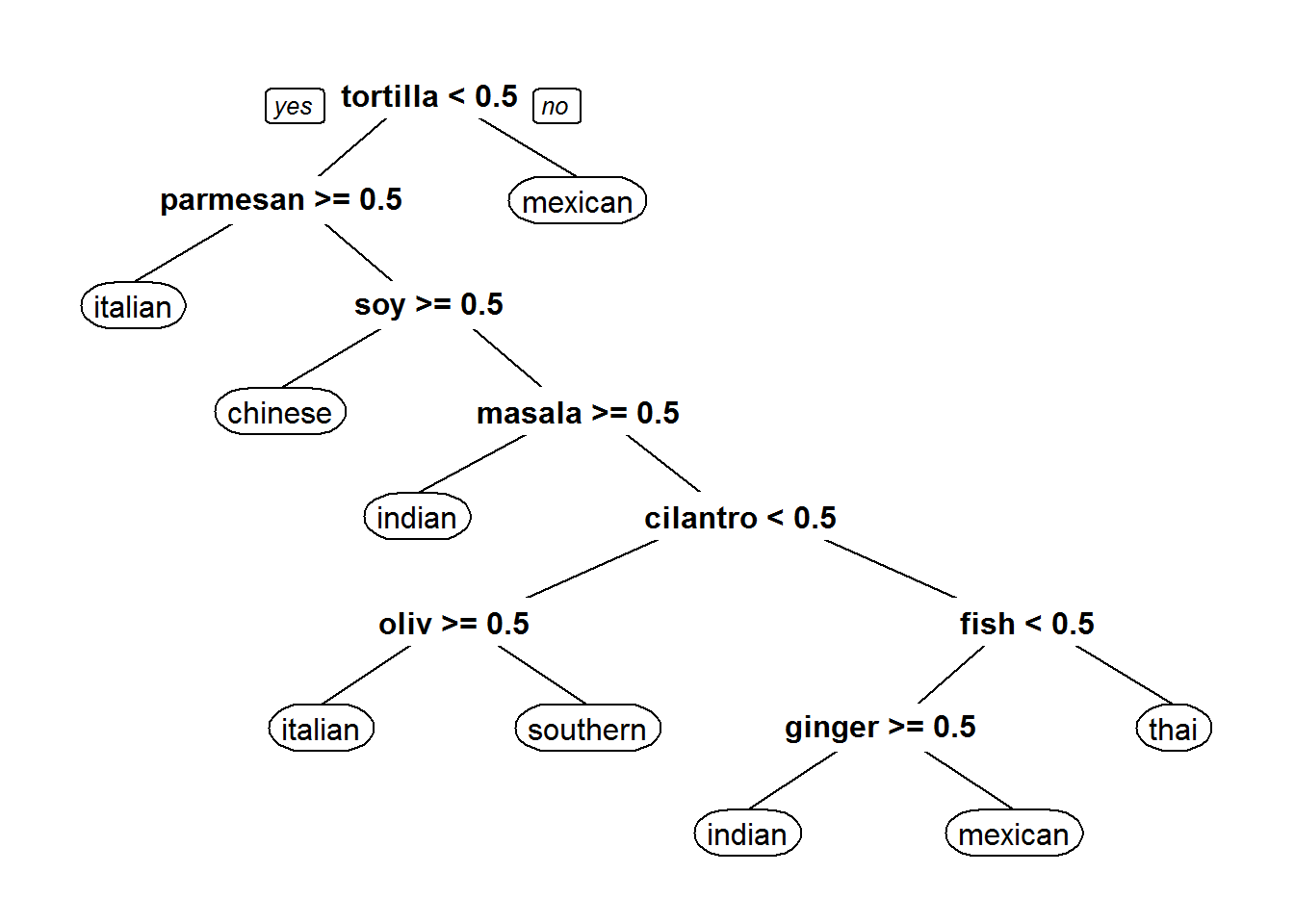
Classification and regression technique is used to predict cuisines. Predicted cuisines with three models, one model using the most used ingredients, one model with all ingredients and last one using PCA. All three models results are shown below.

|  |  |  |
| --- | --- | --- |
| Model# | Variation | Accuracy |
| Model1 | Using Most used ingredient | 42.09% |
| Model2 | Using all ingredients | 42.03% |
| Model3 | Using PCA | 40.09% |

**Model1 Regression tree:**



**Model 2 Regression tree:**



## Results Discussion:

* In Model 1 Using the most used ingredients resulted in lesser accuracy, this might be because the ingredients less used are also a good factor in determining the cuisine.
* Using all the ingredients there is almost no improvement.
* Interestingly, model using PCA is least accurate. Have to research more on it.

# Future Work

Future work will be

* Model using XgBoost
* Visualizing popular cuisines in different countries based on the cuisine categorization and twitter data!
* For a new restaurant, can we predict cuisines which may become the best cuisines given an area/country?

# Acknowledgements

I would like to thank Amit Dingare for his advice and guidance throughout the project. Even though I’m a novice in programming, I was able to pick up and learn lot of new techniques and packages (tm, caret, dummyvar) in a very small span of time because of his guidance.

# Addendum R- code

##loading libraries

```{r}

library(jsonlite)

library(dplyr)

library(tidyr)

library(stringr)

library(ggplot2)

library(reshape2)

library(tm)s

```

## loading Data

```{r}

#setting the working directory

setwd("C:/Users/Bimal/Desktop/Data Science/Slide Rule/Yummly Data")

# loading the train dataset using the Json library

train\_data <- fromJSON(txt = "train.json",flatten = TRUE)

test\_data <- fromJSON(txt = "test.json",flatten = TRUE)

#Converting the data to tbl\_df format in dplyr.

train <- tbl\_df(train\_data)

train

test <- tbl\_df(train\_data)

test

nrow(train)

```

## Freq of cuisines

```{r, echo=FALSE}

# Trying the understand the cuisines present and their frequency

ggplot(train,aes(cuisine)) + geom\_bar()

#We can see from the graph that the train dataset has lots of Italian cuisines followed by Mexican and southern\_us

```

```{r}

# understanding cuisines and their frequency in data form

train\_cuisine\_sum <- train %>%

group\_by(cuisine) %>%

summarise(number = n()) %>%

arrange(number)

train\_cuisine\_sum

```

# Data cleaning

```{r}

# trying to identify any issues with the ingredients

x <- rbind(train$ingredients)

unique\_ingredients <- data\_frame(unique(sort(unlist(train$ingredients))))

unique\_ingredients #ingredients data doesn’t look clean, words are identical but not same. For example, same data is represented lower cases in some cases and upper cases in other (braeburn apple/Braeburn Apple).

```

# Cleaning data Using Corpus

```{r}

library(SnowballC)

ingredients <- Corpus(VectorSource(train$ingredients)) # Using Corpus in tm package and to create Document Matrix.

# Converting all the text to lower case

ingredients <- tm\_map(ingredients,content\_transformer(tolower))

# removing the Quantity from the ingrdients.

ingredients <- tm\_map(ingredients, removeNumbers)

removeBrackets <- content\_transformer(function(x){gsub(pattern = "\\(|\\)|,",replacement = " ",x)})

ingredients <- tm\_map(ingredients,removeBrackets)# Converting the list into Corpus vector resulted in brackets in list so removing them by creting the content transformer remove bracket function.

# removing the punctuations

ingredients <- tm\_map(ingredients,removePunctuation)

#Removing common word endings like(-es,-s)

ingredients <- tm\_map(ingredients, stemDocument)

# Stripping all the whitespaces

ingredients <- tm\_map(ingredients, stripWhitespace)

# Converting ingredinets into a Document matrix

ingredientsMatirx <- DocumentTermMatrix(ingredients)

ingredientsMatirx

#Converting Corpus matrix into df

ingredientsDTM <- as.data.frame(as.matrix(ingredientsMatirx))

```

# Exploring data

```{r}

#Organizing ingredients by thier frequency

freq <- colSums(as.matrix(ingredientsMatirx))

length(freq)

# Ordering them in order

freq\_order <- order(freq)

freq[head(freq\_order)]

freq[tail(freq\_order)]

# Visualizing data

freq\_df <- data.frame(word = names(freq), freq = freq)

head(freq\_df)#we can observe that the Italian is highest in number followed by Mexican and southern us

# Plotting terms which appear more than 10,000 times

ggplot(subset(freq\_df, freq >10000), aes(x = word, y = freq)) + geom\_bar(stat = 'identity')# we can observe that pepper, salt ,oil are the most used ingredients

library(wordcloud)

#PLotting on word cloud

wordcloud(names(freq), freq, min.freq = 2500, scale = c(6, .1), colors = brewer.pal(4, "BuPu"))

# Plotting most 5000 used words

wordcloud(names(freq), freq, max.words = 5000, scale = c(6, .1), colors = brewer.pal(6, 'Dark2'))

```

## Modeling {.tabset .tabset-fade .tabset-pills}

### Using CART removing the least used recipies.

```{r}

#only keep terms that appear in 1% or more of the recipes.

sparse <- removeSparseTerms(ingredientsMatirx, 0.99)

sparse

#Converting Corpus matrix into df

ingredientsDTM\_Sparse <- as.data.frame(as.matrix(sparse))

dim(ingredientsDTM\_Sparse)

ingredientsDTM\_Sparse$cuisine <- as.factor(train$cuisine)

ingredientsDTM\_Cuisine <- ingredientsDTM

ingredientsDTM\_Cuisine$cuisine <- as.factor(train$cuisine)

# Creating Model

library(caret)

# Partitioning the data with 75% in train

inTrain <- createDataPartition(y = ingredientsDTM\_Sparse$cuisine, p = 0.75, list = FALSE)

training <- ingredientsDTM\_Sparse[inTrain,]

testing <- ingredientsDTM\_Sparse[-inTrain,]

#CART

library(rpart)

library(rpart.plot)

set.seed(6000)

cartModelFit <- rpart(cuisine ~ ., data = training, method = "class")

## Plot the tree

prp(cartModelFit)

# Predict

cartPredict <- predict(cartModelFit, newdata = testing, type = "class")

cartCM <- confusionMatrix(cartPredict, testing$cuisine)

cartCM

cartPredict\_test <- predict(cartModelFit, newdata = testing, type = "class")

# Not much accuracy as it's 41.39% , lets try using all recipies.

```

###Using CART & using all recipies

```{r}

ingredientsDTM\_Cuisine <- ingredientsDTM

ingredientsDTM\_Cuisine$cuisine <- as.factor(train$cuisine)

dim(ingredientsDTM\_Cuisine)

# Creating Model

library(caret)

# Partitioning the data with 75% in train

inTrain1 <- createDataPartition(y = ingredientsDTM\_Cuisine$cuisine, p = 0.75, list = FALSE)

training1 <- ingredientsDTM\_Cuisine[inTrain1,]

testing1 <- ingredientsDTM\_Cuisine[-inTrain1,]

#CART

library(rpart)

library(rpart.plot)

set.seed(6000)

cartModelFit1 <- rpart(cuisine ~ ., data = training1, method = "class")

## Plot the tree

prp(cartModelFit1)

# Predict

cartPredict1 <- predict(cartModelFit1, newdata = testing1, type = "class")

cartCM1 <- confusionMatrix(cartPredict1, testing1$cuisine)

cartCM1

# Accuracy is 42.97% ; Not much improvement when compared to the previous model, lets use the dimension reduction technique PCA

```

### Modeling with PCA

```{r}

ingredientsDTM\_Cuisine <- ingredientsDTM

ingredientsDTM\_Cuisine$cuisine <- as.factor(train$cuisine)

str(ingredientsDTM\_Cuisine)

# Partitioning the data with 75% in train

inTrain3 <- createDataPartition(y = ingredientsDTM\_Cuisine$cuisine, p = 0.75, list = FALSE)

training3 <- ingredientsDTM\_Cuisine[inTrain3,]

pca.training3 <- subset(training3,select = -c(cuisine))# removing the cuisine(dependent) column

testing3 <- ingredientsDTM\_Cuisine[-inTrain3,]

# Principle component Analysis

prin\_comp <- prcomp(pca.training3)

names(prin\_comp)

dim(prin\_comp$x)

#standard deviation of each principal component

std\_dev <-prin\_comp$sdev

# Computing variance a higher variance implies more informaton is contained in that components.

pr\_var <- std\_dev^2

head(pr\_var)

# proportion of the variation

prop\_varex <- pr\_var/sum(pr\_var)

head(prop\_varex)

# cumulative scree plot to understand teh impact of principle componenets

plot(cumsum(prop\_varex), xlab = "Principal Component",

ylab = "Cumulative Proportion of Variance Explained",

type = "b")

#adding training data set with principle components

training3\_data <- data.frame(cuisine = training3$cuisine,prin\_comp$x)

# Using only the 250 principle componenets.

training3\_data <- training3\_data[, 1:1000]

# CART

library(caret)

library(rpart)

library(rpart.plot)

cartModelFit3 <- rpart(cuisine ~ ., data = training3\_data, method = "class")

prp(cartModelFit3)

#transform test into PCA

testing3\_data <- predict(prin\_comp,newdata = testing3)

testing3\_data <- as.data.frame(testing3\_data)

# Using only the first 250 principle componenets.

testing3\_data <- testing3\_data[,1:1000]

# Predict

cartPredict3 <- predict(cartModelFit3, newdata = testing3\_data,type = "class")

cartCM3 <- confusionMatrix(cartPredict3,testing3$cuisine)

cartCM3

```