What's Cooking

The link to the What's Cooking Kaggle competition can be found here: <https://www.kaggle.com/c/whats-cooking>

Let's start by loading packages we will use later:

library(rpart)  
library(jsonlite)  
library(randomForest)  
library(ggplot2)  
library(tm)  
library(caret)  
library(rpart.plot)  
library(xgboost)  
library(SnowballC)  
library(dplyr) #load packages

Import the training and testing data sets:

traincooking <- fromJSON("traincooking.json", flatten=TRUE)  
testcooking <- fromJSON("testcooking.json", flatten=TRUE) #import train and test data

Create a table to get an idea of how many recipes are in each cuisine:

table(traincooking$cuisine)

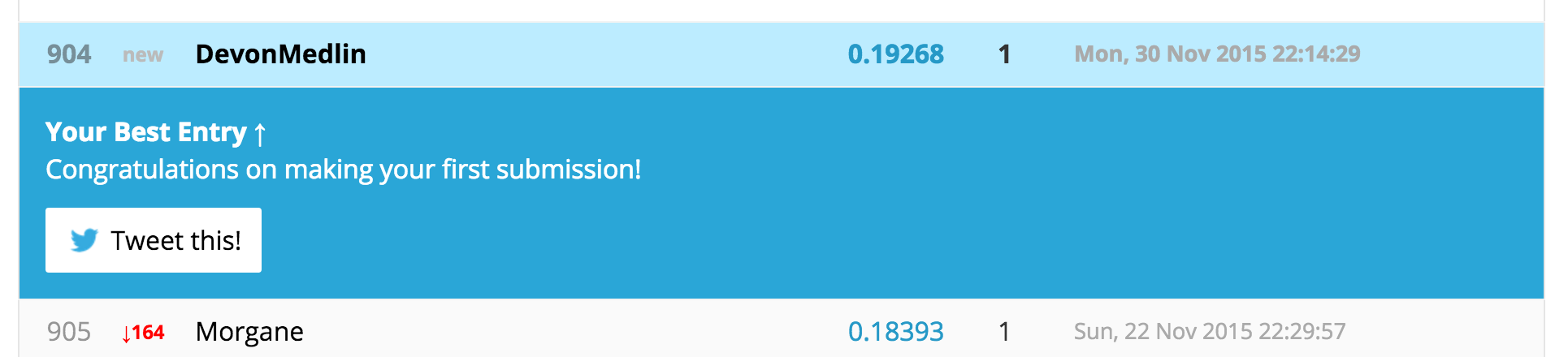
Italian is the most common recipe thus we will set all the recipes equal to Italian for our naïve model:

testcooking$cuisine = "italian"

Create the data frame and csv file used for submission to Kaggle:

naïvemodel <- testcooking[,c('id','cuisine')]   
write.csv(naïvemodel,"model1naïve.csv", row.names=FALSE)

Kaggle results:



This model is not very predictive but sets a baseline that all future models should exceed.

Let's see how a Random Forest model does with this data.  
First we will create a corpus of text to mine:

traincorpus <- Corpus(VectorSource(traincooking$ingredients))  
testcorpus <- Corpus(VectorSource(testcooking$ingredients)) #first step of creating a corpus

Clean the data:

traincorpus <- tm\_map(traincorpus, stemDocument, lazy=TRUE)  
testcorpus <- tm\_map(testcorpus, stemDocument,lazy = TRUE) #reduce words to their base form  
  
traincorpus <- tm\_map(traincorpus, stripWhitespace, lazy=TRUE)  
testcorpus <- tm\_map(testcorpus, stripWhitespace,lazy = TRUE) #trim white space  
  
traincorpus <- tm\_map(traincorpus, tolower, lazy=TRUE)  
testcorpus <- tm\_map(testcorpus, tolower,lazy = TRUE) #make all letters lower case   
  
traincorpus <- tm\_map(traincorpus,PlainTextDocument)  
testcorpus <- tm\_map(testcorpus,PlainTextDocument) #make plain text document

Make a document term matrix:

ingredientsDTM1 <- DocumentTermMatrix(traincorpus)  
ingredientsDTM2 <- DocumentTermMatrix(testcorpus) #create a document term matrix

Remove terms that are very rare:

sparse1 <- removeSparseTerms(ingredientsDTM1, 0.985)  
sparse2 <- removeSparseTerms(ingredientsDTM2, 0.985) #remove terms with a frequency of less than 1.5%

Here we change the edited document term matrices into data frames

ingredientsDTM1df <- as.data.frame(as.matrix(sparse1))  
ingredientsDTM2df <- as.data.frame(as.matrix(sparse2))

View the type of varible of each column:

printout <- sapply(ingredientsDTM1df, class)  
printout

Remove 10 columns that do not exist in both data frames:

traincol <- names(ingredientsDTM1df)  
testcol <- names(ingredientsDTM2df)  
intersect1 <- intersect(traincol, testcol)  
ingredientsDTM1df <- ingredientsDTM1df[,c(intersect1)]  
ingredientsDTM2df <- ingredientsDTM2df[,c(intersect1)]

Set the seed:

set.seed(12345)

More data cleaning:

names(ingredientsDTM1df) <- gsub("-", "", names(ingredientsDTM1df))  
names(ingredientsDTM2df) <- gsub("-", "", names(ingredientsDTM2df))

Change row names from character(0) to a number in the document term matrices:

row.names(ingredientsDTM1df) <- rownames(1:39774)  
row.names(ingredientsDTM2df) <- rownames(1:9944)

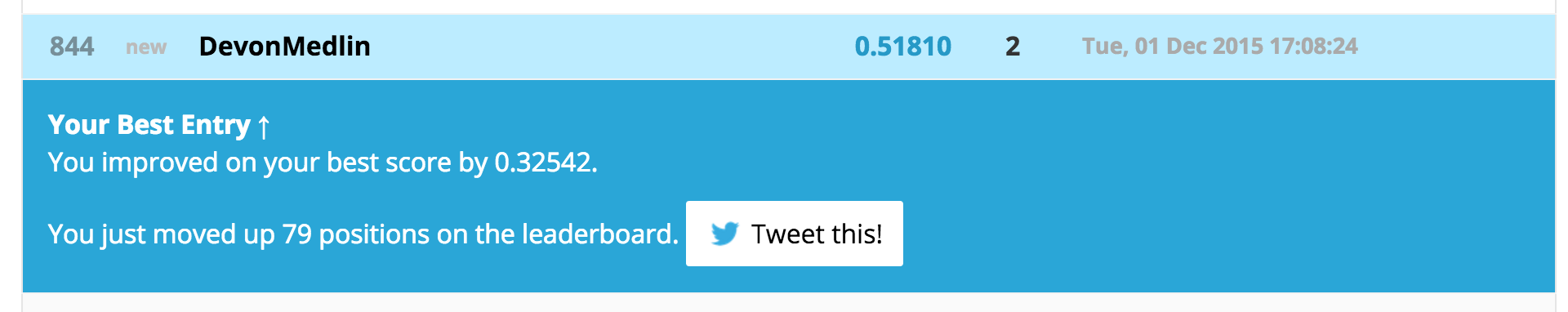
Random Forest model (1 Tree):

rfmodel <- randomForest(as.factor(traincooking$cuisine) ~ . ,data = ingredientsDTM1df,  
importance=TRUE, ntree=1)  
rfpredict <- predict(rfmodel, newdata = ingredientsDTM2df, type = "response")

Create the data frame and csv file used for submission to Kaggle:

submission <- data.frame(id = testcooking$id, cuisine= rfpredict)  
write.csv(submission, file = "randomforest1.csv", row.names = FALSE) #created data frame for submission to Kaggle

Kaggle results:



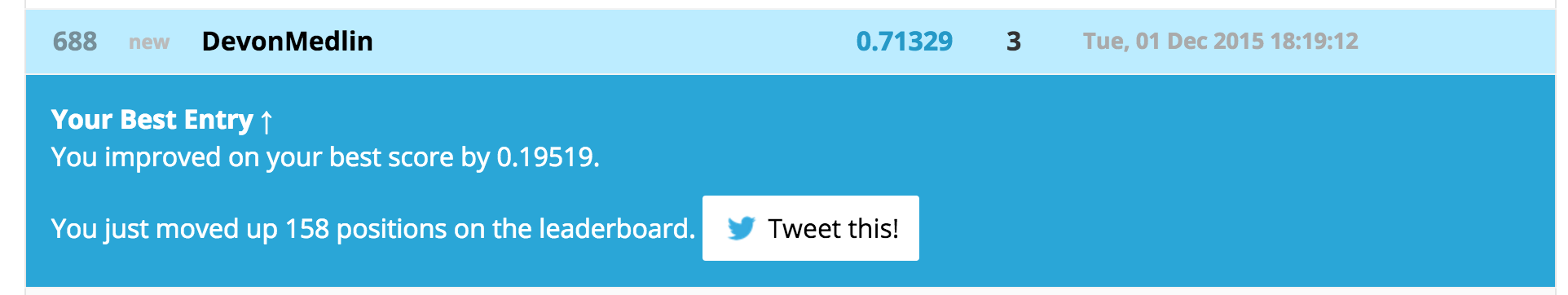
The Random Forest model run with one tree was a substantial improvement on the naïve mode but it still has room to improve. Increasing the number of trees in the model should improve our score.  
Random Forest model (200 Trees):

rfmodel <- randomForest(as.factor(traincooking$cuisine) ~ . ,data = ingredientsDTM1df,  
importance=TRUE, ntree=200)

Create the data frame and csv file used for submission to Kaggle:

submission <- data.frame(id = testcooking$id, cuisine= rfpredict)  
write.csv(submission, file = "randomforest200.csv", row.names = FALSE) #created data frame for submission to Kaggle

Kaggle results:



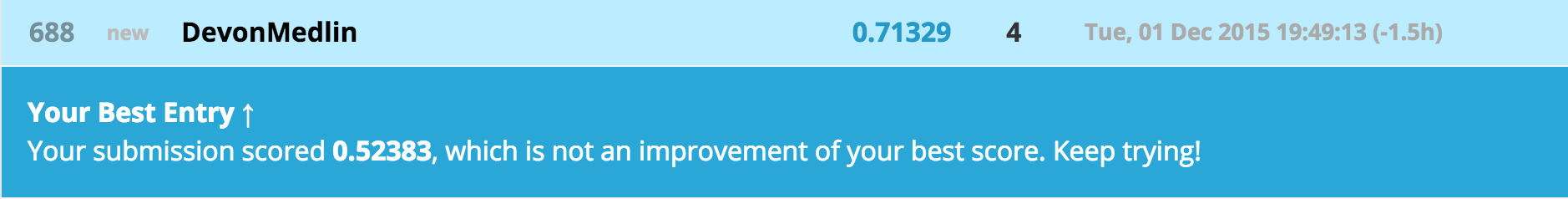
After increasing the number of trees to two hundred the Random Forest model attained an accuracy rating of 0.71329, an improvement of 0.19519.

After I learned that the Random Forest model could predict cuisines correctly at about a 71% accuracy level with two hundred trees I was interested in seeing what would happen if I was able to find certain ingredients that only existed in the cuisine of one country. If I was able to do this, I knew I could filter those ingredients out before running the Random Forest model which would allow Random Forest to create nodes on different ingredients and hopefully increase the prediction accuracy of the overall model. I tried this approach through the use of an if else function with the ingredient garam masala, which is a popular spice in Indian dishes. I only used one tree with this model for processing speed purposes.

Random Forest model (1 Tree) and Garam masala is Indian:

if(traincooking$ingredients %in% "garam masala")  
{testcooking$cuisine="indian"  
} else{rfmodel <- randomForest(as.factor(traincooking$cuisine) ~ . ,data = ingredientsDTM1df,  
importance=TRUE, ntree=1)}

Kaggle results:



This model achieved an accuracy score of 0.52383 (see the lower section of the screenshot) which was a slight improvement on the Random Forest (1 Tree) model’s accuracy level of 0.51810. With this result it was clear that filtering out certain ingredients that were specific to one nationality's cuisine would increase the overall prediction of the model.

Hope you enjoyed the read, and thanks for visiting my site!