Maximizing Recipe Traffic: Insights from Data Analysis

Thomas Roosdorp 2023-01-31

```
knitr::opts_chunk$set(message = FALSE, warning = FALSE)

# for showcasing interpretation of a single seed
set.seed(50)

library(tidyverse)
library(caret)
library(randomForest)
library(pROC)
```

```
# Load the data
url_data <-
  "https://s3.amazonaws.com/talent-assets.datacamp.com/recipe_site_traffic_2212.csv"

recipe_data <- as.data.frame(readr::read_csv(url_data))

# Adding custom functions used in cleaning
detect_outlier <- function(x) {
  Q1 <- quantile(x, probs = 0.25)
  Q3 <- quantile(x, probs = 0.75)
  IQR <- IQR(x)

  x > Q3 + IQR * 1.5 | x < Q1 - IQR * 1.5
}

remove_outlier <- function(df, cols = names(df)) {
  for (col in cols) {
    outliers <- detect_outlier(df[[col]])
    df <- df[!outliers, ]
  }
  return(invisible(df))
}

cube_transform <- function(x) {
  x ^ (1/3)
}
```

Background

I have been tasked to do a data exploration and analysis of a data set on recipes hosted on Tasty Bytes. Also, to use machine learning models to help predict which recipes will receive high traffic, in order to confidently select which recipes to display on on the home page, with the ambition to increase site traffic and in turn increase the number of subscriptions.

Data validation

```
str(recipe_data)
```

```
## 'data.frame':    947 obs. of  8 variables:
##  $ recipe      : chr  "001" "002" "003" "004" ...
##  $ calories    : num  NA 35.5 914.3 97 27.1 ...
```

```
##  $ carbohydrate: num  NA 38.56 42.68 30.56 1.85 ...
##  $ sugar       : num  NA 0.66 3.09 38.63 0.8 ...
##  $ protein     : num  NA 0.92 2.88 0.02 0.53 ...
##  $ category    : chr  "Pork" "Potato" "Breakfast" "Beverages" ...
##  $ servings    : chr  "6" "4" "1" "4" ...
##  $ high_traffic: chr  "High" "High" NA "High" ...
```

The original data set contains 947 observations as well as 8 columns. I began by investigating and validating all variables. The columns are all described as in the presented data dictionary and with summary of changes made:

- recipe: Numeric, unique identifier of recipe
  - Converted to numeric from char; No cleaning needed
- calories: Numeric, number of calories (kcal)
  - Cleaning done: removed outliers and missing
- carbohydrate: Numeric, number of carbohydrates (gram)
  - Cleaning done: removed outliers and missing
- sugar: Numeric, number of sugars (gram)
  - Cleaning done: removed outliers and missing
- protein: Numeric, number of protein (gram)
  - Cleaning done: removed outliers and missing
- category: Nominal, type of recipe. 10 categories
  - Validation: consolidated "Chicken Breast" (n=98) into "Chicken"
- servings: Ordinal, number of servings for the recipe
  - Cleaning and validation: coerced non-valid categories '4 as a snack' (n=2) and '6 as a snack' (n=1) to missing, removed missing
- high_traffic: Nominal, indicator if a recipe received high traffic

Validation done: Only had 'High' values and NA, converted NA to 'Low', and then coerced as factor Validation included compressing Category values "Chicken Breast" into "Chicken" and removing non-valid values from the Servings variable. Outliers that went outside the 1.5 times IQR were removed as well. Missing values were removed completely as imputing them with median values decreased the accuracy in the constructed models.

```
# validate and clean data set
recipe_data_clean <- recipe_data |>
  mutate(
    category = ifelse(category == "Chicken Breast", "Chicken", category), # Consolidate types
    high_traffic = ifelse(is.na(high_traffic), "Low", high_traffic), # Create binary values
    high_traffic = factor(high_traffic, levels = c("Low", "High")), # Turn into factors
    across(c(recipe, servings), as.integer), # Introduces NA for non valid values
    across(c(category, servings), as.factor)) |>
  drop_na() |> # Remove missing values; imputing median did not improve accuracy
  remove_outlier(cols = c("calories", "carbohydrate", "sugar", "protein"))
```

```
# validate 10 types of categories
levels(recipe_data_clean$category)
```

```
##  [1] "Beverages"     "Breakfast"     "Chicken"       "Dessert"
##  [5] "Lunch/Snacks"  "Meat"          "One Dish Meal" "Pork"
```

```
##  [9] "Potato"        "Vegetable"
```

```
# validate 4 types of servings
levels(recipe_data_clean$servings)
```

```
## [1] "1" "2" "4" "6"
```

```
# validate high or low traffic
levels(recipe_data_clean$high_traffic)
```

```
## [1] "Low"  "High"
```

```
# validate any negative values in numeric variables
summary(recipe_data_clean)
```

```
##      recipe         calories       carbohydrate        sugar
##  Min.   :  2.0   Min.   :   0.3   Min.   :  0.03   Min.   : 0.010
##  1st Qu.:245.0   1st Qu.:  96.2   1st Qu.:  7.54   1st Qu.: 1.560
##  Median :485.0   Median : 266.6   Median : 19.16   Median : 3.800
##  Mean   :478.9   Mean   : 364.3   Mean   : 25.99   Mean   : 5.381
##  3rd Qu.:717.0   3rd Qu.: 544.0   3rd Qu.: 38.56   3rd Qu.: 7.820
##  Max.   :947.0   Max.   :1321.8   Max.   :100.70   Max.   :22.390
##
##     protein              category   servings high_traffic
##  Min.   : 0.00   Chicken     :110   1:126    Low :271
##  1st Qu.: 2.88   Breakfast   : 88   2:127    High:390
##  Median : 9.50   Beverages   : 80   4:277
##  Mean   :15.56   Vegetable   : 73   6:131
##  3rd Qu.:23.04   Potato      : 68
##  Max.   :71.88   Lunch/Snacks: 66
##                  (Other)     :176
```

```
str(recipe_data_clean)
```

```
## 'data.frame':    661 obs. of  8 variables:
##  $ recipe      : int  2 3 5 6 7 8 9 11 12 13 ...
##  $ calories    : num  35.5 914.3 27.1 691.1 183.9 ...
##  $ carbohydrate: num  38.56 42.68 1.85 3.46 47.95 ...
##  $ sugar       : num  0.66 3.09 0.8 1.65 9.75 0.4 3.37 4.1 9.78 1.56 ...
##  $ protein     : num  0.92 2.88 0.53 53.93 46.71 ...
##  $ category    : Factor w/ 10 levels "Beverages","Breakfast",..: 9 2 1 7 3 5 8 1 2 9 ...
##  $ servings    : Factor w/ 4 levels "1","2","4","6": 3 1 3 2 3 3 4 1 4 3 ...
##  $ high_traffic: Factor w/ 2 levels "Low","High": 2 1 1 2 1 1 2 1 1 2 ...
```

The data set was reduced down to 661 observations after cleaning.

Exploratory Analysis

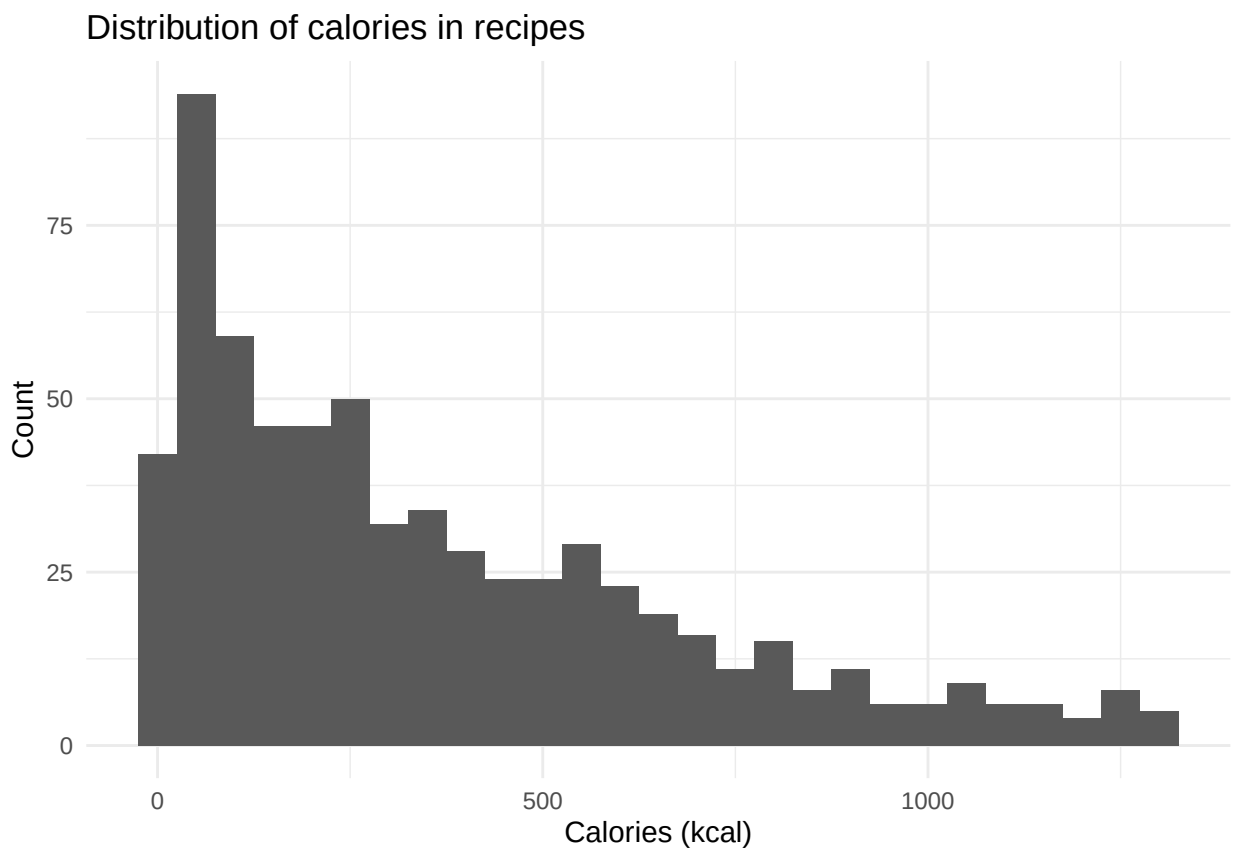
Visual exploration was used to investigate the target variable 'high traffic' and features of the recipe data, as well as the relationship between the target variable and features. Following the preliminary analysis, I decided to implement the following changes to enable more accurate modeling, cube root transformation on all numerical variables (calories, carbohydrate, sugar, protein) as they were heavily skewed in their distribution.

```
# transform numerical variables in cube scale
recipe_data_transf <- recipe_data_clean |>
  mutate(across(c(calories, carbohydrate, sugar, protein), cube_transform))
```

Numeric Variables

We can see from the histogram below that the distribution of calories is substantially skewed to the right, as was the case for all of the other numerical variables. Therefore, I applied a cube transformation (log and square root did not normalize as much) of the numerical variables as previously mentioned, the distribution of the transformed values is closer to a normal distribution.

```
# plot histogram distribution of a single variable
recipe_data_clean |>
  ggplot(aes(x = calories)) +
  geom_histogram(binwidth = 50) +
  labs(title = "Distribution of calories in recipes",
       x = "Calories (kcal)",
       y = "Count") +
  theme_minimal()
```



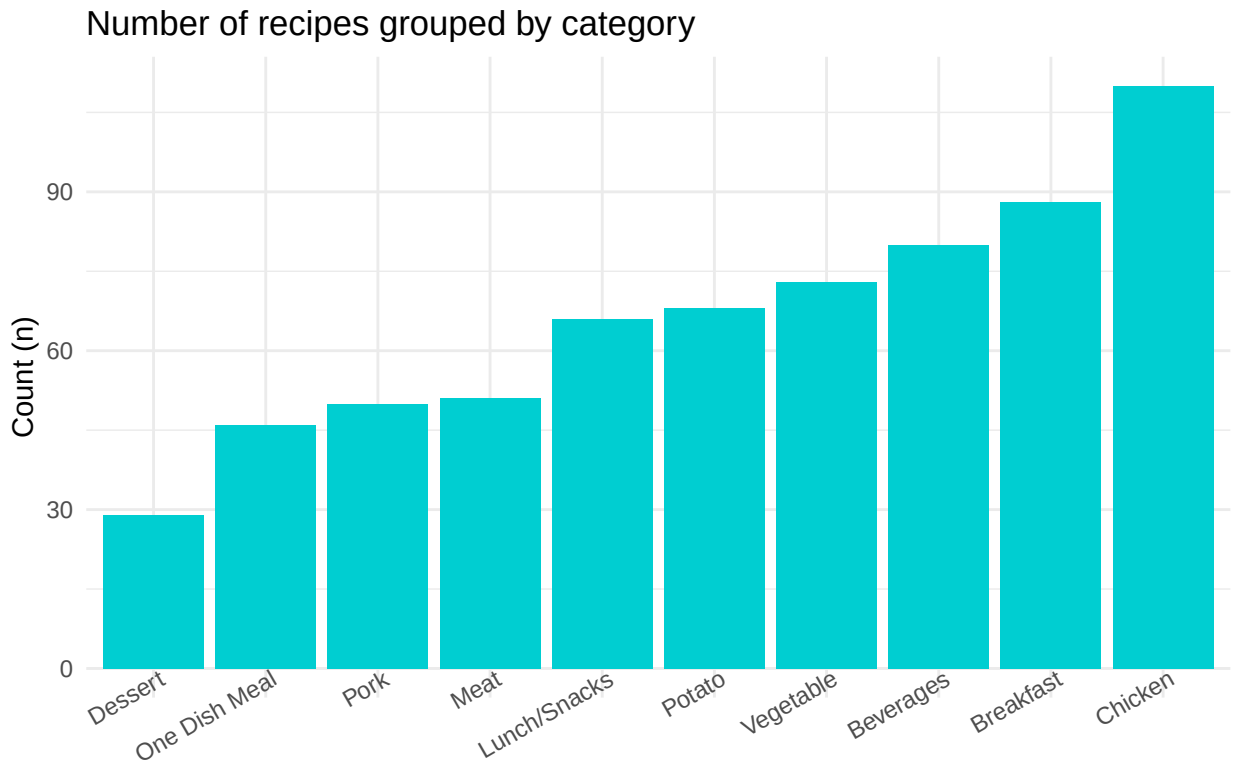
Target Variable - High Traffic

Because we need to predict if a recipe will receive a lot of traffic, we'll use the high_traffic variable as our target variable, also known as the *response variable*.

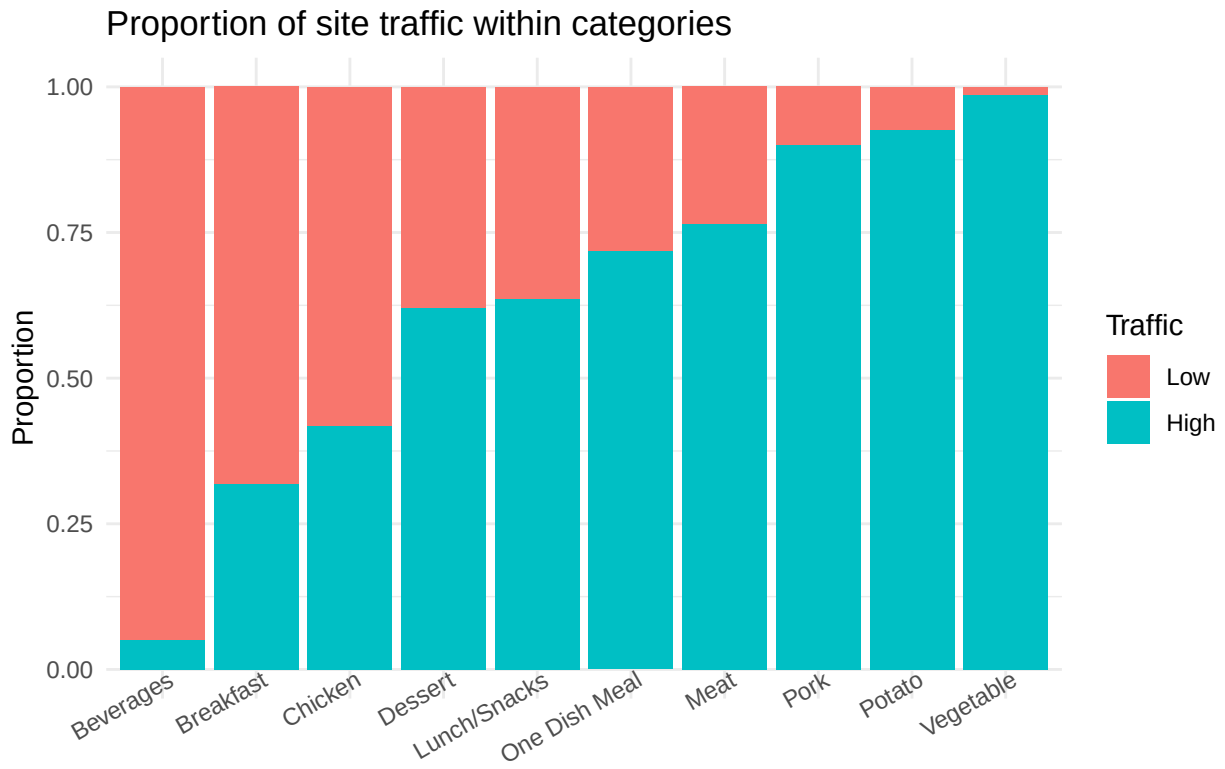
We can see in the graph below that the categories with the fewest recipes are 'Dessert', 'One Dish Meal', and 'Pork', while the categories with the most recipes are 'Beverages', 'Breakfast', and 'Chicken'.

```
# plotting a single variable
recipe_data_clean |>
  ggplot(aes(x = fct_rev(fct_infreq(category)))) +
  geom_bar(fill = "darkturquoise") +
```

```
  labs(title = "Number of recipes grouped by category",
       x = "",
       y = "Count (n)") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 30, vjust = 1.5, hjust = 1))
```




```
# plot the relationship between high_traffic and category
recipe_data_clean |>
  mutate(category = fct_rev(fct_reorder(category,
                                        high_traffic,
                                        .fun = function(.x) mean(as.numeric(.x)),
                                        .desc = TRUE))) |>
  ggplot(aes(x = category, fill = high_traffic)) +
  geom_bar(position = "fill") +
  labs(title = "Proportion of site traffic within categories",
       x = "",
       y = "Proportion",
       fill = "Traffic") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 30, vjust = 1.5, hjust = 1))
```



At first appearance, high traffic recipes appear to be more focused in the categories 'Vegetables' and 'Potato'. Low traffic recipes are typically found in the 'Breakfast' and 'Beverage' categories. In comparison, the 'Chicken' category with the greatest frequency count appears to be closer to the bottom ranked categories in terms of high traffic counts, indicating the possibility for improvement in site traffic. The same goes for the category 'Beverages' which is among the highest count of recipes but being in the most bottom of high traffic proportions.

Model development

Predicting whether or not a recipe will be popular is a classification problem in machine learning. Because the response variable is binary, I've chosen *logistic regression* as the baseline model. The comparison model I am choosing is a *Random Forest classification*.

Model preparation

To enable modeling, I chose *high_traffic* as response variable and the following as explanatory variables: calories, carbohydrate, sugar, protein, category, servings. I have also split the data into a training set and a test set.

```
# Assemble formula for models
response <- "high_traffic"
predictors <- c("calories",
                "carbohydrate",
                "sugar",
                "protein",
                "category",
```

```
                "servings")
fmla_all <- formula(paste(response, "~", paste0(predictors, collapse = " + ")))

# split data set into a training and testing sets
split_index <- createDataPartition(recipe_data_clean$high_traffic,
                                   p = 0.5,
                                   list = FALSE)

train_data  <- recipe_data_clean[split_index, -1]
test_data   <- recipe_data_clean[-split_index, -1]
```

Baseline Model - Logistic Regression

With all explanatory variables, a baseline model was created using logistic regression. Even when normal or transformed, the numerical variables added little significance or accuracy to the models. As a result, they were dropped for the comparison model. The most important predictions were selected (without losing accuracy). The 'category' variable remained the most significant predictor.

```
# Baseline model: Logistic regression with all predictors
model_logistic <- glm(fmla_all, family = "binomial", data = train_data)

summary(model_logistic)
```

```
##
## Call:
## glm(formula = fmla_all, family = "binomial", data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6376  -0.8325   0.2511   0.8278   2.4920
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -2.9157244  0.7268084  -4.012 6.03e-05 ***
## calories               0.0008602  0.0004660   1.846 0.064927 .
## carbohydrate           0.0012243  0.0062924   0.195 0.845732
## sugar                  0.0093200  0.0294278   0.317 0.751465
## protein                0.0014644  0.0093670   0.156 0.875769
## categoryBreakfast      1.9388445  0.7154829   2.710 0.006732 **
## categoryChicken        2.3275211  0.7382481   3.153 0.001617 **
## categoryDessert        2.5050837  0.8266288   3.030 0.002442 **
## categoryLunch/Snacks   2.8543966  0.7518290   3.797 0.000147 ***
## categoryMeat           3.3146191  0.8193116   4.046 5.22e-05 ***
## categoryOne Dish Meal  3.3549794  0.8370361   4.008 6.12e-05 ***
## categoryPork           4.8489599  1.0114644   4.794 1.63e-06 ***
## categoryPotato         4.9429247  0.8840019   5.592 2.25e-08 ***
## categoryVegetable      6.4336200  1.1989424   5.366 8.05e-08 ***
## servings2             -0.2970744  0.4742995  -0.626 0.531090
## servings4             -0.1609684  0.4073113  -0.395 0.692697
## servings6              0.5703399  0.4784820   1.192 0.233270
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 448.29  on 330  degrees of freedom
## Residual deviance: 307.41  on 314  degrees of freedom
## AIC: 341.41
##
## Number of Fisher Scoring iterations: 6
```

```
# Assemble adjusted formula for feature importance selection
fmla_adj <- formula(paste(response, "~", predictors[5]))

# predict data on test set from model
test_data <- test_data |>
  mutate(
    predicted_prob = predict(model_logistic,
                             newdata = test_data,
                             type = "response"),
    predicted = factor(round(predicted_prob), labels = c("Low", "High"))
  )

# create confusion matrix for performance estimates
cm_logistic <- confusionMatrix(test_data$predicted, test_data$high_traffic, positive = "High")
```

Comparison Model

The chosen comparison model is a *Random Forest*, which fared just marginally better than a Decision Tree model. First, I try to find the optimal cutoff point for the Random Forest Model, and then I predict using the model with the best fit.

```
# random forest with single explanatory variable
model_forest <- randomForest(fmla_adj, data = train_data, importance = TRUE)
mtry <- tuneRF(train_data[-7], train_data$high_traffic, trace = FALSE, plot = FALSE)
```

```
## -0.02083333 0.05
## -0.02083333 0.05
```

```
# determine the cutoff for the best fit
best_m <- mtry[mtry[, 2] == min(mtry[, 2]), 1]

model_forest2 <- randomForest(fmla_adj,
                              data = train_data,
                              mtry = best_m,
                              importance = TRUE)

# predict data on test set from model
prob_forest <- predict(model_forest2, newdata = test_data, type = "prob")
pred_forest <- predict(model_forest2, newdata = test_data, type = "class")

# create confusion matrix for performance estimates
cm_forest <- confusionMatrix(test_data$high_traffic, pred_forest, positive = "High")
```

Model evaluation

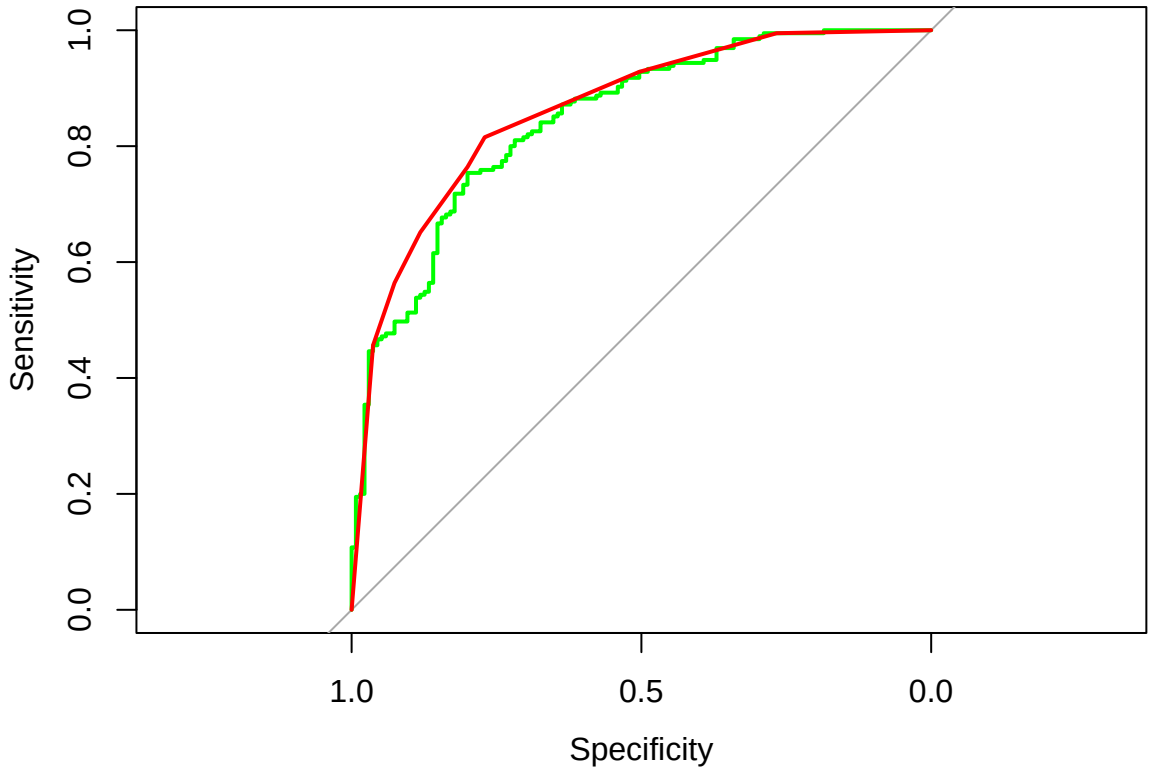
I am evaluating the models using the following metrics, accuracy and AUC (Area Under the Receiver Operating Characteristic). Accuracy is a common metric for evaluating binary classification models which represents the proportion of true results. AUC is a another common metric, and it measures the model's ability to distinguish between the two classes.

```r
# create performance evaluators
roc_logistic <- roc(response = test_data$high_traffic,
                    predictor = test_data$predicted_prob)
roc_forest <- roc(response = test_data$high_traffic,
                  predictor = prob_forest[, 2])

message(paste("Logistic Regression AUC:", round(auc(roc_logistic), 2)))
message(paste("Random Forest AUC:", round(auc(roc_forest), 2)))

message(paste0("Logistic Regression Accuracy: ", round(cm_logistic$overall[1] * 100), "%"))
message(paste0("Random Forest Accuracy: ", round(cm_forest$overall[1] * 100), "%"))

plot(roc_logistic, col = "green")
plot(roc_forest, add = TRUE, col = "red")
```



The AUC of the Logistic Regression is 0.85, and the Random Forest model is 0.86, meaning they perform almost equally well (as seen in the ROC plot). The accuracy of the Logistic Regression is 77% and for the Random Forest model is 80%, meaning the latter is slightly better in predicting values.

Business metrics

Tasty Bytes wants to predict which recipes will achieve high traffic, with an accuracy of approximately 80%. The created models tend to predict with an accuracy close to 80%, but not all the time. I was unable to improve the models accuracy to be consistently above 80%. When adjusting features for the baseline model of Logistic Regression, it produced similar values to the Random Forest. One consideration for future evaluation is that Logistic Regression could rather be used as it tends to be less resource demanding on

systems.

After evaluating the models, I would recommend the company to use the Random Forest model's accuracy as a metric or KPI, in order to compare prediction abilities.

For comparing the performance of the two models with future data, I would suggest using the AUC in order to maintain consistent performance and see if prediction rates improve as well.

Summary

We can employ this Random Forest model to assist the product manager in better predicting which recipes would generate high traffic. By adopting this model, approximately 80% of the predictions will ensure that recipes with high traffic are chosen. This will assist the product manager gain confidence in increasing site visitors.

Based on the initial findings of the analysis, I would propose that the company focus on promoting particular recipe categories that have been observed to be connected with high traffic, such as 'Pork', 'Potato', and 'Vegetable'. This could be accomplished through targeted marketing efforts or by emphasizing specific recipe categories on the website. Furthermore, the company should research why breakfast and beverages are related with low traffic and develop measures to increase the appeal of these dish categories. It is possible that the recipes for breakfast and beverages are not enticing enough, or that they are not adequately marketed. The company might revise the recipes or promote them more effectively.