Project

Predicting High Site Traffic

Introduction:

This project aims to analyze and predict high site traffic using a data-driven approach. Relying on a dataset describing recipe characteristics, we investigate how different features—such as recipe category, nutritional content, and serving size—influence the probability of attracting high traffic.

Several statistical and machine-learning models are implemented and compared, including a Linear Probability Model (LPM), a Logit model, and a Random Forest. Model performance is evaluated using out-of-sample predictions and standard classification metrics, notably confusion matrices and ROC–AUC measures. Beyond predictive accuracy, particular attention is paid to model interpretability to derive actionable insights.

```r
# Packages
library(tidyverse)

library(readxl)

library(rsample)

library(marginaleffects)

library(ranger)

library(yardstick)

library(dbplyr)

############################################################

# 1) DATA WRANGLING
# 1.1 Load data (Excel)
traffic_raw <- read_excel("C:/Users/user/Downloads/recipe_site_traffic_2212
.xlsx")

# Inspect data
glimpse(traffic_raw)

## Rows: 947
## Columns: 8
## $ recipe       <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 
16, 17…
## $ calories     <chr> "NA", "35.48", "914.28", "97.03", "27.05", "691.15", 
"183…
## $ carbohydrate <chr> "NA", "38.56", "42.68", "30.56", "1.85", "3.46", 
"47.95",…
## $ sugar        <chr> "NA", "0.66", "3.09", "38.63", "0.8", "1.65", "9.75", 
"0.…
## $ protein      <chr> "NA", "0.92", "2.88", "0.02", "0.53", "53.93", 
"46.71", "…
## $ category     <chr> "Pork", "Potato", "Breakfast", "Beverages", 
"Beverages", …
## $ servings     <chr> "6", "4", "1", "4", "4", "2", "4", "4", "6", "2", 
"1", "6…
## $ high_traffic <chr> "High", "High", "NA", "High", "NA", "High", "NA", 
"NA", "…

summary(traffic_raw)

##      recipe        calories          carbohydrate          sugar          
##  Min.   :  1.0   Length:947         Length:947         Length:947        
##  1st Qu.:237.5   Class :character   Class :character   Class :character  
##  Median :474.0   Mode  :character   Mode  :character   Mode  :character  
##  Mean   :474.0                                                           
##  3rd Qu.:710.5                                                           
##  Max.   :947.0                                                           
##    protein            category           servings          high_traffic      
##  Length:947         Length:947         Length:947         Length:947        
##  Class :character   Class :character   Class :character   Class :character  
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character  
##                                                                              
##                                                                              
##                                                                              

# 1.2 Recode + type conversion

traffic <- traffic_raw %>%
  mutate(
    high_traffic = case_when(
      high_traffic == "High" ~ 1,
      high_traffic == "NA"   ~ 0,
      is.na(high_traffic)    ~ 0,
      TRUE                   ~ 0
    ),
    category = as.factor(gsub("Chicken Breast", "Chicken", category)),

    # parse_number extracts numeric part safely
    servings     = readr::parse_number(as.character(servings)),
    calories     = readr::parse_number(as.character(calories)),
    carbohydrate = readr::parse_number(as.character(carbohydrate)),
    sugar        = readr::parse_number(as.character(sugar)),
    protein      = readr::parse_number(as.character(protein)),

    recipe = as.character(recipe))


# 1.3 Check missing values share (like Titanic)
miss_share <- traffic %>%
  summarise(across(everything(), ~ mean(is.na(.x)))) %>%
  pivot_longer(everything(), names_to = "variable", values_to =
"share_missing") %>%
  arrange(desc(share_missing))

miss_share
```

```
## # A tibble: 8 × 2
##   variable     share_missing
##   <chr>                <dbl>
## 1 calories            0.0549
## 2 carbohydrate        0.0549
## 3 sugar               0.0549
## 4 protein             0.0549
## 5 recipe              0
## 6 category            0
## 7 servings            0
## 8 high_traffic        0
```

```
# 1.4 Drop missing values (course-consistent approach)
# Here we drop rows missing predictors (and servings) after converting to
numeric.
traffic_clean <- traffic %>%
  drop_na(calories, carbohydrate, sugar, protein, servings, category,
high_traffic)

# 1.5 Distortion check (outcome before vs after dropping missings)
prop_before <- mean(traffic$high_traffic)          # no NA now because we set
NA -> 0
prop_after  <- mean(traffic_clean$high_traffic)

cat("\nProportion High Traffic BEFORE drop_na():", round(prop_before, 4),
"\n")
```

```
##
## Proportion High Traffic BEFORE drop_na(): 0.6061
```

```
cat("Proportion High Traffic AFTER  drop_na():", round(prop_after, 4), "\n")
```

```
## Proportion High Traffic AFTER  drop_na(): 0.5978
```

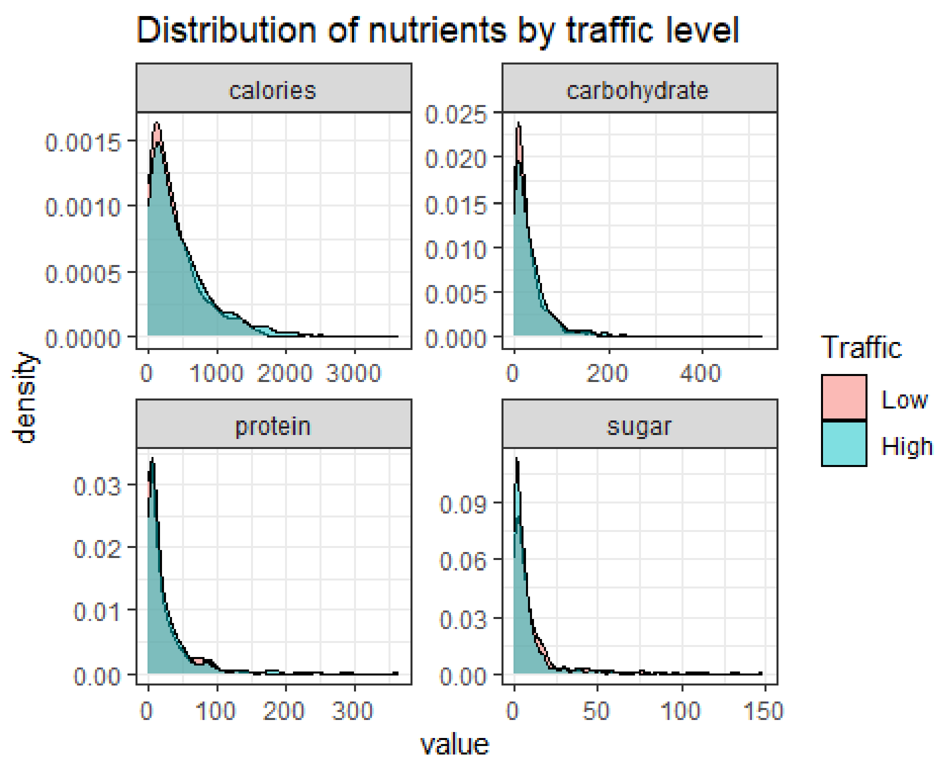
```
cat("Difference (after - before):", round(prop_after - prop_before, 4), "\n")
```

```
## Difference (after - before): -0.0084

cat("\nN before:", nrow(traffic), "\n")

##
## N before: 947

cat("N after :", nrow(traffic_clean), "\n")

## N after : 895
```

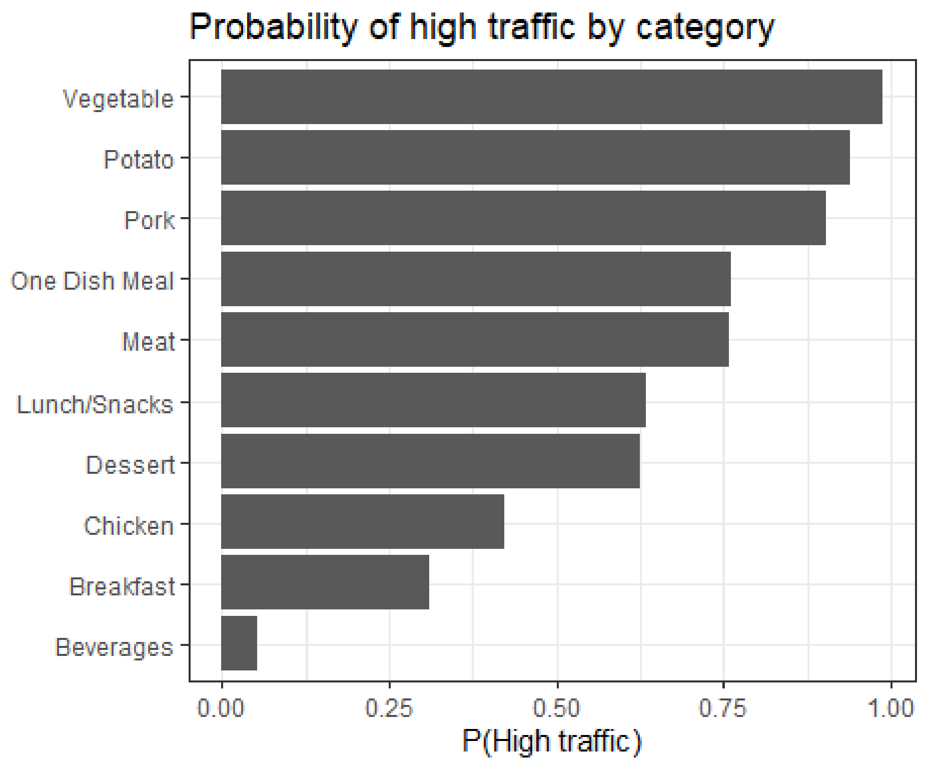
Interpretation :

```
# The dataset studied contains 947 recipes identified by nutritional
characteristics (calories, carbohydrate, sugar, protein), recipe category,
number of servings, and an indicator of whether the traffic to the site was
high when this recipe was shown.


# We created a response variable "high_traffic" from the same data frame as a
subset, originally recorded as "High" for recipes that generate high traffic,
and recoded into a binary variable equal to 1 and 0 otherwise ("NA"). We also
created two categories (“Chicken” and “Chicken Breast”) for consistency and
converted the variable "category" in the data into a factor and nutritional
variables and "servings" into numeric after cleaning the data.


# The data had missing values in the nutritional variables that needed to be
dropped. To verify that we had the desired outcome, we compared the
proportion of high-traffic recipes before and after dropping missing values.
The proportion decreased slightly from 0.606 to 0.598, which showed that the
outcome distribution was not distorted.
```

```
############################################################
# 2) EDA (lesson-style: group means = probabilities)
############################################################

# Make outcome as factor for nicer plots
traffic_clean <- traffic_clean %>%
  mutate(high_traffic_f = factor(high_traffic, levels = c(0, 1), labels =
c("Low", "High")))

# 2.1 Nutrient distributions by traffic (density plots)
traffic_clean %>%
  pivot_longer(cols = c(calories, carbohydrate, sugar, protein),
               names_to = "nutrient", values_to = "value") %>%
  ggplot(aes(x = value, fill = high_traffic_f)) +
  geom_density(alpha = 0.5) +
  facet_wrap(~ nutrient, scales = "free") +
  theme_bw() +
  labs(title = "Distribution of nutrients by traffic level",
       fill = "Traffic")
```



```
# 2.2 Probability of High traffic by category (mean of binary outcome)
traffic_clean %>%
  group_by(category) %>%
  summarise(prob_high = mean(high_traffic), .groups = "drop") %>%
  ggplot(aes(x = reorder(category, prob_high), y = prob_high)) +
  geom_col() +
  coord_flip() +
  theme_bw() +
  labs(title = "Probability of high traffic by category",
       x = "", y = "P(High traffic)")
```



Interpretation :

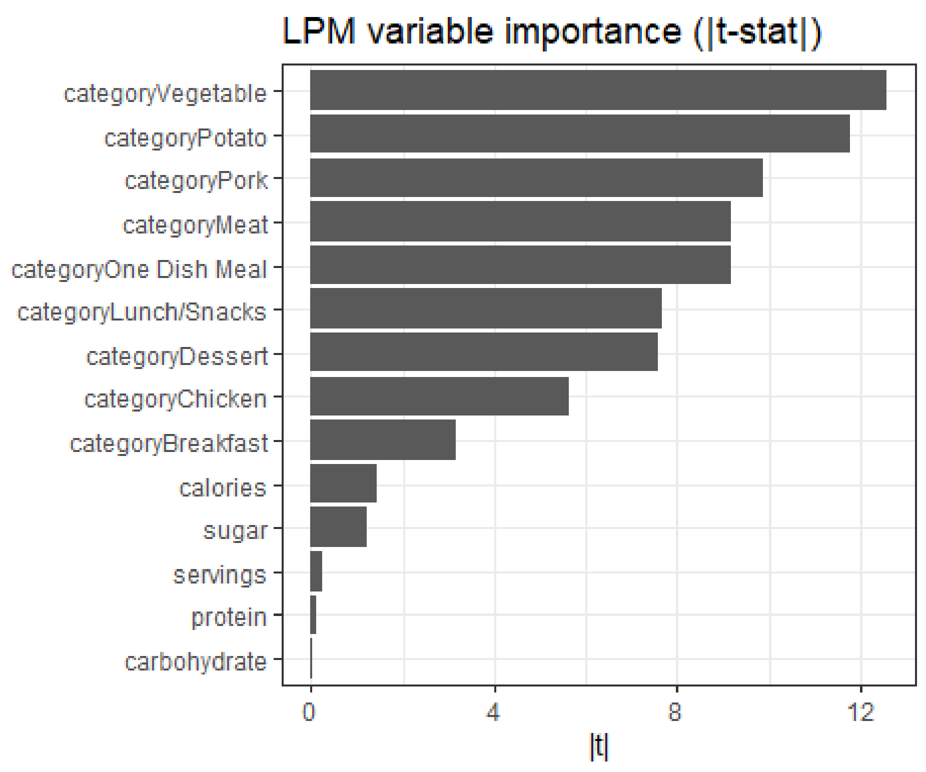
```
# Density plots indicate that recipes generating high traffic tend to have
higher calorie and protein content, while sugar and carbohydrate
distributions overlap substantially across varying traffic levels. This
suggests that nutritional richness may matter, but its effect is likely non-
linear or secondary relative to other factors.


# Talking about grouping (categories of recipes), we see that high traffic is
generated by recipes listed in the categories "Vegetable", "Potato", "Pork",
"One Dish Meal", and "Meat" compared to "Beverages", which serves as the
baseline category. This means that site visits are more likely to be
generated by full-meal recipes than lighter items such as drinks or breakfast
recipes.


# Overall, high site traffic is largely explained by recipe category, while
nutritional variables may play a more limited role.

############################################################
# 3) TRAIN/TEST SPLIT + (where relevant) 10-fold CV
############################################################

set.seed(123)
split_obj <- initial_split(traffic_clean, prop = 0.7, strata = high_traffic)
train_data <- training(split_obj)
```

```
test_data  <- testing(split_obj)

# 10-fold CV object (lesson-style)
set.seed(123)
cv_splits <- vfold_cv(train_data, v = 10, strata = high_traffic)
cv_splits

## #  10-fold cross-validation using stratification
## # A tibble: 10 × 2
##    splits            id
##    <list>            <chr>
##  1 <split [561/64]> Fold01
##  2 <split [562/63]> Fold02
##  3 <split [562/63]> Fold03
##  4 <split [562/63]> Fold04
##  5 <split [563/62]> Fold05
##  6 <split [563/62]> Fold06
##  7 <split [563/62]> Fold07
##  8 <split [563/62]> Fold08
##  9 <split [563/62]> Fold09
## 10 <split [563/62]> Fold10
```

Interpretation :

```
# The train-test split was divided between a training set (70%) and a test
set (30%) by using stratification on the response variable to maintain the
proportion of high-traffic recipes among samples. This helps to make sure
that the model performance is evaluated using unseen data while conserving
class balance.


############################################################
# 4) LPM + LOGIT, Marginal Effects, Variable Importance (|t|)
############################################################

# 4.1 LPM (Linear Probability Model)
lpm_model <- lm(
  high_traffic ~ calories + carbohydrate + sugar + protein + servings +
category,
  data = train_data
)
summary(lpm_model)

##
## Call:
## lm(formula = high_traffic ~ calories + carbohydrate + sugar +
##     protein + servings + category, data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.00978 -0.26319  0.02936  0.24564  1.01750
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)            3.420e-02  6.364e-02   0.537  0.59125
## calories               5.231e-05  3.621e-05   1.445  0.14904
## carbohydrate          -1.672e-05  3.902e-04  -0.043  0.96585
## sugar                 -1.697e-03  1.385e-03  -1.226  0.22083
## protein               -6.154e-05  5.064e-04  -0.122  0.90332
## servings               2.326e-03  9.167e-03   0.254  0.79980
## categoryBreakfast      2.183e-01  6.884e-02   3.172  0.00159 **
## categoryChicken        3.950e-01  6.990e-02   5.651 2.45e-08 ***
## categoryDessert        6.368e-01  8.423e-02   7.560 1.48e-13 ***
## categoryLunch/Snacks   5.669e-01  7.402e-02   7.659 7.39e-14 ***
## categoryMeat           7.200e-01  7.861e-02   9.160  < 2e-16 ***
## categoryOne Dish Meal  7.611e-01  8.311e-02   9.158  < 2e-16 ***
## categoryPork           8.116e-01  8.199e-02   9.898  < 2e-16 ***
## categoryPotato         9.076e-01  7.722e-02  11.754  < 2e-16 ***
## categoryVegetable      9.372e-01  7.457e-02  12.568  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3965 on 610 degrees of freedom
## Multiple R-squared:  0.3615, Adjusted R-squared:  0.3468
## F-statistic: 24.67 on 14 and 610 DF,  p-value: < 2.2e-16
```

```r
# Variable importance for LPM = |t-stat|
lpm_t <- summary(lpm_model)$coefficients[-1, 3]
lpm_varimp <- tibble(variable = names(lpm_t), score = abs(lpm_t)) %>%
  arrange(desc(score))

ggplot(lpm_varimp, aes(x = reorder(variable, score), y = score)) +
  geom_col() +
  coord_flip() +
  theme_bw() +
  labs(title = "LPM variable importance (|t-stat|)", x = "", y = "|t|")
```



```r
# 4.2 Logit (GLM)
logit_model <- glm(
  high_traffic ~ calories + carbohydrate + sugar + protein + servings +
category,
  data = train_data,
  family = binomial(link = "logit")
)
summary(logit_model)
```

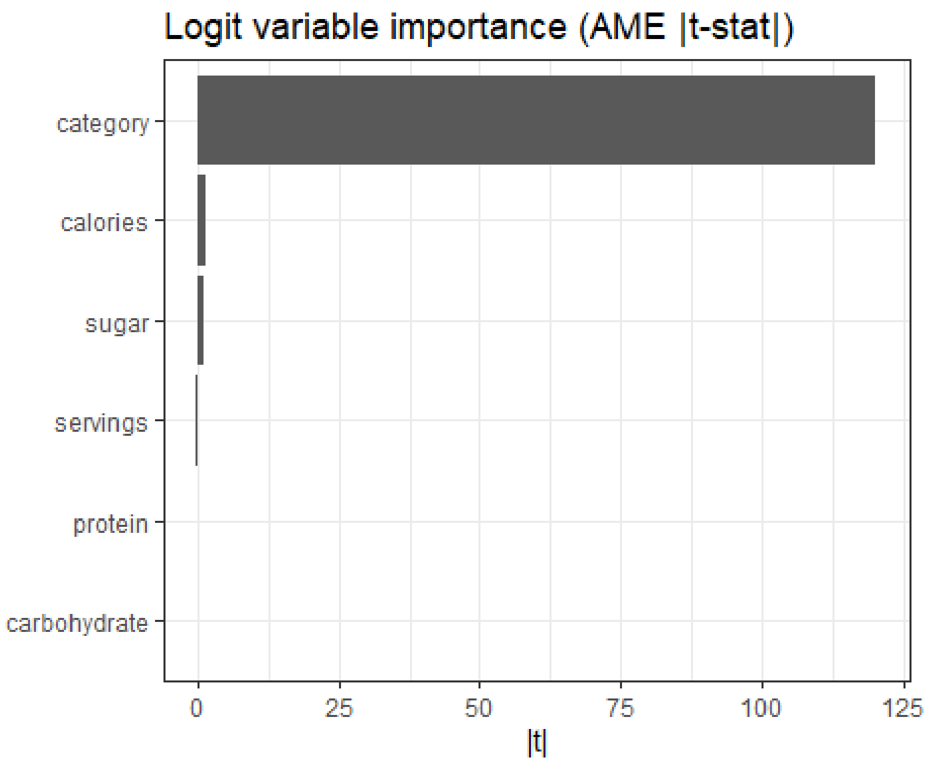
```
##
## Call:
## glm(formula = high_traffic ~ calories + carbohydrate + sugar +
##     protein + servings + category, family = binomial(link = "logit"),
##     data = train_data)
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -3.3837373  0.7581768  -4.463 8.08e-06 ***
## calories                 0.0003115  0.0002263   1.377  0.16863
## carbohydrate            -0.0001891  0.0026870  -0.070  0.94389
## sugar                   -0.0086345  0.0079913  -1.080  0.27993
## protein                 -0.0003480  0.0028545  -0.122  0.90297
## servings                 0.0141884  0.0585342   0.242  0.80847
## categoryBreakfast        2.2774918  0.7668707   2.970  0.00298 **
## categoryChicken          3.0421764  0.7626770   3.989 6.64e-05 ***
## categoryDessert          4.1142623  0.8133598   5.058 4.23e-07 ***
## categoryLunch/Snacks     3.7490150  0.7737162   4.845 1.26e-06 ***
## categoryMeat           4.4963286  0.8038411   5.594 2.22e-08 ***
## categoryOne Dish Meal  4.7577764  0.8341402   5.704 1.17e-08 ***
## categoryPork           5.1663116  0.8555245   6.039 1.55e-09 ***
## categoryPotato         6.5190298  1.0267397   6.349 2.16e-10 ***
## categoryVegetable      7.3080014  1.2401195   5.893 3.79e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 842.07  on 624  degrees of freedom
## Residual deviance: 577.37  on 610  degrees of freedom
## AIC: 607.37
##
## Number of Fisher Scoring iterations: 6
```

```
# Average Marginal Effects (AME)
ame <- avg_slopes(logit_model)
ame
```

```
##
##         Term                  Contrast  Estimate Std. Error       z
Pr(>|z|)
##  calories     dY/dX                      4.78e-05   3.45e-05  1.3850
0.166
##  carbohydrate dY/dX                     -2.90e-05   4.12e-04 -0.0704
0.944
##  category     Breakfast - Beverages      2.32e-01   5.52e-02  4.2062
<0.001
##  category     Chicken - Beverages        4.03e-01   5.79e-02  6.9611
<0.001
##  category     Dessert - Beverages        6.58e-01   8.25e-02  7.9747
<0.001
##  category     Lunch/Snacks - Beverages   5.76e-01   6.71e-02  8.5897
<0.001
##  category     Meat - Beverages           7.32e-01   6.50e-02 11.2515
<0.001
##  category     One Dish Meal - Beverages  7.75e-01   6.62e-02 11.7024
<0.001
##  category     Pork - Beverages           8.29e-01   5.78e-02 14.3578
<0.001
##  category     Potato - Beverages         9.25e-01   3.73e-02 24.8092
<0.001
##  category     Vegetable - Beverages      9.45e-01   3.11e-02 30.3512
<0.001
##  protein      dY/dX                     -5.33e-05   4.38e-04 -0.1219
0.903
##  servings     dY/dX                      2.17e-03   8.97e-03  0.2425
0.808
##  sugar        dY/dX                     -1.32e-03   1.22e-03 -1.0851
```

```
0.278
##      S     2.5 %   97.5 %
##    2.6 -1.98e-05 0.000115
##    0.1 -8.36e-04 0.000778
##   15.2  1.24e-01 0.340614
##   38.1  2.90e-01 0.516824
##   49.2  4.96e-01 0.819410
##   56.7  4.45e-01 0.707521
##   95.1  6.04e-01 0.859321
##  102.7  6.45e-01 0.904554
##  152.9  7.16e-01 0.942486
##  448.9  8.52e-01 0.997720
##  669.8  8.84e-01 1.006408
##    0.1 -9.11e-04 0.000804
##    0.3 -1.54e-02 0.019753
##    1.8 -3.71e-03 0.001067
##
## Type: response
```

```r
# Variable importance for Logit = |t-stat| of AME
logit_varimp <- ame %>%
  mutate(score = abs(statistic)) %>%
  arrange(desc(score))

ggplot(logit_varimp, aes(x = reorder(term, score), y = score)) +
  geom_col() +
  coord_flip() +
  theme_bw() +
  labs(title = "Logit variable importance (AME |t-stat|)", x = "", y = "|t|")
```



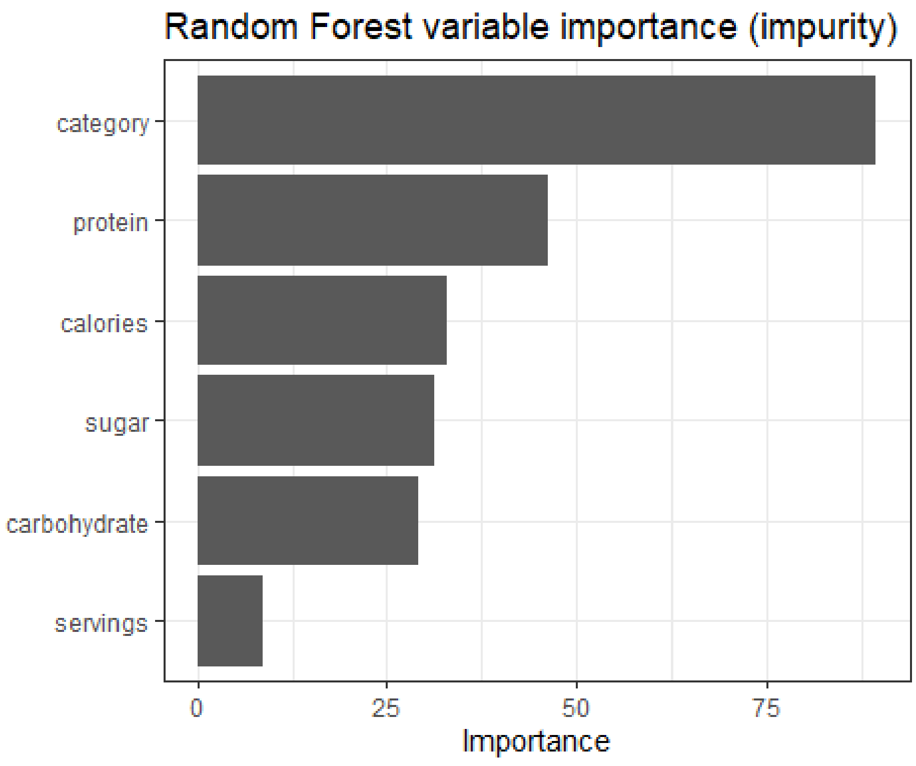
Interpretation :

# We used OLS to estimate a linear probability model (LPM). As a result, most nutritional variables are not statistically significant. However, the variable "category" shows a strong and statistically significant effect on high-traffic probability. Also, given the baseline category "Beverages", all main-dish categories are related to higher probabilities of high traffic, showing larger effects for "Vegetable", "Potato", "Pork", and "one dish meal" recipes.

# Given the binary nature of the response variable, we had to estimate a Logit model and observe the results. It is shown that the signs and significance patterns are equivalent to the LPM model. To allow direct comparison with the LPM coefficients, we computed Average Marginal Effects (AMEs); they confirmed that category effects dominate. For example, for a "vegetable" recipe, the probability of high traffic is increased by more than 90 percentage points relative to Beverages, but in contrast, other nutritional variables are statistically insignificant.

# In order to show the variable importance, we used the absolute value of the t-statistics of the marginal effects. We say that both models show that recipe category is the most important key factor, with nutritional variables playing a less significant role.

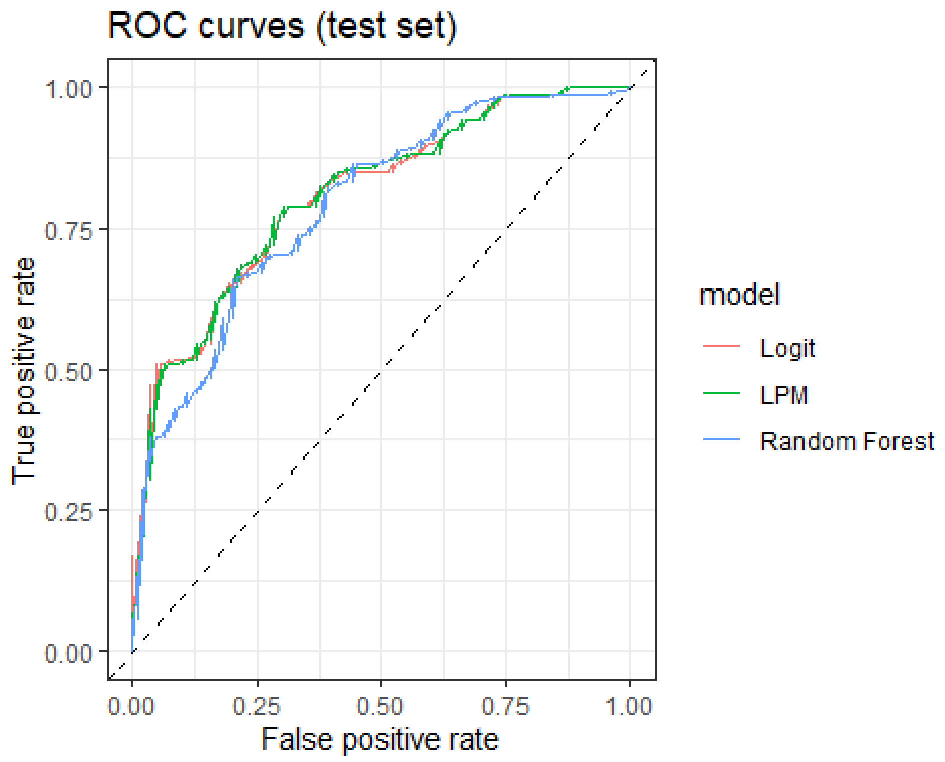
```r
############################################
# 5) RANDOM FOREST
############################################

# 5.1 We'll choose mtry that maximizes AUC on validation folds.

# Helper function to compute AUC for one fold
compute_auc_rf <- function(train_df, valid_df, mtry_value) {
  fit <- ranger(
    high_traffic_f ~ calories + carbohydrate + sugar + protein + servings +
category,
    data = train_df %>% mutate(high_traffic_f = factor(high_traffic,
levels=c(0,1), labels=c("Low","High"))),
    probability = TRUE,
    mtry = mtry_value,
    num.trees = 500,
    importance = "impurity"
  )
  valid_df2 <- valid_df %>%
    mutate(high_traffic_f = factor(high_traffic, levels=c(0,1),
labels=c("Low","High")))
  prob <- predict(fit, valid_df2)$predictions[, "High"]
  yardstick::roc_auc(valid_df2, truth = high_traffic_f, prob, event_level =
"second")$.estimate
}

# Grid for mtry
library(tidyverse)
library(rsample)
library(ranger)
library(yardstick)
library(marginaleffects)
mtry_grid <- 1:6

set.seed(123)
compute_auc_rf <- function(train_df, valid_df, mtry_value) {

  # We wanna ensure outcome is a factor with levels Low/High
  train_df2 <- train_df %>%
    mutate(high_traffic_f = factor(high_traffic, levels = c(0, 1), labels =
c("Low", "High")))

  valid_df2 <- valid_df %>%
    mutate(high_traffic_f = factor(high_traffic, levels = c(0, 1), labels =
c("Low", "High")))

  # Fit Random Forest
  fit <- ranger(
    high_traffic_f ~ calories + carbohydrate + sugar + protein + servings +
category,
    data = train_df2,
    probability = TRUE,
    num.trees = 500,
    mtry = mtry_value,
    importance = "impurity")

  # We get predicted probabilities for "High"
  valid_df2 <- valid_df2 %>%
    mutate(prob_rf = predict(fit, valid_df2)$predictions[, "High"])

  # AUC
  yardstick::roc_auc(valid_df2, truth = high_traffic_f, prob_rf, event_level
= "second")$.estimate}

cv_results <- map_dfr(mtry_grid, function(m) {
  aucs <- map_dbl(cv_splits$splits, function(spl) {
    tr <- analysis(spl)
    va <- assessment(spl)
    compute_auc_rf(tr, va, m)
  })
  tibble(mtry = m, auc_mean = mean(aucs), auc_sd = sd(aucs))})

cv_results

## # A tibble: 6 × 3
##    mtry auc_mean auc_sd
##   <int>    <dbl>  <dbl>
## 1     1    0.821 0.0507
## 2     2    0.824 0.0441
## 3     3    0.819 0.0432
## 4     4    0.813 0.0441
## 5     5    0.810 0.0444
## 6     6    0.805 0.0427

best_mtry <- cv_results %>% arrange(desc(auc_mean)) %>% slice(1) %>%
pull(mtry)
best_mtry

## [1] 2

# 5.2 Fit final RF on full training set
train_data <- train_data %>%
  mutate(high_traffic_f = factor(high_traffic, levels=c(0,1),
labels=c("Low","High")))
test_data <- test_data %>%
  mutate(high_traffic_f = factor(high_traffic, levels=c(0,1),
labels=c("Low","High")))

rf_model <- ranger(
  high_traffic_f ~ calories + carbohydrate + sugar + protein + servings +
category,
  data = train_data,
  probability = TRUE,
  num.trees = 500,
  mtry = best_mtry,
  importance = "impurity")

rf_model

## Ranger result
##
## Call:
##  ranger(high_traffic_f ~ calories + carbohydrate + sugar + protein +
servings + category, data = train_data, probability = TRUE,      num.trees =
500, mtry = best_mtry, importance = "impurity")
##
## Type:                             Probability estimation
## Number of trees:                  500
## Sample size:                      625
## Number of independent variables:  6
## Mtry:                             2
## Target node size:                 10
## Variable importance mode:         impurity
## Splitrule:                        gini
## OOB prediction error (Brier s.):  0.1694704

# Variable importance
rf_varimp <- tibble(
  variable = names(rf_model$variable.importance),
  importance = as.numeric(rf_model$variable.importance)
) %>% arrange(desc(importance))

ggplot(rf_varimp, aes(x = reorder(variable, importance), y = importance)) +
  geom_col() +
  coord_flip() +
  theme_bw() +
  labs(title = "Random Forest variable importance (impurity)", x = "", y =
"Importance")
```



Interpretation :

```
# As in the previous results, variable importance measures from the Random
Forest confirm that recipe category is the most influential predictor,
followed by nutritional variables such as calories and protein. This result
is consistent with the linear models but we used it to highlight the
flexibility of the RF in capturing non-linearities and interactions between
predictors in the data.

#################################################
# 6) PREDICTION + CONFUSION MATRICES + ROC/AUC
#################################################

# 6.1 Predicted probabilities on test set
test_data <- test_data %>%
  mutate(
    prob_lpm   = predict(lpm_model, newdata = test_data),
    prob_logit = predict(logit_model, newdata = test_data, type =
"response"))

# RF probabilities
test_data$prob_rf <- predict(rf_model, test_data)$predictions[, "High"]

# 6.2 Class predictions
test_data <- test_data %>%
  mutate(
    pred_lpm   = factor(ifelse(prob_lpm   >= 0.5, "High", "Low"), levels =
c("Low","High")),
    pred_logit = factor(ifelse(prob_logit >= 0.5, "High", "Low"), levels =
c("Low","High")),
    pred_rf    = factor(ifelse(prob_rf    >= 0.5, "High", "Low"), levels =
c("Low","High")))

# 6.3 Confusion matrices
conf_mat(test_data, truth = high_traffic_f, estimate = pred_lpm)

##           Truth
## Prediction Low High
##       Low   74   34
##       High  35  127

conf_mat(test_data, truth = high_traffic_f, estimate = pred_logit)

##           Truth
## Prediction Low High
##       Low   73   34
##       High  36  127

conf_mat(test_data, truth = high_traffic_f, estimate = pred_rf)

##           Truth
## Prediction Low High
##       Low   67   36
##       High  42  125

# 6.4 ROC + AUC
roc_lpm   <- roc_curve(test_data, truth = high_traffic_f, prob_lpm,
event_level = "second")
roc_logit <- roc_curve(test_data, truth = high_traffic_f, prob_logit,
event_level = "second")
roc_rf    <- roc_curve(test_data, truth = high_traffic_f, prob_rf,
event_level = "second")

auc_lpm   <- roc_auc(test_data, truth = high_traffic_f, prob_lpm,
event_level = "second")
auc_logit <- roc_auc(test_data, truth = high_traffic_f, prob_logit,
event_level = "second")
auc_rf    <- roc_auc(test_data, truth = high_traffic_f, prob_rf,
event_level = "second")

auc_lpm

## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.808
```

```
auc_logit

## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.809

auc_rf

## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.790

# Plot ROC curves together
bind_rows(
  roc_lpm   %>% mutate(model = "LPM"),
  roc_logit %>% mutate(model = "Logit"),
  roc_rf    %>% mutate(model = "Random Forest")
) %>%
  ggplot(aes(x = 1 - specificity, y = sensitivity, color = model)) +
  geom_line() +
  geom_abline(linetype = "dashed") +
  theme_bw() +
  labs(title = "ROC curves (test set)", x = "False positive rate", y = "True
positive rate")
```



Interpretation :

```
# The three models almost have the same reasonable performance; as for the
LPM anf Logit models, they achieved very equal AUC values (0.808 and 0.809,
respectively), and for the Random Forest model, a slightly lower AUC was
observed (0.790). Also, confusion matrices were accurate in terms of
classification across models, with no significant or clear dominance of the
Random Forest.


# We can say that the Logit model is the preferred model, because it achieves
the highest AUC on the test set (≈ 0.81), which is slightly greater than the
LPM (0.808) and the Random Forest (0.79), also the performance of the LPM and
Logit models are nearly identical. Moreover, the Logit model gives better
insights for interpreting and understanding through average marginal effects
(AME), which is beneficial for actionable decisions. In contrast, despite its
flexibility, the Random Forest doesn't help predict performance since it
suggests that simpler parametric models largely capture the relationship
between predictors and traffic. Given that, the Logit model provides the
optimal balance between interpretability and forecast accuracy.

############################################################

# 7 Interpretations and recommendations:

# Based on the preferred Logit model and the previous results, the variable
recipe category is shown to be by far the most significant predictor of high
site traffic. The other variables such as vegetable-based dishes, potatoes,
pork, meat, and one-dish meals are much more likely to generate high traffic
than beverages or breakfast items. Nutritional variables and the number of
servings have less explanatory effect once the variable category is
controlled for.


# Yes, the model performs reasonably well, given an AUC around 0.81 shows
good identifying ability. In about 81% of cases, the model can correctly
classify high-traffic recipes above low-traffic ones. As for confusion
matrices, they helped by showing a good balance between true positives and
false positives, with no extreme misclassification. For that, the results are
robust rather than model-specific given the stable performance across models.
Despite the good performance of the model, it remains not perfect, which
means that unobserved factors such as visual appeal, marketing, seasonality…
are also necessary to increase the precision of the model.


# In order to increase site traffic, many concrete actions can be recommended
such as making vegetable, potato, pork, meat, and one-dish meal recipes more
visible and frequent and promote them on the homepage, newsletters, or
recommendation panels. Also, we can improve categories with systematically
lower traffic (e.g. Beverages) through better presentation and adjustments of
the nutritional metrics. Regarding the model, it can be enriched to further
```

```
improve predictive power by adding other predictors such as images quality,
user ratings, seasonality, or other promotion indicators.


# Overall, we conclude that the analysis conducted shows that the
presentation of the type of recipe weighs far more than its nutritional
composition. Given this result, promoting editorial and marketing efforts on
the most attractive recipe categories can be an effective strategy to
increase site traffic.
```