P3_code

1. Setup

References

Data dictionary: https://nhts.ornl.gov/tables09/CodebookBrowser.aspx (https://nhts.ornl.gov/tables09/CodebookBrowser.aspx)

P3 assignment instructions: https://gsd-ses-5394-sp2025.github.io/examples/P3/P3 (https://gsd-ses-5394-sp2025.github.io/examples/P3/P3)

Loading libraries

```
library(tidyverse)
library(dplyr)
library(caret)
library(pROC)
library(MASS)
library(mgcv)
library(smotefamily)
```

Loading data

```
# Loading person-level data
person_data <- read.csv("/Users/paigelee/Documents/SES 5394/ses5394/P2_vehicle_availability/data/perpub.csv")

# Loading trip data
trip_data <- read.csv("/Users/paigelee/Documents/SES 5394/ses5394/P2_vehicle_availability/data/trippub.csv")
```

2. Feature selection and engineering

Creating the response variable

Create a binary response variable with values of 1 if the trip destination purpose (WHYTO) was for medical trips, or 0 otherwise.

```
trip_data$medical_trips_response <- ifelse(trip_data$WHYTO == 18, 1, 0)
trip_data$medical_trips_response <- as.integer(trip_data$medical_trips_response)

table(trip_data$medical_trips_response)
```

```
##
##      0      1
## 906788  16784
```

There is a severe class imbalance in the response variable, so we'll need to perform weighted logistic regression later.

Calculating weights for weighted logistic regression

Due to the big class imbalance in the response variable (`trip_data$medical_trips_response`), we need to perform weighted logistic regression since medical trips (what we're focusing on) are very rare in the dataset

```
# Compute class proportions
p1 <- sum(trip_data$medical_trips_response == 1) / nrow(trip_data)
p0 <- sum(trip_data$medical_trips_response == 0) / nrow(trip_data)

# Assign weights inversely proportional to class size
trip_data$weights <- ifelse(trip_data$medical_trips_response == 1,
                            1 / p1,  # Weight for medical trips (higher if rare)
                            1 / p0)  # Weight for non-medical trips
```

Creating binary predictors for age groups

Create two binary predictors: 1. whether a person is ≥ 65 years old, 2. whether a person is < 18 years old.

```
person_data$is_65_older <- ifelse(person_data$R_AGE >= 65, 1, 0)
table(person_data$is_65_older)
```

```
##
##      0      1
## 190808  73426
```

```
person_data$is_18_younger <- ifelse(person_data$R_AGE < 18, 1, 0)
table(person_data$is_18_younger)
```

```
##
##      0      1
## 230592  33642
```

Merge trip data with person data

```
# Left join trip data with person data -> keep all records from trip data, find their matches with person data
# Merge on the unique (HOUSEID, PERSONID) pairs -> merge on two columns

trip_data_merged <- merge(x = trip_data, y = person_data, by = c("HOUSEID", "PERSONID"), all.x = TRUE)
```

Clean the merged data for duplicate columns .x and .y

```
# Function to check if .x and .y versions are identical
check_identical_columns <- function(df) {
  cols_x <- grep(".x$", names(df), value = TRUE)  # Get .x columns
  cols_y <- sub(".x$", ".y", cols_x)  # Corresponding .y columns

  identical_cols <- c()  # To store columns where .x and .y are identical
  non_identical_cols <- c()  # To store columns where .x and .y are different

  for (i in seq_along(cols_x)) {
    col_x <- df[[cols_x[i]]]
    col_y <- df[[cols_y[i]]]

    # Check if they are identical (ignoring NAs)
    if (all(col_x == col_y, na.rm = TRUE)) {
      identical_cols <- c(identical_cols, cols_x[i])  # Store identical ones
    } else {
      non_identical_cols <- c(non_identical_cols, cols_x[i])  # Store non-identical ones
    }
  }

  list(identical = identical_cols, non_identical = non_identical_cols)
}

cat("Number of columns in trip_data_merged before cleaning:", ncol(trip_data_merged), "\n")
```

```
## Number of columns in trip_data_merged before cleaning: 238
```

```
# Check which columns are identical
col_check <- check_identical_columns(trip_data_merged)

# Process only if columns are identical
trip_data_merged <- trip_data_merged %>%
  # Keep .x version if .x and .y are identical
  mutate(across(all_of(col_check$identical), ~ coalesce(.x, get(sub(".x$", ".y", cur_column()))))) %>%
  # Drop the corresponding .y columns
  dplyr::select(-all_of(sub(".x$", ".y", col_check$identical))) %>%
  # Rename .x columns by removing .x suffix
  rename_with(~ sub(".x$", "", .), all_of(col_check$identical))

# Output non-identical columns for manual review
if (length(col_check$non_identical) > 0) {
  cat("Warning: These columns have different .x and .y values and were not merged:\n")
  print(sub(".x$", "", col_check$non_identical))
}

cat("Number of columns in trip_data_merged after cleaning:", ncol(trip_data_merged))
```

```
## Number of columns in trip_data_merged after cleaning: 203
```

Transforming the household race variable

As shown below, white people make up 83% of the sample → we should convert the household race variable into a binary variable (yes white, or not white); I did this on the P2 assignment

```
# Percentage of each category in the HH_RACE variable
table(trip_data_merged$HH_RACE) / nrow(trip_data_merged) * 100
```

```
## 
##          -8          -7           1           2           3           4 
##  0.08748641  0.42909486 83.62120116  6.67224645  3.92151343  0.58955880 
##           5           6          97 
##  0.17020871  2.90816525  1.60052492
```

```
# 1 if white, 0 otherwise
trip_data_merged$HH_RACE_new <- ifelse(trip_data_merged$HH_RACE == 1, 1, 0)

table(trip_data_merged$HH_RACE_new)
```

```
## 
##      0      1 
## 151270 772302
```

Transforming the household income variable

I used the sample household income categorization that was used in P2 in the code that was given to us

```
trip_data_merged <- trip_data_merged |>
  mutate(HHFAMINC = as.numeric(HHFAMINC)) |>
  filter(HHFAMINC > 0) |>
  mutate(HHFAMINC_new = case_when(HHFAMINC < 4 ~ "low",
                             HHFAMINC < 5 & HHSIZE > 1 ~ "low",
                             HHFAMINC < 6 & HHSIZE > 3 ~ "low",
                             HHFAMINC < 7 & HHSIZE > 5 ~ "low",
                             HHFAMINC < 8 & HHSIZE > 7 ~ "low",
                             HHFAMINC > 8 ~ "high",
                            TRUE ~ "medium")) |>
    mutate(HHFAMINC_new = factor(HHFAMINC_new, levels = c("medium", "low", "high")))

table(trip_data_merged$HHFAMINC_new)
```

```
## 
## medium    low   high 
## 501986 194453 202793
```

Convert all categorical variables (that we’ll use in the model) to factors

```
categorical_variables <- c("TRPTRANS", "PUBTRANS", "URBRUR", "EDUC", "R_SEX", "PHYACT", "CONDTRAV", "CONDRIDE",
"CONDNIGH", "CONDRIVE", "CONDPUB", "CONDSPEC", "CONDTAX", "is_65_older", "is_18_younger", "HH_RACE_new")

# Filter out all observations with negative values (NAs) in the categorical variable columns
trip_data_merged <- trip_data_merged %>%
  filter(if_all(all_of(categorical_variables), ~ . >= 0))

# Convert categorical variables to factors
trip_data_merged[categorical_variables] <- lapply(trip_data_merged[categorical_variables], as.factor)
```

Cleaning the medical device variables

We want to convert all positive values to 1 (uses the medical device) and all other values to 0 (doesn’t use the medical device)

```
medical_device_variables <- c("W_CANE", "W_WLKR", "W_WHCANE", "W_DOG", "W_CRUTCH",
                         "W_SCOOTR", "W_CHAIR", "W_MTRCHR")

# Convert all positive values to 1, all others to 0
trip_data_merged <- trip_data_merged %>%
  mutate(across(all_of(medical_device_variables), ~ ifelse(. > 0, 1, 0))) %>%
  mutate(across(all_of(medical_device_variables), as.factor))
```

Removing negative values (NAs) from numeric variables

```
numeric_variables <- c("TRVLCMIN", "DWELTIME", "R_AGE")

# Filter out all observations with negative values (NAs) in the numeric variable columns
trip_data_merged <- trip_data_merged %>%
  filter(if_all(all_of(numeric_variables), ~ . >= 0))
```

Collapsing the TRPTRANS variable

There are too many levels (20), and some don’t contain many observations

Categories I’m using:

- Personal vehicle: car, SUV, van, pickup truck, motorcycle
- Walk/bike
- Public transit: school bus, public or commuter bus, city to city bus, Amtrak commuter rail, subway
- Other

```
table(trip_data_merged$TRPTRANS) / nrow(trip_data_merged) * 100
```

```
## 
##           1           2           3           4           5           6 
##  8.352144470  0.481885276 49.434064966 20.211965484  7.745385268  9.720954805 
##           7           8           9          10          11          12 
##  0.161695724  0.123272978  0.132878664  0.142484351  1.460064358  0.385828410 
##          13          14          15          16          17          18 
##  0.267358276  0.036821799  0.051230328  0.139282455  0.337799978  0.145686246 
##          19          20          97 
##  0.100859709  0.008004739  0.560331716
```

```
trip_data_merged$TRPTRANS_new <- ifelse(trip_data_merged$TRPTRANS %in% c(3, 4, 5, 6, 8), "personal_vehicle",
                               ifelse(trip_data_merged$TRPTRANS %in% c(1, 2), "walk_bike",
                               ifelse(trip_data_merged$TRPTRANS %in% c(10, 11, 14, 15, 16), "public_transit",
                               "other")))

trip_data_merged$TRPTRANS_new <- as.factor(trip_data_merged$TRPTRANS_new)
table(trip_data_merged$TRPTRANS_new)
```

```
## 
##            other personal_vehicle   public_transit        walk_bike 
##             1312            54490             1143             5518
```

Making sure medical condition variables are binary

```
# 1 = yes, 0 = no
trip_data_merged$CONDTRAV <- ifelse(trip_data_merged$CONDTRAV == 2, 0, trip_data_merged$CONDTRAV)
trip_data_merged$CONDRIDE <- ifelse(trip_data_merged$CONDRIDE == 2, 0, trip_data_merged$CONDRIDE)
trip_data_merged$CONDNIGH <- ifelse(trip_data_merged$CONDNIGH == 2, 0, trip_data_merged$CONDNIGH)
trip_data_merged$CONDPUB <- ifelse(trip_data_merged$CONDPUB == 2, 0, trip_data_merged$CONDPUB)
trip_data_merged$CONDSPEC <- ifelse(trip_data_merged$CONDSPEC == 2, 0, trip_data_merged$CONDSPEC)
```

3. Modeling

Model 1: weighted logistic regression with quadratic age (WITHOUT binary age group predictors)

Predictors to add:

- Trip characteristics
    - TRVLCMIN: trip duration in minutes
    - TRPMILES: travel distance in miles
    - TRPTRANS: transportation mode
    - PUBTRANS: public transportation used on the trip
    - DWELTIME: time at the destination
- Demographic characteristics
    - HHSIZE: household size
    - HHVEHCNT: household vehicle count
    - HHFAMINC: household income
    - WRKCOUNT: number of workers in the household
    - URBRUR: is the household urban or rural?
    - HH_RACE: household race
    - EDUC: education of the respondent
    - R_SEX: sex of the respondent
    - PHYACT: level of physical activity
    - Medical devices used: W_CANE, W_WLKR, W_WHCANE, W_DOG, W_CRUTCH, W_SCOOTR, W_CHAIR, W_MTRCHR
    - Medical condition results in: CONDTRAV, CONDRIDE, CONDNIGH, CONDRIVE, CONDPUB, CONDSPEC, CONDTAX

```
# eval = FALSE for all previous models before my final model
# Fit the model
model1 <- glm(medical_trips_response ~ R_AGE + I(R_AGE^2) + TRVLCMIN + TRPMILES + TRPTRANS + DWELTIME + HHSIZE +
HHVEHCNT + HHFAMINC_new + WRKCOUNT + URBRUR + HH_RACE_new + EDUC + R_SEX + PHYACT + W_CANE + W_WLKR + W_WHCANE +
W_DOG + W_CRUTCH + W_SCOOTR + W_CHAIR + W_MTRCHR + CONDTRAV + CONDRIDE + CONDNIGH + CONDRIVE + CONDPUB + CONDSPEC
+ CONDTAX,
                      family = binomial(link = "logit"),
                      data = trip_data_merged,
                      weights = trip_data_merged$weights)

summary(model1)
```

Now, we need to adjust the cutoff from the default threshold of 0.5 due to the class imbalance → this cutoff is causing the model accuracy to be very low, low sensitivity, high specificity

We need to find the optimal cutoff using the ROC curve and Youden's index (the point of maximizing sensitivity and specificity)

```
# Evaluate the model

# Make predictions
trip_data_merged$predicted_probs <- predict(model1, type = "response")

# ROC curve
roc_curve <- roc(trip_data_merged$medical_trips_response, trip_data_merged$predicted_probs)

# Plot the ROC curve
plot(roc_curve, main = "ROC curve for model 1", col = "blue")

# Compute AUC
auc(roc_curve)

# Compute best threshold using Youden's Index
best_threshold <- coords(roc_curve, "best", ret = "threshold")

# Convert to binary predictions using a threshold of best_threshold
trip_data_merged$predicted_class <- ifelse(trip_data_merged$predicted_probs > as.numeric(best_threshold), 1, 0)

# Create confusion matrix
confusionMatrix(as.factor(trip_data_merged$predicted_class), as.factor(trip_data_merged$medical_trips_response))
```

Model 2: weighted logistic regression with quadratic age (WITH binary age group predictors)

Predictors to add: `person_data$is_65_older` and `person_data$is_18_younger`

```
# eval = FALSE for all previous models before my final model
# Fit the model
model2 <- glm(medical_trips_response ~ R_AGE + I(R_AGE^2) + TRVLCMIN + TRPMILES + TRPTRANS + DWELTIME + HHSIZE +
HHVEHCNT + HHFAMINC_new + WRKCOUNT + URBRUR + HH_RACE_new + EDUC + R_SEX + R_AGE_IMP + R_SEX_IMP + PHYACT + W_CAN
E + W_WLKR + W_WHCANE + W_DOG + W_CRUTCH + W_SCOOTR + W_CHAIR + W_MTRCHR + CONDTRAV + CONDRIDE + CONDNIGH + CONDR
IVE + CONDPUB + CONDSPEC + CONDTAX + is_65_older + is_18_younger,
                      family = binomial(link = "logit"),
                      data = trip_data_merged,
                      weights = trip_data_merged$weights)

summary(model2)
```

```
# Evaluate the model

# Make predictions
trip_data_merged$predicted_probs <- predict(model2, type = "response")

# ROC curve
roc_curve <- roc(trip_data_merged$medical_trips_response, trip_data_merged$predicted_probs)

# Plot the ROC curve
plot(roc_curve, main = "ROC curve for model 2", col = "blue")

# Compute AUC
auc(roc_curve)

# Compute best threshold using Youden’s Index
best_threshold <- coords(roc_curve, "best", ret = "threshold")

# Convert to binary predictions using a threshold of best_threshold
trip_data_merged$predicted_class <- ifelse(trip_data_merged$predicted_probs > as.numeric(best_threshold), 1, 0)

# Create confusion matrix
confusionMatrix(as.factor(trip_data_merged$predicted_class), as.factor(trip_data_merged$medical_trips_response))
```

Analysis of model performance from models 1 and 2

Next, I will use stepwise feature selection to find the best combination of features. I hand chose all the predictors based on what I thought would be best predictors of medical visits, but I now want to use AIC to determine which variables provide the most predictive value.

Model 3: applying stepwise feature selection to model 2

```
# Setting eval = FALSE so we don't run it every time we knit the file
model3 <- stepAIC(model2, direction = "both")
```

```
# eval = FALSE for all previous models before my final model
model3 <- glm(formula = medical_trips_response ~ I(R_AGE^2) + TRVLCMIN +
    TRPTRANS + DWELTIME + HHSIZE + HHFAMINC_new + WRKCOUNT +
    URBRUR + HH_RACE_new + EDUC + R_SEX + R_AGE_IMP + PHYACT +
    W_CANE + W_WLKR + W_CRUTCH + W_SCOOTR + W_CHAIR + CONDTRAV +
    CONDRIDE + CONDNIGH + CONDRIVE + CONDPUB + CONDSPEC, family = binomial(link = "logit"),
    data = trip_data_merged, weights = trip_data_merged$weights)

summary(model3)
```

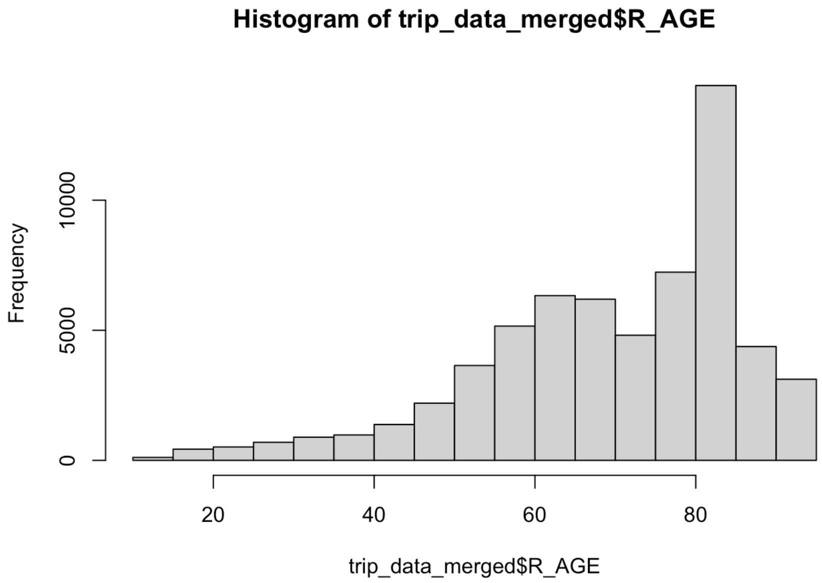
```
# eval = FALSE for all previous models before my final model
# Evaluate the model

# Make predictions
trip_data_merged$predicted_probs <- predict(model3, type = "response")

# ROC curve
roc_curve <- roc(trip_data_merged$medical_trips_response, trip_data_merged$predicted_probs)

# Plot the ROC curve
plot(roc_curve, main = "ROC curve for model 3", col = "blue")

# Compute AUC
auc(roc_curve)

# Compute best threshold using Youden’s Index
best_threshold <- coords(roc_curve, "best", ret = "threshold")

# Convert to binary predictions using a threshold of best_threshold
trip_data_merged$predicted_class <- ifelse(trip_data_merged$predicted_probs > as.numeric(best_threshold), 1, 0)

# Create confusion matrix
confusionMatrix(as.factor(trip_data_merged$predicted_class), as.factor(trip_data_merged$medical_trips_response))
```

Model 4: Weighted Generalized Additive Model (GAM)

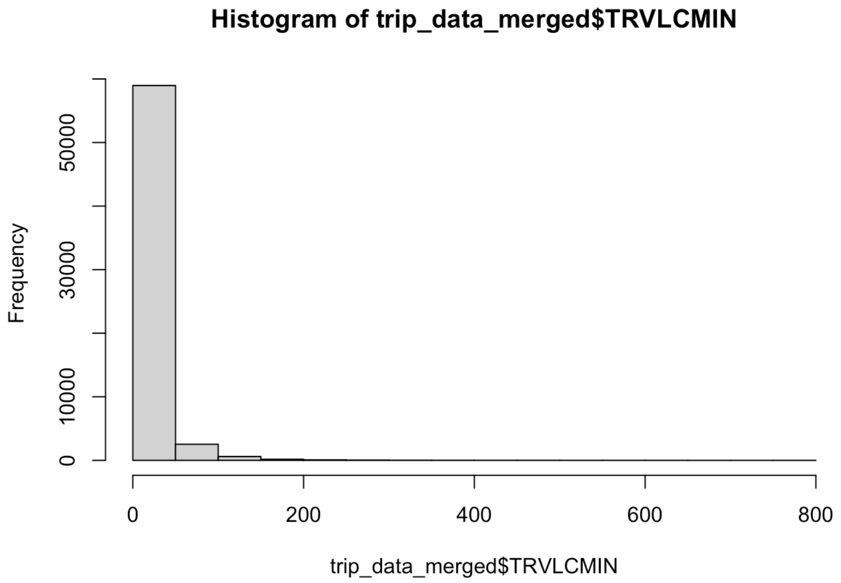
GAM model allows for smooth relationships in variables that we think may be nonlinear

We’ll want to transform the numeric variables appropriately

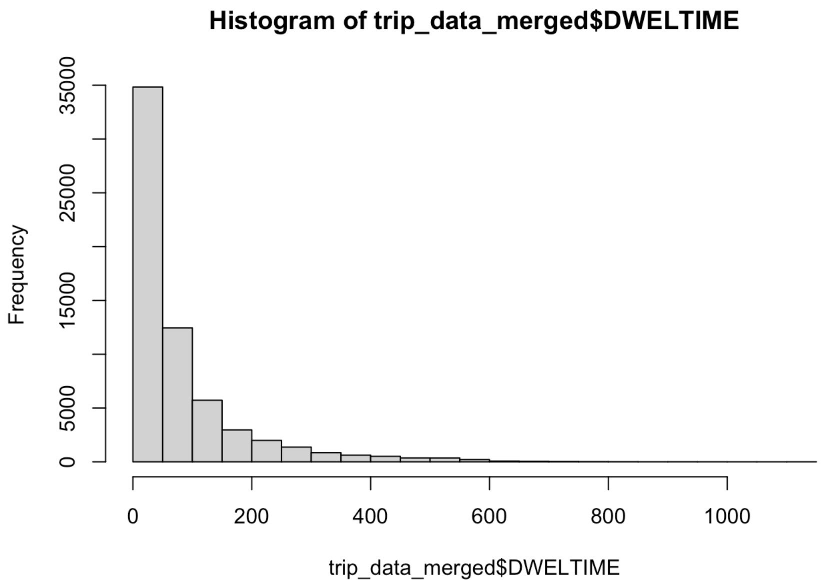
```
# Plotting numeric variables to look at their distributions

hist(trip_data_merged$R_AGE)
```



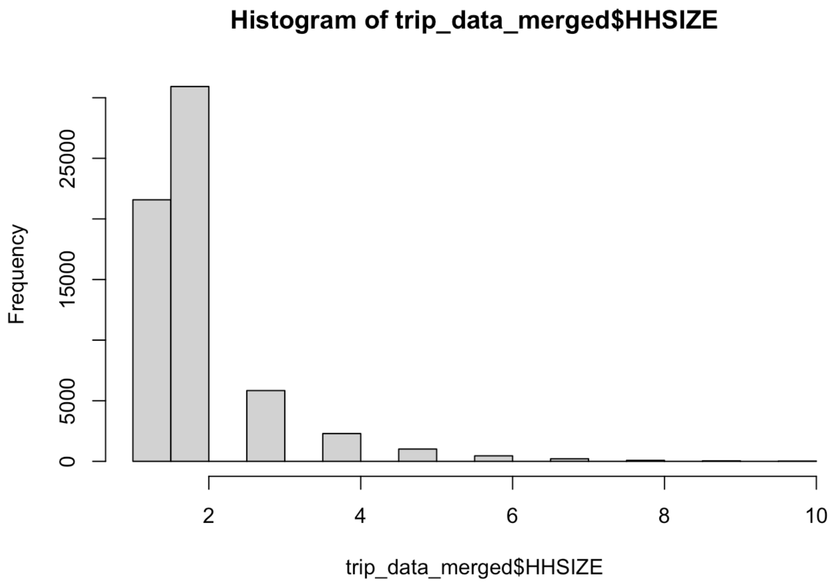
```
hist(trip_data_merged$TRVLCMIN)
```



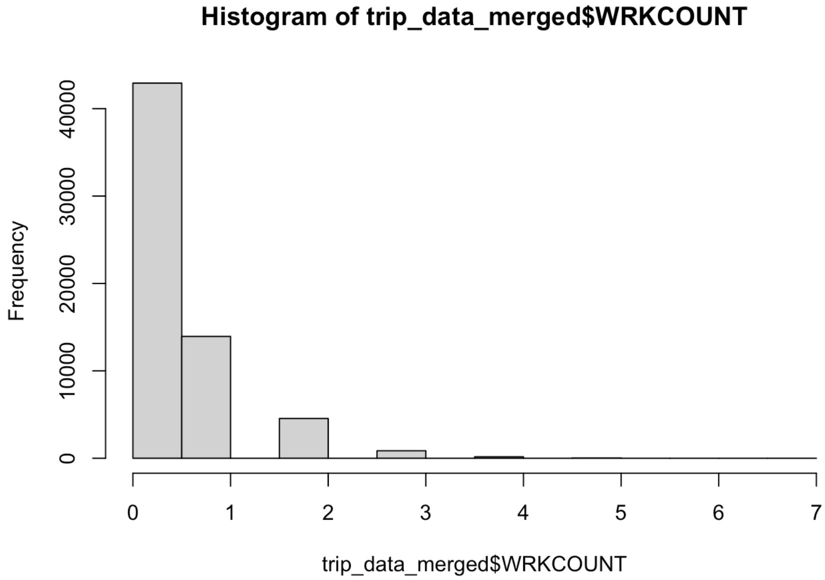
```
hist(trip_data_merged$DWELTIME)
```



```
hist(trip_data_merged$HHSIZE)
```



```
hist(trip_data_merged$WRKCOUNT)
```



As we can see, `TRVLCMIN` , `DWELTIME` , `HHSIZE` , `WRKCOUNT` are all right skewed → cubic spline transform `TRVLCMIN` and `DWELTIME` . But, we should leave `HHSIZE` alone `WRKCOUNT` since they're discrete counts

`R_AGE` is left skewed → model as nonlinear U-shaped

```
# eval = FALSE for all previous models before my final model

model4 <- gam(medical_trips_response ~ s(R_AGE) + s(TRVLCMIN, bs = "cs") + s(DWELTIME, bs = "cs") + TRPTRANS + HH
SIZE + HHFAMINC_new + WRKCOUNT + URBRUR + HH_RACE_new + EDUC + R_SEX + PHYACT + W_CANE + W_WLKR + W_CRUTCH + W_SC
OOTR + W_CHAIR + CONDTRAV + CONDRIDE + CONDNIGH + CONDRIVE + CONDPUB + CONDSPEC, family = binomial(link = "logi
t"), data = trip_data_merged, weights = trip_data_merged$weights)

summary(model4)
```

```
# eval = FALSE for all previous models before my final model
# Evaluate the model

# Make predictions
trip_data_merged$predicted_probs <- predict(model4, type = "response")

# ROC curve
roc_curve <- roc(trip_data_merged$medical_trips_response, trip_data_merged$predicted_probs)

# Plot the ROC curve
plot(roc_curve, main = "ROC curve for model 4", col = "blue")

# Compute AUC
auc(roc_curve)

# Compute best threshold using Youden's Index
best_threshold <- coords(roc_curve, "best", ret = "threshold")

# Convert to binary predictions using a threshold of best_threshold
trip_data_merged$predicted_class <- ifelse(trip_data_merged$predicted_probs > as.numeric(best_threshold), 1, 0)

# Create confusion matrix
confusionMatrix(as.factor(trip_data_merged$predicted_class), as.factor(trip_data_merged$medical_trips_response))
```

It seems like the GAM model is definitely better than logistic regression, but we should still do more to address the big class imbalance in the response variable (there are not many medical trips).

Model 5: model 4 + SMOTE resampling

SMOTE resampling oversamples the minority class by creating synthetic data

```
# eval = FALSE for all previous models before my final model

# Separate features (X) and target variable (y)
model_variables <- c("R_AGE", "TRVLCMIN", "DWELTIME", "TRPTRANS", "HHSIZE",
                "HHFAMINC_new", "WRKCOUNT", "URBRUR", "HH_RACE_new", "EDUC",
                "R_SEX", "PHYACT", "W_CANE", "W_WLKR", "W_CRUTCH", "W_SCOOTR",
                "W_CHAIR", "CONDTRAV", "CONDRIDE", "CONDNIGH", "CONDRIVE",
                "CONDPUB", "CONDSPEC")
X <- trip_data_merged[, model_variables]
y <- trip_data_merged$medical_trips_response

# Clean X and y so they can be passed through the SMOTE function
X <- X %>%
  mutate(across(where(is.factor), as.numeric))  # Convert all factors to numeric
y <- as.numeric(as.character(y))

# Apply SMOTE (K = 5 means nearest neighbors, dup_size controls oversampling)
smote_result <- SMOTE(X, y, K = 5, dup_size = 5)

# Extract balanced dataset
trip_data_balanced <- smote_result$data

# Rename target variable back to "medical_trips_response" and convert it to a factor
trip_data_balanced$medical_trips_response <- as.factor(trip_data_balanced$class)
trip_data_balanced$class <- NULL  # Remove the generated class column
```

```
# eval = FALSE for all previous models before my final model
# Refit the model with balanced data
model5 <- gam(medical_trips_response ~
                       s(R_AGE) + s(TRVLCMIN, bs = "cs") + s(DWELTIME, bs = "cs") +
                       TRPTRANS + HHSIZE + HHFAMINC_new + WRKCOUNT +
                       URBRUR + HH_RACE_new + EDUC + R_SEX + PHYACT +
                       W_CANE + W_WLKR + W_CRUTCH + W_SCOOTR +
                       W_CHAIR + CONDTRAV + CONDRIDE + CONDNIGH +
                       CONDRIVE + CONDPUB + CONDSPEC,
                       family = binomial(link = "logit"),
                       data = trip_data_balanced)

summary(model5)
```

```
# eval = FALSE for all previous models before my final model

# Make sure data types are compatible between trip_data_balanced and trip_data_merged

trip_data_merged_copy <- data.frame(trip_data_merged, stringsAsFactors = FALSE) # Make a copy
categorical_variables <- c("TRPTRANS", "HHFAMINC_new", "URBRUR", "HH_RACE_new",
                      "EDUC", "R_SEX", "PHYACT", "W_CANE", "W_WLKR",
                      "W_CRUTCH", "W_SCOOTR", "W_CHAIR", "CONDTRAV",
                      "CONDRIDE", "CONDNIGH", "CONDRIVE", "CONDPUB", "CONDSPEC")

trip_data_merged_copy <- trip_data_merged_copy %>%
  mutate(across(all_of(categorical_variables), as.numeric))
```

```
# eval = FALSE for all previous models before my final model
# Evaluate the model (on trip_data_merged, not trip_data_balanced)

# Make predictions
trip_data_merged_copy$predicted_probs <- predict(model5, newdata = trip_data_merged_copy, type = "response")

# ROC curve
roc_curve <- roc(trip_data_merged_copy$medical_trips_response, trip_data_merged_copy$predicted_probs)

# Plot the ROC curve
plot(roc_curve, main = "ROC curve for model 5", col = "blue")

# Compute AUC
auc(roc_curve)

# Compute best threshold using Youden’s Index
best_threshold <- coords(roc_curve, "best", ret = "threshold")

# Convert to binary predictions using a threshold of best_threshold
trip_data_merged_copy$predicted_class <- ifelse(trip_data_merged_copy$predicted_probs > as.numeric(best_threshol
d), 1, 0)

# Create confusion matrix
confusionMatrix(as.factor(trip_data_merged_copy$predicted_class), as.factor(trip_data_merged_copy$medical_trips_r
esponse))
```

I tried model 5 with the SMOTE parameter `dup_size = 5` vs. `dup_size = 10`, and `dup_size = 5` performed slightly better.

Since the model didn’t improve much after SMOTE resampling, next I will try downsampling

Model 6: model 4 + downsampling

Downsampling downsamples the majority class (non-medical trips)

```
model_variables <- c("R_AGE", "TRVLCMIN", "DWELTIME", "TRPTRANS_new", "HHSIZE",
                "HHFAMINC_new", "WRKCOUNT", "URBRUR", "HH_RACE_new", "EDUC",
                "R_SEX", "PHYACT", "W_CANE", "W_WLKR", "W_CRUTCH", "W_SCOOTR",
                "W_CHAIR", "CONDTRAV", "CONDRIDE", "CONDNIGH", "CONDRIVE",
                "CONDPUB", "CONDSPEC", "medical_trips_response")

# Only use the variables that are in the model
trip_data_merged_copy <- data.frame(trip_data_merged, stringsAsFactors = FALSE) # Make a copy
trip_data_merged_copy <- trip_data_merged_copy[, model_variables]

# Define categorical variables
categorical_vars <- c("TRPTRANS_new", "HHFAMINC_new", "URBRUR", "HH_RACE_new",
                      "EDUC", "R_SEX", "PHYACT", "W_CANE", "W_WLKR",
                      "W_CRUTCH", "W_SCOOTR", "W_CHAIR", "CONDTRAV",
                      "CONDRIDE", "CONDNIGH", "CONDRIVE", "CONDPUB", "CONDSPEC")

# Convert categorical variables to factors in the original dataset
for (var in categorical_vars) {
  trip_data_merged_copy[[var]] <- as.factor(trip_data_merged_copy[[var]])
}

# Perform downsampling
set.seed(100)
trip_data_downsampled <- trip_data_merged_copy %>%
  group_by(medical_trips_response) %>%
  sample_n(min(table(trip_data_merged_copy$medical_trips_response)), replace = FALSE) %>%
  ungroup()

# Ensure all factor levels are preserved in downsampled dataset
for (var in categorical_vars) {
  trip_data_downsampled[[var]] <- factor(trip_data_downsampled[[var]], levels = levels(trip_data_merged_copy[[va
r]]))
}
```

```r
# eval = FALSE for all previous models before my final model

# Refit the model with balanced data
model6 <- gam(medical_trips_response ~
                       s(R_AGE) + s(TRVLCMIN, bs = "cs") + s(DWELTIME, bs = "cs") +
                       TRPTRANS + HHSIZE + HHFAMINC_new + WRKCOUNT +
                       URBRUR + HH_RACE_new + EDUC + R_SEX + PHYACT +
                       W_CANE + W_WLKR + W_CRUTCH + W_SCOOTR +
                       W_CHAIR + CONDTRAV + CONDRIDE + CONDNIGH +
                       CONDRIVE + CONDPUB + CONDSPEC,
                       family = binomial(link = "logit"),
                       data = trip_data_downsampled)

summary(model6)
```

```r
# Make sure trip_data_merged and trip_data_downsampled are compatible
# In this case, there are factor categories in trip_data_merged that were not seen in trip_data_downsampled, which the model was trained on

# Ensure factor levels in prediction dataset match those in training
for (var in categorical_vars) {
  trip_data_merged_copy[[var]] <- factor(trip_data_merged_copy[[var]], levels = levels(trip_data_downsampled[[var]]))
}

# Find rows where any categorical variable has an unseen level
rows_with_unseen_levels <- apply(trip_data_merged_copy[categorical_vars], 1, function(row) {
  any(is.na(row))  # NA values indicate unseen factor levels
})

# Remove these rows before making predictions
trip_data_merged_copy <- trip_data_merged_copy[!rows_with_unseen_levels, ]

# Explicitly Reset Factor Levels Again
for (var in categorical_vars) {
  trip_data_merged_copy[[var]] <- factor(trip_data_merged_copy[[var]], levels = levels(trip_data_downsampled[[var]]))
}

# Drop unused levels
trip_data_merged_copy <- droplevels(trip_data_merged_copy)
trip_data_downsampled <- droplevels(trip_data_downsampled)

# Verify that model matrices align
train_matrix <- model.matrix(~ ., data = trip_data_downsampled)
predict_matrix <- model.matrix(~ ., data = trip_data_merged_copy)

# Identify remaining extra/missing columns
extra_cols_matrix <- setdiff(colnames(predict_matrix), colnames(train_matrix))
missing_cols_matrix <- setdiff(colnames(train_matrix), colnames(predict_matrix))

# Print results
if (length(extra_cols_matrix) > 0) {
  message("Extra columns in prediction dataset (should be empty): ", paste(extra_cols_matrix, collapse=", "))
}
if (length(missing_cols_matrix) > 0) {
  message("Missing columns in prediction dataset (should be empty): ", paste(missing_cols_matrix, collapse=", "))
}

# Final check
if (ncol(train_matrix) != ncol(predict_matrix)) {
  stop("Mismatch in model matrices: Training has ", ncol(train_matrix),
       " columns, but Prediction has ", ncol(predict_matrix), " columns.")
} else {
  message("Model matrices are now aligned")
}
```

```
# eval = FALSE for all previous models before my final model
# Evaluate the model (on trip_data_merged, not trip_data_balanced)

# Make predictions
trip_data_merged_copy$predicted_probs <- predict(model6, newdata = trip_data_merged_copy, type = "response")

# ROC curve
roc_curve <- roc(trip_data_merged_copy$medical_trips_response, trip_data_merged_copy$predicted_probs)

# Plot the ROC curve
plot(roc_curve, main = "ROC curve for model 6", col = "blue")

# Compute AUC
auc(roc_curve)

# Compute best threshold using Youden's Index
best_threshold <- coords(roc_curve, "best", ret = "threshold")

# Convert to binary predictions using a threshold of best_threshold
trip_data_merged_copy$predicted_class <- ifelse(trip_data_merged_copy$predicted_probs > as.numeric(best_threshold), 1, 0)

# Create confusion matrix
confusionMatrix(as.factor(trip_data_merged_copy$predicted_class), as.factor(trip_data_merged_copy$medical_trips_response))
```

Comparison of model (weighted GAM) 4 vs. model 5 (SMOTE) vs. model 6 (downsampling)

Model	AUC	Accuracy	Sensitivity	Specificity	Balanced Accuracy	Kappa
Model 4	0.7809	0.6915	0.6885	0.7355	0.7120	0.1404
Model 5 (SMOTE)	0.7731	0.7017	0.7009	0.7145	0.7077	0.142
Model 6 (Downsampled)	0.7808	0.7206	0.7216	0.7062	0.7139	0.1539

Model 6 is the best model

Model 7: building off of model 6, but add interaction terms

Interactions to try

- `TRPTRANS * URBRUR` → different transportation modes are used depending on if the area is urban or rural
- `PHYACT * W_CHAIR`, `PHYACT * W_WLKR`, `PHYACT * W_CANE`, `PHYACT * W_CRUTCH`, `PHYACT * W_SCOOTR` → people with medical devices are likely to perform less physical activity
- `HHFAMINC_new * EDUC` → household income and education level are often related
- `WRKCOUNT * HHSIZE` → the number of working people in a household may influence trip frequency
- `R_AGE * DWELTIME` → older people may spend longer at the hospital (the destination in this case)
- `TRVLCMIN * TRPTRANS` → transportation mode influences total travel time

After trying the physical activity and medical device interactions, I noticed none of them are very significant, and the model didn't improve much from model 6. So, I ended up removing them. Specifically this part: `ti(PHYACT, by = W_CANE, k = 3) + ti(PHYACT, by = W_WLKR, k = 3) + ti(PHYACT, by = W_CRUTCH, k = 3) + ti(PHYACT, by = W_SCOOTR, k = 3)`

I also removed the interaction between household income and education level `ti(HHFAMINC_new, EDUC, k = c(3, 5))` and the interaction between number of workers and household size `ti(WRKCOUNT,HHSIZE)` since they didn't seem to significantly contribute either

I couldn't actually do any of the interactions that involved unordered factors (ex. `TRPTRANS` transportation mode)

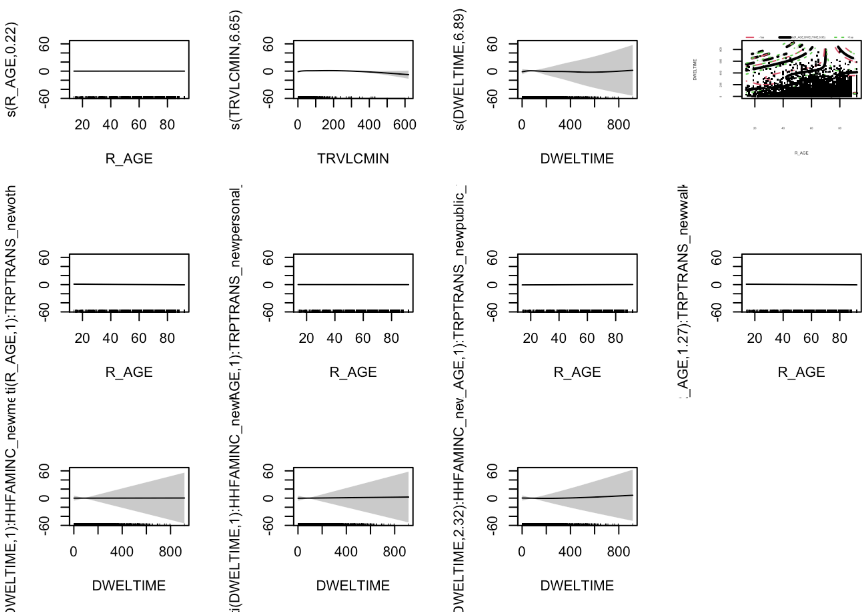
I implemented the new TRPTRANS variable collapsed this time since there were too many levels previously with not many observations

```
# Ensure data is the correct format for interaction terms
trip_data_downsampled$TRPTRANS_new <- as.factor(trip_data_downsampled$TRPTRANS_new)
trip_data_downsampled$URBRUR <- as.factor(trip_data_downsampled$URBRUR)
trip_data_downsampled$PHYACT <- as.factor(trip_data_downsampled$PHYACT)
trip_data_downsampled$HHFAMINC_new <- as.factor(trip_data_downsampled$HHFAMINC_new)
trip_data_downsampled$EDUC <- as.factor(trip_data_downsampled$EDUC)
```

```
# Use the by = inside the interaction terms for binary variables
# Manually specify k when a categorical variable has fewer than 5 levels

model7 <- gam(medical_trips_response ~ s(R_AGE, bs = "cs") + s(TRVLCMIN, bs = "cs") + s(DWELTIME, bs = "cs") + TR
PTRANS_new + HHSIZE + HHFAMINC_new + WRKCOUNT + URBRUR + HH_RACE_new + EDUC + R_SEX + PHYACT + W_CANE + W_WLKR +
W_CRUTCH + W_SCOOTR + W_CHAIR + CONDTRAV + CONDRIDE + CONDNIGH + CONDRIVE + CONDPUB + CONDSPEC  + ti(R_AGE, DWELT
IME) + ti(R_AGE, by = TRPTRANS_new) + ti(DWELTIME, by = HHFAMINC_new), family = binomial(link = "logit"), data =
trip_data_downsampled)

summary(model7)
```

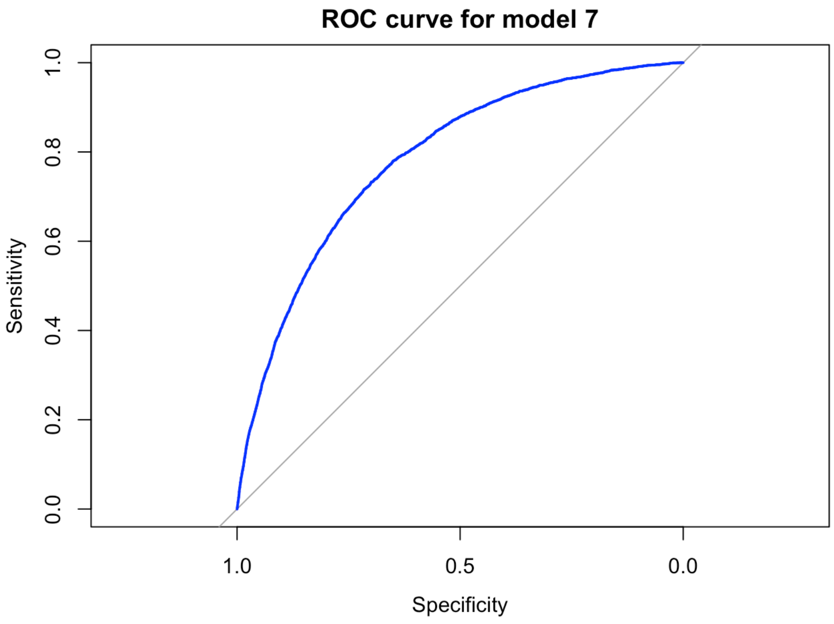
```
## 
## Family: binomial 
## Link function: logit 
## 
## Formula:
## medical_trips_response ~ s(R_AGE, bs = "cs") + s(TRVLCMIN, bs = "cs") + 
##     s(DWELTIME, bs = "cs") + TRPTRANS_new + HHSIZE + HHFAMINC_new + 
##     WRKCOUNT + URBRUR + HH_RACE_new + EDUC + R_SEX + PHYACT + 
##     W_CANE + W_WLKR + W_CRUTCH + W_SCOOTR + W_CHAIR + CONDTRAV + 
##     CONDRIDE + CONDNIGH + CONDRIVE + CONDPUB + CONDSPEC + ti(R_AGE, 
##     DWELTIME) + ti(R_AGE, by = TRPTRANS_new) + ti(DWELTIME, by = HHFAMINC_new)
## 
## Parametric coefficients:
##                              Estimate Std. Error z value Pr(>|z|)    
## (Intercept)                   0.553304   0.224531   2.464 0.013729 *  
## TRPTRANS_newpersonal_vehicle -0.654546   0.161611  -4.050 5.12e-05 ***
## TRPTRANS_newpublic_transit   -0.554947   0.245714  -2.259 0.023914 *  
## TRPTRANS_newwalk_bike        -2.228568   0.216225 -10.307  < 2e-16 ***
## HHSIZE                        0.127636   0.031766   4.018 5.87e-05 ***
## HHFAMINC_newlow               0.012967   0.059424   0.218 0.827263    
## HHFAMINC_newhigh              0.197685   0.110521   1.789 0.073668 .  
## WRKCOUNT                     -0.281095   0.047486  -5.920 3.23e-09 ***
## URBRUR2                      -0.072316   0.063591  -1.137 0.255452    
## HH_RACE_new1                  0.054909   0.074250   0.740 0.459597    
## EDUC2                         0.113992   0.105630   1.079 0.280514    
## EDUC3                        -0.009459   0.105677  -0.090 0.928681    
## EDUC4                        -0.016275   0.116896  -0.139 0.889272    
## EDUC5                         0.032747   0.120192   0.272 0.785273    
## R_SEX2                       -0.046134   0.053384  -0.864 0.387483    
## PHYACT2                      -0.189075   0.070634  -2.677 0.007433 ** 
## PHYACT3                      -0.406049   0.108035  -3.758 0.000171 ***
## W_CANE1                       0.112162   0.061800   1.815 0.069538 .  
## W_WLKR1                       0.236373   0.078349   3.017 0.002554 ** 
## W_CRUTCH1                     0.105681   0.173112   0.610 0.541547    
## W_SCOOTR1                    -0.142248   0.132383  -1.075 0.282590    
## W_CHAIR1                      0.226684   0.120706   1.878 0.060383 .  
## CONDTRAV1                     0.256806   0.058271   4.407 1.05e-05 ***
## CONDRIDE1                     0.273795   0.060464   4.528 5.95e-06 ***
## CONDNIGH1                    -0.076607   0.059745  -1.282 0.199760    
## CONDRIVE2                    -0.232905   0.083912  -2.776 0.005510 ** 
## CONDPUB1                     -0.173546   0.110525  -1.570 0.116369    
## CONDSPEC1                    -0.046077   0.114920  -0.401 0.688458    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Approximate significance of smooth terms:
##                                         edf Ref.df  Chi.sq  p-value    
## s(R_AGE)                             0.2199  8.000   0.277   0.2543    
## s(TRVLCMIN)                          6.6523  9.000 390.615  < 2e-16 ***
## s(DWELTIME)                          6.8921  8.000 627.156  < 2e-16 ***
## ti(R_AGE,DWELTIME)                   6.9506  8.742  42.538 4.95e-08 ***
## ti(R_AGE):TRPTRANS_newother          1.0002  1.000   3.770   0.0522 .  
## ti(R_AGE):TRPTRANS_newpersonal_vehicle 1.0005  1.001   0.583   0.4452    
## ti(R_AGE):TRPTRANS_newpublic_transit   1.0011  1.002   1.170   0.2803    
## ti(R_AGE):TRPTRANS_newwalk_bike        1.2724  1.488   5.707   0.0235 *  
## ti(DWELTIME):HHFAMINC_newmedium        1.0002  1.000   0.000   0.9974    
## ti(DWELTIME):HHFAMINC_newlow           1.0014  1.003   0.007   0.9399    
## ti(DWELTIME):HHFAMINC_newhigh          2.3219  2.819   7.938   0.0298 *  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## R-sq.(adj) =  0.236   Deviance explained = 19.1%
## UBRE = 0.1365  Scale est. = 1         n = 7924
```

```
# Plot the nonlinear terms and interactions
plot(model7, pages = 1, shade = TRUE, rug = TRUE)
```



```
# Make sure trip_data_merged data is compatible with the transformations we made on trip_data_downsampled
trip_data_merged_copy$TRPTRANS_new <- as.factor(trip_data_merged_copy$TRPTRANS_new)
trip_data_merged_copy$URBRUR <- as.factor(trip_data_merged_copy$URBRUR)
trip_data_merged_copy$PHYACT <- as.factor(trip_data_merged_copy$PHYACT)
trip_data_merged_copy$HHFAMINC_new <- as.factor(trip_data_merged_copy$HHFAMINC_new)
trip_data_merged_copy$EDUC <- as.factor(trip_data_merged_copy$EDUC)
```

```
# Evaluate the model (on trip_data_merged, not trip_data_balanced)

# Make predictions
trip_data_merged_copy$predicted_probs <- predict(model7, newdata = trip_data_merged_copy, type = "response")

# ROC curve
roc_curve <- roc(trip_data_merged_copy$medical_trips_response, trip_data_merged_copy$predicted_probs)

# Plot the ROC curve
plot(roc_curve, main = "ROC curve for model 7", col = "blue")
```



```
# Compute AUC
auc(roc_curve)
```

```
## Area under the curve: 0.7833
```

```
# Compute best threshold using Youden’s Index
best_threshold <- coords(roc_curve, "best", ret = "threshold")

# Convert to binary predictions using a threshold of best_threshold
trip_data_merged_copy$predicted_class <- ifelse(trip_data_merged_copy$predicted_probs > as.numeric(best_threshol
d), 1, 0)

# Create confusion matrix
confusionMatrix(as.factor(trip_data_merged_copy$predicted_class), as.factor(trip_data_merged_copy$medical_trips_r
esponse))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 40909  1059
##          1 17592  2903
##
##                Accuracy : 0.7014
##                  95% CI : (0.6978, 0.705)
##     No Information Rate : 0.9366
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1467
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6993
##             Specificity : 0.7327
##          Pos Pred Value : 0.9748
##          Neg Pred Value : 0.1416
##              Prevalence : 0.9366
##          Detection Rate : 0.6549
##    Detection Prevalence : 0.6719
##       Balanced Accuracy : 0.7160
##
##        'Positive' Class : 0
##
```

Model 7 is still bad because the model, which was trained on downsampled data, doesn’t appear to generalize well to new, unseen, imbalanced data

I’m going to try weighted GAM again (but using a different weight calculation method) but using the improvements from model 7 (interactions and collapsed categorical variables)

Model 8: weighted GAM with model 7’s improvements

```
# eval = FALSE for all previous models before my final model
model8 <- gam(medical_trips_response ~ s(R_AGE, bs = "cs") + s(TRVLCMIN, bs = "cs") + s(DWELTIME, bs = "cs") + TR
PTRANS_new + HHSIZE + HHFAMINC_new + WRKCOUNT + URBRUR + HH_RACE_new + EDUC + R_SEX + PHYACT + W_CANE + W_WLKR +
W_CRUTCH + W_SCOOTR + W_CHAIR + CONDTRAV + CONDRIDE + CONDNIGH + CONDRIVE + CONDPUB + CONDSPEC  + ti(R_AGE, DWELT
IME) + ti(R_AGE, by = TRPTRANS_new) + ti(DWELTIME, by = HHFAMINC_new), family = binomial(link = "logit"), data =
trip_data_merged, weights = trip_data_merged$weights)

summary(model8)
```

```
# eval = FALSE for all previous models before my final model
# Evaluate the model

# Make predictions
trip_data_merged$predicted_probs <- predict(model8, type = "response")

# ROC curve
roc_curve <- roc(trip_data_merged$medical_trips_response, trip_data_merged$predicted_probs)

# Plot the ROC curve
plot(roc_curve, main = "ROC curve for model 8", col = "blue")

# Compute AUC
auc(roc_curve)

# Compute best threshold using Youden’s Index
best_threshold <- coords(roc_curve, "best", ret = "threshold")

# Convert to binary predictions using a threshold of best_threshold
trip_data_merged$predicted_class <- ifelse(trip_data_merged$predicted_probs > as.numeric(best_threshold), 1, 0)

# Create confusion matrix
confusionMatrix(as.factor(trip_data_merged$predicted_class), as.factor(trip_data_merged$medical_trips_response))
```

Model 8 performed very similarly to model 7, so I’m just going to keep model 7