CYO Project: Harvard PH125.9x Capstone

Ashley Reinhart

2024-04-15

Introduction

This project is to satisfy the Choose Your Own portion of the HarvardX: PH125.9x Data Science Capstone course. The overall aim of this project is to demonstrate the knowledge and skills obtained from the Data Science Professional Certificate series.

This author chose to analyse data from the National Poll on Healthy Aging's (NPHA) survey, as it directly relates to her career in the Senior Living Industry. The aim of the NPHA survey was to gather data to help inform providers, policy makers, and advocates of the various healthcare and health related issues affecting older adults. The original data set captured various responses on various topics such as health insurance, doctors visits, dental health, employment, and much more.

Objectives

The objectives in this analysis are to (1) test predictive models to determine which model provides the best accuracy in predicting the target variable Number.of.Doctors.Visited (2) identify the important features related to the Number.of.Doctors.Visited.

Key Steps

This analysis will cover data exploration, additional data cleaning, modeling approaches, results, and future work recommendations.

Examining the Data

The NPHA data was obtained from the UCI Machine Learning Repository. The downloaded CSV file, as well as the key file can be found in this author's GitHub repository.

The data was pre-processed and only includes 14 features and one target variable from the original NPHA data set. It was further pre-processed by removing any respondents with missing responses.

Loading Data

```
data_url <-
"https://raw.githubusercontent.com/anreinhart/edx-capstone/main/NPHA-doctor-visits.csv"
NPHA_Original <- read.csv(data_url)
```

```r
str(NPHA_Original)
```

```
## 'data.frame':    714 obs. of  15 variables:
##  $ Number.of.Doctors.Visited                 : int  3 2 3 1 3 2 3 2 2 1 ...
##  $ Age                                       : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ Phyiscal.Health                           : int  4 4 3 3 3 3 4 3 3 2 ...
##  $ Mental.Health                             : int  3 2 2 2 3 2 1 2 1 1 ...
##  $ Dental.Health                             : int  3 3 3 3 3 4 1 6 2 3 ...
##  $ Employment                                : int  3 3 3 3 3 3 3 3 3 1 ...
##  $ Stress.Keeps.Patient.from.Sleeping        : int  0 1 0 0 1 0 0 1 0 0 ...
##  $ Medication.Keeps.Patient.from.Sleeping    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Pain.Keeps.Patient.from.Sleeping          : int  0 0 0 0 0 0 1 0 1 0 ...
##  $ Bathroom.Needs.Keeps.Patient.from.Sleeping: int  0 1 0 1 0 1 1 0 1 0 ...
##  $ Uknown.Keeps.Patient.from.Sleeping        : int  1 0 1 0 0 0 0 0 0 1 ...
##  $ Trouble.Sleeping                          : int  2 3 3 3 2 3 2 3 3 3 ...
##  $ Prescription.Sleep.Medication             : int  3 3 3 3 3 3 1 3 3 3 ...
##  $ Race                                      : int  1 1 4 4 1 1 1 1 1 1 ...
##  $ Gender                                    : int  2 1 1 2 2 1 1 1 2 1 ...
```

Based on the description from the UCI repository, we know the data is categorical and the values for each feature correspond to a response value.

We can see that there are 714 observations(responses) and 15 variables (1 target; 14 features)

A key was provided by the UCI repository/survey authors. To easily examine and plot the data, a transformed data frame will be created to replaces the variable values the corresponding string response.

Loading the Key

```r
key_url <-
"https://raw.githubusercontent.com/anreinhart/edx-capstone/main/NPHA_Data_Info%20-%20Sheet1.csv"
NPHA_key <- read.csv(key_url)
```

Variable.Name	Description	Units
Number.of.Doctors.Visited	The total count of different doctors the patient has seen	{1: 0-1 doctors, 2: 2-3 doctors , 3: 4 or more doctors}
Age	The patient's age group	{1: 50-64, 2: 65-80}
Physical.Health	A self-assessment of the patient's physical well-being	{-1: Refused, 1: Excellent, 2: Very Good, 3: Good, 4: Fair, 5: Poor}
Mental.Health	A self-evaluation of the patient's mental or psychological health	{-1: Refused, 1: Excellent, 2: Very Good, 3: Good, 4: Fair, 5: Poor}
Dental.Health	A self-assessment of the patient's oral or dental health	{-1: Refused, 1: Excellent, 2: Very Good, 3: Good, 4: Fair, 5: Poor}

Variable.Name	Description	Units
Employment	The patient's employment status or work-related information	{-1: Refused, 1: Working full-time, 2: Working part-time, 3: Retired, 4: Not working at this time }
Stress.Keeps.Patient.from.Sleeping	Whether stress affects the patient's ability to sleep	{0: No , 1: Yes}
Medication.Keeps.Patient.from.Sleeping	Whether medication impacts the patient's sleep	{0: No , 1: Yes}
Pain.Keeps.Patient.from.Sleeping	Whether physical pain disturbs the patient's sleep	{0: No , 1: Yes}
Bathroom.Needs.Keeps.Patient.from.Sleeping	Whether the need to use the bathroom affects the patient's sleep	{0: No , 1: Yes}
Uknown.Keeps.Patient.from.Sleeping	Unidentified factors affecting the patient's sleep	{0: No , 1: Yes}
Trouble.Sleeping	General issues or difficulties the patient faces with sleeping	{0: No , 1: Yes}
Prescription.Sleep.Medication	Information about any sleep medication prescribed to the patient	{-1: Refused, 1: Use regularly, 2: Use occasionally, 3: Do not use}
Race	The patient's racial or ethnic background	{-2: Not asked, -1: REFUSED, 1: White Non-Hispanic, 2: Black Non-Hispanic, 3: Other Non-Hispanic, 4: Hispanic, 5: 2+ Races Non-Hispanic}
Gender	The gender identity of the patient	{-2: Not asked, -1: REFUSED, 1: Male, 2: Female}

A visual inspection of both data frames show a misspelling of a column in the NPHA_Original data frame. This data frame has "Phyiscal.Health", where the key is "Physical.Health" (the correct spelling). To avoid any errors in transformation, we will update the spelling in the original data frame.

Cleaning Data

```
NPHA_Original <- NPHA_Original %>%
  rename(Physical.Health = Phyiscal.Health)

str(NPHA_Original)
```

```
## 'data.frame':    714 obs. of  15 variables:
##  $ Number.of.Doctors.Visited                 : int  3 2 3 1 3 2 3 2 2 1 ...
##  $ Age                                       : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ Physical.Health                           : int  4 4 3 3 3 3 4 3 3 2 ...
##  $ Mental.Health                             : int  3 2 2 2 3 2 1 2 1 1 ...
##  $ Dental.Health                             : int  3 3 3 3 3 4 1 6 2 3 ...
##  $ Employment                                : int  3 3 3 3 3 3 3 3 3 1 ...
##  $ Stress.Keeps.Patient.from.Sleeping        : int  0 1 0 0 1 0 0 1 0 0 ...
##  $ Medication.Keeps.Patient.from.Sleeping    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Pain.Keeps.Patient.from.Sleeping          : int  0 0 0 0 0 0 1 0 1 0 ...
##  $ Bathroom.Needs.Keeps.Patient.from.Sleeping: int  0 1 0 1 0 1 1 0 1 0 ...
##  $ Uknown.Keeps.Patient.from.Sleeping        : int  1 0 1 0 0 0 0 0 0 1 ...
##  $ Trouble.Sleeping                          : int  2 3 3 3 2 3 2 3 3 3 ...
##  $ Prescription.Sleep.Medication             : int  3 3 3 3 3 3 1 3 3 3 ...
##  $ Race                                      : int  1 1 4 4 1 1 1 1 1 1 ...
##  $ Gender                                    : int  2 1 1 2 2 1 1 1 2 1 ...
```

The column name is corrected.

Creating Data Frame with Response Values

A function to replace the numeric values with their corresponding descriptions is created to easily apply to future data frames. This function maps the values in the NPHA_Original data frame to values in the NPHA_key data frame. As visualized earlier, the values and responses in the NPHA_key data frame are combined within {} brackets. This function separates each value & response by "," then returns corresponding response on the right side of the ":".

```r
# function to replace numeric values with descriptions
replace_with_descriptions <- function(df, key_df) {
  for (col_name in colnames(df)) { # iterating over each column
    var_name <- col_name
    mapping <- key_df$Units[key_df$Variable.Name == var_name]  # find matching units from
'key_df' based on 'var_name'
    if (length(mapping) > 0) { # proceed if there are matches in 'key_df'
      unit_str <- key_df$Units[key_df$Variable.Name == var_name] # extract the unit
string from 'key_df'
      unit_map <- strsplit(gsub("[{}]", "", unit_str), ", ")[[1]] #spliting the unit
string into key-value pairs and create a named vector ('unit_map')
  # creating vector where keys are numeric values and values are corresponding labels
      unit_map <- setNames(
        sapply(unit_map, function(x) {
          key_val <- strsplit(x, ": ")[[1]] # splitting key-value pair
          as.character(key_val[2]) #extracting label (value) and convert to character
        }),
        sapply(unit_map, function(x) {
          key_val <- strsplit(x, ": ")[[1]]# extracting numeric key and convert to
numeric type
          as.numeric(key_val[1])
        })
      )
      # Replace column values in 'df' with labels from 'unit_map'
      df[[col_name]] <- factor(df[[col_name]], levels = names(unit_map), labels =
unname(unit_map))
    }
  }
  return(df)
}

# applying function to create new data frame
```

```
NPHA_transformed <- replace_with_descriptions(NPHA_Original, NPHA_key)

str(NPHA_transformed)
```

```
## 'data.frame':    714 obs. of  15 variables:
##  $ Number.of.Doctors.Visited                 : Factor w/ 3 levels "0-1 doctors",..: 3 2 3 1 3 2 3 2
##  $ Age                                       : Factor w/ 2 levels "50-64","65-80": 2 2 2 2 2 2 2 2 2
##  $ Physical.Health                           : Factor w/ 6 levels "Refused","Excellent",..: 5 5 4 4
##  $ Mental.Health                             : Factor w/ 6 levels "Refused","Excellent",..: 4 3 3 3
##  $ Dental.Health                             : Factor w/ 6 levels "Refused","Excellent",..: 4 4 4 4
##  $ Employment                                : Factor w/ 5 levels "Refused","Working full-time",..:
##  $ Stress.Keeps.Patient.from.Sleeping        : Factor w/ 2 levels "No ","Yes": 1 2 1 1 2 1 1 2 1 1 .
##  $ Medication.Keeps.Patient.from.Sleeping    : Factor w/ 2 levels "No ","Yes": 1 1 1 1 1 1 1 1 1 1 .
##  $ Pain.Keeps.Patient.from.Sleeping          : Factor w/ 2 levels "No ","Yes": 1 1 1 1 1 1 2 1 2 1 .
##  $ Bathroom.Needs.Keeps.Patient.from.Sleeping: Factor w/ 2 levels "No ","Yes": 1 2 1 2 1 2 2 1 2 1 .
##  $ Uknown.Keeps.Patient.from.Sleeping        : Factor w/ 2 levels "No ","Yes": 2 1 2 1 1 1 1 1 1 2 .
##  $ Trouble.Sleeping                          : Factor w/ 2 levels "No ","Yes": NA NA NA NA NA NA NA
##  $ Prescription.Sleep.Medication             : Factor w/ 4 levels "Refused","Use regularly",..: 4 4
##  $ Race                                      : Factor w/ 7 levels "Not asked","REFUSED",..: 3 3 6 6
##  $ Gender                                    : Factor w/ 4 levels "Not asked","REFUSED",..: 4 3 3 4
```

Inspection of the transformed data frame reveals NA values in the Dental.Health column and the Trouble.Sleeping column.

```
NPHA_transformed %>%
  select(Dental.Health, Trouble.Sleeping) %>%
  slice_head(n = 8)
```

```
##   Dental.Health Trouble.Sleeping
## 1          Good             <NA>
## 2          Good             <NA>
## 3          Good             <NA>
## 4          Good             <NA>
## 5          Good             <NA>
## 6          Fair             <NA>
## 7     Excellent             <NA>
## 8          <NA>             <NA>
```

Handeling NA Values

While the description from the UCI repository states that there are no missing values in the data set, further inspection of the NPHA_original data frame shows that the Dental.Health column includes a key value of 6, which is not an documented key value.

The Trouble.Sleeping column contains values that do not correspond the the NPHA_key. Per the key the values for this column should be binary values of 0:No, 1:Yes. The values populating in the NPHA_Original data frame range from -1 to 3. This seemed odd, thus this author re-downloaded the data set from the repository and checked the raw CSV file to ensure data corruption did not occur. Unfotunely, values in the raw file match the incorrect values in the loaded data frame.

This author also investigated the original data source for the NPHA survey to ensure there was not a misrepresentation of the response values. According the the original data source, the response values for this column should be binary values of 0:No, 1:Yes. Since this data was pre-processed, it is possible that the original authors miss-labeled the column.

Because NAs will pose problems for model development, they will be removed from the data set.

Only rows containing NAs/6s in the Dental.Health column will be removed and the entire Trouble.Sleeping column will be removed since all values are corrupted.

Prior to removing the NAs and the Trouble.Sleep column, we must ensure there are no other NAs in the data.

First, NAs in the Dental.Health column are summed.

```
sum(is.na(NPHA_transformed$Dental.Health))
```

```
## [1] 55
```

Check to ensure this matches the number of 6 responses in the original data frame.

```
sum(NPHA_Original$Dental.Health == 6, na.rm = TRUE)
```

```
## [1] 55
```

Both match., which confirms there are no additional NAs are in this column.

Now to check the number of NAs in the Trouble.Sleeping column. This is done to ensure that the combined sum of NAs in the Dental.Health and the Trouble.Sleep column match the sum of NAs for the entire data frame. If there are more NAs in the entire data frame, further investigation will be needed to determine the source.

Summing NA's for Trouble.Sleeping

```
sum(is.na(NPHA_transformed$Trouble.Sleeping))
```

```
## [1] 652
```

The combined sum of the two columns is 707.

The sum of NAs for the entire transformed data frame:

```
sum(is.na(NPHA_transformed))
```

```
## [1] 707
```

707! This matches the combined sum, thus it is safe to move forward with removing NAs.

Since this needs to be applied to both the NPHA_Original and NPHA_transformed data frames, a function is created to remove rows with NAs and 6s in the Dental.Health column, and remove the entire Trouble.Sleeping column.

```
remove_column_and_na_rows <- function(df, column_to_remove) {
  # check if the column_to_remove exists in the data frame
  if (column_to_remove %in% colnames(df)) {
    # remove the specified column
    df <- df[, !colnames(df) %in% column_to_remove]
  } else {
    cat(paste("Column", column_to_remove, "does not exist in the dataframe.\n"))
    return(NULL)
  }

  # filter out rows with NA values across remaining columns
  df <- df[complete.cases(df), ]

  # filter out rows where Dental.Health is 6
  df <- df[df$Dental.Health != 6, ]

  return(df)
}

#applying function to transformed dfs
NPHA_transformed <- remove_column_and_na_rows(NPHA_transformed, "Trouble.Sleeping")

NPHA_Original <- remove_column_and_na_rows(NPHA_Original, "Trouble.Sleeping")
```

The data frames have been cleaned, and now contains 659 observations (responses) and 14 variables (1 target; 13 features)

Final check to ensure all NAs are removed:

```
sum(is.na(NPHA_transformed))
```

```
## [1] 0
```

All NAs removed, we can continue with our data exploration.

Exploratory Data Analysis

The target variable in this data frame is Number.of.Doctors visited.

Doctor Visits

```
table(NPHA_transformed$Number.of.Doctors.Visited)
```

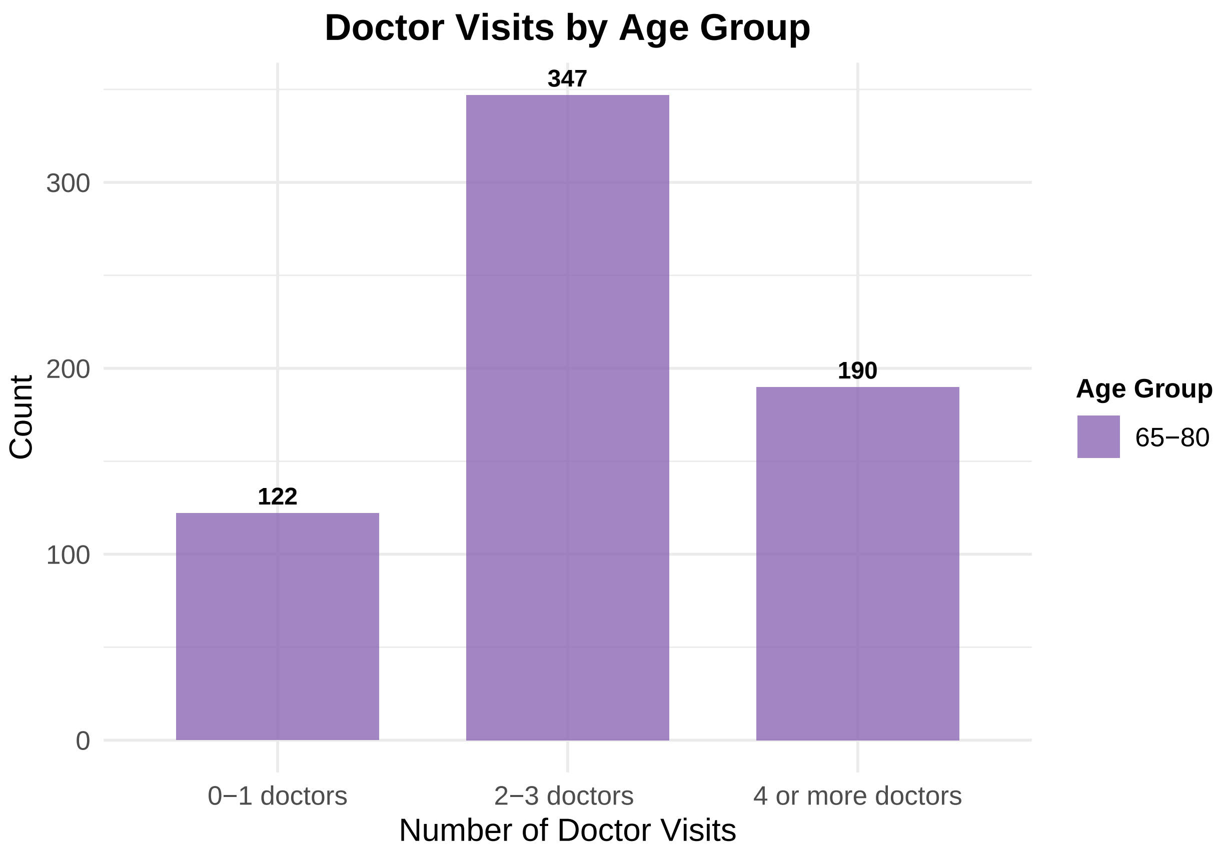
```
##
##       0-1 doctors      2-3 doctors  4 or more doctors
##               122              347               190
```

The data is imbalanced, as there are more responses in the "2-3 doctors" category and than the other two categories combined. The influence of an imbalanced data set will be further explored in the Methods and Analysis: Model section.

Age & Doctor Visits

Per the data set description observations fall within two age categories: 50-64, or 65-80.

```r
# create the basic bar chart
ggplot(NPHA_transformed, aes(x = factor(Number.of.Doctors.Visited), fill = Age)) +

  # add bars with dodged position and outline
  geom_bar(position = "dodge", color = NA, alpha = 0.8, width = 0.7) +

  # add text labels on top of bars (showing counts)
  geom_text(
    aes(label = stat(count)),
    stat = "count",
    position = position_dodge(width = 0.9),
    vjust = -0.5, # adjust vertical position of labels
    size = 3, # label size
    color = "black", # text color
    fontface = "bold" # label text style
  ) +

  # set fill colors for each Age group
  scale_fill_manual(
    values = c("50-64" = "#4CAF50", "65-80" = "#8C67B5"),
    name = "Age Group" # Customize legend title
  ) +
  labs(
    x = "Number of Doctor Visits",
    y = "Count",
    title = "Doctor Visits by Age Group"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(size = 14, face = "bold", hjust = 0.5),
    axis.title = element_text(size = 12),
    axis.text = element_text(size = 10),
    legend.title = element_text(size = 10, face = "bold"),
    legend.text = element_text(size = 10)
  )
```

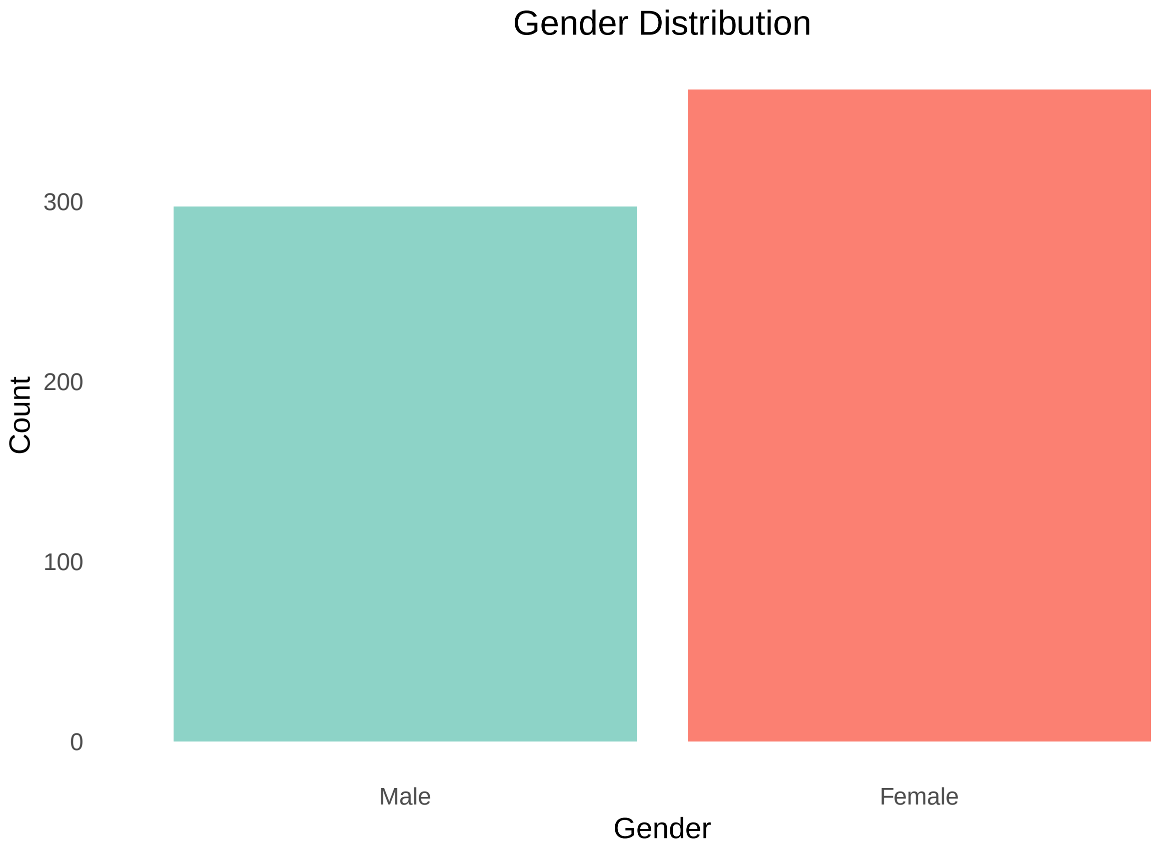


This data only contains responses from the 65-80 age group, therefore we can remove this column from the data set, as it is not relevant to this analysis.

```
# removing age column, as there is only one age group
NPHA_Original <- subset(NPHA_Original, select = -Age)
NPHA_transformed <- subset(NPHA_transformed, select = -Age)
```

Gender & Race

```
NPHA_transformed %>%
  ggplot(aes(x = factor(Gender, levels = c("Male", "Female")), fill = Gender)) +
  # using factor to specify the order of 'Gender' levels in x-axis
  geom_bar() +
  # creating a bar plot
  scale_fill_manual(values = c("Male" = "#8dd3c7", "Female" = "#fb8072")) +
  # fill colors
  labs(x = "Gender", y = "Count", title = "Gender Distribution") +
  # Label x-axis, y-axis, and plot title
  geom_text(stat = "count", aes(label = ..count..), vjust = -0.5, size = 4, color =
  "white") +
  # adding count labels above bars, adjust text position and size
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 0, hjust = 0.5)) +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank()) +
  theme(legend.position = "none")
```



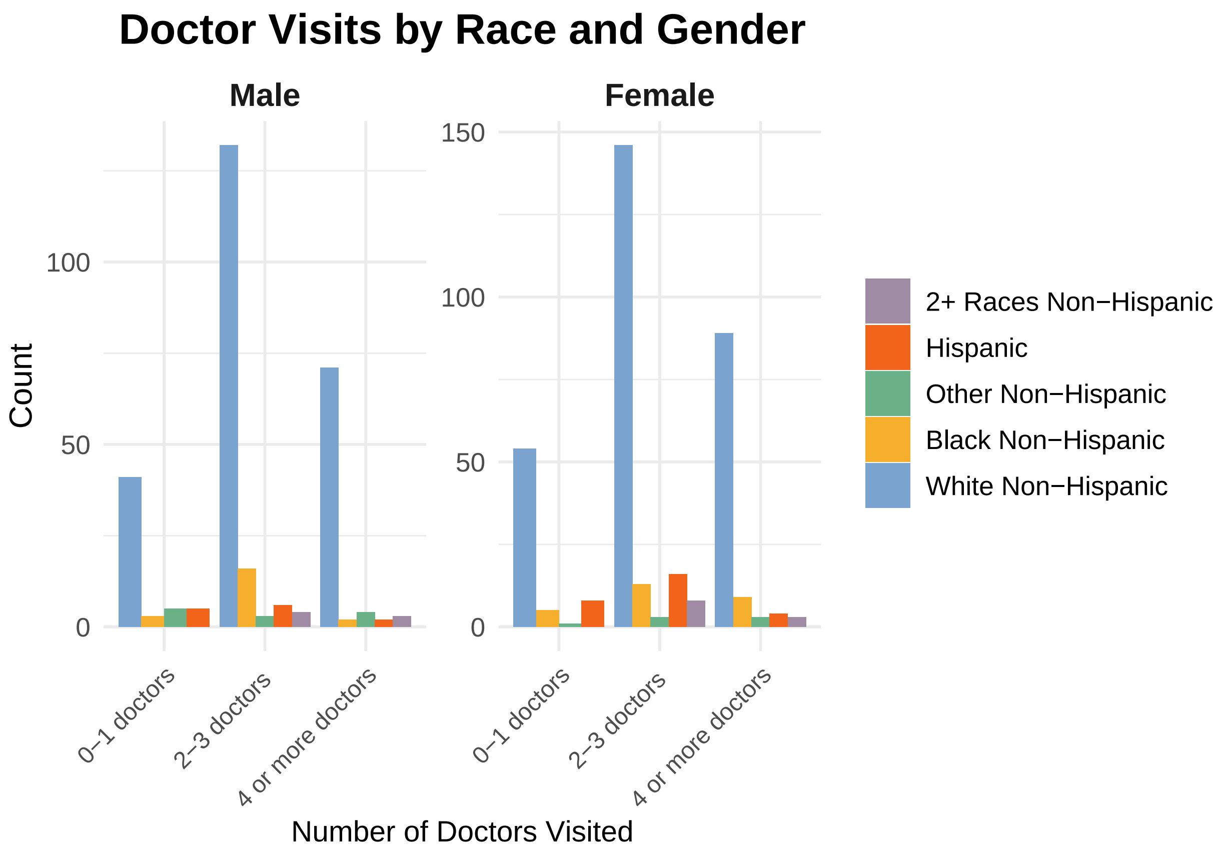
The data includes more responses from females than males. This is not surprising, as it is well known that on average women live longer than men.

```
table(NPHA_transformed$Race, NPHA_transformed$Gender)
```

```
##
##                           Not asked REFUSED Male Female
##   Not asked                       0       0    0      0
##   REFUSED                         0       0    0      0
##   White Non-Hispanic              0       0  244    289
##   Black Non-Hispanic              0       0   21     27
##   Other Non-Hispanic              0       0   12      7
##   Hispanic                        0       0   13     28
##   2+ Races Non-Hispanic           0       0    7     11
```

The total number of white respondents is greater than all other races combined.

```
# creating custome color pallet for race categories
race_palette <- c(
  "#7BA3D0", "#F6AE2D", "#6AB187", "#F26419", "#A08BA5", "#FF7F00",
  "#4B4E6D"
)
# plotting doctor visits by race and gender with tilted x-axis labels
ggplot(NPHA_transformed, aes(x = Number.of.Doctors.Visited, fill = Race)) +
  # add bar chart with dodged bars for each Race category
  geom_bar(position = "dodge", color = NA, size = 0.2) +
  scale_fill_manual(values = race_palette) +
  # facet by Gender to create separate plots for each gender
  facet_wrap(~Gender, scales = "free") +
  labs(
    title = "Doctor Visits by Race and Gender",
    x = "Number of Doctors Visited",
    y = "Count"
  ) +
  # Customize axis and legend labels, and tilt x-axis labels
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
    axis.title.y = element_text(size = 12),
    axis.text.y = element_text(size = 10),
    strip.text = element_text(size = 12, face = "bold"),
    legend.title = element_blank(),
    legend.text = element_text(size = 10),
    axis.text.x = element_text(angle = 45, hjust = 1)
  ) + # Rotate x-axis labels
  # Remove legend title and customize legend appearance
  guides(fill = guide_legend(reverse = TRUE, title.position = "top", title.hjust = 0.5))
  +

  # Adjust facet labels
  theme(
    strip.background = element_blank(),
    strip.placement = "outside"
  )
```



"2-3 doctors" a year is the most common among both male/female and difference races, followed by 4 or more doctors and then 0-1 doctors. We can also tell from this visual that a higher number of white non-Hispanic responses were recorded than other races.

Average Ratings

```
avg_responses <- colMeans(NPHA_Original[3:5])
# obtaining means by col
print(avg_responses)
```

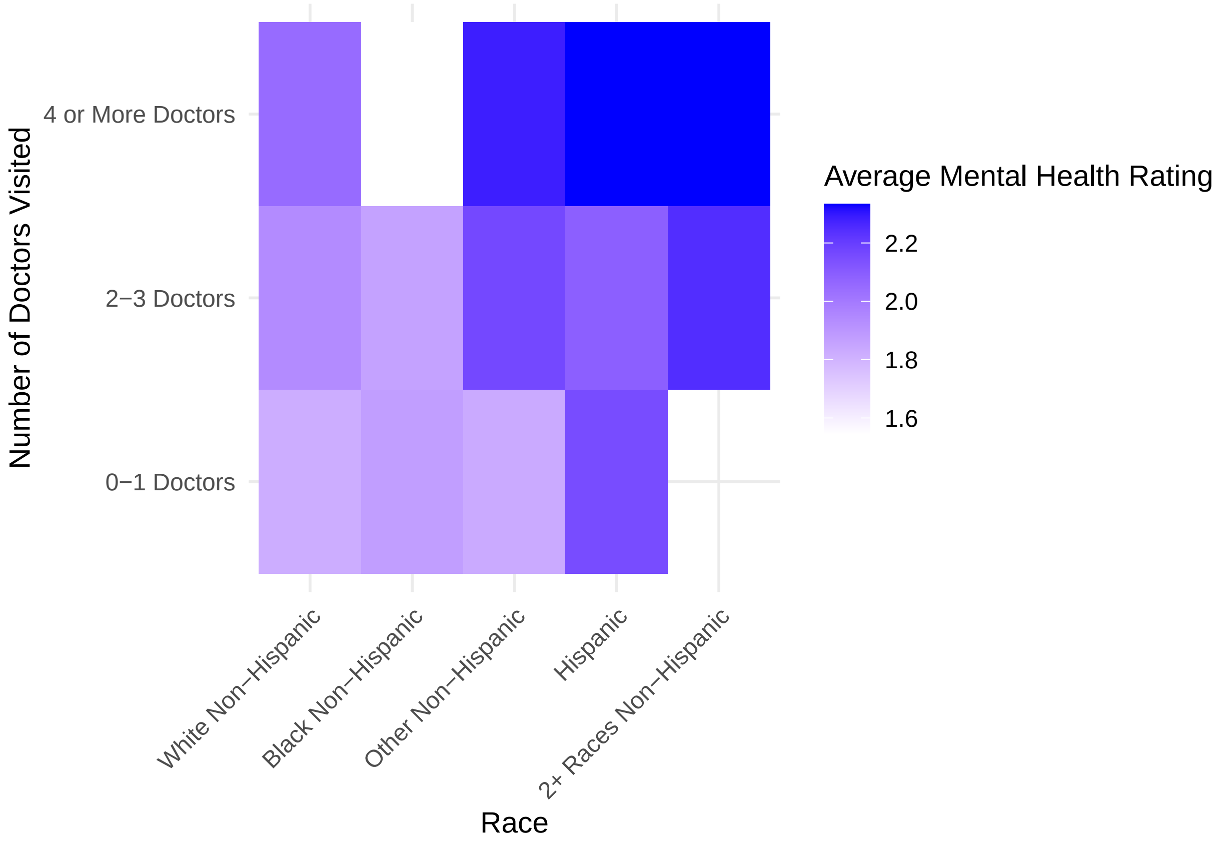
```
## Mental.Health Dental.Health    Employment
##      1.965099      2.760243      2.798179
```

A simple average of responses for each variable shows that the respondents did not rate their Mental.Health very high. Physical.Health and Dental.Health were rated higher. This data can be skewed by the different respondents. A common response, or average might be different for a different cohort (i.e. White Non-Hispanic, vs Hispanic Vs, Black Non-Hispanic, and etc.) These relationships will be further explored.

Mental Health

```
avg_mental_health <- aggregate(Mental.Health ~ Race + Number.of.Doctors.Visited, data =
NPHA_Original, FUN = mean)
# calculating mean of mental.health var grouped by race and number of doctor visits
```

```r
#creating labels for plot
doctor_labels <- c("0-1 Doctors", "2-3 Doctors", "4 or More Doctors")
race_labels <- c("White Non-Hispanic", "Black Non-Hispanic", "Other Non-Hispanic",
"Hispanic", "2+ Races Non-Hispanic")

# plotting tiled heatmap
ggplot(avg_mental_health, aes(x = factor(Race, levels = 1:5, labels = race_labels),#
defining x-axis w/ custom labels
                              y = factor(Number.of.Doctors.Visited, levels = 1:3, labels
                              = doctor_labels),  #defining y-axis w/ custom labels
                              fill = Mental.Health)) + #fill color base on avg
                              mental.health value
  geom_tile() +
  scale_fill_gradient(low = "white", high = "blue") + # color gradient
  labs(x = "Race", y = "Number of Doctors Visited", fill = "Average Mental Health
  Rating") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) #rotatin axis to fit
```



The Mental.Health ratings are categorized as:

- -1: Refused
- 1: Excellent

- 2: Very Good
- 3: Good
- 4: Fair
- 5: Poor

On average, White Non-Hispanic and Black Non-Hispanic respondents rated their mental health lower than Other Non-Hispanic, Hispanic, and 2+ Races Non-Hispanic.

Physical Health

```
physhealth_results <- table(NPHA_transformed$Physical.Health)
physhealth_results
```

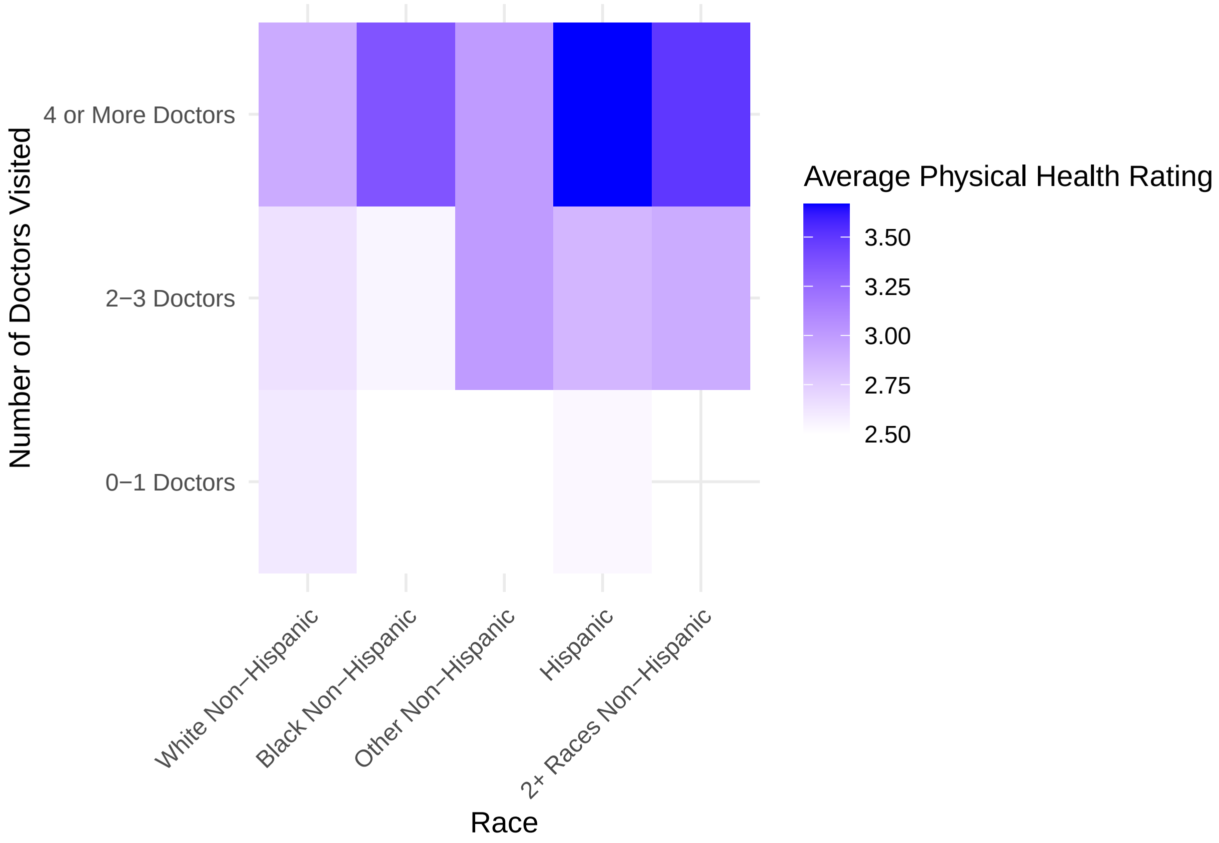
```
##
##   Refused Excellent Very Good      Good      Fair      Poor
##         1        36       232       268       103        19
```

```
percentage_physhealth <- prop.table(physhealth_results) * 100
# combine categories and percentages into a data frame
results_df <- data.frame(
  Response_Category = names(physhealth_results),
  Count = as.numeric(physhealth_results), # convert counts to numeric
  Percentage = percentage_physhealth
)

print(percentage_physhealth)
```

```
##
##    Refused  Excellent  Very Good       Good       Fair       Poor
##  0.1517451  5.4628225 35.2048558 40.6676783 15.6297420  2.8831563
```

A majority of respondents rate their physical health as Very Good or Good.

```
avg_physical_health <- aggregate(Physical.Health ~ Race + Number.of.Doctors.Visited, data
= NPHA_Original, FUN = mean)
# calculating mean of physical.health var grouped by race and number of doctor visits

# Plot heatmap
ggplot(avg_physical_health, aes(x = factor(Race, levels = 1:5, labels = race_labels), #
defining x axis w/custom labels
                                y = factor(Number.of.Doctors.Visited, levels = 1:3,
                                labels = doctor_labels), # defining y axis w/custom
                                labels
                                fill = Physical.Health)) + #setting fill to
                                physical.health value
  geom_tile() +
  scale_fill_gradient(low = "white", high = "blue") + # color gradient
  labs(x = "Race", y = "Number of Doctors Visited", fill = "Average Physical Health
  Rating") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) #tilting axis to fit
```



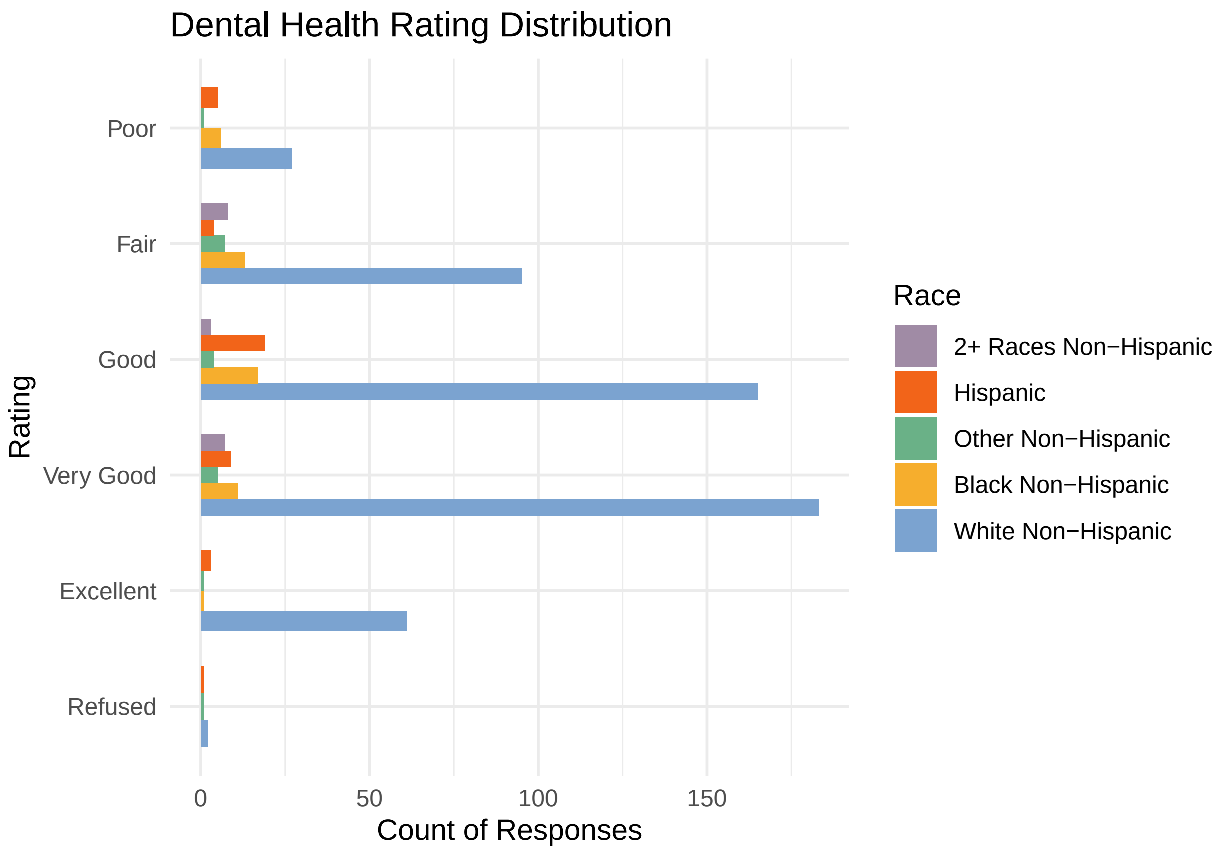
The Physical.Health ratings are categorized as:

- -1: Refused
- 1: Excellent
- 2: Very Good
- 3: Good
- 4: Fair
- 5: Poor

Again, we see that Hispanic respondents are rating their Physical.Health higher than other respondents, but also seeing more doctors. An assumption is that a respondent who rates their physical health poor (or even mental or dental health) would see more doctors than someone who rates their health as < "Good". This may imply that other factors play a significant roll in the number of doctor visits.

Dental Health

```r
ggplot(NPHA_transformed, aes(x = factor(Dental.Health), fill = Race)) +

  # plot frequencies
  geom_bar(position = "dodge", stat = "count", width = 0.7) +
  scale_fill_manual(values = race_palette) +

  # plot titles and axis labels
  labs(
    title = "Dental Health Rating Distribution",
    x = "Rating",
    y = "Count of Responses",
    fill = "Race"
  ) +
  theme_minimal() +
  theme(axis.text.y = element_text(angle = 0, hjust = 1)) +
  guides(fill = guide_legend(reverse = TRUE)) + # reverse the order of legend keys
  coord_flip() # flip coordinates for horizontal bar plot
```



Hispanic and Black Non-Hispanic respondents rated their dental health as "Good" more frequently than other responses, where as a majority of white respondents rated their dental health as "Good", "Very Good", or "Excellent". Very few respondents of the non-white races rated their dental health as "Excellent".

Employment

This author was interested in how respondents categorized their employment vs how they rated their mental, physical, and dental health.

```
table(NPHA_transformed$Employment)
```

```
##
##                   Refused         Working full-time         Working part-time
##                         0                        47                        53
##                   Retired Not working at this time
##                       545                        14
```

```
#calcualting retired %
Retiredpercent <- NPHA_transformed %>%
  filter(NPHA_transformed$Employment == "Retired") #filtering table to just retired

(nrow(Retiredpercent) / length(NPHA_transformed$Employment)) * 100
```

```
## [1] 82.70106
```

*82 % of respondents are Retired. The remaining respondents are either working full time, part time, or not working nor retired.

Let's view the average ratings per Physical, Mental, and Dental health by employment status.

```
#grouping 'NPHA_Original' by recoded 'Employment' categories and calculate averages
avg_health_by_employment <- NPHA_Original %>%
  group_by(Employment_label = recode(Employment,
    "1" = "Woirking full-time",
    "2" = "Working part-time",
    "3" = "Retired",
    "4" = "Not working at this time"
  )) %>%
  # grouping by label and for visual
  summarise(
    avg_phys_health = mean(Physical.Health, na.rm = TRUE),
    avg_ment_health = mean(Mental.Health, na.rm = TRUE),
    avg_dent_health = mean(Dental.Health, na.rm = TRUE)
  )


kable(avg_health_by_employment)
```

Employment_label	avg_phys_health	avg_ment_health	avg_dent_health
Not working at this time	3.714286	2.500000	3.571429
Retired	2.774312	1.981651	2.761468
Woirking full-time	2.553192	1.851064	2.723404
Working part-time	2.377359	1.754717	2.566038

Respondents who listed their employment status as “Not working at this time” had, on average, higher ratings of of Physical, Mental, and Dental health.

It is unclear in the data and response key, what the difference is between “Not working at this time” and “Retired”. If directions were unclear in the original survey, it is possible respondents could have mistakenly classified themselves.

```
avg_health_by_employment_visits <- NPHA_Original %>%
  mutate(Employment_label = case_when(
    Employment == 1 ~ "Working full-time",
    Employment == 2 ~ "Working part-time",
    Employment == 3 ~ "Retired",
    Employment == 4 ~ "Not working at this time"
  )) %>%
  # grouping by label and recoding for visual
  group_by(Employment_label, Number.of.Doctors.Visited) %>%
  summarise(
    avg_phys_health = mean(Physical.Health, na.rm = TRUE),
    avg_ment_health = mean(Mental.Health, na.rm = TRUE),
    avg_dent_health = mean(Dental.Health, na.rm = TRUE)
  ) %>%
  ungroup() %>% # removing grouping for further operations
  mutate(Number.of.Doctors.Visited = recode(Number.of.Doctors.Visited,
  !!!setNames(doctor_labels, 1:3)))


kable(avg_health_by_employment_visits)
```

Employment_label	Number.of.Doctors.Visited	avg_phys_health	avg_ment_health	avg_dent_health
Not working at this time	0-1 Doctors	3.250000	2.500000	4.250000
Not working at this time	2-3 Doctors	4.000000	4.000000	3.000000
Not working at this time	4 or More Doctors	3.888889	2.333333	3.333333
Retired	0-1 Doctors	2.648936	1.851064	2.914894
Retired	2-3 Doctors	2.672414	1.972414	2.706897
Retired	4 or More Doctors	3.031056	2.074534	2.770186
Working full-time	0-1 Doctors	2.272727	2.000000	2.909091
Working full-time	2-3 Doctors	2.793103	1.862069	2.689655
Working full-time	4 or More Doctors	2.000000	1.571429	2.571429
Working part-time	0-1 Doctors	2.230769	1.615385	2.461539
Working part-time	2-3 Doctors	2.444444	1.814815	2.518518
Working part-time	4 or More Doctors	2.384615	1.769231	2.769231

Once split to view by Number.of.Doctors.Visited, the “Not working at this time” maintained a higher average than all other respondents.

Interestingly, Retired respondents who visit “4 or more” doctors, tend to rate their mental, physical, and dental health more highly than their counterparts,

And unsurprisingly, respondents who work full time follow an an expected trend of lower health ratings = greater number of doctor visits.

Sleep

The data on sleep includes binary responses (yes/no) to several questions regarding issues that can affect sleep. Those issues are:

- Stress
- Medication
- Pain
- Bathroom Needs
- Unknown

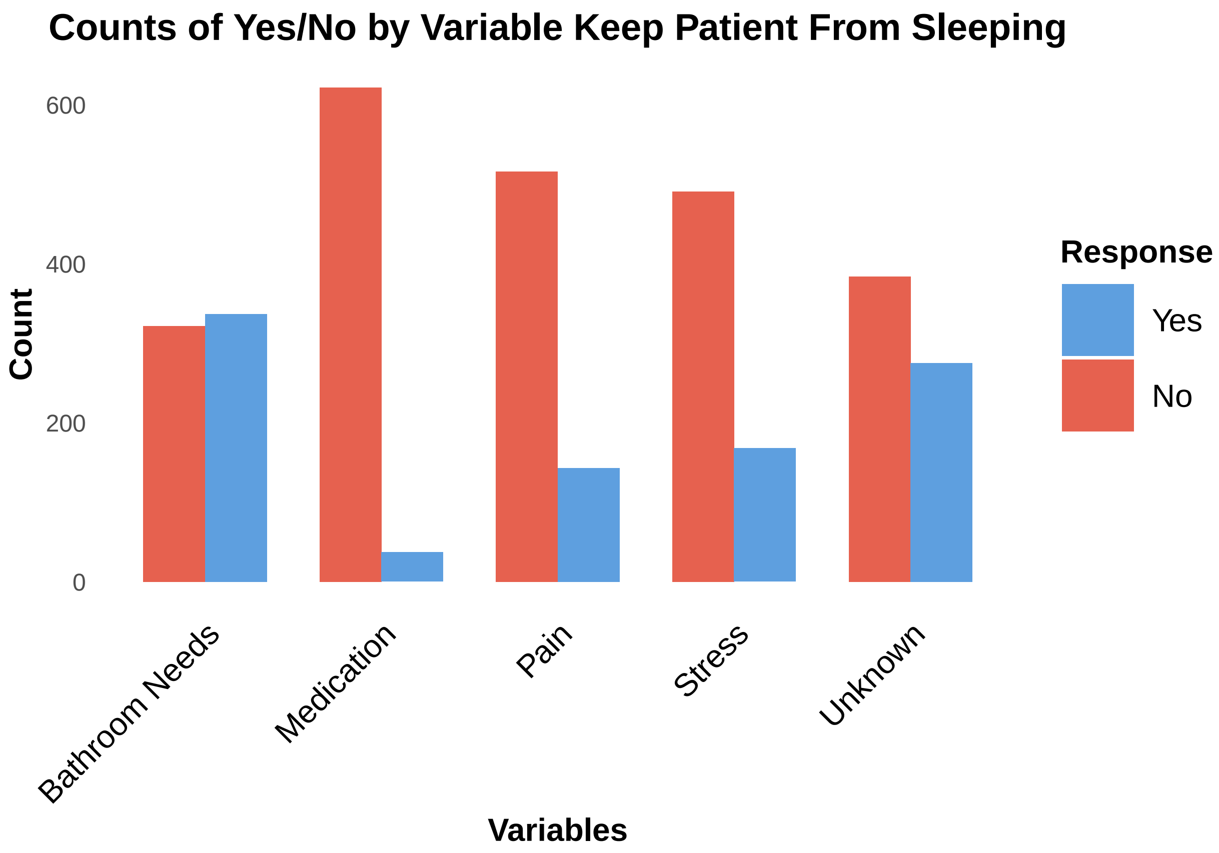
The survey also captured if a respondent utilizes prescription sleep aides with a range of:

- -1: Refused
- 1: Use regularly
- 2: Use occasionally
- 3: Do not use

As aforementioned, we removed the Trouble.Sleeping question, as the values did not match the response key.

```r
NPHAtrans_sleep_subset <- NPHA_transformed[, c("Stress.Keeps.Patient.from.Sleeping",
"Medication.Keeps.Patient.from.Sleeping", "Pain.Keeps.Patient.from.Sleeping",
"Uknown.Keeps.Patient.from.Sleeping", "Bathroom.Needs.Keeps.Patient.from.Sleeping")]
# subetting for visualizaition

# reshape data into long format
NPHASleep_long <- pivot_longer(NPHAtrans_sleep_subset,
  cols = everything(),
  names_to = "Variable",
  values_to = "Response"
)

# coounts of yes * no
sleepcounts <- table(NPHASleep_long$Variable, NPHASleep_long$Response)
sleepcounts <- data.frame(sleepcounts)


# creating custome labels for x axis
sleep_labels <- c(
  "Stress.Keeps.Patient.from.Sleeping" = "Stress",
  "Medication.Keeps.Patient.from.Sleeping" = "Medication",
  "Pain.Keeps.Patient.from.Sleeping" = "Pain",
  "Uknown.Keeps.Patient.from.Sleeping" = "Unknown",
  "Bathroom.Needs.Keeps.Patient.from.Sleeping" = "Bathroom Needs"
)

# color palette
custom_colors <- c("Yes" = "#5E9FDF", "No " = "#E6614F")
```

```r
# grouped bar chart
ggplot(sleepcounts, aes(x = Var1, y = Freq, fill = Var2)) +
  geom_bar(stat = "identity", position = "dodge", width = 0.7) +
  labs(
    title = "Counts of Yes/No by Variable Keep Patient From Sleeping",
    x = "Variables", y = "Count"
  ) +
   #customize the fill (color) scale
  scale_fill_manual(
    values = custom_colors, name = "Response",
    labels = c("Yes" = "Yes", "No " = "No")
  ) +
  # customizing x-axis labels using predefined sleep_labels
  scale_x_discrete(labels = sleep_labels) +
  theme_minimal() +
  theme(
    plot.title = element_text(size = 14, face = "bold", hjust = 0.5),
    axis.title.x = element_text(size = 12, face = "bold"),
    axis.title.y = element_text(size = 12, face = "bold"),
    axis.text.x = element_text(angle = 45, hjust = 1, size = 12, color = "black"),
    legend.title = element_text(size = 12, face = "bold"),
    legend.text = element_text(size = 12),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.border = element_blank(),
    panel.background = element_blank(),
    legend.key.size = unit(1, "cm")
  ) +
  guides(fill = guide_legend(reverse = TRUE)) # reverse legend order
```



We can see that among the sleep questions, both Bathroom.Needs and Unknown.Reasons received the most "Yes" responses, but otherwise not many respondents report their sleep being affected by other sources.

```
table(NPHA_transformed$Prescription.Sleep.Medication)
```

```
##
##            Refused     Use regularly Use occasionally        Do not use
##                  3                34               32               590
```

And the majority of respondents do not use prescription sleep medications.

Methods and Analysis

Correlation

Since this data contains more than a 2x2 categorical variables, conventional correlation techniques like the chi square or the Pearson correlation tests will not be used. Instead to evaluate if any correlation exists between the target variable and other features Cramer's V will be utilized. Cramer's V is used to examine correlations between two categorical variables when the contingency is greater than 2x2.

Cramer's V must lie between 0 and 1, where 0 reflects complete independence and 1 reflects complete dependence or association. (3 SITE)

```r
# Calculate Cramér's V for categorical variables
cramer_v <- function(x, y) {
  tab <- table(x, y)
  chisq <- chisq.test(tab)$statistic
  n <- sum(tab)
  df <- (nrow(tab) - 1) * (ncol(tab) - 1) # degrees of freedom calculation

  # Cramer's V with NaN values excluded
  if (is.nan(chisq)) {
    return(NA) # return NA if chisq is NaN
  } else {
    return(sqrt(chisq / (n * df)))
  }
}

# correlations with target variable
correlations <- sapply(NPHA_transformed[-1], function(x) {
  cramer <- cramer_v(NPHA_transformed$Number.of.Doctors.Visited, x)
  ifelse(is.na(cramer), 0, cramer) # replace NaN with 0
})


kable(correlations)
```

	x
Physical.Health.X-squared	0.0612082
Mental.Health.X-squared	0.0365069
Dental.Health.X-squared	0.0377572
Employment	0.0000000
Stress.Keeps.Patient.from.Sleeping.X-squared	0.0555148
Medication.Keeps.Patient.from.Sleeping.X-squared	0.0840584
Pain.Keeps.Patient.from.Sleeping.X-squared	0.0551769
Bathroom.Needs.Keeps.Patient.from.Sleeping.X-squared	0.0514629
Uknown.Keeps.Patient.from.Sleeping.X-squared	0.0177796
Prescription.Sleep.Medication.X-squared	0.0580096
Race	0.0000000
Gender	0.0000000

The factors Physical.Health and Medications.Keep.Patient.From.Sleeping seem to have a higher correlation to our target variable of Number.of.Doctors.Visited than the others.

Due to the target variable having three levels, this author attempted to run Cramer's V on each level independently. Unfortunately, due to the low number of samples, the formula was unable to properly run.

Creating Partitions

Since the data set is not very large, it will be partitioned in a traditional 70/30 split to prevent the model from over fitting. The partitions will be created with the NPHA_Original data set to provide consistency in the models.

```
set.seed(123) # set seed for reproducibility

index_original <- createDataPartition(NPHA_Original$Number.of.Doctors.Visited, p = 0.7,
list = FALSE)

train_original <- NPHA_Original[index_original, ] # 70% of the data for training
test_original <- NPHA_Original[-index_original, ] # Remaining 30% of the data for testing
```

Model Seleciton

Random Forest

The first model that will be evaluated is the Random Forest model. This model will utilize the randomForest function from the randomForest package.

This model was selected because it is a supervised non-linear model that can handle categorical data without further transformation. The random forest approach is a step-above the decision tree approach, as it builds a forest of decision trees that specifies the categories with higher probability.

Before deploying the model and proceeding model, we'll first convert the target variable "Number.of,Doctors.Visited" into a factor, as these are classification models and both models expect the target to be a factor.

```
train_original$Number.of.Doctors.Visited <-
factor(train_original$Number.of.Doctors.Visited)
test_original$Number.of.Doctors.Visited <-
factor(test_original$Number.of.Doctors.Visited)
```

```
# function for randomforest
fit_random_forest <- function(train_data, test_data, target_variable, to_exclude = NULL,
ntree = 50000) {
  if (!is.null(to_exclude)) {
    x_train <- train_data[, -to_exclude]
    x_test <- test_data[, -to_exclude]
  } else {
    x_train <- train_data
    x_test <- test_data
  }

  y_train <- train_data[[target_variable]]

  # fitting RF
  set.seed(120) # Setting seed
  classifier_RF <- randomForest(x = x_train, y = y_train, ntree = ntree)

  # predicting on test set
  y_pred <- predict(classifier_RF, newdata = x_test)

  # confusion matrix
  confusion_mtx <- table(test_data[[target_variable]], y_pred)

  # calculate accuracy using confusion matrix
  accuracy <- sum(diag(confusion_mtx)) / sum(confusion_mtx)

  return(list(
    model = classifier_RF,
    confusion_matrix = confusion_mtx,
    accuracy = accuracy
  ))
}
```

```
to_exclude <- c(1) # index of the target variable to exclude

# calling RF function
result <- fit_random_forest(
  train_data = train_original,
  test_data = test_original,
  target_variable = "Number.of.Doctors.Visited",
  to_exclude = to_exclude
)

# results
result$confusion_matrix # The confusion matrix on actual predictions
```

```
##    y_pred
##      1  2  3
##   1  5 30  3
##   2  6 83 13
##   3  5 43  9
```

```
result$accuracy # The accuracy of the model
```

```
## [1] 0.4923858
```

This model produced an accuracy of 49%. Not great, but not a bad start. Based on the confusion matrix, which displays how many of the predictions were correct vs incorrect, we can see that this version is good at predicting the “2-3 doctors” level, but it does not perform well on the other levels.

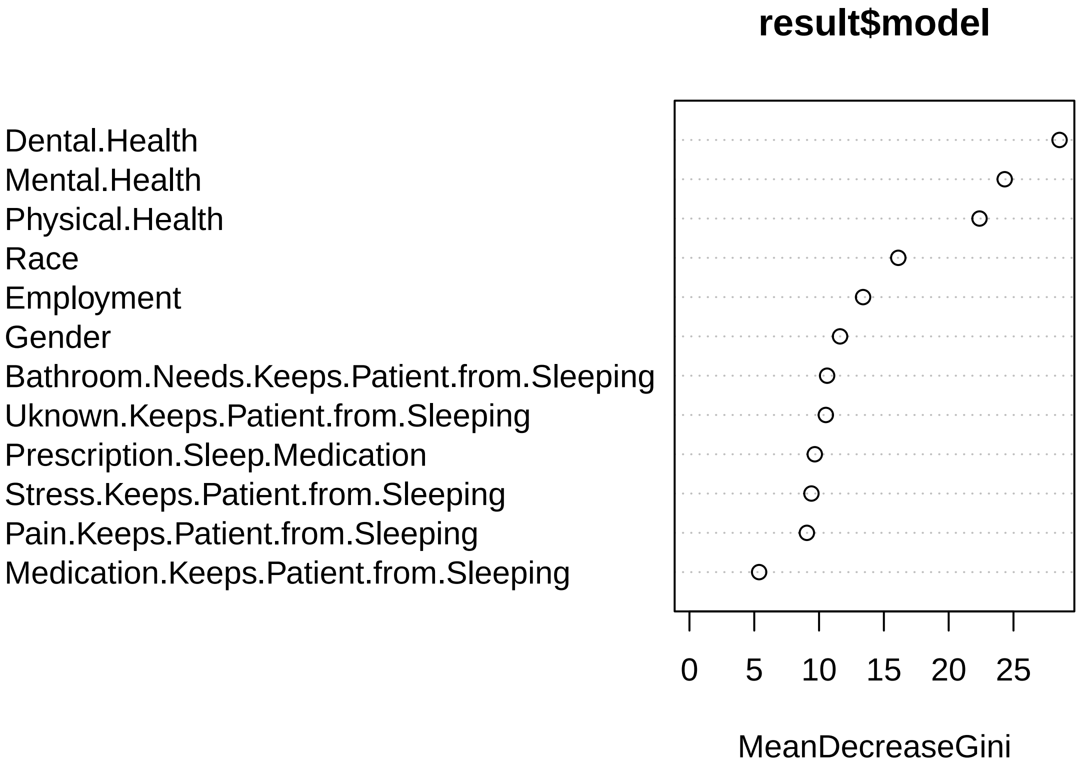
```
# create an empty tibble for model results
model_results <- tibble(
  Model = character(),
  Accuracy = numeric()
)

# updating model_results tibble
model_results <- bind_rows(model_results, tibble(
  Model = "Random Forest",
  Accuracy = result$accuracy
))

kable(model_results)
```

Model	Accuracy
Random Forest	0.4923858

```
# importance plot
importance(result$model)
```

```
##                                          MeanDecreaseGini
## Physical.Health                                 22.379244
## Mental.Health                                   24.322849
## Dental.Health                                   28.554888
## Employment                                      13.395084
## Stress.Keeps.Patient.from.Sleeping               9.411176
## Medication.Keeps.Patient.from.Sleeping           5.375793
## Pain.Keeps.Patient.from.Sleeping                 9.059172
## Bathroom.Needs.Keeps.Patient.from.Sleeping      10.621133
## Uknown.Keeps.Patient.from.Sleeping              10.522681
## Prescription.Sleep.Medication                    9.669698
## Race                                            16.112109
## Gender                                          11.619804
```

```
# variable importance plot
varImpPlot(result$model)
```



This model also provides the MeanDecreaseGini, which is indicative of the importance of each variable. Thus the higher the MeanDeacreseGini, the higher the relative importance.

We will further tune this model with feature selection based on the MeanDecreasGini. We can see that Physical.Health, Mental.Health, Dental.Health., Race, and Employment appear to be th most important features. These features will be isolated and tested.

```
# extracting feature importance scores
feature_importance <- importance(result$model)
# extraxting names of important features based on MeanDecreaseGini values in descending
order
important_features <- names(sort(feature_importance[, "MeanDecreaseGini"], decreasing =
TRUE))

# selecting top n important features
n <- 5
selected_features <- important_features[1:n]


# subset the training and test data with selected features
train_data_selected <- train_original[, c("Number.of.Doctors.Visited",
selected_features)]

test_data_selected <- test_original[, c("Number.of.Doctors.Visited", selected_features)]



# call RF function
result_features <- fit_random_forest(
  train_data = train_data_selected,
  test_data = test_data_selected,
  target_variable = "Number.of.Doctors.Visited",
  to_exclude = to_exclude
)

# results
result_features$model # The trained Random Forest model
```

```
##
## Call:
##  randomForest(x = x_train, y = y_train, ntree = ntree)
##                Type of random forest: classification
##                      Number of trees: 50000
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 49.35%
## Confusion matrix:
##    1   2  3 class.error
## 1 10  60 14   0.8809524
## 2  8 203 34   0.1714286
## 3  6 106 21   0.8421053
```

```
result_features$confusion_matrix # The confusion matrix
```

```
##    y_pred
##      1  2  3
##   1  3 29  6
##   2  0 78 24
##   3  4 41 12
```

```
result_features$accuracy # The accuracy of the model
```

```
## [1] 0.4720812
```

The feature selections based on the MeanDecreaseGini actually decreased the accuracy of the model.

```
# updating model_results tibble
model_results <- model_results %>%
  add_row(Model = "Random Forest w/Feature Selection", Accuracy =
  result_features$accuracy)

kable(model_results)
```

Model	Accuracy
Random Forest	0.4923858
Random Forest w/Feature Selection	0.4720812

The next model evaluated is K Nearest Neighbors classification (KNN). For each row of the test set, the k nearest (in Euclidean distance) training set vectors are found, and the classification is decided by majority vote, with ties broken at random. If there are ties for the kth nearest vector, all candidates are included in the vote.

K-fold cross validation will also be repeatedly applied to resample the data in an effort to prevent bias.

KNN

A function is created for the KNN model. Again, this will make it more efficient to apply this model to new data.

```
# creating function for KNN
train_and_evaluate_knn <- function(train_data, test_data, target_variable, k_values =
15:25,
                                   cv_folds = 10, cv_repeats = 3) {
  # preparing features (X) and target variable (Y) from training dataset
  X <- train_data[, !names(train_data) %in% target_variable] # features excluding target
variable
  Y <- train_data[[target_variable]] # tagret variable

  # Train KNN model
  set.seed(123) # set seed for reproducibility
  knn_model <- train(X, Y,
    method = "knn",
    trControl = trainControl( #training control for model tuning
      method = "repeatedcv", # use repeated k-fold cross-validation for model evaluation
      number = cv_folds, # folds
```

```r
      repeats = cv_repeats #times to repeat CV
      ),
    tuneGrid = expand.grid(k = k_values) #specify grid k-values to tune
)


  # print the best tuned parameter (optimal k)
  knn_best_k <- knn_model$bestTune

  # predict on test data
  predictions <- predict(knn_model, newdata = test_data[, !names(test_data) %in%
target_variable])

  #  model performance
  accuracy <- confusionMatrix(predictions,
test_data[[target_variable]])$overall["Accuracy"]
  print(paste("Accuracy with KNN:", accuracy))

  # confusion matrix
  conf_matrix <- confusionMatrix(predictions, test_data[[target_variable]])

  # return the trained model and evaluation results
  return(list(model = knn_model, accuracy = accuracy, confusion_matrix = conf_matrix))
}
```

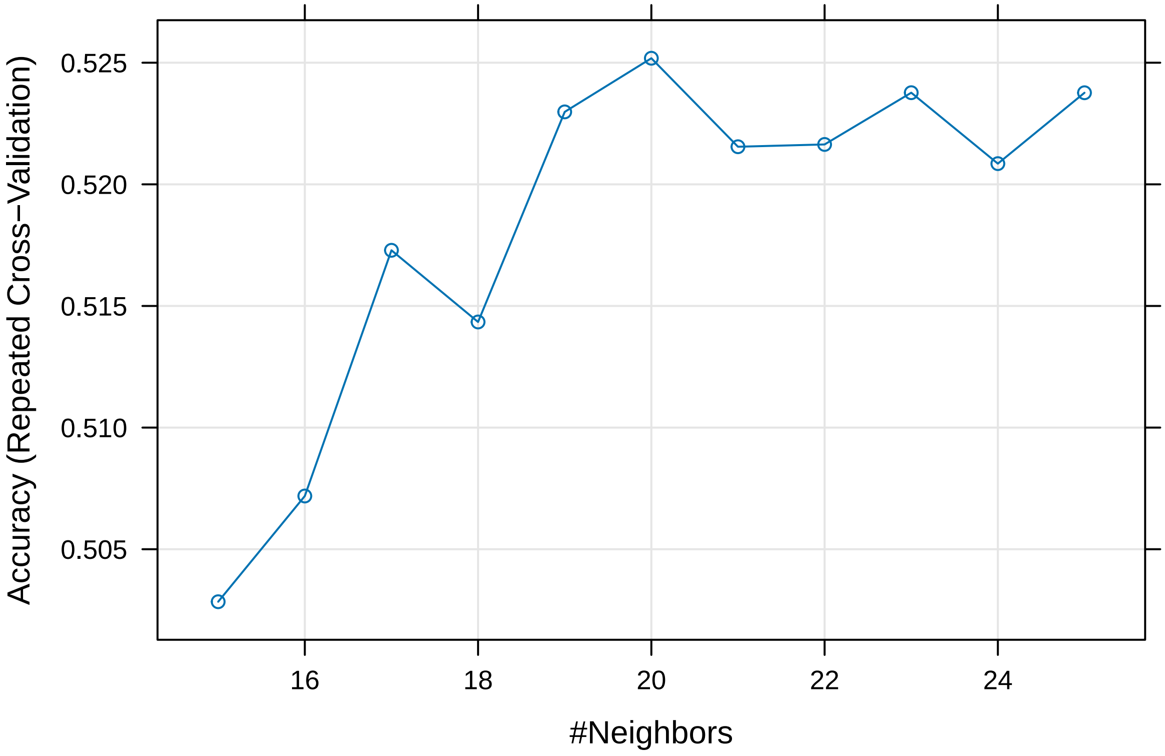
```r
# setting the target variable name
target_variable <- "Number.of.Doctors.Visited"

# calling KNN function
result_knn <- train_and_evaluate_knn(
  train_data = train_original,
  test_data = test_original,
  target_variable = target_variable,
  k_values = 15:25,
  cv_folds = 10,
  cv_repeats = 3
)
```

```
## [1] "Accuracy with KNN: 0.532994923857868"
```

```r
result_knn$model # trained KNN model
```

```
## k-Nearest Neighbors
##
## 462 samples
##  12 predictor
##   3 classes: '1', '2', '3'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 416, 416, 417, 416, 415, 417, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##   15  0.5028412  -0.002272429
##   16  0.5071877   0.001222483
##   17  0.5172878   0.017044190
##   18  0.5143423   0.005051240
##   19  0.5229783   0.018656165
##   20  0.5251837   0.018813288
##   21  0.5215457   0.012991970
##   22  0.5216403   0.009329478
##   23  0.5237666   0.010949339
##   24  0.5208513   0.001657362
##   25  0.5237652   0.009116075
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 20.
```

```
plot(result_knn$model)
```



The most effective fold = 20.

```
result_knn$confusion_matrix # confusion matrix of predictions
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction   1   2   3
##          1   0   0   2
##          2  37 101  51
##          3   1   1   4
##
## Overall Statistics
##
##                Accuracy : 0.533
##                  95% CI : (0.4607, 0.6042)
##     No Information Rate : 0.5178
##     P-Value [Acc > NIR] : 0.3611
##
##                   Kappa : 0.0517
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity           0.00000  0.99020  0.07018
## Specificity           0.98742  0.07368  0.98571
## Pos Pred Value        0.00000  0.53439  0.66667
## Neg Pred Value        0.80513  0.87500  0.72251
## Prevalence            0.19289  0.51777  0.28934
## Detection Rate        0.00000  0.51269  0.02030
## Detection Prevalence  0.01015  0.95939  0.03046
## Balanced Accuracy     0.49371  0.53194  0.52794
```

```r
result_knn$accuracy # accuracy of the KNN model
```

```
##  Accuracy
## 0.5329949
```

Accuracy improved with KNN, with the number of neighbors selected at fold 20. Once again, we see that the model is good at predicting the “2-3 doctors category”

```r
# Updating model_results tibble
model_results <- model_results %>%
  add_row(Model = "KNN", Accuracy = result_knn$accuracy)

kable(model_results)
```

Model	Accuracy
Random Forest	0.4923858
Random Forest w/Feature Selection	0.4720812
KNN	0.5329949

Both models have done well classifying the “2-3 doctors” level. But if we remember from the earlier data exploration, this data set is imbalanced and has a much higher number of observations in the”2-3 doctors” level. Thus this author suspects the models are biased and overfitting.

Blanaced Weights

To remove bias from the models, balanced class weights will be added. This will be done by random sampling to provide a balanced number of observations per Number.of.Doctors.Visited level, based on the smallest level.

```
# create manual balanced sampling
class_levels <- levels(train_original$Number.of.Doctors.Visited)
# choosing the minimum class size for balanced sampli
sample_size <- min(table(train_original$Number.of.Doctors.Visited)) # choosing the
minimum class size for balanced sampling

#create empty df top store data
balanced_data <- NULL
#loop thru each class level in class_levels
for (level in class_levels) {
  #subset original data for class level
  level_data <- train_original[train_original$Number.of.Doctors.Visited == level, , drop
= FALSE]
  #randomly sampling rows from subet data & combining
  balanced_data <- rbind(balanced_data, level_data[sample(1:nrow(level_data),
sample_size), ])
}

# shuffle the balanced data
balanced_data <- balanced_data[sample(nrow(balanced_data)), ]

# confirm new distribution
table(balanced_data$Number.of.Doctors.Visited)
```

```
##
##  1  2  3
## 84 84 84
```

The training set is now balanced. This is only performed on the training set, as the aim of the test set is to mimic real world data, where balance will not necessarily occur.

Now the models will be trained on the balanced_data set.

Balanced Weights: Random Forest

```
to_exclude <- c(1) # index of the target variable to exclude

# calling function
result_RF_balanced <- fit_random_forest(
  train_data = balanced_data,
  test_data = test_original,
  target_variable = "Number.of.Doctors.Visited",
  to_exclude = to_exclude
)
```

```
# results
result_RF_balanced$model # trained Random Forest model
```

```
##
## Call:
##  randomForest(x = x_train, y = y_train, ntree = ntree)
##                Type of random forest: classification
##                      Number of trees: 50000
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 59.13%
## Confusion matrix:
##    1  2  3 class.error
## 1 32 32 20   0.6190476
## 2 30 31 23   0.6309524
## 3 21 23 40   0.5238095
```

```
result_RF_balanced$confusion_matrix # confusion matrix
```

```
##    y_pred
##      1  2  3
##   1 15 16  7
##   2 42 33 27
##   3 20 13 24
```

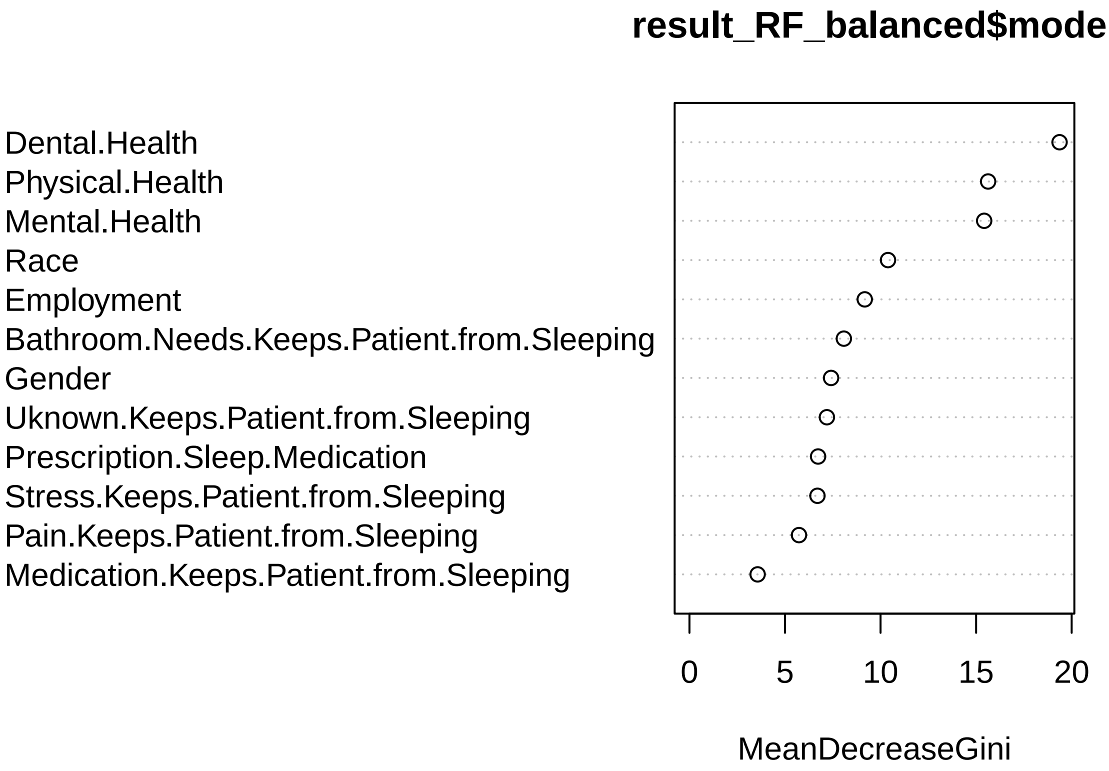
```
result_RF_balanced$accuracy # accuracy of the model
```

```
## [1] 0.3654822
```

```
# importance plot
importance(result_RF_balanced$model)
```

```
##                                            MeanDecreaseGini
## Physical.Health                                   15.628007
## Mental.Health                                     15.422335
## Dental.Health                                     19.366512
## Employment                                         9.173261
## Stress.Keeps.Patient.from.Sleeping                 6.697724
## Medication.Keeps.Patient.from.Sleeping             3.568472
## Pain.Keeps.Patient.from.Sleeping                   5.732190
## Bathroom.Needs.Keeps.Patient.from.Sleeping         8.079294
## Uknown.Keeps.Patient.from.Sleeping                 7.191700
## Prescription.Sleep.Medication                      6.731064
## Race                                              10.390791
## Gender                                             7.410429
```

```
# variable importance plot
varImpPlot(result_RF_balanced$model)
```



We see a decrease in accuracy, which confirms that the first iteration of the model was biased towards the "2-3 doctors" observations. Important features did not change from the initial model. These will now be added in and trained on the balanced data.

```r
# Updating the model_results tibble to add a new column
model_results <- model_results %>%
  mutate(`Accuracy w/Balanced Weights` = NA_real_) # Initialize a new column with NA

# assign  accuracy value for "Random Forest" rows in the new column
model_results$`Accuracy w/Balanced Weights`[model_results$Model == "Random Forest"] <-
result_RF_balanced$accuracy

# Print
kable(model_results)
```

Model	Accuracy	Accuracy w/Balanced Weights
Random Forest	0.4923858	0.3654822
Random Forest w/Feature Selection	0.4720812	NA
KNN	0.5329949	NA

Balanced Weights: Random Forest w/Feature Selection

```r
# extracting feature importance scores
feature_importance <- importance(result_RF_balanced$model)

important_features <- names(sort(feature_importance[, "MeanDecreaseGini"], decreasing =
TRUE))

# selecting top n important features
n <- 5
selected_features <- important_features[1:n]


# subset the training and test data with selected features
train_data_selected <- balanced_data[, c("Number.of.Doctors.Visited", selected_features)]

test_data_selected <- test_original[, c("Number.of.Doctors.Visited", selected_features)]



# calling function
result_features_balanced <- fit_random_forest(
  train_data = train_data_selected,
  test_data = test_data_selected,
  target_variable = "Number.of.Doctors.Visited",
  to_exclude = to_exclude
)

# results
result_features_balanced$model # trained Random Forest model
```

```
## 
## Call:
##  randomForest(x = x_train, y = y_train, ntree = ntree) 
##                Type of random forest: classification
##                      Number of trees: 50000
## No. of variables tried at each split: 2
## 
##         OOB estimate of  error rate: 59.52%
## Confusion matrix:
##    1  2  3 class.error
## 1 30 27 27   0.6428571
## 2 29 37 18   0.5595238
## 3 24 25 35   0.5833333
```

```r
result_features_balanced$confusion_matrix #  confusion matrix
```

```
##    y_pred
##      1  2  3
##   1 11 16 11
##   2 21 41 40
##   3 14 18 25
```

```r
result_features_balanced$accuracy # accuracy of the model
```

```
## [1] 0.3908629
```

With a balanced training set, the random forest model accuracy has improved with feature selection, which we did not see previously.

```r
# add the second balanced weight value for the "Random Forest" model
# assign the accuracy value for "Random Forest" rows in the new column
model_results$`Accuracy w/Balanced Weights`[model_results$Model == "Random Forest
w/Feature Selection"] <- result_features_balanced$accuracy

# Print updated model_results tibble
print(model_results)
```

```
## # A tibble: 3 x 3
##   Model                            Accuracy `Accuracy w/Balanced Weights`
##   <chr>                               <dbl>                         <dbl>
## 1 Random Forest                       0.492                         0.365
## 2 Random Forest w/Feature Selection   0.472                         0.391
## 3 KNN                                 0.533                        NA
```

Balanced Weights: KNN

Finally we test on the KNN model.

```r
# setting target variable
target_variable <- "Number.of.Doctors.Visited"

# calling function
result_knn_balanced <- train_and_evaluate_knn(
  train_data = balanced_data,
  test_data = test_original,
  target_variable = target_variable,
  k_values = 15:25,
  cv_folds = 10,
  cv_repeats = 3
)
```

```
## [1] "Accuracy with KNN: 0.406091370558376"
```

```r
# results
result_knn_balanced$model # trained KNN model
```

```
## k-Nearest Neighbors
##
## 252 samples
##  12 predictor
##   3 classes: '1', '2', '3'
```

```
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 227, 227, 227, 228, 227, 227, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##   15  0.4140556  0.12083128
##   16  0.4219444  0.13259296
##   17  0.4237018  0.13517211
##   18  0.4168086  0.12488892
##   19  0.4129240  0.11880817
##   20  0.3970532  0.09564363
##   21  0.4038390  0.10577604
##   22  0.4141505  0.12143555
##   23  0.4262104  0.13922084
##   24  0.4247825  0.13713680
##   25  0.4184278  0.12773607
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 23.
```

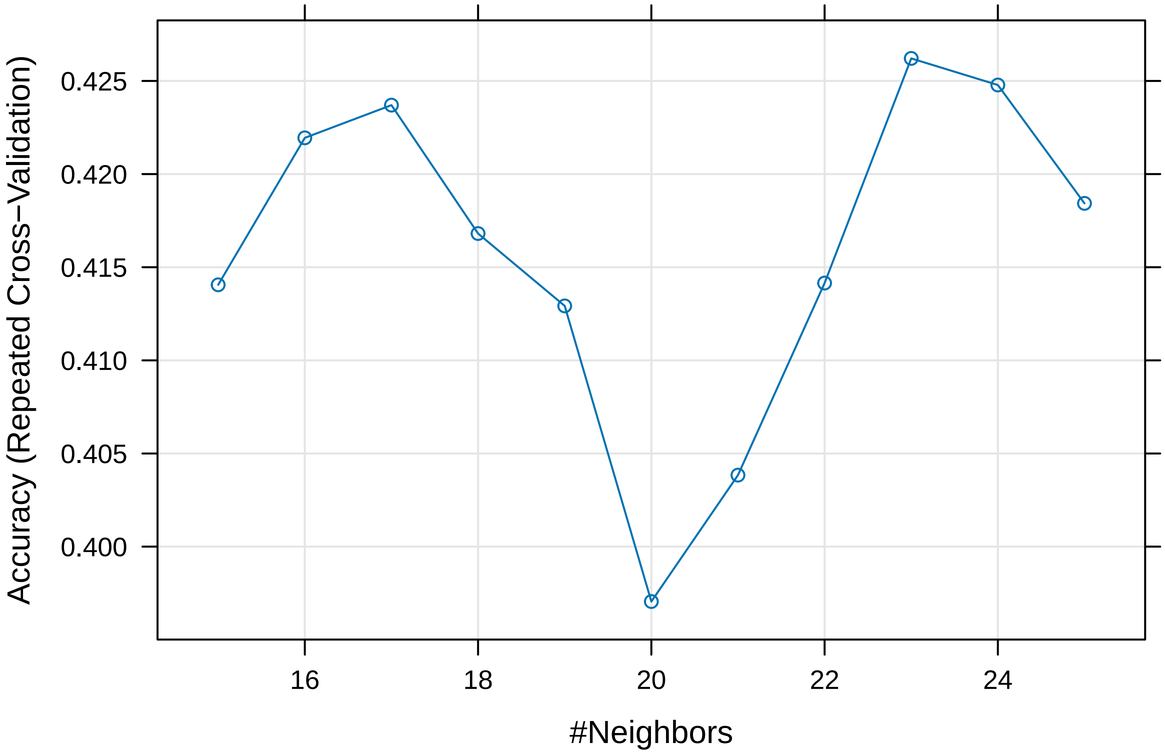
```
result_knn_balanced$accuracy # accuracy of the KNN model
```

```
##  Accuracy
## 0.4060914
```

```
result_knn_balanced$confusion_matrix # confusion matrix of prediction
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3
##          1 17 36 19
##          2 17 40 15
##          3  4 26 23
##
## Overall Statistics
##
##                Accuracy : 0.4061
##                  95% CI : (0.3369, 0.4782)
##     No Information Rate : 0.5178
##     P-Value [Acc > NIR] : 0.9993432
##
##                   Kappa : 0.1034
##
##  Mcnemar's Test P-Value : 0.0002109
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity           0.44737   0.3922   0.4035
## Specificity          0.65409   0.6632   0.7857
## Pos Pred Value       0.23611   0.5556   0.4340
## Neg Pred Value       0.83200   0.5040   0.7639
## Prevalence           0.19289   0.5178   0.2893
## Detection Rate       0.08629   0.2030   0.1168
## Detection Prevalence 0.36548   0.3655   0.2690
## Balanced Accuracy    0.55073   0.5277   0.5946
```

```
plot(result_knn_balanced$model)
```



Applying the balanced data to the KNN model shows a slight increase in accuracy, and an increase in the K-fold value used for the model to 23. Thus we have found a sufficient model to continue future work on.

```
# adding the balanced weight for KNN
# assign the accuracy value for "KNN" rows in the new column
model_results$`Accuracy w/Balanced Weights`[model_results$Model == "KNN"] <-
result_knn_balanced$accuracy

# Print
kable(model_results)
```

Model	Accuracy	Accuracy w/Balanced Weights
Random Forest	0.4923858	0.3654822

Model	Accuracy	Accuracy w/Balanced Weights
Random Forest w/Feature Selection	0.4720812	0.3908629
KNN	0.5329949	0.4060914

Conclusion

The KNN with a balanced training model data preformed the best. Key features that may relate to the target variable Number.of.Doctors.Visited included, Physical.Health, Mental.Health, Dental.Health, and Employment. While accuracy of this model was not above 50%, this does provide a starting point for future work.

Future work should include a larger subset of the original survey data, as this author believes the small portion of features does not tell the whole story behind number of doctors visited. Factors such as socioeconomic status, medical insurance, geographic location, and etc., all contribute to the complexity of healthcare decisions and utilization. Likewise, an entire column (Trouble.Sleeping) had to be removed because of incorrect data. This author considered creating an additional column that provided a similar metric (i.e. if any of the preceding sleep questions = 1, then return 1 for sleep troubles, otherwise 0). This author decided against this step to avoid assumptions, as a respondent could theoretically answer no to all the sleep questions, but still answer yes to Trouble.Sleeping.

Future work on the models themselves should include additional tuning and parameter selection, for example, the Random Forest method could have specified the number of variables to randomly sample(mtry) at each split, and/or tested larger forests.This author was limited by her computer, as it could not process a forest greater than 50000 trees. Further work should be done on a system that can handle large complex data. Additional models should also be investigated, such as XGBoost, which is a distributed gradient boosting library, that combines predictions of multiple weak models to produce strong predictions. This analysis is limited to only Random Forest and KNN methods, and additional methods may prove to be more accurate.

References

- Malani, Preeti N., Kullgren, Jeffrey, and Solway, Erica. National Poll on Healthy Aging (NPHA), [United States], April 2017.Inter-university Consortium for Political and Social Research [distributor], 2019-05-29. https://doi.org/10.3886/ICPSR37305.v1
- Ginter E, Simko V. Women live longer than men. Bratisl Lek Listy. 2013;114(2):45-9. doi: 10.4149/bll_2013_011. PMID: 23331196.
- "Lectur15." Web.pdx.edu, web.pdx.edu/newsomj/pa551/lectur15.htm#::text=Cramer.
- "Random Forest Approach for Classification in R Programming." GeeksforGeeks, 5 July 2020, www.geeksforgeeks.org/random-forest-approach-for-classification-in-r-programming/#. Accessed 15 Apr. 2024.
- "Machine Learning - Random Forest - Q." Wiki.q-Researchsoftware.com, 20 Oct. 2023, wiki.q-researchsoftware.com/wiki/Machine_Learning_-_Random_Forest. Accessed 15 Apr. 2024.
- "Knn Function - RDocumentation." Www.rdocumentation.org,www.rdocumentation.org/packages/class/versions/7.3-22/topics/knn.
- "XGBoost." GeeksforGeeks, 18 Sept. 2021, www.geeksforgeeks.org/xgboost/#. Accessed 15 Apr. 2024.