Report-Pilot-Model

S M Mustaquim, Solaiman Emon, James Arthur, Ocansey Thompson

2023-11-21

Data Preparation and Cleaning

```r
library(readxl)
library(dplyr)
library(tidyr)

demographics1 <- read_excel("Non-veterans.xlsx", sheet = "Demographics")
via1 <- read_excel("Non-veterans.xlsx", sheet = "VIA Results")

demographics2 <- read_excel("Veterans.xlsx", sheet = "Demographics")
via2 <- read_excel("Veterans.xlsx", sheet = "VIA Results")

nonvet <- merge(demographics1, via1, by = "Session ID", all = TRUE)
vet <- merge(demographics2, via2, by = "Session ID", all = TRUE)

# Add a new column to the vet dataset to indicate the veteran status
vet <- vet %>% mutate(`Are you a veteran?` = 'Yes')

# Add a new column to the vet dataset to indicate the non-veteran status
nonvet <- nonvet %>% mutate(`Are you a veteran?` = 'No')
```

Now, we find common columns and stack the dataframes on top of each other.

```r
#Find the common columns
common_columns <- intersect(names(vet), names(nonvet))

# Extract only the common columns from each dataset
vet_common <- vet[, common_columns, drop = FALSE]
nonvet_common <- nonvet[, common_columns, drop = FALSE]

# Stack the datasets on top of each other
merged_data <- rbind(vet_common, nonvet_common)

# Check if all session IDs are unique
all_unique <- !duplicated(merged_data$Session_ID)

cat("Are all session IDs unique? ", all(all_unique), "\n")
```

```
## Are all session IDs unique?  TRUE
```

Now, we filter the data for US locations.

```
# Extract the column "Where are you located?"
location_column <- merged_data$`Where are you located?`

# Use the table function to get the counts of each unique category
location_counts <- as.data.frame(table(location_column))

# Create a filter for U.S. locations
us_filter <- grepl("United States|USA|Alabama|Alaska|Arizona|Arkansas|California|Colorado|Connecticut|De

# Apply the filter to the dataframe
us_locations <- merged_data[us_filter, ]
```

Now, we break down the disability column into specific disabilities.

```
# Extract the column "Are you experiencing any of the following?"
symptoms_column <- us_locations$`Are you experiencing any of the following? (Please check all that apply

#unique symptoms and their counts
symptoms_counts <- as.data.frame(table(symptoms_column))


# Replace "N/A" with actual NA values
us_locations <- us_locations %>%
  mutate(symptoms_column = na_if(symptoms_column, "N/A"))


#Now we create binary variable for each disability symptoms and fill them true or false
#according to the observed symptoms.

symptoms_list <- strsplit(as.character(symptoms_column), ", ")

unique_symptoms <- unique(unlist(symptoms_list))

for (symptom in unique_symptoms) {
  us_locations[paste0("Symptom_", symptom)] <- sapply(symptoms_list, function(x) symptom %in% x)
}

#now, this is our master dataset

dataset_final <- us_locations

#write.csv(dataset_final, "dataset_final.csv", row.names = FALSE)
```

A simple chi-squared test to see if there is association between homelessness and veteran status.

```
unique_values <- unique(dataset_final[, 13])
occurrences <- table(dataset_final[, 13])

# Display unique values and their occurrences
for (value in unique_values) {
  print(paste("Value:", value, ", Occurrences:", occurrences[value]))
}
```

```
## [1] "Value: NO , Occurrences: 33876"
## [1] "Value: YES , Occurrences: 369"
## [1] "Value: N/A , Occurrences: 344"
```

```r
# Let us delete the missing values

# Replace variations of missing values with standard NA representation
dataset_final$`Are you currently experiencing homelessness?` <- na_if(dataset_final$`Are you currently e
dataset_final$`Are you currently experiencing homelessness?` <- na_if(dataset_final$`Are you currently e

# Remove rows with missing values
dataset_final <- dataset_final[!is.na(dataset_final$`Are you currently experiencing homelessness?`), ]

# Chi-Squared Test

# Create a contingency table
contingency_table_homelessness <- table(dataset_final$`Are you a veteran?`, dataset_final$`Are you curre

# Display the contingency table
print(contingency_table_homelessness)
```

```
##
##         NO   YES
##   No  11378    39
##   Yes 22498   330
```

```r
# Perform the chi-squared test
chi_square_result_homelessness <- chisq.test(contingency_table_homelessness)

# Print the chi-squared test result
print(chi_square_result_homelessness)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  contingency_table_homelessness
## X-squared = 85.99, df = 1, p-value < 2.2e-16
```

```r
if (chi_square_result_homelessness$p.value < 0.05) {
  print("There is a significant association between veteran status and homelessness.")
} else {
  print("There is no significant association between veteran status and homelessness.")
}
```

```
## [1] "There is a significant association between veteran status and homelessness."
```

Also, we show the prevalence of homelessness in both veterans and nonveterans.

```r
# Get the contingency table
contingency_table_homelessness <- table(dataset_final$`Are you a veteran?`, dataset_final$`Are you curre
```

```r
# Extract counts from the contingency table
count_homeless_veterans <- contingency_table_homelessness["Yes", "YES"]
count_homeless_civilians <- contingency_table_homelessness["No", "YES"]

# Calculate total counts in each group
total_veterans <- sum(contingency_table_homelessness["Yes", ])
total_civilians <- sum(contingency_table_homelessness["No", ])

# Calculate prevalence
prevalence_homeless_veterans <- count_homeless_veterans / total_veterans
prevalence_homeless_civilians <- count_homeless_civilians / total_civilians

# Display the results
print(paste("Prevalence of homelessness among veterans: ", round(prevalence_homeless_veterans * 100, 2)
```

```
## [1] "Prevalence of homelessness among veterans:  1.45 %"
```

```r
print(paste("Prevalence of homelessness among civilians: ", round(prevalence_homeless_civilians * 100, 
```

```
## [1] "Prevalence of homelessness among civilians:  0.34 %"
```

Now, we do some simple data cleaning before we apply our model.

```r
# The types of variables in the dataset

column_types <- sapply(dataset_final, class)
print(column_types)

# Find the occurrences of each unique value in the column
occurrences <- table(dataset_final$`How many children under 18 years old live in your household?`)

# Display the occurrences
print(occurrences)


# Convert the result to a data frame
occurrences_df <- as.data.frame(occurrences)

# Display the occurrences data frame
print(occurrences_df)


# Extract the column to be cleaned
children_column <- dataset_final$`How many children under 18 years old live in your household?`

# Replace "None" and "One" with NA
children_column[children_column %in% c("None", "One")] <- NA

# Convert the column to numeric
children_column <- as.numeric(children_column)

# Replace values less than 0 or greater than 20 with NA
```

```r
children_column[children_column < 0 | children_column > 20] <- NA

# Impute missing values with the mean
mean_value <- mean(children_column, na.rm = TRUE)
children_column[is.na(children_column)] <- mean_value

# Round the values to the nearest integer
children_column <- round(children_column)

# Create a new column in the dataframe with cleaned and imputed values
dataset_final$`Cleaned_Children_Column` <- children_column

#---------------------------------------------------------------------------

date_of_birth_column <- dataset_final$`What is your date of birth?`

# Convert the character column to Date format
date_of_birth_column <- as.Date(date_of_birth_column)

# Specify today's date
current_date <- as.Date("2023-11-20")  # Replace with the current date in "YYYY-MM-DD" format

# Calculate ages based on the specified current date
ages_in_days <- as.integer(difftime(current_date, date_of_birth_column, units = "days"))
ages_in_years <- ages_in_days / 365.25  # Considering leap years

# Impute missing values with the mean age
mean_age <- mean(ages_in_years, na.rm = TRUE)
ages_in_years[is.na(ages_in_years)] <- mean_age

# Round ages to the nearest integer
ages_in_years <- round(ages_in_years)

# Create a new column in the dataframe with ages
dataset_final$`Age_Column` <- ages_in_years


#------------------------------------------------------------------------

column_name <- 'What is your current employment status?'

# Calculate the mode of the column
mode_value <- names(sort(table(dataset_final[[column_name]]), decreasing = TRUE))[1]

# Impute missing values with the mode
dataset_final[[column_name]] <- ifelse(is.na(dataset_final[[column_name]]), mode_value, dataset_final[[

#--------------------------------------------------------------------------

column_name_education <- 'What is your highest level of education?'

# Calculate the mode of the column
mode_value_education <- names(sort(table(dataset_final[[column_name_education]]), decreasing = TRUE))[1]
```

```r
# Impute missing values with the mode
dataset_final[[column_name_education]] <- ifelse(is.na(dataset_final[[column_name_education]]), mode_val

#---------------------------------------------------------------------------
column_name_gender <- 'How do you identify your gender?'

# Calculate the mode of the column
mode_value_gender <- names(sort(table(dataset_final[[column_name_gender]]), decreasing = TRUE))[1]

# Impute missing values with the mode
dataset_final[[column_name_gender]] <- ifelse(is.na(dataset_final[[column_name_gender]]), mode_value_ge

#-----------------------------------------------------------------------------

column_name_marital_status <- 'What is your current marital status?'

# Calculate the mode of the column
mode_value_marital_status <- names(sort(table(dataset_final[[column_name_marital_status]]), decreasing =

# Impute missing values with the mode
dataset_final[[column_name_marital_status]] <- ifelse(is.na(dataset_final[[column_name_marital_status]]

#-----------------------------------------------------------------------------

column_name_income <- 'What is your total annual household income'

# Calculate the proportions of each category
income_category_proportions <- table(dataset_final[[column_name_income]]) / sum(!is.na(dataset_final[[c

# Impute missing values by sampling from existing categories
dataset_final[[column_name_income]] <- ifelse(
  is.na(dataset_final[[column_name_income]]),
  sample(names(income_category_proportions), sum(is.na(dataset_final[[column_name_income]])), replace =
  dataset_final[[column_name_income]]
)

#----------------------------------------------------------------------------

missing_percentage <- colMeans(is.na(dataset_final)) * 100

# Display the result
print(missing_percentage)


#----------------------------------------------------------------------------

write.csv(dataset_final, "dataset_final.csv", row.names = FALSE)


#----------------------------------------------------------------------------

data_types <- sapply(dataset_final, class)
print(data_types)
```

```
columns_to_convert <- c(14:37, 52, 53)

# Convert selected columns to numeric
dataset_final[, columns_to_convert] <- lapply(dataset_final[, columns_to_convert], as.numeric)

columns_to_convert <- c(2, 6, 7, 8, 11)
dataset_final[, columns_to_convert] <- lapply(dataset_final[, columns_to_convert], as.factor)

#---------------------------------------------------------------------------

columns_to_keep <- c(-1, -3, -4, -5, -9, -10, -12, -39)
dataset <- dataset_final[, columns_to_keep]

data_types <- sapply(dataset, class)
print(data_types)
```

Logistic Regression Model

Now, we apply a simple logistic regression model.

```
# Convert "Are you a veteran?" to factor
dataset$`Are you a veteran?` <- factor(dataset$`Are you a veteran?`)

# Convert "Are you currently experiencing homelessness?" to factor
dataset$`Are you currently experiencing homelessness?` <- factor(dataset$`Are you currently experiencin

# Build a logistic regression model
model <- glm(`Are you currently experiencing homelessness?` ~ `Are you a veteran?` + .,
             data = dataset,
             family = "binomial")
```

Evaluation of the model:

```
# Install and load necessary libraries
# install.packages("caret")
library(caret)

# Make predictions on the original dataset
predicted_probabilities <- predict(model, type = "response")

# Convert probabilities to class predictions
predicted_classes <- ifelse(predicted_probabilities > 0.5, "YES", "NO")

# Evaluate the model using confusion matrix and other metrics
conf_matrix <- confusionMatrix(factor(predicted_classes), dataset$`Are you currently experiencing homel
print(conf_matrix)
```

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

```
## Prediction    NO   YES
##        NO  33845   280
##        YES    31    89
##
##                Accuracy : 0.9909
##                  95% CI : (0.9899, 0.9919)
##     No Information Rate : 0.9892
##     P-Value [Acc > NIR] : 0.001022
##
##                   Kappa : 0.3606
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9991
##             Specificity : 0.2412
##          Pos Pred Value : 0.9918
##          Neg Pred Value : 0.7417
##              Prevalence : 0.9892
##          Detection Rate : 0.9883
##    Detection Prevalence : 0.9965
##       Balanced Accuracy : 0.6201
##
##        'Positive' Class : NO
##
```

The data is highly imbalanced on the response variable. The model could not provide a good prediction on the true YES classes.

So, we apply a weights on imbalanced classes.

Weighted Logistic Regression

```r
# Calculate class weights based on the inverse of class frequencies
class_weights <- ifelse(dataset$`Are you currently experiencing homelessness?` == "YES", 1/sum(dataset$


# Build a logistic regression model with adjusted class weights
model2 <- glm(`Are you currently experiencing homelessness?` ~ `Are you a veteran?` + .,
             data = dataset,
             family = "binomial",
             weights = class_weights)
```

Now, we evaluate this revised model.

```r
library(caret)

# Make predictions on the original dataset using model2
predicted_probabilities <- predict(model2, type = "response")

# Convert probabilities to class predictions
predicted_classes <- ifelse(predicted_probabilities > 0.5, "YES", "NO")
```

```r
# Evaluate the model using confusion matrix and other metrics
conf_matrix <- confusionMatrix(factor(predicted_classes), dataset$`Are you currently experiencing homel
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    NO   YES
##        NO  29314    70
##        YES  4562   299
##
##                Accuracy : 0.8647
##                  95% CI : (0.8611, 0.8683)
##     No Information Rate : 0.9892
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0962
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.86533
##             Specificity : 0.81030
##          Pos Pred Value : 0.99762
##          Neg Pred Value : 0.06151
##              Prevalence : 0.98922
##          Detection Rate : 0.85601
##    Detection Prevalence : 0.85805
##       Balanced Accuracy : 0.83782
##
##        'Positive' Class : NO
##
```

It has better prediction on the yes classes than the previous model.