Capstone Project Pre-Processing and Feature Engineering

Steve Uzupis

2024-07-07

*Packages Load*

# load packages  
  
library(readr) # load csv files  
library(readxl) # load excel files  
library(dplyr) # data manipulation

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(lubridate) # date & time manipulation

##   
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':  
##   
## date, intersect, setdiff, union

library(ggplot2) # data visualization  
library(tidyr) # collection of statistical packages, packages loaded individually  
library(corrplot) # to visualize correlations

## corrplot 0.92 loaded

library(leaps) # for subset selection  
library(caret) # test for correlation

## Loading required package: lattice

library(car) # for VIF

## Loading required package: carData

##   
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':  
##   
## recode

library(scales) # for visualizing plots in %

##   
## Attaching package: 'scales'

## The following object is masked from 'package:readr':  
##   
## col\_factor

library(forcats) # ordering data frames  
library(codebookr) # adding appendix to r code  
library(gtsummary) # creating tables  
library(cardx) # to include statistic results  
library(moments) # to calculate skewness and kurtosis  
library(VIM) # to run K- Nearest Neighbour

## Loading required package: colorspace

## Loading required package: grid

## VIM is ready to use.

## Suggestions and bug-reports can be submitted at: https://github.com/statistikat/VIM/issues

##   
## Attaching package: 'VIM'

## The following object is masked from 'package:datasets':  
##   
## sleep

library(pROC) # to analyse and display reciever operating characteristics(ROC) curvees

## Type 'citation("pROC")' for a citation.

##   
## Attaching package: 'pROC'

## The following object is masked from 'package:colorspace':  
##   
## coords

## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

library(randomForest) # to impletement Random Forest algorithm

## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

## The following object is masked from 'package:dplyr':  
##   
## combine

library(glmnet)

## Loading required package: Matrix

##   
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':  
##   
## expand, pack, unpack

## Loaded glmnet 4.1-8

library(tidymodels)

## ── Attaching packages ────────────────────────────────────── tidymodels 1.2.0 ──

## ✔ broom 1.0.6 ✔ rsample 1.2.1  
## ✔ dials 1.3.0 ✔ tibble 3.2.1  
## ✔ infer 1.0.7 ✔ tune 1.2.1  
## ✔ modeldata 1.4.0 ✔ workflows 1.1.4  
## ✔ parsnip 1.2.1 ✔ workflowsets 1.1.0  
## ✔ purrr 1.0.2 ✔ yardstick 1.3.1  
## ✔ recipes 1.1.0

## ── Conflicts ───────────────────────────────────────── tidymodels\_conflicts() ──  
## ✖ randomForest::combine() masks dplyr::combine()  
## ✖ purrr::discard() masks scales::discard()  
## ✖ Matrix::expand() masks tidyr::expand()  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ✖ purrr::lift() masks caret::lift()  
## ✖ randomForest::margin() masks ggplot2::margin()  
## ✖ Matrix::pack() masks tidyr::pack()  
## ✖ yardstick::precision() masks caret::precision()  
## ✖ recipes::prepare() masks VIM::prepare()  
## ✖ yardstick::recall() masks caret::recall()  
## ✖ car::recode() masks dplyr::recode()  
## ✖ yardstick::sensitivity() masks caret::sensitivity()  
## ✖ purrr::some() masks car::some()  
## ✖ yardstick::spec() masks readr::spec()  
## ✖ yardstick::specificity() masks caret::specificity()  
## ✖ recipes::step() masks stats::step()  
## ✖ Matrix::unpack() masks tidyr::unpack()  
## ✖ recipes::update() masks Matrix::update(), stats::update()  
## • Search for functions across packages at https://www.tidymodels.org/find/

library(recipes)

*Data load*

# load data  
sdoh\_data <- read\_csv("data/sdoh\_data.csv")

## Warning: One or more parsing issues, call `problems()` on your data frame for details,  
## e.g.:  
## dat <- vroom(...)  
## problems(dat)

## Rows: 3229 Columns: 682  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (15): COUNTYFIPS, STATEFIPS, STATE, COUNTY, REGION, CAF\_ADJ\_COUNTY\_1, C...  
## dbl (664): YEAR, TERRITORY, ACS\_TOT\_POP\_WT, ACS\_TOT\_POP\_US\_ABOVE1, ACS\_TOT\_P...  
## lgl (3): CAF\_ADJ\_COUNTY\_12, CAF\_ADJ\_COUNTY\_13, CAF\_ADJ\_COUNTY\_14  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

dim(sdoh\_data)

## [1] 3229 682

# remove unwanted features, create calculated feature, convert fips\_code to data type matching chr\_data  
sdoh\_data <- sdoh\_data %>%   
 select("COUNTYFIPS",   
 "STATE",   
 "COUNTY",   
 "REGION",   
 "ACS\_TOT\_POP\_WT",   
 "ACS\_AVG\_HH\_SIZE",   
 "ACS\_PCT\_MALE",   
 "ACS\_PCT\_AIAN",   
 "ACS\_PCT\_ASIAN",   
 "ACS\_PCT\_BLACK",   
 "ACS\_PCT\_HISPANIC",   
 "ACS\_PCT\_OTHER\_RACE",   
 "ACS\_PCT\_WHITE",   
 "ACS\_PCT\_CHILD\_1FAM",   
 "ACS\_PCT\_CHILDREN\_GRANDPARENT",   
 "ACS\_PCT\_GRANDP\_RESPS\_NO\_P",   
 "ACS\_PCT\_GRANDP\_RESPS\_P",  
 "ACS\_PCT\_HH\_OTHER\_COMP",   
 "ACS\_PCT\_HH\_INTERNET",  
 "ACS\_PCT\_EMPLOYED",  
 "ACS\_PCT\_HH\_INC\_99999",  
 "ACS\_PCT\_MEDICARE\_ONLY",  
 "AHRF\_CLIN\_NURSE\_SPEC\_RATE",   
 "AHRF\_DENTISTS\_RATE",  
 "AHRF\_PHYSICIAN\_ASSIST\_RATE",  
 "AMFAR\_MHFAC\_RATE",  
 "CEN\_POPDENSITY\_COUNTY",  
 "NEPHTN\_HEATIND\_90",  
 "SAIPE\_MEDIAN\_HH\_INCOME",  
 "POS\_MEDIAN\_DIST\_ED",  
 "POS\_MEDIAN\_DIST\_PED\_ICU",  
 "POS\_MEDIAN\_DIST\_CLINIC",   
 "POS\_MEDIAN\_DIST\_ALC",   
 ) %>%   
 mutate(percent\_grandparents\_as\_guardians = ACS\_PCT\_CHILDREN\_GRANDPARENT \* ((ACS\_PCT\_GRANDP\_RESPS\_P + ACS\_PCT\_GRANDP\_RESPS\_NO\_P)/100)) %>%   
 select(-ACS\_PCT\_GRANDP\_RESPS\_P, -ACS\_PCT\_GRANDP\_RESPS\_NO\_P, -ACS\_PCT\_CHILDREN\_GRANDPARENT) %>%   
 rename("fips\_code" = "COUNTYFIPS",  
 "state" = "STATE",  
 "county" = "COUNTY",  
 "region" = "REGION",  
 "weighted\_population" = "ACS\_TOT\_POP\_WT",  
 "average\_hh\_size" = "ACS\_AVG\_HH\_SIZE",  
 "pct\_male" = "ACS\_PCT\_MALE",  
 "pct\_native\_american" = "ACS\_PCT\_AIAN",  
 "pct\_asian" = "ACS\_PCT\_ASIAN",  
 "pct\_black" = "ACS\_PCT\_BLACK",  
 "pct\_hispanic" = "ACS\_PCT\_HISPANIC",  
 "pct\_other\_race" = "ACS\_PCT\_OTHER\_RACE",  
 "pct\_white" = "ACS\_PCT\_WHITE",  
 "pct\_single\_parent" = "ACS\_PCT\_CHILD\_1FAM",  
 "pct\_hh\_other\_computer" = "ACS\_PCT\_HH\_OTHER\_COMP",  
 "pct\_hh\_internet" = "ACS\_PCT\_HH\_INTERNET",  
 "pct\_employed" = "ACS\_PCT\_EMPLOYED",  
 "pct\_hh\_inc\_99999" = "ACS\_PCT\_HH\_INC\_99999", # renamed by mg  
 "pct\_w\_medicare" = "ACS\_PCT\_MEDICARE\_ONLY",  
 "clinical\_nurse\_pt" = "AHRF\_CLIN\_NURSE\_SPEC\_RATE",  
 "dentist\_pt" = "AHRF\_DENTISTS\_RATE",  
 "pa\_pt" = "AHRF\_PHYSICIAN\_ASSIST\_RATE",  
 "mental\_health\_faciliy\_pt" = "AMFAR\_MHFAC\_RATE",  
 "population\_density" = "CEN\_POPDENSITY\_COUNTY",  
 "days\_over\_90\_f" = "NEPHTN\_HEATIND\_90",  
 "median\_hh\_income" = "SAIPE\_MEDIAN\_HH\_INCOME",  
 "median\_er\_dist" = "POS\_MEDIAN\_DIST\_ED",  
 "median\_pediatric\_icu\_dist" = "POS\_MEDIAN\_DIST\_PED\_ICU",  
 "median\_health\_clinic\_dist" = "POS\_MEDIAN\_DIST\_CLINIC",  
 "median\_drug\_alcohol\_care\_dist" = "POS\_MEDIAN\_DIST\_ALC"  
   
 ) %>%   
 mutate(fips\_code = as.numeric(fips\_code))  
  
chr\_data <- read\_csv("data/chr\_data.csv", skip = 1)

## Rows: 3194 Columns: 720  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (2): state, county  
## dbl (572): statecode, countycode, fipscode, year, county\_ranked, v001\_rawval...  
## lgl (146): v002\_numerator, v002\_denominator, v036\_numerator, v036\_denominato...  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

dim(chr\_data)

## [1] 3194 720

# remove unwanted features  
# convert principal care providers from per 100,000 people to per 1,000 people to match other data  
  
chr\_data <- chr\_data %>%  
 select("fipscode",  
 "v002\_rawvalue",  
 "v009\_rawvalue",  
 "v011\_rawvalue",  
 "v070\_rawvalue",   
 "v049\_rawvalue",  
 "v085\_rawvalue",  
 "v168\_rawvalue",   
 "v069\_rawvalue",  
 "v044\_rawvalue",   
 "v140\_rawvalue",  
 "v125\_rawvalue",  
 "v124\_rawvalue",  
 "v136\_other\_data\_1",  
 "v136\_other\_data\_2",  
 "v137\_rawvalue",  
 "v147\_rawvalue",  
 "v139\_rawvalue",  
 "v177\_rawvalue",  
 "v153\_rawvalue",  
 "v053\_rawvalue",   
 "v058\_rawvalue",   
 "v004\_rawvalue",  
 ) %>%   
 mutate(pcp\_pt = v004\_rawvalue/100) %>%   
 select(-v004\_rawvalue) %>%   
 rename("fips\_code" = "fipscode",  
 "pct\_poor\_to\_fair\_health" = "v002\_rawvalue",  
 "pct\_adult\_smokers" = "v009\_rawvalue",  
 "pct\_obese\_adults" = "v011\_rawvalue",  
 "pct\_no\_exercise" = "v070\_rawvalue",  
 "pct\_binge\_drinkers" = "v049\_rawvalue",  
 "pct\_under\_65\_no\_health\_insurance" = "v085\_rawvalue",  
 "pct\_highschool\_diploma" = "v168\_rawvalue",  
 "pct\_some\_college" = "v069\_rawvalue",  
 "inequality\_ratio" = "v044\_rawvalue",  
 "social\_clubs\_per\_10k" = "v140\_rawvalue",  
 "air\_polution\_metric" = "v125\_rawvalue",  
 "water\_quality" = "v124\_rawvalue", # renamed by mg  
 "pct\_high\_housing\_costs" = "v136\_other\_data\_1",  
 "pct\_overcrowded\_hh" = "v136\_other\_data\_2",  
 "pct\_food\_insecurities" = "v139\_rawvalue",  
 "pct\_voters" = "v177\_rawvalue",  
 "pct\_home\_owner" = "v153\_rawvalue",  
 "pct\_65\_plus" = "v053\_rawvalue",  
 "pct\_rural\_population" = "v058\_rawvalue",  
 "life\_expectancy\_years" = "v147\_rawvalue",  
 "pct\_30\_min\_plus\_commute" = "v137\_rawvalue")  
  
# full data sets are extremely large, initial dimension reduction was performed previously

*Combine datasets*

# Create and clean the qol\_data dataset  
qol\_data <- sdoh\_data %>%  
 inner\_join(chr\_data, by = "fips\_code") %>%  
 mutate(response = ifelse(pct\_poor\_to\_fair\_health >= 0.154, "worse", "better")) %>%  
 mutate(response = as.factor(response)) %>%  
 #select(-pct\_poor\_to\_fair\_health) %>% # keep until analysis has been performed  
 mutate\_at(vars(state, county, region), as.factor) # convert characters to factors

eliminate variables with no predictive value: fipscode, county and state, pct\_poor\_to\_fair\_health:

qol\_data <- qol\_data %>% select(-c(fips\_code,county,state,pct\_poor\_to\_fair\_health))

calcSplitRatio <- function(p = NA, df) {  
 ## @p = the number of parameters. by default, if none are provided, the number of columns (predictors) in the dataset are used  
 ## @df = the dataframe that will be used for the analysis  
   
 ## If the number of parameters isn't supplied, set it to the number of features minus 1 for the target  
 if(is.na(p)) {  
 p <- ncol(df) -1 ## COMMENT HERE  
 }  
   
 ## Calculate the ideal number of testing set  
 test\_N <- (1/sqrt(p))\*nrow(df)  
 ## Turn that into a testing proportion  
 test\_prop <- round((1/sqrt(p))\*nrow(df)/nrow(df), 2)  
 ## And find the training proportion  
 train\_prop <- 1-test\_prop  
   
 ## Tell us the results!  
 print(paste0("The ideal split ratio is ", train\_prop, ":", test\_prop, " training:testing"))  
   
 ## Return the size of the training set  
 return(train\_prop)  
}  
  
calcSplitRatio(df = qol\_data)

## [1] "The ideal split ratio is 0.86:0.14 training:testing"

## [1] 0.86

# Find columns with missing data  
colSums(is.na(qol\_data))

## region weighted\_population   
## 0 1   
## average\_hh\_size pct\_male   
## 1 1   
## pct\_native\_american pct\_asian   
## 1 1   
## pct\_black pct\_hispanic   
## 1 1   
## pct\_other\_race pct\_white   
## 1 1   
## pct\_single\_parent pct\_hh\_other\_computer   
## 2 1   
## pct\_hh\_internet pct\_employed   
## 1 1   
## pct\_hh\_inc\_99999 pct\_w\_medicare   
## 1 1   
## clinical\_nurse\_pt dentist\_pt   
## 0 0   
## pa\_pt mental\_health\_faciliy\_pt   
## 0 0   
## population\_density days\_over\_90\_f   
## 1 34   
## median\_hh\_income median\_er\_dist   
## 2 1   
## median\_pediatric\_icu\_dist median\_health\_clinic\_dist   
## 13 1   
## median\_drug\_alcohol\_care\_dist percent\_grandparents\_as\_guardians   
## 1 40   
## pct\_adult\_smokers pct\_obese\_adults   
## 2 2   
## pct\_no\_exercise pct\_binge\_drinkers   
## 2 2   
## pct\_under\_65\_no\_health\_insurance pct\_highschool\_diploma   
## 1 0   
## pct\_some\_college inequality\_ratio   
## 0 7   
## social\_clubs\_per\_10k air\_polution\_metric   
## 0 27   
## water\_quality pct\_high\_housing\_costs   
## 43 0   
## pct\_overcrowded\_hh pct\_30\_min\_plus\_commute   
## 0 0   
## life\_expectancy\_years pct\_food\_insecurities   
## 70 0   
## pct\_voters pct\_home\_owner   
## 30 0   
## pct\_65\_plus pct\_rural\_population   
## 0 7   
## pcp\_pt response   
## 147 2

# Delete rows with missing values in the "response" column  
qol\_data <- qol\_data[!is.na(qol\_data$response), ]  
# delete column with large number of missing values 'pcp\_pt'  
  
qol\_data <- qol\_data[, !names(qol\_data) %in% "pcp\_pt"]  
  
set.seed(112)  
# Splitting the data into train and test sets  
#qol\_train <- qol\_data[sample(nrow(qol\_data), 0.86\*nrow(qol\_data)), ]  
#qol\_test <- qol\_data[!(rownames(qol\_data) %in% rownames(qol\_train)), ]  
  
data\_split <- initial\_split(qol\_data, strata = "response", prop = 0.86)  
  
qol\_train <- training(data\_split)  
qol\_test <- testing(data\_split)  
  
  
# Selecting only numeric columns for imputation  
numeric\_cols <- sapply(qol\_train, is.numeric)  
qol\_train\_numeric <- qol\_train[, numeric\_cols]  
qol\_test\_numeric <- qol\_test[, numeric\_cols]  
  
# Imputing missing values using mice  
library(mice)

##   
## Attaching package: 'mice'

## The following object is masked from 'package:stats':  
##   
## filter

## The following objects are masked from 'package:base':  
##   
## cbind, rbind

# Imputing missing values in the training set  
qol\_train\_imputed <- mice(qol\_train\_numeric, m = 1, method = "cart", # Using CART for imputation  
 maxit = 5) # Maximum iterations for imputation

##   
## iter imp variable  
## 1 1 pct\_single\_parent days\_over\_90\_f median\_hh\_income median\_pediatric\_icu\_dist percent\_grandparents\_as\_guardians pct\_under\_65\_no\_health\_insurance inequality\_ratio air\_polution\_metric water\_quality life\_expectancy\_years pct\_voters pct\_rural\_population  
## 2 1 pct\_single\_parent days\_over\_90\_f median\_hh\_income median\_pediatric\_icu\_dist percent\_grandparents\_as\_guardians pct\_under\_65\_no\_health\_insurance inequality\_ratio air\_polution\_metric water\_quality life\_expectancy\_years pct\_voters pct\_rural\_population  
## 3 1 pct\_single\_parent days\_over\_90\_f median\_hh\_income median\_pediatric\_icu\_dist percent\_grandparents\_as\_guardians pct\_under\_65\_no\_health\_insurance inequality\_ratio air\_polution\_metric water\_quality life\_expectancy\_years pct\_voters pct\_rural\_population  
## 4 1 pct\_single\_parent days\_over\_90\_f median\_hh\_income median\_pediatric\_icu\_dist percent\_grandparents\_as\_guardians pct\_under\_65\_no\_health\_insurance inequality\_ratio air\_polution\_metric water\_quality life\_expectancy\_years pct\_voters pct\_rural\_population  
## 5 1 pct\_single\_parent days\_over\_90\_f median\_hh\_income median\_pediatric\_icu\_dist percent\_grandparents\_as\_guardians pct\_under\_65\_no\_health\_insurance inequality\_ratio air\_polution\_metric water\_quality life\_expectancy\_years pct\_voters pct\_rural\_population

# Imputing missing values in the test set using the trained model  
qol\_test\_imputed <- mice(qol\_test\_numeric, m = 1, method = "cart",  
 maxit = 5,  
 use.all = FALSE) # Using only the training data for imputation

##   
## iter imp variable  
## 1 1 days\_over\_90\_f median\_pediatric\_icu\_dist percent\_grandparents\_as\_guardians air\_polution\_metric water\_quality life\_expectancy\_years pct\_voters  
## 2 1 days\_over\_90\_f median\_pediatric\_icu\_dist percent\_grandparents\_as\_guardians air\_polution\_metric water\_quality life\_expectancy\_years pct\_voters  
## 3 1 days\_over\_90\_f median\_pediatric\_icu\_dist percent\_grandparents\_as\_guardians air\_polution\_metric water\_quality life\_expectancy\_years pct\_voters  
## 4 1 days\_over\_90\_f median\_pediatric\_icu\_dist percent\_grandparents\_as\_guardians air\_polution\_metric water\_quality life\_expectancy\_years pct\_voters  
## 5 1 days\_over\_90\_f median\_pediatric\_icu\_dist percent\_grandparents\_as\_guardians air\_polution\_metric water\_quality life\_expectancy\_years pct\_voters

# Combining the imputed numeric columns back with the original data  
qol\_train <- cbind(qol\_train[, !numeric\_cols], complete(qol\_train\_imputed))  
qol\_test <- cbind(qol\_test[, !numeric\_cols], complete(qol\_test\_imputed))  
  
sum(colSums(is.na(qol\_test)))

## [1] 0

Encode character Variables

# use count of areas (state and region) for features that are characters  
# Create dummy variables for 'region' in training data  
#set.seed(1)  
#dummies\_train <- dummyVars(~ region, data = qol\_train, fullRank = FALSE)  
#qol\_train\_encoded <- predict(dummies\_train, newdata = qol\_train)  
#qol\_train <- cbind(qol\_train, qol\_train\_encoded) %>%  
 #select(-region) # Remove the original 'region' column  
  
#set.seed(1)  
# Create dummy variables for 'region' in test data  
#dummies\_test <- dummyVars(~ region, data = qol\_test, fullRank = FALSE)  
#qol\_test\_encoded <- predict(dummies\_test, newdata = qol\_test)  
#qol\_test <- cbind(qol\_test, qol\_test\_encoded) %>%  
 #select(-region) # Remove the original 'region' column  
  
#Encode response variable as 1(worse) and 0(better)   
  
#encode response variable in training set  
#qol\_train <- qol\_train %>%  
 #mutate(response = ifelse(response == "better", 0, 1))  
#table(qol\_train$response)  
# Convert response variable to factor   
#qol\_train$response <- as.factor(qol\_train$response)  
  
  
  
#encode response variable in testing set  
#qol\_test <- qol\_test %>%  
 #mutate(response = ifelse(response == "better", 0, 1))  
#table(qol\_test$response)  
# Convert response variable to factor   
#qol\_test$response <- as.factor(qol\_test$response)

library(recipes)  
set.seed(12)  
# Create a recipe  
response\_recipe <- recipe(response ~ ., data = qol\_train) %>%  
 step\_mutate(response = ifelse(response == "better", 0, 1)) %>% # Encode 'response' as 1 and 0  
 step\_dummy(region, one\_hot = TRUE) %>% # One-hot encode the 'region' feature to include all 4 regions  
 step\_center(all\_predictors()) %>%  
 step\_scale(all\_predictors()) %>%  
 #step\_rm(region) %>% # Remove the original 'region' column  
 #step\_rm(response) %>% # Remove the original 'response' column  
 prep(training = qol\_train, retain = TRUE)  
  
# Apply the recipe to the training and testing datasets  
qol\_train <- bake(response\_recipe, new\_data = qol\_train)  
qol\_test <- bake(response\_recipe, new\_data = qol\_test)  
  
#show distribution of response variable in   
# For qol\_train dataset  
table(qol\_train$response)

##   
## 0 1   
## 1359 1340

# For qol\_test dataset  
table(qol\_test$response)

##   
## 0 1   
## 222 219

# use to identify which transformation to use  
  
# use square root + 3 (from q-q values) to get shapiro wilk p-value of around 0.5, if less than 0.05, it is not normally distributed  
  
# Q-Q plotsare useful if greater than 5000 observations

# run a pca (DO this with 0, 1) or lda (prefered) on numeric data

Build a simple OLS and RF model

# Convert response variable to numeric  
qol\_train$response <- as.numeric(as.character(qol\_train$response))  
  
# --- OLS Model ---  
  
# Fit the OLS model  
ols\_model <- lm(response ~ ., data = qol\_train)  
  
# Print OLS coefficients  
print(summary(ols\_model)$coefficients)

## Estimate Std. Error t value  
## (Intercept) 0.4964801778 0.005418819 91.62146448  
## weighted\_population 0.0089464501 0.006896139 1.29731298  
## average\_hh\_size 0.0243631205 0.010227761 2.38205811  
## pct\_male -0.0242662250 0.006949771 -3.49165837  
## pct\_native\_american -0.0069018112 0.023034388 -0.29963077  
## pct\_asian -0.0007704927 0.013060153 -0.05899569  
## pct\_black 0.0284404930 0.037411872 0.76019968  
## pct\_hispanic 0.0505781587 0.012953036 3.90473379  
## pct\_other\_race 0.0324762294 0.011900663 2.72894276  
## pct\_white 0.0226478722 0.043306762 0.52296388  
## pct\_single\_parent 0.0006293589 0.009261174 0.06795670  
## pct\_hh\_other\_computer -0.0036238794 0.005752333 -0.62998426  
## pct\_hh\_internet 0.0095325097 0.010034122 0.95000934  
## pct\_employed 0.0179736014 0.008112687 2.21549297  
## pct\_hh\_inc\_99999 -0.0148898972 0.007944334 -1.87427892  
## pct\_w\_medicare 0.0120993829 0.008081453 1.49717921  
## clinical\_nurse\_pt 0.0040767992 0.005806001 0.70216994  
## dentist\_pt 0.0212414670 0.007312702 2.90473567  
## pa\_pt -0.0188355185 0.006407077 -2.93979877  
## mental\_health\_faciliy\_pt -0.0159326464 0.005906869 -2.69730807  
## population\_density -0.0041293235 0.006720298 -0.61445541  
## days\_over\_90\_f 0.0424241761 0.010400891 4.07889841  
## median\_hh\_income 0.0036107492 0.012713846 0.28400134  
## median\_er\_dist -0.0043476135 0.008230263 -0.52824718  
## median\_pediatric\_icu\_dist -0.0142014779 0.007534933 -1.88475183  
## median\_health\_clinic\_dist -0.0029548842 0.006621209 -0.44627560  
## median\_drug\_alcohol\_care\_dist 0.0236493492 0.008162836 2.89719760  
## percent\_grandparents\_as\_guardians 0.0064473228 0.007798499 0.82673892  
## pct\_adult\_smokers 0.1568963800 0.018366141 8.54269699  
## pct\_obese\_adults 0.0281949877 0.012275571 2.29683721  
## pct\_no\_exercise 0.0267856549 0.018211314 1.47082492  
## pct\_binge\_drinkers -0.0260296319 0.008517594 -3.05598403  
## pct\_under\_65\_no\_health\_insurance 0.0187185427 0.009198963 2.03485361  
## pct\_highschool\_diploma 0.0002995497 0.012368565 0.02421863  
## pct\_some\_college -0.0390191691 0.011350992 -3.43751193  
## inequality\_ratio 0.0048139008 0.008236069 0.58449012  
## social\_clubs\_per\_10k -0.0014126193 0.006872383 -0.20555013  
## air\_polution\_metric 0.0094938544 0.008076558 1.17548276  
## water\_quality -0.0133358226 0.005799055 -2.29965462  
## pct\_high\_housing\_costs 0.0098093416 0.008459723 1.15953463  
## pct\_overcrowded\_hh -0.0101635778 0.009523151 -1.06724950  
## pct\_30\_min\_plus\_commute 0.0112510997 0.008747745 1.28617139  
## life\_expectancy\_years -0.0210434480 0.009738927 -2.16075640  
## pct\_food\_insecurities 0.0457803478 0.011567561 3.95764930  
## pct\_voters -0.0616108220 0.010125395 -6.08478210  
## pct\_home\_owner -0.0122532750 0.009881835 -1.23997971  
## pct\_65\_plus -0.0042739239 0.011498179 -0.37170442  
## pct\_rural\_population 0.0185240861 0.010412873 1.77896011  
## region\_Midwest -0.0930435854 0.013534358 -6.87462136  
## region\_Northeast -0.0566797247 0.007783826 -7.28173056  
## region\_South -0.0214077652 0.014327242 -1.49420000  
## Pr(>|t|)  
## (Intercept) 0.000000e+00  
## weighted\_population 1.946365e-01  
## average\_hh\_size 1.728640e-02  
## pct\_male 4.878868e-04  
## pct\_native\_american 7.644823e-01  
## pct\_asian 9.529600e-01  
## pct\_black 4.472029e-01  
## pct\_hispanic 9.668553e-05  
## pct\_other\_race 6.395768e-03  
## pct\_white 6.010431e-01  
## pct\_single\_parent 9.458252e-01  
## pct\_hh\_other\_computer 5.287592e-01  
## pct\_hh\_internet 3.421942e-01  
## pct\_employed 2.681102e-02  
## pct\_hh\_inc\_99999 6.100201e-02  
## pct\_w\_medicare 1.344658e-01  
## clinical\_nurse\_pt 4.826349e-01  
## dentist\_pt 3.706089e-03  
## pa\_pt 3.312683e-03  
## mental\_health\_faciliy\_pt 7.034562e-03  
## population\_density 5.389672e-01  
## days\_over\_90\_f 4.658538e-05  
## median\_hh\_income 7.764316e-01  
## median\_er\_dist 5.973721e-01  
## median\_pediatric\_icu\_dist 5.957278e-02  
## median\_health\_clinic\_dist 6.554347e-01  
## median\_drug\_alcohol\_care\_dist 3.796028e-03  
## percent\_grandparents\_as\_guardians 4.084595e-01  
## pct\_adult\_smokers 2.177647e-17  
## pct\_obese\_adults 2.170576e-02  
## pct\_no\_exercise 1.414573e-01  
## pct\_binge\_drinkers 2.265610e-03  
## pct\_under\_65\_no\_health\_insurance 4.196503e-02  
## pct\_highschool\_diploma 9.806800e-01  
## pct\_some\_college 5.961451e-04  
## inequality\_ratio 5.589404e-01  
## social\_clubs\_per\_10k 8.371581e-01  
## air\_polution\_metric 2.399073e-01  
## water\_quality 2.154525e-02  
## pct\_high\_housing\_costs 2.463429e-01  
## pct\_overcrowded\_hh 2.859565e-01  
## pct\_30\_min\_plus\_commute 1.984957e-01  
## life\_expectancy\_years 3.080357e-02  
## pct\_food\_insecurities 7.768148e-05  
## pct\_voters 1.335223e-09  
## pct\_home\_owner 2.150928e-01  
## pct\_65\_plus 7.101427e-01  
## pct\_rural\_population 7.536099e-02  
## region\_Midwest 7.722988e-12  
## region\_Northeast 4.324858e-13  
## region\_South 1.352425e-01

# Predict on the test set  
predictions\_ols <- predict(ols\_model, newdata = qol\_test)  
  
# Evaluate the OLS model  
confusion\_matrix\_ols <- table(qol\_test$response, predictions\_ols > 0.5)  
accuracy\_ols <- sum(diag(confusion\_matrix\_ols)) / sum(confusion\_matrix\_ols)  
print(paste("Accuracy (OLS):", accuracy\_ols))

## [1] "Accuracy (OLS): 0.920634920634921"

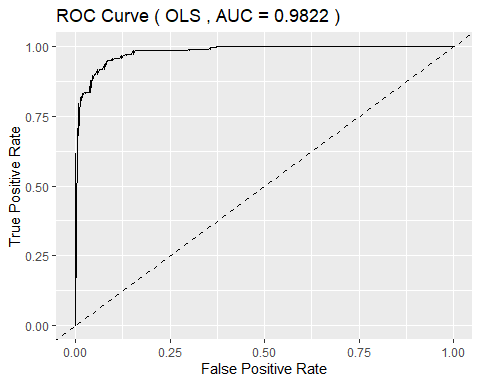
# Calculate feature importance for OLS  
feature\_importance\_ols <- abs(summary(ols\_model)$coefficients[, "Estimate"])  
importance\_df\_ols <- data.frame(Feature = names(feature\_importance\_ols), Importance = feature\_importance\_ols)  
importance\_df\_ols <- importance\_df\_ols[order(-importance\_df\_ols$Importance), ]  
  
# --- Random Forest Model ---  
  
# Train a random forest model  
rf\_model <- randomForest(as.factor(response) ~ ., data = qol\_train, ntree = 1000, na.action = na.omit)  
  
# Predict on the test set  
predictions\_rf <- predict(rf\_model, qol\_test, type = "prob")[, 2]  
  
# Evaluate the Random Forest model  
confusion\_matrix\_rf <- table(qol\_test$response, predictions\_rf > 0.5)  
accuracy\_rf <- sum(diag(confusion\_matrix\_rf)) / sum(confusion\_matrix\_rf)  
print(paste("Accuracy (RF):", accuracy\_rf))

## [1] "Accuracy (RF): 0.936507936507937"

# --- ROC Curve (Both Models) ---  
  
# Function to plot ROC curves  
plot\_roc <- function(predictions, model\_name) {  
 roc\_obj <- roc(qol\_test$response, predictions)  
 auc\_value <- auc(roc\_obj)  
   
 roc\_df <- data.frame(  
 tpr = roc\_obj$sensitivities,  
 fpr = 1 - roc\_obj$specificities  
 )  
   
 ggplot(roc\_df, aes(x = fpr, y = tpr)) +  
 geom\_line() +  
 geom\_abline(linetype = "dashed") +  
 xlab("False Positive Rate") +  
 ylab("True Positive Rate") +  
 ggtitle(paste("ROC Curve (", model\_name, ", AUC =", round(auc\_value, 4), ")"))  
}  
  
# Plot ROC curves for both models  
plot\_roc(predictions\_ols, "OLS")

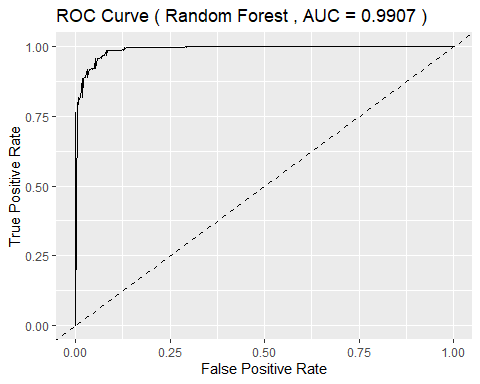
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

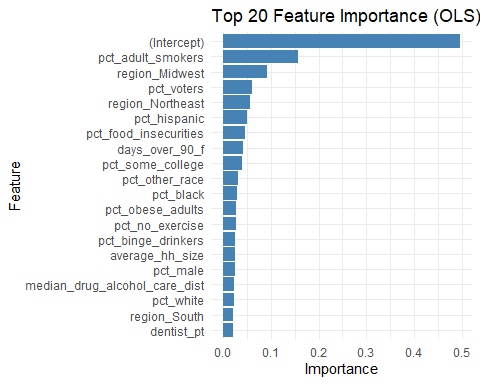


plot\_roc(predictions\_rf, "Random Forest")

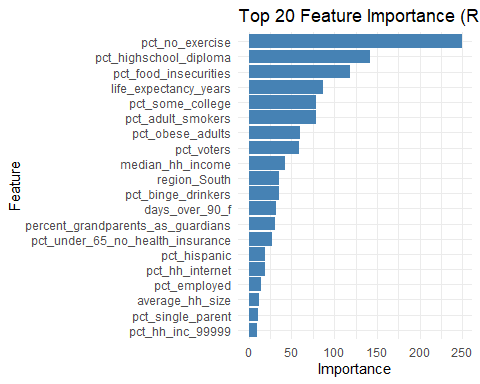
## Setting levels: control = 0, case = 1  
## Setting direction: controls < cases



# --- Feature Importance Visualization (OLS) ---  
  
# Select top 20 features  
top\_20\_features\_ols <- head(importance\_df\_ols, 20)  
  
# Plot top 20 feature importance for OLS  
ggplot(top\_20\_features\_ols, aes(x = reorder(Feature, Importance), y = Importance)) +  
 geom\_bar(stat = "identity", fill = "steelblue") +  
 coord\_flip() +  
 labs(title = "Top 20 Feature Importance (OLS)", x = "Feature", y = "Importance") +  
 theme\_minimal()



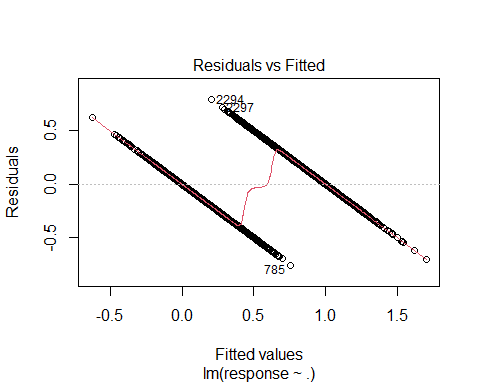
# --- Feature Importance Visualization (RF) ---  
  
# Calculate feature importance for RF  
feature\_importance\_rf <- importance(rf\_model)  
importance\_df\_rf <- data.frame(Feature = rownames(feature\_importance\_rf), Importance = feature\_importance\_rf[, 1])  
importance\_df\_rf <- importance\_df\_rf[order(-importance\_df\_rf$Importance), ]  
  
# Select top 20 features  
top\_20\_features\_rf <- head(importance\_df\_rf, 20)  
  
# Plot top 20 feature importance for RF  
ggplot(top\_20\_features\_rf, aes(x = reorder(Feature, Importance), y = Importance)) +  
 geom\_bar(stat = "identity", fill = "steelblue") +  
 coord\_flip() +  
 labs(title = "Top 20 Feature Importance (RF)", x = "Feature", y = "Importance") +  
 theme\_minimal()



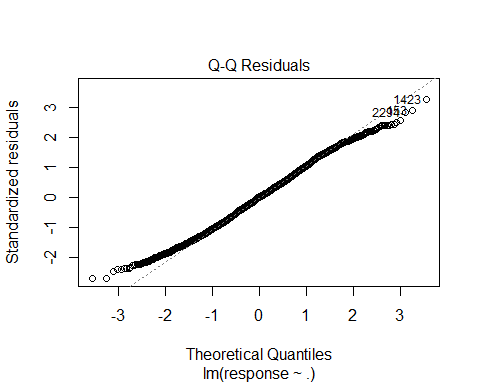
Assumptions OLS Model Linearity: Assumes a linear relationship between the predictors and the response variable. Independence: Assumes that the observations are independent. Homoscedasticity: Assumes constant variance of the errors. Normality: Assumes that the errors are normally distributed. To check these assumptions, use the following diagnostics:

Residuals Plot: To check for homoscedasticity and linearity. Q-Q Plot: To check for normality of residuals. Durbin-Watson Test: To check for independence of residuals.

# Residuals plot  
plot(ols\_model, which = 1)



# Q-Q plot  
plot(ols\_model, which = 2)



# Durbin-Watson test  
library(lmtest)

## Loading required package: zoo

##   
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

dwtest(ols\_model)

##   
## Durbin-Watson test  
##   
## data: ols\_model  
## DW = 0.89266, p-value = 0.3085  
## alternative hypothesis: true autocorrelation is greater than 0

build Lasso Model

# Convert response variable to factor  
qol\_train$response <- as.factor(qol\_train$response)  
qol\_test$response <- as.factor(qol\_test$response)  
  
# Create design matrices (excluding response variable)  
x\_train <- model.matrix(~ ., data = qol\_train[,-which(names(qol\_train) == "response")])  
x\_test <- model.matrix(~ ., data = qol\_test[,-which(names(qol\_test) == "response")])  
  
# Fit the Lasso model  
lasso\_model <- glmnet(x\_train, qol\_train$response, family = "binomial", alpha = 1)   
  
# Find optimal lambda using cross-validation  
cv\_lasso <- cv.glmnet(x\_train, qol\_train$response, family = "binomial", alpha = 1)  
  
# Predict on the test set using the optimal lambda  
predictions\_lasso <- predict(lasso\_model, newx = x\_test, s = cv\_lasso$lambda.min, type = "response")  
  
# Convert probabilities to class predictions (0 or 1)  
predicted\_classes\_lasso <- ifelse(predictions\_lasso > 0.5, 1, 0)  
  
# Calculate accuracy  
accuracy\_lasso <- mean(qol\_test$response == predicted\_classes\_lasso)  
cat("Accuracy (Lasso):", accuracy\_lasso, "\n")

## Accuracy (Lasso): 0.9319728

# Plot ROC curve  
roc\_lasso <- roc(qol\_test$response, predictions\_lasso)

## Setting levels: control = 0, case = 1

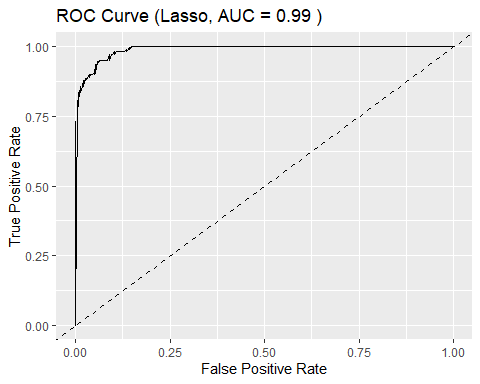
## Warning in roc.default(qol\_test$response, predictions\_lasso): Deprecated use a  
## matrix as predictor. Unexpected results may be produced, please pass a numeric  
## vector.

## Setting direction: controls < cases

auc\_lasso <- auc(roc\_lasso)  
cat("AUC (Lasso):", auc\_lasso, "\n")

## AUC (Lasso): 0.9896129

ggplot(data.frame(fpr = 1 - roc\_lasso$specificities, tpr = roc\_lasso$sensitivities),   
 aes(x = fpr, y = tpr)) +  
 geom\_line() +  
 geom\_abline(linetype = "dashed") +  
 ggtitle(paste("ROC Curve (Lasso, AUC =", round(auc\_lasso, 2), ")")) +  
 xlab("False Positive Rate") +  
 ylab("True Positive Rate")



# Feature importance  
  
# Extract coefficients as a dense vector  
coef\_lasso <- coef(lasso\_model, s = cv\_lasso$lambda.min)  
  
# Extract non-zero coefficients  
non\_zero\_coefs <- coef\_lasso[which(coef\_lasso != 0)]  
  
# Get the names of the non-zero coefficients  
feature\_names <- rownames(coef\_lasso)[which(coef\_lasso != 0)]  
  
# Create the data frame  
importance\_df\_lasso <- data.frame(Feature = feature\_names, Importance = abs(non\_zero\_coefs))  
  
  
importance\_df\_lasso <- importance\_df\_lasso[order(-importance\_df\_lasso$Importance), ]  
top\_20\_features\_lasso <- head(importance\_df\_lasso, 20)  
  
# Plot top 20 feature importance  
ggplot(top\_20\_features\_lasso, aes(x = reorder(Feature, Importance), y = Importance)) +  
 geom\_bar(stat = "identity", fill = "steelblue") +  
 coord\_flip() +  
 labs(title = "Top 20 Feature Importance (Lasso)", x = "Feature", y = "Importance") +  
 theme\_minimal()

