Untitled

Udumaga Onyeukwu

2024-08-06

*Packages Load*

# load packages  
  
library(readr) # load csv files  
library(readxl) # load excel files  
library(dplyr) # data manipulation  
library(lubridate) # date & time manipulation  
library(ggplot2) # data visualization  
library(tidyr) # collection of statistical packages, packages loaded individually  
library(corrplot) # to visualize correlations  
library(leaps) # for subset selection  
library(caret) # test for correlation  
library(car) # for VIF  
library(scales) # for visualizing plots in %  
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  
library(pROC) # to analyse and display reciever operating characteristics(ROC) curvees  
library(randomForest) # to impletement Random Forest algorithm  
library(glmnet)  
library(tidymodels)  
library(recipes)  
library(lmtest)  
library(xgboost)  
library(openxlsx)

*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))

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

Split data intraining and testing sets, delete non useful features, input missing values

# Find columns with missing data  
#sum(colSums(is.na(qol\_data)))  
  
# 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  
  
  
data\_split <- initial\_split(qol\_data, strata = "pct\_poor\_to\_fair\_health", 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)  
  
# 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 weighted\_population average\_hh\_size pct\_male pct\_native\_american pct\_asian pct\_black pct\_hispanic pct\_other\_race pct\_white pct\_single\_parent pct\_hh\_other\_computer pct\_hh\_internet pct\_employed pct\_hh\_inc\_99999 pct\_w\_medicare population\_density days\_over\_90\_f median\_hh\_income median\_er\_dist median\_pediatric\_icu\_dist median\_health\_clinic\_dist median\_drug\_alcohol\_care\_dist percent\_grandparents\_as\_guardians pct\_poor\_to\_fair\_health pct\_adult\_smokers pct\_obese\_adults pct\_no\_exercise pct\_binge\_drinkers pct\_under\_65\_no\_health\_insurance inequality\_ratio air\_polution\_metric water\_quality life\_expectancy\_years pct\_voters pct\_rural\_population  
## 2 1 weighted\_population average\_hh\_size pct\_male pct\_native\_american pct\_asian pct\_black pct\_hispanic pct\_other\_race pct\_white pct\_single\_parent pct\_hh\_other\_computer pct\_hh\_internet pct\_employed pct\_hh\_inc\_99999 pct\_w\_medicare population\_density days\_over\_90\_f median\_hh\_income median\_er\_dist median\_pediatric\_icu\_dist median\_health\_clinic\_dist median\_drug\_alcohol\_care\_dist percent\_grandparents\_as\_guardians pct\_poor\_to\_fair\_health pct\_adult\_smokers pct\_obese\_adults pct\_no\_exercise pct\_binge\_drinkers pct\_under\_65\_no\_health\_insurance inequality\_ratio air\_polution\_metric water\_quality life\_expectancy\_years pct\_voters pct\_rural\_population  
## 3 1 weighted\_population average\_hh\_size pct\_male pct\_native\_american pct\_asian pct\_black pct\_hispanic pct\_other\_race pct\_white pct\_single\_parent pct\_hh\_other\_computer pct\_hh\_internet pct\_employed pct\_hh\_inc\_99999 pct\_w\_medicare population\_density days\_over\_90\_f median\_hh\_income median\_er\_dist median\_pediatric\_icu\_dist median\_health\_clinic\_dist median\_drug\_alcohol\_care\_dist percent\_grandparents\_as\_guardians pct\_poor\_to\_fair\_health pct\_adult\_smokers pct\_obese\_adults pct\_no\_exercise pct\_binge\_drinkers pct\_under\_65\_no\_health\_insurance inequality\_ratio air\_polution\_metric water\_quality life\_expectancy\_years pct\_voters pct\_rural\_population  
## 4 1 weighted\_population average\_hh\_size pct\_male pct\_native\_american pct\_asian pct\_black pct\_hispanic pct\_other\_race pct\_white pct\_single\_parent pct\_hh\_other\_computer pct\_hh\_internet pct\_employed pct\_hh\_inc\_99999 pct\_w\_medicare population\_density days\_over\_90\_f median\_hh\_income median\_er\_dist median\_pediatric\_icu\_dist median\_health\_clinic\_dist median\_drug\_alcohol\_care\_dist percent\_grandparents\_as\_guardians pct\_poor\_to\_fair\_health pct\_adult\_smokers pct\_obese\_adults pct\_no\_exercise pct\_binge\_drinkers pct\_under\_65\_no\_health\_insurance inequality\_ratio air\_polution\_metric water\_quality life\_expectancy\_years pct\_voters pct\_rural\_population  
## 5 1 weighted\_population average\_hh\_size pct\_male pct\_native\_american pct\_asian pct\_black pct\_hispanic pct\_other\_race pct\_white pct\_single\_parent pct\_hh\_other\_computer pct\_hh\_internet pct\_employed pct\_hh\_inc\_99999 pct\_w\_medicare population\_density days\_over\_90\_f median\_hh\_income median\_er\_dist median\_pediatric\_icu\_dist median\_health\_clinic\_dist median\_drug\_alcohol\_care\_dist percent\_grandparents\_as\_guardians pct\_poor\_to\_fair\_health pct\_adult\_smokers pct\_obese\_adults pct\_no\_exercise pct\_binge\_drinkers 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 percent\_grandparents\_as\_guardians inequality\_ratio air\_polution\_metric water\_quality life\_expectancy\_years pct\_voters  
## 2 1 days\_over\_90\_f percent\_grandparents\_as\_guardians inequality\_ratio air\_polution\_metric water\_quality life\_expectancy\_years pct\_voters  
## 3 1 days\_over\_90\_f percent\_grandparents\_as\_guardians inequality\_ratio air\_polution\_metric water\_quality life\_expectancy\_years pct\_voters  
## 4 1 days\_over\_90\_f percent\_grandparents\_as\_guardians inequality\_ratio air\_polution\_metric water\_quality life\_expectancy\_years pct\_voters  
## 5 1 days\_over\_90\_f percent\_grandparents\_as\_guardians inequality\_ratio 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))

one hot encode character Variables,center and scale all predictors except one hot encoded variables.

set.seed(12)  
# Create a recipe  
response\_recipe <- recipe(pct\_poor\_to\_fair\_health ~ ., data = qol\_train) %>%  
   
 step\_dummy(region, one\_hot = TRUE) %>% # One-hot encode the 'region' feature to include all 4 regions  
 step\_center(all\_predictors(), -starts\_with("region\_")) %>% # Center all predictors except the one-hot encoded 'region' columns  
 step\_scale(all\_predictors(), -starts\_with("region\_")) %>% # Scale all predictors except the one-hot encoded 'region' columns  
 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)

Build and compare OLs, RF and lasso models

# Fit the OLS model  
ols\_model <- lm(pct\_poor\_to\_fair\_health ~ ., data = qol\_train)  
  
# Print OLS coefficients  
print(summary(ols\_model)$coefficients)

## Estimate Std. Error t value  
## (Intercept) 0.1667531239 0.0007402773 225.2576492  
## weighted\_population -0.0001832448 0.0002244152 -0.8165437  
## average\_hh\_size 0.0012214552 0.0003438822 3.5519581  
## pct\_male 0.0008986935 0.0002289776 3.9248099  
## pct\_native\_american 0.0004253901 0.0007519222 0.5657368  
## pct\_asian 0.0005004165 0.0004255643 1.1758893  
## pct\_black 0.0036175517 0.0011814665 3.0619166  
## pct\_hispanic 0.0063347871 0.0004276278 14.8137867  
## pct\_other\_race -0.0001299606 0.0003844120 -0.3380763  
## pct\_white 0.0009844123 0.0013762409 0.7152907  
## pct\_single\_parent 0.0005423958 0.0002991155 1.8133327  
## pct\_hh\_other\_computer -0.0002905630 0.0001873700 -1.5507444  
## pct\_hh\_internet -0.0008739334 0.0003257878 -2.6825236  
## pct\_employed -0.0006194164 0.0002626579 -2.3582630  
## pct\_hh\_inc\_99999 -0.0014631466 0.0002577000 -5.6777122  
## pct\_w\_medicare -0.0002824302 0.0002682869 -1.0527170  
## clinical\_nurse\_pt -0.0000357989 0.0001902213 -0.1881961  
## dentist\_pt 0.0001455579 0.0002461408 0.5913601  
## pa\_pt -0.0002644818 0.0002080331 -1.2713444  
## mental\_health\_faciliy\_pt 0.0002876091 0.0001962302 1.4656717  
## population\_density -0.0005634245 0.0002130007 -2.6451769  
## days\_over\_90\_f -0.0010073222 0.0003387878 -2.9733130  
## median\_hh\_income -0.0005651299 0.0004226812 -1.3370122  
## median\_er\_dist -0.0008034986 0.0002696165 -2.9801533  
## median\_pediatric\_icu\_dist 0.0000323153 0.0002417681 0.1336624  
## median\_health\_clinic\_dist -0.0010167693 0.0002165432 -4.6954574  
## median\_drug\_alcohol\_care\_dist 0.0010989580 0.0002692086 4.0821797  
## percent\_grandparents\_as\_guardians 0.0002012997 0.0002549438 0.7895848  
## pct\_adult\_smokers 0.0090831870 0.0006014870 15.1012199  
## pct\_obese\_adults 0.0043190441 0.0004044524 10.6787446  
## pct\_no\_exercise 0.0175265377 0.0005828287 30.0715090  
## pct\_binge\_drinkers -0.0015395616 0.0002782273 -5.5334676  
## pct\_under\_65\_no\_health\_insurance -0.0012962860 0.0003050287 -4.2497178  
## pct\_highschool\_diploma -0.0038438246 0.0003963252 -9.6986638  
## pct\_some\_college 0.0006251624 0.0003690030 1.6941934  
## inequality\_ratio 0.0005590815 0.0002688658 2.0794072  
## social\_clubs\_per\_10k -0.0004348399 0.0002231403 -1.9487283  
## air\_polution\_metric 0.0014700099 0.0002650512 5.5461352  
## water\_quality 0.0003260659 0.0001906233 1.7105245  
## pct\_high\_housing\_costs 0.0006638655 0.0002805090 2.3666457  
## pct\_overcrowded\_hh 0.0013419150 0.0003074762 4.3642892  
## pct\_30\_min\_plus\_commute -0.0003714686 0.0002852210 -1.3023887  
## life\_expectancy\_years 0.0003029483 0.0003266542 0.9274279  
## pct\_food\_insecurities 0.0076641066 0.0003829749 20.0120347  
## pct\_voters 0.0015272981 0.0003322259 4.5971669  
## pct\_home\_owner 0.0006649507 0.0003183508 2.0887356  
## pct\_65\_plus 0.0004789357 0.0003718530 1.2879703  
## pct\_rural\_population 0.0001099482 0.0003438049 0.3197981  
## region\_Midwest -0.0140374670 0.0009233961 -15.2019987  
## region\_Northeast -0.0132849220 0.0010057865 -13.2084907  
## region\_South -0.0020169893 0.0009273715 -2.1749528  
## Pr(>|t|)  
## (Intercept) 0.000000e+00  
## weighted\_population 4.142627e-01  
## average\_hh\_size 3.890459e-04  
## pct\_male 8.900852e-05  
## pct\_native\_american 5.716206e-01  
## pct\_asian 2.397448e-01  
## pct\_black 2.221335e-03  
## pct\_hispanic 9.228516e-48  
## pct\_other\_race 7.353325e-01  
## pct\_white 4.744925e-01  
## pct\_single\_parent 6.989365e-02  
## pct\_hh\_other\_computer 1.210824e-01  
## pct\_hh\_internet 7.352290e-03  
## pct\_employed 1.843296e-02  
## pct\_hh\_inc\_99999 1.513570e-08  
## pct\_w\_medicare 2.925667e-01  
## clinical\_nurse\_pt 8.507373e-01  
## dentist\_pt 5.543296e-01  
## pa\_pt 2.037178e-01  
## mental\_health\_faciliy\_pt 1.428564e-01  
## population\_density 8.213056e-03  
## days\_over\_90\_f 2.972607e-03  
## median\_hh\_income 1.813334e-01  
## median\_er\_dist 2.907243e-03  
## median\_pediatric\_icu\_dist 8.936797e-01  
## median\_health\_clinic\_dist 2.795647e-06  
## median\_drug\_alcohol\_care\_dist 4.593590e-05  
## percent\_grandparents\_as\_guardians 4.298410e-01  
## pct\_adult\_smokers 1.720990e-49  
## pct\_obese\_adults 4.309915e-26  
## pct\_no\_exercise 3.530838e-171  
## pct\_binge\_drinkers 3.447243e-08  
## pct\_under\_65\_no\_health\_insurance 2.214370e-05  
## pct\_highschool\_diploma 7.027500e-22  
## pct\_some\_college 9.034609e-02  
## inequality\_ratio 3.767591e-02  
## social\_clubs\_per\_10k 5.143332e-02  
## air\_polution\_metric 3.209358e-08  
## water\_quality 8.728601e-02  
## pct\_high\_housing\_costs 1.802165e-02  
## pct\_overcrowded\_hh 1.324223e-05  
## pct\_30\_min\_plus\_commute 1.928968e-01  
## life\_expectancy\_years 3.537889e-01  
## pct\_food\_insecurities 4.413767e-83  
## pct\_voters 4.483610e-06  
## pct\_home\_owner 3.682677e-02  
## pct\_65\_plus 1.978688e-01  
## pct\_rural\_population 7.491466e-01  
## region\_Midwest 4.196444e-50  
## region\_Northeast 1.268686e-38  
## region\_South 2.972184e-02

# Predict on the test set  
predictions\_ols <- predict(ols\_model, new\_data = qol\_test)  
  
# Evaluate the OLS model  
mse\_ols <- mean((qol\_test$pct\_poor\_to\_fair\_health - predictions\_ols)^2)

## Warning in qol\_test$pct\_poor\_to\_fair\_health - predictions\_ols: longer object  
## length is not a multiple of shorter object length

rmse\_ols <- sqrt(mse\_ols)  
r2\_ols <- summary(ols\_model)$r.squared  
print(paste("MSE (OLS):", mse\_ols))

## [1] "MSE (OLS): 0.0039069415003134"

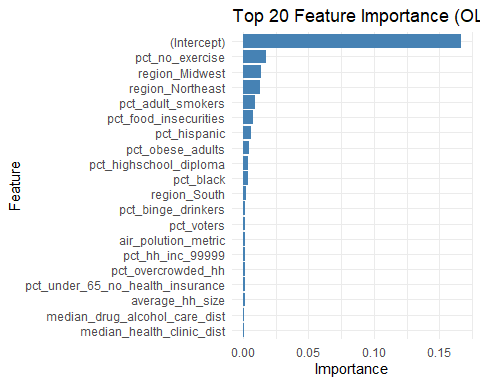
print(paste("RMSE (OLS):", rmse\_ols))

## [1] "RMSE (OLS): 0.0625055317577044"

print(paste("R-squared (OLS):", r2\_ols))

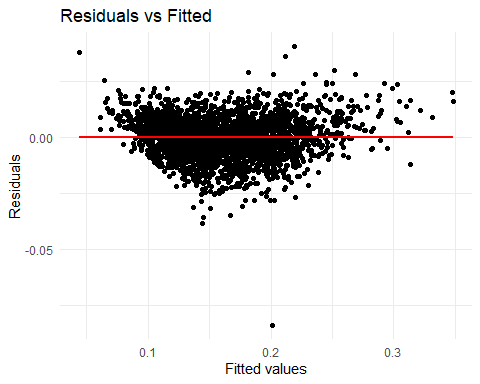
## [1] "R-squared (OLS): 0.957627822349486"

# --- Feature Importance Visualization (OLS) ---  
  
# 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), ]  
  
# 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()

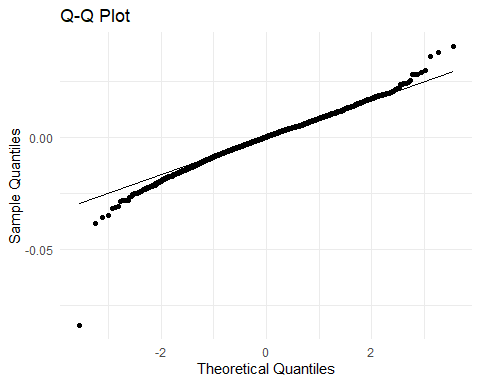


# --- Assumptions Diagnostics for OLS Model ---  
  
# Residuals plot  
ggplot(data.frame(residuals = residuals(ols\_model), fitted = fitted(ols\_model)), aes(x = fitted, y = residuals)) +  
 geom\_point() +  
 geom\_smooth(method = "lm", se = FALSE, color = "red") +  
 labs(title = "Residuals vs Fitted", x = "Fitted values", y = "Residuals") +  
 theme\_minimal()

## `geom\_smooth()` using formula = 'y ~ x'



# Q-Q plot  
ggplot(data.frame(sample = residuals(ols\_model)), aes(sample = sample)) +  
 stat\_qq() +  
 stat\_qq\_line() +  
 labs(title = "Q-Q Plot", x = "Theoretical Quantiles", y = "Sample Quantiles") +  
 theme\_minimal()



# Durbin-Watson test  
#dwtest(ols\_model)  
  
# Perform the Breusch-Godfrey test for autocorrelation  
bg\_test <- bgtest(ols\_model, order = 1) # You can change the order as needed  
  
# Print the test results  
print(bg\_test)

##   
## Breusch-Godfrey test for serial correlation of order up to 1  
##   
## data: ols\_model  
## LM test = 738.41, df = 1, p-value < 2.2e-16

Random forest model

# Load necessary library  
library(randomForest)  
  
# Train a random forest model  
rf\_model <- randomForest(pct\_poor\_to\_fair\_health ~ ., data = qol\_train, ntree = 1000, na.action = na.omit)  
  
# Predict on the test set  
predictions\_rf <- predict(rf\_model, qol\_test)  
  
# Evaluate the Random Forest model  
mse\_rf <- mean((qol\_test$pct\_poor\_to\_fair\_health - predictions\_rf)^2)  
rmse\_rf <- sqrt(mse\_rf)  
r2\_rf <- 1 - (sum((qol\_test$pct\_poor\_to\_fair\_health - predictions\_rf)^2) / sum((qol\_test$pct\_poor\_to\_fair\_health - mean(qol\_test$pct\_poor\_to\_fair\_health))^2))  
print(paste("MSE (RF):", mse\_rf))

## [1] "MSE (RF): 6.5813230305314e-05"

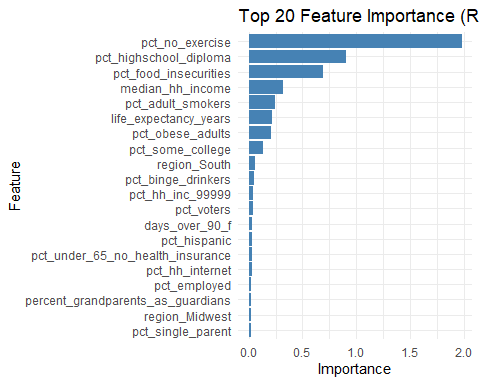
print(paste("RMSE (RF):", rmse\_rf))

## [1] "RMSE (RF): 0.00811253538083588"

print(paste("R-squared (RF):", r2\_rf))

## [1] "R-squared (RF): 0.966253647924883"

# --- 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 = abs(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()



# Create design matrices (excluding response variable)  
x\_train <- model.matrix(pct\_poor\_to\_fair\_health ~ ., data = qol\_train)[, -1]  
x\_test <- model.matrix(pct\_poor\_to\_fair\_health ~ ., data = qol\_test)[, -1]  
  
# Fit the Lasso model  
lasso\_model <- glmnet(x\_train, qol\_train$pct\_poor\_to\_fair\_health, alpha = 1)  
  
# Find optimal lambda using cross-validation  
cv\_lasso <- cv.glmnet(x\_train, qol\_train$pct\_poor\_to\_fair\_health, alpha = 1)  
  
# Predict on the test set using the optimal lambda  
predictions\_lasso <- predict(lasso\_model, newx = x\_test, s = cv\_lasso$lambda.min)  
  
# Evaluate the Lasso model  
mse\_lasso <- mean((qol\_test$pct\_poor\_to\_fair\_health - predictions\_lasso)^2)  
rmse\_lasso <- sqrt(mse\_lasso)  
r2\_lasso <- 1 - (sum((qol\_test$pct\_poor\_to\_fair\_health - predictions\_lasso)^2) / sum((qol\_test$pct\_poor\_to\_fair\_health - mean(qol\_test$pct\_poor\_to\_fair\_health))^2))  
print(paste("MSE (Lasso):", mse\_lasso))

## [1] "MSE (Lasso): 7.98139624859371e-05"

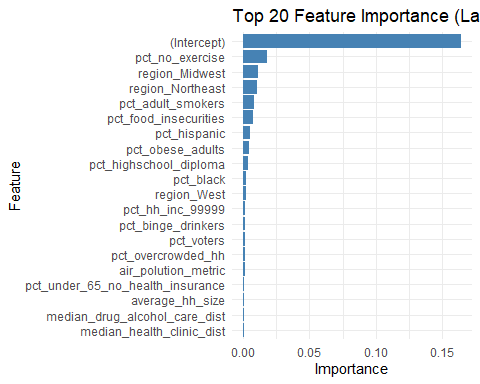
print(paste("RMSE (Lasso):", rmse\_lasso))

## [1] "RMSE (Lasso): 0.00893386604365306"

print(paste("R-squared (Lasso):", r2\_lasso))

## [1] "R-squared (Lasso): 0.959074640979245"

# --- Feature Importance Visualization (Lasso) ---  
  
# 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))  
  
# Select top 20 features  
top\_20\_features\_lasso <- head(importance\_df\_lasso[order(-importance\_df\_lasso$Importance), ], 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()



# Prepare data for xgboost  
dtrain <- xgb.DMatrix(data = as.matrix(qol\_train[, -which(names(qol\_train) == "pct\_poor\_to\_fair\_health")]), label = qol\_train$pct\_poor\_to\_fair\_health)  
dtest <- xgb.DMatrix(data = as.matrix(qol\_test[, -which(names(qol\_test) == "pct\_poor\_to\_fair\_health")]), label = qol\_test$pct\_poor\_to\_fair\_health)  
  
# Set parameters for xgboost  
params <- list(  
 objective = "reg:squarederror",  
 eta = 0.1,  
 max\_depth = 6,  
 subsample = 0.8,  
 colsample\_bytree = 0.8  
)  
  
# Train the GBM model  
gbm\_model <- xgb.train(params, dtrain, nrounds = 100)  
  
# Predict on the test set  
predictions\_gbm <- predict(gbm\_model, dtest)  
  
# Evaluate the GBM model  
mse\_gbm <- mean((qol\_test$pct\_poor\_to\_fair\_health - predictions\_gbm)^2)  
rmse\_gbm <- sqrt(mse\_gbm)  
r2\_gbm <- 1 - (sum((qol\_test$pct\_poor\_to\_fair\_health - predictions\_gbm)^2) / sum((qol\_test$pct\_poor\_to\_fair\_health - mean(qol\_test$pct\_poor\_to\_fair\_health))^2))  
print(paste("MSE (GBM):", mse\_gbm))

## [1] "MSE (GBM): 5.05616020737363e-05"

print(paste("RMSE (GBM):", rmse\_gbm))

## [1] "RMSE (GBM): 0.00711066818757115"

print(paste("R-squared (GBM):", r2\_gbm))

## [1] "R-squared (GBM): 0.974074063571311"

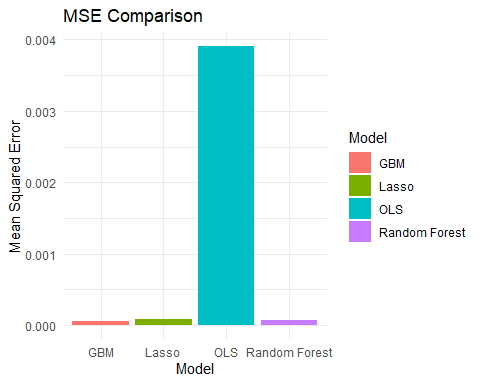
Compare the 4 models based on the evaluation metrics (MSE, RMSE, and R-squared),

Lower MSE and RMSE indicate better model performance. Higher R-squared indicates better model performance. After running the code, you can visually compare the performance of the three models. The model with the lowest MSE and RMSE and the highest R-squared would be the recommended choice.

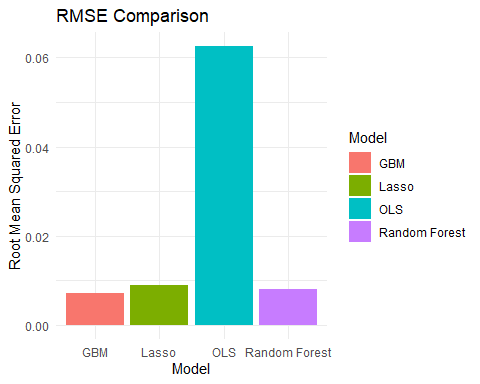
# Create a data frame to store the results  
results <- data.frame(  
 Model = c("OLS", "Random Forest", "Lasso", "GBM"),  
 MSE = c(mse\_ols, mse\_rf, mse\_lasso,mse\_gbm),  
 RMSE = c(rmse\_ols, rmse\_rf, rmse\_lasso,rmse\_gbm),  
 R\_squared = c(r2\_ols, r2\_rf, r2\_lasso,r2\_gbm)  
)  
  
# Print the results  
print(results)

## Model MSE RMSE R\_squared  
## 1 OLS 3.906942e-03 0.062505532 0.9576278  
## 2 Random Forest 6.581323e-05 0.008112535 0.9662536  
## 3 Lasso 7.981396e-05 0.008933866 0.9590746  
## 4 GBM 5.056160e-05 0.007110668 0.9740741

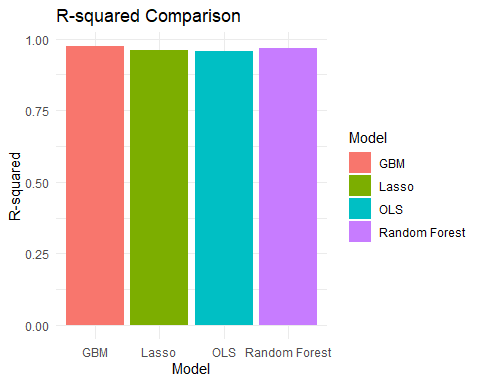
# Plot the results  
library(ggplot2)  
  
# MSE Comparison  
ggplot(results, aes(x = Model, y = MSE, fill = Model)) +  
 geom\_bar(stat = "identity") +  
 labs(title = "MSE Comparison", x = "Model", y = "Mean Squared Error") +  
 theme\_minimal()



# RMSE Comparison  
ggplot(results, aes(x = Model, y = RMSE, fill = Model)) +  
 geom\_bar(stat = "identity") +  
 labs(title = "RMSE Comparison", x = "Model", y = "Root Mean Squared Error") +  
 theme\_minimal()



# R-squared Comparison  
ggplot(results, aes(x = Model, y = R\_squared, fill = Model)) +  
 geom\_bar(stat = "identity") +  
 labs(title = "R-squared Comparison", x = "Model", y = "R-squared") +  
 theme\_minimal()



# Create a data frame with the results  
results <- data.frame(  
 Model = c("OLS", "Random Forest", "Lasso", "GBM"),  
 MSE = c(3.906942e-03, 6.581323e-05, 7.981396e-05, 5.056160e-05),  
 RMSE = c(0.062505532, 0.008112535, 0.008933866, 0.007110668),  
 R\_squared = c(0.9576278, 0.9662536, 0.9590746, 0.9740741)  
)  
  
# Write the data frame to an Excel file  
#write.xlsx(results, "model\_comparison.xlsx")

Analysis 1. Mean Squared Error (MSE): a. GBM has the lowest MSE (5.056160e-05), indicating the best performance in terms of prediction accuracy. b. Forest follows with an MSE of 6.581323e-05. c, Lasso has an MSE of 7.981396e-05. d, OLS has the highest MSE (3.906942e-03), indicating the least accurate predictions among the models.

1. Root Mean Squared Error (RMSE): a, GBM again has the lowest RMSE (0.007110668), reinforcing its superior performance. b, Random Forest has an RMSE of 0.008112535.
2. Lasso has an RMSE of 0.008933866. d, OLS has the highest RMSE (0.062505532).
3. R-squared:
4. GBM has the highest R-squared (0.9740741), indicating the best fit to the data.
5. Random Forest follows with an R-squared of 0.9662536.
6. Lasso has an R-squared of 0.9590746.
7. OLS has the lowest R-squared (0.9576278).

Recommendation Based on the evaluation metrics, the GBM (Gradient Boosting Machine) model performs the best, with the lowest MSE and RMSE and the highest R-squared. Therefore, I recommend using the GBM model for our analysis.

# Combine training and testing data for cross-validation  
data <- rbind(qol\_train, qol\_test)  
  
# Define cross-validation method  
train\_control <- trainControl(method = "cv", number = 10) # 10-fold cross-validation  
  
# OLS Model  
ols\_model <- train(pct\_poor\_to\_fair\_health ~ ., data = data, method = "lm", trControl = train\_control)  
# Predicted values and residuals for OLS model  
ols\_predicted <- predict(ols\_model, data)  
ols\_residuals <- data$pct\_poor\_to\_fair\_health - ols\_predicted  
  
# Random Forest Model  
rf\_model <- train(pct\_poor\_to\_fair\_health ~ ., data = data, method = "rf", trControl = train\_control, ntree = 1000)  
# Predicted values and residuals for Random Forest model  
rf\_predicted <- predict(rf\_model, data)  
rf\_residuals <- data$pct\_poor\_to\_fair\_health - rf\_predicted  
  
# Lasso Model  
lasso\_model <- train(pct\_poor\_to\_fair\_health ~ ., data = data, method = "glmnet", trControl = train\_control, tuneGrid = expand.grid(alpha = 1, lambda = seq(0.001, 0.1, by = 0.001)))

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,  
## : There were missing values in resampled performance measures.

# Predicted values and residuals for Lasso model  
lasso\_predicted <- predict(lasso\_model, data)  
lasso\_residuals <- data$pct\_poor\_to\_fair\_health - lasso\_predicted  
  
# GBM Model  
gbm\_model <- train(pct\_poor\_to\_fair\_health ~ ., data = data, method = "xgbTree", trControl = train\_control, tuneGrid = expand.grid(nrounds = 100, max\_depth = 6, eta = 0.1, gamma = 0, colsample\_bytree = 0.8, min\_child\_weight = 1, subsample = 0.8))  
# Predicted values and residuals for GBM model  
gbm\_predicted <- predict(gbm\_model, data)  
gbm\_residuals <- data$pct\_poor\_to\_fair\_health - gbm\_predicted  
  
# Create ggplot objects for each model  
p1 <- ggplot(data, aes(x = ols\_predicted, y = pct\_poor\_to\_fair\_health)) +  
 geom\_point() +  
 geom\_abline(slope = 1, intercept = 0, color = "red") +  
 labs(title = "OLS: Predicted vs Actual Values", x = "Predicted", y = "Actual")  
  
p2 <- ggplot(data, aes(x = ols\_predicted, y = ols\_residuals)) +  
 geom\_point() +  
 geom\_hline(yintercept = 0, color = "red") +  
 labs(title = "OLS: Residuals vs Predicted Values", x = "Predicted", y = "Residuals")  
  
p3 <- ggplot(data, aes(x = rf\_predicted, y = pct\_poor\_to\_fair\_health)) +  
 geom\_point() +  
 geom\_abline(slope = 1, intercept = 0, color = "red") +  
 labs(title = "Random Forest: Predicted vs Actual Values", x = "Predicted", y = "Actual")  
  
p4 <- ggplot(data, aes(x = rf\_predicted, y = rf\_residuals)) +  
 geom\_point() +  
 geom\_hline(yintercept = 0, color = "red") +  
 labs(title = "Random Forest: Residuals vs Predicted Values", x = "Predicted", y = "Residuals")  
  
p5 <- ggplot(data, aes(x = lasso\_predicted, y = pct\_poor\_to\_fair\_health)) +  
 geom\_point() +  
 geom\_abline(slope = 1, intercept = 0, color = "red") +  
 labs(title = "Lasso: Predicted vs Actual Values", x = "Predicted", y = "Actual")  
  
p6 <- ggplot(data, aes(x = lasso\_predicted, y = lasso\_residuals)) +  
 geom\_point() +  
 geom\_hline(yintercept = 0, color = "red") +  
 labs(title = "Lasso: Residuals vs Predicted Values", x = "Predicted", y = "Residuals")  
  
p7 <- ggplot(data, aes(x = gbm\_predicted, y = pct\_poor\_to\_fair\_health)) +  
 geom\_point() +  
 geom\_abline(slope = 1, intercept = 0, color = "red") +  
 labs(title = "GBM: Predicted vs Actual Values", x = "Predicted", y = "Actual")  
  
p8 <- ggplot(data, aes(x = gbm\_predicted, y = gbm\_residuals)) +  
 geom\_point() +  
 geom\_hline(yintercept = 0, color = "red") +  
 labs(title = "GBM: Residuals vs Predicted Values", x = "Predicted", y = "Residuals")  
  
# Combine plots into one  
library(gridExtra)

##   
## Attaching package: 'gridExtra'

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

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

grid.arrange(p1, p2, p3, p4, p5, p6, p7, p8, ncol = 2)

