# Load data

library(readxl)

library(foreign)

library(rms)

library(rrtable) # Output table as Word document

library(ggplot2) # Save images

library(psych)

library(caret)

library(Hmisc)

library(psych)

library(car)

setwd('C:/Users/Administrator/Desktop/DementiaYing/results/3.frontier/fixed/data\_filled.xlsx')

data\_filled <- read\_excel("C:/Users/Administrator/Desktop/DementiaYing/results/3.frontier/fixed/data\_filled.xlsx")

completed\_data <- as.data.frame(data\_filled)

table(completed\_data$Dementia)

for (col in names(completed\_data)) {

 completed\_data[[col]] <- as.numeric(as.factor(completed\_data[[col]]))

} # Convert all variables from factors to numeric

table(completed\_data$Dementia)

completed\_data$Dementia <- ifelse(completed\_data$Dementia == 2, 1, ifelse(completed\_data$Dementia == 1, 0, completed\_data$Dementia))

x1 <- as.matrix(completed\_data[, -1])

y1 <- completed\_data$Dementia

library(MASS)

# Random number seed

set.seed(123) # Random number generator

ind <- sample(2, nrow(completed\_data), replace = TRUE, prob = c(0.7, 0.3))

# Split dataset

train1 <- completed\_data[ind == 1, ] # The training data set

test1 <- completed\_data[ind == 2, ]

str(test1) # Confirm it worked

str(train1) # Confirm it worked

train\_x1 <- as.matrix(train1[,-1])

train\_y1 <- train1$Dementia

test\_x1 <- as.matrix(test1[,-1])

test\_y1 <- test1$Dementia

#### Lambda minimum corresponding to the number of variables

library(glmnet)

lasso <- glmnet(train\_x1, train\_y1, family = "binomial", alpha = 1) # y is continuous then use gaussian

print(lasso)

#### Draw working plot

# par(lwd=2, font=2) # lwd controls line thickness, font controls font (2 means bold)

plot(lasso, xvar = "lambda", label = TRUE)

##### Compute regression coefficients

lasso.coef <- predict(lasso, s = 0.000180, type = "coefficients") # s is the lambda from above

lasso.coef

####### Cross-validation

set.seed(317)

train\_y1 <- as.numeric(train\_y1)

lasso.cv = cv.glmnet(train\_x1, train\_y1, nfolds = 10) # 10-fold

plot(lasso.cv)

lasso.cv$lambda.min # minimum, check this for this dataset

lasso.cv$lambda.1se # one standard error away

coef(lasso.cv, s = "lambda.min") # If lambda.1se is better, choose that one

############################################################# Factor Variables

completed\_data$Gender <- as.factor(completed\_data$性别)

completed\_data$PhysicalDisability <- as.factor(completed\_data$身体残疾)

completed\_data$HearingAbnormality <- as.factor(completed\_data$听力异常)

completed\_data$Hyperlipidemia <- as.factor(completed\_data$高血脂)

completed\_data$HeartDisease <- as.factor(completed\_data$心脏病)

completed\_data$AttentionDeficit <- as.factor(completed\_data$注意力不集中)

completed\_data$Hope <- as.factor(completed\_data$希望)

completed\_data$SelfRatedHealth <- as.factor(completed\_data$自评健康状况)

completed\_data$SleepDuration <- as.factor(completed\_data$睡眠时长)

completed\_data$Napping <- as.factor(completed\_data$午睡)

completed\_data$EducationLevel <- as.factor(completed\_data$文化程度)

completed\_data$Residence <- as.factor(completed\_data$居住地)

completed\_data$Socialization <- as.factor(completed\_data$社交)

completed\_data$ModeratePhysicalActivity <- as.factor(completed\_data$中度体力活动)

completed\_data$LightPhysicalActivity <- as.factor(completed\_data$轻度体力活动)

completed\_data$HealthInsurance <- as.factor(completed\_data$医疗保险)

# Ensure dependent variable "Dementia" is either numeric or factor

completed\_data$Dementia <- as.factor(completed\_data$Dementia)

# Fit logistic regression model

fit1 <- glm(Dementia ~ Gender + PhysicalDisability + HearingAbnormality + Hyperlipidemia + HeartDisease + AttentionDeficit + Hope +

 SelfRatedHealth + SleepDuration +

 Napping + EducationLevel + Residence + Socialization + ModeratePhysicalActivity + LightPhysicalActivity + HealthInsurance,

 data = completed\_data, family = "binomial")

fit1

# Stepwise regression based on AIC for variable selection

# fit2 <- step(fit1, direction = "backward", trace = FALSE)

# fit2

library(questionr)

# Calculate odds ratios for selected model

orresults <- odds.ratio(fit1)

# Convert odds ratio results to dataframe

ordf <- as.data.frame(orresults)

# View dataframe

print(ordf)

# Fit logistic regression model

fit3 <- glm(Dementia ~ AttentionDeficit + SelfRatedHealth

 + Napping + EducationLevel + Residence + Socialization + HealthInsurance,

 data = completed\_data, family = "binomial")

# Variance inflation factor (VIF)

vif.values <- vif(fit3)

# Output results

print(vif.values)

########################################################################

table(completed\_data$Dementia)

dd <- datadist(completed\_data)

options(datadist = 'dd')

#### Nomogram

library(rms)

# Fit logistic regression model

full <- lrm(Dementia ~ AttentionDeficit + SelfRatedHealth + Napping + EducationLevel + Residence + Socialization + HealthInsurance, data = completed\_data)

# Create nomogram

nom1 <- nomogram(full, fun = plogis, fun.at = c(.001, .01, .05, seq(.1, .9, by = .1), .95, .99, .999), lp = F, funlabel = "rate")

par(mar = c(5, 8, 2, 2), family = "Times New Roman")

plot(nom1, col.grid = gray(c(0.8, 0.95)))

# Evaluation

train <- subset(train1, select = c("Dementia", "AttentionDeficit", "SelfRatedHealth", "Napping", "EducationLevel"

 ,"Residence", "Socialization", "HealthInsurance"))

test <- subset(test1, select = c("Dementia", "AttentionDeficit", "SelfRatedHealth", "Napping", "EducationLevel"

 ,"Residence", "Socialization", "HealthInsurance"))

dd <- datadist(train)

options(datadist = 'dd')

dd <- datadist(test)

options(datadist = 'dd')

train\_x <- train[,-1]

train\_y <- train1$Dementia

test\_x <- test[,-1]

test\_y <- test1$Dementia

# Convert all categorical variables to factors

train$Dementia <- as.factor(train$Dementia)

# Convert categorical variables to numeric

train$AttentionDeficit <- as.numeric(train$AttentionDeficit)

train$SelfRatedHealth <- as.numeric(train$SelfRatedHealth)

train$Napping <- as.numeric(train$Napping)

train$EducationLevel <- as.numeric(train$EducationLevel)

train$Residence <- as.numeric(train$Residence)

train$Socialization <- as.numeric(train$Socialization)

train$HealthInsurance <- as.numeric(train$HealthInsurance)

# Convert all categorical variables to factors

test$Dementia <- as.factor(test$Dementia)

# Convert categorical variables to numeric

test$AttentionDeficit <- as.numeric(test$AttentionDeficit)

test$SelfRatedHealth <- as.numeric(test$SelfRatedHealth)

test$Napping <- as.numeric(test$Napping)

test$EducationLevel <- as.numeric(test$EducationLevel)

test$Residence <- as.numeric(test$Residence)

test$Socialization <- as.numeric(test$Socialization)

test$HealthInsurance <- as.numeric(test$HealthInsurance)

## ROC

full.fit <- glm(Dementia ~ AttentionDeficit + SelfRatedHealth

 + Napping + EducationLevel + Residence + Socialization + HealthInsurance, data = train, family = "binomial")

full.fit1 <- glm(Dementia ~ AttentionDeficit + SelfRatedHealth

 + Napping + EducationLevel + Residence + Socialization + HealthInsurance, data = test, family = "binomial")

# Generate predicted probabilities

train$predicted\_prob <- predict(full.fit, type = "response")

test$predicted\_prob <- predict(full.fit1, type = "response")

# Calculate ROC curve for each predictor

roc\_model <- roc(train$Dementia, train$predicted\_prob)

roc\_attention <- roc(train$Dementia, train$AttentionDeficit)

roc\_health <- roc(train$Dementia, train$SelfRatedHealth)

roc\_nap <- roc(train$Dementia, train$Napping)

roc\_education <- roc(train$Dementia, train$EducationLevel)

roc\_residence <- roc(train$Dementia, train$Residence)

roc\_social <- roc(train$Dementia, train$Socialization)

roc\_insurance <- roc(train$Dementia, train$HealthInsurance)

auc\_value

auc\_value1 <- auc(roc\_model)

# Plot ROC curves

plot(roc\_model, col = "black", main = "ROC Curves")

lines(roc\_attention, col = "red")

lines(roc\_health, col = "blue")

lines(roc\_nap, col = "green")

lines(roc\_education, col = "purple")

lines(roc\_residence, col = "orange")

lines(roc\_social, col = "pink")

lines(roc\_insurance, col = "brown")

# Add light gray grid lines

grid(col = "gray70") # Light gray grid lines

# Adjust legend position and add space

legend(x = 0.55, y = 0.35, cex = 0.5, text.font = 2, legend = c(

 paste("Dementia Model (AUC =", round(auc(roc\_model), 3), ")"),

 paste("Lack of concentration (AUC =", round(auc(roc\_attention), 3), ")"),

 paste("Self-assessed health status (AUC =", round(auc(roc\_health), 3), ")"),

 paste("Napping (AUC =", round(auc(roc\_nap), 3), ")"),

 paste("Educational level (AUC =", round(auc(roc\_education), 3), ")"),

 paste("Residence (AUC =", round(auc(roc\_residence), 3), ")"),

 paste("Socialization (AUC =", round(auc(roc\_social), 3), ")"),

 paste("Medical Insurance (AUC =", round(auc(roc\_insurance), 3), ")")

),

col = c("black", "red", "blue", "green", "purple", "orange", "pink", "brown"),

lwd = 2, inset = c(0.005, 0.005), bty = "n" # bty="n" removes the box, inset adjusts distance

)

# Calculate ROC curve for test data

roc\_model <- roc(test$Dementia, test$predicted\_prob)

roc\_attention <- roc(test$Dementia, test$AttentionDeficit)

roc\_health <- roc(test$Dementia, test$SelfRatedHealth)

roc\_nap <- roc(test$Dementia, test$Napping)

roc\_education <- roc(test$Dementia, test$EducationLevel)

roc\_residence <- roc(test$Dementia, test$Residence)

roc\_social <- roc(test$Dementia, test$Socialization)

roc\_insurance <- roc(test$Dementia, test$HealthInsurance)

auc\_value2 <- auc(roc\_model)

# Plot ROC curves

plot(roc\_model, col = "black", main = "ROC Curves")

lines(roc\_attention, col = "red")

lines(roc\_health, col = "blue")

lines(roc\_nap, col = "green")

lines(roc\_education, col = "purple")

lines(roc\_residence, col = "orange")

lines(roc\_social, col = "pink")

lines(roc\_insurance, col = "brown")

# Add light gray grid lines

grid(col = "gray70") # Light gray grid lines

# Adjust legend position and add space

legend(x = 0.55, y = 0.35, cex = 0.5, text.font = 2, legend = c(

 paste("Dementia Model (AUC =", round(auc(roc\_model), 3), ")"),

 paste("Lack of concentration (AUC =", round(auc(roc\_attention), 3), ")"),

 paste("Self-assessed health status (AUC =", round(auc(roc\_health), 3), ")"),

 paste("Napping (AUC =", round(auc(roc\_nap), 3), ")"),

 paste("Educational level (AUC =", round(auc(roc\_education), 3), ")"),

 paste("Residence (AUC =", round(auc(roc\_residence), 3), ")"),

 paste("Socialization (AUC =", round(auc(roc\_social), 3), ")"),

 paste("Medical Insurance (AUC =", round(auc(roc\_insurance), 3), ")")

),

col = c("black", "red", "blue", "green", "purple", "orange", "pink", "brown"),

lwd = 2, inset = c(0.005, 0.005), bty = "n" # bty="n" removes the box, inset adjusts distance

)

# Calibration curve

c\_train <- lrm(Dementia ~ AttentionDeficit + SelfRatedHealth

 + Napping + EducationLevel + Residence + Socialization + HealthInsurance, data = train, x = T, y = T)

c\_test <- lrm(Dementia ~ AttentionDeficit + SelfRatedHealth

 + Napping + EducationLevel + Residence + Socialization + HealthInsurance, data = test, x = T, y = T)

# Use calibrate() to draw calibration curve

cal1 <- calibrate(c\_train, method = "boot", B = 1000) # B is the number of bootstraps

plot(cal1, main = "Calibration Curve")

cal2 <- calibrate(c\_test, method = "boot", B = 1000) # B is the number of bootstraps

plot(cal2, main = "Calibration Curve")

# Decision Curve Analysis (DCA)

library(rmda)

train$Dementia <- as.numeric(train$Dementia) - 1 # Convert "No" to 0, "Yes" to 1

# Refit the model

model1 <- decision\_curve(Dementia ~ AttentionDeficit + SelfRatedHealth +

 Napping + EducationLevel + Residence + Socialization + HealthInsurance,

 family = binomial(link = 'logit'),

 data = train,

 thresholds = seq(0, .8, by = .05),

 bootstraps = 10)

# Plot DCA curve

plot\_decision\_curve(model1,

 curve.names = "Model",

 col = c("red"),

 confidence.intervals = F # Do not show confidence intervals

)

test$Dementia <- as.numeric(test$Dementia) - 1 # Convert "No" to 0, "Yes" to 1

# Refit the model

model2 <- decision\_curve(Dementia ~ AttentionDeficit + SelfRatedHealth +

 Napping + EducationLevel + Residence + Socialization + HealthInsurance,

 family = binomial(link = 'logit'),

 data = test,

 thresholds = seq(0, .8, by = .05),

 bootstraps = 10)

# Plot DCA curve

plot\_decision\_curve(model2,

 curve.names = "Model",

 col = c("red"),

 confidence.intervals = F # Do not show confidence intervals

)