**Afraid of Needles? Machine Learning Might Help to Determine Your LDL Levels**

Brianne Bell, Hunter Blum, and Eva Chow

ADS503 - Applied Predictive Modeling

University of San Diego

**Abstract**

LDL or “bad” cholesterol can lead to fatty deposits in blood vessels, causing angina, heart attack and stroke (MedlinePlus, 2018; Mayo Foundation, 2021). Despite the dangers, LDL cholesterol testing can be expensive and cause anxiety in patients who are uncomfortable with needles. This project aims to utilize a set of demographic and body size measurements to create a model to warn patients if their cholesterol is dangerously high. The data was collected from the National Health and Nutrition Examination Survey pre-pandemic data set. Multiple training data sets were used throughout the project, but each was normalized, imputed, and constructed from 80% of the observations. Five linear regression methods were employed to train the model using the various training data sets, with the ridge regression having the best RMSE of 32.9 mg/dL when predicted on the test data. Next five non-linear models were trained, the best being the radial SVM with an RMSE of 32.95 mg/dL. Finally, six tree-based models were created. The tree-based cubist was the best overall, with an RMSE of 32.64 mg/dL. Due to the high RMSE, the performance of the models was deemed to be insufficient for use in any medical setting. The cubist model predictions lacked the variance in the original observations needed to create a more accurate model (Figure 4.1). Future studies should focus on adding more easily measured predictors, such as blood pressure, to increase variance and improve model performance.

**Introduction**

According to the Mayo Clinic (2021), high levels of LDL cholesterol can lead to deposits building up in the blood vessels. Cholesterol has two sides, the good HDL (high-density lipoproteins) which helps carry cholesterol back to the liver for removal. LDL (low-density lipoprotein) is the “bad” part of the general cholesterol group since it is the form that can build up in arteries (MedlinePlus, 2018). These deposits can lead to complications such as angina, heart attack, and stroke (Mayo Foundation, 2021). Complications from cholesterol cause an estimated 2.6 million deaths annually worldwide (World Health Organization, 2012). Due to the dangers of high cholesterol, the CDC recommends that adults get their levels tested every 4-6 years (CDC, 2019). However, visiting the doctor for an LDL cholesterol test can have high costs and drawing blood can be uncomfortable for the patient. The goal of this project was to utilize a set of demographic and easily attainable body size measurements to create a model that accurately estimated the LDL cholesterol of a patient. With this model, patients can be notified if they should seek a more robust test of their cholesterol levels from their physician.

**Data Preparation**

**Exploratory Data Analysis**

The data was collected from three data sets from the National Health and Nutrition Examination Survey (NHANES) pre-pandemic data set, which included observations from 2017 to March of 2020. The first data set focused on demographic data and included variables such as age, gender, ethnicity, language, and income. The project’s next data set focused on body measurements such as height, weight, BMI, and various lengths and circumferences of body parts. Finally, a third data set provided LDL cholesterol levels (mg/dL). The final data set was created by left-joining the two prior data sets to the complete observations of the cholesterol data set. Variables relating to the physician’s comments were removed, leaving the data with 4,617 observations and 27 variables.

When evaluating the missing observations for each variable, six predictors had over 3,000 observations missing, with age in months, recumbent length, and head circumference all missing. No other variables had over 1,000 missing observations. These six predictors were then removed from the data set and the missing cells for each row was calculated. The maximum number of missing cells for one row was 12, so the remaining data was kept for later imputation.

Histograms were generated utilizing ggplot2 (Wickham, 2016) to evaluate the distribution of each numeric variable. Figure 2.1 shows that LDL cholesterol has a distribution skewed slightly in the positive direction, and a skewness value of 0.79 confirms this finding. Age (RIDAGEYR) has a relatively uniform distribution with spikes every 5 to 10 years. Family income ratio (IMDFMPIR) is centered around a mode slightly above one and is heavily skewed to the right (Figure 2.2). Most body measurement predictors were relatively normal, however weight (BMXWT), BMI (BMXBMI), arm circumference (BMXARMC), and hip circumference (BMXHIP) were skewed in the positive direction (Figure 2.3).

Bar plots were also created using ggplot2 to evaluate the class balance of each categorical predictor. For language predictors, English observations dominated the counts of each class (Figure 2.4). Other categorical demographic variables were evaluated in Figure 2.5, with gender (RIAGENDR) being the only balanced predictor.

Next, categorical features were one-hot encoded using the dummy\_cols function (Kaplan, 2020) to create a data set for modeling. This data set was used to evaluate the correlation between each variable. Cholesterol showed no strong correlation with any variable (Figure 2.6 and 2.7). Language variables exhibited a strong positive correlation when representing the same language and a strong negative correlation with different languages (Figure 2.6). The male class showed a strong positive correlation to leg circumference, height, and arm length, while the female class displayed the opposite relationship. Many body measurements were strongly, directly correlated such as BMI, arm circumference, hip circumference, and waist circumference (Figure 2.7)

Chart, histogram

Description automatically generated

Figure 2.1. Histogram of target variable, LDL cholesterol.

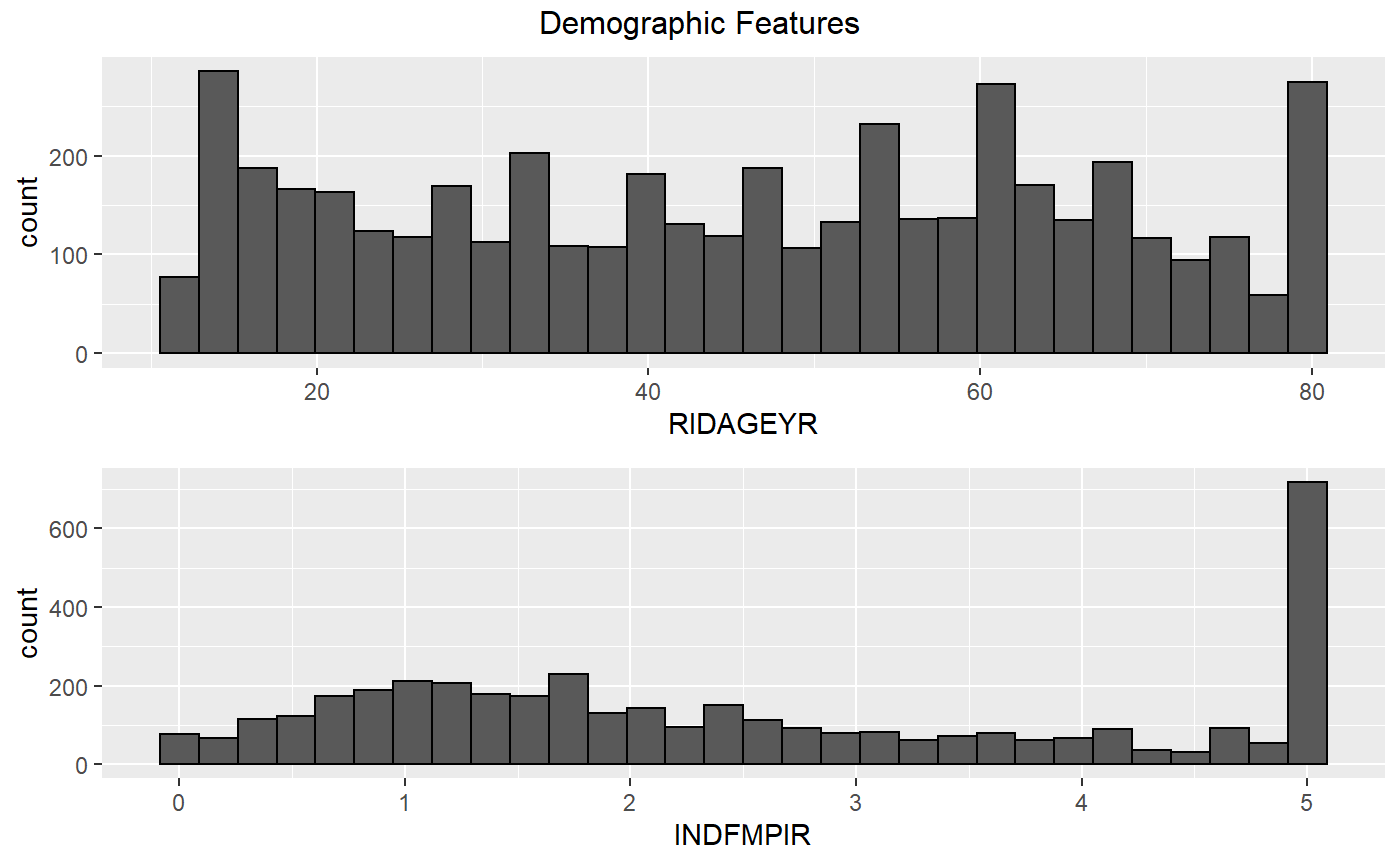
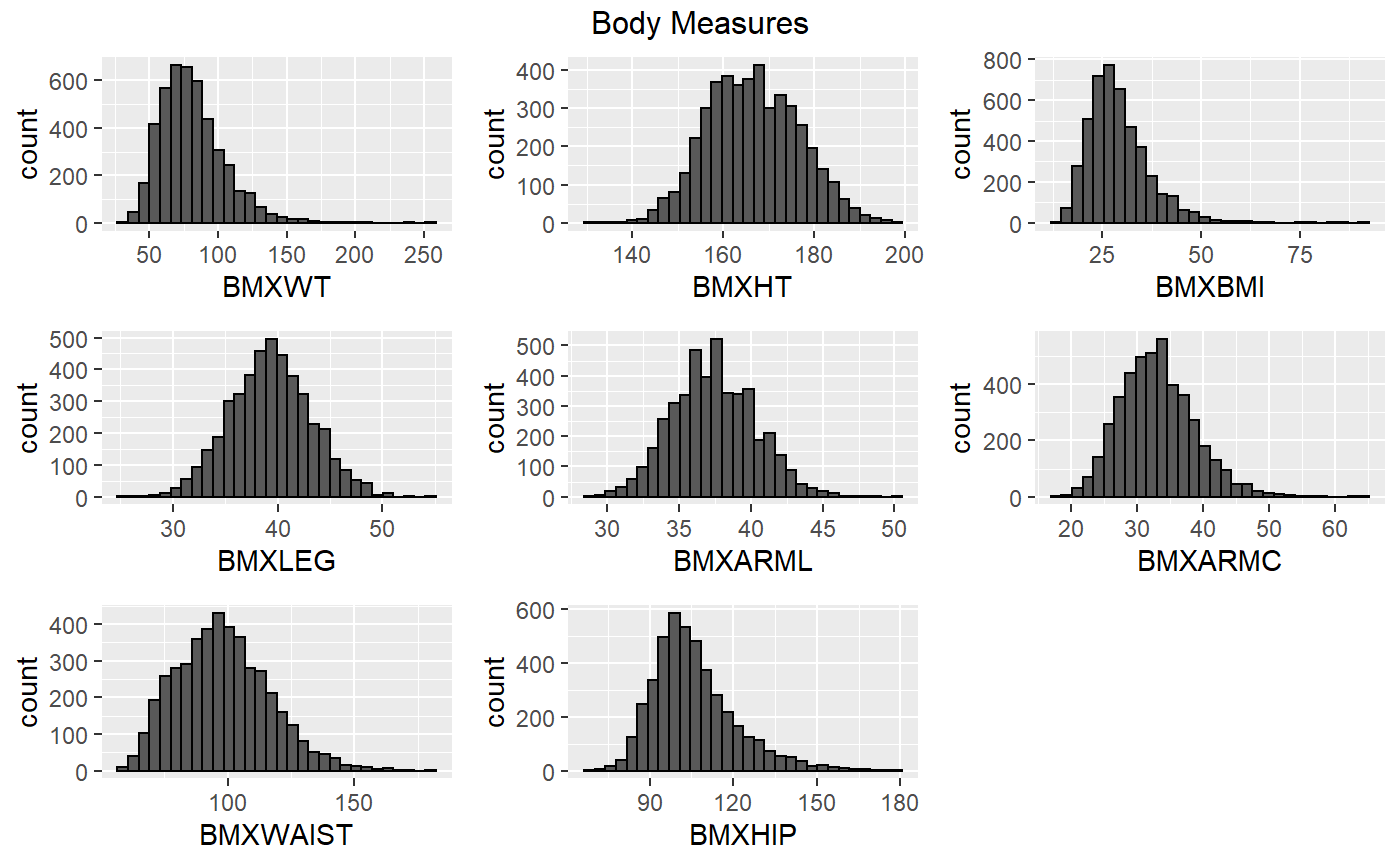


Figure 2.3. Histograms of numeric body measurement features.

Figure 2.2. Histograms of numeric demographic predictors.

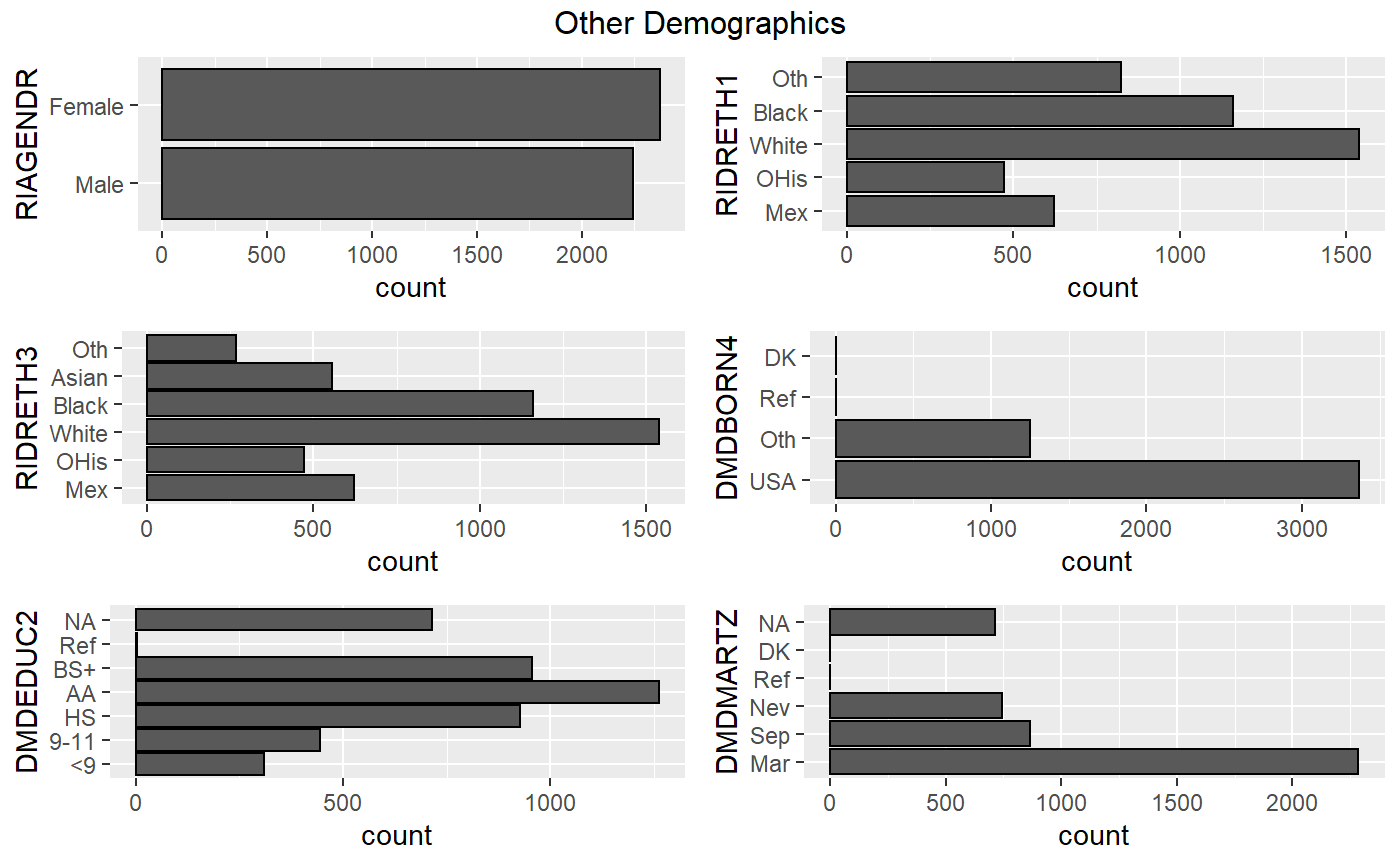
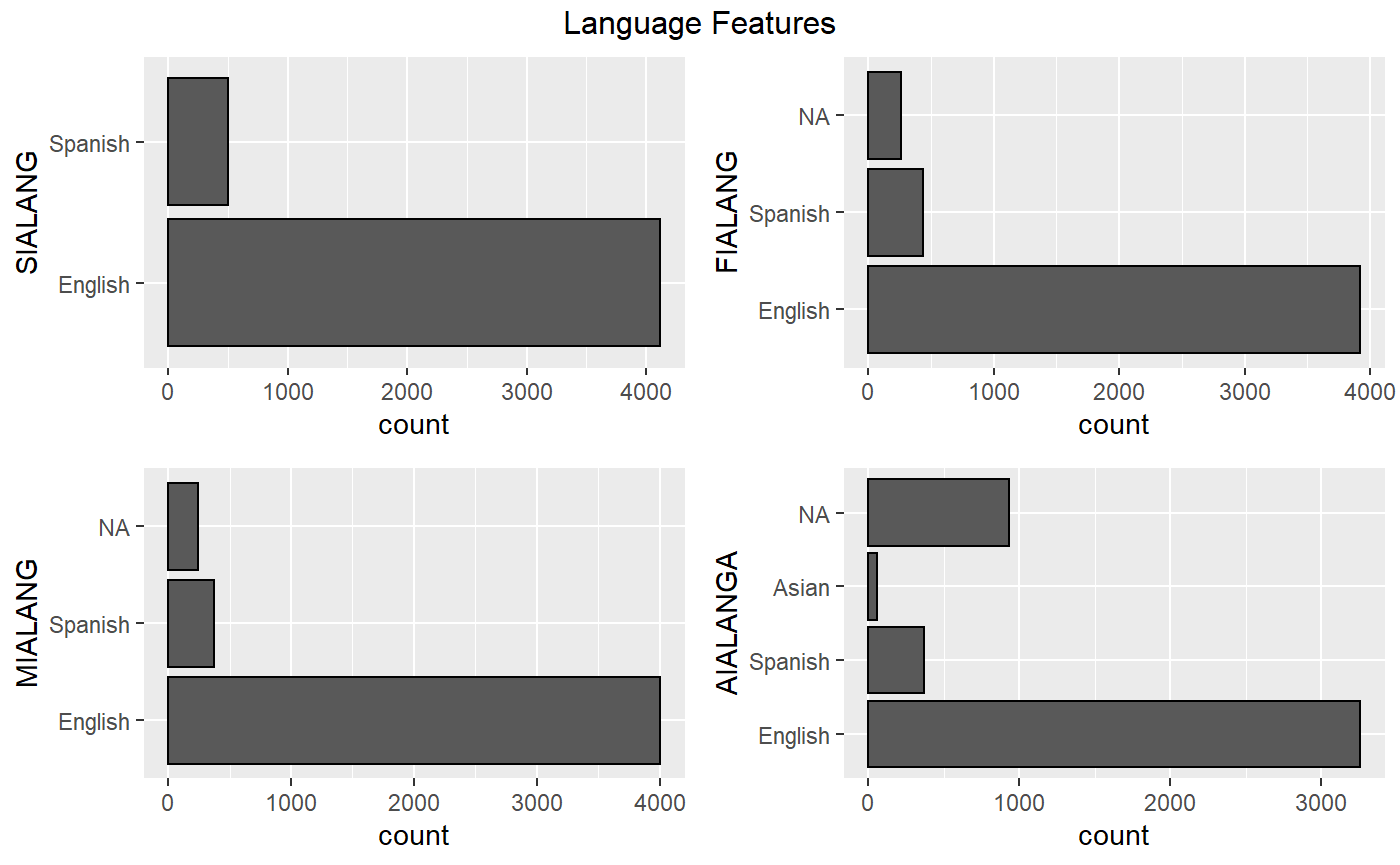


Figure 2.5. Bar plots of non-language, demographic feature classes.

Figure 2.4. Bar plots of language feature classes.

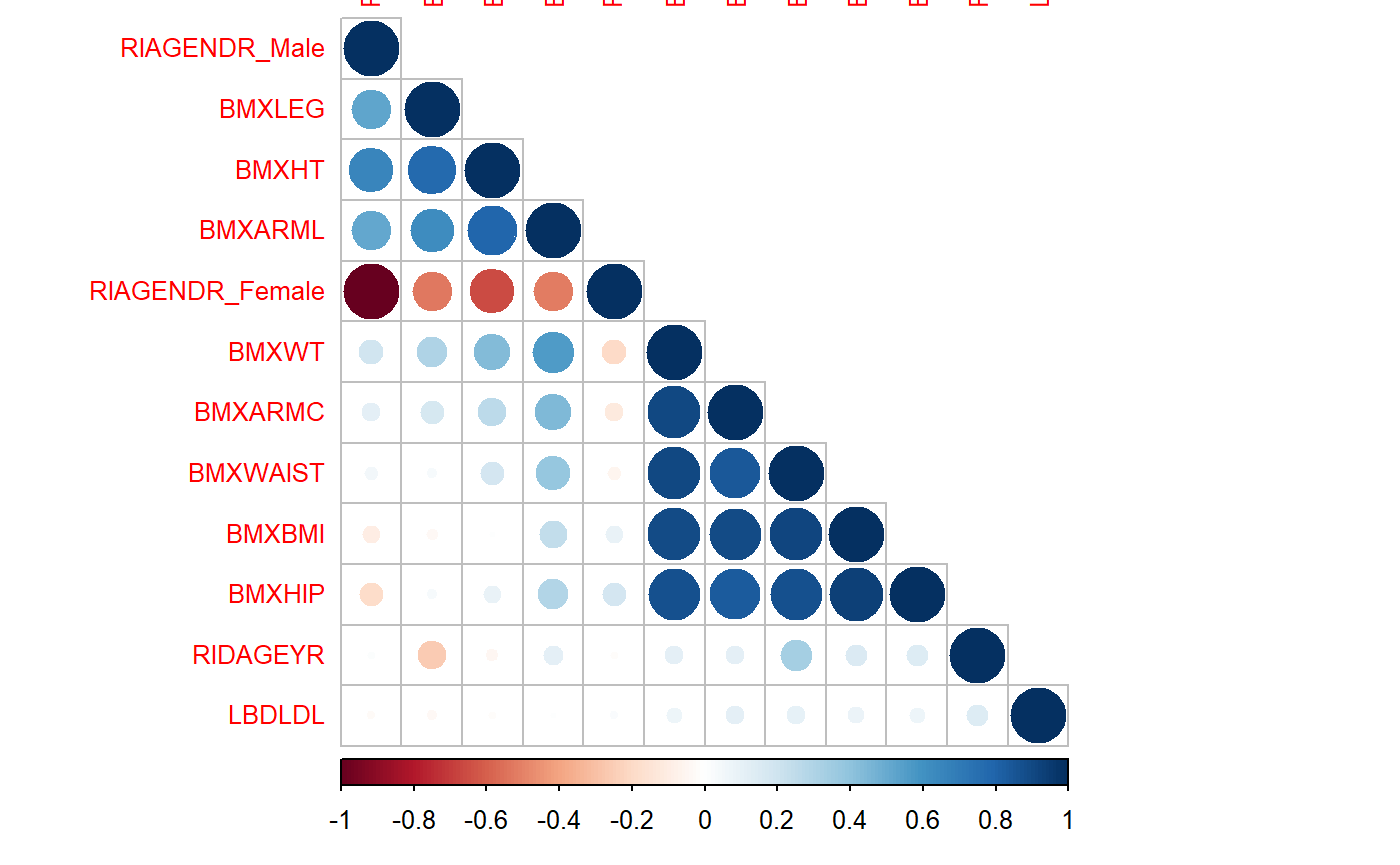
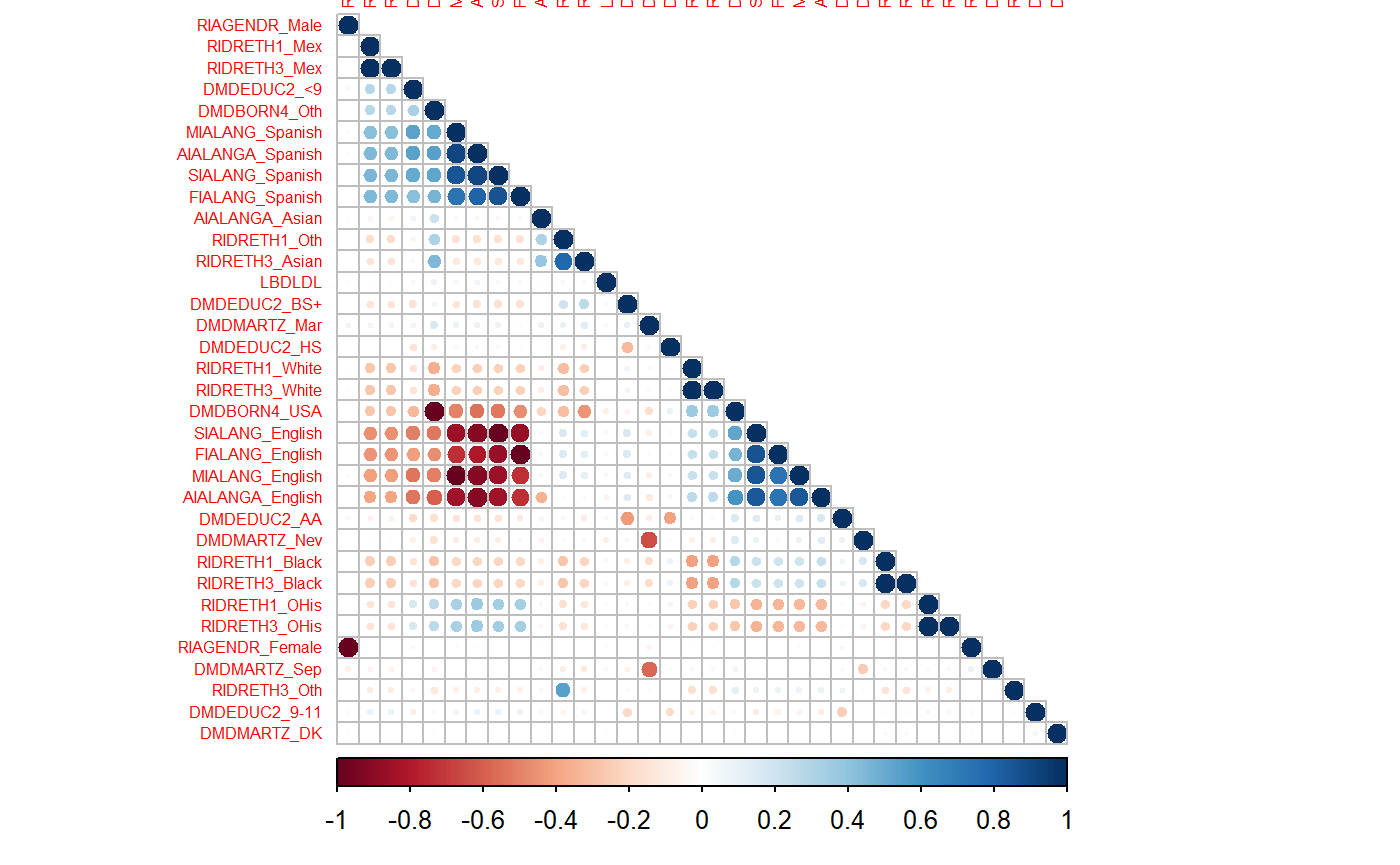


Figure 2.7. Correlations between LDL cholesterol and body measurement predictors.

Figure 2.6. Correlations between LDL cholesterol and demographic predictors.

**Data Pre-Processing and Data Splitting**

To split the data, a stratified random sample was taken from the data set with dummy variables, with 80 percent of observations going to the training data set. The training and test predictors were processed by normalizing and then imputing using the five nearest neighbors. Ten fold cross-validation was utilized to evaluate tuning models. Finally, another data set was created using the same methods using the numeric variables and gender, in hopes of creating a simpler, interpretable model.

**Model Building**

**Linear Regression Models**

An Ordinary Least Squares Regression (OLS) model was created using all predictors in the training data using the “lm” method in caret (Kuhn, 2021). When evaluating diagnostic plots this model struggled with the normality predictors and collinearity, so predictors with above 0.9 Pearson’s correlation to each other were removed. This data set was utilized to create all other models and transformations for the remainder of the project. Another model was trained with the reduced data set, but this model also had issues with the normality of predictors. Two other models were trained, one using Box Cox transformations and another using the numeric training data, however each did nothing to change the issues with the QQ diagnostic plot. So, a final OLS model was trained using the significant predictors from the correlation filtered training data.

Three Principal Component Regression (PCR) and three Partial Least Squares Regression (PLS) were trained using three different data sets. The PCR models were trained using the “pcr” method in caret, while the PLS models were trained using the “pls” method in caret (Kuhn, 2021). One PCR and one PLS model was tuned on the training data evaluating 1-32 components. Another model was trained for each method in the same fashion utilizing the Box Cox transformed data. Finally, a model was created for each method using the numerical data evaluating 1-8 components. When evaluating resampled root mean squared error (RMSE), each model showed an increase in error as components increased.

Three Ridge Regression models were trained using the training, box cox, and numerical data sets and the “ridge” method in caret (Kuhn, 2021). Lambda values from 0 to 0.5 were evaluated for each model. For each model around 0.05 showed a RMSE around 40 mg/dL. After the penalty of 0.05, RMSE leveled off around the 40 mg/dL value. Finally, three Elastic Net models were trained using the same three data sets and the “enet” method and caret. These models were evaluated with lambdas of 0, 0.01, and 0.1 and a fraction of the full solution from 0.05 to 1. Each model exhibited the best RMSE with a lambda of 0.1 and had increases in RMSE with an increase in the fraction of the full solution.

**Non-Linear Regression Models**

Multiple non-linear regression models were developed and tuned including Support Vector Machine models (SVM), K-Nearest Neighbors, Multivariate Adaptive Regression Splines (MARS) model, and a Neural Network model (nnet). Through each initial rendition of the models, the predictors "MIALANG\_Spanish", "DMDEDUC2\_<9", "MIALANG\_NA" were causing issues of still having zero variance, despite the above-mentioned preprocessing of the training and testing data. As a result, these columns were dropped for each of these models, leaving 29 predictors to develop the models with. All of the models utilized a cross-validation system with ten fold cross validation.

The first SVM model explored was with the radial basis kernel function, designated as SVM-Radial or SVM-R. The radial kernel function has sigma as its scaling parameter that acts with the cost parameter of the SVM model (Kuhn & Johnson, 2013). Sigma values were explored utilizing the sigest function from the kernlab library of R, where it was able to explore the sigma values from the predictor training set. The cost parameter was explored as a sequence between 2-4 and 2+4.  During model tuning, the predictors were centered and scaled as a method to preprocess the data for optimal model development. The model was optimized to select the lowest RMSE model which ended up having a cost metric of .25, sigma value of .0350, and utilized 3377 support vectors, producing an RMSE value of 35 mg/dL.

The second SVM model explored was the polynomial based kernel function that incorporates a degree parameter and phi (φ) as a scaling parameter (Kuhn & Johnson, 2013). The degree parameter is the power of the polynomial and for this setting, a range of 1 to 3 degrees was explored. The scaling parameter was explored at three values, (.01, .005, .001, .0005). The cost parameter was explored in a range in the same range as the SVM-Radial model (2-4 and 2+4). Again, the parameters were centered and scaled as part of the model preprocessing method. The final model had epsilon at .1, cost at 2, degree at 2, and scale at .001. This model utilized 3366 support vectors to obtain an RMSE value of 35 mg/dL.

The K-Nearest Neighbor model had only the k value to optimize so that value was explored between one and 40. Additionally, the model went through preprocessing to center and scale the parameters. The optimal number of neighbors was determined to be 20 since it had the lowest RMSE value at 35 mg/dL of the k values explored. However, this RMSE value is neither worse nor better than the SVM models explored.

The MARS model has two tuning parameters: the degree that parameters are added and the number of retained terms (Kuhn & Johnson, 2013). The degree that parameters were added was explored at values of 1-3 while retention was explored as a range of pruning values of 2-38. Ultimately, the model selected two of the 18 terms and only one of the 29 predictors, resulting in an RMSE value of 35 mg/dL. This doesn’t place the MARS model as the best or worst trained model of the non-linear varieties so far.

The final non-linear model explored was the neural network model which is controlled by parameters for decay and size. Decay or weight decay, penalizing large regression coefficients while the size refers to the hidden layers (Kuhn & Johnson, 2013). The predictors were centered and scaled as part of the preprocessing then the model was applied over a grid of decay values (0, .01, .1) and size of (3, 7, 11, 13). The maximum number of weights was held at 404 (13\*(number of columns +1) +13+1) and the maximum number of iterations was held at 100. The final neural network model had 13 hidden layers (size), decay of .1 which resulted in an RMSE value of 43 mg/dL.  This is the highest RMSE of the non-linear training models and therefore, the worst model choice of nonlinear options.

The models went through a final evaluation of how well they predicted the LDL values. The actual predictions were stored and the RMSE of the tested models was calculated for comparisons. Of the non-linear models explored, the SVM-radial model had the lowest RMSE value at 33 mg/dL while the neural network was the worst with its RMSE value of 37 mg/dL.

**Regression Trees**

After modeling some linear and non-linear regression models, regression trees were tested. These included single regression trees, bagged trees, random forest, boosted trees, and model trees such as M5 and cubist. All the tree models mentioned were modeled using the imputed training set that was reduced to 32 variables, with missing values imputed using K-Nearest Neighbors imputation methods mentioned prior.

The first regression tree model that was explored was a single regression tree with CART based splits. In the first model, the tuning parameter that was adjusted was the complexity parameter (cp). By default, this value is set to 0.01 (Kassambra, 2018). An increase in this value reduces tree size, while a decrease in this value increases tree size. At the risk of overfitting the model, a cp value of 0.003 was selected. This resulted in an RMSE value of 33.0 mg/dL.

The second regression tree model also used CART based splits but used maxdepth to tune the model. For our model, a maxdepth of 6 was used to tune the model. This depth was chosen to reflect the number of splits that were seen in the regression model tuned using the complexity parameter. With an RMSE value of 32.9 mg/dL, the maxdepth tuned CART based model slightly outperformed the complexity parameter tuned model. For our final model comparisons, the CART model displayed will refer to the model tuned using maxdepth.

The next tree model tested was a bagged tree model. Tuning this model required indication of both the complexity parameter (cp) as well indicating how many bootstrapped replications will be performed using nbagg. Reflecting the complexity parameter used in the regression trees, a cp value of 0.003 was used. By default, nbagg is set to 25. An increase in this parameter will increase the number of bootstrapped repetitions and can increase model performance in exchange for computation power. For the model used, the nbagg value was tested by increasing the value by 5 from the default, noting minor improvements in model performance measured using RMSE values. By doing this, an optimal nbagg value of 70 was decided upon after no further improvements in RMSE values were seen. The final RMSE value for the bagged tree was 33.1 mg/dL.

Next, a Stochastic Gradient Boosting Tree model was used.  For this model, there were several parameters that could be tuned for optimal model performance. At the risk of overfitting, the number of trees (n.trees) was set to 300. The maximum number of nodes per tree (interaction.depth) was set to 1, 3, 5, 7, and 9 rather than the default of 6. Shrinkage is the model’s learning rate and was set to 0.01 and 0.1. The closer this value gets to 1, the more overfit the model will become. Finally, the last parameter that was tuned was the minimum number of observations in the terminal nodes (n.minobsinnode). This value was set to 5. The bag.fraction and train.fraction parameters were left to their default settings. With this, the final boosted tree model obtained an RMSE value of 32.9 mg/dL.

Model trees were evaluated next. Using both an M5 model tree and a rule based M5 model tree, the tuning parameter used was the M value within weka\_control. This value was set to 5. For the normal M5 model tree with no other tuning adjustments, an RMSE value of 33.6 mg/dL was obtained. The rule based M5 model obtained an RMSE of 32.8 mg/dL.

Finally, the last tree model that was evaluated was the cubist model. For this model, two tuning parameters are available. They are the number of committees and the number of instances. For model training, the default parameters were used, with instances of 0, 5, and 9 and the number of committees between 0 and 20 instances. This resulted in the best performing model with an RMSE value of 32.6 mg/dL with 0 instances and 10 committees.

**Results**

**Metrics of Evaluation**

All of the models that our team explored were compared within each model type and between model types using the RMSE.

Equation 1. Root Mean Squared Error.

Chart

Description automatically generatedThe final RMSE values for final model comparison were obtained from running the test set through each model and comparing the resulting predictions to the original test set for LDL.  The best linear model was the ridge regression with a RMSE of 32.87 mg/dL. The best non-linear model was the SVM-radial model with RMSE of 32.95 mg/dL. The best regression tree model was the cubist model with RMSE of 32.64 mg/dL, making it the best performing model overall (Figure 4.2). Plotting the cubist model predictions against the test data reveals that the model struggles to predict the variance of the original data (Figure 4.1).

Figure 4.1. Observed test LDL cholesterol (light blue) versus the cubist predicted LDL Cholesterol (dark blue).

**Chart

Description automatically generated**

Figure 4.2. RMSE of each model evaluated on the test data.

**Discussion**

The models did not perform well and therefore are not recommended for any medical or home use. According to the U.S. National Library of Medicine (n.d.), less than 100 mg/dL is optimal and over 160 mg/dL is high. Given that the RMSE of the best model was over 30 mg/dL, a prediction by the model could egregiously misguide a patient on their true cholesterol levels. Since the model exhibited high bias in Figure 4.1,the best recommendation to improve the performance is to utilize other predictors to introduce more variance into the training process. Perhaps, another easily acquired measurement such as blood pressure would help the performance of a future model.  Another particularly useful enhancement would be to make the process interactive through RStudio’s Shiny application feature (*Shiny - Tutorial*, n.d.). By developing and sharing the Shiny application, people could input the relevant data (body measurements, demographic data) to determine if they should be concerned about LDL cholesterol levels. Additionally, clinics could also use the application to help screen patients, particularly those with needle phobias or other complications to blood draw techniques.

**Conclusion**

Having high cholesterol is not good in and of itself but having high cholesterol is often a contributing factor to many health conditions that greatly impact quality of life and life expectancy. While checking LDL levels is easy enough in theory, there are compounding issues of people having phobias of needles or blood, there is also the issue of cost. Blood draws and the accompanying laboratory tests are typically not cheap and the costs are passed on to the patients. For these reasons, it is ideal that an alternative early warning method of predicting LDL levels is developed, rather than relying on drawing blood. That being said, there are many factors that play a role in predicting LDL cholesterol levels that the project likely did not utilize. In the future, utilizing more predictors could produce a model that will reduce patients’ anxiety and expense while visiting the doctor.

**References**

CDC. (2019, February 6). High Cholesterol Facts. Centers for Disease Control and Prevention. <https://www.cdc.gov/cholesterol/facts.htm>

Kaplan, J. (2020).  *fastDummies: Fast Creation of Dummy (Binary) Columns and Rows from Categorical Variables.* <https://CRAN.R-project.org/package=fastDummies>.

Kassambara. (2018). *CART model: Decision tree essentials*. STHDA. Retrieved June 24, 2022, from <http://www.sthda.com/english/articles/35-statistical-machine-learning-essentials/141-cart-model-> decision-tree-essentials/

Kuhn M. (2021). *caret: Classification and Regression Training.* https://CRAN.Rproject.org/package=caret

Kuhn, M., & Johnson, K. (2013). *Applied Predictive Modeling.* New York City: Springer. ISBN-13: 978- 1461468486

Mayo Foundation for Medical Education and Research. (2021, July 20). *High cholesterol*. Mayo Clinic. Retrieved June 20, 2022, from [https://www.mayoclinic.org/diseases-conditions/high-blood- cholesterol/symptoms-causes/syc- 20350800#:~:text=With%20high%20cholesterol%2C%20you%20can,a%20heart%20attack%20o r%20str oke](https://www.mayoclinic.org/diseases-conditions/high-blood-%09cholesterol/symptoms-causes/syc-%0920350800#:~:text=With%20high%20cholesterol%2C%20you%20can,a%20heart%20attack%20o r%20str oke).

MedlinePlus. (2018). LDL: The “Bad” Cholesterol. Medlineplus.gov; National Library of Medicine. <https://medlineplus.gov/ldlthebadcholesterol.html>

Shiny - Tutorial. (n.d.). Shiny.rstudio.com. Retrieved June 23, 2022, from <https://shiny.rstudio.com/tutorial/#written-tutorials>

U.S. National Library of Medicine. (n.d.). *LDL: The "bad" cholesterol*. MedlinePlus. Retrieved June 23, 2022, from https://medlineplus.gov/ldlthebadcholesterol.html

World Health Organization. (2012). *Indicator metadata registry details*. World Health Organization. Retrieved June 23, 2022, from [https://www.who.int/data/gho/indicator-metadata-registry/imr- details/3236](https://www.who.int/data/gho/indicator-metadata-registry/imr-%09details/3236)

Wickham, H. (2016). *ggplot2 : Elegant Graphics for Data Analysis*. <https://CRAN.Rproject.org/package=ggplot2>

**Appendix**

Code utilized for the project

ADS 503 Project

Hunter Blum, Brianne Bell, Eva Chow

2022-06-21

### Libraries

library(haven)  
library(caret)  
library(gridExtra)  
library(corrplot)  
library(e1071)  
library(car)  
library(lattice)  
library(doParallel)  
library(RANN)  
library(rpart)  
library(party)  
library(partykit)  
library(rpart.plot)  
library(randomForest)  
library(RWeka)  
library(gbm)  
library(Cubist)  
  
library(tidyverse)

### Set up Parallelization

cl <- makeCluster(6)  
registerDoParallel(cl)

# Creating Dataset

### Read in data

Demographic <- read\_xpt("P\_DEMO.XPT")  
BodySize <- read\_xpt("P\_BMX.XPT")  
Chol\_ldl <- read\_xpt("P\_TRIGLY.XPT")

### Chol-ldl

#Select Variables of interest  
Chol\_ldl <- Chol\_ldl %>% select(SEQN, LBDLDL)  
#NA in target feature won't be useful  
Chol\_ldl <- Chol\_ldl %>% drop\_na()

### Demographic

#Get rid of variables we don't need  
Drop\_col <- c('SDDSRVYR', 'RIDSTATR', 'RIDEXMON', 'SIAPROXY', 'SIAINTRP', 'FIAPROXY', 'FIAINTRP', 'MIAPROXY', 'MIAINTRP', 'WTINTPRP', 'WTMECPRP', 'SDMVPSU', 'SDMVSTRA')  
Demographic <- Demographic %>% select(-one\_of(Drop\_col))

### BodySize

Drop\_col <- c('BMIWT', 'BMIRECUM', 'BMIHEAD', 'BMIHT', 'BMILEG', 'BMIARML', 'BMIARMC', 'BMIWAIST', 'BMIHIP', 'BMDSTATS')  
BodySize <- BodySize %>% select(-one\_of(Drop\_col))

### Join

J1 <- Chol\_ldl %>% left\_join(Demographic, by = "SEQN")  
Chol <- J1 %>% left\_join(BodySize, by = "SEQN")  
Chol <- Chol %>% select(!SEQN)

# Cleaning

## Changing factors for EDA

### Changing Variables to the Correct Type

Chol\_2 <- Chol  
factors <- c("RIAGENDR", "RIDRETH1", "RIDRETH3", "DMDBORN4", "DMDEDUC2", "DMDMARTZ", "RIDEXPRG", "SIALANG", "FIALANG", "MIALANG", "AIALANGA")  
Chol\_2[,factors] <- lapply(Chol\_2[,factors], factor)

### Change factor levels to be more interpretable

levels(Chol\_2$RIAGENDR) <- c("Male", "Female")  
levels(Chol\_2$RIDRETH1) <- c("Mex", "OHis", "White", "Black", "Oth")  
levels(Chol\_2$RIDRETH3) <- c("Mex", "OHis", "White", "Black", "Asian", "Oth")  
levels(Chol\_2$DMDBORN4) <- c("USA", "Oth", "Ref", "DK")  
levels(Chol\_2$DMDYRUSZ) <- c("<5", "5-15", "15-30", ">30", "Ref", "DK")  
levels(Chol\_2$DMDEDUC2) <- c("<9", "9-11", "HS", "AA", "BS+", "Ref", "DK")  
levels(Chol\_2$DMDMARTZ) <- c("Mar", "Sep", "Nev", "Ref", "DK")  
levels(Chol\_2$RIDEXPRG) <- c("Yes", "No", "DK")  
levels(Chol\_2$SIALANG) <- c("English", "Spanish")  
levels(Chol\_2$FIALANG) <- c("English", "Spanish")  
levels(Chol\_2$MIALANG) <- c("English", "Spanish")  
levels(Chol\_2$AIALANGA) <- c("English", "Spanish", "Asian")

# EDA

## NAs

### By Variable - Removed variables with over 3000 observations missing

Variable\_na <- Chol\_2 %>% select(everything()) %>% summarise\_all(funs(sum(is.na(.)))) %>% pivot\_longer(cols = c(colnames(Chol\_2[,1:ncol(Chol\_2)])), names\_to = "Variable", values\_to = "Missing") %>% arrange(desc(Missing))  
Drop\_col <- c("RIDAGEMN", "BMXRECUM", "BMXHEAD", "BMDBMIC", "RIDEXPRG", "DMDYRUSZ")  
Chol\_2 <- Chol\_2 %>% select(-one\_of(Drop\_col))

### By Row

row\_na <- rowSums(is.na(Chol\_2))  
row\_na <- data.frame(row\_na, Row = c(1:length(row\_na)))  
row\_na <- row\_na %>% arrange(desc(row\_na))  
#Most missing values in a row is 12, not bad

## Distributions

### Response - LDL Cholesterol

#Looks like a fairly normal distribution, maybe a little skewed to the right.   
ggplot(Chol\_2, aes(x = LBDLDL)) + geom\_histogram() + ggtitle("Distribution of LDL Cholesterol") + xlab("LDL Chol.") + ylab("Count")

Chart, histogram

Description automatically generated

skewness(Chol\_2$LBDLDL)

## [1] 0.7886403

#skewness value .7886403 confirms very mild skewness to the right

### Predictors

Factors

Chol\_fact <- Chol\_2 %>% select\_if(is.factor)  
Chol.bar <- function(xvar){  
 ggplot(Chol\_fact, aes\_(x = as.name(xvar))) +  
 geom\_bar(color = "black") + coord\_flip()  
}  
Lang\_barplots <- lapply(names(Chol\_fact[,7:10]), Chol.bar)  
Oth\_barplots <- lapply(names(Chol\_fact[,1:6]), Chol.bar)  
grid.arrange(grobs = Lang\_barplots, top = "Language Features")

A picture containing text, crossword puzzle

Description automatically generated

grid.arrange(grobs = Oth\_barplots, top = "Other Demographics")

Graphical user interface, timeline

Description automatically generated

Numeric

Chol\_num <- Chol\_2 %>% select\_if(is.numeric) %>% select(!LBDLDL)  
Chol.hist <- function(xvar){  
 ggplot(Chol\_num, aes\_(x = as.name(xvar))) +  
 geom\_histogram(color = "black")   
}  
Dem\_hist <- lapply(names(Chol\_num[,1:2]), Chol.hist)  
Body\_hist <- lapply(names(Chol\_num[,3:10]), Chol.hist)  
grid.arrange(grobs = Dem\_hist, top = "Demographic Features")

Chart, histogram

Description automatically generated

grid.arrange(grobs = Body\_hist, top = "Body Measures")

A picture containing logo

Description automatically generated

## Correlations

Heatmap

Chol\_dummy <- fastDummies::dummy\_cols(Chol\_2)  
Chol\_dummy <- Chol\_dummy %>% select\_if(~!is.factor(.))  
Chol\_dummy[] <- lapply(Chol\_dummy, as.numeric)  
Chol\_cor <- cor(Chol\_dummy, use = "complete.obs")  
Chol\_corplot <- corrplot(cor(Chol\_dummy, use = "complete.obs"), tl.pos = 'n')

Chart

Description automatically generated

#Looks like some dummy variables that are refusal could be messing up correlations  
Drop\_col <- c("DMDBORN4\_Ref", "DMDBORN4\_DK", "DMDEDUC2\_Ref", "DMDEDUC2\_DK", "DMDEDUC2\_NA", "DMDMARTZ\_Ref", "DMDMARTZ\_NA", "FIALANG\_NA", "MIALANG\_NA", "AIALANGA\_NA")  
Chol\_dummy\_2 <- Chol\_dummy %>% select(-one\_of(Drop\_col))  
invisible(cor(Chol\_dummy\_2, use = "complete.obs")) # using invisible() to reduce extensive output  
corrplot(cor(Chol\_dummy\_2, use = "complete.obs"), tl.pos = 'n', type = 'lower')

Chart, scatter chart

Description automatically generated ### Smaller plots for easier interpretation.

# Mini Correlations: Sociological Measures:  
socio <- Chol\_dummy\_2[,c("RIAGENDR\_Male","RIAGENDR\_Female",  
 "RIDRETH1\_Mex", "RIDRETH1\_OHis", "RIDRETH1\_White",   
 "RIDRETH1\_Black", "RIDRETH1\_Oth", "RIDRETH3\_Mex", "RIDRETH3\_OHis",  
 "RIDRETH3\_White", "RIDRETH3\_Black", "RIDRETH3\_Asian", "RIDRETH3\_Oth",  
 "DMDBORN4\_USA", "DMDBORN4\_Oth", "DMDEDUC2\_<9", "DMDEDUC2\_9-11",  
 "DMDEDUC2\_HS", "DMDEDUC2\_AA", "DMDEDUC2\_BS+","DMDMARTZ\_Mar",  
 "DMDMARTZ\_Sep", "DMDMARTZ\_Nev", "DMDMARTZ\_DK", "SIALANG\_English",   
 "SIALANG\_Spanish", "FIALANG\_English", "FIALANG\_Spanish",  
 "MIALANG\_English","MIALANG\_Spanish", "AIALANGA\_English",  
 "AIALANGA\_Spanish", "AIALANGA\_Asian", "LBDLDL")]  
  
invisible(cor(socio, use = "complete.obs")) # using invisible() to reduce extensive output  
corrplot(cor(socio, use = "complete.obs"), tl.pos = 'y', type = 'lower',   
 order = "hclust", tl.cex = 0.5)

Chart, scatter chart

Description automatically generated

# Mini Correlations: Biological Measures:  
biologic <- Chol\_dummy\_2[,c("RIDAGEYR", "BMXWT", "BMXHT", "BMXBMI",   
 "BMXLEG", "BMXARML", "BMXARMC", "BMXWAIST",  
 "BMXHIP", "RIAGENDR\_Male", "RIAGENDR\_Female","LBDLDL")]  
invisible(cor(biologic, use = "complete.obs")) # using invisible() to reduce extensive output  
corrplot(cor(biologic, use = "complete.obs"), tl.pos = 'y', type = 'lower',   
 order = "hclust", tl.cex = 0.8)

Chart, bubble chart

Description automatically generated

# correlations between certain biological measures make sense. BMI is derived from the MASS and height of an individual, so it makes sense that many of the BMI measurements correlate with each other. (i.e. hip, waist, and weight measurements correlate with a higher BMI. Being female correlates negatively with leg and arm length as well as height)

### Highly-correlated variables in cholesterol

Note: Highly correlated variables were removed in the model training process - this was just for EDA

# let's check for highly correlated predictors  
# we'll do this on our non factor transformed dataset  
dim(Chol)

## [1] 4617 27

# 27 variables. let's find correlations greater than 0.80 and see how the data looks if removed  
corr\_Chol <- cor(Chol)  
  
# if removed, how many variables are left  
high\_corr\_Chol <- findCorrelation(corr\_Chol, cutoff = 0.80)  
no\_corr\_Chol <- Chol[, -high\_corr\_Chol]  
dim(no\_corr\_Chol)

## [1] 4617 26

### Looking at a simple ols model to get an idea of important predictors.

# looking at a base linear model, to see significant variables   
model0 <- lm(LBDLDL~., Chol\_2)  
invisible(summary(model0)) # invisible() used to reduce output. Key observations will be noted below:  
 # significant contributors: variable (Pr(>|t|))   
 # RIDAGEYR (0.000132)  
 # (Intercept) (0.043474)  
 # MDBORN4Oth (0.076414)   
 # AIALANGASpanish (0.041803)  
 # BMXLEG (0.033051)  
 # BMXARMC (0.004507)   
 # BMXWAIST (0.071888)

Looking at VIF of simple model showed aliased coefficients. Removed them in the model training process.

# looking at VIF for baseline linear:  
# vif(model0)  
# highly/perfectly correlated factors, we might need to drop some

### Degenerate Predictors in the non-dummy dataset

# Let's check for degenerate predictors from the original dataset  
nearZeroVar(Chol, saveMetrics = FALSE)

## [1] 4 11 16 18 19

deg\_chol <- subset(Chol, select=c(4,11,16,18,19))  
colnames(deg\_chol)

## [1] "RIDAGEMN" "RIDEXPRG" "INDFMPIR" "BMXRECUM" "BMXHEAD"

# Do it again on factor dataset  
nearZeroVar(Chol\_2, saveMetrics = FALSE)

## [1] 13

deg\_chol2 <- subset(Chol\_2, select=c(13))  
colnames(deg\_chol2)

## [1] "INDFMPIR"

# we may have to consider removing depending on the data used for modeling

# Preparing data for modeling

### Splitting dummy-variable data set and resampling

# set the seed and split the data. We'll do an 80/20 split  
set.seed(123)  
Chol\_split <- createDataPartition(Chol$LBDLDL, p=0.80, list=FALSE)  
  
# split into train and test  
Chol\_train <- Chol\_dummy[Chol\_split,]  
Chol\_test <- Chol\_dummy[-Chol\_split,]  
  
# split predictors from the target  
Chol\_train\_X <- as.data.frame(subset(Chol\_train, select=-c(LBDLDL)))   
Chol\_train\_y <- Chol\_train$LBDLDL  
  
Chol\_test\_X <- as.data.frame(subset(Chol\_test, select=-c(LBDLDL)))  
Chol\_test\_y <- Chol\_test$LBDLDL  
  
# Creating imputed data sets  
Chol\_imp <- preProcess(Chol\_train\_X, method = c("center", "scale", "knnImpute"))  
Chol\_train\_X\_imp <- predict(Chol\_imp, Chol\_train\_X)  
Chol\_test\_X\_imp <- predict(Chol\_imp, Chol\_test\_X)  
  
# Adding Resampling/Validation Set and Control   
set.seed(123)  
Chol\_folds <- createFolds(y = Chol\_train\_X, k = 10, returnTrain = T)  
Chol\_control <- trainControl(method = "cv", index = Chol\_folds)

### Numeric training data set with just the numeric variables and gender (Just going off a hunch that having too many dummy variables is hurting linear model performance).

Drop\_col <- c('RIDRETH1', 'RIDRETH3', 'DMDBORN4', 'DMDEDUC2', 'DMDMARTZ', 'SIALANG', 'FIALANG', 'MIALANG', 'AIALANGA', 'LBDLDL')  
Chol\_num <- Chol\_2 %>% select(-one\_of(Drop\_col))  
Chol\_dummy <- fastDummies::dummy\_cols(Chol\_num)  
Chol\_num <- Chol\_dummy %>% select\_if(~!is.factor(.))  
Chol\_num[] <- lapply(Chol\_num, as.numeric)  
  
Chol\_num\_tr\_X <- as.data.frame(Chol\_num[Chol\_split, ])  
Chol\_num\_test\_X <- as.data.frame(Chol\_num[-Chol\_split, ])  
  
#Preprocess  
Chol\_imp <- preProcess(Chol\_num\_tr\_X, method = c("center", "scale", "knnImpute"))  
Chol\_num\_tr\_X <- predict(Chol\_imp, Chol\_num\_tr\_X)  
Chol\_num\_test\_X <- predict(Chol\_imp, Chol\_num\_test\_X)  
  
# Adding Resampling/Validation Set and Control   
set.seed(123)  
Chol\_folds\_num <- createFolds(y = Chol\_num\_tr\_X, k = 10, returnTrain = T)  
Chol\_control\_num <- trainControl(method = "cv", index = Chol\_folds\_num)

# Linear Models - Hunter

## OLS

### Create Initial Model

Chol\_ols\_tune <- train(x = Chol\_train\_X\_imp, y = Chol\_train\_y, method = "lm", trControl = Chol\_control)  
plot(Chol\_ols\_tune$finalModel)

Chart, scatter chart

Description automatically generatedChart

Description automatically generatedChart, scatter chart

Description automatically generatedChart, line chart

Description automatically generated ### FIX TRAINING SET

# VIF shows aliased coefficients, need to get rid of those by removing high cor predictors  
test <- cor(Chol\_train\_X\_imp)  
# Also have an issue with DMDEDUC2\_DK all being zero so get rid of high var predictors  
Chol\_tr\_x\_imp\_vr <- Chol\_train\_X\_imp[, -nearZeroVar(Chol\_train\_X\_imp)]  
Chol\_tr\_X\_imp\_fin <- Chol\_tr\_x\_imp\_vr[, -findCorrelation(cor(Chol\_tr\_x\_imp\_vr), cutoff = 0.9)]

### Tune Another Model

Chol\_ols\_tune2 <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y, method = "lm", trControl = Chol\_control)  
plot(Chol\_ols\_tune2$finalModel)

Chart, scatter chart

Description automatically generatedChart

Description automatically generatedChart, scatter chart

Description automatically generatedChart

Description automatically generated

invisible(summary(Chol\_ols\_tune2$finalModel)) # invisible used to minimize output clutter. Key observations to be noted in final report

### Tune model with BoxCox to see if it will help normality issues - Didn’t do much, we’ll just stick with the non-transformed data. Also tried transformin LDL, didn’t work either.

Chol\_bct <- preProcess(Chol\_tr\_X\_imp\_fin, method = "BoxCox")  
Chol\_tr\_boxcox <- predict(Chol\_bct, Chol\_tr\_X\_imp\_fin)  
  
Chol\_ols\_tune3 <- train(x = Chol\_tr\_boxcox, y = Chol\_train\_y, method = "lm", trControl = Chol\_control)  
plot(Chol\_ols\_tune3$finalModel)

Chart, scatter chart

Description automatically generatedChart

Description automatically generatedChart, scatter chart

Description automatically generatedChart

Description automatically generated

invisible(summary(Chol\_ols\_tune3$finalModel)) #invisible used to reduce cluttered output. Key observations noted in final report

### Try reduced data - Didn’t really help our diagnostic plot, so we’ll go with the regular dummy data

Chol\_ols\_tune\_num <- train(x = Chol\_num\_tr\_X, y = Chol\_train\_y, method = "lm", trControl = Chol\_control\_num)  
plot(Chol\_ols\_tune\_num$finalModel)

Chart, scatter chart

Description automatically generatedChart, line chart

Description automatically generatedChart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generated

invisible(summary(Chol\_ols\_tune\_num$finalModel)) # hiding long output to reduce clutter. Key observations to be noted in final report

### Final OLS Model

Chol\_sig\_tr <- Chol\_tr\_X\_imp\_fin %>% select(RIDAGEYR, BMXHT, BMXBMI, BMXLEG, BMXARMC, DMDBORN4\_Oth, DMDEDUC2\_NA, MIALANG\_NA, AIALANGA\_NA)  
Chol\_ols <- train(x = Chol\_sig\_tr, y = Chol\_train\_y, method = "lm", trControl = Chol\_control)  
plot(Chol\_ols$finalModel)

Chart, scatter chart

Description automatically generatedChart

Description automatically generatedChart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generated

invisible(summary(Chol\_ols$finalModel)) # hidden to reduce output clutter. Key observations to be noted  
  
#Predict on test data  
Chol\_ols\_res <- predict(Chol\_ols, Chol\_test\_X)

## PCR and PLS

### PCR

set.seed(123)  
Chol\_pcr <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y, method = "pcr", tuneGrid = expand.grid(ncomp=1:32), trControl = Chol\_control)  
invisible(Chol\_pcr) # output hidden in final report to reduce clutter  
  
set.seed(123)  
Chol\_pcr\_box <- train(x = Chol\_tr\_boxcox, y = Chol\_train\_y, method = "pcr", tuneGrid = expand.grid(ncomp=1:32), trControl = Chol\_control)  
invisible(Chol\_pcr\_box) # output hidden in final report to reduce clutter  
  
set.seed(123)  
Chol\_pcr\_num <- train(x = Chol\_num\_tr\_X, y = Chol\_train\_y, method = "pcr", tuneGrid = expand.grid(ncomp=1:8), trControl = Chol\_control\_num)  
invisible(Chol\_pcr\_num) # output hidden in final report to reduce clutter  
  
pcr\_resamp <- Chol\_pcr$results  
pcr\_resamp$Model <- "PCR"  
  
box\_pcr\_resamp <- Chol\_pcr\_box$results  
box\_pcr\_resamp$Model <- "BPCR"  
  
num\_pcr\_resamp <- Chol\_pcr\_num$results  
num\_pcr\_resamp$Model <- "PCR"  
# key observations and summary to be noted in final report

### PLS

set.seed(123)  
Chol\_pls <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y, method = "pls", tuneGrid = expand.grid(ncomp = 1:32), trControl = Chol\_control)  
invisible(Chol\_pls) # output hidden to reduce clutter  
  
set.seed(123)  
Chol\_pls\_box <- train(x = Chol\_tr\_boxcox, y = Chol\_train\_y, method = "pls", tuneGrid = expand.grid(ncomp = 1:32), trControl = Chol\_control)  
invisible(Chol\_pls\_box) # output hidden to reduce clutter  
  
set.seed(123)  
Chol\_pls\_num <- train(x = Chol\_num\_tr\_X, y = Chol\_train\_y, method = "pls", tuneGrid = expand.grid(ncomp = 1:8), trControl = Chol\_control\_num)  
invisible(Chol\_pls\_num) # output hidden to reduce clutter  
  
pls\_resamp <- Chol\_pls$results  
pls\_resamp$Model <- "PLS"  
  
pls\_box\_resamp <- Chol\_pls\_box$results  
pls\_box\_resamp$Model <- "BPLS"  
  
pls\_num\_resamp <- Chol\_pls\_num$results  
pls\_num\_resamp$Model <- "PLS"

### Compare

plot\_data <- rbind(pcr\_resamp, box\_pcr\_resamp, pls\_resamp, pls\_box\_resamp)  
xyplot(RMSE ~ ncomp, data = plot\_data, xlab = "# of Components", ylab = "RMSE (Cross-validation)", auto.key = list(columns = 4), groups = Model, type = c("o", "g"))

Chart, line chart

Description automatically generated

plot2\_data <- rbind(num\_pcr\_resamp, pls\_num\_resamp)  
xyplot(RMSE ~ ncomp, data = plot2\_data, xlab = "# of Components", ylab = "RMSE (Cross-validation)", auto.key = list(columns = 2), groups = Model, type = c("o", "g"))

Chart, line chart

Description automatically generated

## Penalized Models

### Ridge

set.seed(123)  
Chol\_ridge <- train(x = Chol\_tr\_X\_imp\_fin, y= Chol\_train\_y, method = "ridge", tuneGrid = expand.grid(lambda = seq(0, .5, length = 15)), trControl = Chol\_control)  
invisible(Chol\_ridge) # model output hidden to reduce clutter  
  
set.seed(123)  
Chol\_ridge\_box <- train(x = Chol\_tr\_boxcox, y= Chol\_train\_y, method = "ridge", tuneGrid = expand.grid(lambda = seq(0, .5, length = 15)), trControl = Chol\_control)  
invisible(Chol\_ridge\_box) # model output hidden to reduce clutter  
  
set.seed(123)  
Chol\_ridge\_num <- train(x = Chol\_num\_tr\_X, y= Chol\_train\_y, method = "ridge", tuneGrid = expand.grid(lambda = seq(0, .5, length = 15)), trControl = Chol\_control\_num)  
invisible(Chol\_ridge\_num) # model output hidden to reduce clutter  
  
print(update(plot(Chol\_ridge), xlab = "Penalty"))

Chart, line chart

Description automatically generated

print(update(plot(Chol\_ridge\_box), xlab = "Penalty"))

Chart, line chart

Description automatically generated

print(update(plot(Chol\_ridge\_num), xlab = "Penalty"))

Chart, line chart

Description automatically generated

### Elastic Net

enet\_grid <- expand.grid(lambda = c(0, 0.01, 0.1), fraction = seq(0.05, 1, length = 20))  
  
Chol\_enet <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y, method = "enet", tuneGrid = enet\_grid, trControl = Chol\_control)  
invisible(Chol\_enet) # model output hidden to reduce clutter  
  
Chol\_enet\_box <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y, method = "enet", tuneGrid = enet\_grid, trControl = Chol\_control)  
invisible(Chol\_enet\_box) # model output hidden to reduce clutter  
  
Chol\_enet\_num <- train(x = Chol\_num\_tr\_X, y = Chol\_train\_y, method = "enet", tuneGrid = enet\_grid, trControl = Chol\_control\_num)  
  
plot(Chol\_enet)

Chart

Description automatically generated

plot(Chol\_enet\_box)

Chart

Description automatically generated

plot(Chol\_enet\_num)

Chart

Description automatically generated

## Gather Results from Linear Models

Note: No diagnostic plot showed much of a difference between boxcox vs. regular (not suprising given normality in histograms), so we’ll just use the regular constructed models

### Create Data set of predictions and observed

Res\_OLS <- predict(Chol\_ols, Chol\_test\_X\_imp)  
Res\_OLS\_num <- predict(Chol\_ols\_tune\_num, Chol\_num\_test\_X)  
Res\_PLS <- predict(Chol\_pls, Chol\_test\_X\_imp)  
Res\_PLS\_num <- predict(Chol\_pls\_num, Chol\_num\_test\_X)  
Res\_PCR <- predict(Chol\_pcr, Chol\_test\_X\_imp)  
Res\_PCR\_num <- predict(Chol\_pcr\_num, Chol\_num\_test\_X)  
Res\_Ridge <- predict(Chol\_ridge, Chol\_test\_X\_imp)  
Res\_Ridge\_num <- predict(Chol\_ridge\_num, Chol\_num\_test\_X)  
Res\_Enet <- predict(Chol\_enet, Chol\_test\_X\_imp)  
Res\_Enet\_num <- predict(Chol\_enet\_num, Chol\_num\_test\_X)  
  
Linear\_res <- cbind.data.frame(Observed = Chol\_test\_y, OLS = Res\_OLS, OLS\_num = Res\_OLS\_num, PLS = Res\_PLS, PLS\_num = Res\_PLS\_num, PCR = Res\_PCR, PCR\_num = Res\_PCR\_num ,Ridge = Res\_Ridge, Ridge\_num = Res\_Ridge\_num, ENet = Res\_Enet, ENet\_num = Res\_Enet\_num)

### Get RMSE and Plot

find\_rmse <- function(x){  
 caret::RMSE(x, Linear\_res[,"Observed"])  
}  
  
RMSE\_results <- apply(X = Linear\_res[,2:11], FUN = find\_rmse, MARGIN = 2)  
RMSE\_results <- data.frame(RMSE\_results)  
RMSE\_results$Model <- rownames(RMSE\_results)  
  
  
ggplot(RMSE\_results, aes(x=reorder(Model, -RMSE\_results), y=RMSE\_results)) + geom\_segment(aes(x=reorder(Model, -RMSE\_results), xend = reorder(Model, -RMSE\_results), y=30, yend=RMSE\_results), color = "cadetblue") + geom\_point(color = "darkblue", size = 10) + coord\_flip() + ylab("RMSE") + geom\_text(aes(label = round(RMSE\_results, 2)), color = "white", size = 2.5)

Chart, bubble chart

Description automatically generated

# Non-Linear Models - Brianne

## Support Vector Machine (SVM)

### Create Initial Radial Model

# initial SVM model with radial basis and processed Chol\_tr\_X\_imp\_fin and Chol\_train\_y  
set.seed(123)  
svmR0 <- train(x=Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y,  
 method = "svmRadial",  
 preProcess = c("center", "scale"),  
 tuneLength = 14,  
 trControl = Chol\_control)  
invisible(svmR0) # model output hidden to reduce clutter. Key observations noted below:  
 # final model uses: sigma = 0.0207376 and C = 0.25  
 # RMSE: 35.40559, Rsquared: 0.019455264  
plot(svmR0, scales = list(x = list(log = 2)), main="SVM-Radial Initial")

Chart, line chart

Description automatically generated

### final radial model

# svm radial model v2   
 # issue causing variables in X: MIALANG\_Spanish, DMDEDUC2\_<9, MIALANG\_NA (zero var)  
Chol\_tr\_X\_impfin\_drop <- c("MIALANG\_Spanish", "DMDEDUC2\_<9", "MIALANG\_NA")  
Chol\_tr\_X\_impfin\_sv <- subset(Chol\_tr\_X\_imp\_fin,   
 select = !(names(Chol\_tr\_X\_imp\_fin) %in% Chol\_tr\_X\_impfin\_drop))  
  
#making test X have same columns available  
Chol\_te\_X\_sv <- subset(Chol\_test\_X\_imp, select = c(names(Chol\_tr\_X\_impfin\_sv)))  
dim(Chol\_tr\_X\_impfin\_sv)

## [1] 3696 29

dim(Chol\_te\_X\_sv)

## [1] 921 29

# sigma grid instead of using tuneLength = 14  
sigmaEst <- kernlab::sigest(as.matrix(Chol\_tr\_X\_impfin\_sv[,1:29]))  
Csearch <- 2^seq(-4,+4)  
# sigma estimates using kernlab's sigest function  
svmgrid <- expand.grid(sigma = sigmaEst, C = Csearch)  
  
#model  
set.seed(123)  
svmR1 <- train(x=Chol\_tr\_X\_impfin\_sv, y = Chol\_train\_y,  
 method = "svmRadial",  
 preProcess = c("center", "scale"),  
 tuneGrid = svmgrid,  
 trControl = Chol\_control)  
invisible(svmR1) # model output hidden to reduce clutter. Key observations noted below:  
 # final model uses: sigma = 0.03504524 and C = 0.25.  
 # RMSE: 35.30443, Rsquared: 0.020245225   
plot(svmR1, scales = list(x = list(log = 2)), main="SVM-Radial v2")

Chart, line chart

Description automatically generated

svmR1$finalModel

## Support Vector Machine object of class "ksvm"   
##   
## SV type: eps-svr (regression)   
## parameter : epsilon = 0.1 cost C = 0.25   
##   
## Gaussian Radial Basis kernel function.   
## Hyperparameter : sigma = 0.035045237945048   
##   
## Number of Support Vectors : 3377   
##   
## Objective Function Value : -557.6121   
## Training error : 0.82428

### Create Polynomial Model

# going to use the x training set from final svm-radial due to assuming there will be the same problem causing factors of nearZeroVar.  
set.seed(123)  
svmP <- train(x=Chol\_tr\_X\_impfin\_sv, y = Chol\_train\_y,  
 method = "svmPoly",  
 preProcess = c("center", "scale"),  
 tuneGrid = expand.grid(degree = 1:3,   
 scale = c(0.01, 0.005, 0.001, 0.0005),   
 C = Csearch),  
 trControl = Chol\_control)  
invisible(svmP) # model output hidden to reduce clutter. Key observations noted below:  
 # final model uses: degree = 2, scale = 0.001, offset = 1   
 # sigma = 0.02231109 and C = 2.  
 # RMSE: 35.44929, Rsquared: 0.011746076  
plot(svmP, scales = list(x = list(log = 2),  
 between=list(x=.5, y=1)), main="SVM-Polynomial")

Chart

Description automatically generated

svmP$finalModel

## Support Vector Machine object of class "ksvm"   
##   
## SV type: eps-svr (regression)   
## parameter : epsilon = 0.1 cost C = 2   
##   
## Polynomial kernel function.   
## Hyperparameters : degree = 2 scale = 0.001 offset = 1   
##   
## Number of Support Vectors : 3366   
##   
## Objective Function Value : -4741.157   
## Training error : 0.932111

## K Nearest Neighbors (KNN)

### Create Initial KNN Model

# KNN Model needs to have NZV removed so again we are using the x=Chol\_tr\_X\_impfin\_sv to train and Chol\_te\_X\_sv to test  
set.seed(123)  
knnTune <- train(x=Chol\_tr\_X\_impfin\_sv, y = Chol\_train\_y,  
 method = "knn",  
 preProcess = c("center", "scale"),  
 tuneGrid = data.frame(k=1:40),  
 trControl = Chol\_control)  
invisible(knnTune) # model output hidden to reduce clutter. Key observations noted below:  
 # final model uses: k=40  
 # RMSE: 35.63394, Rsquared: 0.0006650456  
plot(knnTune, main="KNN")

Chart

Description automatically generated

knnTune$finalModel

## 20-nearest neighbor regression model

## Multivariate Adaptive Regression Splines (MARS)

### Create Initial MARS Model

# MARS model doesn't need preprocessing, so first rendition will be with Chol\_tr\_X\_imp\_fin   
set.seed(123)  
mars1 <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y,  
 method = "earth",  
 tuneGrid = expand.grid(degree = 1:3, nprune = 2:38),  
 trControl = Chol\_control)

## Loading required package: earth

## Loading required package: Formula

## Loading required package: plotmo

## Loading required package: plotrix

## Loading required package: TeachingDemos

mars1$finalModel

## Selected 2 of 18 terms, and 1 of 32 predictors (nprune=2)  
## Termination condition: RSq changed by less than 0.001 at 18 terms  
## Importance: RIDAGEYR, INDFMPIR-unused, BMXHT-unused, BMXBMI-unused, ...  
## Number of terms at each degree of interaction: 1 1 (additive model)  
## GCV 1192.459 RSS 4400176 GRSq 0.05246562 RSq 0.05349109

invisible(mars1$results) # model results hidden to reduce clutter. Key observations noted below:  
 #used 1 of 32 predictors, 2 of 18 terms (nprune =2) degree=1  
 # RMSE: 35.02993, Rsquared: 0.05354368   
plot(mars1, main="MARS Initial")

Chart

Description automatically generated

## Multivariate Adaptive Regression Splines (MARS)

### Create Secondary MARS Model

# MARS model using same X sets as SVM models:   
set.seed(123)  
mars2 <- train(x = Chol\_tr\_X\_impfin\_sv, y = Chol\_train\_y,  
 method = "earth",  
 tuneGrid = expand.grid(degree = 1:3, nprune = 2:38),  
 trControl = Chol\_control)  
mars2$finalModel

## Selected 2 of 18 terms, and 1 of 29 predictors (nprune=2)  
## Termination condition: RSq changed by less than 0.001 at 18 terms  
## Importance: RIDAGEYR, INDFMPIR-unused, BMXHT-unused, BMXBMI-unused, ...  
## Number of terms at each degree of interaction: 1 1 (additive model)  
## GCV 1192.459 RSS 4400176 GRSq 0.05246562 RSq 0.05349109

plot(mars2, main="MARS Secondary")

Chart

Description automatically generated

# No change between the two MARS models. Drops all but two factors for both.  
 #nprune=2, degree=1, RMSE: 34.80857, Rsquared: 0.05457991

## Neural Network Model (nnet)

### Create Initial NNET Model

set.seed(123)  
nnetGrid <- expand.grid(decay = c(0, 0.01, .1), size = c(3, 7, 11, 13))  
# NNET first rendition will be with Chol\_tr\_X\_imp\_fin   
set.seed(100)  
nnet1 <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y,  
 method = "nnet",  
 tuneGrid = nnetGrid,  
 trControl = Chol\_control,  
 preProc = c("center", "scale"),  
 linout = TRUE,  
 trace = FALSE,  
 MaxNWts = 13 \* (ncol(Chol\_tr\_X\_imp\_fin) + 1) + 13 + 1,  
 maxit = 100)  
invisible(nnet1) # model output hidden to reduce clutter. Key observations noted below:  
 # size=3, decay=0.1  
 # RMSE: 40.97813, Rsquared: 0.0051328366  
plot(nnet1, main="NNET first")

Chart, line chart

Description automatically generated

set.seed(123)  
# NNET second rendition will be with Chol\_tr\_X\_impfin\_sv   
set.seed(100)  
nnet2 <- train(x = Chol\_tr\_X\_impfin\_sv, y = Chol\_train\_y,  
 method = "nnet",  
 tuneGrid = nnetGrid,  
 trControl = Chol\_control,  
 preProc = c("center", "scale"),  
 linout = TRUE,  
 trace = FALSE,  
 MaxNWts = 13 \* (ncol(Chol\_tr\_X\_impfin\_sv) + 1) + 13 + 1,  
 maxit = 100)  
invisible(nnet2) # model output hidden to reduce clutter. Key observations noted below:  
 # size=13, decay=0.1  
 # RMSE: 44.38061, Rsquared: 0.001743654  
plot(nnet2, main="NNET second")

Chart, line chart

Description automatically generated ## Comparing Nonlinear Models: ### Saving Results

NonLpred <- data.frame(obs=Chol\_test\_y)  
NonLpred$svmR <- predict(svmR1, Chol\_te\_X\_sv)  
NonLpred$svmP <- predict(svmP, Chol\_te\_X\_sv)  
NonLpred$KNN <- predict(knnTune, Chol\_te\_X\_sv)  
NonLpred$MARS2 <- predict(mars2, Chol\_te\_X\_sv)  
NonLpred$NNET2 <- predict(nnet2, Chol\_te\_X\_sv)  
plotpred <- data.frame(x=1:921, y1=NonLpred$obs, y2=NonLpred$svmR,   
 y3=NonLpred$svmP, y4=NonLpred$KNN,   
 y5=NonLpred$MARS2[,"y"], y6=NonLpred$NNET2)  
plot(plotpred$x, plotpred$y1, type = "l", col = 1,   
 xlab = "prediction #", ylab = "prediction values",   
 main = "Nonlinear Model Predictions")  
lines(plotpred$x, plotpred$y2, col = 2)  
lines(plotpred$x, plotpred$y3, col = 3)  
lines(plotpred$x, plotpred$y4, col = 4)  
lines(plotpred$x, plotpred$y5, col = 5)  
lines(plotpred$x, plotpred$y6, col = 6)  
legend("bottomleft", cex=0.5, legend = c("Observed", "SVM-Radial",  
 "SVM-Polynomial", "KNN", "MARS", "NNET"),  
 col = 1:6, lwd = 2)

Chart

Description automatically generated ### Getting RMSE and Plotting

# RMSE = sqrt(sum((obs-pred)^2)/n), n=921  
getRMSE <- function(x,y) {  
 sqrt(sum((x-y)^2)/length(x))  
}  
nonlin\_rmse <- data.frame(c("svmRad", "svmPoly", "KNN",  
 "MARS", "NNet"))  
nonlin\_rmse$RMSE <- c(getRMSE(NonLpred$obs, NonLpred$svmR),  
 getRMSE(NonLpred$obs, NonLpred$svmP),  
 getRMSE(NonLpred$obs, NonLpred$KNN),  
 getRMSE(NonLpred$obs, NonLpred$MARS2),  
 getRMSE(NonLpred$obs, NonLpred$NNET2))  
colnames(nonlin\_rmse)[1]<-"Model Type"  
nonlin\_rmse[order(nonlin\_rmse$RMSE),]

## Model Type RMSE  
## 1 svmRad 32.94845  
## 2 svmPoly 33.03741  
## 4 MARS 33.52618  
## 3 KNN 33.85140  
## 5 NNet 36.94634

# best non linear model is svm radial with   
 # sigma = sigma = 0.03504524 and C = 0.25. (RMSE is 32.95)

### Messing around with predictor set and log transforming the response

expDropC <- c("RIDRETH1\_Black", "RIDRETH1\_Mex", "RIDRETH1\_OHis",   
 "RIDRETH1\_Oth", "RIDRETH1\_White", "BMXBMI")  
Xtrial\_tr <- subset(Chol\_train\_X\_imp, select = !(names(Chol\_train\_X\_imp) %in% expDropC))  
# looking for near zero var:  
Xtri\_nzv <- nearZeroVar(Xtrial\_tr)  
 # "DMDBORN4\_Ref", "DMDBORN4\_DK", "DMDEDUC2\_Ref", "DMDEDUC2\_DK", "DMDMARTZ\_Ref",  
 # "DMDMARTZ\_DK", "AIALANGA\_Asian"  
Xtri\_tr\_nz <- subset(Xtrial\_tr, select = -c(Xtri\_nzv))  
print(paste("Xtrial\_tr ncol: ", ncol(Xtrial\_tr), " NZV removed, new ncol: ", ncol(Xtri\_tr\_nz)))

## [1] "Xtrial\_tr ncol: 47 NZV removed, new ncol: 40"

# dropping SIALANG groups (sample person interview instrument lang)  
expDropC <- c("SIALANG\_English", "SIALANG\_Spanish",   
 "MIALANG\_English", "MIALANG\_Spanish", "MIALANG\_NA")  
Xtri\_tr\_nz <- subset(Xtri\_tr\_nz, select = !(names(Xtri\_tr\_nz) %in% expDropC))  
print(paste("Xtrial\_tr ncol: ", ncol(Xtrial\_tr), " NZV removed, new ncol: ", ncol(Xtri\_tr\_nz)))

## [1] "Xtrial\_tr ncol: 47 NZV removed, new ncol: 35"

#looking for high corr:  
Xtritr\_hiC <- findCorrelation(cor(Xtri\_tr\_nz), cutoff = 0.8)  
Xtritr\_hiC

## [1] 33 18 3 30 8 9 25 10

# "AIALANGA\_English", "DMDBORN4\_USA", "BMXWT", "FIALANG\_English",   
 # "BMXWAIST", "BMXHIP", "DMDEDUC2\_NA", "RIAGENDR\_Male"  
  
Xtritr\_hiC <- subset(Xtri\_tr\_nz, select = c("AIALANGA\_English", "DMDBORN4\_USA",   
 "BMXWT", "FIALANG\_English", "BMXWAIST",  
 "BMXHIP", "DMDEDUC2\_NA", "RIAGENDR\_Male"))  
invisible(cor(Xtritr\_hiC)) # invisible used to reduce extensive output  
corrplot(cor(Xtritr\_hiC), order = "hclust", type="lower")

Chart, bubble chart

Description automatically generated

#dropping "BMXHIP", "BMXWAIST", DMDEDUC2\_<9 (recurring issues in model attempts)  
drophiC <- c("BMXHIP", "BMXWAIST", "DMDEDUC2\_<9")  
X\_trial\_train <- subset(Xtri\_tr\_nz, select = !(names(Xtri\_tr\_nz) %in% drophiC))  
corrplot(cor(X\_trial\_train), order = "hclust", type="lower", tl.cex = 0.7)

Chart, scatter chart

Description automatically generated

### Making test and train columns match

keepsies <- colnames(X\_trial\_train)  
X\_trial\_test <- subset(Chol\_test\_X\_imp, select = c(keepsies))

### Radial SVM with the matched data and log adjusted y

#svm radial with X\_trial\_train and log adjusted y  
log\_Y\_train <- log10(Chol\_train\_y)  
hist(log\_Y\_train)

Chart, histogram

Description automatically generated

log\_Y\_test <- log10(Chol\_test\_y)  
  
#model  
set.seed(123)  
svmR\_trial <- train(x=X\_trial\_train, y = log\_Y\_train,  
 method = "svmRadial",  
 preProcess = c("center", "scale"),  
 tuneLength = 14,  
 trControl = Chol\_control)  
invisible(svmR\_trial) # output hidden to reduce clutter. Key observations noted below:  
# issues in: DMDEDUC2\_<9,   
 # final model uses: sigma = 0.02019005 and C = 0.25.  
 # RMSE: 0.1531765, Rsquared: 0.021637157  
 # while these are the lowest values, the graph is identical to the non-log adjusted SVM radial model with just different RMSE values.   
plot(svmR\_trial, scales = list(x = list(log = 2)), main="SVM-Radial with X\_trial\_train and Log Y")

Chart, line chart

Description automatically generated

svmR\_trial$finalModel

## Support Vector Machine object of class "ksvm"   
##   
## SV type: eps-svr (regression)   
## parameter : epsilon = 0.1 cost C = 0.25   
##   
## Gaussian Radial Basis kernel function.   
## Hyperparameter : sigma = 0.0201900463201106   
##   
## Number of Support Vectors : 3367   
##   
## Objective Function Value : -574.0137   
## Training error : 0.850126

trialPred <- data.frame(obs=log\_Y\_test)  
trialPred$svmRad <- predict(svmR\_trial, X\_trial\_test)

# Regression Trees - Eva

## Single Regression Trees

### Tree with CART based splits (rpart) and optimization with default parameters

set.seed(123)  
# let's see how it splits the training data  
Chol\_rpart <- rpart(Chol\_train\_y ~., data = Chol\_tr\_X\_imp\_fin, cp = 0.003)  
  
invisible(summary(Chol\_rpart)) # output hidden to reduce clutter. Key observations noted below:

## Call:  
## rpart(formula = Chol\_train\_y ~ ., data = Chol\_tr\_X\_imp\_fin, cp = 0.003)  
## n= 3696   
##   
## CP nsplit rel error xerror xstd  
## 1 0.054164414 0 1.0000000 1.0004973 0.03300310  
## 2 0.020563310 1 0.9458356 0.9554923 0.03188785  
## 3 0.013308632 2 0.9252723 0.9464258 0.03207678  
## 4 0.007112960 3 0.9119636 0.9311717 0.03205759  
## 5 0.005855210 4 0.9048507 0.9260955 0.03176146  
## 6 0.004972343 6 0.8931403 0.9256549 0.03150486  
## 7 0.004824224 7 0.8881679 0.9238887 0.03142018  
## 8 0.004142728 8 0.8833437 0.9264225 0.03141473  
## 9 0.003828087 10 0.8750582 0.9339156 0.03236363  
## 10 0.003774630 11 0.8712302 0.9329465 0.03238441  
## 11 0.003599490 12 0.8674555 0.9316994 0.03236427  
## 12 0.003384505 13 0.8638560 0.9351374 0.03241022  
## 13 0.003021559 14 0.8604715 0.9307203 0.03231526  
## 14 0.003000000 15 0.8574500 0.9348469 0.03254034  
##   
## Variable importance  
## RIDAGEYR BMXBMI DMDEDUC2\_NA DMDMARTZ\_Nev BMXARMC   
## 32 13 12 9 8   
## BMXARML AIALANGA\_NA BMXLEG BMXHT RIAGENDR\_Male   
## 5 5 4 4 4   
## INDFMPIR DMDMARTZ\_Sep DMDEDUC2\_BS+ DMDMARTZ\_Mar   
## 1 1 1 1   
##   
## Node number 1: 3696 observations, complexity param=0.05416441  
## mean=105.2411, MSE=1257.805   
## left son=2 (1098 obs) right son=3 (2598 obs)  
## Primary splits:  
## RIDAGEYR < -0.7177214 to the left, improve=0.05416441, (0 missing)  
## DMDEDUC2\_NA < 0.9305478 to the right, improve=0.04807959, (0 missing)  
## DMDMARTZ\_Nev < -0.2411516 to the right, improve=0.02845850, (0 missing)  
## BMXBMI < -0.7742706 to the left, improve=0.02378829, (0 missing)  
## BMXARMC < -0.8110234 to the left, improve=0.01974683, (0 missing)  
## Surrogate splits:  
## DMDEDUC2\_NA < 0.9305478 to the right, agree=0.862, adj=0.536, (0 split)  
## DMDMARTZ\_Nev < -0.2411516 to the right, agree=0.846, adj=0.482, (0 split)  
## BMXBMI < -0.9678924 to the left, agree=0.735, adj=0.108, (0 split)  
## BMXARMC < -1.151012 to the left, agree=0.730, adj=0.091, (0 split)  
##   
## Node number 2: 1098 observations, complexity param=0.01330863  
## mean=92.54463, MSE=805.3737   
## left son=4 (536 obs) right son=5 (562 obs)  
## Primary splits:  
## BMXBMI < -0.5935568 to the left, improve=0.06996470, (0 missing)  
## BMXARMC < -0.113152 to the left, improve=0.06346379, (0 missing)  
## RIDAGEYR < -1.398075 to the left, improve=0.04298442, (0 missing)  
## DMDEDUC2\_NA < 0.9305478 to the right, improve=0.03832009, (0 missing)  
## DMDEDUC2\_AA < -0.4815614 to the right, improve=0.02453409, (0 missing)  
## Surrogate splits:  
## BMXARMC < -0.4352465 to the left, agree=0.884, adj=0.763, (0 split)  
## BMXARML < -0.3506071 to the left, agree=0.622, adj=0.226, (0 split)  
## RIDAGEYR < -1.203688 to the left, agree=0.605, adj=0.190, (0 split)  
## DMDEDUC2\_NA < 0.9305478 to the right, agree=0.604, adj=0.188, (0 split)  
## DMDEDUC2\_BS+ < -0.3327601 to the right, agree=0.587, adj=0.153, (0 split)  
##   
## Node number 3: 2598 observations, complexity param=0.02056331  
## mean=110.607, MSE=1352.096   
## left son=6 (968 obs) right son=7 (1630 obs)  
## Primary splits:  
## RIDAGEYR < 0.7887756 to the right, improve=0.027213960, (0 missing)  
## AIALANGA\_NA < 0.7694033 to the right, improve=0.019766670, (0 missing)  
## BMXBMI < 0.6714394 to the right, improve=0.007005182, (0 missing)  
## DMDBORN4\_Oth < 0.5241397 to the left, improve=0.006384595, (0 missing)  
## BMXARML < 0.8611078 to the right, improve=0.005282776, (0 missing)  
## Surrogate splits:  
## AIALANGA\_NA < 0.7694033 to the right, agree=0.787, adj=0.428, (0 split)  
## DMDMARTZ\_Sep < 0.6618522 to the right, agree=0.653, adj=0.068, (0 split)  
## BMXLEG < -2.129812 to the left, agree=0.630, adj=0.007, (0 split)  
## BMXHT < -2.253535 to the left, agree=0.629, adj=0.005, (0 split)  
## BMXARML < -2.618245 to the left, agree=0.628, adj=0.001, (0 split)  
##   
## Node number 4: 536 observations  
## mean=84.85821, MSE=627.8344   
##   
## Node number 5: 562 observations, complexity param=0.003828087  
## mean=99.87544, MSE=864.6108   
## left son=10 (146 obs) right son=11 (416 obs)  
## Primary splits:  
## RIDAGEYR < -1.398075 to the left, improve=0.03662437, (0 missing)  
## BMXBMI < 0.32292 to the left, improve=0.02895393, (0 missing)  
## BMXARMC < 0.3163073 to the left, improve=0.02481897, (0 missing)  
## DMDEDUC2\_NA < 0.9305478 to the right, improve=0.02327611, (0 missing)  
## DMDEDUC2\_AA < -0.4815614 to the right, improve=0.01764382, (0 missing)  
## Surrogate splits:  
## DMDEDUC2\_NA < 0.9305478 to the right, agree=0.826, adj=0.329, (0 split)  
## INDFMPIR < -1.53041 to the left, agree=0.746, adj=0.021, (0 split)  
## BMXBMI < -0.5548324 to the left, agree=0.746, adj=0.021, (0 split)  
## BMXARMC < -1.043647 to the left, agree=0.746, adj=0.021, (0 split)  
## BMXHT < -2.298376 to the left, agree=0.744, adj=0.014, (0 split)  
##   
## Node number 6: 968 observations, complexity param=0.00711296  
## mean=102.7355, MSE=1416.277   
## left son=12 (495 obs) right son=13 (473 obs)  
## Primary splits:  
## RIAGENDR\_Male < 0.02868757 to the right, improve=0.02411971, (0 missing)  
## BMXHT < -0.4649097 to the right, improve=0.02055340, (0 missing)  
## BMXARML < 0.8611078 to the right, improve=0.01581797, (0 missing)  
## BMXBMI < 0.3706801 to the right, improve=0.01427778, (0 missing)  
## BMXARMC < -0.1310462 to the right, improve=0.01285898, (0 missing)  
## Surrogate splits:  
## BMXHT < -0.06633024 to the right, agree=0.853, adj=0.700, (0 split)  
## BMXLEG < -0.2429857 to the right, agree=0.773, adj=0.535, (0 split)  
## BMXARML < 0.2864087 to the right, agree=0.770, adj=0.529, (0 split)  
## DMDMARTZ\_Mar < -0.1600155 to the right, agree=0.616, adj=0.214, (0 split)  
## DMDMARTZ\_Sep < 0.6618522 to the left, agree=0.606, adj=0.195, (0 split)  
##   
## Node number 7: 1630 observations, complexity param=0.00585521  
## mean=115.2816, MSE=1255.334   
## left son=14 (464 obs) right son=15 (1166 obs)  
## Primary splits:  
## RIDAGEYR < -0.2317546 to the left, improve=0.011590290, (0 missing)  
## BMXBMI < 0.7617963 to the right, improve=0.011509240, (0 missing)  
## BMXLEG < -1.843139 to the right, improve=0.007419219, (0 missing)  
## DMDBORN4\_Oth < 0.5241397 to the left, improve=0.005611986, (0 missing)  
## BMXARMC < 1.085755 to the right, improve=0.004611308, (0 missing)  
## Surrogate splits:  
## BMXBMI < -1.613299 to the left, agree=0.718, adj=0.009, (0 split)  
## BMXARMC < -1.741519 to the left, agree=0.718, adj=0.009, (0 split)  
## BMXLEG < 2.27452 to the right, agree=0.717, adj=0.006, (0 split)  
## BMXARML < 2.817162 to the right, agree=0.717, adj=0.006, (0 split)  
##   
## Node number 10: 146 observations  
## mean=90.37671, MSE=687.0841   
##   
## Node number 11: 416 observations  
## mean=103.2091, MSE=884.1366   
##   
## Node number 12: 495 observations, complexity param=0.004824224  
## mean=97.02222, MSE=1260.83   
## left son=24 (257 obs) right son=25 (238 obs)  
## Primary splits:  
## RIDAGEYR < 1.177549 to the right, improve=0.03593447, (0 missing)  
## AIALANGA\_NA < 0.7694033 to the right, improve=0.03260307, (0 missing)  
## BMXLEG < 1.59693 to the left, improve=0.02390759, (0 missing)  
## BMXBMI < -0.2837618 to the right, improve=0.02065733, (0 missing)  
## BMXARMC < 1.318379 to the right, improve=0.01047926, (0 missing)  
## Surrogate splits:  
## AIALANGA\_NA < 0.7694033 to the right, agree=0.970, adj=0.937, (0 split)  
## RIDRETH3\_White < 0.3645385 to the right, agree=0.586, adj=0.139, (0 split)  
## BMXARMC < 0.2268366 to the left, agree=0.578, adj=0.122, (0 split)  
## RIDRETH3\_Black < 0.5542921 to the left, agree=0.570, adj=0.105, (0 split)  
## AIALANGA\_English < -2.469158 to the right, agree=0.570, adj=0.105, (0 split)  
##   
## Node number 13: 473 observations, complexity param=0.00377463  
## mean=108.7146, MSE=1509.045   
## left son=26 (156 obs) right son=27 (317 obs)  
## Primary splits:  
## BMXBMI < 0.3964963 to the right, improve=0.02458421, (0 missing)  
## INDFMPIR < -0.05039062 to the left, improve=0.02189095, (0 missing)  
## BMXARMC < 1.19312 to the right, improve=0.01510029, (0 missing)  
## BMXLEG < -2.520729 to the right, improve=0.01028420, (0 missing)  
## BMXARML < 0.6187648 to the right, improve=0.01009494, (0 missing)  
## Surrogate splits:  
## BMXARMC < 0.2304154 to the right, agree=0.854, adj=0.558, (0 split)  
## BMXARML < 0.7226261 to the right, agree=0.706, adj=0.109, (0 split)  
## INDFMPIR < -1.308407 to the left, agree=0.672, adj=0.006, (0 split)  
## BMXHT < 0.6610774 to the right, agree=0.672, adj=0.006, (0 split)  
##   
## Node number 14: 464 observations, complexity param=0.004972343  
## mean=109.2349, MSE=1026.456   
## left son=28 (91 obs) right son=29 (373 obs)  
## Primary splits:  
## BMXBMI < -0.6839137 to the left, improve=0.04853424, (0 missing)  
## RIAGENDR\_Male < 0.02868757 to the left, improve=0.03776510, (0 missing)  
## BMXARMC < -0.6499762 to the left, improve=0.03739263, (0 missing)  
## BMXLEG < 0.03326233 to the left, improve=0.01273680, (0 missing)  
## DMDBORN4\_Oth < 0.5241397 to the left, improve=0.01123883, (0 missing)  
## Surrogate splits:  
## BMXARMC < -0.8110234 to the left, agree=0.894, adj=0.462, (0 split)  
##   
## Node number 15: 1166 observations, complexity param=0.00585521  
## mean=117.6878, MSE=1326.074   
## left son=30 (265 obs) right son=31 (901 obs)  
## Primary splits:  
## BMXBMI < 0.7230719 to the right, improve=0.019870600, (0 missing)  
## RIDAGEYR < 0.6915823 to the right, improve=0.007605915, (0 missing)  
## BMXARMC < 1.085755 to the right, improve=0.006445164, (0 missing)  
## BMXLEG < -1.843139 to the right, improve=0.005717149, (0 missing)  
## BMXARML < 0.5841444 to the right, improve=0.004316487, (0 missing)  
## Surrogate splits:  
## BMXARMC < 1.085755 to the right, agree=0.886, adj=0.498, (0 split)  
## BMXLEG < -2.416484 to the left, agree=0.776, adj=0.015, (0 split)  
## BMXARML < 2.869092 to the right, agree=0.774, adj=0.004, (0 split)  
##   
## Node number 24: 257 observations  
## mean=90.54475, MSE=899.7889   
##   
## Node number 25: 238 observations, complexity param=0.003021559  
## mean=104.0168, MSE=1556.462   
## left son=50 (228 obs) right son=51 (10 obs)  
## Primary splits:  
## BMXLEG < 1.570869 to the left, improve=0.03791935, (0 missing)  
## DMDEDUC2\_9-11 < 1.224965 to the left, improve=0.03211565, (0 missing)  
## BMXBMI < -0.3483024 to the right, improve=0.02225825, (0 missing)  
## BMXHT < 1.478165 to the left, improve=0.02098695, (0 missing)  
## DMDMARTZ\_Nev < 0.7669317 to the right, improve=0.01875647, (0 missing)  
## Surrogate splits:  
## BMXHT < 2.235466 to the left, agree=0.966, adj=0.2, (0 split)  
##   
## Node number 26: 156 observations  
## mean=100.0321, MSE=1199.569   
##   
## Node number 27: 317 observations  
## mean=112.9874, MSE=1605.987   
##   
## Node number 28: 91 observations  
## mean=94.94505, MSE=776.6014   
##   
## Node number 29: 373 observations, complexity param=0.003384505  
## mean=112.7212, MSE=1025.44   
## left son=58 (215 obs) right son=59 (158 obs)  
## Primary splits:  
## RIAGENDR\_Male < 0.02868757 to the left, improve=0.04113595, (0 missing)  
## BMXLEG < 0.03326233 to the left, improve=0.01961291, (0 missing)  
## BMXBMI < 2.071971 to the right, improve=0.01668855, (0 missing)  
## DMDBORN4\_Oth < 0.5241397 to the left, improve=0.01517982, (0 missing)  
## BMXHT < -0.0165078 to the left, improve=0.01417552, (0 missing)  
## Surrogate splits:  
## BMXHT < 0.5016456 to the left, agree=0.831, adj=0.601, (0 split)  
## BMXARML < 0.2033197 to the left, agree=0.769, adj=0.456, (0 split)  
## BMXLEG < 0.2938737 to the left, agree=0.753, adj=0.418, (0 split)  
## INDFMPIR < -0.1527586 to the left, agree=0.619, adj=0.101, (0 split)  
## BMXARMC < -0.0952579 to the left, agree=0.590, adj=0.032, (0 split)  
##   
## Node number 30: 265 observations  
## mean=108.2226, MSE=1089.109   
##   
## Node number 31: 901 observations, complexity param=0.004142728  
## mean=120.4717, MSE=1361.67   
## left son=62 (867 obs) right son=63 (34 obs)  
## Primary splits:  
## BMXLEG < -1.843139 to the right, improve=0.014884930, (0 missing)  
## BMXARMC < 1.211014 to the left, improve=0.009461288, (0 missing)  
## RIDRETH3\_Black < 0.5542921 to the right, improve=0.008043446, (0 missing)  
## INDFMPIR < -0.7231829 to the left, improve=0.007061246, (0 missing)  
## BMXBMI < -0.2708537 to the left, improve=0.006776726, (0 missing)  
## Surrogate splits:  
## BMXHT < -1.954601 to the right, agree=0.971, adj=0.235, (0 split)  
## BMXARML < -2.323971 to the right, agree=0.966, adj=0.088, (0 split)  
##   
## Node number 50: 228 observations  
## mean=102.4079, MSE=1440.917   
##   
## Node number 51: 10 observations  
## mean=140.7, MSE=2786.21   
##   
## Node number 58: 215 observations  
## mean=107.1535, MSE=854.0927   
##   
## Node number 59: 158 observations  
## mean=120.2975, MSE=1159.019   
##   
## Node number 62: 867 observations, complexity param=0.00359949  
## mean=119.5802, MSE=1272.986   
## left son=124 (210 obs) right son=125 (657 obs)  
## Primary splits:  
## INDFMPIR < -0.7743669 to the left, improve=0.015161550, (0 missing)  
## BMXARMC < 1.211014 to the left, improve=0.011285510, (0 missing)  
## BMXBMI < -0.2708537 to the left, improve=0.008664953, (0 missing)  
## RIDAGEYR < 0.6915823 to the right, improve=0.006674608, (0 missing)  
## RIDRETH3\_Black < 0.5542921 to the right, improve=0.006344337, (0 missing)  
## Surrogate splits:  
## BMXARMC < -2.036772 to the left, agree=0.760, adj=0.010, (0 split)  
## BMXBMI < -1.548758 to the left, agree=0.759, adj=0.005, (0 split)  
##   
## Node number 63: 34 observations, complexity param=0.004142728  
## mean=143.2059, MSE=3085.987   
## left son=126 (27 obs) right son=127 (7 obs)  
## Primary splits:  
## BMXARML < -1.164187 to the left, improve=0.19305520, (0 missing)  
## BMXHT < -1.112601 to the left, improve=0.16323750, (0 missing)  
## INDFMPIR < -0.8033506 to the right, improve=0.11564990, (0 missing)  
## RIDAGEYR < 0.1084221 to the left, improve=0.07737871, (0 missing)  
## DMDEDUC2\_AA < 0.3718316 to the right, improve=0.07447832, (0 missing)  
## Surrogate splits:  
## BMXHT < -1.022921 to the left, agree=0.912, adj=0.571, (0 split)  
## BMXBMI < 0.4971797 to the left, agree=0.824, adj=0.143, (0 split)  
##   
## Node number 124: 210 observations  
## mean=111.8095, MSE=1186.621   
##   
## Node number 125: 657 observations  
## mean=122.0639, MSE=1275.122   
##   
## Node number 126: 27 observations  
## mean=130.7778, MSE=1924.099   
##   
## Node number 127: 7 observations  
## mean=191.1429, MSE=4673.837

# 127 nodes. means and MSE of the terminal nodes vary widely  
rpart.plot(Chol\_rpart)

Diagram

Description automatically generated

# tune and predict  
Chol\_rpart\_tune <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y, method = "rpart", cp = 0.003)  
Chol\_rpart\_pred <- predict(Chol\_rpart\_tune, Chol\_test\_X\_imp)  
postResample(pred = Chol\_rpart\_pred, obs = Chol\_test\_y)

## RMSE Rsquared MAE   
## 33.0113573 0.0659519 25.8550917

# Rsquared value of 0.066 isn't too great. RMSE of 33.0. Let's compare to other trees

### Tree with CART based splits (rpart2 to tune over max depth)

set.seed(123)  
Chol\_rpart2\_tune <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y, method = "rpart2", maxdepth = 6)  
Chol\_rpart2\_pred <- predict(Chol\_rpart2\_tune, Chol\_test\_X\_imp)  
postResample(pred = Chol\_rpart2\_pred, obs = Chol\_test\_y)

## RMSE Rsquared MAE   
## 32.93397878 0.07117066 25.65328561

# minor improvement? Rsquared value of 0.071 and RMSE of 32.9

## Bagged Trees

set.seed(123)  
Chol\_bagtree <- train(x=Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y, method = "treebag", nbagg = 70, cp = 0.003, trControl = Chol\_control)  
Chol\_bagtree

## Bagged CART   
##   
## 3696 samples  
## 32 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 51, 50, 48, 50, 46, 46, ...   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 35.31841 0.03281359 27.35183

# Rsquared value of 0.033. RMSE is 35.3. Still not fantastic

## Random Forest

set.seed(123)  
  
rfmtryValues <- seq(1,10,1)  
  
Chol\_rf <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y, method = "rf", ntree = 300, tuneGrid = data.frame(mtry=rfmtryValues), trControl=Chol\_control) # 300 trees provides the highest Rsquared value when checking with test data  
invisible(Chol\_rf) # output hidden to reduce clutter  
plot(Chol\_rf)

Chart, line chart

Description automatically generated

Chol\_rf\_pred <- predict(Chol\_rf, Chol\_test\_X\_imp)  
postResample(pred = Chol\_rf\_pred, obs = Chol\_test\_y)

## RMSE Rsquared MAE   
## 33.06977523 0.07717907 26.17119444

# Rsquared value of 0.077. RMSE of 33.1

## Boosted Trees

# some control parameters  
gbmGrid <- expand.grid(interaction.depth = c(1,3,5,7,9), n.trees=300, shrinkage = c(0.01, 0.1), n.minobsinnode=5)  
  
set.seed(123)  
Chol\_gbm <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y, method = "gbm", tuneGrid = gbmGrid, verbose = FALSE, trControl = Chol\_control)  
invisible(Chol\_gbm) # output hidden to reduce clutter. Key observations noted below:  
# shrinkage of 0.01 and a smaller interaction depth provided models with the lowest RMSE values  
  
Chol\_gbm\_pred <- predict(Chol\_gbm, Chol\_test\_X\_imp)  
postResample(pred = Chol\_gbm\_pred, obs = Chol\_test\_y)

## RMSE Rsquared MAE   
## 32.86446826 0.08137564 25.82653202

# RMSE 32.9 and Rsquared 0.081

## Model Trees

### Model Trees (M5)

# decision tree with linear regression at terminal nodes to predict continuous variables  
set.seed(123)  
Chol\_M5 <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y, method = "M5", trControl = Chol\_control, control = Weka\_control(M=10))  
  
plot(Chol\_M5)

Chart, line chart

Description automatically generated

Chol\_M5\_pred <- predict(Chol\_M5, Chol\_test\_X\_imp)  
postResample(pred = Chol\_M5\_pred, obs = Chol\_test\_y)

## RMSE Rsquared MAE   
## 33.61012224 0.07641278 26.16774336

# Rsquared value of 0.076 and RMSE of 33.6

### Model Tree (Rule Based M5)

# decision tree with linear regression at terminal nodes to predict continuous variables  
set.seed(123)  
Chol\_M5rules <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y, method = "M5Rules", trControl = Chol\_control, control = Weka\_control(M=10))  
  
plot(Chol\_M5rules)

Chart, line chart

Description automatically generated

Chol\_M5rules\_pred <- predict(Chol\_M5rules, Chol\_test\_X\_imp)  
postResample(pred = Chol\_M5rules\_pred, obs = Chol\_test\_y)

## RMSE Rsquared MAE   
## 32.75281564 0.08668215 25.26610200

# Slight improvement, Rsquared value of 0.087 and RMSE of 32.8

## Cubist

set.seed(123)  
Chol\_cube <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y, method = "cubist", trControl = Chol\_control)  
  
plot(Chol\_cube)

Chart, line chart

Description automatically generated

Chol\_cube\_pred <- predict(Chol\_cube, Chol\_test\_X\_imp)  
postResample(pred = Chol\_cube\_pred, obs = Chol\_test\_y)

## RMSE Rsquared MAE   
## 32.63815970 0.09218122 25.11596950

# "Best performing" so far, but ever so slightly over the other models. Rsquared of 0.092 and RMSE of 32.6

# Results

### Create Data set with all of the models

NonLpred <- NonLpred %>% select(!obs)  
Trees <- cbind.data.frame(Cubist = Chol\_cube\_pred, GBM = Chol\_gbm\_pred, M5 = Chol\_M5\_pred, M5Rules = Chol\_M5rules\_pred, RF = Chol\_rf\_pred, CART = Chol\_rpart2\_pred)  
  
Results <- cbind(Linear\_res, NonLpred, Trees)

### Get RMSE and Plot

find\_rmse <- function(x){  
 caret::RMSE(x, Results[,"Observed"])  
}  
  
RMSE\_results <- apply(X = Results[,2:22], FUN = find\_rmse, MARGIN = 2)  
RMSE\_results <- data.frame(RMSE\_results)  
RMSE\_results$Model <- rownames(RMSE\_results)  
RMSE\_results$Model\_Type <- "Linear"  
RMSE\_results$Model\_Type[11:15] <- "Non-Linear"  
RMSE\_results$Model\_Type[16:21] <- "Tree"  
  
ggplot(RMSE\_results, aes(x=reorder(Model, -RMSE\_results), y=RMSE\_results)) + geom\_segment(aes(x=reorder(Model, -RMSE\_results), xend = reorder(Model, -RMSE\_results), y=30, yend=RMSE\_results, color = Model\_Type)) + geom\_point(aes(color=Model\_Type), size = 9) + coord\_flip() + ylab("RMSE") + geom\_text(aes(label = round(RMSE\_results, 2)), color = "black", size = 2.5, fontface = "bold") + labs(color = "Model Type") + xlab("Model")

Chart

Description automatically generated

### Plot best model (cubist) against the original data

cubist\_plot <- Results %>% select(Observed, Cubist)  
cubist\_plot$x <- c(1:921)  
cubist\_plot <- cubist\_plot %>% gather(key = "Data Source", value = "LDL", -x)  
# cubist\_plot$alpha <- ifelse(cubist\_plot$Observed == "Observed", 0.8, 1)  
ggplot(cubist\_plot, aes(x = x, y = LDL)) + geom\_line(aes(color = `Data Source`, alpha = `Data Source`)) + scale\_color\_manual(values = c("royalblue1", "royalblue4")) + scale\_alpha\_manual(values = c(1,.3)) + theme\_classic()

Chart, bar chart

Description automatically generated