Midterm Project: Heart Study Data

Nathan Kurtz-Enko, Sawyer Jacobsen, Daniel Kindem

April 7, 2019

Background:

`One of the most important factors in the health of citizens around the world is heart disease. Through `

The data set we are using is the framingham data set. The Framingham Heart Study is an investigation into the set of causes for cardiovascular disease among a population in the community of Framingham, Massachusetts. Much of today's common knowledge concerning heart disease and the factors associated with it are based off of this study.

Prediction Question:

What factors are most associated with Ten Year coronary heart disease, and can we create a model that accurately predicts when a patient will contract coronary heart disease.

Summary of our approach:

The models we will be using are Logistic Regression, Lasso, KNN, and decision tree

KNN needs all variables to be numeric, so we need a separate dataset for this. We will evaluate a K Nearest Neighbor model over a range of possible values for K when training. From here we will select whichever values achieves the best result and test the model on a test data set.

Use Lasso to narrow the number of predictors and then use logistic regression to see how the data fairs

Reading in data set

```
#loading and cleaning the data
directory_name = "~/ADM/midterm/"
file_name = "framingham.csv"
paste(directory_name, file_name, sep = "")
```

```
## [1] "~/ADM/midterm/framingham.csv"
```

```
framingham = read.csv(paste(directory_name, file_name, sep = ""))
#removing the NAs
clean_framingham = na.omit(framingham)
```

```
nrow(clean_framingham)
```

```
## [1] 3658
```

```
framingham.df = data.frame(scale(clean_framingham[,-ncol(clean_framingham)]),
                           TenYearCHD = clean_framingham$TenYearCHD)
summary(framingham.df)
```

```
##       male              age             education        currentSmoker
##  Min.   :-0.8929   Min.   :-2.04997   Min.   :-0.95860   Min.   :-0.9782
##  1st Qu.:-0.8929   1st Qu.:-0.88203   1st Qu.:-0.95860   1st Qu.:-0.9782
##  Median :-0.8929   Median :-0.06446   Median : 0.01925   Median :-0.9782
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.0000
##  3rd Qu.: 1.1196   3rd Qu.: 0.75310   3rd Qu.: 0.99709   3rd Qu.: 1.0220
```

```
##  Max.   : 1.1196   Max.   : 2.38823   Max.   : 1.97494   Max.   : 1.0220
##    cigsPerDay          BPMeds        prevalentStroke     prevalentHyp
##  Min.   :-0.7571   Min.   :-0.1769   Min.   :-0.07598   Min.   :-0.6728
##  1st Qu.:-0.7571   1st Qu.:-0.1769   1st Qu.:-0.07598   1st Qu.:-0.6728
##  Median :-0.7571   Median :-0.1769   Median :-0.07598   Median :-0.6728
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000
##  3rd Qu.: 0.9206   3rd Qu.:-0.1769   3rd Qu.:-0.07598   3rd Qu.: 1.4860
##  Max.   : 5.1146   Max.   : 5.6521   Max.   :13.15839   Max.   : 1.4860
##     diabetes          totChol             sysBP             diaBP
##  Min.   :-0.1668   Min.   :-2.80849   Min.   :-2.2127   Min.   :-2.91601
##  1st Qu.:-0.1668   1st Qu.:-0.69953   1st Qu.:-0.6959   1st Qu.:-0.66117
##  Median :-0.1668   Median :-0.06458   Median :-0.1979   Median :-0.07658
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.00000
##  3rd Qu.:-0.1668   3rd Qu.: 0.59305   3rd Qu.: 0.5209   3rd Qu.: 0.59152
##  Max.   : 5.9950   Max.   : 8.23518   Max.   : 7.3632   Max.   : 4.97592
##       BMI             heartRate           glucose          TenYearCHD
##  Min.   :-2.51938   Min.   :-2.64830   Min.   :-1.7509   Min.   :0.0000
##  1st Qu.:-0.66480   1st Qu.:-0.64522   1st Qu.:-0.4540   1st Qu.:0.0000
##  Median :-0.09908   Median :-0.06099   Median :-0.1612   Median :0.0000
##  Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.0000   Mean   :0.1523
##  3rd Qu.: 0.55458   3rd Qu.: 0.52324   3rd Qu.: 0.2153   3rd Qu.:0.0000
##  Max.   : 7.62918   Max.   : 5.61442   Max.   :13.0583   Max.   :1.0000
```

KNN

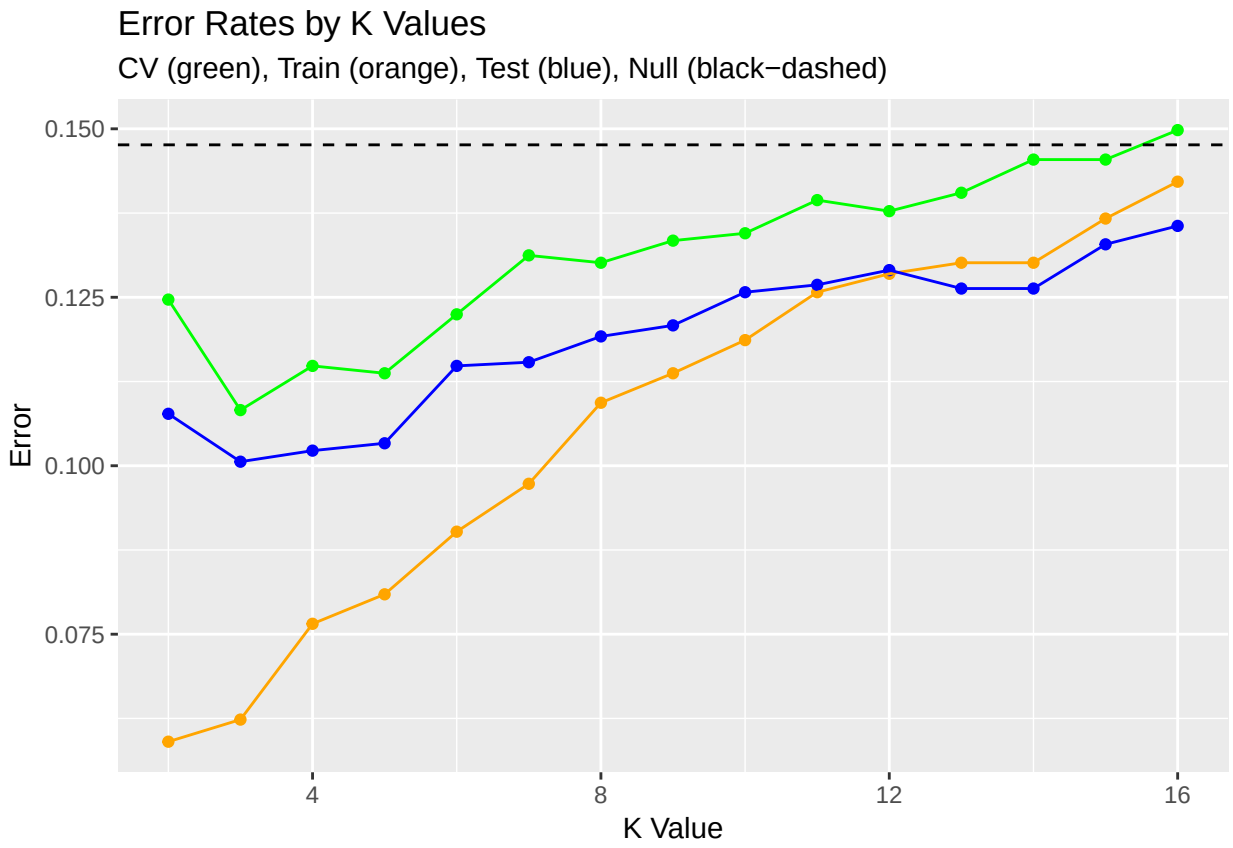
```
#more_stuff <- tibble(k=0, err_cv=0, err_train=0, err_test=0)
#for(i in 1:5){

n = nrow(framingham.df)
train = sample(1:n, n/2, rep = F)
train.df = framingham.df[train,]
test.df = framingham.df[-train,]
#define k values
kval_rng <- 2:16
#tibbles to store stuff
stuff <- tibble(k = kval_rng, err_cv = kval_rng,
                err_train = kval_rng, err_test = kval_rng)
#rearrange data from knn() and knn.cv()
train_result <- train.df$TenYearCHD
knn_train <- train.df[,-1]
test_result <- test.df$TenYearCHD
knn_test <- test.df[,-1]
for(k in kval_rng){
  #errs <- tibble(cv_err = 1:10, train_err = 1:10, test_err = 1:10)
  #for(i in 1:10){
  mod.knn <- knn.cv(knn_train, train_result, k)
  err <- mean(train_result != mod.knn)
  # errs[i, 1] = err
  stuff[k-1, 2] <- err
  mod.knn <- knn(knn_train, knn_train, train_result, k)
  err <- mean(train_result != mod.knn)
  #errs[i,2] = err
  stuff[k-1, 3] <- err
```

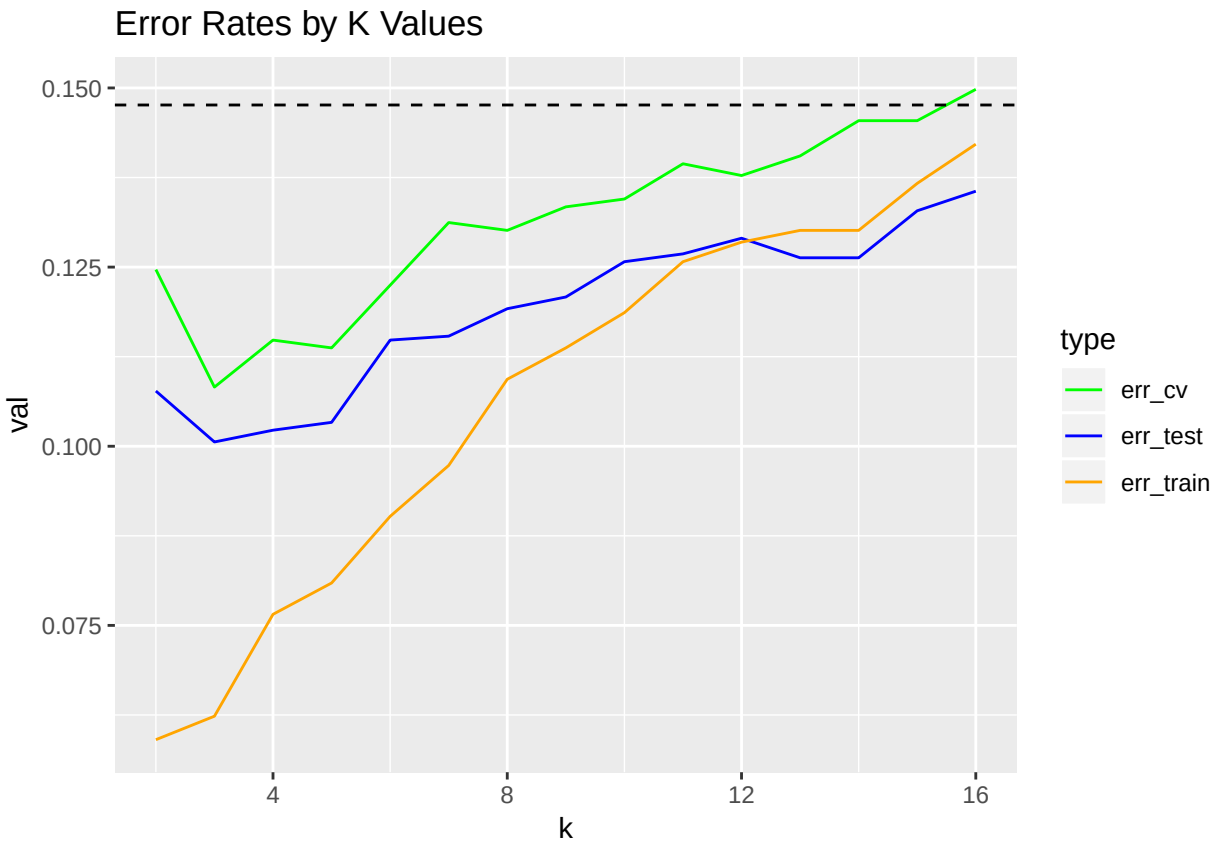
```
  mod.knn <- knn(knn_train, knn_test, train_result, k)
  err <- mean(test_result != mod.knn)
  #errs[i,3] = err
  stuff[k-1, 4] <- err
#}
#mean_cv <- mean(errs$cv_err)
#mean_train <- mean(errs$train_err)
#mean_test <- mean(errs$test_err)
#stuff[k-1, 2] = mean_cv
#stuff[k-1, 3] = mean_train
#stuff[k-1, 4] = mean_test
}
#more_stuff <- rbind(more_stuff, stuff)
#}
#more_stuff <- more_stuff[-1,] %>%
#  group_by(k) %>%
#  summarise(err_cv = mean(err_cv),
#            err_train = mean(err_train),
#            err_test = mean(err_test))
```

```
#null rate for test data set
nr <- sum(test.df$TenYearCHD)/nrow(test.df)


ggplot(stuff)+
  geom_point(aes(x = k, y = err_cv), color = "green")+
  geom_point(aes(x = k, y = err_train), color = "orange")+
  geom_point(aes(x = k, y = err_test), color = "blue")+
  geom_line(aes(x = k, y = err_cv), color = "green")+
  geom_line(aes(x = k, y = err_train), color = "orange")+
  geom_line(aes(x = k, y = err_test), color = "blue")+
  geom_hline(yintercept = nr, linetype = "dashed")+
  labs(y = "Error", x = "K Value",
       title = "Error Rates by K Values",
       subtitle = "CV (green), Train (orange), Test (blue), Null (black-dashed)")
```




```
stuff %>%
    gather(type,val,err_cv:err_test) %>%
    ggplot()+
    geom_line(aes(k,val,color=type))+
    scale_color_manual(values=c("green","blue","orange"))+
    geom_hline(yintercept = nr, linetype = "dashed")+
    labs(title = "Error Rates by K Values")
```




```
#best k value
best_k <- (filter(stuff, err_cv == min(stuff$err_cv)))$k

#testing model
  test.knn <- knn(knn_train, knn_test, train_result)
  err_knn <- mean(test_result != test.knn)



c(err, nr)
```

```
## [1] 0.1355932 0.1476217
```

```
#confusion matrix
cm <- table(test.df$TenYearCHD, test.knn)
cm[1,1]
```

```
## [1] 1503
```

```
## [1] 0.09
```

(Jonathan's stuff) ##Determine Best Predictors: Linear Regression

Setup

Calculate Thresh Error

```
calcErr <- function(thresh) {
  pred = ifelse(test.df$vals < thresh,0,1)
  mean(test.df$TenYearCHD != pred)
```

```
}
```

Kfold CV Function

```
mseCV <- function(data.df,kfolds=10){
  threshVals<-seq(0,1,length.out = 100)
  sampleSize <- nrow(data.df)
  folds <- sample(1:kfolds,sampleSize,rep=T)
    #mse <- rep(0,kfolds)
  mse=foreach(k = 1:kfolds, .combine=cbind) %dopar% {

calcErr <- function(thresh) {
  pred = ifelse(test.df$vals < thresh,0,1)
  mean(test.df$TenYearCHD != pred)
}

    library(purrr)
      train.df <- data.df[folds !=k,]
      test.df <- data.df[folds==k,]
      mod <- lm(TenYearCHD~.,data=train.df)
      test.df$vals <- predict(mod,newdata=test.df)
      err.preds<-map_dbl(threshVals,calcErr)
      min(err.preds)
    }
  mean(mse)
}
```

Bootstrap CV Function

```
mseBoot <- function(data.df,M=50){
  sampleSize <- nrow(data.df)
  threshVals<-seq(0,1,length.out = 100)
  #mse <- rep(0,M)
  mse=foreach(m = 1:M, .combine=cbind) %dopar% {

calcErr <- function(thresh) {
  pred = ifelse(test.df$vals < thresh,0,1)
  mean(test.df$TenYearCHD != pred)
}

    library(purrr)
      bootSamp <- sample(1:sampleSize,sampleSize,rep=T)
      outOfBag <- setdiff(1:sampleSize,bootSamp)
      train.df <- data.df[bootSamp,]
      test.df <-   data.df[outOfBag,]
      mod <- lm(TenYearCHD~.,data=train.df)
      test.df$vals <- predict(mod,newdata=test.df)
      err.preds<-map_dbl(threshVals,calcErr)
      min(err.preds)
  }
  mean(mse)
}
```

Setup For Main Loop

The last field is the response variable in this case

```
numPreds <- length(names(train.df))-1
(allPreds <- 1:(numPreds))
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
```

```
currPreds <- c()
availPreds <- setdiff(allPreds,currPreds)
maxPreds <- numPreds
minMSE <- numeric(maxPreds)
nr <- sum(framingham.df$TenYearCHD)/nrow(framingham.df)
```

Main Loop

```
currPreds <- c()
availPreds <- setdiff(allPreds,currPreds)
maxPreds <- numPreds
minMSE <- numeric(maxPreds)
cores=detectCores()-1 #not to overload your computer
cl <- makeCluster(cores)
registerDoParallel(cl)

tot <- 0
while( tot < maxPreds){
    ##add predictor which decreases MSE (as determined by CV or
    ##Bootstrapping)
    ## The MSEs computed as we add each of the available predictors
    allMSE <- numeric(length(availPreds))
    ct<-1
    for(id in availPreds){
        #library(foreach)
        data2.df <- framingham.df[,c(currPreds,id,numPreds+1)]
        mse <- (mseCV(data2.df)+mseBoot(data2.df))/2
        allMSE[ct]<-mse
        ct<-ct+1
    }
    ##Find the min
    id <- which.min(allMSE)
    ##get the best predictor and MSW
    bestPred <- availPreds[id]
    bestMSE <- min(allMSE)
    ##Add these into the collection
    currPreds <- c(currPreds,bestPred)
    tot <-tot+1
    minMSE[tot] <- bestMSE
    availPreds <- setdiff(allPreds,currPreds)
    ## Print stuff out for debugging and attention-grabbing
    print(sprintf("Predictor Added: %s: %s  MeanError Value: %s",bestPred,
                  colnames(framingham.df[bestPred]),bestMSE))
    print(currPreds)
}
```

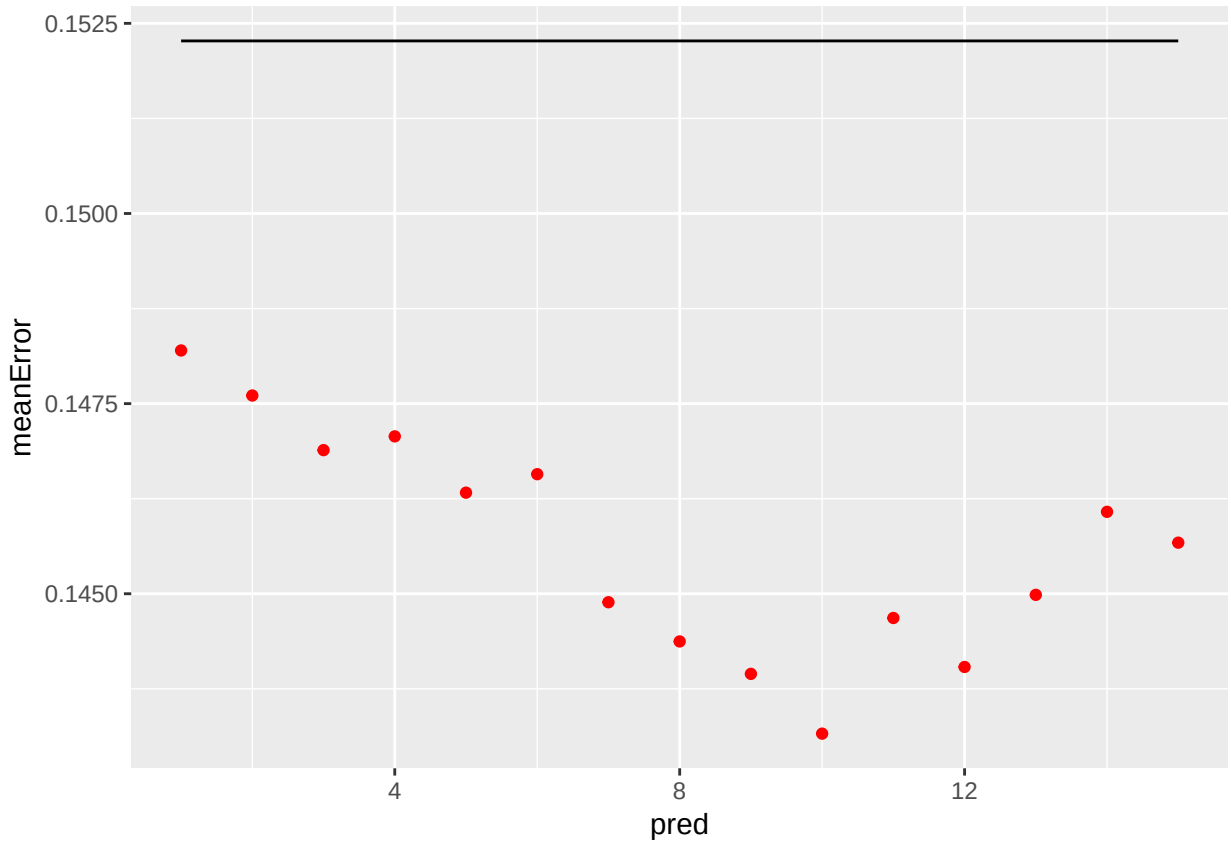
```
## [1] "Predictor Added: 11: sysBP  MeanError Value: 0.148198842646725"
```

```
## [1] 11
## [1] "Predictor Added: 1: male  MeanError Value: 0.1476065263572"
## [1] 11  1
## [1] "Predictor Added: 6: BPMeds  MeanError Value: 0.146888491392433"
## [1] 11  1  6
## [1] "Predictor Added: 2: age  MeanError Value: 0.147069494968354"
## [1] 11  1  6  2
## [1] "Predictor Added: 4: currentSmoker  MeanError Value: 0.146329344192945"
## [1] 11  1  6  2  4
## [1] "Predictor Added: 15: glucose  MeanError Value: 0.146572087748906"
## [1] 11  1  6  2  4 15
## [1] "Predictor Added: 5: cigsPerDay  MeanError Value: 0.144888508294562"
## [1] 11  1  6  2  4 15  5
## [1] "Predictor Added: 7: prevalentStroke  MeanError Value: 0.144373069317167"
## [1] 11  1  6  2  4 15  5  7
## [1] "Predictor Added: 8: prevalentHyp  MeanError Value: 0.143946621272505"
## [1] 11  1  6  2  4 15  5  7  8
## [1] "Predictor Added: 14: heartRate  MeanError Value: 0.143160993647078"
##  [1] 11  1  6  2  4 15  5  7  8 14
## [1] "Predictor Added: 10: totChol  MeanError Value: 0.144681601142625"
##  [1] 11  1  6  2  4 15  5  7  8 14 10
## [1] "Predictor Added: 9: diabetes  MeanError Value: 0.144037413648152"
##  [1] 11  1  6  2  4 15  5  7  8 14 10  9
## [1] "Predictor Added: 13: BMI  MeanError Value: 0.144985544502881"
##  [1] 11  1  6  2  4 15  5  7  8 14 10  9 13
## [1] "Predictor Added: 12: diaBP  MeanError Value: 0.146077493460915"
##  [1] 11  1  6  2  4 15  5  7  8 14 10  9 13 12
## [1] "Predictor Added: 3: education  MeanError Value: 0.14567182371884"
##  [1] 11  1  6  2  4 15  5  7  8 14 10  9 13 12  3
```

```
stopCluster(cl)
```

Plot

```
forplot.df<-data.frame(pred=1:maxPreds,meanError=minMSE,
                       nullRate=numeric(length(maxPreds))+nr)
forplot<-forplot.df %>%
    ggplot()+
    geom_point(aes(pred,meanError),color="red")+
    geom_line(aes(pred,nullRate),color="black")
forplot
```

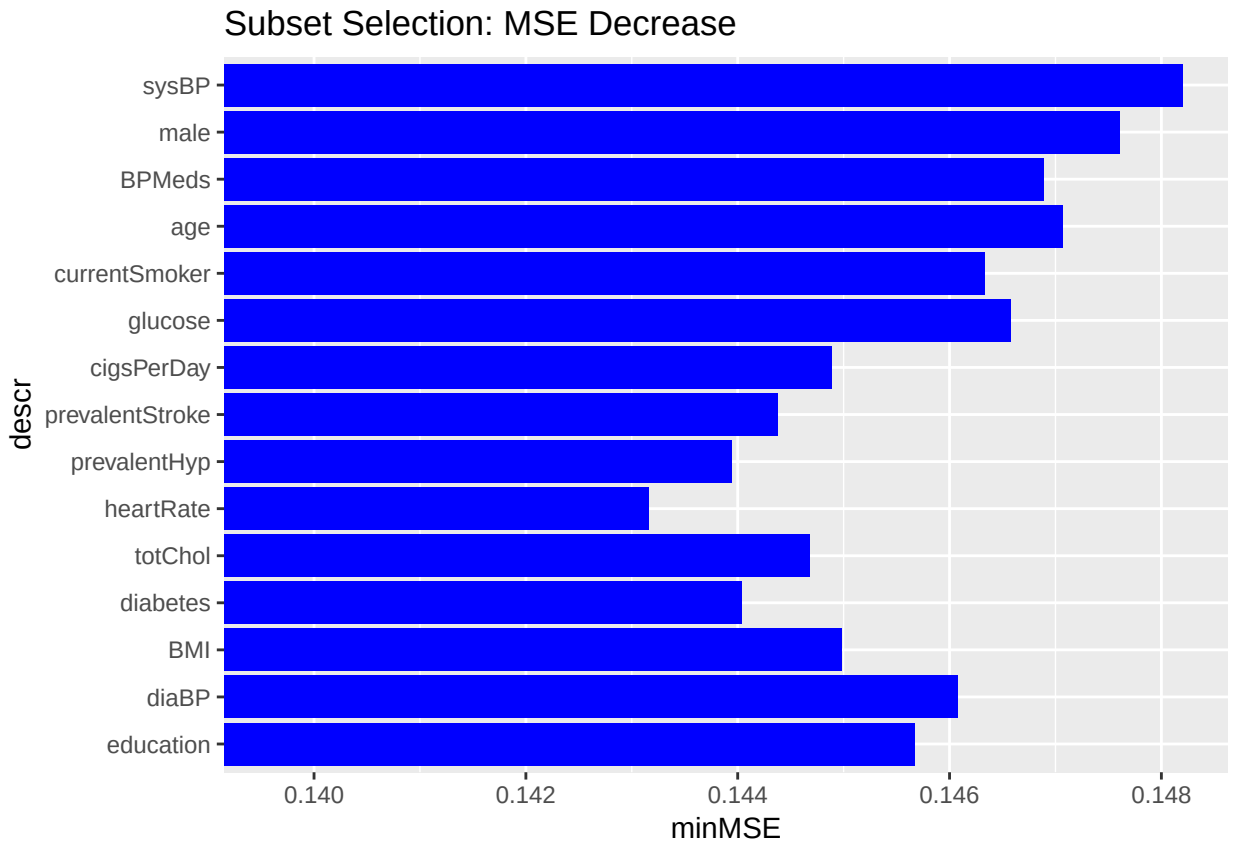



```
descr <- colnames(framingham.df[currPreds])
diffMSE <- c(0,minMSE[-length(minMSE)]-minMSE[-1])
head(diffMSE)
```

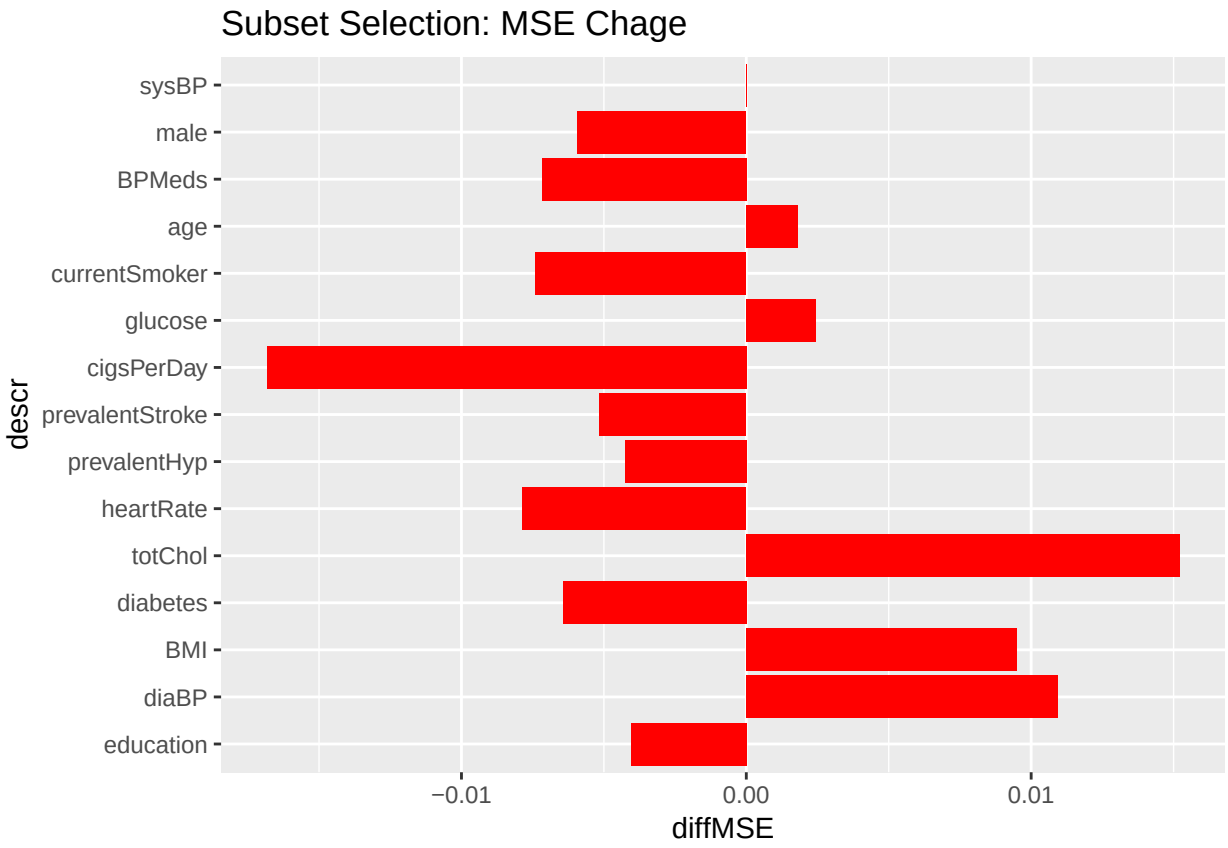
```
## [1]  0.0000000000  0.0005923163  0.0007180350 -0.0001810036  0.0007401508
## [6] -0.0002427436
```

```
result.df <- data.frame(id=1:length(descr),descr,minMSE,diffMSE=-10*diffMSE)
result.df <- result.df%>%
    mutate(descr=factor(descr,levels=rev(descr)))
```

```
ggplot(result.df,
    aes(descr,minMSE))+
    geom_bar(stat="identity",fill="blue")+
    coord_flip(ylim=c(min(minMSE)*.975,max(minMSE)))+
    ggtitle("Subset Selection: MSE Decrease")
```




```
ggplot(result.df,aes(descr,diffMSE))+
    geom_bar(stat="identity",fill="red")+
    coord_flip()+
    ggtitle("Subset Selection: MSE Chage")
```



Based on our work and calculated errors, we found that for this particular data set, the best method to use in order to accurately predict instances of heart disease is KNN with an error rate of 0.1011482