Predicting Coronary Artery Disease with Neural Networks using H2O

David Noonan

Coronary artery disease (CAD) is a group of diseases which are the most common type of heart disease. It is estimated to kill 8.9 million people each year worldwide. An important source of data on CAD comes from the National Health and Nutrition Examination Survey (NHANES). It is a survey of physical examinations, laboratory exams, and interviews from a nationwide representative sample in the United States collected by the National Center for Health Statistics. Using this data, I aim to predict whether a person has coronary heart disease based on laboratory test data using artificial neural networks with the h2o package in R. The goal of the model is classification: whether the disease is present or not.

Step 1 Collecting the Data

The data were collected from the NHANES website's 2013-2014 survey. It is formatted in SAS xport files (.XPT). These were imported into R using the **sasxport.get** function from the **Hmisc** package in R.

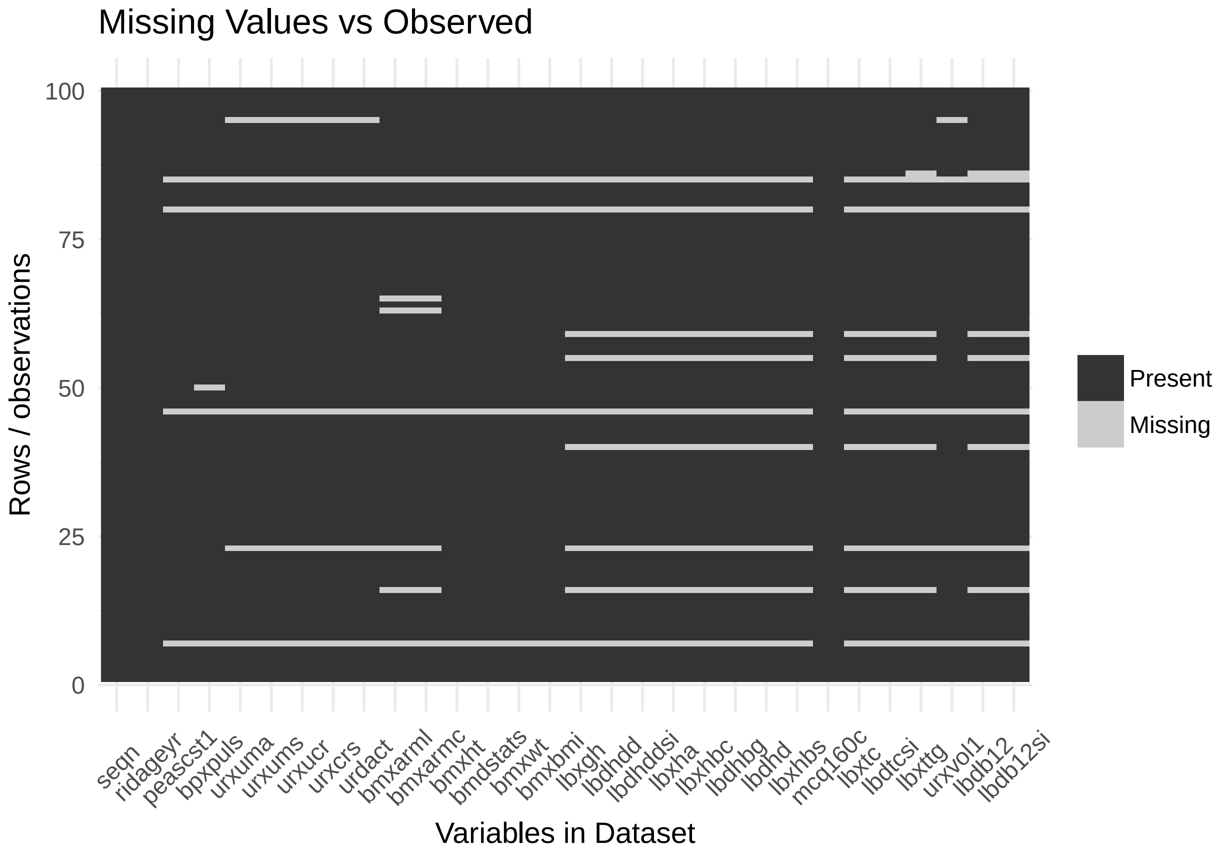
Step 2 Exploring and Preparing the data

Variable subset selection

After import, I merged the data frames. The result was a data frame of 10175 observations from 719 variables. I needed to decide which variables to use. Arbitrarily, I chose to remove variables with greater than 1000 missing observations. This was a very large proportion of the data. The result was a data frame of 5769 observations from 30 variables.

Handling NAs

Next, I needed to address the missing values in the data. Below is a map of the missing dataor the first 100 observatons:



From this plot, we see that the missing data is correlated. If an observation is missing in one laboratory test, it's likely missing in all or many of the others. I removed the observations that had this pattern. The result was 5046 observations from 30 variables. I also removed observations with missing values when they made up a small fraction of the dataset (less than .01%).

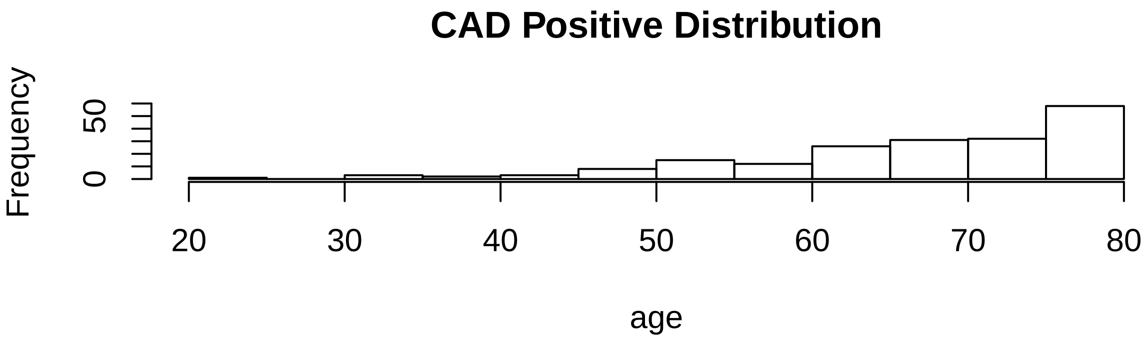
Another change to the data was to recode the outcome variable. Its name is "mcq160c", which is an observation based on an interview question: "have you been told [by a medical professional] that you have coronary [artery] disease?". The answer is "1" for positive, and "2" for negative. I changed "1" to "positive" and "2" to "negative." Finally, I renamed the variable "CAD", for coronary artery disease. Below is the distriubtion of that variable:

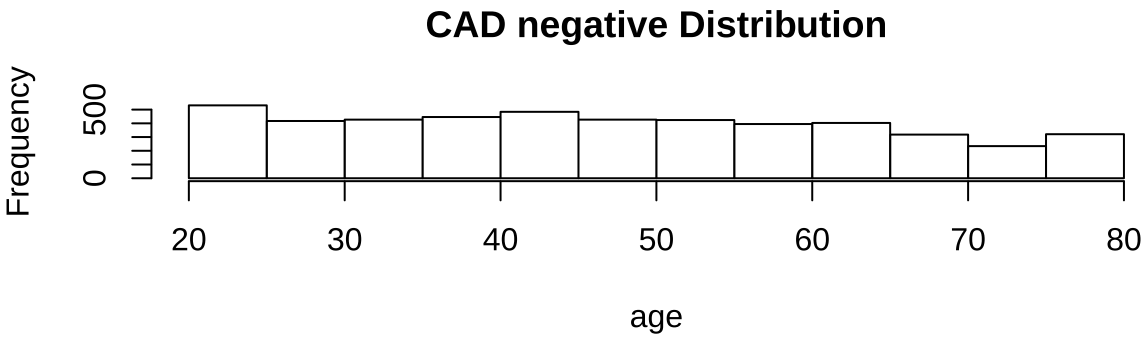
```
##
##    1    2
##  191 4827
```

The number of diseased patients is slightly less than 4% of the data. Next we look at the mean age for each outcome:

```
## # A tibble: 2 × 3
##        CAD count mean.age
##     <fctr> <int>    <dbl>
## 1 positive   191 67.49738
## 2 negative  4827 47.80029
```

It is interesting to note that age is strongly associated with CAD here. Below is a comparison of the age distributions for each outcome:





We see that CAD positive observations tend to occur with greater ages. However, there are some CAD cases in the 20s and 30s.

Step 3: Training the model on the data

Our next step is to train the model on the data. We will use the h2o package in R to fit an artificial neural net model to the data. We will begin with default settings. A neural net with two hidden layers, and the activation function "Rectifier." I used the h2o.splitFrame command to specify training and test set proportions at .75 and .25 respectively.

After training the model on the data, we get an output of the most important variables. Below is a table of the most important variables according to the model:

```
##    variable relative_importance scaled_importance percentage
## 1:   lbdhbg           1.0000000         1.0000000 0.05863132
## 2:    lbdhd           0.9300222         0.9300222 0.05452843
## 3: ridageyr           0.7565607         0.7565607 0.04435815
## 4:   urxucr           0.7223135         0.7223135 0.04235019
## 5:   lbdb12           0.6931577         0.6931577 0.04064075
## 6:    lbxtc           0.6862274         0.6862274 0.04023442
```

Interestingly, the two most important variables have to do with hepatitis b. Next in order of importance is age, which I would have expected to be at the top, given the histograms shown earlier in this report. The next three variables involve hepatitis B again, vitamin B 12, and urine volume discharge.

Step 4: Evaluating model performance

I used the model to predict classifications in the test dataset. Below is a confusion matrix of the predictions:

```
## 
## 
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
## 
## 
## Total Observations in Table:  1241 
## 
## 
##                | predicted default 
## actual default |  negative |  positive | Row Total | 
## ---------------|-----------|-----------|-----------|
##       negative |      1164 |        28 |      1192 | 
##                |     0.938 |     0.023 |           | 
## ---------------|-----------|-----------|-----------|
##       positive |        44 |         5 |        49 | 
##                |     0.035 |     0.004 |           | 
## ---------------|-----------|-----------|-----------|
##   Column Total |      1208 |        33 |      1241 | 
## ---------------|-----------|-----------|-----------|
## 
## 
```

The accuracy of this model is $(1161 + 11)/1241 = .9444$, which looks good but is misleading. Of the positive cases, about 74% of the cases were misclassified as negative, with a true positive rate of 0.2619. Further, there were 38 false positives, a rate of 3.3%. Perhaps a modification of the model can help improve the predictions.

Step 5: Improving model performance

One route to improving the model is to introduce hyperparameter tuning, which involves splitting the dataset into test, training, and validation partitions. In the next step, we create a model with data split to proportions .6, .2 and .2 for training, validation, and testing respectively. For the tuning parameters, we introduce stopping_rounds = 4, stopping_metric = "misclassification", and stopping_tolerance = 0.1, and epochs = 100. Below is a list of the model's most important variables:

```
##    variable relative_importance scaled_importance percentage
## 1:   lbdhbg           1.0000000         1.0000000 0.04904539
## 2:  bpxpuls           0.9942517         0.9942517 0.04876346
## 3:    lbdhd           0.8982791         0.8982791 0.04405645
## 4:  bmxarml           0.8560902         0.8560902 0.04198728
## 5: lbdhddsi           0.8394769         0.8394769 0.04117247
## 6:   urxucr           0.8385633         0.8385633 0.04112766
```

We see hepatitis variables being the most important, but interestingly, age is no longer on the list. Vitamin b12 remains on the list, but we see the addition of a factor for irregular pulse (bpxpuls), urine creatine (urxcrs), and albumin creatine ratio (urdact).

Next, we make predictions and build a confusion matrix:

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  974
##
##
##                 | predicted default
## actual default  |  negative  |  positive  | Row Total |
## ----------------|-----------|-----------|-----------|
##        negative |       910 |        27 |       937 |
##                 |     0.934 |     0.028 |           |
## ----------------|-----------|-----------|-----------|
##        positive |        29 |         8 |        37 |
##                 |     0.030 |     0.008 |           |
## ----------------|-----------|-----------|-----------|
##    Column Total |       939 |        35 |       974 |
## ----------------|-----------|-----------|-----------|
##
##
```

The accuracy is $(918 + 7)/974 = .9497$, which is slightly better than the previous model. The true positive rate is .2692, which is also slightly better.

Two predictions to showcase are from observation id 73562 and 80241. 73562 has CAD, and the model correctly classified it as positive. 80251 on the other hand was negative, and was also classified as positive. It is more often the case that predictions misclassify as positive rather than correctly classify as positive. This issue makes the usefulness of the model questionable in a medical setting.

Conclusion

Overall, the improved model learned to correctly predict coronary artery disease about 27% of positive predictions. This rate is low, but it is better than guessing from the ~4% rate of CAD in the dataset. For future improvements, we could try tree-based methods to see if the classification rate improves. Further, we could add more predictor variables to see if they improve the predictive power of our models.

Code Appendix

```r
library(Hmisc) #contains sasxport.get function

#Nhanes data from 2013-2014 survey

demo <- sasxport.get("https://wwwn.cdc.gov/Nchs/Nhanes/2013-2014/DEMO_H.XPT")
#demographic data

MCQ <- sasxport.get("https://wwwn.cdc.gov/Nchs/Nhanes/2013-2014/MCQ_H.XPT")
#medical conditions data

BPX <- sasxport.get("https://wwwn.cdc.gov/Nchs/Nhanes/2013-2014/BPX_H.XPT")
#blood pressure data

BMX <- sasxport.get("https://wwwn.cdc.gov/Nchs/Nhanes/2013-2014/BMX_H.XPT")
#body measures

ALB <- sasxport.get("https://wwwn.cdc.gov/Nchs/Nhanes/2013-2014/ALB_CR_H.XPT")
#albumin and creatine (Urine)

HDL <- sasxport.get("https://wwwn.cdc.gov/Nchs/Nhanes/2013-2014/HDL_H.XPT")
#HDL cholesterol

TCHOL <- sasxport.get("https://wwwn.cdc.gov/Nchs/Nhanes/2013-2014/TCHOL_H.XPT")
#total cholesterol

GHB <- sasxport.get("https://wwwn.cdc.gov/Nchs/Nhanes/2013-2014/GHB_H.XPT")
#glycohemoglobin

HEPA <- sasxport.get("https://wwwn.cdc.gov/Nchs/Nhanes/2013-2014/HEPA_H.XPT")
#hepatitis A

HEPBS <- sasxport.get("https://wwwn.cdc.gov/Nchs/Nhanes/2013-2014/HEPB_S_H.XPT")
#hepatitis B surface antibody

HEPBD <- sasxport.get("https://wwwn.cdc.gov/Nchs/Nhanes/2013-2014/HEPBD_H.XPT")
#hepatitis B core antibody, surface antigen, and hepatitis D antibody

TGEMA <- sasxport.get("https://wwwn.cdc.gov/Nchs/Nhanes/2013-2014/TGEMA_H.XPT")
#tissue transglutaminase assay and IgA endomyseal antibody assay

UTAS <- sasxport.get("https://wwwn.cdc.gov/Nchs/Nhanes/2013-2014/UTAS_H.XPT")
#total urinary arsenic

UCFLOW <- sasxport.get("https://wwwn.cdc.gov/Nchs/Nhanes/2013-2014/UCFLOW_H.XPT")
#urine flow rate

VITB12 <- sasxport.get("https://wwwn.cdc.gov/Nchs/Nhanes/2013-2014/VITB12_H.XPT")
#vitamin B12
```

```r
library(dplyr)
#From dataframes with greater than 5000 observations:
#Select variables with less than 1000 NAs
#create new data frame with "1" added to the end of the name.
#some dataframes renamed without changing

#select subsets from dataframes with greater than 5000 observations
demo1 <- select(demo, seqn, ridageyr)
BPX1 <- select(BPX, seqn, peascst1, bpxpuls)
ALB1 <- select(ALB, seqn, urxuma, urxums, urxucr, urxcrs, urdact)
BMX1 <- select(BMX, seqn, bmxarml, bmxarmc, bmxht, bmdstats, bmxwt, bmxbmi)
GHB1 <- GHB
HDL1 <- HDL
HEPA1 <- HEPA
HEPBD1 <- HEPBD
HEPBS1 <- HEPBS
MCQ1 <- select(MCQ, seqn, mcq160c)  #outcome variable
TCHOL1 <- TCHOL
TGEMA1 <- select(TGEMA, seqn, lbxttg)
UCFLOW1 <- select(UCFLOW, seqn, urxvol1)
UTAS1 <- select(UTAS, seqn, -wtsa2yr, -urxuas)
VITB121 <- VITB12


#Merge subset dataframes

NHANES <- list(demo1, BPX1, ALB1, BMX1, GHB1, HDL1, HEPA1, HEPBD1,
               HEPBS1, MCQ1, TCHOL1, TGEMA1,UCFLOW1, UTAS1,
               VITB121) %>%
  Reduce(function(dtf1,dtf2) left_join(dtf1,dtf2,by="seqn"), .)

#create subset of observations for ages greater than 19
nhs1 <- filter(NHANES, ridageyr>19, mcq160c != "NA")

#look for patterns in missing data
library(reshape2)
library(ggplot2)
library(dplyr)

#missing data map function from Nicholas Tierney:
#http://www.njtierney.com/r/missing%20data/rbloggers/2015/12/01/ggplot-missing-data/
ggplot_missing <- function(x, title){

  x %>%
    is.na %>%
    melt %>%
    ggplot(data = .,
           aes(x = Var2,
               y = Var1)) +
    geom_raster(aes(fill = value)) +
    scale_fill_grey(name = "",
                    labels = c("Present","Missing")) +
    theme_minimal() +
    theme(axis.text.x  = element_text(angle=45, vjust=0.5)) +
    labs(x = "Variables in Dataset",
         y = "Rows / observations")
}

ggplot_missing(nhs1[1:100,]) +
  labs(title = "Missing Values vs Observed")

#remove correlated missing observations
nhs2 <- filter(nhs1, bmxht != "NA", urxcrs != "NA", lbxha != "NA",
               bmxarmc != "NA", lbdb12 != "NA", lbdhd!= "NA", bmxarml != "NA",
               bmxbmi != "NA", lbxgh != "NA", mcq160c != "9")

#recode missing factor variables as "unknown"

nhs2$bpxpuls <- ifelse(is.na(nhs2$bpxpuls), "unknown", nhs2$bpxpuls)
summary(nhs2$bpxpuls)


#remove irrelevant variables:

nhs3 <- select(nhs2, -c(peascst1,bmdstats,lbxttg,seqn))
names(nhs3)

#Specify factor variables:

nhs3$bpxpuls<- as.factor(nhs3$bpxpuls)
nhs3$lbxha<- as.factor(nhs3$lbxha)
nhs3$lbxhbc<- as.factor(nhs3$lbxhbc)
nhs3$lbdhbg<- as.factor(nhs3$lbdhbg)
nhs3$lbdhd<- as.factor(nhs3$lbdhd)
nhs3$lbxhbs<- as.factor(nhs3$lbxhbs)
nhs3$mcq160c<- as.factor(nhs3$mcq160c)

#change outcome variable levels
levels(nhs3$mcq160c) <- c("positive", "negative")
summary(nhs3$mcq160c)

#change outcome variable name to cAD: coronary art
colnames(nhs3)[21] <- "CAD"
colnames(nhs3)[21]

#Train model on data

library(h2o)

h2o.init(nthreads=4, max_mem_size="2G")
h2o.removeAll() ## clean slate - just in case the cluster was already running

h2o.init()

write.csv(x = nhs3, "nhs3.csv")  #write csv to for making h2o object
```

```r
nhs3.hex <- h2o.importFile("nhs3.csv") #create h2o object from csv file

splits1 <- h2o.splitFrame(nhs3.hex, 0.75, seed=8827) #create splits
predictors <- names(nhs3)[-21] #predictor variables
fit1labels <- as.vector(splits1[[2]]$CAD) #labels for test set

#fit basic neural net model with two hidden layers
fit1 <- h2o.deeplearning(
  x= predictors,
  y="CAD",
  training_frame=splits1[[1]],
  activation = "Rectifier",
  hidden = c(200,200),
  variable_importances = TRUE
  )

library(data.table)
#variable importance
head(as.data.table(h2o.varimp(fit1)))


fit1.predict <- h2o.predict(fit1, splits1[[2]]) #make predictions

#convert predictions to vector
fit1.predictions <- as.vector(fit1.predict$predict)

#confusion matrix for first model
library(gmodels)
CrossTable(fit1labels, fit1.predictions,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual default', 'predicted default'))


#model 2 with hyperparameter tuning

splits2 <- h2o.splitFrame(nhs3.hex, c(.6, .2), seed=8827) #create splits
training <- h2o.assign(splits2[[1]], "train")# training set
validation <- h2o.assign( splits2[[2]], "valid") # validation set
testing  <- h2o.assign( splits2[[3]], "test")  # testing set
fit2labels <- as.vector(splits2[[3]]$CAD) #labels for test set


fit2 <- h2o.deeplearning(
  x= predictors,
  y="CAD",
  training_frame=training,
  validation_frame = validation,
  activation = "Rectifier",
  hidden = c(200,200),
  variable_importances = TRUE,
  epochs = 100,
  stopping_rounds = 4,
  stopping_metric = "misclassification",
  stopping_tolerance = 0.1
)

head(as.data.table(h2o.varimp(fit2)))

#model 2 predictions
fit2.predict <- h2o.predict(fit2, splits2[[3]])

#convert predictions to vector
fit2.predictions <- as.vector(fit2.predict$predict)

#confusion matrix for first model
library(gmodels)
CrossTable(fit2labels, fit2.predictions,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual default', 'predicted default'))


##plots and tables##

library(Amelia)

table(nhs2$mcq160c) #missing data table

#summarize age by classification
cads <- group_by(nhs3, CAD)
summarise(cads,
          count = n(),
          mean.age = mean(ridageyr)
)

#create histograms of age distributions by classification
ages <- select(nhs3, ridageyr, CAD)
pos.ages <- filter(ages, CAD == "positive")
neg.ages <- filter(ages, CAD == "negative")
par(mfrow = c(2,1))
hist(pos.ages$ridageyr, main = "CAD Positive Distribution", xlab = "age")
hist(neg.ages$ridageyr, main = "CAD negative Distribution", xlab = "age")

library(data.table)
#variable importance
head( as.data.table( h2o.varimp(fit1) ) )

#confusion matrix for model 1
library(gmodels)
CrossTable(fit1labels, fit1.predictions,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual default', 'predicted default'))

#important variables for model 2
head(as.data.table(h2o.varimp(fit2)))

#confusion matrix for model 2
```

```
library(gmodels)
CrossTable(fit2labels, fit2.predictions,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual default', 'predicted default'))


h2o.performance(fit2)

h2o.shutdown()
```