Eryn Yuasa

ML_Final_Eryn_Yuasa

Introduction

This final assignment is broken up into three sections: 1) Exploratory Data Analysis and Data Cleaning, 2) Modeling (10 models are considered), and 3) Choosing a Model (Selection and Analysis).

1. Exploratory Data Analysis and Data Cleaning

Filter to only include indicated predictors in assignment

Indicated Predictors, Description, and Categories for References

"RIDAGEYR": Age in years at time of screening interview
"RIAGENDR":Gender 1 = male, 2 = female
"BPQ010": Last blood pressure reading by doctor. 1 = less than 6 months ago, 2 = 6m - 1 year, 3 = more than 1 year to 2 years, 4 = more than 2 years ago, 5 = never, 7 = refused, 9 = don't know, . = missing.
"BPQ060": Ever had blood cholesterol checked 1 = Yes, 2 = No, 7 = Refused, 9 = don't know . = missing.
"DIQ010": Doctor told you have diabetes 1 = yes, 2 = no, 3= borderline, 7 = refused, 9 = don't know, . = missing.
"DIQ050": Taking insulin now 1 = yes, 2 = no, 7 = refused, 9 = don't know . = missing.
"DIQ090": Past year told control weight, increasing physical activity, reduce fat or category
"MCQ010": Ever been told you have asthma 1 = Yes, 2 = No, 7 = Refused, 9 = don't know, . = missing.
"MCQ053": Taking treatment for anemia part 3 months: 1 = Yes, 2 = No, 7 = Refused, 9 = don't know, . = missing.
"MCQ160A": Doctor ever said you have arthritis 1 = Yes, 2 = No, 7 = Refused, 9 = don't know, . = missing.
"MCQ160B": Ever told had congestive heart failure 1 = Yes, 2 = No, 7 = Refused, 9 = don't know, . = missing.
"MCQ160K": Ever told you had chronic bronchitis 1 = Yes, 2 = No, 7 = Refused, 9 = don't know, . = missing.
"MCQ160L": Ever told you had any liver condition 1 = Yes, 2 = No, 7 = Refused, 9 = don't know, . = missing.
"BMXWAIST": Waist Circumference (cm) . = missing
"MCQ160M": Ever told you had a thyroid problem 1 = Yes, 2 = No, 7 = Refused, 9 = don't know, . = missing.
"MCQ220": Ever told you have cancer or malignancy 1 = Yes, 2 = No, 7 = Refused, 9 = don't know, . = missing.

"MCQ245A" - Discontinued
"MCQ250A": blood relatives have diabetes
"MCQ250B": blood relatives have Alzheimer
"MCQ250C": blood relatives have asthma
"MCQ250E": blood relatives have osteoporosis
"MCQ250F": blood relatives have high blood pressure or stroke before 50
"MCQ250G": blood relatives have heart attack or anginia before 50
"MCQ265" - Discontinued
"SSQ011": Anyone to help with emotional support 1 = yes, 2 = no, 3 = doesn't need, 7 = refused, 9 = don't know . = missing.
"SSQ051": Anyone to help with financial support 1 = yes, 2 = no, 3 = wouldn't accept it but offered, 7 = refused, 9 = don't know . = missing.
"WHQ030": How do you consider your weight? 1=over 2=under, 3=about the right weight, 7 = refused, 9 = don't know, . = missing.
"WHQ040": Like to weight more, less, or same 1=more, 2=less 3=same, 7 = refused, 9=don't know, .=missing.
"LBXRDW": red cell distribution width (%) range of values
"HSD010": General health condition 1 = excellent, 2=very good, 3 = good, 4 = fair, 5 = poor, 7 = refused, 9 = don't know, . = missing.
"BPXPULS": Pulse regular or irregular (1 = regular, 2 = irregular, . = missing)
"BPXML1": Maximum inflation levels (mm HG) "VIQ200": Eye surgery for cataracts 1 = yes 2 = no 9 = don't know . = missing
"BMXBMI": Body mass index (kp / m ** 2) "BPXSY1": Systolic: Blood pres (1st rdg) mm Hg
"BPXDI1": Diastolic: Blood pres (1st rdg) mm Hg
mortstat: 0 = assumed alive, 1 = assumed deceased

```
predictors <- c("RIDAGEYR", "RIAGENDR", "BPQ010", "BPQ060", "DIQ010", "DIQ050
", "DIQ090", "MCQ010", "MCQ053", "MCQ160A", "MCQ160B", "MCQ160K", "MCQ160L",
"BMXWAIST", "MCQ160M", "MCQ220", "MCQ245A", "MCQ250A", "MCQ250B", "MCQ250C",
"MCQ250E", "MCQ250F", "MCQ250G", "MCQ265", "SSQ011", "SSQ051", "WHQ030", "WHQ
040", "LBXRDW", "HSD010", "BPXPULS", "BPXML1", "VIQ200", "BMXBMI", "BPXSY1",
"BPXDI1")

load(file='nhanes2003-2004.Rda')

nhanes_select <- nhanes2003_2004 %>%
  select(predictors, "mortstat")

## Note: Using an external vector in selections is ambiguous.
## ℹ Use `all_of(predictors)` instead of `predictors` to silence this messag
e.
## ℹ See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

# Display first 6 rows of dataframe
head(nhanes_select)
```

```
##   RIDAGEYR RIAGENDR BPQ010 BPQ060 DIQ010 DIQ050 DIQ090 MCQ010 MCQ053 MCQ16
0A
## 1       19        1      2   <NA>      2      2   <NA>      2      2    <N
A>
## 2       16        2      2   <NA>      2      2   <NA>      2      2    <N
A>
## 3       14        2   <NA>   <NA>      2      2   <NA>      2      2    <N
A>
## 4       17        1      1   <NA>      2      2   <NA>      1      2    <N
A>
## 5       55        1      2      1      2      2      2      2      2      
2
## 6       52        2      3      1      2      2      2      1      2      
2
##   MCQ160B MCQ160K MCQ160L BMXWAIST MCQ160M MCQ220 MCQ245A MCQ250A MCQ250B
## 1    <NA>    <NA>    <NA>    135.9    <NA>   <NA>       1    <NA>    <NA>
## 2    <NA>    <NA>    <NA>     73.6    <NA>   <NA>       2    <NA>    <NA>
## 3    <NA>    <NA>    <NA>     69.5    <NA>   <NA>    <NA>    <NA>    <NA>
## 4    <NA>    <NA>    <NA>     74.7    <NA>   <NA>       1    <NA>    <NA>
## 5       2       2       2    118.4       2      2       1       2       2
## 6       2       2       2     91.4       2      2       1       2       2
##   MCQ250C MCQ250E MCQ250F MCQ250G MCQ265 SSQ011 SSQ051 WHQ030 WHQ040 LBXRD
W
## 1    <NA>    <NA>    <NA>    <NA>   <NA>   <NA>   <NA>      3      2   13.
8
## 2    <NA>    <NA>    <NA>    <NA>   <NA>   <NA>   <NA>      3      3   13.
4
## 3    <NA>    <NA>    <NA>    <NA>   <NA>   <NA>   <NA>   <NA>   <NA>   11.
4
## 4    <NA>    <NA>    <NA>    <NA>   <NA>   <NA>   <NA>      3      1   11.
6
## 5       2       2       2       2      9      1      1      1      2   12.
1
## 6       1       2       2       2      2      1      1      1      2     1
2
##   HSD010 BPXPULS BPXML1 VIQ200 BMXBMI BPXSY1 BPXDI1 mortstat
## 1      3       1   <NA>      2  50.85   <NA>   <NA>        0
## 2      3       1    130      2  20.78    100     58       NA
## 3      2       1    130      2  18.43   <NA>   <NA>       NA
## 4      2       1    140      2  20.65   <NA>   <NA>       NA
## 5      2       1    150      2  31.26    124     88        0
## 6      3       1    160      2  25.49    128     86        0
```

Convert to numeric and only age >= 50

```
# Convert all predictor columns to numeric class
nhanes_select[ , predictors] <- apply(nhanes_select[ , predictors], 2,
                    function(x)
                     as.numeric(as.character(x)))
# Filter for age >= 50 as stated in the assignment
nhanes_df <- nhanes_select %>%
```

```
  filter(RIDAGEYR >= 50)
# Omit NAs
nhanes_df <- na.omit(nhanes_df)
# Summarize data columns
summary(nhanes_df)
```

```
##     RIDAGEYR        RIAGENDR         BPQ010          BPQ060     
##  Min.   :50.00   Min.   :1.000   Min.   :1.000   Min.   :1.000  
##  1st Qu.:59.00   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:1.000  
##  Median :66.00   Median :1.000   Median :1.000   Median :1.000  
##  Mean   :66.98   Mean   :1.495   Mean   :1.273   Mean   :1.315  
##  3rd Qu.:75.00   3rd Qu.:2.000   3rd Qu.:1.000   3rd Qu.:1.000  
##  Max.   :85.00   Max.   :2.000   Max.   :9.000   Max.   :9.000  
##      DIQ010          DIQ050          DIQ090          MCQ010     
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000  
##  1st Qu.:2.000   1st Qu.:2.000   1st Qu.:2.000   1st Qu.:2.000  
##  Median :2.000   Median :2.000   Median :2.000   Median :2.000  
##  Mean   :1.836   Mean   :1.959   Mean   :1.963   Mean   :1.908  
##  3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:2.000  
##  Max.   :3.000   Max.   :2.000   Max.   :9.000   Max.   :9.000  
##      MCQ053         MCQ160A         MCQ160B         MCQ160K     
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000  
##  1st Qu.:2.000   1st Qu.:1.000   1st Qu.:2.000   1st Qu.:2.000  
##  Median :2.000   Median :2.000   Median :2.000   Median :2.000  
##  Mean   :1.973   Mean   :1.571   Mean   :1.995   Mean   :1.933  
##  3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:2.000  
##  Max.   :9.000   Max.   :9.000   Max.   :9.000   Max.   :9.000  
##     MCQ160L         BMXWAIST        MCQ160M          MCQ220     
##  Min.   :1.000   Min.   : 61.8   Min.   :1.000   Min.   :1.000  
##  1st Qu.:2.000   1st Qu.: 91.5   1st Qu.:2.000   1st Qu.:2.000  
##  Median :2.000   Median :100.0   Median :2.000   Median :2.000  
##  Mean   :1.978   Mean   :100.8   Mean   :1.877   Mean   :1.851  
##  3rd Qu.:2.000   3rd Qu.:109.7   3rd Qu.:2.000   3rd Qu.:2.000  
##  Max.   :9.000   Max.   :157.1   Max.   :9.000   Max.   :9.000  
##     MCQ245A         MCQ250A        MCQ250B         MCQ250C        MCQ250E     
##  Min.   :1.000   Min.   :1.00   Min.   :1.000   Min.   :1.00   Min.   :1.0
00  
##  1st Qu.:1.000   1st Qu.:1.00   1st Qu.:2.000   1st Qu.:2.00   1st Qu.:2.0
00  
##  Median :2.000   Median :2.00   Median :2.000   Median :2.00   Median :2.0
00  
##  Mean   :1.629   Mean   :1.66   Mean   :1.973   Mean   :1.95   Mean   :2.1
07  
##  3rd Qu.:2.000   3rd Qu.:2.00   3rd Qu.:2.000   3rd Qu.:2.00   3rd Qu.:2.0
00  
##  Max.   :2.000   Max.   :9.00   Max.   :9.000   Max.   :9.00   Max.   :9.0
00  
##     MCQ250F        MCQ250G         MCQ265          SSQ011          SSQ051     
##  Min.   :1.00   Min.   :1.00   Min.   :1.000   Min.   :1.000   Min.   :1.0
00  
```

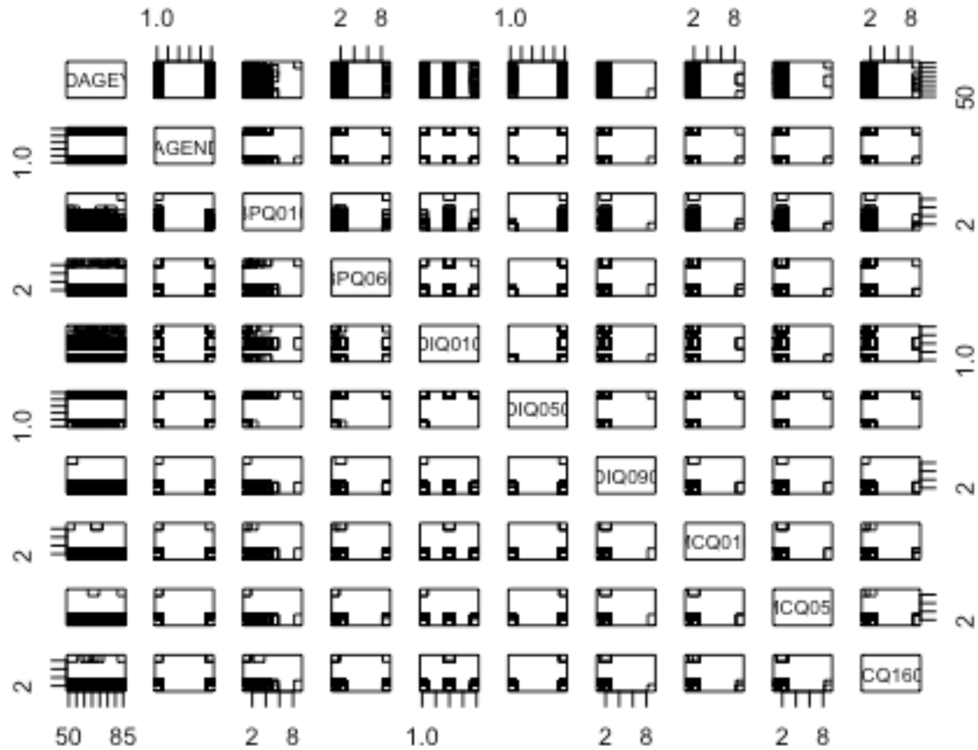
```
##  1st Qu.:2.00   1st Qu.:2.00   1st Qu.:2.000   1st Qu.:1.000   1st Qu.:1.0
00
##  Median :2.00   Median :2.00   Median :2.000   Median :1.000   Median :1.0
00
##  Mean   :2.11   Mean   :2.12   Mean   :2.065   Mean   :1.096   Mean   :1.3
57
##  3rd Qu.:2.00   3rd Qu.:2.00   3rd Qu.:2.000   3rd Qu.:1.000   3rd Qu.:2.0
00
##  Max.   :9.00   Max.   :9.00   Max.   :9.000   Max.   :9.000   Max.   :9.0
00
##      WHQ030          WHQ040          LBXRDW          HSD010     
##  Min.   :1.000   Min.   :1.000   Min.   :11.00   Min.   :1.000  
##  1st Qu.:1.000   1st Qu.:2.000   1st Qu.:12.30   1st Qu.:2.000  
##  Median :1.000   Median :2.000   Median :12.70   Median :3.000  
##  Mean   :1.836   Mean   :2.324   Mean   :12.97   Mean   :2.916  
##  3rd Qu.:3.000   3rd Qu.:3.000   3rd Qu.:13.30   3rd Qu.:4.000  
##  Max.   :9.000   Max.   :9.000   Max.   :22.40   Max.   :5.000  
##     BPXPULS          BPXML1          VIQ200          BMXBMI     
##  Min.   :1.000   Min.   :120.0   Min.   :1.000   Min.   :14.70  
##  1st Qu.:1.000   1st Qu.:150.0   1st Qu.:2.000   1st Qu.:24.75  
##  Median :1.000   Median :160.0   Median :2.000   Median :27.70  
##  Mean   :1.096   Mean   :166.7   Mean   :1.834   Mean   :28.53  
##  3rd Qu.:1.000   3rd Qu.:180.0   3rd Qu.:2.000   3rd Qu.:31.57  
##  Max.   :2.000   Max.   :260.0   Max.   :9.000   Max.   :56.03  
##      BPXSY1          BPXDI1          mortstat     
##  Min.   : 80.0   Min.   :  0.00   Min.   :0.0000  
##  1st Qu.:122.0   1st Qu.: 64.00   1st Qu.:0.0000  
##  Median :134.0   Median : 70.00   Median :0.0000  
##  Mean   :137.5   Mean   : 69.86   Mean   :0.2153  
##  3rd Qu.:150.0   3rd Qu.: 78.00   3rd Qu.:0.0000  
##  Max.   :228.0   Max.   :120.00   Max.   :1.0000  
```

Visually examine some of the possible predictors

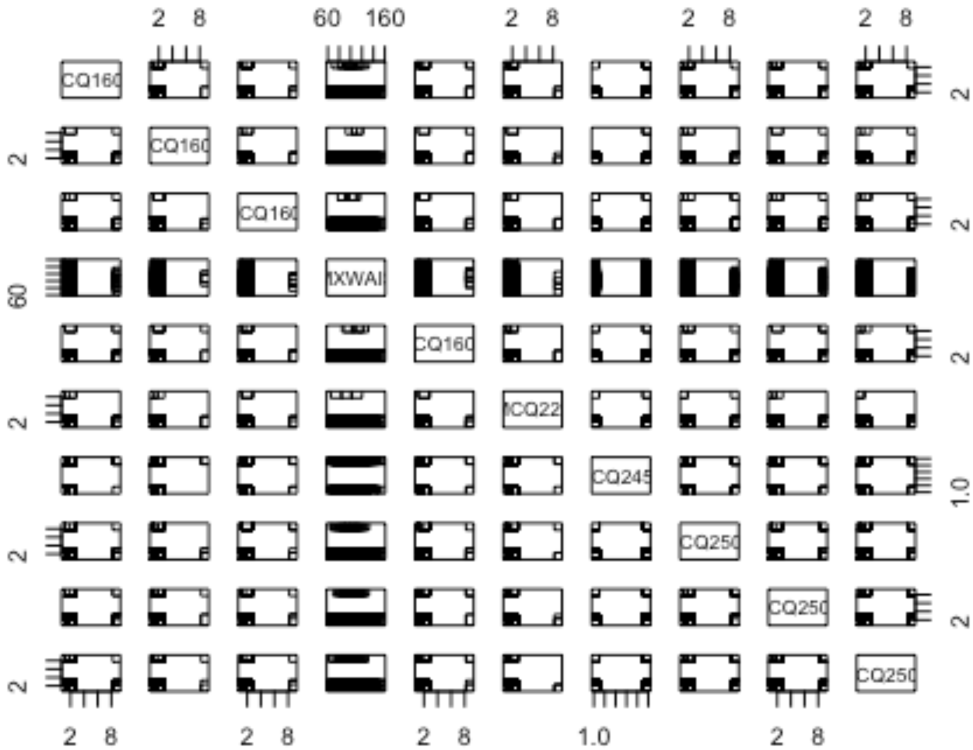
```
prop.table(table(nhanes_df$mortstat))*100

## 
##       0       1 
## 78.4692 21.5308

prop.table(table(nhanes_df$mortstat, nhanes_df$RIAGENDR))*100

##    
##             1         2
##   0 38.083385 40.385812
##   1 12.445551  9.085252
```

Identifying highly correlated pairs for possible confounding: BPXML1 with BPXSY1 and BPXDI1 (both could be measures of blood pressure) as well as BPXSY1 and BPXDI1.

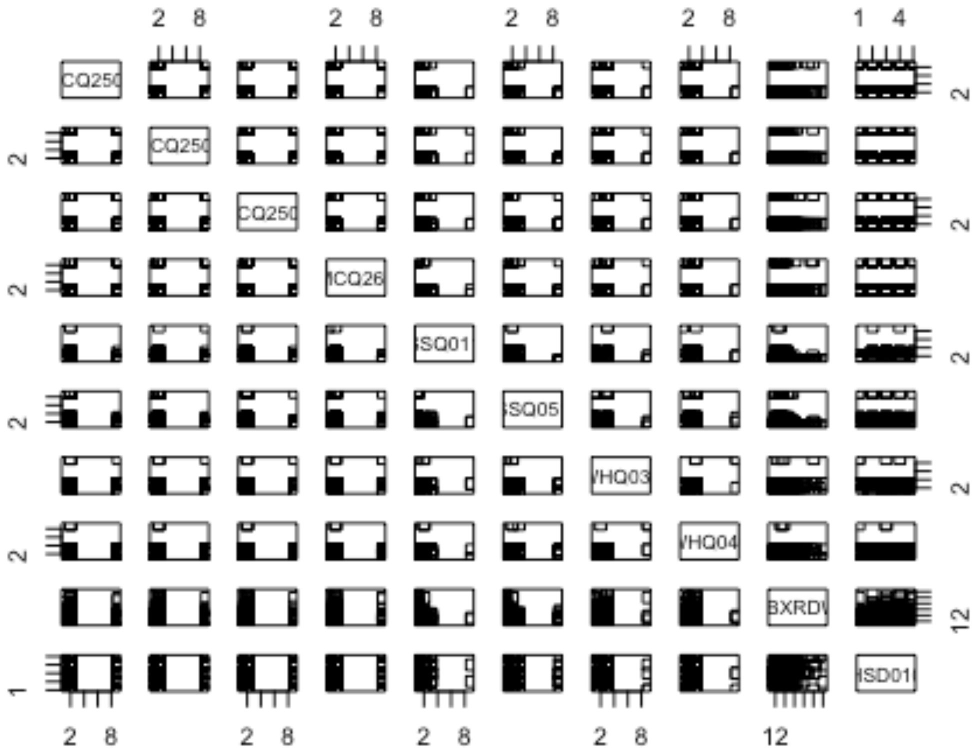
```
pairs(nhanes_df[,1:10], pch=0.5)
```



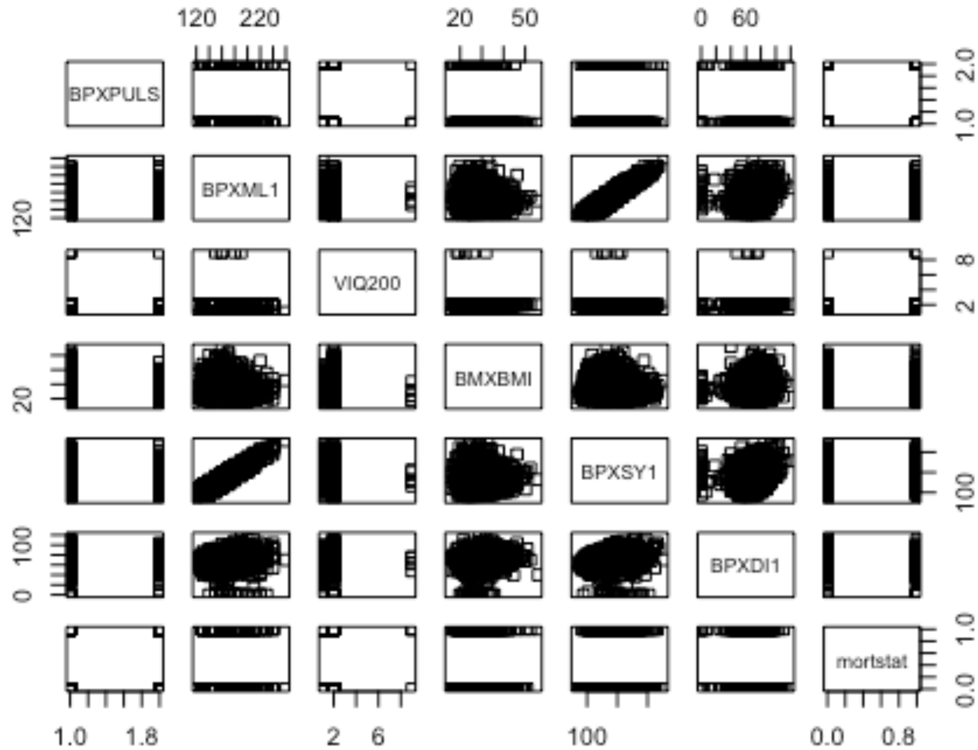
```
pairs(nhanes_df[,11:20], pch=0.5)
```




```
pairs(nhanes_df[,21:30], pch=0.5)
```




```
pairs(nhanes_df[,31:37], pch=0.5)
```



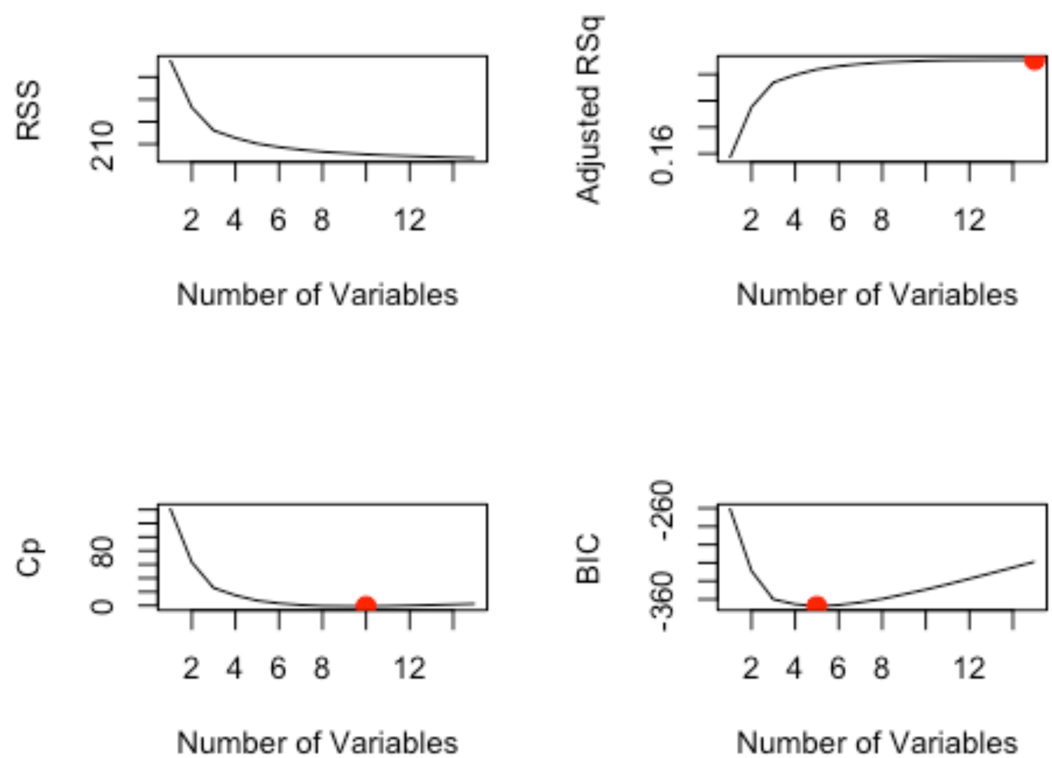
2. Modeling

Subset Selection

Because of the large number of potential variables that can be considered as predictors for mortality status, I first ran best subset selection fitting up to a 15-variable model in the linear space to aid in prediction accuracy and model interpretability for the first few models. Subset selection was chosen to identify a subset of the p predictors that were identified as being related to the response. After subset selection was run, Cp, BIC, and R^2 statistics were analyzed in order to select models for use in the first few models.

```
library(leaps)
# Fitting up to a 15 variable model
regfit.full = regsubsets(mortstat~., nhanes_df, nvmax=15)
reg.summary <- summary(regfit.full)

par(mfrow=c(2,2))
plot(reg.summary$rss ,xlab="Number of Variables ",ylab="RSS",
type="l")
plot(reg.summary$adjr2 ,xlab="Number of Variables ",
```

```
ylab="Adjusted RSq",type="l")
points(15,reg.summary$adjr2[15], col="red",cex=2,pch=20)
plot(reg.summary$cp ,xlab="Number of Variables ",
ylab="Cp",type="l")
points(10,reg.summary$cp[10], col="red",cex=2,pch=20)
plot(reg.summary$bic ,xlab="Number of Variables ",
ylab="BIC",type="l")
points(5,reg.summary$bic[5], col="red",cex=2,pch=20)
```



Cp predicts the 10 variable model and BIC predicts the 5-variable model. Those models are as follows using our best subset selection: 5: RIDAGEYR, RIAGENDR, MCQ160K, LBXRDW, HSD010 10: RIDAGEYR, RIAGENDR, BPQ060, DIQ090, MCQ160A, MCQ160K, SSQ011, LBXRDW, HSD010, BPXPULS

```
reg.summary

## Subset selection object
## Call: regsubsets.formula(mortstat ~ ., nhanes_df, nvmax = 15)
## 36 Variables  (and intercept)
##          Forced in Forced out
## RIDAGEYR     FALSE      FALSE
## RIAGENDR     FALSE      FALSE
## BPQ010       FALSE      FALSE
```

```
## BPQ060       FALSE      FALSE
## DIQ010       FALSE      FALSE
## DIQ050       FALSE      FALSE
## DIQ090       FALSE      FALSE
## MCQ010       FALSE      FALSE
## MCQ053       FALSE      FALSE
## MCQ160A      FALSE      FALSE
## MCQ160B      FALSE      FALSE
## MCQ160K      FALSE      FALSE
## MCQ160L      FALSE      FALSE
## BMXWAIST     FALSE      FALSE
## MCQ160M      FALSE      FALSE
## MCQ220       FALSE      FALSE
## MCQ245A      FALSE      FALSE
## MCQ250A      FALSE      FALSE
## MCQ250B      FALSE      FALSE
## MCQ250C      FALSE      FALSE
## MCQ250E      FALSE      FALSE
## MCQ250F      FALSE      FALSE
## MCQ250G      FALSE      FALSE
## MCQ265       FALSE      FALSE
## SSQ011       FALSE      FALSE
## SSQ051       FALSE      FALSE
## WHQ030       FALSE      FALSE
## WHQ040       FALSE      FALSE
## LBXRDW       FALSE      FALSE
## HSD010       FALSE      FALSE
## BPXPULS      FALSE      FALSE
## BPXML1       FALSE      FALSE
## VIQ200       FALSE      FALSE
## BMXBMI       FALSE      FALSE
## BPXSY1       FALSE      FALSE
## BPXDI1       FALSE      FALSE
## 1 subsets of each size up to 15
## Selection Algorithm: exhaustive
##           RIDAGEYR RIAGENDR BPQ010 BPQ060 DIQ010 DIQ050 DIQ090 MCQ010 MCQ0
53
## 1  ( 1 )  "*"      " "      " "    " "    " "    " "    " "    " "    " "
## 2  ( 1 )  "*"      " "      " "    " "    " "    " "    " "    " "    " "
## 3  ( 1 )  "*"      " "      " "    " "    " "    " "    " "    " "    " "
## 4  ( 1 )  "*"      "*"      " "    " "    " "    " "    " "    " "    " "
## 5  ( 1 )  "*"      "*"      " "    " "    " "    " "    " "    " "    " "
## 6  ( 1 )  "*"      "*"      " "    " "    " "    "*"    " "    " "    " "
## 7  ( 1 )  "*"      "*"      " "    "*"    " "    "*"    " "    " "    " "
## 8  ( 1 )  "*"      "*"      " "    "*"    " "    "*"    " "    " "    " "
## 9  ( 1 )  "*"      "*"      " "    "*"    " "    "*"    " "    " "    " "
## 10  ( 1 ) "*"      "*"      " "    "*"    " "    "*"    " "    " "    " "
## 11  ( 1 ) "*"      "*"      " "    "*"    " "    "*"    " "    " "    " "
## 12  ( 1 ) "*"      "*"      " "    "*"    " "    "*"    " "    "*"    " "
## 13  ( 1 ) "*"      "*"      " "    "*"    "*"    "*"    " "    "*"    " "
```

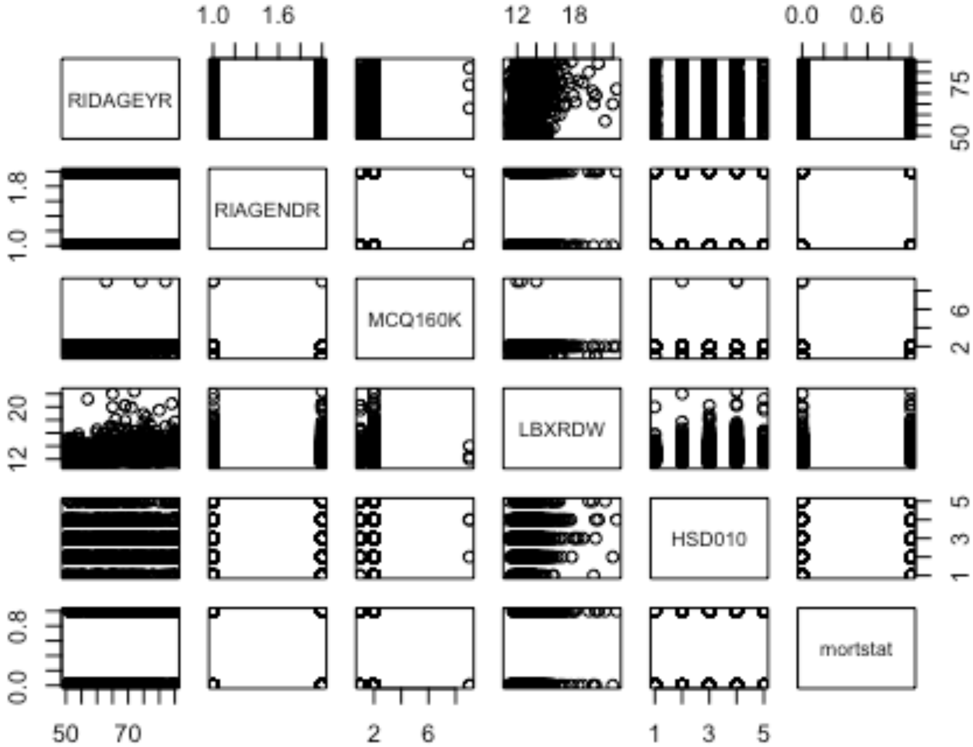
```
## 14  ( 1 ) "*"      "*"      " "     "*"     "*"     "*"     " "     "*"     " "
## 15  ( 1 ) "*"      "*"      " "     "*"     "*"     "*"     " "     "*"     " "
##           MCQ160A MCQ160B MCQ160K MCQ160L BMXWAIST MCQ160M MCQ220 MCQ245A
## 1  ( 1 )  " "     " "     " "     " "     " "      " "     " "    " "
## 2  ( 1 )  " "     " "     " "     " "     " "      " "     " "    " "
## 3  ( 1 )  " "     " "     " "     " "     " "      " "     " "    " "
## 4  ( 1 )  " "     " "     " "     " "     " "      " "     " "    " "
## 5  ( 1 )  " "     " "     "*"     " "     " "      " "     " "    " "
## 6  ( 1 )  " "     " "     "*"     " "     " "      " "     " "    " "
## 7  ( 1 )  " "     " "     "*"     " "     " "      " "     " "    " "
## 8  ( 1 )  " "     " "     "*"     " "     " "      " "     " "    " "
## 9  ( 1 )  " "     " "     "*"     " "     " "      " "     " "    " "
## 10  ( 1 ) "*"     " "     "*"     " "     " "      " "     " "    " "
## 11  ( 1 ) "*"     " "     "*"     " "     " "      " "     " "    " "
## 12  ( 1 ) "*"     " "     "*"     " "     " "      " "     " "    " "
## 13  ( 1 ) "*"     " "     "*"     " "     " "      " "     " "    " "
## 14  ( 1 ) "*"     " "     "*"     " "     " "      " "     " "    " "
## 15  ( 1 ) "*"     " "     "*"     " "     " "      " "     " "    " "
##           MCQ250A MCQ250B MCQ250C MCQ250E MCQ250F MCQ250G MCQ265 SSQ011 SS
Q051
## 1  ( 1 )  " "     " "     " "     " "     " "     " "     " "    " "    "
"
## 2  ( 1 )  " "     " "     " "     " "     " "     " "     " "    " "    "
"
## 3  ( 1 )  " "     " "     " "     " "     " "     " "     " "    " "    "
"
## 4  ( 1 )  " "     " "     " "     " "     " "     " "     " "    " "    "
"
## 5  ( 1 )  " "     " "     " "     " "     " "     " "     " "    " "    "
"
## 6  ( 1 )  " "     " "     " "     " "     " "     " "     " "    " "    "
"
## 7  ( 1 )  " "     " "     " "     " "     " "     " "     " "    " "    "
"
## 8  ( 1 )  " "     " "     " "     " "     " "     " "     " "    " "    "
"
## 9  ( 1 )  " "     " "     " "     " "     " "     " "     " "    "*"    "
"
## 10  ( 1 ) " "     " "     " "     " "     " "     " "     " "    "*"    "
"
## 11  ( 1 ) " "     " "     " "     " "     " "     " "     " "    "*"    "
"
## 12  ( 1 ) " "     " "     " "     " "     " "     " "     " "    "*"    "
"
## 13  ( 1 ) " "     " "     " "     " "     " "     " "     " "    "*"    "
"
## 14  ( 1 ) " "     " "     " "     " "     " "     " "     " "    "*"    "
"
## 15  ( 1 ) "*"     " "     " "     " "     " "     " "     " "    "*"    "
"
```

```
##            WHQ030 WHQ040 LBXRDW HSD010 BPXPULS BPXML1 VIQ200 BMXBMI BPXSY1
## 1  ( 1 )  " "    " "    " "    " "    " "     " "    " "    " "    " "   
## 2  ( 1 )  " "    " "    "*"    " "    " "     " "    " "    " "    " "   
## 3  ( 1 )  " "    " "    "*"    "*"    " "     " "    " "    " "    " "   
## 4  ( 1 )  " "    " "    "*"    "*"    " "     " "    " "    " "    " "   
## 5  ( 1 )  " "    " "    "*"    "*"    " "     " "    " "    " "    " "   
## 6  ( 1 )  " "    " "    "*"    "*"    " "     " "    " "    " "    " "   
## 7  ( 1 )  " "    " "    "*"    "*"    " "     " "    " "    " "    " "   
## 8  ( 1 )  " "    " "    "*"    "*"    "*"     " "    " "    " "    " "   
## 9  ( 1 )  " "    " "    "*"    "*"    "*"     " "    " "    " "    " "   
## 10  ( 1 ) " "    " "    "*"    "*"    "*"     " "    " "    " "    " "   
## 11  ( 1 ) " "    " "    "*"    "*"    "*"     " "    "*"    " "    " "   
## 12  ( 1 ) " "    " "    "*"    "*"    "*"     " "    "*"    " "    " "   
## 13  ( 1 ) " "    " "    "*"    "*"    "*"     " "    "*"    " "    " "   
## 14  ( 1 ) " "    " "    "*"    "*"    "*"     " "    "*"    "*"    " "   
## 15  ( 1 ) " "    " "    "*"    "*"    "*"     " "    "*"    "*"    " "   
##            BPXDI1
## 1  ( 1 )  " "   
## 2  ( 1 )  " "   
## 3  ( 1 )  " "   
## 4  ( 1 )  " "   
## 5  ( 1 )  " "   
## 6  ( 1 )  " "   
## 7  ( 1 )  " "   
## 8  ( 1 )  " "   
## 9  ( 1 )  " "   
## 10  ( 1 ) " "   
## 11  ( 1 ) " "   
## 12  ( 1 ) " "   
## 13  ( 1 ) " "   
## 14  ( 1 ) " "   
## 15  ( 1 ) " "   
```

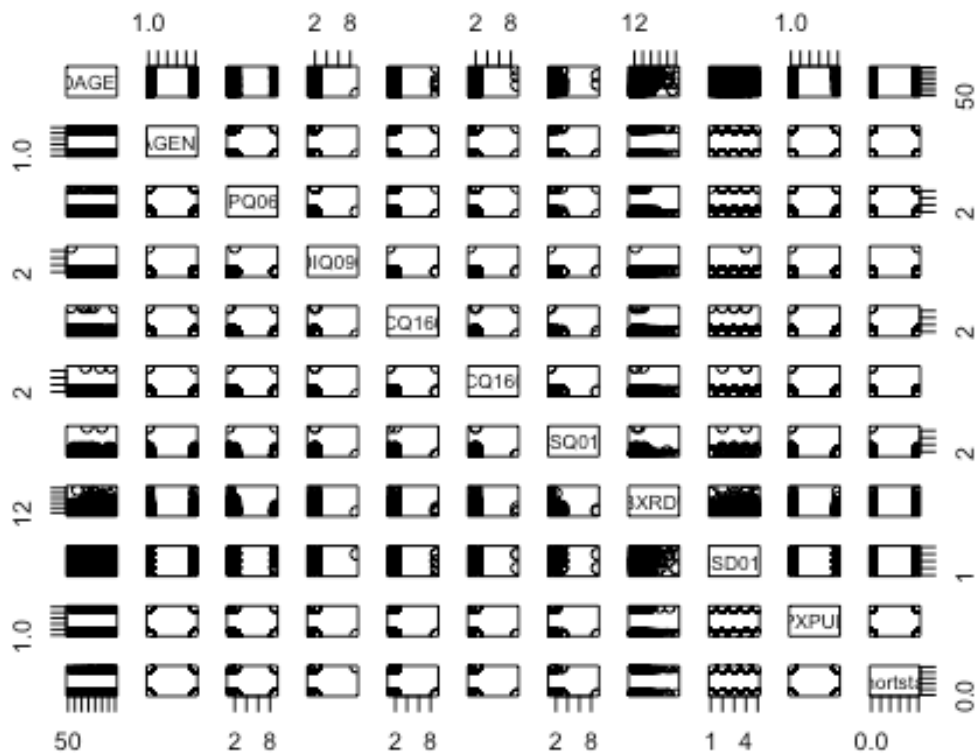
Examine possible correlation between the 5 and 10 selected variables

Visually, no major correlation trends between variables stand out in the five or ten variable selected models.

```
five_var <- c("RIDAGEYR", "RIAGENDR", "MCQ160K", "LBXRDW", "HSD010")
ten_var <- c("RIDAGEYR", "RIAGENDR", "BPQ060", "DIQ090", "MCQ160A", "MCQ160K"
, "SSQ011", "LBXRDW", "HSD010", "BPXPULS")
five_df <- nhanes_df %>%
  select(five_var, mortstat)

## Note: Using an external vector in selections is ambiguous.
## ℹ Use `all_of(five_var)` instead of `five_var` to silence this message.
## ℹ See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

pairs(five_df)
```




```
ten_df <- nhanes_df %>%
  select(ten_var, mortstat)

## Note: Using an external vector in selections is ambiguous.
## ℹ Use `all_of(ten_var)` instead of `ten_var` to silence this message.
## ℹ See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

pairs(ten_df)
```



Classification Methods: Logistic Regression

Model 1: Logistic Regression using 5 predictors in best subset selection

In each of the logistic regression classification methods (5 and 10-variable), I first separated the data into two equal test and train samples using the sample() command in R. Once the test and training data was complete, I ran a proptable() on mortstatus to understand if the training and testing data had similar mortality status proportions. For both models, I used the testing data to run a glm with family binomial to create the model. I then used the predict() function in R to use the model to predict on the test data set. I created a table to compare the predicted probabilities to the actual mortality test data. For logistic regression models, I finally compared models for preference using AIC.

```
attach(nhanes_df)
model <- glm(mortstat ~ RIAGENDR + RIDAGEYR + MCQ160K + LBXRDW + HSD010, data
=nhanes_df, family=binomial)
summary(model)

##
## Call:
## glm(formula = mortstat ~ RIAGENDR + RIDAGEYR + MCQ160K + LBXRDW +
```

```
##     HSD010, family = binomial, data = nhanes_df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0581  -0.6348  -0.3559  -0.1716   2.8434
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept) -13.499423   1.125706 -11.992  < 2e-16 ***
## RIAGENDR     -0.540351   0.143050  -3.777 0.000159 ***
## RIDAGEYR      0.110753   0.007988  13.865  < 2e-16 ***
## MCQ160K      -0.677409   0.233626  -2.900 0.003737 **
## LBXRDW        0.395899   0.060877   6.503 7.86e-11 ***
## HSD010        0.436580   0.071078   6.142 8.14e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1674.2  on 1606  degrees of freedom
## Residual deviance: 1275.2  on 1601  degrees of freedom
## AIC: 1287.2
##
## Number of Fisher Scoring iterations: 5
```

Creating training and testing dataset and making sure percentages of mortality split are similar to original dataset.

```
set.seed(1)
prop.table(table(nhanes_df$mortstat))*100

##
##       0       1
## 78.4692 21.5308

train = sample(1607, 1607/2)
train_data = nhanes_df[train,]
prop.table(table(train_data$mortstat))*100

##
##        0        1
## 79.32752 20.67248

test_data = nhanes_df[-train,]
prop.table(table(test_data$mortstat))*100

##
##        0        1
## 77.61194 22.38806
```

Running logistic regression

```
model_1 <- glm(mortstat ~ RIAGENDR + RIDAGEYR + MCQ160K + LBXRDW + HSD010, da
ta=train_data, family=binomial)
summary(model_1)

##
## Call:
## glm(formula = mortstat ~ RIAGENDR + RIDAGEYR + MCQ160K + LBXRDW +
##     HSD010, family = binomial, data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8811  -0.6174  -0.3499  -0.1707   2.6962
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -13.02620    1.59991  -8.142 3.89e-16 ***
## RIAGENDR     -0.63202    0.20684  -3.056  0.00225 **
## RIDAGEYR      0.11001    0.01158   9.503  < 2e-16 ***
## MCQ160K      -0.77659    0.35659  -2.178  0.02942 *
## LBXRDW        0.37667    0.08244   4.569 4.90e-06 ***
## HSD010        0.47465    0.10150   4.676 2.92e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 818.39  on 802  degrees of freedom
## Residual deviance: 621.92  on 797  degrees of freedom
## AIC: 633.92
##
## Number of Fisher Scoring iterations: 5

log_probs <- predict(model_1, test_data, type="response")
test_probs <-  (log_probs - test_data$mortstat) ^ 2
#mean((log_probs - test_data$mortstat) ^ 2)
```

Find sensitivity and specificity of model.

```
log_pred =rep("0",804)
log_pred[log_probs >.5]="1"
table(log_pred, test_data$mortstat)

##
## log_pred   0   1
##        0 586 123
##        1  38  57

mean(log_pred == test_data$mortstat)

## [1] 0.7997512
```

This model is correct predicting 79.98% of the time.

```
# Sensitivity - Detecting those who are going to have a mort status of 1 corr
ectly
56 / (57 + 38)

## [1] 0.5894737

# Specificity - Detecting those who are not going to have a mort status of 0
correctly
586 / (568+123)

## [1] 0.8480463
```

Model 2: Logistic Regression using 10 predictors in best subset selection

Running logistic regression

```
set.seed(1)
model_2 <- glm(mortstat ~ RIAGENDR + RIDAGEYR + BPQ060 + DIQ090 + MCQ160A + M
CQ160K + SSQ011 + LBXRDW + HSD010 + BPXPULS, data=train_data, family=binomial
)
summary(model_2)

## 
## Call:
## glm(formula = mortstat ~ RIAGENDR + RIDAGEYR + BPQ060 + DIQ090 + 
##     MCQ160A + MCQ160K + SSQ011 + LBXRDW + HSD010 + BPXPULS, family = binom
ial, 
##     data = train_data)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -1.9265  -0.6107  -0.3480  -0.1690   2.7552  
## 
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)    
## (Intercept) -13.09875    1.83984  -7.120 1.08e-12 ***
## RIAGENDR     -0.64567    0.21153  -3.052  0.00227 ** 
## RIDAGEYR      0.10782    0.01199   8.989  < 2e-16 ***
## BPQ060        0.04084    0.06910   0.591  0.55447    
## DIQ090        0.19719    0.26911   0.733  0.46372    
## MCQ160A      -0.18901    0.19870  -0.951  0.34148    
## MCQ160K      -0.73053    0.35463  -2.060  0.03940 *  
## SSQ011       -0.15273    0.28940  -0.528  0.59768    
## LBXRDW        0.38034    0.08258   4.606 4.11e-06 ***
## HSD010        0.45631    0.10347   4.410 1.03e-05 ***
## BPXPULS       0.15404    0.30528   0.505  0.61384    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 818.39  on 802  degrees of freedom
## Residual deviance: 619.73  on 792  degrees of freedom
## AIC: 641.73
##
## Number of Fisher Scoring iterations: 5
```

```
log_probs_2 <- predict(model_2, test_data, type="response")
#mean((log_probs - test_data$mortstat) ^ 2)
```

Find sensitivity and specificity of model.

```
log_pred_2 =rep("0",804)
log_pred_2[log_probs_2 >.5]="1"
table(log_pred_2, test_data$mortstat)
```

```
##
## log_pred_2   0   1
##          0 589 119
##          1  35  61
```

```
mean(log_pred_2 == test_data$mortstat)
```

```
## [1] 0.8084577
```

This model is correct predicting 80.84% of the time.

```
# Sensitivity - Detecting those who are going to have a mort status of 1 corr
ectly
61 / (61+35)
```

```
## [1] 0.6354167
```

```
# Specificity - Detecting those who are not going to have a mort status of 0
correctly
589 / (589+119)
```

```
## [1] 0.8319209
```

The testing accuracy in this specific logistic regression with 10 variables (model 2) is slightly better than the model 1 with 5 variables, but not a lot.

AIC on Logistic Regression Models

```
AIC(model_1, model_2)
```

```
##         df      AIC
## model_1  6 633.9173
## model_2 11 641.7271
```

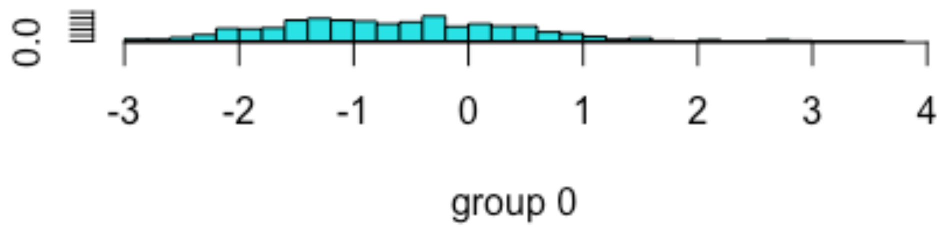
Using AIC, we see that model 1 with 5 variables is the preferred model for logistic regression between the two models.

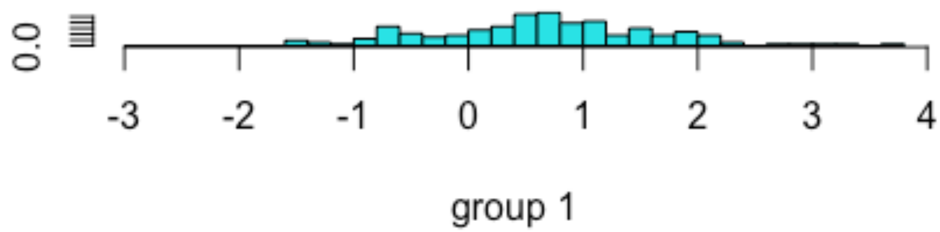
Classification Methods: Quadratic and Linear Discriminant Analysis

In models 3 – 5, I used the lda() or qda() commands on the training data to get the original model. From there, I predicted the models on the test data using the predict() command in R and compared the classes of data predicted by the model against the actual mortality status of the test data.

Model 3: LDA - 5 variables

```
library(MASS)

##
## Attaching package: 'MASS'

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

set.seed(1)
model_3 <- lda(mortstat ~ RIAGENDR + RIDAGEYR + MCQ160K + LBXRDW + HSD010, da
ta=train_data)
model_3

## Call:
## lda(mortstat ~ RIAGENDR + RIDAGEYR + MCQ160K + LBXRDW + HSD010,
##     data = train_data)
##
## Prior probabilities of groups:
##         0         1
## 0.7932752 0.2067248
##
## Group means:
##   RIAGENDR RIDAGEYR  MCQ160K   LBXRDW   HSD010
## 0 1.529042 64.93564 1.968603 12.81727 2.800628
## 1 1.379518 74.33133 1.891566 13.60482 3.289157
##
## Coefficients of linear discriminants:
##                  LD1
## RIAGENDR -0.45938214
## RIDAGEYR  0.08080262
## MCQ160K  -0.43162366
## LBXRDW    0.38052725
## HSD010    0.33344846

plot(model_3)
```





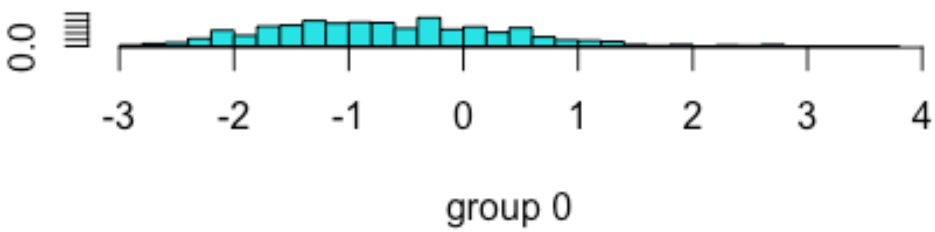
This LDA output tells us that 77.5% of the training observations correspond to 0 mortstat (alive) and 22.4% to 1 mortstat (dead). We'll now use it on our prediction data.

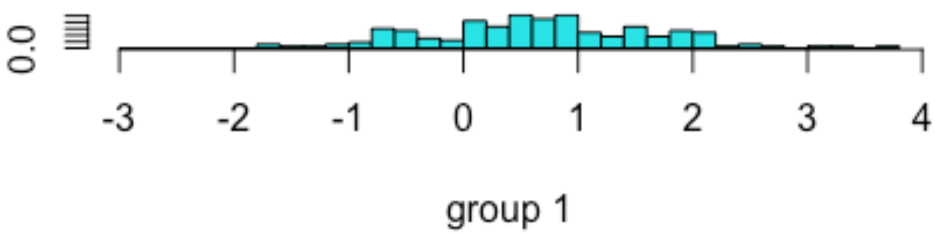
```
lda_pred_5 <- predict(model_3, test_data)
lda_class_5 <- lda_pred_5$class
test_mort_status <- test_data$mortstat
mean(lda_class_5== test_mort_status)

## [1] 0.8022388

mean(lda_class_5!= test_mort_status)

## [1] 0.1977612

table(lda_class_5, test_mort_status)

##              test_mort_status
## lda_class_5   0   1
##             0 586 121
##             1  38  59

(59) / (59+38)

## [1] 0.6082474
```

```
(586) / (586+121)

## [1] 0.8288543
```

Model 4: LDA - 10 variables

```
library(MASS)
model_4 <- lda(mortstat ~ RIAGENDR + RIDAGEYR + BPQ060 + DIQ090 + MCQ160A + M
CQ160K + SSQ011 + LBXRDW + HSD010 + BPXPULS, data=train_data)
model_4

## Call:
## lda(mortstat ~ RIAGENDR + RIDAGEYR + BPQ060 + DIQ090 + MCQ160A + 
##     MCQ160K + SSQ011 + LBXRDW + HSD010 + BPXPULS, data = train_data)
## 
## Prior probabilities of groups:
##         0         1 
## 0.7932752 0.2067248 
## 
## Group means:
##   RIAGENDR RIDAGEYR   BPQ060   DIQ090  MCQ160A  MCQ160K   SSQ011   LBXRDW
## 0 1.529042 64.93564 1.309262 1.976452 1.594976 1.968603 1.094192 12.81727
## 1 1.379518 74.33133 1.415663 1.975904 1.469880 1.891566 1.072289 13.60482
##     HSD010  BPXPULS
## 0 2.800628 1.072214
## 1 3.289157 1.180723
## 
## Coefficients of linear discriminants:
##                  LD1
## RIAGENDR -0.44586997
## RIDAGEYR  0.07725378
## BPQ060    0.06119776
## DIQ090    0.14880666
## MCQ160A  -0.08624917
## MCQ160K  -0.44550853
## SSQ011   -0.20534965
## LBXRDW    0.38287327
## HSD010    0.31911646
## BPXPULS   0.30391641

plot(model_4)
```






```
set.seed(1)
lda_pred_10 <- predict(model_4, test_data)
lda_class_10 <- lda_pred_10$class
test_mort_status <- test_data$mortstat
mean(lda_class_10== test_mort_status)

## [1] 0.8034826

mean(lda_class_10!= test_mort_status)

## [1] 0.1965174

table(lda_class_10, test_mort_status)

##             test_mort_status
## lda_class_10   0   1
##            0 586 120
##            1  38  60

60 / (60+38)

## [1] 0.6122449

586 / (586 + 120)
```

```
## [1] 0.8300283
```

The results that we got from LDA were similar to that from Logistic Regression.

Model 5 QDA - 10 variables

```
set.seed(1)
model_5 <- qda(mortstat ~ RIAGENDR + RIDAGEYR + BPQ060 + DIQ090 + MCQ160A + M
CQ160K + SSQ011 + LBXRDW + HSD010 + BPXPULS, data=train_data)
model_5

## Call:
## qda(mortstat ~ RIAGENDR + RIDAGEYR + BPQ060 + DIQ090 + MCQ160A +
##     MCQ160K + SSQ011 + LBXRDW + HSD010 + BPXPULS, data = train_data)
##
## Prior probabilities of groups:
##         0         1
## 0.7932752 0.2067248
##
## Group means:
##   RIAGENDR RIDAGEYR   BPQ060   DIQ090  MCQ160A  MCQ160K   SSQ011   LBXRDW
## 0 1.529042 64.93564 1.309262 1.976452 1.594976 1.968603 1.094192 12.81727
## 1 1.379518 74.33133 1.415663 1.975904 1.469880 1.891566 1.072289 13.60482
##     HSD010  BPXPULS
## 0 2.800628 1.072214
## 1 3.289157 1.180723

qda_pred_10 <- predict(model_5, test_data)
qda_class_10 <- qda_pred_10$class
test_mort_status <- test_data$mortstat
mean(qda_class_10== test_mort_status)

## [1] 0.7674129

mean(qda_class_10!= test_mort_status)

## [1] 0.2325871
```

The test accuracy rate is 76.74%. So far, this is the lower testing accuracy we've seen in this dataset.

```
table(qda_class_10, test_mort_status)

##             test_mort_status
## qda_class_10   0   1
##            0 559 122
##            1  65  58

58/(58+65)

## [1] 0.4715447

559/(559+122)
```

```
## [1] 0.8208517
```

This model appears to have the highest specificity, but lowest sensitivity.

Tree-Based Methods: Random Forest

In models 6 and 7, I used the randomForest package in R, built under my current R version of 4.1.2 (randomForest 4.7-1). I used the randomForest() command to run a random forest on the 36 variables present in the training dataset. I used importance() and varImpPlot() functions on the models to understand the variables deemed most important in the modeling. From there, I used the predict() function to do prediction on the test data. I then compared the prediction from random forest methods to the actual test mortality status data.

Model 6: RF with all 36 predictor variables = bagging.

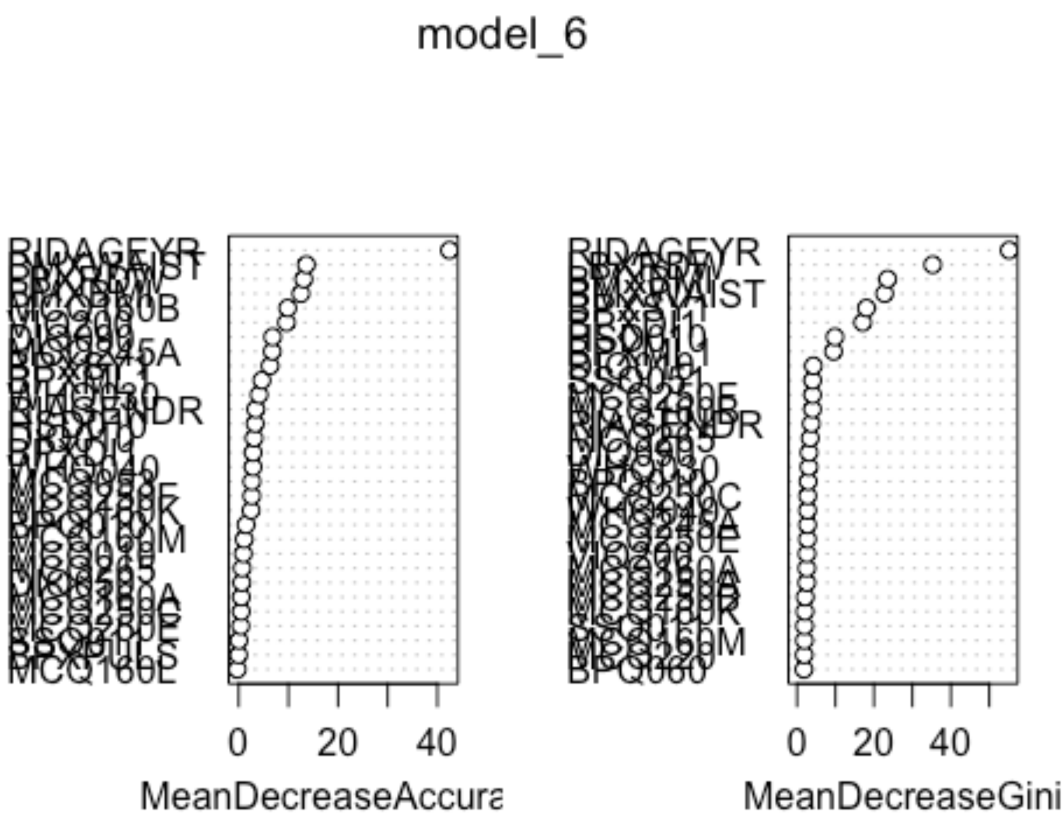
```
library(randomForest)

## Warning: package 'randomForest' was built under R version 4.1.2

## randomForest 4.7-1

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

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

## The following object is masked from 'package:ggplot2':
##
##     margin

set.seed(1)
model_6 =randomForest(as.factor(mortstat) ~., data=train_data, mtry=36,import
ance =TRUE)
model_6

##
## Call:
##  randomForest(formula = as.factor(mortstat) ~ ., data = train_data,      m
try = 36, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 36
##
##         OOB estimate of  error rate: 20.67%
## Confusion matrix:
##     0  1 class.error
```

```
## 0 582 55  0.08634223
## 1 111 55  0.66867470
```

```
importance(model_6)
```

```
##                    0          1 MeanDecreaseAccuracy MeanDecreaseGini
## RIDAGEYR 33.74249570 27.1460674          42.43806705        55.181874
## RIAGENDR  3.25043392  1.3983346           3.45043818         3.707251
## BPQ010    2.22776490 -0.5937684           1.48412139         2.891952
## BPQ060   -3.51126897  2.3111330          -1.12269298         1.788888
## DIQ010    1.73414777  3.9382384           3.10046435         4.251602
## DIQ050    1.90410965 -2.0114196           0.62368445         1.354015
## DIQ090    6.58480684  1.9636690           6.84431108         3.118907
## MCQ010    2.13440371 -1.2911928           0.92402273         1.704373
## MCQ053    1.40941869  2.3982068           2.65379372         1.617083
## MCQ160A   1.43922264 -1.3784505           0.52867834         2.545484
## MCQ160B   9.83532843  2.0281806           9.86997562         4.015880
## MCQ160K   2.18744019  1.2461160           2.48950448         2.040418
## MCQ160L  -0.90633192  1.1973668          -0.28088322         1.128233
## BMXWAIST 14.42128835 -2.3480447          13.63186625        22.776546
## MCQ160M   1.76643277 -0.3830419           1.10824891         1.928222
## MCQ220   -0.68062404 -1.1129521          -1.18033681         1.860957
## MCQ245A   9.28174733 -6.7435241           6.76006989         2.740696
## MCQ250A   1.64235113 -2.9828790          -0.35819537         2.537334
## MCQ250B  -1.16439991 -2.3479148          -2.12852522         2.121831
## MCQ250C   0.37194138  0.4878726           0.51398075         2.781672
## MCQ250E   1.49941559 -1.8135998           0.41736447         2.709222
## MCQ250F   0.54307004  4.1408788           2.64101465         4.045061
## MCQ250G  -3.86615757  1.2502165          -2.92663163         1.547404
## MCQ265   -0.02211148  1.5127132           0.70904096         3.433817
## SSQ011   -0.39266572  0.7234931           0.03848513         1.955175
## SSQ051   -1.24434744  1.2728775          -0.30362942         4.062478
## WHQ030    6.32809830 -3.3316081           4.10795994         3.101702
## WHQ040    3.82563970 -1.8591665           2.88240815         2.745353
## LBXRDW    7.13622439 13.4031076          13.10936580        35.287691
## HSD010    2.66424664  1.9117351           3.32552802         9.861563
## BPXPULS  -0.20433502  0.3007218          -0.01348982         1.623682
## BPXML1    6.21732655 -2.4711613           4.68377970         9.492746
## VIQ200   11.35952467 -2.7304346           9.63170240         2.585662
## BMXBMI   13.56295444 -2.0334029          12.53607010        23.563439
## BPXSY1    9.10298565 -3.4316857           6.30545971        18.071895
## BPXDI1    8.87986382 -7.8088690           2.89565563        17.087297
```

```
varImpPlot(model_6)
```




```
yhat_bag =predict(model_6,newdata=test_data)
table(yhat_bag, test_mort_status)

##          test_mort_status
## yhat_bag   0   1
##         0 578 115
##         1  46  65

(65) / (65+46)

## [1] 0.5855856

(578) / (578 + 115)

## [1] 0.8340548
```

Model 7: RF without bagging

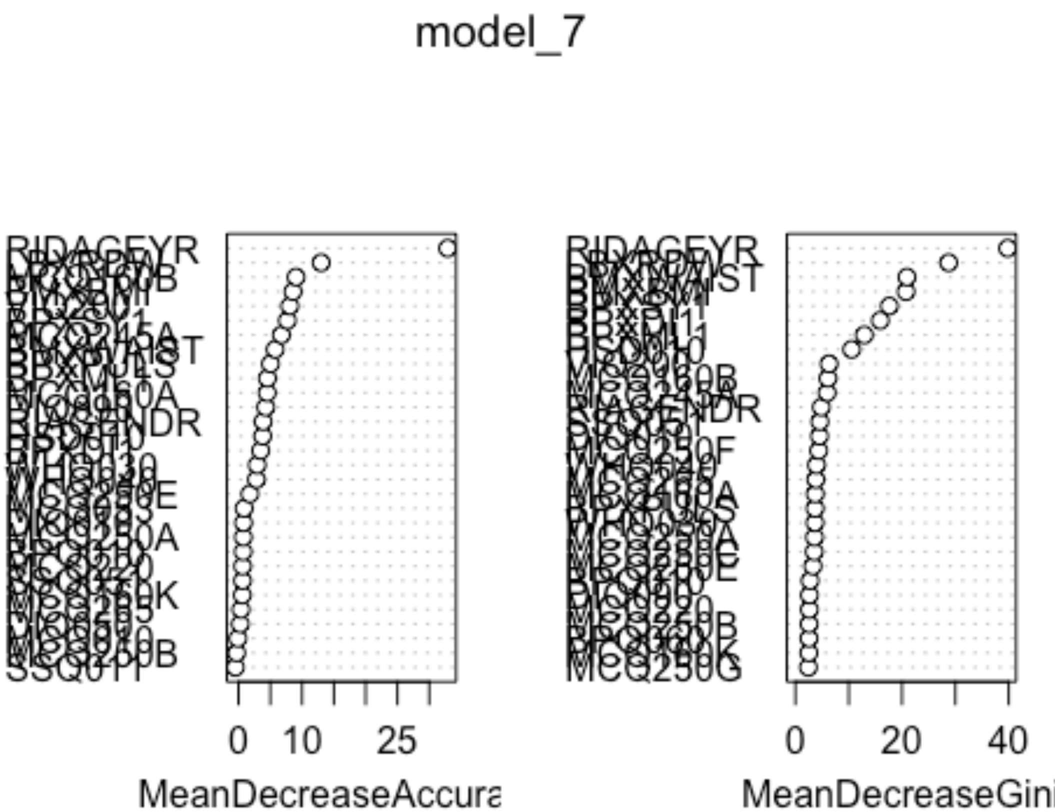
```
set.seed(1)
model_7 =randomForest(as.factor(mortstat) ~., data=train_data, mtry=6,importa
nce =TRUE)
model_7

##
## Call:
```

```
##  randomForest(formula = as.factor(mortstat) ~ ., data = train_data,      m
try = 6, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 19.43%
## Confusion matrix:
##     0  1 class.error
## 0 611 26  0.04081633
## 1 130 36  0.78313253
```

```
importance(model_7)
```

```
##                    0           1 MeanDecreaseAccuracy MeanDecreaseGini
## RIDAGEYR 25.248637706 23.84116670          32.90777347       39.8043742
## RIAGENDR  3.284721381  1.98335764           3.86031787        4.8592858
## BPQ010   -0.509139977  2.10264862           0.72202617        2.7578316
## BPQ060   -2.520578528  0.81349179          -1.54885637        2.5502491
## DIQ010    0.642897126  0.35067182           0.77640826        4.5250667
## DIQ050    0.007616475  0.63384480           0.28229875        1.8035327
## DIQ090    4.309539056  1.36443918           4.28095854        2.6286019
## MCQ010    0.494180757 -1.10817994          -0.09720318        1.6717996
## MCQ053    0.020434252  1.31569209           0.87200588        1.5676737
## MCQ160A   4.262576457  1.52398716           4.51388941        3.7209164
## MCQ160B   7.905550893  5.55422758           9.04943501        6.1064980
## MCQ160K   0.735137889 -0.31035767           0.58819980        2.5300294
## MCQ160L  -0.734592688 -0.34287199          -0.75621520        0.9835438
## BMXWAIST  6.177030088 -0.09125322           5.75749784       20.9237992
## MCQ160M  -1.117305821  0.10760416          -0.87423931        2.2487612
## MCQ220    0.987229359 -0.38146637           0.64488683        2.6204055
## MCQ245A   6.096160177  0.78940513           6.74559367        5.9468047
## MCQ250A   0.238511204  1.09395387           0.74298789        3.5861493
## MCQ250B  -0.792670690  0.81149225          -0.30170373        2.5768425
## MCQ250C  -2.756662141  2.64977560          -0.94424004        3.5329964
## MCQ250E   1.982024381  0.19336151           1.78393269        3.2927717
## MCQ250F  -2.310579239  2.84637225          -0.65741277        4.3657694
## MCQ250G  -0.708604510 -2.60842958          -2.00692065        2.4396419
## MCQ265    0.068557215  0.44856428           0.37761752        3.8273246
## SSQ011   -0.804756959  0.26905810          -0.49538474        1.7938707
## SSQ051    0.579777534  0.33226653           0.62752278        4.5901427
## WHQ030    3.505586780 -0.99265962           2.93099620        3.6612484
## WHQ040    4.150241964 -1.80096075           2.81816239        3.9622935
## LBXRDW    6.398935246 15.08275306          12.99976239       28.7393940
## HSD010    2.853597186  2.22444389           3.79669121       10.5530180
## BPXPULS   3.391860624  3.59581384           5.01449351        3.6868765
## BPXML1    5.517160797 -1.32744237           4.54630013       12.9245684
## VIQ200    8.395901848  1.11240806           8.06681127        6.3144599
## BMXBMI    8.457606701  0.67023857           8.56753802       20.7454883
```

```
## BPXSY1     9.233452499 -1.77921958            7.68224160        17.6126250
## BPXDI1     5.328646070 -2.07108546            3.48438380        15.9114056

varImpPlot(model_7)
```




```
yhat_rf =predict(model_7,newdata=test_data)
table(yhat_rf, test_mort_status)

##         test_mort_status
## yhat_rf   0   1
##       0 598 128
##       1  26  52

(52) / (52+25)

## [1] 0.6753247

(598) / (598+128)

## [1] 0.8236915
```

Support Vector Machines: SVM

Using SVM modeling, I tried modeling with linear, polynomial, and radial kernels. I tried out differente cost features in the SVM model and utilized the `tune()` function in R and the e1071 package to find the best svm model for different costs for polynomial and radial kernels. From there, I used the `predict()` function to predict on the test data. I constructed ROC curves for the final radial model with best fit cost.

Model 8: SVM: Linear Model with Cost = 10

Note: I manually ran cost = 1, cost = 0.5, and cost = 10 to choose a model from those three options.

```
library(e1071)

set.seed(1)
model_8 <- svm(mortstat~., data=train_data, kernel="linear", cost=10)
summary(model_8)

## 
## Call:
## svm(formula = mortstat ~ ., data = train_data, kernel = "linear", 
##     cost = 10)
## 
## 
## Parameters:
##    SVM-Type:  eps-regression 
##  SVM-Kernel:  linear 
##        cost:  10 
##       gamma:  0.02777778 
##     epsilon:  0.1 
## 
## 
## Number of Support Vectors:  390

pred_model_8 =predict(model_8, newdata=test_data)
# not probability - cutoff should then be 0.
# svm fits intercept term
# ?predict can help to convert to probability, 0 or 1, etc.
summary(pred_model_8)

##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max. 
## -0.103264  0.002668  0.032017  0.036307  0.063170  0.292814

svm_pred =rep("0",804)
svm_pred[pred_model_8 > 0]="1"
table(true=test_data$mortstat, pred=svm_pred)

##     pred
## true   0   1
```

```
##    0 173 451
##    1   8 172
```

```
172 / (172+8)
```

```
## [1] 0.9555556
```

```
173 / (173+451)
```

```
## [1] 0.2772436
```

Model 9: SVM: Polynomial

```
set.seed(1)
model_9=tune(svm, mortstat~., data=train_data, kernel="polynomial",
ranges=list(cost=c(0.1,1,10,100,1000)))
summary(model_9)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.488858
##
## - Detailed performance results:
##    cost     error dispersion
## 1 1e-01  0.488858  0.9478462
## 2 1e+00  7.163293 20.6265774
## 3 1e+01 13.368810 28.1196013
## 4 1e+02 12.582465 25.9917215
## 5 1e+03  8.496388 16.8504720
```

The best polynomial model has a cost of 0.1.

```
pred_model_9 =predict(model_9$best.model, newdata=test_data)
svm_pred_9 =rep("0",804)
svm_pred_9[pred_model_9 >0]="1"
table(true=test_data$mortstat, pred=svm_pred_9)
```

```
##     pred
## true   0   1
##    0   9 615
##    1   1 179
```

```
179 / (179+1)
```

```
## [1] 0.9944444
```

```
(9) / (9+615)

## [1] 0.01442308
```

Model 10: SVM: Radial

```
set.seed(1)
model_10=tune(svm, mortstat~., data=train_data, kernel="radial",
ranges=list(cost=c(0.1,1,10,100,1000)))
summary(model_10)

##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.1610457
##
## - Detailed performance results:
##    cost     error dispersion
## 1 1e-01 0.1863994 0.05857509
## 2 1e+00 0.1610457 0.04769782
## 3 1e+01 0.1710888 0.03643432
## 4 1e+02 0.2045542 0.03951186
## 5 1e+03 0.2109801 0.04055448
```

The best radial model has a cost of 1.

```
pred_model_10 =predict(model_10$best.model, newdata=test_data)
svm_pred_10 =rep("0",804)
svm_pred_10[pred_model_10 >0]="1"
table(true=test_data$mortstat, pred=svm_pred_10)

##     pred
## true   0   1
##    0 174 450
##    1   9 171

171 / (179+9)

## [1] 0.9095745

(174) / (174+450)

## [1] 0.2788462
```

ROC for Radial SVM

```
library(ROCR)
rocplot=function(pred, truth, ...){
```

```
  predob = prediction (pred, truth)
  perf = performance (predob , "tpr", "fpr")
  plot(perf ,...)}

svmfit.opt_final=svm(mortstat~., data=train_data, kernel="radial", cost=1,dec
ision.values=T)
fitted=attributes(predict(svmfit.opt_final,train_data,decision.values=TRUE))$
decision.values

par(mfrow=c(1,2))
rocplot(fitted,train_data$mortstat,main="Training Data")
svmfit.flex=svm(mortstat~., data=train_data, kernel="radial", cost=1,decision
.values=T)
fitted=attributes(predict(svmfit.flex,train_data, decision.values=T))$decisio
n.values
rocplot(fitted ,train_data$mortstat,add=T,col="red")
fitted=attributes(predict(svmfit.opt_final,test_data,decision.values=T))$deci
sion.values
rocplot(fitted,test_data$mortstat,main="Test Data")
abline(coef = c(0,1), col="red")
```

As expected, the training data shows good performance on ROC measures. However, it's important to look at the ROC of the test data. This model is performing over the baseline (indicated in red at the 0.5 mark) and can be helpful to consider moving forward.

3. Choosing a Model

Sensitivity and Specificity

In assessing model selection, there are multiple criteria that we can use to assess. One of these model selection tools is sensitivity and specificity. Sensitivity refers to the ability of our model to detect those whose mortality status is truly going to be 1, or that truly dies over the 9-year study period. Having high sensitivity means that there are a few false negative results and that we will miss less cases of death (mort status = 1). Specificity is the ability of the model to detect those who are not going to die (mort status = 0) as truly predicting their mortality status in the 9-year study period to be 0. With maximizing specificity, we'll have less false positive results.

In this case of this data and predicting mortality status, we are more interested in ensuring sensitivity. We want to make sure that we catch those who have a higher risk of predicted mortality so an intervention can be possible within the needed time frame. This is not to say specificity does not matter, but rather, in a mortality analysis model, we want to focus on helping those who could be at risk of a particular disease.

Let's take a look at the sensitivity and specificity of all the models that we created in the table below.

Model	Sensitivity	Specificity	Notes
Model 1: Classification: Logistic Regression using 5 predictors with best subset selection	58.95%	84.80%	Predictors used: RIDAGEYR, RIAGENDR, MCQ160K, LBXRDW, HSD010
Model 2: Classification: Logistic Regression using 10 predictors with best subset selection	63.54%	83.19%	Predictors Used: RIDAGEYR, RIAGENDR, BPQ060, DIQ090, MCQ160A, MCQ160K, SSQ011, LBXRDW, HSD010, BXPULS
Model 3: Classification: LDA with 5 predictors with best subset selection	60.82%	82.89%	Predictors used: RIDAGEYR, RIAGENDR, MCQ160K, LBXRDW, HSD010

Model 4: Classification: LDA with 10 predictors with best subset selection	61.22%	83.00%	Predictors Used: RIDAGEYR, RIAGENDR, BPQ060, DIQ090, MCQ160A, MCQ160K, SSQ011, LBXRDW, HSD010, BXPULS
Model 5: Classification: QDA with 10 predictors with best subset selection	47.15%	82.09%	Predictors Used: RIDAGEYR, RIAGENDR, BPQ060, DIQ090, MCQ160A, MCQ160K, SSQ011, LBXRDW, HSD010, BXPULS
Model 6: Random forest: Bagging on all 36 predictors	58.59%	83.41%	Examined MeanDecreaseAccuracy and MeanDecreaseGini with each predictor. The three highest predictor variables for MeanDecreaseAccuracy were RIDAGEYR (42.43), BMXWAIST (13.63), LBXRDW (13.11). The three highest for MeanDecreaseGini were RIDAGEYR (55.12), LBXRDW (35.29), and BMXBMI (23.56).
Model 7: Random forest without bagging	67.53%	82.37%	Examined MeanDecreaseAccuracy and MeanDecreaseGini with each predictor. The three highest predictor variables for MeanDecreaseAccuracy were RIDAGEYR (32.90), LBXRDW (13.00), and MCQ160B (9.05) – the later of which is different than random forest with bagging which included BMXWAIST instead. The three highest for MeanDecreaseGini were RIDAGEYR (39.80), LBXRDW (28.74), and BMIWAIST (20.92).
Model 8: SVM Linear	95.55%	27.7%	Note: Manually ran cost 0.5, 1, and 10 to choose best model since the tuning parameter on linear was not running quickly.

Model 9: SVM Polynomial	99.44%	1.44%	Best model with tune function had a cost of 0.1
Model 10: SVM Radial	90.95%	27.88%	Best model with tune function had a cost of 1

Model Selection

Three different types of models immediate stand out for performance on specificity and sensitivity. The first type is the SVM models. All three SVM models (linear, polynomial, and radial kernel models) have high sensitivity, but relatively low specificity compared to the rest of the models. Compared to the rest of the models, the predictions for SVM appear to have more predictions that are positive as opposed to negative, which is different than the actual data frame which has only about a quarter of the cases having a mortality status of 1 and the remaining cases that have a mortality status of 0. The SVM models can be considered if we want the highest amount of specificity with its mechanics in this case. However, although we have declared that sensitivity is the model choosing factor that is most important in the case of this mortality data, the low specificity, and the imbalance of prediction versus true across the entire data frame is what makes the SVM models not as preferable to the next two that can be called out.

We now turn attention to the performance of models 2 and 7. These have the next highest specificity after SVM models 8-10. Model 2 is the classification logistic regression method with 10 predictor variables selected from best subset selection. Model 7 is the random forest without bagging model. These two models perform similarly in terms of sensitivity (63.54% and 67.53%, respectively) and specificity (83.19% and 82.37%, respectively). Performance, then, can depend on the actual implementation of the model. If we were working with the base dataset of 813 variables, then random forest might be preferable as it's better able to incorporate a large number of variables relative to sample size into the model. In the assignment, since we were given a list of predictors to start from, this difference is not as important. Random forest and other tree-based methods also commonly improve prediction accuracy at the expense of model and feature interpretability. In this case, since we want to understand and predict mortality status in order to be able to possibly act on predictor variables, model 2, logistic regression with 10 variables chosen from best subset selection, is preferable to advance with. This model has high sensitivity relative to other models and allows us to directly understand how each of the ten variables in the model play a role on prediction of mortality status.