Code Demo

Yupeng Miao

2020/10/14

```r
library(ggplot2)
library(caret)
library(xlsx)
```

1 Data Wrangling

1.0 Load the dataset (No NA entry)

```r
cad <- read.xlsx("DataWrangling/Z-Alizadeh sani dataset.xlsx", 1, header=TRUE)
head(cad)
```

```
##   Age Weight Length   Sex      BMI DM HTN Current.Smoker EX.Smoker FH Obesity
## 1  53     90    175  Male 29.38776  0   1              1         0  0       Y
## 2  67     70    157 Fmale 28.39872  0   1              0         0  0       Y
## 3  54     54    164  Male 20.07733  0   0              1         0  0       N
## 4  66     67    158 Fmale 26.83865  0   1              0         0  0       Y
## 5  50     87    153 Fmale 37.16519  0   1              0         0  0       Y
## 6  50     75    175  Male 24.48980  0   0              1         0  0       N
##   CRF CVA Airway.disease Thyroid.Disease CHF DLP  BP  PR Edema
## 1   N   N              N               N   N   Y 110  80     0
## 2   N   N              N               N   N   N 140  80     1
## 3   N   N              N               N   N   N 100 100     0
## 4   N   N              N               N   N   N 100  80     0
## 5   N   N              N               N   N   N 110  80     0
## 6   N   N              N               N   N   N 118  70     0
##   Weak.Peripheral.Pulse Lung.rales Systolic.Murmur Diastolic.Murmur
## 1                     N          N               N                N
## 2                     N          N               N                N
## 3                     N          N               N                N
## 4                     N          N               N                Y
## 5                     N          N               Y                N
## 6                     N          N               N                N
##   Typical.Chest.Pain Dyspnea Function.Class Atypical Nonanginal Exertional.CP
## 1                  0       N              0        N          N             N
## 2                  1       N              0        N          N             N
## 3                  1       N              0        N          N             N
## 4                  0       Y              3        N          Y             N
## 5                  0       Y              2        N          N             N
```

```
## 6                    1       N              3        N          N             N
##   LowTH.Ang Q.Wave St.Elevation St.Depression Tinversion LVH Poor.R.Progression
## 1         N      0            0             1          1   N                  N
## 2         N      0            0             1          1   N                  N
## 3         N      0            0             0          0   N                  N
## 4         N      0            0             1          0   N                  N
## 5         N      0            0             0          0   N                  N
## 6         N      0            0             0          0   N                  N
##   BBB FBS  CR  TG LDL HDL BUN ESR   HB   K  Na   WBC Lymph Neut PLT EF.TTE
## 1   N  90 0.7 250 155  30   8   7 15.6 4.7 141  5700    39   52 261     50
## 2   N  80 1.0 309 121  36  30  26 13.9 4.7 156  7700    38   55 165     40
## 3   N  85 1.0 103  70  45  17  10 13.5 4.7 139  7400    38   60 230     40
## 4   N  78 1.2  63  55  27  30  76 12.1 4.4 142 13000    18   72 742     55
## 5   N 104 1.0 170 110  50  16  27 13.2 4.0 140  9200    55   39 274     50
## 6   N  86 1.0 139 119  34  13  18 15.6 4.2 141  7300    26   66 194     50
##   Region.RWMA    VHD   Cath
## 1           0      N    Cad
## 2           4      N    Cad
## 3           2   mild    Cad
## 4           0 Severe Normal
## 5           0 Severe Normal
## 6           0      N    Cad
```

1.1 Factorising variables in character

```r
for (i in 1:length(cad)) {
  if (class(cad[,i]) == "character"){
    cad[,i] <- as.factor(cad[,i])
  }
  else if (cad[,i][1] == 0 | cad[,i][1] == 1 ){
    cad[,i] <- as.factor(cad[,i])
  }
}
str(cad)
```

```
## 'data.frame':    303 obs. of  56 variables:
##  $ Age                  : num  53 67 54 66 50 50 55 72 58 60 ...
##  $ Weight               : num  90 70 54 67 87 75 80 80 84 71 ...
##  $ Length               : num  175 157 164 158 153 175 165 175 163 170 ...
##  $ Sex                  : Factor w/ 2 levels "Fmale","Male": 2 1 2 1 1 2 2 2 1 2 ...
##  $ BMI                  : num  29.4 28.4 20.1 26.8 37.2 ...
##  $ DM                   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 2 ...
##  $ HTN                  : Factor w/ 2 levels "0","1": 2 2 1 2 2 1 1 1 1 1 ...
##  $ Current.Smoker       : Factor w/ 2 levels "0","1": 2 1 2 1 1 2 1 2 1 1 ...
##  $ EX.Smoker            : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 2 1 1 1 ...
##  $ FH                   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Obesity              : Factor w/ 2 levels "N","Y": 2 2 1 2 2 1 2 2 2 1 ...
##  $ CRF                  : Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CVA                  : Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Airway.disease       : Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Thyroid.Disease      : Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CHF                  : Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 1 1 1 ...
```

```
##  $ DLP                 : Factor w/ 2 levels "N","Y": 2 1 1 1 1 1 1 2 1 1 ...
##  $ BP                  : num  110 140 100 100 110 118 110 130 90 130 ...
##  $ PR                  : num  80 80 100 80 80 70 80 70 50 70 ...
##  $ Edema               : Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 1 1 1 ...
##  $ Weak.Peripheral.Pulse: Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Lung.rales          : Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Systolic.Murmur     : Factor w/ 2 levels "N","Y": 1 1 1 1 2 1 2 1 1 1 ...
##  $ Diastolic.Murmur    : Factor w/ 2 levels "N","Y": 1 1 1 2 1 1 1 1 1 1 ...
##  $ Typical.Chest.Pain  : Factor w/ 2 levels "0","1": 1 2 2 1 1 2 2 2 1 2 ...
##  $ Dyspnea             : Factor w/ 2 levels "N","Y": 1 1 1 2 2 1 1 1 2 2 ...
##  $ Function.Class      : Factor w/ 4 levels "0","1","2","3": 1 1 1 4 3 4 1 1 1 3 ...
##  $ Atypical            : Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Nonanginal          : Factor w/ 2 levels "N","Y": 1 1 1 2 1 1 1 1 2 1 ...
##  $ Exertional.CP       : Factor w/ 1 level "N": 1 1 1 1 1 1 1 1 1 1 ...
##  $ LowTH.Ang           : Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Q.Wave              : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 2 1 1 1 ...
##  $ St.Elevation        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 2 1 1 1 ...
##  $ St.Depression       : Factor w/ 2 levels "0","1": 2 2 1 2 1 1 1 2 1 1 ...
##  $ Tinversion          : Factor w/ 2 levels "0","1": 2 2 1 1 1 1 2 2 1 1 ...
##  $ LVH                 : Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Poor.R.Progression  : Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 1 1 1 ...
##  $ BBB                 : Factor w/ 3 levels "LBBB","N","RBBB": 2 2 2 2 2 2 2 2 2 1 ...
##  $ FBS                 : num  90 80 85 78 104 86 80 130 69 209 ...
##  $ CR                  : num  0.7 1 1 1.2 1 1 0.8 0.9 0.6 1.3 ...
##  $ TG                  : num  250 309 103 63 170 139 83 80 79 80 ...
##  $ LDL                 : num  155 121 70 55 110 119 85 90 90 90 ...
##  $ HDL                 : num  30 36 45 27 50 34 34 55 59 44 ...
##  $ BUN                 : num  8 30 17 30 16 13 12 19 15 16 ...
##  $ ESR                 : num  7 26 10 76 27 18 38 4 5 8 ...
##  $ HB                  : num  15.6 13.9 13.5 12.1 13.2 15.6 14.1 16.1 11.6 13.9 ...
##  $ K                   : num  4.7 4.7 4.7 4.4 4 4.2 4.8 4.3 3.4 4.6 ...
##  $ Na                  : num  141 156 139 142 140 141 139 142 139 140 ...
##  $ WBC                 : num  5700 7700 7400 13000 9200 7300 9400 12200 5100 4900 ...
##  $ Lymph               : num  39 38 38 18 55 26 58 25 49 55 ...
##  $ Neut                : num  52 55 60 72 39 66 33 74 50 42 ...
##  $ PLT                 : num  261 165 230 742 274 194 292 410 370 380 ...
##  $ EF.TTE              : num  50 40 40 55 50 50 40 45 50 40 ...
##  $ Region.RWMA         : Factor w/ 5 levels "0","1","2","3",..: 1 5 3 1 1 1 5 5 1 3 ...
##  $ VHD                 : Factor w/ 4 levels "mild","Moderate",..: 3 3 1 4 4 3 1 1 3 3 ...
##  $ Cath                : Factor w/ 2 levels "Cad","Normal": 1 1 1 2 2 1 1 1 2 1 ...
```

We observe that the feature Exertional CP only has one level, so we drop it from the dataset

```r
cad <- subset(cad, select = -Exertional.CP)
```

set up the dataset for svm

```r
svm.df <- cad
```

1.2 Catrgorical Random Variables: One-Hot Encoding

```r
dummy <- dummyVars(" ~ .", data = cad)
cad <- data.frame(predict(dummy, newdata = cad))
head(cad)
```

```
##   Age Weight Length Sex.Fmale Sex.Male      BMI DM.0 DM.1 HTN.0 HTN.1
## 1  53     90    175         0        1 29.38776    1    0     0     1
## 2  67     70    157         1        0 28.39872    1    0     0     1
## 3  54     54    164         0        1 20.07733    1    0     1     0
## 4  66     67    158         1        0 26.83865    1    0     0     1
## 5  50     87    153         1        0 37.16519    1    0     0     1
## 6  50     75    175         0        1 24.48980    1    0     1     0
##   Current.Smoker.0 Current.Smoker.1 EX.Smoker.0 EX.Smoker.1 FH.0 FH.1 Obesity.N
## 1                0                1           1           0    1    0         0
## 2                1                0           1           0    1    0         0
## 3                0                1           1           0    1    0         1
## 4                1                0           1           0    1    0         0
## 5                1                0           1           0    1    0         0
## 6                0                1           1           0    1    0         1
##   Obesity.Y CRF.N CRF.Y CVA.N CVA.Y Airway.disease.N Airway.disease.Y
## 1         1     1     0     1     0                1                0
## 2         1     1     0     1     0                1                0
## 3         0     1     0     1     0                1                0
## 4         1     1     0     1     0                1                0
## 5         1     1     0     1     0                1                0
## 6         0     1     0     1     0                1                0
##   Thyroid.Disease.N Thyroid.Disease.Y CHF.N CHF.Y DLP.N DLP.Y  BP  PR Edema.0
## 1                 1                 0     1     0     0     1 110  80       1
## 2                 1                 0     1     0     1     0 140  80       0
## 3                 1                 0     1     0     1     0 100 100       1
## 4                 1                 0     1     0     1     0 100  80       1
## 5                 1                 0     1     0     1     0 110  80       1
## 6                 1                 0     1     0     1     0 118  70       1
##   Edema.1 Weak.Peripheral.Pulse.N Weak.Peripheral.Pulse.Y Lung.rales.N
## 1       0                       1                       0            1
## 2       1                       1                       0            1
## 3       0                       1                       0            1
## 4       0                       1                       0            1
## 5       0                       1                       0            1
## 6       0                       1                       0            1
##   Lung.rales.Y Systolic.Murmur.N Systolic.Murmur.Y Diastolic.Murmur.N
## 1            0                 1                 0                  1
## 2            0                 1                 0                  1
## 3            0                 1                 0                  1
## 4            0                 1                 0                  0
## 5            0                 0                 1                  1
## 6            0                 1                 0                  1
##   Diastolic.Murmur.Y Typical.Chest.Pain.0 Typical.Chest.Pain.1 Dyspnea.N
## 1                  0                    1                    0         1
## 2                  0                    0                    1         1
## 3                  0                    0                    1         1
## 4                  1                    1                    0         0
```

```
## 5                 0                    1                 0         0
## 6                 0                    0                 1         1
##   Dyspnea.Y Function.Class.0 Function.Class.1 Function.Class.2 Function.Class.3
## 1         0                1                0                0                0
## 2         0                1                0                0                0
## 3         0                1                0                0                0
## 4         1                0                0                0                1
## 5         1                0                0                1                0
## 6         0                0                0                0                1
##   Atypical.N Atypical.Y Nonanginal.N Nonanginal.Y LowTH.Ang.N LowTH.Ang.Y
## 1          1          0            1            0           1           0
## 2          1          0            1            0           1           0
## 3          1          0            1            0           1           0
## 4          1          0            0            1           1           0
## 5          1          0            1            0           1           0
## 6          1          0            1            0           1           0
##   Q.Wave.0 Q.Wave.1 St.Elevation.0 St.Elevation.1 St.Depression.0
## 1        1        0              1              0               0
## 2        1        0              1              0               0
## 3        1        0              1              0               1
## 4        1        0              1              0               0
## 5        1        0              1              0               1
## 6        1        0              1              0               1
##   St.Depression.1 Tinversion.0 Tinversion.1 LVH.N LVH.Y Poor.R.Progression.N
## 1               1            0            1     1     0                    1
## 2               1            0            1     1     0                    1
## 3               0            1            0     1     0                    1
## 4               1            1            0     1     0                    1
## 5               0            1            0     1     0                    1
## 6               0            1            0     1     0                    1
##   Poor.R.Progression.Y BBB.LBBB BBB.N BBB.RBBB FBS  CR  TG LDL HDL BUN ESR   HB
## 1                    0        0     1        0  90 0.7 250 155  30   8   7 15.6
## 2                    0        0     1        0  80 1.0 309 121  36  30  26 13.9
## 3                    0        0     1        0  85 1.0 103  70  45  17  10 13.5
## 4                    0        0     1        0  78 1.2  63  55  27  30  76 12.1
## 5                    0        0     1        0 104 1.0 170 110  50  16  27 13.2
## 6                    0        0     1        0  86 1.0 139 119  34  13  18 15.6
##     K  Na   WBC Lymph Neut PLT EF.TTE Region.RWMA.0 Region.RWMA.1 Region.RWMA.2
## 1 4.7 141  5700    39   52 261     50             1             0             0
## 2 4.7 156  7700    38   55 165     40             0             0             0
## 3 4.7 139  7400    38   60 230     40             0             0             1
## 4 4.4 142 13000    18   72 742     55             1             0             0
## 5 4.0 140  9200    55   39 274     50             1             0             0
## 6 4.2 141  7300    26   66 194     50             1             0             0
##   Region.RWMA.3 Region.RWMA.4 VHD.mild VHD.Moderate VHD.N VHD.Severe Cath.Cad
## 1             0             0        0            0     1          0        1
## 2             0             1        0            0     1          0        1
## 3             0             0        1            0     0          0        1
## 4             0             0        0            0     0          1        0
## 5             0             0        0            0     0          1        0
## 6             0             0        0            0     1          0        1
##   Cath.Normal
## 1           0
## 2           0
```

```
## 3           0
## 4           1
## 5           1
## 6           0
```

1.3 Remove Zero Variance or Near-zero variance variable

```r
nz <- nearZeroVar(cad, saveMetrics = TRUE)
nz[nz$nzv,]
```

```
##                           freqRatio percentUnique zeroVar  nzv
## EX.Smoker.0                29.30000      0.660066   FALSE TRUE
## EX.Smoker.1                29.30000      0.660066   FALSE TRUE
## CRF.N                      49.50000      0.660066   FALSE TRUE
## CRF.Y                      49.50000      0.660066   FALSE TRUE
## CVA.N                      59.60000      0.660066   FALSE TRUE
## CVA.Y                      59.60000      0.660066   FALSE TRUE
## Airway.disease.N           26.54545      0.660066   FALSE TRUE
## Airway.disease.Y           26.54545      0.660066   FALSE TRUE
## Thyroid.Disease.N          42.28571      0.660066   FALSE TRUE
## Thyroid.Disease.Y          42.28571      0.660066   FALSE TRUE
## CHF.N                     302.00000      0.660066   FALSE TRUE
## CHF.Y                     302.00000      0.660066   FALSE TRUE
## Edema.0                    24.25000      0.660066   FALSE TRUE
## Edema.1                    24.25000      0.660066   FALSE TRUE
## Weak.Peripheral.Pulse.N    59.60000      0.660066   FALSE TRUE
## Weak.Peripheral.Pulse.Y    59.60000      0.660066   FALSE TRUE
## Lung.rales.N               26.54545      0.660066   FALSE TRUE
## Lung.rales.Y               26.54545      0.660066   FALSE TRUE
## Diastolic.Murmur.N         32.66667      0.660066   FALSE TRUE
## Diastolic.Murmur.Y         32.66667      0.660066   FALSE TRUE
## Function.Class.1          302.00000      0.660066   FALSE TRUE
## LowTH.Ang.N               150.50000      0.660066   FALSE TRUE
## LowTH.Ang.Y               150.50000      0.660066   FALSE TRUE
## St.Elevation.0             20.64286      0.660066   FALSE TRUE
## St.Elevation.1             20.64286      0.660066   FALSE TRUE
## Poor.R.Progression.N       32.66667      0.660066   FALSE TRUE
## Poor.R.Progression.Y       32.66667      0.660066   FALSE TRUE
## BBB.LBBB                   22.30769      0.660066   FALSE TRUE
## BBB.RBBB                   36.87500      0.660066   FALSE TRUE
## Region.RWMA.3              20.64286      0.660066   FALSE TRUE
## Region.RWMA.4              20.64286      0.660066   FALSE TRUE
## VHD.Severe                 26.54545      0.660066   FALSE TRUE
```

```r
nz.svm <- nearZeroVar(svm.df, saveMetrics = T)
nz.svm[nz.svm$nzv,]
```

```
##                         freqRatio percentUnique zeroVar  nzv
## EX.Smoker                29.30000      0.660066   FALSE TRUE
## CRF                      49.50000      0.660066   FALSE TRUE
## CVA                      59.60000      0.660066   FALSE TRUE
```

```
## Airway.disease         26.54545      0.660066   FALSE TRUE
## Thyroid.Disease        42.28571      0.660066   FALSE TRUE
## CHF                   302.00000      0.660066   FALSE TRUE
## Edema                  24.25000      0.660066   FALSE TRUE
## Weak.Peripheral.Pulse  59.60000      0.660066   FALSE TRUE
## Lung.rales             26.54545      0.660066   FALSE TRUE
## Diastolic.Murmur       32.66667      0.660066   FALSE TRUE
## LowTH.Ang             150.50000      0.660066   FALSE TRUE
## St.Elevation           20.64286      0.660066   FALSE TRUE
## Poor.R.Progression     32.66667      0.660066   FALSE TRUE
## BBB                    21.69231      0.990099   FALSE TRUE
```

There are 32 near-zero-variance features which may cause problems when the data are split into cross validation or boostraping samples, we eliminated those features.

```
nzv <- nearZeroVar(cad)
cad <- cad[,-nzv]
nzv.svm <- nearZeroVar(svm.df)
svm.df <- svm.df[,-nzv.svm]
```

1.4 Remove Correlated predictors

```
cad_cor <- cor(cad)
high.cor <- sum(abs(cad_cor[upper.tri(cad_cor)])>0.999)
summary(cad_cor[upper.tri(cad_cor)])
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -1.000000 -0.063363 -0.002277 -0.008947  0.060996  0.725005
```

We observed 17 highly correlated features, most of them are generated by one-hot encoding (e.g. for a variable with 3 levels, drop one of the dummies will not lose any information), only one of them (neut) isn't a column generated by One-Hot Encoding. we drop them and the effect of removing those with absolute correlations above 0.75 are shown below:

```
high.cor.feature <- findCorrelation(cad_cor, cutoff = 0.75)
high.cor.feature[1] = 65 #remove the Cath.Normal rather than Cath.cad
cad <- cad[, -high.cor.feature]
cad_cor2 <- cor(cad)
summary(cad_cor2[upper.tri(cad_cor2)])
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.54585 -0.05229  0.01302  0.01417  0.06687  0.72501
```

```
svm.df <- svm.df[, -c(36, 65)]
```

1.5 Features with linear dependencies:

```
ld <- findLinearCombos(cad)
ld
```

```
## $linearCombos
## list()
##
## $remove
## NULL
```

###Nothing observed so we are ok.

1.6 factorising binary variables:

```
for (i in 1:length(cad)) {
  if (cad[,i][1] == 0 | cad[,i][1] == 1 ){
    cad[,i] <- as.factor(cad[,i])
  }
}
```

2 Train/Test split:

```
set.seed(3164)
train.index <- createDataPartition(cad$Cath.Cad, times = 1, p = 0.8, list = FALSE)
cad.train <- cad[train.index,]
cad.test <- cad[-train.index,]
```

3 Set up 10-cross-validation

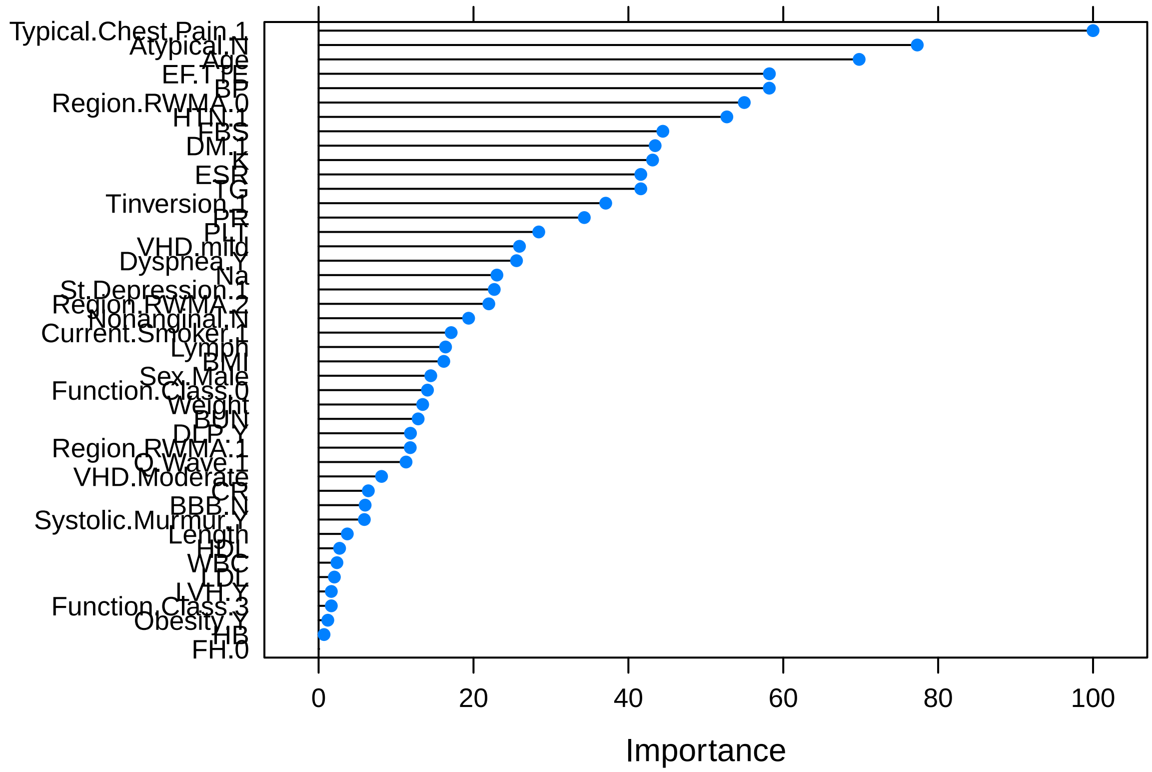
```
parameters = trainControl(method = "repeatedcv",
                          number = 10,
                          repeats = 10,
                          classProbs = FALSE)
```

4 Feature selection by SVM

```
# train SVM
SVM = train(Cath.Cad ~ ., data = cad.train,
            method = "svmPoly",
            trControl = parameters,
            tuneGrid = data.frame(degree = 1, scale = 1, C = 1),
```

```r
             preProcess = c("pca", "scale", "center"),
             na.action = na.omit)

# Ranking features by Importance
feature.rank = varImp(SVM, scale = TRUE)
plot(feature.rank)
```



Generate a dataset of features with importance > 40

```r
important <- c(feature.rank$importance$X0>40, TRUE)
im <- cad[,important]
str(im)
```

```
## 'data.frame':    303 obs. of  13 variables:
##  $ Age                 : num  53 67 54 66 50 50 55 72 58 60 ...
##  $ DM.1                : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 2 ...
##  $ HTN.1               : Factor w/ 2 levels "0","1": 2 2 1 2 2 1 1 1 1 1 ...
##  $ BP                  : num  110 140 100 100 110 118 110 130 90 130 ...
##  $ Typical.Chest.Pain.1: Factor w/ 2 levels "0","1": 1 2 2 1 1 2 2 2 1 2 ...
##  $ Atypical.N          : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ FBS                 : num  90 80 85 78 104 86 80 130 69 209 ...
##  $ TG                  : num  250 309 103 63 170 139 83 80 79 80 ...
##  $ ESR                 : num  7 26 10 76 27 18 38 4 5 8 ...
```

```
##  $ K                   : num  4.7 4.7 4.7 4.4 4 4.2 4.8 4.3 3.4 4.6 ...
##  $ EF.TTE              : num  50 40 40 55 50 50 40 45 50 40 ...
##  $ Region.RWMA.0       : Factor w/ 2 levels "0","1": 2 1 1 2 2 2 1 1 2 1 ...
##  $ Cath.Cad            : Factor w/ 2 levels "0","1": 2 2 2 1 1 2 2 2 1 2 ...
```

```r
# train/test split
im.train <- im[train.index,]
im.test <- im[-train.index,]
```

```r
svm.df <- subset(svm.df, select = c(Age, DM, HTN, BP, Typical.Chest.Pain, Atypical, FBS, TG, ESR, K, EF
str(svm.df)
```

```
## 'data.frame':    303 obs. of  13 variables:
##  $ Age               : num  53 67 54 66 50 50 55 72 58 60 ...
##  $ DM                : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 2 ...
##  $ HTN               : Factor w/ 2 levels "0","1": 2 2 1 2 2 1 1 1 1 1 ...
##  $ BP                : num  110 140 100 100 110 118 110 130 90 130 ...
##  $ Typical.Chest.Pain: Factor w/ 2 levels "0","1": 1 2 2 1 1 2 2 2 1 2 ...
##  $ Atypical          : Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 1 1 1 ...
##  $ FBS               : num  90 80 85 78 104 86 80 130 69 209 ...
##  $ TG                : num  250 309 103 63 170 139 83 80 79 80 ...
##  $ ESR               : num  7 26 10 76 27 18 38 4 5 8 ...
##  $ K                 : num  4.7 4.7 4.7 4.4 4 4.2 4.8 4.3 3.4 4.6 ...
##  $ EF.TTE            : num  50 40 40 55 50 50 40 45 50 40 ...
##  $ Region.RWMA       : Factor w/ 5 levels "0","1","2","3",..: 1 5 3 1 1 1 5 5 1 3 ...
##  $ Cath              : Factor w/ 2 levels "Cad","Normal": 1 1 1 2 2 1 1 1 2 1 ...
```

3 Claasifier Training and Testing

3.0 Initialising training settings

```r
# set up results table
result = data.frame(Classifier = c("SVM.Poly", "SVM.Radial", "SVM.Linear", "Random Forest", "Neurual Ne
```

```r
control = trainControl(method = "repeatedcv",
                       number = 10,
                       repeats = 10,
                       classProbs = TRUE,
                       summaryFunction = twoClassSummary
                       )
```

```r
# Test/ train split for SVM
set.seed(3164)
train.index.svm <- createDataPartition(svm.df$Cath, times = 1, p = 0.8, list = FALSE)
svm.train <- svm.df[train.index.svm,]
svm.test <- svm.df[-train.index.svm,]
```

3.1 SVM Polynomial

```r
set.seed(3164)
# Train SVM polynomial with ROC metric
svm.poly= train(Cath ~ ., data = svm.train,
            method = "svmPoly",
            trControl = control,
            tuneGrid = data.frame(degree = c(1,1), scale = c(1,2), C = c(1,3)),
            preProcess = c("pca", "scale", "center"),
            metric = "ROC",
            na.action = na.omit)
print(svm.poly$results)
```

```
##   degree scale C       ROC      Sens      Spec      ROCSD     SensSD    SpecSD
## 1      1     1 1 0.9152148 0.9198693 0.6985714 0.05763243 0.06684049 0.1623053
## 2      1     2 3 0.9134314 0.9146405 0.7000000 0.05721340 0.06724725 0.1618029
```

```r
# Train without ROC
SVM = train(Cath.Cad ~ ., data = im.train,
            method = "svmPoly",
            trControl = parameters,
            tuneGrid = data.frame(degree = c(1,1), scale = c(1,2), C = c(1,3)),
            preProcess = c("pca", "scale", "center"),
            na.action = na.omit)
print(SVM$results)
```

```
##   degree scale C  Accuracy     Kappa AccuracySD   KappaSD
## 1      1     1 1 0.8643667 0.6612775 0.06272514 0.1605093
## 2      1     2 3 0.8622500 0.6564865 0.06229360 0.1579216
```

```r
# Test SVM polynomial with ROC
svm.poly.predict = predict(svm.poly, svm.test)
# confusion matrix
cm.svm.poly <- confusionMatrix(svm.poly.predict, svm.test$Cath)
print(cm.svm.poly)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Cad Normal
##     Cad     40      5
##     Normal   3     12
##
##                Accuracy : 0.8667
##                  95% CI : (0.7541, 0.9406)
##     No Information Rate : 0.7167
##     P-Value [Acc > NIR] : 0.004937
##
##                   Kappa : 0.6596
##
##  Mcnemar's Test P-Value : 0.723674
```

```
##
##             Sensitivity : 0.9302
##             Specificity : 0.7059
##          Pos Pred Value : 0.8889
##          Neg Pred Value : 0.8000
##              Prevalence : 0.7167
##          Detection Rate : 0.6667
##    Detection Prevalence : 0.7500
##       Balanced Accuracy : 0.8181
##
##        'Positive' Class : Cad
##
```

```
# Test SVM polynomial without ROC
svm.predict = predict(SVM, im.test)
# confusion matrix
cm.svm <- confusionMatrix(svm.predict, im.test$Cath.Cad)
print(cm.svm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 13  8
##          1  4 35
##
##                Accuracy : 0.8
##                  95% CI : (0.6767, 0.8922)
##     No Information Rate : 0.7167
##     P-Value [Acc > NIR] : 0.09575
##
##                   Kappa : 0.5402
##
##  Mcnemar's Test P-Value : 0.38648
##
##             Sensitivity : 0.7647
##             Specificity : 0.8140
##          Pos Pred Value : 0.6190
##          Neg Pred Value : 0.8974
##              Prevalence : 0.2833
##          Detection Rate : 0.2167
##    Detection Prevalence : 0.3500
##       Balanced Accuracy : 0.7893
##
##        'Positive' Class : 0
##
```

We notice that training classifier with ROC as metric yields higher testing accuracy, thus, we'll train the left classifiers with ROC.

```
result$ROC[1] = max(svm.poly$results$ROC)
result$Testing_Accuracy[1] = cm.svm.poly$overall[1]
```

3.2 SVM Radial

```
set.seed(3164)
# Train SVM Radial
svm.rad= train(Cath ~ ., data = svm.train,
            method = "svmRadial",
            trControl = control,
            tuneLength = 10,
            preProcess = c("pca", "scale", "center"),
            metric = "ROC",
            na.action = na.omit)
print(svm.rad$results)
```

```
##        sigma      C       ROC      Sens      Spec      ROCSD     SensSD
## 1  0.0622846   0.25 0.9048553 0.9100000 0.7085714 0.06768556 0.06919869
## 2  0.0622846   0.50 0.9000047 0.9202288 0.6371429 0.07110917 0.06684309
## 3  0.0622846   1.00 0.8957423 0.9283660 0.6271429 0.07244670 0.05976535
## 4  0.0622846   2.00 0.8927358 0.9168954 0.6414286 0.07541720 0.06588594
## 5  0.0622846   4.00 0.8882353 0.9104575 0.6400000 0.07641461 0.06862824
## 6  0.0622846   8.00 0.8802708 0.9057190 0.6214286 0.07787928 0.06998145
## 7  0.0622846  16.00 0.8840056 0.9085948 0.6142857 0.07712949 0.07069320
## 8  0.0622846  32.00 0.8633333 0.9114379 0.5700000 0.08064663 0.07697575
## 9  0.0622846  64.00 0.8471615 0.9109804 0.5457143 0.08502766 0.07488678
## 10 0.0622846 128.00 0.8317414 0.9118627 0.5071429 0.08819426 0.07386473
##       SpecSD
## 1  0.1721965
## 2  0.1820880
## 3  0.1780526
## 4  0.1630656
## 5  0.1680278
## 6  0.1678990
## 7  0.1498986
## 8  0.1471155
## 9  0.1641806
## 10 0.1641743
```

```
# Test SVM Radial
svm.rad.predict = predict(svm.rad, svm.test)
# confusion matrix
cm.svm.rad <- confusionMatrix(svm.rad.predict, svm.test$Cath)
print(cm.svm.rad)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Cad Normal
##     Cad     40      7
##     Normal   3     10
##
##                Accuracy : 0.8333
##                  95% CI : (0.7148, 0.9171)
##     No Information Rate : 0.7167
```

```
##     P-Value [Acc > NIR] : 0.02687
##
##                   Kappa : 0.5582
##
##  Mcnemar's Test P-Value : 0.34278
##
##             Sensitivity : 0.9302
##             Specificity : 0.5882
##          Pos Pred Value : 0.8511
##          Neg Pred Value : 0.7692
##              Prevalence : 0.7167
##          Detection Rate : 0.6667
##    Detection Prevalence : 0.7833
##       Balanced Accuracy : 0.7592
##
##        'Positive' Class : Cad
##
```

```
result$ROC[2] = max(svm.rad$results$ROC)
result$Testing_Accuracy[2] = cm.svm.rad$overall[1]
```

3.3 SVM linear

```
set.seed(3164)
# Train SVM linear
svm.l= train(Cath ~ ., data = svm.train,
            method = "svmLinear",
            trControl = control,
            tuneLength = 10,
            preProcess = c("pca", "scale", "center"),
            metric = "ROC",
            na.action = na.omit)
print(svm.l$results)
```

```
##   C       ROC      Sens      Spec      ROCSD   SensSD    SpecSD
## 1 1 0.9152148 0.9245098 0.6885714 0.05763243 0.062564 0.1666729
```

```
# Test SVM linear
svm.l.predict = predict(svm.l, svm.test)
# confusion matrix
cm.svm.l <- confusionMatrix(svm.l.predict, svm.test$Cath)
print(cm.svm.l)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Cad Normal
##     Cad     42      5
##     Normal   1     12
##
```

```
##                Accuracy : 0.9
##                  95% CI : (0.7949, 0.9624)
##     No Information Rate : 0.7167
##     P-Value [Acc > NIR] : 0.0005383
##
##                   Kappa : 0.7349
##
##  Mcnemar's Test P-Value : 0.2206714
##
##             Sensitivity : 0.9767
##             Specificity : 0.7059
##          Pos Pred Value : 0.8936
##          Neg Pred Value : 0.9231
##              Prevalence : 0.7167
##          Detection Rate : 0.7000
##    Detection Prevalence : 0.7833
##       Balanced Accuracy : 0.8413
##
##        'Positive' Class : Cad
##
```

```
result$ROC[3] = max(svm.l$results$ROC)
result$Testing_Accuracy[3] = cm.svm.l$overall[1]
```

```
library("pROC")
```

```
svm.poly.probs = predict(svm.poly,svm.test[,!names(svm.test) %in% c("Cath")],type = "prob")
svm.rad.probs = predict(svm.rad,svm.test[,!names(svm.test) %in% c("Cath")],type = "prob")
svm.l.probs = predict(svm.l,svm.test[,!names(svm.test) %in% c("Cath")],type = "prob")
# plot ROC
svm.poly.ROC = roc(response = svm.test$Cath,
              predictor = svm.poly.probs$Cad,
              levels = levels(svm.test$Cath),
              percent = T)

svm.rad.ROC = roc(response = svm.test$Cath,
              predictor = svm.rad.probs$Cad,
              levels = levels(svm.test$Cath),
              percent = T)

svm.l.ROC = roc(response = svm.test$Cath,
              predictor = svm.l.probs$Cad,
              levels = levels(svm.test$Cath),
              percent = T)
```

```
result$Test_AUC[1] = svm.poly.ROC$auc
result$Test_AUC[2] = svm.rad.ROC$auc
result$Test_AUC[3] = svm.l.ROC$auc
```

3.4 Random Forest

```r
# train the random forest
rf <- train(Cath ~., data = svm.train,
                 method = "rf",
                 trControl = control,
                 preProc = c("center", "scale"),
                 tuneLength = 10,
                 metric = "ROC")
print(rf)
```

```
## Random Forest
##
## 243 samples
##  12 predictor
##   2 classes: 'Cad', 'Normal'
##
## Pre-processing: centered (15), scaled (15)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 219, 218, 219, 218, 219, 218, ...
## Resampling results across tuning parameters:
##
##   mtry  ROC        Sens       Spec
##    2    0.9215780  0.9130392  0.6971429
##    3    0.9158310  0.9068301  0.7214286
##    4    0.9125934  0.9009804  0.7214286
##    6    0.9118511  0.8916993  0.7342857
##    7    0.9117787  0.8923529  0.7371429
##    9    0.9102171  0.8934967  0.7414286
##   10    0.9088772  0.8887255  0.7357143
##   12    0.9074416  0.8901307  0.7400000
##   13    0.9060224  0.8848693  0.7442857
##   15    0.9042554  0.8871895  0.7528571
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```r
# Test the random forest
rf.predict = predict(rf, svm.test)
# confusion matrix
cm.rf <- confusionMatrix(rf.predict, svm.test$Cath)
print(cm.rf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Cad Normal
##     Cad     40      5
##     Normal   3     12
##
##                Accuracy : 0.8667
##                  95% CI : (0.7541, 0.9406)
```

```
##     No Information Rate : 0.7167
##     P-Value [Acc > NIR] : 0.004937
##
##                   Kappa : 0.6596
##
##  Mcnemar's Test P-Value : 0.723674
##
##             Sensitivity : 0.9302
##             Specificity : 0.7059
##          Pos Pred Value : 0.8889
##          Neg Pred Value : 0.8000
##              Prevalence : 0.7167
##          Detection Rate : 0.6667
##    Detection Prevalence : 0.7500
##       Balanced Accuracy : 0.8181
##
##        'Positive' Class : Cad
##
```

```
# store the ROC and Testing accuracy to the results table
result$ROC[4] = max(rf$results$ROC)
result$Testing_Accuracy[4] = cm.rf$overall[1]

# Save the testing ROC for plotting later
rf.probs = predict(rf,svm.test[,!names(svm.test) %in% c("Cath")],type = "prob")
rf.ROC = roc(response = svm.test$Cath,
             predictor = rf.probs$Cad,
             levels = levels(svm.test$Cath),
             percent = T)
result$Test_AUC[4] = rf.ROC$auc
```

3.5 Neural Network

```
# train the neural network
cath_index = grep("Cath", colnames(svm.train))
nn = train(svm.train[, -cath_index],
            svm.train$Cath,
            method = "nnet",
            trControl = control,
            preProcess = c("scale", "center"),
            tuneLength = 5,
            verbose = F,
            trace = F,
            metric = "ROC",
            na.action = na.omit)
print(nn)
```

```
## Neural Network
##
## 243 samples
##  12 predictor
```

```
##   2 classes: 'Cad', 'Normal'
##
## Pre-processing: scaled (7), centered (7), ignore (5)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 218, 219, 218, 219, 219, 219, ...
## Resampling results across tuning parameters:
##
##   size  decay  ROC        Sens       Spec
##   1     0e+00  0.8444935  0.8625817  0.7971429
##   1     1e-04  0.8717110  0.8568301  0.7857143
##   1     1e-03  0.8980252  0.8647386  0.7871429
##   1     1e-02  0.9149253  0.8681373  0.7914286
##   1     1e-01  0.9207843  0.8879085  0.7614286
##   3     0e+00  0.8070215  0.8605882  0.7028571
##   3     1e-04  0.8328735  0.8534641  0.7442857
##   3     1e-03  0.8578618  0.8758497  0.7200000
##   3     1e-02  0.8530439  0.8589542  0.6800000
##   3     1e-01  0.9024697  0.8878105  0.7185714
##   5     0e+00  0.7889052  0.8517647  0.6571429
##   5     1e-04  0.8278175  0.8631699  0.6600000
##   5     1e-03  0.8374510  0.8638562  0.6528571
##   5     1e-02  0.8569328  0.8715033  0.6157143
##   5     1e-01  0.9035434  0.9028431  0.6857143
##   7     0e+00  0.7809687  0.8523203  0.6571429
##   7     1e-04  0.8320121  0.8742484  0.6142857
##   7     1e-03  0.8449090  0.8684314  0.5957143
##   7     1e-02  0.8593931  0.8778758  0.6357143
##   7     1e-01  0.8979458  0.9022549  0.6871429
##   9     0e+00  0.7843044  0.8584967  0.6385714
##   9     1e-04  0.8561064  0.8838562  0.6471429
##   9     1e-03  0.8666270  0.8763072  0.6571429
##   9     1e-02  0.8732610  0.8784967  0.6385714
##   9     1e-01  0.8984687  0.8993464  0.6842857
##
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were size = 1 and decay = 0.1.
```

```r
# testing the neural network
nn.predict = predict(nn, svm.test)
# confusion matrix
cm.nn <- confusionMatrix(nn.predict, svm.test$Cath)
print(cm.nn)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Cad Normal
##     Cad     37      4
##     Normal   6     13
##
##                Accuracy : 0.8333
##                  95% CI : (0.7148, 0.9171)
##     No Information Rate : 0.7167
##     P-Value [Acc > NIR] : 0.02687
```

```
##
##                   Kappa : 0.6037
##
##  Mcnemar's Test P-Value : 0.75183
##
##             Sensitivity : 0.8605
##             Specificity : 0.7647
##          Pos Pred Value : 0.9024
##          Neg Pred Value : 0.6842
##              Prevalence : 0.7167
##          Detection Rate : 0.6167
##    Detection Prevalence : 0.6833
##       Balanced Accuracy : 0.8126
##
##        'Positive' Class : Cad
##
```

```
# store the ROC and Testing accuracy to the results table
result$ROC[5] = max(nn$results$ROC)
result$Testing_Accuracy[5] = cm.nn$overall[1]
# plot the testing ROC
nn.probs = predict(nn,svm.test[,!names(svm.test) %in% c("Cath")],type = "prob")
nn.ROC = roc(response = svm.test$Cath,
              predictor = nn.probs$Cad,
              levels = levels(svm.test$Cath),
              percent = T)
result$Test_AUC[5] = nn.ROC$auc
```

3.6 Naive Bayes

```
set.seed(3164)
# train Naive Bayes
nb <- train(Cath ~., data = svm.train,
                 method = "nb",
                 trControl = control,
                 preProc = c("center", "scale"),
                 tuneGrid = data.frame(fL = 0, usekernel = T, adjust = 1),
                 metric = "ROC")
print(nb)
```

```
## Naive Bayes
##
## 243 samples
##  12 predictor
##   2 classes: 'Cad', 'Normal'
##
## Pre-processing: centered (15), scaled (15)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 219, 219, 218, 219, 219, 218, ...
## Resampling results:
##
```

```
##   ROC        Sens       Spec
##   0.8881139  0.5108497  0.9428571
##
## Tuning parameter 'fL' was held constant at a value of 0
## Tuning
##  parameter 'usekernel' was held constant at a value of TRUE
## Tuning
##  parameter 'adjust' was held constant at a value of 1
```

```r
# Test the Naive Bayes
nb.predict = predict(nb, svm.test)
# confusion matrix
cm.nb <- confusionMatrix(nb.predict, svm.test$Cath)
print(cm.nb)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction Cad Normal
##     Cad     24      1
##     Normal  19     16
##
##                Accuracy : 0.6667
##                  95% CI : (0.5331, 0.7831)
##     No Information Rate : 0.7167
##     P-Value [Acc > NIR] : 0.8421385
##
##                   Kappa : 0.3782
##
##  Mcnemar's Test P-Value : 0.0001439
##
##             Sensitivity : 0.5581
##             Specificity : 0.9412
##          Pos Pred Value : 0.9600
##          Neg Pred Value : 0.4571
##              Prevalence : 0.7167
##          Detection Rate : 0.4000
##    Detection Prevalence : 0.4167
##       Balanced Accuracy : 0.7497
##
##        'Positive' Class : Cad
##
```

```r
# store the ROC and Testing accuracy to the results table
result$ROC[6] = max(nb$results$ROC)
result$Testing_Accuracy[6] = cm.nb$overall[1]

# Save the testing ROC for plotting later
nb.probs = predict(nb,svm.test[,!names(svm.test) %in% c("Cath")],type = "prob")
nb.ROC = roc(response = svm.test$Cath,
              predictor = nb.probs$Cad,
              levels = levels(svm.test$Cath),
              percent = T)
result$Test_AUC[6] = nb.ROC$auc
```

3.7 Logistic Regression

```
set.seed(3164)
# train the logistic regression
lr <- train(Cath ~., data = svm.train,
                 method = "glm",
                 family = "binomial",
                 trControl = control,
                 preProc = c("center", "scale"),
                 tuneLength = 10,
                 metric = "ROC")
print(lr)
```

```
## Generalized Linear Model
##
## 243 samples
##  12 predictor
##   2 classes: 'Cad', 'Normal'
##
## Pre-processing: centered (15), scaled (15)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 219, 219, 218, 219, 219, 218, ...
## Resampling results:
##
##   ROC        Sens      Spec
##   0.9166153  0.905915  0.74
```

```
# Test the logistic regression
lr.predict = predict(lr, svm.test)
# confusion matrix
cm.lr <- confusionMatrix(lr.predict, svm.test$Cath)
print(cm.lr)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Cad Normal
##     Cad     39      4
##     Normal   4     13
##
##                Accuracy : 0.8667
##                  95% CI : (0.7541, 0.9406)
##     No Information Rate : 0.7167
##     P-Value [Acc > NIR] : 0.004937
##
##                   Kappa : 0.6717
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.9070
##             Specificity : 0.7647
##          Pos Pred Value : 0.9070
```

```
##          Neg Pred Value : 0.7647
##              Prevalence : 0.7167
##          Detection Rate : 0.6500
##    Detection Prevalence : 0.7167
##       Balanced Accuracy : 0.8358
##
##        'Positive' Class : Cad
##
```

```
# store the ROC and Testing accuracy to the results table
result$ROC[7] = max(lr$results$ROC)
result$Testing_Accuracy[7] = cm.lr$overall[1]

# Save the testing ROC for plotting later
lr.probs = predict(lr,svm.test[,!names(svm.test) %in% c("Cath")],type = "prob")
lr.ROC = roc(response = svm.test$Cath,
              predictor = lr.probs$Cad,
              levels = levels(svm.test$Cath),
              percent = T)
result$Test_AUC[7] = lr.ROC$auc
```

3.8 LDA

```
set.seed(3164)
# train the LDA
lda <- train(Cath ~., data = svm.train,
                method = "lda",
                trControl = control,
                preProc = c("center", "scale"),
                tuneLength = 10,
                metric = "ROC")
print(lda)
```

```
## Linear Discriminant Analysis
##
## 243 samples
##  12 predictor
##   2 classes: 'Cad', 'Normal'
##
## Pre-processing: centered (15), scaled (15)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 219, 219, 218, 219, 219, 218, ...
## Resampling results:
##
##   ROC        Sens       Spec
##   0.9234874  0.8832353  0.8142857
```

```
# Test the lda
lda.predict = predict(lda, svm.test)
# confusion matrix
cm.lda <- confusionMatrix(lda.predict, svm.test$Cath)
print(cm.lda)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Cad Normal
##     Cad     39      4
##     Normal   4     13
##
##                Accuracy : 0.8667
##                  95% CI : (0.7541, 0.9406)
##     No Information Rate : 0.7167
##     P-Value [Acc > NIR] : 0.004937
##
##                   Kappa : 0.6717
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.9070
##             Specificity : 0.7647
##          Pos Pred Value : 0.9070
##          Neg Pred Value : 0.7647
##              Prevalence : 0.7167
##          Detection Rate : 0.6500
##    Detection Prevalence : 0.7167
##       Balanced Accuracy : 0.8358
##
##        'Positive' Class : Cad
##
```

```
# store the ROC and Testing accuracy to the results table
result$ROC[8] = max(lda$results$ROC)
result$Testing_Accuracy[8] = cm.lda$overall[1]

# Save the testing ROC for plotting later
lda.probs = predict(lda,svm.test[,!names(svm.test) %in% c("Cath")],type = "prob")
lda.ROC = roc(response = svm.test$Cath,
              predictor = lda.probs$Cad,
              levels = levels(svm.test$Cath),
              percent = T)
result$Test_AUC[8] = lda.ROC$auc
```

3.9 KNN

```
set.seed(3164)
# train the knn
knn <- train(Cath ~., data = svm.train,
                 method = "knn",
                 trControl = control,
                 preProc = c("center", "scale"),
                 tuneLength = 10,
                 metric = "ROC")
print(knn)
```

```
## k-Nearest Neighbors
##
## 243 samples
##  12 predictor
##   2 classes: 'Cad', 'Normal'
##
## Pre-processing: centered (15), scaled (15)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 219, 219, 218, 219, 219, 218, ...
## Resampling results across tuning parameters:
##
##   k   ROC        Sens       Spec
##    5  0.8788235  0.8977451  0.6814286
##    7  0.8883380  0.9076144  0.7185714
##    9  0.8888025  0.9065686  0.7485714
##   11  0.8864169  0.9006536  0.7600000
##   13  0.8926471  0.9012092  0.7700000
##   15  0.8988049  0.9041503  0.7542857
##   17  0.8996615  0.9006863  0.7371429
##   19  0.8984757  0.9008170  0.7314286
##   21  0.9008730  0.9019608  0.7100000
##   23  0.8997946  0.9025490  0.7085714
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 21.
```

```r
# Test the knn
knn.predict = predict(knn, svm.test)
# confusion matrix
cm.knn <- confusionMatrix(knn.predict, svm.test$Cath)
print(cm.knn)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Cad Normal
##     Cad     39      6
##     Normal   4     11
##
##                Accuracy : 0.8333
##                  95% CI : (0.7148, 0.9171)
##     No Information Rate : 0.7167
##     P-Value [Acc > NIR] : 0.02687
##
##                   Kappa : 0.5745
##
##  Mcnemar's Test P-Value : 0.75183
##
##             Sensitivity : 0.9070
##             Specificity : 0.6471
##          Pos Pred Value : 0.8667
##          Neg Pred Value : 0.7333
##              Prevalence : 0.7167
##          Detection Rate : 0.6500
```

```
##    Detection Prevalence : 0.7500
##       Balanced Accuracy : 0.7770
##
##        'Positive' Class : Cad
##
```

```r
# store the ROC and Testing accuracy to the results table
result$ROC[9] = max(knn$results$ROC)
result$Testing_Accuracy[9] = cm.knn$overall[1]

# Save the testing ROC for plotting later
knn.probs = predict(knn,svm.test[,!names(svm.test) %in% c("Cath")],type = "prob")
knn.ROC = roc(response = svm.test$Cath,
              predictor = knn.probs$Cad,
              levels = levels(svm.test$Cath),
              percent = T)
result$Test_AUC[9] = knn.ROC$auc
```

3.10 GBM

```r
set.seed(3164)
# train the GBM
gbm <- train(Cath ~., data = svm.train,
                 method = "gbm",
                 trControl = control,
                 verbose = FALSE,
                 preProc = c("center", "scale"),
                 tuneLength = 10,
                 metric = "ROC")
print(gbm)
```

```
## Stochastic Gradient Boosting
##
## 243 samples
##  12 predictor
##   2 classes: 'Cad', 'Normal'
##
## Pre-processing: centered (15), scaled (15)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 219, 219, 218, 219, 219, 218, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  ROC        Sens       Spec
##    1                  50      0.9198413  0.9299673  0.6685714
##    1                 100      0.9258497  0.9248366  0.7242857
##    1                 150      0.9243371  0.9143791  0.7300000
##    1                 200      0.9221849  0.9069281  0.7285714
##    1                 250      0.9213725  0.9085948  0.7257143
##    1                 300      0.9184407  0.9022222  0.7300000
##    1                 350      0.9167554  0.8992157  0.7157143
##    1                 400      0.9154155  0.8986928  0.7157143
```

```
##     1                        450       0.9134314  0.8963399  0.7128571
##     1                        500       0.9109104  0.8934641  0.7100000
##     2                         50       0.9195985  0.9214052  0.7085714
##     2                        100       0.9158777  0.9167320  0.7128571
##     2                        150       0.9127218  0.9107843  0.7057143
##     2                        200       0.9088936  0.9022549  0.7128571
##     2                        250       0.9083987  0.9028431  0.7042857
##     2                        300       0.9044164  0.9046732  0.6957143
##     2                        350       0.9045285  0.9017320  0.6885714
##     2                        400       0.9046359  0.8987908  0.6928571
##     2                        450       0.9033894  0.8959150  0.7000000
##     2                        500       0.9010131  0.8970915  0.6928571
##     3                         50       0.9132213  0.9133660  0.7014286
##     3                        100       0.9062325  0.9041503  0.6928571
##     3                        150       0.9006863  0.9024837  0.6857143
##     3                        200       0.8991690  0.8964706  0.6928571
##     3                        250       0.8976657  0.8959477  0.6900000
##     3                        300       0.8961625  0.8942484  0.6942857
##     3                        350       0.8962792  0.8914052  0.6928571
##     3                        400       0.8937628  0.8948366  0.6957143
##     3                        450       0.8930812  0.8913725  0.6957143
##     3                        500       0.8922782  0.8907516  0.6957143
##     4                         50       0.9087162  0.9138889  0.6900000
##     4                        100       0.9031746  0.9001307  0.6885714
##     4                        150       0.8989356  0.8948366  0.6971429
##     4                        200       0.8956022  0.8872876  0.6971429
##     4                        250       0.8958870  0.8862418  0.7000000
##     4                        300       0.8934500  0.8896078  0.6928571
##     4                        350       0.8911251  0.8855882  0.6914286
##     4                        400       0.8908450  0.8843791  0.6971429
##     4                        450       0.8893184  0.8826797  0.6900000
##     4                        500       0.8908310  0.8832680  0.6928571
##     5                         50       0.9092810  0.9161111  0.7100000
##     5                        100       0.9019468  0.8999020  0.7042857
##     5                        150       0.8982446  0.8964706  0.6942857
##     5                        200       0.8957516  0.8930065  0.7000000
##     5                        250       0.8915033  0.8924183  0.6928571
##     5                        300       0.8907563  0.8879085  0.6928571
##     5                        350       0.8892764  0.8850000  0.6900000
##     5                        400       0.8869608  0.8809477  0.6871429
##     5                        450       0.8871382  0.8768627  0.6828571
##     5                        500       0.8848880  0.8774183  0.6871429
##     6                         50       0.9030019  0.9075490  0.6885714
##     6                        100       0.8936181  0.8925163  0.6914286
##     6                        150       0.8908777  0.8900654  0.6828571
##     6                        200       0.8903688  0.8894771  0.6842857
##     6                        250       0.8872782  0.8842484  0.6871429
##     6                        300       0.8870962  0.8831046  0.6842857
##     6                        350       0.8877591  0.8837908  0.6842857
##     6                        400       0.8865453  0.8826471  0.6928571
##     6                        450       0.8862792  0.8769935  0.6942857
##     6                        500       0.8840289  0.8768627  0.6914286
##     7                         50       0.9033707  0.9052614  0.7057143
##     7                        100       0.8953455  0.8929739  0.6914286
```

```
##    7                   150      0.8906162  0.8877778  0.6900000
##    7                   200      0.8892390  0.8854248  0.6871429
##    7                   250      0.8888329  0.8792157  0.6871429
##    7                   300      0.8866387  0.8808170  0.6885714
##    7                   350      0.8864939  0.8784967  0.6814286
##    7                   400      0.8846545  0.8784967  0.6857143
##    7                   450      0.8831979  0.8756863  0.6900000
##    7                   500      0.8836461  0.8720261  0.6885714
##    8                    50      0.9052007  0.9046405  0.6914286
##    8                   100      0.8976797  0.8983007  0.6885714
##    8                   150      0.8928478  0.8919935  0.6928571
##    8                   200      0.8912885  0.8889869  0.7071429
##    8                   250      0.8907843  0.8866340  0.7000000
##    8                   300      0.8889683  0.8849346  0.7014286
##    8                   350      0.8888002  0.8831046  0.7014286
##    8                   400      0.8885247  0.8790523  0.7028571
##    8                   450      0.8875163  0.8791176  0.6885714
##    8                   500      0.8868861  0.8768301  0.6885714
##    9                    50      0.9036835  0.9035621  0.7028571
##    9                   100      0.8968347  0.8884641  0.6771429
##    9                   150      0.8947059  0.8845098  0.6800000
##    9                   200      0.8913632  0.8797386  0.6814286
##    9                   250      0.8891923  0.8774183  0.6957143
##    9                   300      0.8860831  0.8750654  0.6914286
##    9                   350      0.8869141  0.8733987  0.6985714
##    9                   400      0.8839683  0.8728105  0.7000000
##    9                   450      0.8834827  0.8721895  0.6957143
##    9                   500      0.8831373  0.8726797  0.6942857
##   10                    50      0.9031232  0.9000000  0.7042857
##   10                   100      0.9006956  0.8936928  0.7028571
##   10                   150      0.8959244  0.8885621  0.7014286
##   10                   200      0.8925023  0.8862092  0.6942857
##   10                   250      0.8907423  0.8833333  0.7042857
##   10                   300      0.8890009  0.8868301  0.6900000
##   10                   350      0.8884127  0.8844444  0.7000000
##   10                   400      0.8873716  0.8815686  0.6871429
##   10                   450      0.8868908  0.8785948  0.6900000
##   10                   500      0.8875724  0.8843464  0.6900000
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 100, interaction.depth =
##  1, shrinkage = 0.1 and n.minobsinnode = 10.
```

```r
# Test the gbm
gbm.predict = predict(gbm, svm.test)
# confusion matrix
cm.gbm <- confusionMatrix(gbm.predict, svm.test$Cath)
print(cm.gbm)
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction Cad Normal
##     Cad     38      4
##     Normal   5     13
##
##                Accuracy : 0.85
##                  95% CI : (0.7343, 0.929)
##     No Information Rate : 0.7167
##     P-Value [Acc > NIR] : 0.01221
##
##                   Kappa : 0.6371
##
##  Mcnemar's Test P-Value : 1.00000
##
##             Sensitivity : 0.8837
##             Specificity : 0.7647
##          Pos Pred Value : 0.9048
##          Neg Pred Value : 0.7222
##              Prevalence : 0.7167
##          Detection Rate : 0.6333
##    Detection Prevalence : 0.7000
##       Balanced Accuracy : 0.8242
##
##        'Positive' Class : Cad
##
```

```
# store the ROC and Testing accuracy to the results table
result$ROC[10] = max(gbm$results$ROC)
result$Testing_Accuracy[10] = cm.gbm$overall[1]

# Save the testing ROC for plotting later
gbm.probs = predict(gbm,svm.test[,!names(svm.test) %in% c("Cath")],type = "prob")
gbm.ROC = roc(response = svm.test$Cath,
              predictor = gbm.probs$Cad,
              levels = levels(svm.test$Cath),
              percent = T)
result$Test_AUC[10] = gbm.ROC$auc
```

3.11 Decision Tree

```
set.seed(3164)
# train the decision tree
dt <- train(Cath ~., data = svm.train,
                 method = "C5.0",
                 trControl = control,
                 verbose = FALSE,
                 preProc = c("center", "scale"),
                 tuneLength = 10,
                 metric = "ROC")
print(dt)
```

```
## C5.0
```

```
##
## 243 samples
##  12 predictor
##   2 classes: 'Cad', 'Normal'
##
## Pre-processing: centered (15), scaled (15)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 219, 219, 218, 219, 219, 218, ...
## Resampling results across tuning parameters:
##
##   model  winnow  trials  ROC        Sens       Spec
##   rules  FALSE    1      0.8246032  0.8885294  0.6857143
##   rules  FALSE   10      0.8984174  0.9158497  0.6685714
##   rules  FALSE   20      0.9100373  0.9163072  0.7014286
##   rules  FALSE   30      0.9118254  0.9144444  0.7100000
##   rules  FALSE   40      0.9131839  0.9144771  0.7085714
##   rules  FALSE   50      0.9132586  0.9157190  0.7057143
##   rules  FALSE   60      0.9143184  0.9157190  0.7171429
##   rules  FALSE   70      0.9135481  0.9157516  0.7171429
##   rules  FALSE   80      0.9142157  0.9162745  0.7114286
##   rules  FALSE   90      0.9149813  0.9156536  0.7157143
##   rules   TRUE    1      0.8276517  0.8772876  0.6785714
##   rules   TRUE   10      0.8701774  0.9000000  0.6714286
##   rules   TRUE   20      0.8764776  0.8902614  0.6900000
##   rules   TRUE   30      0.8781816  0.8884967  0.6900000
##   rules   TRUE   40      0.8780182  0.8885621  0.6957143
##   rules   TRUE   50      0.8793347  0.8896078  0.6957143
##   rules   TRUE   60      0.8799977  0.8879085  0.6957143
##   rules   TRUE   70      0.8799276  0.8890850  0.6971429
##   rules   TRUE   80      0.8818231  0.8884967  0.6957143
##   rules   TRUE   90      0.8813329  0.8884967  0.6985714
##   tree   FALSE    1      0.8182493  0.8884967  0.6771429
##   tree   FALSE   10      0.8977848  0.9040850  0.6871429
##   tree   FALSE   20      0.9047806  0.9086928  0.7100000
##   tree   FALSE   30      0.9068161  0.9064052  0.7071429
##   tree   FALSE   40      0.9090803  0.9041503  0.7171429
##   tree   FALSE   50      0.9114099  0.9063725  0.7171429
##   tree   FALSE   60      0.9103315  0.9064706  0.7100000
##   tree   FALSE   70      0.9117507  0.9064706  0.7142857
##   tree   FALSE   80      0.9124416  0.9094444  0.7200000
##   tree   FALSE   90      0.9126284  0.9065033  0.7157143
##   tree    TRUE    1      0.8200980  0.8756209  0.6814286
##   tree    TRUE   10      0.8711438  0.8913725  0.6914286
##   tree    TRUE   20      0.8765033  0.8872876  0.7000000
##   tree    TRUE   30      0.8789309  0.8872876  0.7071429
##   tree    TRUE   40      0.8789799  0.8843791  0.7085714
##   tree    TRUE   50      0.8785341  0.8837582  0.7100000
##   tree    TRUE   60      0.8787068  0.8837908  0.7085714
##   tree    TRUE   70      0.8796125  0.8826471  0.7057143
##   tree    TRUE   80      0.8797899  0.8832353  0.7057143
##   tree    TRUE   90      0.8795472  0.8849346  0.7100000
##
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were trials = 90, model = rules and
```

```
##  winnow = FALSE.
```

```r
# Test the decision tree
dt.predict = predict(dt, svm.test)
# confusion matrix
cm.dt <- confusionMatrix(dt.predict, svm.test$Cath)
print(cm.dt)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Cad Normal
##     Cad     39      3
##     Normal   4     14
##
##                Accuracy : 0.8833
##                  95% CI : (0.7743, 0.9518)
##     No Information Rate : 0.7167
##     P-Value [Acc > NIR] : 0.001754
##
##                   Kappa : 0.7177
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.9070
##             Specificity : 0.8235
##          Pos Pred Value : 0.9286
##          Neg Pred Value : 0.7778
##              Prevalence : 0.7167
##          Detection Rate : 0.6500
##    Detection Prevalence : 0.7000
##       Balanced Accuracy : 0.8653
##
##        'Positive' Class : Cad
##
```

```r
# store the ROC and Testing accuracy to the results table
result$ROC[11] = max(dt$results$ROC)
result$Testing_Accuracy[11] = cm.dt$overall[1]

# Save the testing ROC for plotting later
dt.probs = predict(dt,svm.test[,!names(svm.test) %in% c("Cath")],type = "prob")
dt.ROC = roc(response = svm.test$Cath,
              predictor = dt.probs$Cad,
              levels = levels(svm.test$Cath),
              percent = T)
result$Test_AUC[11] = dt.ROC$auc
```

3.12 AdaBoost Classification tree

```
set.seed(3164)
# train adaBoost Classification Tree
adaB = train(Cath ~ ., data = svm.train,
            method = "AdaBoost.M1",
            tuneGrid = data.frame(mfinal = (1:3)*5, maxdepth = c(5,5,5), coeflearn = c("Breiman", "Freun
            preProcess = c("scale", "center"),
            na.action = na.omit)
print(adaB)
```

```
## AdaBoost.M1
##
## 243 samples
##  12 predictor
##   2 classes: 'Cad', 'Normal'
##
## Pre-processing: scaled (15), centered (15)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 243, 243, 243, 243, 243, 243, ...
## Resampling results across tuning parameters:
##
##   coeflearn  mfinal  Accuracy   Kappa
##   Breiman     5      0.8049827  0.5145849
##   Freund     10      0.7937542  0.4847262
##   Zhu        15      0.7944090  0.4817667
##
## Tuning parameter 'maxdepth' was held constant at a value of 5
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were mfinal = 5, maxdepth = 5 and
##  coeflearn = Breiman.
```

```
# Testing AdaBoost Classification tree
adab.tree.predict = predict(adaB, svm.test)
# confusion matrix
cm.adab.tree <- confusionMatrix(adab.tree.predict, svm.test$Cath)
print(cm.adab.tree)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Cad Normal
##     Cad     38      4
##     Normal   5     13
##
##                Accuracy : 0.85
##                  95% CI : (0.7343, 0.929)
##     No Information Rate : 0.7167
##     P-Value [Acc > NIR] : 0.01221
##
##                   Kappa : 0.6371
##
##  Mcnemar's Test P-Value : 1.00000
##
```

```
##             Sensitivity : 0.8837
##             Specificity : 0.7647
##          Pos Pred Value : 0.9048
##          Neg Pred Value : 0.7222
##              Prevalence : 0.7167
##          Detection Rate : 0.6333
##    Detection Prevalence : 0.7000
##       Balanced Accuracy : 0.8242
##
##        'Positive' Class : Cad
##
```

```
# store the ROC and Testing accuracy to the results table
result$ROC[12] = paste(as.character(max(adaB$results$Accuracy)), "(Accuracy)", sep = " ")
result$Testing_Accuracy[12] = cm.adab.tree$overall[1]

# Save the testing ROC for plotting later
adab.probs = predict(adaB,svm.test[,!names(svm.test) %in% c("Cath")],type = "prob")
adab.ROC = roc(response = svm.test$Cath,
              predictor = adab.probs$Cad,
              levels = levels(svm.test$Cath),
              percent = T)
result$Test_AUC[12] = adab.ROC$auc
```

3.13 Boosted Logistic Regression

```
set.seed(3164)
# train Boosted Logistic Regression
blr = train(Cath ~ ., data = svm.train,
            method = "LogitBoost",
            tuneGrid = data.frame(nIter = c(5,10,20,50)),
            trControl = control,
            preProcess = c("scale", "center"),
            metric = "ROC",
            na.action = na.omit)
print(blr)
```

```
## Boosted Logistic Regression
##
## 243 samples
##  12 predictor
##   2 classes: 'Cad', 'Normal'
##
## Pre-processing: scaled (15), centered (15)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 219, 219, 218, 219, 219, 218, ...
## Resampling results across tuning parameters:
##
##   nIter  ROC        Sens       Spec
##    5     0.8263515  0.8718301  0.6014286
##   10     0.8794748  0.9193723  0.7221429
```

```
##   20     0.8829342  0.9162630  0.7043571
##   50     0.8844981  0.9043243  0.6862381
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was nIter = 50.
```

```
# Testing Boosted logistic regression
blr.predict = predict(blr, svm.test)
# confusion matrix
cm.blr <- confusionMatrix(blr.predict, svm.test$Cath)
print(cm.blr)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Cad Normal
##     Cad     34      4
##     Normal   5     11
##
##                Accuracy : 0.8333
##                  95% CI : (0.7071, 0.9208)
##     No Information Rate : 0.7222
##     P-Value [Acc > NIR] : 0.04242
##
##                   Kappa : 0.593
##
##  Mcnemar's Test P-Value : 1.00000
##
##             Sensitivity : 0.8718
##             Specificity : 0.7333
##          Pos Pred Value : 0.8947
##          Neg Pred Value : 0.6875
##              Prevalence : 0.7222
##          Detection Rate : 0.6296
##    Detection Prevalence : 0.7037
##       Balanced Accuracy : 0.8026
##
##        'Positive' Class : Cad
##
```

```
# store the ROC and Testing accuracy to the results table
result$ROC[13] = max(blr$results$ROC)
result$Testing_Accuracy[13] = cm.blr$overall[1]

# Save the testing ROC for plotting later
blr.probs = predict(blr,svm.test[,!names(svm.test) %in% c("Cath")],type = "prob")
blr.ROC = roc(response = svm.test$Cath,
              predictor = blr.probs$Cad,
              levels = levels(svm.test$Cath),
              percent = T)
result$Test_AUC[13] = blr.ROC$auc
```

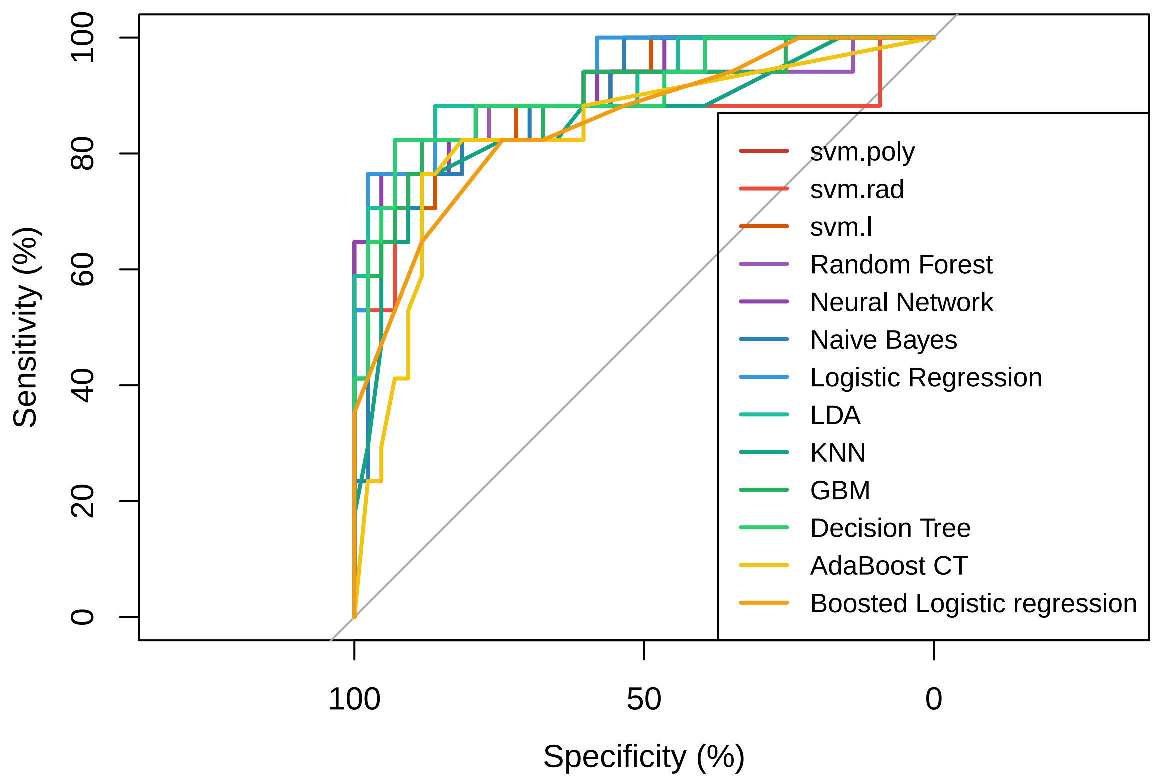
4 Review the Results and Plotting the ROC

```r
# view the results
print(result)
```

```
##                      Classifier Training_ROC Testing_Accuracy Test_AUC
## 1                      SVM.Poly            0        0.8666667 90.69767
## 2                    SVM.Radial            1        0.8333333 85.36252
## 3                    SVM.Linear            2        0.9000000 90.69767
## 4                 Random Forest            3        0.8666667 89.60328
## 5                Neurual Network           4        0.8333333 91.79207
## 6                   Naive Bayes            5        0.6666667 90.01368
## 7           Logistic Regression            6        0.8666667 92.61286
## 8                           LDA            7        0.8666667 91.51847
## 9                           KNN            8        0.8333333 85.49932
## 10                          GBM            9        0.8500000 89.05609
## 11                Decision Tree           10        0.8833333 90.42408
## 12 AdaBoost Classification Tree           11        0.8500000 83.51573
## 13  Boosted Logistic Regression           12        0.8333333 85.22572
##                             ROC
## 1             0.915214752567694
## 2             0.904855275443511
## 3             0.915214752567694
## 4             0.921577964519141
## 5              0.92078431372549
## 6             0.888113912231559
## 7             0.916615312791783
## 8             0.923487394957983
## 9             0.900873015873016
## 10            0.925849673202614
## 11            0.914981325863679
## 12 0.804982729619275 (Accuracy)
## 13            0.884498132586368
```

```r
# ploting the ROC
plot(svm.poly.ROC,type = "S",col = "#C0392B")
plot(svm.rad.ROC,add = TRUE,col = "#E74C3C")
plot(svm.l.ROC,add = TRUE,col = "#D35400")
plot(rf.ROC,add = TRUE,col = "#9B59B6")
plot(nn.ROC,add = TRUE,col = "#8E44AD")
plot(nb.ROC,add = TRUE,col = "#2980B9")
plot(lr.ROC,add = TRUE,col = "#3498DB")
plot(lda.ROC,add = TRUE,col = "#1ABC9C")
plot(knn.ROC,add = TRUE,col = "#16A085")
plot(gbm.ROC,add = TRUE,col = "#27AE60")
plot(dt.ROC,add = TRUE,col = "#2ECC71")
plot(adab.ROC,add = TRUE,col = "#F1C40F")
plot(blr.ROC,add = TRUE,col = "#F39C12")

legend("bottomright", legend = c("svm.poly","svm.rad", "svm.l", "Random Forest", "Neural Network", "Nai
                                 "Logistic Regression", "LDA", "KNN", "GBM", "Decision Tree", "AdaBoost
       col = c("#C0392B", "#E74C3C", "#D35400", "#9B59B6", "#8E44AD", "#2980B9", "#3498DB", "#1ABC9C",
               "#F1C40F", "#F39C12"),lwd = 2, cex = 0.8)
```



5 Combining the models

```
# generate a dataframe that contains the prediction results of all models
generate_ensemble_df <- function(caddataset){
  #Input: caddataset - the training dataset
  #Output: a data frame of size (nrow x 12). Each feature (column) is comprised of the predictions made
  #Runtime: Linear

  aggregate_pred.df <- as.data.frame(caddataset$Cath)
  colnames(aggregate_pred.df)=c("Cath")
  #attach predictions
  aggregate_pred.df$knnres <-  predict(knn, caddataset, type = "prob")$Cad
  aggregate_pred.df$ldares <-  predict(lda, caddataset, type = "prob")$Cad
  aggregate_pred.df$lrres <-  predict(lr, caddataset, type = "prob")$Cad
  aggregate_pred.df$rfres <-  predict(rf, caddataset, type = "prob")$Cad
  aggregate_pred.df$svmLres <-  predict(svm.l, caddataset, type = "prob")$Cad
  aggregate_pred.df$svmPres <-  predict(svm.poly, caddataset, type = "prob")$Cad
  aggregate_pred.df$svmRres <-  predict(svm.rad, caddataset, type = "prob")$Cad
  aggregate_pred.df$NNres <-  predict(nn, caddataset, type = "prob")$Cad
  aggregate_pred.df$GBMres <-  predict(gbm, caddataset, type = "prob")$Cad
  aggregate_pred.df$AdaBres <-  predict(adaB, caddataset, type = "prob")$Cad
  aggregate_pred.df$dtres <-  predict(dt, caddataset, type = "prob")$Cad
  aggregate_pred.df$blrres <-  predict(blr, caddataset, type = "prob")$Cad
  return(aggregate_pred.df)
}
```

5.1 Naive Voting Ensemble

```
vote_ensemble <- function(dataset, label="Cath"){
  #Input: dataset - Any data set.
  #Input label - A column name to predict values for.
  #Output: a vector containing the average value of all features for each input row.
  #Converts Y or N in input df to 1 and 0 respectively.
  #To be used to take a unweighted vote of columns, aka. vote ensembling when combined with generate_en

  df = dataset[,names(dataset) != c(label)]
  num = dim(df)[2]
  vote = apply(df, 1, function(x) sum(as.numeric(x)))/num
  return(as.factor(ifelse(round(vote) == 0,"Normal","Cad")))
}
```

```
ensem_result <- vote_ensemble(generate_ensemble_df(svm.train))
confusionMatrix(ensem_result,svm.train$Cath)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Cad Normal
##     Cad    163      8
##     Normal  10     62
##
##                Accuracy : 0.9259
##                  95% CI : (0.8855, 0.9555)
##     No Information Rate : 0.7119
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8209
##
##  Mcnemar's Test P-Value : 0.8137
##
##             Sensitivity : 0.9422
##             Specificity : 0.8857
##          Pos Pred Value : 0.9532
##          Neg Pred Value : 0.8611
##              Prevalence : 0.7119
##          Detection Rate : 0.6708
##    Detection Prevalence : 0.7037
##       Balanced Accuracy : 0.9140
##
##        'Positive' Class : Cad
##
```

```
ensem_result_test <- vote_ensemble(generate_ensemble_df(svm.test))
confusionMatrix(ensem_result_test,svm.test$Cath)
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction Cad Normal
##     Cad     39      4
##     Normal   4     13
##
##                Accuracy : 0.8667
##                  95% CI : (0.7541, 0.9406)
##     No Information Rate : 0.7167
##     P-Value [Acc > NIR] : 0.004937
##
##                   Kappa : 0.6717
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.9070
##             Specificity : 0.7647
##          Pos Pred Value : 0.9070
##          Neg Pred Value : 0.7647
##              Prevalence : 0.7167
##          Detection Rate : 0.6500
##    Detection Prevalence : 0.7167
##       Balanced Accuracy : 0.8358
##
##        'Positive' Class : Cad
##
```

5.2 Train logistic regression on result

```
set.seed(3164)
ensem_train <- generate_ensemble_df(svm.train)
ensem_test <- generate_ensemble_df(svm.test)

# set up 10 cross validation
control <- trainControl(method="repeatedcv",
                        number=10,
                        classProbs = TRUE,
                        summaryFunction = twoClassSummary)

# train logistic regression on result
lr_ensem<-train(Cath ~., data = ensem_train,
                method="glm",
                family = "binomial",
                trControl=control,
                metric = "ROC")

# testing the logistic regression model
ensem_lr_test=predict(lr_ensem, ensem_test)
cm.lr_ensem = confusionMatrix(ensem_lr_test, svm.test$Cath)
cm.lr_ensem
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction Cad Normal
##     Cad     39      3
##     Normal   4     14
##
##                Accuracy : 0.8833
##                  95% CI : (0.7743, 0.9518)
##     No Information Rate : 0.7167
##     P-Value [Acc > NIR] : 0.001754
##
##                   Kappa : 0.7177
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.9070
##             Specificity : 0.8235
##          Pos Pred Value : 0.9286
##          Neg Pred Value : 0.7778
##              Prevalence : 0.7167
##          Detection Rate : 0.6500
##    Detection Prevalence : 0.7000
##       Balanced Accuracy : 0.8653
##
##        'Positive' Class : Cad
##
```

```r
# ROC and Testing accuracy to the results table
print("The training ROC is:", lr_ensem$results$ROC)
```

```
## [1] "The training ROC is:"
```

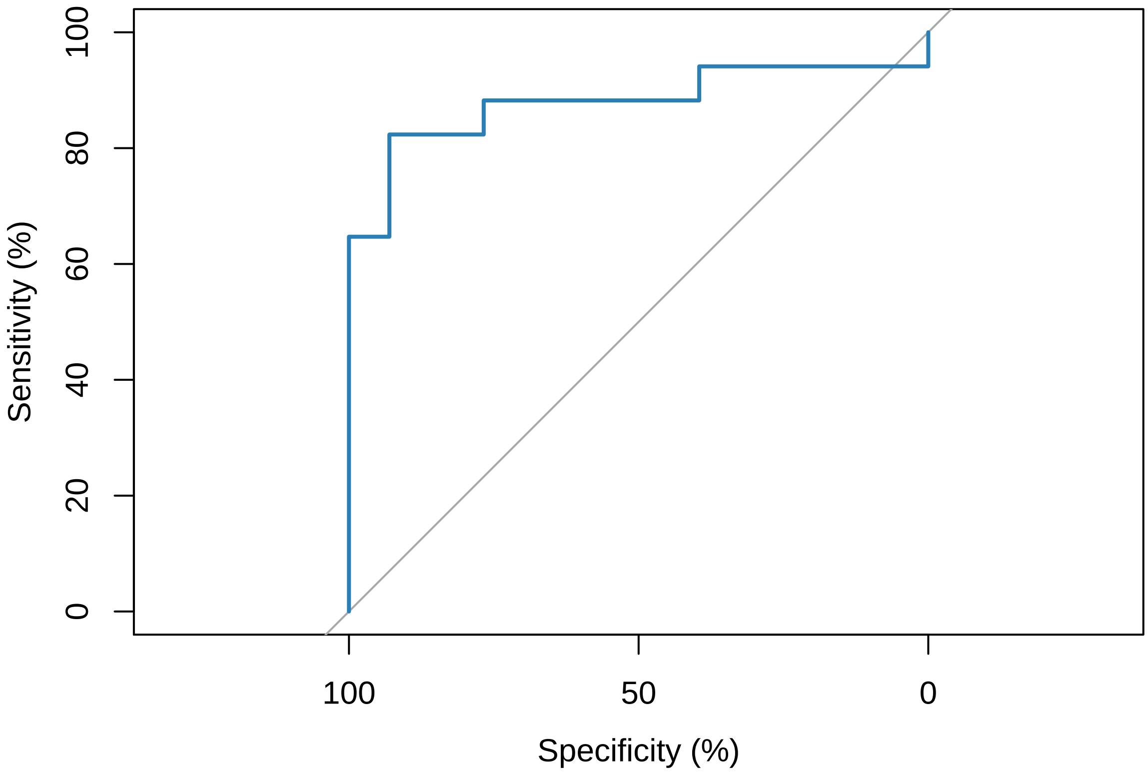
```r
print("The Testing Accuracy is:", cm.lr_ensem$overall[1])
```

```
## [1] "The Testing Accuracy is:"
```

```r
# Save the testing ROC for plotting later
lr.ensem.probs = predict(lr_ensem, ensem_test[,2:length(ensem_test)],type = "prob")
lr.ensem.ROC = roc(response = svm.test$Cath,
              predictor = lr.ensem.probs$Cad,
              levels = levels(svm.test$Cath),
              percent = T)
cat("The test AUC is: ", as.character(lr.ensem.ROC$auc))
```

```
## The test AUC is:  88.8508891928865
```

```r
# Plot the Testing ROC curve
plot(lr.ensem.ROC,type = "S",col = "#2980B9")
```



100
80
60
40
20
0
Sensitivity (%)
100
50
0
Specificity (%)