Equity Premium - Prediction

Druce Vertes

July 2017

Equity Premium Prediction in R

- Using dataset from Prof. Amit Goyal, attempt to predict quarterly equity outperformance based on fundamental data like interest rates, valuation.
- This should be viewed as exploratory data analysis and a demo of R regression and classification using caret.
- No predictive value from regression.
- Binary classification predicting whether next quarter's equity premium will be above or below long term average is more successful, with some machine learning classifiers achieving > 60% accuracy out of sample.

Initialize

- Libraries to use

```
options(java.parameters='Xmx5g')
library(plyr)
library(reshape2)
library(lattice)
library(ggplot2)
library(MASS)
library(caret)
library(mlbench)
library(rpart)
library(boot)
```

```
##
## Attaching package: 'boot'
## The following object is masked from 'package:lattice':
##
##     melanoma
```

Import and clean data

- Import from CSV

```
setwd("C:/Users/druce/R/EquityPremium2017")
data<-read.csv('PredictorData2016q.csv',na.strings = c("NA","#DIV/0!", "","NaN"))
```

Clean... Trim NA valued columns

```r
countMissing <- function(mycol) {
  return (sum(is.na(data[, mycol]))/ nrow(data))
}
countNAs <- data.frame(countNA=sapply(colnames(data), countMissing))
subset(countNAs, countNAs$countNA > 0.5)
```

```
##       countNA
## cay 0.5547945
## csp 0.5496575
## ik  0.5205479
## D3  0.8013699
```

```r
colsToDeleteNA <- countNAs$countNA > 0.5
data <- data[, !colsToDeleteNA]
```

Clean... Trim NA valued rows

```r
rowsToDelete <- data$yyyyq <= 19254
data <- data[!rowsToDelete,]
```

Add EqPrem column, numeric date column for charts

```r
data$EqPrem = data$CRSP_SPvw - data$Rfree
data$numdate = as.numeric(substring(data$yyyyq, 1,4))+as.numeric(substring(data$yyyyq, 5,5))/4
```

```r
# functions to do leads and lags
mylag <- function(v, n){
  c(rep(NA, n),v[(seq(length(v)-n))])
}

mylead <- function(v, n){
  c(v[-n], rep(NA, n))
}

data$EqPrem = mylead(data$EqPrem,1)
```

Create a big data frame including all predictors, first diffs lagged up to 2 quarters

```r
#keep 12 predictors plus EqPrem
# truncate last quarter, no EqPrem to predict
data2 <- data[1:359,c("D12","E12","b.m","tbl","AAA","BAA","lty","ntis","infl","ltr","corpr","svar","EqP

# use trailing 1 year inflation
# should really do cum product of 1+infl , 70s/80s compounding would have made small difference
rsum.cumsum <- function(x, n = 4L) {
  tail(cumsum(x) - cumsum(c(rep(0, n), head(x, -n))), -n + 1)
}

# use real long term yields sted nominal
data2$infl <- tail(c(rep(NA,3), rsum.cumsum(data$infl)), 359)
```

```r
data2$AAA <- data2$AAA - data2$infl
data2$BAA <- data2$BAA - data2$infl
data2$lty <- data2$lty - data2$infl

# compute first diffs
diffs <- tail(data2, -1) - head(data2, -1)
diffs <- diffs[complete.cases(diffs),]

# truncate oldest 2 qs, no trailing diffs
bigdata <- tail(data2,-2)

# truncate oldest q
diffs <- tail(diffs,-1)
diffs <- diffs[,c("D12","E12","b.m", "tbl","AAA","BAA","lty","ntis","infl","ltr","corpr","svar")]
names(diffs)<-c("D12.diff","E12.diff","b.m.diff","tbl.diff","AAA.diff","BAA.diff","lty.diff","ntis.diff
bigdata=merge(bigdata, diffs,by=0)

# add previous quarter's 1st diff for tbl, AAA, BAA, lty, ltr, corpr
# compute first diffs
diffs <- tail(data2, -1) - head(data2, -1)
diffs <- head(diffs, -1)
diffs <- diffs[,c("tbl","AAA","BAA","lty","ltr","corpr")]
names(diffs)<-c("tbl.lagdiff","AAA.lagdiff","BAA.lagdiff","lty.lagdiff","ltr.lagdiff","corpr.lagdiff")
bigdata$rownums=1:nrow(bigdata)
diffs$rownums=1:nrow(diffs)
bigdata=merge(bigdata, diffs,by="rownums")

colsToDelete = names(bigdata) %in% c("Row.names", "rownums")
bigdata <- bigdata[,!colsToDelete]

# truncate oldest 2q, no ntis diff
bigdata <- tail(bigdata,-2)
```

Run models

Run a linear model

```r
fit <- lm(EqPrem~., data=bigdata)
summary(fit) # show results
```

```
##
## Call:
## lm(formula = EqPrem ~ ., data = bigdata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.37364 -0.04684  0.00633  0.05215  0.66648
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.0009795  0.0299087  -0.033  0.97390
```

```
## D12            0.0037751  0.0030161   1.252  0.21160
## E12           -0.0017093  0.0012399  -1.379  0.16898
## b.m            0.0991786  0.0385382   2.574  0.01052 *
## tbl            0.2906472  0.6362728   0.457  0.64813
## AAA            3.1659295  3.8239441   0.828  0.40833
## BAA           -1.9822910  1.8930672  -1.047  0.29583
## lty           -2.1357846  2.4266374  -0.880  0.37944
## ntis          -0.8521807  0.3465108  -2.459  0.01445 *
## infl          -0.8739380  0.7394369  -1.182  0.23812
## ltr           -0.2771483  0.6073886  -0.456  0.64849
## corpr          1.4604838  0.5758054   2.536  0.01167 *
## svar           0.7372439  0.7543373   0.977  0.32914
## D12.diff       0.0408510  0.0354457   1.152  0.24997
## E12.diff       0.0085082  0.0027303   3.116  0.00200 **
## b.m.diff       0.1104591  0.0894700   1.235  0.21788
## tbl.diff      -0.8878974  0.9566036  -0.928  0.35401
## AAA.diff       3.3849598  5.1999998   0.651  0.51554
## BAA.diff       5.2456596  2.7342694   1.918  0.05594 .
## lty.diff      -2.1819500  5.4468046  -0.401  0.68899
## ntis.diff     -0.0264819  0.7725883  -0.034  0.97268
## infl.diff      6.7678900  5.1953702   1.303  0.19362
## ltr.diff      -0.0655907  0.3043687  -0.215  0.82952
## corpr.diff    -0.4019124  0.3403049  -1.181  0.23846
## svar.diff     -0.4045074  0.6306586  -0.641  0.52172
## tbl.lagdiff   -0.4285723  0.8182731  -0.524  0.60081
## AAA.lagdiff    8.8501413  4.1469635   2.134  0.03359 *
## BAA.lagdiff   -5.3904002  1.9800521  -2.722  0.00684 **
## lty.lagdiff   -3.4549028  3.2807786  -1.053  0.29310
## ltr.lagdiff   -0.0572854  0.2152449  -0.266  0.79030
## corpr.lagdiff -0.0143397  0.2155922  -0.067  0.94701
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1071 on 321 degrees of freedom
## Multiple R-squared:  0.1877, Adjusted R-squared:  0.1118
## F-statistic: 2.473 on 30 and 321 DF,  p-value: 5.527e-05
```

```
#plot(fit)
```

Run a stepwise regression for variable selection

```
library(MASS)

step <- stepAIC(fit, direction="both")
```

```
step$anova # display results
```

```
## Stepwise Model Path
## Analysis of Deviance Table
##
## Initial Model:
## EqPrem ~ D12 + E12 + b.m + tbl + AAA + BAA + lty + ntis + infl +
##     ltr + corpr + svar + D12.diff + E12.diff + b.m.diff + tbl.diff +
```

```
##     AAA.diff + BAA.diff + lty.diff + ntis.diff + infl.diff +
##     ltr.diff + corpr.diff + svar.diff + tbl.lagdiff + AAA.lagdiff +
##     BAA.lagdiff + lty.lagdiff + ltr.lagdiff + corpr.lagdiff
##
## Final Model:
## EqPrem ~ b.m + lty + ntis + infl + corpr + E12.diff + BAA.diff +
##     infl.diff + corpr.diff + AAA.lagdiff + BAA.lagdiff
##
##
##              Step Df     Deviance Resid. Df Resid. Dev       AIC
## 1                                        321   3.681286 -1543.250
## 2      - ntis.diff  1 1.347400e-05       322   3.681300 -1545.249
## 3  - corpr.lagdiff  1 5.471588e-05       323   3.681354 -1547.243
## 4       - ltr.diff  1 5.282828e-04       324   3.681883 -1549.193
## 5       - lty.diff  1 1.665440e-03       325   3.683548 -1551.034
## 6            - tbl  1 2.479686e-03       326   3.686028 -1552.797
## 7    - tbl.lagdiff  1 2.225238e-03       327   3.688253 -1554.584
## 8    - ltr.lagdiff  1 2.185306e-03       328   3.690438 -1556.376
## 9       - AAA.diff  1 3.402517e-03       329   3.693841 -1558.051
## 10     - svar.diff  1 5.207370e-03       330   3.699048 -1559.556
## 11      - tbl.diff  1 4.996482e-03       331   3.704045 -1561.080
## 12          - svar  1 8.004671e-03       332   3.712049 -1562.321
## 13           - ltr  1 6.524716e-03       333   3.718574 -1563.702
## 14   - lty.lagdiff  1 6.450171e-03       334   3.725024 -1565.092
## 15      - D12.diff  1 1.006347e-02       335   3.735088 -1566.143
## 16           - BAA  1 1.243909e-02       336   3.747527 -1566.972
## 17           - AAA  1 1.772375e-03       337   3.749299 -1568.806
## 18      - b.m.diff  1 1.438072e-02       338   3.763680 -1569.458
## 19           - E12  1 1.728183e-02       339   3.780962 -1569.846
## 20           - D12  1 4.810846e-03       340   3.785773 -1571.398
```

Run a model, with just the useful predictors

Slightly lower R-squared, higher adjusted R-squared

```r
fit2<-lm(EqPrem ~ D12 + E12 + b.m + AAA + BAA + ntis + infl + corpr +
    E12.diff + BAA.diff + infl.diff + corpr.diff + AAA.lagdiff +
    BAA.lagdiff, data=bigdata)
summary(fit2) # show results
```

```
##
## Call:
## lm(formula = EqPrem ~ D12 + E12 + b.m + AAA + BAA + ntis + infl +
##     corpr + E12.diff + BAA.diff + infl.diff + corpr.diff + AAA.lagdiff +
##     BAA.lagdiff, data = bigdata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.39058 -0.04618  0.00521  0.04981  0.67198
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.0136926  0.0269917  -0.507  0.61228
```

```
## D12          0.0034137  0.0021893   1.559  0.11987    
## E12         -0.0009793  0.0007446  -1.315  0.18933    
## b.m          0.1109008  0.0354543   3.128  0.00191 ** 
## AAA         -0.1095278  1.2058290  -0.091  0.92768    
## BAA         -0.6228837  1.1576973  -0.538  0.59091    
## ntis        -0.6481864  0.2947502  -2.199  0.02855 *  
## infl        -0.6995310  0.2521236  -2.775  0.00584 ** 
## corpr        1.2560077  0.2610201   4.812 2.26e-06 ***
## E12.diff     0.0064898  0.0019667   3.300  0.00107 ** 
## BAA.diff     6.9506101  1.5986165   4.348 1.82e-05 ***
## infl.diff    7.3477536  1.6875288   4.354 1.77e-05 ***
## corpr.diff  -0.4406723  0.1409287  -3.127  0.00192 ** 
## AAA.lagdiff  4.5744270  1.8220242   2.511  0.01252 *  
## BAA.lagdiff -4.6491075  1.8146445  -2.562  0.01084 *  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.1057 on 337 degrees of freedom
## Multiple R-squared:  0.169,  Adjusted R-squared:  0.1345 
## F-statistic: 4.897 on 14 and 337 DF,  p-value: 2.84e-08
```

```r
#plot(fit2)
```

Run an out of sample test

TODO: don't select the variables using the whole set, which is bad practice/cheating

```r
# test set v. training set
# sample(1000,1)
set.seed(710)

trainindex <- sample(nrow(bigdata), trunc(nrow(bigdata)*.75))
trainingset <- bigdata[trainindex,]
testset <- bigdata[-trainindex, ]

fit3<-lm(EqPrem ~ D12 + E12 + b.m + AAA + BAA + ntis + infl + corpr +
    E12.diff + BAA.diff + infl.diff + corpr.diff + AAA.lagdiff +
    BAA.lagdiff, data=trainingset)
summary(fit3) # show results
```

```
## 
## Call:
## lm(formula = EqPrem ~ D12 + E12 + b.m + AAA + BAA + ntis + infl + 
##     corpr + E12.diff + BAA.diff + infl.diff + corpr.diff + AAA.lagdiff + 
##     BAA.lagdiff, data = trainingset)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -0.40279 -0.05135  0.00578  0.05055  0.49197 
## 
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0076413  0.0285657   0.267 0.789306
```

```
## D12          0.0051626  0.0025524   2.023 0.044179 *
## E12         -0.0016600  0.0008668  -1.915 0.056637 .
## b.m          0.1159292  0.0374576   3.095 0.002193 **
## AAA          1.0814256  1.3168452   0.821 0.412304
## BAA         -1.9362790  1.2517262  -1.547 0.123160
## ntis        -0.9318413  0.3168770  -2.941 0.003583 **
## infl        -0.9271829  0.2668676  -3.474 0.000604 ***
## corpr        1.5392040  0.2820778   5.457 1.17e-07 ***
## E12.diff     0.0150214  0.0028823   5.212 3.93e-07 ***
## BAA.diff     9.9555854  1.6706069   5.959 8.61e-09 ***
## infl.diff   10.3463154  1.7876559   5.788 2.14e-08 ***
## corpr.diff  -0.4671172  0.1580101  -2.956 0.003413 **
## AAA.lagdiff -2.9540555  2.5622188  -1.153 0.250045
## BAA.lagdiff  3.7257000  2.6472452   1.407 0.160560
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09848 on 249 degrees of freedom
## Multiple R-squared:  0.2409, Adjusted R-squared:  0.1982
## F-statistic: 5.645 on 14 and 249 DF,  p-value: 1.744e-09
```

```r
# R-squared goes up when all we did was reduce the sample size
# suggests overfitting

# in-sample RMSE
mdss <- function (var1, var2) {
  mean((var1 - var2)^2)
}

MSEis <- mdss(predict(fit3), trainingset$EqPrem)
print(sqrt(MSEis))
```

```
## [1] 0.09564563
```

```r
# in-sample population standard dev
print(sqrt(mdss(trainingset$EqPrem, mean(trainingset$EqPrem))))
```

```
## [1] 0.1097795
```

```r
# check vs. sd function
sqrt((sd(trainingset$EqPrem))^2 * (nrow(trainingset)-1) / nrow(trainingset))
```

```
## [1] 0.1097795
```

```r
# bigdata population standard dev
print(sqrt(mdss(bigdata$EqPrem, mean(bigdata$EqPrem))))
```

```
## [1] 0.113469
```

```r
# if out-of-sample RMSE is better than those we are probably predicting something

# in-sample MSE / Population Variance = R-squared  (as a check)
print(1- MSEis / mdss(trainingset$EqPrem, mean(trainingset$EqPrem)))
```

```
## [1] 0.2409198
```

```r
# out-of-sample RMSE
mypredict <- predict(fit3, newdata = testset)
```

```r
MSEos <- mdss(mypredict, testset$EqPrem)
print(sqrt(MSEos))
```

```
## [1] 0.148684
```

```r
# suppose we just used the mean of training set as predictor, RMSE would be
print(sqrt(mdss(testset$EqPrem, mean(trainingset$EqPrem))))
```

```
## [1] 0.1239497
```

```r
# our model predicts worse out-of-sample than just using the training set mean (!)
#print("out-of-sample MSE / Variance") # out-of-sample R-squared maybe different
#print("not sure what is correct out-of-sample R-squared formula but")
#print(1- MSEos / mean((testset$EqPrem - mean(trainingset$EqPrem))^2))
#print(1- MSEos / mean((testset$EqPrem - mean(testset$EqPrem))^2))

# leave one out cross-validation
# glm same as lm but supports cross-validation

glm.fit <- glm(EqPrem ~ D12 + E12 + b.m + AAA + BAA + ntis + infl + corpr +
             E12.diff + BAA.diff + infl.diff + corpr.diff + AAA.lagdiff +
             BAA.lagdiff, data=bigdata)
# same as fit2
summary(glm.fit)
```

```
##
## Call:
## glm(formula = EqPrem ~ D12 + E12 + b.m + AAA + BAA + ntis + infl +
##     corpr + E12.diff + BAA.diff + infl.diff + corpr.diff + AAA.lagdiff +
##     BAA.lagdiff, data = bigdata)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.39058  -0.04618   0.00521   0.04981   0.67198
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.0136926  0.0269917  -0.507  0.61228
## D12          0.0034137  0.0021893   1.559  0.11987
## E12         -0.0009793  0.0007446  -1.315  0.18933
## b.m          0.1109008  0.0354543   3.128  0.00191 **
## AAA         -0.1095278  1.2058290  -0.091  0.92768
## BAA         -0.6228837  1.1576973  -0.538  0.59091
## ntis        -0.6481864  0.2947502  -2.199  0.02855 *
## infl        -0.6995310  0.2521236  -2.775  0.00584 **
## corpr        1.2560077  0.2610201   4.812 2.26e-06 ***
## E12.diff     0.0064898  0.0019667   3.300  0.00107 **
## BAA.diff     6.9506101  1.5986165   4.348 1.82e-05 ***
## infl.diff    7.3477536  1.6875288   4.354 1.77e-05 ***
## corpr.diff  -0.4406723  0.1409287  -3.127  0.00192 **
## AAA.lagdiff  4.5744270  1.8220242   2.511  0.01252 *
## BAA.lagdiff -4.6491075  1.8146445  -2.562  0.01084 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for gaussian family taken to be 0.01117507)
##
##     Null deviance: 4.5321  on 351  degrees of freedom
## Residual deviance: 3.7660  on 337  degrees of freedom
## AIC: -566.31
##
## Number of Fisher Scoring iterations: 2
```

```r
cv.err <- cv.glm(bigdata, glm.fit)

print("Leave one out cross-validation RMSE")
```

```
## [1] "Leave one out cross-validation RMSE"
```

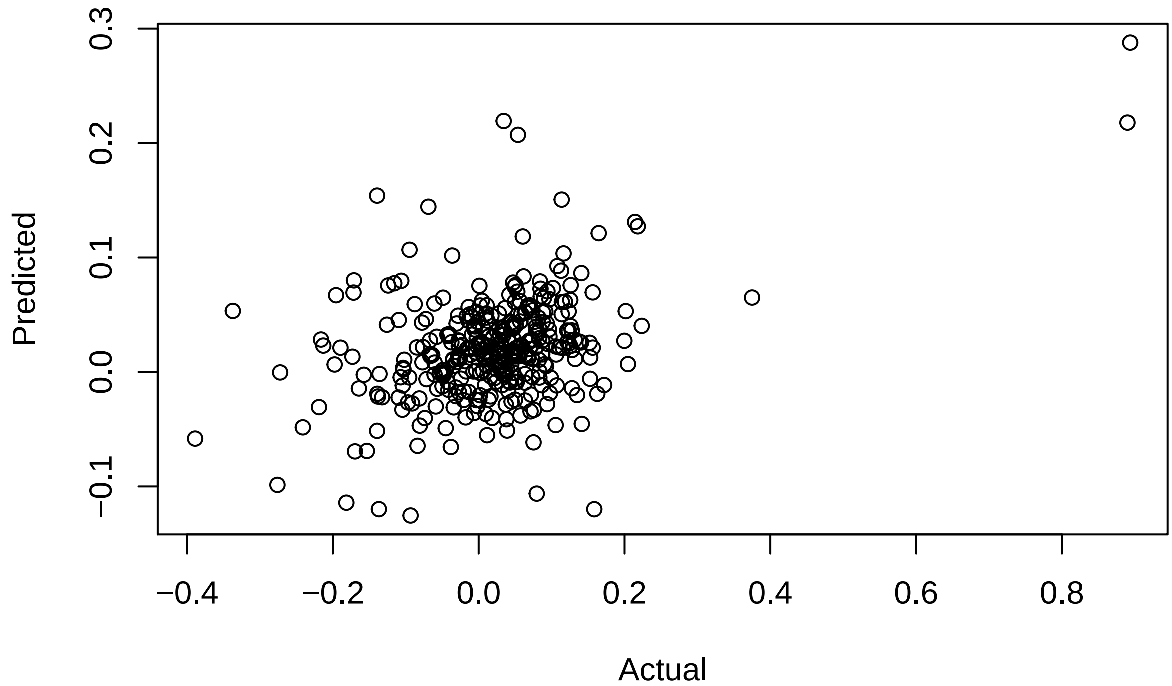
```r
MSEos <- cv.err$delta[1]
print(sqrt(MSEos))
```

```
## [1] 0.1176198
```

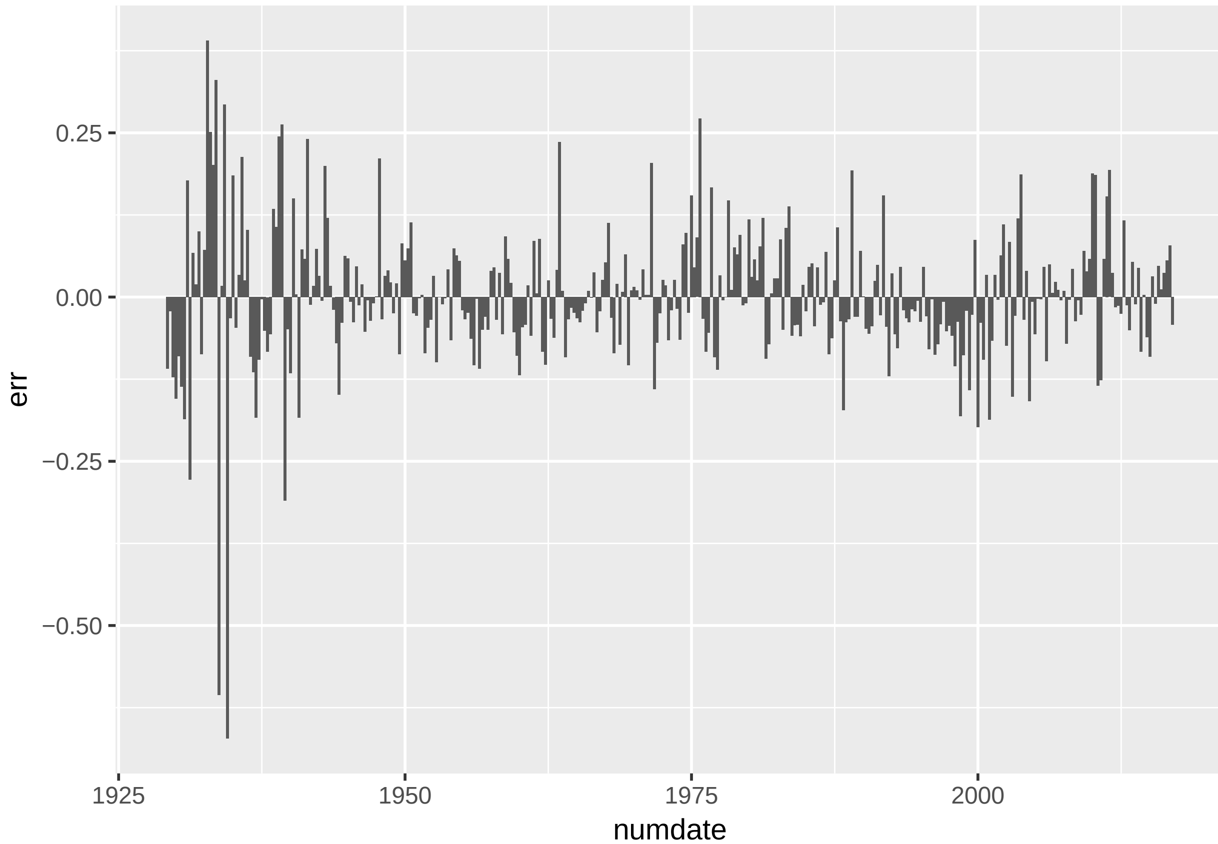
```r
# larger than in-sample RMSE which makes sense
# smaller than OOSE RMSE we found with training/test 75%/25% which makes sense
# smaller than RMSE we get just using the mean of the training set
# so, if you leave one out, estimate model on remainder, test on one you left out,
# error is a little smaller than just using a constant
# a wee bit but not much useful prediction going on

#print("LOOCV MSE / Variance") # out-of-sample R-squared maybe different
#print(1- MSEos / mean((bigdata$EqPrem - mean(bigdata$EqPrem))^2))
```

Plot predicted vs. actual

```r
# scatter plot
plotframe=data.frame(bigdata$EqPrem, fitted(fit2))
plot(plotframe, ylab="Predicted", xlab="Actual")
```




```
## error plot
plotframe$numdate <- tail(data$numdate, 352)
plotframe$err <- plotframe$fitted.fit2. - plotframe$bigdata.EqPrem
ggplot(data=plotframe, aes(x=numdate, y=err)) + geom_bar(stat="identity")
```




```
## bars since 1974
plotframe2 = plotframe[plotframe$numdate > 2000, c("bigdata.EqPrem", "fitted.fit2." ,  "numdate") ]
plotframe3 = melt(plotframe2,id="numdate")
ggplot(plotframe3, aes(x=numdate, y=value, fill=variable)) + geom_bar(stat="identity", position="dodge")
```

Run caret regression models

- Observe OOS RMSE with various nonlinear models v. linear model

```
library(frbs)
library(pls)
```

```
##
## Attaching package: 'pls'

## The following object is masked from 'package:caret':
##
##     R2

## The following object is masked from 'package:stats':
##
##     loadings
```

```
library(monomvn)
```

```
## Loading required package: lars

## Loaded lars 1.2
```

```
library(elasticnet)
library(foba)
library(fastICA)
```

```r
library(kernlab)
```

```
##
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```r
library(KRLS)
```

```
## ## KRLS Package for Kernel-based Regularized Least Squares.

## ## See Hainmueller and Hazlett (2014) for details.
```

```r
library(lars)
library(neuralnet)
library(nnls)
library(leaps)

# use same as before
trainingset <- bigdata[trainindex,]
testset <- bigdata[-trainindex, ]

# these returned valid values at one time, maybe a version hell situation, subsequently loaded package
# "lars" "lasso", "neuralnet", 'rqlasso', , 'superpc', , 'lasso', "krlsRadial", "krlsPoly", , "rlm"", '
# , "lmStepAIC" # this one just generates too much annoying output

regressionMethods <- c("lm", "enet", "leapBackward", "leapForward","leapSeq",
                       "nnls", "pcr", 'rvmLinear', 'rvmRadial', 'ridge'
                       )

regressionModels <- array(1:length(regressionMethods))
regressionTrainPredicts <- data.frame(row.names=row.names(trainingset))
regressionTestPredicts <- data.frame(row.names=row.names(testset))

print("Out of sample RMSE using various methods")
```

```
## [1] "Out of sample RMSE using various methods"
```

```r
# trc_cv = trainControl(method="cv")
i <- 0
for(mx in regressionMethods) {
  i <- i + 1
  print(mx)
  mymodel <- train(EqPrem ~ D12 + E12 + b.m + AAA + BAA + ntis + infl + corpr +
    E12.diff + BAA.diff + infl.diff + corpr.diff + AAA.lagdiff +
    BAA.lagdiff, data=trainingset, method=mx, preProc = c("center", "scale"), verbose=FALSE)

  mypredict <- predict(mymodel, newdata = testset)
  MSEos <- mdss(mypredict, testset$EqPrem)
  print(sqrt(MSEos))

  regressionModels[i] <- mymodel
  regressionTrainPredicts[, mx] <- predict(mymodel, newdata=trainingset)
  regressionTestPredicts[, mx] <- mypredict
```

```
}
```

```
## [1] "lm"
## [1] 0.148684
## [1] "enet"
## [1] 0.1223865
## [1] "leapBackward"
## [1] 0.1276375
## [1] "leapForward"
## [1] 0.1236289
## [1] "leapSeq"
## [1] 0.1236289
## [1] "nnls"
## [1] 0.1362005
## [1] "pcr"
## [1] 0.1233096
## [1] "rvmLinear"
## [1] 0.136446
## [1] "rvmRadial"
## [1] 0.1327488
## [1] "ridge"
## [1] 0.1485132
```

the nonlinear methods do better, sometimes significantly better

- note lm model has same OOS RMSE as we found earlier, all the others are smaller

```
mx <- 'leapBackward'
mymodel <- train(EqPrem ~ D12 + E12 + b.m + AAA + BAA + ntis + infl + corpr +
    E12.diff + BAA.diff + infl.diff + corpr.diff + AAA.lagdiff +
    BAA.lagdiff, data=trainingset, method=mx, preProc = c("center", "scale"), verbose=FALSE)
mypredict <- predict(mymodel, newdata = testset)

MSEos <- mdss(mypredict, testset$EqPrem)
print("Out of sample RMSE")
```

```
## [1] "Out of sample RMSE"
```

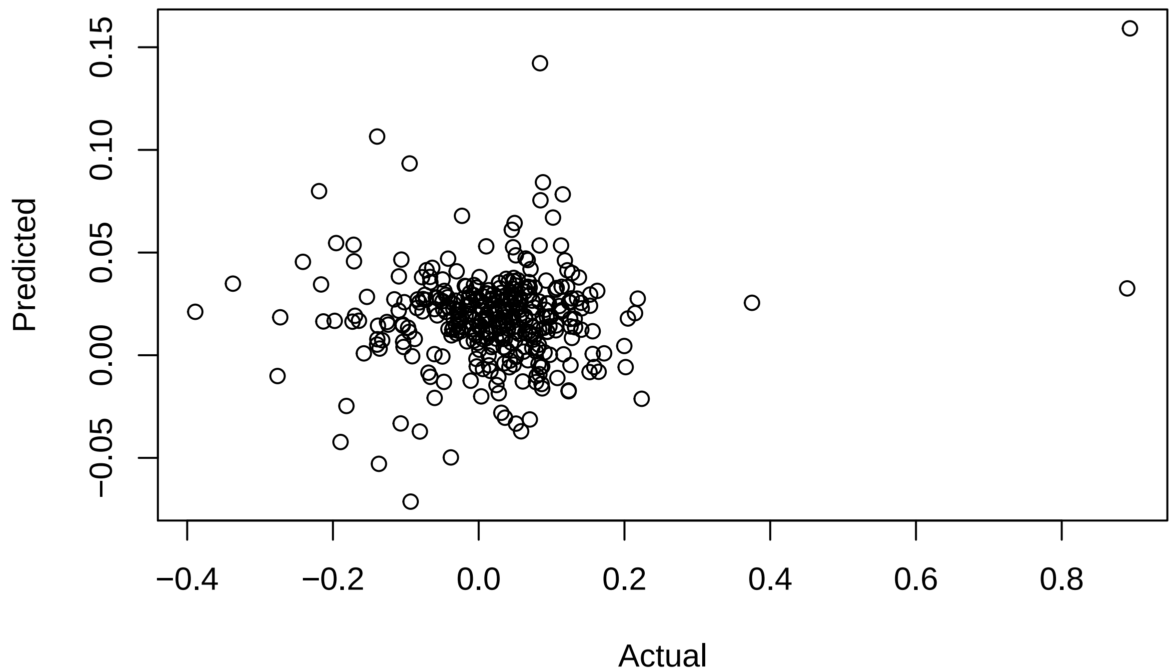
```
print(sqrt(MSEos))
```

```
## [1] 0.1276375
```

```
# suppose we just used the mean of training set as predictor, RMSE would be
print(sqrt(mdss(mean(trainingset$EqPrem), testset$EqPrem)))
```

```
## [1] 0.1239497
```

```
#print(1- MSEos / mean((testset$EqPrem - mean(trainingset$EqPrem))^2))
#print(1- MSEos / mean((testset$EqPrem - mean(testset$EqPrem))^2))

plotframe <- data.frame(bigdata$EqPrem, predict(mymodel, newdata = bigdata))

plot(plotframe, ylab="Predicted", xlab="Actual")
```



not good but at least a little more predictive than using the mean or linear model

```r
# try preprocessing with PCA

print("Out of sample RMSE using various methods")
```

```
## [1] "Out of sample RMSE using various methods"
```

```r
for(mx in regressionMethods) {

  trc_cv = trainControl(method="cv")

  print(mx)
  mymodel <- train(EqPrem ~ ., data=trainingset, method=mx, preProc = c("center", "scale", "pca"),
                   verbose=FALSE)
  mypredict <- predict(mymodel, newdata = testset)
  MSEos <- mean((mypredict - testset$EqPrem)^2)
  print(sqrt(MSEos))
}
```

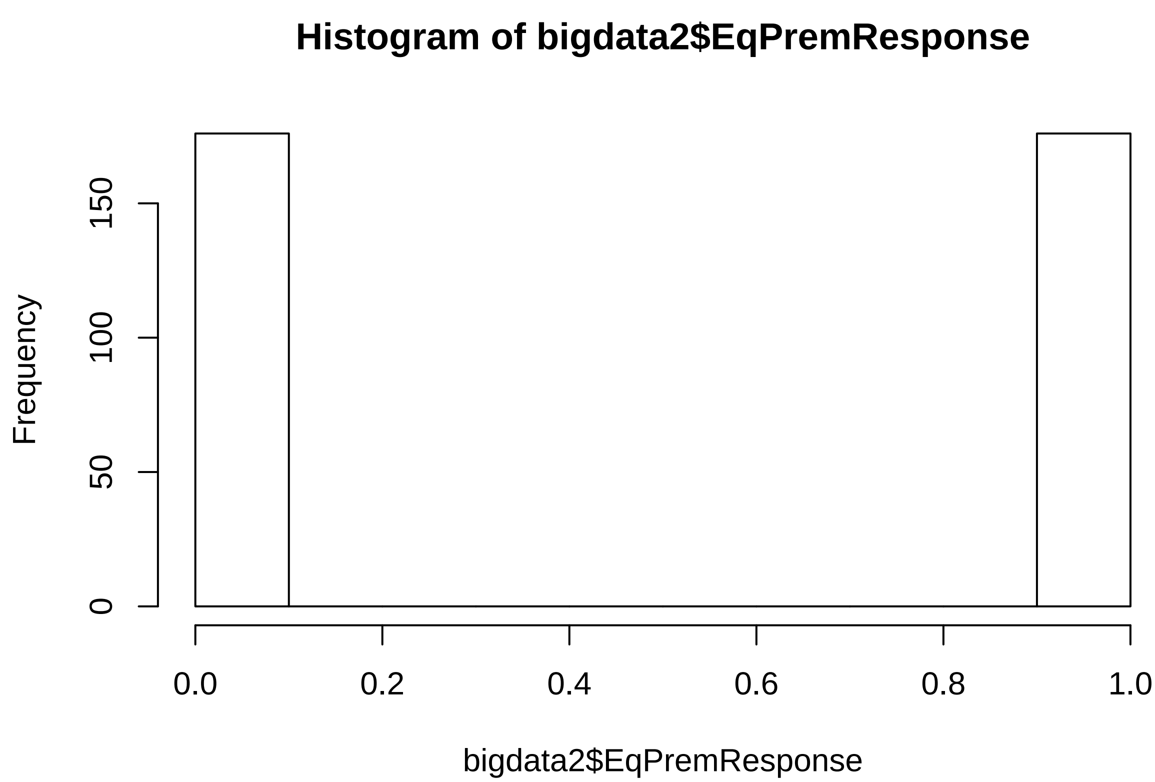
```
## [1] "lm"
## [1] 0.1339107
## [1] "enet"
## [1] 0.1239372
## [1] "leapBackward"
## [1] 0.1286883
```

```
## [1] "leapForward"
## [1] 0.1286883
## [1] "leapSeq"
## [1] 0.1286883
## [1] "nnls"
## [1] 0.126178
## [1] "pcr"
## [1] 0.1252217
## [1] "rvmLinear"
## [1] 0.1296435
## [1] "rvmRadial"
## [1] 0.1249831
## [1] "ridge"
## [1] 0.1339107
```

no real help

- Run a binary classification model
- Create indicator for classification

```r
bigdata2=bigdata
Z <- quantile(bigdata2$EqPrem, probs=c(0,0.5,1)) # really just need 0.5
bigdata2$EqPremResponse=1
bigdata2$EqPremResponse[bigdata$EqPrem < Z[2]] = 0
hist(bigdata2$EqPremResponse)
```



```r
# some algos try todo regression instead of classification on numbers, or error
bigdata2$EqPremResponse <- as.factor(bigdata2$EqPremResponse)
bigdata2 = bigdata2[, !(colnames(bigdata2) == "EqPrem")]
```

Create training and test sets

```r
# create training and test sets
# use same samples as earlier

trainingset <- bigdata2[trainindex,]
testset <- bigdata2[-trainindex, ]
```

Predict quantiles using a variety of algorithms

```r
# "nnet", "pcaNNet", "stepLDA", "stepQDA" don't work great and generate pages of output

myMethods <- c("gbm", "lda", "lda2", "LogitBoost", "multinom", "nb", "qda", "rf", 'rocc', "svmLinear","s

#myMethods <- c("lda")

trc_cv = trainControl(method="cv")

# center and scale for better performance on some methods
runModel <- function(mxpar) {
    return (train(EqPremResponse ~ ., data=trainingset, method=mxpar,
                  preProc = c("center", "scale"), verbose=FALSE))
}

for(mx in myMethods) {
  print(mx)
  mymodel = runModel(mx)

  print("Training set confusion matrix")
  myPredict <- data.frame(prediction=predict(mymodel, trainingset))
  myPredict$EqPremResponse<-trainingset$EqPremResponse
  print(confusionMatrix(myPredict$prediction, myPredict$EqPremResponse))

  print("Test set confusion matrix")
  myPredict <- data.frame(prediction=predict(mymodel, testset))
  myPredict$EqPremResponse<-testset$EqPremResponse
  print(confusionMatrix(myPredict$prediction, myPredict$EqPremResponse))
}
```

```
## [1] "gbm"

## Loading required package: gbm

## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:boot':
##
##     aml

## The following object is masked from 'package:caret':
##
```

```
##     cluster
## Loading required package: splines
## Loading required package: parallel
## Loaded gbm 2.1.3
## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 107  21
##          1  27 109
##
##                Accuracy : 0.8182
##                  95% CI : (0.7663, 0.8628)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6365
##  Mcnemar's Test P-Value : 0.4705
##
##             Sensitivity : 0.7985
##             Specificity : 0.8385
##          Pos Pred Value : 0.8359
##          Neg Pred Value : 0.8015
##              Prevalence : 0.5076
##          Detection Rate : 0.4053
##    Detection Prevalence : 0.4848
##       Balanced Accuracy : 0.8185
##
##        'Positive' Class : 0
##
## [1] "Test set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 22 19
##          1 20 27
##
##                Accuracy : 0.5568
##                  95% CI : (0.447, 0.6627)
##     No Information Rate : 0.5227
##     P-Value [Acc > NIR] : 0.2974
##
##                   Kappa : 0.1109
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.5238
##             Specificity : 0.5870
##          Pos Pred Value : 0.5366
##          Neg Pred Value : 0.5745
##              Prevalence : 0.4773
```

```
##          Detection Rate : 0.2500
##    Detection Prevalence : 0.4659
##       Balanced Accuracy : 0.5554
##
##        'Positive' Class : 0
##
## [1] "lda"
## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 84 47
##          1 50 83
##
##                Accuracy : 0.6326
##                  95% CI : (0.5713, 0.6908)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : 2.847e-05
##
##                   Kappa : 0.2652
##  Mcnemar's Test P-Value : 0.8391
##
##             Sensitivity : 0.6269
##             Specificity : 0.6385
##          Pos Pred Value : 0.6412
##          Neg Pred Value : 0.6241
##              Prevalence : 0.5076
##          Detection Rate : 0.3182
##    Detection Prevalence : 0.4962
##       Balanced Accuracy : 0.6327
##
##        'Positive' Class : 0
##
## [1] "Test set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 29 23
##          1 13 23
##
##                Accuracy : 0.5909
##                  95% CI : (0.4809, 0.6946)
##     No Information Rate : 0.5227
##     P-Value [Acc > NIR] : 0.1200
##
##                   Kappa : 0.1885
##  Mcnemar's Test P-Value : 0.1336
##
##             Sensitivity : 0.6905
##             Specificity : 0.5000
##          Pos Pred Value : 0.5577
##          Neg Pred Value : 0.6389
```

```
##              Prevalence : 0.4773
##          Detection Rate : 0.3295
##    Detection Prevalence : 0.5909
##       Balanced Accuracy : 0.5952
##
##        'Positive' Class : 0
##
## [1] "lda2"
## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 84 47
##          1 50 83
##
##                Accuracy : 0.6326
##                  95% CI : (0.5713, 0.6908)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : 2.847e-05
##
##                   Kappa : 0.2652
##  Mcnemar's Test P-Value : 0.8391
##
##             Sensitivity : 0.6269
##             Specificity : 0.6385
##          Pos Pred Value : 0.6412
##          Neg Pred Value : 0.6241
##              Prevalence : 0.5076
##          Detection Rate : 0.3182
##    Detection Prevalence : 0.4962
##       Balanced Accuracy : 0.6327
##
##        'Positive' Class : 0
##
## [1] "Test set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 29 23
##          1 13 23
##
##                Accuracy : 0.5909
##                  95% CI : (0.4809, 0.6946)
##     No Information Rate : 0.5227
##     P-Value [Acc > NIR] : 0.1200
##
##                   Kappa : 0.1885
##  Mcnemar's Test P-Value : 0.1336
##
##             Sensitivity : 0.6905
##             Specificity : 0.5000
##          Pos Pred Value : 0.5577
```

```
##          Neg Pred Value : 0.6389
##              Prevalence : 0.4773
##          Detection Rate : 0.3295
##    Detection Prevalence : 0.5909
##       Balanced Accuracy : 0.5952
##
##        'Positive' Class : 0
##
## [1] "LogitBoost"
```

```
## Loading required package: caTools
```

```
## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 119  30
##          1  15 100
##
##                Accuracy : 0.8295
##                  95% CI : (0.7786, 0.8729)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.6584
##  Mcnemar's Test P-Value : 0.03689
##
##             Sensitivity : 0.8881
##             Specificity : 0.7692
##          Pos Pred Value : 0.7987
##          Neg Pred Value : 0.8696
##              Prevalence : 0.5076
##          Detection Rate : 0.4508
##    Detection Prevalence : 0.5644
##       Balanced Accuracy : 0.8286
##
##        'Positive' Class : 0
##
## [1] "Test set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 25 23
##          1 17 23
##
##                Accuracy : 0.5455
##                  95% CI : (0.4358, 0.652)
##     No Information Rate : 0.5227
##     P-Value [Acc > NIR] : 0.3751
##
##                   Kappa : 0.0947
##  Mcnemar's Test P-Value : 0.4292
##
```

```
##             Sensitivity : 0.5952
##             Specificity : 0.5000
##          Pos Pred Value : 0.5208
##          Neg Pred Value : 0.5750
##              Prevalence : 0.4773
##          Detection Rate : 0.2841
##    Detection Prevalence : 0.5455
##       Balanced Accuracy : 0.5476
##
##        'Positive' Class : 0
##
## [1] "multinom"
```

```
## Loading required package: nnet
```

```
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 151.489709
## iter  20 value 136.673874
## iter  30 value 134.660504
## iter  40 value 134.651801
## final  value 134.651714
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 152.438587
## iter  20 value 142.471201
## iter  30 value 142.135107
## final  value 142.135100
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 151.490286
## iter  20 value 136.682607
## iter  30 value 134.676745
## iter  40 value 134.668141
## final  value 134.668057
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 159.078528
## iter  20 value 150.908545
## iter  30 value 148.370884
## iter  40 value 147.948811
## final  value 147.948790
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 159.410998
## iter  20 value 153.883982
## iter  30 value 153.579438
## final  value 153.579381
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
```

```
## iter  10 value 159.078874
## iter  20 value 150.913011
## iter  30 value 148.388593
## iter  40 value 147.975116
## final  value 147.975096
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 158.399098
## iter  20 value 152.971982
## iter  30 value 151.815712
## iter  40 value 151.657490
## iter  40 value 151.657489
## iter  40 value 151.657489
## final  value 151.657489
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 158.874618
## iter  20 value 154.980780
## iter  30 value 154.763752
## final  value 154.763714
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 158.399596
## iter  20 value 152.974632
## iter  30 value 151.821657
## iter  40 value 151.666073
## iter  40 value 151.666073
## iter  40 value 151.666073
## final  value 151.666073
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 153.973863
## iter  20 value 142.426534
## iter  30 value 139.796548
## iter  40 value 139.588531
## final  value 139.588507
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 154.427624
## iter  20 value 145.923510
## iter  30 value 145.515035
## final  value 145.515027
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 153.974333
## iter  20 value 142.432665
## iter  30 value 139.812255
## iter  40 value 139.608221
```

```
## final  value 139.608198
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 160.136377
## iter  20 value 148.505308
## iter  30 value 147.283091
## iter  40 value 147.052765
## final  value 147.052752
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 160.640160
## iter  20 value 152.464313
## iter  30 value 152.320053
## final  value 152.320001
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 160.136914
## iter  20 value 148.511003
## iter  30 value 147.292773
## iter  40 value 147.066879
## final  value 147.066866
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 153.441627
## iter  20 value 142.820693
## iter  30 value 140.229251
## final  value 140.225870
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 154.231944
## iter  20 value 146.985354
## iter  30 value 146.286665
## final  value 146.286596
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 153.442474
## iter  20 value 142.827061
## iter  30 value 140.240835
## final  value 140.237476
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 144.915910
## iter  20 value 134.160724
## iter  30 value 132.399838
## iter  40 value 132.383292
## final  value 132.383112
## converged
```

```
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 143.930289
## iter  20 value 137.090047
## iter  30 value 136.723259
## final  value 136.723242
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 144.917540
## iter  20 value 134.165347
## iter  30 value 132.408270
## iter  40 value 132.391890
## final  value 132.391714
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 162.045597
## iter  20 value 151.440354
## iter  30 value 149.501848
## iter  40 value 149.482384
## final  value 149.482320
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 163.138139
## iter  20 value 155.884894
## iter  30 value 155.424577
## final  value 155.424521
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 162.046824
## iter  20 value 151.448600
## iter  30 value 149.512861
## iter  40 value 149.494164
## final  value 149.494101
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 149.849623
## iter  20 value 138.332474
## iter  30 value 134.259268
## iter  40 value 133.948502
## final  value 133.948493
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 150.748130
## iter  20 value 141.364119
## iter  30 value 140.679634
## final  value 140.679572
## converged
## # weights:  32 (31 variable)
```

```
## initial  value 182.990856
## iter  10 value 149.850579
## iter  20 value 138.336834
## iter  30 value 134.281201
## iter  40 value 133.976501
## final  value 133.976493
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 157.378872
## iter  20 value 149.896618
## iter  30 value 147.310796
## iter  40 value 147.123682
## final  value 147.123679
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 158.293749
## iter  20 value 153.248937
## iter  30 value 152.914612
## final  value 152.914567
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 157.379871
## iter  20 value 149.902244
## iter  30 value 147.324252
## iter  40 value 147.140469
## final  value 147.140466
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 156.894250
## iter  20 value 149.956478
## iter  30 value 148.508739
## iter  40 value 148.503159
## final  value 148.503100
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 157.957563
## iter  20 value 153.205439
## iter  30 value 152.775368
## final  value 152.775361
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 156.895436
## iter  20 value 149.960792
## iter  30 value 148.516416
## iter  40 value 148.510899
## final  value 148.510842
## converged
## # weights:  32 (31 variable)
```

```
## initial  value 182.990856
## iter  10 value 146.828280
## iter  20 value 138.363051
## iter  30 value 135.780332
## iter  40 value 135.679002
## final  value 135.678699
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 147.725719
## iter  20 value 142.118841
## iter  30 value 141.670023
## final  value 141.670003
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 146.829247
## iter  20 value 138.368513
## iter  30 value 135.794309
## iter  40 value 135.694692
## final  value 135.694401
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 153.245208
## iter  20 value 147.043941
## iter  30 value 144.689342
## iter  40 value 144.563732
## final  value 144.563696
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 153.620948
## iter  20 value 149.187792
## iter  30 value 148.877737
## final  value 148.877718
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 153.245602
## iter  20 value 147.047225
## iter  30 value 144.700831
## iter  40 value 144.577307
## final  value 144.577272
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 165.686562
## iter  20 value 160.548356
## iter  30 value 159.308748
## iter  40 value 159.252527
## final  value 159.252524
## converged
## # weights:  32 (31 variable)
```

```
## initial  value 182.990856
## iter  10 value 166.027335
## iter  20 value 162.023778
## iter  30 value 161.652563
## final  value 161.652535
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 165.686918
## iter  20 value 160.550191
## iter  30 value 159.313021
## iter  40 value 159.257476
## final  value 159.257474
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 142.259049
## iter  20 value 135.353755
## iter  30 value 133.579685
## iter  40 value 133.531272
## final  value 133.530995
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 144.806293
## iter  20 value 140.577470
## iter  30 value 140.226176
## final  value 140.226166
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 142.262006
## iter  20 value 135.361315
## iter  30 value 133.593065
## iter  40 value 133.545561
## final  value 133.545295
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 153.134011
## iter  20 value 143.721633
## iter  30 value 142.203940
## iter  40 value 141.864677
## final  value 141.864637
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 153.884094
## iter  20 value 148.016218
## iter  30 value 147.691272
## final  value 147.691219
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
```

```
## iter  10 value 153.134806
## iter  20 value 143.727887
## iter  30 value 142.216490
## iter  40 value 141.883685
## final  value 141.883646
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 157.295177
## iter  20 value 146.006838
## iter  30 value 142.004703
## iter  40 value 141.664704
## final  value 141.664674
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 158.421760
## iter  20 value 150.122590
## iter  30 value 149.435407
## final  value 149.435286
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 157.296400
## iter  20 value 146.012469
## iter  30 value 142.022576
## iter  40 value 141.687853
## final  value 141.687824
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 161.426657
## iter  20 value 156.048325
## iter  30 value 154.070104
## iter  40 value 153.893879
## final  value 153.893875
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 161.828783
## iter  20 value 158.225638
## iter  30 value 157.995124
## final  value 157.995115
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 161.427081
## iter  20 value 156.051812
## iter  30 value 154.084076
## iter  40 value 153.912186
## final  value 153.912182
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
```

```
## iter  10 value 154.175901
## iter  20 value 145.972656
## iter  30 value 143.830952
## iter  40 value 143.638626
## iter  40 value 143.638625
## iter  40 value 143.638625
## final  value 143.638625
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 155.235847
## iter  20 value 149.152569
## iter  30 value 148.592108
## final  value 148.592013
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 154.177044
## iter  20 value 145.976759
## iter  30 value 143.839210
## iter  40 value 143.649926
## iter  40 value 143.649926
## iter  40 value 143.649926
## final  value 143.649926
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 163.216716
## iter  20 value 153.650671
## iter  30 value 149.764824
## iter  40 value 149.694792
## final  value 149.694717
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 163.499687
## iter  20 value 157.199010
## iter  30 value 156.691823
## final  value 156.691807
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 163.217009
## iter  20 value 153.656642
## iter  30 value 149.781955
## iter  40 value 149.712886
## final  value 149.712814
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 168.248850
## iter  20 value 160.237945
## iter  30 value 156.753922
## iter  40 value 156.272024
```

```
## final  value 156.272022
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 169.016709
## iter  20 value 163.382376
## iter  30 value 162.889395
## final  value 162.889319
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 168.249693
## iter  20 value 160.242375
## iter  30 value 156.772780
## iter  40 value 156.300060
## final  value 156.300058
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 164.709117
## iter  20 value 156.722484
## iter  30 value 153.021370
## iter  40 value 152.648850
## final  value 152.648846
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 165.085442
## iter  20 value 160.260670
## iter  30 value 159.766601
## final  value 159.766520
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 164.709514
## iter  20 value 156.727986
## iter  30 value 153.039120
## iter  40 value 152.672788
## final  value 152.672784
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 159.263982
## iter  20 value 151.883142
## iter  30 value 149.486386
## iter  40 value 149.276067
## final  value 149.276061
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 160.624773
## iter  20 value 156.360867
## iter  30 value 155.973664
## final  value 155.973571
```

```
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 159.265506
## iter  20 value 151.889909
## iter  30 value 149.500380
## iter  40 value 149.292922
## final  value 149.292916
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 155.961410
## iter  20 value 150.422763
## iter  30 value 149.542314
## iter  40 value 149.540761
## final  value 149.540758
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 156.497230
## iter  20 value 152.080322
## iter  30 value 151.900063
## final  value 151.900060
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 155.961970
## iter  20 value 150.425836
## iter  30 value 149.546246
## iter  40 value 149.544714
## final  value 149.544710
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 159.325142
## iter  20 value 153.201114
## iter  30 value 152.083356
## iter  40 value 151.987978
## iter  40 value 151.987977
## iter  40 value 151.987977
## final  value 151.987977
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 160.018574
## iter  20 value 155.223546
## iter  30 value 154.837506
## final  value 154.837455
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 159.325878
## iter  20 value 153.203638
## iter  30 value 152.088418
```

```
## iter  40 value 151.994208
## iter  40 value 151.994207
## iter  40 value 151.994207
## final  value 151.994207
## converged
## # weights:  32 (31 variable)
## initial  value 182.990856
## iter  10 value 167.395752
## iter  20 value 162.953637
## iter  30 value 161.819741
## final  value 161.815733
## converged
## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 83 48
##          1 51 82
##
##                Accuracy : 0.625
##                  95% CI : (0.5636, 0.6836)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : 8.008e-05
##
##                   Kappa : 0.2501
##  Mcnemar's Test P-Value : 0.8407
##
##             Sensitivity : 0.6194
##             Specificity : 0.6308
##          Pos Pred Value : 0.6336
##          Neg Pred Value : 0.6165
##              Prevalence : 0.5076
##          Detection Rate : 0.3144
##    Detection Prevalence : 0.4962
##       Balanced Accuracy : 0.6251
##
##        'Positive' Class : 0
##
## [1] "Test set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 29 23
##          1 13 23
##
##                Accuracy : 0.5909
##                  95% CI : (0.4809, 0.6946)
##     No Information Rate : 0.5227
##     P-Value [Acc > NIR] : 0.1200
##
##                   Kappa : 0.1885
##  Mcnemar's Test P-Value : 0.1336
```

```
##
##             Sensitivity : 0.6905
##             Specificity : 0.5000
##          Pos Pred Value : 0.5577
##          Neg Pred Value : 0.6389
##              Prevalence : 0.4773
##          Detection Rate : 0.3295
##    Detection Prevalence : 0.5909
##       Balanced Accuracy : 0.5952
##
##        'Positive' Class : 0
##
## [1] "nb"
```

```
## Loading required package: klaR
```

```
## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 51 31
##          1 83 99
##
##                Accuracy : 0.5682
##                  95% CI : (0.5061, 0.6288)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : 0.02803
##
##                   Kappa : 0.1413
##  Mcnemar's Test P-Value : 1.783e-06
##
##             Sensitivity : 0.3806
##             Specificity : 0.7615
##          Pos Pred Value : 0.6220
##          Neg Pred Value : 0.5440
##              Prevalence : 0.5076
##          Detection Rate : 0.1932
##    Detection Prevalence : 0.3106
##       Balanced Accuracy : 0.5711
##
##        'Positive' Class : 0
##
## [1] "Test set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 14 19
##          1 28 27
##
##                Accuracy : 0.4659
##                  95% CI : (0.3588, 0.5754)
##     No Information Rate : 0.5227
##     P-Value [Acc > NIR] : 0.8797
```

```
##
##                   Kappa : -0.0805
##  Mcnemar's Test P-Value : 0.2432
##
##             Sensitivity : 0.3333
##             Specificity : 0.5870
##          Pos Pred Value : 0.4242
##          Neg Pred Value : 0.4909
##              Prevalence : 0.4773
##          Detection Rate : 0.1591
##    Detection Prevalence : 0.3750
##       Balanced Accuracy : 0.4601
##
##        'Positive' Class : 0
##
## [1] "qda"
## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  92  14
##          1  42 116
##
##                Accuracy : 0.7879
##                  95% CI : (0.7336, 0.8356)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.577
##  Mcnemar's Test P-Value : 0.0003085
##
##             Sensitivity : 0.6866
##             Specificity : 0.8923
##          Pos Pred Value : 0.8679
##          Neg Pred Value : 0.7342
##              Prevalence : 0.5076
##          Detection Rate : 0.3485
##    Detection Prevalence : 0.4015
##       Balanced Accuracy : 0.7894
##
##        'Positive' Class : 0
##
## [1] "Test set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 16 22
##          1 26 24
##
##                Accuracy : 0.4545
##                  95% CI : (0.348, 0.5642)
##     No Information Rate : 0.5227
```

```
##     P-Value [Acc > NIR] : 0.9173
##
##                   Kappa : -0.0977
##  Mcnemar's Test P-Value : 0.6650
##
##             Sensitivity : 0.3810
##             Specificity : 0.5217
##          Pos Pred Value : 0.4211
##          Neg Pred Value : 0.4800
##              Prevalence : 0.4773
##          Detection Rate : 0.1818
##    Detection Prevalence : 0.4318
##       Balanced Accuracy : 0.4513
##
##        'Positive' Class : 0
##
## [1] "rf"
```

```
## Loading required package: randomForest
```

```
## randomForest 4.6-12
```

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

```
##
## Attaching package: 'randomForest'
```

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

```
## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 134   0
##          1   0 130
##
##                Accuracy : 1
##                  95% CI : (0.9861, 1)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5076
##          Detection Rate : 0.5076
##    Detection Prevalence : 0.5076
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : 0
```

```
##
## [1] "Test set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 27 23
##          1 15 23
##
##                Accuracy : 0.5682
##                  95% CI : (0.4582, 0.6734)
##     No Information Rate : 0.5227
##     P-Value [Acc > NIR] : 0.2279
##
##                   Kappa : 0.1417
##  Mcnemar's Test P-Value : 0.2561
##
##             Sensitivity : 0.6429
##             Specificity : 0.5000
##          Pos Pred Value : 0.5400
##          Neg Pred Value : 0.6053
##              Prevalence : 0.4773
##          Detection Rate : 0.3068
##    Detection Prevalence : 0.5682
##       Balanced Accuracy : 0.5714
##
##        'Positive' Class : 0
##
## [1] "rocc"

## Loading required package: rocc

## Loading required package: ROCR

## Loading required package: gplots

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess

##
## Attaching package: 'ROCR'

## The following object is masked from 'package:neuralnet':
##
##     prediction

## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 74 48
##          1 60 82
##
```

```
##                Accuracy : 0.5909
##                  95% CI : (0.529, 0.6508)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : 0.00398
##
##                   Kappa : 0.1828
##  Mcnemar's Test P-Value : 0.28984
##
##             Sensitivity : 0.5522
##             Specificity : 0.6308
##          Pos Pred Value : 0.6066
##          Neg Pred Value : 0.5775
##              Prevalence : 0.5076
##          Detection Rate : 0.2803
##    Detection Prevalence : 0.4621
##       Balanced Accuracy : 0.5915
##
##        'Positive' Class : 0
##
## [1] "Test set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 20 28
##          1 22 18
##
##                Accuracy : 0.4318
##                  95% CI : (0.3266, 0.5418)
##     No Information Rate : 0.5227
##     P-Value [Acc > NIR] : 0.9652
##
##                   Kappa : -0.1317
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity : 0.4762
##             Specificity : 0.3913
##          Pos Pred Value : 0.4167
##          Neg Pred Value : 0.4500
##              Prevalence : 0.4773
##          Detection Rate : 0.2273
##    Detection Prevalence : 0.5455
##       Balanced Accuracy : 0.4337
##
##        'Positive' Class : 0
##
## [1] "svmLinear"
## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 81 41
##          1 53 89
```

```
##
##                Accuracy : 0.6439
##                  95% CI : (0.5829, 0.7017)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : 5.369e-06
##
##                   Kappa : 0.2887
##  Mcnemar's Test P-Value : 0.2566
##
##             Sensitivity : 0.6045
##             Specificity : 0.6846
##          Pos Pred Value : 0.6639
##          Neg Pred Value : 0.6268
##              Prevalence : 0.5076
##          Detection Rate : 0.3068
##    Detection Prevalence : 0.4621
##       Balanced Accuracy : 0.6445
##
##        'Positive' Class : 0
##
## [1] "Test set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 27 24
##          1 15 22
##
##                Accuracy : 0.5568
##                  95% CI : (0.447, 0.6627)
##     No Information Rate : 0.5227
##     P-Value [Acc > NIR] : 0.2974
##
##                   Kappa : 0.12
##  Mcnemar's Test P-Value : 0.2002
##
##             Sensitivity : 0.6429
##             Specificity : 0.4783
##          Pos Pred Value : 0.5294
##          Neg Pred Value : 0.5946
##              Prevalence : 0.4773
##          Detection Rate : 0.3068
##    Detection Prevalence : 0.5795
##       Balanced Accuracy : 0.5606
##
##        'Positive' Class : 0
##
## [1] "svmRadial"
## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  98  18
```

```
##          1  36 112
##
##                Accuracy : 0.7955
##                  95% CI : (0.7417, 0.8424)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5917
##  Mcnemar's Test P-Value : 0.0207
##
##             Sensitivity : 0.7313
##             Specificity : 0.8615
##          Pos Pred Value : 0.8448
##          Neg Pred Value : 0.7568
##              Prevalence : 0.5076
##          Detection Rate : 0.3712
##    Detection Prevalence : 0.4394
##       Balanced Accuracy : 0.7964
##
##        'Positive' Class : 0
##
## [1] "Test set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 23 25
##          1 19 21
##
##                Accuracy : 0.5
##                  95% CI : (0.3915, 0.6085)
##     No Information Rate : 0.5227
##     P-Value [Acc > NIR] : 0.7034
##
##                   Kappa : 0.0041
##  Mcnemar's Test P-Value : 0.4510
##
##             Sensitivity : 0.5476
##             Specificity : 0.4565
##          Pos Pred Value : 0.4792
##          Neg Pred Value : 0.5250
##              Prevalence : 0.4773
##          Detection Rate : 0.2614
##    Detection Prevalence : 0.5455
##       Balanced Accuracy : 0.5021
##
##        'Positive' Class : 0
##
## [1] "svmRadialWeights"
## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
```

```
##          0 134 130
##          1   0   0
##
##                Accuracy : 0.5076
##                  95% CI : (0.4456, 0.5694)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : 0.5247
##
##                   Kappa : 0
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.5076
##          Neg Pred Value :    NaN
##              Prevalence : 0.5076
##          Detection Rate : 0.5076
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 0
##
## [1] "Test set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 42 46
##          1  0  0
##
##                Accuracy : 0.4773
##                  95% CI : (0.3696, 0.5865)
##     No Information Rate : 0.5227
##     P-Value [Acc > NIR] : 0.8316
##
##                   Kappa : 0
##  Mcnemar's Test P-Value : 3.247e-11
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.4773
##          Neg Pred Value :    NaN
##              Prevalence : 0.4773
##          Detection Rate : 0.4773
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 0
##
## [1] "treebag"
```

```
## Loading required package: ipred
```

```
## Loading required package: e1071
```

```
## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 134   0
##          1   0 130
##
##                Accuracy : 1
##                  95% CI : (0.9861, 1)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5076
##          Detection Rate : 0.5076
##    Detection Prevalence : 0.5076
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : 0
##
## [1] "Test set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 24 21
##          1 18 25
##
##                Accuracy : 0.5568
##                  95% CI : (0.447, 0.6627)
##     No Information Rate : 0.5227
##     P-Value [Acc > NIR] : 0.2974
##
##                   Kappa : 0.1146
##  Mcnemar's Test P-Value : 0.7488
##
##             Sensitivity : 0.5714
##             Specificity : 0.5435
##          Pos Pred Value : 0.5333
##          Neg Pred Value : 0.5814
##              Prevalence : 0.4773
##          Detection Rate : 0.2727
##    Detection Prevalence : 0.5114
##       Balanced Accuracy : 0.5575
##
##        'Positive' Class : 0
##
```

```
## [1] "bartMachine"
## Loading required package: bartMachine
## Loading required package: rJava
## Loading required package: bartMachineJARs
## Loading required package: car
##
## Attaching package: 'car'
## The following object is masked from 'package:boot':
##
##     logit
## Loading required package: missForest
## Loading required package: foreach
## Loading required package: itertools
## Loading required package: iterators
## Welcome to bartMachine v1.2.3! You have 3.82GB memory available.
##
## If you run out of memory, restart R, and use e.g.
## 'options(java.parameters = "-Xmx5g")' for 5GB of RAM before you call
## 'library(bartMachine)'.
## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 111  24
##          1  23 106
##
##                Accuracy : 0.822
##                  95% CI : (0.7704, 0.8662)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6438
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8284
##             Specificity : 0.8154
##          Pos Pred Value : 0.8222
##          Neg Pred Value : 0.8217
##              Prevalence : 0.5076
##          Detection Rate : 0.4205
##    Detection Prevalence : 0.5114
##       Balanced Accuracy : 0.8219
##
##        'Positive' Class : 0
##
## [1] "Test set confusion matrix"
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction  0  1
##          0 24 23
##          1 18 23
##
##                Accuracy : 0.5341
##                  95% CI : (0.4246, 0.6412)
##     No Information Rate : 0.5227
##     P-Value [Acc > NIR] : 0.4582
##
##                   Kappa : 0.0711
##  Mcnemar's Test P-Value : 0.5322
##
##             Sensitivity : 0.5714
##             Specificity : 0.5000
##          Pos Pred Value : 0.5106
##          Neg Pred Value : 0.5610
##              Prevalence : 0.4773
##          Detection Rate : 0.2727
##    Detection Prevalence : 0.5341
##       Balanced Accuracy : 0.5357
##
##        'Positive' Class : 0
##
## [1] "deepboost"
```

```
## Loading required package: deepboost
```

```
##
## Attaching package: 'deepboost'
```

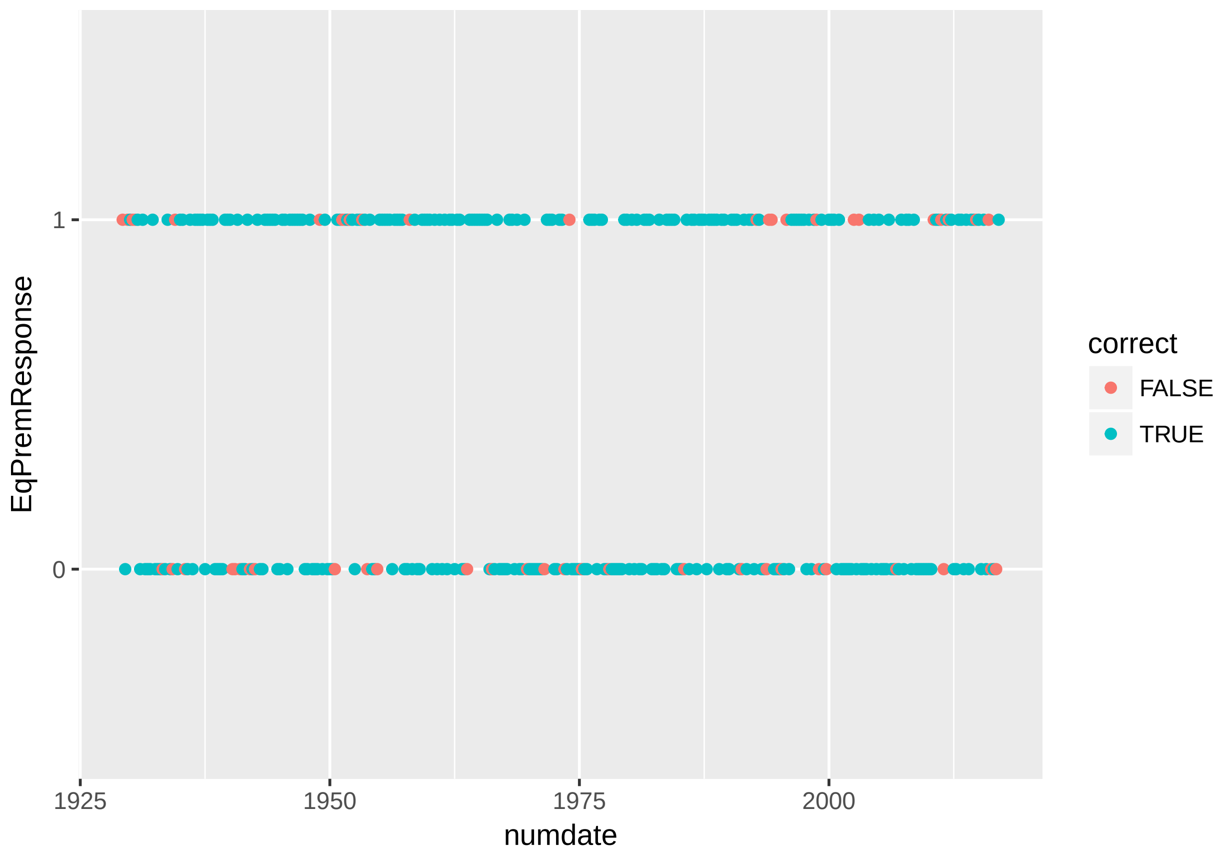
```
## The following object is masked from 'package:survival':
##
##     heart
```

```
## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 129   2
##          1   5 128
##
##                Accuracy : 0.9735
##                  95% CI : (0.9461, 0.9893)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.947
##  Mcnemar's Test P-Value : 0.4497
##
##             Sensitivity : 0.9627
##             Specificity : 0.9846
##          Pos Pred Value : 0.9847
##          Neg Pred Value : 0.9624
```

```
##               Prevalence : 0.5076
##           Detection Rate : 0.4886
##     Detection Prevalence : 0.4962
##        Balanced Accuracy : 0.9737
##
##         'Positive' Class : 0
##
## [1] "Test set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 19 22
##          1 23 24
##
##                Accuracy : 0.4886
##                  95% CI : (0.3805, 0.5975)
##     No Information Rate : 0.5227
##     P-Value [Acc > NIR] : 0.7726
##
##                   Kappa : -0.0259
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.4524
##             Specificity : 0.5217
##          Pos Pred Value : 0.4634
##          Neg Pred Value : 0.5106
##              Prevalence : 0.4773
##          Detection Rate : 0.2159
##    Detection Prevalence : 0.4659
##       Balanced Accuracy : 0.4871
##
##        'Positive' Class : 0
##
```

Chart correct vs. incorrect

```r
myPredict <- data.frame(prediction=predict(mymodel, bigdata2))
myPredict$EqPremResponse<-bigdata2$EqPremResponse
myPredict$numdate <- tail(data$numdate, nrow(myPredict))
myPredict$correct <- (myPredict$prediction==myPredict$EqPremResponse)
ggplot(myPredict, aes(x=numdate, y=EqPremResponse, color=correct)) + geom_point()
```



Just for grins, predict on regressionTestPredicts

- kitchen sink ensemble methods FTW

```
regressionTrainPredicts$EqPremResponse <- trainingset$EqPremResponse
runModel <- function(mxpar) {
    return (train(EqPremResponse ~ ., data=regressionTrainPredicts, method=mxpar,
                  preProc = c("center", "scale"), verbose=FALSE))
}

#myMethods <- c("ada", "AdaBag", "adaboost", "bartMachine", "deepboost", "gbm", "lda", "LogitBoost", "m

myMethods <- c("ada", "AdaBag", "adaboost", "bartMachine", "deepboost", "gbm", "lda", "rf", 'rocc', "svm

for(mx in myMethods) {
  print(Sys.time())
  print(mx)
  mymodel = runModel(mx)

  print("Training set confusion matrix")
  myPredict <- data.frame(prediction=predict(mymodel, regressionTrainPredicts))
  myPredict$EqPremResponse<-trainingset$EqPremResponse
  print(confusionMatrix(myPredict$prediction, myPredict$EqPremResponse))

  print("Test set confusion matrix")
  myPredict <- data.frame(prediction=predict(mymodel, regressionTestPredicts))
  myPredict$EqPremResponse<-testset$EqPremResponse
  print(confusionMatrix(myPredict$prediction, myPredict$EqPremResponse))
```

```
}
```

```
## [1] "2017-07-09 09:49:07 EDT"
## [1] "ada"

## Loading required package: ada

## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 87 38
##          1 47 92
##
##                Accuracy : 0.678
##                  95% CI : (0.618, 0.734)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : 1.511e-08
##
##                   Kappa : 0.3566
##  Mcnemar's Test P-Value : 0.3855
##
##             Sensitivity : 0.6493
##             Specificity : 0.7077
##          Pos Pred Value : 0.6960
##          Neg Pred Value : 0.6619
##              Prevalence : 0.5076
##          Detection Rate : 0.3295
##    Detection Prevalence : 0.4735
##       Balanced Accuracy : 0.6785
##
##        'Positive' Class : 0
##
## [1] "Test set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 20 19
##          1 22 27
##
##                Accuracy : 0.5341
##                  95% CI : (0.4246, 0.6412)
##     No Information Rate : 0.5227
##     P-Value [Acc > NIR] : 0.4582
##
##                   Kappa : 0.0633
##  Mcnemar's Test P-Value : 0.7548
##
##             Sensitivity : 0.4762
##             Specificity : 0.5870
##          Pos Pred Value : 0.5128
##          Neg Pred Value : 0.5510
##              Prevalence : 0.4773
```

```
##          Detection Rate : 0.2273
##    Detection Prevalence : 0.4432
##       Balanced Accuracy : 0.5316
##
##        'Positive' Class : 0
##
## [1] "2017-07-09 09:52:01 EDT"
## [1] "AdaBag"
## Loading required package: adabag
##
## Attaching package: 'adabag'
## The following object is masked from 'package:ipred':
##
##     bagging
## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 76 32
##          1 58 98
##
##                Accuracy : 0.6591
##                  95% CI : (0.5985, 0.7161)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : 4.647e-07
##
##                   Kappa : 0.3201
##  Mcnemar's Test P-Value : 0.008408
##
##             Sensitivity : 0.5672
##             Specificity : 0.7538
##          Pos Pred Value : 0.7037
##          Neg Pred Value : 0.6282
##              Prevalence : 0.5076
##          Detection Rate : 0.2879
##    Detection Prevalence : 0.4091
##       Balanced Accuracy : 0.6605
##
##        'Positive' Class : 0
##
## [1] "Test set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 17 20
##          1 25 26
##
##                Accuracy : 0.4886
##                  95% CI : (0.3805, 0.5975)
##     No Information Rate : 0.5227
```

```
##     P-Value [Acc > NIR] : 0.7726
##
##                   Kappa : -0.0302
##  Mcnemar's Test P-Value : 0.5510
##
##             Sensitivity : 0.4048
##             Specificity : 0.5652
##          Pos Pred Value : 0.4595
##          Neg Pred Value : 0.5098
##              Prevalence : 0.4773
##          Detection Rate : 0.1932
##    Detection Prevalence : 0.4205
##       Balanced Accuracy : 0.4850
##
##        'Positive' Class : 0
##
## [1] "2017-07-09 10:01:34 EDT"
## [1] "adaboost"
## Loading required package: fastAdaboost
## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 134   0
##          1   0 130
##
##                Accuracy : 1
##                  95% CI : (0.9861, 1)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5076
##          Detection Rate : 0.5076
##    Detection Prevalence : 0.5076
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : 0
##
## [1] "Test set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 24 27
##          1 18 19
```

```
##
##                Accuracy : 0.4886
##                  95% CI : (0.3805, 0.5975)
##     No Information Rate : 0.5227
##     P-Value [Acc > NIR] : 0.7726
##
##                   Kappa : -0.0154
##  Mcnemar's Test P-Value : 0.2330
##
##             Sensitivity : 0.5714
##             Specificity : 0.4130
##          Pos Pred Value : 0.4706
##          Neg Pred Value : 0.5135
##              Prevalence : 0.4773
##          Detection Rate : 0.2727
##    Detection Prevalence : 0.5795
##       Balanced Accuracy : 0.4922
##
##        'Positive' Class : 0
##
## [1] "2017-07-09 10:05:51 EDT"
## [1] "bartMachine"
## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 96 38
##          1 38 92
##
##                Accuracy : 0.7121
##                  95% CI : (0.6534, 0.766)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : 1.082e-11
##
##                   Kappa : 0.4241
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.7164
##             Specificity : 0.7077
##          Pos Pred Value : 0.7164
##          Neg Pred Value : 0.7077
##              Prevalence : 0.5076
##          Detection Rate : 0.3636
##    Detection Prevalence : 0.5076
##       Balanced Accuracy : 0.7121
##
##        'Positive' Class : 0
##
## [1] "Test set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
```

```
##          0 19 19
##          1 23 27
##
##                Accuracy : 0.5227
##                  95% CI : (0.4135, 0.6304)
##     No Information Rate : 0.5227
##     P-Value [Acc > NIR] : 0.5431
##
##                   Kappa : 0.0395
##  Mcnemar's Test P-Value : 0.6434
##
##             Sensitivity : 0.4524
##             Specificity : 0.5870
##          Pos Pred Value : 0.5000
##          Neg Pred Value : 0.5400
##              Prevalence : 0.4773
##          Detection Rate : 0.2159
##    Detection Prevalence : 0.4318
##       Balanced Accuracy : 0.5197
##
##        'Positive' Class : 0
##
## [1] "2017-07-09 10:21:58 EDT"
## [1] "deepboost"
## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 95 48
##          1 39 82
##
##                Accuracy : 0.6705
##                  95% CI : (0.6102, 0.7268)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : 6.251e-08
##
##                   Kappa : 0.3401
##  Mcnemar's Test P-Value : 0.3911
##
##             Sensitivity : 0.7090
##             Specificity : 0.6308
##          Pos Pred Value : 0.6643
##          Neg Pred Value : 0.6777
##              Prevalence : 0.5076
##          Detection Rate : 0.3598
##    Detection Prevalence : 0.5417
##       Balanced Accuracy : 0.6699
##
##        'Positive' Class : 0
##
## [1] "Test set confusion matrix"
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction  0  1
##          0 24 24
##          1 18 22
##
##                Accuracy : 0.5227
##                  95% CI : (0.4135, 0.6304)
##     No Information Rate : 0.5227
##     P-Value [Acc > NIR] : 0.5431
##
##                   Kappa : 0.0494
##  Mcnemar's Test P-Value : 0.4404
##
##             Sensitivity : 0.5714
##             Specificity : 0.4783
##          Pos Pred Value : 0.5000
##          Neg Pred Value : 0.5500
##              Prevalence : 0.4773
##          Detection Rate : 0.2727
##    Detection Prevalence : 0.5455
##       Balanced Accuracy : 0.5248
##
##        'Positive' Class : 0
##
## [1] "2017-07-09 10:27:03 EDT"
## [1] "gbm"
## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 90 32
##          1 44 98
##
##                Accuracy : 0.7121
##                  95% CI : (0.6534, 0.766)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : 1.082e-11
##
##                   Kappa : 0.4249
##  Mcnemar's Test P-Value : 0.207
##
##             Sensitivity : 0.6716
##             Specificity : 0.7538
##          Pos Pred Value : 0.7377
##          Neg Pred Value : 0.6901
##              Prevalence : 0.5076
##          Detection Rate : 0.3409
##    Detection Prevalence : 0.4621
##       Balanced Accuracy : 0.7127
##
##        'Positive' Class : 0
##
## [1] "Test set confusion matrix"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 16 20
##          1 26 26
##
##                Accuracy : 0.4773
##                  95% CI : (0.3696, 0.5865)
##     No Information Rate : 0.5227
##     P-Value [Acc > NIR] : 0.8316
##
##                   Kappa : -0.0542
##  Mcnemar's Test P-Value : 0.4610
##
##             Sensitivity : 0.3810
##             Specificity : 0.5652
##          Pos Pred Value : 0.4444
##          Neg Pred Value : 0.5000
##              Prevalence : 0.4773
##          Detection Rate : 0.1818
##    Detection Prevalence : 0.4091
##       Balanced Accuracy : 0.4731
##
##        'Positive' Class : 0
##
## [1] "2017-07-09 10:27:07 EDT"
## [1] "lda"
## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 86 48
##          1 48 82
##
##                Accuracy : 0.6364
##                  95% CI : (0.5752, 0.6945)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : 1.658e-05
##
##                   Kappa : 0.2726
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.6418
##             Specificity : 0.6308
##          Pos Pred Value : 0.6418
##          Neg Pred Value : 0.6308
##              Prevalence : 0.5076
##          Detection Rate : 0.3258
##    Detection Prevalence : 0.5076
##       Balanced Accuracy : 0.6363
##
##        'Positive' Class : 0
```

```
##
## [1] "Test set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 25 22
##          1 17 24
##
##                Accuracy : 0.5568
##                  95% CI : (0.447, 0.6627)
##     No Information Rate : 0.5227
##     P-Value [Acc > NIR] : 0.2974
##
##                   Kappa : 0.1164
##  Mcnemar's Test P-Value : 0.5218
##
##             Sensitivity : 0.5952
##             Specificity : 0.5217
##          Pos Pred Value : 0.5319
##          Neg Pred Value : 0.5854
##              Prevalence : 0.4773
##          Detection Rate : 0.2841
##    Detection Prevalence : 0.5341
##       Balanced Accuracy : 0.5585
##
##        'Positive' Class : 0
##
## [1] "2017-07-09 10:27:08 EDT"
## [1] "rf"
## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 134   0
##          1   0 130
##
##                Accuracy : 1
##                  95% CI : (0.9861, 1)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5076
##          Detection Rate : 0.5076
##    Detection Prevalence : 0.5076
##       Balanced Accuracy : 1.0000
```

```
##
##        'Positive' Class : 0
##
## [1] "Test set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 24 21
##          1 18 25
##
##                Accuracy : 0.5568
##                  95% CI : (0.447, 0.6627)
##     No Information Rate : 0.5227
##     P-Value [Acc > NIR] : 0.2974
##
##                   Kappa : 0.1146
##  Mcnemar's Test P-Value : 0.7488
##
##             Sensitivity : 0.5714
##             Specificity : 0.5435
##          Pos Pred Value : 0.5333
##          Neg Pred Value : 0.5814
##              Prevalence : 0.4773
##          Detection Rate : 0.2727
##    Detection Prevalence : 0.5114
##       Balanced Accuracy : 0.5575
##
##        'Positive' Class : 0
##
## [1] "2017-07-09 10:27:21 EDT"
## [1] "rocc"
## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 100  54
##          1  34  76
##
##                Accuracy : 0.6667
##                  95% CI : (0.6063, 0.7233)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : 1.24e-07
##
##                   Kappa : 0.3316
##  Mcnemar's Test P-Value : 0.04283
##
##             Sensitivity : 0.7463
##             Specificity : 0.5846
##          Pos Pred Value : 0.6494
##          Neg Pred Value : 0.6909
##              Prevalence : 0.5076
##          Detection Rate : 0.3788
```

```
##    Detection Prevalence : 0.5833
##       Balanced Accuracy : 0.6654
##
##        'Positive' Class : 0
##
## [1] "Test set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 28 23
##          1 14 23
##
##                Accuracy : 0.5795
##                  95% CI : (0.4695, 0.684)
##     No Information Rate : 0.5227
##     P-Value [Acc > NIR] : 0.1685
##
##                   Kappa : 0.1651
##  Mcnemar's Test P-Value : 0.1884
##
##             Sensitivity : 0.6667
##             Specificity : 0.5000
##          Pos Pred Value : 0.5490
##          Neg Pred Value : 0.6216
##              Prevalence : 0.4773
##          Detection Rate : 0.3182
##    Detection Prevalence : 0.5795
##       Balanced Accuracy : 0.5833
##
##        'Positive' Class : 0
##
## [1] "2017-07-09 10:27:23 EDT"
## [1] "svmLinear"
## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 81 36
##          1 53 94
##
##                Accuracy : 0.6629
##                  95% CI : (0.6024, 0.7197)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : 2.42e-07
##
##                   Kappa : 0.3269
##  Mcnemar's Test P-Value : 0.08989
##
##             Sensitivity : 0.6045
##             Specificity : 0.7231
##          Pos Pred Value : 0.6923
##          Neg Pred Value : 0.6395
```

```
##              Prevalence : 0.5076
##          Detection Rate : 0.3068
##    Detection Prevalence : 0.4432
##       Balanced Accuracy : 0.6638
##
##        'Positive' Class : 0
##
## [1] "Test set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 20 20
##          1 22 26
##
##                Accuracy : 0.5227
##                  95% CI : (0.4135, 0.6304)
##     No Information Rate : 0.5227
##     P-Value [Acc > NIR] : 0.5431
##
##                   Kappa : 0.0415
##  Mcnemar's Test P-Value : 0.8774
##
##             Sensitivity : 0.4762
##             Specificity : 0.5652
##          Pos Pred Value : 0.5000
##          Neg Pred Value : 0.5417
##              Prevalence : 0.4773
##          Detection Rate : 0.2273
##    Detection Prevalence : 0.4545
##       Balanced Accuracy : 0.5207
##
##        'Positive' Class : 0
##
## [1] "2017-07-09 10:27:24 EDT"
## [1] "svmRadial"
## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 89 42
##          1 45 88
##
##                Accuracy : 0.6705
##                  95% CI : (0.6102, 0.7268)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : 6.251e-08
##
##                   Kappa : 0.341
##  Mcnemar's Test P-Value : 0.8302
##
##             Sensitivity : 0.6642
##             Specificity : 0.6769
```

```
##          Pos Pred Value : 0.6794
##          Neg Pred Value : 0.6617
##              Prevalence : 0.5076
##          Detection Rate : 0.3371
##    Detection Prevalence : 0.4962
##       Balanced Accuracy : 0.6706
##
##        'Positive' Class : 0
##
## [1] "Test set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 28 26
##          1 14 20
##
##                Accuracy : 0.5455
##                  95% CI : (0.4358, 0.652)
##     No Information Rate : 0.5227
##     P-Value [Acc > NIR] : 0.37509
##
##                   Kappa : 0.1002
##  Mcnemar's Test P-Value : 0.08199
##
##             Sensitivity : 0.6667
##             Specificity : 0.4348
##          Pos Pred Value : 0.5185
##          Neg Pred Value : 0.5882
##              Prevalence : 0.4773
##          Detection Rate : 0.3182
##    Detection Prevalence : 0.6136
##       Balanced Accuracy : 0.5507
##
##        'Positive' Class : 0
##
## [1] "2017-07-09 10:27:27 EDT"
## [1] "svmRadialWeights"
## [1] "Training set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 91 52
##          1 43 78
##
##                Accuracy : 0.6402
##                  95% CI : (0.579, 0.6981)
##     No Information Rate : 0.5076
##     P-Value [Acc > NIR] : 9.51e-06
##
##                   Kappa : 0.2794
##  Mcnemar's Test P-Value : 0.4118
##
```

```
##             Sensitivity : 0.6791
##             Specificity : 0.6000
##          Pos Pred Value : 0.6364
##          Neg Pred Value : 0.6446
##              Prevalence : 0.5076
##          Detection Rate : 0.3447
##    Detection Prevalence : 0.5417
##       Balanced Accuracy : 0.6396
##
##        'Positive' Class : 0
##
## [1] "Test set confusion matrix"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 28 26
##          1 14 20
##
##                Accuracy : 0.5455
##                  95% CI : (0.4358, 0.652)
##     No Information Rate : 0.5227
##     P-Value [Acc > NIR] : 0.37509
##
##                   Kappa : 0.1002
##  Mcnemar's Test P-Value : 0.08199
##
##             Sensitivity : 0.6667
##             Specificity : 0.4348
##          Pos Pred Value : 0.5185
##          Neg Pred Value : 0.5882
##              Prevalence : 0.4773
##          Detection Rate : 0.3182
##    Detection Prevalence : 0.6136
##       Balanced Accuracy : 0.5507
##
##        'Positive' Class : 0
##
```