Final Project

Josh Gardner, Brody Johnson, Zach Reimers, Sara Shields-Menard

2024-07-30

## Libraries

## Data

diabetes = as.data.frame(read.csv(text = getURL('https://raw.githubusercontent.com/zachbreimers/STA6543\_Final\_Project/main/diabetes.csv')))

## Check for null and missing values

which(is.na(diabetes))

## integer(0)

which(is.null(diabetes))

## integer(0)

### Conclusion

No null or missing values in data set

## Check for predictors with near zero variances

nearZeroVar(diabetes[1:8])

## integer(0)

### Conclusion

No predictors with near zero variance in data set.

## Check for skewness

apply(diabetes[1:8], 2, function(x) skewness(x))

## Pregnancies Glucose BloodPressure   
## 0.8981549 0.1730754 -1.8364126   
## SkinThickness Insulin BMI   
## 0.1089456 2.2633826 -0.4273073   
## DiabetesPedigreeFunction Age   
## 1.9124179 1.1251880

### Conclusion

Several predictors exhibit a higher degree of skewness, data would benefit from pre-processing including: centering, scaling, Box-Cox transformation, and normalization.

## Train Test Split

set.seed(100)  
train\_partition = createDataPartition(diabetes[,9], p = 0.8)[[1]]  
  
# Response split  
diabetes\_train = diabetes[train\_partition,9]  
diabetes\_test = diabetes[-train\_partition,9]  
  
# Predictor split  
x\_train = diabetes[train\_partition,1:8]  
x\_test = diabetes[-train\_partition,1:8]

## Continuous Response Train Control

indx = createFolds(diabetes\_train, returnTrain = TRUE)  
ctrl = trainControl(method = "cv", index = indx)

## Continuous Response Models

## Linear Regression and It’s Cousins

### Linear Regression

set.seed(100)  
  
diabetes\_lm = train(x = x\_train, y = diabetes\_train,  
 method = 'lm',  
 preProcess = c('center', 'scale', 'BoxCox', 'spatialSign'),  
 trControl = ctrl)

## Warning in train.default(x = x\_train, y = diabetes\_train, method = "lm", : You  
## are trying to do regression and your outcome only has two possible values Are  
## you trying to do classification? If so, use a 2 level factor as your outcome  
## column.

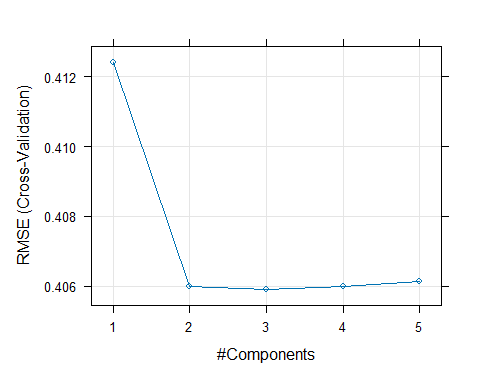
#diabetes\_lm  
summary(diabetes\_lm)

##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.86130 -0.29804 -0.06385 0.31538 1.04581   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.36928 0.01640 22.518 < 2e-16 \*\*\*  
## Pregnancies 0.08838 0.05662 1.561 0.119099   
## Glucose 0.49763 0.05000 9.952 < 2e-16 \*\*\*  
## BloodPressure -0.12807 0.05774 -2.218 0.026916 \*   
## SkinThickness -0.01104 0.05057 -0.218 0.827236   
## Insulin -0.05351 0.06120 -0.874 0.382279   
## BMI 0.27916 0.05477 5.097 4.62e-07 \*\*\*  
## DiabetesPedigreeFunction 0.16695 0.04431 3.768 0.000181 \*\*\*  
## Age 0.19116 0.05537 3.453 0.000594 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.4024 on 606 degrees of freedom  
## Multiple R-squared: 0.2954, Adjusted R-squared: 0.2861   
## F-statistic: 31.75 on 8 and 606 DF, p-value: < 2.2e-16

diabetes\_results = data.frame(obs = diabetes\_test,  
 LM = predict(diabetes\_lm, x\_test))

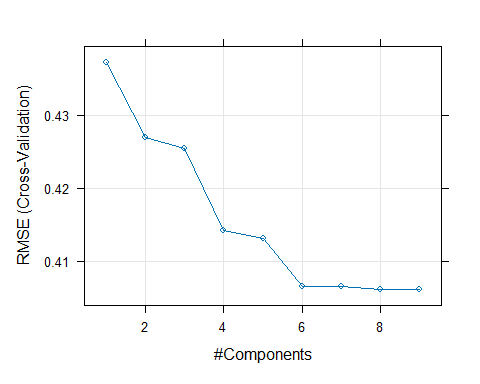
### Partial Least Squares

#diabetes\_pls  
plot(diabetes\_pls)



### PCR

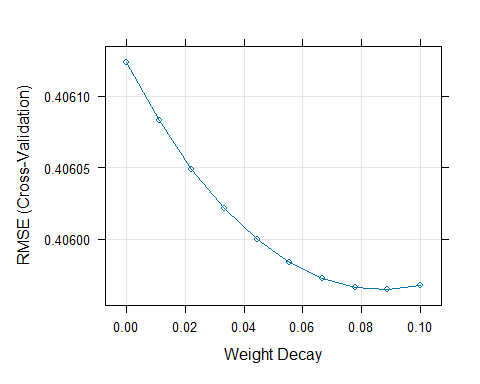
#diabetes\_pcr  
plot(diabetes\_pcr)



## Shrinkage-Penalized Models

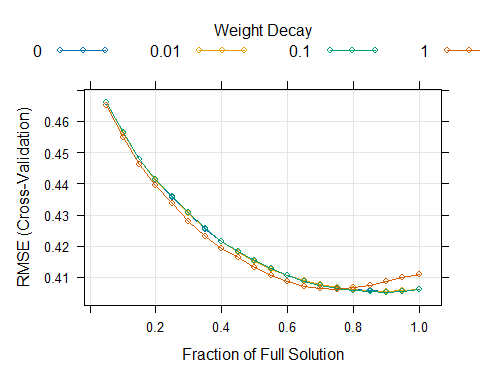
### Ridge Regression

#diabetes\_ridge  
#names(diabetes\_ridge)  
#summary(diabetes\_ridge)  
plot(diabetes\_ridge)



### ENET

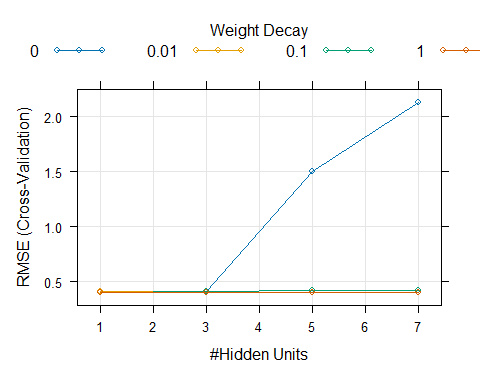
#diabetes\_enet  
#names(diabetes\_enet)  
#summary(diabetes\_enet)  
plot(diabetes\_enet)



## NonLinear Regression Models

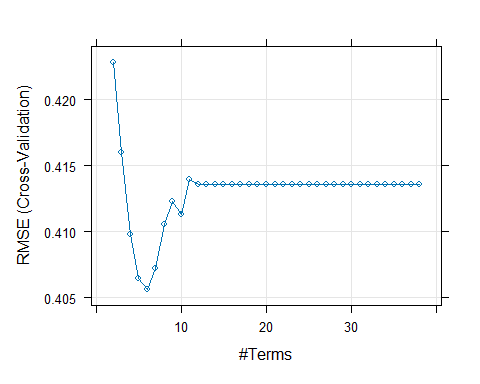
### Neural Net

#diabetes\_nnet  
plot(diabetes\_nnet)

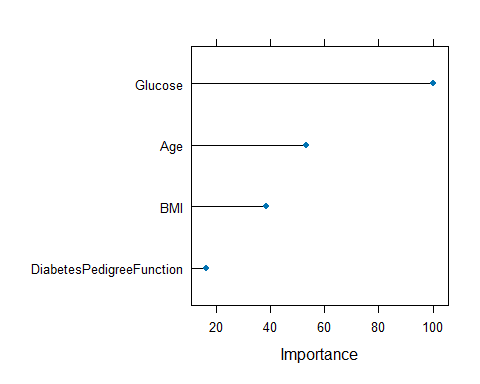


### MARS

#diabetes\_mars  
plot(diabetes\_mars)

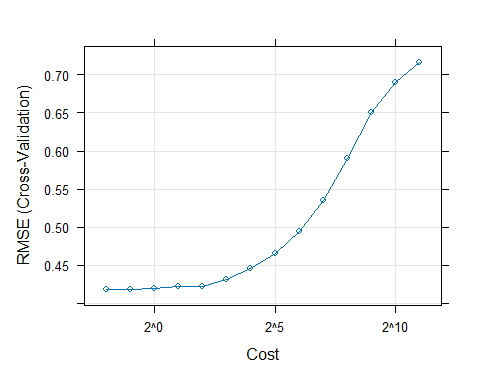


MARSimp = varImp(diabetes\_mars, scale = FALSE)  
plot(MARSimp)



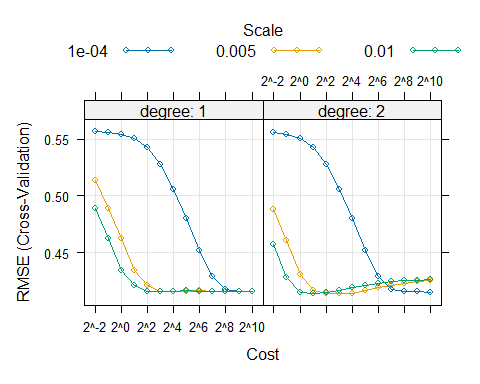
### Radial SVM

#diabetes\_svmr  
plot(diabetes\_svmr, scales = list(x = list(log = 2)))



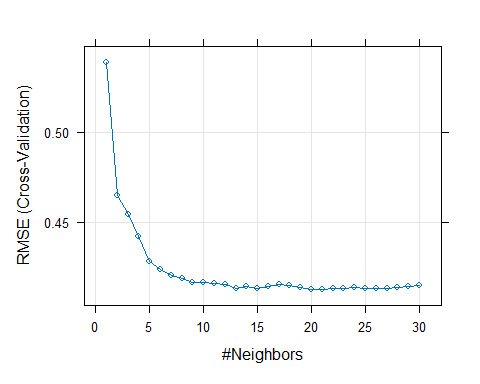
### Poly SVM

#diabetes\_svmp  
plot(diabetes\_svmp,  
 scales = list(x = list(log = 2),  
 between = list(x = 0.5, y = 1)))



### KNN

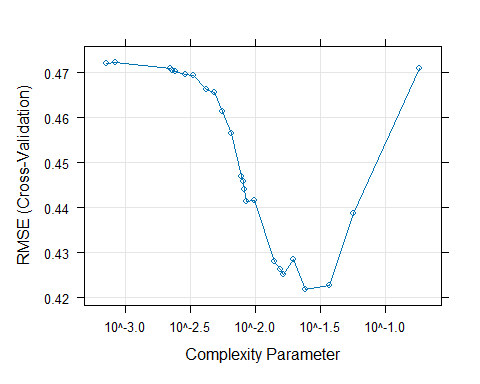
#diabetes\_knn  
plot(diabetes\_knn)



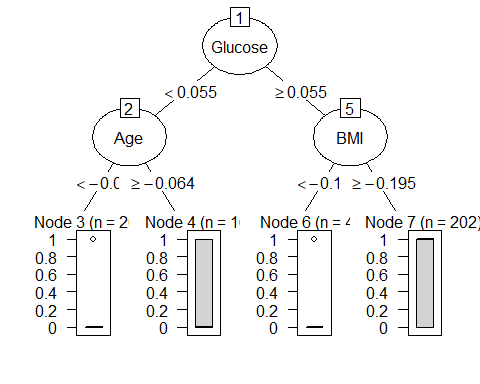
## Regression Trees and Rule-Based Models

### Basic Regression Tree

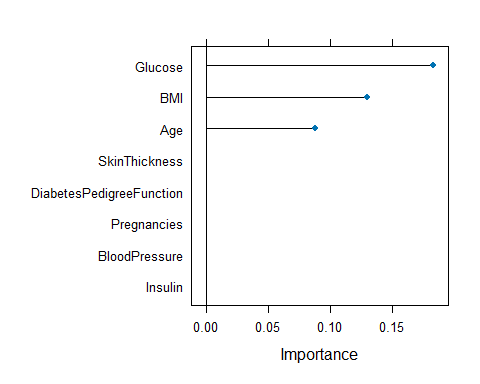
#diabetes\_tree  
plot(diabetes\_tree, scales = list(x = list(log = 10)))



Tree = as.party(diabetes\_tree$finalModel)  
plot(Tree)

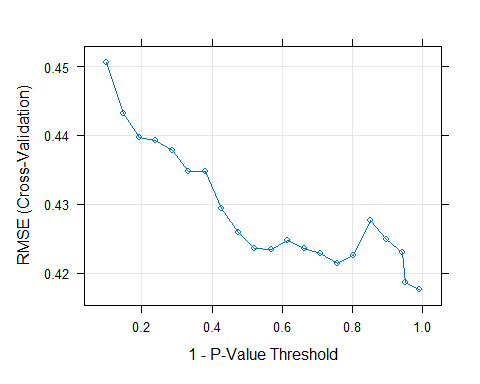


tree\_Imp = varImp(diabetes\_tree, scale = FALSE, competes = FALSE)  
#tree\_Imp  
plot(tree\_Imp)

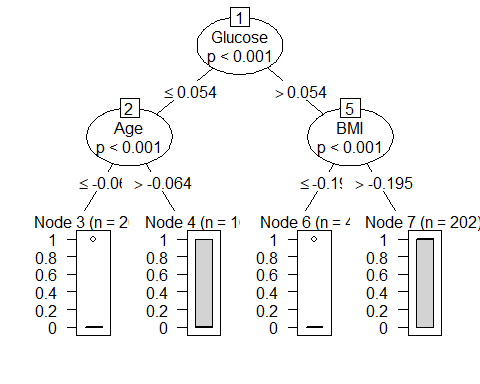


### Conditional Inference Tree

#diabetes\_ctree  
  
plot(diabetes\_ctree)



plot(diabetes\_ctree$finalModel)



### Bagged Tree

set.seed(100)  
  
diabetes\_bagged = train(x = x\_train, y = diabetes\_train,  
 method = "treebag",  
 nbagg = 50,  
 preProcess = c('center', 'scale', 'BoxCox', 'spatialSign'),  
 trControl = ctrl)

## Warning in train.default(x = x\_train, y = diabetes\_train, method = "treebag", :  
## You are trying to do regression and your outcome only has two possible values  
## Are you trying to do classification? If so, use a 2 level factor as your  
## outcome column.

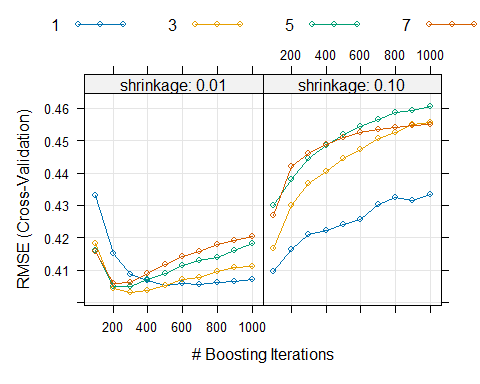
diabetes\_bagged

## Bagged CART   
##   
## 615 samples  
## 8 predictor  
##   
## Pre-processing: centered (8), scaled (8), Box-Cox transformation (2),  
## spatial sign transformation (8)   
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 554, 554, 553, 553, 554, 554, ...   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 0.4108167 0.2659006 0.3269407

diabetes\_results$Bagged = predict(diabetes\_bagged, x\_test)

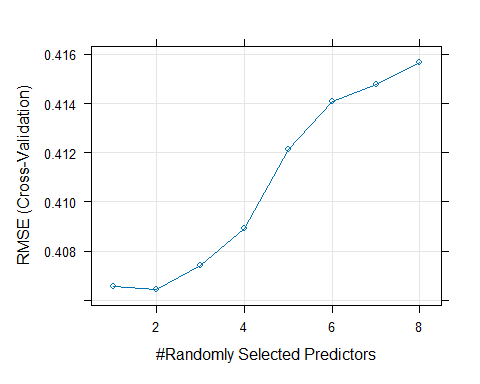
### Boosting

#diabetes\_boost  
plot(diabetes\_boost, auto.key = list(columns = 4, lines = TRUE))

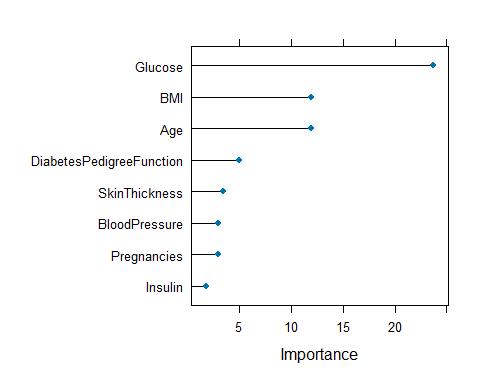


### Random Forest

#diabetes\_rf  
plot(diabetes\_rf)

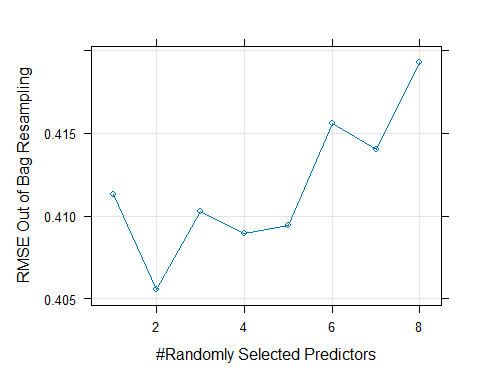


rf\_Imp = varImp(diabetes\_rf, scale = FALSE)  
plot(rf\_Imp)

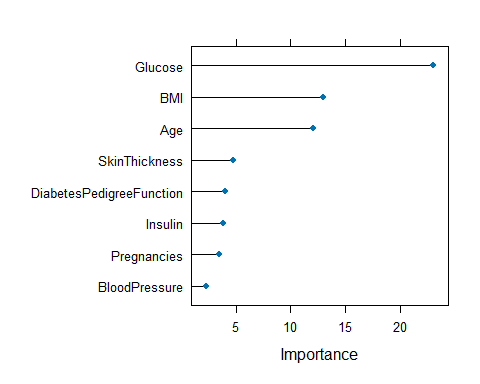


### Random Forest Tuned with OOB Estimates

#diabetes\_rfOOB  
plot(diabetes\_rfOOB)

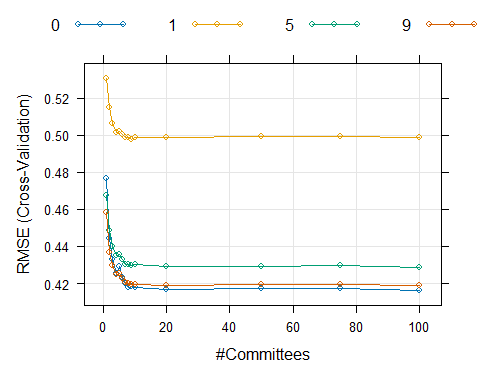


rfOOB\_Imp = varImp(diabetes\_rfOOB, scale = FALSE)  
plot(rfOOB\_Imp)

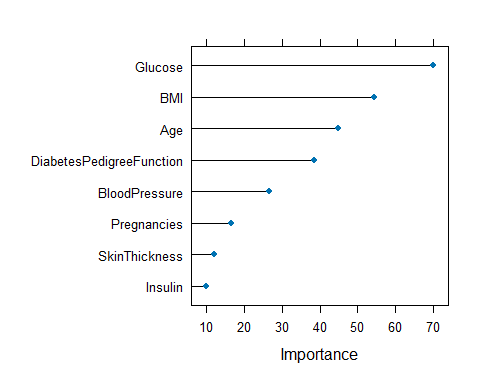


### Cubist

#diabetes\_cubist  
plot(diabetes\_cubist, auto.key = list(columns = 4, lines = TRUE))



cb\_Imp = varImp(diabetes\_cubist, scale = FALSE)  
#cb\_Imp  
plot(cb\_Imp)



### Continuous Model Performance Results

modelPerf = data.frame(rbind(OLS = postResample(pred = diabetes\_results$LM, obs = diabetes\_results$obs),  
 PLS = postResample(pred = diabetes\_results$PLS, obs = diabetes\_results$obs),  
 PCR = postResample(pred = diabetes\_results$PCR, obs = diabetes\_results$obs),  
 RIDGE = postResample(pred = diabetes\_results$RIDGE, obs = diabetes\_results$obs),  
 ENET = postResample(pred = diabetes\_results$ENET, obs = diabetes\_results$obs),  
 NNET = postResample(pred = diabetes\_results$NNET, obs = diabetes\_results$obs),  
 MARS = postResample(pred = diabetes\_results$MARS, obs = diabetes\_results$obs),  
 SVMr = postResample(pred = diabetes\_results$SVMr, obs = diabetes\_results$obs),  
 SVMp = postResample(pred = diabetes\_results$SVMp, obs = diabetes\_results$obs),  
 KNN = postResample(pred = diabetes\_results$KNN, obs = diabetes\_results$obs),  
 TREE = postResample(pred = diabetes\_results$TREE, obs = diabetes\_results$obs),  
 CTREE = postResample(pred = diabetes\_results$CTREE, obs = diabetes\_results$obs),  
 Bagged = postResample(pred = diabetes\_results$Bagged, obs = diabetes\_results$obs),  
 Boosting = postResample(pred = diabetes\_results$Boosting, obs = diabetes\_results$obs),  
 RF = postResample(pred = diabetes\_results$RF, obs = diabetes\_results$obs),  
 RF\_OOB = postResample(pred = diabetes\_results$RF\_OOF, obs = diabetes\_results$obs),  
 Cubist = postResample(pred = diabetes\_results$Cubist, obs = diabetes\_results$obs)  
 ))  
  
modelPerf[order(modelPerf$RMSE),]

## RMSE Rsquared MAE  
## SVMp 0.3708600 0.4190313 0.2885276  
## NNET 0.3714095 0.4155444 0.2997784  
## RIDGE 0.3756752 0.3944619 0.3197081  
## OLS 0.3759304 0.3958001 0.3204703  
## PCR 0.3759304 0.3958001 0.3204703  
## ENET 0.3770866 0.3930822 0.3214585  
## PLS 0.3772633 0.3905902 0.3213515  
## MARS 0.3899699 0.3421542 0.3287546  
## Boosting 0.3902915 0.3482048 0.3166133  
## SVMr 0.3913172 0.3688944 0.2772273  
## KNN 0.3914698 0.3447018 0.3172891  
## Bagged 0.3940577 0.3319123 0.3122140  
## Cubist 0.3945550 0.3416291 0.2952572  
## RF 0.4006786 0.3109951 0.3234270  
## TREE 0.4281954 0.2108013 0.3443732  
## CTREE 0.4281954 0.2108013 0.3443732  
## RF\_OOB NA NA NA

## Classification Response Models

## Classification Response Train Control

set.seed(100)  
class\_ctrl = trainControl(method = "LGOCV",  
 summaryFunction = twoClassSummary,  
 classProbs = TRUE,  
 savePredictions = TRUE)

## Convert Response Variables to Factor

diabetes\_train\_class = as.factor(make.names(diabetes\_train))  
diabetes\_test\_class = as.factor(make.names(diabetes\_test))  
  
diabetes\_results$obs\_class = diabetes\_test\_class

## Discriminant and Linear Classification Models

### Logistic Regression

set.seed(100)  
  
diabetes\_logistic = train(x = x\_train, y = diabetes\_train\_class,  
 method = "glm",  
 metric = "ROC",  
 preProcess = c('center', 'scale', 'BoxCox', 'spatialSign'),  
 trControl = class\_ctrl)  
  
diabetes\_logistic

## Generalized Linear Model   
##   
## 615 samples  
## 8 predictor  
## 2 classes: 'X0', 'X1'   
##   
## Pre-processing: centered (8), scaled (8), Box-Cox transformation (2),  
## spatial sign transformation (8)   
## Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)   
## Summary of sample sizes: 462, 462, 462, 462, 462, 462, ...   
## Resampling results:  
##   
## ROC Sens Spec   
## 0.8106189 0.8464 0.6007547

diabetes\_results$Log = predict(diabetes\_logistic, x\_test)  
diabetes\_results$Log\_prob = predict(diabetes\_logistic, x\_test, type = 'prob')[,1]

### Linear discriminant analysis

set.seed(100)  
  
diabetes\_lda = train(x = x\_train, y = diabetes\_train\_class,  
 method = "lda",  
 metric = "ROC",  
 preProcess = c('center', 'scale', 'BoxCox', 'spatialSign'),  
 trControl = class\_ctrl)  
  
diabetes\_lda

## Linear Discriminant Analysis   
##   
## 615 samples  
## 8 predictor  
## 2 classes: 'X0', 'X1'   
##   
## Pre-processing: centered (8), scaled (8), Box-Cox transformation (2),  
## spatial sign transformation (8)   
## Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)   
## Summary of sample sizes: 462, 462, 462, 462, 462, 462, ...   
## Resampling results:  
##   
## ROC Sens Spec   
## 0.8102113 0.8376 0.6158491

diabetes\_results$LDA = predict(diabetes\_lda, x\_test)  
diabetes\_results$LDA\_prob = predict(diabetes\_lda, x\_test, type = 'prob')[,1]

### Partial least squares discriminant analysis

set.seed(100)  
   
diabetes\_plsda = train(x = x\_train, y = diabetes\_train\_class,  
 method = "pls",  
 metric = "ROC",  
 tuneGrid = expand.grid(.ncomp = 1:5),  
 preProcess = c('center', 'scale', 'BoxCox', 'spatialSign'),  
 trControl = class\_ctrl)  
  
diabetes\_plsda

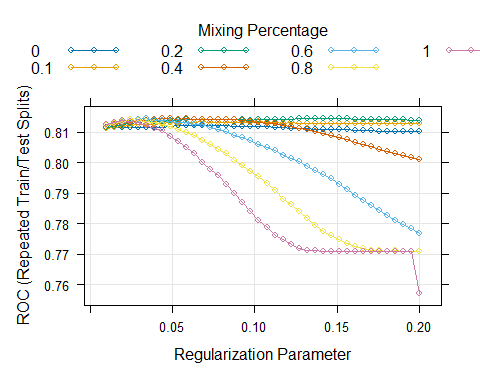
## Partial Least Squares   
##   
## 615 samples  
## 8 predictor  
## 2 classes: 'X0', 'X1'   
##   
## Pre-processing: centered (8), scaled (8), Box-Cox transformation (2),  
## spatial sign transformation (8)   
## Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)   
## Summary of sample sizes: 462, 462, 462, 462, 462, 462, ...   
## Resampling results across tuning parameters:  
##   
## ncomp ROC Sens Spec   
## 1 0.7998491 0.8144 0.6158491  
## 2 0.8099547 0.8328 0.6037736  
## 3 0.8100755 0.8400 0.6022642  
## 4 0.8104981 0.8432 0.6037736  
## 5 0.8101962 0.8408 0.6015094  
##   
## ROC was used to select the optimal model using the largest value.  
## The final value used for the model was ncomp = 4.

diabetes\_results$PLSDA = predict(diabetes\_plsda, x\_test)  
diabetes\_results$PLSDA\_prob = predict(diabetes\_plsda, x\_test, type = 'prob')[,1]

## Penalized Models & Nearest Shrunken Centroids

### Penalized Models

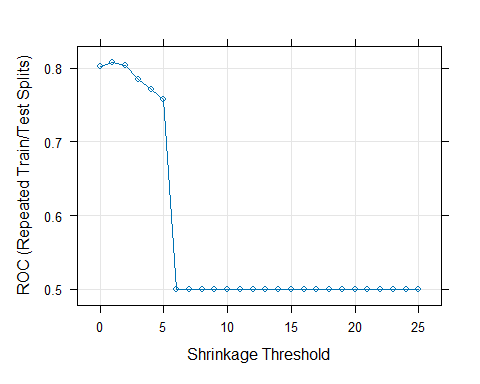
set.seed(100)   
  
glmnGrid = expand.grid(.alpha = c(0, .1, .2, .4, .6, .8, 1),  
 .lambda = seq(.01, .2, length = 40))  
   
diabetes\_glmnet = train(x\_train, y = diabetes\_train\_class,  
 method = "glmnet",  
 tuneGrid = glmnGrid,  
 metric = "ROC",  
 preProcess = c('center', 'scale', 'BoxCox', 'spatialSign'),  
 trControl = class\_ctrl)  
  
#diabetes\_glmnet  
plot(diabetes\_glmnet)



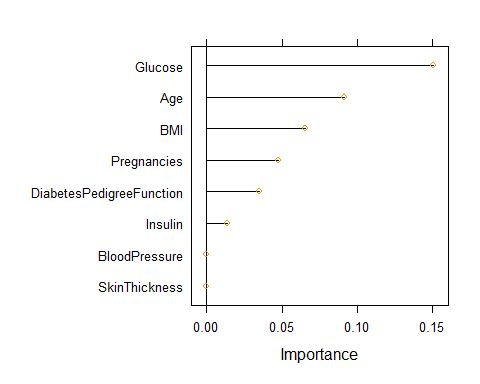
diabetes\_results$GLMNET = predict(diabetes\_glmnet, x\_test)  
diabetes\_results$GLMNET\_prob = predict(diabetes\_glmnet, x\_test, type = 'prob')[,1]

### Nearest Shrunken Centroids

#diabetes\_nsc  
plot(diabetes\_nsc)



plot(varImp(diabetes\_nsc, scale =FALSE))



## Non-Linear Classification Models

### Quadratic discriminant analysis

set.seed(100)  
  
diabetes\_qda = train(x = x\_train, y = diabetes\_train\_class,  
 method = "qda",  
 preProcess = c('center', 'scale', 'BoxCox', 'spatialSign'),  
 metric = "ROC",  
 trControl = class\_ctrl)  
  
diabetes\_qda

## Quadratic Discriminant Analysis   
##   
## 615 samples  
## 8 predictor  
## 2 classes: 'X0', 'X1'   
##   
## Pre-processing: centered (8), scaled (8), Box-Cox transformation (2),  
## spatial sign transformation (8)   
## Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)   
## Summary of sample sizes: 462, 462, 462, 462, 462, 462, ...   
## Resampling results:  
##   
## ROC Sens Spec   
## 0.8040453 0.7904 0.6762264

diabetes\_results$QDA = predict(diabetes\_qda, x\_test)  
diabetes\_results$QDA\_prob = predict(diabetes\_qda, x\_test, type = 'prob')[,1]

### Regularized discriminant analysis

set.seed(100)  
  
diabetes\_rda = train(x = x\_train, y = diabetes\_train\_class,  
 method = "rda",  
 preProcess = c('center', 'scale', 'BoxCox', 'spatialSign'),   
 metric = "ROC",  
 trControl = class\_ctrl)  
  
diabetes\_rda

## Regularized Discriminant Analysis   
##   
## 615 samples  
## 8 predictor  
## 2 classes: 'X0', 'X1'   
##   
## Pre-processing: centered (8), scaled (8), Box-Cox transformation (2),  
## spatial sign transformation (8)   
## Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)   
## Summary of sample sizes: 462, 462, 462, 462, 462, 462, ...   
## Resampling results across tuning parameters:  
##   
## gamma lambda ROC Sens Spec   
## 0.0 0.0 0.8040453 0.7904 0.6762264  
## 0.0 0.5 0.8121660 0.8136 0.6566038  
## 0.0 1.0 0.8102113 0.8376 0.6158491  
## 0.5 0.0 0.8107849 0.7928 0.6769811  
## 0.5 0.5 0.8114491 0.8004 0.6633962  
## 0.5 1.0 0.8092453 0.8108 0.6422642  
## 1.0 0.0 0.7997434 0.7668 0.6716981  
## 1.0 0.5 0.7997736 0.7656 0.6732075  
## 1.0 1.0 0.7998491 0.7644 0.6747170  
##   
## ROC was used to select the optimal model using the largest value.  
## The final values used for the model were gamma = 0 and lambda = 0.5.

diabetes\_results$RDA = predict(diabetes\_rda, x\_test)  
diabetes\_results$RDA\_prob = predict(diabetes\_rda, x\_test, type = 'prob')[,1]

### Mixture discriminant analysis

set.seed(100)  
  
diabetes\_mda = train(x = x\_train, y = diabetes\_train\_class,  
 method = "mda",  
 tuneGrid = expand.grid(.subclasses = 1:8),  
 preProcess = c('center', 'scale', 'BoxCox', 'spatialSign'),   
 metric = "ROC",  
 trControl = class\_ctrl)  
  
diabetes\_mda

## Mixture Discriminant Analysis   
##   
## 615 samples  
## 8 predictor  
## 2 classes: 'X0', 'X1'   
##   
## Pre-processing: centered (8), scaled (8), Box-Cox transformation (2),  
## spatial sign transformation (8)   
## Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)   
## Summary of sample sizes: 462, 462, 462, 462, 462, 462, ...   
## Resampling results across tuning parameters:  
##   
## subclasses ROC Sens Spec   
## 1 0.8102113 0.8376 0.6158491  
## 2 0.7994189 0.8048 0.6264151  
## 3 0.7996981 0.7840 0.6754717  
## 4 0.7973057 0.7816 0.6581132  
## 5 0.7963547 0.7824 0.6513208  
## 6 0.7816906 0.7880 0.6166038  
## 7 0.7818038 0.7812 0.6233962  
## 8 0.7760226 0.7940 0.6150943  
##   
## ROC was used to select the optimal model using the largest value.  
## The final value used for the model was subclasses = 1.

diabetes\_results$MDA = predict(diabetes\_mda, x\_test)  
diabetes\_results$MDA\_prob = predict(diabetes\_mda, x\_test, type = 'prob')[,1]

### Naïve Bayes

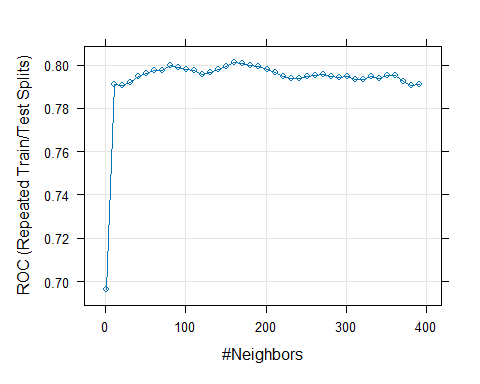
set.seed(100)  
  
diabetes\_nb = train(x = x\_train, y = diabetes\_train\_class,  
 method = "nb",  
 preProcess = c('center', 'scale', 'BoxCox', 'spatialSign'),   
 metric = "ROC",  
 trControl = class\_ctrl)  
  
diabetes\_nb

## Naive Bayes   
##   
## 615 samples  
## 8 predictor  
## 2 classes: 'X0', 'X1'   
##   
## Pre-processing: centered (8), scaled (8), Box-Cox transformation (2),  
## spatial sign transformation (8)   
## Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)   
## Summary of sample sizes: 462, 462, 462, 462, 462, 462, ...   
## Resampling results across tuning parameters:  
##   
## usekernel ROC Sens Spec   
## FALSE 0.8013585 0.7836 0.6762264  
## TRUE 0.8036075 0.7848 0.6898113  
##   
## Tuning parameter 'fL' was held constant at a value of 0  
## Tuning  
## parameter 'adjust' was held constant at a value of 1  
## ROC was used to select the optimal model using the largest value.  
## The final values used for the model were fL = 0, usekernel = TRUE and adjust  
## = 1.

diabetes\_results$NB = predict(diabetes\_nb, x\_test)  
diabetes\_results$NB\_prob = predict(diabetes\_nb, x\_test, type = 'prob')[,1]

### K-nearest neighbors

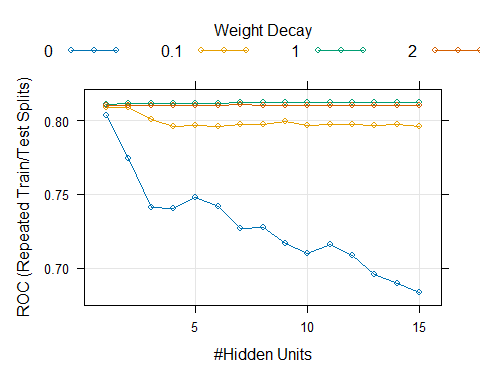
set.seed(100)  
  
diabetes\_class\_knn = train(x = x\_train, y = diabetes\_train\_class,  
 method = "knn",  
 metric = "ROC",  
 preProcess = c('center', 'scale', 'BoxCox', 'spatialSign'),   
 tuneGrid = data.frame(.k = seq(1,400, by=10)),  
 trControl = class\_ctrl)  
  
#diabetes\_class\_knn  
plot(diabetes\_class\_knn)



diabetes\_results$CLASS\_KNN = predict(diabetes\_class\_knn, x\_test)  
diabetes\_results$CLASS\_KNN\_prob = predict(diabetes\_class\_knn, x\_test, type = 'prob')[,1]

### Neural networks

set.seed(100)  
  
nnetGrid\_class = expand.grid(.size = 1:15,  
 .decay = c(0, .1, 1, 2))  
  
#maxSize\_class = max(nnetGrid\_class$.size)  
  
diabetes\_class\_nnet <- train(x = x\_train, y = diabetes\_train\_class,  
 method = "nnet",  
 metric = "ROC",  
 preProcess = c('center', 'scale', 'BoxCox', 'spatialSign'),   
 tuneGrid = nnetGrid\_class,  
 trace = FALSE,  
 maxit = 2000,  
 MaxNWts = 200,  
 trControl = class\_ctrl)  
  
#diabetes\_class\_nnet  
plot(diabetes\_class\_nnet)



diabetes\_results$CLASS\_NNET = predict(diabetes\_class\_nnet, x\_test)  
diabetes\_results$CLASS\_NNET\_prob = predict(diabetes\_class\_nnet, x\_test, type = 'prob')[,1]

### Flexible discriminant analysis

set.seed(100)  
  
diabetes\_FDA = train(x = x\_train, y = diabetes\_train\_class,  
 method = "fda",  
 preProcess = c('center', 'scale', 'BoxCox', 'spatialSign'),   
 metric = "ROC",  
 trControl = class\_ctrl)  
  
diabetes\_FDA

## Flexible Discriminant Analysis   
##   
## 615 samples  
## 8 predictor  
## 2 classes: 'X0', 'X1'   
##   
## Pre-processing: centered (8), scaled (8), Box-Cox transformation (2),  
## spatial sign transformation (8)   
## Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)   
## Summary of sample sizes: 462, 462, 462, 462, 462, 462, ...   
## Resampling results across tuning parameters:  
##   
## nprune ROC Sens Spec   
## 2 0.7708906 0.8584 0.4996226  
## 9 0.7965887 0.8312 0.5871698  
## 16 0.7961660 0.8276 0.5969811  
##   
## Tuning parameter 'degree' was held constant at a value of 1  
## ROC was used to select the optimal model using the largest value.  
## The final values used for the model were degree = 1 and nprune = 9.

diabetes\_results$FDA = predict(diabetes\_FDA, x\_test)  
diabetes\_results$FDA\_prob = predict(diabetes\_FDA, x\_test, type = 'prob')[,1]

### Support Vector Machines

set.seed(100)  
  
sigmaRangeReduced <- sigest(as.matrix(x\_train))  
svmRGridReduced <- expand.grid(.sigma = sigmaRangeReduced[1],  
 .C = 2^(seq(-4, 4)))  
  
diabetes\_class\_svmr = train(x = x\_train, y = diabetes\_train\_class,  
 method = "svmRadial", #svmLinear #svmPoly #Radial is common in practice  
 metric = "ROC",  
 preProcess = c('center', 'scale', 'BoxCox', 'spatialSign'),   
 tuneGrid = svmRGridReduced,  
 fit = FALSE,  
 trControl = class\_ctrl)  
  
diabetes\_class\_svmr

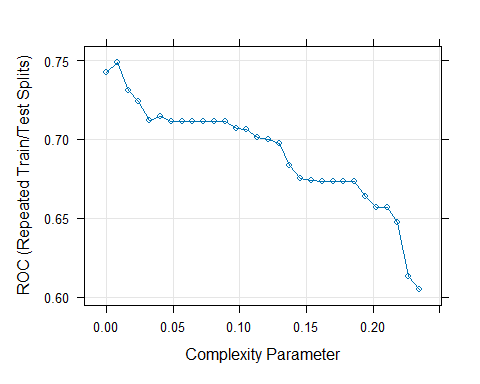
## Support Vector Machines with Radial Basis Function Kernel   
##   
## 615 samples  
## 8 predictor  
## 2 classes: 'X0', 'X1'   
##   
## Pre-processing: centered (8), scaled (8), Box-Cox transformation (2),  
## spatial sign transformation (8)   
## Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)   
## Summary of sample sizes: 462, 462, 462, 462, 462, 462, ...   
## Resampling results across tuning parameters:  
##   
## C ROC Sens Spec   
## 0.0625 0.8157509 0.7732 0.6950943  
## 0.1250 0.8157811 0.7812 0.6928302  
## 0.2500 0.8163547 0.8424 0.6075472  
## 0.5000 0.8163849 0.8656 0.5683019  
## 1.0000 0.8142491 0.8732 0.5562264  
## 2.0000 0.8103849 0.8724 0.5381132  
## 4.0000 0.8066340 0.8760 0.5283019  
## 8.0000 0.8010415 0.8816 0.5139623  
## 16.0000 0.7946717 0.8832 0.4943396  
##   
## Tuning parameter 'sigma' was held constant at a value of 0.0332444  
## ROC was used to select the optimal model using the largest value.  
## The final values used for the model were sigma = 0.0332444 and C = 0.5.

diabetes\_results$CLASS\_SVMr = predict(diabetes\_class\_svmr, x\_test)  
diabetes\_results$CLASS\_SVMr\_prob = predict(diabetes\_class\_svmr, x\_test, type = 'prob')[,1]

## Classification Trees

### Classification Tree

set.seed(100)  
  
diabetes\_class\_tree = train(x = x\_train, y = diabetes\_train\_class,  
 method = 'rpart',  
 metric = 'ROC',  
 preProcess = c('center', 'scale', 'BoxCox', 'spatialSign'),  
 tuneLength = 30,  
 trControl = class\_ctrl)  
  
#diabetes\_class\_tree  
plot(diabetes\_class\_tree)



diabetes\_results$CLASS\_TREE = predict(diabetes\_class\_tree, x\_test)  
diabetes\_results$CLASS\_TREE\_prob = predict(diabetes\_class\_tree, x\_test, type = 'prob')[,1]

### Bagged Classification Tree

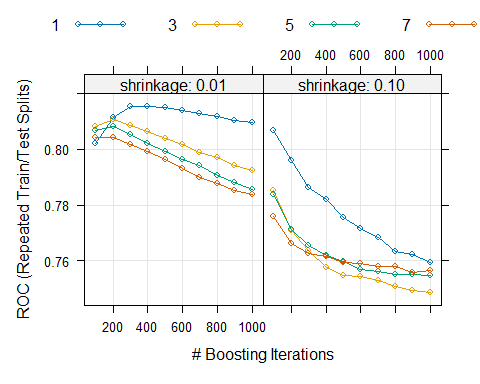
set.seed(100)  
  
diabetes\_class\_bagged = train(x = x\_train, y = diabetes\_train\_class,  
 method = 'treebag',  
 metric = 'ROC',  
 preProcess = c('center', 'scale', 'BoxCox', 'spatialSign'),  
 nbagg = 50,  
 trControl = class\_ctrl)  
  
diabetes\_class\_bagged

## Bagged CART   
##   
## 615 samples  
## 8 predictor  
## 2 classes: 'X0', 'X1'   
##   
## Pre-processing: centered (8), scaled (8), Box-Cox transformation (2),  
## spatial sign transformation (8)   
## Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)   
## Summary of sample sizes: 462, 462, 462, 462, 462, 462, ...   
## Resampling results:  
##   
## ROC Sens Spec   
## 0.7814415 0.8236 0.5811321

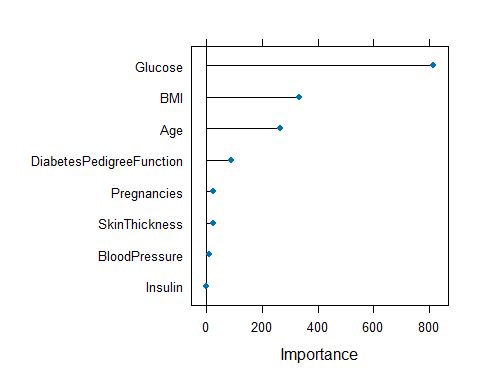
diabetes\_results$CLASS\_Bagged = predict(diabetes\_class\_bagged, x\_test)  
diabetes\_results$CLASS\_Bagged\_prob = predict(diabetes\_class\_bagged, x\_test, type = 'prob')[,1]

### Classification Boosting

set.seed(100)  
  
gbmGrid\_class = expand.grid( interaction.depth = seq( 1, 7, by=2 ),  
 n.trees = seq( 100, 1000, by=100 ),  
 shrinkage = c(0.01, 0.1),  
 n.minobsinnode = 10 )  
  
diabetes\_class\_boosting = train(x = x\_train, y = diabetes\_train\_class,  
 method = 'gbm',  
 metric = 'ROC',  
 preProcess = c('center', 'scale', 'BoxCox', 'spatialSign'),  
 tuneGrid = gbmGrid\_class,  
 trControl = class\_ctrl,  
 verbose = FALSE)  
  
#diabetes\_class\_boosting  
plot(diabetes\_class\_boosting, auto.key = list(columns = 4, lines = TRUE))



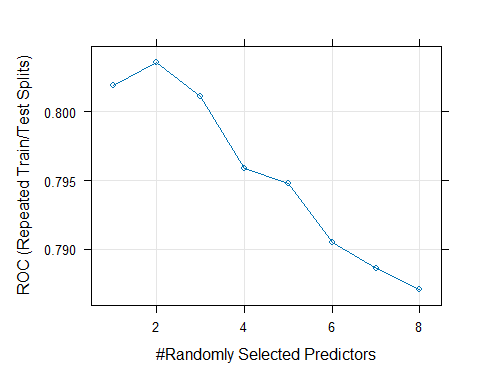
plot(varImp(diabetes\_class\_boosting, scale = FALSE))



diabetes\_results$CLASS\_Boosting = predict(diabetes\_class\_boosting, x\_test)  
diabetes\_results$CLASS\_Boosting\_prob = predict(diabetes\_class\_boosting, x\_test, type = 'prob')[,1]

### Classification Random Forest

set.seed(100)  
  
mtryGrid\_class = data.frame(mtry = 1:8)  
  
diabetes\_class\_rf = train(x = x\_train, y = diabetes\_train\_class,  
 method = 'rf',  
 metric = 'ROC',  
 preProcess = c('center', 'scale', 'BoxCox', 'spatialSign'),  
 tuneGrid = mtryGrid\_class,  
 ntree = 200,  
 importance = TRUE,  
 trControl = class\_ctrl)  
  
#diabetes\_class\_rf  
plot(diabetes\_class\_rf)



diabetes\_results$CLASS\_RF = predict(diabetes\_class\_rf, x\_test)  
diabetes\_results$CLASS\_RF\_prob = predict(diabetes\_class\_rf, x\_test, type = 'prob')[,1]

## Model Performance and Comparison

### ROC

# ROC for logistic model  
plot(roc(diabetes\_results$obs\_class, diabetes\_results$Log\_prob), col=1, lty=1, lwd=2)

## Setting levels: control = X0, case = X1

## Setting direction: controls > cases

# ROC for LDA  
lines(roc(diabetes\_results$obs\_class, diabetes\_results$LDA\_prob), col=2, lty=2, lwd=2)

## Setting levels: control = X0, case = X1  
## Setting direction: controls > cases

# ROC for PLSDA  
lines(roc(diabetes\_results$obs\_class, diabetes\_results$PLSDA\_prob), col=3, lty=3, lwd=2)

## Setting levels: control = X0, case = X1  
## Setting direction: controls > cases

# ROC for penalized model  
lines(roc(diabetes\_results$obs\_class, diabetes\_results$GLMNET\_prob), col=4, lty=4, lwd=2)

## Setting levels: control = X0, case = X1  
## Setting direction: controls > cases

# ROC for NSC  
lines(roc(diabetes\_results$obs\_class, diabetes\_results$NSC\_prob), col=5, lty=5, lwd=2)

## Setting levels: control = X0, case = X1  
## Setting direction: controls > cases

# ROC for Classification Tree  
lines(roc(diabetes\_results$obs\_class, diabetes\_results$CLASS\_TREE\_prob), col=6, lty=6, lwd=2)

## Setting levels: control = X0, case = X1  
## Setting direction: controls > cases

# ROC for Bagged Tree  
lines(roc(diabetes\_results$obs\_class, diabetes\_results$CLASS\_Bagged\_prob), col=7, lty=7, lwd=2)

## Setting levels: control = X0, case = X1  
## Setting direction: controls > cases

# ROC for Boosting  
lines(roc(diabetes\_results$obs\_class, diabetes\_results$CLASS\_Boosting\_prob), col=8, lty=8, lwd=2)

## Setting levels: control = X0, case = X1  
## Setting direction: controls > cases

# ROC for Random Forest  
lines(roc(diabetes\_results$obs\_class, diabetes\_results$CLASS\_RF\_prob), col=9, lty=9, lwd=2)

## Setting levels: control = X0, case = X1  
## Setting direction: controls > cases

# ROC for QDA  
lines(roc(diabetes\_results$obs\_class, diabetes\_results$QDA\_prob), col=10, lty=10, lwd=2)

## Setting levels: control = X0, case = X1  
## Setting direction: controls > cases

# ROC for RDA  
lines(roc(diabetes\_results$obs\_class, diabetes\_results$RDA\_prob), col=11, lty=11, lwd=2)

## Setting levels: control = X0, case = X1  
## Setting direction: controls > cases

# ROC for MDA  
lines(roc(diabetes\_results$obs\_class, diabetes\_results$MDA\_prob), col=12, lty=12, lwd=2)

## Setting levels: control = X0, case = X1  
## Setting direction: controls > cases

# ROC for NB  
lines(roc(diabetes\_results$obs\_class, diabetes\_results$NB\_prob), col=13, lty=13, lwd=2)

## Setting levels: control = X0, case = X1  
## Setting direction: controls > cases

# ROC for KNN  
lines(roc(diabetes\_results$obs\_class, diabetes\_results$CLASS\_KNN\_prob), col=14, lty=14, lwd=2)

## Setting levels: control = X0, case = X1  
## Setting direction: controls > cases

# ROC for NNET  
lines(roc(diabetes\_results$obs\_class, diabetes\_results$CLASS\_NNET\_prob), col=15, lty=15, lwd=2)

## Setting levels: control = X0, case = X1  
## Setting direction: controls > cases

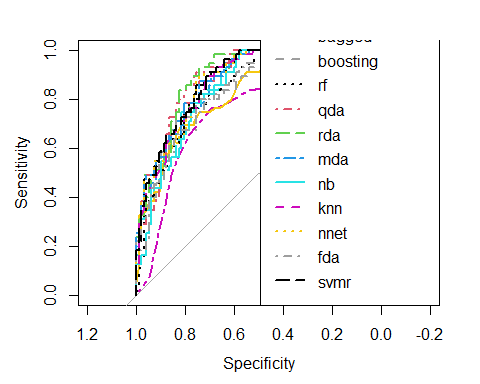
# ROC for FDA  
lines(roc(diabetes\_results$obs\_class, diabetes\_results$FDA\_prob), col=16, lty=16, lwd=2)

## Setting levels: control = X0, case = X1  
## Setting direction: controls > cases

# ROC for SVMr  
lines(roc(diabetes\_results$obs\_class, diabetes\_results$CLASS\_SVMr\_prob), col=17, lty=17, lwd=2)

## Setting levels: control = X0, case = X1  
## Setting direction: controls > cases

legend('bottomright',  
 c('logistic', 'lda', 'plsda',  
 'penalized model', 'nsc',   
 'tree', 'bagged', 'boosting',  
 'rf', 'qda', 'rda', 'mda', 'nb',   
 'knn', 'nnet', 'fda', 'svmr'),  
 col=1:17, lty=1:17,lwd=2)



### Model Accuracy

log\_conf = confusionMatrix(data = diabetes\_results$Log, reference = diabetes\_results$obs\_class)  
LDA\_conf = confusionMatrix(data = diabetes\_results$LDA, reference = diabetes\_results$obs\_class)  
PLSDA\_conf = confusionMatrix(data = diabetes\_results$PLSDA, reference = diabetes\_results$obs\_class)  
GLMNET\_conf = confusionMatrix(data = diabetes\_results$GLMNET, reference = diabetes\_results$obs\_class)  
NSC\_conf = confusionMatrix(data = diabetes\_results$NSC, reference = diabetes\_results$obs\_class)  
tree\_conf = confusionMatrix(data = diabetes\_results$CLASS\_TREE, reference = diabetes\_results$obs\_class)  
bagged\_conf = confusionMatrix(data = diabetes\_results$CLASS\_Bagged, reference = diabetes\_results$obs\_class)  
boosting\_conf = confusionMatrix(data = diabetes\_results$CLASS\_Boosting, reference = diabetes\_results$obs\_class)  
rf\_conf = confusionMatrix(data = diabetes\_results$CLASS\_RF, reference = diabetes\_results$obs\_class)  
QDA\_conf = confusionMatrix(data = diabetes\_results$QDA, reference = diabetes\_results$obs\_class)  
RDA\_conf = confusionMatrix(data = diabetes\_results$RDA, reference = diabetes\_results$obs\_class)  
MDA\_conf = confusionMatrix(data = diabetes\_results$MDA, reference = diabetes\_results$obs\_class)  
NB\_conf = confusionMatrix(data = diabetes\_results$NB, reference = diabetes\_results$obs\_class)  
KNN\_conf = confusionMatrix(data = diabetes\_results$CLASS\_KNN, reference = diabetes\_results$obs\_class)  
NNET\_conf = confusionMatrix(data = diabetes\_results$CLASS\_NNET, reference = diabetes\_results$obs\_class)  
FDA\_conf = confusionMatrix(data = diabetes\_results$FDA, reference = diabetes\_results$obs\_class)  
SVMr\_conf = confusionMatrix(data = diabetes\_results$CLASS\_SVMr, reference = diabetes\_results$obs\_class)  
  
model\_perf\_class = data.frame(rbind(log = c(log\_conf$overall[1:2], log\_conf$byClass[1:2]),  
 LDA = c(LDA\_conf$overall[1:2], LDA\_conf$byClass[1:2]),  
 PLSDA = c(PLSDA\_conf$overall[1:2], PLSDA\_conf$byClass[1:2]),  
 Penalized = c(GLMNET\_conf$overall[1:2], GLMNET\_conf$byClass[1:2]),  
 NSC = c(NSC\_conf$overall[1:2], NSC\_conf$byClass[1:2]),  
 Tree = c(tree\_conf$overall[1:2], tree\_conf$byClass[1:2]),  
 Bagged = c(bagged\_conf$overall[1:2], bagged\_conf$byClass[1:2]),  
 Boosting = c(boosting\_conf$overall[1:2], boosting\_conf$byClass[1:2]),  
 RF = c(rf\_conf$overall[1:2], rf\_conf$byClass[1:2]),  
 QDA = c(QDA\_conf$overall[1:2], QDA\_conf$byClass[1:2]),  
 RDA = c(RDA\_conf$overall[1:2], RDA\_conf$byClass[1:2]),  
 MDA = c(MDA\_conf$overall[1:2], MDA\_conf$byClass[1:2]),  
 NB = c(NB\_conf$overall[1:2], NB\_conf$byClass[1:2]),  
 KNN = c(KNN\_conf$overall[1:2], KNN\_conf$byClass[1:2]),  
 NNET = c(NNET\_conf$overall[1:2], NNET\_conf$byClass[1:2]),  
 FDA = c(FDA\_conf$overall[1:2], FDA\_conf$byClass[1:2]),  
 SVMr = c(SVMr\_conf$overall[1:2], SVMr\_conf$byClass[1:2])  
 ))  
  
model\_perf\_class[order(model\_perf\_class$Kappa, decreasing = TRUE),]

## Accuracy Kappa Sensitivity Specificity  
## QDA 0.8169935 0.59940153 0.8673469 0.72727273  
## PLSDA 0.7843137 0.50182536 0.9081633 0.56363636  
## LDA 0.7777778 0.48889762 0.8979592 0.56363636  
## MDA 0.7777778 0.48889762 0.8979592 0.56363636  
## log 0.7777778 0.48454221 0.9081633 0.54545455  
## NNET 0.7777778 0.48454221 0.9081633 0.54545455  
## FDA 0.7647059 0.48076923 0.8367347 0.63636364  
## NB 0.7516340 0.46913806 0.7857143 0.69090909  
## SVMr 0.7712418 0.46711116 0.9081633 0.52727273  
## Bagged 0.7647059 0.46336711 0.8775510 0.56363636  
## RDA 0.7581699 0.45076162 0.8673469 0.56363636  
## KNN 0.7712418 0.44328932 0.9591837 0.43636364  
## Boosting 0.7581699 0.43179765 0.9081633 0.49090909  
## RF 0.7516340 0.42390012 0.8877551 0.50909091  
## Penalized 0.7581699 0.41669243 0.9387755 0.43636364  
## Tree 0.7320261 0.39642067 0.8367347 0.54545455  
## NSC 0.6535948 0.04611222 1.0000000 0.03636364