P8106_yiminchen_secondaryanalysis

Yimin Chen (yc4195), Yang Yi (yy3307), Qingyue Zhuo (qz2493)

Contents

Import and data manipulation

```r
# Load recovery.RData environment
load("./recovery.Rdata")

dat %>% na.omit()

# dat1 draw a random sample of 2000 participants Uni:3307
set.seed(3307)

dat1 = dat[sample(1:10000, 2000),]

dat1 =
  dat1[, -1] %>%
  mutate(
    recovery_time = as.factor(
      case_when(recovery_time <= 30 ~ "long", recovery_time > 30 ~ "short")
    ),
    gender = as.factor(gender),
    race = as.factor(race),
    smoking = as.factor(smoking),
    hypertension = as.factor(hypertension),
    diabetes = as.factor(diabetes),
    vaccine = as.factor(vaccine),
    severity = as.factor(severity),
    study = as.factor(
      case_when(study == "A" ~ 1, study == "B" ~ 2, study == "C" ~ 3)
      )
    )

# dat2 draw a random sample of 2000 participants Uni:2493
set.seed(2493)

dat2 = dat[sample(1:10000, 2000),]

dat2 =
  dat2[, -1] %>%
  mutate(
    recovery_time = as.factor(
      case_when(recovery_time <= 30 ~ "long", recovery_time > 30 ~ "short")
    ),
    gender = as.factor(gender),
    race = as.factor(race),
    smoking = as.factor(smoking),
    hypertension = as.factor(hypertension),
    diabetes = as.factor(diabetes),
    vaccine = as.factor(vaccine),
    severity = as.factor(severity),
    study = as.factor(
      case_when(study == "A" ~ 1, study == "B" ~ 2, study == "C" ~ 3)
      )
    )

# Merged dataset with unique observation
covid_dat = rbind(dat1, dat2) %>%
  unique()

covid_dat2 = model.matrix(recovery_time ~ ., covid_dat)[, -1] #ignore intercept


# Partition dataset into two parts: training data (70%) and test data (30%)
rowTrain = createDataPartition(y = covid_dat$recovery_time, p = 0.7, list = FALSE)

trainData = covid_dat[rowTrain, ]
testData = covid_dat[-rowTrain, ]

# matrix of predictors
x1 = covid_dat2[rowTrain,]
# vector of response
y1 = covid_dat$recovery_time[rowTrain]
# matrix of predictors
x2 = covid_dat2[-rowTrain,]
```

```r
# vector of response
y2 = covid_dat$recovery_time[-rowTrain]

ctrl1 = trainControl(method = "repeatedcv", number = 10, repeats = 5)
ctrl2 = trainControl(method = "cv",
                          classProbs = TRUE,
                          summaryFunction = twoClassSummary)
```

Data visualization

Model training

classification

- glm + penalized logistice regreesion L8
- GAM L8
- MARS L8
- QDA L9
- LDA L9
- Navie Bayes L9
- classification tree: L11
- random forest L12
- boosting L12
- support vecotr machines L13

Logistic regression and its cousins

GLM

```r
set.seed(2)
model.glm <- train(x = covid_dat2[rowTrain,],
                   y = covid_dat$recovery_time[rowTrain],
                   method = "glm",
                   metric = "ROC",
                   trControl = ctrl2)
```

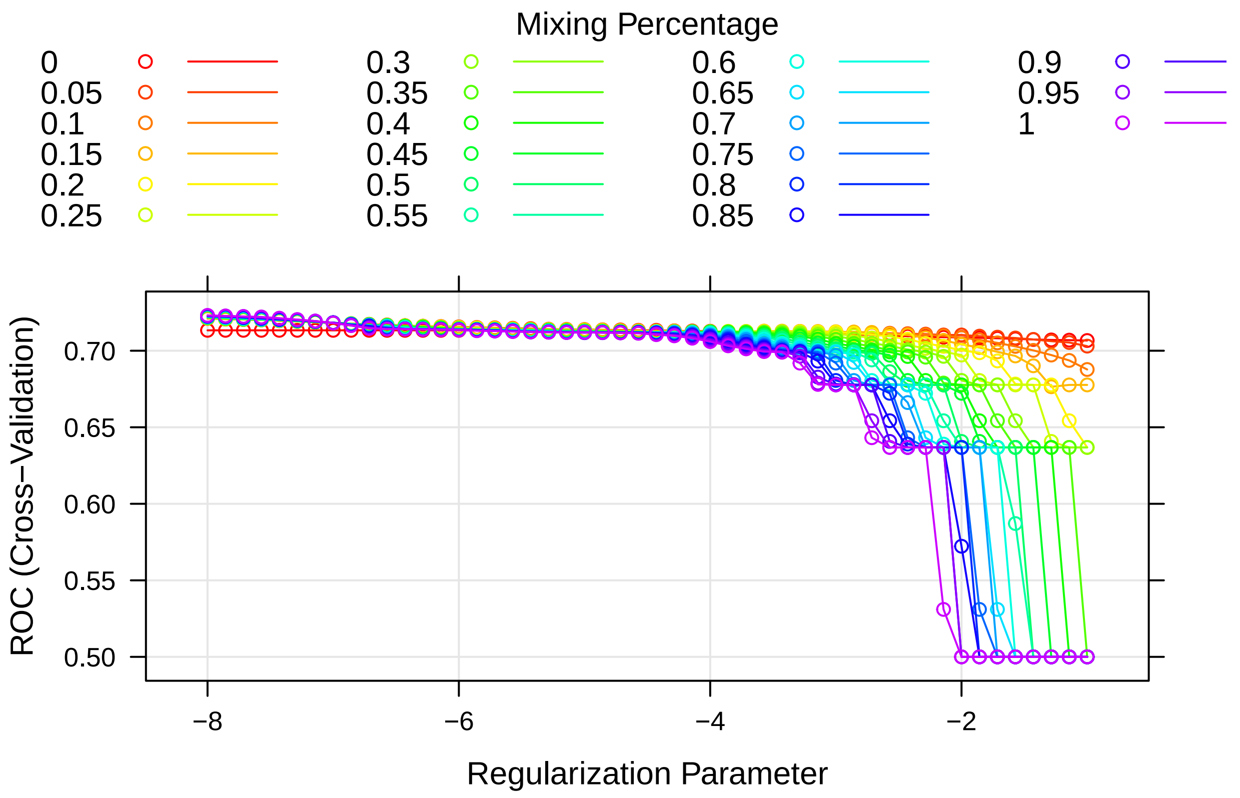
Penalized logistic regression

Penalized logistic regression can be fitted using `glmnet`. We use the `train` function to select the optimal tuning parameters.

```r
glmnGrid <- expand.grid(.alpha = seq(0, 1, length = 21),
                        .lambda = exp(seq(-8, -1, length = 50)))
set.seed(2)
model.glmn <- train(x = covid_dat2[rowTrain,],
                   y = covid_dat$recovery_time[rowTrain],
                    method = "glmnet",
                    tuneGrid = glmnGrid,
                    metric = "ROC",
                    trControl = ctrl2)

model.glmn$bestTune
```

```
##      alpha       lambda
## 1001     1 0.0003354626
```

```
myCol<- rainbow(25)
myPar <- list(superpose.symbol = list(col = myCol),
              superpose.line = list(col = myCol))

plot(model.glmn, par.settings = myPar, xTrans = function(x) log(x))
```



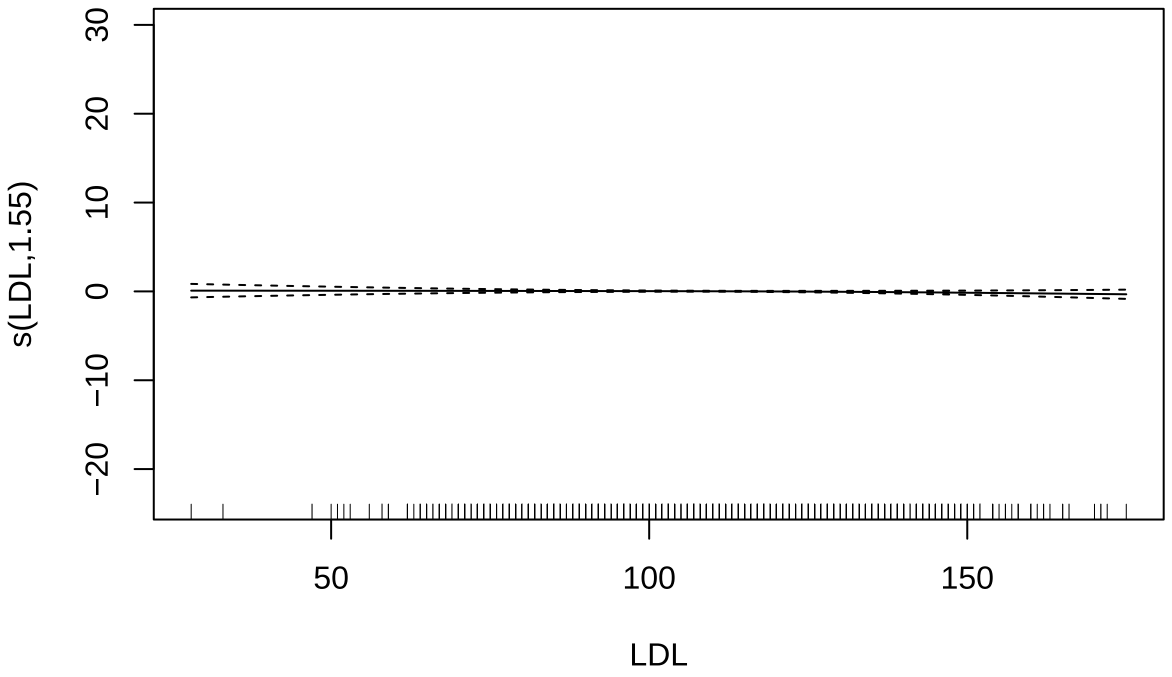
GAM

```
set.seed(2)
model.gam <- train(x = covid_dat2[rowTrain,],
                   y = covid_dat$recovery_time[rowTrain],
                   method = "gam",
                   metric = "ROC",
                   trControl = ctrl2)


model.gam$finalModel
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## .outcome ~ gender1 + race3 + race4 + smoking1 + smoking2 + hypertension1 +
##     diabetes1 + vaccine1 + severity1 + study2 + study3 + s(age) +
##     s(SBP) + s(LDL) + s(bmi) + s(height) + s(weight)
##
```

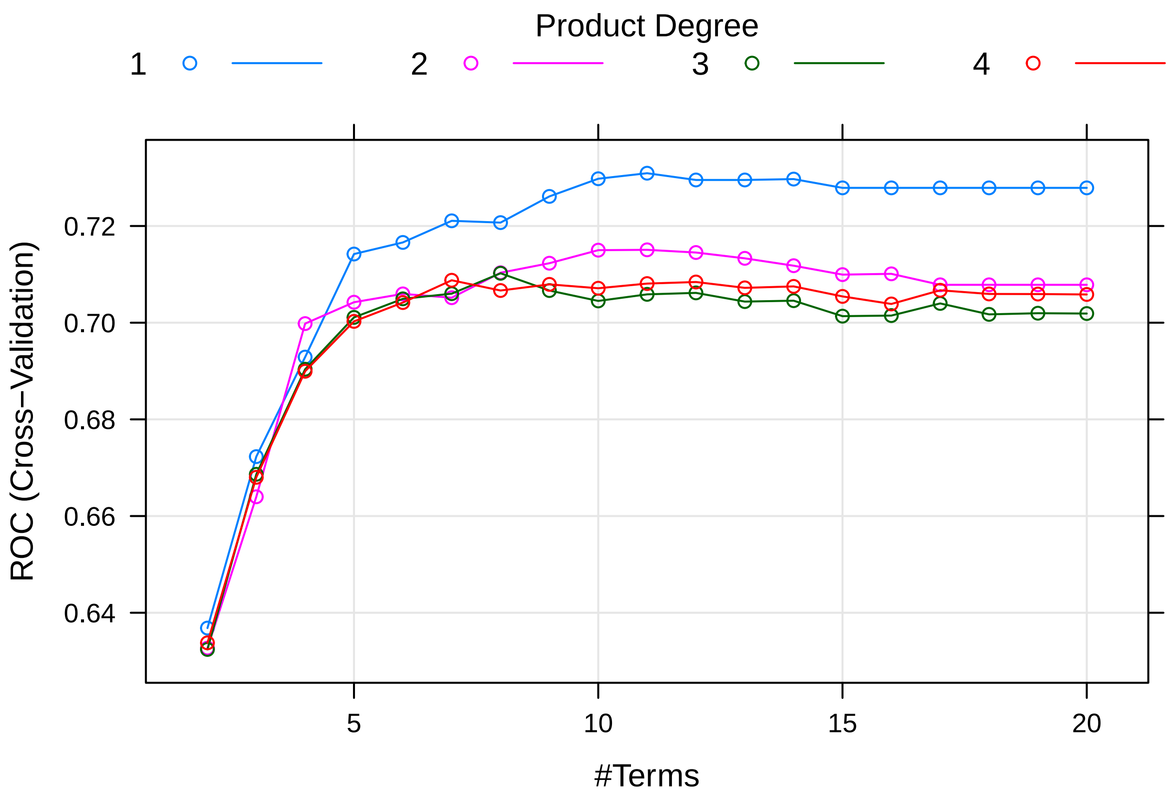
```
## Estimated degrees of freedom:
## 1.00 1.00 1.55 2.79 1.00 2.29  total = 21.63
##
## UBRE score: 0.06815373
```

```
plot(model.gam$finalModel, select = 3)
```



MARS

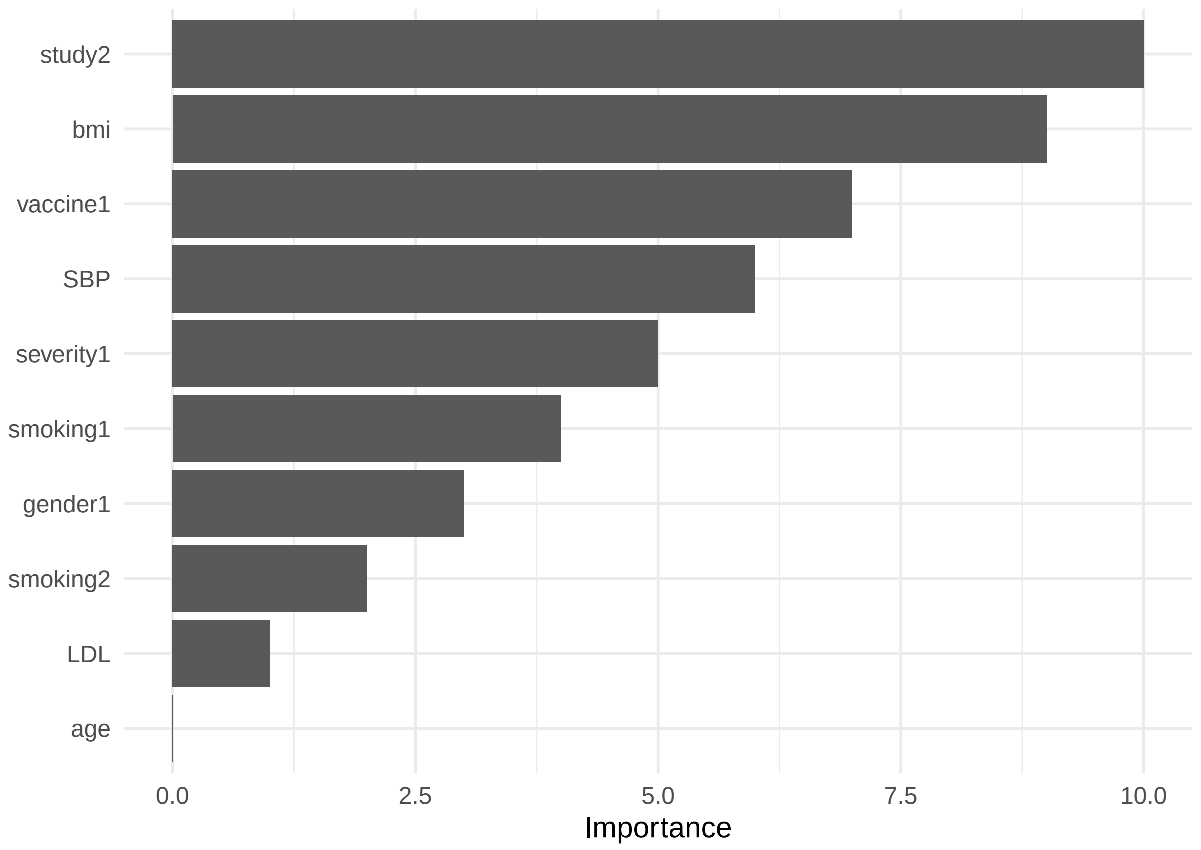
```
set.seed(2)
model.mars <- train(x = covid_dat2[rowTrain,],
                   y = covid_dat$recovery_time[rowTrain],
                    method = "earth",
                    tuneGrid = expand.grid(degree = 1:4,
                                           nprune = 2:20),
                    metric = "ROC",
                    trControl = ctrl2)

plot(model.mars)
```



```
coef(model.mars$finalModel)
```

```
## (Intercept)      study2 h(28.6-bmi)    vaccine1  h(135-SBP)   severity1
## -0.32524568 -1.35310824  0.51047027 -0.73109733 -0.03262848  0.80307433
##    smoking1     gender1    smoking2  h(LDL-145) h(bmi-23.1)
##  0.43021337 -0.32207625  0.55022116 -0.05342548  0.41456148
```

```
vip(model.mars$finalModel)
```

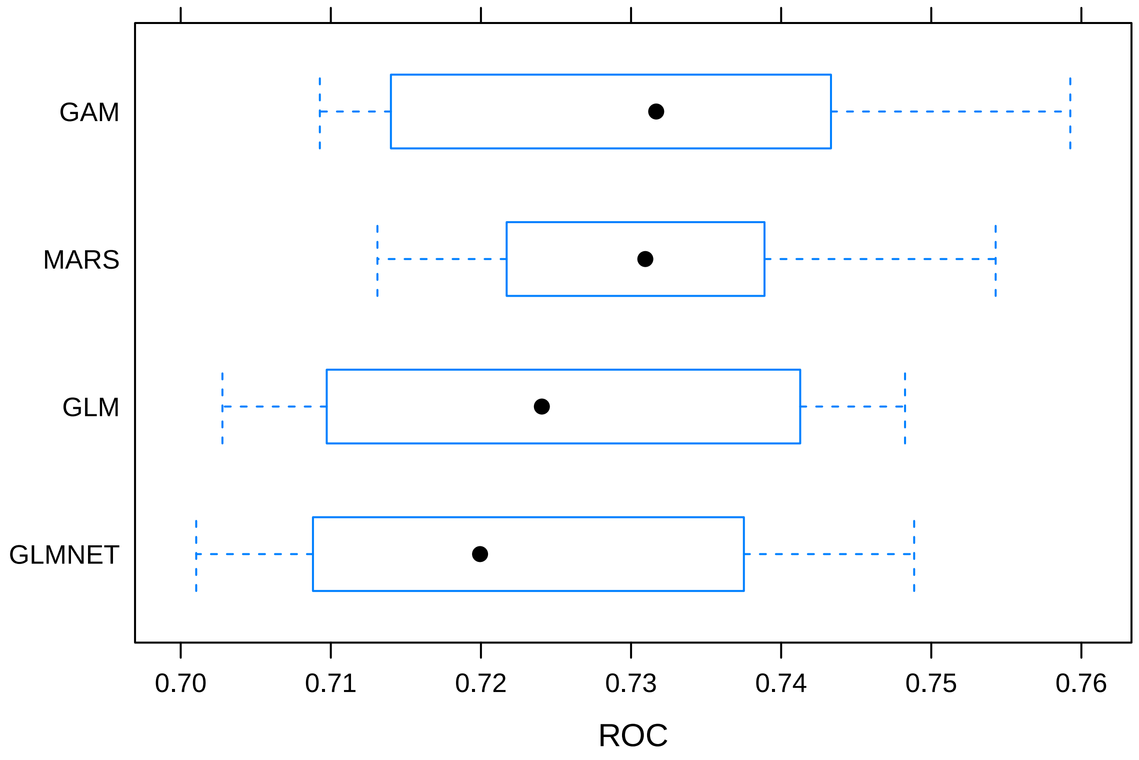


```r
res <- resamples(list(GLM = model.glm,
                      GLMNET = model.glmn,
                      GAM = model.gam,
                      MARS = model.mars))
summary(res)
```

```
##
## Call:
## summary.resamples(object = res)
##
## Models: GLM, GLMNET, GAM, MARS
## Number of resamples: 10
##
## ROC
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM    0.7027786 0.7100512 0.7240580 0.7243182 0.7396824 0.7482539    0
## GLMNET 0.7010325 0.7091541 0.7199426 0.7230782 0.7374113 0.7488612    0
## GAM    0.7092686 0.7156054 0.7316766 0.7319728 0.7432725 0.7592621    0
## MARS   0.7131036 0.7220759 0.7309513 0.7309203 0.7385390 0.7542894    0
##
## Sens
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM    0.2162162 0.2702703 0.2789708 0.2814698 0.3074324 0.3378378    0
## GLMNET 0.2027027 0.2466216 0.2837838 0.2693262 0.2969733 0.3243243    0
## GAM    0.2027027 0.2627730 0.3175676 0.3017031 0.3378378 0.3648649    0
## MARS   0.2297297 0.2837838 0.2924843 0.3044428 0.3344595 0.3783784    0
##
```

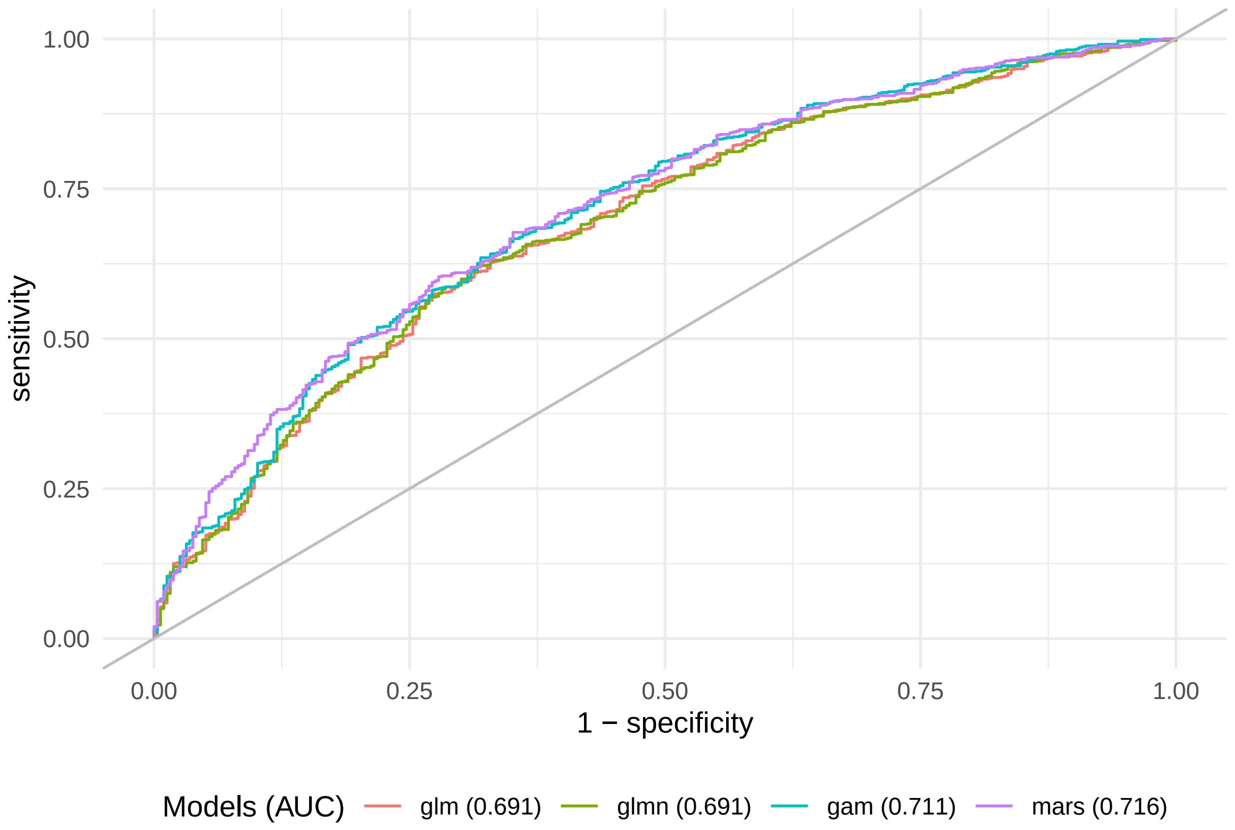
```
## Spec
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM    0.8531073 0.9039548 0.9154605 0.9119755 0.9324890 0.9438202    0
## GLMNET 0.8644068 0.9053672 0.9180791 0.9187425 0.9382022 0.9548023    0
## GAM    0.8644068 0.8884181 0.9124294 0.9018314 0.9157303 0.9269663    0
## MARS   0.8644068 0.8912429 0.9098584 0.9057862 0.9196106 0.9325843    0
```

```
bwplot(res, metric = "ROC")
```



test data performance for Logistic regression and its cousins

```
glm.pred <- predict(model.glm, newdata = covid_dat2[-rowTrain,], type = "prob")[,2]
glmn.pred <- predict(model.glmn, newdata = covid_dat2[-rowTrain,], type = "prob")[,2]
gam.pred <- predict(model.gam, newdata = covid_dat2[-rowTrain,], type = "prob")[,2]
mars.pred <- predict(model.mars, newdata = covid_dat2[-rowTrain,], type = "prob")[,2]

roc.glm <- roc(covid_dat$recovery_time[-rowTrain], glm.pred)
roc.glmn <- roc(covid_dat$recovery_time[-rowTrain], glmn.pred)
roc.gam <- roc(covid_dat$recovery_time[-rowTrain], gam.pred)
roc.mars <- roc(covid_dat$recovery_time[-rowTrain], mars.pred)

auc <- c(roc.glm$auc[1], roc.glmn$auc[1],
         roc.gam$auc[1], roc.mars$auc[1])

modelNames <- c("glm","glmn","gam","mars")

ggroc(list(roc.glm, roc.glmn, roc.gam, roc.mars), legacy.axes = TRUE) +
  scale_color_discrete(labels = paste0(modelNames, " (", round(auc,3),")"),
                       name = "Models (AUC)") +
  geom_abline(intercept = 0, slope = 1, color = "grey")
```



Discriminant Analysis

LDA

```
set.seed(2)

model.lda <- train(x = covid_dat2[rowTrain,],
                   y = covid_dat$recovery_time[rowTrain],
                   method = "lda",
                   metric = "ROC",
                   trControl = ctrl2)
```

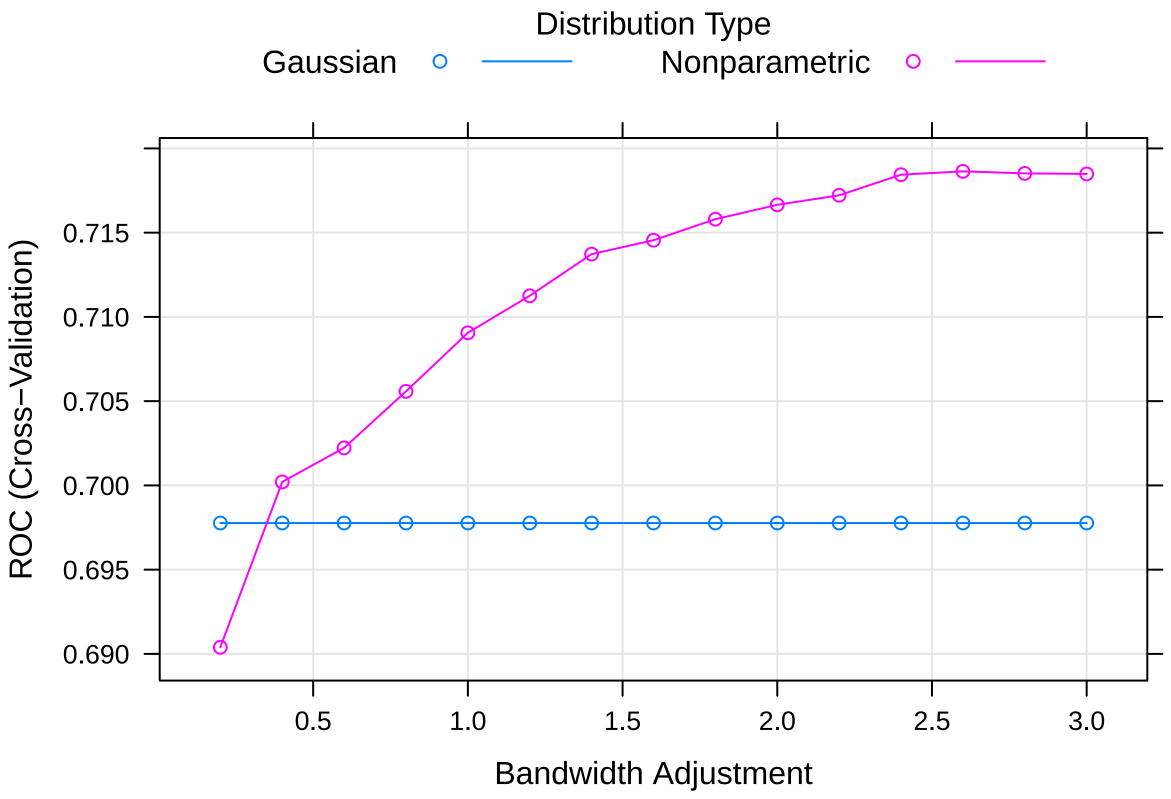
QDA

```
set.seed(2)
model.qda <- train(x = covid_dat2[rowTrain,],
                   y = covid_dat$recovery_time[rowTrain],
                   method = "qda",
                   metric = "ROC",
                   trControl = ctrl2)
```

Naive Bayes (NB)

There is one practical issue with the NB classifier when nonparametric estimators are used. When a new data point includes a feature value that never occurs for some response class, the posterior probability can become zero. To avoid this, we increase the count of the value with a zero occurrence to a small value, so that the overall probability doesn't become zero. In practice, a value of one or two is a common choice. This correction is called "Laplace Correction," and is implemented via the parameter `fL`. The parameter `adjust` adjusts the bandwidths of the kernel density estimates, and a larger value means a more flexible estimate.

```
nbGrid <- expand.grid(usekernel = c(FALSE,TRUE),
                      fL = 1,
                      adjust = seq(.2, 3, by = .2))

set.seed(2)
model.nb <- train(x = covid_dat2[rowTrain,],
                  y = covid_dat$recovery_time[rowTrain],
                  method = "nb",
                  tuneGrid = nbGrid,
                  metric = "ROC",
                  trControl = ctrl2)

plot(model.nb)
```

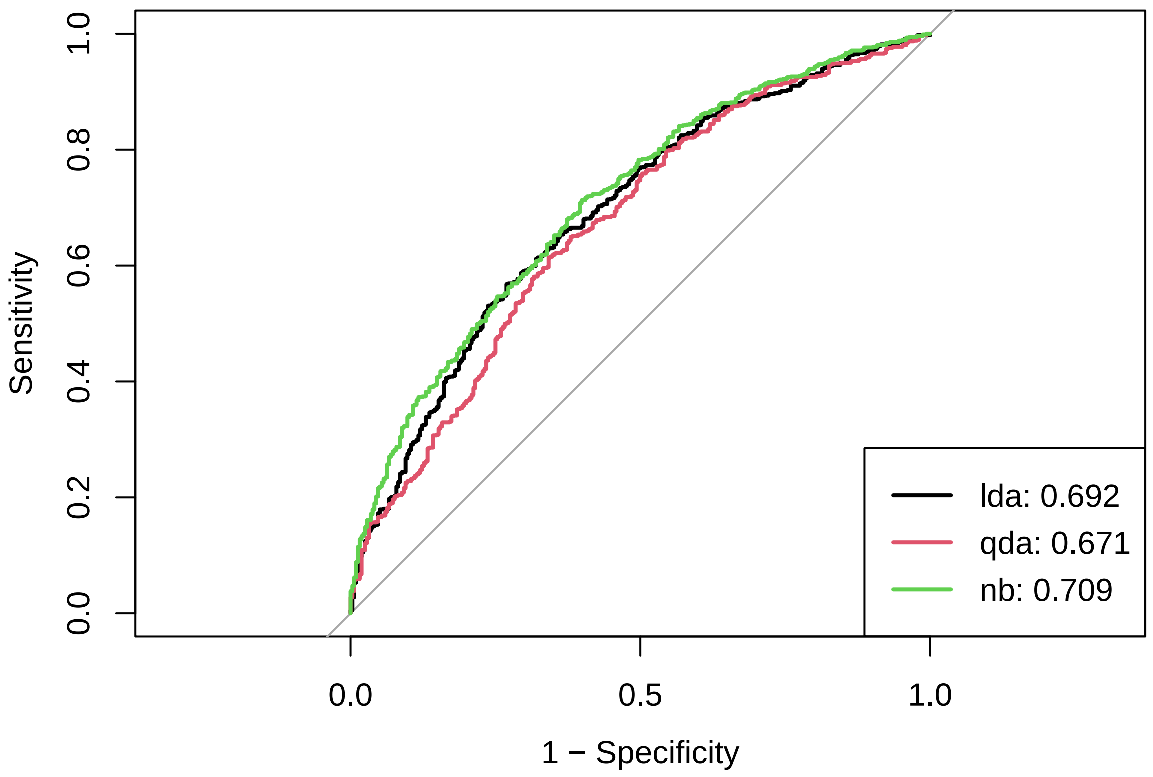


```
res <- resamples(list(LDA = model.lda, QDA = model.qda, NB = model.nb))
summary(res)
```

```
##
## Call:
## summary.resamples(object = res)
##
## Models: LDA, QDA, NB
## Number of resamples: 10
##
## ROC
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LDA 0.6982994 0.7092113 0.7204289 0.7236334 0.7385363 0.7509869    0
## QDA 0.6825470 0.6870515 0.7021897 0.7067165 0.7212097 0.7436988    0
## NB  0.6882730 0.7120171 0.7177622 0.7186405 0.7223240 0.7519739    0
##
## Sens
##          Min.    1st Qu.     Median       Mean    3rd Qu.       Max. NA's
## LDA 0.1891892 0.26013514 0.27702703 0.26525361 0.29489078 0.31081081    0
## QDA 0.5270270 0.55743243 0.60135135 0.59546464 0.63175676 0.67567568    0
## NB  0.0000000 0.01351351 0.01351351 0.01488338 0.02369493 0.02702703    0
##
## Spec
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LDA 0.8587571 0.8997175 0.9239351 0.9170507 0.9324890 0.9606742    0
## QDA 0.6440678 0.7090395 0.7211325 0.7185298 0.7299562 0.7683616    0
## NB  0.9887006 0.9957627 1.0000000 0.9977401 1.0000000 1.0000000    0
```

test set performance for Discriminant Analysis

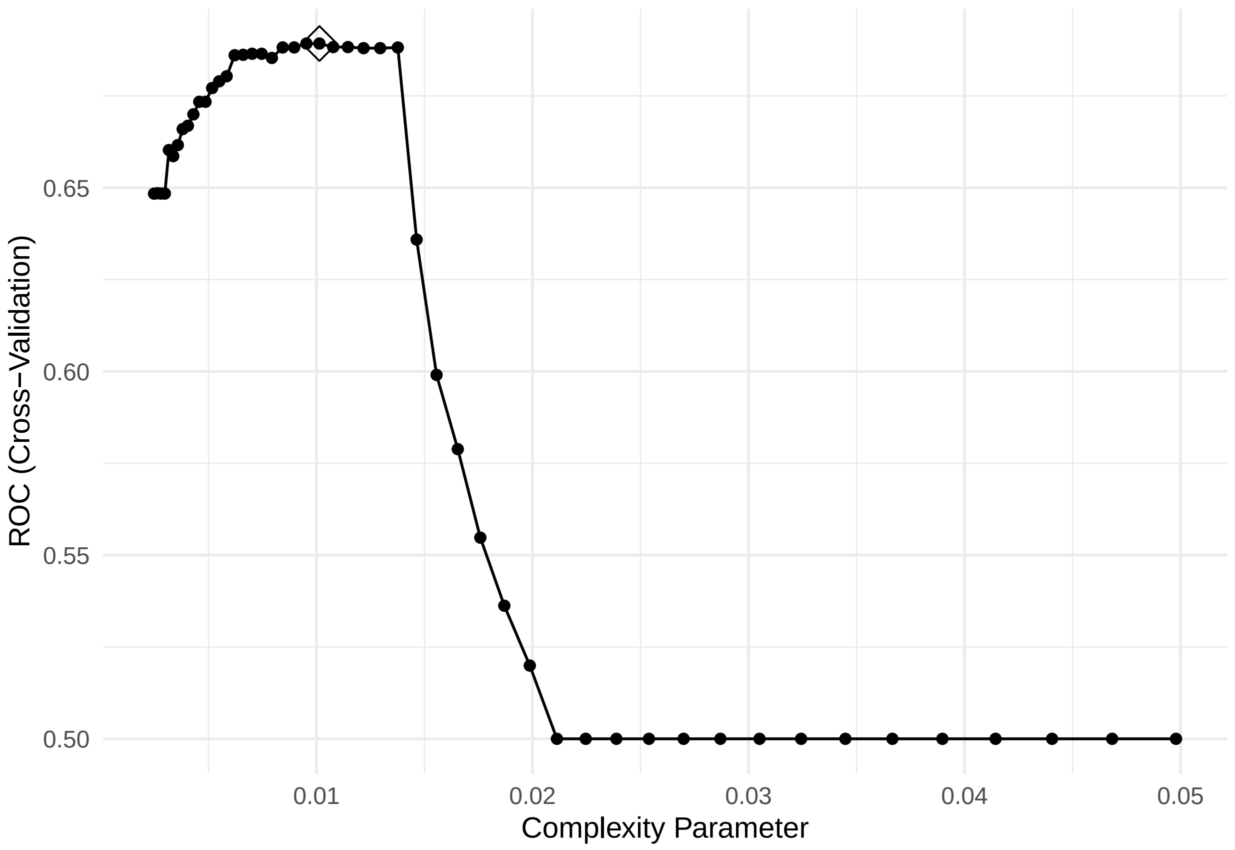
```r
lda.pred <- predict(model.lda, newdata = covid_dat2[-rowTrain,], type = "prob")[,2]
nb.pred <- predict(model.nb, newdata = covid_dat2[-rowTrain,], type = "prob")[,2]
qda.pred <- predict(model.qda, newdata = covid_dat2[-rowTrain,], type = "prob")[,2]


roc.lda <- roc(covid_dat$recovery_time[-rowTrain], lda.pred)
roc.nb <- roc(covid_dat$recovery_time[-rowTrain], nb.pred)
roc.qda <- roc(covid_dat$recovery_time[-rowTrain], qda.pred)


auc <- c(roc.lda$auc[1], roc.qda$auc[1], roc.nb$auc[1])

plot(roc.lda, legacy.axes = TRUE)
plot(roc.qda, col = 2, add = TRUE)
plot(roc.nb, col = 3, add = TRUE)

modelNames <- c("lda","qda","nb")
legend("bottomright", legend = paste0(modelNames, ": ", round(auc,3)),
       col = 1:3, lwd = 2)
```



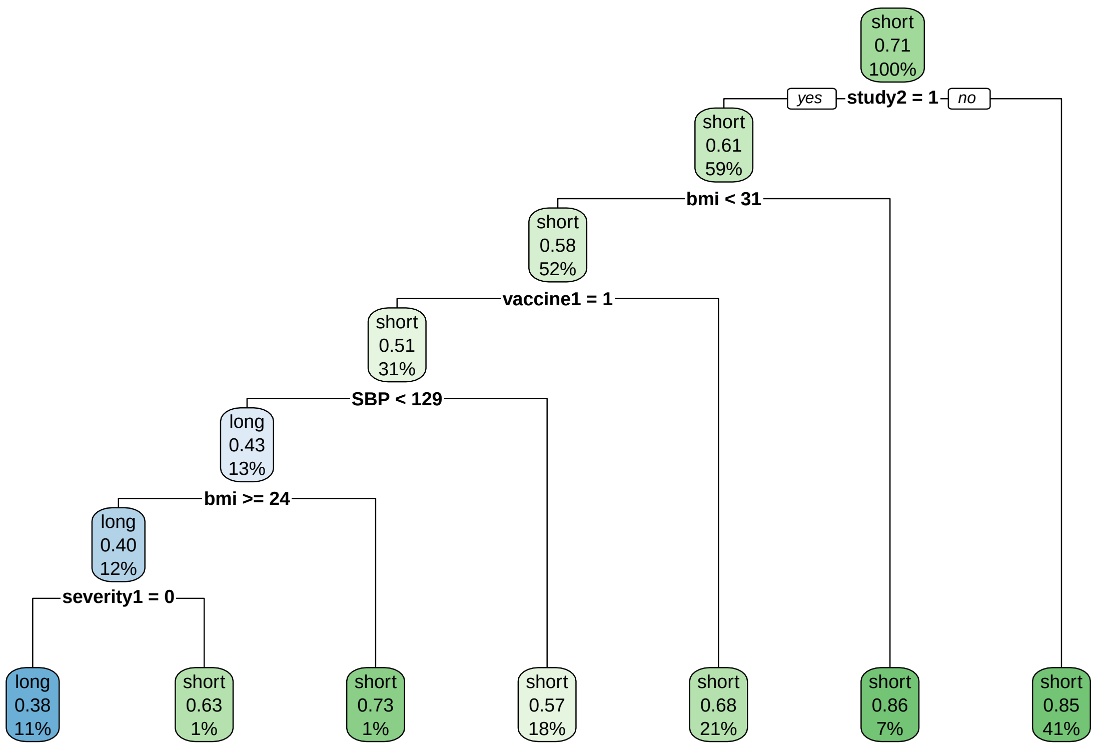
classification tree models

rpart

```
set.seed(2)

model.rpart = train(recovery_time ~ .,
                    covid_dat,
                    subset = rowTrain,
                    method = "rpart",
                    tuneGrid = data.frame(cp = exp(seq(-6, -3, len = 50))),
                    trControl = ctrl2,
                    metric = "ROC")

ggplot(model.rpart, highlight = TRUE)
```

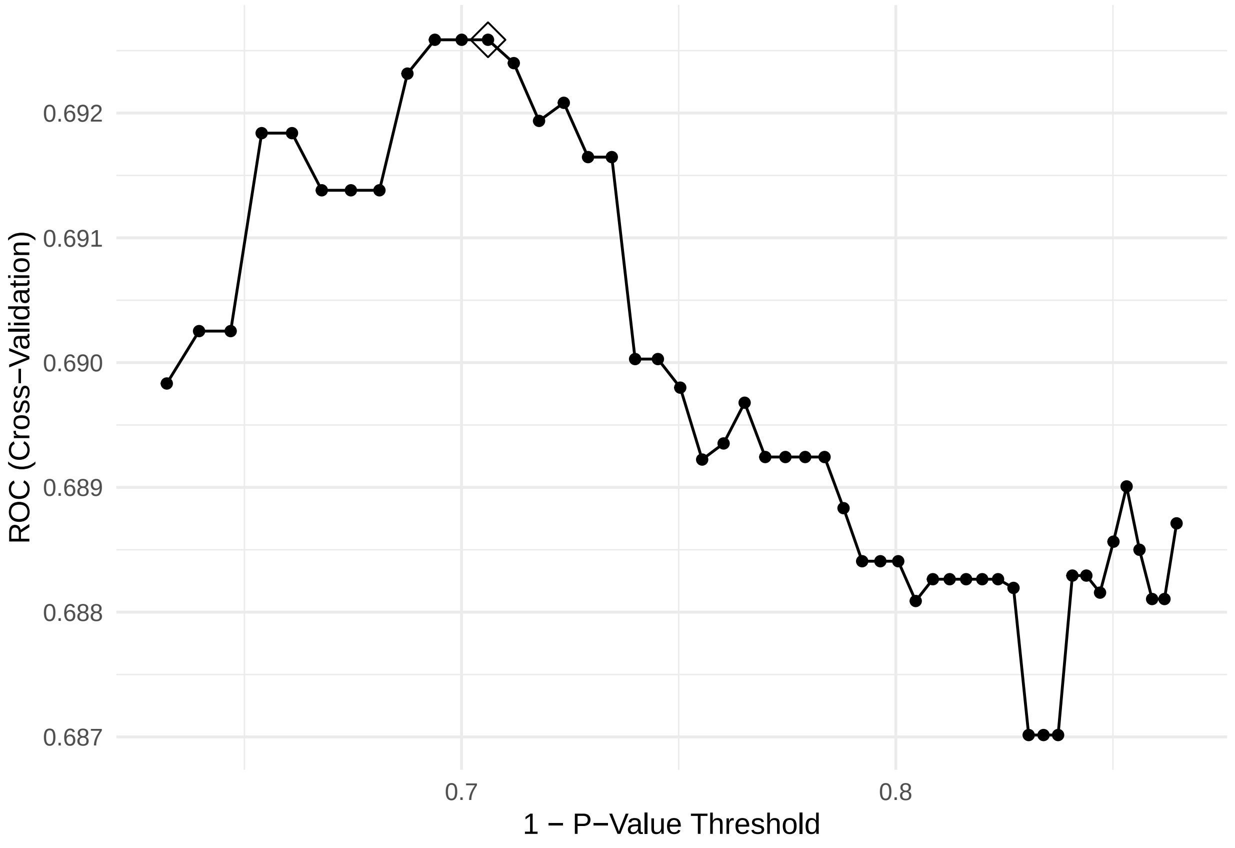


```
rpart.plot(model.rpart$finalModel)
```

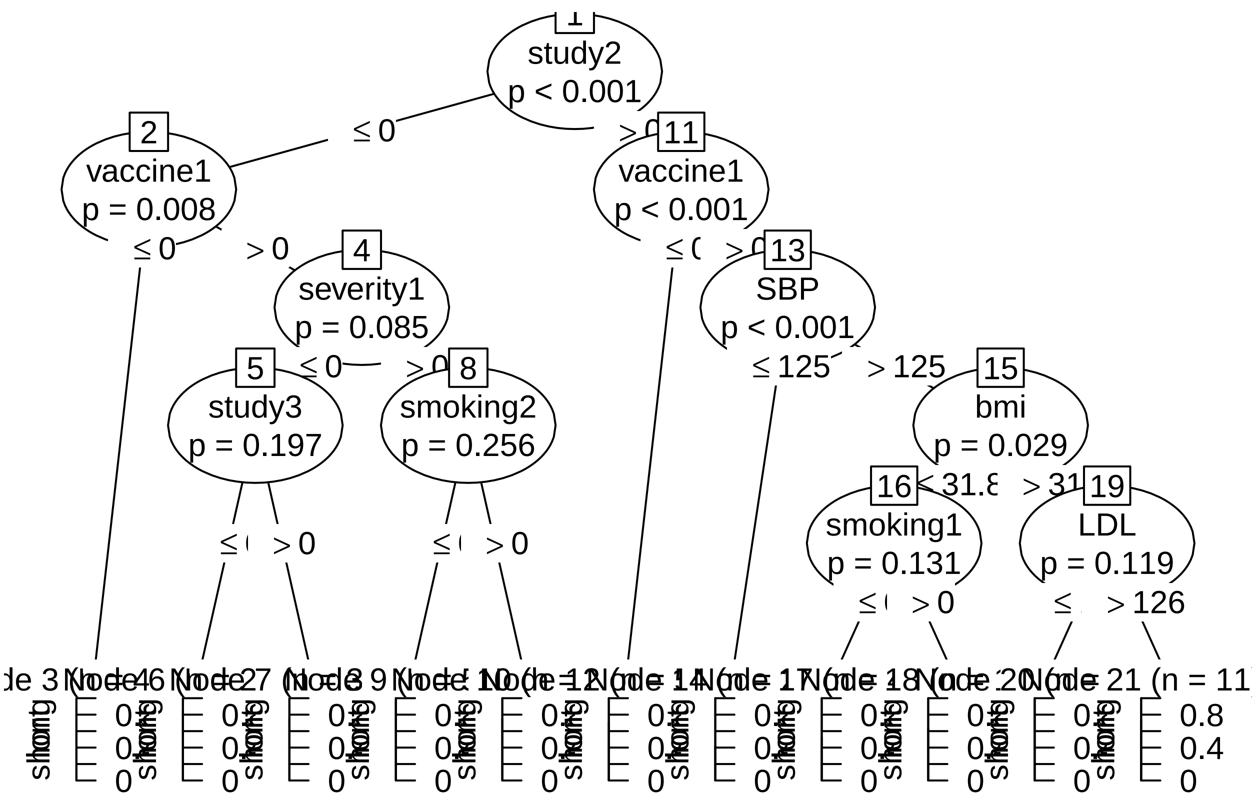


ctree

```
set.seed(2)


model.ctree = train(recovery_time ~ .,
                    covid_dat,
                    subset = rowTrain,
                    method = "ctree",
                    tuneGrid = data.frame(mincriterion = 1 - exp(seq(-2, -1, length = 50))),
                    metric = "ROC",
                    trControl = ctrl2)

ggplot(model.ctree, highlight = TRUE)
```



```
plot(model.ctree$finalModel)
```



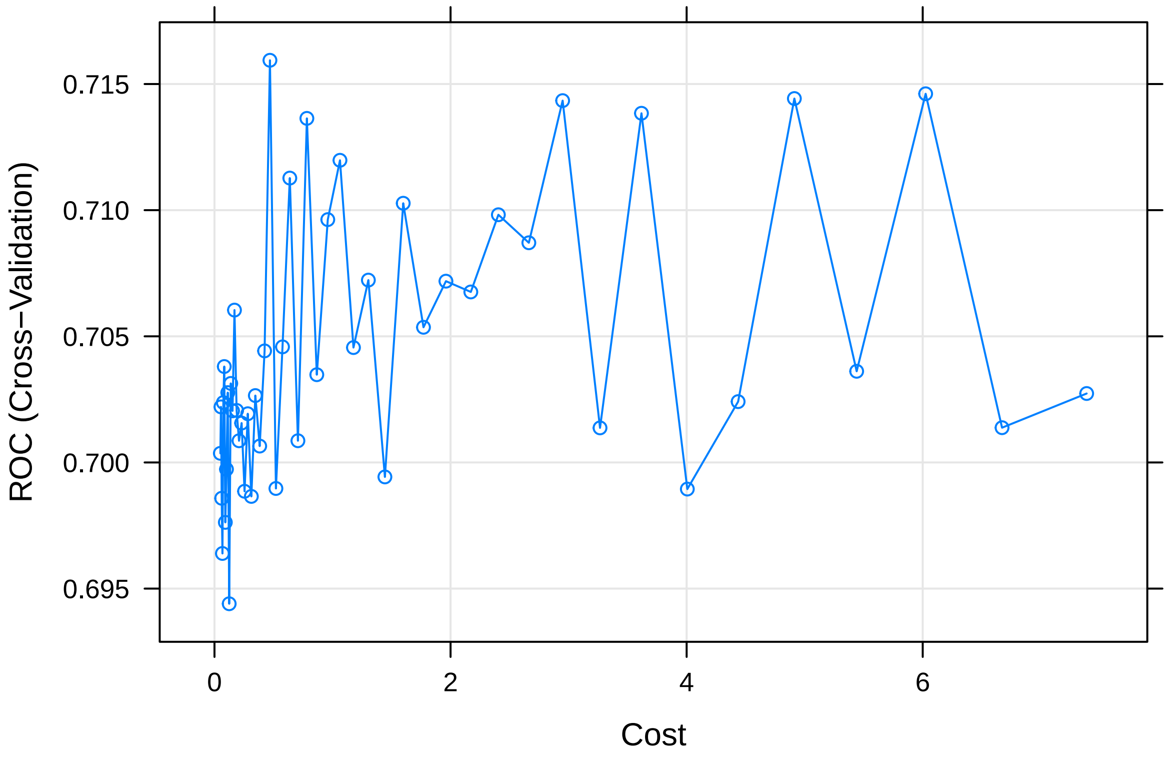
test set performance for classification tree models

```r
resamp_tree <- resamples(list(rpart = model.rpart,
                              ctree = model.ctree))
summary(resamp_tree)
```

```
##
## Call:
## summary.resamples(object = resamp_tree)
##
## Models: rpart, ctree
## Number of resamples: 10
##
## ROC
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rpart 0.6570851 0.6730605 0.6885818 0.6892499 0.6958410 0.7433196    0
## ctree 0.6718201 0.6806484 0.6832281 0.6925873 0.7081043 0.7224385    0
##
## Sens
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rpart 0.2027027 0.2432432 0.2500000 0.2476490 0.2593947 0.2837838    0
## ctree 0.1621622 0.1790541 0.1959459 0.2288227 0.2837838 0.3287671    0
##
## Spec
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rpart 0.8644068 0.8956151 0.9180791 0.9125627 0.9268631 0.9548023    0
## ctree 0.8474576 0.8997175 0.9152542 0.9103314 0.9255618 0.9604520    0
```

Support Vector Machines

```r
set.seed(2)
# kernal linear

#model.svml <- train(recovery_time ~ .,
                    #data = covid_dat[rowTrain, ],
                    #method = "svmLinear",
                    #tuneGrid = data.frame(C = exp(seq(-2,5,len=5))),
                    #trControl = ctrl2)


model.svml <- train(recovery_time ~ .,
                    data = covid_dat[rowTrain, ],
                    method = "svmLinear2",
                    preProcess = c("center", "scale"),
                    tuneGrid = data.frame(cost = exp(seq(-3,2,len = 50))),
                    trControl = ctrl2)

plot(model.svml, highlight = TRUE)
```



```r
model.svml$bestTune
```

```
##         cost
## 23 0.4699627
```

```r
model.svml$finalModel
```

```
##
## Call:
## svm.default(x = as.matrix(x), y = y, kernel = "linear", cost = param$cost,
##     probability = classProbs)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.4699627
##
## Number of Support Vectors:  1706
```

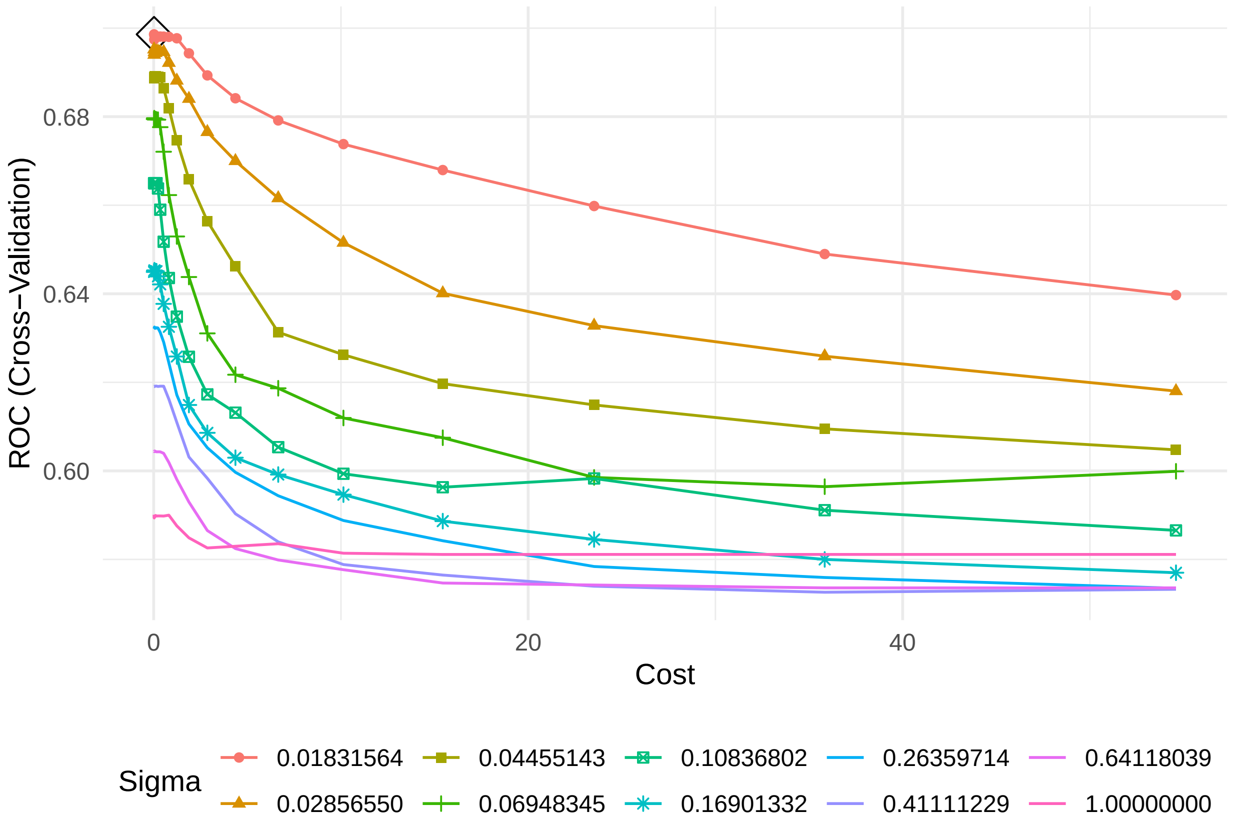
```
#test error
linear_test_preds = predict(model.svml, newdata = covid_dat[-rowTrain, ])
confusionMatrix(data = linear_test_preds,
                reference = covid_dat$recovery_time[-rowTrain])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction long short
##      long     0     0
##      short  316   759
##
##                Accuracy : 0.706
##                  95% CI : (0.6778, 0.7331)
##     No Information Rate : 0.706
##     P-Value [Acc > NIR] : 0.5152
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.000
##             Specificity : 1.000
##          Pos Pred Value :   NaN
##          Neg Pred Value : 0.706
##              Prevalence : 0.294
##          Detection Rate : 0.000
##    Detection Prevalence : 0.000
##       Balanced Accuracy : 0.500
##
##        'Positive' Class : long
##
```

```
svmr.grid <- expand.grid(C = exp(seq(-4,4,len=20)),
                         sigma = exp(seq(-4,0,len=10)))
#radial kernel
set.seed(2)

model.svmr <- train(recovery_time ~ .,
                  data = covid_dat[rowTrain, ],
                  method = "svmRadialSigma",
                  preProcess = c("center", "scale"),
                  tuneGrid = svmr.grid,
                  trControl = ctrl2)
```

```
## line search fails -0.4823254 -0.5646101 1.084874e-05 7.909748e-06 -1.823444e-08 -6.944499e-09 -2.527
```

```
myCol<- rainbow(20)
myPar <- list(superpose.symbol = list(col = myCol),
              superpose.line = list(col = myCol))
ggplot(model.svmr, highlight = TRUE, par.settings = myPar)
```



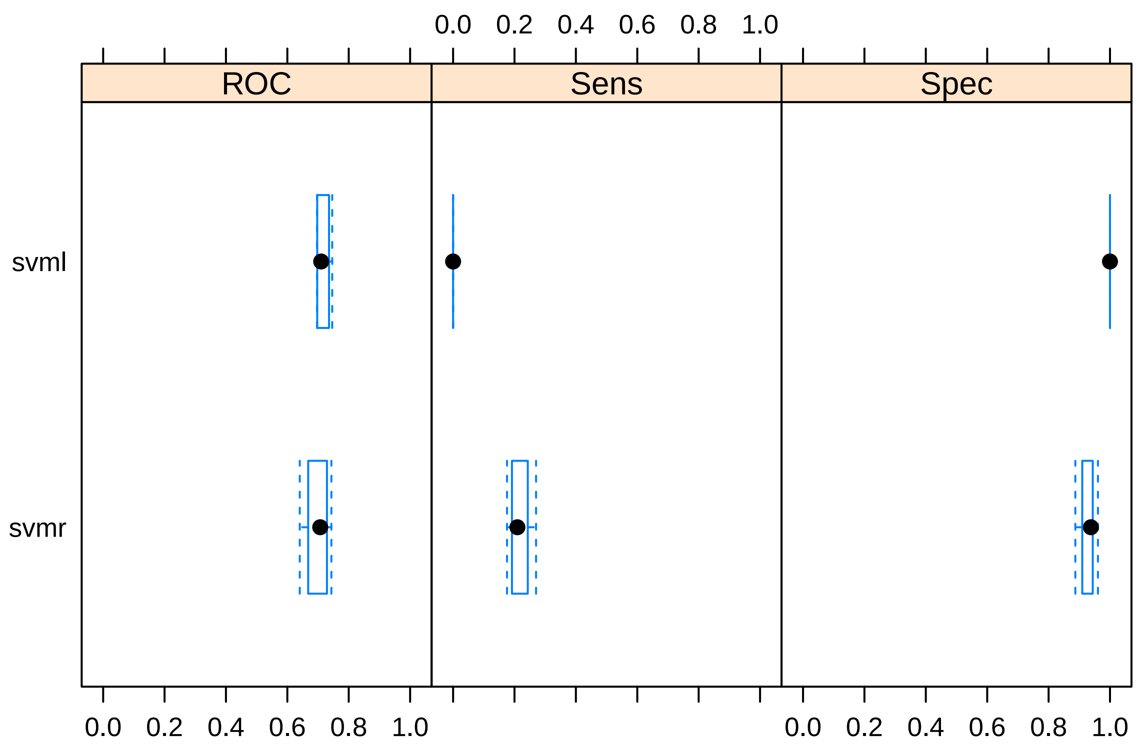
```
# test error
radial_test_preds = predict(model.svmr, newdata = covid_dat[-rowTrain, ])
confusionMatrix(data = radial_test_preds ,
                reference = covid_dat$recovery_time[-rowTrain])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction long short
##      long    77    67
##      short  239   692
##
##                Accuracy : 0.7153
##                  95% CI : (0.6873, 0.7422)
##     No Information Rate : 0.706
##     P-Value [Acc > NIR] : 0.2633
##
##                   Kappa : 0.1847
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.24367
##             Specificity : 0.91173
##          Pos Pred Value : 0.53472
##          Neg Pred Value : 0.74329
##              Prevalence : 0.29395
##          Detection Rate : 0.07163
##    Detection Prevalence : 0.13395
##       Balanced Accuracy : 0.57770
##
##        'Positive' Class : long
##
```

```
#radial kernel
#set.seed(2)

#model.svmr <- tune.svm(recovery_time ~ .,
                  #data = covid_dat[rowTrain, ],
                  #kernel = "radial",
                        #cost = exp(seq(-1,5,len=20)),
                        #gamma = exp(seq(-4,3,len=20)))

#plot(model.svmr, transform.y = log, transform.x = log,
#color.palette = terrain.colors)
# test error
#radial_test_preds = predict(model.svmr, newdata = covid_dat[-rowTrain, ])
#confusionMatrix(data = radial_test_preds ,
                #reference = covid_dat$recovery_time[-rowTrain])
```

test data performance of SVM methods

```
resamp <- resamples(list(svml = model.svml,
                         svmr = model.svmr
                        ))
bwplot(resamp)
```



```
pred.svml <- predict(model.svml, newdata = covid_dat[-rowTrain,])
pred.svmr <- predict(model.svmr, newdata = covid_dat[-rowTrain,])

confusionMatrix(data = pred.svml,
                reference = covid_dat$recovery_time[-rowTrain])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction long short
##      long     0     0
##      short  316   759
##
##                Accuracy : 0.706
##                  95% CI : (0.6778, 0.7331)
##     No Information Rate : 0.706
##     P-Value [Acc > NIR] : 0.5152
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.000
##             Specificity : 1.000
##          Pos Pred Value :   NaN
##          Neg Pred Value : 0.706
##              Prevalence : 0.294
##          Detection Rate : 0.000
##    Detection Prevalence : 0.000
##       Balanced Accuracy : 0.500
##
```

```
##        'Positive' Class : long
##
```

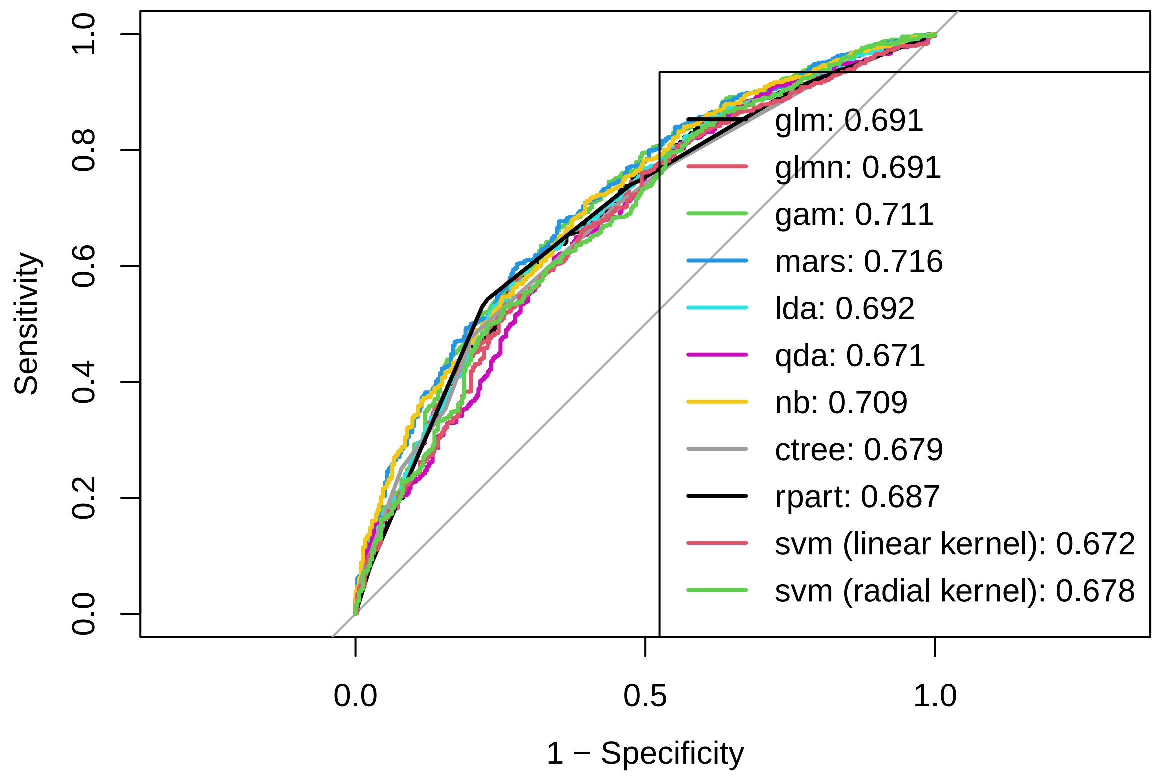
```r
confusionMatrix(data = pred.svmr,
                reference = covid_dat$recovery_time[-rowTrain])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction long short
##      long    77    67
##      short  239   692
##
##                Accuracy : 0.7153
##                  95% CI : (0.6873, 0.7422)
##     No Information Rate : 0.706
##     P-Value [Acc > NIR] : 0.2633
##
##                   Kappa : 0.1847
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.24367
##             Specificity : 0.91173
##          Pos Pred Value : 0.53472
##          Neg Pred Value : 0.74329
##              Prevalence : 0.29395
##          Detection Rate : 0.07163
##    Detection Prevalence : 0.13395
##       Balanced Accuracy : 0.57770
##
##        'Positive' Class : long
##
```

```r
res <- resamples(list(GLM = model.glm, GLMNET = model.glmn, GAM = model.gam, MARS = model.mars, CTREE =
trainROC <- bwplot(res, metric = "ROC")
summary(res)
```

```
##
## Call:
## summary.resamples(object = res)
##
## Models: GLM, GLMNET, GAM, MARS, CTREE, RPART, LDA, QDA, NB, SVML, SVMR
## Number of resamples: 10
##
## ROC
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM    0.7027786 0.7100512 0.7240580 0.7243182 0.7396824 0.7482539    0
## GLMNET 0.7010325 0.7091541 0.7199426 0.7230782 0.7374113 0.7488612    0
## GAM    0.7092686 0.7156054 0.7316766 0.7319728 0.7432725 0.7592621    0
## MARS   0.7131036 0.7220759 0.7309513 0.7309203 0.7385390 0.7542894    0
## CTREE  0.6718201 0.6806484 0.6832281 0.6925873 0.7081043 0.7224385    0
## RPART  0.6570851 0.6730605 0.6885818 0.6892499 0.6958410 0.7433196    0
## LDA    0.6982994 0.7092113 0.7204289 0.7236334 0.7385363 0.7509869    0
## QDA    0.6825470 0.6870515 0.7021897 0.7067165 0.7212097 0.7436988    0
## NB     0.6882730 0.7120171 0.7177622 0.7186405 0.7223240 0.7519739    0
## SVML   0.6968239 0.6982173 0.7103583 0.7159424 0.7324267 0.7462041    0
## SVMR   0.6404795 0.6755612 0.7070927 0.6985938 0.7242925 0.7435469    0
##
## Sens
##             Min.    1st Qu.     Median       Mean    3rd Qu.       Max. NA's
## GLM    0.2162162 0.27027027 0.27897075 0.28146983 0.30743243 0.33783784    0
## GLMNET 0.2027027 0.24662162 0.28378378 0.26932618 0.29697334 0.32432432    0
## GAM    0.2027027 0.26277305 0.31756757 0.30170307 0.33783784 0.36486486    0
## MARS   0.2297297 0.28378378 0.29248427 0.30444280 0.33445946 0.37837838    0
## CTREE  0.1621622 0.17905405 0.19594595 0.22882266 0.28378378 0.32876712    0
## RPART  0.2027027 0.24324324 0.25000000 0.24764902 0.25939467 0.28378378    0
## LDA    0.1891892 0.26013514 0.27702703 0.26525361 0.29489078 0.31081081    0
## QDA    0.5270270 0.55743243 0.60135135 0.59546464 0.63175676 0.67567568    0
## NB     0.0000000 0.01351351 0.01351351 0.01488338 0.02369493 0.02702703    0
## SVML   0.0000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000    0
## SVMR   0.1756757 0.19451129 0.20945946 0.21917808 0.24324324 0.27027027    0
##
## Spec
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM    0.8531073 0.9039548 0.9154605 0.9119755 0.9324890 0.9438202    0
## GLMNET 0.8644068 0.9053672 0.9180791 0.9187425 0.9382022 0.9548023    0
## GAM    0.8644068 0.8884181 0.9124294 0.9018314 0.9157303 0.9269663    0
## MARS   0.8644068 0.8912429 0.9098584 0.9057862 0.9196106 0.9325843    0
## CTREE  0.8474576 0.8997175 0.9152542 0.9103314 0.9255618 0.9604520    0
## RPART  0.8644068 0.8956151 0.9180791 0.9125627 0.9268631 0.9548023    0
## LDA    0.8587571 0.8997175 0.9239351 0.9170507 0.9324890 0.9606742    0
## QDA    0.6440678 0.7090395 0.7211325 0.7185298 0.7299562 0.7683616    0
## NB     0.9887006 0.9957627 1.0000000 0.9977401 1.0000000 1.0000000    0
## SVML   1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000    0
## SVMR   0.8870056 0.9124294 0.9378531 0.9288929 0.9423284 0.9606742    0
```

```r
#pred
glm.pred <- predict(model.glm, newdata = covid_dat2[-rowTrain,], type = "prob")[,2]
glmn.pred <- predict(model.glmn, newdata = covid_dat2[-rowTrain,], type = "prob")[,2]
gam.pred <- predict(model.gam, newdata = covid_dat2[-rowTrain,], type = "prob")[,2]
mars.pred <- predict(model.mars, newdata = covid_dat2[-rowTrain,], type = "prob")[,2]

ctree.pred <- predict(model.ctree, newdata = covid_dat[-rowTrain,], type = "prob")[,2]
rpart.pred <- predict(model.rpart, newdata = covid_dat[-rowTrain,], type = "prob")[,2]

lda.pred <- predict(model.lda, newdata =covid_dat2[-rowTrain,], type = "prob")[,2]
qda.pred <- predict(model.qda, newdata = covid_dat2[-rowTrain,], type = "prob")[,2]
nb.pred <- predict(model.nb, newdata = covid_dat2[-rowTrain,], type = "prob")[,2]

svml.pred <- predict(model.svml, newdata = covid_dat[-rowTrain,], type = "prob")[,2]
svmr.pred <- predict(model.svmr, newdata = covid_dat[-rowTrain,], type = "prob")[,2]

#roc
roc.glm <- roc(covid_dat$recovery_time[-rowTrain], glm.pred)
roc.glmn <- roc(covid_dat$recovery_time[-rowTrain], glmn.pred)
roc.gam <- roc(covid_dat$recovery_time[-rowTrain], gam.pred)
roc.mars <- roc(covid_dat$recovery_time[-rowTrain], mars.pred)
roc.ctree <- roc(covid_dat$recovery_time[-rowTrain], ctree.pred)
roc.rpart <- roc(covid_dat$recovery_time[-rowTrain], rpart.pred)

roc.lda <- roc(covid_dat$recovery_time[-rowTrain], lda.pred)
roc.qda <- roc(covid_dat$recovery_time[-rowTrain], qda.pred)
roc.nb <- roc(covid_dat$recovery_time[-rowTrain], nb.pred)

roc.svml <- roc(covid_dat$recovery_time[-rowTrain], svml.pred)
roc.svmr <- roc(covid_dat$recovery_time[-rowTrain], svmr.pred)


auc <- c(roc.glm$auc[1], roc.glmn$auc[1],
         roc.gam$auc[1], roc.mars$auc[1],
         roc.lda$auc[1],roc.qda$auc[1], roc.nb$auc[1],
         roc.ctree$auc[1], roc.rpart$auc[1],
         roc.svml$auc[1], roc.svmr$auc[1]
         )

plot(roc.glm, legacy.axes = TRUE)
plot(roc.glmn, col = 2, add = TRUE)

plot(roc.gam, col = 3, add = TRUE)
plot(roc.mars, col = 4, add = TRUE)

plot(roc.lda, col = 5, add = TRUE)
plot(roc.qda, col = 6, add = TRUE)
plot(roc.nb, col = 7, add = TRUE)

plot(roc.ctree, col = 8, add = TRUE)
plot(roc.rpart, col = 9, add = TRUE)

plot(roc.svml, col = 10, add = TRUE)
plot(roc.svmr, col = 11, add = TRUE)


modelNames <- c("glm","glmn","gam","mars","lda","qda","nb", "ctree",  "rpart","svm (linear kernel)", "s
legend("bottomright", legend = paste0(modelNames, ": ", round(auc,3)),
       col = 1:11, lwd = 2)
```



Sensitivity
1 − Specificity
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.5
1.0
glm: 0.691
glmn: 0.691
gam: 0.711
mars: 0.716
lda: 0.692
qda: 0.671
nb: 0.709
ctree: 0.679
rpart: 0.687
svm (linear kernel): 0.672
svm (radial kernel): 0.678