Binary classification with the caret R package

Evan Muzzall September 14, 2016

This R code was compiled from:

- Package 'caret
- The caret help page
- Kuhn M. 2015. A Short Introduction to the caret Package.
- Kuhn M. 2013. Predictive modeling with R and the caret package. useR! The R User Conference, July 10-12, University of Castilla-La Mancha, Albacete, Spain
- Kuhn M. 2008. Building predictive models in R using the caret package. J Stat Softw 28:1-26.

install packages

```
install.packages("car")
install.packages("caret")
install.packages("e1071")
install.packages("gamlss")
install.packages("gbm")
install.packages("kernlab")
install.packages("plyr")
install.packages("pROC")
library(caret)
library(e1071)
library(gamlss)
library(gbm)
library(kernlab)
library(plyr)
library(pROC)
```

load the Mroz dataset

```
library(car)
data(Mroz)
str(Mroz)

## 'data.frame':    753 obs. of  8 variables:
##  $ lfp : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ k5  : int  1 0 1 0 1 0 0 0 0 0 ...
##  $ k618: int  0 2 3 3 2 0 2 0 2 2 ...
##  $ age : int  32 30 35 34 31 54 37 54 48 39 ...
##  $ wc  : Factor w/ 2 levels "no","yes": 1 1 1 1 2 1 2 1 1 1 ...
##  $ hc  : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ lwg : num  1.2102 0.3285 1.5141 0.0921 1.5243 ...
##  $ inc : num  10.9 19.5 12 6.8 20.1 ...
```

See variable definitions with ?Mroz

use createDataPartition() to create a 75/25 stratified random split

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
split <- createDataPartition(Mroz$lfp, p=0.75, list=FALSE)
training.set <- Mroz[split,]
test.set <- Mroz[-split,]
```

sanity check

```
nrow(training.set) + nrow(test.set) == nrow(Mroz)
```

```
## [1] TRUE
```

train() a GBM model

```
set.seed(1)
gbm.fit1 <- train(lfp ~ ., data=training.set, method="gbm")
```

```
## Loading required package: gbm
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

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

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.1
```

```
## Loading required package: plyr
```

'.' comes from Perl's regex library and stands for "everything else"

caret shows us the optimal model based on its attributes

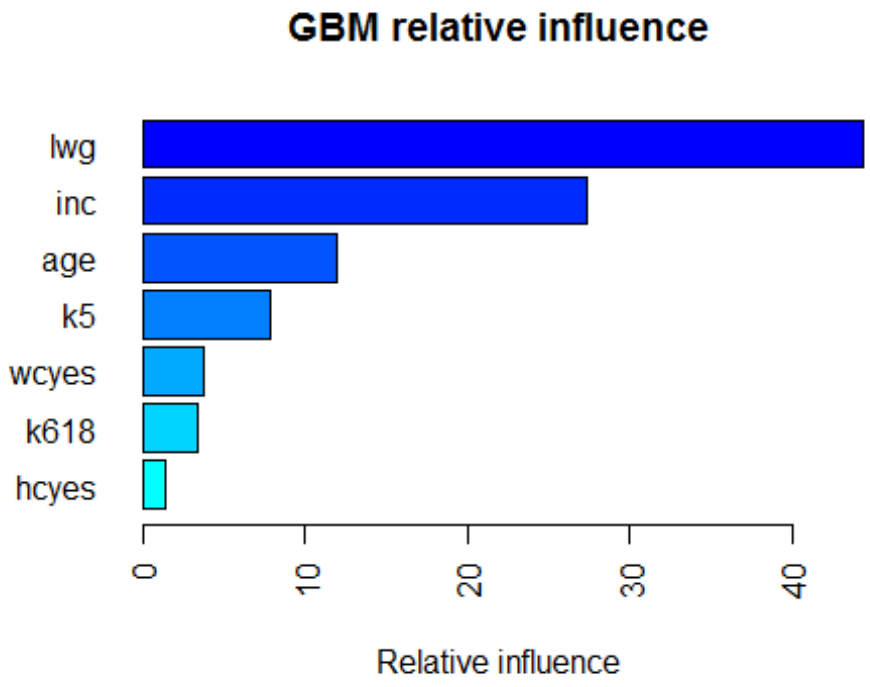
View a model summary table by calling the object

```
gbm.fit1
```

```
## Stochastic Gradient Boosting
##
## 565 samples
##   7 predictor
##   2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 565, 565, 565, 565, 565, 565, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.6971243  0.3814156
##   1                  100      0.7049356  0.3977827
##   1                  150      0.7033739  0.3952230
##   2                   50      0.7083650  0.4048771
##   2                  100      0.7084643  0.4062031
##   2                  150      0.7090828  0.4064878
##   3                   50      0.7026401  0.3929582
##   3                  100      0.7061140  0.4011811
##   3                  150      0.7092976  0.4065118
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

Plot bargraph of variable relative influence with summary()

```
summary(gbm.fit1, las=2, main="GBM relative influence")
```




```
##         var   rel.inf
## lwg     lwg 44.372260
## inc     inc 27.323192
## age     age 11.960572
## k5       k5  7.792741
## wcyes wcyes  3.751020
## k618   k618  3.400282
## hcyes hcyes  1.399934
```

trainControl()

define the parameters of the control mechanism

```
control1 <- trainControl(method="repeatedcv", repeats=5)
```

train the model via trControl()

```
set.seed(1)
gbm.fit2 <- train(lfp ~ ., data=training.set,
    method="gbm",
    verbose=FALSE,
    trControl=control1)
```

model summary table

```
gbm.fit2
```

bargraph of variable relative influence

```
summary(gbm.fit2, las=2)
```

model tuning within trainControl()

```
control2 <- trainControl(method="repeatedcv",
    repeats=5,
    classProbs=TRUE,
    summaryFunction=twoClassSummary)

set.seed(1)
gbm.fit3 <- train(lfp ~ ., data=training.set,
    method="gbm",
    metric="ROC",
    verbose=FALSE,
    trControl=control2)
```

model summary table

```
gbm.fit3
```

bargraph of variable relative influence

```
summary(gbm.fit3, las=2)
```

compare multiple models at once with expand.grid()

```
grid <- expand.grid(n.trees=seq(100,5100, by=500),
    interaction.depth=seq(1,3,5),
    shrinkage=c(0.01,0.05, 0.1),
    n.minobsinnode=10)

set.seed(1)
gbm.fit4 <- train(lfp ~ ., data=training.set,
    method="gbm",
    metric="ROC",
    tuneGrid=grid,
    verbose=FALSE,
    trControl=control2)
```

model summary table

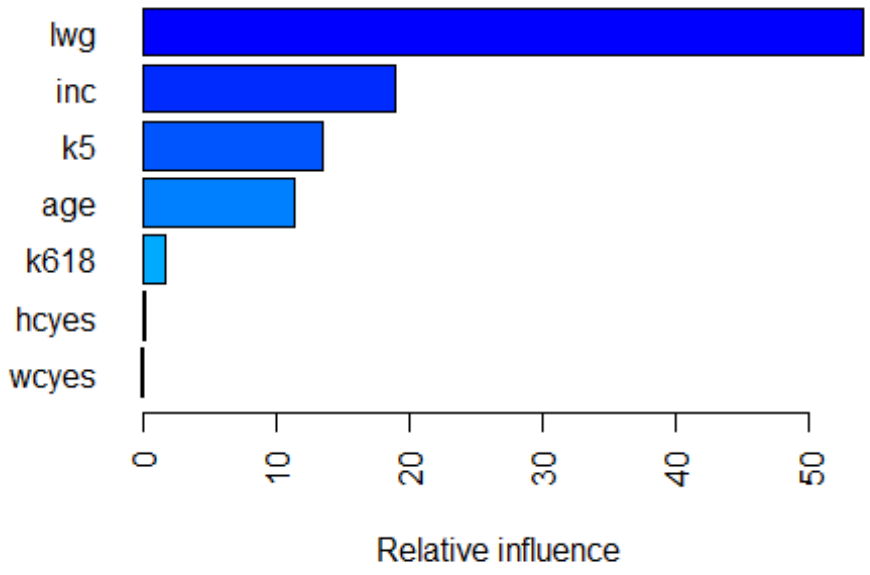
```
gbm.fit4
```

```
## Stochastic Gradient Boosting
##
## 565 samples
##   7 predictor
##   2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 509, 509, 508, 508, 507, 509, ...
## Resampling results across tuning parameters:
##
##   shrinkage  n.trees  ROC        Sens       Spec
##   0.01        100     0.7514858  0.3178333  0.8796970
```

```
##   0.01        600     0.7902775  0.6632333  0.7558144
##   0.01       1100     0.7934049  0.6853667  0.7476515
##   0.01       1600     0.7902823  0.6804000  0.7464205
##   0.01       2100     0.7892753  0.6813333  0.7432765
##   0.01       2600     0.7881934  0.6871000  0.7470265
##   0.01       3100     0.7867107  0.6871000  0.7538636
##   0.01       3600     0.7863194  0.6805333  0.7532576
##   0.01       4100     0.7857652  0.6821333  0.7495076
##   0.01       4600     0.7850192  0.6805333  0.7519886
##   0.01       5100     0.7846124  0.6780000  0.7513636
##   0.05        100     0.7863477  0.6607333  0.7588068
##   0.05        600     0.7857294  0.6788000  0.7501136
##   0.05       1100     0.7850699  0.6756000  0.7538258
##   0.05       1600     0.7839003  0.6632333  0.7544508
##   0.05       2100     0.7805825  0.6674333  0.7519508
##   0.05       2600     0.7824095  0.6730333  0.7544318
##   0.05       3100     0.7811982  0.6688667  0.7507197
##   0.05       3600     0.7812330  0.6671333  0.7457576
##   0.05       4100     0.7819305  0.6597667  0.7482576
##   0.05       4600     0.7808585  0.6589333  0.7488826
##   0.05       5100     0.7796546  0.6604667  0.7457765
##   0.10        100     0.7908012  0.6771667  0.7470265
##   0.10        600     0.7823807  0.6605000  0.7425947
##   0.10       1100     0.7811461  0.6639667  0.7513636
##   0.10       1600     0.7822016  0.6679667  0.7544886
##   0.10       2100     0.7808256  0.6663000  0.7563447
##   0.10       2600     0.7802844  0.6613667  0.7550947
##   0.10       3100     0.7787305  0.6621000  0.7495265
##   0.10       3600     0.7754369  0.6637000  0.7451326
##   0.10       4100     0.7755464  0.6613000  0.7519697
##   0.10       4600     0.7752049  0.6753000  0.7538258
##   0.10       5100     0.7739857  0.6727333  0.7469318
##
## Tuning parameter 'interaction.depth' was held constant at a value of
##  1
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## ROC was used to select the optimal model using  the largest value.
## The final values used for the model were n.trees = 1100,
##  interaction.depth = 1, shrinkage = 0.01 and n.minobsinnode = 10.
```

bargraph of variable relative influence

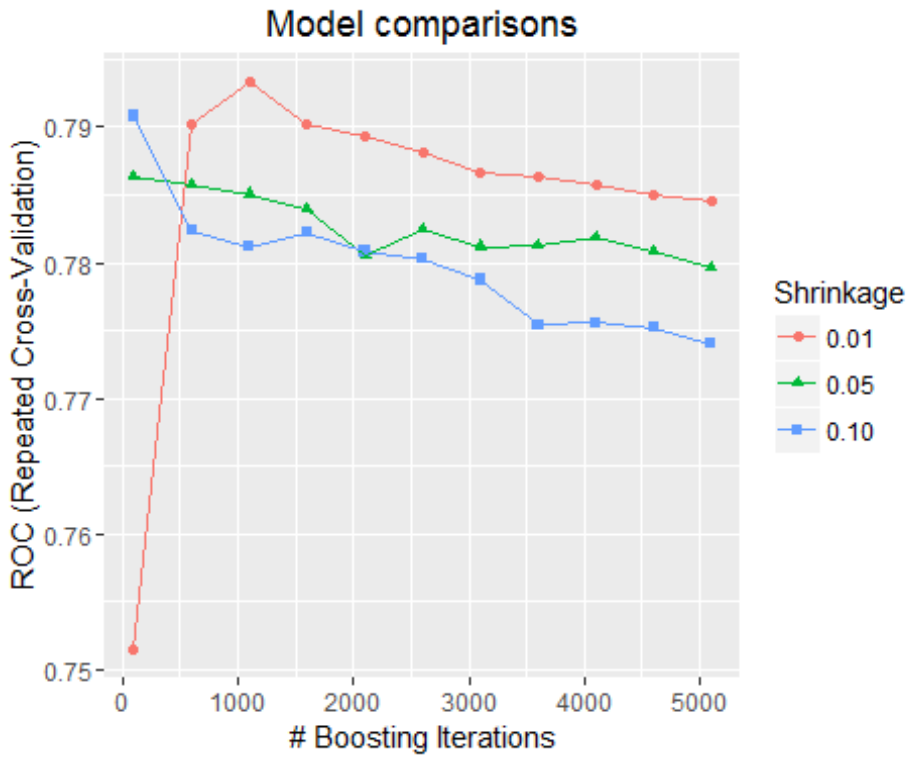
```
summary(gbm.fit4, las=2)
```




```
##          var    rel.inf
## lwg      lwg 54.1133058
## inc      inc 18.9101705
## k5        k5 13.5024530
## age      age 11.3526652
## k618   k618  1.7660524
## hcyes hcyes  0.2514238
## wcyes wcyes  0.1039293
```

ggplot line graph

```
ggplot(gbm.fit4) + theme_grey() + ggtitle("Model comparisons")
```



save as .PNG

```
png(width=9, heigh=6, unit="in", res=620)
ggplot(gbm.fit4) + theme_grey() + ggtitle("Model comparisons")
dev.off()
```

generate GBM predicted values and probabilities with with predict()

predicted values

```
set.seed(1)
gbm.pred <- predict(gbm.fit4, test.set)
gbm.prob <- predict(gbm.fit4, test.set, type="prob")
```

view GBM final model

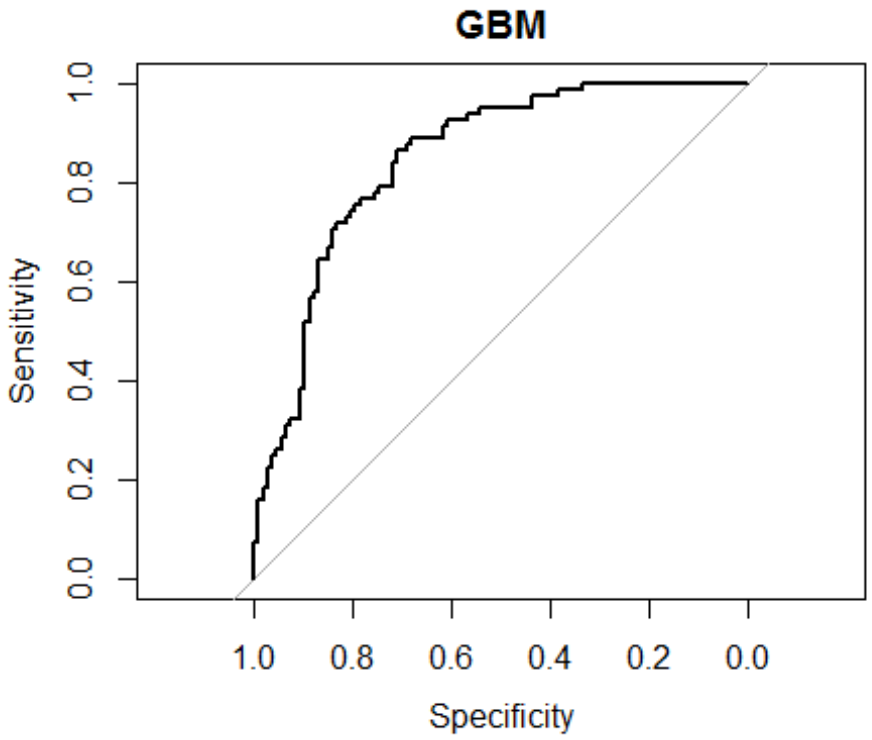
```
gbm.cm <- confusionMatrix(gbm.pred, test.set$lfp)
gbm.cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction no yes
##        no  52  14
##        yes 29  93
##
##                Accuracy : 0.7713
##                  95% CI : (0.7045, 0.8293)
```

```
##     No Information Rate : 0.5691
##     P-Value [Acc > NIR] : 5.455e-09
##
##                   Kappa : 0.5229
##  Mcnemar's Test P-Value : 0.03276
##
##             Sensitivity : 0.6420
##             Specificity : 0.8692
##          Pos Pred Value : 0.7879
##          Neg Pred Value : 0.7623
##              Prevalence : 0.4309
##          Detection Rate : 0.2766
##    Detection Prevalence : 0.3511
##       Balanced Accuracy : 0.7556
##
##        'Positive' Class : no
##
```

plot GBM ROC curve

```
library(pROC)

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var

rocCurve <- roc(response=test.set$lfp,
    predictor = gbm.prob[, "yes"],
    levels = rev(levels(test.set$lfp)),
    auc=TRUE, ci=TRUE)

plot(rocCurve, main="GBM")
```




```
##
## Call:
## roc.default(response = test.set$lfp, predictor = gbm.prob[, "yes"],     level
s = rev(levels(test.set$lfp)), auc = TRUE, ci = TRUE)
##
## Data: gbm.prob[, "yes"] in 107 controls (test.set$lfp yes) > 81 cases (test.s
et$lfp no).
## Area under the curve: 0.8461
## 95% CI: 0.7912-0.901 (DeLong)
```

fit a second model SVM

```
grid2 <- expand.grid(sigma=0.05, C=c(0.01, 0.05, 0.10))
set.seed(1)
svm.fit1 <- train(lfp ~ ., data=training.set,
    method="svmRadial",
    trControl=control2,
    tuneGrid=grid2,
    metric="ROC")
```

```
## Loading required package: kernlab
```

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

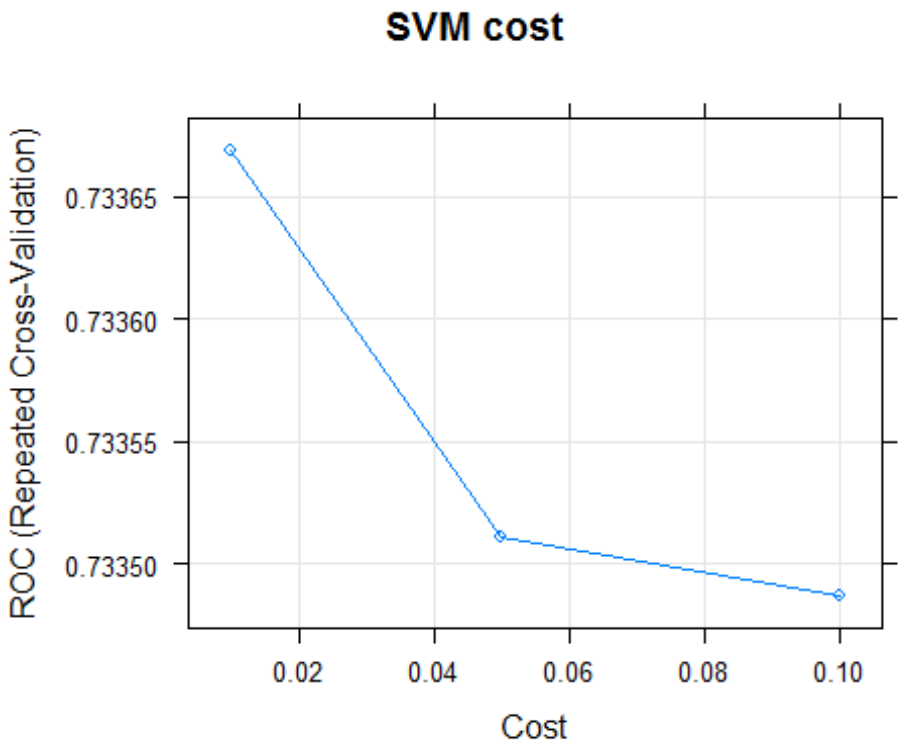
```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

model summary

```
svm.fit1

## Support Vector Machines with Radial Basis Function Kernel
##
## 565 samples
##   7 predictor
##   2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 509, 509, 508, 508, 507, 509, ...
## Resampling results across tuning parameters:
##
##   C     ROC        Sens       Spec
##   0.01  0.7336691  0.6118000  0.7021402
##   0.05  0.7335108  0.6125667  0.7071402
##   0.10  0.7334868  0.6094000  0.7039773
##
## Tuning parameter 'sigma' was held constant at a value of 0.05
## ROC was used to select the optimal model using  the largest value.
## The final values used for the model were sigma = 0.05 and C = 0.01.
```

plot SVM cost

```
plot(svm.fit1, las=2, main="SVM cost")
```

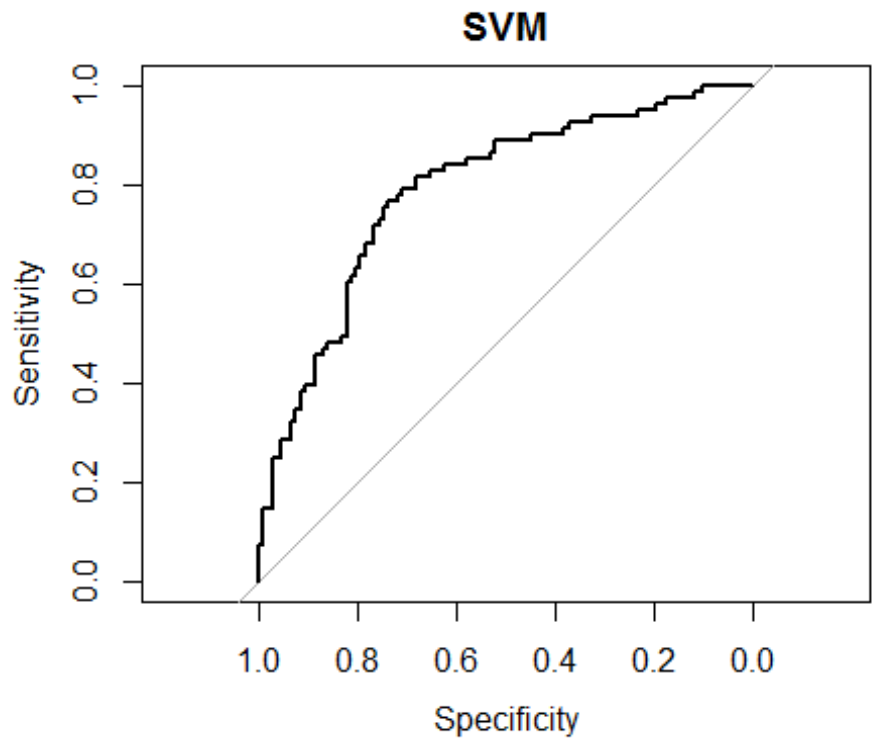


generate SVM predicted values and probabilities

```
set.seed(1)
svm.pred <- predict(svm.fit1, test.set)
svm.prob <- predict(svm.fit1, test.set, type="prob")
confusionMatrix(svm.fit1)

## Cross-Validated (10 fold, repeated 5 times) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction   no  yes
##        no  26.4 16.9
##        yes 16.7 39.9
##
##  Accuracy (average) : 0.6634
```

Plot SVM ROC curve

```
rocCurve2 <- roc(response=test.set$lfp,
    predictor=svm.prob[, "yes"],
    levels=rev(levels(test.set$lfp)),
    auc=TRUE, ci=TRUE)
plot(rocCurve2, main="SVM")
```




```
##
## Call:
## roc.default(response = test.set$lfp, predictor = svm.prob[, "yes"],     level
```

```
s = rev(levels(test.set$lfp)), auc = TRUE, ci = TRUE)
##
## Data: svm.prob[, "yes"] in 107 controls (test.set$lfp yes) > 81 cases (test.s
et$lfp no).
## Area under the curve: 0.7895
## 95% CI: 0.7239-0.8552 (DeLong)
```

resample the GBM and SVM models

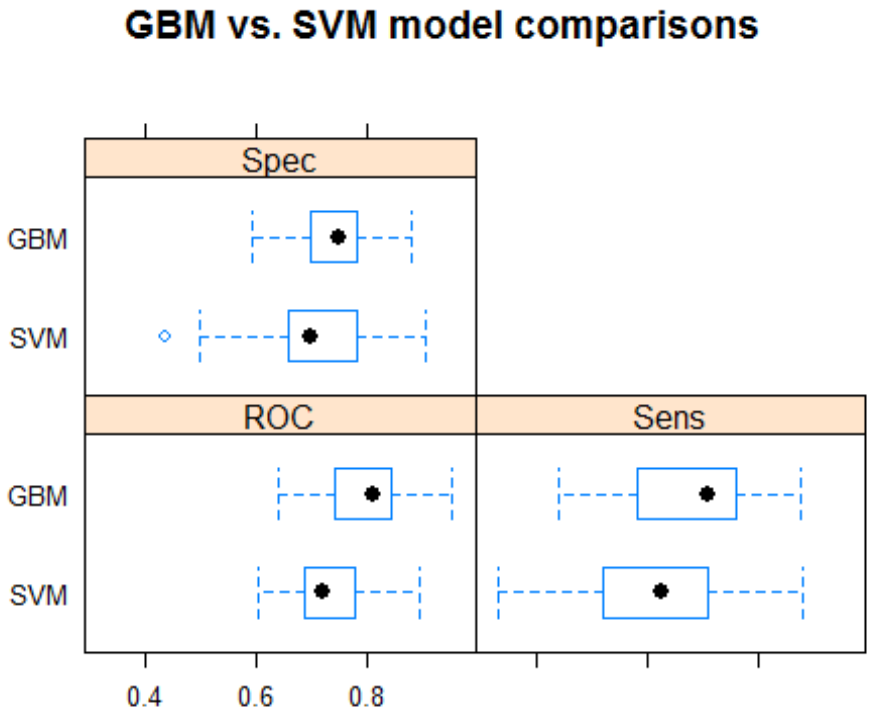
```
set.seed(1)
rsmpl <- resamples(list(GBM=gbm.fit4, SVM=svm.fit1))
rsmpl

##
## Call:
## resamples.default(x = list(GBM = gbm.fit4, SVM = svm.fit1))
##
## Models: GBM, SVM
## Number of resamples: 50
## Performance metrics: ROC, Sens, Spec
## Time estimates for: everything, final model fit
```

five-number summary between the SVM and GBM models

```
summary(rsmpl, resamples=final)

##
## Call:
## summary.resamples(object = rsmpl, resamples = final)
##
## Models: GBM, SVM
## Number of resamples: 50
##
## ROC
##       Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
## GBM 0.6400  0.7448 0.8105 0.7934  0.8422 0.9505    0
## SVM 0.6042  0.6878 0.7181 0.7337  0.7778 0.8932    0
##
## Sens
##       Min. 1st Qu. Median   Mean 3rd Qu.  Max. NA's
## GBM 0.4400  0.5875 0.7083 0.6854  0.7575 0.875    0
## SVM 0.3333  0.5200 0.6250 0.6118  0.7083 0.880    0
##
## Spec
##       Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
## GBM 0.5938  0.6970  0.750 0.7477  0.7812 0.8788    0
## SVM 0.4375  0.6562  0.697 0.7021  0.7812 0.9062    0
```

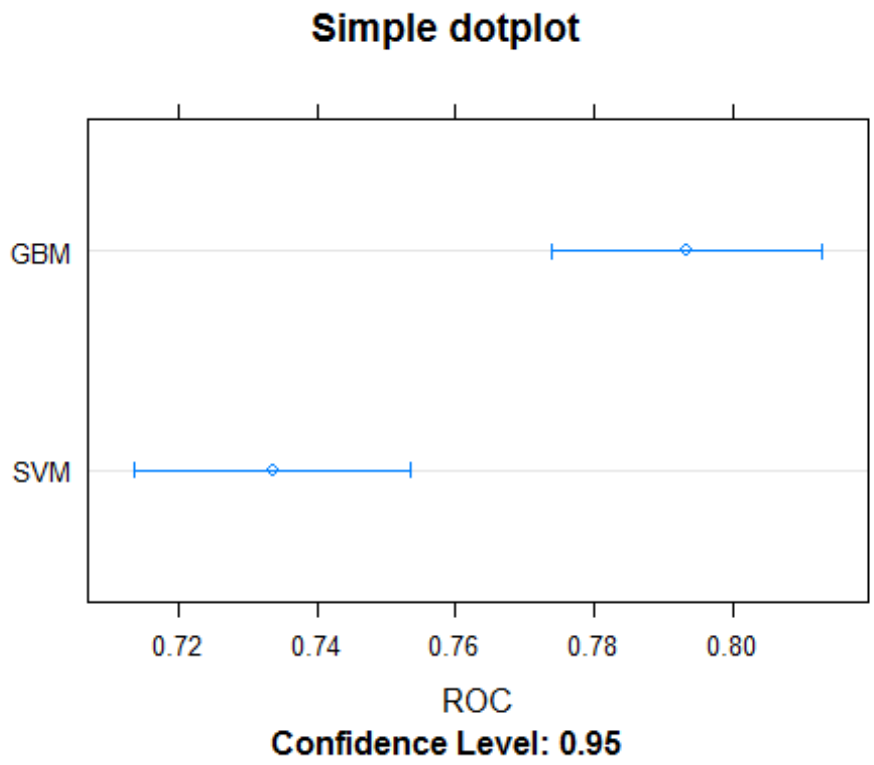
visualize with trellis and dotplots

```
library(lattice)
bwplot(rsmpl, main="GBM vs. SVM model comparisons")
```



save as .PNG

```
png(width=6, height=6, unit="in", res=720)
bwplot(rsmpl, main="Plot of GBM v. SVM models")
dev.off()

dotplot(rsmpl, metric="ROC", main="Simple dotplot")
```



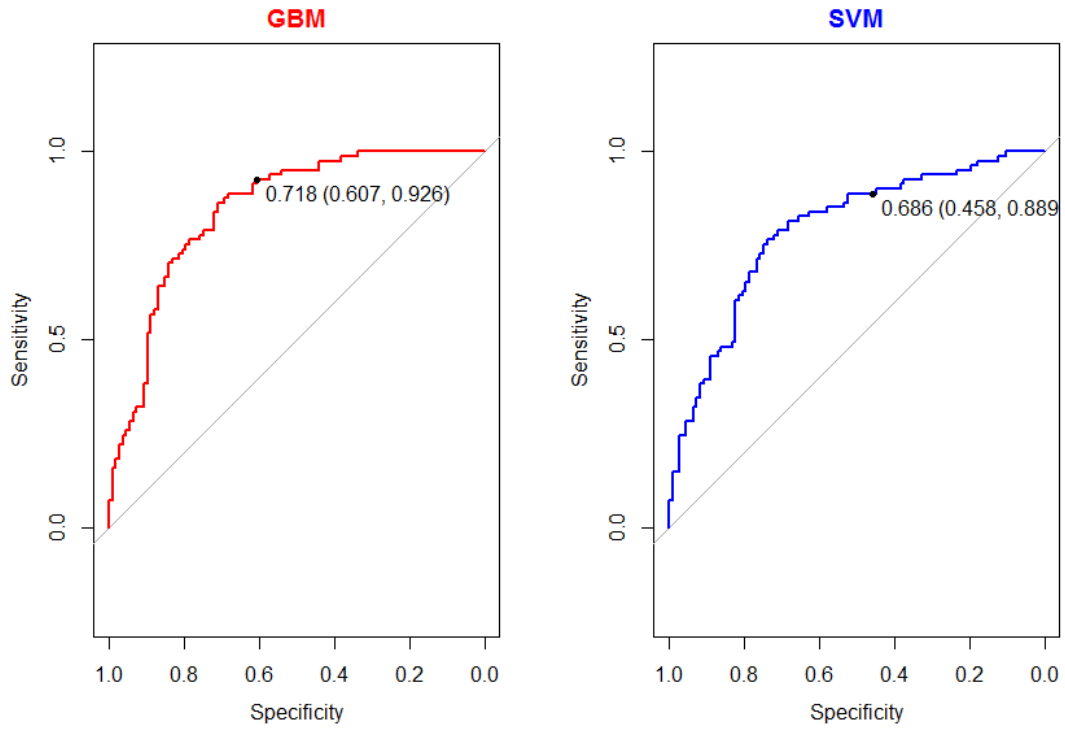
save as .PNG

```
png(width=6, height=6, unit="in", res=720)
dotplot(rsmpl, metric="ROC", main="Simple dotplot")
dev.off()

## png
##   2
```

plot GBM and SCM ROC curves together

Plot results

```
par(mfrow=c(1,2))
plot(rocCurve, print.thres=0.7181, col="red", col.main="red", col.lab="black", m
ain="GBM")

##
## Call:
## roc.default(response = test.set$lfp, predictor = gbm.prob[, "yes"],     level
s = rev(levels(test.set$lfp)), auc = TRUE, ci = TRUE)
##
## Data: gbm.prob[, "yes"] in 107 controls (test.set$lfp yes) > 81 cases (test.s
et$lfp no).
## Area under the curve: 0.8461
## 95% CI: 0.7912-0.901 (DeLong)

plot(rocCurve2, print.thres=0.6864, col="blue", col.main="blue", col.lab="black"
, main="SVM")
```




```
##
## Call:
## roc.default(response = test.set$lfp, predictor = svm.prob[, "yes"],     level
s = rev(levels(test.set$lfp)), auc = TRUE, ci = TRUE)
##
## Data: svm.prob[, "yes"] in 107 controls (test.set$lfp yes) > 81 cases (test.s
et$lfp no).
## Area under the curve: 0.7895
## 95% CI: 0.7239-0.8552 (DeLong)
```

Save as .PNG

```
png(width=12, height=6, unit="in", res=720)
par(mfrow=c(1,2))
plot(rocCurve, print.thres=0.7181, col="red", col.main="red", col.lab="black", m
ain="GBM")
plot(rocCurve2, print.thres=0.6864, col="blue", col.main="blue", col.lab="black"
, main="SVM")
dev.off()
```

t-test to compare p-values of AUC differences

```
set.seed(1)
difValues <- diff(rsmpl)
difValues
```

```
##
## Call:
## diff.resamples(x = rsmpl)
##
## Models: GBM, SVM
## Metrics: ROC, Sens, Spec
## Number of differences: 1
## p-value adjustment: bonferroni
```

```
summary(difValues)
```

```
##
## Call:
## summary.diff.resamples(object = difValues)
##
## p-value adjustment: bonferroni
## Upper diagonal: estimates of the difference
## Lower diagonal: p-value for H0: difference = 0
##
## ROC
##     GBM       SVM
## GBM           0.05974
## SVM 1.316e-09
##
## Sens
##     GBM       SVM
## GBM           0.07357
## SVM 0.0004423
##
## Spec
##     GBM     SVM
## GBM         0.04551
## SVM 0.00262
```