exercise9-10

Lee Rui (BA19008042)

2020/5/4

buidling a model with caret

including following steps: load data, preProcess data,select features, split dataset, build and assess the model

```
rm(list = ls())
```

install and load caret package

install.packages(“caret”, dependencies = TRUE, INSTALL_opts =‘–no-lock’)s

Load a dataset

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```
data <- read.csv("E:/RStudio/workspace/ecology/exercise9-10/npcl11.csv")
str(data)
```

```
## 'data.frame':    14 obs. of  13 variables:
##  $ apiary.no: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ P        : num  0.15 0.1 0.08 0 0 0 0 0.13 0 0.08 ...
##  $ PA       : num  0.23 0.09 0.08 0 0.09 0 0 0.28 0.11 0.27 ...
##  $ Q        : num  0.5 0.55 0.84 0.45 0.45 0.32 0.38 0.69 0.39 0.63 ...
##  $ G        : num  0.29 0.27 0.24 0.32 0.32 0.23 0.22 0.27 0.23 0.25 ...
##  $ GA       : num  0.35 0.32 0.36 0.24 0.26 0.22 0.49 0.82 0.44 1.2 ...
##  $ CF       : num  0.49 0.33 0.37 0.21 0.26 0.32 0.28 1.18 0.37 0.49 ...
##  $ F        : num  0.28 0.03 0.01 0.103 0.086 0.04 0.01 0.053 0.02 0.09 ...
##  $ HB       : num  2.11 1.68 2.73 1.88 4.12 ...
##  $ CA       : num  2.3 1.85 2.23 2.03 1.89 1 1.08 2.03 2.06 1.04 ...
##  $ IA       : num  0 2.11 0 1.39 1.27 0 1.36 2.43 1.76 3.66 ...
##  $ HVA      : num  0.05 0.02 0.11 0.09 0.03 0.029 0.03 0.06 0.07 0.04 ...
##  $ loss_rate: num  0.332 0.691 0.223 0.604 0.57 ...
```

```
head(data)
```

```
##   apiary.no    P   PA    Q    G   GA   CF     F    HB   CA   IA   HVA loss_rate
## 1         1 0.15 0.23 0.50 0.29 0.35 0.49 0.280 2.110 2.30 0.00 0.050    0.3319
## 2         2 0.10 0.09 0.55 0.27 0.32 0.33 0.030 1.679 1.85 2.11 0.020    0.6912
## 3         3 0.08 0.08 0.84 0.24 0.36 0.37 0.010 2.730 2.23 0.00 0.110    0.2233
## 4         4 0.00 0.00 0.45 0.32 0.24 0.21 0.103 1.879 2.03 1.39 0.090    0.6043
## 5         5 0.00 0.09 0.45 0.32 0.26 0.26 0.086 4.120 1.89 1.27 0.030    0.5702
## 6         6 0.00 0.00 0.32 0.23 0.22 0.32 0.040 3.220 1.00 0.00 0.029    0.2312
```

preProcess data

```
library(Hmisc, quietly=TRUE)
```

```
## Warning: package 'Hmisc' was built under R version 3.6.3
```

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

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

```
##
## Attaching package: 'Hmisc'
```

```
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
contents(data)
```

```
##
## Data frame:data  14 observations and 13 variables    Maximum # NAs:0
##
##
##           Storage
## apiary.no integer
## P          double
## PA         double
## Q          double
## G          double
## GA         double
## CF         double
## F          double
## HB         double
## CA         double
## IA         double
## HVA        double
## loss_rate  double
```

```r
summary(data)
```

```
##    apiary.no           P                PA               Q
##  Min.   : 1.00   Min.   :0.0000   Min.   :0.000   Min.   :0.2200
##  1st Qu.: 4.25   1st Qu.:0.0000   1st Qu.:0.035   1st Qu.:0.3950
##  Median : 7.50   Median :0.0400   Median :0.090   Median :0.4500
##  Mean   : 7.50   Mean   :0.2050   Mean   :0.105   Mean   :0.5236
##  3rd Qu.:10.75   3rd Qu.:0.1225   3rd Qu.:0.120   3rd Qu.:0.6100
##  Max.   :14.00   Max.   :2.1600   Max.   :0.280   Max.   :1.0500
##        G                GA               CF               F
##  Min.   :0.1700   Min.   :0.2100   Min.   :0.2100   Min.   :0.01000
##  1st Qu.:0.2325   1st Qu.:0.2750   1st Qu.:0.2900   1st Qu.:0.03000
##  Median :0.2600   Median :0.3550   Median :0.3450   Median :0.04650
##  Mean   :0.5693   Mean   :0.5286   Mean   :0.5143   Mean   :0.07771
##  3rd Qu.:0.3125   3rd Qu.:0.4825   3rd Qu.:0.4600   3rd Qu.:0.09975
##  Max.   :4.5200   Max.   :1.7100   Max.   :2.0000   Max.   :0.28000
##        HB              CA              IA             HVA
##  Min.   :1.679   Min.   :1.000   Min.   :0.000   Min.   :0.02000
##  1st Qu.:2.120   1st Qu.:1.325   1st Qu.:0.215   1st Qu.:0.03000
##  Median :2.745   Median :1.960   Median :1.315   Median :0.04500
##  Mean   :2.947   Mean   :1.747   Mean   :1.303   Mean   :0.05564
##  3rd Qu.:3.475   3rd Qu.:2.075   3rd Qu.:2.022   3rd Qu.:0.07750
##  Max.   :5.460   Max.   :2.300   Max.   :3.660   Max.   :0.12000
##    loss_rate
##  Min.   :0.1387
##  1st Qu.:0.2681
##  Median :0.4853
##  Mean   :0.4666
##  3rd Qu.:0.6309
##  Max.   :0.8357
```

```r
library(fBasics, quietly=TRUE)
```

```
## Warning: package 'fBasics' was built under R version 3.6.3
```

```
## Warning: package 'timeSeries' was built under R version 3.6.3
```

```r
skewness(data, na.rm=TRUE)
```

```
##   apiary.no           P          PA           Q           G          GA
##  0.00000000  2.91358595  0.64858780  0.95606590  2.96669624  1.61279191
##          CF           F          HB          CA          IA         HVA
##  2.09716109  1.37953842  0.74771579 -0.52675413  0.38488055  0.60978347
##   loss_rate
##  0.07743156
```

impute NA in the dataset

```
library(skimr) # for basic statistics
```

```
## Warning: package 'skimr' was built under R version 3.6.3
```

```
skimmed <- skim_to_wide(data)
```

```
## Warning: 'skim_to_wide' is deprecated.
## Use 'skim()' instead.
## See help("Deprecated")
```

```
skimmed[, 2:12]
```

```
## # A tibble: 13 x 11
##    skim_variable n_missing complete_rate numeric.mean numeric.sd numeric.p0
##    <chr>             <int>         <dbl>        <dbl>      <dbl>      <dbl>
##  1 apiary.no             0             1       7.5        4.18        1
##  2 P                     0             1       0.205      0.566       0
##  3 PA                    0             1       0.105      0.0948      0
##  4 Q                     0             1       0.524      0.218       0.22
##  5 G                     0             1       0.569      1.14        0.17
##  6 GA                    0             1       0.529      0.434       0.21
##  7 CF                    0             1       0.514      0.490       0.21
##  8 F                     0             1       0.0777     0.0763      0.01
##  9 HB                    0             1       2.95       1.07        1.68
## 10 CA                    0             1       1.75       0.468       1
## 11 IA                    0             1       1.30       1.10        0
## 12 HVA                   0             1       0.0556     0.0335      0.02
## 13 loss_rate             0             1       0.467      0.221       0.139
## # ... with 5 more variables: numeric.p25 <dbl>, numeric.p50 <dbl>,
## #   numeric.p75 <dbl>, numeric.p100 <dbl>, numeric.hist <chr>
```

```
preProcess_missingdata_model <- preProcess(data[,2:12], #build a model
                                           method='knnImpute')
preProcess_missingdata_model
```

```
## Created from 14 samples and 11 variables
##
## Pre-processing:
##   - centered (11)
##   - ignored (0)
##   - 5 nearest neighbor imputation (11)
##   - scaled (11)
```

```r
library(RANN) # imputing NA algorithm
```

```
## Warning: package 'RANN' was built under R version 3.6.3
```

```r
data_NA <- predict(preProcess_missingdata_model, newdata = data)
anyNA(data_NA)
```

```
## [1] FALSE
```

for one-hot code

```r
dummies_model <- dummyVars(loss_rate ~ .,
                           data= data_NA)# build a model
data_NA_dum_mat <- predict(dummies_model,
                           newdata = data_NA)
data_NA_dum <- data.frame(data_NA_dum_mat)#rebuild a dataframe including target

loss_rate <- data_NA$loss_rate
data_clean <- cbind(loss_rate,data_NA_dum)
head(data_clean)
```

```
##   loss_rate apiary.no           P         PA          Q          G         GA
## 1    0.3319         1 -0.09712975  1.3190257 -0.1080463 -0.2453443 -0.4117625
## 2    0.6912         2 -0.18542953 -0.1582831  0.1211428 -0.2629137 -0.4809386
## 3    0.2233         3 -0.22074943 -0.2638051  1.4504401 -0.2892678 -0.3887038
## 4    0.6043         4 -0.36202907 -1.1079816 -0.3372355 -0.2189902 -0.6654081
## 5    0.5702         5 -0.36202907 -0.1582831 -0.3372355 -0.2189902 -0.6192907
## 6    0.2312         6 -0.36202907 -1.1079816 -0.9331274 -0.2980525 -0.7115255
##            CF          F         HB         CA          IA        HVA
## 1 -0.04953062  2.6509282 -0.7825834  1.1811167 -1.18743573 -0.1684091
## 2 -0.37584996 -0.6252896 -1.1855624  0.2197426  0.73563727 -1.0637485
## 3 -0.29427013 -0.8873870 -0.2028920  1.0315697 -1.18743573  1.6222697
## 4 -0.62058948  0.3313660 -0.9985651  0.6042923  0.07942278  1.0253768
## 5 -0.51861468  0.1085832  1.0967387  0.3051981 -0.02994630 -0.7653020
## 6 -0.39624492 -0.4942409  0.2552512 -1.5961862 -1.18743573 -0.7951467
```

transforming data

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.6.3

## -- Attaching packages ----------------------------------------------- tidyverse 1.3.0 --

## v tibble  3.0.1     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## v purrr   0.3.3
## Warning: package 'tibble' was built under R version 3.6.3

## Warning: package 'tidyr' was built under R version 3.6.3

## Warning: package 'readr' was built under R version 3.6.3

## Warning: package 'purrr' was built under R version 3.6.3

## Warning: package 'dplyr' was built under R version 3.6.3

## Warning: package 'forcats' was built under R version 3.6.3

## -- Conflicts ---------------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter()    masks timeSeries::filter(), stats::filter()
## x dplyr::lag()       masks timeSeries::lag(), stats::lag()
## x purrr::lift()      masks caret::lift()
## x dplyr::src()       masks Hmisc::src()
## x dplyr::summarize() masks Hmisc::summarize()
```

```
data_class <- data [,-1] %>% mutate(loss_rate = case_when(loss_rate >= 0.4 ~ 'serious',loss_rate < 0.4
head(data_class)
```

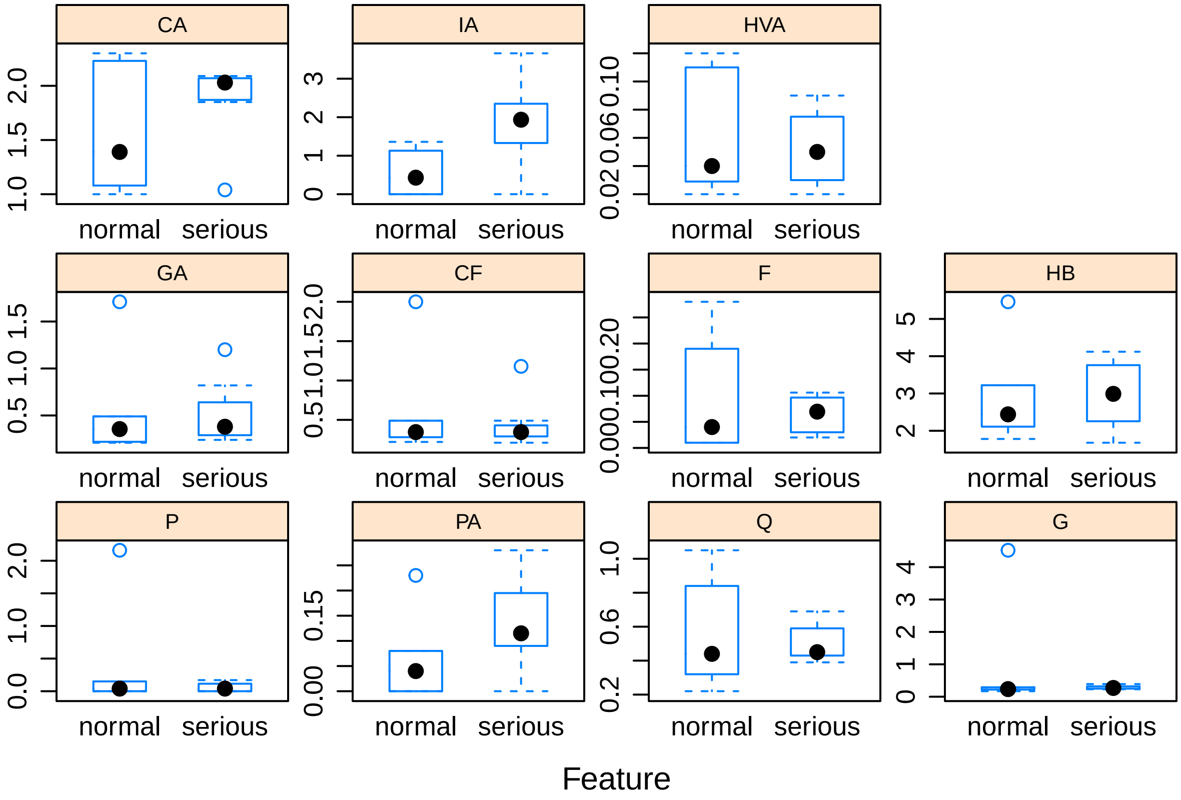
```
##      P   PA    Q    G   GA   CF     F    HB   CA   IA   HVA loss_degree
## 1 0.15 0.23 0.50 0.29 0.35 0.49 0.280 2.110 2.30 0.00 0.050      normal
## 2 0.10 0.09 0.55 0.27 0.32 0.33 0.030 1.679 1.85 2.11 0.020     serious
## 3 0.08 0.08 0.84 0.24 0.36 0.37 0.010 2.730 2.23 0.00 0.110      normal
## 4 0.00 0.00 0.45 0.32 0.24 0.21 0.103 1.879 2.03 1.39 0.090     serious
## 5 0.00 0.09 0.45 0.32 0.26 0.26 0.086 4.120 1.89 1.27 0.030     serious
## 6 0.00 0.00 0.32 0.23 0.22 0.32 0.040 3.220 1.00 0.00 0.029      normal
```

```
write.csv(data_class, file = "E:/RStudio/workspace/ecology/exercise9-10/data_class.csv")
```

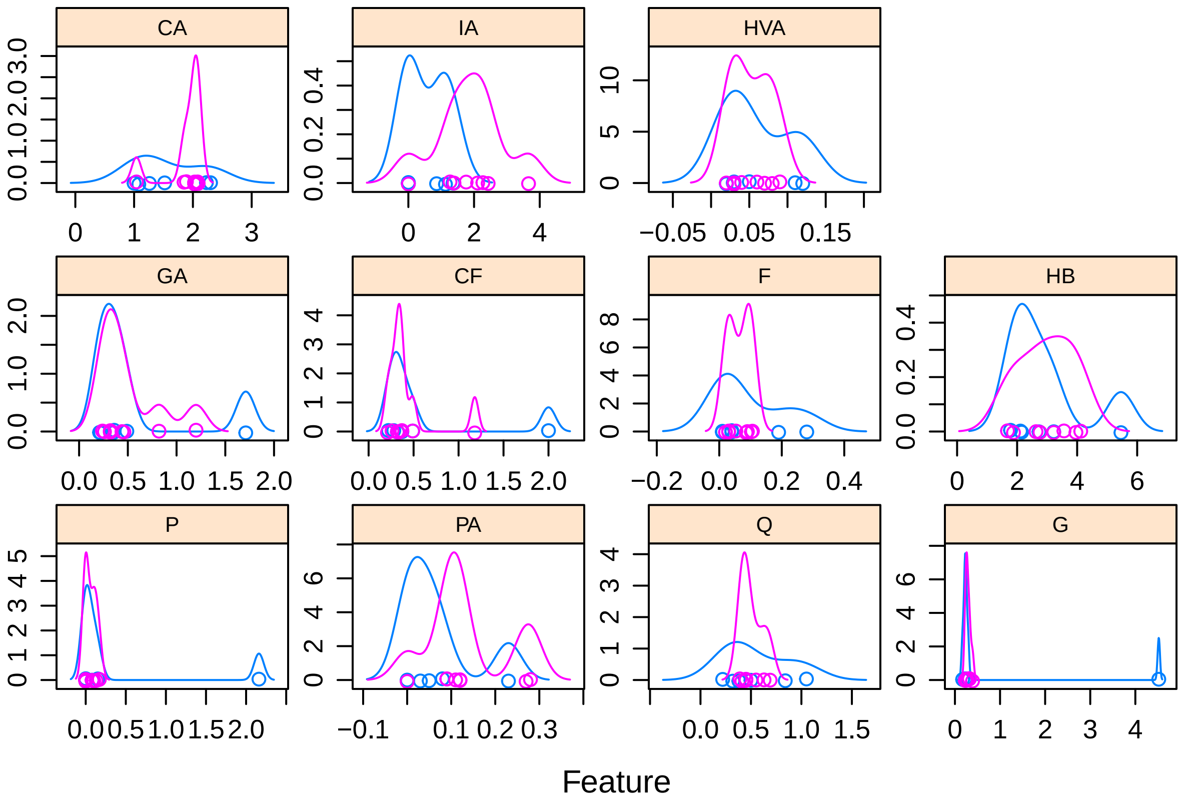
select features

visualize feature importance

```
x = as.matrix(data_class[, 1:11])
y = as.factor(data_class$loss_degree)
featurePlot(x, y, plot = "box",
            strip=strip.custom(par.strip.text=list(cex=.7)),
            scales = list(x = list(relation = "free"),
                          y = list(relation="free")))
```



```
featurePlot(x, y, plot = "density",
            strip=strip.custom(par.strip.text=list(cex=.7)),
            scales = list(x = list(relation="free"),
                          y = list(relation="free")))
```



estimate feature importance using one of three methods (caret)

automatically selecting a subset of the most predictive features

```
options(warn=-1)
set.seed(1234)
```

```
subsets <- c(1:5, 8, 11)
rfectrl <- rfeControl(functions = rfFuncs, #random forest algorithem
                   method = "repeatedcv",
                   repeats = 5,
                   verbose = FALSE)
ImProfile <- rfe(x, y,
                 sizes=subsets,
                 rfeControl=rfectrl)# run the RFE algorithm
print(ImProfile)
```

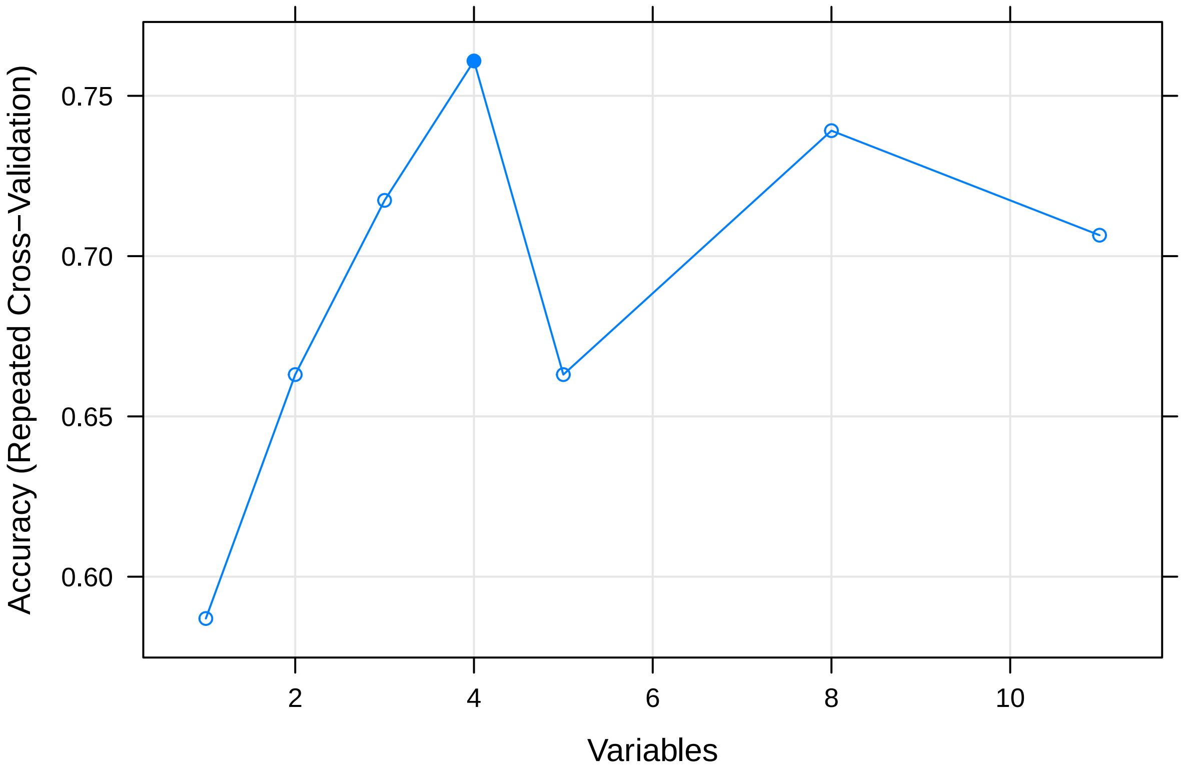
```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (10 fold, repeated 5 times)
##
## Resampling performance over subset size:
```

```
##
##  Variables Accuracy  Kappa AccuracySD KappaSD Selected
##          1   0.5870 0.0625     0.3987  0.5040
##          2   0.6630 0.2258     0.3952  0.5603
##          3   0.7174 0.2143     0.3749  0.6299
##          4   0.7609 0.4138     0.3762  0.6278        *
##          5   0.6630 0.2258     0.3952  0.5603
##          8   0.7391 0.3448     0.3454  0.4837
##         11   0.7065 0.2414     0.3738  0.5766
##
## The top 4 variables (out of 4):
##    IA, PA, CA, G
```

```r
predictors(ImProfile)# list the chosen features
```

```
## [1] "IA" "PA" "CA" "G"
```

```r
plot(ImProfile, type=c("g", "o"))# plot the results
```



searching for and removing redundant features

```r
corr_Matrix <- cor(data_class[,1:11])
print(corr_Matrix)
```

```
##               P          PA          Q          G         GA          CF
## P    1.00000000 -0.09324741 0.73424228  0.9953205  0.8052531  0.89597453
## PA  -0.09324741  1.00000000 0.26435532 -0.1579788  0.3217441  0.21686727
## Q    0.73424228  0.26435532 1.00000000  0.7007531  0.7592811  0.78865266
## G    0.99532051 -0.15797882 0.70075311  1.0000000  0.7832812  0.87249189
## GA   0.80525312  0.32174406 0.75928115  0.7832812  1.0000000  0.86198411
## CF   0.89597453  0.21686727 0.78865266  0.8724919  0.8619841  1.00000000
## F    0.47224473  0.31656694 0.30623763  0.4428204  0.3459145  0.41416496
## HB   0.67805249  0.07120336 0.50064102  0.6895438  0.6362791  0.60027169
## CA  -0.09295721  0.28734410 0.19159324 -0.1154100 -0.2620213 -0.02491238
## IA  -0.08369630  0.44424874 0.05346081 -0.1061270  0.3678361  0.05954336
## HVA  0.54288703 -0.04590664 0.69261345  0.5531851  0.4183219  0.52745162
##              F          HB          CA          IA         HVA
## P    0.4722447  0.67805249 -0.09295721 -0.08369630  0.54288703
## PA   0.3165669  0.07120336  0.28734410  0.44424874 -0.04590664
## Q    0.3062376  0.50064102  0.19159324  0.05346081  0.69261345
## G    0.4428204  0.68954378 -0.11541000 -0.10612695  0.55318510
## GA   0.3459145  0.63627906 -0.26202130  0.36783607  0.41832189
## CF   0.4141650  0.60027169 -0.02491238  0.05954336  0.52745162
## F    1.0000000  0.29344781  0.23446165 -0.13377953  0.17541509
## HB   0.2934478  1.00000000 -0.08971812  0.02928921  0.32445652
## CA   0.2344617 -0.08971812  1.00000000 -0.21469055  0.39911477
## IA  -0.1337795  0.02928921 -0.21469055  1.00000000 -0.33357598
## HVA  0.1754151  0.32445652  0.39911477 -0.33357598  1.00000000
```

```r
highlyCorr <- findCorrelation(corr_Matrix, cutoff=0.5)
print(highlyCorr)
```

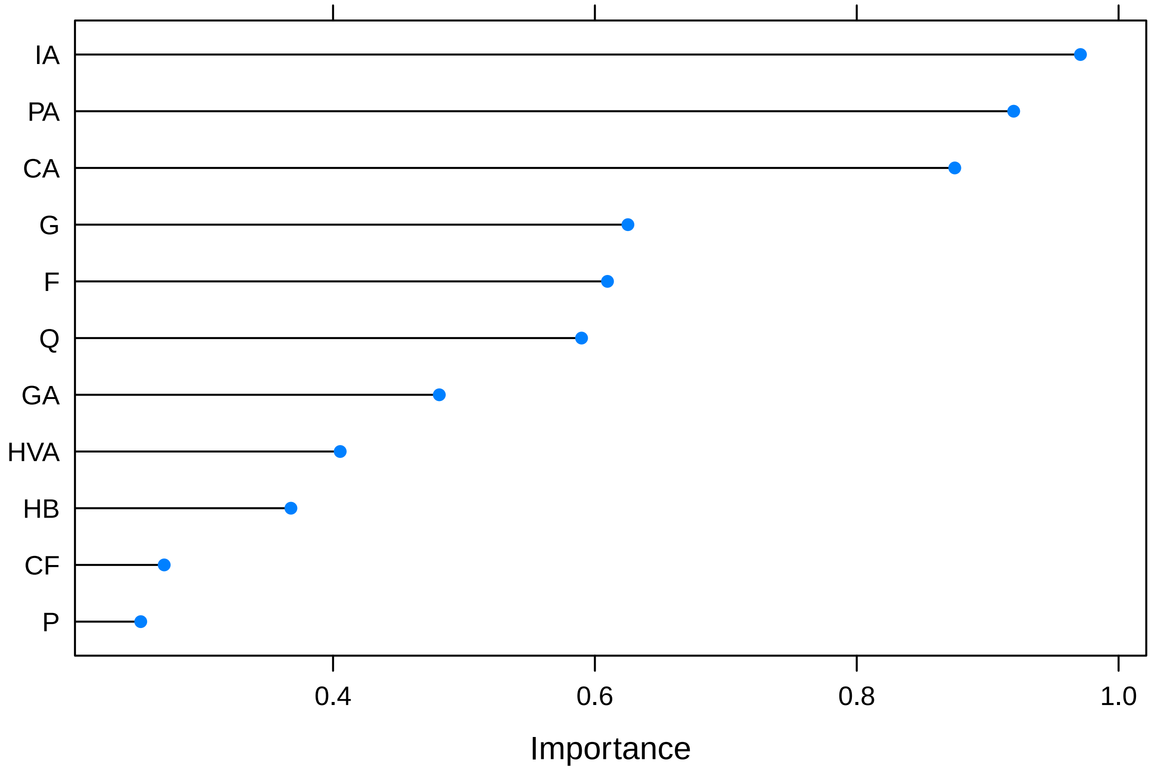
```
## [1] 5 4 1 6 3
```

ranking features by importance

```r
importance_control <- trainControl(method="repeatedcv",
                        number=10, repeats=3)# cross-validation
model <- train(loss_degree~.,
               data=data_class,
               method="rf",
               preProcess="scale",
               trControl=importance_control)# train the model
importance <- varImp(model, scale=FALSE)
print(importance)# summarize importance
```

```
## rf variable importance
##
##     Overall
## IA   0.9709
## PA   0.9199
## CA   0.8749
## G    0.6252
## F    0.6096
## Q    0.5898
## GA   0.4812
## HVA  0.4055
## HB   0.3678
## CF   0.2710
## P    0.2531
```

```
plot(importance)# plot importance
```



train and tune models

split the dataset

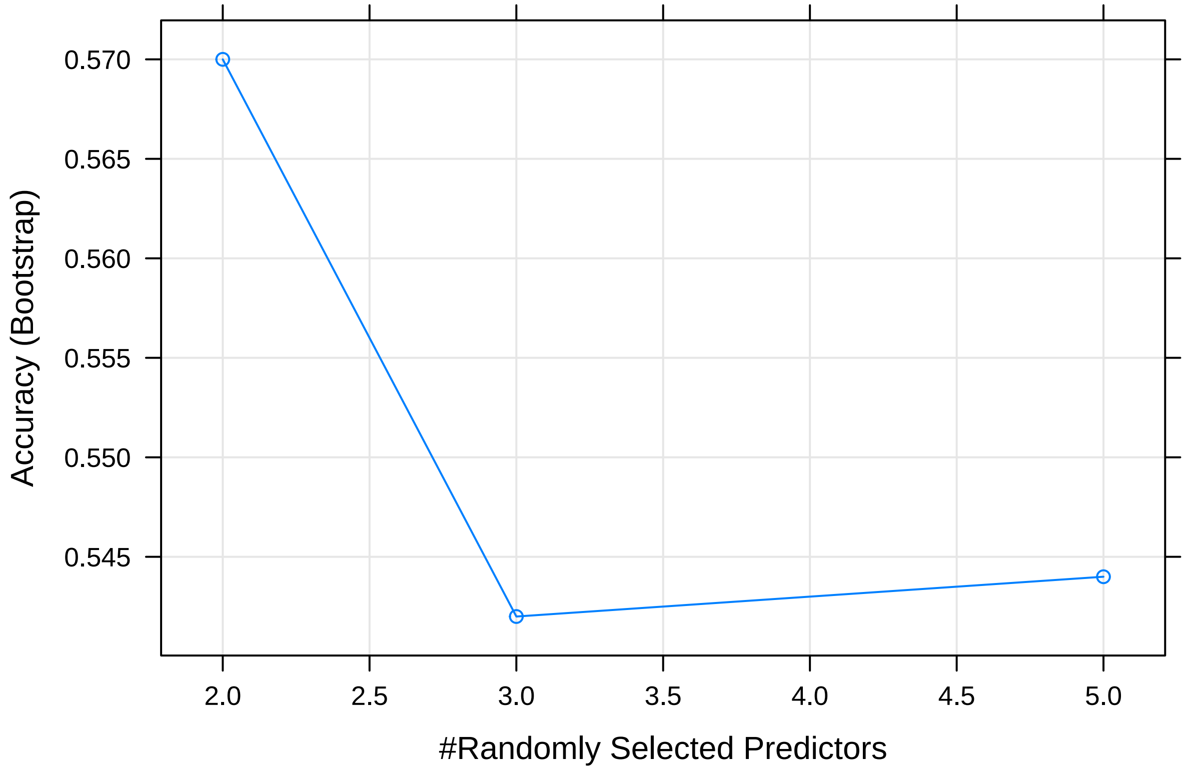
```
set.seed(1234)
train_idx <- createDataPartition(data_class$loss_degree, p=0.75, list=FALSE)
training <- data_class[train_idx,]
test <- data_class[-train_idx,]
```

build rf model and evaluate its performance

```
set.seed(1234)#build rf model
rf_fit <- train(as.factor(loss_degree) ~ IA + PA + CA + Q + G,
                data = training,
                method = "rf")
rf_fit
```

```
## Random Forest
##
## 11 samples
##  5 predictor
##  2 classes: 'normal', 'serious'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 11, 11, 11, 11, 11, 11, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##   2     0.570     0.1587773
##   3     0.542     0.1283578
##   5     0.544     0.1317460
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
plot(rf_fit)
```



```
rf_pred <- predict(rf_fit, test)#evaluate rf performance
rf_pred
```

```
## [1] normal  serious serious
## Levels: normal serious
```

```
confusionMatrix(reference = as.factor(test$loss_degree),
                data = rf_pred,
                mode = "everything")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction normal serious
##    normal       1       0
##    serious      0       2
##
##                Accuracy : 1
##                  95% CI : (0.2924, 1)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.2963
##
##                   Kappa : 1
##
```

```
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##               Precision : 1.0000
##                  Recall : 1.0000
##                      F1 : 1.0000
##              Prevalence : 0.3333
##          Detection Rate : 0.3333
##    Detection Prevalence : 0.3333
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : normal
##
```

set uneLength or tuneGrid for better model performance

```
uneLength_ctrl <- trainControl(
  method = 'cv',
  number = 5,
  savePredictions = 'final',
  classProbs = T,
  summaryFunction=twoClassSummary)

rf_fit <- train(as.factor(loss_degree) ~., #optimize mtry with tuneLength
                data = training,
                method = "rf",
                tuneLength = 5,
                trControl = uneLength_ctrl,
                verbose = FALSE
)
```

evaluate rf performance

```
rf_pred <- predict(rf_fit, test)
rf_pred
```

```
## [1] normal  serious serious
## Levels: normal serious
```

```
confusionMatrix(reference = as.factor(test$loss_degree),
                data = rf_pred,
                mode = "everything")
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction normal serious
##    normal       1       0
##    serious      0       2
##
##                Accuracy : 1
##                  95% CI : (0.2924, 1)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.2963
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##               Precision : 1.0000
##                  Recall : 1.0000
##                      F1 : 1.0000
##              Prevalence : 0.3333
##          Detection Rate : 0.3333
##    Detection Prevalence : 0.3333
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : normal
##
```

```r
library(MLeval)
x <- evalm(rf_fit)
```

```
## ***MLeval: Machine Learning Model Evaluation***
```

```
## Input: caret train function object
```

```
## Not averaging probs.
```

```
## Group 1 type: cv
```

```
## Observations: 11
```

```
## Number of groups: 1
```

```
## Observations per group: 11
```

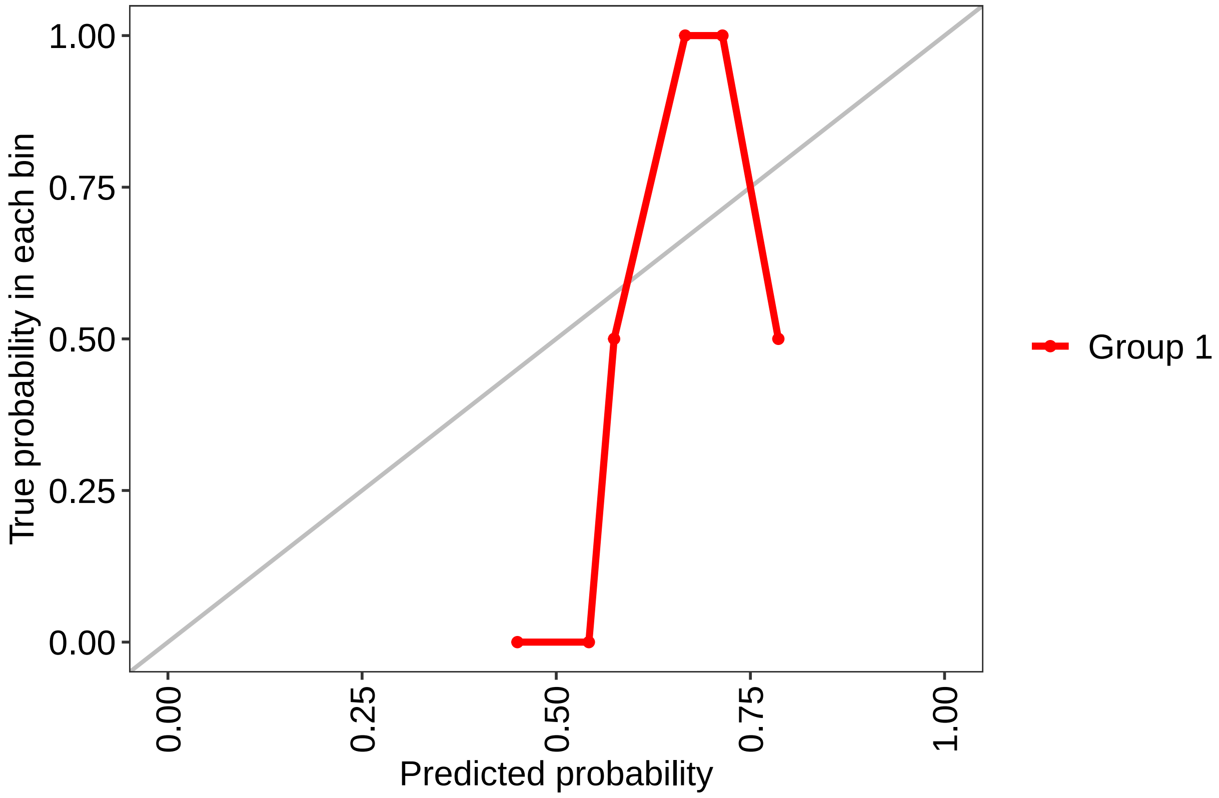
```
## Positive: serious
```

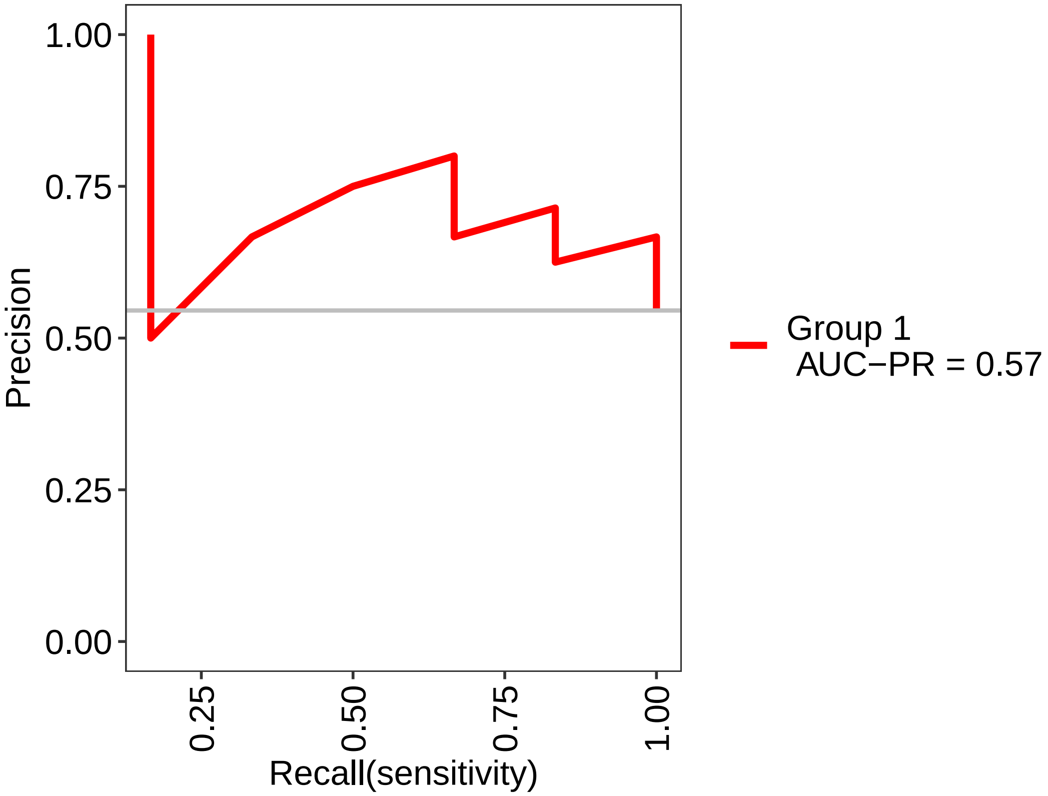
```
## Negative: normal
```

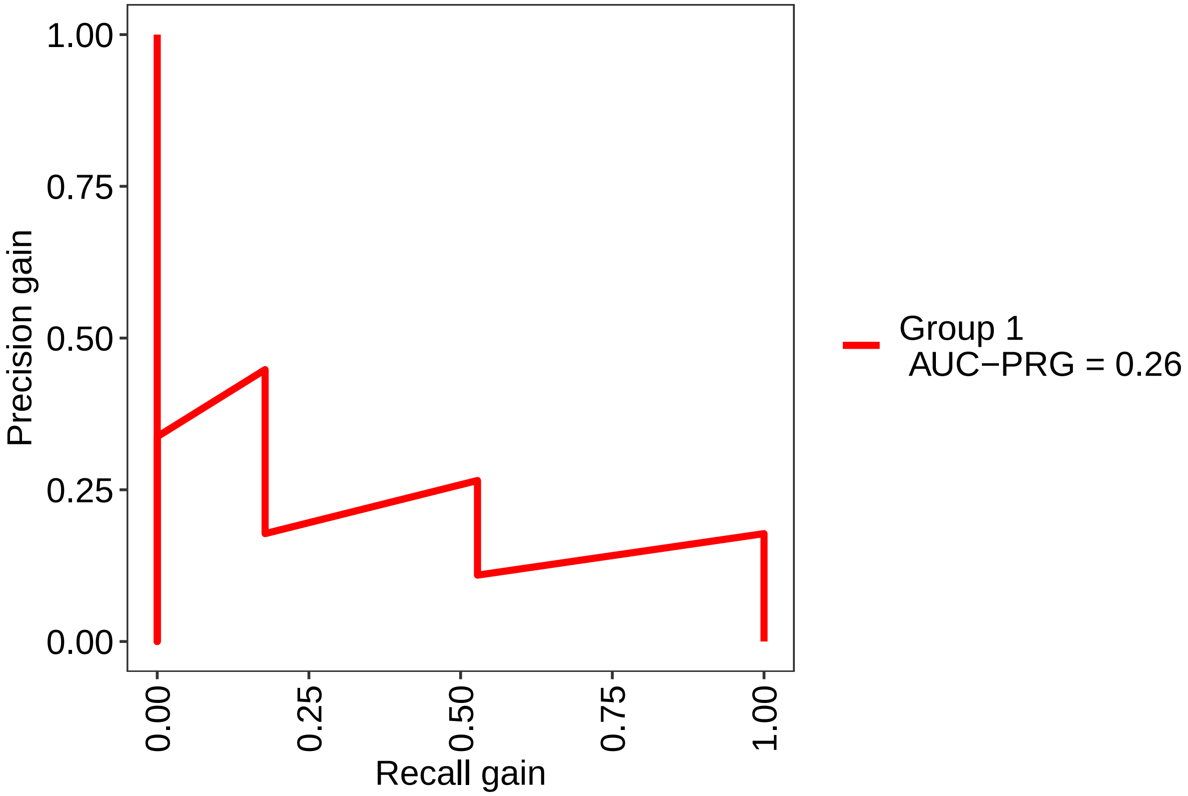
```
## Group: Group 1
## Positive: 6
```

```
## Negative: 5
```

```
## ***Performance Metrics***
```

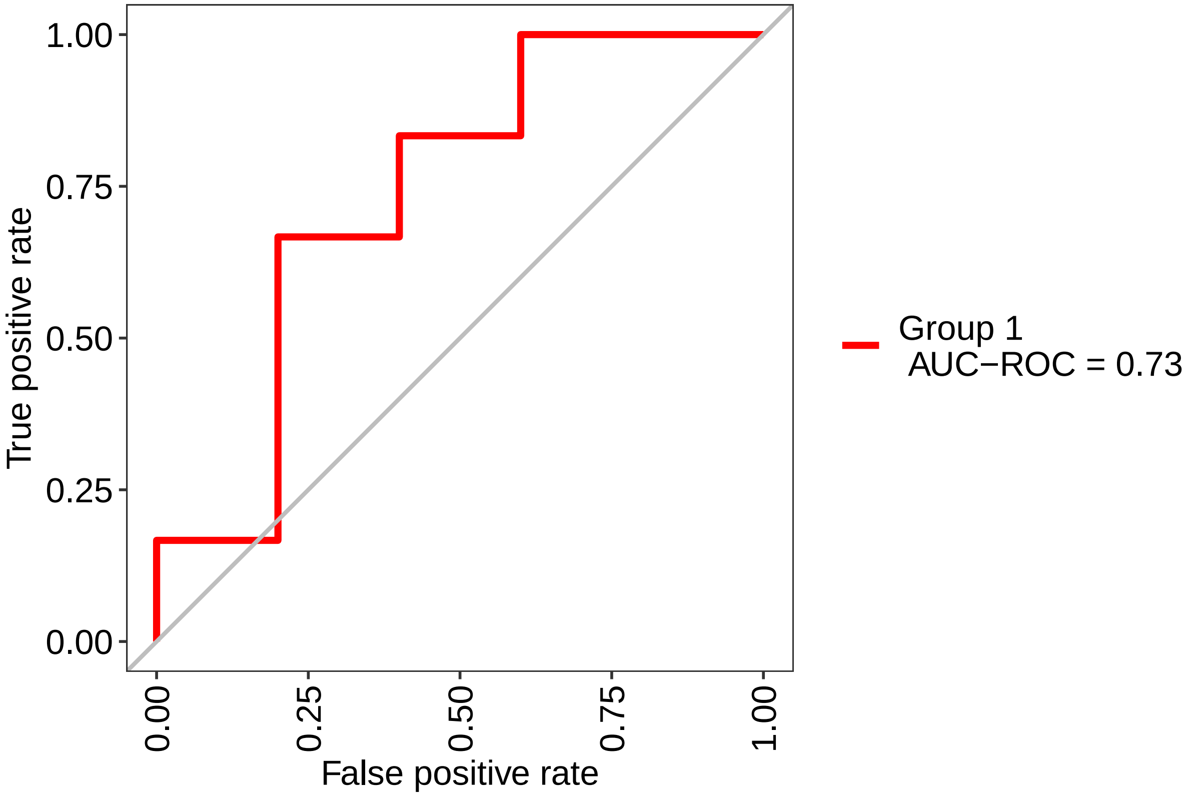



1.00
0.75
0.50
0.25
0.00
Precision
0.25
0.50
0.75
1.00
Recall(sensitivity)
Group 1
AUC−PR = 0.57

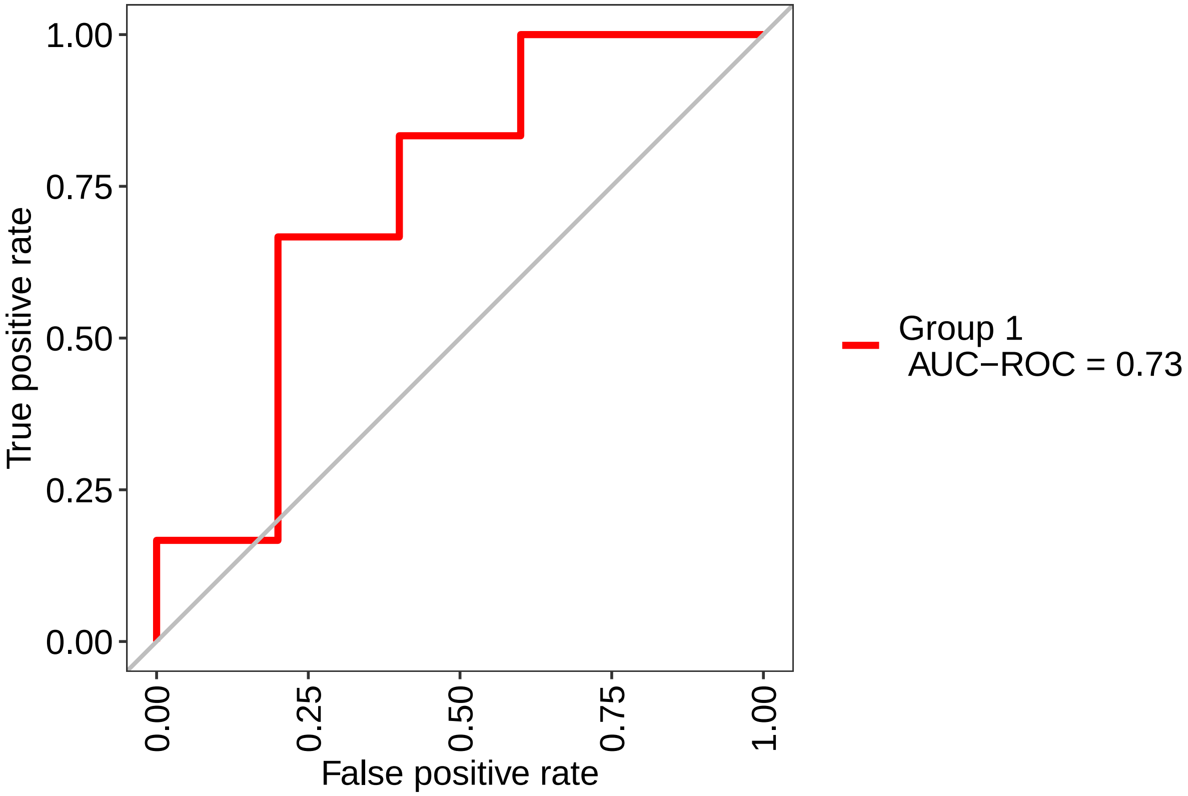


```
## Group 1 Optimal Informedness = 0.466666666666667
```

```
## Group 1 AUC-ROC = 0.73
```


1.00
0.75
0.50
0.25
0.00
True positive rate
0.00
0.25
0.50
0.75
1.00
False positive rate
Group 1
AUC−ROC = 0.73

```
x$roc
```



```
x$stdres
```

```
## $`Group 1`
##             Score        CI
## SENS        1.000    0.61-1
## SPEC        0.200 0.04-0.62
## MCC         0.346      <NA>
## Informedness 0.200      <NA>
## PREC        0.600 0.31-0.83
## NPV         1.000    0.21-1
## FPR         0.800      <NA>
## F1          0.750      <NA>
## TP          6.000      <NA>
## FP          4.000      <NA>
## TN          1.000      <NA>
## FN          0.000      <NA>
## AUC-ROC     0.730 0.42-1.04
## AUC-PR      0.570      <NA>
## AUC-PRG     0.260      <NA>
```

build and compare models

Set up training control

```
ctrl <- trainControl(method = "repeatedcv",
                     number = 5,
                     # Use AUC to pick the best model
                     summaryFunction=twoClassSummary,
                     classProbs=TRUE,
                     allowParallel = TRUE)
```

training multiple models

```
set.seed(1234)
rpart_model = train(as.factor(loss_degree) ~.,
                    data=training,
                    method='rpart',
                    tuneLength=15,
                    trControl = ctrl)
rf_model = train(as.factor(loss_degree) ~.,
                 data=training,
                 method='rf',
                 tuneLength=15,
                 trControl = ctrl)
```

```
## note: only 10 unique complexity parameters in default grid. Truncating the grid to 10 .
```

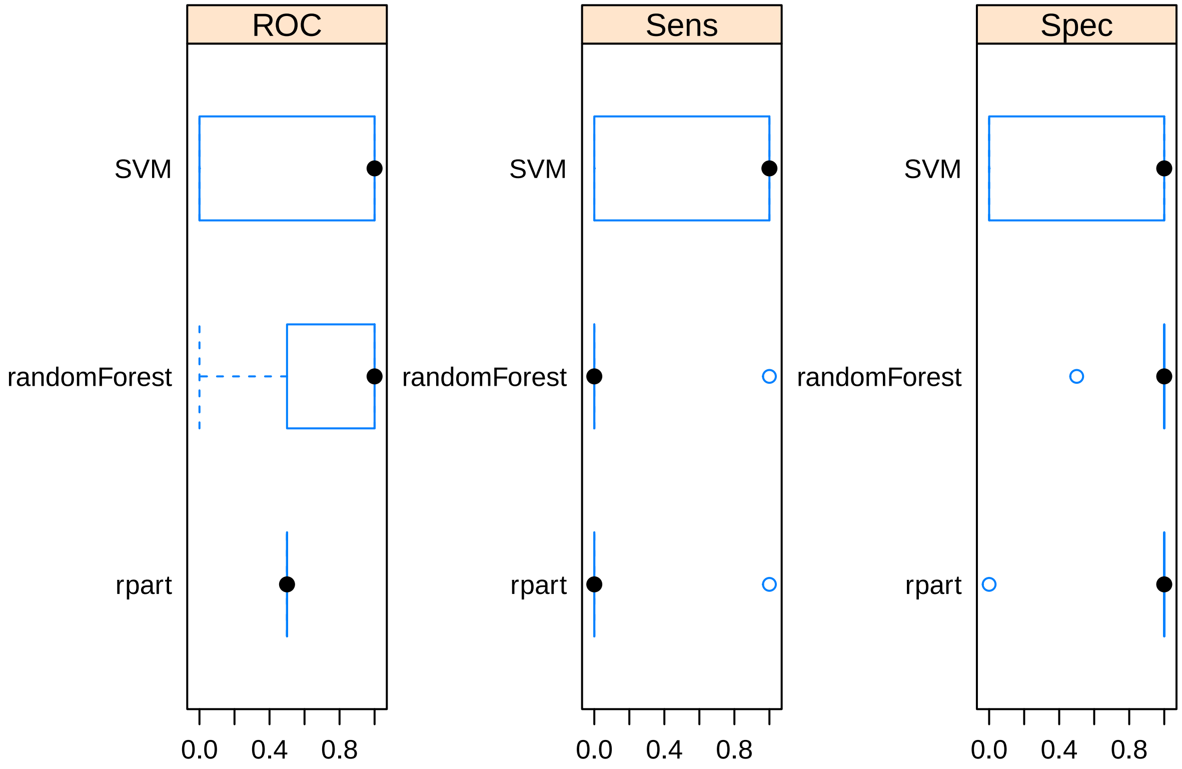
```
svm_model = train(as.factor(loss_degree) ~ .,
                  data=training,
                  method='svmRadial',
                  tuneLength=15,
                  trControl = ctrl)

models_compare <- resamples(list(rpart = rpart_model, randomForest = rf_model, SVM= svm_model))
summary(models_compare)
```

```
##
## Call:
## summary.resamples(object = models_compare)
##
## Models: rpart, randomForest, SVM
## Number of resamples: 5
##
## ROC
##              Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
## rpart         0.5     0.5    0.5  0.5     0.5  0.5    0
## randomForest  0.0     0.5    1.0  0.7     1.0  1.0    0
## SVM           0.0     0.0    1.0  0.6     1.0  1.0    0
##
```

```
## Sens
##              Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
## rpart           0       0      0  0.2       0    1    0
## randomForest    0       0      0  0.2       0    1    0
## SVM             0       0      1  0.6       1    1    0
##
## Spec
##              Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
## rpart         0.0       1      1  0.8       1    1    0
## randomForest  0.5       1      1  0.9       1    1    0
## SVM           0.0       0      1  0.6       1    1    0
```

```
scales <- list(x=list(relation="free"), y=list(relation="free"))
bwplot(models_compare, scales=scales)
```



Stacking Algorithms - Run multiple algos in one call.

```
library(caretEnsemble)
```

```
##
## Attaching package: 'caretEnsemble'

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

```
##
##     autoplot
```

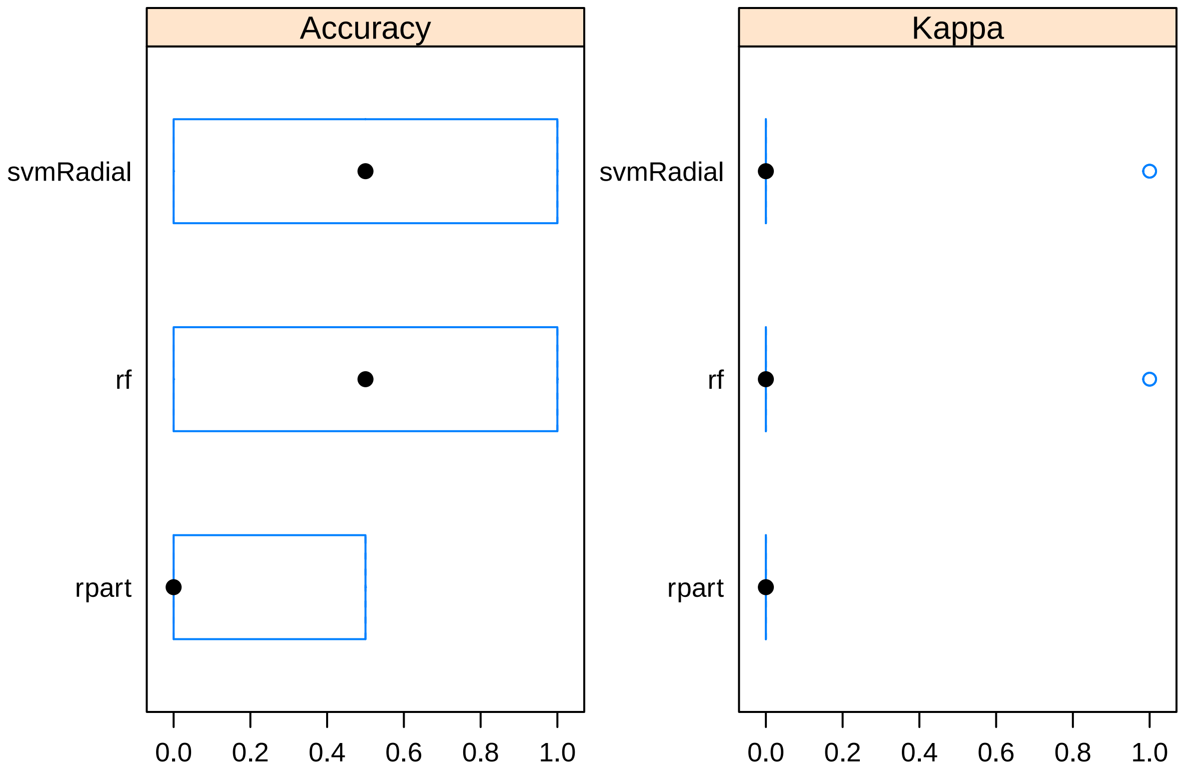
```
caret_ctrl <- trainControl(method="repeatedcv",
                      number=10,
                      repeats=3,
                      savePredictions=TRUE,
                      classProbs=TRUE)

algorithmList <- c('rf', 'rpart', 'svmRadial')

set.seed(1234)
models <- caretList(as.factor(loss_degree) ~ .,
                     data=training,
                     trControl=caret_ctrl,
                     methodList=algorithmList)
results <- resamples(models)
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: rf, rpart, svmRadial
## Number of resamples: 25
##
## Accuracy
##           Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
## rf           0       0    0.5 0.60     1.0  1.0    0
## rpart        0       0    0.0 0.16     0.5  0.5    0
## svmRadial    0       0    0.5 0.48     1.0  1.0    0
##
## Kappa
##           Min. 1st Qu. Median      Mean 3rd Qu. Max. NA's
## rf           0       0      0 0.1333333       0    1   10
## rpart        0       0      0 0.0000000       0    0    0
## svmRadial    0       0      0 0.1111111       0    1    7
```

```
scales <- list(x=list(relation="free"), y=list(relation="free"))
bwplot(results, scales=scales)
```


Accuracy
Kappa
svmRadial
rf
rpart
svmRadial
rf
rpart
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0