exercise_10

chenchunpeng

2020/5/11

```
setwd('E:/Ecology/exercise_10/')

rm(list = ls())

# Load package
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

# Load a dataset
data <- read.csv("./data/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 0.0
4 ...
##  $ loss_rate: num  0.332 0.691 0.223 0.604 0.531 ...

#preProcess data
library(Hmisc, quietly=TRUE)

##
## 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
```

```
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.   :0.2000   Min.   :0.2000   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   :0.5714   Mean   :0.5279   Mean   :0.5143   Mean   :0.07700
```

```
##  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.:2.112   1st Qu.:1.302   1st Qu.:0.215   1st Qu.:0.03000
##  Median :2.745   Median :1.960   Median :1.315   Median :0.04500
##  Mean   :2.945   Mean   :1.745   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.:0.2706
##  Median :0.4853
##  Mean   :0.4645
##  3rd Qu.:0.6309
##  Max.   :0.8357
```

```
library(fBasics, quietly=TRUE)
# Calculate skewness
skewness(data, na.rm=TRUE)
```

```
##  apiary.no          P         PA          Q          G         GA
      CF
##  0.0000000  2.9135860  0.6485878  0.9560659  2.9682123  1.6082590  2.
0971611
##          F         HB         CA         IA        HVA  loss_rate
##  1.3739392  0.7461853 -0.5213997  0.3848806  0.6097835  0.1050796
```

```
#impute NA in the dataset
library(skimr)
# Data integrity and basic statistics
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 num
eric.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       0.571      1.14
  0.2
##  6 GA                    0             1       0.528      0.434
  0.2
##  7 CF                    0             1       0.514      0.490
  0.21
##  8 F                     0             1       0.077      0.0767
  0.01
##  9 HB                    0             1       2.94       1.07
  1.68
## 10 CA                    0             1       1.74       0.471
  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       0.465      0.219
  0.139
## # ... with 5 more variables: numeric.p25 <dbl>, numeric.p50 <dbl>,
## #   numeric.p75 <dbl>, numeric.p100 <dbl>, numeric.hist <chr>

#building model
preProcess_missingdata_model <- preProcess(data[,2:12],
                                           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)

library(RANN)

data_NA <- predict(preProcess_missingdata_model,
                   newdata = data)

# There is no NA
anyNA(data_NA)

## [1] FALSE

# for one-hot code
# Type code to digital code
dummies_model <- dummyVars(loss_rate ~ .,
                           data= data_NA)
```

```
data_NA_dum_mat <- predict(dummies_model,
                           newdata = data_NA)
data_NA_dum <- data.frame(data_NA_dum_mat)#rebuild a dataframe includin
g 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 -0.2473966 -
0.4095743
## 2    0.6912         2 -0.18542953 -0.1582831  0.1211428 -0.2649781 -
0.4786592
## 3    0.2233         3 -0.22074943 -0.2638051  1.4504401 -0.2913503 -
0.3865461
## 4    0.6043         4 -0.36202907 -1.1079816 -0.3372355 -0.2210243 -
0.6628854
## 5    0.5312         5 -0.36202907 -0.1582831 -0.3372355 -0.2210243 -
0.6168288
## 6    0.2312         6 -0.36202907 -1.1079816 -0.9331274 -0.3001410 -
0.7089419
##            CF          F         HB         CA          IA        HV
A
## 1 -0.04953062  2.6455266 -0.7793060  1.1794862 -1.18743573 -0.168409
1
## 2 -0.37584996 -0.6125111 -1.1816273  0.2231460  0.73563727 -1.063748
5
## 3 -0.29427013 -0.8731541 -0.2005606  1.0307222 -1.18743573  1.622269
7
## 4 -0.62058948  0.3388359 -0.9949353  0.6056821  0.07942278  1.025376
8
## 5 -0.51861468  0.1172894  1.0969491  0.3081540 -0.02994630 -0.765302
0
## 6 -0.39624492 -0.4821896  0.2568349 -1.5832742 -1.18743573 -0.795146
7
```

```
##========================================
# transforming data

library(tidyverse)
```

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

## √ tibble  3.0.1     √ dplyr   0.8.5
## √ tidyr   1.0.2     √ stringr 1.4.0
## √ readr   1.3.1     √ forcats 0.5.0
## √ purrr   0.3.3

## -- 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()
```

```
#Classification according to loss rate
data_class <- data [,-1] %>%
  mutate(loss_rate =
           case_when(loss_rate >= 0.4 ~ 'serious',
                     loss_rate < 0.4 ~ 'normal')) %>%
  rename(loss_degree=loss_rate)
head(data_class)
```

```
##      P   PA    Q    G   GA   CF     F    HB   CA   IA   HVA loss_deg
ree
## 1 0.15 0.23 0.50 0.29 0.35 0.49 0.280 2.110 2.30 0.00 0.050      nor
mal
## 2 0.10 0.09 0.55 0.27 0.32 0.33 0.030 1.679 1.85 2.11 0.020     seri
ous
## 3 0.08 0.08 0.84 0.24 0.36 0.37 0.010 2.730 2.23 0.00 0.110      nor
mal
## 4 0.00 0.00 0.45 0.32 0.24 0.21 0.103 1.879 2.03 1.39 0.090     seri
ous
## 5 0.00 0.09 0.45 0.32 0.26 0.26 0.086 4.120 1.89 1.27 0.030     seri
ous
## 6 0.00 0.00 0.32 0.23 0.22 0.32 0.040 3.220 1.00 0.00 0.029      nor
mal
```

```
write.csv(data_class, file = "./data/data_class.csv")
```

```
##========================================
##select features
library('randomForest')
```

```
## randomForest 4.6-14
```

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

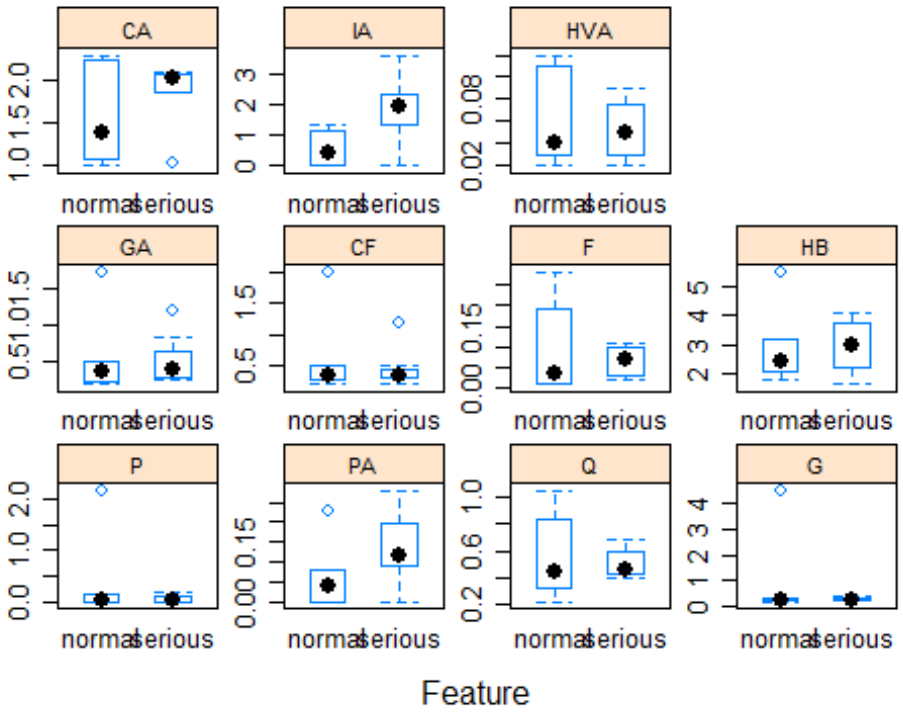
```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

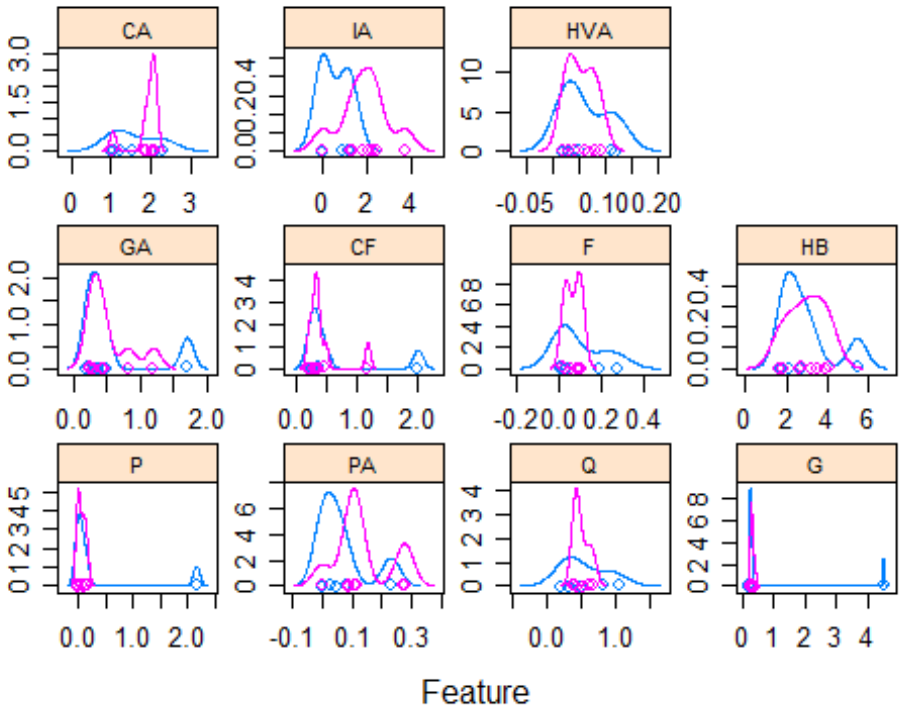
```
## The following object is masked from 'package:timeSeries':
##
##     outlier
```

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

```
x = as.matrix(data_class[, 1:11])
y = as.factor(data_class$loss_degree)

#visualize feature importance
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")))
```



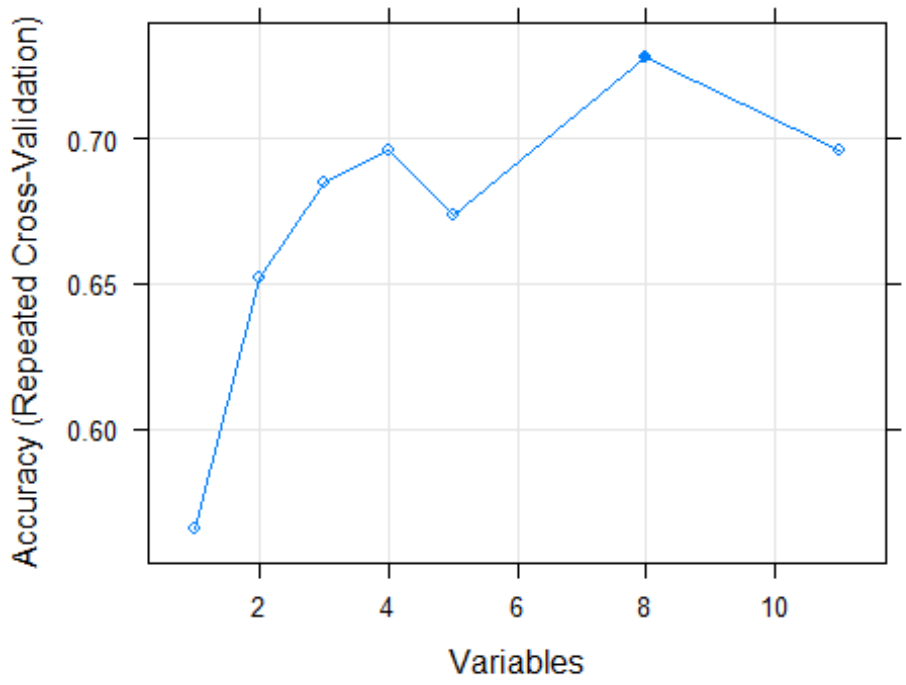

```
##estimate feature importance using one of three methods (caret)
#automatically selecting a subset of the most predictive features
options(warn=-1)
set.seed(1234)

subsets <- c(1:5, 8, 11)
ctrl <- rfeControl(functions = rfFuncs, #random forest algorithem
                   method = "repeatedcv",
                   repeats = 5,
                   verbose = FALSE)
ImProfile <- rfe(x, y,
                 sizes=subsets,
                 rfeControl=ctrl)
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   0.5652 0.0000     0.4029  0.5080
##          2   0.6522 0.1935     0.4065  0.6011
##          3   0.6848 0.1724     0.3853  0.6017
```

```
##          4   0.6957 0.2667     0.4011  0.6397
##          5   0.6739 0.2000     0.3832  0.5509
##          8   0.7283 0.3103     0.3607  0.5414        *
##         11   0.6957 0.2667     0.3870  0.5833
##
## The top 5 variables (out of 8):
##    IA, PA, CA, Q, G
```

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

```
## [1] "IA"  "PA"  "CA"  "Q"   "G"   "F"   "HVA" "GA"
```

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




```
# searching for and removing redundant features
corr_Matrix <- cor(data_class[,1:11])
print(corr_Matrix)
```

```
##               P          PA          Q          G         GA
  CF
## P    1.00000000 -0.09324741 0.73424228  0.9952699  0.8048320  0.8959
7453
## PA  -0.09324741  1.00000000 0.26435532 -0.1596928  0.3227215  0.2168
6727
## Q    0.73424228  0.26435532 1.00000000  0.6984117  0.7607443  0.7886
5266
## G    0.99526985 -0.15969284 0.69841166  1.0000000  0.7818753  0.8718
7370
```

```
## GA   0.80483201  0.32272149 0.76074432  0.7818753  1.0000000  0.8619
1005
## CF   0.89597453  0.21686727 0.78865266  0.8718737  0.8619100  1.0000
0000
## F    0.47325345  0.32274401 0.31848776  0.4429418  0.3519805  0.4178
8351
## HB   0.67772580  0.07279197 0.50282150  0.6880855  0.6373468  0.6005
8498
## CA  -0.09069496  0.28972086 0.19741440 -0.1153834 -0.2547688 -0.0218
3839
## IA  -0.08369630  0.44424874 0.05346081 -0.1065195  0.3676298  0.0595
4336
## HVA  0.54288703 -0.04590664 0.69261345  0.5514072  0.4196543  0.5274
5162
##               F           HB          CA          IA          HVA
## P    0.4732534  0.67772580 -0.09069496 -0.08369630  0.54288703
## PA   0.3227440  0.07279197  0.28972086  0.44424874 -0.04590664
## Q    0.3184878  0.50282150  0.19741440  0.05346081  0.69261345
## G    0.4429418  0.68808552 -0.11538344 -0.10651946  0.55140717
## GA   0.3519805  0.63734683 -0.25476878  0.36762985  0.41965432
## CF   0.4178835  0.60058498 -0.02183839  0.05954336  0.52745162
## F    1.0000000  0.30012071  0.24532204 -0.13145784  0.18510549
## HB   0.3001207  1.00000000 -0.08309639  0.02958079  0.32621848
## CA   0.2453220 -0.08309639  1.00000000 -0.21279375  0.40224188
## IA  -0.1314578  0.02958079 -0.21279375  1.00000000 -0.33357598
## HVA  0.1851055  0.32621848  0.40224188 -0.33357598  1.00000000
```

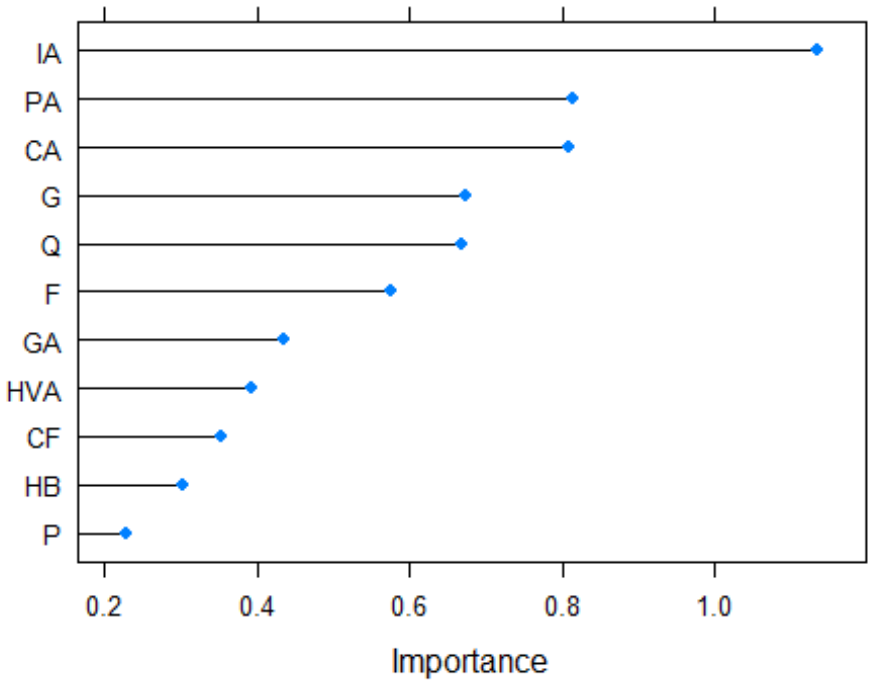
```
highlyCorr <- findCorrelation(corr_Matrix, cutoff=0.5)
print(highlyCorr)
```

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

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

```
## rf variable importance
##
##     Overall
## IA   1.1356
## PA   0.8131
## CA   0.8095
## G    0.6748
## Q    0.6695
```

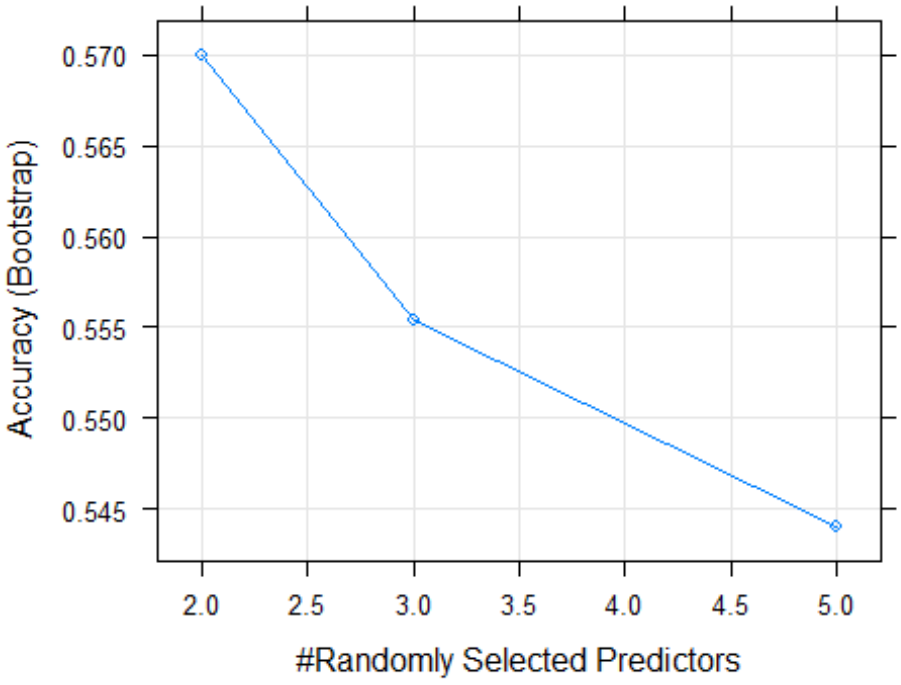
```
## F    0.5770
## GA   0.4353
## HVA  0.3922
## CF   0.3527
## HB   0.3026
## P    0.2282

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=F
ALSE)
training <- data_class[train_idx,]
test <- data_class[-train_idx,]

##build rf model and evaluate its performance
#build rf model
set.seed(1234)
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     0.5700000  0.1587773
##   3     0.5553333  0.1700244
##   5     0.5440000  0.1317460
##
## Accuracy was used to select the optimal model using the largest valu
e.
## The final value used for the model was mtry = 2.
```

```
plot(rf_fit)
```

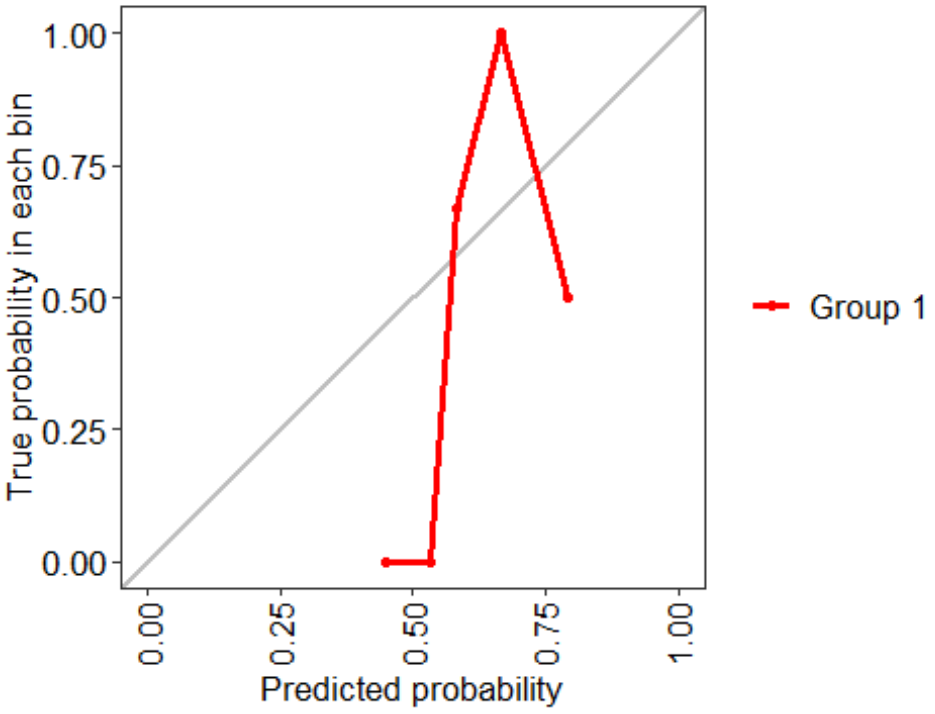


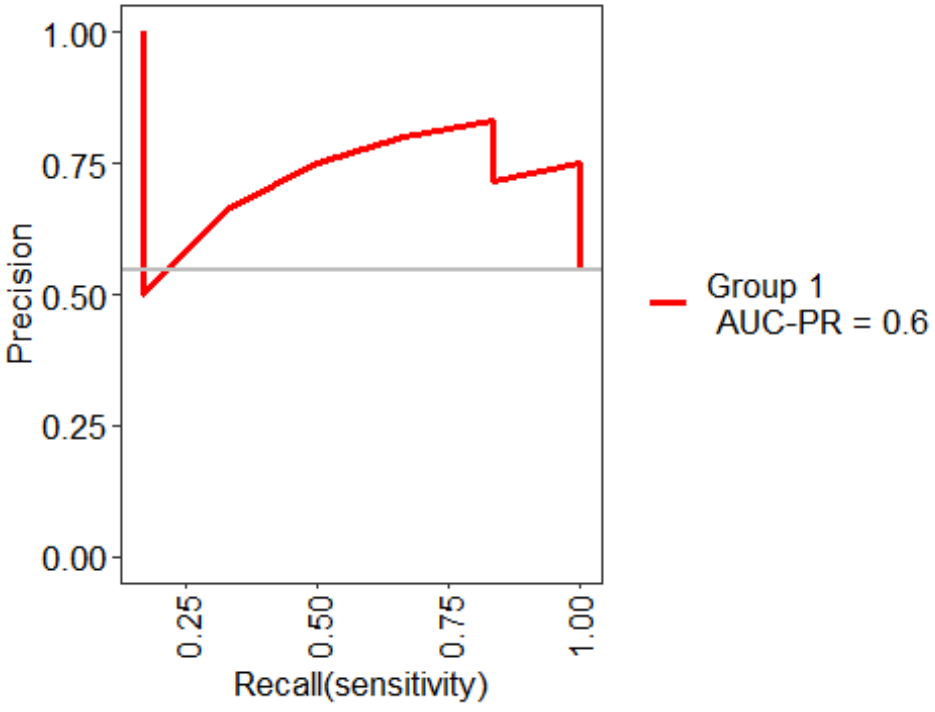

```
#evaluate rf performance
rf_pred <- predict(rf_fit, test)
rf_pred
```

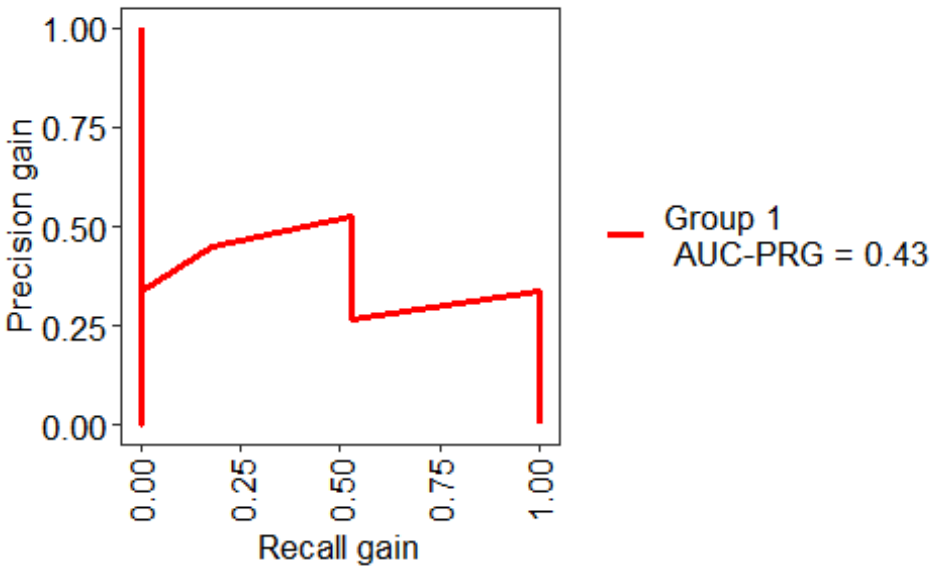
```
## [1] normal  serious serious
## Levels: normal serious

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
##

# Completely accurate

#set uneLength or tuneGrid for better model performance

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

rf_fit <- train(as.factor(loss_degree) ~., #optimize mtry with tuneLeng
th
```

```
                data = training,
                method = "rf",
                tuneLength = 5,
                trControl = 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
##
```

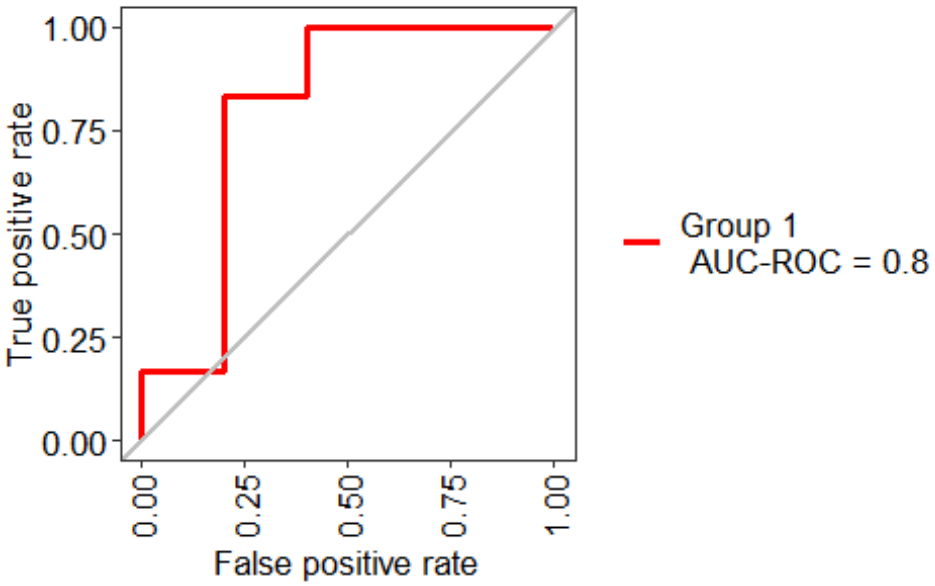
```r
library(MLeval)

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
True probability in each bin
0.00
0.25
0.50
0.75
1.00
Predicted probability
Group 1

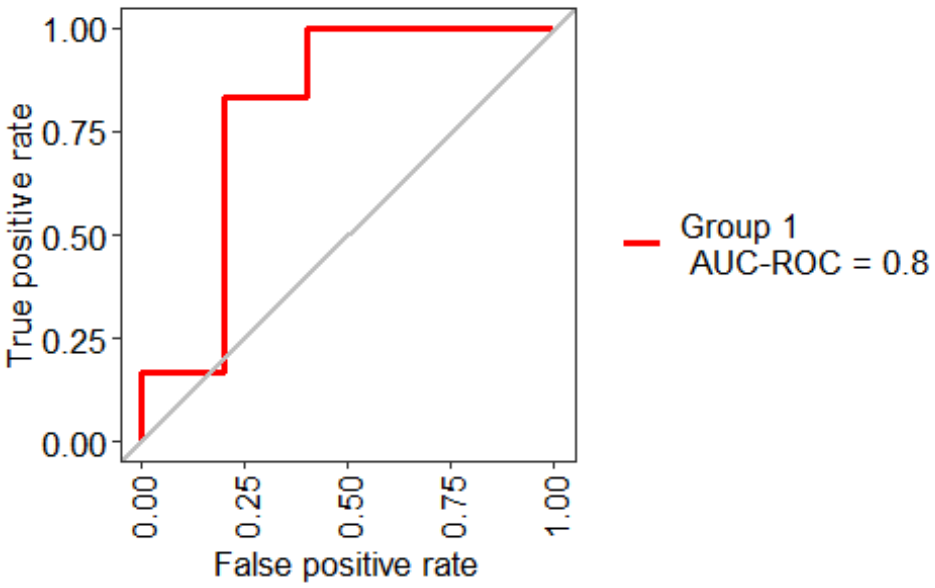

1.00
0.75
0.50
0.25
0.00
Precision
0.25
0.50
0.75
1.00
Recall(sensitivity)
Group 1
AUC-PR = 0.6




```
## Group 1 Optimal Informedness = 0.633333333333333

## Group 1 AUC-ROC = 0.8
```

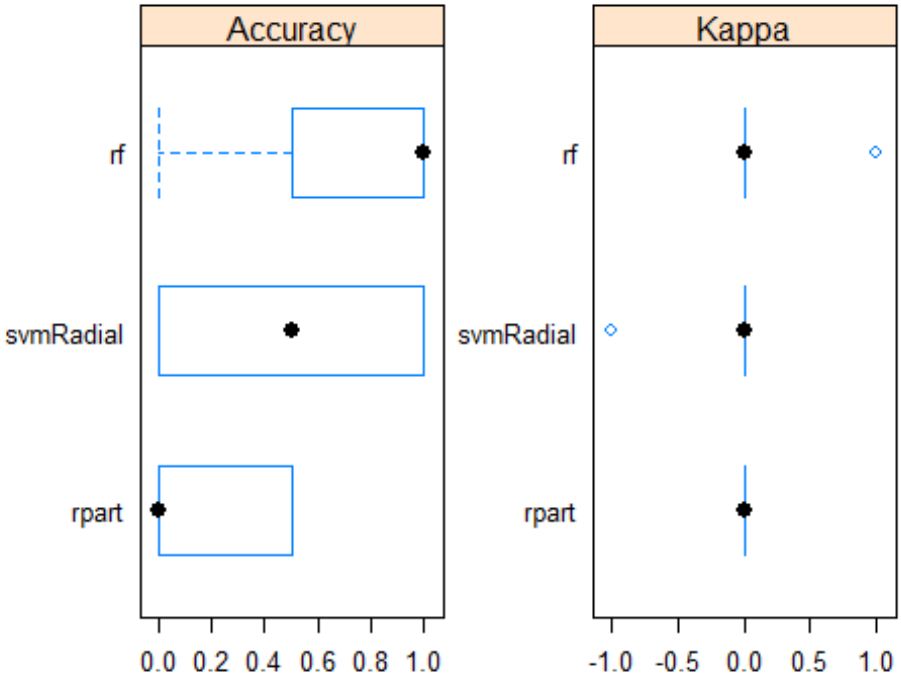


```
x$roc
```




```
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        0.800 0.53-1.07
## AUC-PR         0.600      <NA>
## AUC-PRG        0.430      <NA>

##================================================
# compare algorithms
library(caretEnsemble)
```

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

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

# Stacking Algorithms - Run multiple algos in one call.
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=ctrl,
                    methodList=algorithmList)

# show results
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     0.5    1.0 0.64     1.0  1.0    0
## rpart        0     0.0    0.0 0.16     0.5  0.5    0
## svmRadial    0     0.0    0.5 0.50     1.0  1.0    0
##
## Kappa
##           Min. 1st Qu. Median       Mean 3rd Qu. Max. NA's
## rf           0       0      0  0.1428571       0    1   11
## rpart        0       0      0  0.0000000       0    0    0
## svmRadial   -1       0      0 -0.0625000       0    0    9

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


Accuracy
Kappa
rf
svmRadial
rpart
0.0 0.2 0.4 0.6 0.8 1.0
-1.0 -0.5 0.0 0.5 1.0