exercise 4：基于树模型的特征选择

YangMenglei（杨梦磊 SA20008161）

2021/4/09

Introduction

> 在模型构建过程中，特征选择（Feature Selection）又称为属性筛选，通过在数据集中选取有意义的特征，输入到算法模型中进行训练，特征选择主要从两个方面进行考虑:

- 特征是否具有发散性：可通过方差来衡量；方差越小，说明对目标变化的贡献度越小
- 特征与目标的相关性，如两个特征的相关性很大，其中一者往往可以用来替代另一者以此来减少特征量

特征选择可以提高算法运行效率。 去除不相关的特征往往会降低学习任务的难度，使模型更易理解，比如，使决策树的规则变得更加清晰，去除不相关的变量还可以尽量减少过拟合的风险。 文献中采用基于SVM和ANN的分类模型来预测柑橘是有机种植还是传统化肥培育，并通过Feature selection filter methods来确定最具有代表性意义的特征组合。 基于此，下面的代码采用了rattle包进行基本的图形绘制，如数据分布，特征相关性，特征重要性，主要目的是探索影响糖尿病的特征因素及建立预测模型。

Methods

> 常见的特征选择方法:

1. 过滤式（Filter)：对数据集进行特征选择，然后再训练模型，特征选择过程与后续模型训练无关。常用的特征子集评价标准包括相关系数、互信息、信息增益
2. 包裹式（wrapper）：直接把最终要使用的模型的性能作为特征子集的评价标准，性能好，计算开销大
3. 嵌入式（embedding)：将特征选择过程与学习过程融为一体，在同一个优化过程中完成。

> 这里参考文献的Filter Method进行学习建模，思路为:

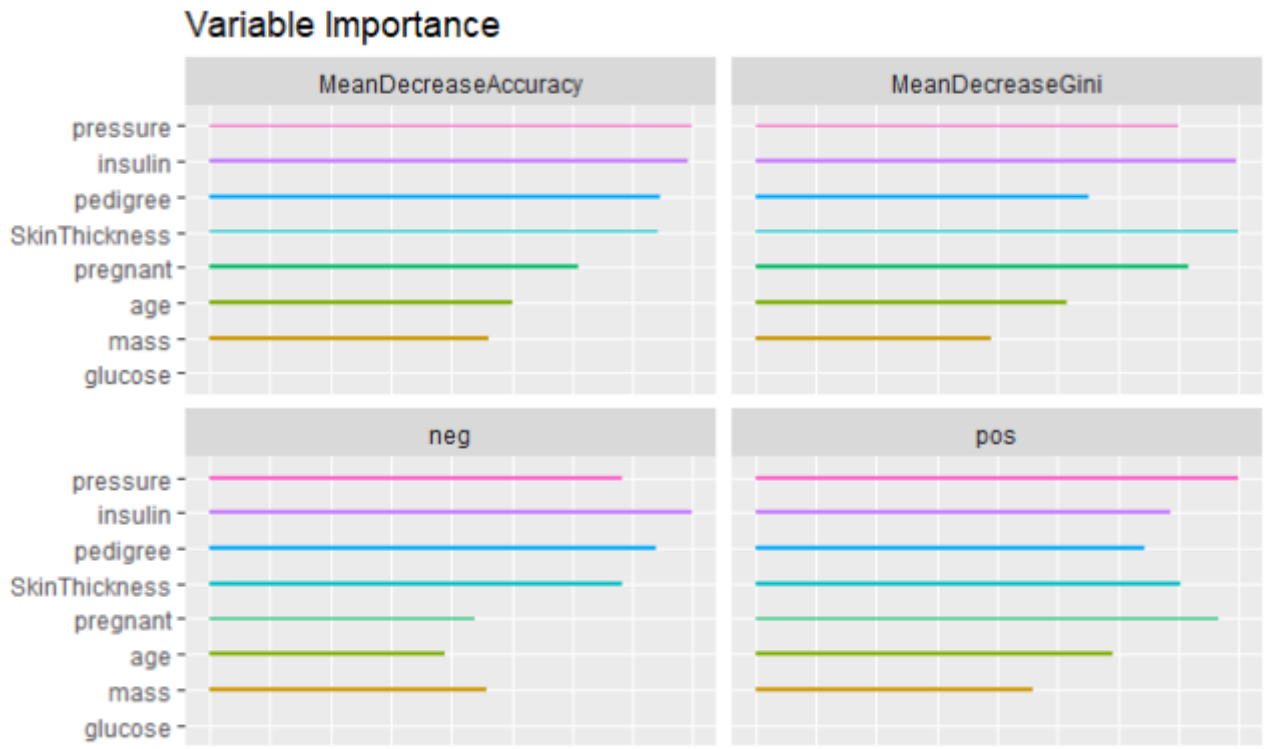
1. 可视化数据集，及相关系数、方差、特征重要性，如果方差比较低，则说明对目标变量的贡献值越小
2. 移除冗余特征，以及高度关联的特征（其他特征可以反映该特征）
3. 构建模型获取特征重要性，利用ROC曲线分析获取，或决策树特征重要性获取

特征选择：递归特征消除（Recursive Feature Elimination）构建不同子集的许多模型，识别哪些特征有助于构建准确模型，同时采用基于树结构的特征选择

其中随机森林算法用于每一轮迭代中评估模型的方法，支持向量机SVM用于评估特征组合的准确度:

Main Results:

#变量（特征)重要性



#特征重要性排序(caret包)

Variable	Importance
glucose	0.7881306
mass	0.6875672
age	0.6969403
pregnant	0.6196149
pedigree	0.6062015
pressure	0.5864590
triceps	0.5536269
insulin	0.5378619

#模型评估

Model ID	Varible subset	Ac curacy(%)
#1	glucose	69.30
#2	glucose mass	71.35
#3	glucose mass age	73.96
#4	glucose mass age pregnant	75.40
#5	glucose mass age pregnant insulin	75,90
#6	glucose mass age pregnant insulin pedigree	72.65
#7	glucose mass age pregnant insulin pedigree triceps	74.00
#8	glucose mass age pregnant insulin pedigree triceps pressure	78.35

调用 rattle 进行数据可视化

```
library(RGtk2)
library(tibble)
library(bitops)
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## XXXX 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## 键入'rattle()'去轻摇、晃动、翻滚你的数据。
```

```
rattle()
```

数据导入及划分训练集、验证集、测试集

```
# Build the train/validate/test datasets.
# nobs=768 train=538 validate=115 test=115
library(rattle)   # Access the weather dataset and utilities
library(magrittr) # Utilise %>% and %<>% pipeline operators.
building <- TRUE
scoring  <- ! building
crv$seed <- 42
# Load a dataset from file.

fname         <- "file:///D:/RStudio/project/initial exercise/predictdiabetes.csv"
crs$dataset <- read.csv(fname,
            na.strings=c(".", "NA", "", "?"),
            strip.white=TRUE, encoding="UTF-8")

#=====================================================================
# Build the train/validate/test datasets.

# nobs=768 train=538 validate=115 test=115

set.seed(crv$seed)

crs$nobs <- nrow(crs$dataset)

crs$train <- sample(crs$nobs, 0.7*crs$nobs)

crs$nobs %>%
  seq_len() %>%
  setdiff(crs$train) %>%
  sample(0.15*crs$nobs) ->
crs$validate

crs$nobs %>%
  seq_len() %>%
  setdiff(crs$train) %>%
  setdiff(crs$validate) ->
crs$test

# The following variable selections have been noted.

crs$input     <- c("pregnant", "glucose", "pressure",
                 "SkinThickness", "insulin", "mass", "pedigree",
                 "age")

crs$numeric   <- c("pregnant", "glucose", "pressure",
                 "SkinThickness", "insulin", "mass", "pedigree",
                 "age")

crs$categoric <- NULL

crs$target    <- "diabetes"
crs$risk      <- NULL
crs$ident     <- NULL
crs$ignore    <- NULL
crs$weights   <- NULL
```

随机森林建模评估变量重要性

#根据随机森林度量变量重要性方法

```
# Build a Random Forest model using the traditional approach.
set.seed(crv$seed)

crs$rf <- randomForest::randomForest(as.factor(diabetes) ~. ,
  data=crs$dataset[crs$train, c(crs$input, crs$target)],
  ntree=500,
  mtry=2,
  importance=TRUE,
  na.action=randomForest::na.roughfix,
  replace=FALSE)

# Generate textual output of the 'Random Forest' model.

crs$rf
```

```
##
## Call:
##  randomForest(formula = as.factor(diabetes) ~ ., data = crs$dataset[crs$train,      c(crs$in
put, crs$target)], ntree = 500, mtry = 2, importance = TRUE,      replace = FALSE, na.action =
randomForest::na.roughfix)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 24.21%
## Confusion matrix:
##     neg pos class.error
## neg 289  55   0.1598837
## pos  75 118   0.3886010
```

```
# The `pROC' package implements various AUC functions.

# Calculate the Area Under the Curve (AUC).

pROC::roc(crs$rf$y, as.numeric(crs$rf$predicted))
```

```
## Setting levels: control = neg, case = pos
```

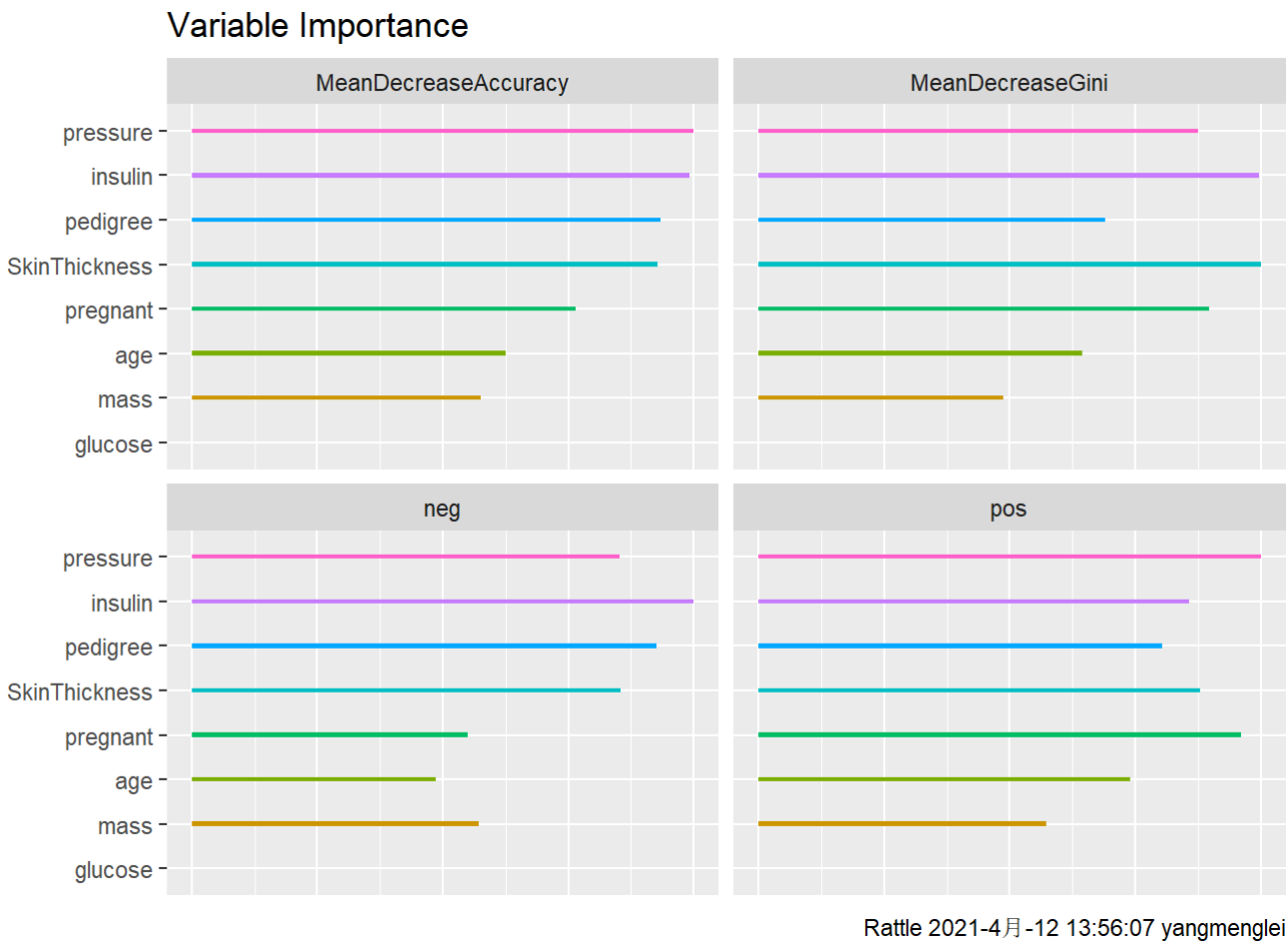
```
## Setting direction: controls < cases
```

```
##
## Call:
## roc.default(response = crs$rf$y, predictor = as.numeric(crs$rf$predicted))
##
## Data: as.numeric(crs$rf$predicted) in 344 controls (crs$rf$y neg) < 193 cases (crs$rf$y po
s).
## Area under the curve: 0.7258
```

```
# List the importance of the variables.
rn <- round(randomForest::importance(crs$rf), 2)
rn[order(rn[,3], decreasing=TRUE),]
```

```
##                  neg   pos MeanDecreaseAccuracy MeanDecreaseGini
## glucose        28.39 31.52                38.83            38.84
## mass           13.22 13.58                18.96            25.28
## age            15.49  8.33                17.24            20.90
## pregnant       13.80  1.43                12.51            13.83
## SkinThickness   5.73  3.97                 6.87            10.98
## pedigree        3.84  6.31                 6.64            19.64
## insulin         1.86  4.63                 4.63            11.06
## pressure        5.76  0.18                 4.36            14.45
```

```
# Plot the relative importance of the variables.

p <- ggVarImp(crs$rf,
              title="Variable Importance Random Forest predictdiabetes.csv")
p
```



```
#构建支持向量机模型

library(kernlab, quietly=TRUE)

# Build a Support Vector Machine model.

set.seed(crv$seed)
crs$ksvm <- ksvm(as.factor(diabetes) ~ .,
      data=crs$dataset[crs$train,c(crs$input, crs$target)],
      kernel="rbfdot",
      prob.model=TRUE)

# Generate a textual view of the SVM model.

crs$ksvm
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.134649827397974
##
## Number of Support Vectors : 319
##
## Objective Function Value : -249.3915
## Training error : 0.158287
## Probability model included.
```

```
# Time taken: 0.06 secs
```

生成特征之间的相关性

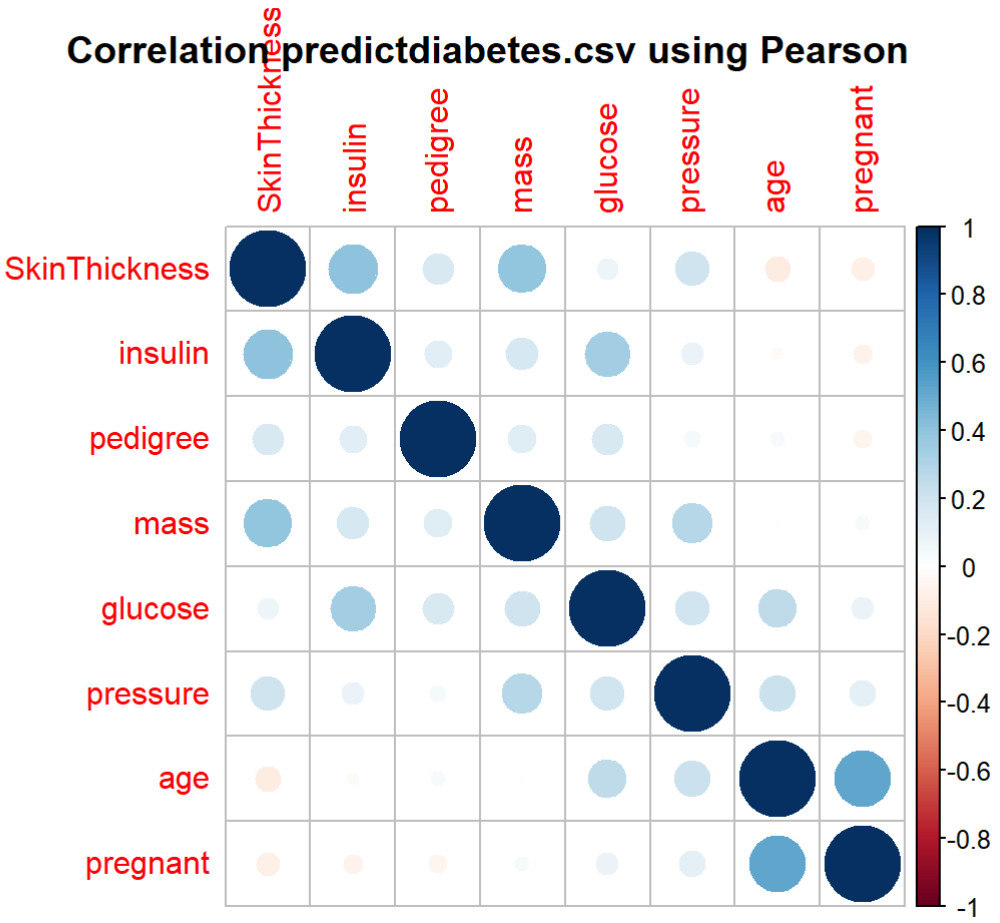
```
# Generate a correlation plot for the variables.
library(corrplot, quietly=TRUE)
```

```
## corrplot 0.84 loaded
```

```
# Correlations work for numeric variables only.

crs$cor <- cor(crs$dataset[crs$train, crs$numeric], use="pairwise", method="pearson")

# Order the correlations by their strength.

crs$ord <- order(crs$cor[1,])
crs$cor <- crs$cor[crs$ord, crs$ord]

# Display the actual correlations.

print(crs$cor)
```

```
##               SkinThickness     insulin    pedigree        mass    glucose
## SkinThickness    1.00000000  0.40581770  0.16158240 0.392335757 0.07685683
## insulin          0.40581770  1.00000000  0.12321381 0.179588822 0.34371918
## pedigree         0.16158240  0.12321381  1.00000000 0.136582186 0.16050021
## mass             0.39233576  0.17958882  0.13658219 1.000000000 0.20804835
## glucose          0.07685683  0.34371918  0.16050021 0.208048352 1.00000000
## pressure         0.20391675  0.08558097  0.04189904 0.280771979 0.19322958
## age             -0.10544153 -0.02460145  0.03338416 0.009659328 0.25153588
## pregnant        -0.08885364 -0.06375357 -0.05698262 0.034955256 0.08352818
##                  pressure          age    pregnant
## SkinThickness 0.20391675 -0.105441531 -0.08885364
## insulin       0.08558097 -0.024601447 -0.06375357
## pedigree      0.04189904  0.033384155 -0.05698262
## mass          0.28077198  0.009659328  0.03495526
## glucose       0.19322958  0.251535884  0.08352818
## pressure      1.00000000  0.219031025  0.11534378
## age           0.21903102  1.000000000  0.52906116
## pregnant      0.11534378  0.529061158  1.00000000
```

```
# Graphically display the correlations.

corrplot(crs$cor, mar=c(0,0,1,0))
title(main="Correlation predictdiabetes.csv using Pearson",
    sub=)
```



#主成分分析验证

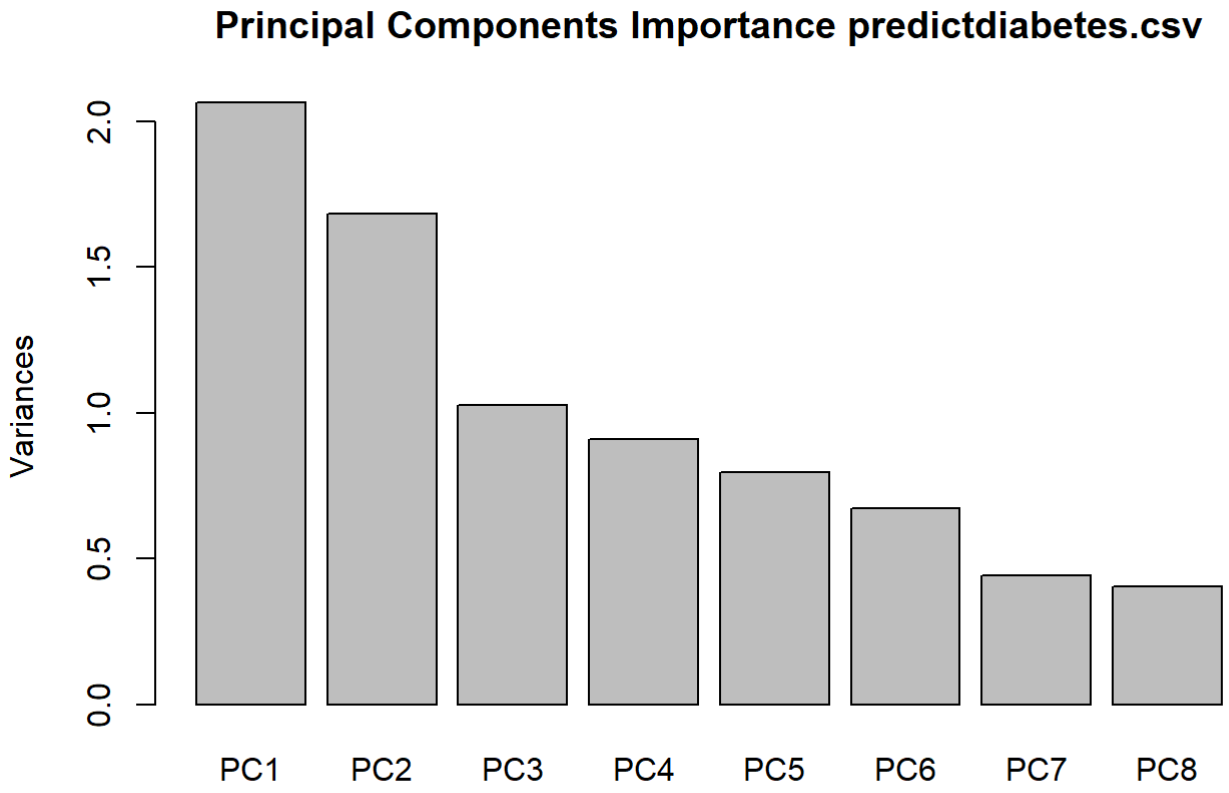
```
# Principal Components Analysis (on numerics only).
pc <- prcomp(na.omit(crs$dataset[crs$train, crs$numeric]), scale=TRUE, center=TRUE, tol=0)

# Show the output of the analysis.
pc
```

```
## Standard deviations (1, .., p=8):
## [1] 1.4369484 1.2971847 1.0131345 0.9538638 0.8927472 0.8200588 0.6659375
## [8] 0.6350014
##
## Rotation (n x k) = (8 x 8):
##                       PC1        PC2         PC3         PC4         PC5
## pregnant      -0.1034377  0.6066566 -0.08604011  0.02477517  0.48851644
## glucose       -0.4150702  0.1546956  0.47885083  0.22412195 -0.45826688
## pressure      -0.3722607  0.1767848 -0.46540181 -0.09430320 -0.53756553
## SkinThickness -0.4415946 -0.3220747 -0.25327494  0.04981098  0.43845384
## insulin       -0.4270006 -0.2180178  0.33837568  0.48576470  0.21783487
## mass          -0.4513410 -0.1043274 -0.39773385 -0.17362379  0.01657073
## pedigree      -0.2517700 -0.1088333  0.44828243 -0.81904470  0.09968193
## age           -0.1844105  0.6346584  0.09436356 -0.02888827  0.11163144
##                       PC6        PC7          PC8
## pregnant       0.07310465 -0.4395532  0.419585609
## glucose        0.31990471  0.1788687  0.424164642
## pressure      -0.51450089 -0.2208885  0.046878315
## SkinThickness -0.21583561  0.5120744  0.365493883
## insulin       -0.25008140 -0.4040431 -0.382661430
## mass           0.69396975 -0.1336809 -0.311978323
## pedigree      -0.15559719 -0.1371873  0.002911614
## age           -0.11258471  0.5136588 -0.514276383
```

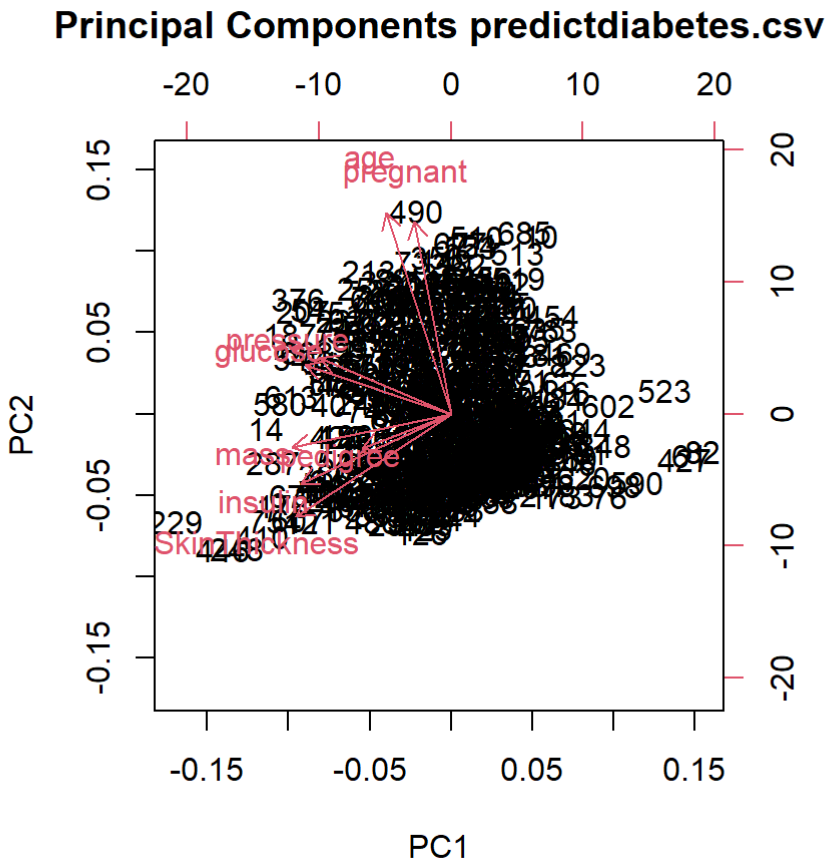
```
# Summarise the importance of the components found.
summary(pc)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5     PC6     PC7
## Standard deviation     1.4369 1.2972 1.0131 0.9539 0.89275 0.82006 0.66594
## Proportion of Variance 0.2581 0.2103 0.1283 0.1137 0.09962 0.08406 0.05543
## Cumulative Proportion  0.2581 0.4684 0.5967 0.7105 0.81010 0.89416 0.94960
##                           PC8
## Standard deviation     0.6350
## Proportion of Variance 0.0504
## Cumulative Proportion  1.0000
```

```
# Display a plot showing the relative importance of the components.
plot(pc, main="")
title(main="Principal Components Importance predictdiabetes.csv",
    sub=)
axis(1, at=seq(0.7, ncol(pc$rotation)*1.2, 1.2), labels=colnames(pc$rotation), lty=0)
```




```
# Display a plot showing the two most principal components.
biplot(pc, main="")
title(main="Principal Components predictdiabetes.csv",
    sub=)
```



caret包评估变量相关性

```
set.seed(1234)
library(mlbench)
library(caret)
```

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

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

```
##
## Attaching package: 'ggplot2'
```

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

```
library(readr)
predictdiabetes <- read_csv("predictdiabetes.csv",
    col_types = cols(diabetes = col_factor(levels = c("pos",
        "neg"))))
data(predictdiabetes)
```

```
## Warning in data(predictdiabetes): data set 'predictdiabetes' not found
```

```
Matrix <- predictdiabetes[,1:8]

library(Hmisc)
```

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

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

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

```
## Loading required package: Formula
```

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

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

```
up_CorMatrix <- function(cor,p) {ut <- upper.tri(cor)
data.frame(row = rownames(cor)[row(cor)[ut]] ,
           column = rownames(cor)[col(cor)[ut]],
           cor =(cor)[ut] ) }

res <- rcorr(as.matrix(Matrix))
cor_data <- up_CorMatrix (res$r)
cor_data <- subset(cor_data, cor_data$cor > 0.38) #关联度大于0.38认为一者可以代表其余变量
cor_data
```

```
##              row  column       cor
## 10 SkinThickness insulin 0.4367826
## 14 SkinThickness    mass 0.3925732
## 22      pregnant     age 0.5443412
```

##如果两个变量的关联程度很高，那么其中一种往往能代表其余的一种或多种，可以移除掉高度关联的特征，如上图结果所示，pregnant和age 以及skinThickness和insulin和BMI

caret包生成特征重要性排序

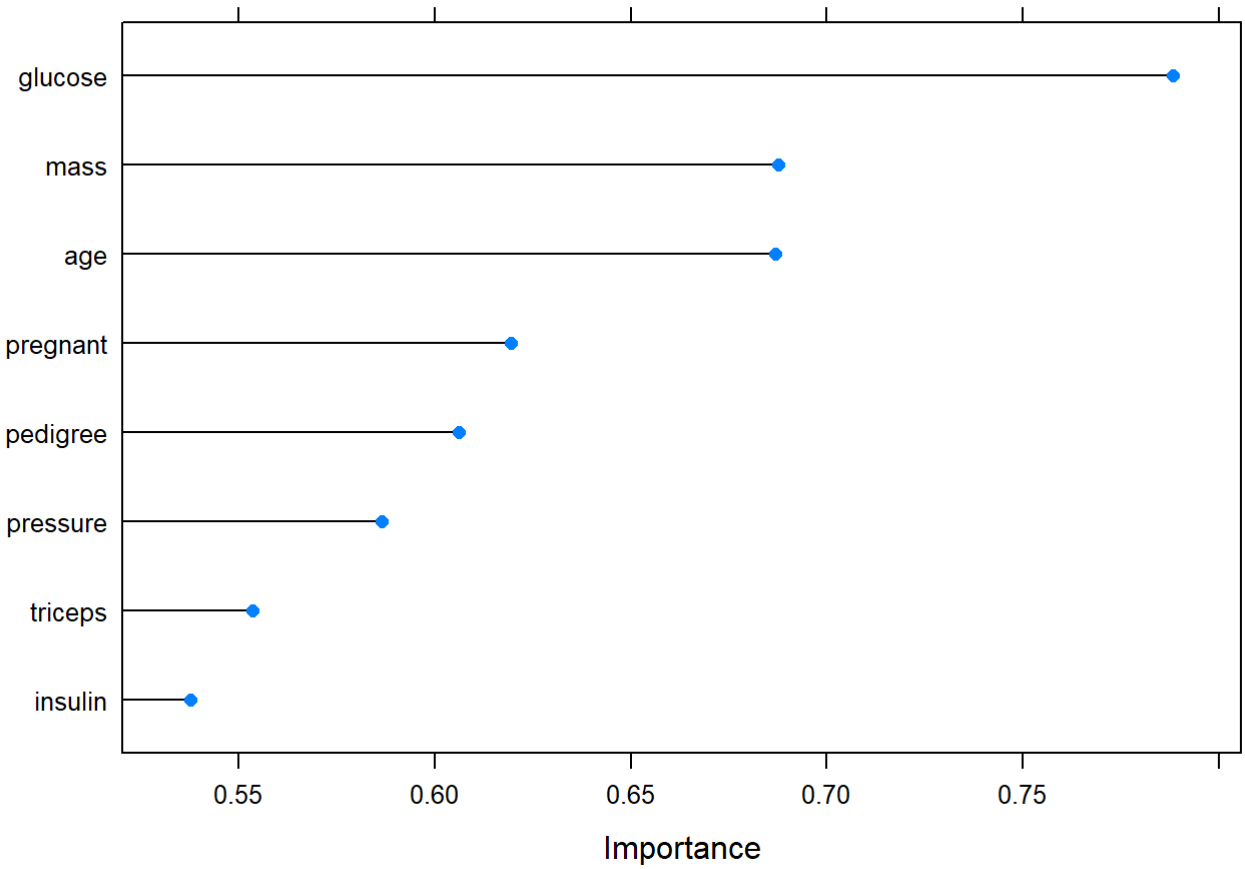
```
# ensure results are repeatable
set.seed(1234)
# load the library
library(mlbench)
library(caret)
# load the dataset
data(predictdiabetes)
```

```
## Warning in data(predictdiabetes): data set 'predictdiabetes' not found
```

```
data(PimaIndiansDiabetes)
# prepare training scheme
control <- trainControl(method="repeatedcv", number=10, repeats=3)

# train the model
model <- train(diabetes~., data=PimaIndiansDiabetes, method="lvq", preProcess="scale", trContro
l=control)

# estimate variable importance
importance <- varImp(model, scale=FALSE)

# summarize importance
print(importance)
```

```
## ROC curve variable importance
##
##          Importance
## glucose      0.7881
## mass         0.6876
## age          0.6869
## pregnant     0.6195
## pedigree     0.6062
## pressure     0.5865
## triceps      0.5536
## insulin      0.5379
```

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



caret包 递归交叉检验评估准确度（accuracy）

```
# ensure the results are repeatable
set.seed(7)
# load the library
library(mlbench)
library(caret)
# load the data
library(readr)
data(predictdiabetes)
```

```
## Warning in data(predictdiabetes): data set 'predictdiabetes' not found
```

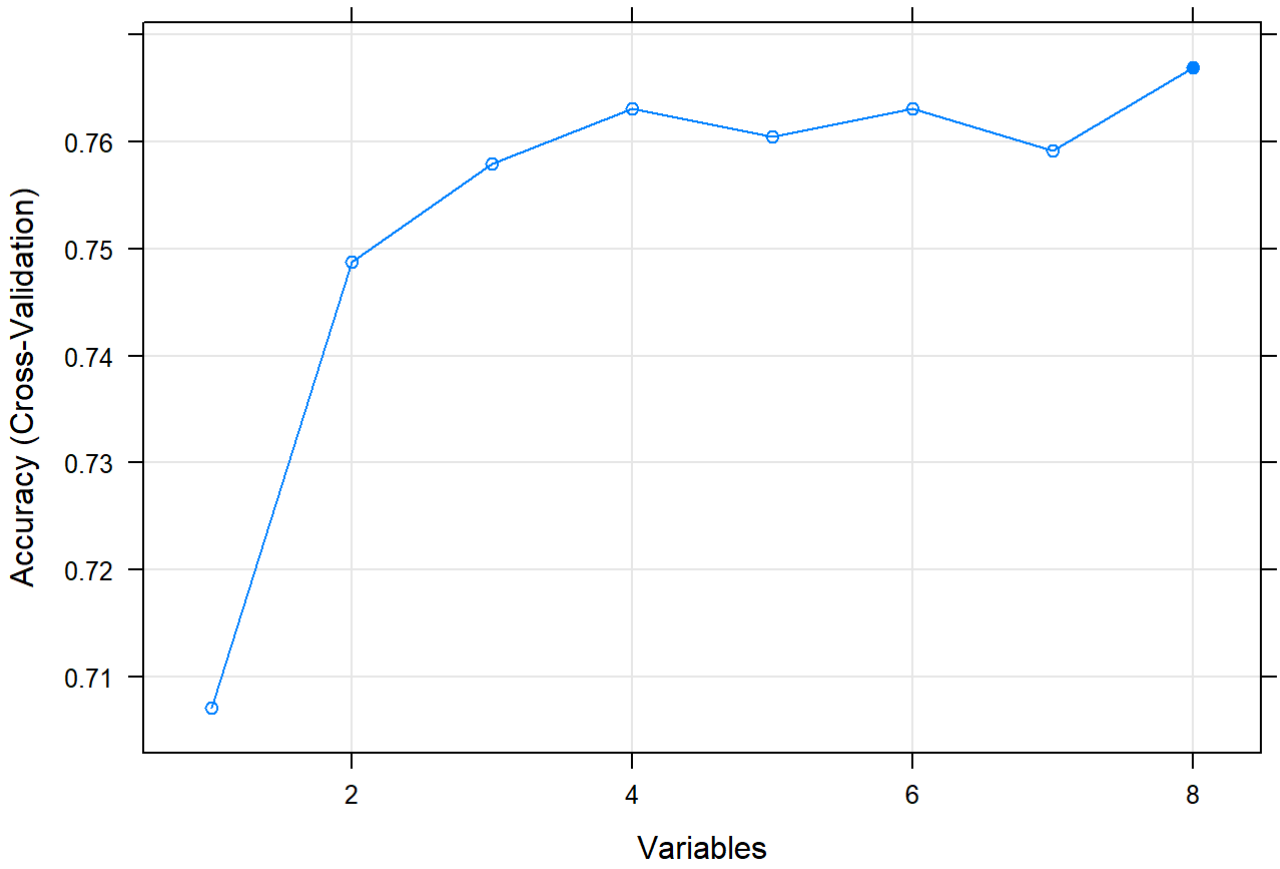
```
# define the control using a random forest selection function
control <- rfeControl(functions=rfFuncs, method="cv", number=10)
# run the RFE algorithm
results <- rfe(PimaIndiansDiabetes[,1:8], PimaIndiansDiabetes[,9], sizes=c(1:8),  rfeControl=co
ntrol)
# summarize the results
print(results)
```

```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (10 fold)
##
## Resampling performance over subset size:
##
##  Variables Accuracy  Kappa AccuracySD KappaSD Selected
##          1   0.7071 0.2957    0.03235 0.09669
##          2   0.7487 0.4263    0.03914 0.08962
##          3   0.7579 0.4539    0.03966 0.08767
##          4   0.7631 0.4699    0.05373 0.11706
##          5   0.7605 0.4656    0.04037 0.07910
##          6   0.7631 0.4660    0.03316 0.06588
##          7   0.7591 0.4579    0.03739 0.06966
##          8   0.7670 0.4696    0.03926 0.08056        *
##
## The top 5 variables (out of 8):
##    glucose, mass, age, pregnant, insulin
```

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

```
## [1] "glucose"  "mass"     "age"      "pregnant" "insulin"  "pedigree" "triceps"
## [8] "pressure"
```

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




```
## The top 5 variables (out of 8):
# glucose, mass, age, pregnant, insulin
```

#自动特征选择用于构建不同子集的许多模型，识别哪些特征有助于构建准确模型，哪些特征没什么帮助。特征选择的一个流行的自动方法称为 递归特征消除（Recursive Feature Elimination）或RFE。 。随机森林算法用于每一轮迭代中评估模型的方法。该算法用于探索所有可能的特征子集。从图中可以看出当使用5个特征时即可获取与最高性能相差无几的结果。

利用SVM建模对筛选的特征组合进行评估

Model ID	Varible subset	Accuracy(%)
#1	glucose	69.30
#2	glucose mass	71.35
#3	glucose mass age	73.96
#4	glucose mass age pregnant	75.40
#5	glucose mass age pregnant insulin	75,90
#6	glucose mass age pregnant insulin pedigree	72.65
#7	glucose mass age pregnant insulin pedigree triceps	74.00
#8	glucose mass age pregnant insulin pedigree triceps pressure	78.35

```
#导入数据
# nobs=768 train=538 validate=115 test=115
# Load a dataset from file.

fname         <- "file:///D:/RStudio/project/initial exercise/predictdiabetes.csv"
crs$dataset <- read.csv(fname,
            na.strings=c(".", "NA", "", "?"),
            strip.white=TRUE, encoding="UTF-8")

#=====================================================================
# Action the user selections from the Data tab.
# Build the train/validate/test datasets.
# nobs=768 train=538 validate=115 test=115

set.seed(crv$seed)

crs$nobs <- nrow(crs$dataset)

crs$train <- sample(crs$nobs, 0.7*crs$nobs)

crs$nobs %>%
  seq_len() %>%
  setdiff(crs$train) %>%
  sample(0.15*crs$nobs) ->
crs$validate

crs$nobs %>%
  seq_len() %>%
  setdiff(crs$train) %>%
  setdiff(crs$validate) ->
crs$test

# The following variable selections have been noted.

crs$input     <- c("Pregnancies", "Glucose", "BloodPressure",
                   "SkinThickness", "Insulin", "BMI",
                   "DiabetesPedigreeFunction", "Age")

crs$numeric   <- c("Pregnancies", "Glucose", "BloodPressure",
                   "SkinThickness", "Insulin", "BMI",
                   "DiabetesPedigreeFunction", "Age")

crs$categoric <- NULL

crs$target    <- "Outcome"
crs$risk      <- NULL
crs$ident     <- NULL
crs$ignore    <- NULL
crs$weights   <- NULL
```

```
library(rattle)   # Access the weather dataset and utilities.
library(magrittr) # Utilise %>% and %<>% pipeline operators.

building <- TRUE
scoring  <- ! building

crv$seed <- 42
# Load a dataset from file.

fname         <- "file:///D:/RStudio/project/initial exercise/predictdiabetes.csv"
crs$dataset <- read.csv(fname,
            na.strings=c(".", "NA", "", "?"),
            strip.white=TRUE, encoding="UTF-8")


# Build the train/validate/test datasets.

# nobs=768 train=538 validate=115 test=115

set.seed(crv$seed)

crs$nobs <- nrow(crs$dataset)

crs$train <- sample(crs$nobs, 0.7*crs$nobs)

crs$nobs %>%
  seq_len() %>%
  setdiff(crs$train) %>%
  sample(0.15*crs$nobs) ->
crs$validate

crs$nobs %>%
  seq_len() %>%
  setdiff(crs$train) %>%
  setdiff(crs$validate) ->
crs$test

# The following variable selections have been noted.

crs$input     <- c("pregnant", "glucose", "pressure",
                 "SkinThickness", "insulin", "mass", "pedigree",
                 "age")

crs$numeric   <- c("pregnant", "glucose", "pressure",
                 "SkinThickness", "insulin", "mass", "pedigree",
                 "age")

crs$categoric <- NULL

crs$target    <- "diabetes"
crs$risk      <- NULL
crs$ident     <- NULL
crs$ignore    <- NULL
crs$weights   <- NULL

# nobs=768 train=538 validate=115 test=115
```

决策树模型

```
# Decision Tree

# The 'rpart' package provides the 'rpart' function.

library(rpart, quietly=TRUE)

# Reset the random number seed to obtain the same results each time.

set.seed(crv$seed)

# Build the Decision Tree model.

crs$rpart <- rpart(diabetes ~ .,
    data=crs$dataset[crs$train, c(crs$input, crs$target)],
    method="class",
    parms=list(split="information"),
    control=rpart.control(usesurrogate=0,
        maxsurrogate=0),
    model=TRUE)

# Generate a textual view of the Decision Tree model.

print(crs$rpart)
```

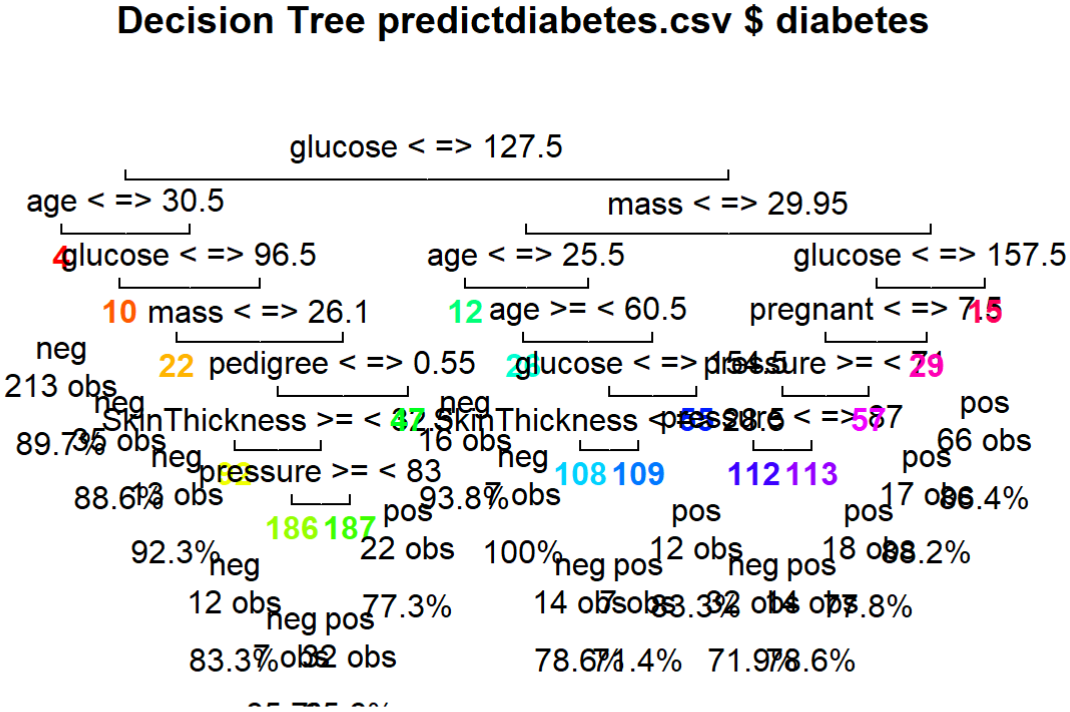
```
## n= 537
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 537 193 neg (0.64059590 0.35940410)
##     2) glucose< 127.5 334  68 neg (0.79640719 0.20359281)
##       4) age< 30.5 213  22 neg (0.89671362 0.10328638) *
##       5) age>=30.5 121  46 neg (0.61983471 0.38016529)
##        10) glucose< 96.5 35   4 neg (0.88571429 0.11428571) *
##        11) glucose>=96.5 86  42 neg (0.51162791 0.48837209)
##          22) mass< 26.1 13   1 neg (0.92307692 0.07692308) *
##          23) mass>=26.1 73  32 pos (0.43835616 0.56164384)
##            46) pedigree< 0.5485 51  24 neg (0.52941176 0.47058824)
##              92) SkinThickness>=32.5 12   2 neg (0.83333333 0.16666667) *
##              93) SkinThickness< 32.5 39  17 pos (0.43589744 0.56410256)
##               186) pressure>=83 7   1 neg (0.85714286 0.14285714) *
##               187) pressure< 83 32  11 pos (0.34375000 0.65625000) *
##            47) pedigree>=0.5485 22   5 pos (0.22727273 0.77272727) *
##     3) glucose>=127.5 203  78 pos (0.38423645 0.61576355)
##       6) mass< 29.95 56  19 neg (0.66071429 0.33928571)
##        12) age< 25.5 16   1 neg (0.93750000 0.06250000) *
##        13) age>=25.5 40  18 neg (0.55000000 0.45000000)
##          26) age>=60.5 7   0 neg (1.00000000 0.00000000) *
##          27) age< 60.5 33  15 pos (0.45454545 0.54545455)
##            54) glucose< 154.5 21   8 neg (0.61904762 0.38095238)
##             108) SkinThickness< 28.5 14   3 neg (0.78571429 0.21428571) *
##             109) SkinThickness>=28.5 7   2 pos (0.28571429 0.71428571) *
##            55) glucose>=154.5 12   2 pos (0.16666667 0.83333333) *
##       7) mass>=29.95 147  41 pos (0.27891156 0.72108844)
##        14) glucose< 157.5 81  32 pos (0.39506173 0.60493827)
##          28) pregnant< 7.5 64  30 pos (0.46875000 0.53125000)
##            56) pressure>=71 46  20 neg (0.56521739 0.43478261)
##             112) pressure< 87 32   9 neg (0.71875000 0.28125000) *
##             113) pressure>=87 14   3 pos (0.21428571 0.78571429) *
##            57) pressure< 71 18   4 pos (0.22222222 0.77777778) *
##          29) pregnant>=7.5 17   2 pos (0.11764706 0.88235294) *
##        15) glucose>=157.5 66   9 pos (0.13636364 0.86363636) *
```

```
printcp(crs$rpart)
```

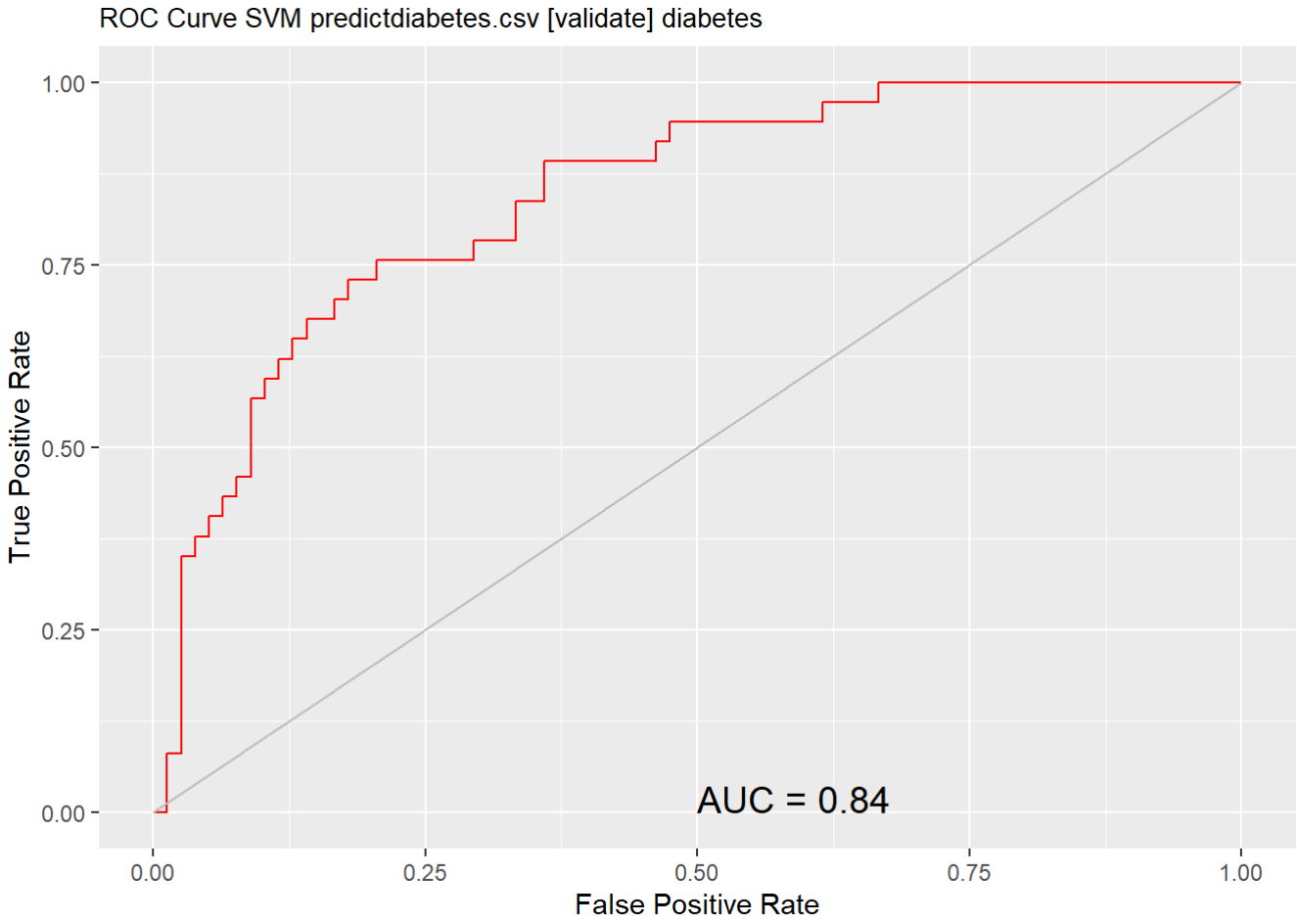
```
##
## Classification tree:
## rpart(formula = diabetes ~ ., data = crs$dataset[crs$train, c(crs$input,
##     crs$target)], method = "class", model = TRUE, parms = list(split = "information"),
##     control = rpart.control(usesurrogate = 0, maxsurrogate = 0))
##
## Variables actually used in tree construction:
## [1] age           glucose       mass          pedigree      pregnant
## [6] pressure      SkinThickness
##
## Root node error: 193/537 = 0.3594
##
## n= 537
##
##         CP nsplit rel error  xerror     xstd
## 1 0.243523      0   1.00000 1.00000 0.057612
## 2 0.093264      1   0.75648 0.83938 0.055110
## 3 0.018135      2   0.66321 0.78756 0.054089
## 4 0.015544      6   0.59067 0.74093 0.053073
## 5 0.013817     12   0.47668 0.74611 0.053190
## 6 0.010000     16   0.41969 0.72539 0.052713
```

```
cat("\n")
```

```
drawTreeNodes(crs$rpart)
title(main="Decision Tree predictdiabetes.csv $ diabetes",
    sub=)
```



```
# Evaluate model performance on the validation dataset.

library(ROCR)

library(ggplot2, quietly=TRUE)

crs$pr <- kernlab::predict(crs$ksvm, newdata=na.omit(crs$dataset[crs$validate, c(crs$input, crs
$target)]),
    type    = "probabilities")[,2]

# Remove observations with missing target.

no.miss   <- na.omit(na.omit(crs$dataset[crs$validate, c(crs$input, crs$target)])$diabetes)
miss.list <- attr(no.miss, "na.action")
attributes(no.miss) <- NULL

if (length(miss.list))
{
  pred <- prediction(crs$pr[-miss.list], no.miss)
} else
{
  pred <- prediction(crs$pr, no.miss)
}

pe <- performance(pred, "tpr", "fpr")
au <- performance(pred, "auc")@y.values[[1]]
pd <- data.frame(fpr=unlist(pe@x.values), tpr=unlist(pe@y.values))
p <- ggplot(pd, aes(x=fpr, y=tpr))
p <- p + geom_line(colour="red")
p <- p + xlab("False Positive Rate") + ylab("True Positive Rate")
p <- p + ggtitle("ROC Curve SVM predictdiabetes.csv [validate] diabetes")
p <- p + theme(plot.title=element_text(size=10))
p <- p + geom_line(data=data.frame(), aes(x=c(0,1), y=c(0,1)), colour="grey")
p <- p + annotate("text", x=0.50, y=0.00, hjust=0, vjust=0, size=5,
                  label=paste("AUC =", round(au, 2)))
print(p)
```




```
# Calculate the area under the curve for the plot.
#Remove observations with missing target.
no.miss   <- na.omit(na.omit(crs$dataset[crs$validate, c(crs$input, crs$target)])$diabetes)
miss.list <- attr(no.miss, "na.action")
attributes(no.miss) <- NULL

if (length(miss.list))
{
  pred <- prediction(crs$pr[-miss.list], no.miss)
} else
{
  pred <- prediction(crs$pr, no.miss)
}
performance(pred, "auc")
```

```
## A performance instance
##   'Area under the ROC curve'
```

```
#Area under the ROC curve for the ksvm model on predictdiabetes.csv [validate] is 0.8427
```

```
# Obtain the response from the Decision Tree model.
crs$pr <- predict(crs$rpart, newdata=crs$dataset[crs$validate, c(crs$input, crs$target)],
    type="class")
# Generate the confusion matrix showing counts.
rattle::errorMatrix(crs$dataset[crs$validate, c(crs$input, crs$target)]$diabetes, crs$pr, count
=TRUE)
```

```
##       Predicted
## Actual neg pos Error
##    neg  62  16  20.5
##    pos  13  24  35.1
```

```
# Generate the confusion matrix showing proportions.
(per <- rattle::errorMatrix(crs$dataset[crs$validate, c(crs$input, crs$target)]$diabetes, crs$p
r))
```

```
##       Predicted
## Actual  neg  pos Error
##    neg 53.9 13.9  20.5
##    pos 11.3 20.9  35.1
```

```
# Calculate the overall error percentage.
cat(100-sum(diag(per), na.rm=TRUE))
```

```
## 25.2
```

```
# Calculate the averaged class error percentage.
cat(mean(per[,"Error"], na.rm=TRUE))
```

```
## 27.8
```

```
crs$pr <- kernlab::predict(crs$ksvm, newdata=na.omit(crs$dataset[crs$validate, c(crs$input, crs
$target)]))

# Generate the confusion matrix showing counts.

rattle::errorMatrix(na.omit(crs$dataset[crs$validate, c(crs$input, crs$target)])$diabetes, crs
$pr, count=TRUE)
```

```
##       Predicted
## Actual neg pos Error
##    neg  69   9  11.5
##    pos  14  23  37.8
```

```
(per <- rattle::errorMatrix(na.omit(crs$dataset[crs$validate, c(crs$input, crs$target)])$diabet
es, crs$pr))
```

```
##       Predicted
## Actual  neg  pos Error
##    neg 60.0  7.8  11.5
##    pos 12.2 20.0  37.8
```

```
cat(100-sum(diag(per), na.rm=TRUE))
```

```
## 20
```

```
cat(mean(per[,"Error"], na.rm=TRUE))
```

```
## 24.65
```

```
#=======================================================================
# Rattle timestamp: 2021-04-12 11:15:04 x86_64-w64-mingw32

# Action the user selections from the Data tab.

# Build the train/validate/test datasets.

# nobs=768 train=538 validate=115 test=115

set.seed(crv$seed)

crs$nobs <- nrow(crs$dataset)

crs$train <- sample(crs$nobs, 0.7*crs$nobs)

crs$nobs %>%
  seq_len() %>%
  setdiff(crs$train) %>%
  sample(0.15*crs$nobs) ->
crs$validate

crs$nobs %>%
  seq_len() %>%
  setdiff(crs$train) %>%
  setdiff(crs$validate) ->
crs$test

# The following variable selections have been noted.

crs$input     <- c("pregnant", "glucose", "insulin", "mass",
                   "age")

crs$numeric   <- c("pregnant", "glucose", "insulin", "mass",
                   "age")

crs$categoric <- NULL

crs$target    <- "diabetes"
crs$risk      <- NULL
crs$ident     <- NULL
crs$ignore    <- c("pressure", "SkinThickness", "pedigree")
crs$weights   <- NULL
```

```
library(rattle)   # Access the weather dataset and utilities.
library(magrittr) # Utilise %>% and %<>% pipeline operators.

# This log generally records the process of building a model.
# However, with very little effort the log can also be used
# to score a new dataset. The logical variable 'building'
# is used to toggle between generating transformations,
# when building a model and using the transformations,
# when scoring a dataset.

building <- TRUE
scoring  <- ! building

# A pre-defined value is used to reset the random seed
# so that results are repeatable.

crv$seed <- 42

#=====================================================================
# Rattle timestamp: 2021-04-12 11:21:40 x86_64-w64-mingw32

# Load a dataset from file.

fname         <- "file:///C:/Users/yangmenglei/Desktop/predictdiabetes.csv"
crs$dataset <- read.csv(fname,
            na.strings=c(".", "NA", "", "?"),
            strip.white=TRUE, encoding="UTF-8")

#=====================================================================
# Rattle timestamp: 2021-04-12 11:21:40 x86_64-w64-mingw32

# Action the user selections from the Data tab.

# Build the train/validate/test datasets.

# nobs=768 train=538 validate=115 test=115

set.seed(crv$seed)

crs$nobs <- nrow(crs$dataset)

crs$train <- sample(crs$nobs, 0.7*crs$nobs)

crs$nobs %>%
  seq_len() %>%
  setdiff(crs$train) %>%
  sample(0.15*crs$nobs) ->
crs$validate

crs$nobs %>%
  seq_len() %>%
  setdiff(crs$train) %>%
  setdiff(crs$validate) ->
crs$test

# The following variable selections have been noted.
```

```
crs$input     <- c("pregnant", "glucose", "pressure",
                   "SkinThickness", "insulin", "mass", "pedigree",
                   "age")

crs$numeric   <- c("pregnant", "glucose", "pressure",
                   "SkinThickness", "insulin", "mass", "pedigree",
                   "age")

crs$categoric <- NULL

crs$target    <- "diabetes"
crs$risk      <- NULL
crs$ident     <- NULL
crs$ignore    <- NULL
crs$weights   <- NULL

#=======================================================================
# Rattle timestamp: 2021-04-12 11:21:49 x86_64-w64-mingw32

# Action the user selections from the Data tab.

# Build the train/validate/test datasets.

# nobs=768 train=538 validate=115 test=115

set.seed(crv$seed)

crs$nobs <- nrow(crs$dataset)

crs$train <- sample(crs$nobs, 0.7*crs$nobs)

crs$nobs %>%
  seq_len() %>%
  setdiff(crs$train) %>%
  sample(0.15*crs$nobs) ->
crs$validate

crs$nobs %>%
  seq_len() %>%
  setdiff(crs$train) %>%
  setdiff(crs$validate) ->
crs$test

# The following variable selections have been noted.

crs$input     <- c("pregnant", "glucose", "insulin", "mass",
                   "age")

crs$numeric   <- c("pregnant", "glucose", "insulin", "mass",
                   "age")

crs$categoric <- NULL

crs$target    <- "diabetes"
crs$risk      <- NULL
crs$ident     <- NULL
crs$ignore    <- c("pressure", "SkinThickness", "pedigree")
crs$weights   <- NULL
```

```
set.seed(crv$seed)

crs$rf <- randomForest::randomForest(as.factor(diabetes) ~ .,
  data=crs$dataset[crs$train, c(crs$input, crs$target)],
  ntree=500,
  mtry=2,
  importance=TRUE,
  na.action=randomForest::na.roughfix,
  replace=FALSE)

# Generate textual output of the 'Random Forest' model.

crs$rf
```

```
##
## Call:
##  randomForest(formula = as.factor(diabetes) ~ ., data = crs$dataset[crs$train,      c(crs$in
put, crs$target)], ntree = 500, mtry = 2, importance = TRUE,      replace = FALSE, na.action =
randomForest::na.roughfix)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 26.44%
## Confusion matrix:
##     neg pos class.error
## neg 280  64   0.1860465
## pos  78 115   0.4041451
```

```
# List the importance of the variables.

rn <- round(randomForest::importance(crs$rf), 2)
rn[order(rn[,3], decreasing=TRUE),]
```

```
##            neg   pos MeanDecreaseAccuracy MeanDecreaseGini
## glucose  30.58 35.95                43.75            51.91
## mass     10.82 17.46                19.21            37.88
## age      15.10  5.36                15.87            30.41
## pregnant 14.94  4.06                14.86            18.50
## insulin   3.20  3.91                 4.98            16.84
```

```
# Time taken: 0.53 secs

#=======================================================================
# Rattle timestamp: 2021-04-12 11:24:51 x86_64-w64-mingw32

# Build a Random Forest model using the traditional approach.

set.seed(crv$seed)

crs$rf <- randomForest::randomForest(as.factor(diabetes) ~ .,
  data=crs$dataset[crs$train, c(crs$input, crs$target)],
  ntree=500,
  mtry=2,
  importance=TRUE,
  na.action=randomForest::na.roughfix,
  replace=FALSE)

# Generate textual output of the 'Random Forest' model.

crs$rf
```

```
##
## Call:
##  randomForest(formula = as.factor(diabetes) ~ ., data = crs$dataset[crs$train,      c(crs$in
put, crs$target)], ntree = 500, mtry = 2, importance = TRUE,      replace = FALSE, na.action =
randomForest::na.roughfix)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 26.44%
## Confusion matrix:
##     neg pos class.error
## neg 280  64   0.1860465
## pos  78 115   0.4041451
```

```
# List the importance of the variables.

rn <- round(randomForest::importance(crs$rf), 2)
rn[order(rn[,3], decreasing=TRUE),]
```

```
##            neg   pos MeanDecreaseAccuracy MeanDecreaseGini
## glucose  30.58 35.95                43.75            51.91
## mass     10.82 17.46                19.21            37.88
## age      15.10  5.36                15.87            30.41
## pregnant 14.94  4.06                14.86            18.50
## insulin   3.20  3.91                 4.98            16.84
```

```
# Time taken: 0.51 secs
```

混淆矩阵评估筛选的特征组合

```
#============================================================
# Rattle timestamp: 2021-04-12 11:27:42 x86_64-w64-mingw32

# Support vector machine.

# The 'kernlab' package provides the 'ksvm' function.

library(kernlab, quietly=TRUE)

# Build a Support Vector Machine model.

set.seed(crv$seed)
crs$ksvm <- ksvm(as.factor(diabetes) ~ .,
      data=crs$dataset[crs$train,c(crs$input, crs$target)],
      kernel="rbfdot",
      prob.model=TRUE)

# Generate a textual view of the SVM model.

crs$ksvm
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.270344395789267
##
## Number of Support Vectors : 308
##
## Objective Function Value : -257.613
## Training error : 0.201117
## Probability model included.
```

```
# Time taken: 0.09 secs#============================================================================

# Rattle is Copyright (c) 2006-2020 Togaware Pty Ltd.
# It is free (as in libre) open source software.
# It is licensed under the GNU General Public License,
# Version 2. Rattle comes with ABSOLUTELY NO WARRANTY.
# Rattle was written by Graham Williams with contributions
# from others as acknowledged in 'library(help=rattle)'.
# Visit https://rattle.togaware.com/ for details.

# Rattle timestamp: 2021-04-12 11:23:20 x86_64-w64-mingw32

# Build a Random Forest model using the traditional approach.

set.seed(crv$seed)

crs$rf <- randomForest::randomForest(as.factor(diabetes) ~ .,
  data=crs$dataset[crs$train, c(crs$input, crs$target)],
  ntree=500,
  mtry=2,
  importance=TRUE,
  na.action=randomForest::na.roughfix,
  replace=FALSE)

# Generate textual output of the 'Random Forest' model.

crs$rf
```

```
##
## Call:
##  randomForest(formula = as.factor(diabetes) ~ ., data = crs$dataset[crs$train,      c(crs$in
put, crs$target)], ntree = 500, mtry = 2, importance = TRUE,      replace = FALSE, na.action =
randomForest::na.roughfix)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 26.44%
## Confusion matrix:
##     neg pos class.error
## neg 280  64   0.1860465
## pos  78 115   0.4041451
```

```
# List the importance of the variables.

rn <- round(randomForest::importance(crs$rf), 2)
rn[order(rn[,3], decreasing=TRUE),]
```

```
##            neg   pos MeanDecreaseAccuracy MeanDecreaseGini
## glucose  30.58 35.95                43.75            51.91
## mass     10.82 17.46                19.21            37.88
## age      15.10  5.36                15.87            30.41
## pregnant 14.94  4.06                14.86            18.50
## insulin   3.20  3.91                 4.98            16.84
```

```
# Time taken: 0.53 secs

#=======================================================================
# Rattle timestamp: 2021-04-12 11:24:51 x86_64-w64-mingw32

# Build a Random Forest model using the traditional approach.

set.seed(crv$seed)

crs$rf <- randomForest::randomForest(as.factor(diabetes) ~ .,
  data=crs$dataset[crs$train, c(crs$input, crs$target)],
  ntree=500,
  mtry=2,
  importance=TRUE,
  na.action=randomForest::na.roughfix,
  replace=FALSE)

# Generate textual output of the 'Random Forest' model.

crs$rf
```

```
##
## Call:
##  randomForest(formula = as.factor(diabetes) ~ ., data = crs$dataset[crs$train,      c(crs$in
put, crs$target)], ntree = 500, mtry = 2, importance = TRUE,      replace = FALSE, na.action =
randomForest::na.roughfix)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 26.44%
## Confusion matrix:
##     neg pos class.error
## neg 280  64   0.1860465
## pos  78 115   0.4041451
```

```
# List the importance of the variables.

rn <- round(randomForest::importance(crs$rf), 2)
rn[order(rn[,3], decreasing=TRUE),]
```

```
##            neg   pos MeanDecreaseAccuracy MeanDecreaseGini
## glucose  30.58 35.95                43.75            51.91
## mass     10.82 17.46                19.21            37.88
## age      15.10  5.36                15.87            30.41
## pregnant 14.94  4.06                14.86            18.50
## insulin   3.20  3.91                 4.98            16.84
```

```
# Time taken: 0.51 secs
```

ROC曲线评估筛选的特征组合

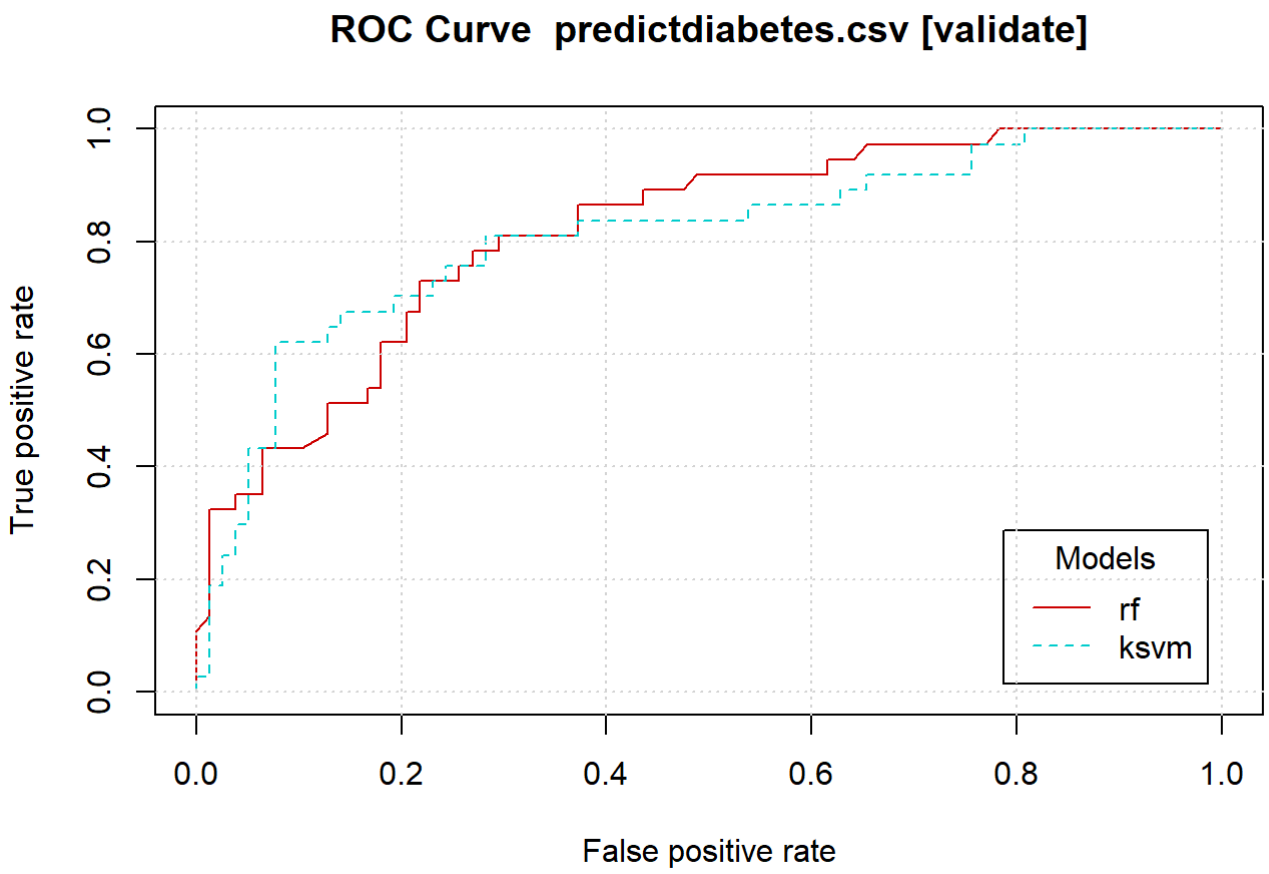
```
#=======================================================================
# Rattle timestamp: 2021-04-12 11:33:15 x86_64-w64-mingw32

# Evaluate model performance on the validation dataset.

# ROC Curve: requires the ROCR package.

library(ROCR)

# Generate an ROC Curve for the rf model on predictdiabetes.csv [validate].

crs$pr <- predict(crs$rf, newdata=na.omit(crs$dataset[crs$validate, c(crs$input, crs$target)]),
    type    = "prob")[,2]

# Remove observations with missing target.

no.miss   <- na.omit(na.omit(crs$dataset[crs$validate, c(crs$input, crs$target)])$diabetes)
miss.list <- attr(no.miss, "na.action")
attributes(no.miss) <- NULL

if (length(miss.list))
{
  pred <- prediction(crs$pr[-miss.list], no.miss)
} else
{
  pred <- prediction(crs$pr, no.miss)
}
ROCR::plot(performance(pred, "tpr", "fpr"), col="#CC0000", lty=1, add=FALSE)

# Calculate the area under the curve for the plot.


# Remove observations with missing target.

no.miss   <- na.omit(na.omit(crs$dataset[crs$validate, c(crs$input, crs$target)])$diabetes)
miss.list <- attr(no.miss, "na.action")
attributes(no.miss) <- NULL

if (length(miss.list))
{
  pred <- prediction(crs$pr[-miss.list], no.miss)
} else
{
  pred <- prediction(crs$pr, no.miss)
}
performance(pred, "auc")
```

```
## A performance instance
##   'Area under the ROC curve'
```

```
# ROC Curve: requires the ROCR package.

library(ROCR)

# Generate an ROC Curve for the ksvm model on predictdiabetes.csv [validate].

crs$pr <- kernlab::predict(crs$ksvm, newdata=na.omit(crs$dataset[crs$validate, c(crs$input, crs
$target)]),
    type    = "probabilities")[,2]

# Remove observations with missing target.

no.miss   <- na.omit(na.omit(crs$dataset[crs$validate, c(crs$input, crs$target)])$diabetes)
miss.list <- attr(no.miss, "na.action")
attributes(no.miss) <- NULL

if (length(miss.list))
{
  pred <- prediction(crs$pr[-miss.list], no.miss)
} else
{
  pred <- prediction(crs$pr, no.miss)
}
ROCR::plot(performance(pred, "tpr", "fpr"), col="#00CCCC", lty=2, add=TRUE)

# Calculate the area under the curve for the plot.



# Remove observations with missing target.

no.miss   <- na.omit(na.omit(crs$dataset[crs$validate, c(crs$input, crs$target)])$diabetes)
miss.list <- attr(no.miss, "na.action")
attributes(no.miss) <- NULL

if (length(miss.list))
{
  pred <- prediction(crs$pr[-miss.list], no.miss)
} else
{
  pred <- prediction(crs$pr, no.miss)
}
performance(pred, "auc")
```

```
## A performance instance
##   'Area under the ROC curve'
```

```
# Add a legend to the plot.

legend("bottomright", c("rf","ksvm"), col=rainbow(2, 1, .8), lty=1:2, title="Models", inset=c(
0.05, 0.05))

# Add decorations to the plot.

title(main="ROC Curve  predictdiabetes.csv [validate]",
    sub=)
grid()
```




```
## Area under the ROC curve for the rf model on predictdiabetes.csv [validate] is 0.8170
#=====================================================================
## Area under the ROC curve for the ksvm model on predictdiabetes.csv [validate] is 0.8105
```

#由此可见，所选特征组合可以很好地预测分类（ROC=0.81）结果