哪些客户特征可以预测客户是否流失？-基于 R 语言的 SVM 分类、模型优化及模型评估

王昊 *(学号: 201821061107)* 指导教师: 段小刚

2019 年 *11* 月 *30* 日

目录

1 摘要

一个电话公司感兴趣的是确定哪些客户特征对预测客户流失 (客户将离开他们的服务) 是有用的。本文采用支持向量机的方法探究哪些客户特征对预测客户流失有意义，并对参数进行调整，最后评价了各种不同的模型，得到了最好的模型。

2 背景

一个电话公司感兴趣的是确定哪些客户特征对预测客户流失 (客户将离开他们的服务) 是有用的。

3 研究问题

哪些客户特征对预测客户流失 (客户将离开他们的服务) 是有用的。

4 对数据的描述性分析

telecom churn 数据集来训练 SVM。

```
# 从 C50 中获取 telecom churn 数据集
#install.packages("C50")
library(C50)
data(churn)
```

```
# 查看数据集结构
str(churnTrain)
```

```
## 'data.frame':    3333 obs. of  20 variables:
##  $ state                          : Factor w/ 51 levels "AK","AL","AR",..: 17 36 32 36 37 2 20 25
##  $ account_length                 : int  128 107 137 84 75 118 121 147 117 141 ...
##  $ area_code                      : Factor w/ 3 levels "area_code_408",..: 2 2 2 1 2 3 3 2 1 2 ..
##  $ international_plan             : Factor w/ 2 levels "no","yes": 1 1 1 2 2 2 1 2 1 2 ...
##  $ voice_mail_plan                : Factor w/ 2 levels "no","yes": 2 2 1 1 1 1 2 1 1 2 ...
##  $ number_vmail_messages          : int  25 26 0 0 0 0 24 0 0 37 ...
##  $ total_day_minutes              : num  265 162 243 299 167 ...
##  $ total_day_calls                : int  110 123 114 71 113 98 88 79 97 84 ...
##  $ total_day_charge               : num  45.1 27.5 41.4 50.9 28.3 ...
##  $ total_eve_minutes              : num  197.4 195.5 121.2 61.9 148.3 ...
##  $ total_eve_calls                : int  99 103 110 88 122 101 108 94 80 111 ...
##  $ total_eve_charge               : num  16.78 16.62 10.3 5.26 12.61 ...
```

```
##  $ total_night_minutes          : num  245 254 163 197 187 ...
##  $ total_night_calls            : int  91 103 104 89 121 118 118 96 90 97 ...
##  $ total_night_charge           : num  11.01 11.45 7.32 8.86 8.41 ...
##  $ total_intl_minutes           : num  10 13.7 12.2 6.6 10.1 6.3 7.5 7.1 8.7 11.2 ...
##  $ total_intl_calls             : int  3 3 5 7 3 6 7 6 4 5 ...
##  $ total_intl_charge            : num  2.7 3.7 3.29 1.78 2.73 1.7 2.03 1.92 2.35 3.02 ...
##  $ number_customer_service_calls: int  1 1 0 2 3 0 3 0 1 0 ...
##  $ churn                        : Factor w/ 2 levels "yes","no": 2 2 2 2 2 2 2 2 2 2 ...
```

```
# 删除一些没有贡献的属性
churnTrain=churnTrain[,! names(churnTrain) %in% c("state","area_code","account_length")]

# 划分训练集和测试集
set.seed(2)
ind=sample(2,
           nrow(churnTrain),
           replace=TRUE,
           prob=c(0.7,0.3))
trainset=churnTrain[ind==1,]
testset=churnTrain[ind==2,]



# 查看维度
dim(trainset)
```

```
## [1] 2315   17
```

```
dim(testset)
```

```
## [1] 1018   17
```

5 统计模型的具体形式

5.1 使用支持向量机完成数据分类

e1071 包提供了 libsvm 的实现

```
#install.packages("e1071")
library(e1071)

model=svm(churn~.,#churn 是分类类别.
          data=trainset,
          kernel="radial",
```

```
          cost=1,
          gamma=1/ncol(trainset))#gamma 函数确定了分离超平面的形状，默认为数据维度的倒数，提高 gamma 会

summary(model)
```

```
##
## Call:
## svm(formula = churn ~ ., data = trainset, kernel = "radial", cost = 1,
##     gamma = 1/ncol(trainset))
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  691
##
##  ( 394 297 )
##
##
## Number of Classes:  2
##
## Levels:
##  yes no
```

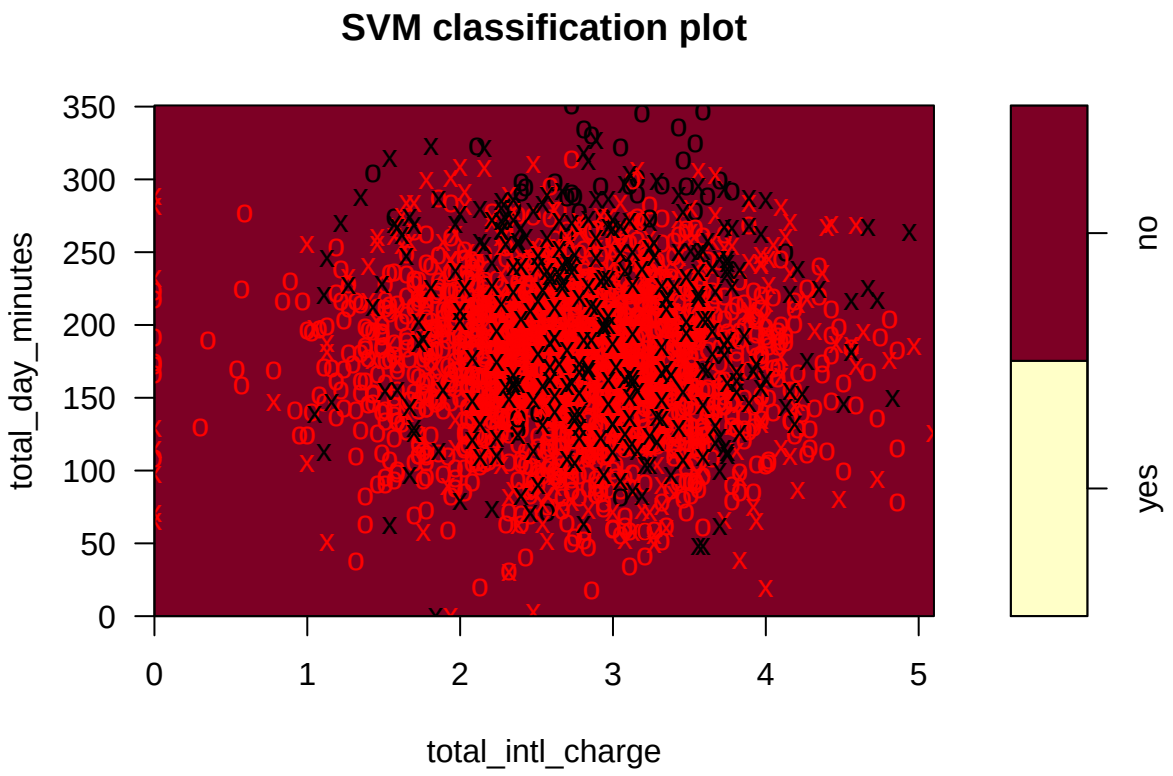
5.2 选择支持向量机的惩罚因子

能实现 SVM 对分类误差及分类边界的控制。惩罚因子比较小，分类间隔会比较大（软间隔），将产生比较多的被错分样本。

5.3 实现 SVM 模型的可视化

支持向量和类别被高亮显示。等高线图绘制类的边缘。

```
plot(model,# 模型名称
     trainset,# 样本数据集（和构建模型的数据集一致）
     total_day_minutes ~ total_intl_charge)# 分类图坐标轴的说明
```



红色的支持向量和黑色的数据样例在图中心区域排列很紧密，不能直接分开。

5.4 基于支持向量机训练模型实现类预测

1. 利用已构建的模型和测试数据集预测它的类别

```
svm.pred=predict(model,
                 testset[,!names(testset) %in% c("churn")])
```

2. 建立分类表

```
svm.table=table(svm.pred,
                testset$churn)
```

3. 分析一致性系数

```
classAgreement(svm.table)
```

```
## $diag
## [1] 0.9184676
##
## $kappa
## [1] 0.5855903
```

```
##
## $rand
## [1] 0.850083
##
## $crand
## [1] 0.5260472
```

4. 基于分类表评测预测性能

```
library(caret)
```

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

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

```
confusionMatrix(svm.table)
```

```
## Confusion Matrix and Statistics
##
##
## svm.pred yes  no
##      yes  70  12
##      no   71 865
##
##                Accuracy : 0.9185
##                  95% CI : (0.8999, 0.9345)
##     No Information Rate : 0.8615
##     P-Value [Acc > NIR] : 1.251e-08
##
##                   Kappa : 0.5856
##
##  Mcnemar's Test P-Value : 1.936e-10
##
##             Sensitivity : 0.49645
##             Specificity : 0.98632
##          Pos Pred Value : 0.85366
##          Neg Pred Value : 0.92415
##              Prevalence : 0.13851
##          Detection Rate : 0.06876
##    Detection Prevalence : 0.08055
##       Balanced Accuracy : 0.74139
##
##        'Positive' Class : yes
```

```
##
```

5.5 调整支持向量机

1. tune.svm

```
tuned=tune.svm(churn~.,
               data=trainset,
               gamma=10^(-6:-1),
               cost=10^(1:2))
summary(tuned)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  gamma cost
##   0.01  100
##
## - best performance: 0.07992051
##
## - Detailed performance results:
##    gamma cost      error dispersion
## 1  1e-06   10 0.14773474 0.01763545
## 2  1e-05   10 0.14773474 0.01763545
## 3  1e-04   10 0.14773474 0.01763545
## 4  1e-03   10 0.14773474 0.01763545
## 5  1e-02   10 0.09116099 0.02007738
## 6  1e-01   10 0.09288513 0.02507137
## 7  1e-06  100 0.14773474 0.01763545
## 8  1e-05  100 0.14773474 0.01763545
## 9  1e-04  100 0.14773474 0.01763545
## 10 1e-03  100 0.11880505 0.01581561
## 11 1e-02  100 0.07992051 0.02049819
## 12 1e-01  100 0.12226638 0.02627152
```

3. 最佳参数设置 SVM

```
model.tuned=svm(churn~.,
                data=trainset,
```

```
                gamma=tuned$best.parameters$gamma,
                cost=tuned$best.parameters$cost)
summary(model.tuned)
```

```
##
## Call:
## svm(formula = churn ~ ., data = trainset, gamma = tuned$best.parameters$gamma,
##     cost = tuned$best.parameters$cost)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  100
##
## Number of Support Vectors:  547
##
##  ( 304 243 )
##
##
## Number of Classes:  2
##
## Levels:
##  yes no
```

4. 类标号预测

```
svm.tuned.pred=predict(model.tuned,
                       testset[, !names(testset) %in% c("churn")])
```

5. 分类表

```
svm.tuned.table=table(svm.tuned.pred,
                      testset$churn)
svm.tuned.table
```

```
##
## svm.tuned.pred yes  no
##            yes  95  24
##            no   46 853
```

6. 得到相关系数, 完成算法性能评测

```
classAgreement(svm.tuned.table)
```

```
## $diag
## [1] 0.9312377
##
## $kappa
## [1] 0.691678
##
## $rand
## [1] 0.871806
##
## $crand
## [1] 0.6303615
```

7. 评测优化后的模型性能

```
confusionMatrix(svm.tuned.table)
```

```
## Confusion Matrix and Statistics
##
##
## svm.tuned.pred yes  no
##            yes  95  24
##            no   46 853
##
##                Accuracy : 0.9312
##                  95% CI : (0.9139, 0.946)
##     No Information Rate : 0.8615
##     P-Value [Acc > NIR] : 1.56e-12
##
##                   Kappa : 0.6917
##
##  Mcnemar's Test P-Value : 0.01207
##
##             Sensitivity : 0.67376
##             Specificity : 0.97263
##          Pos Pred Value : 0.79832
##          Neg Pred Value : 0.94883
##              Prevalence : 0.13851
##          Detection Rate : 0.09332
##    Detection Prevalence : 0.11690
```

```
##       Balanced Accuracy : 0.82320
##
##        'Positive' Class : yes
##
```

试错法寻找最佳的 gamma 和惩罚因子。

6 模型的结果分析与解释

6.1 模型评估

6.1.1 基于 k 折交叉验证方法评测模型性能

```
# 索引分成 10 份
ind=cut(1:nrow(churnTrain),
        breaks=10,
        labels=F)

#SVM 依赖
library(e1071)
#10 折交叉验证
accuracies=c()
for (i in 1:10){
  fit=svm(churn~.,
          churnTrain[ind!=i,])
  predictions=predict(fit,
                      churnTrain[ind==i, !names(churnTrain) %in% c("churn")])
  correct_count=sum(predictions==churnTrain[ind== i,c("churn")])
  accuracies= append(correct_count/nrow(churnTrain[ind==i,]),
                     accuracies)
}

# 输出准确率
accuracies
```

```
##  [1] 0.9341317 0.8948949 0.8978979 0.9459459 0.9219219 0.9281437 0.9219219
##  [8] 0.9249249 0.9189189 0.9251497
```

```
mean(accuracies)
```

```
## [1] 0.9213852
```

6.1.2 利用 e1071 包完成交叉验证

tuning 函数获取最小误差值。

```
tuned=tune.svm(churn~.,
               data=trainset,
               gamma=10^-2,
               cost=10^-2,
               tunecontrol=tune.control(cross=10))

summary(tuned)
```

```
##
## Error estimation of 'svm' using 10-fold cross validation: 0.1477142
```

3. 模型的性能细节

```
tuned$performances
```

```
##   gamma cost     error dispersion
## 1  0.01 0.01 0.1477142  0.0258799
```

4. 使用优化后的模型产生分类表

```
svmfit=tuned$best.model
table(trainset[,c("churn")],
      predict(svmfit))
```

```
##
##        yes   no
##   yes    0  342
##   no     0 1973
```

6.1.3 利用 caret 包完成交叉检验

```
#install.packages("caret")
library(caret)

# 重复的 k 折交叉验证，被用于检测模型的稳定性。如果稳定，用户将得到类似的测试结果。
# 设置控制训练参数，进行重复 3 次的 10 折交叉验证
control=trainControl(method="repeatedcv",
                     number=10,
                     repeats=3)
```

```
model=train(churn~.,
            data=trainset,
            method="rpart",
            preProcess="scale",
            trControl=control)

model
```

```
## CART
##
## 2315 samples
##   16 predictor
##    2 classes: 'yes', 'no'
##
## Pre-processing: scaled (16)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 2083, 2083, 2084, 2084, 2084, 2084, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.05555556  0.9072814  0.5580218
##   0.07456140  0.8711367  0.3141883
##   0.07602339  0.8621988  0.2009855
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.05555556.
```

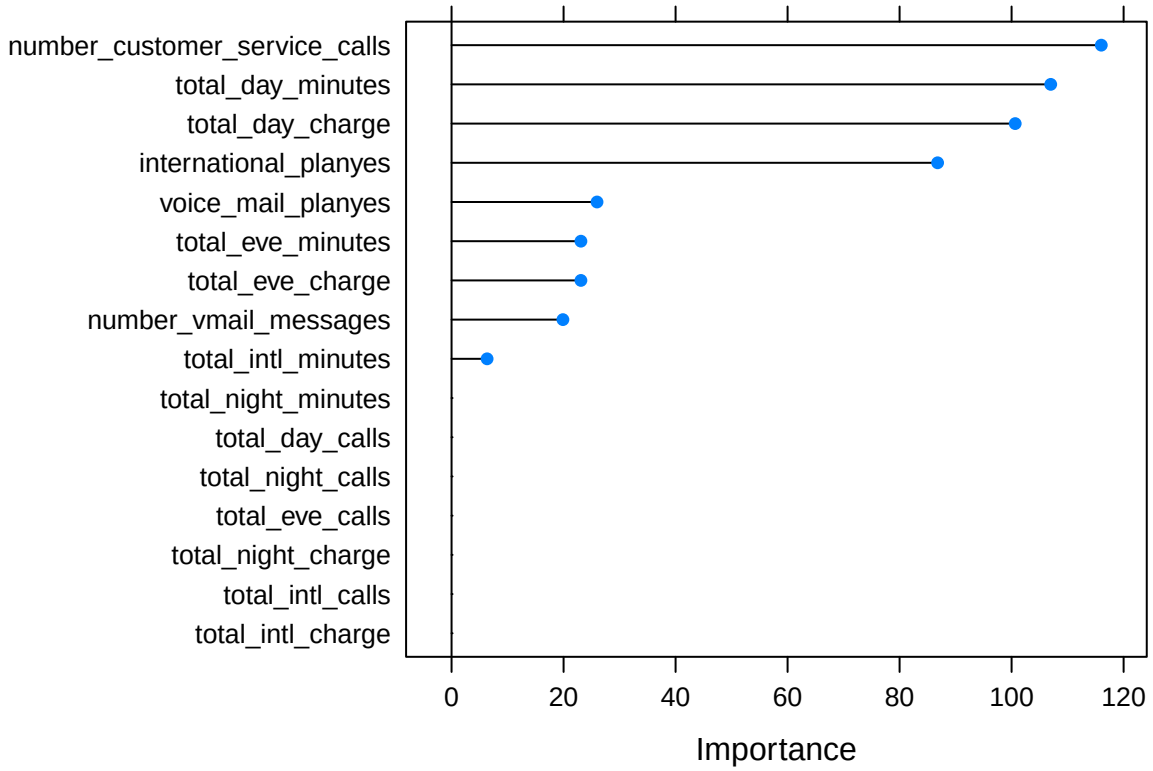
模型输出了 3 次重新采样的结果，其中 cp 为 0.555 的模型准确率最高。### 利用 caret 包对变量重要程度排序比较给定模型输出效果的变化敏感程度来评估不同特征对模型的重要性。

```
importance=varImp(model,scale=FALSE)
importance
```

```
## rpart variable importance
##
##                               Overall
## number_customer_service_calls 116.015
## total_day_minutes             106.988
## total_day_charge              100.648
## international_planyes          86.789
## voice_mail_planyes             25.974
```

```
## total_eve_charge               23.097
## total_eve_minutes              23.097
## number_vmail_messages          19.885
## total_intl_minutes              6.347
## total_day_calls                 0.000
## total_night_minutes             0.000
## total_eve_calls                 0.000
## total_intl_charge               0.000
## total_night_charge              0.000
## total_night_calls               0.000
## total_intl_calls                0.000
```

```
plot(importance)
```



6.1.4 利用 caret 包找到高度关联的特征

如果能提前去掉高度关联的属性，训练模型的性能会更好。

```
# 去掉非数值类型的属性
new_train=trainset[, !names(churnTrain)  %in% c("churn","international_plan","voice_mail_plan")]
```

```r
# 计算每个属性之间的关联度
cor_mat=cor(new_train)

# 找到关联度超过 0.75 的属性
highlyCorrelated=findCorrelation(cor_mat,cutoff=0.75)

names(new_train)[highlyCorrelated]
```

```
## [1] "total_intl_minutes"  "total_day_charge"    "total_eve_minutes"
## [4] "total_night_minutes"
```

6.1.5 利用 caret 包选择特征

递归特征排除函数 rfe

```r
#international_plan 的特征转换为 intl_yes 和 Intl_no
intl_plan=model.matrix(~ trainset.international_plan - 1,
                       data=data.frame(trainset$international_plan))

colnames(intl_plan)=c("trainset.international_planno"="intl_no",
                      "trainset.international_planyes"="intl_yes")


#voice_mail_plan 的特征转换为 voice_yes 和 voice_no
voice_plan=model.matrix(~ trainset.voice_mail_plan-1,
                        data=data.frame(trainset$voice_mail_plan))

colnames(voice_plan)=c("trainset.voice_planno"="voice_no",
                       "trainset.voice_planyes"="voice_yes")

# 去掉 2 个属性，并将训练数据与两个数据框合并
trainset$international_plan=NULL
trainset$voice_mail_plan=NULL
trainset=cbind(intl_plan,voice_plan,trainset)

#test 数据集同理如上
intl_plan=model.matrix(~ testset.international_plan - 1,
                       data=data.frame(testset$international_plan))

colnames(intl_plan)=c("testset.international_planno"="intl_no",
                      "testset.international_planyes"="intl_yes")
```

```
voice_plan=model.matrix(~ testset.voice_mail_plan-1, data=data.frame(testset$voice_mail_plan))

colnames(voice_plan)=c("testset.voice_planno"="voice_no",
                       "testset.voice_planyes"="voice_yes")



testset$international_plan=NULL
testset$voice_mail_plan=NULL
testset=cbind(intl_plan,voice_plan,testset)
```

7. 使用 LDA 创建一个特征筛选

```
ldaControl=rfeControl(functions = ldaFuncs,method="cv")
```

8. 利用从编号 1-18 的数据子集对训练数据集 trainset 进行反向特征筛选

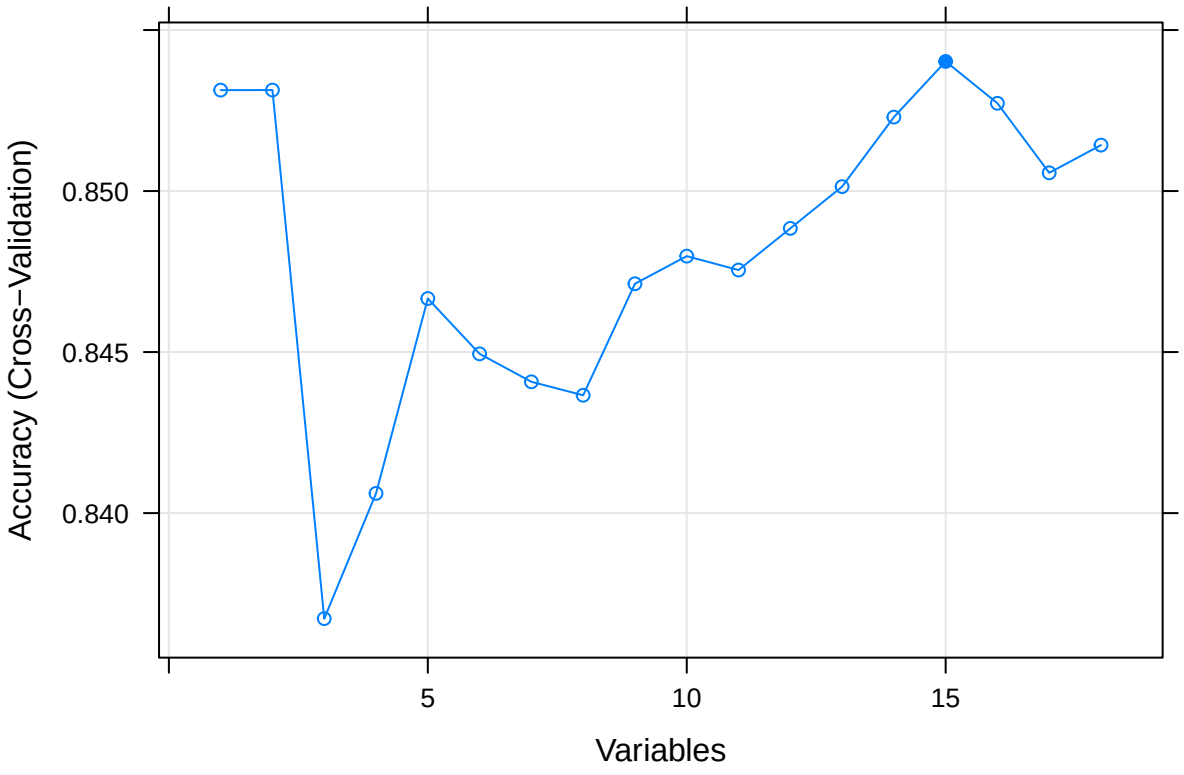
```
ldaProfile=rfe(trainset[, !names(trainset) %in% c("churn")],trainset[,c("churn")],
               sizes = c(1:18),
               rfeControl = ldaControl)

ldaProfile
```

```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (10 fold)
##
## Resampling performance over subset size:
##
##  Variables Accuracy    Kappa AccuracySD KappaSD Selected
##          1   0.8531 0.009672   0.001767 0.02039
##          2   0.8531 0.009829   0.002581 0.02072
##          3   0.8367 0.130371   0.017907 0.09934
##          4   0.8406 0.204617   0.021050 0.10803
##          5   0.8467 0.226621   0.024150 0.14245
##          6   0.8449 0.219809   0.021163 0.11980
##          7   0.8441 0.216697   0.021582 0.12753
##          8   0.8437 0.216058   0.025947 0.14637
##          9   0.8471 0.236037   0.027635 0.14082
##         10   0.8480 0.237765   0.026364 0.14168
```

```
##          11   0.8475 0.235295   0.027232 0.14220
##          12   0.8488 0.242297   0.027482 0.14613
##          13   0.8501 0.251306   0.027321 0.14812
##          14   0.8523 0.255369   0.026387 0.15208
##          15   0.8540 0.266462   0.026225 0.15073       *
##          16   0.8527 0.258823   0.024326 0.14263
##          17   0.8506 0.252423   0.025551 0.14221
##          18   0.8514 0.254383   0.025252 0.14281
##
## The top 5 variables (out of 15):
##    total_day_charge, total_day_minutes, intl_yes, intl_no, number_customer_service_calls
```

9. 选择结果

```
plot(ldaProfile,type=c("o","g"))
```



10. 检测最佳变量子集

```
ldaProfile$optVariables
```

```
##  [1] "total_day_charge"              "total_day_minutes"
##  [3] "intl_yes"                      "intl_no"
```

```
##  [5] "number_customer_service_calls" "total_eve_minutes"
##  [7] "total_eve_charge"              "total_intl_calls"
##  [9] "voice_yes"                     "voice_no"
## [11] "number_vmail_messages"         "total_intl_charge"
## [13] "total_intl_minutes"            "total_night_charge"
## [15] "total_night_minutes"
```

11. 检测合适的模型

```
ldaProfile$fit
```

```
## Call:
## lda(x, y)
##
## Prior probabilities of groups:
##       yes        no
## 0.1477322 0.8522678
##
## Group means:
##     total_day_charge total_day_minutes   intl_yes   intl_no
## yes         35.00143          205.8877 0.29532164 0.7046784
## no          29.62402          174.2555 0.06487582 0.9351242
##     number_customer_service_calls total_eve_minutes total_eve_charge
## yes                      2.204678          213.7269         18.16702
## no                       1.441460          199.6197         16.96789
##     total_intl_calls voice_yes  voice_no number_vmail_messages
## yes         4.134503 0.1666667 0.8333333              5.099415
## no          4.514445 0.2954891 0.7045109              8.674607
##     total_intl_charge total_intl_minutes total_night_charge total_night_minutes
## yes          2.899386           10.73684           9.245994            205.4640
## no           2.741343           10.15119           9.063882            201.4184
##
## Coefficients of linear discriminants:
##                                        LD1
## total_day_charge                0.671376021
## total_day_minutes              -0.123085519
## intl_yes                       -1.130269257
## intl_no                         1.130269257
## number_customer_service_calls  -0.421346323
## total_eve_minutes               0.183124313
## total_eve_charge               -2.210668112
```

```
## total_intl_calls                0.066574547
## voice_yes                       0.330197066
## voice_no                       -0.330197066
## number_vmail_messages          -0.003558459
## total_intl_charge               2.316437631
## total_intl_minutes             -0.694000839
## total_night_charge            -14.513932481
## total_night_minutes             0.651028494
```

12. 重采样评估模型

```
postResample(predict(ldaProfile,
                     testset[, !names(testset) %in% c("churn")]),
             testset[,c("churn")])
```

```
##  Accuracy     Kappa
## 0.8585462 0.2568816
```

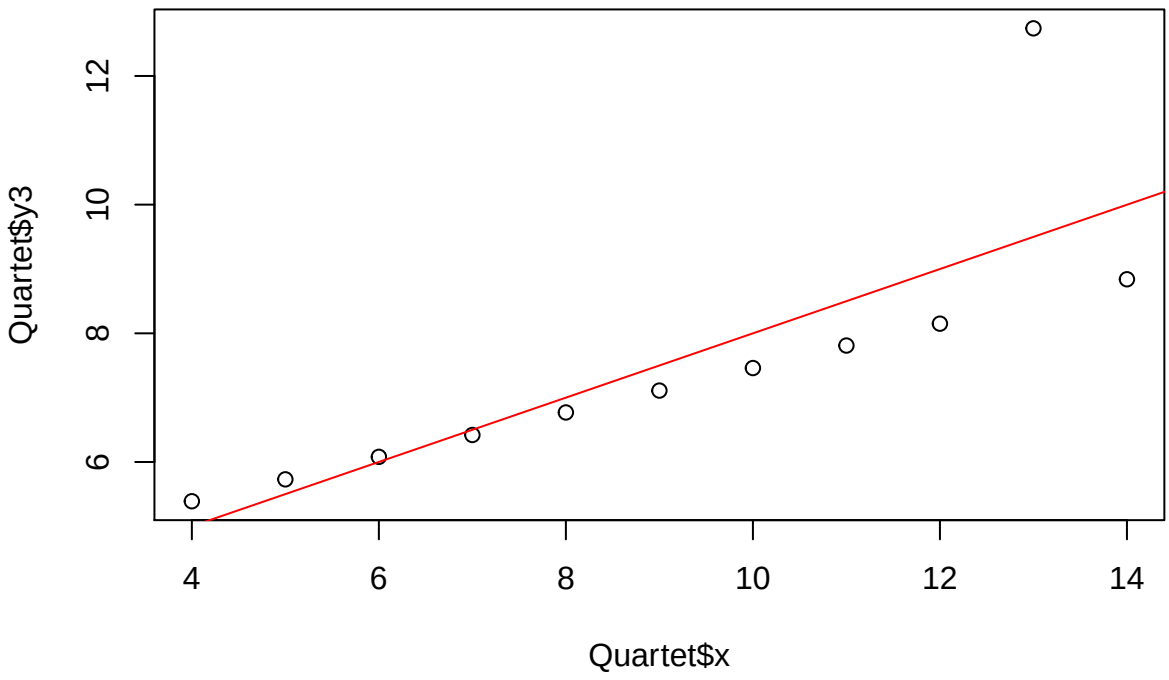
6.1.6 评测回归模型的性能

相对平方差 (Relative Square Error)。数据换用 Quartet 数据集。

```
#install.packages("car")
library(car)
```

```
## Loading required package: carData
```

```
data(Quartet)

plot(Quartet$x,Quartet$y3)
lmfit=lm(Quartet$y3~Quartet$x)
abline(lmfit,col="red")
```



3. 预测结果

```
predicted=predict(lmfit,newdata=Quartet[c("x")])
```

4. 均方误差

```
actual=Quartet$y3
rmse=(mean((predicted-actual)^2))^0.5
rmse
```

```
## [1] 1.118286
```

5. 相对平方误差

```
mu=mean(actual)
rse=mean((predicted-actual)^2)/mean((mu-actual)^2)
rse
```

```
## [1] 0.333676
```

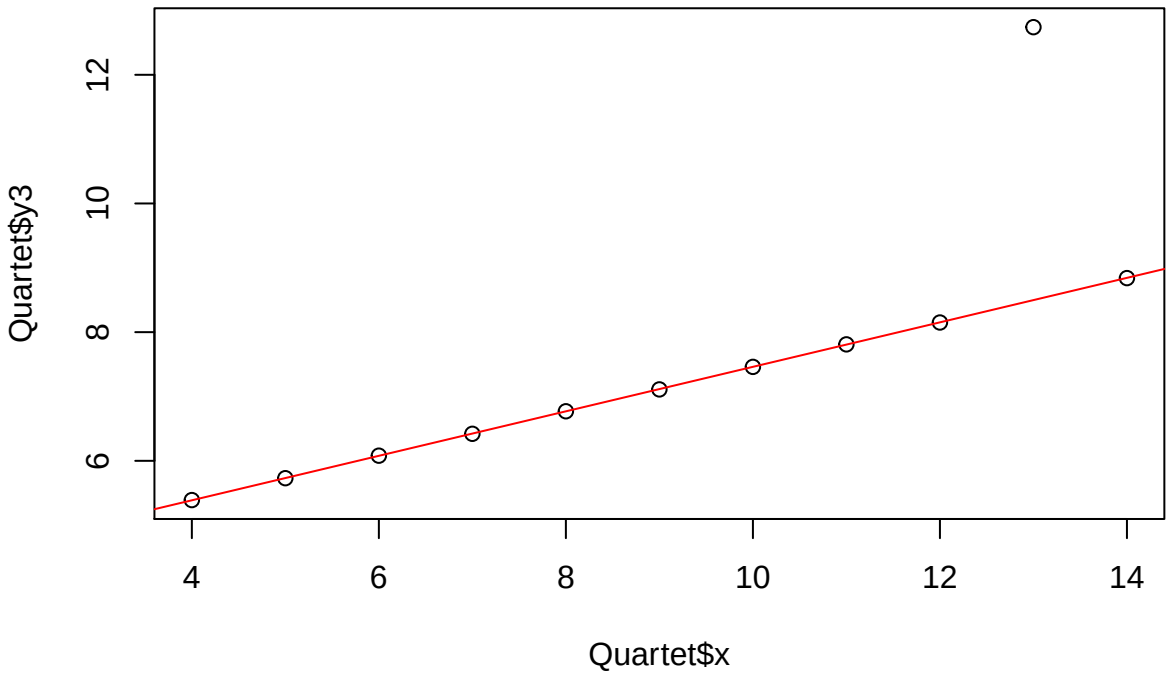
6. R^2

```
rsquare=1-rse
rsquare
```

```
## [1] 0.666324
```

7. 采用 MASS 包重新计算 y3 的值。用模糊线性回归处理数据集。

```
library(MASS)
plot(Quartet$x,Quartet$y3)
rlmfit=rlm(Quartet$y3~Quartet$x)
abline(rlmfit, col="red")
```



8. 预测

```
predicted=predict(rlmfit,newdata = Quartet[c("x")])
```

9. 均方根误差

```
actual=Quartet$y3
rmse=(mean((predicted-actual)^2))^0.5
rmse
```

```
## [1] 1.279045
```

10. 相对平方误差

```
mu=mean(actual)
rse=mean((predicted-actual)^2)/mean((mu-actual)^2)
rse
```

```
## [1] 0.4365067
```

11. R^2

```
rsquare=1-rse
rsquare
```

```
## [1] 0.5634933
```

lm 方法建立的模型其 RMSE 和 RSE 要低于 rlm，在 R^2 的比较中显示出 lm 建立的有更高的预测能力。实际操作中，我们会首先去掉 x=13 这个异常值。#### 线性回归模型上交叉验证

```
tune(lm,y3~x,data=Quartet)
```

```
##
## Error estimation of 'lm' using 10-fold cross validation: 2.346036
```

6.1.7 利用混淆矩阵评测模型的预测能力

```
#install.packages("kernlab")
svm.model=train(churn~.,
                data=trainset,
                method="svmRadial")
svm.pred=predict(svm.model,
                 testset[,!names(testset) %in% c("churn")])

# 分类表
table(svm.pred,testset[,c("churn")])
```

```
##
## svm.pred yes  no
##      yes  76  13
##      no   65 864
```

```
# 预测结果和实际类标号的混淆矩阵
confusionMatrix(svm.pred,testset[,c("churn")])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction yes  no
```

```
##        yes  76  13
##        no   65 864
##
##                Accuracy : 0.9234
##                  95% CI : (0.9053, 0.939)
##     No Information Rate : 0.8615
##     P-Value [Acc > NIR] : 5.191e-10
##
##                   Kappa : 0.6202
##
##  Mcnemar's Test P-Value : 7.713e-09
##
##             Sensitivity : 0.53901
##             Specificity : 0.98518
##          Pos Pred Value : 0.85393
##          Neg Pred Value : 0.93003
##              Prevalence : 0.13851
##          Detection Rate : 0.07466
##    Detection Prevalence : 0.08743
##       Balanced Accuracy : 0.76209
##
##        'Positive' Class : yes
##
```

6.1.8 利用 ROCR 评测模型的预测能力

```
#install.packages("ROCR")
library(ROCR)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

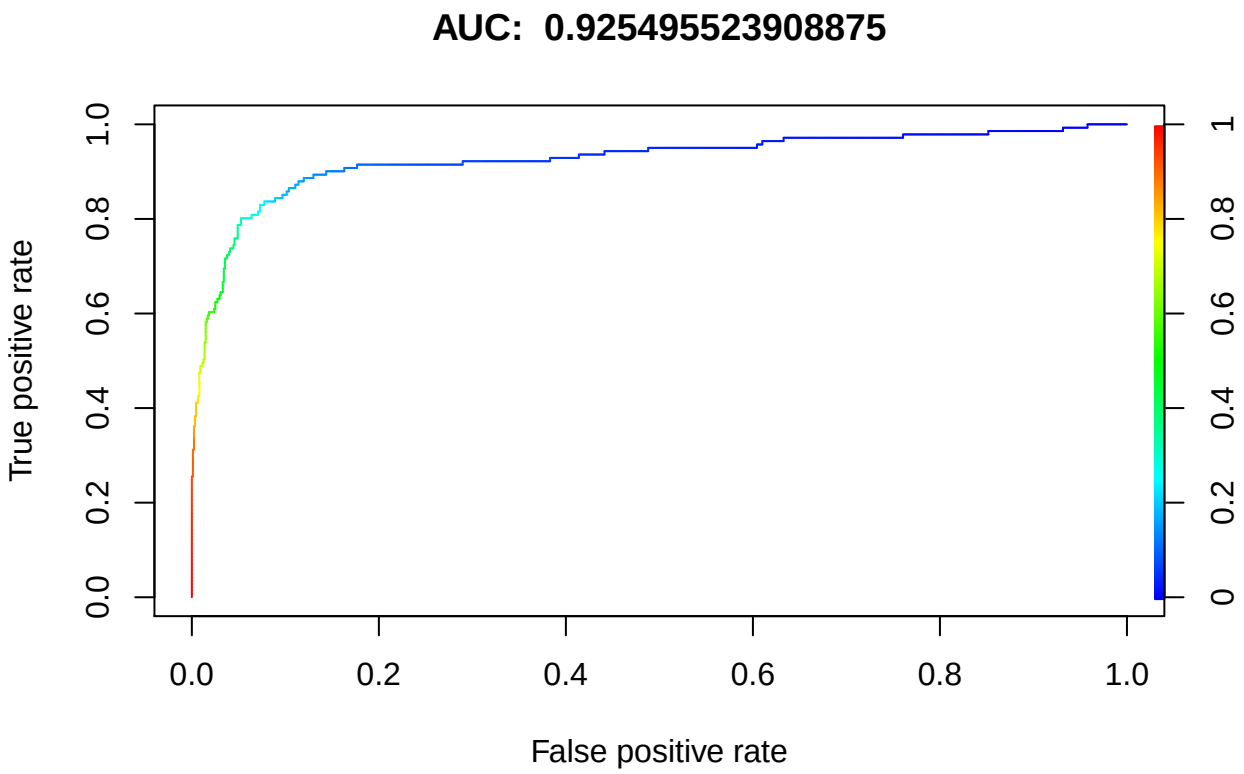
```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
svmfit=svm(churn~.,
           data=trainset,
           prob=TRUE)
```

```
pred=predict(svmfit,
             testset[,!names(testset) %in% c("churn")],
             probability = TRUE)

# 得到标号为 yes 的概率
pred.prob=attr(pred,"probabilities")
pred.to.roc=pred.prob[,2]

# 预测结果
pred.rocr=prediction(pred.to.roc,
                     testset$churn)
```

6. 性能评估

```
perf.rocr=performance(pred.rocr,
                      measure="auc",
                      x.measure = "cutoff")
perf.tpr.rocr=performance(pred.rocr,
                          "tpr",
                          "fpr")
plot(perf.tpr.rocr,
     colorize=T,
     main=paste("AUC: ",
                (perf.rocr@y.values)))
```



6.1.9 利用 caret 包比较 ROC 曲线

```
#install.packages("pROC")

library("pROC")
```

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

```
##
```

```
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
```

```
##
```

```
##     cov, smooth, var
```

```
# 训练控制方法
control=trainControl(method="repeatedcv",
                     number=10,
                     repeats=3,
                     classProbs=TRUE,
                     summaryFunction = twoClassSummary)
```

```
# 使用 glm 在训练数据集上训练一个分类器
glm.model= train(churn~.,
                 data=trainset,
                 method="glm",
                 metric="ROC",
                 trControl=control)

# 使用 svm 在训练数据集上训练一个分类器
svm.model=train(churn ~ .,
                data = trainset,
                method="svmRadial",
                metric="ROC",
                trControl = control)
```

5. 查看 rpart 在训练数据上的运行情况

```
rpart.model = train(churn ~ .,
                    data = trainset,
                    method="rpart",
                    metric="ROC",
                    trControl=control)
```

6. 使用不同的已训练好的模型分别进行预测

```
glm.probs  =predict(glm.model,
                  testset[, ! names(testset) %in% c("churn")],
                  type= "prob")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
svm.probs  =predict(svm.model,
                  testset[, ! names(testset) %in% c("churn")],
                  type= "prob")
rpart.probs=predict(rpart.model,
                  testset[, ! names(testset) %in% c("churn")],
                  type= "prob")
```

7. 生成 ROC 曲线

```
glm.ROC=roc(response=testset[,c("churn")],
            predictor=glm.probs$yes,
            levels=levels(testset[,c("churn")]))
```

```
## Setting direction: controls > cases
```

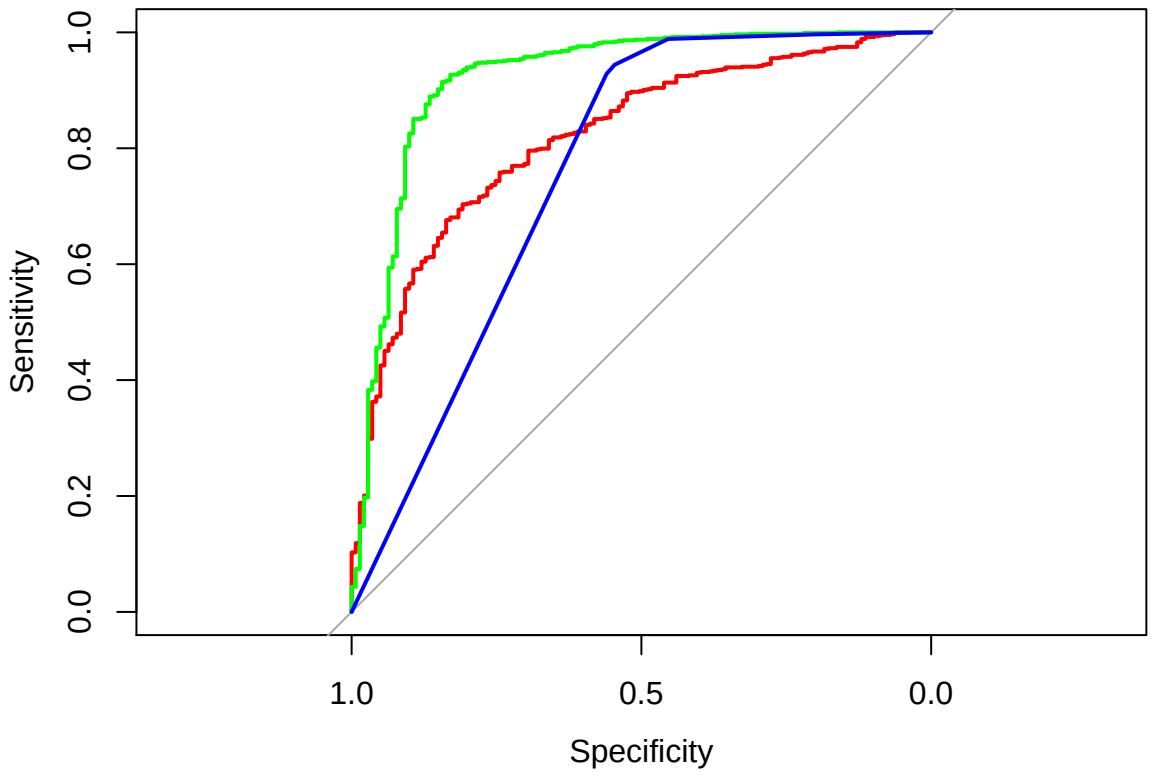
```
plot(glm.ROC,type="S",col="red")


svm.ROC=roc(response=testset[,c("churn")],
            predictor=svm.probs$yes,
            levels=levels(testset[,c("churn")]))
```

```
## Setting direction: controls > cases
```

```
plot(svm.ROC,add=TRUE,col="green")


rpart.ROC=roc(response=testset[,c("churn")],
            predictor=rpart.probs$yes,
            levels=levels(testset[,c("churn")]))
```

```
## Setting direction: controls > cases
```

```
plot(rpart.ROC,add=TRUE,col="blue")
```



6.1.10 利用 caret 包比较模型性能差异

重采样的得到每一个匹配模型的统计信息，包括 ROC、灵敏度、特异度。

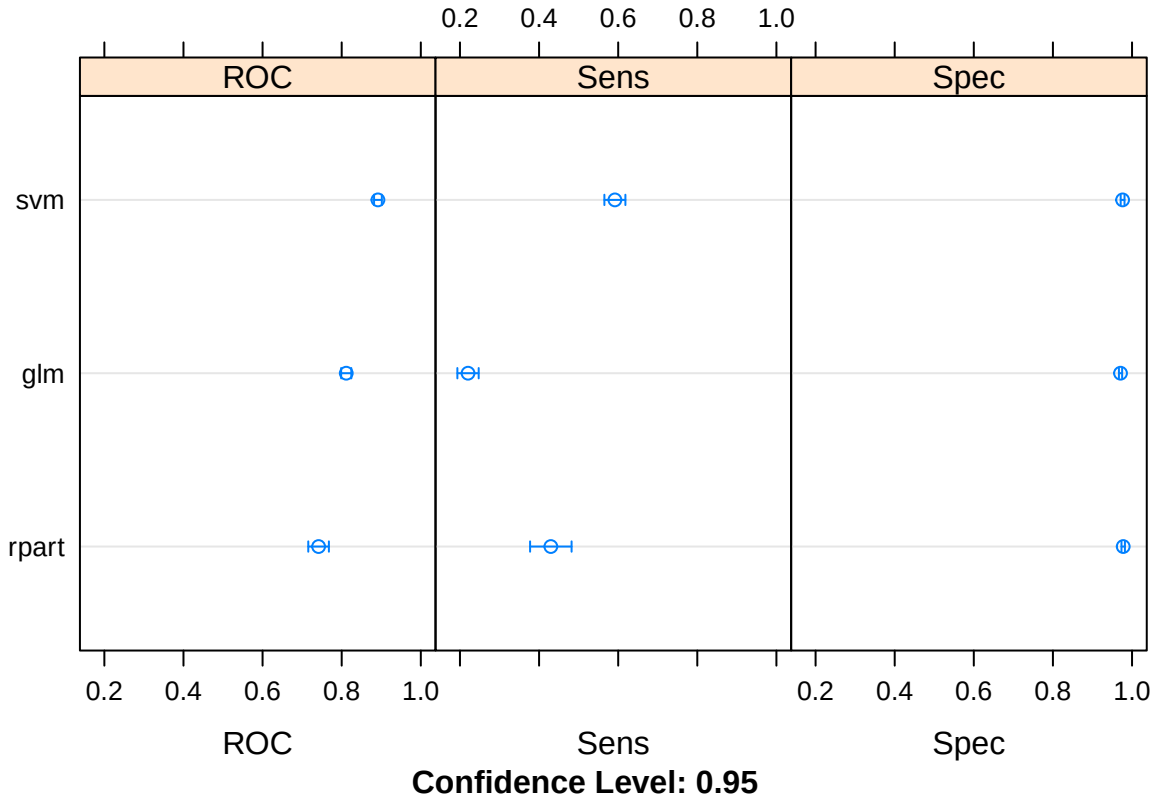
1. 重采样

```
cv.values=resamples(list(glm=glm.model,
                         svm=svm.model,
                         rpart=rpart.model))
summary(cv.values)
```

```
##
## Call:
## summary.resamples(object = cv.values)
##
## Models: glm, svm, rpart
## Number of resamples: 30
##
## ROC
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## glm   0.7393764 0.7914810 0.8097574 0.8114992 0.8321514 0.9034040    0
## svm   0.8287549 0.8785213 0.8924306 0.8916343 0.9108113 0.9384891    0
## rpart 0.5715885 0.6851976 0.7424457 0.7415887 0.8042699 0.8567483    0
##
## Sens
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## glm   0.08823529 0.1764706 0.2058824 0.2203641 0.2573529 0.4117647    0
## svm   0.48571429 0.5294118 0.6029412 0.5916807 0.6258403 0.7352941    0
## rpart 0.14705882 0.3235294 0.4201681 0.4297199 0.5514706 0.6857143    0
##
## Spec
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## glm   0.9543147 0.9644670 0.9695431 0.9709335 0.9796954 0.9898990    0
## svm   0.9492386 0.9695431 0.9772215 0.9761686 0.9847716 1.0000000    0
## rpart 0.9492386 0.9709275 0.9796954 0.9777009 0.9848485 0.9949239    0
```

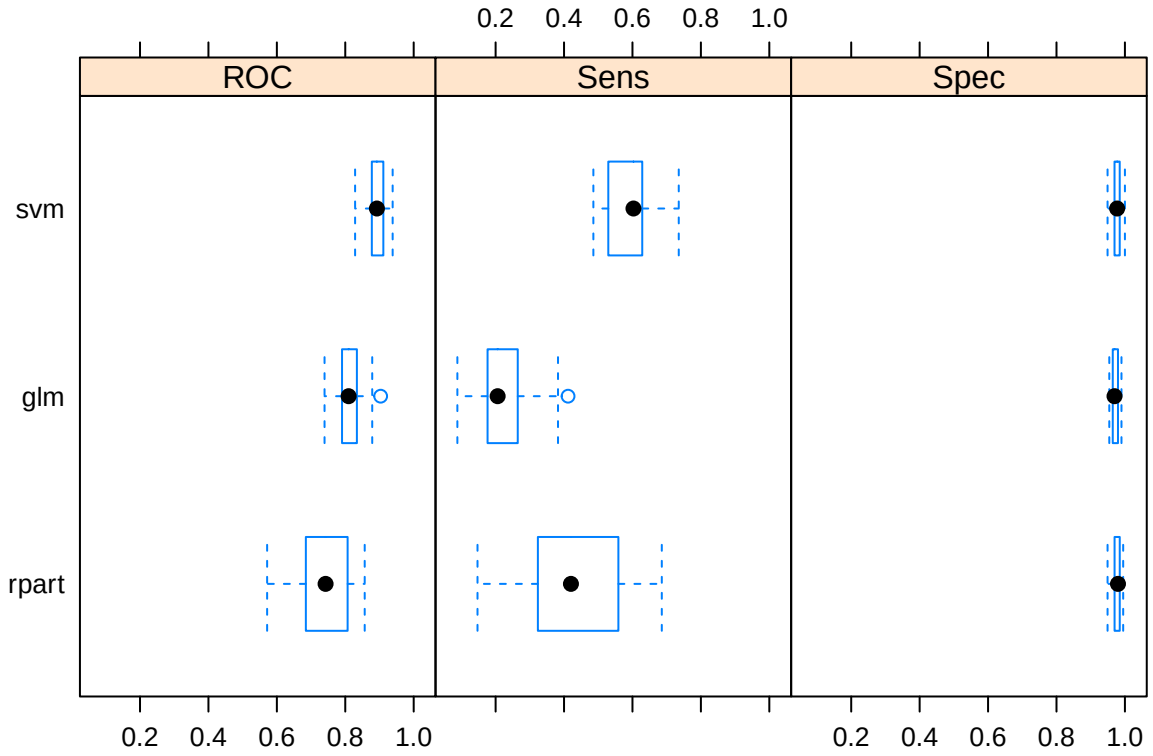
3. 重采样在 ROC 曲线度量中的结果

```
dotplot(cv.values,metrics="ROC")
```



4. 箱线绘制重采样结果

```
bwplot(cv.values,layout=c(3,1))
```



7 参考文献

[1]CHIU Y-W. Machine Learning with R Cookbook[M]. Packt Publishing Ltd, 2015.

8 附录 (代码)

(将以文件形式附在压缩包内)