大数据统计与计量分析——第三章作业

姓名：吴博成　班级：大数据91　　学号：2193211134

---

1 课本复现

1.1 数据背景

Cell segmentation数据集是将微观图像域分割成代表单个细胞实例的片段的任务。它是许多生物医学研究的基础步骤，被认为是基于图像的细胞研究的基石。Cell segmentation数据集是数据集包“Applied Predictive Modeling”中的一个重要数据集，数据集有119维，2019条，维数较高。数据获取的方法:

```
library ( AppliedPredictiveModeling)
data( segmentationOriginal )
```

1.2 数据初步处理

使用过滤法(filter)和封装法(wrapper)来选择一个最优的预测变量子集。对于 linear discriminant analysis 和 logistic regression ,使用嵌套(build-in)的特征选择方法(如 R 包 glmnet 和稀疏的 LDA)选择变量。对比这些方法,从预测变量个数、预测性能、训练时间来对比不同方法。

- **过滤法和封装法**
  1. 数据准备: 包括将数据分为训练集和测试集,将训练集用于训练模型,以及删掉不需要进入特征筛选的变量,得到**解释变量集合。**
  2. 无监督过滤法: 采用无监督过滤,即**不需要因变量的参与**对解释变量进行初步过滤。包括**剔除方差为0**的变量以及彼此高度相关的变量以消除多重共线性。
  3. 有监督过滤法: 需要**因变量参与**的过滤法,剔除**与因变量不够相关**的变量。至于相关性的度量,若解释变量为二分类变量,则采用 Fisher 检验;若解释变量为数值型变量,则采用t检验,删除 p 值较大的变量。
  4. 封装法: 用逐步回归法建立广义线性模型的**二项式回归**, 模型品质的度量选择了 AIC 信息量
- **嵌套方法**

此处分别运用选择过滤器(selected by filter , SBF)和循环特征选择器(recursive feature elimination , RFE)的线性判别分析法(linear discriminant analysis , LDA)和随机森林法(random forest，PF) 四种方法对训练集建立模型,并将模型结果运用于测试集以确定方法的预测准确率。

1.3 过滤法和封装法

初始数据集包含119个变量,进行必要的选择: 剔除不需要的细胞标识ID(Cell)、是否正确分割（Class）、数据用于训练集还是测试集(Case) 这3个变重，剩余 116个变量。剔除方差为0的变量3个，剩余113个变量。剔除彼此高度相关(相关系数阈值：0.75) 的变量32个,剩余81个。剔除与因变量不够相关(p 值阈值定为 0.05) 的变量50个 , 剩余28个变量。最后利用 AIC 准则逐步回归法进行封装,选择出14个变量, AIC信息量为853.8。

最终选择的14个解释变量包括: ConvexHullPerimRatioCh1 、 FiberWidthCh1 、 IntenCoocASMCh3 、 IntenCoocASMCh4 、 IntenCoocContrastCh3 、 TotalIntenCh2 、 VarIntenCh1 、 VarIntenCh4 、 AvgIntenStatusCh1 、 IntenCoocASMStatusCh3 、 IntenCoocMaxStatusCh3 、 SkewIntenStatusCh1 、 TotalIntenStatusCh2 、 VarIntenStatusCh1。

最终得到的预测模型如图1所示:

```
> resultstep$call
glm(formula = yclass ~ ConvexHullPerimRatioCh1 + FiberWidthCh1 +
    IntenCoocASMCh3 + IntenCoocASMCh4 + IntenCoocContrastCh3 +
    TotalIntenCh2 + VarIntenCh1 + VarIntenCh4 + AvgIntenStatusCh1 +
    IntenCoocASMStatusCh3 + IntenCoocMaxStatusCh3 + SkewIntenStatusCh1 +
    TotalIntenStatusCh2 + VarIntenStatusCh1, family = binomial,
    data = xyfilted)
```

图1 逐步回归运行结果

1.4 嵌套方法

运用 SBF — LDA 、 SBF — RF 、 RFE — LDA 、 RFE — RF 四种方法,均采用十折交叉验证,根据运行结果我们发现:

- 预测变量数量方面, SBF 方法由于使用规则对变量进行过滤,得到的变量较多，RFE方法得到的变量数量较少。
- 预测稳定性方面,四种方法交叉验证的准确率以及方差差距不大, SBF — RF 准确最高, SBF — LDA 预测方差较小。
- 训练时间方面, RFE 方法运行时间普遍高于 SBF 方法,这与过滤方法，这与过滤方法选择过程独立于训练过程有关，但可能删除有用的特征，四种方法中RFE-RF运行时间最长。

分类器	变量数	准确率	准确率标准差	训练时长
SBF-LDA	71	80.42%	0.03422	13.48 s
SBF-RF	71	83.29%	0.03274	90.94 s
RFE-LDA	16	80.22%	0.03732	5.12 s
RFE-RF	16	82.51%	0.04146	313.58 s

表2 不同嵌套方法比较

1.5 代码展示

```
#-----------------数据预处理及加载工作包等准备工作------------
library(AppliedPredictiveModeling) #加载数据所在的包
library(caret)
library(MASS) #加载逐步回归需要的包
library("e1071") #加载SBF方法中提示需要的包
```

```r
library("randomForest") #加载随机森林算法需要的包
data(segmentationOriginal) #启用数据
head(segmentationOriginal)
seg = subset(segmentationOriginal,Case =="Train") #使用训练集作为样本
segtest = subset(segmentationOriginal,Case =="Test")#使用测试集检验
yclass = seg$Class
xseg = seg[,-(1:3)] #删除非解释变量,得到解释变量的集合

#----------------------无监督过滤法-------------------------
##剔除方差为0的变量
nearZeroVar(xseg)              #查找方差为0的变量
xseg = xseg[,c(-68,-73,-74)]  #剔除方差为0的变量

#-----------------剔除彼此高度相关的变量--------------------
statuscolnum = grep("Status",names(xseg))  #筛选出二分类定性变量
xseg1= xseg[,-statuscolnum] #暂时删掉定性变量,因为要考察自变量之间的相关性以减弱多重共线性
xsegstatus = xseg[,statuscolnum]
correlations = cor(xseg1)
dim(correlations)
highcor = findCorrelation(correlations,0.75) #筛选出彼此高度和关的变量
length(highcor)
xsegnum =xseg1[,-highcor]
xsegdata = cbind(xsegnum,xsegstatus) #去除强相关变量

#--------------------有监督过滤法--------------------------
##剔除与因变量y不够相关的变量
#编写函数 pScore(),考察x与y的相关性
pScore = function(x,y)
{
numX = length(unique(x))
if(numX > 2)#定量变量采用t检验
  {out = t.test(x~y)$p.value}
else #二分类定性变量采用 Fisher 检验
  {out = fisher.test(factor(x),y)$p.value}
out
}

#编写函数 cal(),为每个 x 计算与 y 的相关性
cal = function(x){
  print(length(unique(x)))
  return(length(unique(x)))
  }
scores = apply(X = xsegdata,MARGIN =2, FUN = pScore,y = yclass)
tail(scores)

#编写函数 pCorrection(),调整 p 值并按照阈值筛选变量
pCorrection = function(score ,p0){
  score = p.adjust(score,"bonferroni")
  keepers =(score<=p0)
  print(keepers)
  return(keepers)
}
result1= pCorrection(scores,0.05)#运用上述函数进行变量过滤 colnames(xsegdata)[result1]
xsegfilted = xsegdata[,result1]
dim(xsegfilted)
xyfilted = cbind(xsegfilted,yclass)
```

```r

#----------------------------封装法----------------------------------
initial = glm(yclass~., data = xyfilted,family = binomial)#使用逐步回归建立二项问归
resultstep = stepAIC(initial ,direction ="both")#运用 AIC 准则筛选变鼠
resultstep$call

#---------------------- built-in方法比较---------------------------
##训练集测试集数据准备
xtrain = seg [,-c(1:3,71,76,77)]
ytrain = seg [,3]
train = cbind(xtrain,ytrain)
xtest = segtest[,-c(1:3,71,76,77)]
ytest = segtest[,3]
test = cbind(xtest,ytest)

## SBF-LDA用时
ldfCtrl= sbfControl(method ="repeatedcv", repeats = 5, functions = ldaSBF,verbose = F)
t1= Sys.time()
ldaFilter = sbf(xtrain,ytrain,tol = 1.0e-12,sbfControl = ldfCtrl)
t2= Sys.time()
t2-t1#计算方法运行时间,下同
ldaFilter

## SBF-RE用时
rffCtrl = sbfControl(method ="repeatedcv", repeats = 5, functions = rfSBF, verbose = F)
t1= Sys.time()
rfFilter = sbf(xtrain,ytrain,sbfControl = rffCtrl)
t2= Sys.time()
t2-t1
rfFilter

## RFE-LDA用时
ldrctrl = rfeControl(method = "repeatedcv", repeats = 5, verbose = F , functions =ldaFuncs)
set.seed(721)
t1= Sys.time ()
ldaRFE = rfe (xtrain,ytrain,metric ="Roc",rfeControl = ldrctrl)
t2= Sys.time ()
t2-t1
ldaRFE

## RFE-RF
rfrctrl = rfeControl(method ="repeatedcv", repeats = 5, verbose = F , functions = rfFuncs)
set.seed(100)
t1= Sys.time()
rfRFE = rfe(xtrain,ytrain,metric = "Roc", rfeControl = rfrctrl)
t2= Sys.time()
t2-t1
rfRFE
```

2 自行获取数据——汽车销售价格

2.1 数据背景介绍

获得的数据为汽车4S店客户购买车辆的具体信息，数据来源为2020第四届全国应用统计专业学位研究生案例大赛企业选题A题。MAS数据来源地址

数据共有18个维度，共38387条非空数据。下面对数据集car_info.csv(下载链接)各列进行描述:

列名	描述
CUST_ID	客户ID
CUST_SEX	客户性别：1=男 2=女
CUST_AGE	客户年龄
CUST_MARRY	客户婚姻状况
BUYERPART	车主性质：1=个人, 2=公司, 3=机关, 4=组织,5=其他
CAR_MODEL	车型代码
CAR_COLOR	车型颜色
CAR_AGE	车龄：单位：天
CAR_PRICE	车辆销售价格
IS_LOAN	是否贷款买车
LOAN_PERIED	贷款期限：1=6个月,2=9个月,3=12个月，4=24个月，5=36个月，6=48个月，7=60个月，8=其他
LOAN_AMOUNT	贷款金额
F_INSORNOT	新车投保是否在4s店
ALL_BUYINS_N	在4S店购买保险总次数
DLRSI_CNT	购买4S店专修险的次数
GLASSBUYSEPARATE_CNT	购买玻璃单独险的次数
SII_CNT	购买自燃险的次数
IS_LOST	是否流失：1=流失，0=未流失

表3 car_info数据意义描述

解释变量为之前的17列变量，IS_LOST为类别变量，我们希望观察客户流失情况与之前这些解释变量的关系。

2.2 数据预处理

首先查看数据缺失值情况，原数据集共有51075条数据，各列变量空值数量见表4

变量列名	空值个数
CUST_ID	0
CUST_SEX	0
CUST_AGE	475
CUST_MARRY	39038
BUYERPART	0
CAR_MODEL	0
CAR_COLOR	21312
CAR_AGE	0
CAR_PRICE	0
IS_LOAN	0
LOAN_PERIED	5607
LOAN_AMOUNT	5607
F_INSORNOT	8151
ALL_BUYINS_N	4631
DLRSI_CNT	4631
GLASSBUYSEPARATE_CNT	4631
SII_CNT	4631
IS_LOST	0

表4 car_info各列数据空值个数

由于CAR_COLOR和CUST_MARRY列缺失值太多，不具有分析价值，故予以删除。

2.3 过滤法和封装法

初始数据集**去空后**包含16个变量,进行必要的选择: 剔除不需要的CUST_ID、IS_LOST，剩余 14个变量。剔除方差为0的变量3个F_INSORNOT、IS_LOAN、BUYERPART，剩余11个变量。剔除彼此高度相关(相关系数阈值：0.75) 的变量0个,剩余11个。剔除与因变量不够相关(p 值阈值定为 0.05) 的变量6个，剩余6个变量。最后利用 AIC 准则逐步回归法进行封装,选择出5个变量, AIC信息量为34403.45。

最终选择的5个解释变量包括: CUST_AGE、CAR_AGE、CAR_PRICE、LOAN_PERIED、ALL_BUYINS_N。最终得到的预测模型如图5所示：

```
> resultstep$call
glm(formula = yclass ~ CUST_AGE + CAR_AGE + CAR_PRICE + LOAN_PERIED +
    ALL_BUYINS_N, family = binomial, data = xyfilted)
```

图5 车辆数据逐步回归运行结果

2.4 嵌套方法

运用 SBF — LDA 、 SBF — RF 、 RFE — LDA 、 RFE — RF 四种方法,均采用十折交叉验证,根据运行结果我们发现:

- 预测变量数量方面, SBF 方法由于使用规则对变量进行过滤,得到的变量较多，RFE方法得到的变量数量较少。
- 预测稳定性方面,四种方法交叉验证的准确率以及方差差距不大, SBF — RF 准确最高, SBF — LDA 预测方差很小。
- 训练时间方面, RFE 方法运行时间普遍高于 SBF 方法,这与过滤方法，这与过滤方法选择过程独立于训练过程有关，但可能删除有用的特征，四种方法中RFE-RF运行时间最长。
- 由于R语言为解释性语言，只能调用单核心CPU，运算速度较慢，不建议进行机器学习模型构建

分类器	变量数	准确率	准确率标准差	训练时长
SBF-LDA	7	79.1%	0.003889	12.88 s
SBF-RF	7	83.29%	0.003274	937.94 s
RFE-LDA	5	80.22%	0.03732	9.89 s
RFE-RF	未知	未知	未知	挂在服务器跑了5小时之后内存不够，崩了

表2 不同嵌套方法比较

2.5 代码展示

```
#-----------------数据预处理及加载工作包等准备工作------------

library(caret)
library(MASS) #加载逐步回归需要的包
library("e1071") #加载SBF方法中提示需要的包
library("randomForest") #加载随机森林算法需要的包
library(knitr)

sale_data = read.csv("car_info_train.csv", header = T, sep = ",")
sale_data_test = read.csv("car_info_test.csv", header = T, sep = ",")

# 将无法识别为NA的中文确实变量转为NA
sale_data[sale_data==""] <- NA
sale_data_test[sale_data_test==""] <- NA

# 查看变量缺失情况
count_na <- function(x){
  return(sum(is.na(x)))
}
m = apply(sale_data, 2, count_na)
kable(m)
sale_data = na.omit(sale_data[,-c(1,4,7)])
sale_data_test = na.omit(sale_data_test[,-c(1,4,7)]) #删除非解释变量和缺失值过多的变量，剔除剩余有空的行,得到解释变量的集合
yclass = sale_data$IS_LOST
xsale_data = sale_data[,-15]

```

```r
#----------------------无监督过滤法-------------------------
##剔除方差为0的变量
nearZeroVar(xsale_data)              #查找方差接近0的变量
xsale_data = xsale_data[,-c(3,7,10)]  #剔除方差为0的变量

#-----------------剔除彼此高度相关的变量--------------------
correlations = cor(xsale_data)
dim(correlations)
highcor = findCorrelation(correlations,0.75) #筛选出彼此高度和关的变量
length(highcor)
xsale_data_num =xsale_data[,-highcor]
xsale_data = cbind(xsale_data_num,xsale_data) #去除强相关变量

#--------------------有监督过滤法--------------------------
##剔除与因变量y不够相关的变量
#编写函数 pScore(),考察x与y的相关性
pScore = function(x,y)
{
  numX = length(unique(x))
  if(numX > 2)#定量变量采用t检验
  {out = t.test(x~y)$p.value}
  else #二分类定性变量采用 Fisher 检验
  {out = fisher.test(factor(x),y)$p.value}
  out
}

#编写函数 cal(),为每个 x 计算与 y 的相关性
cal = function(x){
  print(length(unique(x)))
  return(length(unique(x)))
}
scores = apply(X = xsale_data,MARGIN =2, FUN = pScore,y = yclass)
tail(scores)

#编写函数 pCorrection(),调整 p 值并按照阈值筛选变量
pCorrection = function(score ,p0){
  score = p.adjust(score,"bonferroni")
  keepers =(score<=p0)
  print(keepers)
  return(keepers)
}
result1= pCorrection(scores,0.05)#运用上述函数进行变量过滤 colnames(xsegdata)
[result1]
xsale_data_filted = xsale_data[,result1]
dim(xsale_data_filted)
xyfilted = cbind(xsale_data_filted,yclass)

#----------------------------封装法-----------------------------------
initial = glm(yclass~., data = xyfilted,family = binomial)#使用逐步回归建立二
项问归
resultstep = stepAIC(initial ,direction ="both")#运用 AIC 准则筛选变鼠
resultstep$call

#---------------------- built-in方法比较--------------------------
##训练集测试集数据准备
xtrain = sale_data[,-15]
ytrain = as.factor(sale_data[,15])
train = cbind(xtrain,ytrain)
```

```r
xtest = sale_data_test[,-15]
ytest = as.factor(sale_data_test[,15])
test = cbind(xtest,ytest)

## SBF-LDA用时
ldfCtrl= sbfControl(method ="repeatedcv", repeats = 5, functions =
ldaSBF,verbose = F)
t1= Sys.time()
ldaFilter = sbf(xtrain,ytrain,tol = 1.0e-12,sbfControl = ldfCtrl)
t2= Sys.time()
t2-t1#计算方法运行时间,下同
ldaFilter

## SBF-RE用时
rffCtrl = sbfControl(method ="repeatedcv", repeats = 5, functions = rfSBF,
verbose = F)
t1= Sys.time()
rfFilter = sbf(xtrain,ytrain,sbfControl = rffCtrl)
t2= Sys.time()
t2-t1
rfFilter

## RFE-LDA用时
ldrctrl = rfeControl(method = "repeatedcv", repeats = 5, verbose = F ,
functions =ldaFuncs)
set.seed(721)
t1= Sys.time ()
ldaRFE = rfe (xtrain,ytrain,metric ="Roc",rfeControl = ldrctrl)
t2= Sys.time ()
t2-t1
ldaRFE

## RFE-RF
rfrctrl = rfeControl(method ="repeatedcv", repeats = 5, verbose = F ,
functions = rfFuncs)
set.seed(100)
t1= Sys.time()
rfRFE = rfe(xtrain,ytrain,metric = "Roc", rfeControl = rfrctrl)
t2= Sys.time()
t2-t1
rfRFE

```