高级统计方法 第6次作业:

序号：1 姓名：杨程瑜 学号：20202241107 班级：网2001

**第五章**

**概念**

3.问题（略）

（a）问题（略）

将数据集随机分割成k（k<=n）个大小相等的子集，模型在k-1个子集上训练，在剩下的那个子集上进行测试。这个过程重复k次，k个折叠中的每一个子集都充当一次测试数据集。最后记录每一次的测试误差的估计并将其平均。

（b）问题（略）

i.问题（略）

优点：k折交叉验证的结果比验证集方法更加稳定，且测试误差更低。

对于验证集方法，数据仅区分一次，存在很高的偶然性，意味着测试误差的结果高度依赖于训练集和测试集的分割。同时仅区分一次训练集与测试集，导致验证集方法容易高估测试误差。而k折交叉验证的所有数据都用于训练集和测试集的不同分割子集中，结果偶然性很低，比较稳定和可信。

缺点：k折交叉验证相比验证集方法，不具有计算优势，同时解释性差些。

对于验证集方法，只进行一次拟合与结果测试，计算次数少。当对于较大的数据集和较大的k值时，k折交叉验证方法会花费更多时间。

ii.问题（略）

优点：k折交叉验证计算花费时间更少，且结果的方差更低。

当数据量比较大时，使用k折交叉验证的拟合和测试次数更少，花费的计算时间更少。比如n=10000，取k=10，k折交叉验证方法的计算时间是留一验证法的千分之一。同时由于偏差-方差的权衡，留一验证法近乎将所有的数据用来训练，导致测试集上的方差较大，而k折交叉验证的测试结果方差低一些。

缺点：k折交叉验证的偏差更高，同时测试结果存在一定随机性。

留一验证法近乎将所有的数据用来训练，在测试集上的偏差更小，而k折交叉验证的测试结果偏差高一些。并且k折交叉验证与验证集方法一样，存在分割的随机性，测试集上的结果一定程度上依赖于训练集和测试集是如何分割的，而留一验证法不存在这个问题。

4.问题（略）

要预测模型中变量的测试标准差，可以使用自助法。

使用自助法对响应变量Y进行预测的流程如下：

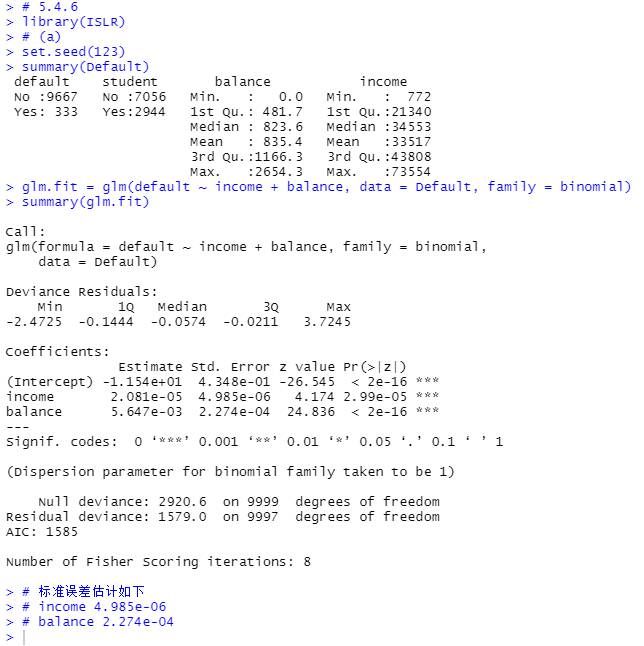
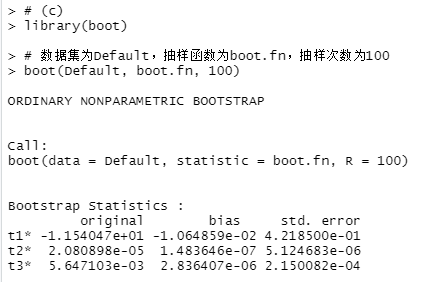
假设数据集包含n个观测值，从数据集中有放回的抽样，创建一个抽样样本，并重复这个过程B次（每个抽样样本包含的抽样数相同）。在每个样本数据集上训练一个模型，然后对该特定的X值产生一个Y的预测值，得到了所有B个样本上对Y的预测值后，就可以使用书中5.8的公式（下图）来估计Y预测结果的标准误差。



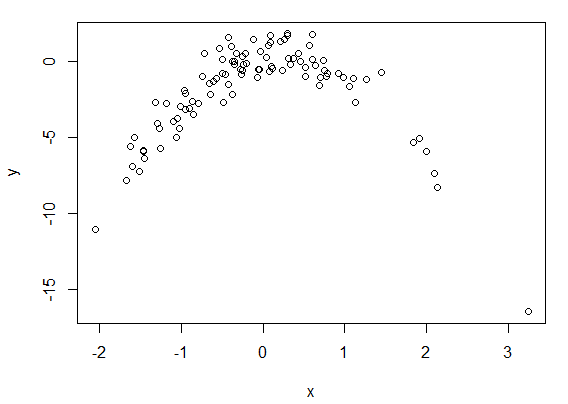
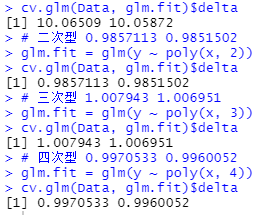
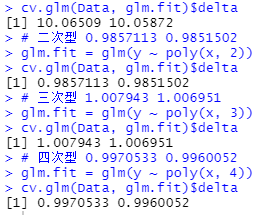
如果原始数据包含 n 个观测值，我们从数据中创建 B 个引导样本（对 n 个观测值进行替换抽样，重复 B 次）。 然后，在每个数据集上，我们将训练一种监督学习方法，并使用它来估计“X 的特定值”。 一旦我们有了这些 B 估计值，我们就可以计算它们的标准差，如 (5.8) 所示。 这样做为我们估计的标准误差提供了引导估计。

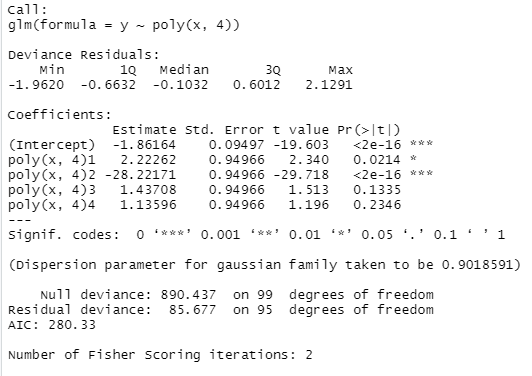
**应用**

6.问题（略）

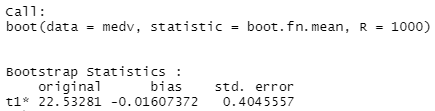
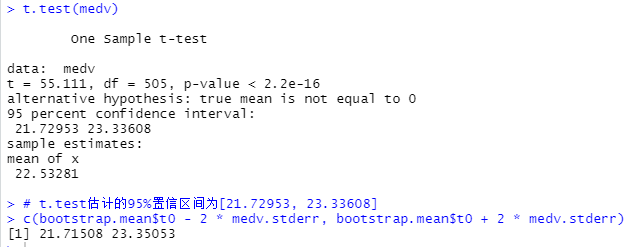
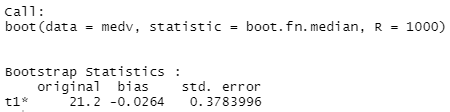
# 5.4.6
  
library(ISLR)
  
（a）问题（略）  
set.seed(123)
  
summary(Default)
  
glm.fit = glm(default ~ income + balance, data = Default, family = binomial)
  
summary(glm.fit)
  
# 标准误差估计如下
  
# income 4.985e-06
  
# balance 2.274e-04
  
  
（b）问题（略）  
boot.fn = function(data, index) {
  
 # 拟合模型, 子集条件为输入的index
  
 glm.fit = glm(default ~ income + balance, data = data, family = binomial, subset = index)
  
 # coef函数返回拟合模型的系数估计
  
 return (coef(glm.fit))
  
}
  
  
（c）问题（略）  
library(boot)
  
# 数据集为Default，抽样函数为boot.fn，抽样次数为100
  
boot(Default, boot.fn, 100)
  
  
（d）问题（略）  
# income balance
  
# glm估计结果 4.985e-06 2.274e-04
  
# boot.fn估计结果 5.124683e-06 2.150082e-04
  
# 发现估计结果相差不大

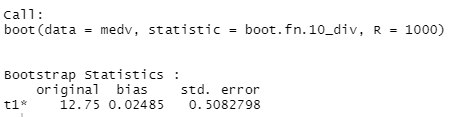
8.问题（略）

# 5.4.8
  
（a）问题（略）  
set.seed(123)
  
y=rnorm(100)
  
x=rnorm(100)
  
y=x-2\*x^2+rnorm(100)
  
# n=100, p=2
  
# y=x-2\*x^2+ε
  
  
（b）问题（略）  
plot(x, y)
  
# 可以看出散点图为下凹曲线，x与y是二次型关系
  
  
（c）问题（略）  
library(boot)
  
# 创建同时包含x和y的数据集
  
Data = data.frame(x, y)
  
set.seed(123)
  
# 线性 10.06509 10.05872
  
glm.fit = glm(y ~ x)
  
cv.glm(Data, glm.fit)$delta
  
# 二次型 0.9857113 0.9851502
  
glm.fit = glm(y ~ poly(x, 2))
  
cv.glm(Data, glm.fit)$delta
  
# 三次型 1.007943 1.006951
  
glm.fit = glm(y ~ poly(x, 3))
  
cv.glm(Data, glm.fit)$delta
  
# 四次型 0.9970533 0.9960052
  
glm.fit = glm(y ~ poly(x, 4))
  
cv.glm(Data, glm.fit)$delta
  
  
# 结果线性的误差最大，二次型误差降低很多
  
# 随着次数再增加，误差不减反增
  
  
（d）问题（略）  
set.seed(123456)
  
# 线性 10.06509 10.05872
  
glm.fit = glm(y ~ x)
  
cv.glm(Data, glm.fit)$delta
  
# 二次型 0.9857113 0.9851502
  
glm.fit = glm(y ~ poly(x, 2))
  
cv.glm(Data, glm.fit)$delta
  
# 三次型 1.007943 1.006951
  
glm.fit = glm(y ~ poly(x, 3))
  
cv.glm(Data, glm.fit)$delta
  
# 四次型 0.9970533 0.9960052
  
glm.fit = glm(y ~ poly(x, 4))
  
cv.glm(Data, glm.fit)$delta
  
  
# 结果一样，因为留一验证法会分割n次数据
  
# 每一次都将不重复的一条数据当作验证集，剩下的当作训练集
  
# 方法不存在随机性，改变随机数种子对留一验证不产生任何影响
  
  
（e）问题（略）  
# 二次型具有最小的LOOCV误差
  
# 原因是最接近y与x的真实关系，拟合效果最好，所以误差最小
  
  
（f）问题（略）  
summary(glm.fit)
  
# 发现二次项的回归系数最显著，而其他项的系数不够显著，与交叉验证的结论相符



9.问题（略）

# 5.4.9
  
（a）问题（略）  
library(MASS)
  
attach(Boston)
  
set.seed(123)
  
medv.mean = mean(medv)
  
medv.mean
  
# 22.53281
  
  
（b）问题（略）  
# medv.stderr medv的标准误差
  
medv.stderr = sd(medv)/sqrt(length(medv))
  
medv.stderr
  
# 使用提示给出的计算公式，得到标准误差为0.4088611
  
  
（c）问题（略）  
library(boot)
  
# 定义均值的抽样函数
  
boot.fn.mean = function(data, index){
  
 # 返回的是数据的总体均值
  
 return (mean(data[index]))
  
}
  
bootstrap.mean=boot(medv, boot.fn.mean, 1000)
  
bootstrap.mean
  
# 结果为0.4045557，和(b)相比相差不大，几乎相同
  
  
（d）问题（略）  
t.test(medv)
  
# t.test估计的95%置信区间为[21.72953, 23.33608]
  
c(bootstrap.mean$t0 - 2 \* medv.stderr, bootstrap.mean$t0 + 2 \* medv.stderr)
  
# 自助法估计的95%置信区间为[21.71508, 23.35053]
  
# 可以看出置信区间基本一致
  
  
（e）问题（略）  
medv.median = median(medv)
  
medv.median
  
# medv的中位数为21.2
  
  
（f）问题（略）  
# 定义中位数的抽样函数
  
boot.fn.median = function(data, index) {
  
 return(median(data[index]))
  
}
  
bootstrap.median=boot(medv, boot.fn.median, 1000)
  
bootstrap.median
  
# medv的中位数标准误差为0.3783996
  
  
（g）问题（略）
  
# quantile函数定义0.1比例的分位数
  
medv.10\_div = quantile(medv, c(0.1))
  
medv.10\_div
  
  
（h）问题（略）  
# 定义十分位的抽样函数
  
boot.fn.10\_div = function(data, index) {
  
 return(quantile(data[index], c(0.1)))
  
}
  
bootstrap.10\_div=boot(medv, boot.fn.10\_div, 1000)
  
bootstrap.10\_div
  
# 十分位数的标准误差为0.5082798



**第六章**

1.问题（略）

（a）问题（略）

最优子集选择模型的效果最好，因为最优子集选择将会对这p个变量拟合个可能的模型，能够找到含有k个变量时训练集RSS最小的模型。而前向和后向逐步选择不一定能够找到对于k个变量来说训练集RSS最小的模型。

（b）问题（略）

不一定。三种方法都有可能达到最小的测试集RSS。因为所有的拟合都是在训练集上进行的，只能得出训练集的RSS。但是最优子集选择模型得到最小的测试集RSS的概率更大，原因是其会考虑所有变量可能的组合。

（c）问题（略）

i.

对。向前逐步选择法中，k+1变量模型是在k变量模型的基础上选择一个对RSS降低效果最好的变量。

ii.

对。向后逐步选择法中，k变量模型是在k+1变量模型的基础上丢弃一个对RSS降低效果最差的变量。

iii.

错。向前和向后逐步选择法的起点不同（分别为空模型和完整模型），选择路径也完全有可能不同，故k变量模型不一定是k+1变量模型的子集。

iv.

错。理由与iii一致。

v.

错。k变量模型和k+1变量模型的变量选择之间没有直接关系，包含于k变量模型中的变量完全有可能不包含于k+1变量模型中，故不能保证k变量模型是k+1变量模型的子集。

2.问题（略）

（a）问题（略）

iii正确，其他错误。

lasso方法最小化的是RSS+，追加了一个惩罚项，不是简单的最小二乘RSS，故灵活性更差。当模型控制一个比较大的λ时，系数收缩，惩罚项降低。此时惩罚项的重要性大于RSS，其大大减少了预测的方差，并小幅提高了预测的偏差，故模型的预测值更加精准。

（b）问题（略）

iii正确，其他错误。

岭回归方法最小化的是RSS+，与lasso相同，都是追加了一个惩罚项，不是简单的最小二乘RSS，故灵活性更差。当模型控制一个比较大的λ时，系数收缩，惩罚项降低。此时惩罚项的重要性大于RSS，其大大减少了预测的方差，并小幅提高了预测的偏差，故模型的预测值更加精准。

（c）问题（略）

ii正确，其他错误。

非线性方法不假定一个线性的模型，具有更高的灵活性。当真实情况更倾向于非线性时，非线性方法拟合的模型偏差很低，方差略高，但是偏差的减少大于方差的增加，整体均方误差比较低，因此预测精度更高。