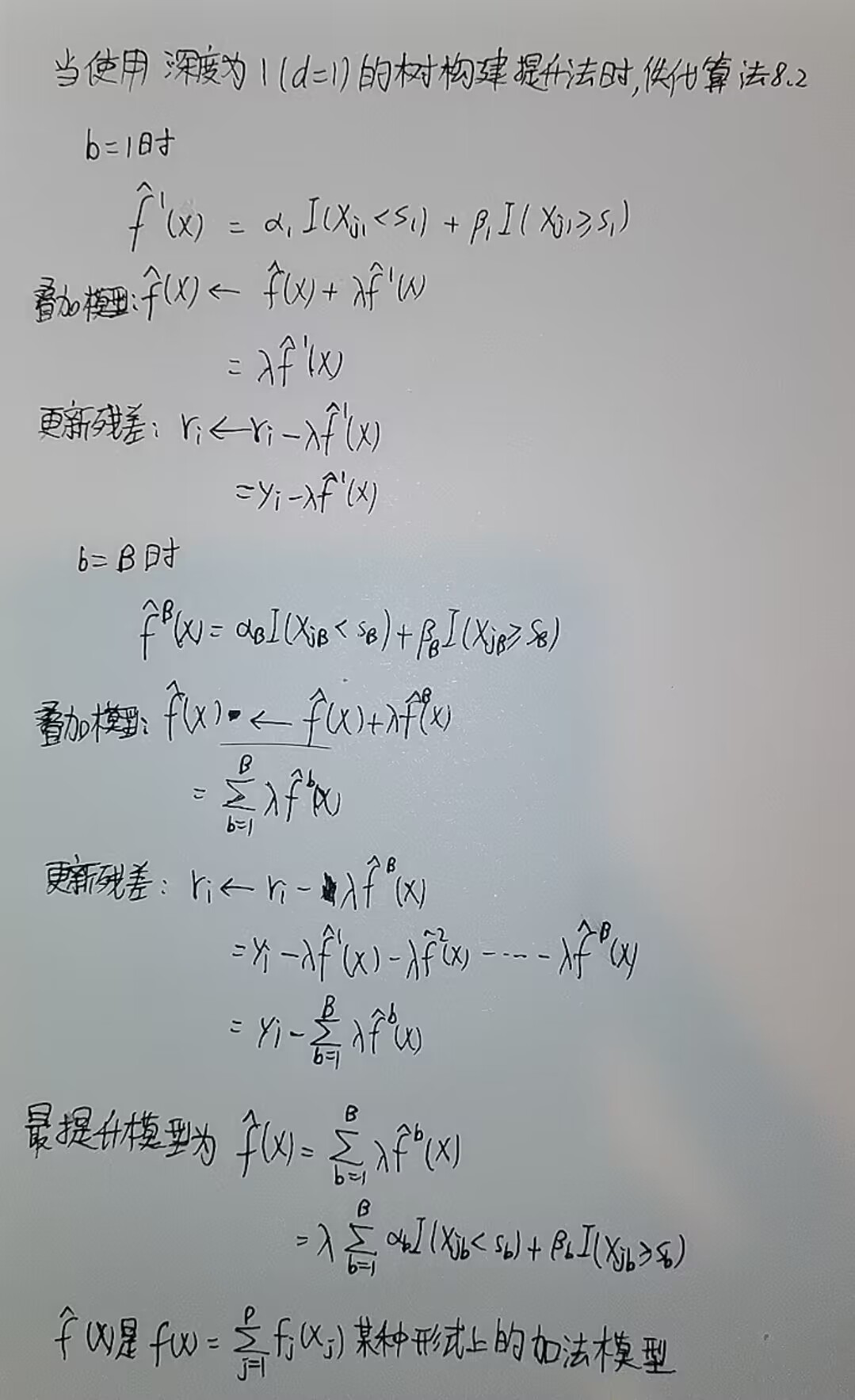
高级统计方法 第9次作业:

序号：1 姓名：杨程瑜 学号：20202241107 班级：网2001

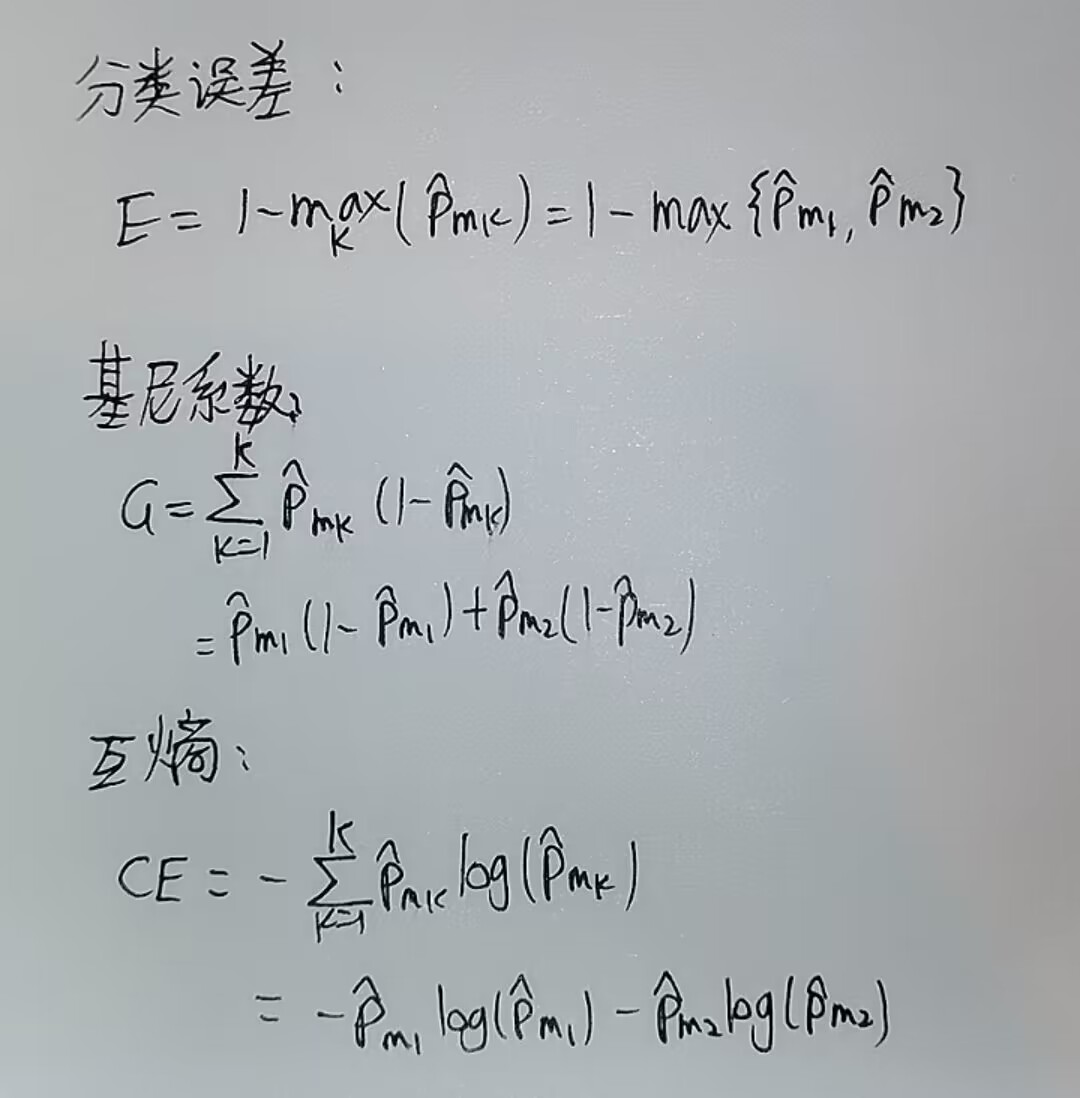
**概念**

2.问题（略）



3.问题（略）

三种指标在p=2的具体情况的化简如下



根据这三个公式，使用R语言绘图

R语言代码如下

library(magrittr)

library(ggplot2)

library(tidyr)

prop\_class\_1 <- seq(0, 1, 0.001)

prop\_class\_2 <- 1 - prop\_class\_1

classification\_error <- 1 - pmax(prop\_class\_1, prop\_class\_2)

gini <- prop\_class\_1\*(1-prop\_class\_1) + prop\_class\_2\*(1-prop\_class\_2)

entropy <- -prop\_class\_1\*log(prop\_class\_1) - prop\_class\_2\*log(prop\_class\_2)

data.frame(prop\_class\_1, prop\_class\_2, classification\_error, gini, entropy) %>%

pivot\_longer(cols = c(classification\_error, gini, entropy), names\_to = "metric") %>%

ggplot(aes(x = prop\_class\_1, y = value, col = factor(metric))) +

geom\_line(size = 1) +

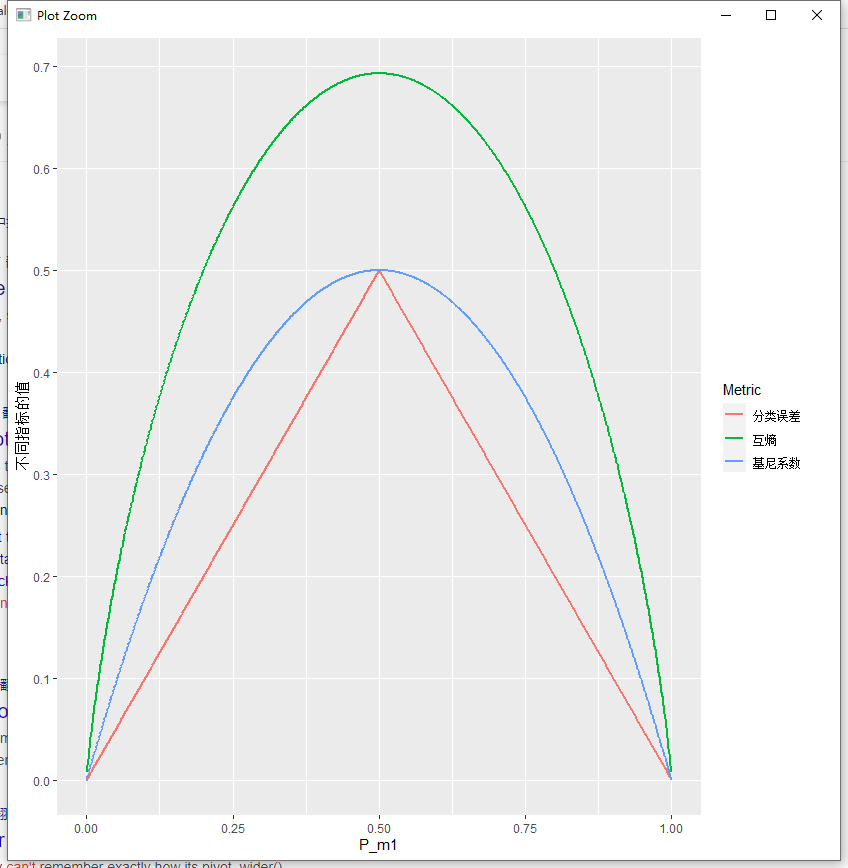
scale\_y\_continuous(breaks = seq(0, 1, 0.1), minor\_breaks = NULL) +

scale\_color\_hue(labels = c("分类误差", "互熵", "基尼系数")) +

labs(col = "Metric",

y = "不同指标的值",

x = "P\_m1")



5.问题（略）

应用多数投票：

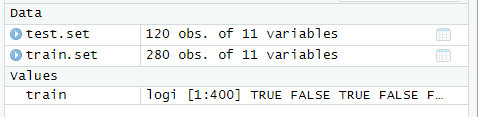
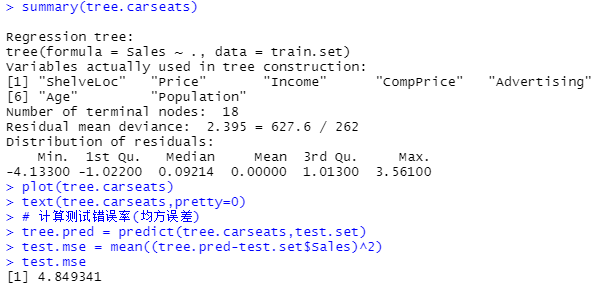
十个预测中统计结果发现P大于0.5的有6个，小于0.5的有4个，多数当选，即综合预测结果为红色

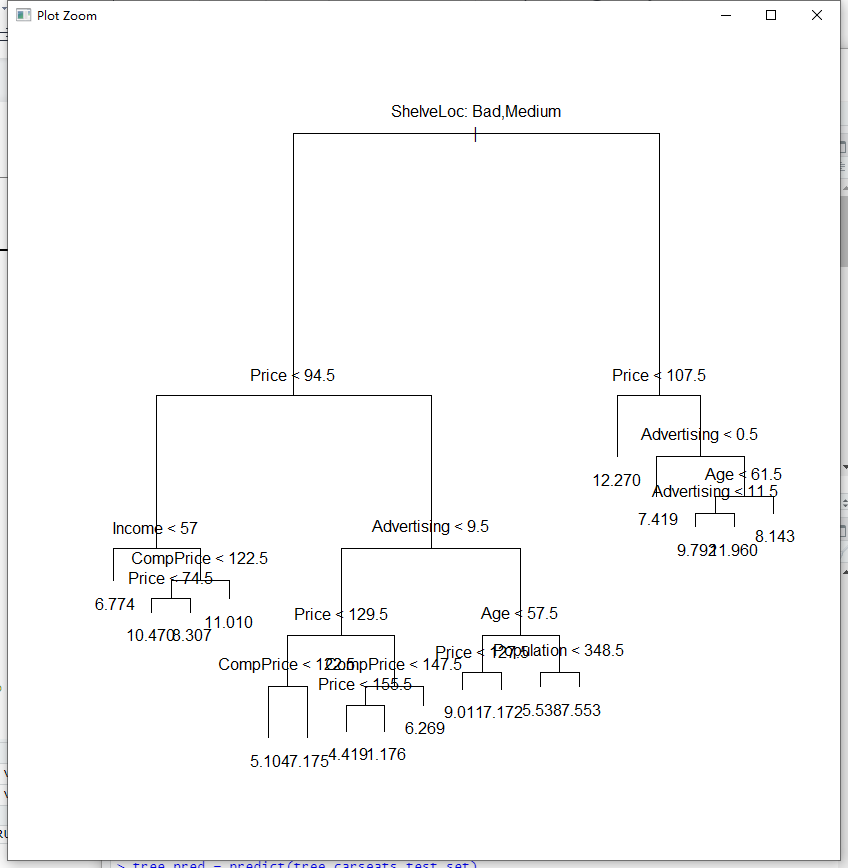
应用平均概率：

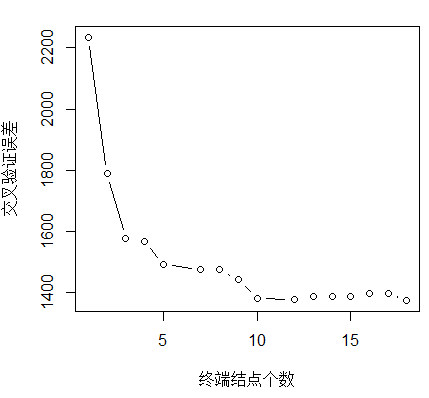
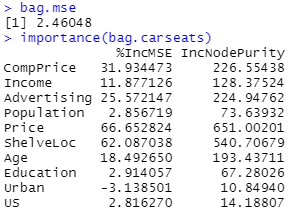
将十个预测概率进行平均：(0.1+0.15+0.2+0.2+0.55+0.6+0.6+0.65+0.7+0.75)/10=0.45<0.5，故综合预测结果为绿色

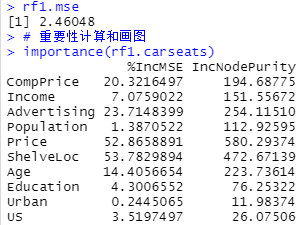
**应用**

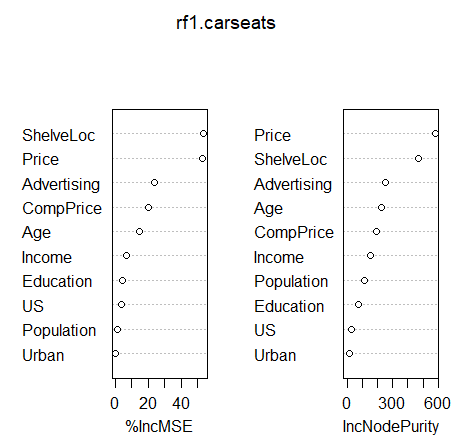
8.问题（略）

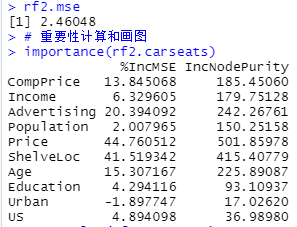
# 8.4.8
  
library(ISLR)
  
library(caTools)
  
library(tree)
  
library(randomForest)
  
（a）问题（略）  
set.seed(123)
  
# 划分测试集与训练集(随机7:3)
  
train = sample.split(Carseats$Sales, SplitRatio = 0.7)
  
train.set = subset(Carseats, train==T)
  
test.set = subset(Carseats, train==F)
  
  
（b）问题（略）  
# Carseats为响应变量的回归决策树
  
tree.carseats = tree(Sales~.,data=train.set)
  
summary(tree.carseats)
  
plot(tree.carseats)
  
text(tree.carseats,pretty=0)
  
# 计算测试错误率(均方误差)
  
tree.pred = predict(tree.carseats,test.set)
  
test.mse = mean((tree.pred-test.set$Sales)^2)
  
test.mse
  
# 回归决策树没有测试错误率，估计得到的均方误差为4.84
  


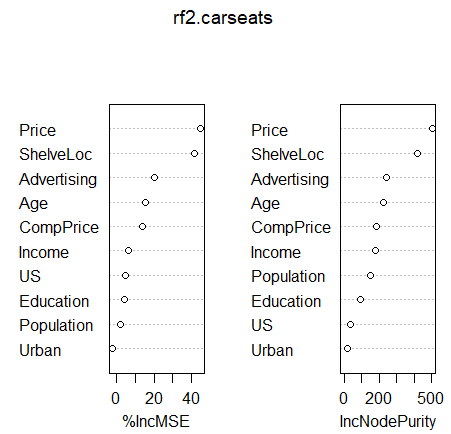
  
（c）问题（略）  
set.seed(1)
  
cv.carseats = cv.tree(tree.carseats)
  
plot(cv.carseats$size,cv.carseats$dev,xlab="终端结点个数",ylab="交叉验证误差",type="b")
  
# 从图中可以看到剪枝能够减小测试错误率，因为交叉验证误差并不是单调递减的
  
prune.carseats = prune.tree(tree.carseats,best=11)
  
tree.pred = predict(prune.carseats,test.set)
  
test.mse = mean((tree.pred-test.set$Sales)^2)
  
test.mse
  
# 最优树在结点数为11处，其测试误差为4.6139，比未剪枝略微更小
  


  
（d）问题（略）  
set.seed(123)
  
# 装袋法
  
bag.carseats = randomForest(Sales~.,data=train.set,mtry=10,importance=T)
  
bag.yhat = predict(bag.carseats,newdata = test.set)
  
bag.mse=mean((bag.yhat-test.set$Sales)^2)
  
bag.mse
  
importance(bag.carseats)
  
  
# 使用装袋法得到的测试均方误差为2.489959
  
# 从结果能够看出Price和ShelveLoc是最重要的变量
  
  
（e）问题（略）  
set.seed(123)
  
# 随机森林
  
# p=m/2
  
rf1.carseats = randomForest(Sales~.,data=train.set,mtry=10/2,importance=T)
  
# 计算mse
  
rf1.yhat = predict(rf1.carseats,newdata = test.set)
  
rf1.mse=mean((bag.yhat-test.set$Sales)^2)
  
rf1.mse
  
# 重要性计算和画图
  
importance(rf1.carseats)
  
varImpPlot(rf1.carseats)
  
  
# p=根号下m
  
rf2.carseats = randomForest(Sales~.,data=train.set,mtry=sqrt(10),importance=T)
  
# 计算mse
  
rf2.yhat = predict(rf1.carseats,newdata = test.set)
  
rf2.mse=mean((bag.yhat-test.set$Sales)^2)
  
rf2.mse
  
# 重要性计算和画图
  
importance(rf2.carseats)
  
varImpPlot(rf2.carseats)
  
  
# p=m/2时，均方误差估计为2.649312
  
# p=根号下m时，均方误差估计为2.46048
  
# 两种情况下Price和ShelveLoc都是最重要的变量

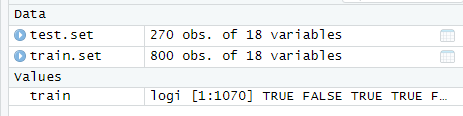
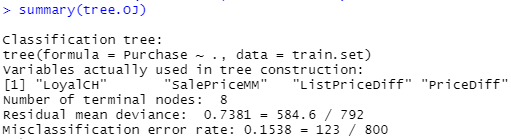
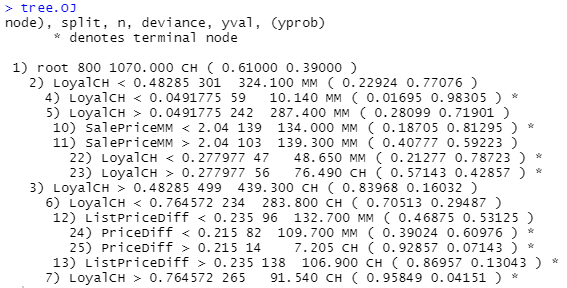
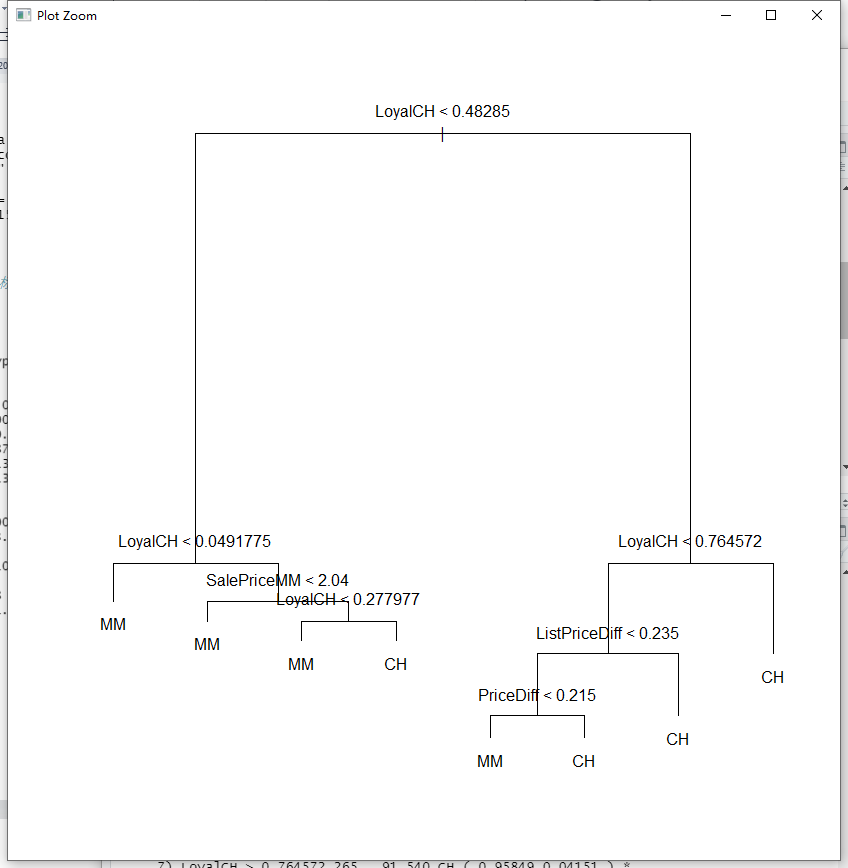
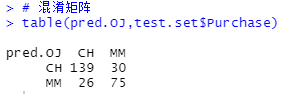


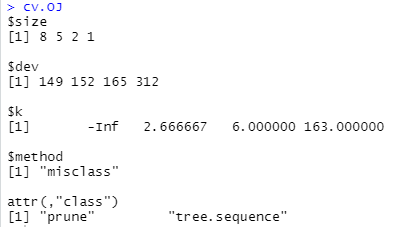
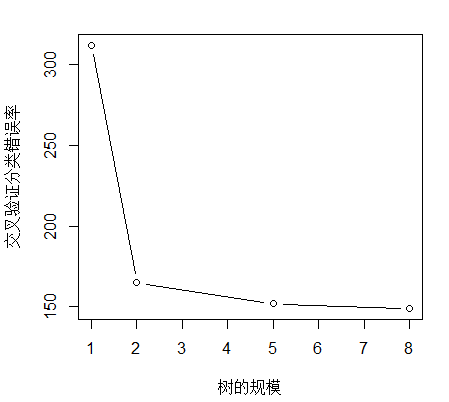






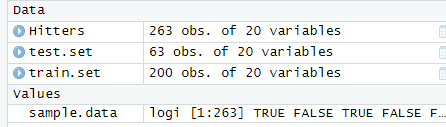
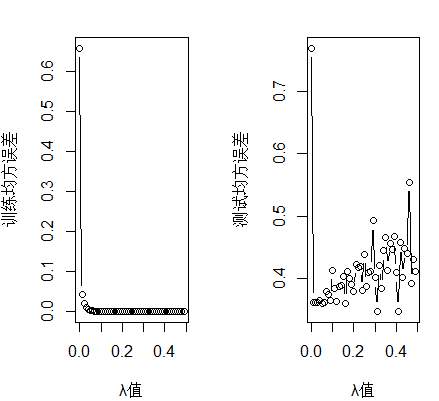
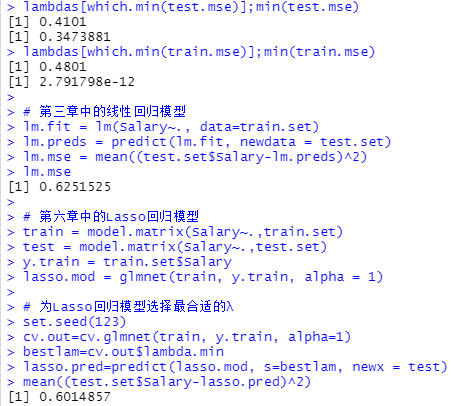
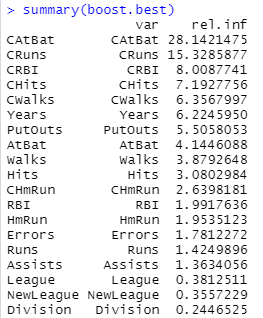
9.问题（略）

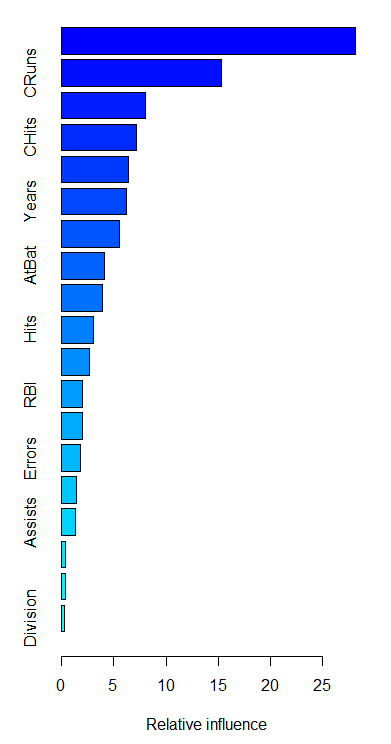
# 8.4.9
  
library(ISLR)
  
library(caTools)
  
library(tree)
  
# 回归树
  
library(gbm)
  
（a）问题（略）  
set.seed(123)
  
# 从1070个数据中随机抽取800个作为训练集
  
train = sample.split(OJ$Purchase, SplitRatio = 800/1070)
  
train.set = subset(OJ,train==T)
  
test.set = subset(OJ,train==F)
  
  
（b）问题（略）  
tree.OJ = tree(Purchase~.,data=train.set)
  
summary(tree.OJ)
  
  
# 残差平均偏差为0.7381，训练错误率为0.1538，有8个终端结点
  
  
（c）问题（略）  
tree.OJ
  
# 观察22终端结点，归入此终端节点的分类标准是LoyalCH<0.277977，共有47个观测值被分类到该结点，该分支分类结果是MM
  
  
（d）问题（略）  
# 树画图
  
plot(tree.OJ)
  
# 加上结点文字信息
  
text(tree.OJ,pretty=0)
  
  
# 从决策树图能够看出，root处使用的是LoyalCH进行分支，其是最关键的变量
  
# 然后还能看出此决策树只使用了4个变量
  
  
（e）问题（略）  
# 用建立的决策树对测试集进行预测
  
pred.OJ = predict(tree.OJ, newdata = test.set, type = "class")
  
# 混淆矩阵
  
table(pred.OJ,test.set$Purchase)
  
# 测试错误率为(30+26)/270=0.2074
  
  
（f）问题（略）  
set.seed(123)
  
# 训练集
  
cv.OJ = cv.tree(tree.OJ, FUN=prune.misclass)
  
cv.OJ

  
（g）问题（略）  
plot(cv.OJ$size, cv.OJ$dev, xlab = "树的规模", ylab = "交叉验证分类错误率", type = "b")
  
  
（h）问题（略）  
# 从cv.OJ以及图中能够看出来8个结点的树交叉验证分类错误率最低
  
  
（i）问题（略）  
# 从图中能够看出来由于交叉验证错误率随属的规模增大而单调递减
  
# 故无法对剪枝后的树进行选择，建立一棵终端结点数为5的树
  
prune.OJ = prune.misclass(tree.OJ,best=5)
  
pred.prune = predict(prune.OJ, newdata = train.set, type = "class")
  
  
（j）问题（略）  
table(pred.prune,train.set$Purchase)
  
# 剪枝后的树训练错误率为(30+101)/800=0.1638
  
# 未剪枝的树训练错误率0.1538
  
# 剪枝后相比剪枝前稍高一些
  
  
（k）问题（略）  
pred.prune = predict(prune.OJ, newdata = test.set, type = "class")
  
table(pred.prune,test.set$Purchase)
  
# 剪枝后的树测试错误率为(18+38)/270=0.2074
  
# 未剪枝的树测试错误率为0.2074
  
# 剪枝后与剪枝前几乎一样



10.问题（略）

# 8.4.10
  
rm(list=ls())
  
library(magrittr)
  
library(tidyr)
  
library(glmnet)
  
（a）问题（略）  
# 批量使用drop\_na函数处理数据集
  
Hitters = Hitters %>% drop\_na(Salary)
  
# 对数变换
  
Hitters$Salary = log(Hitters$Salary)
  
  
（b）问题（略）  
set.seed(123)
  
# 划分训练集(200)和测试集(63)
  
sample.data = sample.split(Hitters$Salary, SplitRatio = 200/263)
  
train.set = subset(Hitters, sample.data==T)
  
test.set = subset(Hitters, sample.data==F)
  
  
（c）问题（略）  
# 使用提升法迭代1000次，产生1000棵树
  
# λ序列
  
lambdas = seq(0.0001,0.5,0.01)
  
train.mse = rep(NA,length(lambdas))
  
test.mse = rep(NA,length(lambdas))
  
set.seed(4)
  
for (i in lambdas){
  
 boost.Hitters = gbm(Salary~., data=train.set,distribution = "gaussian", n.trees = 1000,
  
 interaction.depth = 4, shrinkage = i)
  
 yhat.train = predict(boost.Hitters,newdata = train.set, n.trees = 1000)
  
 train.mse[which(i==lambdas)] = mean((yhat.train-train.set$Salary)^2)
  
  
 yhat.test = predict(boost.Hitters,newdata = test.set, n.trees = 1000)
  
 test.mse[which(i==lambdas)] = mean((yhat.test-test.set$Salary)^2)
  
}
  
  
（d）问题（略）  
par(mfrow=c(1,2))
  
plot(lambdas,train.mse,type="b",xlab="λ值", ylab="训练均方误差")
  
plot(lambdas,test.mse,type="b",xlab="λ值", ylab="测试均方误差")
  
  
（e）问题（略）  
# mse最小时的λ值
  
lambdas[which.min(test.mse)];min(test.mse)
  
lambdas[which.min(train.mse)];min(train.mse)
  
  
# 第三章中的线性回归模型
  
lm.fit = lm(Salary~., data=train.set)
  
lm.preds = predict(lm.fit, newdata = test.set)
  
lm.mse = mean((test.set$Salary-lm.preds)^2)
  
lm.mse
  
  
# 第六章中的Lasso回归模型
  
train = model.matrix(Salary~.,train.set)
  
test = model.matrix(Salary~.,test.set)
  
y.train = train.set$Salary
  
lasso.mod = glmnet(train, y.train, alpha = 1)
  
  
# 为Lasso回归模型选择最合适的λ
  
set.seed(123)
  
cv.out=cv.glmnet(train, y.train, alpha=1)
  
bestlam=cv.out$lambda.min
  
lasso.pred=predict(lasso.mod, s=bestlam, newx = test)
  
mean((test.set$Salary-lasso.pred)^2)
  
  
（f）问题（略）  
# 使用gbm函数进行提升法
  
boost.best = gbm(Salary~., data=train.set, distribution = "gaussian", n.trees = 1000,
  
 interaction.depth = 4, shrinkage = 0.01)
  
summary(boost.best)
  
  
# 从图中以及可以看出CAtBat、CRuns以及CWalks是最重要的三个变量
  


  
（g）问题（略）  
bag.Hitters = randomForest(Salary~.,train.set,mtry=19,importance=T)
  
bag.pred = predict(bag.Hitters,newdata = test.set)
  
mean((test.set$Salary-bag.pred)^2)
  
  
# 装袋法测试均方误差为0.3284043

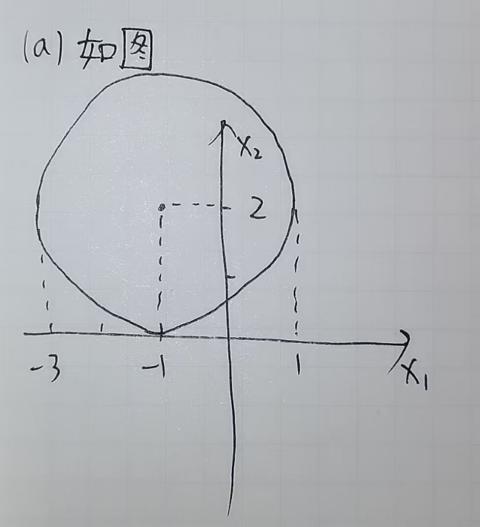


**第九章**

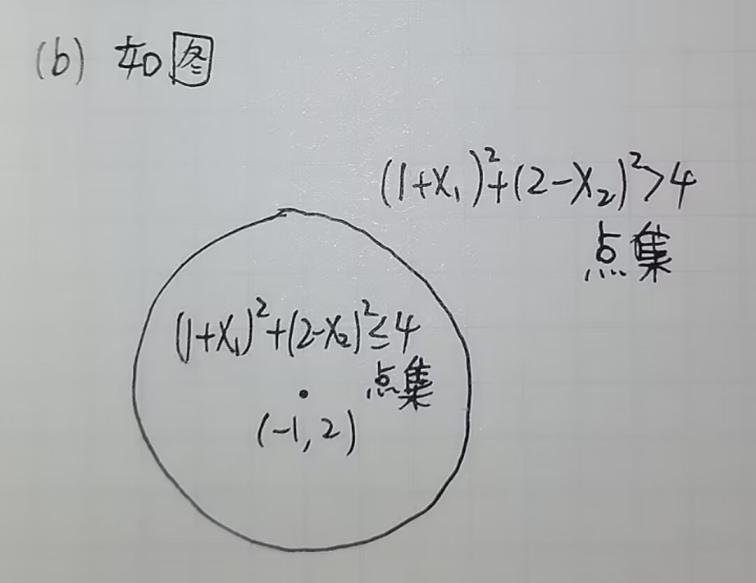
**概念**

2.问题（略）

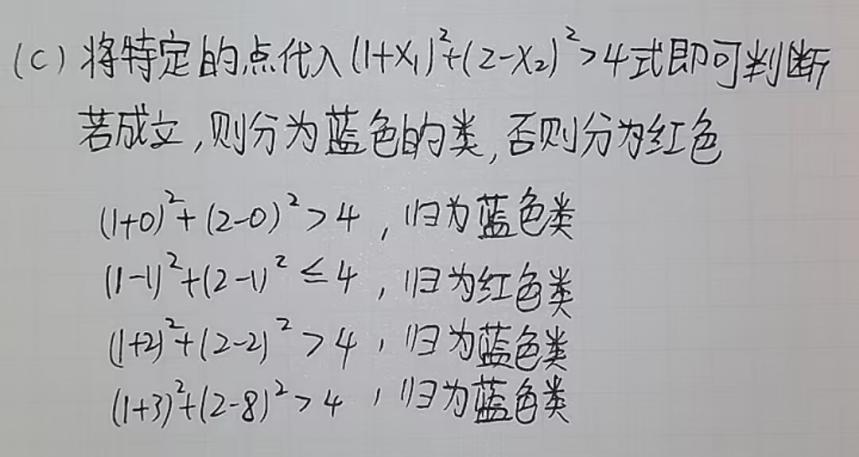
（a）问题（略）



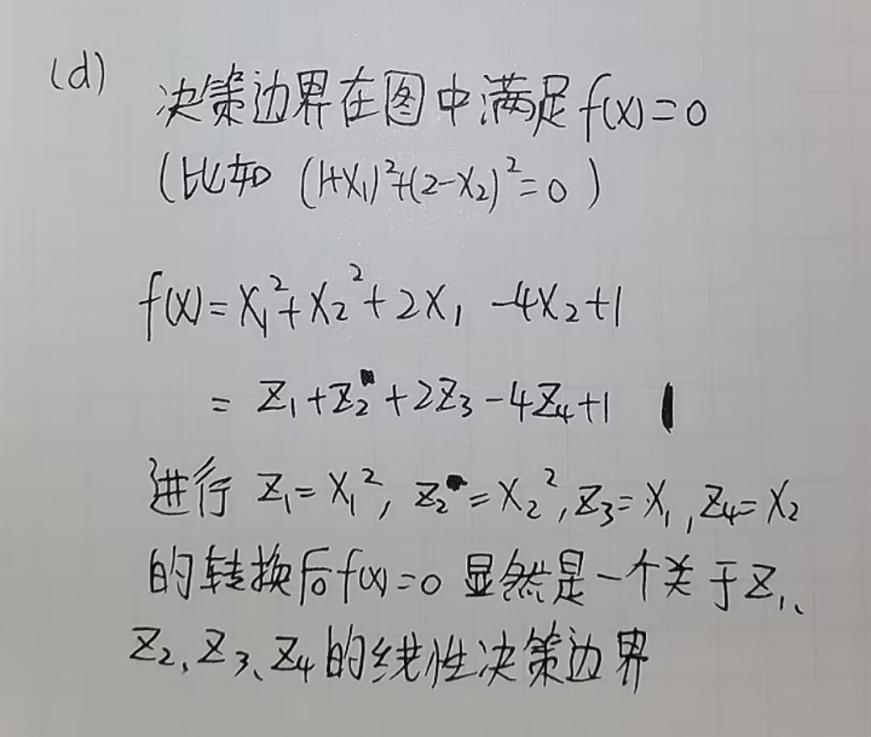
（b）问题（略）



（c）问题（略）

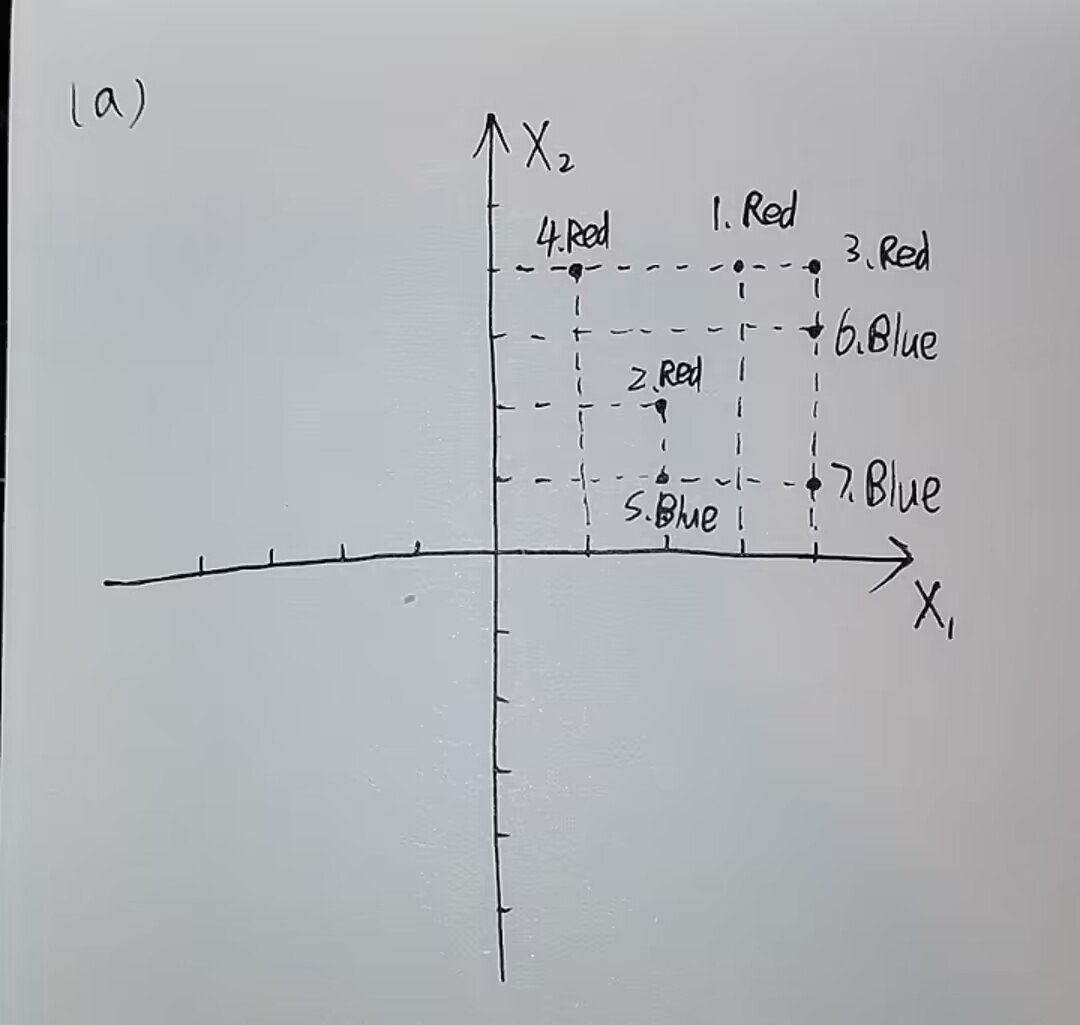


（d）问题（略）

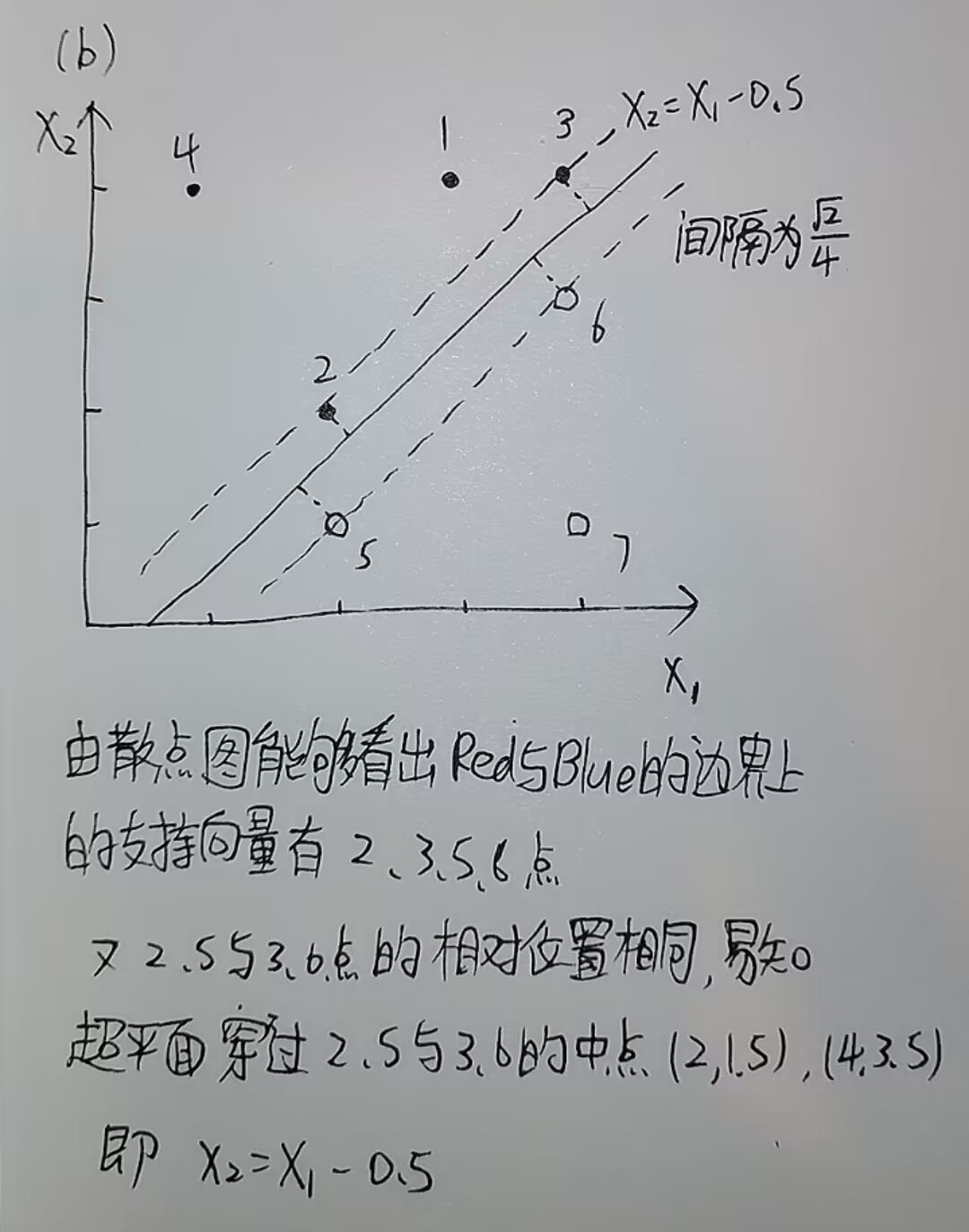


3.问题（略）

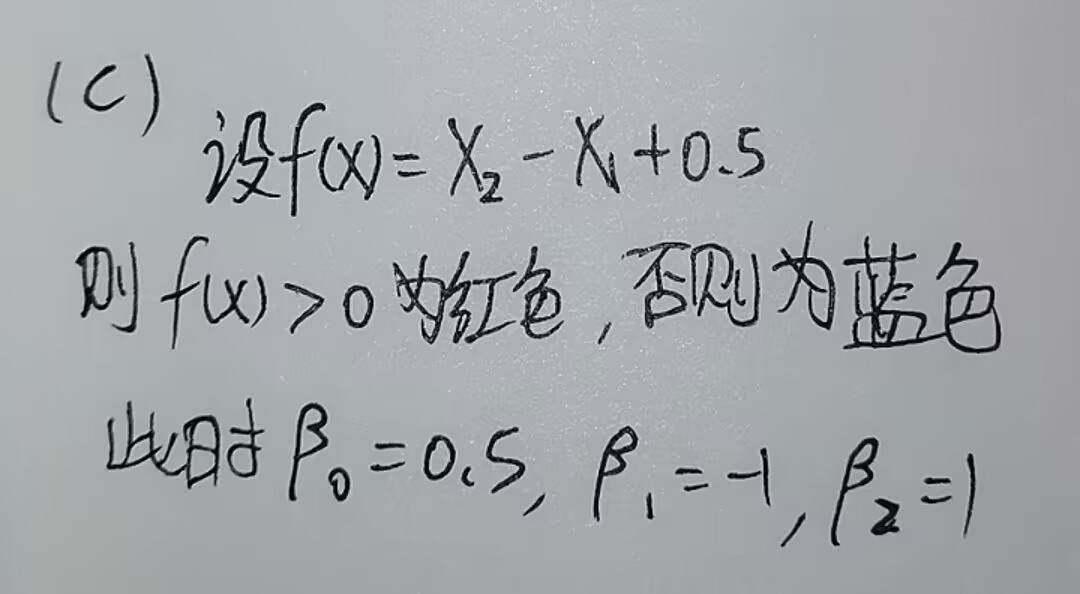
（a）问题（略）



（b）问题（略）



（c）问题（略）



（d）问题（略）

如（b）问图中所示，已给出

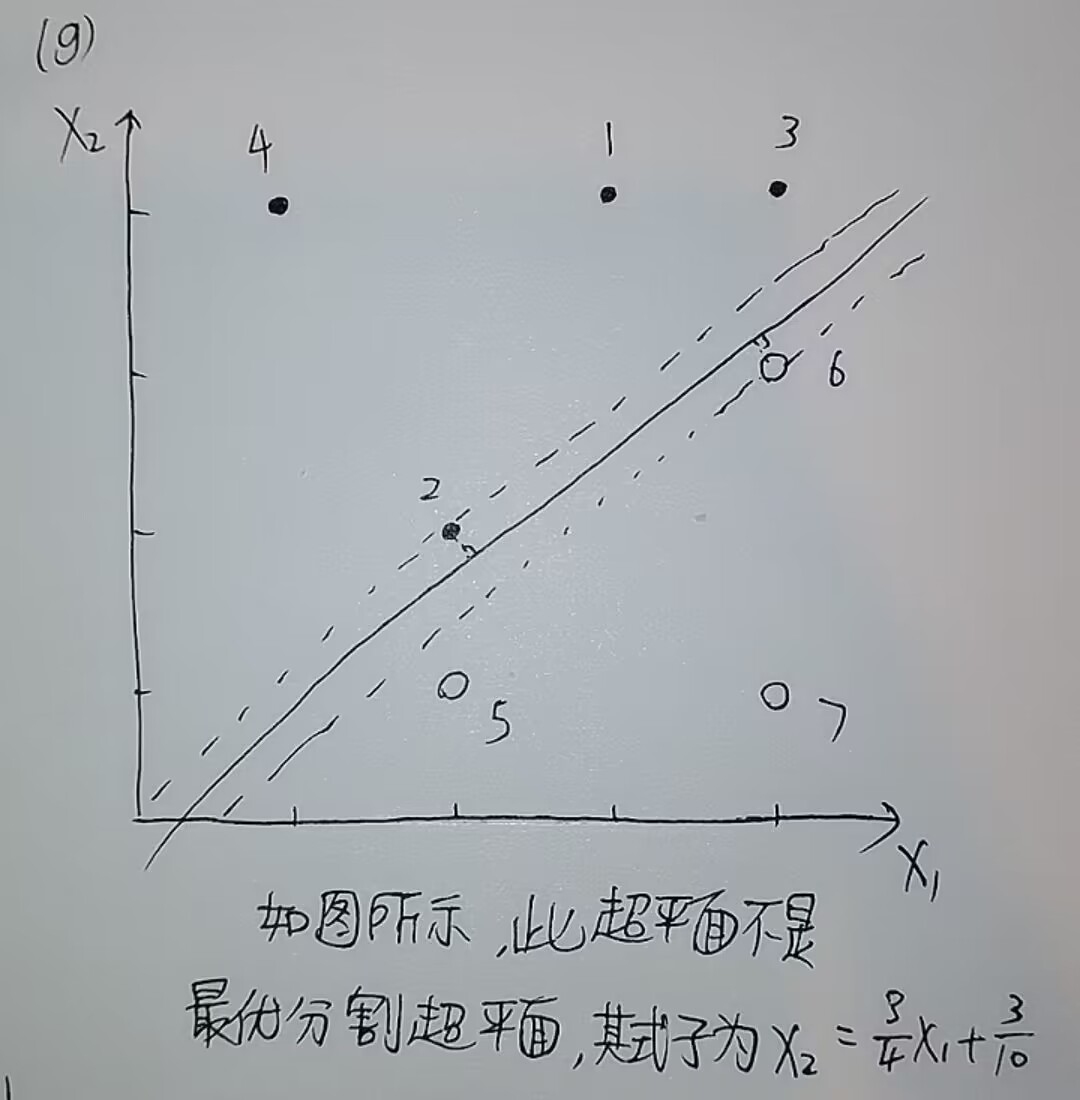
（e）问题（略）

此最大间隔分类器的支持向量有点2、3、5、6

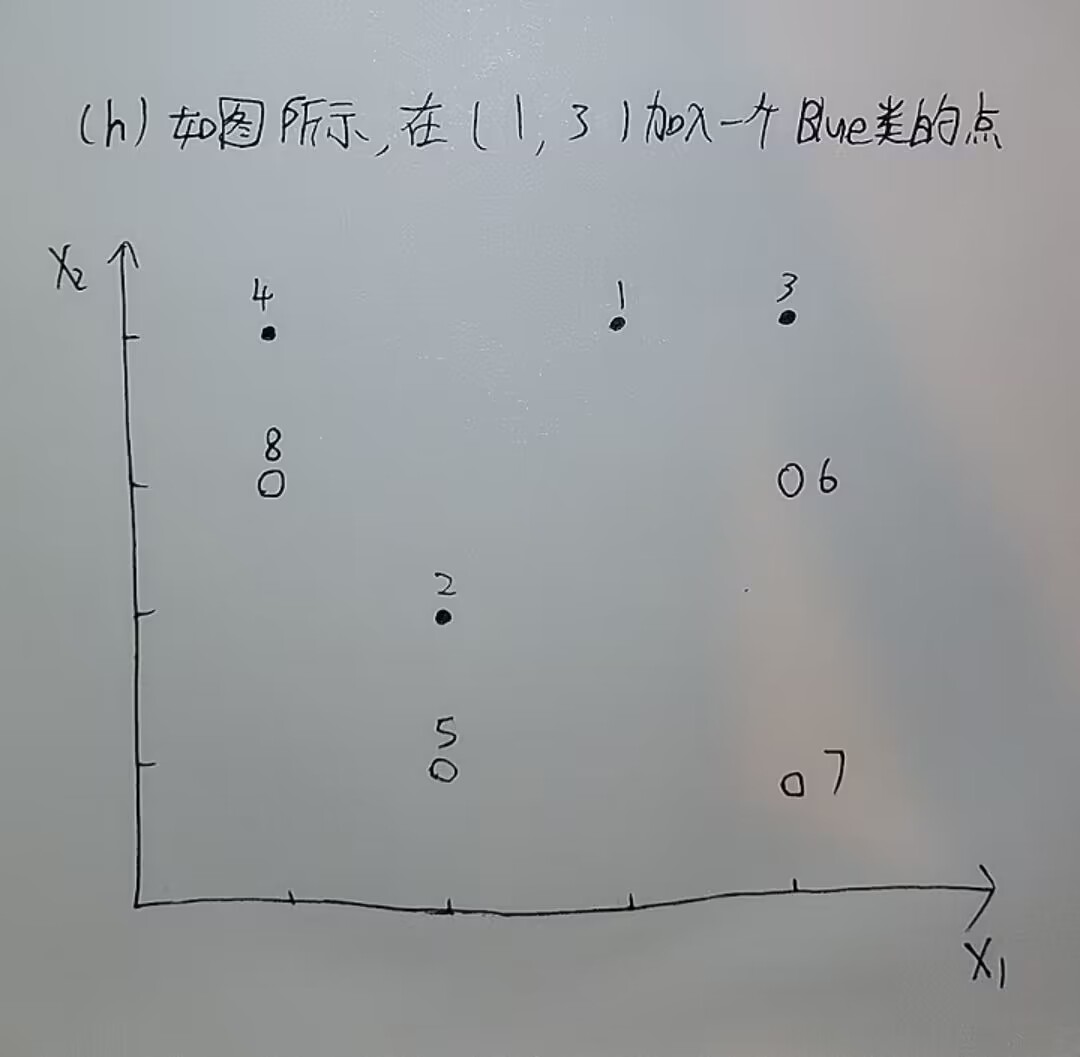
（f）问题（略）

第七个点距离超平面的间隔有一定距离，并不位于间隔上或间隔边缘的位置。因此第七个观测点轻微移动并不会影响最大间隔超平面

（g）问题（略）



（h）问题（略）

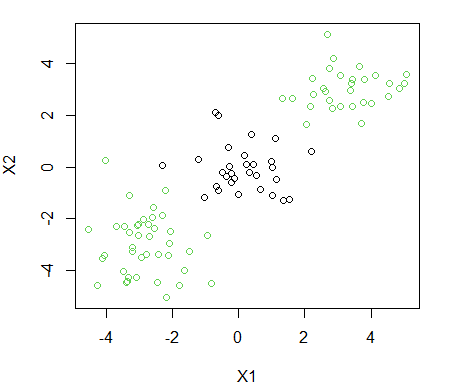


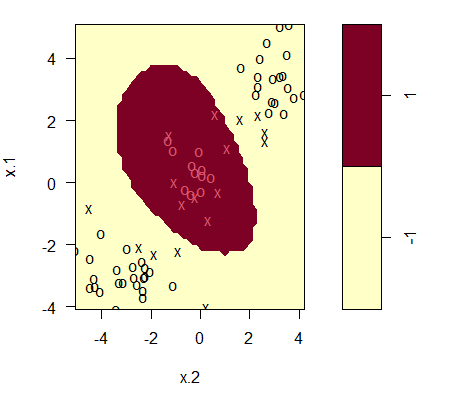
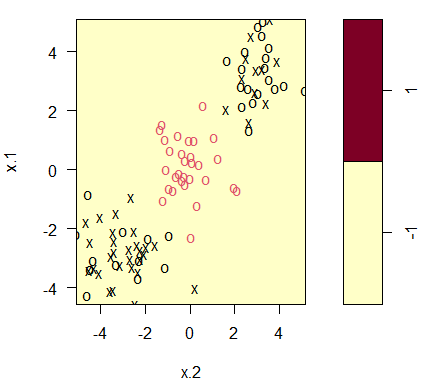
**应用**

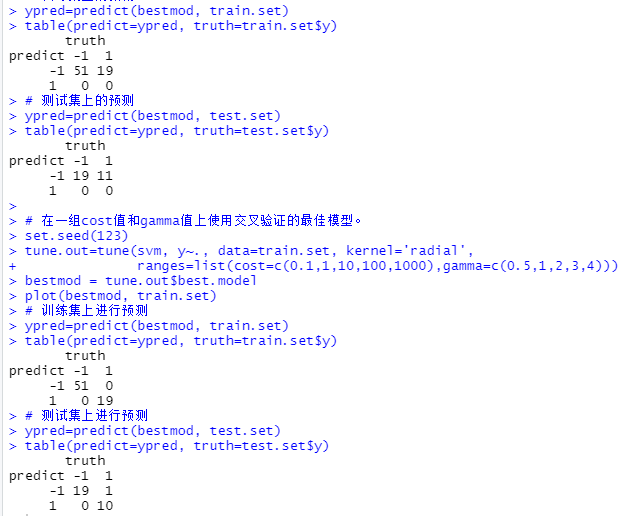
4.问题（略）

# 9.7.4
  
rm(list=ls())
  
library(ISLR)
  
library(caTools)
  
library("e1071")

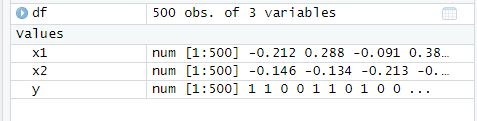
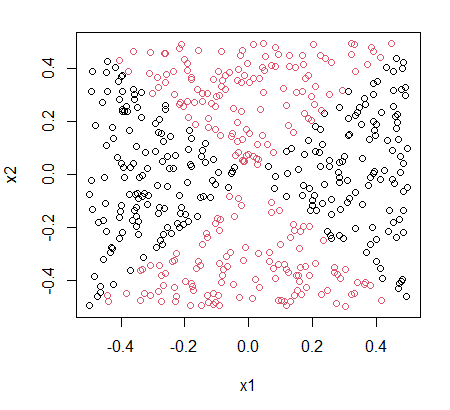
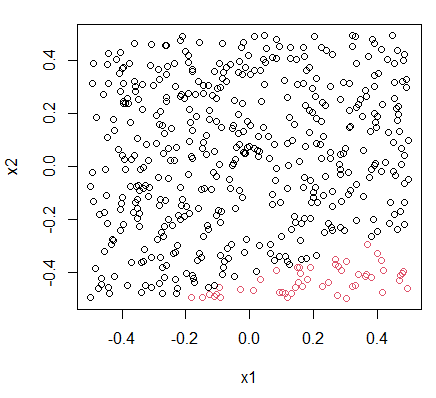
set.seed(123)
  
# 构造数据
  
x=matrix(rnorm(100\*2), ncol=2)
  
y=c(rep(-1,70), rep(1,30))
  
x[1:30,]=x[1:30,]+3.3
  
x[31:70,]=x[31:70,]-3
  
dat=data.frame(x=x, y=as.factor(y))
  
# 训练集和测试集
  
sample.data = sample.split(dat$x.1, SplitRatio = 0.7)
  
train.set = subset(dat, sample.data==T)
  
test.set = subset(dat, sample.data==F)
  
  
plot(x,col=(2-y), xlab='X1', ylab='X2', main='Dataset with non-linear separation')
  
  
# 寻找最好的模型
  
set.seed(123)
  
tune.out=tune(svm,y~.,data = train.set,kernel='linear',
  
 ranges=list(cost=c(0.001,0.01,0.1,1,5,10,100)))
  
bestmod=tune.out$best.model
  
plot(bestmod, dat)
  
  
# 训练集上的预测
  
ypred=predict(bestmod, train.set)
  
table(predict=ypred, truth=train.set$y)
  
# 测试集上的预测
  
ypred=predict(bestmod, test.set)
  
table(predict=ypred, truth=test.set$y)
  
  
# 在一组cost值和gamma值上使用交叉验证的最佳模型。
  
set.seed(123)
  
tune.out=tune(svm, y~., data=train.set, kernel='radial',
  
 ranges=list(cost=c(0.1,1,10,100,1000),gamma=c(0.5,1,2,3,4)))
  
bestmod = tune.out$best.model
  
plot(bestmod, train.set)
  
# 训练集上进行预测
  
ypred=predict(bestmod, train.set)
  
table(predict=ypred, truth=train.set$y)
  
# 测试集上进行预测
  
ypred=predict(bestmod, test.set)
  
table(predict=ypred, truth=test.set$y)
  
  
# 具有线性核的 svm 错误地分类了 19/70 的训练观察和 11/30 的测试观察
  
# 具有径向内核的 svm 正确分类所有训练观察和仅错误分类了1/30的测试观察

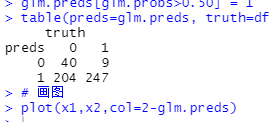
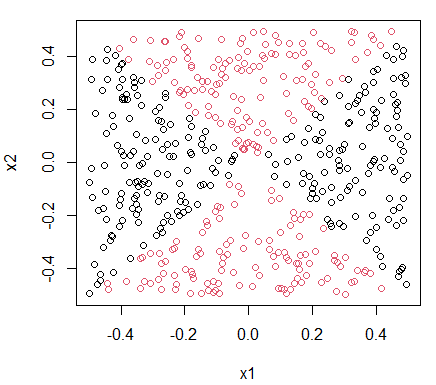
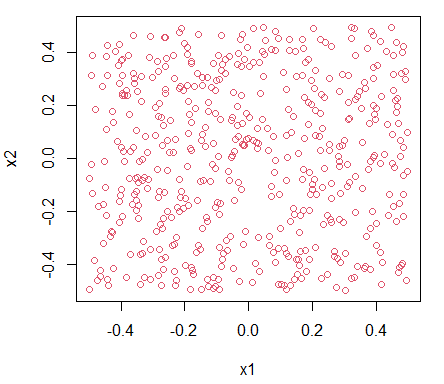
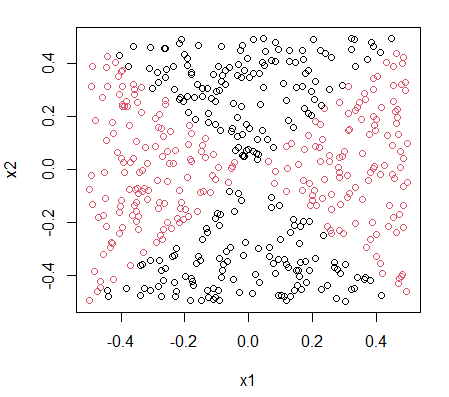




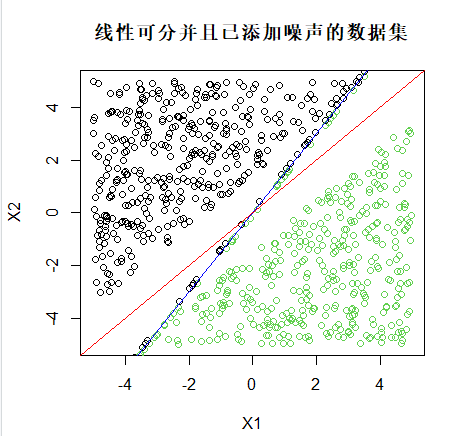
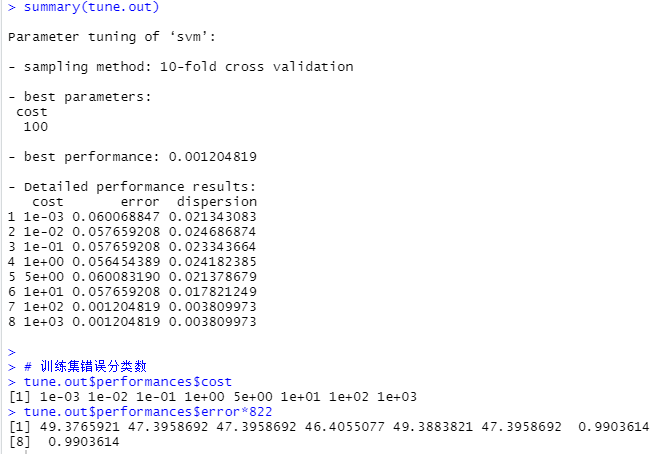
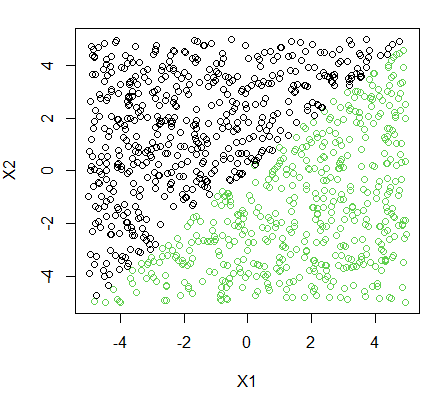


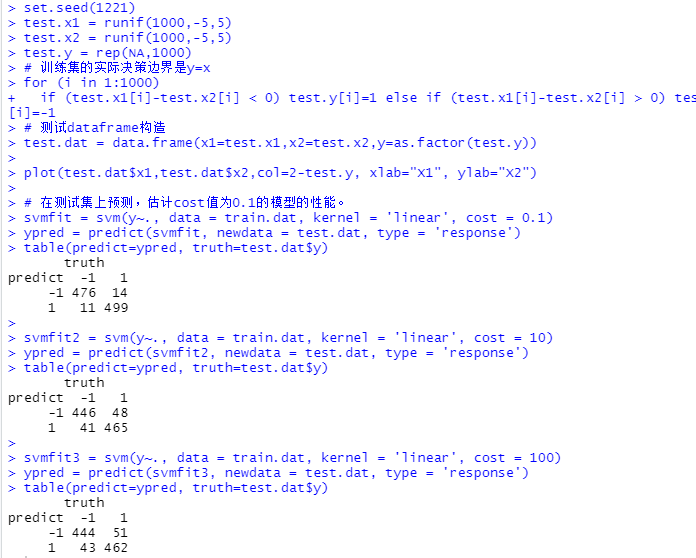
5.问题（略）

# 9.7.5
  
rm(list=ls())
  
library(ISLR)
  
library(caTools)
  
library("e1071")
  
（a）问题（略）  
set.seed(123)
  
x1 = runif(500)-0.5
  
x2 = runif(500)-0.5
  
y = 1\*(x1^2-x2^2 > 0)
  
df=data.frame(x1=x1, x2=x2, y=as.factor(y))
  
  
（b）问题（略）  
plot(x1,x2,col = (2 - y))
  
  
（c）问题（略）  
glm.fit = glm(y~x1+x2, data=df, family = 'binomial')
  
  
（d）问题（略）  
# 预测
  
glm.probs = predict(glm.fit, newdata=df, type = 'response')
  
glm.preds = rep(0,500)
  
glm.preds[glm.probs>0.50] = 1
  
table(preds=glm.preds, truth=df$y)
  
# 画图
  
plot(x1,x2,col=2-glm.preds)
  


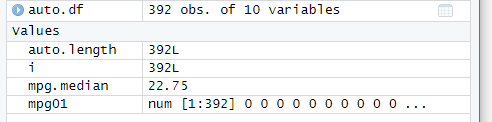
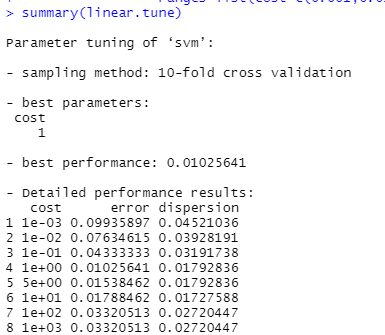
  
（e）问题（略）  
glm.fit = glm(y~I(x1^2)+I(x2^2), data = df, family = 'binomial')
  
  
（f）问题（略）  
glm.probs = predict(glm.fit, newdata = df, type = 'response')
  
glm.preds = rep(0,500)
  
glm.preds[glm.probs>0.5] = 1
  
table(preds=glm.preds, truth=df$y)
  
plot(x1,x2,col=2-glm.preds)
  
  
# X1和X2的二次变换非常完美的分离了，结果是所有观察都分类正确
  
# 即分类错误率为0
  
  
（g）问题（略）  
# 最佳模型
  
tune.out=tune(svm,y~.,data = df,kernel='linear',
  
 ranges=list(cost=c(0.001,0.01,0.1,1,5,10,100)))
  
bestmod=tune.out$best.model
  
  
# 进行预测
  
ypred=predict(bestmod, newdata=df, type='response')
  
table(predict=ypred, truth=df$y)
  
plot(x1,x2,col=ypred)
  
  
（h）问题（略）  
# 拟合径向内核函数的svm
  
tune.out=tune(svm, y~., data=df, kernel='radial',
  
 ranges=list(cost=c(0.1,1,10,100,1000),gamma=c(0.5,1,2,3,4)))
  
bestmod=tune.out$best.model
  
  
# 进行预测
  
ypred=predict(bestmod, newdata=df, type='response')
  
table(predict=ypred, truth=df$y)
  
plot(x1,x2,col=ypred)
  
  
（i）问题（略）  
# 使用线性项和非线性项的逻辑回归模型分别与线性和径向内核完美拟合
  
# 即结果为具有非线性项的逻辑回归和具有径向内核的SVM优于使用线性项的模型
  
# 并且结果显示这些模型在预测方面准确性非常高，证实逻辑斯蒂回归和SVM是相似的方法

6.问题（略）

# 9.7.6
  
rm(list=ls())
  
library(ISLR)
  
library(caTools)
  
library("e1071")
  
  
（a）问题（略）  
set.seed(123)
  
x1 = runif(1000,-5,5)
  
x2 = runif(1000,-5,5)
  
x = cbind(x1,x2)
  
  
y = rep(NA,1000)
  
# 将abline(1.5,1)以上的点分类为 1，将abline(-1.5,1)以下的点分类为-1，其余为0
  
# 删除归类为0的点将创建一个更广泛的数据集
  
# 实际决策边界是一条y=x的线，即abline(0,1)
  
for (i in 1:1000)
  
 if (x[i,2]-x[i,1] > 1.5) y[i]=1 else if (x[i,2]-x[i,1] < -1.5) y[i]=-1 else y[i]=0
  
  
# 结合x和y并删除所有y=0的行
  
x = cbind(x,y)
  
x = x[x[,3]!=0,]
  
  
plot(x[,1],x[,2],col=2-x[,3], xlab="X1", ylab="X2",xlim = c(-5,5), ylim = c(-5,5))
  
abline(0,1, col="red")
  
abline(1.5,1)
  
abline(-1.5,1)
  
abline(h=0,v=0)
  
  
# 沿线 y=1.5x(+-0.1) 生成要用作噪声的随机点
  
x.noise = matrix(NA,100,3)
  
x.noise[,1] = runif(100,-5,5)
  
# 前50个点的Y坐标值
  
x.noise[1:50,2] = (1.5\*x.noise[1:50,1])-0.1
  
x.noise[51:100,2] = (1.5\*x.noise[51:100,1])+0.1
  
# 所有噪声观测值的类值
  
x.noise[,3] = c(rep(-1,50), rep(1,50))
  
  
plot(x[,1],x[,2],col=2-x[,3], xlab='X1', ylab='X2',
  
 ylim = c(-5,5),xlim = c(-5,5),
  
 main="线性可分并且已添加噪声的数据集")
  
par(new = TRUE)
  
plot(x.noise[,1],x.noise[,2],col=2-x.noise[,3], axes=F,
  
 xlab="", ylab="", ylim = c(-5,5), xlim = c(-5,5))
  
# 噪声
  
abline(0,1.5,col="blue")
  
# 实际决策边界
  
abline(0,1,col="red")
  
  
（b）问题（略）  
x = rbind(x,x.noise)
  
train.dat = data.frame(x1=x[,1],x2=x[,2], y=as.factor(x[,3]))
  
  
# 具有各种成本值的线性 SVM 模型。
  
tune.out=tune(svm,y~.,data=train.dat,kernel='linear',
  
 ranges=list(cost=c(0.001,0.01,0.1,1,5,10,100,1000)))
  
summary(tune.out)
  
  
# 训练集错误分类数
  
tune.out$performances$cost
  
tune.out$performances$error\*822
  
  
# 随着成本的增加，更少的训练观察被错误分类
  
# 交叉验证错误与训练集上的错误的数量直接相关
  
# 即较低的交叉验证错误意味着更少的训练错误
  
  
（c）问题（略）  
# 生成一个新的合适的测试集
  
set.seed(1221)
  
test.x1 = runif(1000,-5,5)
  
test.x2 = runif(1000,-5,5)
  
test.y = rep(NA,1000)
  
# 训练集的实际决策边界是y=x
  
for (i in 1:1000)
  
 if (test.x1[i]-test.x2[i] < 0) test.y[i]=1 else if (test.x1[i]-test.x2[i] > 0) test.y[i]=-1
  
# 测试dataframe构造
  
test.dat = data.frame(x1=test.x1,x2=test.x2,y=as.factor(test.y))
  
  
plot(test.dat$x1,test.dat$x2,col=2-test.y, xlab="X1", ylab="X2")
  
  
# 在测试集上预测，估计cost值为0.1的模型的性能。
  
svmfit = svm(y~., data = train.dat, kernel = 'linear', cost = 0.1)
  
ypred = predict(svmfit, newdata = test.dat, type = 'response')
  
table(predict=ypred, truth=test.dat$y)
  
  
svmfit2 = svm(y~., data = train.dat, kernel = 'linear', cost = 10)
  
ypred = predict(svmfit2, newdata = test.dat, type = 'response')
  
table(predict=ypred, truth=test.dat$y)
  
  
svmfit3 = svm(y~., data = train.dat, kernel = 'linear', cost = 100)
  
ypred = predict(svmfit3, newdata = test.dat, type = 'response')
  
table(predict=ypred, truth=test.dat$y)
  
  
# 由混淆矩阵可以看出来，cost值为0.1的模型在测试集上的表现要好于cost值更高的模型
  
# 在训练集上，cost值高于100的模型没有出现错误分类错误
  


  
（d）问题（略）  
# cost值高的模型的间隔非常小，因此间隔里或越过决策边界的支持向量会更少
  
# 因此模型能正确分类大多数训练观察
  
# 虽然在训练集上表现非常好，但是偏差小方差大，故在测试集上表现不佳
  
  
# cost值较低的模型将有更大的间隔，因此间隔里或越过决策边界的支持向量会更多
  
# 这意味着模型错误地分类了更多的训练观察
  
# 虽然在训练集上表现不佳，但是偏差大方差小，故在测试集上表现不错
  
  
# 从这些结果中，我们可以发现cost值较高的模型过拟合了训练数据，而cost值较低的模型则相反

7.问题（略）

# 9.7.7
  
rm(list=ls())
  
library(ISLR)
  
library(caTools)
  
library("e1071")
  
  
（a）问题（略）  
set.seed(123)
  
# 创建二分变量
  
auto.length = length(Auto$mpg)
  
mpg.median = median(Auto$mpg)
  
mpg01 = rep(NA,auto.length)
  
# 如果汽车的 mpg 高于中位数，则为1级，否则为0级
  
# 结果存储在 mpg01 变量中
  
for (i in 1:auto.length) if (Auto$mpg[i] > mpg.median) mpg01[i]=1 else mpg01[i]=0
  
# 数据Dataframe
  
auto.df = Auto
  
auto.df$mpg01 = as.factor(mpg01)
  
  
（b）问题（略）  
# 使用线性SVM对mpg01进行预测
  
linear.tune=tune(svm,mpg01~.,data=auto.df,kernel='linear',
  
 ranges=list(cost=c(0.001,0.01,0.1,1,5,10,100,1000)))
  
summary(linear.tune)
  
  
# 发现训练交叉验证误差随着cost值的增加而减少，在cost值为1时最小，此后逐渐增加增加
  
  
（c）问题（略）  
set.seed(123)
  
# 具有各种 gamma 和成本值的径向内核。
  
radial.tune=tune(svm, mpg01~., data=auto.df, kernel='radial',
  
 ranges=list(cost=c(0.1,1,10,100,1000),gamma=c(0.5,1,2,3,4)))
  
radial.tune$best.parameters
  
radial.tune$best.performance
  
  
# cost值为1和gamma=0.5的径向模型的训练交叉验证误差最低，但是比线性模型高约4倍
  
  
# 不同次数的多项式核函数
  
set.seed(123)
  
poly.tune = tune(svm, mpg01~., data=auto.df, kernel='polynomial',
  
 ranges=list(cost=c(0.1,1,10,100,1000), degree=c(1,2,3,4,5)))
  
poly.tune$best.parameters
  
poly.tune$best.performance

