527 Final

Xinyu Gao

June 29, 2019

data and package

```
# setwd('C:/Users/46541/Desktop/myhw/527/project')
library(pls)
library(class)
library(dplyr)
library(plyr)
library(broom)
library(mgcv)
library(rpart)
library(rpart.plot)
library(glmnet)
library(caret)
library(MASS)
library(gam)
library( ISLR)
library(leaps)
```

Problem 2

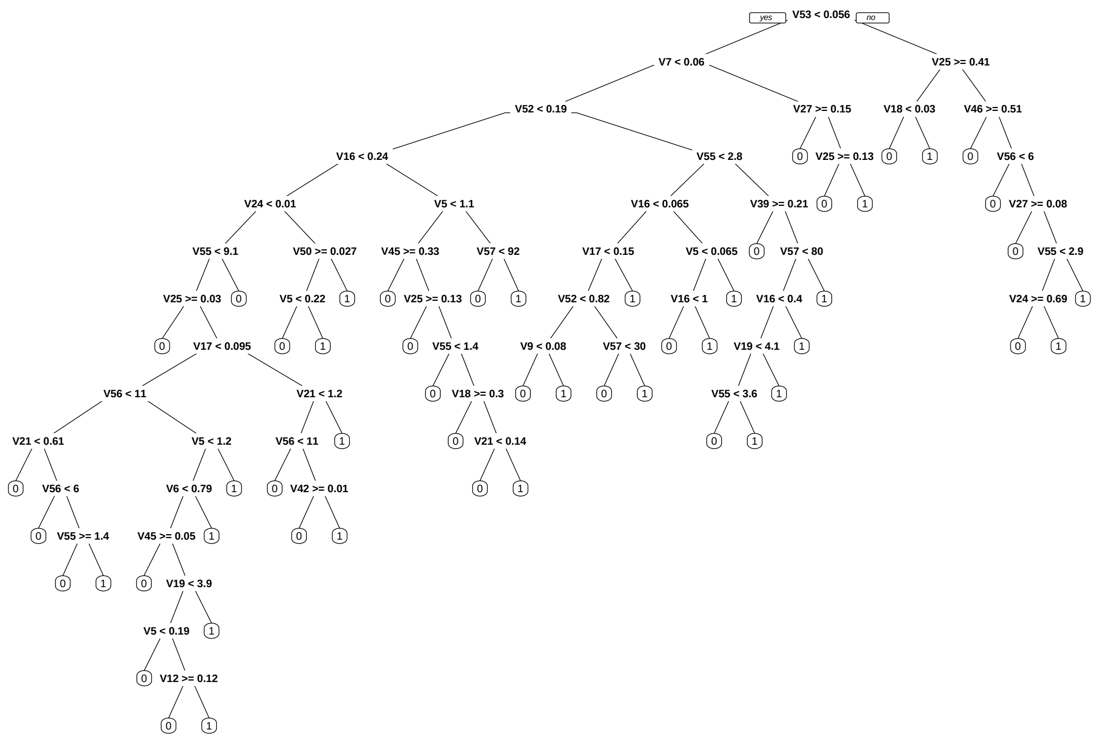
a

```
spam_all = read.csv('spam.txt', header = FALSE, sep = ' ')
indicator = read.csv('spam_traintest.txt', header = FALSE, sep = ' ')
# divide data by indicator
spam_train <- spam_all[which(indicator==0),]
spam_test <- spam_all[which(indicator==1),]

spam_train$V58<- factor(spam_train$V58)
dtree<-rpart(V58~.,data=spam_train,method="class", parms=list(split="gini"),cp=0)
printcp(dtree)
```

```
##
## Classification tree:
## rpart(formula = V58 ~ ., data = spam_train, method = "class",
##     parms = list(split = "gini"), cp = 0)
##
## Variables actually used in tree construction:
##  [1] V12 V16 V17 V18 V19 V21 V24 V25 V27 V39 V42 V45 V46 V5  V50 V52 V53
## [18] V55 V56 V57 V6  V7  V9
##
## Root node error: 1218/3065 = 0.39739
##
```

```
## n= 3065
##
##            CP nsplit rel error  xerror     xstd
## 1  0.49343186      0   1.00000 1.00000 0.022243
## 2  0.14449918      1   0.50657 0.50657 0.018226
## 3  0.04187192      2   0.36207 0.36289 0.015968
## 4  0.02791461      4   0.27833 0.28982 0.014510
## 5  0.01724138      5   0.25041 0.26683 0.013994
## 6  0.01149425      6   0.23317 0.24877 0.013567
## 7  0.00821018      7   0.22167 0.23727 0.013283
## 8  0.00574713      8   0.21346 0.22824 0.013054
## 9  0.00410509     10   0.20197 0.22167 0.012883
## 10 0.00369458     11   0.19787 0.22167 0.012883
## 11 0.00328407     13   0.19048 0.22085 0.012861
## 12 0.00246305     14   0.18719 0.22085 0.012861
## 13 0.00218938     20   0.17241 0.23071 0.013117
## 14 0.00164204     23   0.16585 0.22496 0.012969
## 15 0.00102627     31   0.15271 0.22906 0.013075
## 16 0.00082102     44   0.13629 0.22906 0.013075
## 17 0.00054735     47   0.13383 0.22989 0.013096
## 18 0.00041051     50   0.13218 0.23645 0.013262
## 19 0.00000000     52   0.13136 0.23892 0.013324
```

```r
prp(dtree)
```




```r
spam_fitted <- predict(dtree,newdata=spam_train,type = 'class')
err_train <- 1- sum(spam_fitted == spam_train$V58)/length(spam_fitted)
paste("The error rate on the training set is:",err_train)
```

```
## [1] "The error rate on the training set is: 0.0522022838499184"
```

```r
spam_test_val <-predict(dtree, newdata = spam_test,type='class')
err_test <- 1- sum(spam_test_val == spam_test$V58)/length(spam_test_val)
paste("The error rate on the test set is:",err_test)
```

```
## [1] "The error rate on the test set is: 0.08984375"
```

b

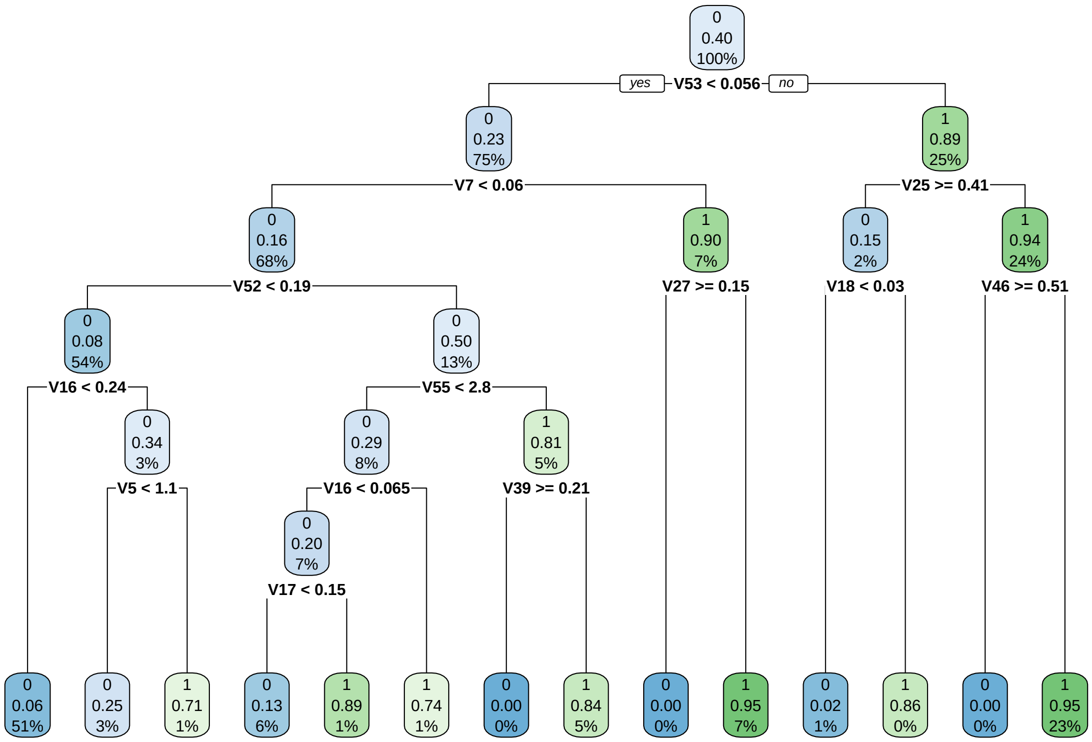
```r
cp <- which.min(dtree$cptable[,"xstd"])
paste("The optimal tunning parameter is:",dtree$cptable[,"CP"][cp])
```

```
## [1] "The optimal tunning parameter is: 0.00328407224958949"
```

```r
tree_pruned<-prune(dtree,cp=dtree$cptable[which.min(dtree$cptable[,"xstd"]),"CP"])
printcp(tree_pruned)
```

```
##
## Classification tree:
## rpart(formula = V58 ~ ., data = spam_train, method = "class",
##     parms = list(split = "gini"), cp = 0)
##
## Variables actually used in tree construction:
##  [1] V16 V17 V18 V25 V27 V39 V46 V5  V52 V53 V55 V7
##
## Root node error: 1218/3065 = 0.39739
##
## n= 3065
##
##           CP nsplit rel error  xerror     xstd
## 1  0.4934319      0   1.00000 1.00000 0.022243
## 2  0.1444992      1   0.50657 0.50657 0.018226
## 3  0.0418719      2   0.36207 0.36289 0.015968
## 4  0.0279146      4   0.27833 0.28982 0.014510
## 5  0.0172414      5   0.25041 0.26683 0.013994
## 6  0.0114943      6   0.23317 0.24877 0.013567
## 7  0.0082102      7   0.22167 0.23727 0.013283
## 8  0.0057471      8   0.21346 0.22824 0.013054
## 9  0.0041051     10   0.20197 0.22167 0.012883
## 10 0.0036946     11   0.19787 0.22167 0.012883
## 11 0.0032841     13   0.19048 0.22085 0.012861
```

```r
rpart.plot::rpart.plot(tree_pruned)
```




```r
pruned_fitted <- predict(tree_pruned, newdata = spam_train,type='class')
err_train_pruned <- 1- sum(pruned_fitted== spam_train$V58)/length(pruned_fitted)
paste("The error rate on the training set is:",err_train_pruned)
```

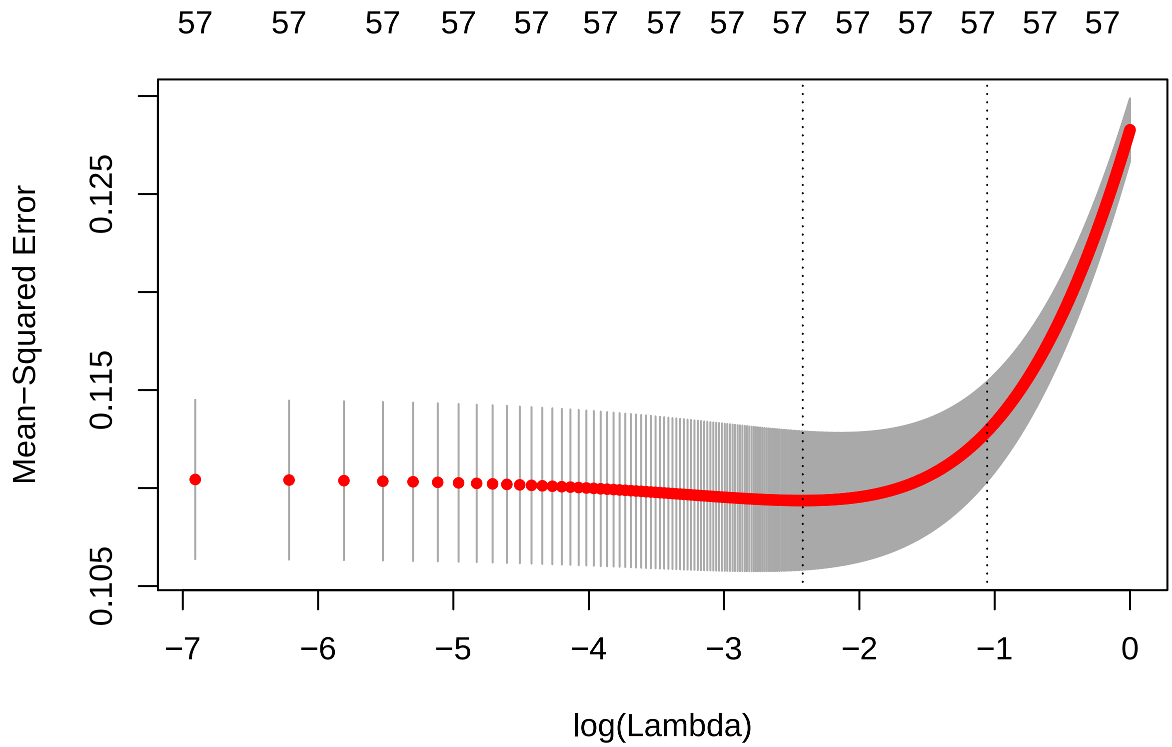
```
## [1] "The error rate on the training set is: 0.0756933115823817"
```

```r
pruned_test_val <-predict(tree_pruned, newdata = spam_test,type='class')
err_test_pruned <- 1- sum(pruned_test_val == spam_test$V58)/length(pruned_test_val)
paste("The error rate on the test set is:",err_test_pruned)
```

```
## [1] "The error rate on the test set is: 0.08984375"
```

Problem 3

```r
spam_all = read.csv('spam.txt', header = FALSE, sep = ' ')
indicator = read.csv('spam_traintest.txt', header = FALSE, sep = ' ')
spam_train <- spam_all[which(indicator==0),]
spam_test <- spam_all[which(indicator==1),]
trainx <- spam_train[,-58]
trainx <- as.matrix(trainx)
trainy <- spam_train[,58]
trainy <- as.matrix(trainy)
lambdas = seq(0, 1, by = 0.001)
set.seed(123)
cv_fit = cv.glmnet(trainx, trainy, alpha = 0,lambda = lambdas,nfolds = 10)
plot(cv_fit)
```




```
opt_lambda = cv_fit$lambda.min
paste("The opyimal lambda is:",opt_lambda)
```

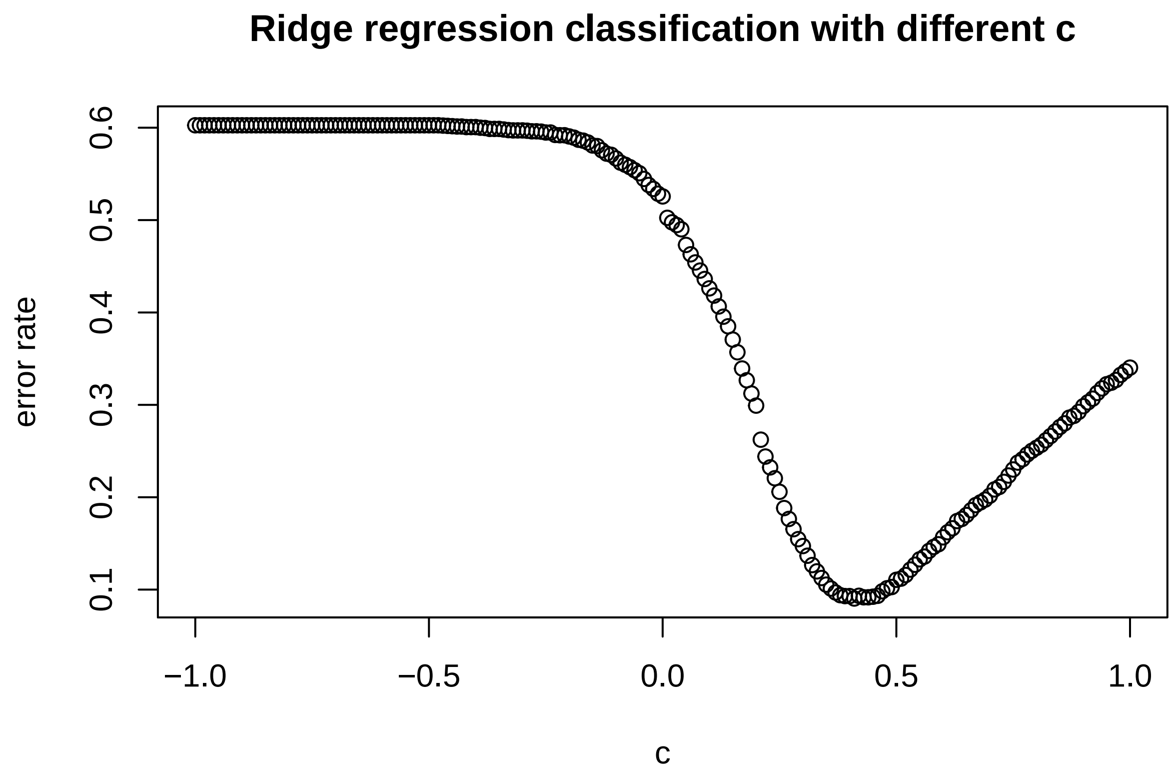
```
## [1] "The opyimal lambda is: 0.089"
```

```
fit_ridge = glmnet(trainx, trainy, alpha = 0, standardize =TRUE,lambda =opt_lambda)
```

pick optimal c

```
spam_train <- spam_all[which(indicator==0),]
train_y_val <- spam_train[,58]
train_y_val <- as.matrix(train_y_val)

spam_test <- spam_all[which(indicator==1),]
testx <- spam_test[,-58]
testx <- as.matrix(testx)
testy <-spam_test[,58]
testy <- as.matrix(testy)

fitted_ridge <- predict(fit_ridge, newx = trainx )
c <- seq(-1,1,by=0.01)

pick_c <- function (c,fit_val, true_class){
    class1 <- which(fit_val > c)
    fit_val[class1] <- 1
    fit_val[-class1] <- 0
    err =1- sum(fit_val == true_class)/length(fit_val)
    return (err)
}
```

```r
opt_c <-function(c){
  pick_c(c, fitted_ridge,train_y_val )
}

c_all <- sapply(c, opt_c)
plot(c,c_all,xlab="c", ylab = "error rate",main="Ridge regression classification with different c" )
```




```r
c_opt <- which(min(c_all)==c_all)
paste("The optimal c is:", c[c_opt])
```

```
## [1] "The optimal c is: 0.41"
```

```r
# resubstitution error
paste("The resubstitution error for ridge regression is:", min(c_all))
```

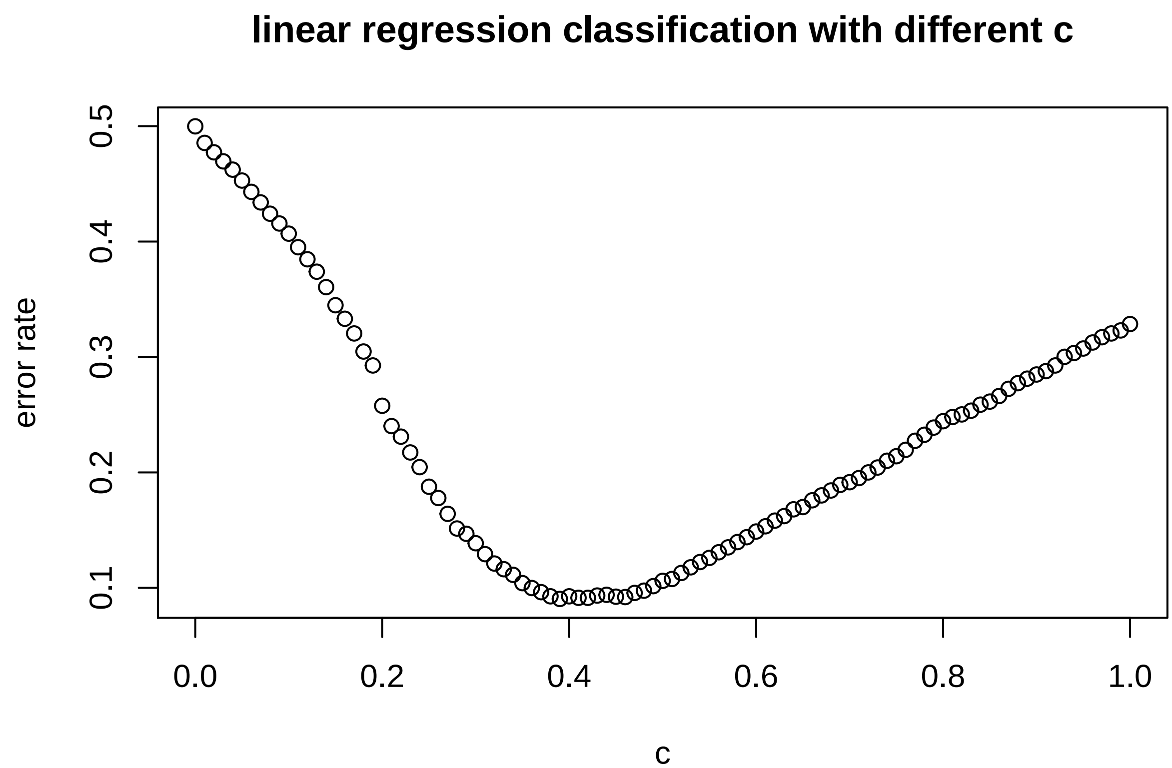
```
## [1] "The resubstitution error for ridge regression is: 0.0903752039151713"
```

```r
# prediction error
predict_ridge <- predict(fit_ridge, newx = testx )
predict_ridge[predict_ridge>c[c_opt]] <- 1
predict_ridge[predict_ridge<=c[c_opt]] <- 0
predict_err <- 1-(sum(predict_ridge == testy))/length(predict_ridge)
paste("The prediction error for ridge regression is:",predict_err )
```

```
## [1] "The prediction error for ridge regression is: 0.0930989583333334"
```

compared with linear regression

```
fit_ols <- lm(V58~.,data = spam_train)
fitted_ols <- predict(fit_ols, newdata = spam_train)

opt_c_ols <-function(c){
  pick_c(c, fitted_ols,spam_train$V58 )
}
c_ols <- seq(0,1,by=0.01)

c_ols_all <-sapply(c_ols,opt_c_ols)
plot(c_ols,c_ols_all,xlab="c",ylab="error rate",main="linear regression classification with different c
```




```
c_ols_opt <- c_ols[which(min(c_ols_all) == c_ols_all)]
paste("The optimal c for linear regression ",c_ols_opt)
```

```
## [1] "The optimal c for linear regression  0.39"
```

```
# resubstitution error
paste("The resubstitution error for linear regression is:",min(c_ols_all))
```

```
## [1] "The resubstitution error for linear regression is: 0.0903752039151713"
```

```
predict_ols <- predict(fit_ols, newdata = spam_test )
predict_ols[predict_ols>c_ols_opt] <- 1
predict_ols[predict_ols<=c_ols_opt] <- 0
predict_err_ols <- 1 - sum(predict_ols == spam_test$V58)/length(predict_ols)
paste("The prediction error for linear regression is:",predict_err_ols )
```

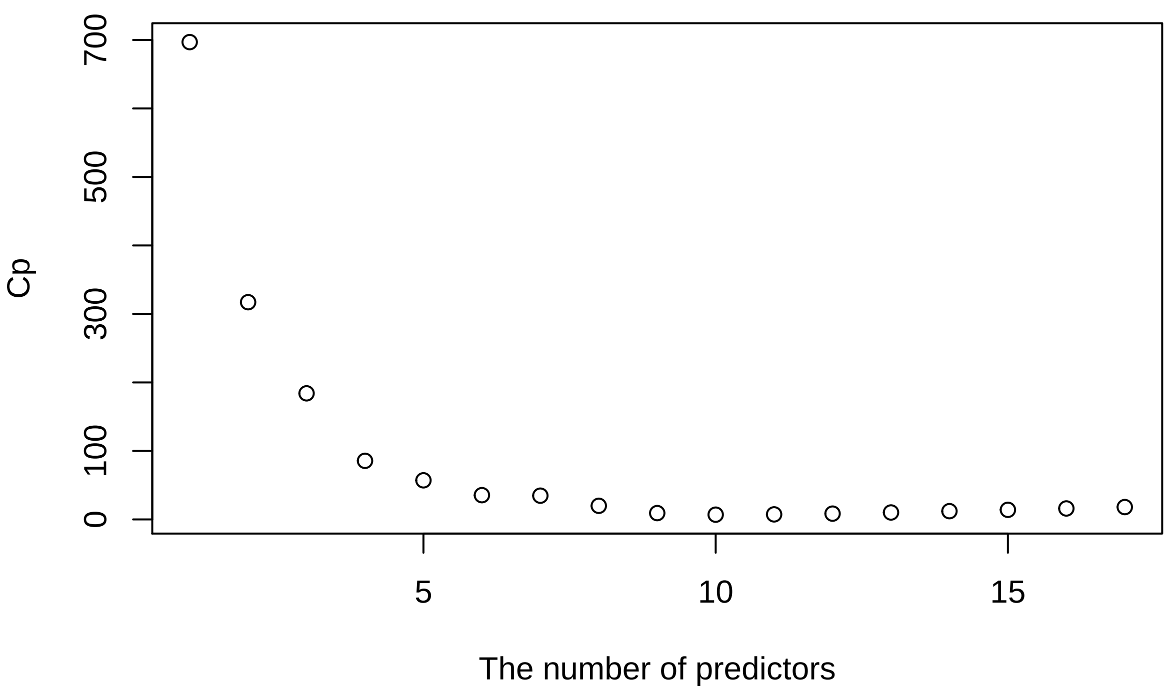
```
## [1] "The prediction error for linear regression is: 0.099609375"
```

Problem 4

```r
data("College")
dt <- College
set.seed(123)
# we chose 70% of data to train
id_sample <- sample(1:777,as.integer(777*0.7),replace=FALSE)
training <- dt[id_sample,]
training$Private<-ifelse(training$Private=="Yes",1,0)
testing <- dt[-id_sample,]
testing$Private<-ifelse(testing$Private=="Yes",1,0)

forward_fit <- regsubsets(Outstate ~ ., data = training, nvmax = 17, method = "forward")
forward_fit_summary <-summary(forward_fit)
```

we used Cp and AIC to choose the pedictors

```r
# cp
plot(1:17,forward_fit_summary$cp, xlab = "The number of predictors",ylab ="Cp")
```




```r
which(min(forward_fit_summary$cp)==forward_fit_summary$cp)
```

```
## [1] 10
```

```r
# AIC
null_model = lm(Outstate~1, data = training)
full_model = lm(Outstate~., data = training)
fit_forward = stepAIC(null_model, scope=list(upper=full_model,lower=null_model),direction = "forward",
# predictors I chose
names(coef(fit_forward ))[-1]
```

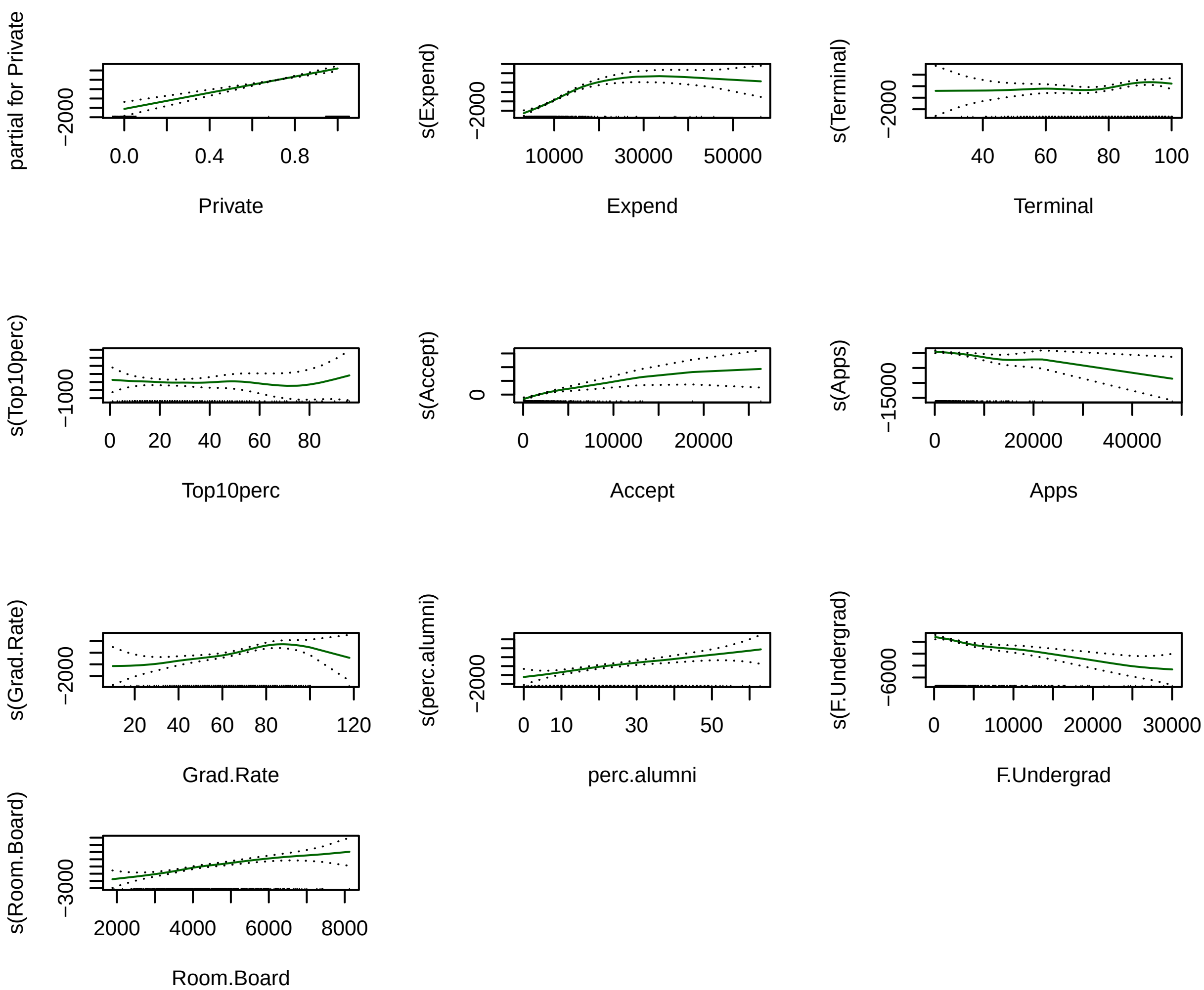
```
##  [1] "Room.Board"  "perc.alumni" "Expend"      "Private"     "Terminal"
##  [6] "Grad.Rate"   "Accept"      "Apps"        "F.Undergrad" "Top10perc"
```

```r
predict_val <- predict(fit_forward, newdata = testing)
err_forward <- mean((testing$Outstate-predict_val)^2)
```

We can find that the number of predictors are the same under Cp and AIC.

b

```r
gam_fit <- gam(Outstate ~ Private+s(Expend)+s(Terminal)+
                 s(Top10perc)+s(Accept)+s(Apps)+s(Grad.Rate)
               +s(perc.alumni) +s(F.Undergrad) +s(Room.Board), data =training)
par(mfrow = c(3, 3))
plot(gam_fit,se = T,col="darkgreen")
```




```r
gam_pred_val <- predict(gam_fit, testing)
gam.err <- mean((testing$Outstate - gam_pred_val)^2)


paste("The predicted RSS by forward stepwise is:",err_forward)
```

```
## [1] "The predicted RSS by forward stepwise is: 3995067.12447107"
```

```r
paste("The predicted RSS by GAM is:",gam.err)
```

```
## [1] "The predicted RSS by GAM is: 3348032.9306974"
```

We can find the performance of GAM is better than ordinary linear regression, which can be explained by GAM can fit well especially when local non-linear relationship appears.

5

```r
train_zip = read.csv('zip-train.txt', header = FALSE, sep = ' ')
train_zip_x <- train_zip[,-1]
train_zip_x$V258 <- c()
train_zip_y <- factor(train_zip[,1])

test_zip = read.csv('zip-test.txt', header = FALSE, sep = ' ')
test_zip_x <- test_zip[,-1]
test_zip_x$V258 <- c()
test_zip_y <- factor(test_zip[,1])
knn.classifier<- function(X.train, y.train, X.test, k.try=1 , pi=rep(1/K,K),CV=F){

  if (CV==FALSE){
    pred_class = sapply(k.try, function(ne){knn(train = X.train, test = X.test, y.train ,k=ne)} )

  }
  else {
    pred_class = sapply(k.try, function(ne){knn.cv(train = X.train,  y.train ,k=ne)} )
  }

  return (pred_class)
}
```

b

```r
data("iris")
iris$Species <- as.integer(iris$Species)
X.train_iris <- iris[,-5]
y.train_iris <- iris[,5]
res1 <- knn.classifier(X.train_iris,y.train_iris,X.train_iris,k.try = 5,pi=rep(1/3,3),CV = T)
paste("The number of classification with 'CV=T' is:",sum(res1 != y.train_iris ))
```

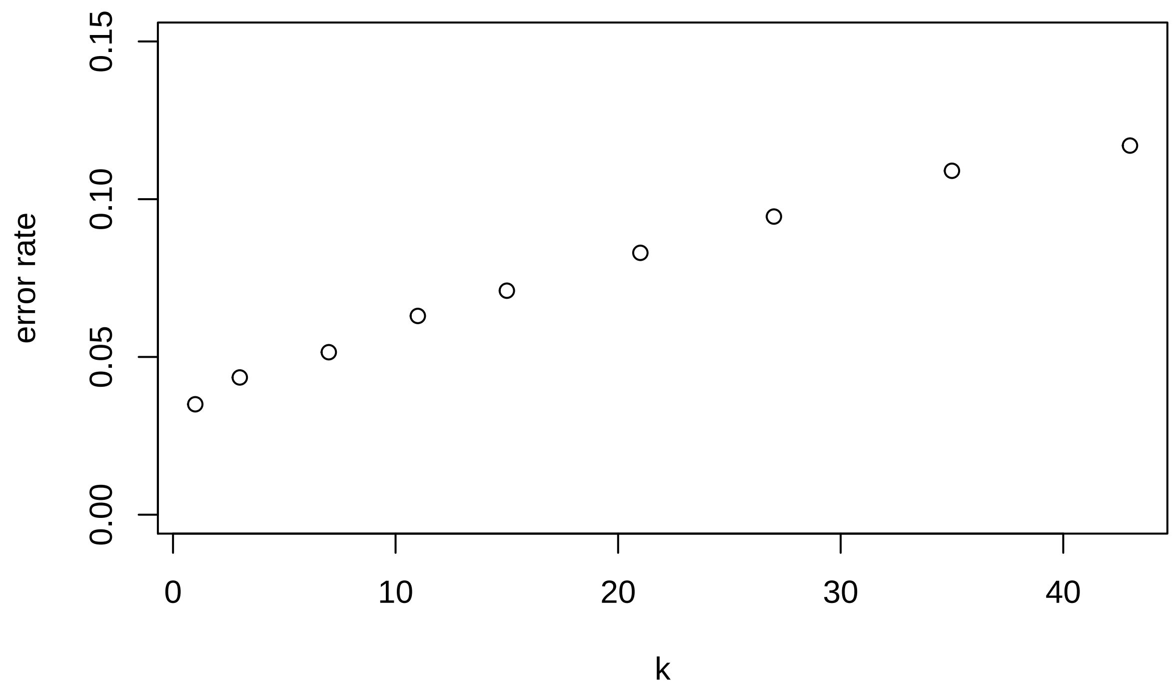
```
## [1] "The number of classification with 'CV=T' is: 5"
```

```r
# not use cv
res2 <- knn.classifier(X.train_iris,y.train_iris,X.train_iris,k.try = 5,pi=rep(1/3,3),CV =F)
paste("The number of classification with 'CV=F' is:",sum(res2 != y.train_iris ))
```

```
## [1] "The number of classification with 'CV=F' is: 5"
```

c

```r
res3 <- knn.classifier(train_zip_x,train_zip_y,test_zip_x,
               k.try = c(1, 3, 7, 11, 15, 21, 27, 35, 43),pi=rep(1/10,10),CV = T)

err <- rep(0,9)
for (i in 1:9){
  err[i] <- 1- sum(res3[,i] == train_zip_y )/length(train_zip_y )

}

plot(c(1, 3, 7, 11, 15, 21, 27, 35, 43),err, xlab = "k",ylab='error rate',ylim=c(0,0.15))
```




```r
paste("The optimal k is:",which(min(err)==err))
```

```
## [1] "The optimal k is: 1"
```

```r
paste("The corresponding error rate is:", min(err))
```

```
## [1] "The corresponding error rate is: 0.035"
```

```r
pred_knn <-  knn(train_zip_x,test_zip_x , train_zip_y ,k=1)
err_predict <- 1- sum(pred_knn == test_zip_y)/length(test_zip_y)
paste("The predict error rate for test data is:",err_predict)
```

```
## [1] "The predict error rate for test data is: 0.0792227204783259"
```