final.R

Charles Hesketh

2020-06-26

We will be using the following equation to calculate exclusivity based on the available data points provided in the dataset, 100*(Apps – Accept)/(Apps) + 100*(Enroll/Accept). In order to find the “best” model given the options we will be using 5-fold cross validation which gives us, roughly, an 80/20 split between training/testing sets, respectively. We will be using SSR as our objective measure of fit throughout the models. To increase the certainty that we are picking the right model we will run this cross validation 1000 times with different seeds values. This will allow us to run the cross validation multiple times and then evaluate which models succeed most often and by how much on average.

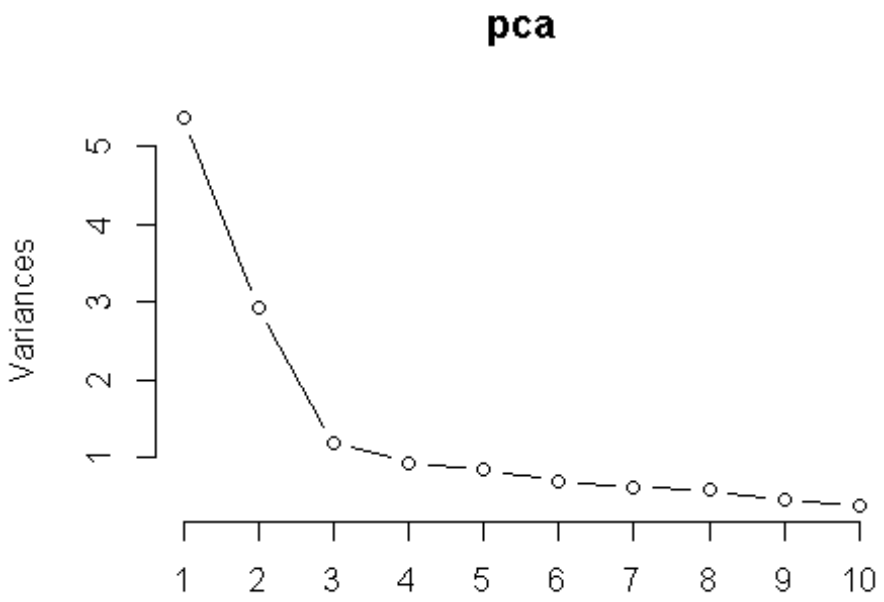
In order to keep things consistent I separate the data into training and testing first, then run each model, with optimal lambda calculated for each fold where required. This resulted in:

Winning Model:	Freq:
Ridge	566
Elastic	275
Lasso	159

If we look at the average for each model we get:

Model Type:	SSR Average:
LM	614.6727
Ridge	611.2132
Lasso	612.4881
Elastic	611.9760
PCR: Num Comp=1	781.7070
PCR: Num Comp=2	719.8120
PCR: Num Comp=3	716.6760
PCR: Num Comp=4	673.7329
PCR: Num Comp=5	674.5847
PCR: Num Comp=6	670.5848
PCR: Num Comp=7	658.6806
PCR: Num Comp=8	659.1356
PCR: Num Comp=9	662.3527
PCR: Num Comp=10	666.8645
PCR: Num Comp=11	671.0446
PCR: Num Comp=12	635.7684
PCR: Num Comp=13	615.1398
PLSR: Num Comp=1	678.1421
PLSR: Num Comp=2	639.6390
PLSR: Num Comp=3	626.4530
PLSR: Num Comp=4	618.7633
PLSR: Num Comp=5	616.3961
PLSR: Num Comp=6	614.2641
PLSR: Num Comp=7	615.6891
PLSR: Num Comp=8	615.2908
PLSR: Num Comp=9	615.2433
PLSR: Num Comp=10	614.9362
PLSR: Num Comp=11	614.8096
PLSR: Num Comp=12	614.7181
PLSR: Num Comp=13	614.6744

As we can see the average is not significantly different for these three models and is not too far off from the linear model as well. This means we are not overfitting this model too much with the linear model. With regards to the PCR and PLSR we can see from our PCA plot that somewhere between 3-4 component vectors is ideal:



When doing some testing 4 appears to be best; however, this is not competitive. I did the analysis using all 13 possible components values ranging from 1 to 13 (seen in the average table above). We can see that even at 13 components we do not see a model that outperforms the Ridge, Elastic, Lasso, or Linear Models. Given that the first three models are so much better, PCR/PLSR does not reduce the dimensions much, and the data set is fairly small thus I do not see a reason to use PCR/PLSR as there is not a noticeable decrease in computing time that would merit the loss in model accuracy.

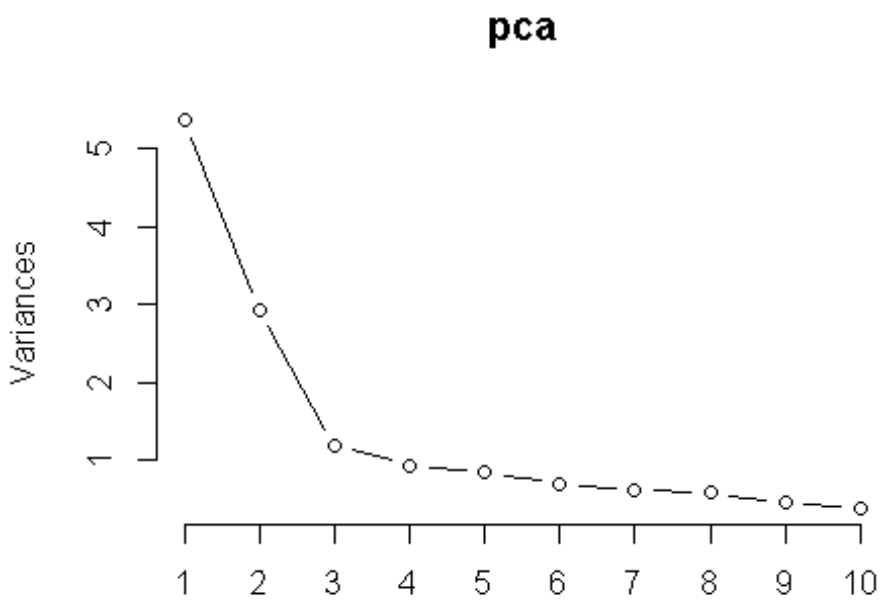
Therefore, given how close the average SSR for Ridge and Elastic is, I recommend the Ridge model because the frequency that this model created a minimal SSR was more then double the frequency of the Elastic model.

Appendix: Code

```
library(MASS)
library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 4.0-2

library(assist)

## Loading required package: nlme

## Loading required package: lattice

##
## Attaching package: 'assist'

## The following object is masked from 'package:Matrix':
##
##     bdiag

library(pls)

##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings

setwd("C:\\Users\\User\\Desktop\\School\\Math_537\\Final")

data=read.csv("College.csv")

data$exclu=100*(data$Apps-data$Accept)/data$Apps+100*(data$Enroll/data$Accept
)
Private=ifelse(data$Private=="Yes", 1,0)
dataS=scale(cbind(Private,data[,7:dim(data)[2]-1]),center=T,scale=T)
y=scale(data[,dim(data)[2]],center=T,scale=T)

modelRes=matrix(0,30,1000)

for(k in 1:1000)
{
  j=5
  set.seed(k)
  folds=cut(seq(1,nrow(data)),breaks=j,labels=F)
```

```
  testIndexes=matrix(0,length(folds[folds==1]),j)
  for (i in 1:j)
  {
    if(length(folds[folds==i])==156)
    {
      testIndexes[,i]=which(folds==i,arr.ind=T)
    }
    else
    {
      testIndexes[,i]=c(which(folds==i,arr.ind=T),0)
    }
  }

  pca=prcomp(dataS)

  plot(pca,type="l")

  #Not a big gain for any addition components after 3, will pick 3.

  i=1


  for(i in 1:j)
  {

    testDatax=dataS[testIndexes[,i],]
    trainDatax=dataS[-testIndexes[,i],]

    testDatay=y[testIndexes[,i]]
    trainDatay=y[-testIndexes[,i]]

    trainData=data.frame(cbind(trainDatay,trainDatax))
    testData=data.frame(cbind(testDatay,testDatax))

    ols <- lm(trainData$trainDatay~ ., data=trainData)#trainData[,2:dim(train
Data)[2]], data=trainData)
    pred=predict.glm(ols,newdata=data.frame(testDatax),type="response")
    modelRes[1,k]=modelRes[1,k]+sum((testDatay-pred)^2)

    cvLam=cv.glmnet(data.matrix(trainDatax),y=trainDatay,alpha=0,nfolds=j)$la
mbda.min
    ridge=glmnet(trainDatax,trainDatay,alpha=0,lambda=cvLam)
    pred=predict(ridge,newx=data.matrix(testDatax),s=cvLam)
    modelRes[2,k]=modelRes[2,k]+sum((testDatay-pred)^2)

    cvLam=cv.glmnet(data.matrix(trainDatax),y=trainDatay,alpha=1,nfolds=j)$la
mbda.min
    lasso=glmnet(data.matrix(trainDatax),trainDatay,alpha=1,lambda=cvLam)
    pred=predict(lasso,newx=data.matrix(testDatax),s=cvLam)
```

```
    modelRes[3,k]=modelRes[3,k]+sum((testDatay-pred)^2)

    cvLam=cv.glmnet(data.matrix(trainDatax),y=trainDatay,alpha=.5,nfolds=j)$l
ambda.min
    elast=glmnet(trainDatax,trainDatay,alpha=.5,lambda=cvLam)
    pred=predict(elast,newx=data.matrix(testDatax),s=cvLam)
    modelRes[4,k]=modelRes[4,k]+sum((testDatay-pred)^2)

    for(j in 1:13)
    {
      mPCR=pcr(trainDatay~trainDatax ,ncomp=j)
      pred=predict(mPCR,testDatax,ncomp=j)
      modelRes[5+j-1,k]=modelRes[5+j-1,k]+sum((testDatay-pred)^2)

      mPLSR = plsr(trainDatay~trainDatax,ncomp=j)
      pred=predict(mPLSR,testDatax,ncomp=j)
      modelRes[6+13+j-2,k]=modelRes[6+13+j-2,k]+sum((testDatay-pred)^2)
    }

  }
  if(k%%100==0)
  {
    print(k)
  }

}
```



```
name=c('LM','Ridge','Lasso','Elastic')
for(i in 1:13)
{
  name=c(name,paste0("PCR: Lambda=",i))
}
for(i in 1:13)
{
  name=c(name,paste0("PLSR: Lambda=",i))
}
rownames(modelRes)=as.vector(name)

winningModel=apply(modelRes,2,which.min)
winningModel[winningModel==2]="Ridge"
winningModel[winningModel==3]="Lasso"
winningModel[winningModel==4]="Elastic"

output=as.data.frame(table(winningModel))
output=output[order(-output$Freq),]
output
```

```
##   winningModel Freq
## 3        Ridge  566
## 1      Elastic  275
## 2        Lasso  159
```

```
rowMeans(modelRes)
```

```
               LM             Ridge             Lasso           Elastic   PCR
: Num Comp=1   PCR: Num Comp=2   PCR: Num Comp=3
         614.6727          611.2132          612.4881          611.9760
781.7070          719.8120          716.6760
  PCR: Num Comp=4   PCR: Num Comp=5   PCR: Num Comp=6   PCR: Num Comp=7   PCR
: Num Comp=8   PCR: Num Comp=9  PCR: Num Comp=10
         673.7329          674.5847          670.5848          658.6806
659.1356          662.3527          666.8645
 PCR: Num Comp=11  PCR: Num Comp=12  PCR: Num Comp=13  PLSR: Num Comp=1  PLSR
: Num Comp=2  PLSR: Num Comp=3  PLSR: Num Comp=4
         671.0446          635.7684          615.1398          678.1421
639.6390          626.4530          618.7633
 PLSR: Num Comp=5  PLSR: Num Comp=6  PLSR: Num Comp=7  PLSR: Num Comp=8  PLSR
: Num Comp=9 PLSR: Num Comp=10 PLSR: Num Comp=11
         616.3961          614.2641          615.6891          615.2908
615.2433          614.9362          614.8096
PLSR: Num Comp=12 PLSR: Num Comp=13
         614.7181          614.6744
```