INFO 420: Assignment 8

Jiawei Liu 9752650

Question 1

1. How many predictors are there, i.e. what is p?

```
Credit = read.csv("D:/Credit.csv",header=TRUE)
head(Credit)
```

```
##   income limit rating cards age education gender student married ethn
icity
## 1  14.89  3606    283     2  34        11   Male      No     Yes Caucas
ian
## 2 106.03  6645    483     3  82        15 Female     Yes     Yes     As
ian
## 3 104.59  7075    514     4  71        11   Male      No      No     Asi
an
## 4 148.92  9504    681     3  36        11 Female      No      No     Asi
an
## 5  55.88  4897    357     2  68        16   Male      No     Yes Caucas
ian
## 6  80.18  8047    569     4  77        10   Male      No      No Caucasi
an
##   balance
## 1     333
## 2     903
## 3     580
## 4     964
## 5     331
## 6    1151
```

```
dim(Credit)
```

```
## [1] 400  11
```

There are 11 variables here. P means the number of features in the dataset.

2. Select the tuning parameter for the ridge regression and lasso models using cross-validation. Make general comments about the final form of each model. How many features have been selected by the lasso?

```
x = model.matrix(balance~.*.,Credit)[,-1]
y = Credit$balance

set.seed(987654312)
train = sample(1:nrow(x),nrow(x)/2)
test = -train
```

```
library(glmnet)

## Warning: package 'glmnet' was built under R version 3.4.4

## Loading required package: Matrix

## Loading required package: foreach

## Warning: package 'foreach' was built under R version 3.4.4

## Loaded glmnet 2.0-16

grid = 10^seq(3,-1,length=100)
set.seed(987654313)
cv.Rout = cv.glmnet(x[train,],y[train],alpha=0,lambda=grid,nfolds=10,th
resh=1e-10)
cv.Rout$lambda.min

## [1] 1.789

set.seed(987654313)
cv.LAout = cv.glmnet(x[train,],y[train],alpha=1,lambda=grid,nfolds=10,t
hresh=1e-10)
cv.LAout$lambda.min

## [1] 1.963
```

In this question, the tuning parameter of the ridge regression is 1.789 and the tuning parameter of the lasso is 1.963.

```
Rbestlam = cv.Rout$lambda.min
bridge.out = glmnet(x,y,alpha=0,lambda=grid,thresh=1e-10)
predict(bridge.out,type="coefficients",s=Rbestlam)[1:12,]

##       (Intercept)            income             limit
##         -278.58059          -2.03231           0.06764
##            rating             cards               age
##           0.54691          10.69344           1.15118
##         education      genderFemale         studentYes
##           1.27168         -66.50707         118.40160
##         marriedYes     ethnicityAsian ethnicityCaucasian
##           41.29797          53.56494           3.68141

LAbestlam = cv.LAout$lambda.min
lasso.out = glmnet(x,y,alpha=1,lambda=grid,thresh=1e-10)
predict(lasso.out,type="coefficients",s=LAbestlam)[1:12,]

##       (Intercept)            income             limit
##         -226.7337          -3.6659            0.1239
##            rating             cards               age
##            0.0000           9.0781            0.0000
```

```
##          education       genderFemale         studentYes
##             0.0000             0.0000          158.2142
##        marriedYes     ethnicityAsian ethnicityCaucasian
##             0.0000             0.0000             0.0000
```

We can easily find that the ridge regression uses all features to build model while the lasso only uses some of them. Therefore, the ridge regression might be more complex than the lasso in this question. In the lasso, there are 4 features: income, limit, cards, studentYes.

3. Compare the test errors for the linear model and ridge regression and lasso models.

```
linear.mod = lm(y[train]~x[train,])
linear.pred = coef(linear.mod)[1]+x[test,] %*% coef(linear.mod)[-1]
mean((linear.pred-y[test])^2)
```

```
## [1] 7056
```

```
ridge.pred = predict(cv.Rout,s=Rbestlam,newx=x[test,])
mean((ridge.pred-y[test])^2)
```

```
## [1] 7076
```

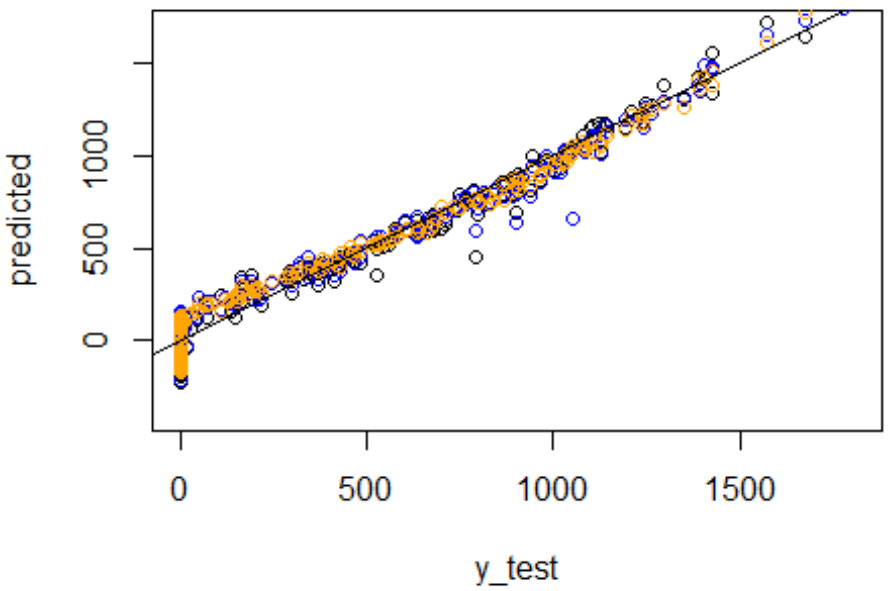
```
lasso.pred = predict(cv.LAout,s=LAbestlam,newx=x[test,])
mean((lasso.pred-y[test])^2)
```

```
## [1] 4907
```

Here are three test errors. For the linear model: 7056 For the ridge regression: 7076 For the loss: 4907

4. Plot a comparison of the test predictions for the three approaches.

```
plot(y[test],linear.pred,ylim=c(-400,1700),xlab="y_test",ylab="predicte
d")
points(y[test],ridge.pred,col="blue")
points(y[test],lasso.pred,col="orange")
abline(0,1)
```


predicted
0
500
1000
0
500
1000
1500
y_test

Question 2 (i)

1. What happens to the test misclassification error as the degrees of freedom of the natural splines for income and limit vary over the 16 combinations (1,1), (1,2), ...,(4,4)? (You may use 0.5 as the classification threshold.)

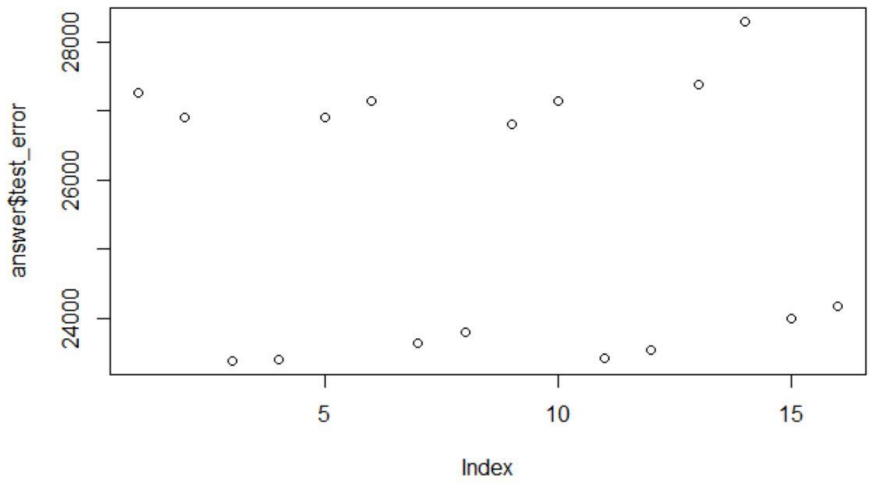
```
library(gam)
library(splines)
library(mgcv)
Credit = read.csv("D:/Credit.csv",header=TRUE)

set.seed(312)
train = sample(1:nrow(Credit),nrow(Credit)/2)
test = -train

answer = NULL
for (i in 1:4){
  for (j in 1:4){
    gam.modi.j = gam(balance~ns(income,df=i)+
                       ns(limit,df=j),data=Credit[train,])
    pred.modi.j = predict(gam.modi.j,newdata=Credit[test,])
    mesi.j = mean((pred.modi.j-Credit$balance[test])^2)
    answeri.j = c(i,j,mesi.j)
    answer = rbind(answer, answeri.j)
  }
}

answer = as.data.frame(answer)
colnames(answer) = c("i", "j", "test_error")

plot(answer$test_error)
```

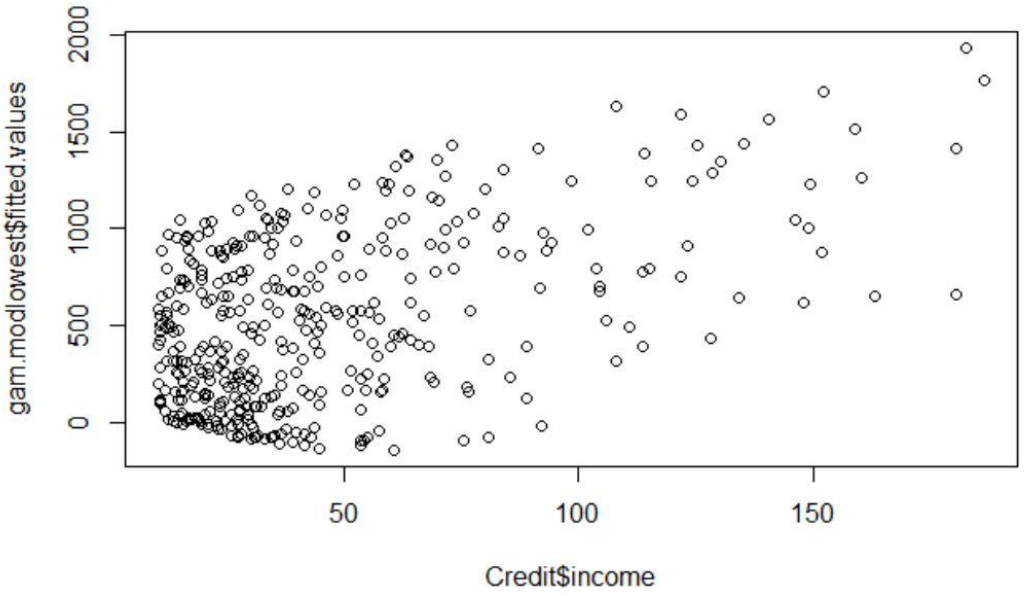


It can be seen that test_error varies from 23000 to more than 28000 with different combinations.

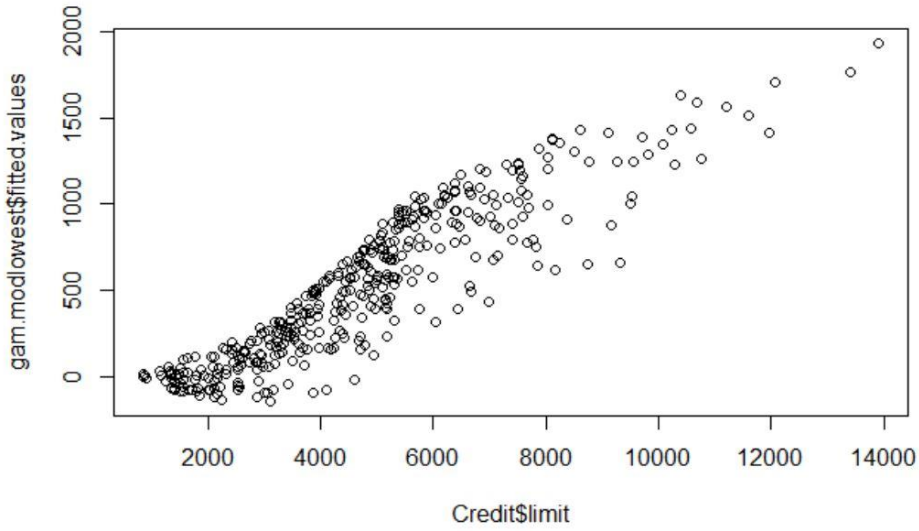
2. Choose the model with the lowest test error. Describe the effect of income and limit in this model. Give a plausible explanation for how the credit company uses limit strategically.

```
minE = min(answer$test_error)
mini.j = answer[which(answer$test_error == minE),]

gam.modlowest = gam(balance~ns(income,df=mini.j$i)+
                      ns(limit,df=mini.j$j),data=Credit)
plot(Credit$income,gam.modlowest$fitted.values)
```




```
plot(Credit$limit,gam.modlowest$fitted.values)
```



It can be seen from these two plots both income and limit have positive influence on predicting the balance, and the effect of limit is clearer. The result of anova for

Parametric Effects also proves that ns(income, df = mini.j$i) and ns(limit,df=mini.j$j) have reasonable influence in the model because the p-vulue of them are much smaller than 0.05. If a credit company wants to use limit strategically, it should bear in mind that limit is one crucial variable to predict customers' balance but other variables should also be considered.

```
y = as.numeric(Credit$balance/(Credit$income*1000/12)>0.2)
Credit_train = Credit[train,]

gam.mod = gam(y[train]~ns(Credit_train$income,df=mini.j$i)+
         ns(Credit_train$limit,df=mini.j$j), family="binomial")

gam.pred = predict(gam.mod,newdata=Credit[test,],type="response")
table(gam.pred>0.5,y[test])
```

```
##
##          0   1
##   FALSE  95  37
##   TRUE   38  30
```

The misclassification error rate is 37.5%.

3. What happens if you now include student in your model?

```
gam.mod = gam(y[train]~ns(Credit_train$income,df=mini.j$i)+
         ns(Credit_train$limit,df=mini.j$j)+Credit_train$student,
family="binomial")

gam.pred = predict(gam.mod,newdata=Credit[test,],type="response")
table(gam.pred>0.5,y[test])
```

```
##
##          0   1
##   FALSE  95  36
##   TRUE   38  31
```

It can be found that one misclassification is corrected after adding student in the model.

Question 2 (ii)

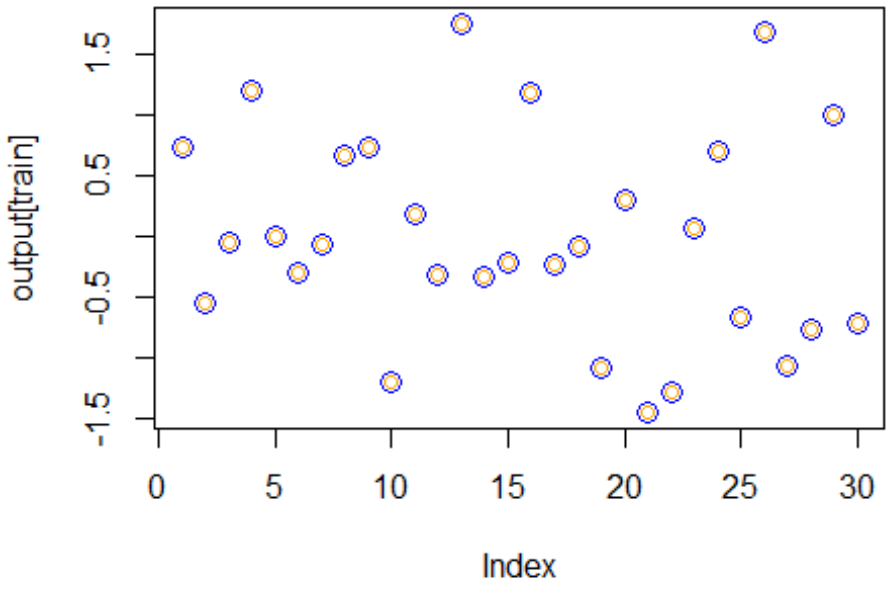
1. Confirm that a linear model can fit the training data exactly. Why is this model not going to be useful?

```
X = read.csv("D:/parkinsons.csv",header=TRUE)
X = scale(X)

set.seed(14312)
train = sample(1:nrow(X),30)
c = 1:nrow(X)
test = c[-train]

input = X[,-ncol(X)]
output = X[,ncol(X)]

linear.mod = lm(output[train]~input[train,])
plot(output[train], col="orange")
points(linear.mod$fitted.values, col="blue", cex = 1.5)
```



It can be seen that linear model can fit the training data exactly because the number of features is bigger than the number of observations. In other words, it a high-dimensional dataset. It would cause the danger of overfitting and might perform reasonably bad when it is applied to new observations so this model is not so useful.

2. Now use the lasso to fit the training data, using leave-one-out cross-validation to find the tuning parameter λ. (This means nfolds=30. You will also

find it convenient to set grid = 10??seq(3,-1,100) and thresh=1e-10.) What is the optimal value of λ and what is the resulting test error?

```
library(glmnet)

## Warning: package 'glmnet' was built under R version 3.4.4

## Loading required package: Matrix

## Loading required package: foreach

## Warning: package 'foreach' was built under R version 3.4.4

## Loaded glmnet 2.0-16

grid = 10^seq(3,-1,length=100)
set.seed(98734545)
cv.LAout = cv.glmnet(input[train,],output[train],
                     alpha=1,lambda=grid,nfolds=30,thresh=1e-10)

## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observ
ations
## per fold

cv.LAout$lambda.min

## [1] 0.1

LAbestlam = cv.LAout$lambda.min
lasso.pred = predict(cv.LAout,s=LAbestlam,newx=input[test,])
mean((lasso.pred-output[test])^2)

## [1] 0.1534199
```

In this question, the optimal value of λ is 0.1 and the resulting test error is 0.1534199.

3. State your final model for the UPDRS. How many features have been selected? What conclusions can you draw?

```
lasso.out = glmnet(input[train,],output[train],alpha=1,lambda=grid,thre
sh=1e-10)
feature_selected = predict(lasso.out,type="coefficients",s=LAbestlam)
[,]
feature_selected[-which(feature_selected == 0),]

##  (Intercept)           X9          X83          X97
## -0.003622642  0.001351132  0.082318259  0.759642355
```

It can be seen that there are three features have been selected in this model: X9, X83, X97. It can be concluded that for the linear model, regardless of whether or not there truly is a relationship between the features and the response, the linear

model will yield a set of coefficient estimates that result in a perfect fit to the data, such that the residuals are zero. In other words, a lot of unnecessary are selected when we applying the linear model to high-dimensional datasets. From the resulting test error in question 2 and selected features of final model in question 3, we may conclude that the lasso works reasonably for this high-dimensional dataset.

4. Repeat your analysis with a different random split into training and test sets. Have the same features been selected in your final model?

```
set.seed(14342342)
train2 = sample(1:nrow(X),30)
c = 1:nrow(X)
test2 = c[-train2]

set.seed(987343334)
cv.LAout2 = cv.glmnet(input[train2,],output[train2],
                      alpha=1,lambda=grid,nfolds=30,thresh=1e-10)

## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observ
ations
## per fold

LAbestlam2 = cv.LAout2$lambda.min

lasso.out2 = glmnet(input[train2,],output[train2],alpha=1,lambda=grid,t
hresh=1e-10)
feature_selected2 = predict(lasso.out2,type="coefficients",s=LAbestlam
2)[,]
feature_selected2[-which(feature_selected2 == 0),]

##  (Intercept)          X83          X97
## -0.008727925  0.002526314  0.846537210
```

X97 is still selected but the rest selected features are different. Therefore, X97 is considerably essential in modeling this dataset and other features may not so important. The estimate of X97 is much bigger than others' estimates, which also proves this.

Question 3

This is a somewhat open-ended question. The file 2 Climate_Sim_Crash.txt gives the 18 input parameters of 540 climate model simulations. The binary outcome of interest is whether the resulting simulation crashed, outcome = 0, or not, outcome = 1. I would like you to analyze the data using LDA, QDA and KNN to see if you can predict which simulations will crash. Feel free to try other classifiers too. You will need to take into account that this is an unbalanced dataset: fewer than 10% of the simulations crashed.

sample the dataset

```
climate = read.table("D:/Climate_Sim_Crash.txt",header=TRUE)

set.seed(5555)
train = c(sample(which(climate$outcome == 0),length(which(climate$outco
me == 0))/2),sample(which(climate$outcome == 1),length(which(climate$ou
tcome == 1))/2))
c = 1:nrow(climate)
test = c[-train]
```

Logistic regression

```
climate_train = climate[train,]

log.mod = glm(outcome~.-Study-Run,data=climate_train,family="binomial")

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

log.prob = predict(log.mod,climate[test,],type="response")
log.pred = log.prob>=0.5
table(log.pred>0.5,climate[test,"outcome"])

##
##          0   1
##   FALSE  16  11
##   TRUE    7 236
```

The misclassification error rate of the Logistic regression is 6.67%.

LDA

```
library(MASS)

## Warning: package 'MASS' was built under R version 3.4.4

lda.fit = lda(outcome~.-Study-Run,data=climate_train)
lda.pred = predict(lda.fit,climate[test,])
table(lda.pred$class,climate[test,"outcome"])
```

```
##
##       0   1
##   0  10   2
##   1  13 245
```

The misclassification error rate of the LDA is 5.56%.

QDA

```
qda.fit = qda(outcome~.-Study-Run,data=climate_train)
qda.pred = predict(qda.fit,climate[test,])
table(qda.pred$class,climate[test,ncol(climate)])
```

```
##
##       0   1
##   0   0   1
##   1  23 246
```

The misclassification error rate of the QDA is 8.89%.

KNN

```
library(class)
set.seed(456463)
 M = 10
 Kmax = 15
 MC = numeric(Kmax)
 for (ctr in 1:M) {
   for (c in train) {
      loo = setdiff(train,c)
      for (k in 1:Kmax) {
      knn.pred = knn(climate[loo,-ncol(climate)],
                     climate[c,-ncol(climate)],climate[loo,ncol(climat
e)],k=k)
      MC[k] = MC[k] + as.numeric(knn.pred!=climate[c,ncol(climate)])
     }
   }
}
MC = MC/M

bestk = which(MC == min(MC))
knn.pred = knn(climate[train,-ncol(climate)],
          climate[test,-ncol(climate)],climate[train,ncol(climate)],k=mi
n(bestk))
table(knn.pred,climate[test,ncol(climate)])
```

```
##
## knn.pred   0   1
##        0   0   0
##        1  23 247
```

The misclassification error rate of the QDA is 8.52%.