```
# Rudy Meza
# Machine Learning in R
# Michael Chang
# Spring
# '***Lab and Quiz***'
# '**********Machine Learning**********'
# 6.5 Lab 1: Subset Selection Methods
library (ISLR)
fix(Hitters )
names(Hitters )
# The sum() function can then be used to count all of the missing elements
dim(Hitters )
sum(is.na(Hitters$Salary))
# Hence we see that Salary is missing for 59 players. The na.omit() function
# removes all of the rows that have missing values in any variable.
Hitters =na.omit(Hitters )
dim(Hitters )
sum(is.na(Hitters ))

library (leaps)
regfit.full=regsubsets (Salary~.,Hitters )
summary (regfit.full)
regfit.full=regsubsets (Salary~.,data=Hitters ,nvmax =19)
reg.summary =summary (regfit.full)
reg.summary$rsq
par(mfrow =c(2,2))
plot(reg.summary$rss ,xlab=" Number of Variables ",ylab=" RSS",
type="l")
 plot(reg.summary$adjr2 ,xlab =" Number of Variables ",
ylab=" Adjusted RSq",type="l")
which.max(reg.summary$adjr2)
points (11, reg.summary$adjr2[11], col ="red",cex =2, pch =20)

# In a similar fashion we can plot the Cp and BIC statistics, and indicate the
# models with the smallest statistic using which.min()
plot(reg.summary$cp ,xlab =" Number of Variables ",ylab="Cp",
type= 'l')
 which.min(reg.summary$cp)
 points (10, reg.summary$cp [10], col ="red",cex =2, pch =20)
which.min (reg.summary$bic)

plot(reg.summary$bic ,xlab=" Number of Variables ",ylab=" BIC",
type= 'l')
points (6, reg.summary$bic [6], col =" red",cex =2, pch =20)
# The follwong command helps with Error in plot.new() : figure margins too large
graphics.off()
par("mar")
par(mar=c(1,1,1,1))
plot(regfit.full ,scale ="r2")
plot(regfit.full ,scale ="adjr2")
plot(regfit.full ,scale ="Cp")
```

```
plot(regfit.full ,scale ="bic")
coef(regfit.full ,6)
# 6.5.2 Forward and Backward Stepwise Selection
# We can also use the regsubsets() function to perform forward stepwise
# or backward stepwise selection, using the argument method="forward" or
# method="backward".
regfit.fwd=regsubsets (Salary~.,data=Hitters ,nvmax =19,
method ="forward")
summary (regfit.fwd )
regfit.bwd=regsubsets (Salary~.,data=Hitters ,nvmax =19,
method ="backward")
summary(regfit.bwd )

coef(regfit.full ,7)
coef(regfit.fwd ,7)
coef(regfit.bwd ,7)

# 6.5.3 Choosing Among Models Using the Validation Set
# Approach and Cross-Validation
# In order to use the validation set approach, we begin by splitting the
# observations into a training set and a test set. We do this by creating
# a random vector, train, of elements equal to TRUE
set.seed (1)
train=sample(c(TRUE ,FALSE), nrow(Hitters ),rep=TRUE)
 test=(!train )
 # Now, we apply regsubsets() to the training set in order to perform best
# subset selection.
regfit.best=regsubsets (Salary~.,data=Hitters [train ,],
nvmax =19)
# Model Matrix from the test data
test.mat=model.matrix(Salary~.,data=Hitters [test ,])
val.errors =rep(NA ,19)
for(i in 1:19){
coefi=coef(regfit.best ,id=i)
pred=test.mat [,names(coefi)]%*% coefi
val.errors [i]= mean(( Hitters$Salary[test]-pred)^2)
}
# We find that the best model is the one that contains ten variables.
val.errors
which.min (val.errors )
coef(regfit.best ,10)
predict.regsubsets =function (object ,newdata ,id ,...){
 form=as.formula (object$call [[2]])
 mat=model.matrix (form ,newdata )
 coefi =coef(object ,id=id)
 xvars =names (coefi )
 mat[,xvars ]%*% coefi
 }
regfit.best=regsubsets (Salary~.,data=Hitters ,nvmax =19)
coef(regfit.best ,10)
k=10
```

```
set.seed (1)
folds=sample (1:k,nrow(Hitters ),replace =TRUE)
cv.errors =matrix (NA ,k,19, dimnames =list(NULL , paste (1:19) ))
for(j in 1:k){
 best.fit =regsubsets (Salary~.,data=Hitters [folds !=j,],
nvmax =19)
 for(i in 1:19) {
 pred=predict (best.fit ,Hitters [folds ==j,], id=i)
cv.errors [j,i]=mean( (Hitters$Salary[folds ==j]-pred)^2)
}
 }
mean.cv.errors =apply(cv.errors ,2, mean)
mean.cv.errors
par(mfrow =c(1,1))
plot(mean.cv.errors ,type='b')
reg.best=regsubsets (Salary~.,data=Hitters , nvmax =19)
coef(reg.best ,11)
# 6.6 Lab 2: Ridge Regression and the Lasso
x=model.matrix (Salary~.,Hitters )[,-1]
y=Hitters$Salary
# 6.6.1 Ridge Regression
library(glmnet)
grid =10^seq(10,-2, length =100)
ridge.mod =glmnet(x,y,alpha =0, lambda =grid)

dim(coef(ridge.mod ))
ridge.mod$lambda [50]
coef(ridge.mod)[,50]
sqrt(sum(coef(ridge.mod)[ -1 ,50]^2) )
ridge.mod$lambda [60]
sqrt(sum(coef(ridge.mod)[ -1 ,60]^2) )
predict(ridge.mod ,s=50, type ="coefficients")[1:20 ,]
set.seed (1)
train=sample (1: nrow(x), nrow(x)/2)
test=(- train )
y.test=y[test]

ridge.mod =glmnet(x[train ,],y[train],alpha =0, lambda =grid ,
thresh =1e-12)
ridge.pred=predict(ridge.mod ,s=4, newx=x[test ,])
mean(( ridge.pred -y.test)^2)

mean(( mean(y[train ])-y.test)^2)
ridge.pred=predict(ridge.mod ,s=1e10 ,newx=x[test ,])
mean(( ridge.pred -y.test)^2)
ridge.pred=predict(ridge.mod,s=0, newx=x[test,], exact=T, x=x[train ,], y=y[train])
mean(( ridge.pred-y.test)^2)
lm(y~x, subset =train)
predict (ridge.mod ,s=0, exact =T,  x=x[train ,], y=y[train],type="coefficients") [1:20 ,]
set.seed (1)
cv.out =cv.glmnet (x[train ,],y[train],alpha =0)
```

```
plot(cv.out)
bestlam =cv.out$lambda.min
bestlam
ridge.pred=predict(ridge.mod ,s=bestlam ,newx=x[test ,])
 mean(( ridge.pred -y.test)^2)
  out=glmnet(x,y,alpha =0)
predict(out ,type="coefficients",s=bestlam )[1:20 ,]

# 6.6.2 The Lasso
lasso.mod=glmnet(x[train ,],y[train],alpha =1, lambda =grid)
plot(lasso.mod)
set.seed (1)
cv.out =cv.glmnet(x[train ,],y[train],alpha =1)
plot(cv.out)
bestlam =cv.out$lambda.min
lasso.pred=predict (lasso.mod ,s=bestlam ,newx=x[test ,])
mean(( lasso.pred-y.test)^2)
out=glmnet (x,y,alpha =1, lambda =grid)
lasso.coef=predict(out ,type ="coefficients",s=bestlam )[1:20 ,]
lasso.coef
lasso.coef[lasso.coef !=0]
# 6.7.1 Principal Components Regression
library (pls)
set.seed (2)
pcr.fit=pcr(Salary~., data=Hitters ,scale=TRUE ,
validation ="CV")
summary(pcr.fit )
validationplot(pcr.fit ,val.type="MSEP")
set.seed (1)
pcr.fit=pcr(Salary~., data=Hitters ,subset =train ,scale =TRUE ,
validation ="CV")
validationplot(pcr.fit ,val.type="MSEP")
pcr.pred=predict (pcr.fit ,x[test ,], ncomp =7)
mean((pcr.pred -y.test)^2)
pcr.fit=pcr(y~x,scale =TRUE ,ncomp =7)
summary(pcr.fit)
# 6.7.2 Partial Least Squares
set.seed (1)
pls.fit=plsr(Salary~., data=Hitters ,subset =train ,scale=TRUE ,
validation ="CV")
 summary(pls.fit )
  validationplot(pls.fit ,val.type="MSEP")
 pls.pred=predict(pls.fit ,x[test ,], ncomp =2)
mean((pls.pred -y.test)^2)
pls.fit=plsr(Salary~., data=Hitters ,scale=TRUE ,ncomp =2)
summary(pls.fit )
# """""""""""""""Week 4 Quiz""""""""""""""""""""
# Question 1
coef(regfit.full, 7)
coef(regfit.full, 8)
# atbat, cruns, cwalks
```

```
# Question 2
regfit.full=regsubsets(Salary~.,data=Hitters ,nvmax =19)
reg.summary =summary(regfit.full)
which.min(reg.summary$bic)
which.max(reg.summary$adjr2)
reg.summary$adjr2[11]
reg.summary$bic[6]
# Question 3
# False
# Question 4
x=model.matrix (Salary~.,Hitters )[,-1]
y=Hitters$Salary
library(glmnet)
grid=10^seq(10,-2, length =100)
ridge.mod =glmnet(x,y,alpha =0, lambda =grid)
 plot(ridge.mod,xvar="lambda",label ="TRUE")
# Zoom helped here
# 15
# Question 5
set.seed (1)
cv.out =cv.glmnet(x[train ,],y[train],alpha =1)
plot(cv.out)
bestlam =cv.out$lambda.min
bestlam
# 16.78
# Question 6
lasso.mod <- glmnet(x[train,], y[train], alpha = 1, lambda = grid)
lasso.pred <- predict(lasso.mod, s = bestlam, newx = x[test,])
mean((lasso.pred - y.test)^2) # 100743.4
out <- glmnet(x, y, alpha = 1,lambda = grid)
lasso.coef <- predict(out, type = "coefficients", s = bestlam)[1:20,]
lasso.coef
# 7
# Question 7
library (pls)
set.seed (2)
pcr.fit=pcr(Salary~., data=Hitters ,scale=TRUE ,
validation ="CV")
summary(pcr.fit)
# 4
```