```
## 422 extra project
library(ISLR)
library(mice)
library(dplyr)
library(plyr)
library(forecast)
library(ggplot2)

House <- read.csv(file.path("kc_house_data.csv"))
summary(House)
str(House)
# dim(House)
# fix(House)
# attach(House)

# sum(is.na(House$price))
# House1 <- House[-1]
# House2 <- House1[-1]
# fix(House2)

##############################EDA

regfit <- lm(price~., data=House2)
par(mfrow=c(2,2))
plot(predict(regfit), residuals(regfit))
mynorm <- rnorm(10000, mean=0, sd=1)
qqplot(mynorm, House1$price, xlab = "Normal", ylab = "Sale Price", main = "QQ Plot
of (Sale Price)")



##visualize missing values
library(Amelia)
missmap(House,main = "Missing map values") ##confirmed that there is no missing
entry


##Create a historgram of log sale price
hist(log(House$price), main = "Histogram of Log Sale Price", xlab = "Log(Sale Price -
$)")
grid()

##Create a qqplot of log sale price
mynorm <- rnorm(10000, mean=0, sd=1)
qqplot(mynorm, log(rawdf$price), xlab = "Normal", ylab = "Log(Sale Price - $)",
main = "QQ Plot of Log(Sale Price)")
```

```
############

## transformations
House.t<-House2
House.t$price<-log(House.t$price)
fix(House.t)
regfit.t<-lm(price~., data=House.t)
plot(predict(regfit.t), residuals(regfit.t))
mynorm <- rnorm(10000, mean=0, sd=1)
qqplot(mynorm, House.t$price, xlab = "Normal", ylab = "log(Sale Price)", main = "QQ
Plot of (Sale Price)")


##a historgram of sale price
hist(House1$price, main = "Histogram of Sale Price", xlab = "Sale Price")
hist(House.t$price, main = "Histogram of Log Sale Price", xlab = "Log(Sale Price)")

## training data and test data
n<-dim(House.t)[1]
n
set.seed(1306)
test<-sample(n,round(n/4))
data.train<-House.t[-test,]
data.test<-House.t[test,]
x<-model.matrix(price~., data=House.t)[,-1]
x.train<-x[-test,]
x.test<-x[test,]
y<-House.t$price
y.train<-y[-test]
y.test<-y[test]
n.train<-dim(data.train)[1]
#n.train
n.test<-dim(data.test)[1]
#n.test

########## linear model with selected variables

library(MASS)
mylm3 = lm(data.train$price~., data=data.train)
mystep3 = stepAIC(mylm3, direction = "stepwise")
mystep3$anova
mypred3 = predict(mystep3, data=data.test)
print(mypred3)
print(mylm3)
summary(mylm3)
```

```
CV(mylm3)

pred3transform <- exp(mypred3)
summary(pred3transform)
#print(pred3transform)

#write.csv(pred3transform, "#2Both.csv")

############## Forward selection

library(MASS)
mylm = lm(data.train$price~., data=data.train)
mystep1 = stepAIC(mylm, direction = "forward")
mystep1$anova
mypred1 = predict(mystep1, data=data.test)
#print(mypred1)
summary(mylm)
CV(mylm)
pred1transform <- exp(mypred1)
summary(pred1transform)
#print(pred1transform)

#write.csv(pred1transform, "#3lmforward.csv")

############## Backward selection

library(MASS)
mylm2 = lm(data.train$price~., data=data.train)
mystep2 = stepAIC(mylm2, direction = "backward")
mystep2$anova
mypred2 = predict(mystep2, data.test)
#print(mypred2)
summary(mylm2)
CV(mylm2)
pred2transform <- exp(mypred2)
#print(pred2transform)
#write.csv(pred2transform, "#4lmbackward.csv")

########## PCA

library(pls)
pcamodel <- pcr(data.train$price~., data=data.train)
pcaprediction <- predict(pcamodel,data.test)
#print(pcaprediction)

fittedpca = pcamodel$fitted
```

```
#print(fittedpca)

pred5transform <- exp(fittedpca)
# print(pred5transform)
# write.csv(pred5transform, "#5pca.csv")
summary(pcamodel)
#pcamodel
RMSEP(pcamodel)
MSEP(pcamodel)

#############Rpart model
library(rpart)
rpartmodel <- rpart(data.train$price~., data=data.train)
rpartprediction <- predict(rpartmodel, data.test)
#print(rpartprediction)

summary(rpartmodel)
str(rpartmodel)

pred6transform <- exp(rpartprediction)
#print(pred6transform)
# write.csv(pred6transform, "#6Rpart.csv")
summary(rpartmodel)
# printcp(rpartmodel)
rsq.rpart(rpartmodel)
# residuals.rpart(rpartmodel)
??rpart

# ########## Random forest
#
# library(randomForest)
# myforest <- randomForest (y=exp(data.train$price),
#                           x = exp(data.train), ntree=1000, replace=TRUE, do.trace = 100,
#                           importance = TRUE)
#
# mypredict <- predict(myforest, exp(data.test))
# #predrftransform <- exp(mypredict)
# #summary(predrftransform)
# #print(mypredict)
# summary(mypredict)
#
# predftransform <- exp(myforest)
# #write.csv(forestsubmit, "#9myforest.csv")
# summary(predftransform)

###################
```

```
## BIC
library(leaps)
regfit.full=regsubsets(price~., data=data.train, nvmax=19)
summary(regfit.full)
reg.summary=summary(regfit.full)
reg.summary$bic
plot(reg.summary$bic,xlab="Number of variables", ylab="BIC", type="l")
which.min(reg.summary$bic)
points(16, reg.summary$bic[16], col="red", cex=2, pch=20)
coef(regfit.full, 16)
cat(names(coef(regfit.full,16)))
coefi=coef(regfit.full,id=16)
pred=coefi[1]+(x.test[,names(coefi[-1])]%*%coefi[-1])
mean((y.test-pred)^2)

## k=10 CV
predict.regsubsets=function(object, newdata,id,...){
 form=as.formula(object$call[[2]])
 mat=model.matrix(form,newdata)
 coefi=coef(object,id=id)
 xvars=names(coefi)
 mat[,xvars]%*%coefi
}
k=10
set.seed(1)
folds=sample(1:k, nrow(data.train), replace=TRUE)
cv.errors=matrix(NA, k, 10,dimnames=list(NULL, paste(1:10)))
for(j in 1:k){
 best.fit=regsubsets(price~., data=data.train[folds!=j,], nvmax=10)
 for(i in 1:10){
  pred=predict(best.fit, data.train[folds==j,],id=i)
  cv.errors[j,i]=mean((data.train$price[folds==j]-pred)^2)
 }
}
mean.cv.errors=apply(cv.errors,2,mean)
mean.cv.errors
which.min(mean.cv.errors)
plot(mean.cv.errors, type="b")
coef(regfit.full, 10)
regfit.best=regsubsets(price~., data.train, nvmax=10)
cv.pred=predict(regfit.best, data.test, id=10)
mean((cv.pred-y.test)^2)

## Lasso
library(glmnet)
```

```
grid=10^seq(10,-2, length=100)
lasso.mod=glmnet(x.train, y.train, alpha=1,lambda=grid)
plot(lasso.mod)
set.seed(1306)
cv.out = cv.glmnet(x.train, y.train, alpha = 1)
plot(cv.out)
largelam = cv.out$lambda.1se
largelam
lasso.pred=predict(lasso.mod, s=largelam, newx=x[test,])
mean((lasso.pred-y.test)^2)
predict(lasso.mod, type="coefficients", s=largelam)[1:19,]

## GAM
library(splines)
library(gam)
gam.house =gam(price~., data=data.train)
plot.gam(gam.house, se=TRUE,col="red")
summary(gam.house)
pred=predict(gam.house, newdata=data.test)
mean((pred-y.test)^2)
gam.house$coef
```