Machine Learning HW1
Carly Zhao, Li Li, Richard Yan

**Question 1**

1. (a) True. As K increases, the model will be less flexible and therefore the bias will increase.
   (b) False. As K increases, the model will be more stable and therefore the variance will reduce.
   (c) True. As K increases, the model is less fitted with the training dataset and therefore the misclassification rate will increase.
   (d) False. Because when K is very small, there is usually an overfitting issue, so as K increases, the misclassification rate will first decrease, and then increase.

2. (a) True. (2) is a more complex model than (1), so the estimate will have more variance.
   (b) True. (2) is less fitted than (3), so the estimate will have more bias
   (c) True. (3) uses the most complicated model to fit the training data, therefore will have the smallest training error
   (d) False. (1) is too simple and does not fit the data well, so it will have bigger test error

3. False. While unlikely, it is possible that the misclassification rate on a validation set is smaller than the training set if the validation set happens to fit the model better than the training set itself.

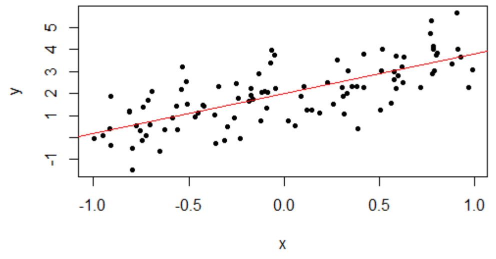
4. False. K-fold cross-validation does not provide an unbiased estimate of the predictive error of the models because the estimate could be sensitive to the random choice of folds.

**Question 2**

1. R code:

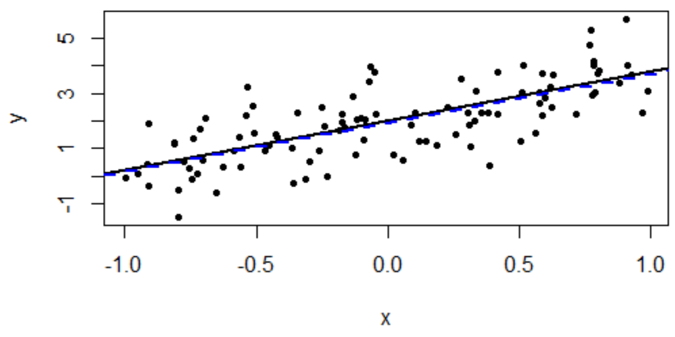
```
set.seed(123)
x = runif(100,-1,1)
e = rnorm(100)
y = 1.8*x+2+e

xt = runif(10000,-1,1)
et = rnorm(10000)
yt = 1.8*xt+2+et
```

2.

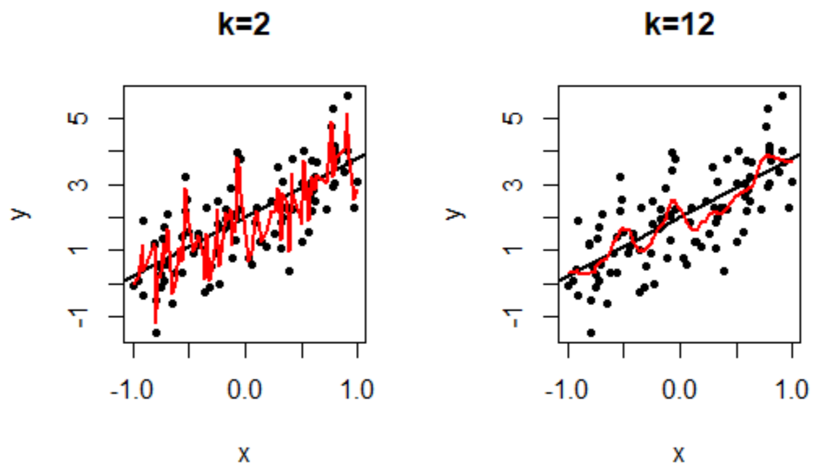


Machine Learning HW1
Carly Zhao, Li Li, Richard Yan

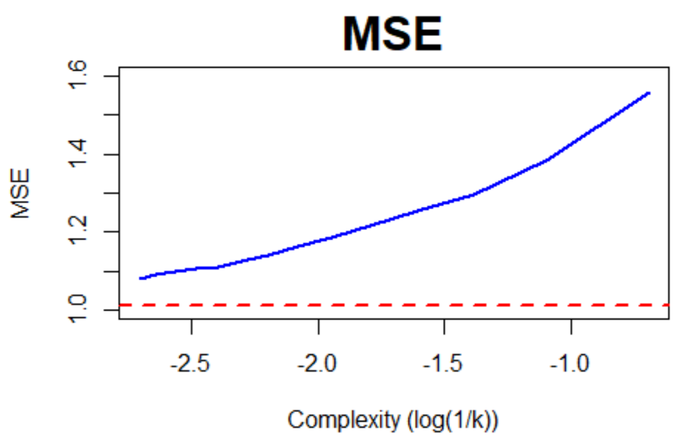
3.



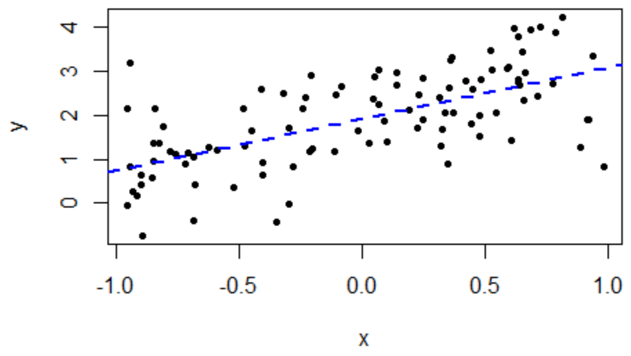
4.

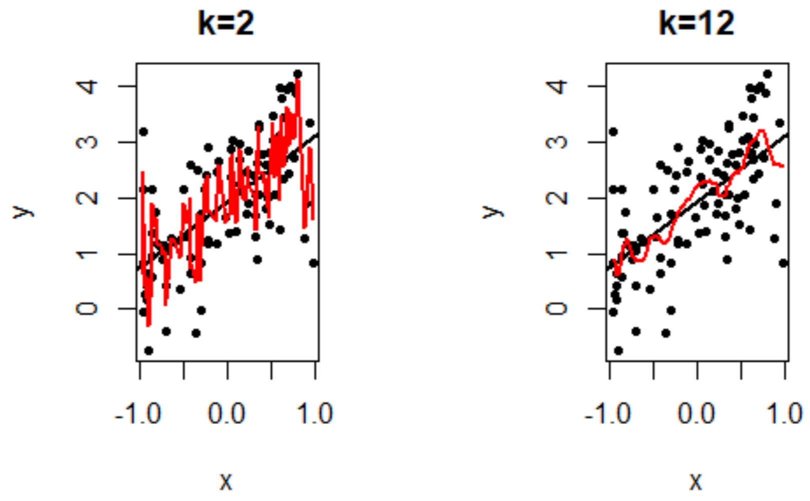


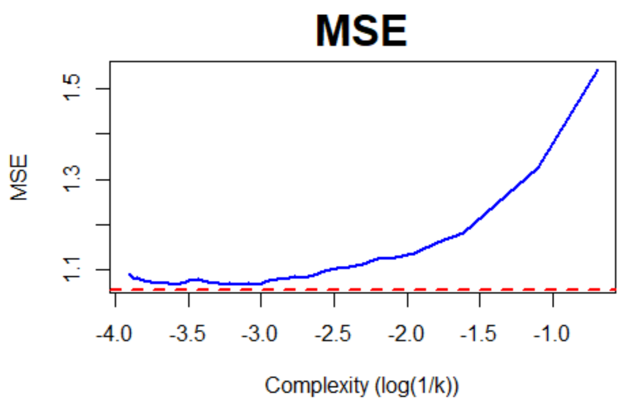
5. Linear regression performs the best. The bigger the k, the better performance the knn is. This is because the true statistical relationship is a simple linear relationship, so the bigger the k, the simpler the model which actually performs better.



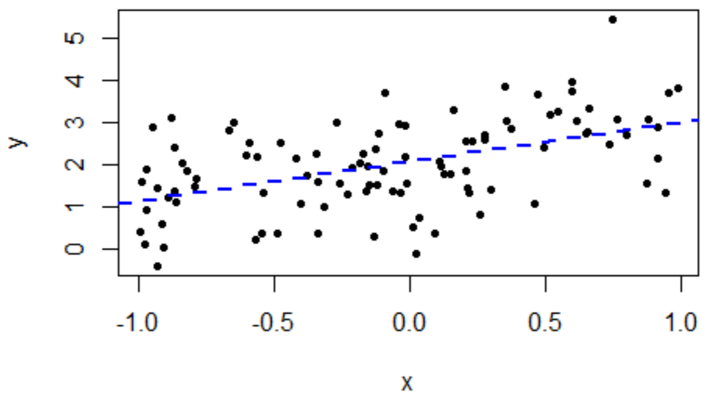
6. The conclusion is the same with question 5.

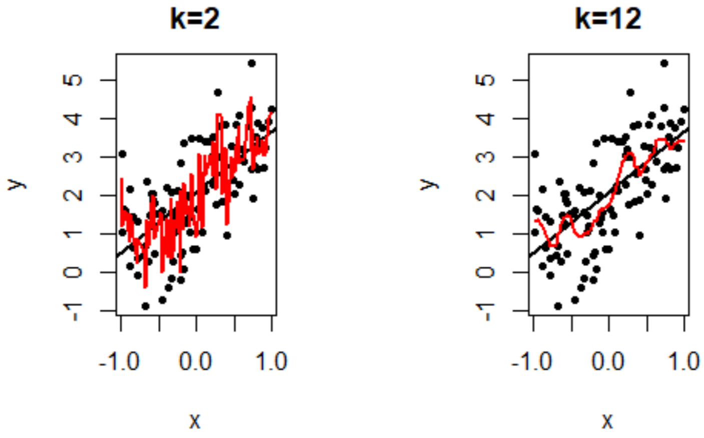


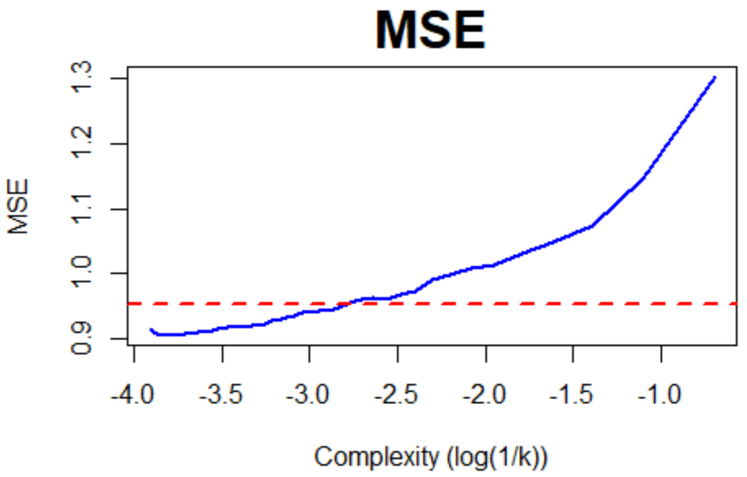




7. Now the knn has relatively better performance than linear regression as the MSE line for knn crossed with the horizontal line which means that when choosing the right k to minimum knn MSE, the knn model has smaller out-of-sample MSE than regression model.

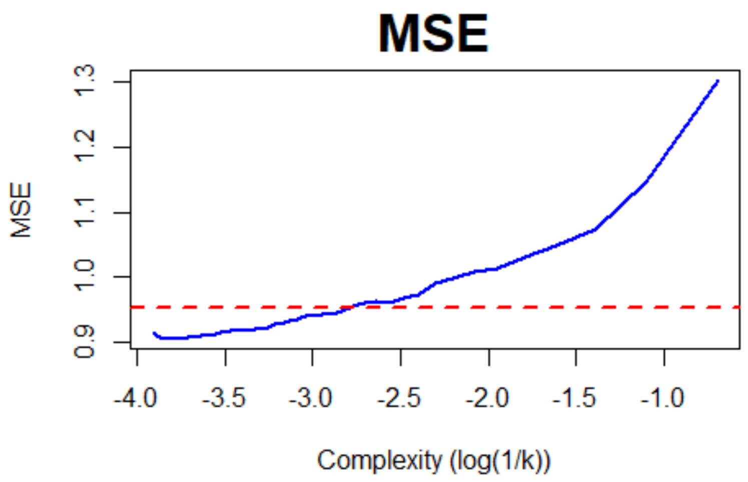






8. S
   P=1

Machine Learning HW1
Carly Zhao, Li Li, Richard Yan



P=2

**Question 3**

3.1 Find User that has rated the most amount of video games – **“U584295664”**

3.2 Find Video Game that has the most user ratings – **“I760611623”**

3.3 Find User that is most Similar to “U141954350” – **“U887577623”**

3.4 Recommend a video game to the user “U141954350” – **“I760611623”**

The code involved uses k fold Cross Validation in order to train the data set and polish it for the model to run the most popular recommendation scheme.

Machine Learning HW1
Carly Zhao, Li Li, Richard Yan

**Corresponding Code** (Also available:https://github.com/ysccoding/ml-hw1 in Code.R)

```
### Q2 ###

#1.#


set.seed(123)


# generate training data #

x = runif(100,-1,1)

e = rnorm(100)

y = 1.8*x+2+e


# generate test data #

xt = runif(10000,-1,1)

et = rnorm(10000)

yt = 1.8*xt+2+et


#2.#


plot(x,y,pch=20)

abline(2,1.8,lwd=2)


#3.#


fit = lm(y~x)
```

```
summary(fit)

b0 = fit$coefficients[1]

b1 = fit$coefficients[2]

abline(b0,b1,col="blue",lty=2,lwd=2)


#4.#

library(kknn)

train = data.frame(x=x, y=y)

test = train #test = train to get in sample knn fit#

ind = order(test[,1]) #sorting test makes the plots below easier to make.

test = test[ind,]

near1 = kknn(y~x,train,test,k=2)

near2 = kknn(y~x,train,test,k=12)


# draw the graph for k=2 and k=12 #

par(mfrow=c(1,2))

plot(x,y,pch=20,main="k=2")

abline(2,1.8,lwd=2)

lines(test[,1],near1$fitted,col=2,lwd=2)


plot(x,y,pch=20,main="k=12")

abline(2,1.8,lwd=2)

lines(test[,1],near2$fitted,col=2,lwd=2)
```

Machine Learning HW1
Carly Zhao, Li Li, Richard Yan

```
#5.#

train = data.frame(x=x,y=y)

test = data.frame(x=xt,y=yt)

out_MSE = NULL

kk = seq(2,15,by=1)


for(i in kk){


  near = kknn(y~x,train,test,k=i,kernel = "rectangular")

  aux = mean((test[,2]-near$fitted)^2)

  out_MSE = c(out_MSE,aux)

}




plot(log(1/kk),out_MSE,xlab="Complexity (log(1/k))",

     ylab="MSE", col=4, lwd=2,type="l", ylim=c(1,1.6))

title(main="MSE",cex.main=2)




fitted_reg = b0 + b1*xt

MSE_reg = mean((yt-fitted_reg)^2)

abline(h=MSE_reg, col="red",lwd=2,lty=2)


#6.#
```

Machine Learning HW1
Carly Zhao, Li Li, Richard Yan

```
x = runif(100,-1,1)

e = rnorm(100)

y = tanh(1.1*x)+2+e


# generate test data #

xt = runif(10000,-1,1)

et = rnorm(10000)

yt = tanh(1.1*xt)+2+e


plot(x,y,pch=20)




fit = lm(y~x)

summary(fit)

b0 = fit$coefficients[1]

b1 = fit$coefficients[2]

abline(b0,b1,col="blue",lty=2,lwd=2)


library(kknn)

train = data.frame(x=x, y=y)

test = train #test = train to get in sample knn fit#

ind = order(test[,1]) #sorting test makes the plots below easier to make.

test = test[ind,]

near1 = kknn(y~x,train,test,k=2)
```

```
near2 = kknn(y~x,train,test,k=12)



# draw the graph for k=2 and k=12 #

par(mfrow=c(1,2))

plot(x,y,pch=20,main="k=2")

abline(b0,b1,lwd=2)

lines(test[,1],near1$fitted,col=2,lwd=2)



plot(x,y,pch=20,main="k=12")

abline(b0,b1,lwd=2)

lines(test[,1],near2$fitted,col=2,lwd=2)



train = data.frame(x=x,y=y)

test = data.frame(x=xt,y=yt)

out_MSE = NULL

kk = seq(2,50,by=1)



for(i in kk){



  near = kknn(y~x,train,test,k=i,kernel = "rectangular")

  aux = mean((test[,2]-near$fitted)^2)

  out_MSE = c(out_MSE,aux)

}
```

```
par(mfrow=c(1,1))


plot(log(1/kk),out_MSE,xlab="Complexity (log(1/k))",

    ylab="MSE", col=4, lwd=2,type="l")

title(main="MSE",cex.main=2)




fitted_reg = b0 + b1*xt

MSE_reg = mean((yt-fitted_reg)^2)

abline(h=MSE_reg, col="red",lwd=2,lty=2)


#7.#

x = runif(100,-1,1)

e = rnorm(100)

y = sin(2*x)+2+e


# generate test data #

xt = runif(10000,-1,1)

et = rnorm(10000)

yt = sin(2*xt)+2+e


plot(x,y,pch=20)
```

Machine Learning HW1
Carly Zhao, Li Li, Richard Yan

```
fit = lm(y~x)

summary(fit)

b0 = fit$coefficients[1]

b1 = fit$coefficients[2]

abline(b0,b1,col="blue",lty=2,lwd=2)


library(kknn)

train = data.frame(x=x, y=y)

test = train #test = train to get in sample knn fit#

ind = order(test[,1]) #sorting test makes the plots below easier to make.

test = test[ind,]

near1 = kknn(y~x,train,test,k=2)

near2 = kknn(y~x,train,test,k=12)


# draw the graph for k=2 and k=12 #

par(mfrow=c(1,2))

plot(x,y,pch=20,main="k=2")

abline(b0,b1,lwd=2)

lines(test[,1],near1$fitted,col=2,lwd=2)


plot(x,y,pch=20,main="k=12")

abline(b0,b1,lwd=2)

lines(test[,1],near2$fitted,col=2,lwd=2)
```

```
train = data.frame(x=x,y=y)

test = data.frame(x=xt,y=yt)

out_MSE = NULL

kk = seq(2,50,by=1)


for(i in kk){


  near = kknn(y~x,train,test,k=i,kernel = "rectangular")

  aux = mean((test[,2]-near$fitted)^2)

  out_MSE = c(out_MSE,aux)

}


par(mfrow=c(1,1))


plot(log(1/kk),out_MSE,xlab="Complexity (log(1/k))",

     ylab="MSE", col=4, lwd=2,type="l")

title(main="MSE",cex.main=2)




fitted_reg = b0 + b1*xt

MSE_reg = mean((yt-fitted_reg)^2)

abline(h=MSE_reg, col="red",lwd=2,lty=2)


#8.#
```

Machine Learning HW1
Carly Zhao, Li Li, Richard Yan

```
x = runif(100,-1,1)

e = rnorm(100)

y = sin(2*x)+2+e


X=NULL

p = seq(2,20,1)

for (n in p){

  X[n]=rnorm(100)


}




# generate test data #

xt = runif(10000,-1,1)

et = rnorm(10000)

yt = sin(2*xt)+2+e


###########################

### Q3 Setup ####

###########################

PackageList=c("MASS",

              "ISLR",

              "animation",
              "ElemStatLearn",

              "glmnet",

              "textir",

              "nnet",

              "methods",

              "statmod",

              "stats",

              "graphics",

              "RCurl",

              "jsonlite",

              "tools",

              "utils",

              "data.table",

              "gbm",

              "ggplot2",

              "randomForest",

              "tree",

              "class",

              "kknn",

              "e1071",

              "data.table",

              "R.utils",

              "recommenderlab")

NewPackages=PackageList[!(PackageList %in%
                     installed.packages()[,"Package"])]

if(length(NewPackages)) install.packages(NewPackages)

lapply(PackageList,require,character.only=TRUE)#array function




download.file('https://github.com/ChicagoBoothML/MLClassData/raw/master/Amazon/videoG
ames.json.gz', 'videoGames.json.gz')


fileConnection <- gzcon(file("videoGames.json.gz", "rb"))

InputData = stream_in(fileConnection)




ratingData = as(InputData[c("reviewerID", "itemID", "rating")], "realRatingMatrix")


# we keep users that have rated more than 2 video games

ratingData = ratingData[rowCounts(ratingData) > 2,]


# we will focus only on popular video games that have

# been rated by more than 3 times

ratingData = ratingData[,colCounts(ratingData) > 3]


# we are left with this many users and items

dim(ratingData)
```

```
# example on how to recommend using Popular method

r = Recommender(ratingData, method="POPULAR")


# recommend 5 items to user it row 13

rec = predict(r, ratingData[13, ], type="topNList", n=5)

as(rec, "list")


# predict ratings

rec = predict(r, ratingData[13, ], type="ratings")

as(rec, "matrix")


#### Q3.1 - Find User that has rated the most amount of video games ####

userIndex = which.max(rowCounts(ratingData, na.rm = TRUE)) # Find Row(Users) with Highest non-NA ratings

mostRatedUser = rownames(ratingData)[userIndex]

mostRatedUser # Printout - TODO: REMOVE


#### Q3.2 - Find Video Game that has the most user ratings ####

videoGameIndex = which.max(colCounts(ratingData, na.rm = TRUE)) # Find Col(Games) with highest non-NA ratings

mostRatedVideoGame = colnames(ratingData)[videoGameIndex]

mostRatedVideoGame # Printout - TODO: REMOVE


#### Q3.3 - Find User that is most Similar to “U141954350” ####

userID = "U141954350"
```

```r
# Transform NA's to 0s for Similarity Analysis

transformedRatingData <- as(ratingData,"matrix") # coerce to matrix so that NA can be transformed

transformedRatingData <- replace(transformedRatingData, is.na(transformedRatingData),0) # replace all NA with 0

transformedRatingData <- as(transformedRatingData, "realRatingMatrix") # coerce back to realRatingMatrix


# Gather Matrix of User & non-User ratings for Similarity Comparison

userRatings = transformedRatingData[userID]

otherUserRatings = transformedRatingData[!rownames(transformedRatingData) == userID]


# Using Pearson method for Ratings Similarities

sim <- similarity(userRatings, otherUserRatings, method = "pearson", which = "users") # Finding similarity vector of all other users

similarUser = colnames(sim)[which.max(sim)] # Finding Most Similar User

similarUser # Printout - TODO: REMOVE


#### Q3.4 - Recommend a video game to the user “U141954350 ####

userID = "U141954350"

currentUser = ratingData[userID]


# Train Data for Recommendation Model

scheme = evaluationScheme(transformedRatingData, method="cross-validation", k = 15, train = 0.8, given = 2)
```

```
trainedRatings = getData(scheme, "train")



# Predict

recModel = Recommender(trainedRatings, method = "POPULAR")

rec = predict(recModel, currentUser, type = "topNList", n = 1)

rec = as(rec, "list")

rec # Printout - TODO: REMOVE
```