IntroCV Assignment #2: Wine Prediction

```
library(tidyverse)
```

```
## -- Attaching packages --------- tidyverse 1.3.0 --

## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Assignment 2: Wine Quality Prediction

Go to: https://archive.ics.uci.edu/ml/datasets/Wine+Quality Use the white wine data set. Build a model to predict quality as a function of the predictors. Compare linear regression with KNN (using knn.reg) For linear regression, use CV and/or bootstrap to determine the best (or at least a good) set of predictors. For KNN, determine the best choice of k.

Note: before starting the modeling, scale the predictors data before applying knn.

Note: treat quality as a quantitative predictor. Could also do this as qualititative but more complicated.

Solution.

Read the data

```
wine.df <- read_delim("wine_quality_white.csv",delim=";")
```

```
## Parsed with column specification:
## cols(
##   `fixed acidity` = col_double(),
##   `volatile acidity` = col_double(),
##   `citric acid` = col_double(),
##   `residual sugar` = col_double(),
##   chlorides = col_double(),
##   `free sulfur dioxide` = col_double(),
##   `total sulfur dioxide` = col_double(),
##   density = col_double(),
##   pH = col_double(),
##   sulphates = col_double(),
##   alcohol = col_double(),
##   quality = col_double()
## )
```

```
dim(wine.df)
```

```
## [1] 4898   12
```

```
N <- nrow(wine.df)
```

Try a simple linear model, use all the predictors.

```
N <- nrow(wine.df)
numFolds <- 10
folds <- sample(1:numFolds,N,rep=T)
errs <- numeric(numFolds)
for(fold in 1:numFolds){
 train.df <- wine.df[folds != fold,]
 test.df <- wine.df[folds == fold,]
 mod.lm <- lm(quality ~ ., data=train.df)
 pred <- predict(mod.lm, newdata=test.df)
 errs[fold] <- with(test.df,mean((quality-pred)^2))
}
mse.lm <- mean(errs)
```

Ok, we have baseline. Instead of systematically going through the predictor set (say via a subset selection), just explore other predictor sets and sizes. We could run over a random selection of the 11 predictors

Let numPreds be the number of predictors to try and use a random subset of the predictors of that size.

```
totPreds <- 11
numPreds <- 2
sampPreds <- sort(sample(1:totPreds,numPreds,rep=F))
data.df <- wine.df[,c(sampPreds,totPreds+1)]
```

Now build a model on these.

```
numFolds <- 10
folds <- sample(1:numFolds,N,rep=T)
errs <- numeric(numFolds)
for(fold in 1:numFolds){
 train.df <- data.df[folds != fold,]
 test.df <- data.df[folds == fold,]
 mod.lm <- lm(quality ~ ., data=train.df)
 pred <- predict(mod.lm, newdata=test.df)
 errs[fold] <- with(test.df,mean((quality-pred)^2))
}
(mse.lm <- mean(errs))
```

```
## [1] 0.7768045
```

Looks ok, make a function. Return both the mse and the set of predictors in case we want to save these.

```
doRandPreds <- function(numPreds){
  sampPreds <- sort(sample(1:totPreds,numPreds,rep=F))
  data.df <- wine.df[,c(sampPreds,totPreds+1)]
  folds <- sample(1:numFolds,N,rep=T)
  errs <- numeric(numFolds)
  for(fold in 1:numFolds){
    train.df <- data.df[folds != fold,]
    test.df <- data.df[folds == fold,]
    mod.lm <- lm(quality ~ ., data=train.df)
    pred <- predict(mod.lm, newdata=test.df)
    errs[fold] <- with(test.df,mean((quality-pred)^2))
  }
  mse.lm <- mean(errs)
  c(mse.lm,sampPreds)
}
```

Try this out...

```
doRandPreds(5)
```

```
## [1]  0.6745467  2.0000000  6.0000000  7.0000000  8.0000000 10.0000000
```

After a lot of searching, it wasn't clear that I could beat using the full set of predictors

```
(res <- doRandPreds(11))
```

```
##  [1]  0.5685813  1.0000000  2.0000000  3.0000000  4.0000000  5.0000000
##  [7]  6.0000000  7.0000000  8.0000000  9.0000000 10.0000000 11.0000000
```

```
(mse.lm <- res[1])
```

```
## [1] 0.5685813
```

Now try KNN (regression)

```
library(FNN) ## knn.reg
```

Build the data matrices

```
wine.mat <- data.matrix(wine.df)
##just the predictors, scaled
wine.x <- scale(wine.mat[,-12])
## the response
wine.y <- wine.mat[,12]
## check that these are close to 0
colMeans(wine.x)
```

```
##        fixed acidity     volatile acidity          citric acid
##        -3.483203e-16        -1.166607e-16         1.446288e-17
##       residual sugar            chlorides  free sulfur dioxide
##         6.149379e-17        -2.049226e-17        -1.207436e-17
## total sulfur dioxide              density                   pH
##        -5.283646e-17        -4.666852e-15        -1.195697e-15
##            sulphates              alcohol
##         1.512514e-16        -3.725335e-16
```

Practice with a single kVal

```
kVal <- 10
## build the folds
numFolds <- 10
folds <- sample(1:numFolds,N,rep=T)
## Ready to cross-validate
errs <- numeric(numFolds)
for(fold in 1:numFolds){
 train.x <- wine.x[folds != fold,]
 test.x <- wine.x[folds == fold,]
 train.y <- wine.y[folds != fold]
 test.y <- wine.y[folds == fold]
 mod.knn <- knn.reg(train.x,test.x,train.y,k=kVal)
 pred <- mod.knn$pred
 errs[fold] <- mean((test.y-pred)^2)
}
mse.knn <- mean(errs)
##compare with lm....
c(mse.lm,mse.knn)
```

```
## [1] 0.5685813 0.4903417
```

Make this a function and repeat over a range of kVals

```
mseKNN <- function(kVal){
  folds <- sample(1:numFolds,N,rep=T)
  errs <- numeric(numFolds)
  for(fold in 1:numFolds){
    train.x <- wine.x[folds != fold,]
    test.x <- wine.x[folds == fold,]
    train.y <- wine.y[folds != fold]
    test.y <- wine.y[folds == fold]
    mod.knn <- knn.reg(train.x,test.x,train.y,k=kVal)
    pred <- mod.knn$pred
    errs[fold] <- mean((test.y-pred)^2)
  }
  mean(errs)
}
```

Try it out...

```
mseKNN(2)
```

```
## [1] 0.5237995
```

```
mseKNN(30)
```

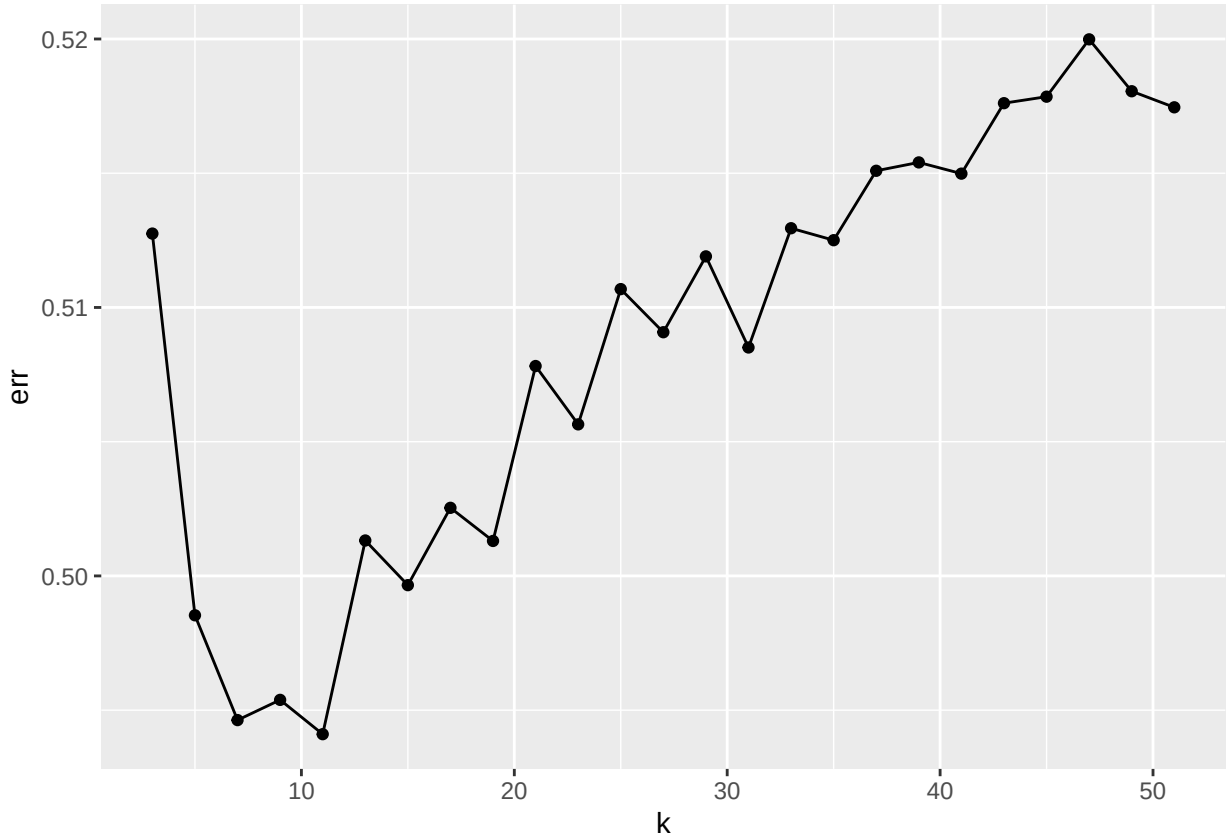
```
## [1] 0.514071
```

Now we can explore a range of k values (only odd values is a good idea)

```
maxK <- 25
#only odd values
kVals <- 2*(1:maxK)+1
## this takes a moment or two...
errsKNN <- map_dbl(kVals,mseKNN)
```

Quick plot..

```
data.frame(k=kVals,err=errsKNN) %>%
  ggplot()+
  geom_point(aes(k,err))+
    geom_line(aes(k,err))
```



Looks like an identifiable min.

```
idOpt <- which.min(errsKNN)
(kOpt <- kVals[idOpt])
```

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

```
(mse.knn <- errsKNN[idOpt])
```

```
## [1] 0.4941063
```

KNN seems to be the winner!

```
c(mse.lm,mse.knn)
```

```
## [1] 0.5685813 0.4941063
```