# Problem Set 2

*Liz Masten*

*10/11/2020*

## Conceptual Questions

**1. The table below contains a training dataset of 6 observations, 3 predictors and 1 qualitative outcome variable. Suppose we wish to use this data set to make a prediction for Y when X1= X2 = X3 = 1 using K-nearest neighbors. (2 pts)**

(a) Compute the Euclidean distance between observation 3 and the test point, X1 = X2 =X3 = 1

$$\sqrt{(0-1)^2 + (0-1)^2 + (0-1)^2} = \sqrt{3} = 1.732$$

(b) Using Euclidean distance, what is our prediction for an observation with X1= X2 = X3 = 1 with K = 1? Why?

The nearest neighbor to test point (1,1,1) is black, because black has the lowest Euclidean distance to the test point (observation 4 with a distance of 1).

(c) Using Euclidean distance, what is our prediction with K = 3? Why?

Using the three lowest Euclidean distances, we get White (Obs 3), Black (Obs 4), and White (Obs 2 OR 5- they both have a Euclidean distance of 2 but we only have room for one. Luckily both observations are the same color). White wins.

(d) Suppose the table looked like this instead (i.e. a regression problem)

2

**2. Let's say I have a model with 30 potential covariates. How many potential variants on the models can I have? What does this imply about the tradeoff between forward/backward selection over best subset selection. (0.5 pts)**

We can have this many potential variants:

```
2^30
```

```
## [1] 1073741824
```

Best subset selection will be compuationally intense. Forward selection will be less intense because it starts with no predictors and adds until no further additions are warranted, while backwards selection will be intense because it starts with all predictors and takes away observations one by one.

**3. If the underlying data is highly linear, we would expect QDA to outperform LDA. True or False? (0.5 pts)**

False- LDA outperforms QDA when data is linear because LDA is Linear Discriminant Analysis.

**4. We have a dataset of genetics sequencing outcome, with 30 observations and 4000 variables. You are trying to determine the best method for regression analysis. Colleague A is advocating for KNN, Colleague B is advocating for linear regression, Colleague C thinks Colleague A and B are both wrong. Who should you side with? (1 pt)**

Parametric models generally outperform non-parametric models on datasets with a small number of observations per feature. Because KNN is non-parametric, this is not a good choice because it may lead to overfitting. However, linear regression does not perform well when the number of features exceeds the number of observations. Therefore, Colleague C is correct- neither KNN nor linear regression would be good for this dataset.

**5. We have a dataset of genetics sequencing outcome, with 3,000 observations and 40,000 variables. You are trying to determine the best method for classification analysis. Colleague A is advocating for QDA but Colleague C is worried. Why might she be concerned? (1 pt)**

Again, we have more variables than observations, making this dataset very wide. QDA requires estimation of more parameters, which is not good when the dataset already has more variables than observations. This will likely lead to overfitting and is a poor choice in this scenario.

**6. You have a dataset that is all dummy variables (i.e. 0/1 categorical variables). If you want to use a linear decision boundary, would you expect LDA or a logistic regression to perform better? (0.5 pts)**

Logistic regression will perform better because logistic regressions are bounded between 0 and 1.

**7. What is the shrinkage penalty for ridge regression? (0.5pts)**

Ridge regression uses L2 for shrinkage penalty, which means that it uses the square of the sum of the coefficients.

**8. What is the shrinkage penalty for lasso regression? (0.5pts)**

The shrinkage penalty for Lasso uses L1, meaning that it takes the absolute value of the sum of the coefficients.

**9. How do the different shrinkage penalties influence variable selection for lasso vs ridge? (0.5pts)**

Lasso regressions can perform variable selection by making some variables 0, meaning they can be excluded from the model. Ridge regressions can drive some coefficients close to zero, but they will never be zero. Thus, Ridge does not perform variable selection (it would then be up to the researcher to decide how to proceed with the coefficients post-Ridge regression).

# Data Questions

**1. How many observations have missing values for at least one feature? Drop those observations for now. (1 pt)**

One observation contained at least one NA value. Here's the code:

```
data("Fatalities")

nrow(Fatalities)
```

```
## [1] 336
```

```
data <- Fatalities %>%
  drop_na()


nrow(data)
```

```
## [1] 335
```

**2. Which variables are categorical variables? How many classes do each of these categorical variables have? (1 pt)**

5 variables are categorical:

state (Factor) year (Factor) breath (Factor) jail (Factor) service (Factor)

```
#this is huge and we don't need to see it:

cat_vars <- str(data)
```

```
## 'data.frame':    335 obs. of  34 variables:
##  $ state       : Factor w/ 48 levels "al","az","ar",..: 1 1 1 1 1 1 1 2 2 2 ...
##  $ year        : Factor w/ 7 levels "1982","1983",..: 1 2 3 4 5 6 7 1 2 3 ...
##  $ spirits     : num  1.37 1.36 1.32 1.28 1.23 ...
##  $ unemp       : num  14.4 13.7 11.1 8.9 9.8 ...
##  $ income      : num  10544 10733 11109 11333 11662 ...
##  $ emppop      : num  50.7 52.1 54.2 55.3 56.5 ...
##  $ beertax     : num  1.54 1.79 1.71 1.65 1.61 ...
##  $ baptist     : num  30.4 30.3 30.3 30.3 30.3 ...
##  $ mormon      : num  0.328 0.343 0.359 0.376 0.393 ...
##  $ drinkage    : num  19 19 19 19.7 21 ...
##  $ dry         : num  25 23 24 23.6 23.5 ...
##  $ youngdrivers: num  0.212 0.211 0.211 0.211 0.213 ...
##  $ miles       : num  7234 7836 8263 8727 8953 ...
##  $ breath      : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ jail        : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 2 2 2 ...
##  $ service     : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 2 2 2 ...
##  $ fatal       : int  839 930 932 882 1081 1110 1023 724 675 869 ...
##  $ nfatal      : int  146 154 165 146 172 181 139 131 112 149 ...
##  $ sfatal      : int  99 98 94 98 119 114 89 76 60 81 ...
```

```
##  $ fatal1517   : int  53 71 49 66 82 94 66 40 40 51 ...
##  $ nfatal1517  : int  9 8 7 9 10 11 8 7 7 8 ...
##  $ fatal1820   : int  99 108 103 100 120 127 105 81 83 118 ...
##  $ nfatal1820  : int  34 26 25 23 23 31 24 16 19 34 ...
##  $ fatal2124   : int  120 124 118 114 119 138 123 96 80 123 ...
##  $ nfatal2124  : int  32 35 34 45 29 30 25 36 17 33 ...
##  $ afatal      : num  309 342 305 277 361 ...
##  $ pop         : num  3942002 3960008 3988992 4021008 4049994 ...
##  $ pop1517     : num  209000 202000 197000 195000 204000 ...
##  $ pop1820     : num  221553 219125 216724 214349 212000 ...
##  $ pop2124     : num  290000 290000 288000 284000 263000 ...
##  $ milestot    : num  28516 31032 32961 35091 36259 ...
##  $ unempus     : num  9.7 9.6 7.5 7.2 7 ...
##  $ emppopus    : num  57.8 57.9 59.5 60.1 60.7 ...
##  $ gsp         : num  -0.0221 0.0466 0.0628 0.0275 0.0321 ...
```

Now, consider the prediction problem where you want to predict number of single vehicle fatalities (Fatalities) given all other variables available in the data set.

**3. Convert the categorical variables to indicator variables (also called "dummy" variables) and run a linear regression. What is the adjusted R2? (1 pt)**

Adjusted R-squared: 0.981

```
# making all factors binary, even year and state, which is kind of a new idea to me.

# move column 19 (sfatal) to last column

binary <- data %>%
          dummy_cols(select_columns = c("state", "breath", "jail", "service", "year"),
                     remove_selected_columns = TRUE, remove_first_dummy = TRUE) %>%
          select(-fatal, -nfatal, -fatal1517, -nfatal1517, -fatal1820, -nfatal1820, -fatal2124, -nfatal
          select_(.dots = c(setdiff(names(.), 'sfatal'), 'sfatal'))
```

```
## Warning: select_() is deprecated.
## Please use select() instead
##
## The 'programming' vignette or the tidyeval book can help you
## to program with select() : https://tidyeval.tidyverse.org
## This warning is displayed once per session.
```

```
model_3 <- lm(sfatal ~ ., data = binary)

#This is where the R Squared is, but the print-out is huge, so I won't show it.

find_r2 <- summary(model_3)
```

**4. Run lasso regression with cross-validation using the canned function cv.glmnet from the package glmnet. You can use the Lambda sequence generated by grid function we used in section notes 4. In order to receive credit for this question, make the line immediately preceding this command say set.seed(222) and run the two lines together. Please report all numbers by rounding to three decimal places. (2 pts)**

```
whereami <- colnames(binary)

# sfatal is now column 76

# will this get rid of a weird error msg?

binary$breath <- as.numeric(binary$breath)
binary$jail <- as.numeric(binary$jail)
binary$service <- as.numeric(binary$service)



set.seed(222)


lasso <- cv.glmnet(x = as.matrix(binary[,1:75]),
                   y = as.numeric(binary[,76]),
                   nfold = 5,
                   standardize = TRUE)

print(lasso$lambda)
```

```
##  [1] 101.84079736  92.79354161  84.55001913  77.03882847  70.19491128
##  [6]  63.95898884  58.27704855  53.09987618  48.38262953  44.08445008
## [11]  40.16810904  36.59968494  33.34826981  30.38570143  27.68631947
## [16]  25.22674315  22.98566882  20.94368536  19.08310607  17.38781553
## [21]  15.84312992  14.43566993  13.15324480  11.98474678  10.92005491
## [26]   9.94994733   9.06602143   8.26062108   7.52677027   6.85811274
## [31]   6.24885691   5.69372568   5.18791079   4.72703109   4.30709468
## [36]   3.92446426   3.57582567   3.25815917   2.96871329   2.70498099
## [41]   2.46467793   2.24572274   2.04621892   1.86443847   1.69880689
## [46]   1.54788957   1.41037933   1.28508512   1.17092170   1.06690024
## [51]   0.97211976   0.88575932   0.80707091   0.73537295   0.67004445
## [56]   0.61051954   0.55628267   0.50686405   0.46183565   0.42080744
## [61]   0.38342407   0.34936173   0.31832540   0.29004625   0.26427934
## [66]   0.24080149   0.21940935   0.19991763   0.18215750   0.16597513
## [71]   0.15123035   0.13779547   0.12555410   0.11440022   0.10423722
## [76]   0.09497707   0.08653957   0.07885163   0.07184667   0.06546401
## [81]   0.05964837   0.05434937   0.04952112   0.04512180   0.04111331
## [86]   0.03746091   0.03413299   0.03110071   0.02833781   0.02582035
## [91]   0.02352655   0.02143651   0.01953215   0.01779697   0.01621594
```

- Which Lambda had the lowest mean cross-validation error for 5 fold cross validation?

```
print(lasso$lambda.min) %>% round(digits = 3)
```

```
## [1] 0.09497707
```

```
## [1] 0.095
```

- What was the cross-validation error?

```
print(lasso$lambda.min) %>% round(digits = 3)
```

```
## [1] 0.09497707
```

```
## [1] 0.095
```

- What was the standard error of the mean cross-validation error for this value of Lambda?

47.12

```
ver1 <- do.call( 'cbind', list(lambda = lasso$lambda,
                               cross_validation_error = lasso$cvm,
                               standard_error = lasso$cvsd,
                               cvup = lasso$cvup,
                               cvlo = lasso$cvlo))

ver2 <- data.frame(ver1) %>%
  arrange(cross_validation_error) %>%
  head(1) %>%
  round(3)

ver2
```

```
##   lambda cross_validation_error standard_error    cvup    cvlo
## 1  0.095                296.596          47.12 343.716 249.476
```

- What was the largest value of Lambda whose mean cross validation error was within one standard deviation of the lowest cross-validation error?

```
#lambda.1se tells us thelambda w/in 1 SE of min lambda

print(lasso$lambda.1se) %>% round(digits = 3)
```

```
## [1] 0.8070709
```

```
## [1] 0.807
```

**5. Using the same data, implement your own 5-fold cross-validation routine for KNN for k = 1, . . . , 20 (e.g. write the cross-validation routine yourself rather than using a canned package). In the 90s, a popular policy response to high rates of alcohol related fatalities was to increase taxes on alcohol. Consider the prediction problem of predicting beer tax (tax on cases of beer) using all of the other variables. Include the snippet of code you wrote here. It should not exceed 20 lines. Which k is best according to CV? (2 pts)**

The best k is 12.

```r
cross_validation_KNN  <- function(data_x, data_y, k_seq, kfolds) {


    fold_ids      <- rep(seq(kfolds),
                         ceiling(nrow(data_x) / kfolds))

    fold_ids      <- fold_ids[1:nrow(data_x)]

    fold_ids      <- sample(fold_ids, length(fold_ids))

    CV_error_mtx  <- matrix(0,
                            nrow = length(k_seq),
                            ncol = kfolds)

    for (k in k_seq) {
      for (fold in 1:kfolds) {

        knn_fold_model    <- knn(train = data_x[which(fold_ids != fold),],
                                 test = data_x[which(fold_ids == fold),],
                                 cl = data_y[which(fold_ids != fold)],
                                 k = k)

        CV_error_mtx[k,fold]  <- mean(knn_fold_model !=
                                        data_y[which(fold_ids == fold)])
      }
    }

    return(CV_error_mtx)
}

set.seed(222)

knn_cv_error  <- cross_validation_KNN(data_x = binary[,-76],
                                                data_y = binary[,76],
                                                k_seq = seq(20),
                                                kfolds = 5)
print(knn_cv_error)
```

```
##            [,1]      [,2]      [,3]      [,4]      [,5]
##  [1,] 0.9850746 0.9552239 0.9402985 1.0000000 0.9701493
##  [2,] 0.9850746 0.9701493 0.9402985 0.9701493 0.9552239
##  [3,] 0.9850746 0.9850746 0.9701493 0.9850746 0.9701493
##  [4,] 0.9552239 0.9402985 0.9552239 0.9701493 0.9701493
##  [5,] 0.9402985 0.9402985 0.9552239 0.9850746 0.9850746
##  [6,] 0.9402985 0.9402985 0.9552239 0.9552239 0.9701493
##  [7,] 0.9552239 0.9552239 0.9701493 0.9552239 0.9701493
##  [8,] 0.9701493 0.9552239 0.9850746 0.9253731 0.9552239
##  [9,] 0.9552239 0.9552239 0.9701493 0.9701493 0.9701493
## [10,] 0.9402985 0.9552239 0.9850746 0.9552239 0.9402985
## [11,] 0.9552239 0.9701493 0.9850746 0.9402985 0.9402985
## [12,] 0.9402985 0.9552239 0.9701493 0.9552239 0.9104478
## [13,] 0.9552239 0.9552239 0.9701493 0.9701493 0.9253731
## [14,] 0.9552239 0.9552239 0.9701493 0.9552239 0.9402985
```

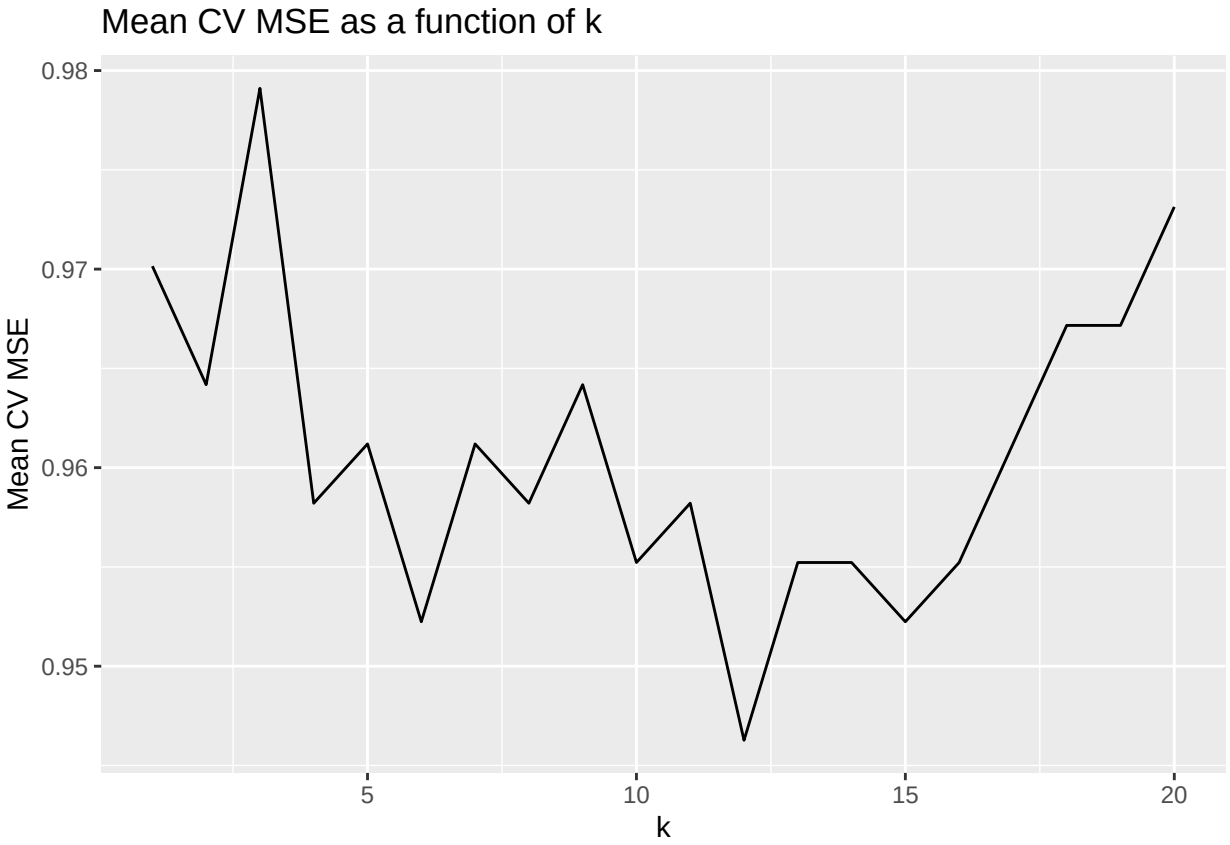
```
## [15,] 0.9701493 0.9701493 0.9253731 0.9552239 0.9402985
## [16,] 0.9701493 0.9701493 0.9402985 0.9552239 0.9402985
## [17,] 0.9701493 0.9402985 0.9552239 0.9701493 0.9701493
## [18,] 0.9850746 0.9701493 0.9402985 0.9701493 0.9701493
## [19,] 0.9701493 0.9850746 0.9402985 0.9850746 0.9552239
## [20,] 0.9701493 0.9850746 0.9701493 0.9701493 0.9701493
```

```
mean_cv_error <- rowMeans(knn_cv_error)

x <- c(1:20)

df <- as.data.frame(mean_cv_error)

df <- tibble::rowid_to_column(df, "k")

df
```

```
##     k mean_cv_error
## 1   1     0.9701493
## 2   2     0.9641791
## 3   3     0.9791045
## 4   4     0.9582090
## 5   5     0.9611940
## 6   6     0.9522388
## 7   7     0.9611940
## 8   8     0.9582090
## 9   9     0.9641791
## 10 10     0.9552239
## 11 11     0.9582090
## 12 12     0.9462687
## 13 13     0.9552239
## 14 14     0.9552239
## 15 15     0.9522388
## 16 16     0.9552239
## 17 17     0.9611940
## 18 18     0.9671642
## 19 19     0.9671642
## 20 20     0.9731343
```

**6. Plot mean cross-validation MSE as a function of k. Label the y-axis "Mean CV MSE" and the x-axis "k". (1 pt)**

```
plot <- ggplot(df,
               aes(x =k,
                   y = mean_cv_error)) +
        geom_line() +
        labs(title = "Mean CV MSE as a function of k",
             x = "k",
             y = "Mean CV MSE")
plot
```


Mean CV MSE as a function of k
Mean CV MSE
0.98
0.97
0.96
0.95
5
10
15
20
k