API 222 Problem Set 2

Machine Learning and Big Data Analytics: Spring 2024

Oscar Boochever

This problem set is worth 30 points in total. To get full credit, submit your code along with a write-up of your answers. This should either be done in R Markdown or Jupyter Notebook, submitted in one knitted PDF.

Final Project Groups (0 pts)

Please join one of 30 final project groups that have been created on this Canvas page. Details about the final project can be found on this Canvas page. You just need to form your group by March 7. The main project milestones will be after the midterm. We recommend forming groups of 5 students. All students working together should join the same group. PhD students need to work individually (see details on Canvas). If you are a PhD student or are otherwise working alone, please form a group by yourself. Please email Jacob if you have questions about this assignment or the final project.

Done – Final Project Team #3 on Canvas.

Conceptual Questions (15 pts)

1. Consider the four main classification methods that have been presented thus far this semester: logistic regression, k-Nearest Neighbors, linear discriminant analysis (LDA), and quadratic discriminant analysis (QDA. Which of these methods may be appropriate if you know the decision boundary between the classes is linear? (3pts)

Logistic regression and LDA.

2. Suppose you had the following data and you are using KNN Regression with Euclidean distance. Consider the prediction problem where you want to predict Y for the data point X1 = X2 = X3 = 0.

X1	X2	X3	Y
0	3.5	2	2
1	2.1	3	1
2	4.7	1	3
1	3.9	1	2
0	2.9	2	4
1	1.5	2	1
1	3.5	4	2

(a) Compute the Euclidean distance between each observation and the test point, X1 = X2 = X3 = 0. (1pt)

```r
# Make the above data a dataframe
euc_data <- data.frame(
  X1 = c(0, 1, 2, 1, 0, 1, 1),
  X2 = c(3.5, 2.1, 4.7, 3.9, 2.9, 1.5, 3.5),
  X3 = c(2, 3, 1, 1, 2, 2, 4),
  Y = c(2, 1, 3, 2, 4, 1, 2)
)

# Define the x1 = x2 = x3 = 0 as a point
test_point <- c(0, 0, 0)

# Create a function for Euclidean distance
euclidean_distance_knn <- function(x1, x2, x3, test_point) {
  sqrt((x1 - test_point[1])^2 + (x2 - test_point[2])^2 + (x3 - test_point[3])^2)
}

# Apply function to data
euc_data$distance_from_test <- round(euclidean_distance_knn(euc_data$X1, euc_data$X2, euc_data$X3, test_

print(euc_data[, c("X1", "X2", "X3", "Y", "distance_from_test")])
```

```
##   X1  X2 X3 Y distance_from_test
## 1  0 3.5  2 2              4.031
## 2  1 2.1  3 1              3.796
## 3  2 4.7  1 3              5.205
## 4  1 3.9  1 2              4.148
## 5  0 2.9  2 4              3.523
## 6  1 1.5  2 1              2.693
## 7  1 3.5  4 2              5.408
```

(b) What is your prediction with K = 2? Why? (1pt)

Your prediction is Y = 2.5 because the two nearest neighbors (lowest distance values: roughly 3.5 and 2.7) have Y values of 4 and 1 respectively. The average of these numbers is 2.5.

(c) If the Bayes decision boundary in this problem is highly nonlinear, then would we expect the best value for K to be large or small? Why? (1pt)

We would expect K to be small so the model could be very flexible and sensitive to slight variation and nonlinearity.

3. Consider we conduct a research study analyzing the risk factors for developing prostate cancer among men, with variables X_1 = age (years), X_2 = family history of prostate cancer (0 = no, 1 = yes), X_3 = smoking status (0 = non-smoker, 1 = smoker), and Y = probability of developing prostate cancer. A logistic regression analysis is performed, resulting in estimated coefficients $\hat{\beta}_1 = 0.06$, $\hat{\beta}_2 = 1.2$, $\hat{\beta}_3 = 0.8$, and $\hat{\beta}_0 = -3.5$.

(a) Interpret $\hat{\beta}_2$. (1 pt)

B2 = 1.2 means that for men with a family history of prostate cancer, the log-odds of developing prostate cancer are 1.2 units higher relative to those without a family history, on average, holding age and smoking status equal.

(b) Estimate the probability that a 60-year-old man with a family history of prostate cancer who is a smoker develops prostate cancer. (2 pts)

$$Y = -3.5 + 0.06Age + 1.2FamilyHistory + 0.8Smoker$$

```
age <- 60
family_history <- 1
smoking_status <- 1

# Model
y <- -3.5 + 0.06 * age + 1.2 * family_history + 0.8 * smoking_status

# Exponentiate
probability <- exp(y) / (1 + exp(y))

probability
```

```
## [1] 0.8909032
```

The probability that a 60yr old man with a history of prostate cancer who smokes develops cancer is 89.10%

4. k-fold cross-validation

(a) Briefly explain how k-fold cross-validation is implemented. (2pts)

K-fold cross-validation involves splitting the dataset into k equal-sized folds, training the model on k-1 folds, and validating on the remaining fold (computing the test error rate), repeating this process k times, each time with a different validation fold.

(b) What are the advantages of k-fold cross-validation relative to the validation set approach? (1pt)

K-fold cross-validation maximizes data use by iteratively using all data for both training and validation, leading to a more reliable test error rate. As seen in lecture 9, validation set approach is highly variable. K-fold cross-validation reduces variance by taking the average test error rate across multiple validation sets.

5. Suppose you want to minimize the false negative rate in your classification. You run two models: A and B. AUC for Model A is 0.7 and for Model B is 0.8. Can you conclude that you should choose Model B? Why or why not? (3 pts)

No, the AUC evaluates the overall performance of the classification model across all thresholds/tradeoffs of true positive and false positive rates. While B performs better overall, you would want to use something like a confusion matrix to examing the false negative rate specifically.

Applied Questions (15 pts)

Predicting Hospital Length of Stay

Download data here.

For the next portion of this assignment you will be working with the `LengthOfStay.csv` dataset. This dataset has data points on patients admitted into hospital, indicators of their health condition and how long they were admitted in the hospital.

```r
# Load data
data <- read.csv('data/LengthOfStay.csv')
```

This is an important problem in healthcare. In order for hospitals to optimize resource allocation, it is important to predict accurately how long a newly admitted patient will stay in the hospital.

1. What are the dimensions of the dataset? (1 pt)

```r
dim(data)
```

```
## [1] 3000   22
```

2. Use the `cor()` function to display the correlations of all **continuous** variables in the dataset. Which variables is most highly correlated with `lengthofstay`? (2 pts)

```r
# Filter to just the continuous variables
continuous_data <- data %>%
  select_if(~ max(.) > 1) %>%
  select(!X)

str(continuous_data)
```

```
## 'data.frame':    3000 obs. of  10 variables:
##  $ hematocrit    : num  16.9 12.4 11.4 9.8 13.5 12.1 11.9 9.2 9.2 12.7 ...
##  $ neutrophils   : num  20.1 7.8 8.7 8.2 4.7 12.9 9.4 13.6 16.7 10.2 ...
##  $ sodium        : num  135 144 138 139 137 ...
##  $ glucose       : num  136 177 139 155 143 ...
##  $ bloodureanitro: num  12 6 28 10.5 12 12 12 4 12 12 ...
##  $ creatinine    : num  0.926 1.236 1.175 1.402 1.412 ...
##  $ bmi           : num  25.5 31.6 33.1 26.2 29.8 ...
##  $ pulse         : int  72 76 83 69 65 91 58 79 74 67 ...
##  $ respiration   : num  6.5 6.5 6.5 4.25 6.5 6.5 6.5 6.5 6 6.5 ...
##  $ lengthofstay  : int  4 6 3 8 1 4 1 4 6 1 ...
```

```r
# Correlations
continuous_corr <- cor(continuous_data)

print(continuous_corr['lengthofstay', ])
```

```
##     hematocrit    neutrophils         sodium        glucose bloodureanitro
##   -0.078573851   -0.034198747   -0.004106144   -0.020379692    0.191832733
##     creatinine            bmi          pulse    respiration   lengthofstay
##   -0.004112273   -0.002354964   -0.012585940   -0.035696740    1.000000000
```

Length of stay is most highly correlated with bloodureanitro with a correlation coefficient of 0.192.

Consider the prediction problem where you want to predict the length of stay for a patient (`lengthofstay`) against all other variables available in the data set.

3. Run ridge regression with cross-validation and standardized features using the canned function `cv.glmnet` from the package `glmnet`. You can use the λ sequence generated by `cv.glment` (you do not need to provide your own λ sequence). 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)

```
set.seed(222)
ridge_cv <- cv.glmnet(x = as.matrix(continuous_data[, -10]),
                      y = continuous_data$lengthofstay,
                      alpha = 0,
                      standardize = TRUE)

ridge_cv$lambda
```

```
##   [1] 465.68931618 424.31876082 386.62345158 352.27688972 320.98158175
##   [6] 292.46646269 266.48454822 242.81079542 221.24015357 201.58578809
##  [11] 183.67746227 167.36006276 152.49225604 138.94526430 126.60174997
##  [16] 115.35479943 105.10699698  95.76958106  87.26167543  79.50958869
##  [21]  72.44617597  66.01025736  60.14608803  54.80287534  49.93433894
##  [26]  45.49830989  41.45636544  37.77349619  34.41780290  31.36022015
##  [31]  28.57426462  26.03580571  23.72285649  21.61538330  19.69513222
##  [36]  17.94547096  16.35124478  14.89864526  13.57509067  12.36911703
##  [41]  11.27027877  10.26905826   9.35678343   8.52555258   7.76816598
##  [46]   7.07806353   6.44926787   5.87633269   5.35429549   4.87863464
##  [51]   4.44523018   4.05032818   3.69050819   3.36265361   3.06392473
##  [56]   2.79173411   2.54372414   2.31774670   2.11184447   1.92423403
##  [61]   1.75329038   1.59753290   1.45561248   1.32629988   1.20847505
##  [66]   1.10111746   1.00329722   0.91416706   0.83295498   0.75895755
##  [71]   0.69153385   0.63009989   0.57412356   0.52312000   0.47664746
##  [76]   0.43430341   0.39572110   0.36056632   0.32853461   0.29934850
##  [81]   0.27275520   0.24852438   0.22644616   0.20632931   0.18799958
##  [86]   0.17129822   0.15608055   0.14221479   0.12958082   0.11806922
##  [91]   0.10758027   0.09802314   0.08931503   0.08138053   0.07415091
##  [96]   0.06756355   0.06156139   0.05609244   0.05110934   0.04656893
```

(a) Which λ had the lowest mean cross-validation error? (1 pt)

```
min_lambda <- ridge_cv$lambda.min
round(min_lambda, 3)
```

```
## [1] 0.477
```

(b) What was the cross-validation error? (1 pt)

```
cv_error <- ridge_cv$cvm[ridge_cv$lambda == ridge_cv$lambda.min]
round(cv_error, 3)
```

```
## [1] 5.7
```

(c) What was the standard error of the mean cross-validation error for this value ofλ? (1 pt)

```r
se_cv_error <- ridge_cv$cvsd[ridge_cv$lambda == ridge_cv$lambda.min]

round(se_cv_error, 3)
```

```
## [1] 0.152
```

(d) What was the largest value of λ whose mean cross validation error was within one standard deviation of the lowest cross-validation error? (1 pt)

```r
lambda_1se <- ridge_cv$lambda.1se
round(lambda_1se, 3)
```

```
## [1] 21.615
```

5. Now consider the same prediction problem. Implement your own 5-fold cross-validation routine for KNN for $K = 1, ..., 50$ (write the cross-validation routine yourself rather than using a canned package). Include the snippet of code you wrote here. It should not exceed 20 lines. (6pts)

```r
# Function to perform cross-validation for KNN
cross_validation_KNN <- function(data_x, data_y, k_seq, kfolds) {

    # Initialize matrix to store CV errors
    CV_error_mtx <- matrix(0, nrow = length(k_seq), ncol = kfolds)

    # Assign fold IDs
    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))  # Shuffle fold IDs

    # Cross-validation loop
    for (k in k_seq) {
        for (fold in 1:kfolds) {
            # Train the KNN model
            knn_fold_model <- knn(train = scale(data_x[which(fold_ids != fold),]),
                                  test = scale(data_x[which(fold_ids == fold),]),
                                  cl = data_y[which(fold_ids != fold)],
                                  k = k)
            # Measure and save error rate
            CV_error_mtx[k, fold] <- mean(knn_fold_model != data_y[which(fold_ids == fold)])
        }
    }

    # Return the matrix of CV errors
    return(CV_error_mtx)
}
```

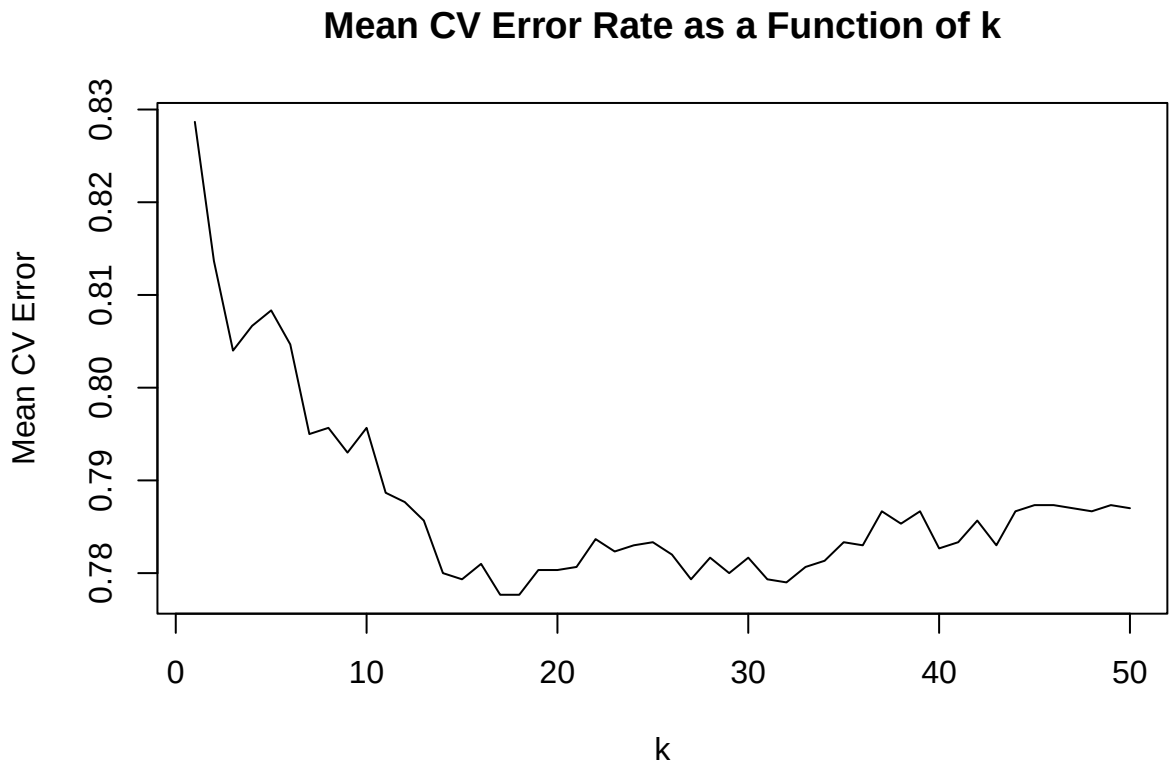
(a) Plot of mean cross-validation MSE as a function of k.

```r
# Applying the function
set.seed(222)
k_seq <- 1:50  # Values of k to test
```

```r
kfolds <- 5      # Number of folds
cv_error_mtx <- cross_validation_KNN(data_x = continuous_data[, -10],
                                     data_y = continuous_data$lengthofstay,
                                     k_seq = k_seq,
                                     kfolds = kfolds)

# Calculate mean CV error for each value of k
mean_cv_error <- rowMeans(cv_error_mtx)

# Plot mean CV error as a function of k
plot(k_seq, mean_cv_error, type = "l",
     main = "Mean CV Error Rate as a Function of k",
     ylab = "Mean CV Error", xlab = "k")
```



(b) The best k according to CV is

```r
# Find the best k according to CV
best_k <- k_seq[which.min(mean_cv_error)]
best_k
```

```
## [1] 17
```

(c) The cross-validation error for the best k is

```r
# Find the cross-validation error for the best k
cv_error_best_k <- min(mean_cv_error)
cv_error_best_k %>% round(3)
```

```
## [1] 0.778
```