A Comparison of Supervised Methods for Classifying Diabetes in the Female Pima Indian Population

Aidan Dykstal, Edward Hammond, & Lilia James
May 4th, 2020

1 Introduction

In this report, we examine the "Pima Indians Diabetes Database", courtesy of the National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK). This dataset – published publicly on May 9th, 1990 – records a variety of measurements for a simple random sample of 768 female members of the Pima Indian tribe in North America. Specifically, for each individual in the data, the following attributes are recorded:

- The number of times the individual has been pregnant.
- The individual's plasma glucose concentration over two hours in an oral glucose tolerance test.
- The individual's diastolic blood pressure in millimeters mercury (mm Hg).
- The individual's tricep skin fold thickness in millimeters (mm).
- The two-hour serum insulin for the individual in micros per milliliter (μU/ml).
- The individual's body mass index (BMI) in kilograms per meter squared (kg/m^2).
- The value of the diabetes pedigree function[1] for the individual.
- The individual's age in years.
- The individual's Type II diabetic status – in particular, whether the individual tests positive for Type II diabetes.

These data were collected as part of a clinical research trial to determine which potential risk factors – if any – are most relevant to Type II diabetes in the Pima Indian population. While we study the same data, our research objective differs.

Our primary research objective focuses on comparing the efficacy of different of statistical learning methods to classify Type II diabetes among females in the Pima Indian population. Specifically, with each method, we classify the diabetic status of individual females in the Pima Indian population using the eight other attributes recorded in the dataset. Then, we estimate the accuracy of each method to determine if we can reasonably predict someone's Type II diabetic status from the variables recorded in the data. Finally, we compare the accuracies of the various classification methods to determine whether different statistical learning models make a noticeable difference in predicting an individual's diabetic status.

In this report, we approach our research question with three distinct methods designed to address binary classification problems. These three methods are:

1. Quadratic Discriminant Analysis (QDA).
2. Logistic Regression.
3. k-Nearest Neighbors (kNN).

For each of these methods, we cover the necessary model assumptions, our model construction procedures, the model results on the Pima Indians Diabetes dataset, and our model appropriateness & limitations in the face of our research objective. Then, we draw comparisons between the performance of these three methods to offer data-driven solutions to our research objective.

[1] A function which scores a person's likelihood of diabetes from 0 to 1 according to family history of the disease.

2 The Quadratic Discriminant Analysis Approach

2.1 Assumptions

The assumptions needed for validity of QDA are

1. Each group has at least as many entries as there are features.
2. Different covariance for each of the response classes. That is, for ex – σ_{k1}, σ_{k2}, σ_{k3} for response class $k1$, $k2$, $k3$ etc.
3. Distribution of observations in each of the response class is normal with a class-specific mean (μ_k) and class-specific covariance (σ_{k2}).

2.2 Model Construction Methods

DANKNUGGIES.

2.3 Application to the Pima Indians Diabetes Study

DANKNUGGIES.

RESULTS:

```
classPredictions
      0   1
  0 307  48
  1  71 106
[1] 0.2236842
   classPredictions
      0   1
  0 299  56
  1  77 100
[1] 0.25
```

2.4 Model Limitations & Appropriateness

Below are the checks we performed for the validity assumptions.

1. There are 177 observations in the smallest group (isDiabetic == 1), and only 8 variables, so this requirement is satisfied.
2. We assume that each response class has a different covariance matrix.
3. By separating the data into the two groups, and then constructing histograms of each variable, we can see which observations appear normally distributed within a particular class. Below is a figure containing these histograms. The first column contains data from all non-diabetic entries, and the second contains all diabetic entries. Then, each row is a particular variable. In order, the rows are "pregnancyNumber", "glucoseConcentration", "bloodPressure", "tricepThiccness", "serumInsulin", "BMI", "pedigreeFunction", and "age".

   AIDAN DONT FORGET TO DO THIS YOU FUCK Insert the plot "ObsHist.png" here

From these plots, we can see that "pregnancyNumber", "serumInsulin", and "age" are all non-normally distributed. As such, we will consider omitting these covariates in order to better preserve the accuracy of the situation.

3 The Logistic Regression Approach

3.1 Assumptions

The logistic regression approach to binary classification problems requires the following assumptions for the validity of the model:

1. The response is binary, meaning observations either belong to class zero or class one. In the Pima Indians Diabetes dataset, each observation is either classified as testing positive for Type II diabetes or not, so this assumption holds in our experiement.
2. The observations in the data are independent. In the Pima Indians Diabetes dataset, individual observations are distinct and are drawn from a simple random sample of the Pima Indian population. Thus, there is reasonable evidence that observations are independent in our experiment.
3. The covariates are not collinear. To check this assumption, we must plot each combination of two covariates against each other to see if there appears to be a strong linear association between them. We will check this assumption in detail in Section 3.4.
4. Each observation is a Bernoulli trial. The success probability of these Bernoulli trials is determined by the model.

According to our logistic regression model, we define the success probability for each observation as

$$p = \frac{\exp(\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_k)}{1 + \exp(\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_k)} = \frac{\exp(\boldsymbol{\beta}^T \boldsymbol{x})}{1 + \exp(\boldsymbol{\beta}^T \boldsymbol{x})}.$$

Above, $\boldsymbol{x}$ is the vector listing the k covariates for the item to be classified and $\boldsymbol{\beta}$ is the vector of parameters. In this case, a success for each Bernoulli trial is considered testing positive for Type II diabetes. With this said, our logistic regression model classifies individual observations as Type II diabetic if the quantity

$$\hat{p} = \frac{\exp\left(\hat{\boldsymbol{\beta}}^T \boldsymbol{x}\right)}{1 + \exp\left(\hat{\boldsymbol{\beta}}^T \boldsymbol{x}\right)}$$

is greater than or equal to 0.5. Here, the vector $\hat{\boldsymbol{\beta}}$ is the maximum likelihood estimator for the parameters $\boldsymbol{\beta}$. This estimator $\hat{\boldsymbol{\beta}}$ maximizes the likelihood function for $\boldsymbol{\beta}$.

3.2 Model Construction Methods

DANKNUGGIES.

3.3 Application to the Pima Indians Diabetes Study

DANKNUGGIES.

RESULTS:

```
class
      0   1
  0 316  39
  1  76 101
[1] 0.2161654
   pcv
      0   1
  0 315  40
```

```
  1  77 100
[1] 0.2199248
```

3.4 Model Limitations & Appropriateness

DANKNUGGIES.

PUT EDWARDS PLOTS HERE.

4 The k-Nearest Neighbors Approach

4.1 Assumptions

The k-nearest neighbors approach explicitly assumes that observations are more likely to belong to the same class if their covariate values are closer in distance. To make this assumption, we must also assume that, for the vector $\boldsymbol{x}$ of covariates corresponding to an item to be classified:

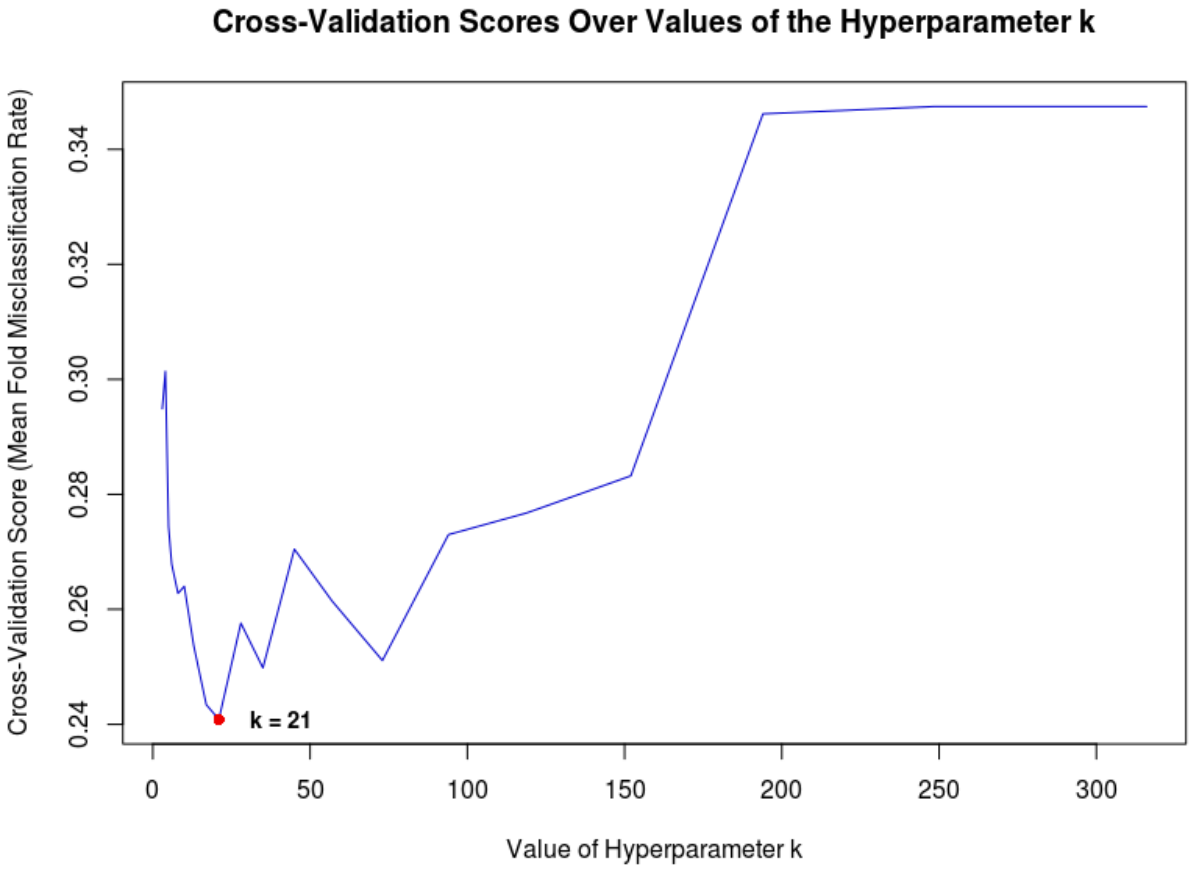
1. We have a *labeled* training set of p examples with labels (classifications) $y_1, \ldots, y_p$ and corresponding covariates $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p$;
2. Every element in each covariate vector is numeric (so we can calculate distances between $\boldsymbol{x}$ and other covariate vectors);
3. There exists some vector norm that we can employ to calculate distances between $\boldsymbol{x}$ and $\boldsymbol{x}_i$ for $i = 1, 2, \ldots, p$.

For our purposes, we meet each of these assumptions. After all, each of the 768 observations in the Pima Indians Diabetes dataset is labeled (since every observation is marked as either positive or negative for Type II diabetes) and has a vector of the same covariates. All of these covariates are numeric and there are no null values, so there is no violation of kNN model assumptions stemming from the data. Additionally, we employ a kNN model that uses the ℓ_2-norm (Euclidean distance) to measure distance between covariate vectors. So, each of the assumptions necessary for this approach are met.

DESCRIBE THE ALGORITHM.

4.2 Model Construction Methods

DANKNUGGIES.



4.3 Application to the Pima Indian Diabetes Study

DANKNUGGIES.

RESULTS:

```
  FinalFit
     0   1
  0 326  29
  1  80  97
[1] 0.2048872
   pcv
      1   2
  0 318  37
  1  85  92
[1] 0.2293233
```

4.4 Model Limitations & Appropriateness

DANKNUGGIES.

5 Conclusions

DANKNUGGIES.

References

[1] Deanna N. Schreiber-Gregory, MS

"Logistic and Linear Regression Assumptions: Violation Recognition and Control"

https://www.lexjansen.com/wuss/2018/130_Final_Paper_PDF.pdf

Published 2018

Accessed 03/29/2020

LILIA SOURCE

DATA SOURCE

Appendix

In this appendix, we feature the R code used to generate the results and visualizations featured in this report. We will feature four primary sections for code: (1) data collection and cleaning, (2) quadratic discriminant analysis model selection and analysis, (3) logistic regression model selection and analysis, & (4) k-nearest neighbors model selection and analysis.

1. Data Collection & Cleaning

The code in this section, written in R, collects and loads the Pima Indian Diabetes data into an R dataframe. It also produces some visualizations used to describe the data in Section 1 of the report.

```
# Collect the Pima Indian Diabetes Data into a Dataframe
BigChungus <- read.csv('Diabetes.csv', header = TRUE)
```

After initially processing the data, we observe some instances of invalid observations in several variables. Specifically, some entries show values of zerp for glucose concentration, blood pressure, tricep thickness, and BMI. Clearly, these values cannot be legitimate. So, we remove rows containing these covariate values.

```
# Remove Rows with Filler Values for Any Covariate
CleanChungus <- subset(BigChungus,
                       glucoseConcentration != 0)
CleanChungus <- subset(CleanChungus,
                       bloodPressure != 0)
CleanChungus <- subset(CleanChungus,
                       tricepThiccness != 0)
CleanChungus <- subset(CleanChungus,
                       BMI != 0)
BigChungus <- CleanChungus
```

Please note that these data are originally owned by the National Institute of Diabetes and Digestive and Kidney Diseases. These data are donated by Vincent Sigillitio, the RMI Group Leader at the Applied Physics Laboratory at The Johns Hopkins University. We use these data as published and made available for academic use by the University of California at Irvine Machine Learning Repository (`https://archive.ics.uci.edu/ml`).

2. Quadratic Discriminant Analysis

The code in this section, written in R, is used to compute all of the results in the report related to the Quadratic Discriminant Analysis (QDA) approach to the research question. In particular, this code produces the results in Section 2 of the report. Comments in the code reveal specific tasks.

```
# Bring in the MASS Library for QDA
library(MASS)

# Fit the QDA Model to the Diabetes Data
fitQDA <- qda(isDiabetic ~ .,
              data = BigChungus,
              prior = c(0.67, 0.33))
predictQDA <- predict(fitQDA, BigChungus)
classPredictions <- predictQDA$class

# Create the Confusion Matrix
Confusion <- table(BigChungus[,9],
                   classPredictions)
Confusion

# Estimate the Misclassification Rate
numCorrect <- sum(diag(Confusion))
numEstimated <- sum(Confusion)
accuracy <- numCorrect / numEstimated
misclassRate <- 1 - accuracy
misclassRate

# Now Use Leave-One-Out Cross Validation
n <- nrow(BigChungus)
classPredictions <- rep(0, n)
for (i in 1:n) {
  fitQDA <- qda(isDiabetic ~ .,
                data = BigChungus[-i,],
                prior = c(0.67, 0.33))
  predictQDA <- predict(fitQDA, BigChungus[i,])
  classPredictions[i] <- as.numeric(predictQDA$class) - 1
}

# Create the Confusion Matrix
Confusion <- table(BigChungus[,9],
                   classPredictions)
Confusion

# Estimate the Misclassification Rate
numCorrect <- sum(diag(Confusion))
numEstimated <- sum(Confusion)
accuracy <- numCorrect / numEstimated
misclassRate <- 1 - accuracy
misclassRate
```

3. Logistic Regression

The code in this section, written in R, is used to compute all of the results in the report related to the Logistic Regression approach to the research question. In particular, this code produces the results in Section 3 of the report. Again, comments in the code reveal specific tasks.

```
# Bring in the NNET Library for Logistic Regression
library(nnet)

# Begin Classifying by Logistic Regression
lrFit <- multinom(isDiabetic ~ .,
                  data = BigChungus,
                  trace = FALSE,
                  maxit = 10000)
coe <- summary(lrFit)$coefficients
LittleChungus <- BigChungus[-9]
nman <- t(LittleChungus)
logodds <- coe[1] + coe[-1] %*% nman
logodds <- cbind(0, t(logodds))
class <- apply(logodds, 1, which.max)
class <- class - 1

# Construct Confusion Matrix
Confusion <- table(BigChungus[,9],
                   class)
Confusion

# Estimate the Misclassification Rate
numCorrect <- sum(diag(Confusion))
numEstimated <- sum(Confusion)
accuracy <- numCorrect / numEstimated
misclassRate <- 1 - accuracy
misclassRate

# Now Try Leave-One-Out Cross-Validation
n <- length(BigChungus$isDiabetic)
pcv <- rep(0, n)

for (i in 1:n) {
  lrFit <- multinom(isDiabetic ~ .,
                    data = BigChungus[-i,],
                    trace = FALSE,
                    maxit = 10000)
  coe <- summary(lrFit)$coefficients
  tman <- t(BigChungus[i,-9])
  logodds <- coe[1] + coe[-1] %*% tman
  logodds <- cbind(0, t(logodds))
  pcv[i] <- which.max(logodds) - 1
}

# Construct Confusion Matrix
Confusion <- table(BigChungus[,9],pcv)
Confusion
```

```r
# Estimate the Misclassification Rate
numCorrect <- sum(diag(Confusion))
numEstimated <- sum(Confusion)
accuracy <- numCorrect / numEstimated
misclassRate <- 1 - accuracy
misclassRate

# Check Assumption of Absence of Multicollinearity

par(mfrow = c(7, 2))
plot(BigChungus$pregnancyNumber, BigChungus$glucoseConcentration,
     col = "blue3", pch = 16,
     xlab = "Number of Pregancies",
     ylab = "Glucose Concentration",
     main = "Number of Pregnancies vs. Glucose Concentration")
plot(BigChungus$pregnancyNumber, BigChungus$bloodPressure,
     col = "blue3", pch = 16,
     xlab = "Number of Pregancies",
     ylab = "Diastolic Blood Pressure",
     main = "Number of Pregnancies vs. Diastolic Blood Pressure")
plot(BigChungus$pregnancyNumber, BigChungus$tricepThiccness,
     col = "blue3", pch = 16,
     xlab = "Number of Pregnancies",
     ylab = "Tricep Fold Thickness",
     main = "Number of Pregnancies vs. Tricep Fold Thickness")
plot(BigChungus$pregnancyNumber, BigChungus$serumInsulin,
     col = "blue3", pch = 16,
     xlab = "Number of Pregnancies",
     ylab = "Serum Insulin",
     main = "Number of Pregnancies vs. Serum Insulin")
plot(BigChungus$pregnancyNumber, BigChungus$BMI,
     col = "blue3", pch = 16,
     xlab = "Number of Pregnancies",
     ylab = "Body Mass Index",
     main = "Number of Pregnancies vs. Body Mass Index")
plot(BigChungus$pregnancyNumber, BigChungus$pedigreeFunction,
     col = "blue3", pch = 16,
     xlab = "Number of Pregnancies",
     ylab = "Diabetic Pedigree Function",
     main = "Number of Pregnancies vs. Diabetic Pedigree Function")
plot(BigChungus$pregnancyNumber, BigChungus$age,
     col = "blue3", pch = 16,
     xlab = "Number of Pregnancies",
     ylab = "Age of Person Observed",
     main = "Number of Pregnancies vs. Age")
plot(BigChungus$glucoseConcentration, BigChungus$bloodPressure,
     col = "blue3", pch = 16,
     xlab = "Glucose Concentration",
     ylab = "Diastolic Blood Pressure",
     main = "Glucose Concentration vs. Diastolic Blood Pressure")
plot(BigChungus$glucoseConcentration, BigChungus$tricepThiccness,
     col = "blue3", pch = 16,
     xlab = "Glucose Concentration",
```

```
     ylab = "Tricep Fold Thickness",
     main = "Glucose Concentration vs. Tricep Fold Thickness")
plot(BigChungus$glucoseConcentration, BigChungus$serumInsulin,
     col = "blue3", pch = 16,
     xlab = "Glucose Concentration",
     ylab = "Serum Insulin",
     main = "Glucose Concentration vs. Serum Insulin")
plot(BigChungus$glucoseConcentration, BigChungus$BMI,
     col = "blue3", pch = 16,
     xlab = "Glucose Concentration",
     ylab = "Body Mass Index",
     main = "Glucose Concentration vs. Body Mass Index")
plot(BigChungus$glucoseConcentration, BigChungus$pedigreeFunction,
     col = "blue3", pch = 16,
     xlab = "Glucose Concentration",
     ylab = "Diabetic Pedigree Function",
     main = "Glucose Concentration vs. Diabetic Pedigree Function")
plot(BigChungus$glucoseConcentration, BigChungus$age,
     col = "blue3", pch = 16,
     xlab = "Glucose Concentration",
     ylab = "Age of Person Observed",
     main = "Glucose Concentration vs. Age")
plot(BigChungus$bloodPressure, BigChungus$tricepThiccness,
     col = "blue3", pch = 16,
     xlab = "Diastolic Blood Pressure",
     ylab = "Tricep Fold Thickness",
     main = "Diastolic Blood Pressure vs. Tricep Fold Thickness")

par(mfrow = c(7, 2))
plot(BigChungus$bloodPressure, BigChungus$serumInsulin,
     col = "blue3", pch = 16,
     xlab = "Diastolic Blood Pressure",
     ylab = "Serum Insulin",
     main = "Diastolic Blood Pressure vs. Serum Insulin")
plot(BigChungus$bloodPressure, BigChungus$BMI,
     col = "blue3", pch = 16,
     xlab = "Diastolic Blood Pressure",
     ylab = "Body Mass Index",
     main = "Diastolic Blood Pressure vs. Body Mass Index")
plot(BigChungus$bloodPressure, BigChungus$pedigreeFunction,
     col = "blue3", pch = 16,
     xlab = "Diastolic Blood Pressure",
     ylab = "Diabetic Pedigree Function",
     main = "Diastolic Blood Pressure vs. Diabetic Pedigree Function")
plot(BigChungus$bloodPressure, BigChungus$age,
     col = "blue3", pch = 16,
     xlab = "Diastolic Blood Pressure",
     ylab = "Age of Person Observed",
     main = "Diastolic Blood Pressure vs. Age")
plot(BigChungus$tricepThiccness, BigChungus$serumInsulin,
     col = "blue3", pch = 16,
     xlab = "Tricep Fold Thickness",
     ylab = "Serum Insulin",
```

```
     main = "Tricep Fold Thickness vs. Serum Insulin")
plot(BigChungus$tricepThiccness, BigChungus$BMI,
     col = "blue3", pch = 16,
     xlab = "Tricep Fold Thickness",
     ylab = "Body Mass Index",
     main = "Tricep Fold Thickness vs. Body Mass Index")
plot(BigChungus$tricepThiccness, BigChungus$pedigreeFunction,
     col = "blue3", pch = 16,
     xlab = "Tricep Fold Thickness",
     ylab = "Diabetic Pedigree Fucntion",
     main = "Tricep Fold Thickness vs. Diabetic Pedigree Function")
plot(BigChungus$tricepThiccness, BigChungus$age,
     col = "blue3", pch = 16,
     xlab = "Tricep Fold Thickness",
     ylab = "Age of Person Observed",
     main = "Tricep Fold Thickness vs. Age")
plot(BigChungus$serumInsulin, BigChungus$BMI,
     col = "blue3", pch = 16,
     xlab = "Serum Insulin",
     ylab = "Body Mass Index",
     main = "Serum Insulin vs. Body Mass Index")
plot(BigChungus$serumInsulin, BigChungus$pedigreeFunction,
     col = "blue3", pch = 16,
     xlab = "Serum Insulin",
     ylab = "Diabetic Pedigree Function",
     main = "Serum Insulin vs. Diabetic Pedigree Function")
plot(BigChungus$serumInsulin, BigChungus$age,
     col = "blue3", pch = 16,
     xlab = "Serum Insulin",
     ylab = "Age of Person Observed",
     main = "Serum Insulin vs. Age")
plot(BigChungus$BMI, BigChungus$pedigreeFunction,
     col = "blue3", pch = 16,
     xlab = "Body Mass Index",
     ylab = "Diabetic Pedigree Function",
     main = "Body Mass Index vs. Diabetic Pedigree Function")
plot(BigChungus$BMI, BigChungus$age,
     col = "blue3", pch = 16,
     xlab = "Body Mass Index",
     ylab = "Age of Person Observed",
     main = "Body Mass Index vs. Age")
plot(BigChungus$pedigreeFunction, BigChungus$age,
     col = "blue3", pch = 16,
     xlab = "Diabetic Pedigree Function",
     ylab = "Age of Person Observed",
     main = "Diabetic Pedigree Function vs. Age")
```

4. k-Nearest Neighbors

The code in this section, written in `R`, is used to compute all of the results in the report related to the k-Nearest Neighbors (kNN) approach to the research question. In particular, this code produces the results in Section 4 of the report. Again, comments in the code reveal specific tasks.

```
# Load the Classifcation Library to Use kNN
library(class)

# Select Potential Values of k (for kNN) on a Log Scale
# Go from k = sqrt(10) to 10^{2.5}
kGrid <- 10^seq(0.5, 2.5, length.out = 20)
kGrid <- floor(kGrid)
MCR <- rep(0, length(kGrid))

# Find the Indices for the 10 Folds for p-Fold CV
n <- nrow(BigChungus)
p <- 10
folds <- sample(c(1:n),replace = FALSE)
foldInds <- seq(1, n, length.out = p + 1)
foldInds <- floor(foldInds)

# Estimate the Mean Misclassification Rate for Each Value
# of k in the kGrid for Each of the p Folds
for(i in 1:length(kGrid)){
  MCRS <- rep(0, p)
  for(j in 1:p){
    testInds <- foldInds[j]:foldInds[j + 1]
    test <- BigChungus[testInds,]
    train <- BigChungus[-testInds,]
    fit <- knn(train,
               test,
               cl = train[,9],
               k = kGrid[i])

    # Construct Confusion Matrix
    Confusion <- table(test[,9], fit)

    # Estimate the Misclassification Rate
    right <- sum(diag(Confusion))
    total <- sum(Confusion)
    MCRS[j] <- 1 - (right / total)
  }

  # Get the Mean Misclassification Rate Over the Folds
  MCR[i] <- mean(MCRS)
}

# Select the Value of k (for kNN) that Minimizes the
# Mean Misclassification Rate Over the p Folds
kBestInd <- which.min(MCR)
kBest <- kGrid[kBestInd]

# Make the Cross-Validation Plot
```

```
plot(kGrid, MCR,
     type = 'l', col = 'blue3',
     xlab = 'Value of Hyperparameter k',
     ylab = 'Cross-Validation Score (Mean Fold Misclassification Rate)',
     main = 'Cross-Validation Scores Over Values of the Hyperparameter k')
points(kBest, MCR[kBestInd],
       pch = 16, col = 'red2')
kBestPlaceHolder <- kBest + 20
text(MCR[kBestInd] ~ kBestPlaceHolder,
     labels = c('k = 21'),
     cex = 0.9,
     font = 2)

# Fit to the Entire Data Set with the Optimal k
FinalFit = knn(BigChungus,
               BigChungus,
               cl = BigChungus[,9],
               k = kBest)
Confusion <- table(BigChungus[,9], FinalFit)
Confusion

# Estimate the Optimal Misclassification Rate
right <- sum(diag(Confusion))
total <- sum(Confusion)
misclassRate <- 1 - (right / total)
misclassRate

# Fit with Leave-One-Out Cross Validation and Optimal k
pcv <- rep(0, n)
for (i in 1:n) {
  fit <- knn(BigChungus[-i,],
             BigChungus[i,],
             cl = BigChungus[-i, 9],
             k = kBest)
  pcv[i] <- fit
}

# Compute the Confusion Matrix
Confusion <- table(BigChungus[,9], pcv)
Confusion

# Estimate the Optimal Misclassification Rate
right <- sum(diag(Confusion))
total <- sum(Confusion)
misclassRate <- 1 - (right / total)
misclassRate
```