Dank Title Here

Aidan Dykstal, Edward Hammond, & Lilia James

May 4th, 2020

1 Introduction

In this project, we choose to focus on a dataset which includes information regarding people of the Pima Indian Tribe and their relationship to diabetes. The dataset contains 768 observations over 9 variables. The variables are as follow:

1. Number of times pregnant -
2. Plasma glucose concentration at two hours in oral glucose tolerance test
3. Diastolic blood pressure in mm/hg
4. Tricep skinfold thickness in mm
5. Two hour serum insulin concentration μ/ml
6. Body Mass Index kg/m^2
7. Diabetes pedigree function
8. Age in years
9. Diabetic status - This categorical variable

2 The Quadratic Discriminant Analysis Approach

2.1 Assumptions

DANKNUGGIES.

2.2 Model Construction Methods

DANKNUGGIES.

2.3 Application to the STUDY-GOES-HERE

DANKNUGGIES.

RESULTS:

```
   classPredictions
      0   1
  0 347 153
  1  52 216
[1] 0.2669271
   classPredictions
      0   1
  0 341 159
  1  60 208
[1] 0.2851562
```

2.4 Model Limitations & Appropriateness

DANKNUGGIES.

3 The Logistic Regression Approach

3.1 Assumptions

DANKNUGGIES.

3.2 Model Construction Methods

DANKNUGGIES.

3.3 Application to the STUDY-GOES-HERE

DANKNUGGIES.

RESULTS:

```
 class
     0   1
  0 445  55
  1 112 156
[1] 0.2174479
   pcv
     0   1
  0 443  57
  1 114 154
[1] 0.2226562
```

3.4 Model Limitations & Appropriateness

DANKNUGGIES.

4 The k-Nearest Neighbors Approach

4.1 Assumptions

DANKNUGGIES.

4.2 Model Construction Methods

DANKNUGGIES.

4.3 Application to the STUDY-GOES-HERE

DANKNUGGIES.

RESULTS:

```
[1] 21
   FinalFit
      0   1
  0 455  45
  1 116 152
[1] 0.2096354
   pcv
      1   2
  0 440  60
  1 126 142
[1] 0.2421875
```

4.4 Model Limitations & Appropriateness

DANKNUGGIES.

5 Conclusions

DANKNUGGIES.

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) STRATEGY-ONE model selection and analysis, (3) STRATEGY-TWO model selection and analysis, & (4) STRATEGY-THREE model selection and analysis.

Data Collection & Cleaning

SUMMARY. Comments in the code highlight specific tasks.

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

Quadratic Discriminant Analysis

SUMMARY. Again, comments in the code reveal specific tasks.

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

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

Logistic Regression

SUMMARY. 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

# Estimate the Misclassification Rate
numCorrect <- sum(diag(Confusion))
```

```r
numEstimated <- sum(Confusion)
accuracy <- numCorrect / numEstimated
misclassRate <- 1 - accuracy
misclassRate
```

p-Fold k-Nearest Neighbors

SUMMARY. Again, comments in the code reveal specific tasks.

```r
# 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
theRightK <- which.min(MCR)
kBest <- kGrid[theRightK]
kBest

# 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)
MCR <- 1 - (right/total)
MCR

# 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)
MCR <- 1 - (right/total)
MCR
```