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

DANKNUGGIES.

2.2 Model Construction Methods

DANKNUGGIES.

2.3 Application to the Pima Indians Diabetes Study

DANKNUGGIES.

RESULTS:

```
classPredictions
      0   1
  0 434  66
  1 116 152
[1] 0.2369792
   classPredictions
      0   1
  0 421  79
  1 123 145
[1] 0.2630208
```

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 Pima Indians Diabetes Study

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 Pima Indian Diabetes Study

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) 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 & Visualization

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)
```

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
```

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]

# Fit to the Entire Data Set with the Optimal k
```

```r
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)
misclassRate <- 1 - (right / total)
misclassRate
```