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. [1] 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 micro units 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. [1] While we study the same data, our research objective differs.

Our primary research objective focuses on comparing the efficacy of different 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.

These data were cleaned by removing all data points which reported a value of zero for the BMI, the glucose concentration, the blood pressure, or the tricep thickness. After all, these values were entered into the dataset as placeholders. The original dataset had 768 observations in total, which was reduced to 532 after being cleaned.

2 The Quadratic Discriminant Analysis Approach

2.1 Assumptions

To ensure the validity of its modeling, we must make the following assumptions for the Quadratic Discriminant Analysis (QDA) approach:

1. Each group (class) has at least as many entries as there are total features in the model. [2]
2. The covariates (features) – stratified by each of the classes in the response – have different covariance matrices. In our case, there are only two classes: diabetic and non-diabetic. Thus, there is a covariance matrix for covariates corresponding to diabetic observations (Σ_{diabetic}) and covariates corresponding to non-diabetic observations ($\Sigma_{\text{non-diabetic}}$).
3. The observations in each of the response classes are normally distributed with a class-specific mean and class-specific variance. That is, the distributions of each individual covariate – stratified by class – is normally distributed. But, these normal distributions need not be the same.

We can summarize the final two assumptions listed above by saying that the diabetic and non-diabetic groups are distributed multivariate normal with the diabetic group having mean $\boldsymbol{\mu}_{\text{diabetic}}$ and covariance Σ_{diabetic} and the non-diabetic group having mean $\boldsymbol{\mu}_{\text{non-diabetic}}$ and covariance $\Sigma_{\text{non-diabetic}}$.

With these assumptions, the QDA classifier works by choosing the group that minimizes the quantity

$$\log|\Sigma_k| + (\boldsymbol{x} - \boldsymbol{\mu}_k)^T \Sigma_k^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_k) - 2\log P_k,$$

where k is either diabetic or non-diabetic, P_k is the prior probability of class k, and $\boldsymbol{x}$ are the covariates for the item to be classified. Ultimately, we do not know $\boldsymbol{\mu}_k$ and Σ_k for any class, so we replace them with their sample counterparts. These sample statistics are constructed from the training data, which are labeled and have the same covariates as the item to be classified.

2.2 Model Construction Methods

To begin fitting our QDA model, we must first select values for the model hyperparameters. In this case, the hyperparameters are the prior probabilties for the diabetic and non-diabetic classes. To estimate these priors, we count the number of observations sampled in each category. The table below visualizes the result.

Diabetic	Non-Diabetic
177	355

To find the prior probabilities of each class, we find the sample proportion of observations in the data set belonging to each class. Doing so, we find that roughly 33.271% of the observations belong to the diabetic class and roughly 66.729% belong to the non-diabetic class. As such, we estimate the prior probability of the diabetic class as $P_{\text{diabetic}} = 0.33$ and we estimate the prior probability of the non-diabetic class as $P_{\text{non-diabetic}} = 0.67$.

With all the model parameters selected, our model selection process involves fitting the QDA model to the Pima Indians Diabetes dataset in two distinct ways:

1. First, we fit the QDA model to the entire dataset. That is, we use each observation for training and testing. In the context of QDA, this means that – for an observation we are trying to classify – the

sample mean and covariance for this observation's labeled class will depend on the covariates associated with this observation.

2. Next, we fit the QDA model to the dataset using leave-one-out cross-validation. That is, for each observation in our data, we predict that point from the model that is trained on every other point. In the context of QDA, leave-one-out cross validation ensures that the sample mean and covariance do not depend on the covariates corresponding to the observation we are trying to classify.

In both of these cases, we keep track of the number of correctly classified and incorrectly classified observations for each group. By doing so, we construct a confusion matrix showing how frequently individuals are either correctly or incorrectly characterized as diabetic or non-diabetic. As a consequence, we can also compute the experimental accuracy and misclassification rate to estimate the goodness-of-fit for our QDA model.

2.3 Application to the Pima Indians Diabetes Study

After fitting our QDA model to the Pima Indians Diabetes dataset, we predict the classifications for each of the individuals in the data. When using the entire data set to both train and test the model, the confusion matrix (with actual values on the rows and predicted values on the columns) is:

	Diabetic	Non-Diabetic
Diabetic	307	48
Non-Diabetic	71	106

In this case, 413 of the 532 observations are classified correctly, giving a misclassification rate of roughly 22.368% and an accuracy of about 77.632%. This suggests that – if we were to classify a random sample of 100 new females from the Pima Indian population – we would expect to correctly classify the diabetic status of about 78 individuals and incorrectly classify the remaining 22.

When using leave-one-out cross validation, the confusion matrix (with actual values on the rows and predicted values on the columns) is:

	Diabetic	Non-Diabetic
Diabetic	299	56
Non-Diabetic	77	100

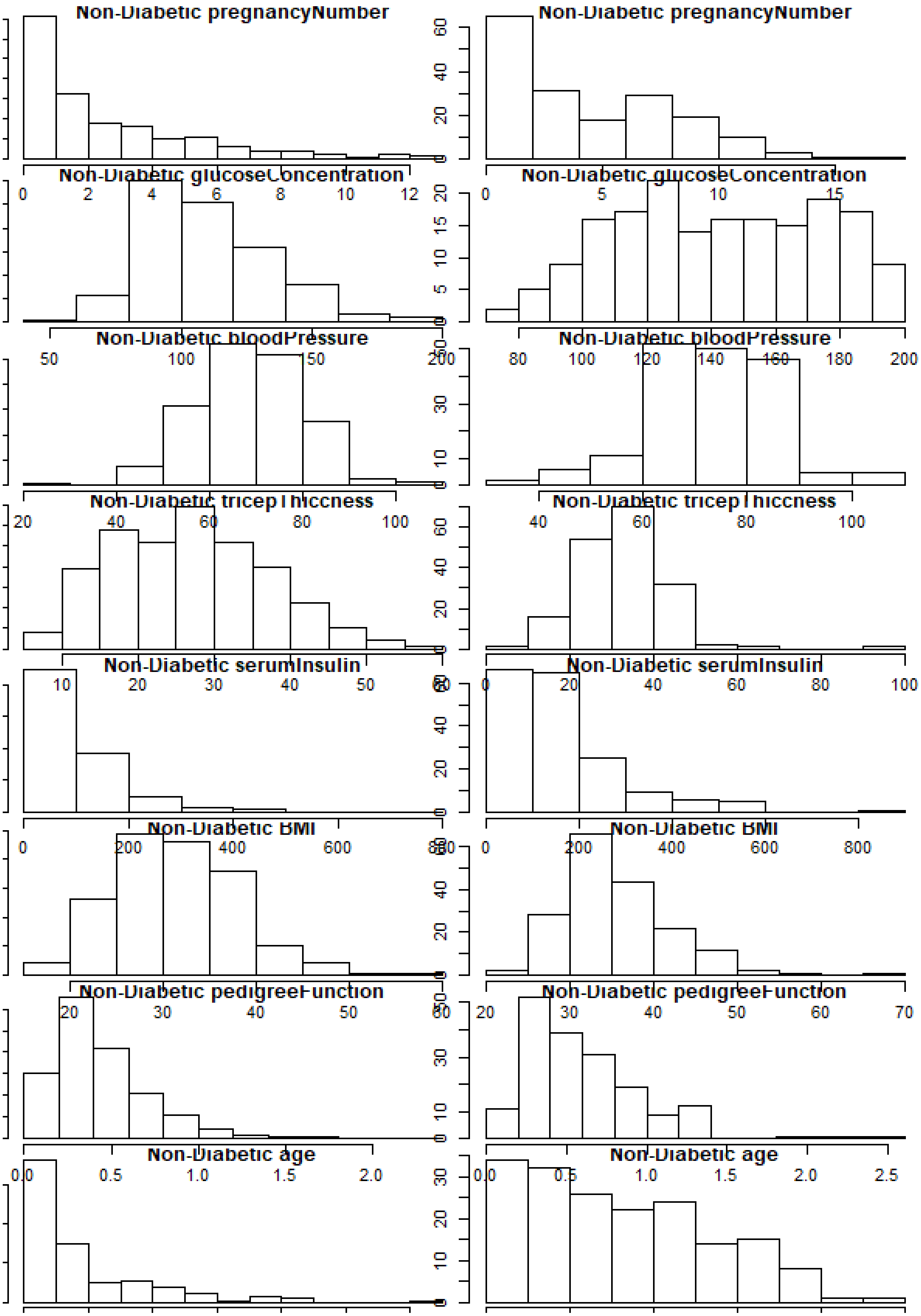
In this case, 399 of the 532 observations are classified correctly, giving a misclassification rate of 25% and an accuracy of 75%. This suggests that – if we were to classify a random sample of 100 new females from the Pima Indian population – we would expect to correctly classify the diabetic status of about 75 individuals and incorrectly classify the remaining 25.

2.4 Model Limitations & Appropriateness

Now, we will check each of the three assumptions stated in Section 2.1 of this report. First, we must assume that classes have a number of observations greater than or equal to the total number of features in the model. As seen in the table in Section 2.2, both classes have at least 177 observations. This is greater than the total number of model features, which is eight (after all, one feature is presented for each covariate).

Next, the capability for distinct covariates to have different covariance matrices is built into the QDA model; this assumption cannot be violated in this sense. So, we can proceed to checking our final assumption. This assumption – which states that the covariates are distributed multivariate normal – requires us to look at the sample distributions of each covariate in our data.

On the following page, we plot histograms for each of the features (covariates) in our model, stratified by class. The first column of plots contains data from all non-diabetic entries and the second contains all diabetic entries. Each row is a particular covariate.


Non-Diabetic pregnancyNumber
Non-Diabetic pregnancyNumber
0
20
40
60
0
2
4
6
8
10
12
0
5
10
15
Non-Diabetic glucoseConcentration
Non-Diabetic glucoseConcentration
0
5
10
15
20
50
100
150
200
80
100
120
140
160
180
200
Non-Diabetic bloodPressure
Non-Diabetic bloodPressure
0
10
30
50
20
40
60
80
100
40
60
80
100
Non-Diabetic tricepThiccness
Non-Diabetic tricepThiccness
0
20
40
60
10
20
30
40
50
60
0
20
40
60
80
100
Non-Diabetic serumInsulin
Non-Diabetic serumInsulin
0
20
40
60
0
200
400
600
800
0
200
400
600
800
Non-Diabetic BMI
Non-Diabetic BMI
0
20
40
60
20
30
40
50
60
20
30
40
50
60
70
Non-Diabetic pedigreeFunction
Non-Diabetic pedigreeFunction
0
10
30
50
0.0
0.5
1.0
1.5
2.0
0.0
0.5
1.0
1.5
2.0
2.5
Non-Diabetic age
Non-Diabetic age
0
10
20
30

From these plots, we can see that the number of pregnancies ("pregnancyNumber"), the two-hour serum insulin ("serumInsulin"), and the age ("age") are clearly non-normally distributed. The remaining features could reasonably be normally-distributed in the sample. As such, we re-fit the QDA model, this time omitting the covariates that clearly violate the assumption of normality.

Upon removing the violating covariates, our new confusion matrix for the QDA model (using the entire data set as the training and testing set) is:

	Diabetic	Non-Diabetic
Diabetic	316	39
Non-Diabetic	81	96

In this case, 412 of the 532 observations are classified correctly, giving a misclassification rate of roughly 22.556% and an accuracy of about 77.444%. This suggests that – if we were to classify a random sample of 100 new females from the Pima Indian population – we would expect to correctly classify the diabetic status of about 77 individuals and incorrectly classify the remaining 23.

When using leave-one-out cross validation, the confusion matrix (with actual values on the rows and predicted values on the columns) is:

	Diabetic	Non-Diabetic
Diabetic	314	41
Non-Diabetic	84	93

In this case, 407 of the 532 observations are classified correctly, giving a misclassification rate of roughly 23.497% and an accuracy of about 76.503%. This suggests that – if we were to classify a random sample of 100 new females from the Pima Indian population – we would expect to correctly classify the diabetic status of about 77 individuals and incorrectly classify the remaining 23.

We now discuss the limitations of the QDA approach. First, we note that we check the normality of the covariates by looking at histograms of their sample distributions, stratified by class. While this visual method for checking normality is often sufficiently robust, it is plausible that the covariates are not actually normally distributed, even when they appear to be in-sample. This means that the assumption of normal covariates could be violated, despite our checks.

Similarly, the prior probabilities we select are based on sample proportions of the data belonging to each class. While this offers an estimate of the prior probabilities, our method of estimation is still entirely subjective and could add misleading bias to the model if it is incorrect.

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. [3]
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]
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. [3]
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

Our model selection process involves fitting the Logistic Regression model to the Pima Indians Diabetes dataset in two distinct ways:

1. First, we fit the Logistic Regression model to the entire dataset. That is, we use each observation for training and testing. In the context of Logistic Regression, this means that – for an individual we want to classify – the likelihood that the individual belongs in a given class is calculated using a set of observations that includes the individual.
2. We continue by choosing the hyperparameter `maxit`. The `maxit` is the maximum number of iterations in the algorithm's search for the minimum of the likelihood. We choose `maxit` as 10,000 because it finds a balance in computational feasibility and accuracy.
3. Next, we fit the Logistic Regression model to the dataset using leave-one-out cross-validation. That is, for each observation in our data, we predict that point from the model that is trained on every other point. In the context of Logistic Regression, leave-one-out cross validation ensures that the likelihood for an individual's class membership is independent of that individual's associated label and covariates.

As with our QDA approach, we keep track of the number of correctly classified and incorrectly classified observations for each group in either of the cases above. That way, we can still construct a confusion matrix and compute the experimental accuracy and misclassification rate to estimate the goodness-of-fit for our Logistic Regression model.

3.3 Application to the Pima Indians Diabetes Study

After fitting the Logistic Regression model to the Pima Indians Diabetes dataset, we predict the classifications for each of the individuals in the data. When using the entire data set to both train and test the model, the confusion matrix (with actual values on the rows and predicted values on the columns) is:

	Diabetic	Non-Diabetic
Diabetic	316	39
Non-Diabetic	76	101

In this case, 417 of the 532 observations are classified correctly, giving a misclassification rate of roughly 21.617% and an accuracy of about 78.383%. This suggests that – if we were to classify a random sample of 100 new females from the Pima Indian population – we would expect to correctly classify the diabetic status of about 78 individuals and incorrectly classify the remaining 22.

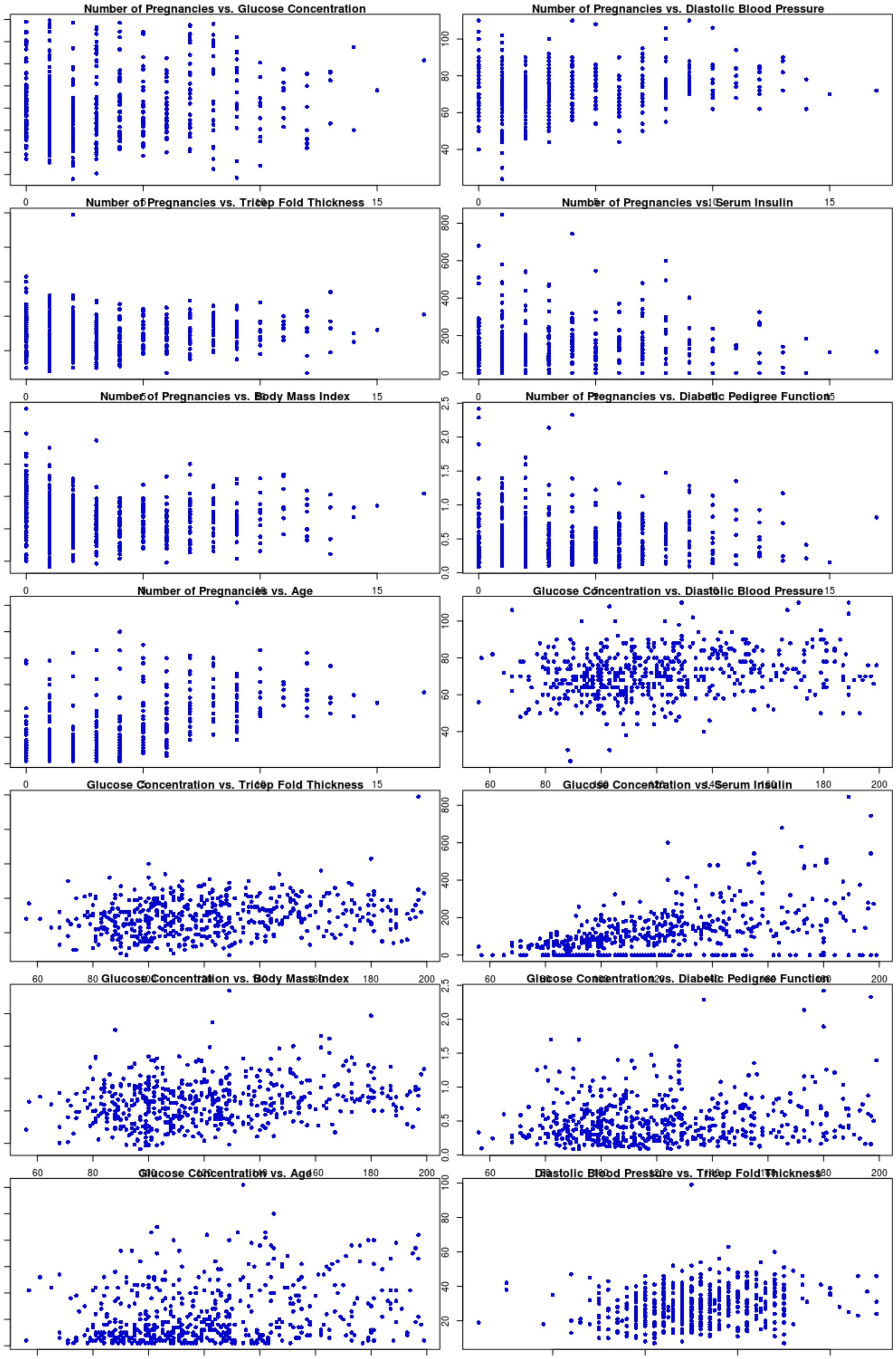
When using leave-one-out cross validation, the confusion matrix (with actual values on the rows and predicted values on the columns) is:

	Diabetic	Non-Diabetic
Diabetic	315	40
Non-Diabetic	77	100

In this case, 415 of the 532 observations are classified correctly, giving a misclassification rate of roughly 21.992% and an accuracy of about 78.007%. This suggests that – if we were to classify a random sample of 100 new females from the Pima Indian population – we would expect to correctly classify the diabetic status of about 78 individuals and incorrectly classify the remaining 22.

3.4 Model Limitations & Appropriateness

Finally, we check the assumption of the absence of multicollinearity by plotting the covariates against one another. Upon looking at the below plots, we see one potential exception, that being, the plot of tricep fold thickness versus BMI. In this plot, we concede that these two factors may be related in a linear fashion.



Number of Pregnancies vs. Glucose Concentration

Number of Pregnancies vs. Diastolic Blood Pressure

Number of Pregnancies vs. Tricep Fold Thickness

Number of Pregnancies vs. Serum Insulin

Number of Pregnancies vs. Body Mass Index

Number of Pregnancies vs. Diabetic Pedigree Function

Number of Pregnancies vs. Age

Glucose Concentration vs. Diastolic Blood Pressure

Glucose Concentration vs. Tricep Fold Thickness

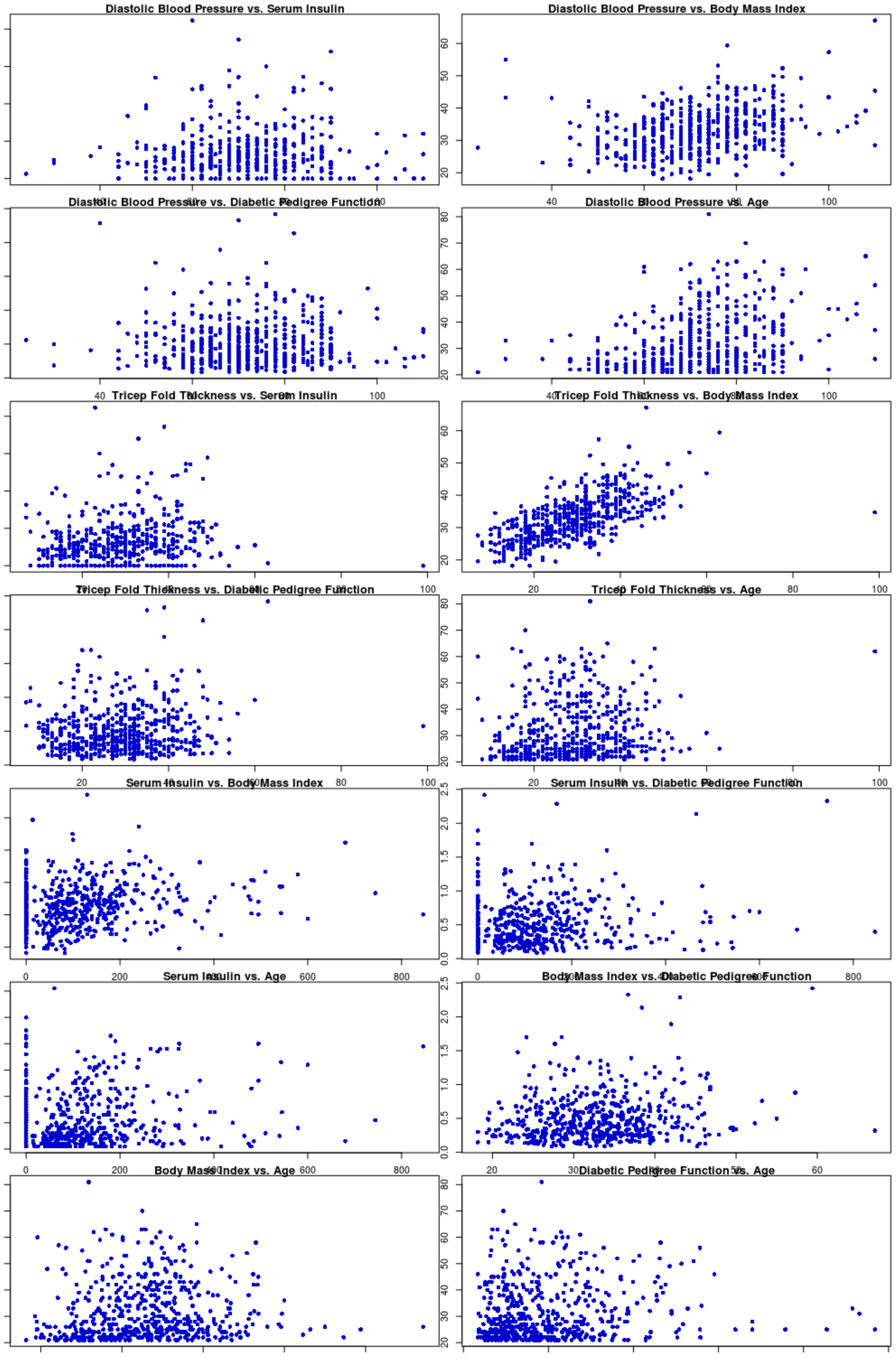
Glucose Concentration vs. Serum Insulin

Glucose Concentration vs. Body Mass Index

Glucose Concentration vs. Diabetic Pedigree Function

Glucose Concentration vs. Age

Diastolic Blood Pressure vs. Tricep Fold Thickness


Diastolic Blood Pressure vs. Serum Insulin
Diastolic Blood Pressure vs. Body Mass Index
20
30
40
50
60
40
60
80
100
Diastolic Blood Pressure vs. Diabetic Pedigree Function
Diastolic Blood Pressure vs. Age
20
30
40
50
60
70
80
Tricep Fold Thickness vs. Serum Insulin
Tricep Fold Thickness vs. Body Mass Index
20
40
60
80
100
Tricep Fold Thickness vs. Diabetic Pedigree Function
Tricep Fold Thickness vs. Age
Serum Insulin vs. Body Mass Index
Serum Insulin vs. Diabetic Pedigree Function
0.0
0.5
1.0
1.5
2.0
2.5
0
200
400
600
800
Serum Insulin vs. Age
Body Mass Index vs. Diabetic Pedigree Function
20
30
40
50
60
Body Mass Index vs. Age
Diabetic Pedigree Function vs. Age

We remove BMI as an explanatory covariate, since tricep fold thickness is a better gauge of body composition and the two covariates could reasonably be collinear. Then, we re-fit our Logistic Regression model to the data without the BMI. Upon doing so, we recalculate the confusion matrix, misclassification rate, and accuracy as before.

After fitting the Logistic Regression model to the Pima Indians Diabetes dataset, we predict the classifications for each of the individuals in the data. When using the entire data set to both train and test the model, the confusion matrix (with actual values on the rows and predicted values on the columns) is:

	Diabetic	Non-Diabetic
Diabetic	317	38
Non-Diabetic	74	103

In this case, 420 of the 532 observations are classified correctly, giving a misclassification rate of roughly 21.053% and an accuracy of about 78.947%. This suggests that – if we were to classify a random sample of 100 new females from the Pima Indian population – we would expect to correctly classify the diabetic status of about 79 individuals and incorrectly classify the remaining 21.

When using leave-one-out cross validation, the confusion matrix (with actual values on the rows and predicted values on the columns) is:

	Diabetic	Non-Diabetic
Diabetic	314	41
Non-Diabetic	77	100

In this case, 414 of the 532 observations are classified correctly, giving a misclassification rate of roughly 22.180% and an accuracy of about 77.820%. This suggests that – if we were to classify a random sample of 100 new females from the Pima Indian population – we would expect to correctly classify the diabetic status of about 78 individuals and incorrectly classify the remaining 22.

Now, we discuss the limitations of our Logistic Regression approach. First, it is possible that – for every observation that we classify – the likelihood estimates for class membership are very similar. For instance, the likelihood that the observation belongs to the diabetic class could be roughly 51% while the likelihood that the observation belongs to the non-diabetic class could be roughly 49%. In this case, our classification has a large amount of uncertainty. So, if this is the case, our model could be overfit to the training data and perform poorly on the testing data. We likely surmount this limitation by estimating the misclassification rate using leave-one-out cross-validation, which tends to mitigate some overfitting. Still, it is a possible limitation to our model that is worth considering.

Given our careful attention to checking the model assumptions, it seems highly unlikely that there are other significant limitations to our Logistic Regression approach.

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.

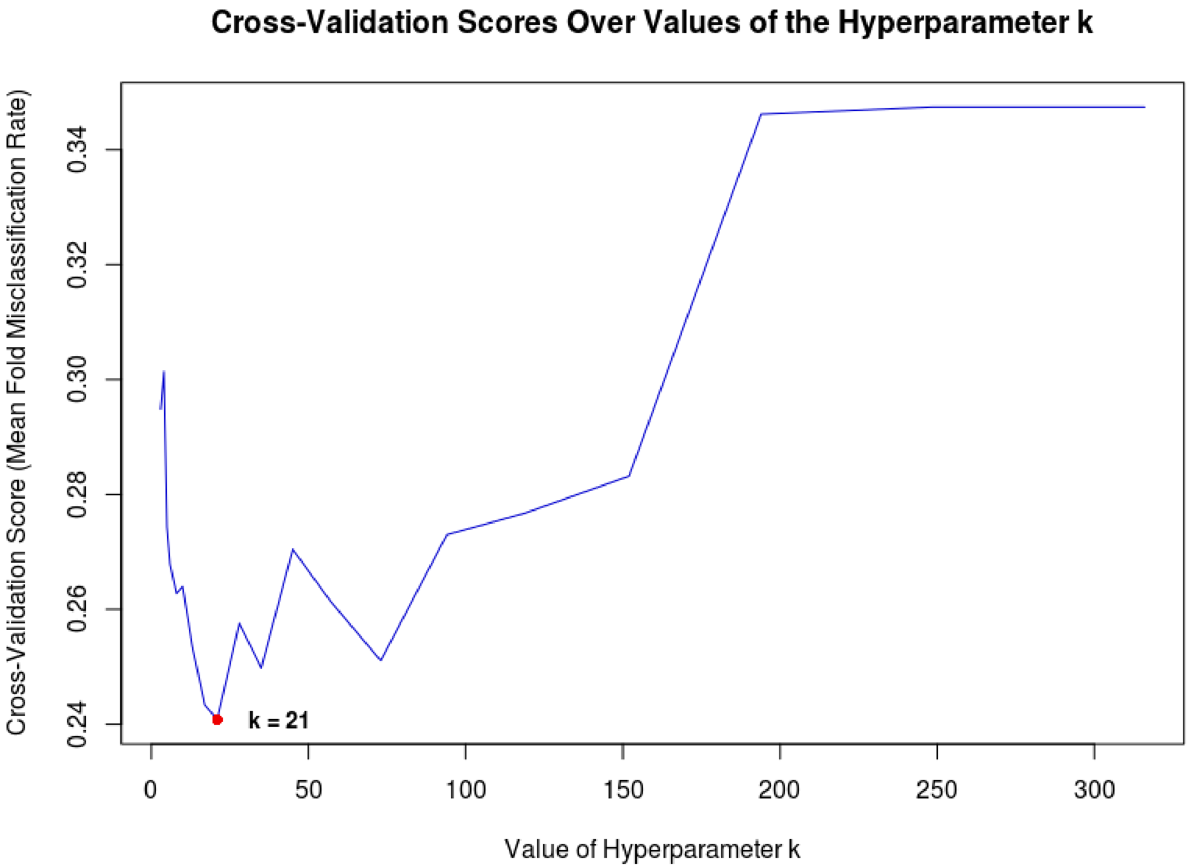
The kNN algorithm assigns a predicted class to an observation whose vector of covariates is $\boldsymbol{x}$. With the hyperparameters we specified, this classification is performed by locating the k closest covariate vectors (using Euclidean distance as the norm) to $\boldsymbol{x}$ in the training data. These k elements of the training data are called the "nearest neighbors" of the item to be classified. Since these training data are labeled, we collect the k classification labels (in the training data) for the k nearest neighbors of the item to be classified. The item is classified to the classification label that is most represented in the sample of k.

4.2 Model Construction Methods

To construct the kNN model, we first select the hyperparameters. As mentioned in the previous section, the hyperparameters are the vector norm to measure distance between covariate vectors and the natural number k that stores the number of "nearest neighbors" to look at during classification. By convention for this course, we choose the ℓ_2 norm to measure the (Euclidean) distance between covariate vectors. But, to select a proper value of k, we employ 10-fold cross validation. Specifically, we split the Pima Indians Diabetes dataset into 10 random folds with roughly the same number of observations. Then, we repeat the following process for each of the 10 distinct folds.

- Select the current fold and specify it as the testing data. Specify the remaining 9 folds as the training data.
- Create a grid of the possible values of k between 1 and $\frac{n}{2}$, where n is the number of observations in the whole data set.
- For each value of k in the grid, fit a kNN model to the training data (which we recall are the 9 folds *not* selected as the testing data). Use this kNN model to predict the classes of the testing data. Compute the estimated misclassification rate.

When this is finished, we compute the mean misclassification rate for each distinct value of k tested in the steps listed above. Then, we plot this mean misclassification rate versus the tested values of k. This way, we can visualize which value of k minimizes the mean misclassification rates over the 10 folds. This plot is given below:



We see that the value $k = 21$ minimizes the mean misclassification rate over the 10 folds. So, we select $k = 21$ as the nearest neighbors hyperparameter for our model.

With all the model parameters selected, our model selection process involves fitting the kNN model to the Pima Indians Diabetes dataset in two distinct ways:

1. First, we fit the kNN model to the entire dataset. That is, we use each observation for training and testing. In the context of kNN, this means that the k nearest neighbors to each point in the test set include the point itself.
2. Next, we fit the kNN model to the dataset using leave-one-out cross-validation. That is, for each observation in our data, we predict that point from the model that is trained on every other point. In the context of kNN, leave-one-out cross validation ensures that the k nearest neighbors for a given point in the test set do not include that point.

As with our QDA and Logistic Regression approaches, we keep track of the number of correctly classified and incorrectly classified observations for each group in either case listed above. That way, we still construct a confusion matrix and calculate the experimental accuracy and misclassification rate to estimate the goodness-of-fit for our kNN model.

4.3 Application to the Pima Indian Diabetes Study

After fitting the kNN model to the Pima Indians Diabetes dataset, we predict the classifications for each of the individuals in the data. When using the entire data set to both train and test the model, the confusion matrix (with actual values on the rows and predicted values on the columns) is:

	Diabetic	Non-Diabetic
Diabetic	326	29
Non-Diabetic	80	97

In this case, 423 of the 532 observations are classified correctly, giving a misclassification rate of roughly 20.489% and an accuracy of about 79.511%. This suggests that – if we were to classify a random sample of 100 new females from the Pima Indian population – we would expect to correctly classify the diabetic status of about 80 individuals and incorrectly classify the remaining 20.

When using leave-one-out cross validation, the confusion matrix (with actual values on the rows and predicted values on the columns) is:

	Diabetic	Non-Diabetic
Diabetic	318	37
Non-Diabetic	85	92

In this case, 410 of the 532 observations are classified correctly, giving a misclassification rate of roughly 22.932% and an accuracy of about 77.067%. This suggests that – if we were to classify a random sample of 100 new females from the Pima Indian population – we would expect to correctly classify the diabetic status of about 77 individuals and incorrectly classify the remaining 23.

4.4 Model Limitations & Appropriateness

Since all of the covariates are numeric, we can calculate distances between $\boldsymbol{x}$ and other covariate vectors. Thus, the conditions are satisfied to perform kNN regression. Additionally, we made sure to select the "best" value for the hyperparameter k using 10-fold cross-validation, as shown in Section 4.2. These measures to promote the accuracy of our model assure that it is appropriate. Since the model is otherwise non-parametric and without assumptions, there is nothing left to check.

We now discuss the potential limitations of our kNN approach. In practice, kNN tends to provide accurate results provided that items closer in covariate distance are actually more likely to be in the same class. In the Pima Indians Diabetes data set, this may not be the case. For instance, consider the age and blood pressure covariates. While Type II diabetes *may* be linked to certain age groups or blood pressure groups, it is also possible that there is no connection between diabetic status and these predictors. In the case that closer covariate values do not actually affect the membership of individuals in a particular class, the primary assumption of kNN is violated and our model is not appropriate.

Additionally, the kNN algorithm can be computationally expensive when datasets become large. We circumvent this limitation by using a reasonably small training data set with 532 observations. But, if we were to collect new data and re-fit the model, we could experience much slower performance times.

5 Conclusions

Now that we have presented three different approaches to classify Type II diabetes among the Pima Indian female population, we can compare the results of these approaches. Summarized, the results among the approaches are as follows:

Method	Accuracy	Cross-Validated Accuracy
QDA	77.444%	76.503%
Logistic Regression	78.947%	**77.820%**
*k*NN	79.511%	77.067%

In this table, accuracy estimates are presented for each approach, both using the entire Pima Indians Diabetes dataset for training ("accuracy") and using leave-one-out cross-validation ("cross-validated accuracy").

To decide on the superior model, we choose the model that maximizes the cross-validated accuracy estimate. After all, using leave-one-out cross-validation to estimate the accuracy ensures that the observation we are classifying is not used to perform the classification. As such, the cross-validation method removes a small amount of bias from the accuracy estimator. We see that the Logistic Regression method yields the best accuracy after cross-validation with a cross-validated accuracy estimate of roughly 77.820%. As such, we conclude that the Logistic Regression approach offers the most accurate means of classifying the Type II diabetic status of females in the Pima Indian population.

In the context of our research question, we conclude that all three tested approaches offer reasonably accurate models to classify the diabetic status of females in the Pima Indian population. Each model possesses its own advantages and limitations, but the Logistic Regression approach seems to have the fewest violations of assumptions and the best accuracy. As such, if we were to classify the diabetic status of new females in the Pima Indian tribe, we would employ the Logistic Regression model presented in Section 3.

References

[1] Vincent Sigillitio, Ph.D
“Pima Indians Diabetes Database”
`https://archive.ics.uci.edu/ml`
Published 05/09/1990
Accessed 03/29/2020

[2] Prashant Shekhar, MS
“How to Perform Logistic Regression, LDA, & QDA in R”
`https://datascienceplus.com/how-to-perform-logistic-regression-lda-qda-in-r/`
Published 01/05/2018
Accessed 03/29/2020

[3] 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

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.

```r
# 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.

```r
# 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.

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

# Check the Assumption that the Group-Split
# Covariate Data is Roughly Multivariate Normal
group1 <- subset(BigChungus,
                 isDiabetic == 0)
group2 <- subset(BigChungus,
                 isDiabetic == 1)
par(mar = c(1, 1, 1, 1))
par(mfrow = c(8, 2))
for (i in 1:8) {
  hist(group1[,i],
       main = NULL,
       xlab = NULL,
       ylab = NULL)
  hist(group2[,i],
       main = NULL,
       xlab = NULL,
       ylab = NULL)
}

# 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
```

We noted in Section 2.4 that there are some variables that violate the assumption of normality among the covariates. To provide a model that more closely meets the assumptions of QDA, we remove these covariates from the dataset and refit our QDA model.

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

# Remove Violating Columns Before Analysis
QDAChungus <- BigChungus[,-c(1,5,8)]

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

# Create the Confusion Matrix
Confusion <- table(QDAChungus[,6],
                   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(QDAChungus)
classPredictions <- rep(0, n)
for (i in 1:n) {
  fitQDA <- qda(isDiabetic ~ .,
                data = QDAChungus[-i,],
                prior = c(0.67, 0.33))
  predictQDA <- predict(fitQDA, QDAChungus[i,])
  classPredictions[i] <- as.numeric(predictQDA$class) - 1
```

```r
}

# Create the Confusion Matrix
Confusion <- table(QDAChungus[,6],
                   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(mar = c(1, 1, 1, 1))
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")
```

Because of the potenital collinearity between the BMI and the tricep fold thickness, we remove the BMI from consideration and fit the model again.

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

# Begin Classifying by Logistic Regression
```

```
lrFit <- multinom(isDiabetic ~ .,
                  data = BigChungus[, -6],
                  trace = FALSE,
                  maxit = 10000)
coe <- summary(lrFit)$coefficients
LittleChungus <- BigChungus[,-c(6,9)]
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,-6],
                    trace = FALSE,
                    maxit = 10000)
  coe <- summary(lrFit)$coefficients
  tman <- t(BigChungus[i,-c(6,9)])
  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))
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.

```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
kBestInd <- which.min(MCR)
kBest <- kGrid[kBestInd]

# Make the Cross-Validation Plot
```

```r
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
```