

Project

Chris Meade

12/7/2016

Abstract

Diabetes occurs more often in the Pima Indian tribe of southern Arizona than any other population of people in the world. By selecting the optimal parameters for three machine learning algorithms – Random Forest decision trees, a Support Vector Machine (SVM) with a radial basis function kernel, and a Multilayer Perceptron (MLP) neural network – this paper proves that diabetes can be accurately diagnosed in Pima women. The results also indicate that classification accuracy is maximized with the neural network algorithm.

1. Introduction

This paper attempts to explore the viability of the application machine learning to the medical field and human health. Can statistical learning algorithms be utilized by doctors to help make accurate diagnoses and aid in decision making as it pertains to patient health? To attempt to answer this question, we train three distinct machine learning algorithms – Random Forest decision trees, a Support Vector Machine (SVM) with a radial basis function kernel, and a Multilayer Perceptron (MLP) neural network – using data about diabetes occurrence in Pima Indian Women. All analysis was performed with the R statistical computing software package and associated machine learning packages in the RStudio integrated development environment. Our results clearly indicate that all three algorithms had a high success rate in correctly diagnosing diabetes in Pima women. This lends support to the proposed hypothesis that machine learning does have a place in doctors' arsenal.

1.1 About the Data

The data about diabetes incidence in Pima women used in our analysis are freely available from the University of California, Irvine machine learning repository at <http://archive.ics.uci.edu/ml/>. The dataset contains 768 observations of nine features. The features in the order they appear in the dataset are as follows:

Feature Description	Feature name in dataset
Number of times pregnant	timePregnant
Plasma glucose concentration at 2 hours in an oral glucose tolerance test	plasmaGlucose
Diastolic blood pressure (mm Hg)	diastolicPressure
Triceps skin fold thickness (mm)	tricepThickness
2-Hour serum insulin (mu U/ml)	serumInsulin
Body mass index (weight in kg/(height in m) ²)	bmi
Diabetes pedigree function	pedigreeFunction
Age (years)	age
Class variable (diabetic or not diabetic)	diabetes

1.2 Software Used in Analysis

All analysis was performed with the R statistical computing environment. The functionality of R was extended by importing the following software libraries: `caret`, `mice`, `randomForest`, `kernlab` VIM, and `RSNNS`. These libraries add support for data imputation and for the three machine learning algorithms used in our analysis.

The analysis was written in the RStudio integrated development environment. This package was generated using the 'rmarkdown' R package.

2. Data Processing

2.1 Data Acquisition

We must first import our data into R and load the required R packages used in our analysis.

```
library(caret)
library(mice)
library(randomForest)
library(kernlab)
library(RSNNS)
library(VIM)

#Create a vector of the feature names
headers <- c("timesPregnant", "plasmaGlucose", "diastolicPressure", "tricepThickness",
             "serumInsulin", "bmi", "pedigreeFunction", "age", "diabetes")

#Import data
www <- paste0("http://archive.ics.uci.edu/ml/machine-learning-databases/",
              "pima-indians-diabetes/pima-indians-diabetes.data")

data <- read.csv(url(www),
                 header = FALSE,
                 col.names = headers)
```

We examine the structure of our data table to ensure that it was imported correctly.

```
str(data)

## 'data.frame': 768 obs. of 9 variables:
## $ timesPregnant : int 6 1 8 1 0 5 3 10 2 8 ...
## $ plasmaGlucose : int 148 85 183 89 137 116 78 115 197 125 ...
## $ diastolicPressure: int 72 66 64 66 40 74 50 0 70 96 ...
## $ tricepThickness : int 35 29 0 23 35 0 32 0 45 0 ...
## $ serumInsulin : int 0 0 0 94 168 0 88 0 543 0 ...
## $ bmi : num 33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
## $ pedigreeFunction : num 0.627 0.351 0.672 0.167 2.288 ...
## $ age : int 50 31 32 21 33 30 26 29 53 54 ...
## $ diabetes : int 1 0 1 0 1 0 1 0 1 1 ...
```

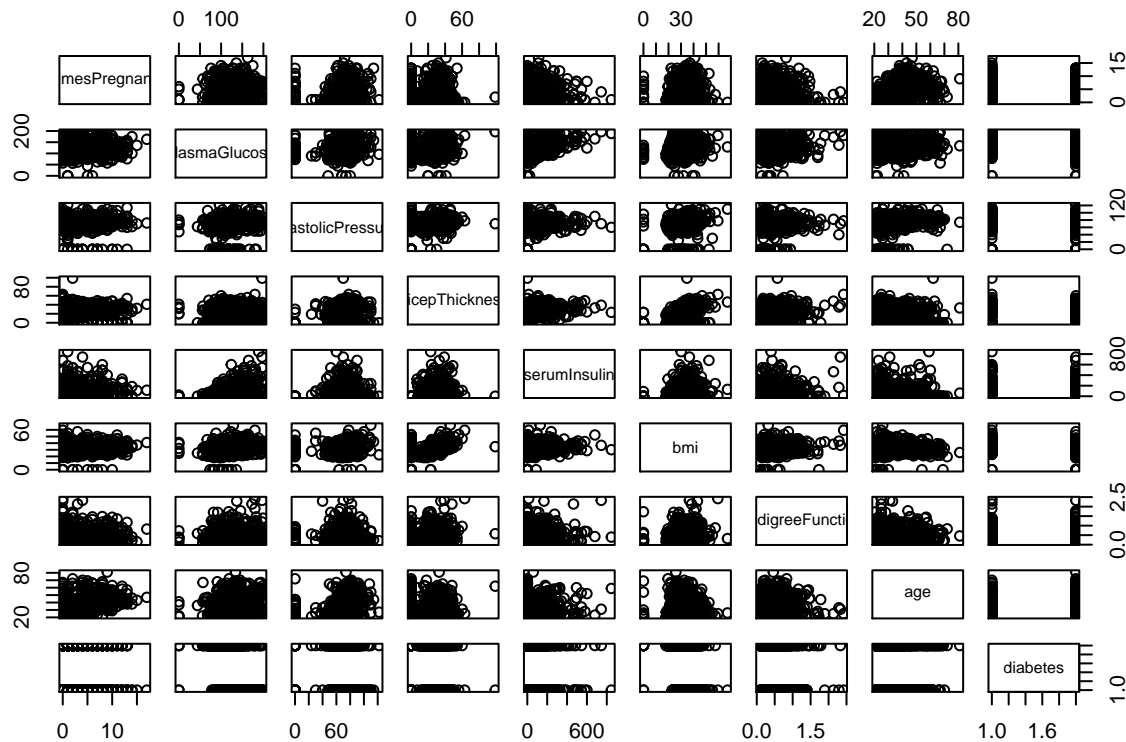
Next, we encode the class label of the `diabetes` feature to be more readable by changing 0 to `notDiabetic` and 1 to `Diabetic`. We then convert this feature to a factor.

```
data$diabetes <- as.factor(ifelse(data$diabetes == 0, "notDiabetic", "Diabetic"))
```

2.2 Exploratory Analysis

Now that we have our data and have imported all required software libraries, we perform some exploratory analysis. We begin by constructing a scatterplot matrix.

```
pairs(data)
```



From this scatterplot matrix we make the following observations:

1. Several features have 0 valued observation.
2. There does not appear to be strong linearity between any two features. We confirm this later in the analysis.

2.3 Missing Data

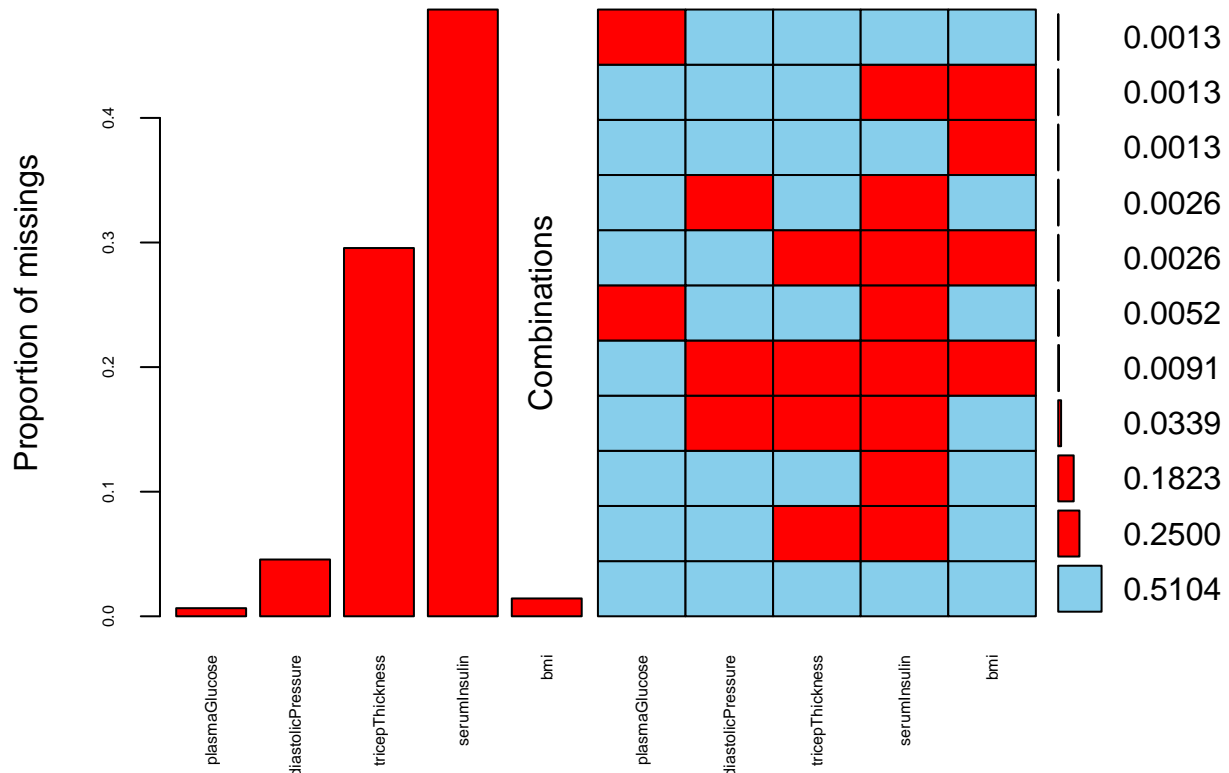
From our exploratory analysis, we found the several features contain values of 0. For several of the features, a 0 value is biologically impossible, namely in `plasamaGlucose`, `diastolicPressure`, `tricepThickness`, `serumInsulin`, `bmi`. We conclude that although the dataset does not explicitly contain missing values, it does contain implicitly missing values encoded as 0s.

We decide to explicitly encode the missing values in the five listed features as NAs.

```
for (i in 2:6){
  for (n in 1:nrow(data)){
    if (data[n, i] == 0){
      data[n, i] <- NA
    }
  }
}
```

Now that missing data is correctly encoded as NA's, we construct an aggregation plot to count the precise number of missing values.

```
aggr(data[,2:6], cex.lab=1, cex.axis = .5, numbers = T, gap = 0)
```



The left plot shows the proportion of missing values to total observations for each feature. We note that over half of all observations for **serumInsulin** and nearly a third **tricepThickness** observations are missing.

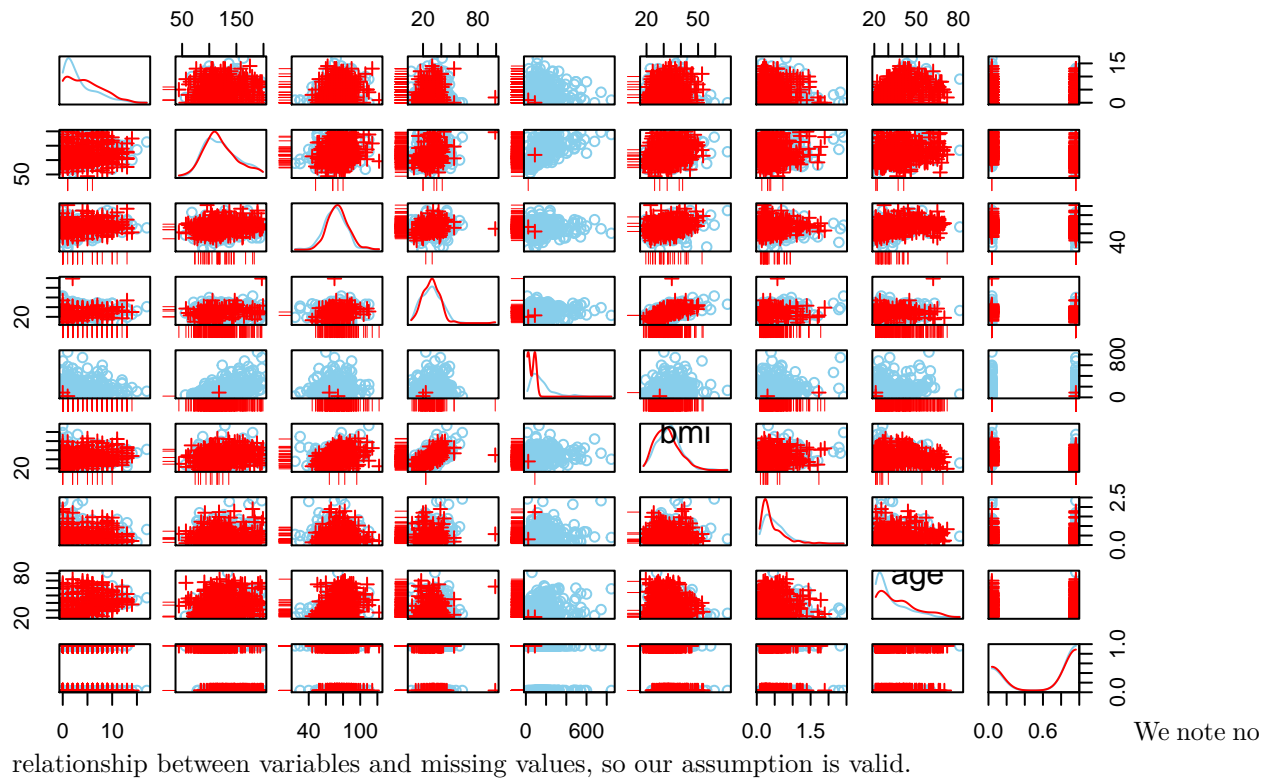
The plot on the right indicates that only little over half of the observations are complete. Due to the number of observations with missing values, we choose not to drop these observations from our analysis. Instead, we will impute missing values using a technique called Imputation by Predicted Mean matching, which “imputes missing values by means of the nearest-neighbor donor with distance based on the expected values of the missing variables conditional on the observed covariates.”(1)

Imputation by Predicted Mean Matching is advantageous over other methods of data imputation because it “can provide valid inference when data are missing at random.” (2)

This introduces an assumption: that data are at least missing at random. That is, we must be sure that our data are not Missing Not at Random (MNAR). In the case of MNAR,

To test if this is a reasonable assumption, we construct a scatterplot matrix highlighting missing values.

```
scattmatrixMiss(data)
```



2.4 Feature Selection

2.5 Data Standardization

3. Training the algorithms

3.1 Predicting the most common label

Base accuracy etc

We begin our analysis by loading data in

References

- (1) <http://www.stefvanbuuren.nl/publications/2014%20Semicontinuous%20-%20Stat%20Neerl.pdf>
- (2) <http://bmcmedresmethodol.biomedcentral.com/articles/10.1186/1471-2288-14-75>