Pima Indians Diabetes CYO Project

Gideon Vos

20 January 2019

1. Executive Summary

Background and Motivation

Diabetes mellitus is one of the major noncommunicable diseases which have great impact on human life today. Many nations are now facing a swiftly rising growth of diabetes among their residents.

According to a study by the World Health Organization (WHO), this number will have raised to 552 million by 2030, denote that one in 10 grownups will have diabetes by 2030 if no serious act is taken. In 2014, the worldwide frequency of diabetes was projected to be 9 % among adults aged 18+ years.

In developing nations, most publics with diabetes are aged between 35 and 64. WHO already made an alarm that Diabetes is the 7th leading cause of death in the world in 2030. In 2012, an estimated 1.5 million deaths were straightly triggered by diabetes. In this, more than 80 % of diabetes deaths occur in low and middleincome countries. Total deaths from diabetes are projected to rise by more than 50 % in the next 10 years.

It is apparent that diabetes is a foremost cause of blindness, amputation and kidney failure. Lack of alertness about diabetes, combined with inadequate access to health services and vital medicines, can lead to many hitches. It is a universal problem with overwhelming human, social, and economic impact, affecting around 300 million people worldwide.

By applying computational analytics on clinical big data, the massive amount of data generated in the healthcare systems, will be used to create medical intelligence which will drive medical prediction and forecasting. Medical analysis is a new trend in medical science. Developing medical intelligence out of the clinical data available will create healthcare system to be patient-centered and will reduce medical cost and hospital readmission too.

DataSet

The data set used for the purpose of this study is Pima Indians Diabetes Database of National Institute of Diabetes and Digestive and Kidney Diseases. This diabetes database, donated by Vincent Sigillito, is a collection of medical diagnostic reports of 768 examples from a population living near Phoenix, Arizona, USA.

The samples consist of examples with 8 attribute values and one of the two possible outcomes, namely whether the patient is tested positive for diabetes (indicated by output one) or not (indicated by zero).

Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

- Pregnancies: Number of times pregnant
- Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
- BloodPressure: Diastolic blood pressure (mm Hg)
- SkinThickness: Triceps skin fold thickness (mm)
- Insulin: 2-Hour serum insulin (mu U/ml)
- BMI: Body mass index (weight in kg/(height in m)^2)
- DiabetesPedigreeFunction: Diabetes pedigree function
- Age: Age (years)
- Outcome: Class variable (0 or 1)

The dataset is relatively small (23KB) and a copy can be downloaded from my GitHub repository here:

(https://github.com/gideonvos/pima/blob/master/pima-indians-diabetes.csv)

Goal

The Pima Indians dataset has been widely studied. According to this post here on Kaggle, an accuracy rate of 75% is considered average, while above 79% being excellent:

(https://www.kaggle.com/general/19387)

A good comparison done in R details a comparison of Decision Tree, Logistic Regression, Random Forest and Support Vector Machines with the highest accuracy achieved being 76.96% using SVM.

This comparison can be reviewed here:

(https://rstudio-pubs-static.s3.amazonaws.com/346228_a62c6c91d5cf40869cd5aef7206826ae.html)

Our project goal is to score above the metrics quoted in this comparison, using R. For our model we will be using the popular Keras framework which offers a layered abstraction between the programmer and one of several possible backends, including CNTK, Theano, MXNet and Tensorflow. We will use Tensorflow in this project. Note that the model is small and trains very fast so a GPU-enabled or high-spec machine is not required, this code will run perfectly on a low-spec machine with as little as 4GB or RAM, probably even less.

I have chosen this dataset and model as I have not yet seen a good example where both R and Tensorflow (with Keras) is used resulting in high accuracy.

2. Methods and Analysis

Exploratory Analysis

The exploratory analysis will hopefully show some interesting indicators, and this has been used in other models. For our neural network model this may or may not add value, and we'll make this decision after the analysis phase.

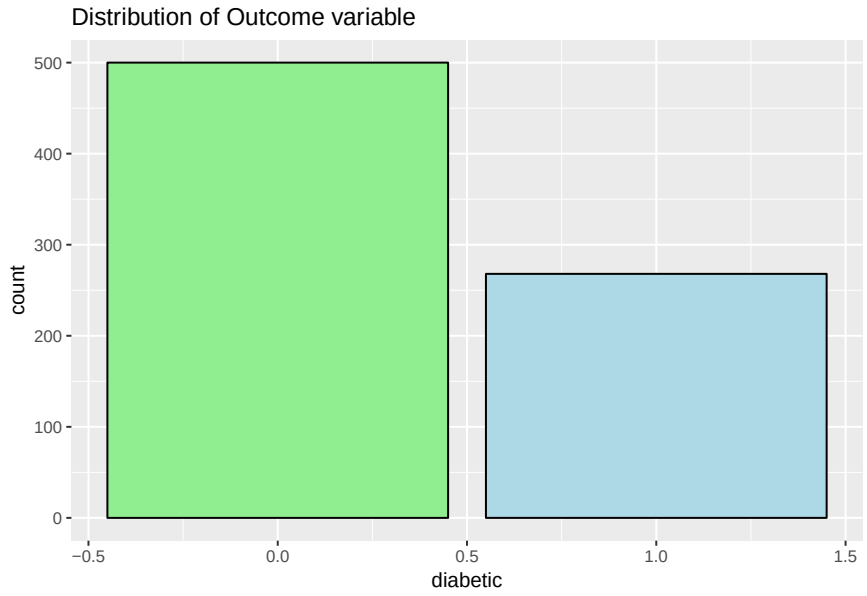
Let's have a quick look at the first few records. Notice the naming we used for the columns. As stated we have 8 predictor variables with an outcome indicator column named 'diabetic' being 1 for positive and 0 for negative.

```
##   pregnant glucose pressure triceps insulin mass pedigree age diabetic
## 1        6     148       72      35       0 33.6    0.627  50        1
## 2        1      85       66      29       0 26.6    0.351  31        0
## 3        8     183       64       0       0 23.3    0.672  32        1
## 4        1      89       66      23      94 28.1    0.167  21        0
## 5        0     137       40      35     168 43.1    2.288  33        1
## 6        5     116       74       0       0 25.6    0.201  30        0
```

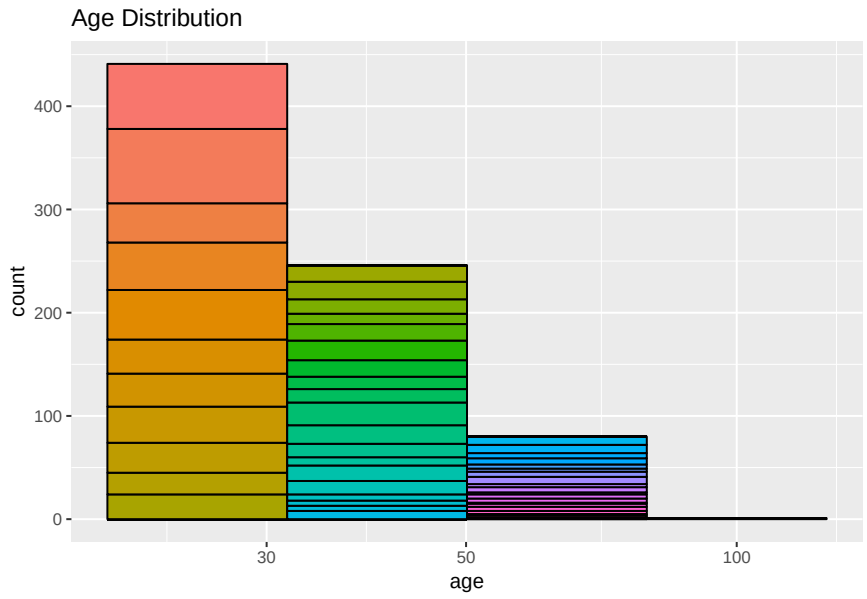
Review a summary of the dataset.

```
##     pregnant         glucose         pressure         triceps     
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00  
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00  
##  Median : 3.000   Median :117.0   Median : 72.00   Median :23.00  
##  Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54  
##  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00  
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00  
##     insulin           mass          pedigree           age        
##  Min.   :  0.0   Min.   : 0.00   Min.   :0.0780   Min.   :21.00  
##  1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437   1st Qu.:24.00  
##  Median : 30.5   Median :32.00   Median :0.3725   Median :29.00  
##  Mean   : 79.8   Mean   :31.99   Mean   :0.4719   Mean   :33.24  
##  3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262   3rd Qu.:41.00  
##  Max.   :846.0   Max.   :67.10   Max.   :2.4200   Max.   :81.00  
##     diabetic   
##  Min.   :0.000  
##  1st Qu.:0.000  
##  Median :0.000  
##  Mean   :0.349  
##  3rd Qu.:1.000  
##  Max.   :1.000
```

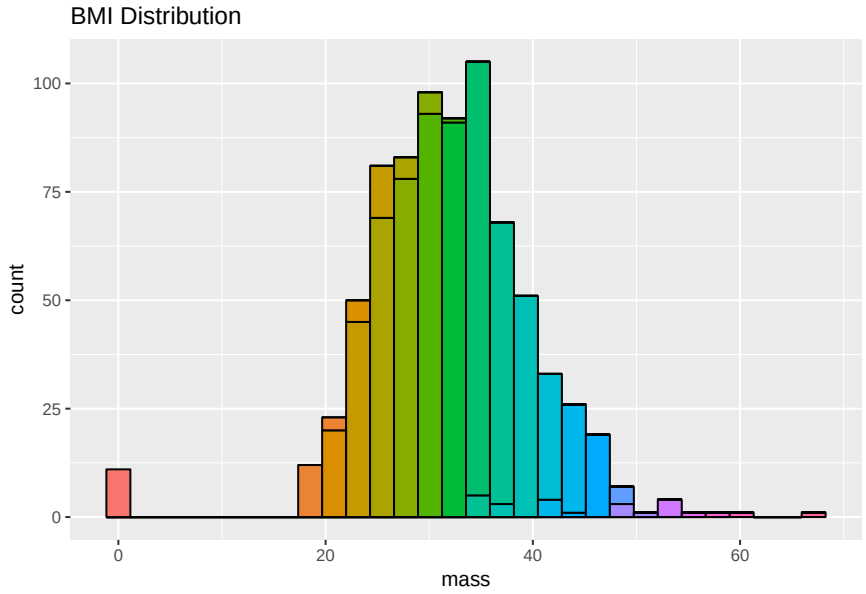
Our goal is to predict the diabetic indicator. We can see the data is a near 50/50 split between diabetic and non-diabetic.



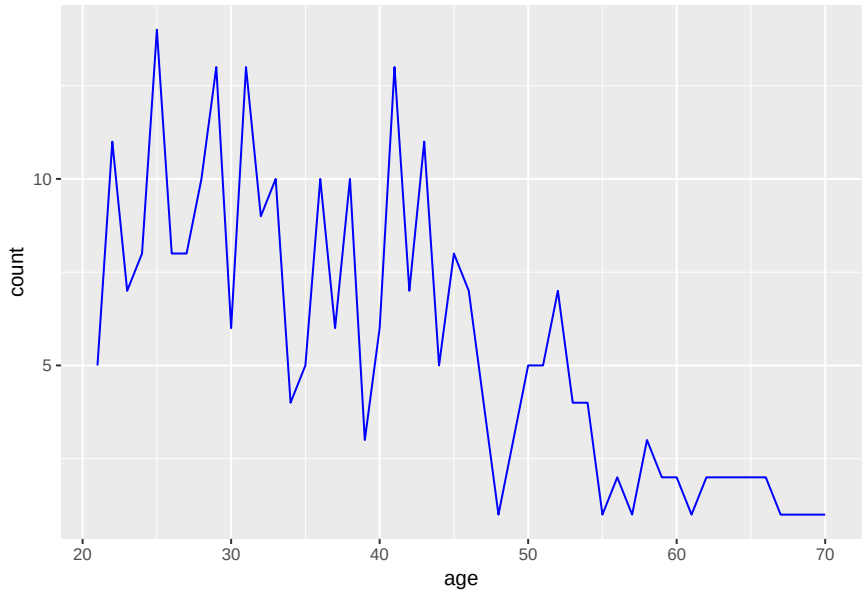
The dataset contains a fair distribution of age groups, most being under 30 with a second large group between 30 and 50 years of age.



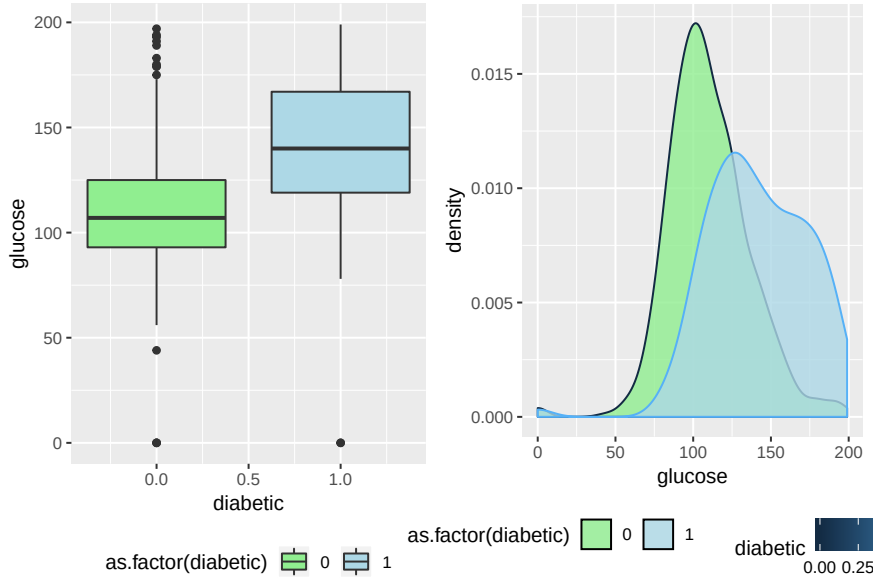
Similarly we can review the BMI index with a significant grouping over 20 upwards. A high or abnormal BMI ratio is often prevalent in diabetes patients.



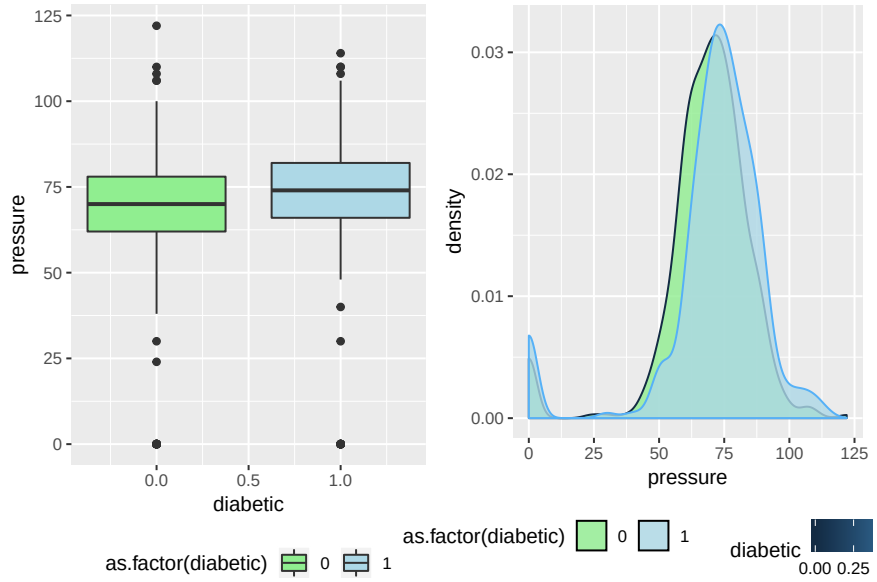
Plotting the ages of the diabetic group we see a clear indicator showing more frequency at lower age groups. Correlation does not equal causation, however diabetes is often associated with lifestyle habits, even in the West, and if younger Pima people are adopting a Western lifestyle and eating habits this could explain the effect on the younger generation versus people aged over 45.



Let's review possible correlations between being diabetic and glucose levels.

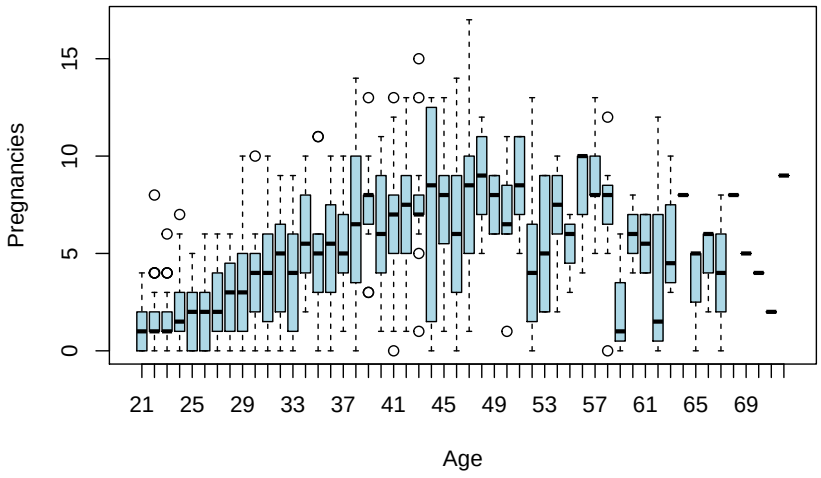


We note a distinct correlation between being diabetic and glucose levels. Similarly, let's review any possible correlation with blood pressure.

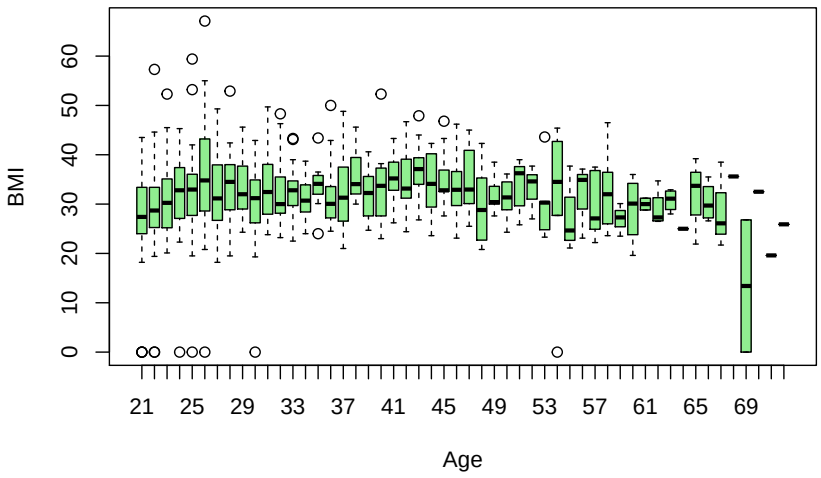


We note there is no real correlation.

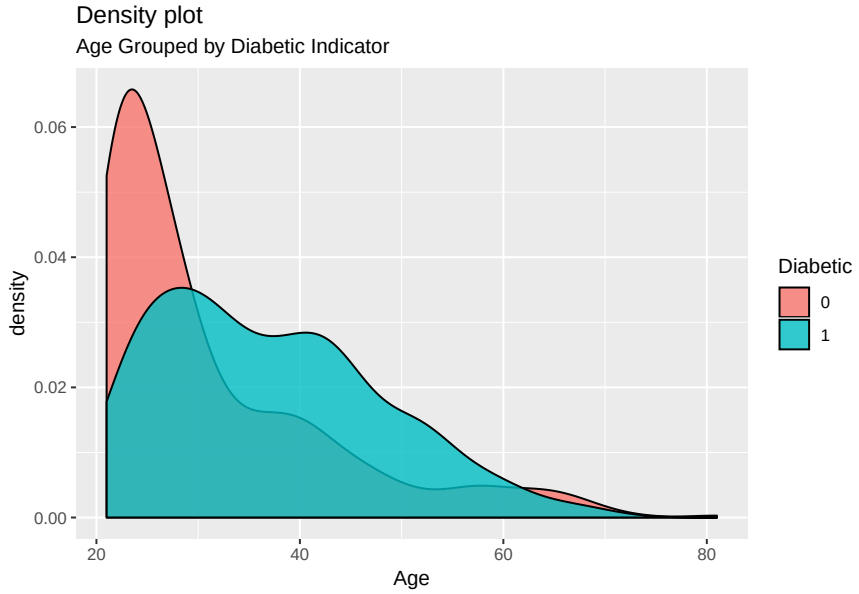
Let's examine the other variables in more depth, such as the number of pregancies over time against age.



We'll also examine BMI over time (age). This seems to remain fairly stable and we find no specific insights in either graph that directly indicates a link to being diabetic.

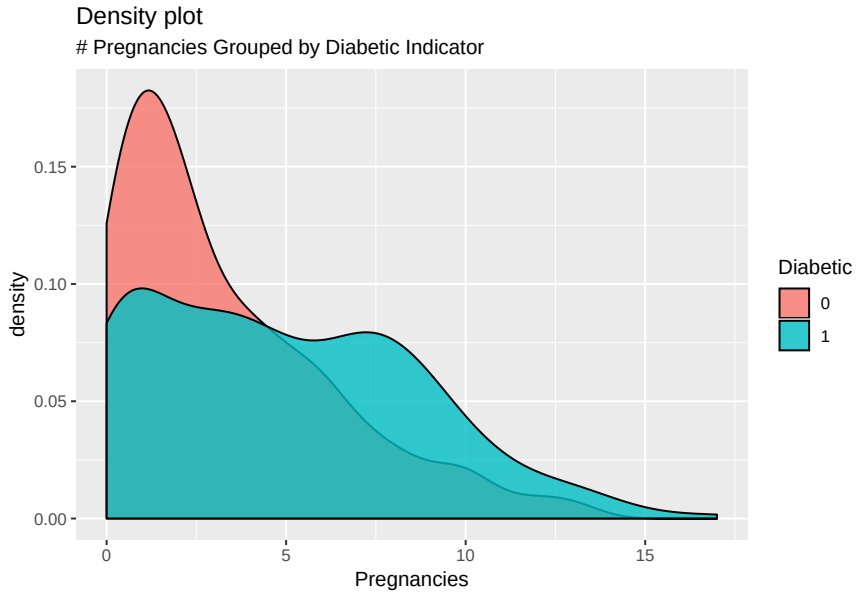


We'll further examine the data using density plots, specifically the age groups of the data samples.

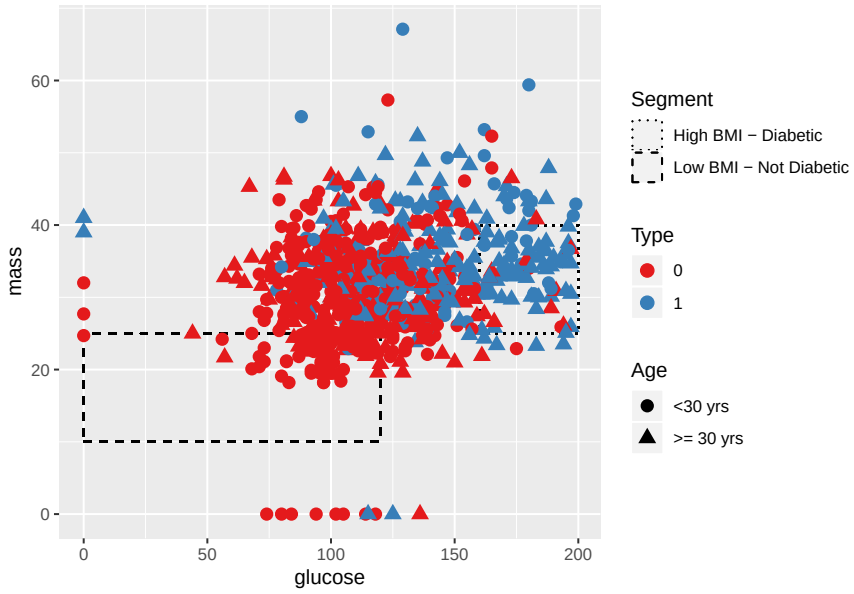


There appears to again be a clear correlation in the distribution of the 'age' variable for those that have diabetes versus those who don't.

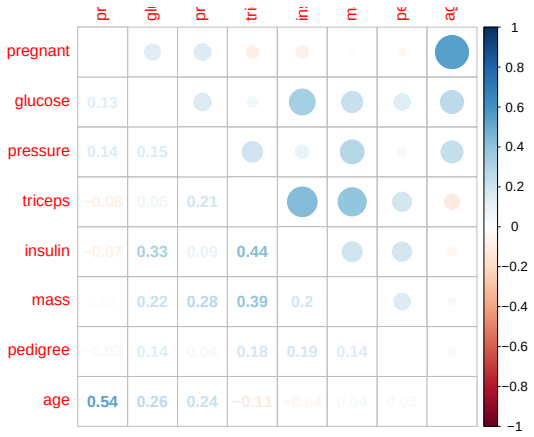
We also note it's clear in the plot below that diabetic patients are associated with a higher number of pregnancies.



Finally, let's try to implement some "basic-level clustering". This is not model-based clustering; rather, it is simply using a scatterplot and a few nice plotting parameters in ggplot2 to make some things pop right out at the viewer - again, with little room for ambiguity. What I like most here is the boxes that we can draw nicely to showcase the "clusters" a little better, along-with the multi-layered information, e.g., age, BMI, glucose, etc. Notice the cluster on the right, showing a clear grouping of how high BMI correlates with being diabetic when combined with glucose levels.



We can add a correlation plot drawn between all the numerical variables to establish the linear association between each other. As observed in the bivariate associations, Insulin and Glucose, BMI and Skin Thickness had a moderate – high linear correlation.



Considering we'll use a relatively simple and small neural network we might as well use all variables as-is. Exploratory analysis has shown some markers that can definitively be used for modelling, but since we are not medical profesionals with a clear understanding of the significance or importance of those variables, we'll proceed with the data as-is.

Training and Prediction

For training we will use the Keras Neural Network framework on Tensorflow to build a deep learning model and see if we can improve on what has been reported with other models. We aim to score at least 75% (low) and get to 78% (good) while 79% is considered very good to excellent.

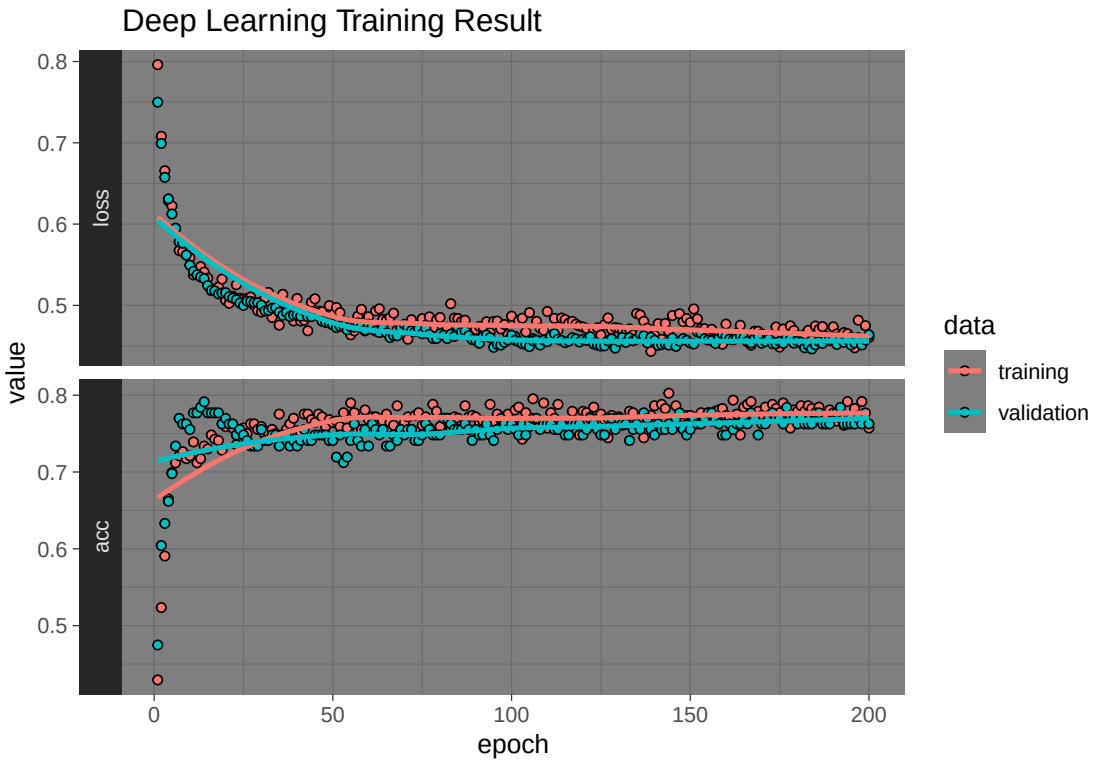
We'll split our dataset into a 90/10 training/validation split and during training the train set will further be split into a 80/20 train/test set. We need to take care against overfitting our model, and our method will be to add drop-out and batch normalization, both well-known and proven techniques to prevent overfitting. We will also monitor and plot our training to measure loss during training.

Our neural network consists of 3 layers (input, hidden, output). We use **relu** as our activation function up to the output layer where we switch to **sigmoid** which is commonly used for classification problems. We add a drop-out level of 0.1 with batch normalization after the first dense input layer. Our final model architecture is shown below.

```
## _________________________________________________________________
## Layer (type)                 Output Shape              Param #
## =================================================================
## dense_1 (Dense)              (None, 12)                108
## _________________________________________________________________
## batch_normalization_1 (BatchNorm (None, 12)                48
## _________________________________________________________________
## dropout_1 (Dropout)          (None, 12)                0
## _________________________________________________________________
## dense_2 (Dense)              (None, 8)                 104
## _________________________________________________________________
## dense_3 (Dense)              (None, 1)                 9
## =================================================================
## Total params: 269
## Trainable params: 245
## Non-trainable params: 24
## _________________________________________________________________
```

During training we split our training set into a 80/20 split for testing. Training takes about 2 minutes on any machine.

We can monitor and plot the training progress as shown below.



Note the model converges well and no overfitting is present.

3. Results

Results can be shown using the **score()** function provided by Keras.

```
## $loss
## [1] 0.4461859
##
## $acc
## [1] 0.8051948
```

We can review the confusion matrix to confirm our results.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 48 12
##          1  3 14
##
##                Accuracy : 0.8052
##                  95% CI : (0.6991, 0.8867)
##     No Information Rate : 0.6623
##     P-Value [Acc > NIR] : 0.004302
##
##                   Kappa : 0.5241
##  Mcnemar's Test P-Value : 0.038867
##
##             Sensitivity : 0.9412
##             Specificity : 0.5385
##          Pos Pred Value : 0.8000
##          Neg Pred Value : 0.8235
##              Prevalence : 0.6623
##          Detection Rate : 0.6234
##    Detection Prevalence : 0.7792
##       Balanced Accuracy : 0.7398
##
##        'Positive' Class : 0
##
```

4. Conclusion

The resulting accuracy of 0.8051948 is exceptional for this problem. Adding dropout and batch normalization boosted our model significantly compared to what has previously been reported. The same model in Python yielded 78% without dropout and batch-normalization.

I believe we've reached our goal of demonstrating the use of Tensorflow and Keras in R, with good accuracy results. The benefit of Keras is that models previously built in Python can be ported fairly painlessly to R, if R is what you prefer to use.