What is the Most Important Factor that affects Diabetes? - Neural Network Approach

Code

0. Load library

Load all required library within our coding and explain what its function.

Hide

```
set.seed(0) # to lock the data
library(gplots) #to visualize correlation through heatmap
```

```

Attaching package: 'gplots'

The following object is masked from 'package:stats':

    lowess
```

Hide

```
library(psych) #to find kurtosis and skewness value
```

```

Attaching package: 'psych'

The following object is masked from 'package:car':

    logit
```

Hide

```
library(caTools) #to split sample data
library(neuralnet) #to neural network machine learning
library(caret) # to calculate accuracy
```

```
Loading required package: lattice
Loading required package: ggplot2

Attaching package: 'ggplot2'

The following objects are masked from 'package:psych':

    %+%, alpha
```

Hide

```
library(e1071) # required by caret library
```

1. Extract, Transform, Load (ETL)

1.1. Load data

Load the data frame called “Prediksi Diabetes.csv” through this link (https://docs.google.com/spreadsheets/d/e/2PACX-1vSiFblgLdNoUumGPmBQLfW38gzqSxl8GIYqKe1IIjjyePQf0pdehYi59XY7jM4qJNuOOQrUBQ40vVZ1/pub?gid=849103461&single=true&output=csv). Then print it with *head* function to take a glance of what the data looks like.

Hide

```
df <- read.csv("https://docs.google.com/spreadsheets/d/e/2PACX-1vSiFblgLdNoUumGPmBQLfW38gzqSxl8GIYqKe1IIjjyePQf0pdehYi59XY7jM4qJNuOOQrUBQ40vVZ1/pub?gid=849103461&single=true&output=csv")
dfcor <- read.csv("https://docs.google.com/spreadsheets/d/e/2PACX-1vSiFblgLdNoUumGPmBQLfW38gzqSxl8GIYqKe1IIjjyePQf0pdehYi59XY7jM4qJNuOOQrUBQ40vVZ1/pub?gid=849103461&single=true&output=csv")
print(head(df))
```

	GlukosaSewaktu <int>	TekananDarahAtas <int>	KetebalanKulit <int>	Insulin <int>	BMI <dbl>	FaktorRisiko <dbl>	U… <int>	Diabet <ir
1	148	72	35	0	33.6	0.627	50	
2	85	66	29	0	26.6	0.351	31	
3	183	64	0	0	23.3	0.672	32	
4	89	66	23	94	28.1	0.167	21	
5	137	40	35	168	43.1	2.288	33	
6	116	74	0	0	25.6	0.201	30	

6 rows

1.2. Data Dictionary - feature, target, independent

The detailed description of the entire available features in tabular format. I use this service (https://www.tablesgenerator.com/markdown_tables) to help me to visualize it.

No	Field Name	Data Type	Description	Classification	Processed
1	GlukosaSewaktu	Integer	The level of blood sugar in mg/dL	Independent	Yes
2	TekananDarahAtas	Integer	The value of diastolic blood pressure in mm Hg	Independent	Yes

No	Field Name	Data Type	Description	Classification	Processed
3	KetebalanKulit	Integer	The value of skin density in mm	Independent	Yes
4	Insulin	Integer	The value of insulin level	Independent	Yes
5	BMI	Double	The value of Body Mass Index, the measure of body fat that based on height and weight	Independent	Yes
6	FaktorRisiko	Double	The likelihood of diabetes based on family history in percentage	Independent	Yes
7	Umur	Integer	The value of age in years old	Independent	Yes
8	Diabetes	Factorial	The classification of whether a person is diagnosed with diabetes or not. 1 = Diabetes 0 = Healthy	Target	Yes

The total of actual data: **768 observations**

2. Feature engineering

2.1. Feature engineering

2.1.1. Remove features that not needed

Inessential

2.1.2. Add features if needed

Inessential

2.2. Check if there are NULL value

This phase evaluates if there is any *empty data* that could possibly ruin the research process.

Hide

```
colSums(is.na(df))
```

```
  GlukosaSewaktu TekananDarahAtas   KetebalanKulit          Insulin              BMI     FaktorR
isiko
               0                0                0                0                0
0
           Umur         Diabetes
              0                0
```

Since there is no NA value within our data. I reevaluate the nature of the data and i figure out that:

1. In this case by considering the nature of the data object, some of the value from certain features are shown as 0 which is not feasible in medical sense.
2. I assume the NA value in this data format is **not represented by "NA" but "0" value.**

Hide

```
colSums(df == 0, )
```

```
  GlukosaSewaktu TekananDarahAtas   KetebalanKulit          Insulin              BMI     FaktorR
isiko
               5               35              227              374               11
0
            Umur         Diabetes
               0              500
```

We ignore *Diabetes* variable since its 0 value represents *Healthy* patient.

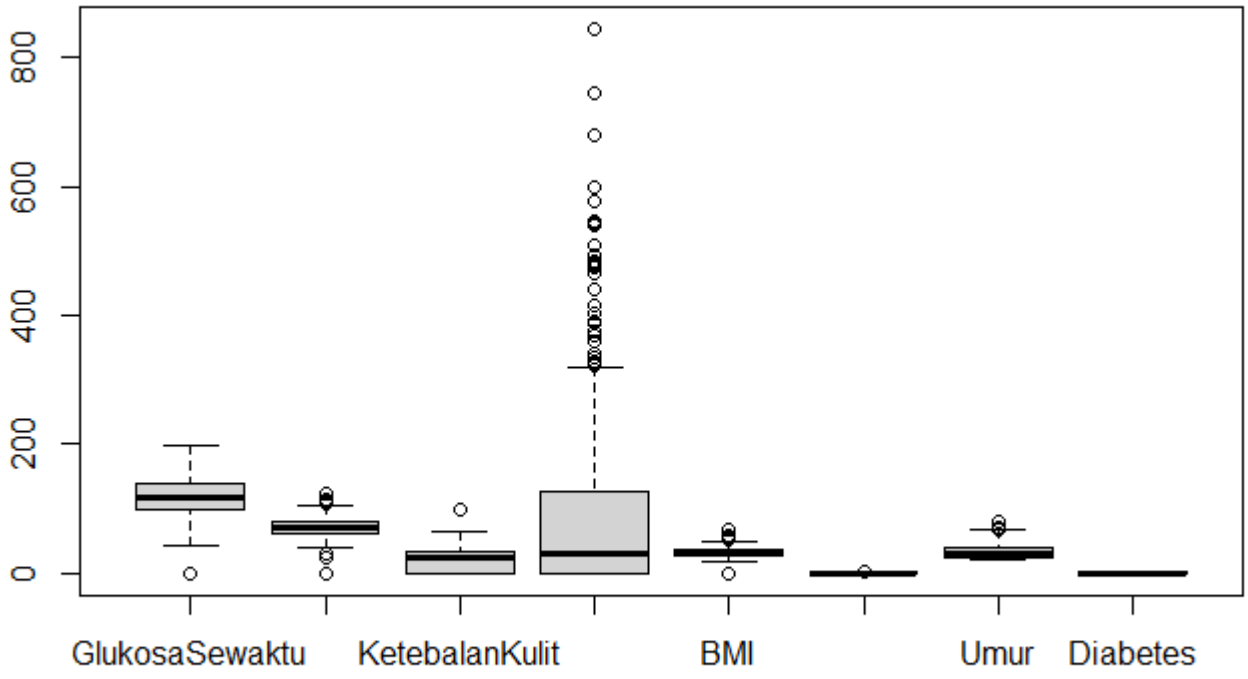
Conclusion: **There are NULL values in several features, hence I must proceed with the data imputing process to solve the issue**

2.2.1 Imputing process (if there's NULL)

In order to choose the appropriate imputing process, I need to **check whether outliers** are exist or not within the data. Creating a boxplot would help me to find the outliers.

Hide

```
boxplot(df)
```



Based on the boxplot shown above, we can identify there are *outliers* that represented by *circles* within the visualization. Hence, I must proceed with **replacing each of the 0 values with median value** of each feature. First by calculating the median value.

Hide

```
median.glucose <- median(df$GlukosaSewaktu)
median.blood <- median(df$TekananDarahAtas)
median.skin <- median(df$KetebalanKulit)
median.insulin <- median(df$Insulin)
median.bmi <- median(df$BMI)

print(paste("Median GlukosaSewaktu =", median.glucose))
```

```
[1] "Median GlukosaSewaktu = 117"
```

Hide

```
print(paste("Median TekananDarahAtas =", median.blood))
```

```
[1] "Median TekananDarahAtas = 72"
```

Hide

```
print(paste("Median KetebelanKulit =", median.skin))
```

```
[1] "Median KetebelanKulit = 23"
```

Hide

```
print(paste("Median Insulin =", median.insulin))
```

```
[1] "Median Insulin = 30.5"
```

Hide

```
print(paste("Median BMI =", median.bmi))
```

```
[1] "Median BMI = 32"
```

After I found the median value for each required feature, I continue to **insert the median amount into each of the feature**, replacing the previously-empty data.

Hide

```
df[df$GlukosaSewaktu == 0, "GlukosaSewaktu"] <- median.glucose
df[df$TekananDarahAtas == 0, "TekananDarahAtas"] <- median.blood
df[df$KetebalanKulit == 0, "KetebalanKulit"] <- median.skin
df[df$Insulin == 0, "Insulin"] <- median.insulin
df[df$BMI == 0, "BMI"] <- median.bmi
head(df)
```

	GlukosaSewaktu <dbl>	TekananDarahAtas <dbl>	KetebalanKulit <dbl>	Insulin <dbl>	BMI <dbl>	FaktorRisiko <dbl>	U… <int>	Diabet <ir
1	148	72	35	30.5	33.6	0.627	50	
2	85	66	29	30.5	26.6	0.351	31	
3	183	64	23	30.5	23.3	0.672	32	
4	89	66	23	94.0	28.1	0.167	21	
5	137	40	35	168.0	43.1	2.288	33	
6	116	74	23	30.5	25.6	0.201	30	

6 rows

Reexamine if there is any empty data left.

Hide

```
colSums(df == 0, )
```

```
  GlukosaSewaktu TekananDarahAtas   KetebalanKulit          Insulin              BMI     FaktorR
isiko
               0                0                0                0                0
0
            Umur         Diabetes
               0              500
```

Conclusion : **There is no empty data left because all considered variables have shown 0 value of NULL data.**

2.3. Check Unary data

In this phase, the data must be checked whether it has similar feature or not (unary data).

Hide

```
apply(df, 2, max) - apply(df, 2, min)
```

```
  GlukosaSewaktu TekananDarahAtas   KetebalanKulit          Insulin              BMI     FaktorR
isiko
         155.000           98.000           92.000          832.000           48.900
2.342
            Umur         Diabetes
          60.000            1.000
```

Conclusion : **Since there is no 0 value shown in the calculation, it means the unary value is not presence within the data.**

3. Exploratory Data Analysis (EDA)

3.1. Descriptive Statistics

Summary

Hide

```
summary(df)
```

```
 GlukosaSewaktu   TekananDarahAtas KetebalanKulit     Insulin           BMI         FaktorRisik
o
 Min.   : 44.00   Min.   : 24.00   Min.   : 7.00   Min.   : 14.00   Min.   :18.20   Min.   :0.07
80
 1st Qu.: 99.75   1st Qu.: 64.00   1st Qu.:23.00   1st Qu.: 30.50   1st Qu.:27.50   1st Qu.:0.24
37
 Median :117.00   Median : 72.00   Median :23.00   Median : 31.25   Median :32.00   Median :0.37
25
 Mean   :121.66   Mean   : 72.39   Mean   :27.33   Mean   : 94.65   Mean   :32.45   Mean   :0.47
19
 3rd Qu.:140.25   3rd Qu.: 80.00   3rd Qu.:32.00   3rd Qu.:127.25   3rd Qu.:36.60   3rd Qu.:0.62
62
 Max.   :199.00   Max.   :122.00   Max.   :99.00   Max.   :846.00   Max.   :67.10   Max.   :2.42
00
      Umur          Diabetes
 Min.   :21.00   Min.   :0.000
 1st Qu.:24.00   1st Qu.:0.000
 Median :29.00   Median :0.000
 Mean   :33.24   Mean   :0.349
 3rd Qu.:41.00   3rd Qu.:1.000
 Max.   :81.00   Max.   :1.000
```

Based on the summary above, the insights we can possibly draw are:

1. **35%** of the people are diagnosed with diabetes.
2. **75%** of the people falls into “overweight” or “obesity” category. Since > 25 means overweight and > 30 means obesity.

Correlation

For correlation, I refer to the original data frame because **the imputed data frame could distort the true correlations between features** since the resolution of the data are increased. Hence the original data frame will better represent the true correlation between target and independent variables.

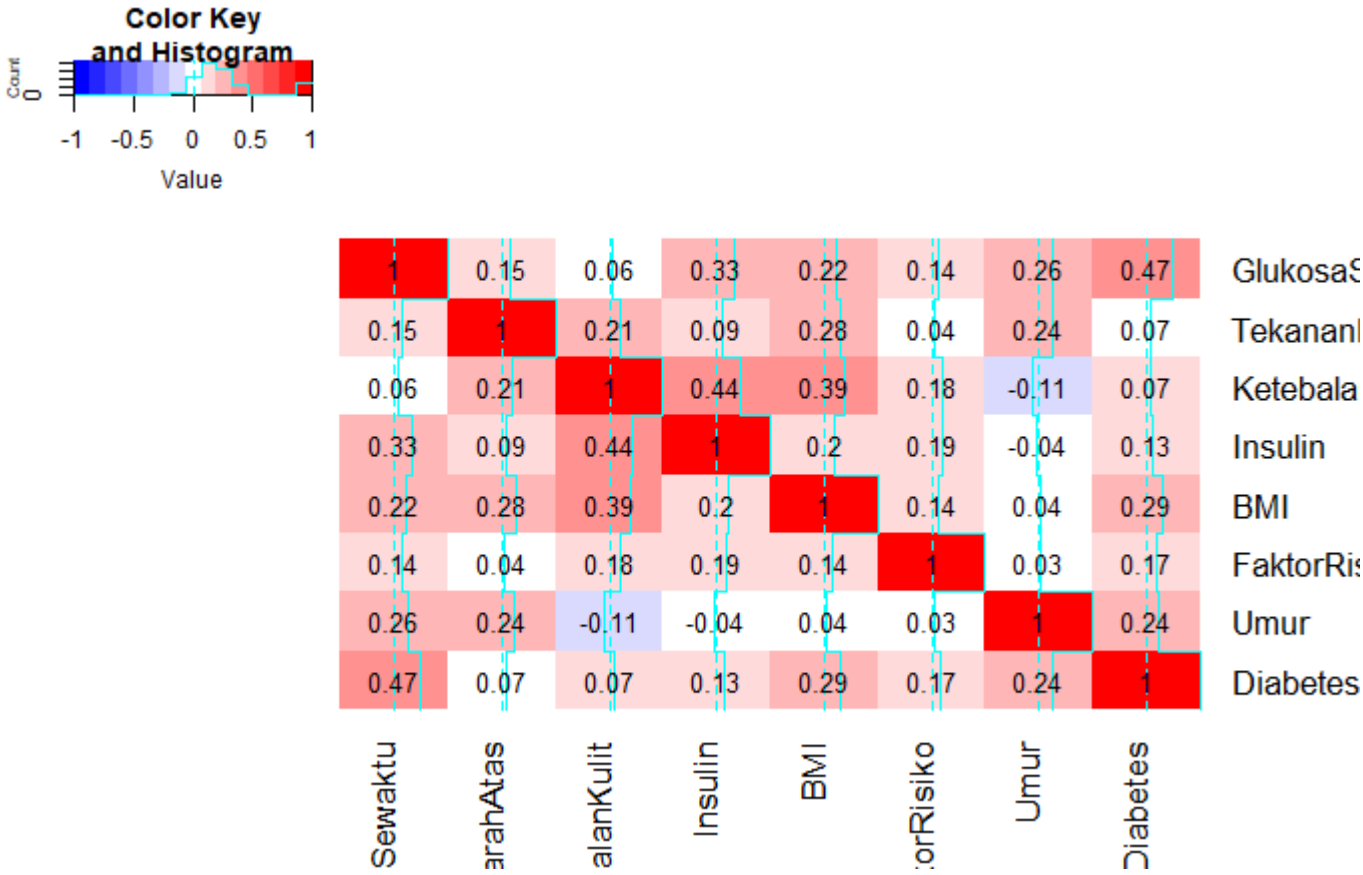
Hide

```
correlation <- cor(dfcor)
print(correlation)
```

```
                GlukosaSewaktu TekananDarahAtas KetebalanKulit     Insulin        BMI FaktorRis
iko        Umur
GlukosaSewaktu       1.00000000       0.15258959     0.05732789  0.33135711 0.22107107   0.13733
730  0.26351432
TekananDarahAtas     0.15258959       1.00000000     0.20737054  0.08893338 0.28180529   0.04126
495  0.23952795
KetebalanKulit       0.05732789       0.20737054     1.00000000  0.43678257 0.39257320   0.18392
757 -0.11397026
Insulin              0.33135711       0.08893338     0.43678257  1.00000000 0.19785906   0.18507
093 -0.04216295
BMI                  0.22107107       0.28180529     0.39257320  0.19785906 1.00000000   0.14064
695  0.03624187
FaktorRisiko         0.13733730       0.04126495     0.18392757  0.18507093 0.14064695   1.00000
000  0.03356131
Umur                 0.26351432       0.23952795    -0.11397026 -0.04216295 0.03624187   0.03356
131  1.00000000
Diabetes             0.46658140       0.06506836     0.07475223  0.13054795 0.29269466   0.17384
407  0.23835598
                   Diabetes
GlukosaSewaktu   0.46658140
TekananDarahAtas 0.06506836
KetebalanKulit   0.07475223
Insulin          0.13054795
BMI              0.29269466
FaktorRisiko     0.17384407
Umur             0.23835598
Diabetes         1.00000000
```

Hide

```
heatmap.2(correlation,
          cellnote = round(correlation, 2),
          notecol = "black",
          Colv = FALSE,
          Rowv = FALSE,
          dendrogram = "none",
          col = colorRampPalette(c("blue", "white", "red")))
```



Based on the correlation above, the insights we can possibly draw are:

1. Glucose level has a less significant correlation against Diabetes but **the most correlation compared to other variables against Diabetes**. It means **glucose level is the most influential factor** in determining the possibility of having Diabetes.
2. All independent variables within the data have **positive** correlation value against Diabetes which means **all of them contribute on incremental value of Diabetes variable.**
3. We should **remove TekananDarahAtas, KetebalanKulit, Insulin, and FaktorRisiko** variables off the calculation. Since they only have tiny correlations which are insignificant *(below 0.2 correlation value)* compared to other provided variables. This method should be done to simplify the machine learning process while still sustaining the optimal accuracy.

Hide

```
kurskew <- describe(df)[c("kurtosis", "skew")]
print(kurskew)
```

	kurtosis <dbl>	skew <dbl>
GlukosaSewaktu	-0.27	0.53
TekananDarahAtas	1.07	0.14
KetebalanKulit	4.66	1.22
Insulin	9.64	2.68
BMI	0.90	0.60

	kurtosis <dbl>	skew <dbl>
FaktorRisiko	5.53	1.91
Umur	0.62	1.13
Diabetes	-1.60	0.63

8 rows

Based on the description of kurtosis and skewness values above, the insights we can possibly draw are:

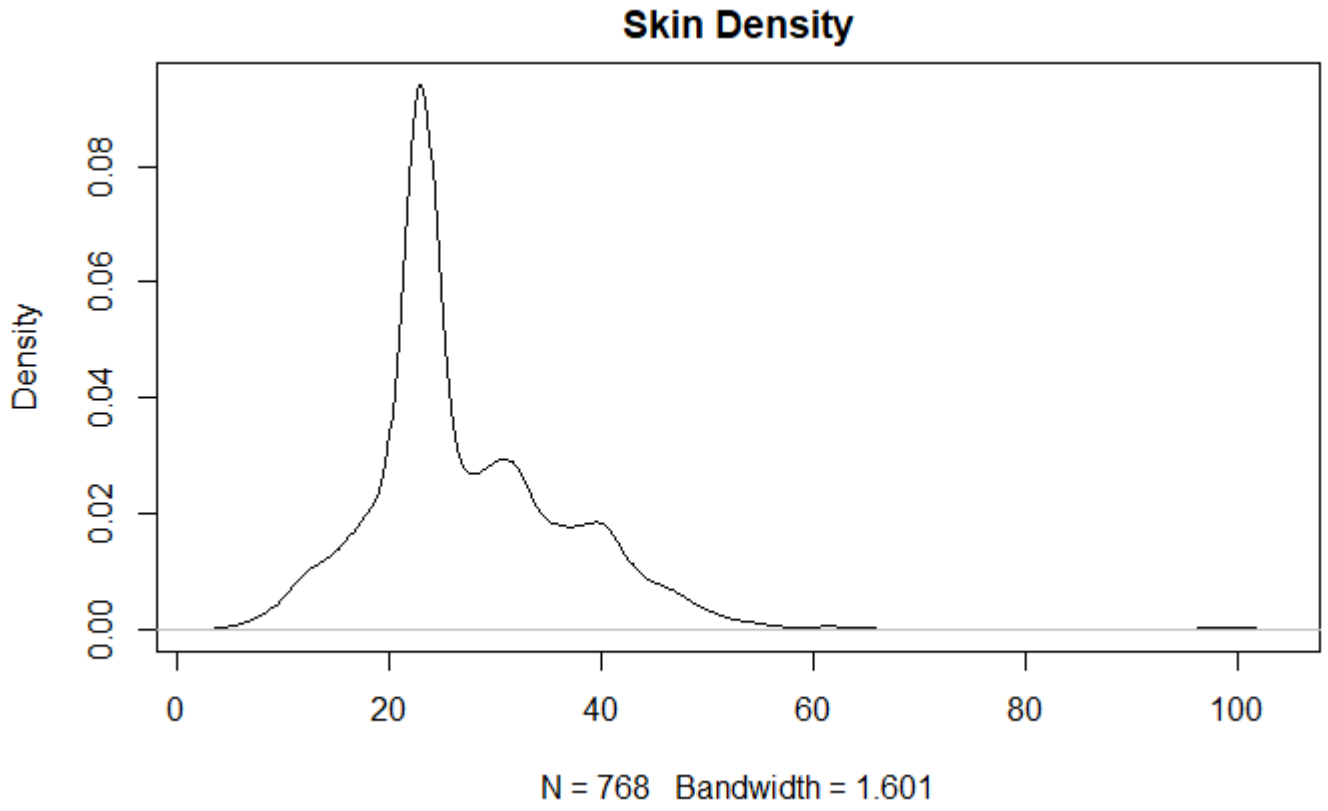
1. **KetebalanKulit value is not normally distributed** since its kurtosis value = +4.66 exceeds the normal distribution limit (> +2).
2. **Insulin variable is not normally distributed** since its kurtosis value = +9.64 exceeds the normal distribution limit (> +2) and its skewness value = +2.68 exceeds normal distribution range (-2 to +2).
3. **FaktorRisiko value is not normally distributed** since its kurtosis value = +5.53 exceeds the normal distribution limit (> +2).
4. **The rest of the variables are normally distributed** because their kurtosis value do not exceed the limit +2 and their skewness value do not exceed the range of -2 until +2.

3.2. Visualisation

The following density plots are provided to show 3 variables that are not normally distributed.

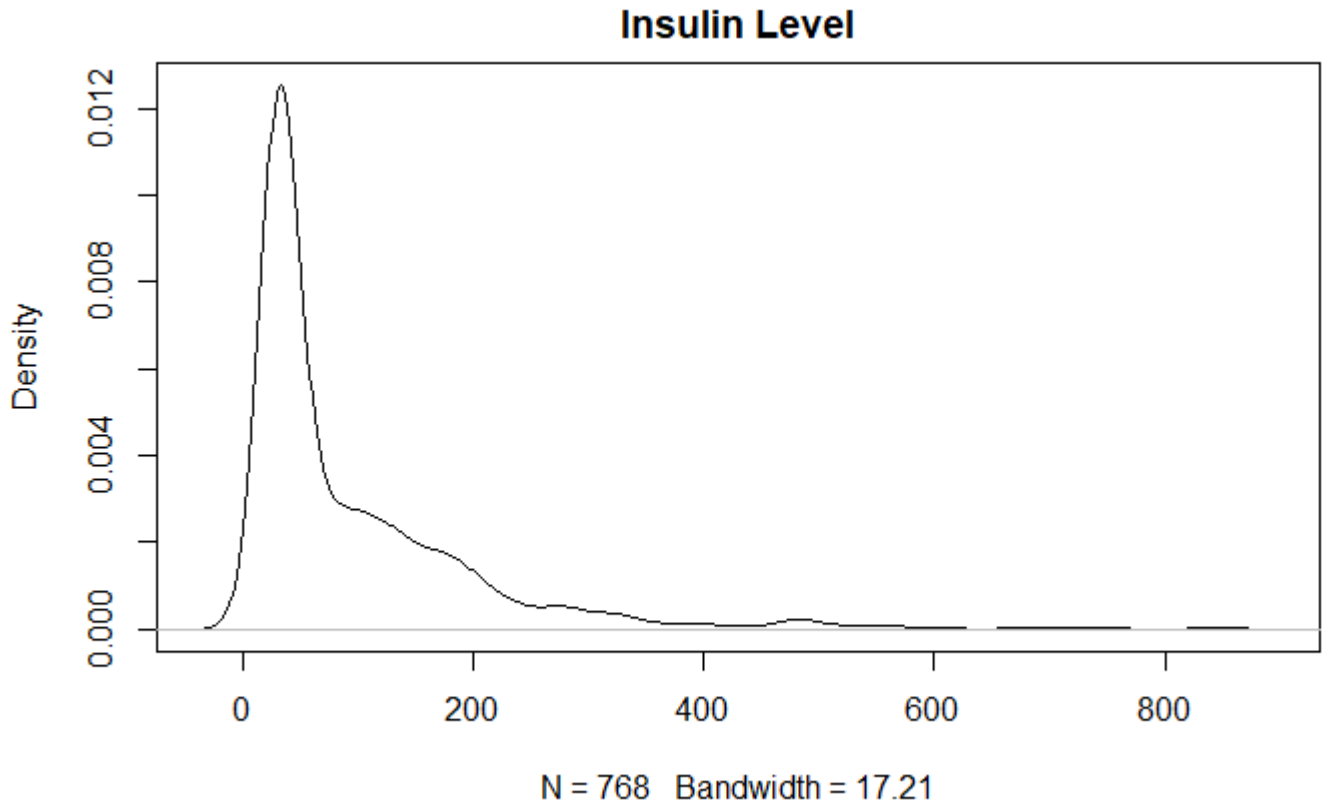
Hide

```
plot(density(df$KetebalanKulit),
     main = "Skin Density")
```



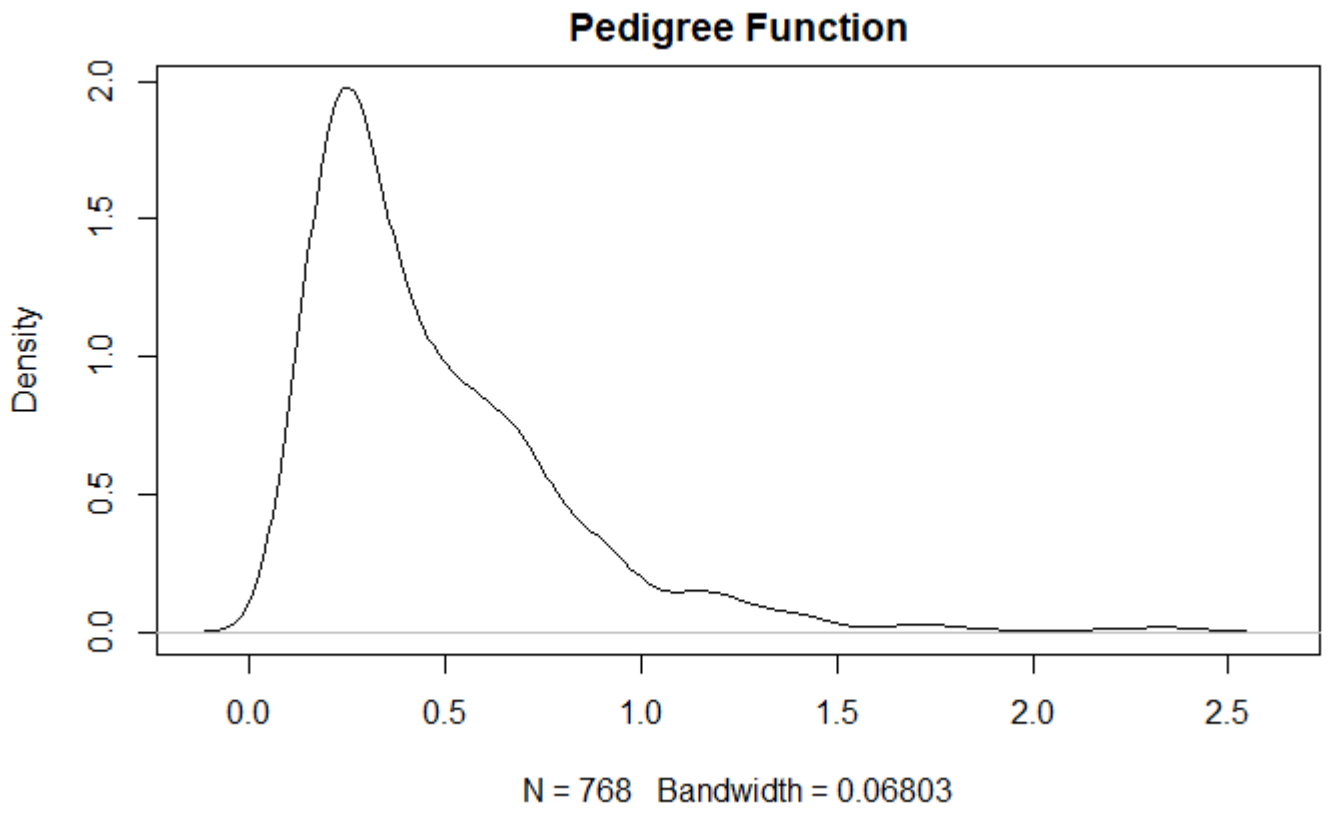
Hide

```
plot(density(df$Insulin),
     main = "Insulin Level")
```



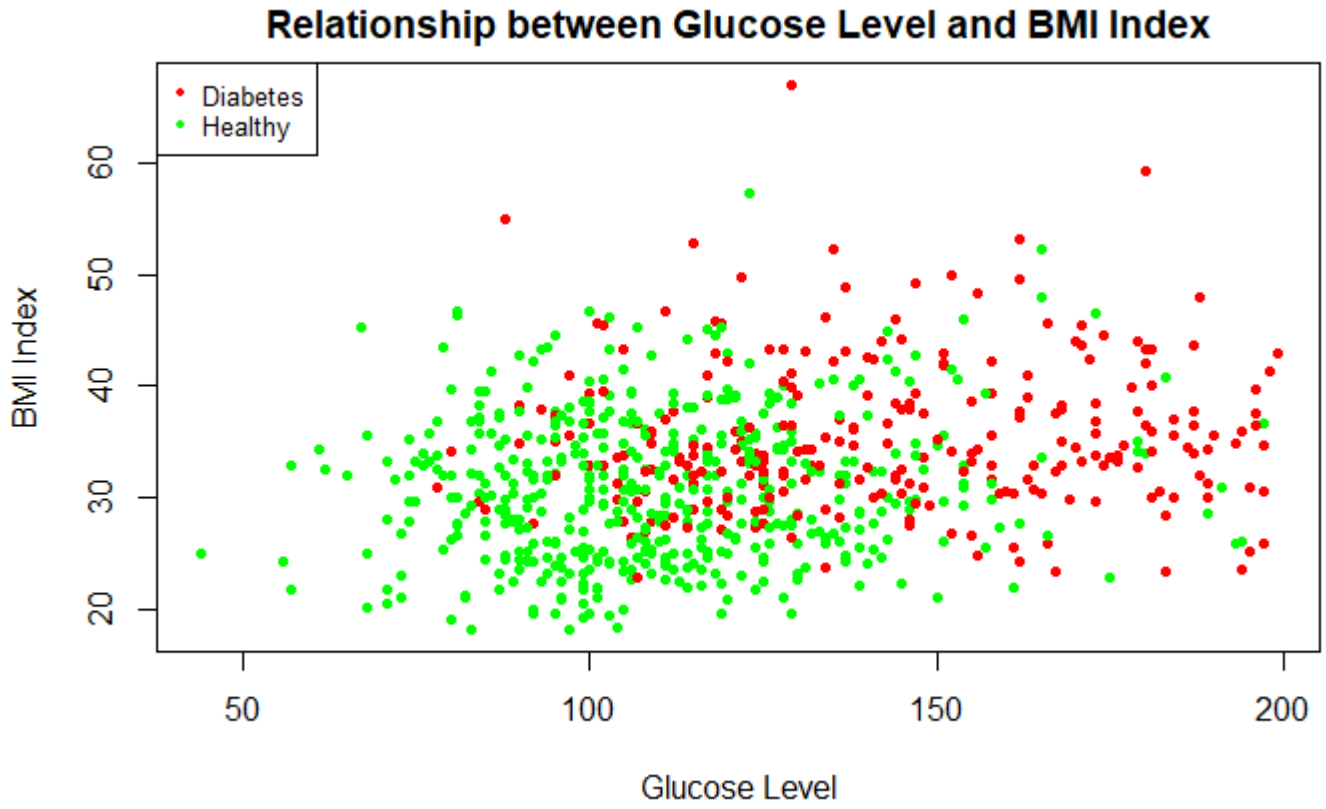
Hide

```
plot(density(df$FaktorRisiko),
     main = "Pedigree Function")
```



Hide

```
plot(df$GlukosaSewaktu, df$BMI,
     main = "Relationship between Glucose Level and BMI Index",
     xlab = "Glucose Level",
     ylab = "BMI Index",
     pch = 20,
     col = ifelse(df$Diabetes == 1, "red", "green"))
legend("topleft", legend = c("Diabetes", "Healthy"),
       col = c("red", "green"), pch = 20:20, cex = 0.8)
```



Based on the chart correlation above, the insights we can possibly draw are:

1. People who have **glucose level lower than 75 are healthy** regardless their BMI Index.
2. People who have **BMI Index under 23 are healthy** regardless their glucose level.
3. People who have **glucose level lower than 110 and maintain its BMI Index under 28 are healthy.**

4. Fitting/Model

4.1 Splitting

In this phase, because there are various range between features, it is necessary to standardize the data value using **scale function**.

This function centers or scales the columns of a numeric matrix where **0 means the observation value equals the mean of each column**. In this part, Diabetes variable is also excluded for easier accuracy calculation.

Hide

```
dfscale <- as.data.frame(scale(df))
dfscale$Diabetes <- df$Diabetes
print(head(dfscale))
```

	GlukosaSewaktu <dbl>	TekananDarahAtas <dbl>	KetebalanKulit <dbl>	Insulin <dbl>	BMI <dbl>	FaktorRisik <dbl
1	0.8654807	-0.0319691	0.8305724	-0.607804864	0.1671312	0.468186
2	-1.2042810	-0.5279745	0.1804488	-0.607804864	-0.8509963	-0.364823

	GlukosaSewaktu <dbl>	TekananDarahAtas <dbl>	KetebalanKulit <dbl>	Insulin <dbl>	BMI <dbl>	FaktorRisik <dbl
3	2.0153484	-0.6933097	-0.4696748	-0.607804864	-1.3309707	0.604003
4	-1.0728676	-0.5279745	-0.4696748	-0.006180565	-0.6328261	-0.920163
5	0.5040938	-2.6773314	0.8305724	0.694924919	1.5488758	5.481337
6	-0.1858268	0.1333660	-0.4696748	-0.607804864	-0.9964431	-0.817545

6 rows | 1-8 of 8 columns

Next, the data should be splitted into 2 parts. **80% of them becomes the data for training purpose** and the **20% becomes the data for accuracy testing purpose.**

Hide

```
split <- sample.split(dfscale$Diabetes, SplitRatio = 0.8)
train <- dfscale[split == TRUE, ]
test <- dfscale[split == FALSE, ]
```

4.2 Modeling

In this phase, I will use **Neural Network** approach to create my machine learning model. Neural network is a series of algorithm that endeavors to recognize underlying relationships in a set of our data through a process **that mimics the way the human brain operates.** Since there is no rule of thumb to optimize my this type of machine learning, I am going to create 4 models initially to find out which one has the best accuracy as the following:

Model 1 with 8 nodes 1 hidden layer

Hide

```
model1 <- neuralnet(Diabetes ~ GlukosaSewaktu + TekananDarahAtas + KetebalanKulit + Insulin + BM
I + FaktorRisiko + Umur,
                    train,
                    hidden = c(8),
                    linear.output = TRUE,
                    stepmax = 1000000)
```

Model 2 with 8 nodes 2 hidden layer

Hide

```
model2 <- neuralnet(Diabetes ~ GlukosaSewaktu + TekananDarahAtas + KetebalanKulit + Insulin + BM
I + FaktorRisiko + Umur,
                    train,
                    hidden = c(8,
                               8),
                    linear.output = TRUE,
                    stepmax = 1000000)
```

Model 3 with 16 nodes 1 hidden layer

Hide

```
model3 <- neuralnet(Diabetes ~ GlukosaSewaktu + TekananDarahAtas + KetebalanKulit + Insulin + BM
I + FaktorRisiko + Umur,
                    train,
                    hidden = c(16),
                    linear.output = TRUE,
                    stepmax = 1000000)
```

Model 4 with 16 nodes 2 hidden layer

Hide

```
model4 <- neuralnet(Diabetes ~ GlukosaSewaktu + TekananDarahAtas + KetebalanKulit + Insulin + BM
I + FaktorRisiko + Umur,
                    train,
                    hidden = c(16,
                               16),
                    linear.output = TRUE,
                    stepmax = 1000000)
```

Model 5 with 6 nodes 1 hidden layer

Based on the first 4 models, increasing nodes and layer only resulting in decreased accuracy, I further approach the model by utilizing simpler nodes from 1 - 7 nodes within the model. I found out that **6 nodes are enough to provide the highest accuracy** among all models.

Hide

```
model5 <- neuralnet(Diabetes ~ GlukosaSewaktu + TekananDarahAtas + KetebalanKulit + Insulin + BM
I + FaktorRisiko + Umur,
                    train,
                    hidden = c(6),
                    linear.output = TRUE,
                    stepmax = 1000000)
```

Removing low-correlation variables to simplify model 5 as Optimized Model

After we found the most accurate model, I proceed to remove 4 variables which are:

- TekananDarahAtas
- KetebalanKulit
- Insulin
- FaktorRisiko

Because all of them have **correlation below 0.2** against Diabetes and it is highly unlikely to contribute as much as other variables while still maintaining the amount of accuracy that it offers.

Hide

```
modelopt <- neuralnet(Diabetes ~ GlukosaSewaktu + BMI + Umur,
                  train,
                  hidden = c(6),
                  linear.output = TRUE,
                  stepmax = 1000000)
```

5. Model Performance

5.1 Inferential Statistic approach

Accuracy testing for Model 1

Hide

```
key <- test$Diabetes
prediction1 <- as.data.frame(compute(model1, test))
prediction11 <- ifelse(prediction1$net.result >= 0.5, 1, 0)
print(head(key))
```

```
[1] 1 1 1 1 1 1
```

Hide

```
print(head(prediction11))
```

```
[1] 0 0 1 1 1 0
```

Accuracy testing for Model 2

Hide

```
key <- test$Diabetes
prediction2 <- as.data.frame(compute(model2, test))
prediction22 <- ifelse(prediction2$net.result >= 0.5, 1, 0)
print(head(key))
```

```
[1] 1 1 1 1 1 1
```

Hide

```
print(head(prediction22))
```

```
[1] 0 0 0 1 1 0
```

Accuracy testing for Model 3

Hide

```
key <- test$Diabetes
prediction3 <- as.data.frame(compute(model3, test))
prediction33 <- ifelse(prediction3$net.result >= 0.5, 1, 0)
print(head(key))
```

```
[1] 1 1 1 1 1 1
```

Hide

```
print(head(prediction33))
```

```
[1] 0 0 1 1 1 0
```

Accuracy testing for Model 4

Hide

```
key <- test$Diabetes
prediction4 <- as.data.frame(compute(model4, test))
prediction44 <- ifelse(prediction4$net.result >= 0.5, 1, 0)
print(head(key))
```

```
[1] 1 1 1 1 1 1
```

Hide

```
print(head(prediction44))
```

```
[1] 0 0 0 1 1 0
```

Accuracy testing for Model 5

Hide

```
key <- test$Diabetes
prediction5 <- as.data.frame(compute(model5, test))
prediction55 <- ifelse(prediction5$net.result >= 0.5, 1, 0)
print(head(key))
```

```
[1] 1 1 1 1 1 1
```

Hide

```
print(head(prediction55))
```

```
[1] 1 0 1 1 1 0
```

Accuracy testing for Optimized Model

Hide

```
key <- test$Diabetes
prediction6 <- as.data.frame(compute(modelopt, test))
predictionopt <- ifelse(prediction6$net.result >= 0.5, 1, 0)
print(head(key))
```

```
[1] 1 1 1 1 1 1
```

Hide

```
print(head(predictionopt))
```

```
[1] 1 0 1 1 1 0
```

Accuracy result for Model 1

Hide

```
confusionMatrix(table(key, prediction11))
```

```
Confusion Matrix and Statistics

   prediction11
key  0  1
  0 82 18
  1 22 32

               Accuracy : 0.7403
                 95% CI : (0.6635, 0.8075)
    No Information Rate : 0.6753
    P-Value [Acc > NIR] : 0.04903

                  Kappa : 0.4197

 Mcnemar's Test P-Value : 0.63526

            Sensitivity : 0.7885
            Specificity : 0.6400
         Pos Pred Value : 0.8200
         Neg Pred Value : 0.5926
             Prevalence : 0.6753
         Detection Rate : 0.5325
   Detection Prevalence : 0.6494
      Balanced Accuracy : 0.7142

       'Positive' Class : 0
```

The accuracy for model 1 is **74%**.

Accuracy result for Model 2

Hide

```
confusionMatrix(table(key, prediction22))
```

```
Confusion Matrix and Statistics

   prediction22
key  0  1
  0 74 26
  1 24 30

               Accuracy : 0.6753
                 95% CI : (0.5953, 0.7485)
    No Information Rate : 0.6364
    P-Value [Acc > NIR] : 0.1787

                  Kappa : 0.2931

 Mcnemar's Test P-Value : 0.8875

            Sensitivity : 0.7551
            Specificity : 0.5357
         Pos Pred Value : 0.7400
         Neg Pred Value : 0.5556
             Prevalence : 0.6364
         Detection Rate : 0.4805
   Detection Prevalence : 0.6494
      Balanced Accuracy : 0.6454

       'Positive' Class : 0
```

The accuracy for model 2 is **68%**.

Accuracy result for Model 3

Hide

```
confusionMatrix(table(key, prediction33))
```

```
Confusion Matrix and Statistics

   prediction33
key  0  1
  0 78 22
  1 22 32

               Accuracy : 0.7143
                 95% CI : (0.636, 0.7841)
    No Information Rate : 0.6494
    P-Value [Acc > NIR] : 0.05265

                  Kappa : 0.3726

 Mcnemar's Test P-Value : 1.00000

            Sensitivity : 0.7800
            Specificity : 0.5926
         Pos Pred Value : 0.7800
         Neg Pred Value : 0.5926
             Prevalence : 0.6494
         Detection Rate : 0.5065
   Detection Prevalence : 0.6494
      Balanced Accuracy : 0.6863

       'Positive' Class : 0
```

The accuracy for model 3 is **71%**.

Accuracy result for Model 4

Hide

```
confusionMatrix(table(key, prediction44))
```

```
Confusion Matrix and Statistics

   prediction44
key  0  1
  0 67 33
  1 29 25

               Accuracy : 0.5974
                 95% CI : (0.5154, 0.6755)
    No Information Rate : 0.6234
    P-Value [Acc > NIR] : 0.7737

                  Kappa : 0.1307

 Mcnemar's Test P-Value : 0.7032

            Sensitivity : 0.6979
            Specificity : 0.4310
         Pos Pred Value : 0.6700
         Neg Pred Value : 0.4630
             Prevalence : 0.6234
         Detection Rate : 0.4351
   Detection Prevalence : 0.6494
      Balanced Accuracy : 0.5645

       'Positive' Class : 0
```

The accuracy for model 4 is **60%**.

Accuracy result for Model 5

Hide

```
confusionMatrix(table(key, prediction55))
```

```
Confusion Matrix and Statistics

   prediction55
key  0  1
  0 80 20
  1 18 36

               Accuracy : 0.7532
                 95% CI : (0.6774, 0.8191)
    No Information Rate : 0.6364
    P-Value [Acc > NIR] : 0.001317

                  Kappa : 0.4627

 Mcnemar's Test P-Value : 0.871131

            Sensitivity : 0.8163
            Specificity : 0.6429
         Pos Pred Value : 0.8000
         Neg Pred Value : 0.6667
             Prevalence : 0.6364
         Detection Rate : 0.5195
   Detection Prevalence : 0.6494
      Balanced Accuracy : 0.7296

       'Positive' Class : 0
```

The accuracy for model 5 is **75%** as the most accurate.

Accuracy result for Optimized Model

Hide

```
confusionMatrix(table(key, predictionopt))
```

```
Confusion Matrix and Statistics

   predictionopt
key  0  1
  0 84 16
  1 20 34

               Accuracy : 0.7662
                 95% CI : (0.6914, 0.8306)
    No Information Rate : 0.6753
    P-Value [Acc > NIR] : 0.008721

                  Kappa : 0.4778

 Mcnemar's Test P-Value : 0.617075

            Sensitivity : 0.8077
            Specificity : 0.6800
         Pos Pred Value : 0.8400
         Neg Pred Value : 0.6296
             Prevalence : 0.6753
         Detection Rate : 0.5455
   Detection Prevalence : 0.6494
      Balanced Accuracy : 0.7438

       'Positive' Class : 0
```

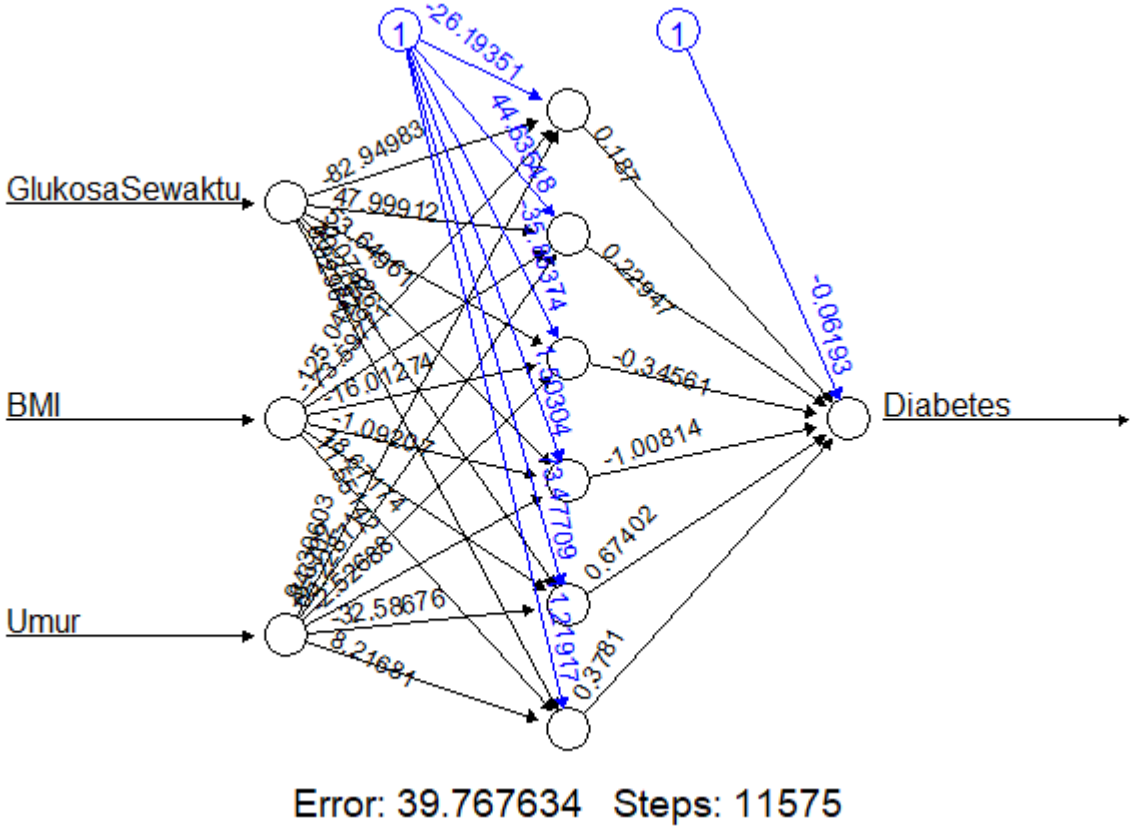
The accuracy for model 5 is **77%** as the most accurate even after removing the unnecessary variables.

5.2 Visualization approach

The following is the visualization for the machine learning model created through Neural Network approach. It utilizes 6 nodes with 1 hidden layer which taken 11575 steps called Optimized Model.

Hide

```
plot(modelopt)
```



6. Prediction/Forecasting

Optimized Model is the chosen model to be used in prediction because it has the highest accuracy. Following is the calculation to predict whether someone is diabetes or not by inputing my own data.

Hide

```
dp <- data.frame(GlukosaSewaktu = 80,
                 TekananDarahAtas = 80,
                 KetebalanKulit = 20,
                 Insulin = 100,
                 BMI = 23,
                 FaktorRisiko = 0.1,
                 Umur = 23,
                 Diabetes = 0)
df1 <- rbind(df, dp)
df2 <- as.data.frame(scale(df1))
df3 <- df2[nrow(df2), ]
```

Because the model based on standardized data, **the new data must be scaled first** before the calculation can be operated.

Hide

```
prediction111 <- as.data.frame(compute(modelopt, df3))
prediction1111 <- ifelse(prediction111$net.result >= 0.5, 1, 0)
print(prediction1111)
```

```
[1] 0
```

In conclusion, the prediction shows that **I do not have diabetes, since the 0 value means “Healthy”.**

7. Further Prescription Analysis/Recommendation

1. I hope this model can be utilized to assist medical practitioner to identify someone who might have diabetes much sooner by eliminating unnecessary variables that do not affect the accuracy that much and start focusing on those variables which have higher correlation which are **glucose level** as the most important factor, followed by BMI index and age variables in determining the possibility of having Diabetes.
2. For future reports, it will be beneficial to find out **what is the meaning of 0s in some columns** so we could gather more insights from the missing data and avoid distortion.
3. For future research, it will be helpful to ask the medical expert **what other relevant variables** against diabetes which should be added into the calculation to further boost the amount of accuracy as there is still much room to fill by around 20% to achieve more optimal accuracy.