CIS690\_GroupAssignment\_Team4

2023-06-15

## Install packages

Install required project packages.

if(!require(tidyverse)) install.packages("tidyverse")

## Loading required package: tidyverse

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.2 ✔ readr 2.1.4  
## ✔ forcats 1.0.0 ✔ stringr 1.5.0  
## ✔ ggplot2 3.4.2 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.2 ✔ tidyr 1.3.0  
## ✔ purrr 1.0.1   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

if(!require(gridExtra)) install.packages("gridExtra")

## Loading required package: gridExtra  
##   
## Attaching package: 'gridExtra'  
##   
## The following object is masked from 'package:dplyr':  
##   
## combine

if(!require(pheatmap)) install.packages("pheatmap")

## Loading required package: pheatmap

if(!require(carat)) install.packages("caret")

## Loading required package: carat

if(!require(pscl)) install.packages("pscl")

## Loading required package: pscl  
## Classes and Methods for R developed in the  
## Political Science Computational Laboratory  
## Department of Political Science  
## Stanford University  
## Simon Jackman  
## hurdle and zeroinfl functions by Achim Zeileis

if(!require(Metrics)) install.packages("Metrics")

## Loading required package: Metrics

if(!require(ggstats)) install.packages("ggstats")

## Loading required package: ggstats

if(!require(DescTools)) install.packages("DescTools")

## Loading required package: DescTools

# Packages required to install DMwR  
if(!require(zoo) | !require(xts) | !require(quantmod)) install.packages(c("zoo","xts","quantmod"))

## Loading required package: zoo  
##   
## Attaching package: 'zoo'  
##   
## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric  
##   
## Loading required package: xts  
##   
## ######################### Warning from 'xts' package ##########################  
## # #  
## # The dplyr lag() function breaks how base R's lag() function is supposed to #  
## # work, which breaks lag(my\_xts). Calls to lag(my\_xts) that you type or #  
## # source() into this session won't work correctly. #  
## # #  
## # Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #  
## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop #  
## # dplyr from breaking base R's lag() function. #  
## # #  
## # Code in packages is not affected. It's protected by R's namespace mechanism #  
## # Set `options(xts.warn\_dplyr\_breaks\_lag = FALSE)` to suppress this warning. #  
## # #  
## ###############################################################################  
##   
## Attaching package: 'xts'  
##   
## The following objects are masked from 'package:dplyr':  
##   
## first, last  
##   
## Loading required package: quantmod  
## Loading required package: TTR  
## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

if(!require(abind) | !require(ROCR)) install.packages(c("abind", "ROCR"))

## Loading required package: abind  
## Loading required package: ROCR

if(!require(DMwR)) install.packages( "https://cran.r-project.org/src/contrib/Archive/DMwR/DMwR\_0.4.1.tar.gz", repos=NULL, type="source" )

## Loading required package: DMwR  
## Loading required package: lattice  
## Loading required package: grid

## Import packages

Import required project packages.

#library import  
library(tidyverse)  
library(gridExtra)  
library(pheatmap)  
library(zoo)  
library(knitr)  
library(caret)

##   
## Attaching package: 'caret'

## The following objects are masked from 'package:DescTools':  
##   
## MAE, RMSE

## The following objects are masked from 'package:Metrics':  
##   
## precision, recall

## The following object is masked from 'package:purrr':  
##   
## lift

library(DMwR)  
library(pscl)  
library(Metrics)  
library(ggstats)  
library(DescTools)

# Importing Data

Before our project starts on building a GDM predictor, we first need to load the provided dataset. We can import the CSV file from our GitHub repository.

# Load the patient data from GitHub  
patient.data <- read.csv("https://raw.github.com/Neatherblok/Pregnancy\_Caused\_Diabetes\_Predictor/main/data/patients.csv")  
  
# Read first few rows of dataset  
head(patient.data)

## Pregnancies Glucose BloodPressure SkinThickness Insulin BMI Pedigree Age  
## 1 6 148 72 35 0 33.6 0.627 50  
## 2 1 85 66 29 0 26.6 0.351 31  
## 3 8 183 64 0 0 23.3 0.672 32  
## 4 1 89 66 23 94 28.1 0.167 21  
## 5 0 137 40 35 168 43.1 2.288 33  
## 6 5 116 74 0 0 25.6 0.201 30  
## Diagnosis  
## 1 1  
## 2 0  
## 3 1  
## 4 0  
## 5 1  
## 6 0

# Removing Duplicates

After loading and looking at the data set we checked if there were duplicate rows in the dataset. We did this by comparing the dimensions of the dataset, first the entire dataset and then what the dimensions were after picking the unique observations. It turned out that there were no duplicate observations.

# Data Dimension before deleting duplicates  
dim(patient.data)

## [1] 768 9

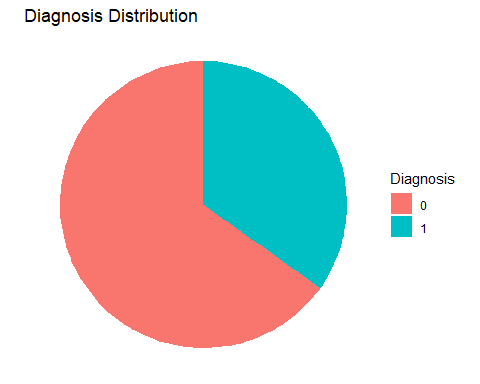
# Select only all unique observations  
unique.data<-unique(patient.data)  
  
# Data Dimension after deleting duplicates  
dim(unique.data)

## [1] 768 9

# Visualizing Dependent Variable

Now that we knew that there were no duplicates, we needed to be sure that the target is balanced. If the target variable is unbalanced the model can not perform as well because the model makes a split between the binary values in the middle and so needs to have about the same data of each feature. After looking at a pie chart of the two values in the **Diagnosis** variable, it seems that the False value appears the most.

# Target distribution pie chart  
ggplot(patient.data, aes(x = "", fill = factor(Diagnosis))) +  
 geom\_bar(width = 1) +  
 coord\_polar(theta = "y") +  
 labs(fill = "Diagnosis") +  
 ggtitle("Diagnosis Distribution") +  
 theme\_void()



# Balancing Dependent Variable

Since the value distribution of **Diagnosis** is not even, the dataset gets downsized on the “0” value. We did this by creating a subset where the dependent variable had a “0” value. Then, we removed the rows that also included 0s for the variables **Glucose**, **BloodPressure**, **SkinThickness**, **Insulin**, and **BMI**. After removing the rows that had both a 0 value in **Diagnosis** and one in the other independent variables (with exception of **Pregnancies**), we merged the subsets of Diagnosis = 0 and 1 back together.

# Create subset of Diagnosis = 0 observations  
no.diabetes <- patient.data[patient.data$Diagnosis == 0, ]  
  
# Finding the odd rows: Check for 0 values in specified columns and remove the rows  
filtered.data <- no.diabetes[!(no.diabetes$Glucose == 0 | no.diabetes$BloodPressure == 0 | no.diabetes$SkinThickness == 0 | no.diabetes$Insulin == 0 | no.diabetes$BMI == 0), ]  
  
# Combine the filtered rows with the remaining rows that have Diagnosis value not equal to 0  
balanced.patient.data <- rbind(filtered.data, patient.data[patient.data$Diagnosis != 0, ])  
  
# row index are reset  
row.names(balanced.patient.data) <- NULL  
  
# Show first few rows of data set  
head(balanced.patient.data)

## Pregnancies Glucose BloodPressure SkinThickness Insulin BMI Pedigree Age  
## 1 1 89 66 23 94 28.1 0.167 21  
## 2 1 103 30 38 83 43.3 0.183 33  
## 3 3 126 88 41 235 39.3 0.704 27  
## 4 1 97 66 15 140 23.2 0.487 22  
## 5 13 145 82 19 110 22.2 0.245 57  
## 6 3 88 58 11 54 24.8 0.267 22  
## Diagnosis  
## 1 0  
## 2 0  
## 3 0  
## 4 0  
## 5 0  
## 6 0

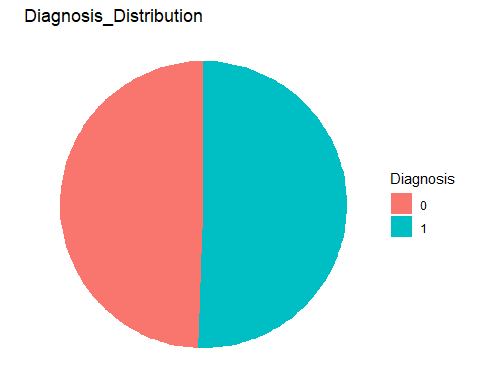
# Show last few rows of data set  
tail(balanced.patient.data)

## Pregnancies Glucose BloodPressure SkinThickness Insulin BMI Pedigree Age  
## 525 8 154 78 32 0 32.4 0.443 45  
## 526 1 128 88 39 110 36.5 1.057 37  
## 527 0 123 72 0 0 36.3 0.258 52  
## 528 6 190 92 0 0 35.5 0.278 66  
## 529 9 170 74 31 0 44.0 0.403 43  
## 530 1 126 60 0 0 30.1 0.349 47  
## Diagnosis  
## 525 1  
## 526 1  
## 527 1  
## 528 1  
## 529 1  
## 530 1

# Visualizing Balanced Dependent Variable

After balancing the target variable **Diagnosis** we can have another look at the pie chart and see that the distribution is now more similar. We also checked the actual number for each value of the target and found that the numbers are really close to each other and approximately follow a 50/50 distribution.

# Distribution of target variable post downscaling  
ggplot(balanced.patient.data, aes(x = "", fill = factor(Diagnosis))) +  
 geom\_bar(width = 1) +  
 coord\_polar(theta = "y") +  
 labs(fill = "Diagnosis") +  
 ggtitle("Diagnosis\_Distribution") +  
 theme\_void()



# Diagnosis distribution in numbers which follows approx 50%:50%  
diagnosis.counts <- table(balanced.patient.data$Diagnosis)  
print(diagnosis.counts)

##   
## 0 1   
## 262 268

# Descriptive Statistic

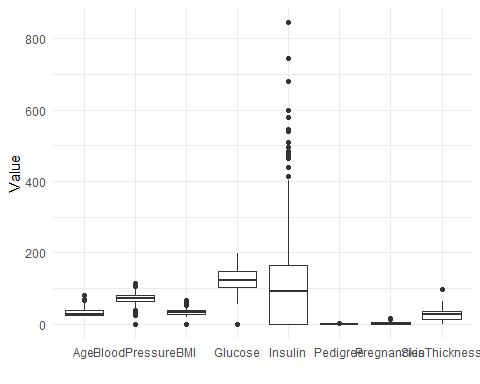
After balancing the data set, we can have a look at how the other independent variables behave. We can do this by summarizing these features and visualizing them in box plots and histograms. By looking at the summary, there are a few points that are popping out. Such as the minimum value for **Glucose, BloodPressure, SkinThickness, Insulin** and **BMI**. These are variables that can’t be having a minimum value of 0 and need to be treated missing values later on. Besides this, it looks like there also might be outliers in the data but that needs to be further explored.

# Summarize descriptive stats  
summary(balanced.patient.data[, 1:8])

## Pregnancies Glucose BloodPressure SkinThickness   
## Min. : 0.000 Min. : 0.0 Min. : 0.00 Min. : 0.00   
## 1st Qu.: 1.000 1st Qu.:102.0 1st Qu.: 64.00 1st Qu.:15.00   
## Median : 3.000 Median :123.0 Median : 72.00 Median :27.00   
## Mean : 3.806 Mean :126.5 Mean : 69.91 Mean :24.68   
## 3rd Qu.: 6.000 3rd Qu.:147.8 3rd Qu.: 80.00 3rd Qu.:35.00   
## Max. :17.000 Max. :199.0 Max. :114.00 Max. :99.00   
## Insulin BMI Pedigree Age   
## Min. : 0.0 Min. : 0.00 Min. :0.0850 Min. :21.00   
## 1st Qu.: 0.0 1st Qu.:28.90 1st Qu.:0.2610 1st Qu.:24.00   
## Median : 92.0 Median :33.20 Median :0.4300 Median :29.00   
## Mean :115.4 Mean :33.47 Mean :0.5118 Mean :32.76   
## 3rd Qu.:165.8 3rd Qu.:37.67 3rd Qu.:0.6777 3rd Qu.:40.00   
## Max. :846.0 Max. :67.10 Max. :2.4200 Max. :81.00

By looking at the box plot, we see that there are quite some outliers in the data that appear above and under the whiskers. These will have to be addressed.

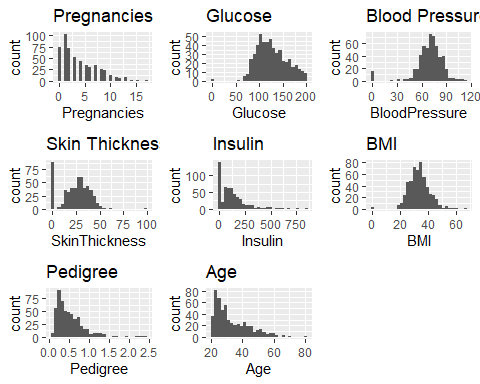
# Box Plot  
independent.features <- balanced.patient.data[, 1:8]  
long\_data <- pivot\_longer(independent.features, everything(), names\_to = "variable", values\_to = "value")  
ggplot(long\_data, aes(x = variable, y = value)) +  
 geom\_boxplot() +  
 xlab("") +  
 ylab("Value") +  
 theme\_minimal()



By looking in the histograms we can say that except **BloodPressure** and **BMI**, non of the other variables follow a normal distribution. But a normal distribution is not assumed by a logistic regression, so not a priority for us.

# Histogram  
Pregnancies <- ggplot(independent.features, aes(x = Pregnancies)) + geom\_histogram() + ggtitle("Pregnancies")  
Glucose <- ggplot(independent.features, aes(x = Glucose)) + geom\_histogram() + ggtitle("Glucose")  
BloodPressure <- ggplot(independent.features, aes(x = BloodPressure)) + geom\_histogram() + ggtitle("Blood Pressure")  
SkinThickness <- ggplot(independent.features, aes(x = SkinThickness)) + geom\_histogram() + ggtitle("Skin Thickness")  
Insulin <- ggplot(independent.features, aes(x = Insulin)) + geom\_histogram() + ggtitle("Insulin")  
BMI <- ggplot(independent.features, aes(x = BMI)) + geom\_histogram() + ggtitle("BMI")  
Pedigree <- ggplot(independent.features, aes(x = Pedigree)) + geom\_histogram() + ggtitle("Pedigree")  
Age <- ggplot(independent.features, aes(x = Age)) + geom\_histogram() + ggtitle("Age")  
grid.arrange(  
 Pregnancies, Glucose, BloodPressure, SkinThickness,  
 Insulin, BMI, Pedigree, Age,  
 nrow = 3  
)

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



## Missing Data Handling

As pointed out, there are still some variables that contain 0 values where this does not make sense. The only independent variable where this would make sense for is **Pregnancies** as it is possible to get diabetes without ever having been pregnant. We will be treated the 0 values for all other independent variables as missing values and be replacing them with the median. We chose for the median as the mean could be heavily influenced by the outliers that will be treated later.

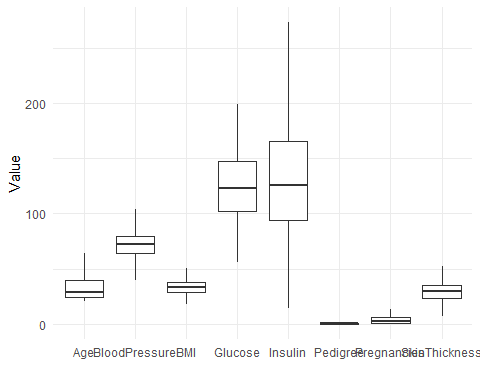
# All zeros are treated as missing values and replaced with median  
for (i in 2:8) {  
 balanced.patient.data[, i] <- ifelse(balanced.patient.data[, i] == 0, median(balanced.patient.data[balanced.patient.data[, i] != 0, i], na.rm = TRUE), balanced.patient.data[, i])  
}  
  
# Change name of data set  
cleaned.patient.data <- balanced.patient.data  
  
# Read First Few rows to see if 0's were replaced by median of the column  
head(cleaned.patient.data)

## Pregnancies Glucose BloodPressure SkinThickness Insulin BMI Pedigree Age  
## 1 1 89 66 23 94 28.1 0.167 21  
## 2 1 103 30 38 83 43.3 0.183 33  
## 3 3 126 88 41 235 39.3 0.704 27  
## 4 1 97 66 15 140 23.2 0.487 22  
## 5 13 145 82 19 110 22.2 0.245 57  
## 6 3 88 58 11 54 24.8 0.267 22  
## Diagnosis  
## 1 0  
## 2 0  
## 3 0  
## 4 0  
## 5 0  
## 6 0

## Removing Outliers

After all missing values are treated by replacing with median, we can replace the outliers. We will use the interquartile range to determine outliers and replace values outside of the 1.5xIQR with the value that is at 1.5xIQR. We will do this for all independent variables. We found out that this was a better method, because we would still have variables if we would have replaced all outliers with the median as the IQR would become smaller. After capping we can once again built a box plot and in it we see that there are no outliers.

# Removing outliers using capping method  
treating.outliers <- function(x) {  
 qntls <- quantile(x, probs = c(0.25, 0.75))  
 iqr <- qntls[2] - qntls[1]  
 lb <- qntls[1] - 1.5 \* iqr  
 ub <- qntls[2] + 1.5 \* iqr  
 x[x < lb] <- lb  
 x[x > ub] <- ub  
 return(x)  
}  
  
# Execute function for all independent variables  
cleaned.patient.data[, 1:8] <- apply(cleaned.patient.data[, 1:8], 2, treating.outliers)  
  
# By the revised box plot you can see that the outliers are treated and not present in the data that we are going to further analyse  
cleaned.independent.variables <- cleaned.patient.data[, 1:8]  
long\_data <- pivot\_longer(cleaned.independent.variables, everything(), names\_to = "variable", values\_to = "value")  
ggplot(long\_data, aes(x = variable, y = value)) +  
 geom\_boxplot() +  
 xlab("") +  
 ylab("Value") +  
 theme\_minimal()



We can now have a look at how the data set looks like after it has been cleaned.

# check summary post data cleaning  
summary(cleaned.patient.data)

## Pregnancies Glucose BloodPressure SkinThickness   
## Min. : 0.000 Min. : 56.0 Min. : 40.0 Min. : 7.00   
## 1st Qu.: 1.000 1st Qu.:102.0 1st Qu.: 64.0 1st Qu.:23.25   
## Median : 3.000 Median :123.0 Median : 72.0 Median :30.00   
## Mean : 3.794 Mean :127.0 Mean : 72.1 Mean :29.53   
## 3rd Qu.: 6.000 3rd Qu.:147.8 3rd Qu.: 80.0 3rd Qu.:35.00   
## Max. :13.500 Max. :199.0 Max. :104.0 Max. :52.62   
## Insulin BMI Pedigree Age   
## Min. : 14.0 Min. :18.20 Min. :0.0850 Min. :21.0   
## 1st Qu.: 94.0 1st Qu.:28.90 1st Qu.:0.2610 1st Qu.:24.0   
## Median :125.5 Median :33.25 Median :0.4300 Median :29.0   
## Mean :135.9 Mean :33.51 Mean :0.5010 Mean :32.7   
## 3rd Qu.:165.8 3rd Qu.:37.67 3rd Qu.:0.6777 3rd Qu.:40.0   
## Max. :273.4 Max. :50.84 Max. :1.3029 Max. :64.0   
## Diagnosis   
## Min. :0.0000   
## 1st Qu.:0.0000   
## Median :1.0000   
## Mean :0.5057   
## 3rd Qu.:1.0000   
## Max. :1.0000

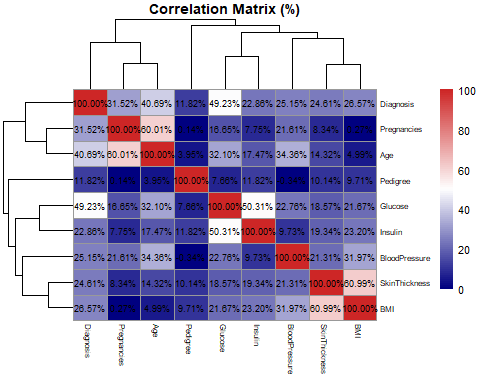
## Correlation Check

We can now start thinking about building the models. We would like to select the best features to train some of the models on, while also training models on all features. With a correlogram we can select the best features, but we can also look for correlations between the independent variables. When looking at the correlation of the independent variables, four variables look the strongest. Those four variables are:

* Glucose
* Age
* Pregnancies
* BMI

We will be training models on these features. But it also looks like independent variables **Age** and **Pregnancies** are correlated, so we will also train a model without age to see how this influences the model in comparison to if it was kept.

# Check for correlation post data cleaning  
Correlation.Mtrx <- cor(cleaned.patient.data)  
Correlation.Percentages <- round(Correlation.Mtrx \* 100, 2)  
pheatmap(Correlation.Percentages,  
 main = "Correlation Matrix (%)",  
 fontsize = 8,  
 fontsize\_row = 6,  
 fontsize\_col = 6,  
 color = colorRampPalette(c("navy", "white", "firebrick3"))(100),  
 show\_rownames = TRUE,  
 show\_colnames = TRUE,  
 display\_numbers = TRUE,  
 number\_color = "black",  
 number\_format = "%.2f%%")



## Splitting Dataset 1

After selecting the best features, we can now split the dataset in a train and test set through a random 80/20 split. We decided to choose for a 80/20 split as we would like to have as much data to train the model on so the model can generalize as much as possible, while also still remaining a good number of observations to test our model on. We will be splitting the data set on random generator seed *130*, to maintain reproducibility.

# pre-set properties for random split   
train.size = floor(0.8\*nrow(cleaned.patient.data))  
set.seed(130)  
  
# randomly split data in r according to generator seed 130  
picked = sample(seq\_len(nrow(cleaned.patient.data)),size = train.size)  
  
# Divide the data set up in two data set according to picked sample  
train.data.not.normal = cleaned.patient.data[picked,]  
test.data.not.normal = cleaned.patient.data[-picked,]  
  
# Reset row index  
rownames(train.data.not.normal) <- NULL  
rownames(test.data.not.normal) <- NULL  
  
# Show the two split data sets  
head(train.data.not.normal)

## Pregnancies Glucose BloodPressure SkinThickness Insulin BMI Pedigree Age  
## 1 4 91 70 32 88.0 33.1 0.446 22  
## 2 7 195 70 33 145.0 25.1 0.163 55  
## 3 5 139 80 35 160.0 31.6 0.361 25  
## 4 6 125 78 31 125.5 27.6 0.565 49  
## 5 4 144 82 32 125.5 38.5 0.554 37  
## 6 2 88 74 19 53.0 29.0 0.229 22  
## Diagnosis  
## 1 0  
## 2 1  
## 3 1  
## 4 1  
## 5 1  
## 6 0

head(test.data.not.normal)

## Pregnancies Glucose BloodPressure SkinThickness Insulin BMI Pedigree Age  
## 1 7 150 66 42 273.375 34.7 0.718 42  
## 2 2 110 74 29 125.000 32.4 0.698 27  
## 3 4 123 80 15 176.000 32.0 0.443 34  
## 4 0 100 70 26 50.000 30.8 0.597 21  
## 5 1 136 74 50 204.000 37.4 0.399 24  
## 6 2 75 64 24 55.000 29.7 0.370 33  
## Diagnosis  
## 1 0  
## 2 0  
## 3 0  
## 4 0  
## 5 0  
## 6 0

## Normalizing Dataset

Even though there is not a normalization for the logistic regression model, we would like to see how normalization influences the performance of the model. We will do this through a min-max scale. This means that all values will be divided by their max, while the min is put to 0, to get all values between 0 and 1. We use normalization to make variables have a similar scale, which improves training stability and performance.

# Set up normalization function  
process <- preProcess(cleaned.patient.data, method=c("range"))  
  
# Execute Min-Max normalization on the cleaned data set  
norm.data <- predict(process, cleaned.patient.data)  
  
# Show normalized data set  
head(norm.data)

## Pregnancies Glucose BloodPressure SkinThickness Insulin BMI  
## 1 0.07407407 0.2307692 0.40625 0.35068493 0.3084337 0.3033321  
## 2 0.07407407 0.3286713 0.00000 0.67945205 0.2660241 0.7690540  
## 3 0.22222222 0.4895105 0.75000 0.74520548 0.8520482 0.6464956  
## 4 0.07407407 0.2867133 0.40625 0.17534247 0.4857831 0.1531980  
## 5 0.96296296 0.6223776 0.65625 0.26301370 0.3701205 0.1225584  
## 6 0.22222222 0.2237762 0.28125 0.08767123 0.1542169 0.2022214  
## Pedigree Age Diagnosis  
## 1 0.06733039 0.00000000 0  
## 2 0.08046803 0.27906977 0  
## 3 0.50826234 0.13953488 0  
## 4 0.33008314 0.02325581 0  
## 5 0.13137637 0.83720930 0  
## 6 0.14944062 0.02325581 0

## Splitting Dataset 2

After normalizing the dataset, we can once again split the dataset in a train and test set through a random 80/20 split, with random generator seed *130*.

# Divide the data set up in two data set according to picked sample  
train.data.normal = norm.data[picked,]  
test.data.normal = norm.data[-picked,]  
  
# Reset row index  
rownames(train.data.normal) <- NULL  
rownames(test.data.normal) <- NULL  
  
# Show the two split data sets  
head(train.data.normal)

## Pregnancies Glucose BloodPressure SkinThickness Insulin BMI  
## 1 0.2962963 0.2447552 0.46875 0.5479452 0.2853012 0.4565301  
## 2 0.5185185 0.9720280 0.46875 0.5698630 0.5050602 0.2114133  
## 3 0.3703704 0.5804196 0.62500 0.6136986 0.5628916 0.4105707  
## 4 0.4444444 0.4825175 0.59375 0.5260274 0.4298795 0.2880123  
## 5 0.2962963 0.6153846 0.65625 0.5479452 0.4298795 0.6219839  
## 6 0.1481481 0.2237762 0.53125 0.2630137 0.1503614 0.3309077  
## Pedigree Age Diagnosis  
## 1 0.29641794 0.02325581 0  
## 2 0.06404598 0.79069767 1  
## 3 0.22662424 0.09302326 1  
## 4 0.39412912 0.65116279 1  
## 5 0.38509699 0.37209302 1  
## 6 0.11823874 0.02325581 0

head(test.data.normal)

## Pregnancies Glucose BloodPressure SkinThickness Insulin BMI  
## 1 0.51851852 0.6573427 0.40625 0.7671233 1.0000000 0.5055534  
## 2 0.14814815 0.3776224 0.53125 0.4821918 0.4279518 0.4350823  
## 3 0.29629630 0.4685315 0.62500 0.1753425 0.6245783 0.4228265  
## 4 0.00000000 0.3076923 0.46875 0.4164384 0.1387952 0.3860590  
## 5 0.07407407 0.5594406 0.53125 0.9424658 0.7325301 0.5882804  
## 6 0.14814815 0.1328671 0.37500 0.3726027 0.1580723 0.3523554  
## Pedigree Age Diagnosis  
## 1 0.5197578 0.48837209 0  
## 2 0.5033357 0.13953488 0  
## 3 0.2939546 0.30232558 0  
## 4 0.4204044 0.00000000 0  
## 5 0.2578261 0.06976744 0  
## 6 0.2340142 0.27906977 0

## Train Logistic Regression Model 1

After splitting, the models can get trained. The first model will be trained on the four most correlated variables of the normalized data set. As dependent variable we will be using Diagnosis. This model trains with family being binomial, with a logit link. The family being binomial means that the output of the model and the dependent variable will be binary value. The logit link specifies that this binary classification is a logistic regression, and it maps the probability of success to the linear predictor.

# Train Logistic Regression model 2  
model.1 <- glm(Diagnosis ~ Pregnancies + Age + Glucose + BMI, data = train.data.normal, family = binomial(link='logit'))

Now that the model has been trained, we can get a summary of what the findings of the model are on the original feature set. It shows which independent features have the most value for the eventual prediction of the dependent variable. The variables with the lowest p-value and most amount of stars have the most influence on the dependent variable.

# Summarize the findings of the first Logistic Regression model  
summary(model.1)

##   
## Call:  
## glm(formula = Diagnosis ~ Pregnancies + Age + Glucose + BMI,   
## family = binomial(link = "logit"), data = train.data.normal)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -4.5109 0.4858 -9.285 < 2e-16 \*\*\*  
## Pregnancies 1.4089 0.6571 2.144 0.032030 \*   
## Age 2.5568 0.6699 3.817 0.000135 \*\*\*  
## Glucose 4.5491 0.6511 6.987 2.8e-12 \*\*\*  
## BMI 2.5043 0.6425 3.898 9.7e-05 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 587.45 on 423 degrees of freedom  
## Residual deviance: 412.68 on 419 degrees of freedom  
## AIC: 422.68  
##   
## Number of Fisher Scoring iterations: 4

Now that the first Logistic Regression model has been trained on the best features normalized set, it can predict the values that are in the normalized test set. It will do this through the response type which will give back a probability for the tested observation between 0 and 1. After all the observations are fitted, the predictions get converted to a binary value depending on the probability score. Values with a probability of 0.5 or higher gets converted to a 1 value, values under 0.5 get converted to the 0 value.

# Predict the outcome using Logistic Regression Model 1  
mdl.1.fitted.results <- predict(model.1, test.data.normal[,c("Pregnancies", "Age", "Glucose", "BMI")], type='response')  
  
# Transform predicted probability values with a probability higher than 0.5 to 1  
# Transform values with probability lower than 0.5 to 0  
predictions.mdl.1 <- ifelse(mdl.1.fitted.results >= 0.5, 1, 0)

After the logistic regression model predicted the values from the test set based on the best feature, a classification report can be generated of the performance of the model predictions, with the confusionMatrix() function from the caret package.

# Create a classification report for Logistic Regression Model 1  
# This includes a confusion matrix, accuracy score, ACI score, confidence interval and other scores  
classification.report.mdl.1 <- caret::confusionMatrix(factor(predictions.mdl.1), factor(test.data.normal$Diagnosis))  
print(classification.report.mdl.1)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 36 15  
## 1 8 47  
##   
## Accuracy : 0.783   
## 95% CI : (0.6924, 0.8572)  
## No Information Rate : 0.5849   
## P-Value [Acc > NIR] : 1.379e-05   
##   
## Kappa : 0.5632   
##   
## Mcnemar's Test P-Value : 0.2109   
##   
## Sensitivity : 0.8182   
## Specificity : 0.7581   
## Pos Pred Value : 0.7059   
## Neg Pred Value : 0.8545   
## Prevalence : 0.4151   
## Detection Rate : 0.3396   
## Detection Prevalence : 0.4811   
## Balanced Accuracy : 0.7881   
##   
## 'Positive' Class : 0   
##

## Train Logistic Regression Model 2

After training Logistic Regression Model 1, we will try out what the difference is with a model that is normalized and still has all features. This model will be trained on all the features from the normalized train data set, with Diagnosis as the dependent variable and tested on the normalized test data with all features in the same way as Logistic Regression Model 1 is trained and tested.

# Train Logistic Regression model 2  
model.2 <- glm(Diagnosis ~.,family=binomial(link='logit'),data=train.data.normal)

# Summarize the findings of the second Logistic Regression model  
summary(model.2)

##   
## Call:  
## glm(formula = Diagnosis ~ ., family = binomial(link = "logit"),   
## data = train.data.normal)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -4.9943 0.5917 -8.441 < 2e-16 \*\*\*  
## Pregnancies 1.3861 0.6687 2.073 0.038200 \*   
## Glucose 5.0748 0.7575 6.699 2.09e-11 \*\*\*  
## BloodPressure -0.1538 0.7217 -0.213 0.831186   
## SkinThickness 1.1133 0.7684 1.449 0.147384   
## Insulin -0.8419 0.6025 -1.397 0.162335   
## BMI 1.9891 0.8242 2.413 0.015807 \*   
## Pedigree 1.1944 0.5146 2.321 0.020289 \*   
## Age 2.4931 0.7015 3.554 0.000379 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 587.45 on 423 degrees of freedom  
## Residual deviance: 403.09 on 415 degrees of freedom  
## AIC: 421.09  
##   
## Number of Fisher Scoring iterations: 5

# Predict the outcome using Logistic Regression Model 2  
mdl.2.fitted.results <- predict(model.2, test.data.normal, type='response')  
  
# Transform predicted probability values with a probability higher than 0.5 to 1  
# Transform values with probability lower than 0.5 to 0  
predictions.mdl.2 <- ifelse(mdl.2.fitted.results >= 0.5, 1, 0)

# Create a classification report  
# This includes a confusion matrix, accuracy score, ACI score, confidence interval and other scores  
classification.report.mdl.2 <- caret::confusionMatrix(factor(predictions.mdl.2), factor(test.data.normal$Diagnosis))  
print(classification.report.mdl.2)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 36 19  
## 1 8 43  
##   
## Accuracy : 0.7453   
## 95% CI : (0.6514, 0.8249)  
## No Information Rate : 0.5849   
## P-Value [Acc > NIR] : 0.0004227   
##   
## Kappa : 0.4938   
##   
## Mcnemar's Test P-Value : 0.0542918   
##   
## Sensitivity : 0.8182   
## Specificity : 0.6935   
## Pos Pred Value : 0.6545   
## Neg Pred Value : 0.8431   
## Prevalence : 0.4151   
## Detection Rate : 0.3396   
## Detection Prevalence : 0.5189   
## Balanced Accuracy : 0.7559   
##   
## 'Positive' Class : 0   
##

## Train Logistic Regression Model 3

After training Logistic Regression Model 1 and 2, we are trying out to see what the difference is with a model that is not normalized and has its best features selected. This model will be trained on the non scaled train data set, with Diagnosis as the dependent variable and tested on the non-scaled test data with the best features in the same way as Logistic Regression Model 1 is trained and tested.

# Train Logistic Regression model 3  
model.3 <- glm(Diagnosis ~Pregnancies + Age + Glucose + BMI, data = train.data.not.normal, family = binomial(link='logit'))

# Summarize the findings of the third Logistic Regression model  
summary(model.3)

##   
## Call:  
## glm(formula = Diagnosis ~ Pregnancies + Age + Glucose + BMI,   
## family = binomial(link = "logit"), data = train.data.not.normal)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -8.937519 0.956942 -9.340 < 2e-16 \*\*\*  
## Pregnancies 0.104361 0.048675 2.144 0.032030 \*   
## Age 0.059460 0.015579 3.817 0.000135 \*\*\*  
## Glucose 0.031812 0.004553 6.987 2.8e-12 \*\*\*  
## BMI 0.076729 0.019684 3.898 9.7e-05 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 587.45 on 423 degrees of freedom  
## Residual deviance: 412.68 on 419 degrees of freedom  
## AIC: 422.68  
##   
## Number of Fisher Scoring iterations: 4

# Predict the outcome using Logistic Regression Model 3  
mdl.3.fitted.results <- predict(model.3, test.data.not.normal[,c("Pregnancies", "Age", "Glucose", "BMI")], type='response')  
  
# Transform predicted probability values with a probability higher than 0.5 to 1  
# Transform values with probability lower than 0.5 to 0  
predictions.mdl.3 <- ifelse(mdl.3.fitted.results >= 0.5, 1, 0)

# Create a classification report for Logistic Regression Model 3  
# This includes a confusion matrix, accuracy score, ACI score, confidence interval and other scores  
classification.report.mdl.3 <- caret::confusionMatrix(factor(predictions.mdl.3), factor(test.data.not.normal$Diagnosis))  
print(classification.report.mdl.3)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 36 15  
## 1 8 47  
##   
## Accuracy : 0.783   
## 95% CI : (0.6924, 0.8572)  
## No Information Rate : 0.5849   
## P-Value [Acc > NIR] : 1.379e-05   
##   
## Kappa : 0.5632   
##   
## Mcnemar's Test P-Value : 0.2109   
##   
## Sensitivity : 0.8182   
## Specificity : 0.7581   
## Pos Pred Value : 0.7059   
## Neg Pred Value : 0.8545   
## Prevalence : 0.4151   
## Detection Rate : 0.3396   
## Detection Prevalence : 0.4811   
## Balanced Accuracy : 0.7881   
##   
## 'Positive' Class : 0   
##

## Train Logistic Regression Model 4

After training Logistic Regression model 3, we also want to know like model 2 how this model trained on the dataset with all features responds to the model trained on the best features. Just like model 3, we will be doing this on the non-scaled dataset. This model will be trained on the non scaled train data set, with Diagnosis as the dependent variable and tested on the non-scaled test data with the all features in the same way as Logistic Regression Model 2 is trained and tested.

# Train Logistic Regression model 4  
model.4 <- glm(Diagnosis ~ ., data = train.data.not.normal, family = binomial(link='logit'))

# Summarize the findings of the fourth Logistic Regression model  
summary(model.4)

##   
## Call:  
## glm(formula = Diagnosis ~ ., family = binomial(link = "logit"),   
## data = train.data.not.normal)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -9.420950 1.100695 -8.559 < 2e-16 \*\*\*  
## Pregnancies 0.102671 0.049535 2.073 0.038200 \*   
## Glucose 0.035488 0.005297 6.699 2.09e-11 \*\*\*  
## BloodPressure -0.002404 0.011276 -0.213 0.831186   
## SkinThickness 0.024401 0.016842 1.449 0.147384   
## Insulin -0.003246 0.002323 -1.397 0.162335   
## BMI 0.060945 0.025253 2.413 0.015807 \*   
## Pedigree 0.980688 0.422535 2.321 0.020289 \*   
## Age 0.057980 0.016314 3.554 0.000379 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 587.45 on 423 degrees of freedom  
## Residual deviance: 403.09 on 415 degrees of freedom  
## AIC: 421.09  
##   
## Number of Fisher Scoring iterations: 5

# Predict the outcome using Logistic Regression Model 4  
mdl.4.fitted.results <- predict(model.4, test.data.not.normal,type='response')  
  
# Transform predicted probability values with a probability higher than 0.5 to 1  
# Transform values with probability lower than 0.5 to 0  
predictions.mdl.4 <- ifelse(mdl.4.fitted.results >= 0.5, 1, 0)

# Create a classification report for Logistic Regression Model 4  
# This includes a confusion matrix, accuracy score, ACI score, confidence interval and other scores  
classification.report.mdl.4 <- caret::confusionMatrix(factor(predictions.mdl.4), factor(test.data.not.normal$Diagnosis))  
print(classification.report.mdl.4)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 36 19  
## 1 8 43  
##   
## Accuracy : 0.7453   
## 95% CI : (0.6514, 0.8249)  
## No Information Rate : 0.5849   
## P-Value [Acc > NIR] : 0.0004227   
##   
## Kappa : 0.4938   
##   
## Mcnemar's Test P-Value : 0.0542918   
##   
## Sensitivity : 0.8182   
## Specificity : 0.6935   
## Pos Pred Value : 0.6545   
## Neg Pred Value : 0.8431   
## Prevalence : 0.4151   
## Detection Rate : 0.3396   
## Detection Prevalence : 0.5189   
## Balanced Accuracy : 0.7559   
##   
## 'Positive' Class : 0   
##

## Train Logistic Regression Model 5

After training Logistic Regression model 4, we also want to know how the model performs when it is trained on the best features but without **Age** as this variable is correlated with **Pregnancies** as well. This model will be trained on the scaled train data set.

# Train Logistic Regression model 5  
model.5 <- glm(Diagnosis ~ Pregnancies + Glucose + BMI, data = train.data.normal, family = binomial(link='logit'))

# Summarize the findings of the fifth Logistic Regression model  
summary(model.5)

##   
## Call:  
## glm(formula = Diagnosis ~ Pregnancies + Glucose + BMI, family = binomial(link = "logit"),   
## data = train.data.normal)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -4.4723 0.4759 -9.398 < 2e-16 \*\*\*  
## Pregnancies 2.8607 0.5480 5.220 1.79e-07 \*\*\*  
## Glucose 4.9681 0.6319 7.862 3.78e-15 \*\*\*  
## BMI 2.5893 0.6330 4.091 4.30e-05 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 587.45 on 423 degrees of freedom  
## Residual deviance: 428.51 on 420 degrees of freedom  
## AIC: 436.51  
##   
## Number of Fisher Scoring iterations: 4

# Predict the outcome using Logistic Regression Model 5  
mdl.5.fitted.results <- predict(model.5, test.data.normal,type='response')  
  
# Transform predicted probability values with a probability higher than 0.5 to 1  
# Transform values with probability lower than 0.5 to 0  
predictions.mdl.5 <- ifelse(mdl.5.fitted.results >= 0.5, 1, 0)

# Create a classification report for Logistic Regression Model 5  
# This includes a confusion matrix, accuracy score, ACI score, confidence interval and other scores  
classification.report.mdl.5 <- caret::confusionMatrix(factor(predictions.mdl.5), factor(test.data.normal$Diagnosis))  
print(classification.report.mdl.5)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 32 14  
## 1 12 48  
##   
## Accuracy : 0.7547   
## 95% CI : (0.6616, 0.8331)  
## No Information Rate : 0.5849   
## P-Value [Acc > NIR] : 0.0001934   
##   
## Kappa : 0.4982   
##   
## Mcnemar's Test P-Value : 0.8445193   
##   
## Sensitivity : 0.7273   
## Specificity : 0.7742   
## Pos Pred Value : 0.6957   
## Neg Pred Value : 0.8000   
## Prevalence : 0.4151   
## Detection Rate : 0.3019   
## Detection Prevalence : 0.4340   
## Balanced Accuracy : 0.7507   
##   
## 'Positive' Class : 0   
##

### Comparing the models

After all models have been trained, we are able to compare the models based on the classification report and accuracy scores that have been generated based on the test sets. In the following code, all accuracy, AIC and confidence interval scores are put beside each other. What stands out in this comparison is the following:

* Both models that are trained on the best features dataset are performing the same
* Both models that are trained on all features perform the same as well

This means that normalization of the data does not influence the result of a logistic regression model.

Other findings are:

* The models trained on the best features have a higher accuracy score than the models trained on all features
* The models trained on all features have a lower AIC score and smaller Confidence Interval ratio.
* The model trained on the best features but without **Age** has a slightly lower accuracy than the best features model.
* The model trained on the best features but without **Age** has a lot worse AIC score compared to the other models.

This means that it is not clear which model actually performs the best. Normally AIC penalizes the model with more features heavier, but in this case this did not happen. This means that the models with all features might actually be a better predictor, which needs to be researched through a Likeliness Ratio Hypothesis test.

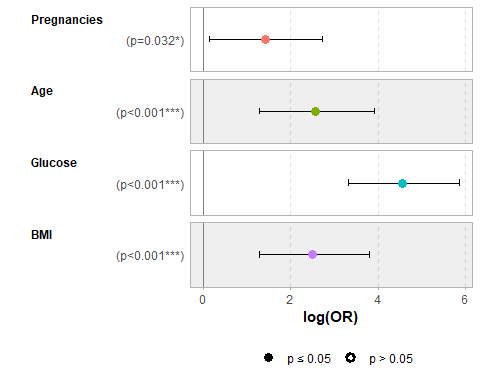
df <- data.frame(Metrics=c("Accuracy", "AIC", "Confidence Interval Lower", "Confidence Interval Higher"),  
 select\_feature\_normal=c(classification.report.mdl.1$overall[1], AIC(model.1),  
 classification.report.mdl.1$overall[3], classification.report.mdl.1$overall[4]),  
   
 all\_normal=c(classification.report.mdl.2$overall[1], AIC(model.2),   
 classification.report.mdl.2$overall[3], classification.report.mdl.2$overall[4]),  
   
 select\_feature\_not\_normal=c(classification.report.mdl.3$overall[1], AIC(model.3),  
 classification.report.mdl.3$overall[3], classification.report.mdl.3$overall[4]),  
   
 all\_not\_normal=c(classification.report.mdl.4$overall[1], AIC(model.4),   
 classification.report.mdl.4$overall[3], classification.report.mdl.4$overall[4]),  
   
 without\_age\_normal=c(classification.report.mdl.5$overall[1], AIC(model.5),   
 classification.report.mdl.5$overall[3], classification.report.mdl.5$overall[4]))  
  
# Set metrics column as index  
rownames(df) <- df$Metrics  
  
# Remove original metrics column from data frame  
df$Metrics <- NULL  
  
# Show results  
df

## select\_feature\_normal all\_normal  
## Accuracy 0.7830189 0.7452830  
## AIC 422.6791460 421.0912215  
## Confidence Interval Lower 0.6924240 0.6514466  
## Confidence Interval Higher 0.8572035 0.8249489  
## select\_feature\_not\_normal all\_not\_normal  
## Accuracy 0.7830189 0.7452830  
## AIC 422.6791460 421.0912215  
## Confidence Interval Lower 0.6924240 0.6514466  
## Confidence Interval Higher 0.8572035 0.8249489  
## without\_age\_normal  
## Accuracy 0.7547170  
## AIC 436.5115330  
## Confidence Interval Lower 0.6616128  
## Confidence Interval Higher 0.8330936

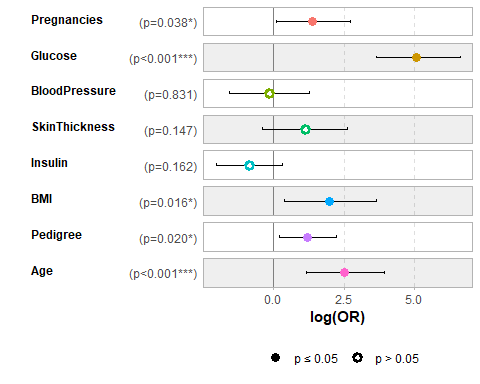
### Visualizing Coefficients

Underneath is the relationship between the coefficients and the dependent variable *Diabetes* visualized. This visualization shows how each coefficient for each model relate to the dependent variable. Where the left means a low logarithmic connection and to the right means a stronger relationship. The visualization also shows the p-value on if the relationship between the coefficient and dependent variable is significant. If the p-value is under 0.05, the dot in the middle is entirely colored in, otherwise its white in the middle. The visualization also shows whiskers for each plot. These whiskers represent the uncertainty of the variance for the coefficient. If whiskers are small, it means there is a small uncertainty of variance meaning that the standard deviation of its residuals is small.

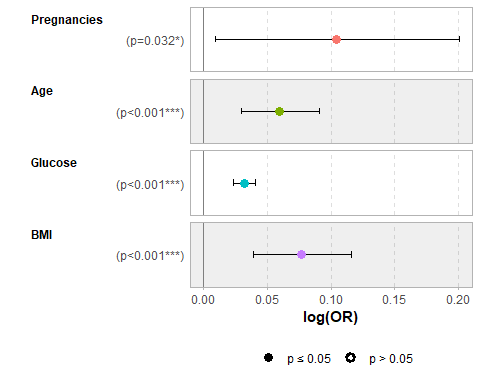
# Visualize logistic regression coefficients  
ggcoef\_model(model.1)



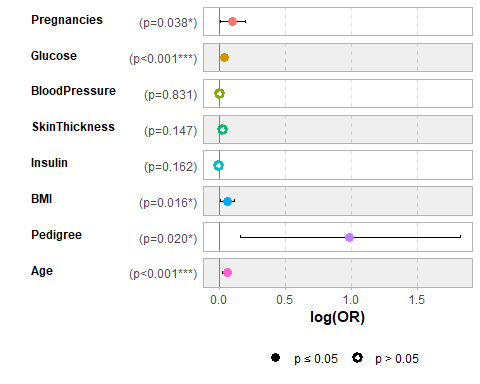
ggcoef\_model(model.2)



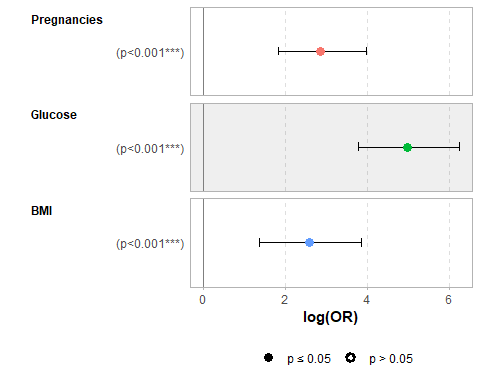
ggcoef\_model(model.3)



ggcoef\_model(model.4)



ggcoef\_model(model.5)



### Hypothesis testing

# Number of times one anova comparison has been run  
anova.runs <- list(  
 anova.1 = 0,  
 anova.2 = 0,  
 anova.3 = 0,  
 anova.4 = 0,  
 anova.5 = 0  
)

All models are trained, and compared by Accuracy, AIC and confidence interval. This has resulted in a few findings and predictions that would need to be tested through a hypothesis. To compare logistic regression models, we are going to be using a Likeliness Ratio test, that tests the maximum likelihood estimates of coefficients to each other where it determines if it needs to reject a restriction of a coefficient or not. It seems like Model 1 and 3 are both equally significant and there is no difference. The same counts for model 2 and 4. When comparing model 1 and 2 and model 3 and 4, we can see that there is a difference in deviance and p-value. The p-value for both likelihood tests are about 0.0497, which is statistically significant and means that the models with all features perform better, but not by a lot (it is just below 0.05). When Model 1 and 5 are compared, we see that the model trained with all best features performs better than the model trained on best features without **Age**. It is shown that the p-value is very low and the deviance is negative meaning that the first model is performing better. What all Likeliness Ratio tests also show us are that the first two Likeliness Ratio tests and the last two deliver us the same results. Based on which models we are comparing, we can also prove that scaling the dataset has not influenced the model results. This means that logistic regression is not tricked by the size of data, but it can be tricked by outliers.

# Testing significance of model 1 compared to model 3  
# H0: Model 1 (min-max scaled;best features selected) is more significant than model 3 (original scale; best features selected)  
execute.anova.1 <- function() {  
 eval.parent(substitute(anova.runs$anova.1 <- anova.runs$anova.1 + 1))  
 return(anova(model.1,model.3, test='LRT'))  
}  
  
# Testing significance of model 2 compared to model 4  
# H0: Model 2 (min-max scaled; all features) is more significant than model 4 (original scale; all features)  
execute.anova.2 <- function() {  
 eval.parent(substitute(anova.runs$anova.2 <- anova.runs$anova.2 + 1))  
 return(anova(model.2, model.4, test='LRT'))  
}  
  
# Testing significance of model 1 compared to model 2  
# H0: Model 1 (min-max scaled; best features selected) is more significant than model 2 (min-max scaled; all features)  
execute.anova.3 <- function() {  
 eval.parent(substitute(anova.runs$anova.3 <- anova.runs$anova.3 + 1))  
 return(anova(model.1, model.2, test='LRT'))  
}  
  
# Testing significance of model 1 compared to model 5  
# H0: Model 1 (min-max scaled; best features selected) is more significant than model 5 (min-max scaled; best features without Age)  
execute.anova.4 <- function() {  
 eval.parent(substitute(anova.runs$anova.4 <- anova.runs$anova.4 + 1))  
 return(anova(model.1, model.5, test='LRT'))  
}  
  
# Testing significance of model 3 compared to model 4  
# H0: Model 3 (original scale; best features selected) is more significant than model 4 (original scale; all features)  
execute.anova.5 <- function() {  
 eval.parent(substitute(anova.runs$anova.5 <- anova.runs$anova.5 + 1))  
 return(anova(model.3, model.4, test='LRT'))  
}  
  
# Execute the four Likeliness Ratio tests  
# Print Likeliness Ratio test results  
anova.1 <- execute.anova.1()  
print(anova.1)

## Analysis of Deviance Table  
##   
## Model 1: Diagnosis ~ Pregnancies + Age + Glucose + BMI  
## Model 2: Diagnosis ~ Pregnancies + Age + Glucose + BMI  
## Resid. Df Resid. Dev Df Deviance Pr(>Chi)  
## 1 419 412.68   
## 2 419 412.68 0 0

anova.2 <- execute.anova.2()  
print(anova.2)

## Analysis of Deviance Table  
##   
## Model 1: Diagnosis ~ Pregnancies + Glucose + BloodPressure + SkinThickness +   
## Insulin + BMI + Pedigree + Age  
## Model 2: Diagnosis ~ Pregnancies + Glucose + BloodPressure + SkinThickness +   
## Insulin + BMI + Pedigree + Age  
## Resid. Df Resid. Dev Df Deviance Pr(>Chi)  
## 1 415 403.09   
## 2 415 403.09 0 0

anova.3 <- execute.anova.3()  
print(anova.3)

## Analysis of Deviance Table  
##   
## Model 1: Diagnosis ~ Pregnancies + Age + Glucose + BMI  
## Model 2: Diagnosis ~ Pregnancies + Glucose + BloodPressure + SkinThickness +   
## Insulin + BMI + Pedigree + Age  
## Resid. Df Resid. Dev Df Deviance Pr(>Chi)   
## 1 419 412.68   
## 2 415 403.09 4 9.5879 0.04797 \*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

anova.4 <- execute.anova.4()  
print(anova.4)

## Analysis of Deviance Table  
##   
## Model 1: Diagnosis ~ Pregnancies + Age + Glucose + BMI  
## Model 2: Diagnosis ~ Pregnancies + Glucose + BMI  
## Resid. Df Resid. Dev Df Deviance Pr(>Chi)   
## 1 419 412.68   
## 2 420 428.51 -1 -15.832 6.921e-05 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

anova.5 <- execute.anova.5()  
print(anova.5)

## Analysis of Deviance Table  
##   
## Model 1: Diagnosis ~ Pregnancies + Age + Glucose + BMI  
## Model 2: Diagnosis ~ Pregnancies + Glucose + BloodPressure + SkinThickness +   
## Insulin + BMI + Pedigree + Age  
## Resid. Df Resid. Dev Df Deviance Pr(>Chi)   
## 1 419 412.68   
## 2 415 403.09 4 9.5879 0.04797 \*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

# Bonferroni correction

To make sure that the models 2 and 4 always perform statistical significantly better, we can run the models a three more times and correct the p-value through Bonferroni correction. After running the Likeliness Tests another three times, we have now run them four times in total.

# Run Likeliness Ratio tests 3 more times, to use bonferroni correction  
for (i in 1:3) {  
 anova.1 <- execute.anova.1()  
 anova.2 <- execute.anova.2()  
 anova.3 <- execute.anova.3()  
 anova.4 <- execute.anova.4()  
 anova.5 <- execute.anova.5()  
   
}  
print("All four Likeliness Ratio tests have run 3 more times.")

## [1] "All four Likeliness Ratio tests have run 3 more times."

After running the Likeliness Ratio tests four times, we have to adjust the p-value and adjust the significance level. We do this according to the Bonferroni correction. This means that the significance level of 0.05 will be divided by the number of tests run, which will be most likely four. We also adjust the p-value with the p.adjust(method="bonferroni") function. We do this for all the four tests and it turns out that model 2 and 4 are not performing statistical significantly better if we run the Likeliness Ratio test four or more times, as it significance is barely below the 0.05 threshold. In this case the earlier Null hypothesis can’t be rejected and the models 1 and 3 perform better. We also see that the best features model still performs better than the model without the **Age** variable after correction. This means that the model trained on the best features is the best model, where this is probably model 1 as this does not require normalizing the data before feeding it. Which results in a faster prediction speed.

bonferroni\_corr <- function(p\_value, num\_run\_test) {  
   
 # Apply Bonferroni correction  
 adjusted.alpha <- 0.05 / num\_run\_test  
   
 # Adjust the p-value using Bonferroni correction  
 adjusted.p.value <- p.adjust(p\_value, method = "bonferroni")  
   
 # Compare the adjusted p-value to the adjusted alpha  
 if(is.na(adjusted.p.value)) {  
 # The two models are the same and have no difference  
 cat("There is no significant difference between the two models, as the two models are the same.", "\nThe adjusted significance level is:", adjusted.alpha, "\nThe adjusted p-value is:", adjusted.p.value, "\n")  
 }  
 else if (adjusted.p.value >= adjusted.alpha) {  
 # Fail to reject the null hypothesis  
 cat("There is no significant difference between the two models (after Bonferroni correction).", "\nThe adjusted significance level is:", adjusted.alpha, "\nThe adjusted p-value is:", adjusted.p.value, "\n")  
 } else {  
 # Reject the null hypothesis and conclude significant difference  
 cat("There is a significant difference between the two models (after Bonferroni correction).", "\nThe adjusted significance level is:", adjusted.alpha, "\nThe adjusted p-value is:", adjusted.p.value, "\n")  
  
 }  
}  
  
# Execute Bonferroni correction for all four Likeliness Ratio tests  
bonferroni\_corr(anova.1$Pr[2], anova.runs$anova.1)

## There is no significant difference between the two models, as the two models are the same.   
## The adjusted significance level is: 0.0125   
## The adjusted p-value is: NA

bonferroni\_corr(anova.2$Pr[2], anova.runs$anova.2)

## There is no significant difference between the two models, as the two models are the same.   
## The adjusted significance level is: 0.0125   
## The adjusted p-value is: NA

bonferroni\_corr(anova.3$Pr[2], anova.runs$anova.3)

## There is no significant difference between the two models (after Bonferroni correction).   
## The adjusted significance level is: 0.0125   
## The adjusted p-value is: 0.04797161

bonferroni\_corr(anova.4$Pr[2], anova.runs$anova.4)

## There is a significant difference between the two models (after Bonferroni correction).   
## The adjusted significance level is: 0.0125   
## The adjusted p-value is: 6.920772e-05

bonferroni\_corr(anova.5$Pr[2], anova.runs$anova.5)

## There is no significant difference between the two models (after Bonferroni correction).   
## The adjusted significance level is: 0.0125   
## The adjusted p-value is: 0.04797161