Diabetes Prediction

Howard Nguyen

2023-06-30

Load libraries and packages

```r
library(tidyverse)
library(ggplot2)
library(ggcorrplot)
library(caret)
library(caretEnsemble)
library(psych)
library(Amelia)
library(mice)
library(GGally)
library(rpart)
library(randomForest)
library(e1071)
library(ROCR)
library(caret)
library(partykit)
library(gridExtra)
```

Load the data

```r
diabetes <- read.csv('diabetes_01.csv')
dim(diabetes)
```

```
## [1] 768   9
```

```r
#str(diabetes)
head(diabetes)
```

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

```r
names(diabetes) <- c("pregnancies", "glucose", "bloodpressure", "skinthickness", "insulin", "bmi",
                     "diabetespedigreefunction", "age", "outcome")
diabetes$outcome <- as.factor(diabetes$outcome) %>% na.omit()
# Change outcome variables as categorical with level = True (1) or False (0)
#diabetes$outcome <- factor(diabetes$outcome, levels = c(0,1), labels = c("False", "True"))
glimpse(diabetes)
```

```
## Rows: 768
## Columns: 9
## $ pregnancies              <int> 6, 1, 8, 1, 0, 5, 3, 10, 2, 8, 4, 10, 10, 1, ~
## $ glucose                  <int> 148, 85, 183, 89, 137, 116, 78, 115, 197, 125~
## $ bloodpressure            <int> 72, 66, 64, 66, 40, 74, 50, 0, 70, 96, 92, 74~
## $ skinthickness            <int> 35, 29, 0, 23, 35, 0, 32, 0, 45, 0, 0, 0, 0, ~
## $ insulin                  <int> 0, 0, 0, 94, 168, 0, 88, 0, 543, 0, 0, 0, 0, ~
## $ bmi                      <dbl> 33.6, 26.6, 23.3, 28.1, 43.1, 25.6, 31.0, 35.~
## $ diabetespedigreefunction <dbl> 0.627, 0.351, 0.672, 0.167, 2.288, 0.201, 0.2~
## $ age                      <int> 50, 31, 32, 21, 33, 30, 26, 29, 53, 54, 30, 3~
## $ outcome                  <fct> 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, ~
```

```r
#deeper dive in the data structure
str(diabetes)
```

```
## 'data.frame':    768 obs. of  9 variables:
##  $ pregnancies             : int  6 1 8 1 0 5 3 10 2 8 ...
##  $ glucose                 : int  148 85 183 89 137 116 78 115 197 125 ...
##  $ bloodpressure           : int  72 66 64 66 40 74 50 0 70 96 ...
##  $ skinthickness           : int  35 29 0 23 35 0 32 0 45 0 ...
##  $ insulin                 : int  0 0 0 94 168 0 88 0 543 0 ...
##  $ bmi                     : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
##  $ diabetespedigreefunction: num  0.627 0.351 0.672 0.167 2.288 ...
##  $ age                     : int  50 31 32 21 33 30 26 29 53 54 ...
##  $ outcome                 : Factor w/ 2 levels "0","1": 2 1 2 1 2 1 2 1 2 2 ...
```

```r
describe(diabetes)
```

```
##                          vars   n   mean     sd median trimmed   mad   min
## pregnancies                 1 768   3.85   3.37   3.00    3.46  2.97  0.00
## glucose                     2 768 120.89  31.97 117.00  119.38 29.65  0.00
## bloodpressure               3 768  69.11  19.36  72.00   71.36 11.86  0.00
## skinthickness               4 768  20.54  15.95  23.00   19.94 17.79  0.00
## insulin                     5 768  79.80 115.24  30.50   56.75 45.22  0.00
## bmi                         6 768  31.99   7.88  32.00   31.96  6.82  0.00
## diabetespedigreefunction    7 768   0.47   0.33   0.37    0.42  0.25  0.08
## age                         8 768  33.24  11.76  29.00   31.54 10.38 21.00
## outcome*                    9 768   1.35   0.48   1.00    1.31  0.00  1.00
##                             max  range  skew kurtosis   se
## pregnancies               17.00  17.00  0.90     0.14 0.12
## glucose                  199.00 199.00  0.17     0.62 1.15
## bloodpressure            122.00 122.00 -1.84     5.12 0.70
## skinthickness             99.00  99.00  0.11    -0.53 0.58
## insulin                  846.00 846.00  2.26     7.13 4.16
## bmi                       67.10  67.10 -0.43     3.24 0.28
## diabetespedigreefunction   2.42   2.34  1.91     5.53 0.01
## age                       81.00  60.00  1.13     0.62 0.42
## outcome*                   2.00   1.00  0.63    -1.60 0.02
```

```
# check if there's any NA values in the dataset
colSums(is.na(diabetes))
```

```
##              pregnancies                  glucose            bloodpressure
##                        0                        0                        0
##            skinthickness                  insulin                      bmi
##                        0                        0                        0
## diabetespedigreefunction                      age                  outcome
##                        0                        0                        0
```

```
sum(is.na(diabetes))
```

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

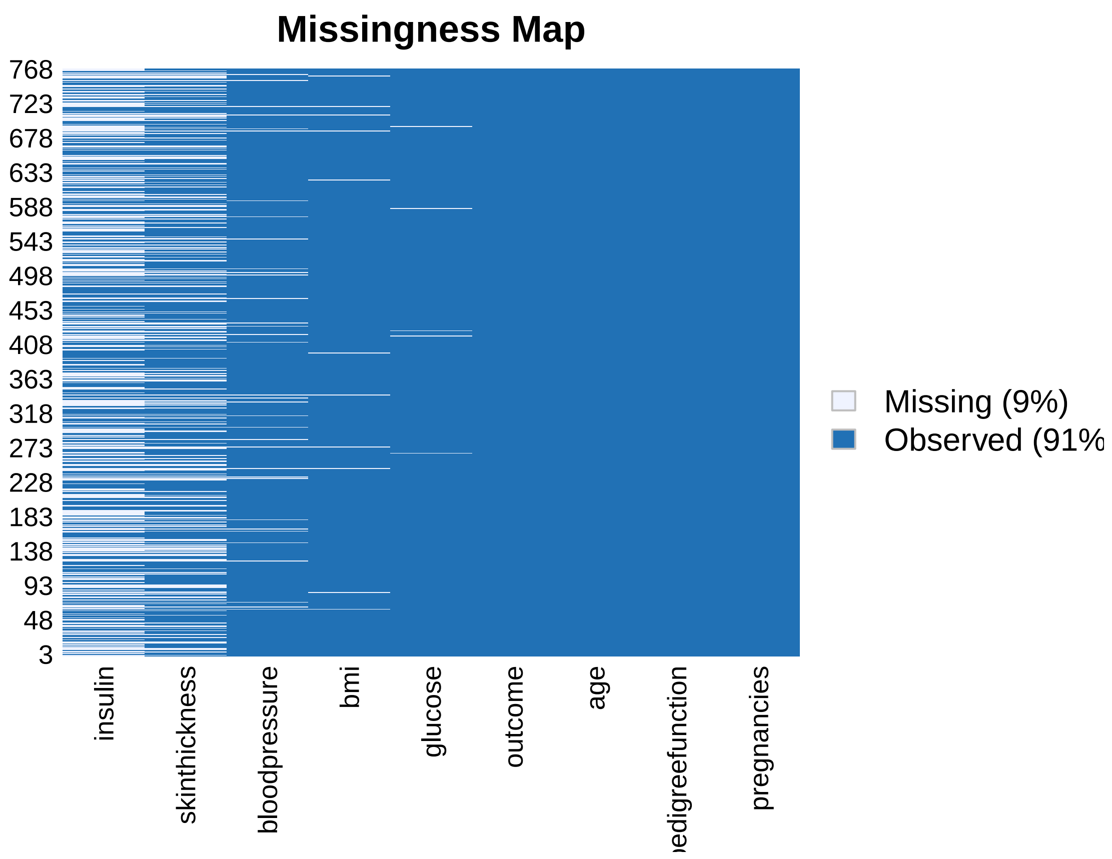
```
anyNA(diabetes)
```

```
## [1] FALSE
```

Data cleansing & reformating

```
# replace '0' values into NA
diabetes[, 2:7][diabetes[, 2:7] == 0] <- NA
```

```
# check missing values in the dataset
diabetes_missmap <- missmap(diabetes)
```



9% of missing values, if we remove all of them, our dataset will be smaller. So, we can use mice package for its imputations

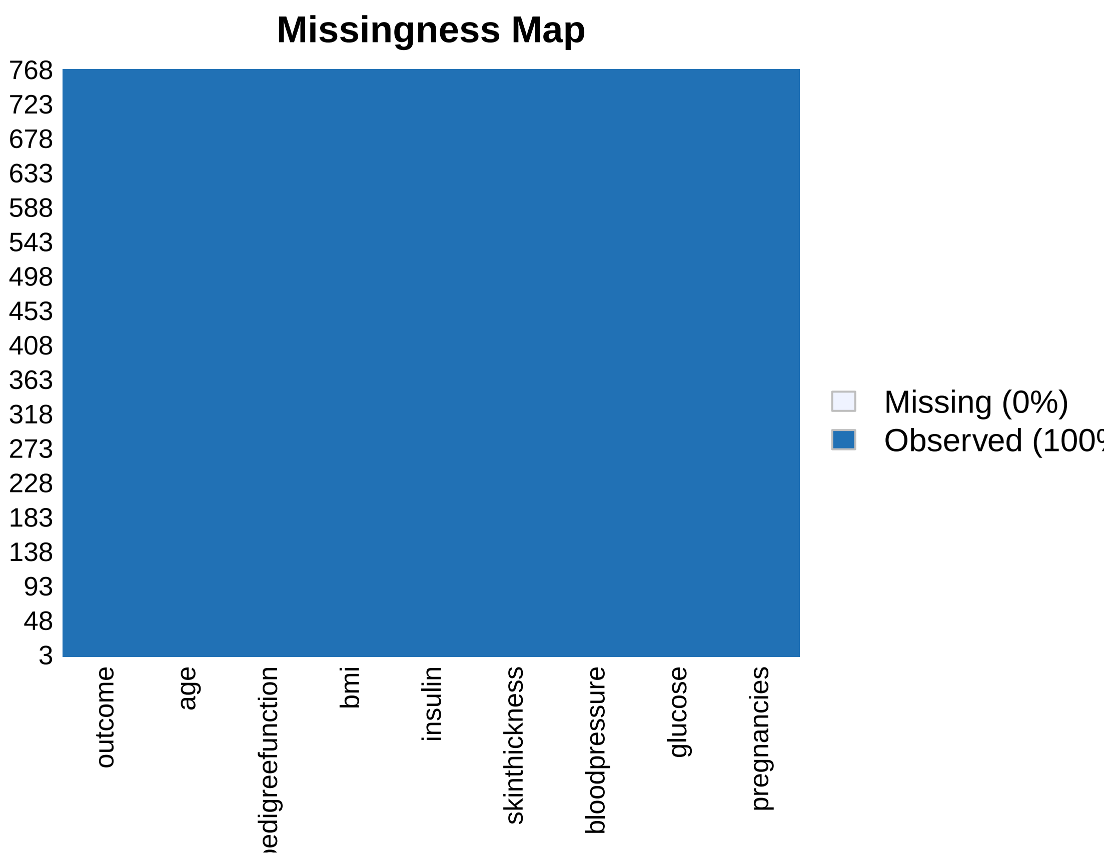
```
# predict the missing values
diabetes_mice <- mice(diabetes[, c("insulin","skinthickness","bloodpressure","bmi","glucose")],
                      method = 'rf')
```

```
##
##  iter imp variable
##   1   1  insulin  skinthickness  bloodpressure  bmi  glucose
##   1   2  insulin  skinthickness  bloodpressure  bmi  glucose
##   1   3  insulin  skinthickness  bloodpressure  bmi  glucose
##   1   4  insulin  skinthickness  bloodpressure  bmi  glucose
##   1   5  insulin  skinthickness  bloodpressure  bmi  glucose
##   2   1  insulin  skinthickness  bloodpressure  bmi  glucose
##   2   2  insulin  skinthickness  bloodpressure  bmi  glucose
##   2   3  insulin  skinthickness  bloodpressure  bmi  glucose
##   2   4  insulin  skinthickness  bloodpressure  bmi  glucose
##   2   5  insulin  skinthickness  bloodpressure  bmi  glucose
##   3   1  insulin  skinthickness  bloodpressure  bmi  glucose
##   3   2  insulin  skinthickness  bloodpressure  bmi  glucose
##   3   3  insulin  skinthickness  bloodpressure  bmi  glucose
##   3   4  insulin  skinthickness  bloodpressure  bmi  glucose
##   3   5  insulin  skinthickness  bloodpressure  bmi  glucose
##   4   1  insulin  skinthickness  bloodpressure  bmi  glucose
##   4   2  insulin  skinthickness  bloodpressure  bmi  glucose
##   4   3  insulin  skinthickness  bloodpressure  bmi  glucose
##   4   4  insulin  skinthickness  bloodpressure  bmi  glucose
##   4   5  insulin  skinthickness  bloodpressure  bmi  glucose
##   5   1  insulin  skinthickness  bloodpressure  bmi  glucose
##   5   2  insulin  skinthickness  bloodpressure  bmi  glucose
##   5   3  insulin  skinthickness  bloodpressure  bmi  glucose
##   5   4  insulin  skinthickness  bloodpressure  bmi  glucose
##   5   5  insulin  skinthickness  bloodpressure  bmi  glucose
```

```
diabetes_result <- complete(diabetes_mice)
```

```
# add the predicted missing values into the dataset
diabetes$insulin <- diabetes_result$insulin
diabetes$skinthickness <- diabetes_result$skinthickness
diabetes$bloodpressure <- diabetes_result$bloodpressure
diabetes$bmi <- diabetes_result$bmi
diabetes$glucose <- diabetes_result$glucose
```

```
# review again the dataset
diabetes_missmap <- missmap(diabetes)
```

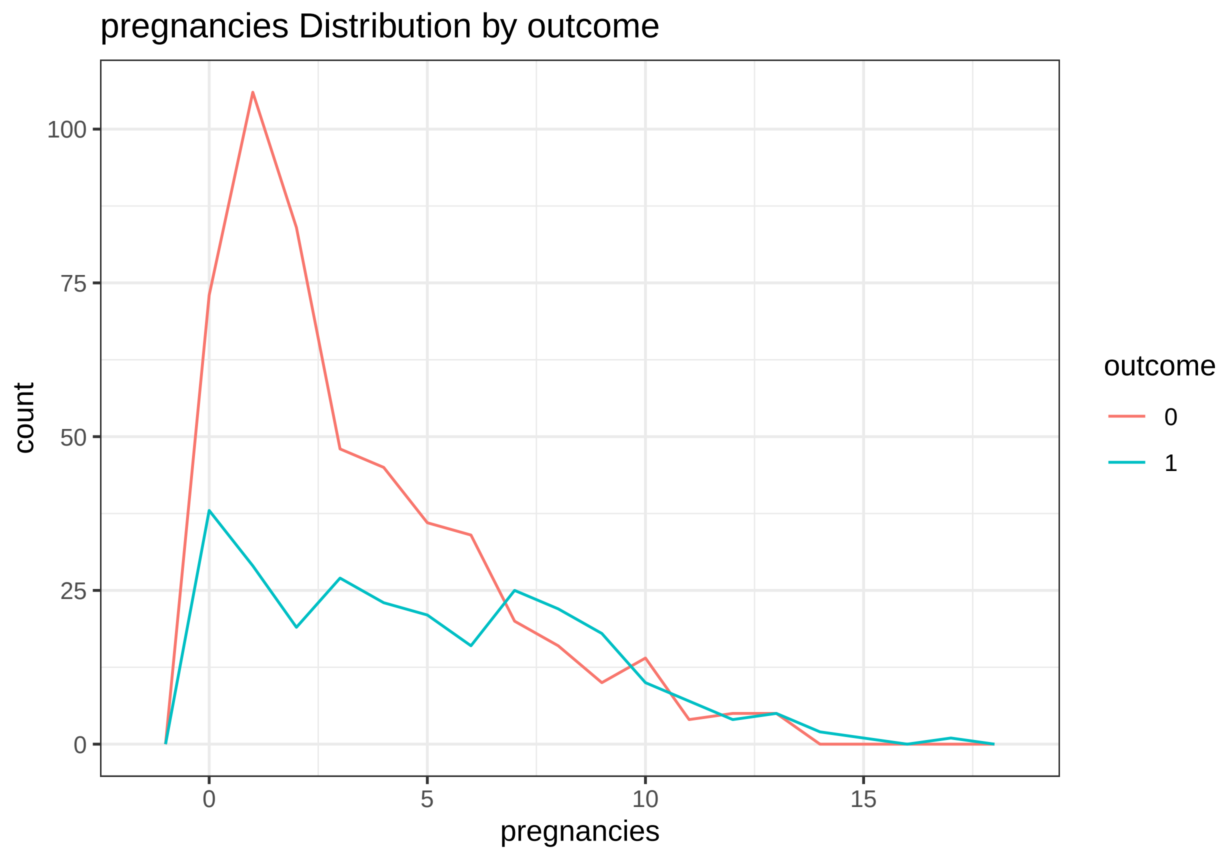


Perfect. There's no more missing values in the dataset

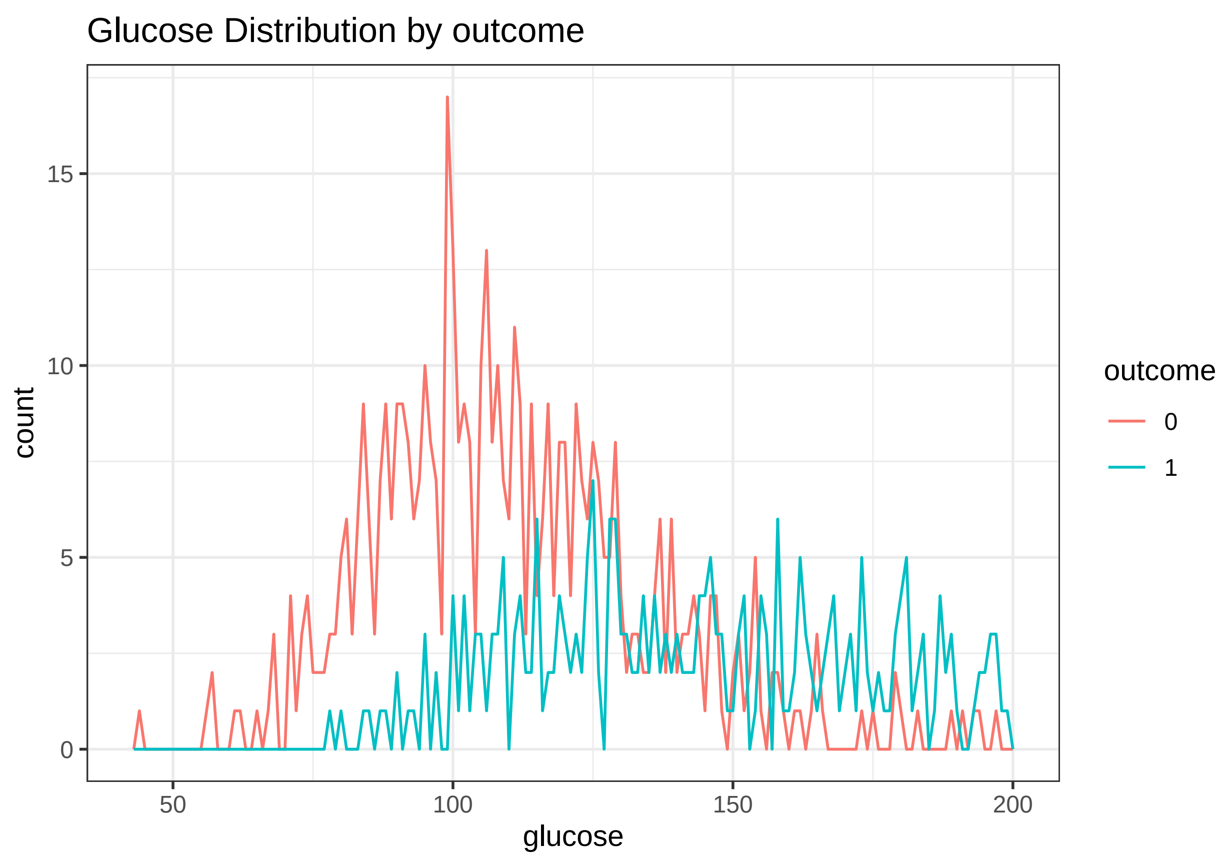
EDA - Exploratory Data Analysis

We'll visualize variables to have a better look and to understand the significance of each predictor variable.

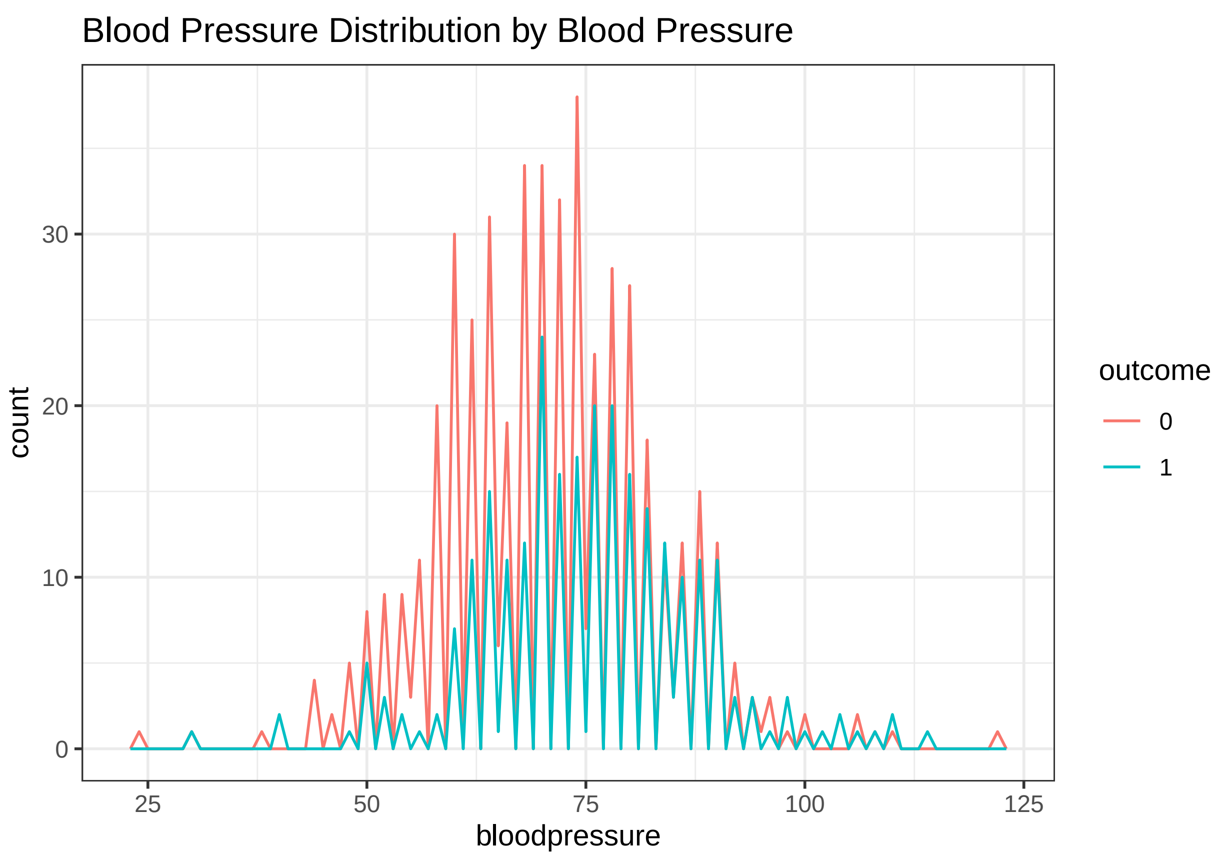
```
# plot pregnancies vs outcome
pregnancies_plot <- ggplot(diabetes, aes(pregnancies, fill=outcome, colour = outcome)) +
  geom_freqpoly(binwidth = 1) + labs(title="pregnancies Distribution by outcome")
pregnancies_plot + theme_bw()
```



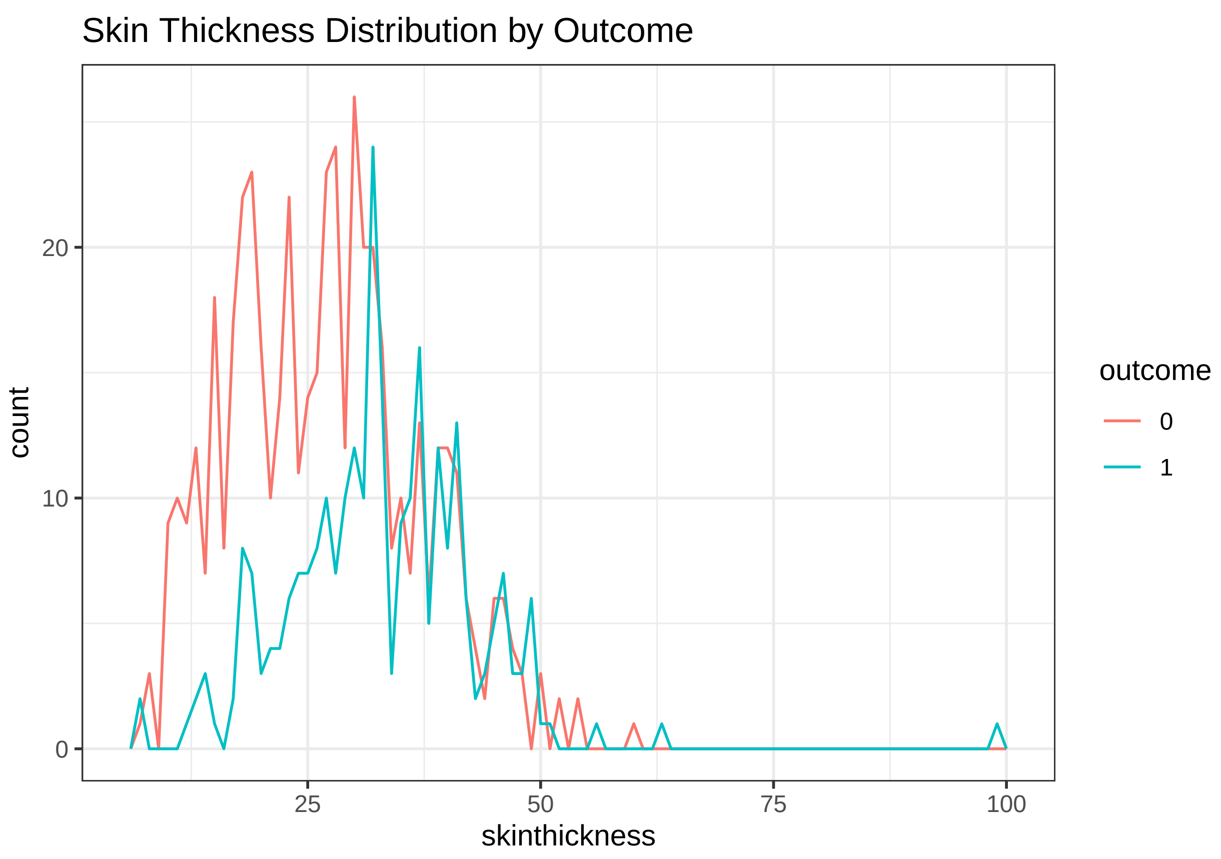
```
#plot glucose vs outcome
glucose_plot <- ggplot(diabetes, aes(x=glucose, fill=outcome, color=outcome)) +
  geom_freqpoly(binwidth = 1) + labs(title="Glucose Distribution by outcome")
glucose_plot + theme_bw()
```



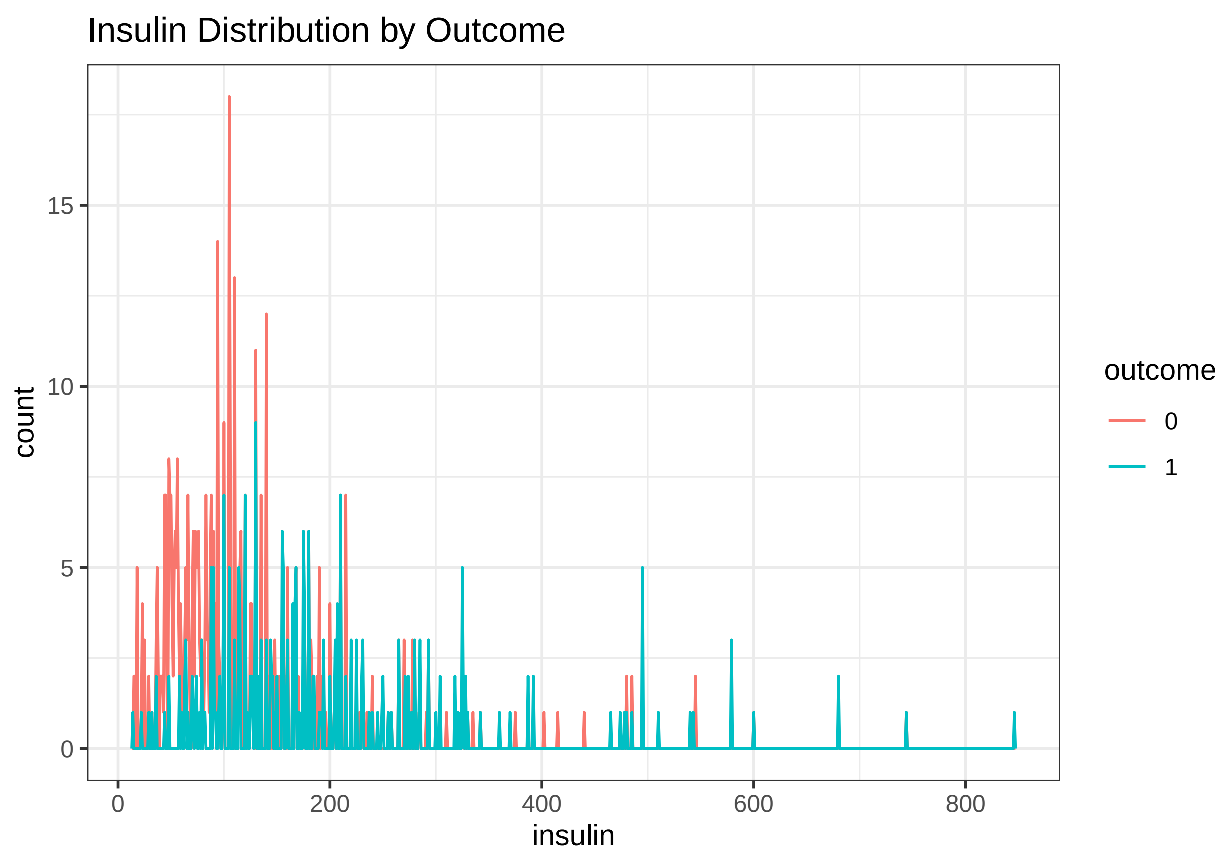
```
#plot bloodpressure vs outcome
bloodpressure_plot <- ggplot(diabetes, aes(x=bloodpressure, fill=outcome, color=outcome)) +
  geom_freqpoly(binwidth = 1) + labs(title="Blood Pressure Distribution by Blood Pressure")
bloodpressure_plot + theme_bw()
```



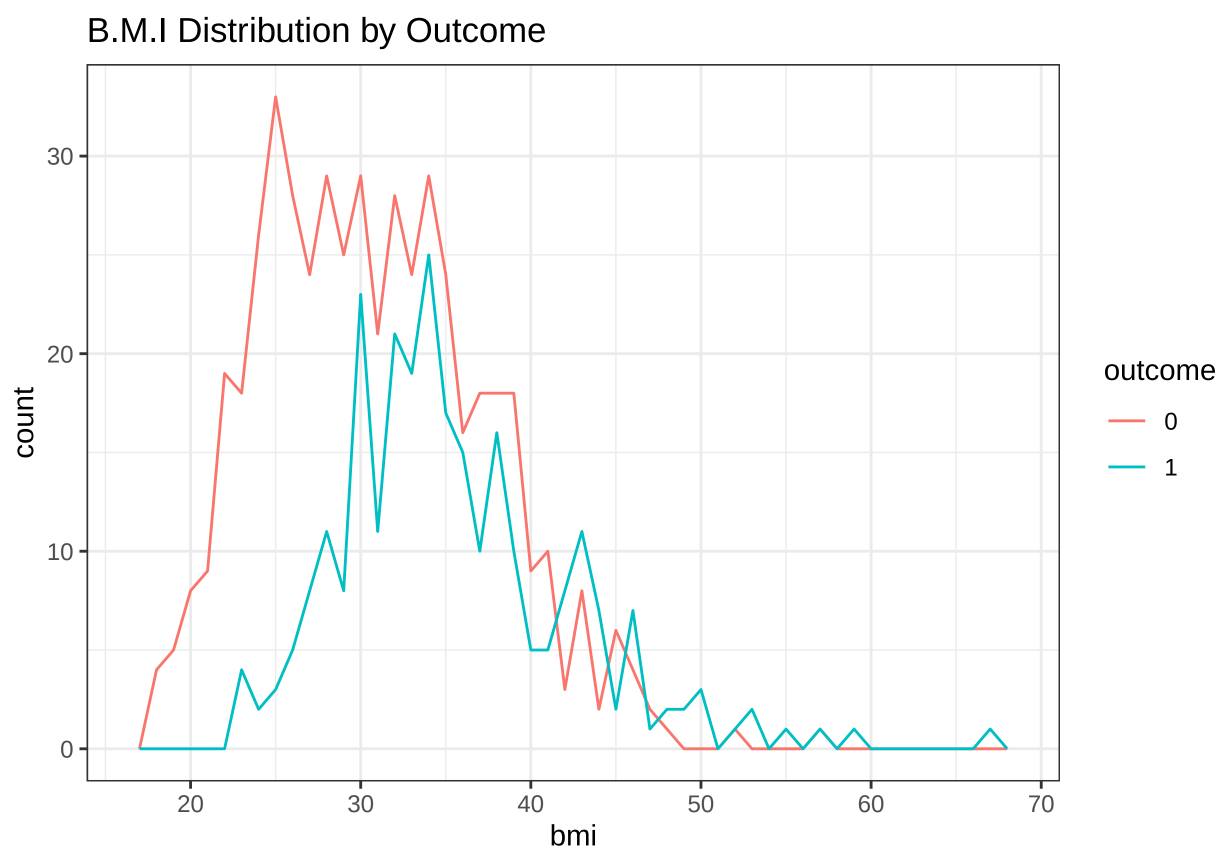
```
# plot skinthickness vs outcome
skinthickness_plot <- ggplot(diabetes, aes(skinthickness, fill=outcome, colour = outcome)) +
  geom_freqpoly(binwidth = 1) + labs(title="Skin Thickness Distribution by Outcome")
skinthickness_plot + theme_bw()
```



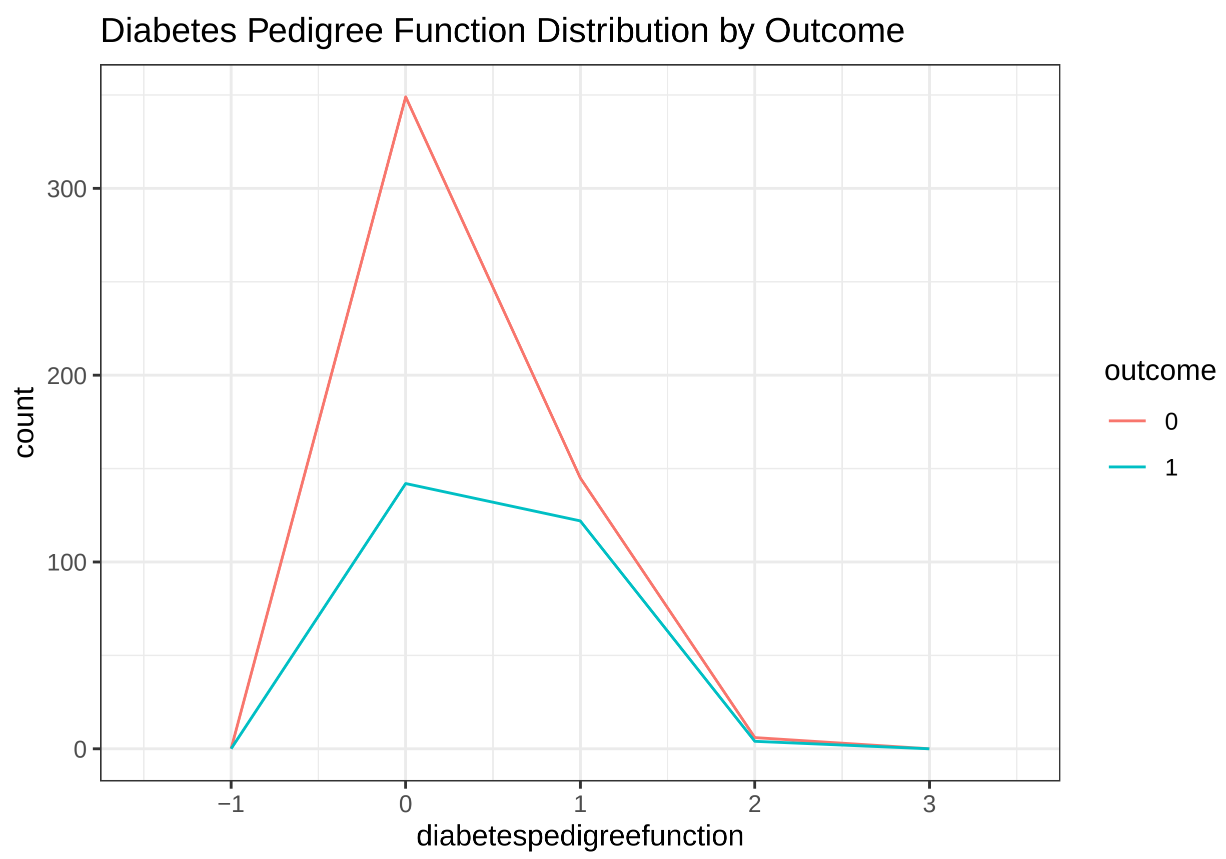
```r
# plot insulin vs Outcome
insulin_plot <- ggplot(diabetes, aes(insulin, fill=outcome, colour = outcome)) +
  geom_freqpoly(binwidth = 1) + labs(title="Insulin Distribution by Outcome")
insulin_plot + theme_bw()
```



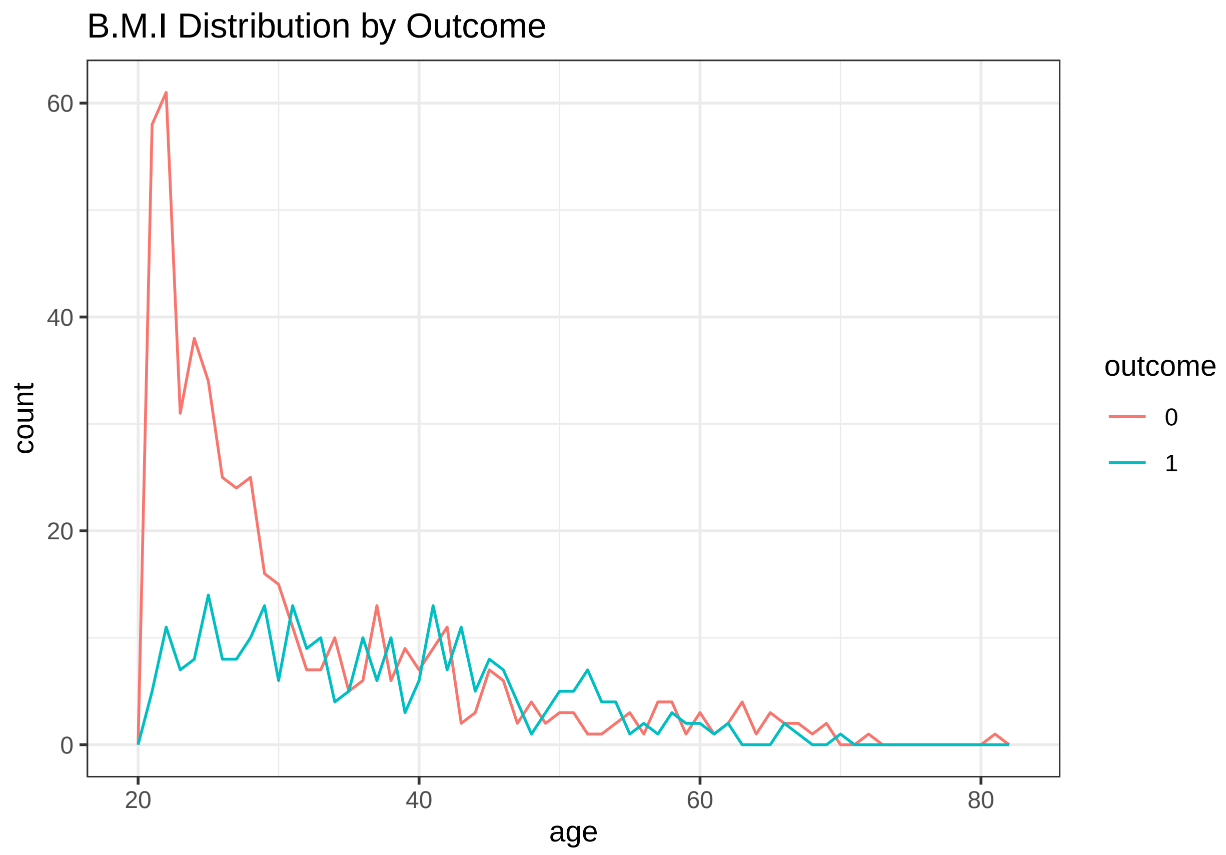
```r
# plot BMI vs Outcome
bmi_plot <- ggplot(diabetes, aes(bmi, fill=outcome, colour = outcome)) +
  geom_freqpoly(binwidth = 1) + labs(title="B.M.I Distribution by Outcome")
bmi_plot + theme_bw()
```



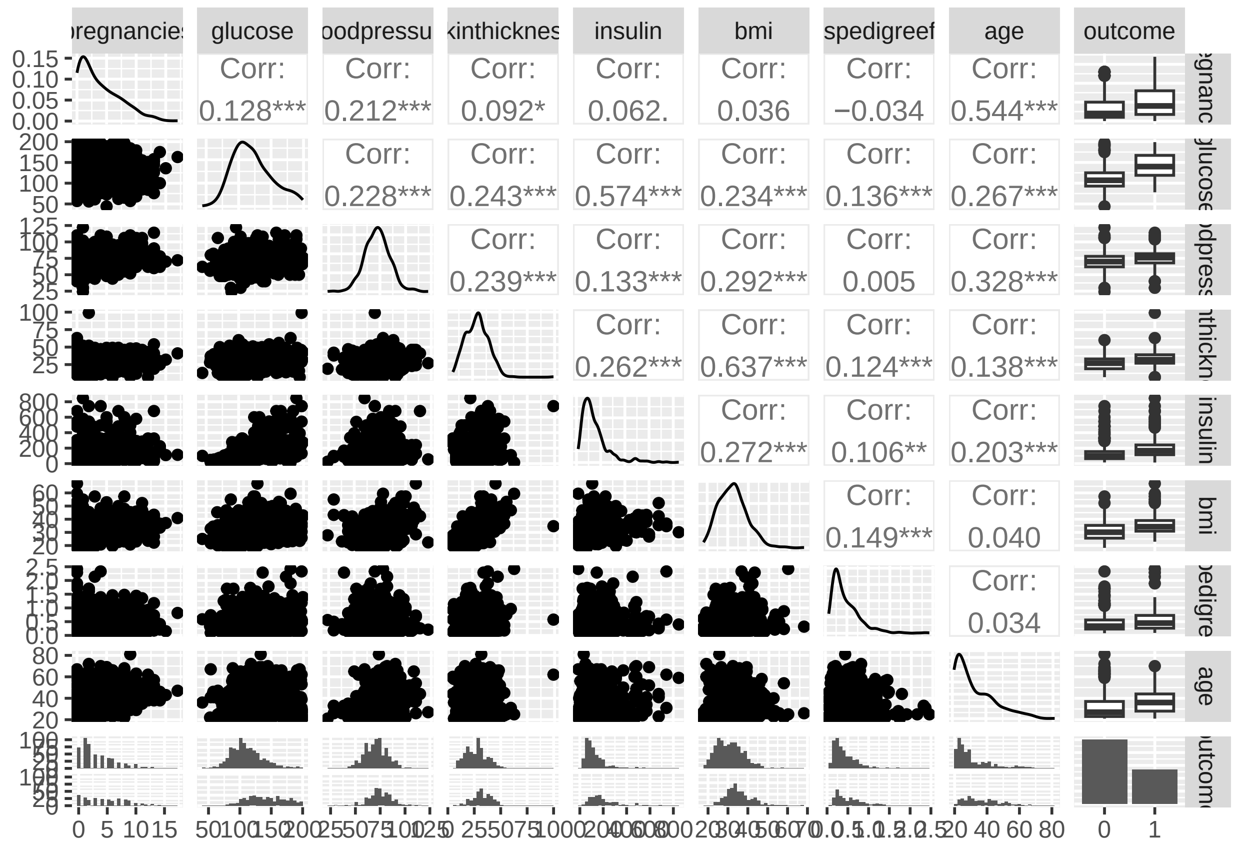
```r
# plot diabetespedigreefunction vs Outcome
diabetespedigreefunction_plot <- ggplot(diabetes, aes(diabetespedigreefunction, fill=outcome, colour = 
  geom_freqpoly(binwidth = 1) + labs(title="Diabetes Pedigree Function Distribution by Outcome")
diabetespedigreefunction_plot + theme_bw()
```



```
# plot age vs Outcome
age_plot <- ggplot(diabetes, aes(age, fill=outcome, colour = outcome)) +
  geom_freqpoly(binwidth = 1) + labs(title="B.M.I Distribution by Outcome")
#geom_histogram(binwidth = 1) + labs(title="Age Distribution by Outcome")
age_plot + theme_bw()
```



```
# Correlation review
ggpairs(diabetes)
```



Model Fitting

Splitting the dataset into training set (80%) and testing set (20%)

```
# split data into training and test data sets
sliced <- createDataPartition(y = diabetes$outcome,p = 0.8,list = FALSE)

diabetes_train <- diabetes[sliced,]
diabetes_test <- diabetes[-sliced,]
```

```
# Check dimensions of the split
prop.table(table(diabetes$outcome)) * 100
```

```
##
##        0        1
## 65.10417 34.89583
```

```
prop.table(table(diabetes_train$outcome)) * 100
```

```
##
##        0        1
## 65.04065 34.95935
```

```
prop.table(table(diabetes_test$outcome)) * 100
```

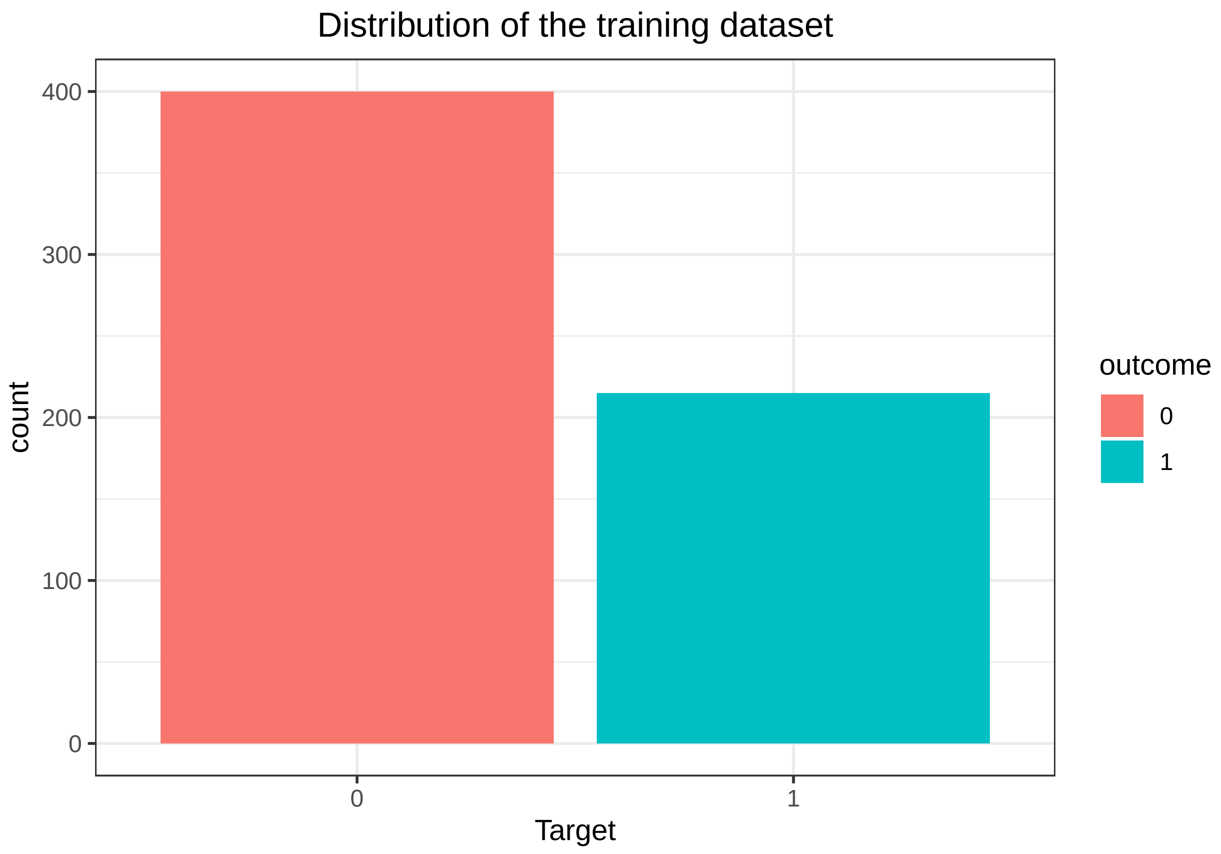
```
##
##        0        1
## 65.35948 34.64052
```

For comparing the outcome of the training and testing phase let's create separate variables that store the value of the response variable:

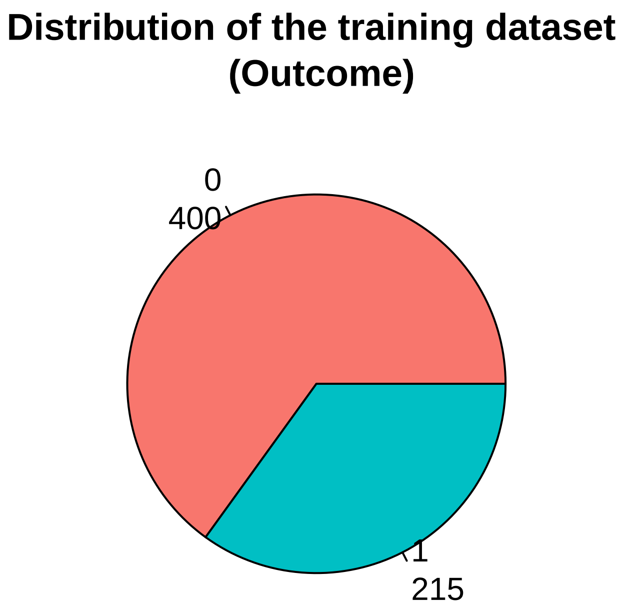
```
# create x_diabetes which holds the predictor variables and y which holds the response variables
x_diabetes = diabetes_train[,-9]
y_diabetes = diabetes_train$outcome
```

Review the distributions of the train dataset

```
ggplot(diabetes_train, aes(outcome, fill = outcome)) +
  geom_bar() +
  theme_bw() +
  labs(title = "Distribution of the training dataset", x = "Target") +
  theme(plot.title = element_text(hjust = 0.5))
```



```r
# view it with pie chart
mytable <- table(diabetes_train$outcome)
lbls <- paste(names(mytable), "\n", mytable, sep="")
pie(mytable, labels = lbls,
    main="Distribution of the training dataset \n (Outcome)",
    col=c("#f8766d", "#00bfc4"))
```



Building the Correlation Matrix

```r
cor_data <- cor(diabetes_train[,setdiff(names(diabetes), 'outcome')])
#Numerical Correlation Matrix
cor_data
```

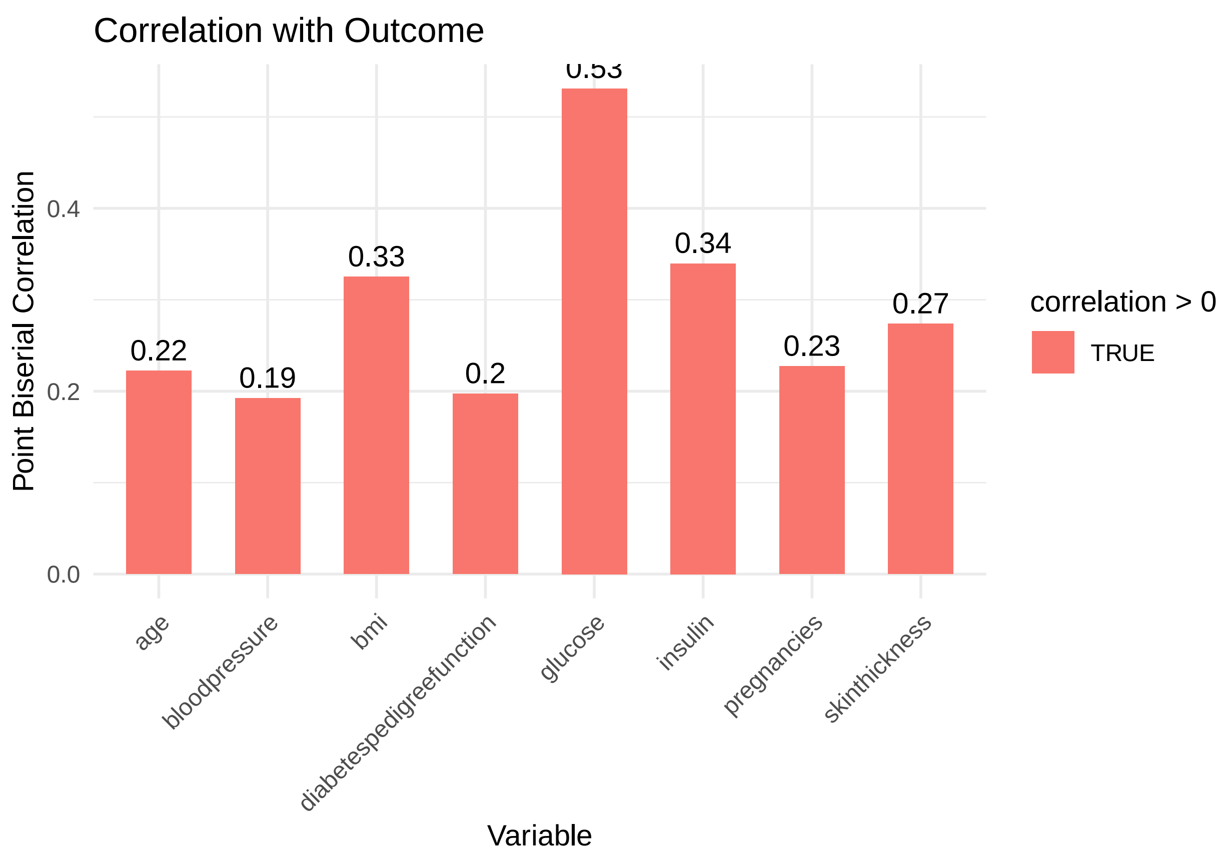
```
##                          pregnancies   glucose bloodpressure skinthickness
## pregnancies               1.00000000 0.1151145    0.19886061    0.09804601
## glucose                   0.11511452 1.0000000    0.22192444    0.23962412
## bloodpressure             0.19886061 0.2219244    1.00000000    0.26537430
## skinthickness             0.09804601 0.2396241    0.26537430    1.00000000
## insulin                   0.05336355 0.5729714    0.14874874    0.28010987
## bmi                       0.04915839 0.2465666    0.33275418    0.62749974
## diabetespedigreefunction -0.03577419 0.1253683    0.02822434    0.11027012
## age                       0.52513253 0.2324188    0.32019895    0.14577367
##                             insulin        bmi diabetespedigreefunction
## pregnancies              0.05336355 0.04915839               -0.03577419
## glucose                  0.57297141 0.24656664                0.12536831
## bloodpressure            0.14874874 0.33275418                0.02822434
## skinthickness            0.28010987 0.62749974                0.11027012
## insulin                  1.00000000 0.30997229                0.11667879
## bmi                      0.30997229 1.00000000                0.12331218
## diabetespedigreefunction 0.11667879 0.12331218                1.00000000
## age                      0.17769933 0.04498255                0.02301554
##                                 age
## pregnancies              0.52513253
## glucose                  0.23241875
## bloodpressure            0.32019895
## skinthickness            0.14577367
## insulin                  0.17769933
## bmi                      0.04498255
## diabetespedigreefunction 0.02301554
## age                      1.00000000
```

```
# Convert 'outcome' to numeric
y_diabetes <- as.numeric(y_diabetes)
```

```
# Calculate the point biserial correlation between 'outcome' and predictor variables
cor_df <- data.frame(variable = names(x_diabetes),
                     correlation = sapply(x_diabetes, function(x) cor(as.numeric(x), y_diabetes)))
```

```
# Sort the data frame by absolute correlation values
cor_df <- cor_df[order(abs(cor_df$correlation), decreasing = TRUE), ]
```

```
# Create the bar plot with correlation values
ggplot(cor_df, aes(x = variable, y = correlation, fill = correlation > 0)) +
  geom_bar(stat = "identity", width = 0.6) +
  geom_text(aes(label = round(correlation, 2)), vjust = -0.5) +  # Add correlation values
  labs(x = "Variable", y = "Point Biserial Correlation", title = "Correlation with Outcome") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Glucose is the top factor to cause diabetes 49%, and follow BMI and insulin to share 29%.

```r
library(ggplot2)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths
```

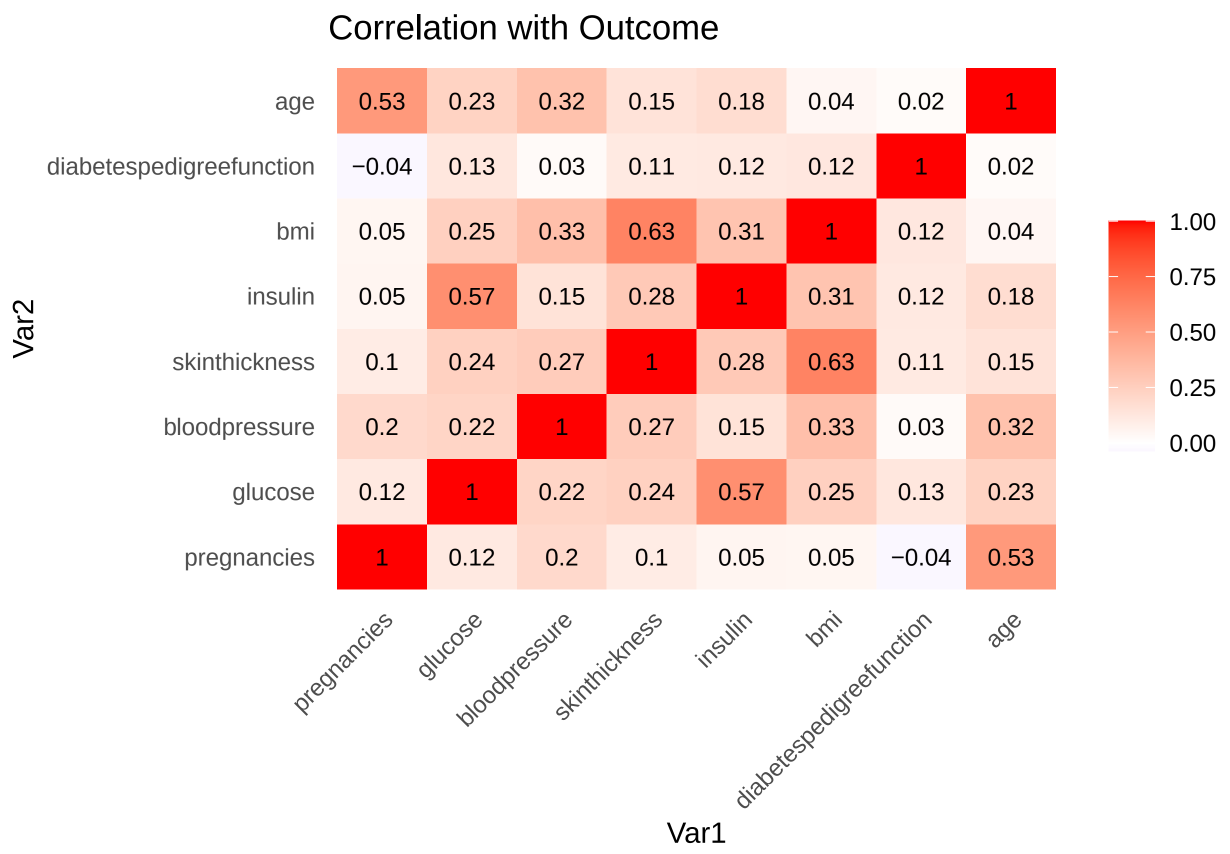
```r
# Convert 'outcome' to numeric
y_diabetes <- as.numeric(y_diabetes)

# Calculate the correlations between 'outcome' and predictor variables
cor_data <- cor(cbind(y_diabetes, x_diabetes))

# Reorder variables in the correlation matrix
variable_order <- c("pregnancies", "glucose", "bloodpressure", "skinthickness",
                    "insulin", "bmi", "diabetespedigreefunction", "age")
cor_data <- cor_data[variable_order, variable_order]

# Convert correlation matrix to long format
cor_data_long <- melt(cor_data)
```

```r
# Create a correlation matrix plot using ggplot2
ggplot(cor_data_long, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", mid = "white", high = "red", midpoint = 0) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        axis.text.y = element_text(angle = 0),
        panel.grid = element_blank(),
        legend.title = element_blank()) +
  labs(title = "Correlation with Outcome") +
  geom_text(aes(label = round(value, 2)), color = "black", size = 3)
```



Var2 \ Var1	pregnancies	glucose	bloodpressure	skinthickness	insulin	bmi	diabetespedigreefunction	age
age	0.53	0.23	0.32	0.15	0.18	0.04	0.02	1
diabetespedigreefunction	−0.04	0.13	0.03	0.11	0.12	0.12	1	0.02
bmi	0.05	0.25	0.33	0.63	0.31	1	0.12	0.04
insulin	0.05	0.57	0.15	0.28	1	0.31	0.12	0.18
skinthickness	0.1	0.24	0.27	1	0.28	0.63	0.11	0.15
bloodpressure	0.2	0.22	1	0.27	0.15	0.33	0.03	0.32
glucose	0.12	1	0.22	0.24	0.57	0.25	0.13	0.23
pregnancies	1	0.12	0.2	0.1	0.05	0.05	−0.04	0.53

With this result from Correlation Coefficients calculation: we can see BMI 68% relates to diabetes and age 56% and insulin 53% relating to diabetes.

```r
#print(rownames(cor_data))
#print(colnames(cor_data))
```

```r
# Correlation matrix plots
#corrplot::corrplot(cor_data)
#Variations
#corrplot::corrplot(cor_data, type = "lower", method = "number")
#corrplot::corrplot(cor_data, type = "lower", method = "pie")
#ggcorr(cor_data, label = T, label_size = 3, hjust = 0.9)
```

We found BMI, Glucose, and Age have a strong correlation with diabetes.

Logistic Regression model

```r
# Load the required libraries
library(caret)
# Convert the outcome variable to a factor
y_diabetes <- as.factor(y_diabetes)
# Create the training control object
train_control <- trainControl(method = "cv", number = 10)
# Train the logistic regression model
diabetes_logistic <- train(x_diabetes, y_diabetes, method = "glm", family = "binomial", trControl = trai
# Print the model results
print(diabetes_logistic)
```

```
## Generalized Linear Model
##
## 615 samples
##   8 predictor
##   2 classes: '1', '2'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 554, 553, 554, 553, 553, 554, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7803543  0.49657
```

Naive Bayes model

```r
# Load the required libraries
library(caret)
# Convert the outcome variable to a factor
y_diabetes <- as.factor(y_diabetes)
# Create the training control object
train_control <- trainControl(method = "cv", number = 10)
# Train the Naive Bayes model
diabetes_naive_bayes <- train(x_diabetes, y_diabetes, method = "naive_bayes", trControl = train_control
# Print the model results
print(diabetes_naive_bayes)
```

```
## Naive Bayes
##
## 615 samples
##   8 predictor
##   2 classes: '1', '2'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 553, 554, 554, 554, 554, 554, ...
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE      0.7658117  0.4763654
##    TRUE      0.7723162  0.5061236
##
## Tuning parameter 'laplace' was held constant at a value of 0
## Tuning
##  parameter 'adjust' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were laplace = 0, usekernel = TRUE
##  and adjust = 1.
```

The Naive Bayes model using the Naive Bayes algorithm has been trained and evaluated with cross-validation. Here are the results:

Accuracy: 0.7576150 Kappa: 0.4783683 The model was trained with 615 samples and 8 predictor variables. It is a classification model with 2 classes: '1' and '2'. No pre-processing was applied to the data before training the model.

During the cross-validation process, 10-fold cross-validation was used, and the sample sizes varied slightly across the folds.

The model was evaluated with two tuning parameters: 'usekernel' and 'laplace'. The optimal values selected for the model were 'laplace = 0', 'usekernel = TRUE', and 'adjust = 1'. The accuracy was used as the criterion to select the optimal model, and the largest accuracy value was chosen.

These results indicate that the Naive Bayes model achieved an accuracy of 0.7576150 and a kappa value of 0.4783683 on the cross-validated data.

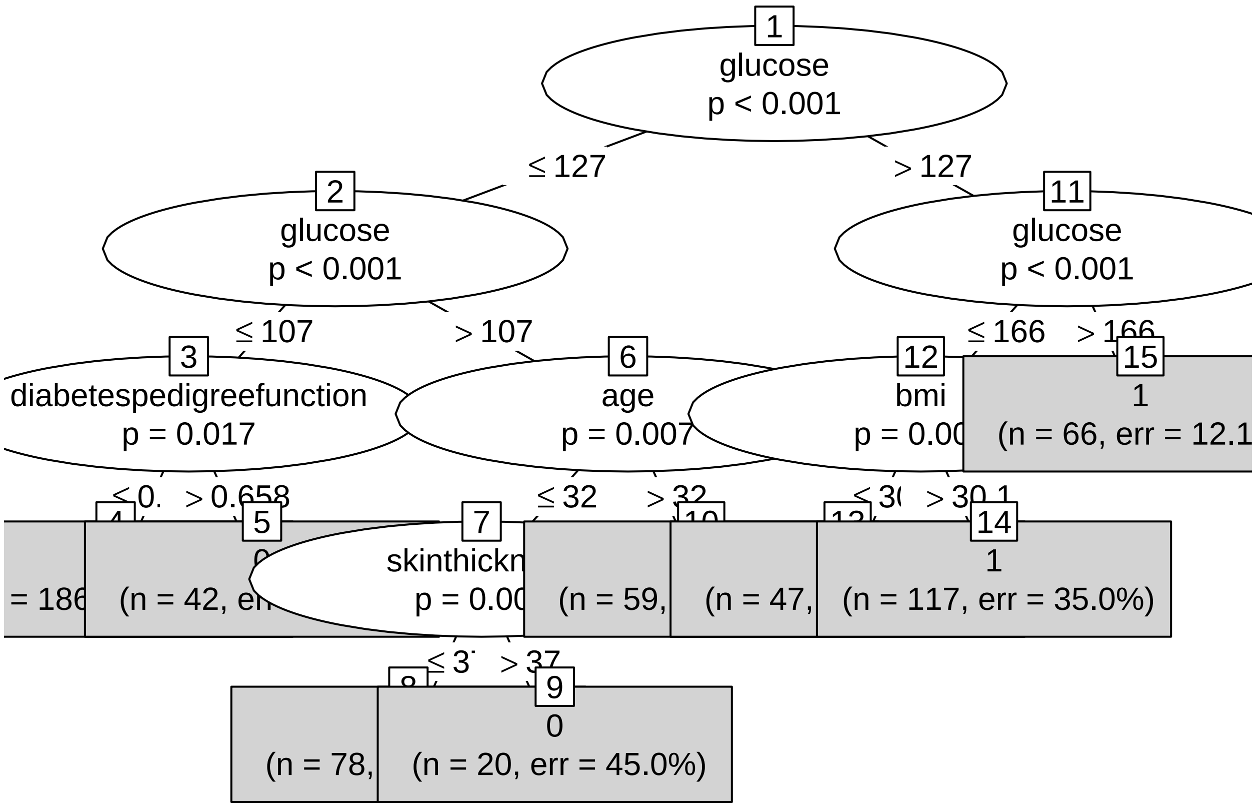
```
#diabetes_naive_bayes = train(x_diabetes,y_diabetes,'nb',trControl=trainControl(method='cv',number=10))
#diabetes_naive_bayes
```

Decision Tree model

```
diabetes_DTree_model <- ctree(formula = outcome ~ ., data = diabetes_train)
diabetes_DTree_model
```

```
##
## Model formula:
## outcome ~ pregnancies + glucose + bloodpressure + skinthickness +
##     insulin + bmi + diabetespedigreefunction + age
##
## Fitted party:
## [1] root
## |   [2] glucose <= 127
## |   |   [3] glucose <= 107
## |   |   |   [4] diabetespedigreefunction <= 0.658: 0 (n = 186, err = 4.8%)
## |   |   |   [5] diabetespedigreefunction > 0.658: 0 (n = 42, err = 26.2%)
## |   |   [6] glucose > 107
## |   |   |   [7] age <= 32
## |   |   |   |   [8] skinthickness <= 37: 0 (n = 78, err = 10.3%)
## |   |   |   |   [9] skinthickness > 37: 0 (n = 20, err = 45.0%)
## |   |   |   [10] age > 32: 1 (n = 59, err = 49.2%)
## |   [11] glucose > 127
## |   |   [12] glucose <= 166
## |   |   |   [13] bmi <= 30.1: 0 (n = 47, err = 29.8%)
## |   |   |   [14] bmi > 30.1: 1 (n = 117, err = 35.0%)
## |   |   [15] glucose > 166: 1 (n = 66, err = 12.1%)
##
## Number of inner nodes:    7
## Number of terminal nodes: 8
```

```r
plot(diabetes_DTree_model, type = 'simple')
```



The root node (node 1) represents the initial split based on the variable "glucose." If a person's glucose level is less than or equal to 154, they move to node 2. Otherwise, they move to node 9.

Node 2 represents the case where glucose is less than or equal to 154. Within this node, there is another split based on the variable "bmi." If a person's BMI is less than or equal to 26.2, they are classified as outcome 0 (non-diabetic) in node 3. Otherwise, they move to node 4.

Node 4 represents the case where BMI is greater than 26.2. Within this node, there is another split based on the variable "glucose." If a person's glucose level is less than or equal to 99, they are classified as outcome 0 (non-diabetic) in node 5. Otherwise, they move to node 6.

Node 6 represents the case where glucose is greater than 99. Within this node, there is another split based on the variable "pregnancies." If a person's number of pregnancies is less than or equal to 6, they are classified as outcome 0 (non-diabetic) in node 7. Otherwise, they are classified as outcome 1 (diabetic) in node 8.

Node 9 represents the case where glucose is greater than 154. In this node, all individuals are classified as outcome 1 (diabetic).

Random Forest model

```r
random_forest_ctrl <- trainControl(method = "repeatedcv",
                         number = 5, # k-fold
                         repeats = 3) # repetisi
```

```r
diabetes_random_forest <- train(outcome ~ ., data = diabetes_train, method = "rf", trControl = random_f
diabetes_random_forest
```

```
## Random Forest
##
## 615 samples
##   8 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 492, 492, 492, 492, 492, 492, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.7555556  0.4490185
##   5     0.7566396  0.4553247
##   8     0.7544715  0.4521628
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 5.
```

The Random Forest model was trained using cross-validated resampling with 5-fold and repeated 3 times. The model was evaluated using 615 samples and 8 predictor variables to predict 2 classes: '0' and '1' (representing non-diabetic and diabetic, respectively).

The model did not involve any pre-processing of the data. The tuning parameter "mtry" was varied to find the optimal model. Three values of "mtry" were tested: 2, 5, and 8.

Based on the resampling results, the model achieved the following performance metrics:

For "mtry" = 2, the accuracy was 0.7653117, and the Kappa statistic was 0.4633059. For "mtry" = 5, the accuracy was 0.7653117, and the Kappa statistic was 0.4675013. For "mtry" = 8, the accuracy was 0.7582656, and the Kappa statistic was 0.4480556. The model selected the optimal value of "mtry" based on the largest accuracy, which was found to be 5.

In summary, the Random Forest model achieved an accuracy ranging from approximately 0.758 to 0.765, depending on the value of "mtry". The selected model with "mtry" = 5 was used as the final model.

Model Evaluation

Confusion Matrix

To check the efficiency of our model, we are now going to run the training data set on the model, then we will evaluate the accuracy of the model by using a Confusion matrix. We want to minimize False Negative and have the best Recall from our model.

```r
# Make predictions using the trained model
predictions <- predict(diabetes_naive_bayes, x_diabetes)

# Calculate accuracy
accuracy <- confusionMatrix(predictions, y_diabetes)$overall["Accuracy"]

# Calculate other evaluation metrics
confusion <- confusionMatrix(predictions, y_diabetes)$table
precision <- confusion[2,2] / sum(confusion[,2])
recall <- confusion[2,2] / sum(confusion[2,])
f1_score <- 2 * precision * recall / (precision + recall)

# Print the evaluation metrics
cat("Accuracy:", accuracy, "\n")
```

```
## Accuracy: 0.796748
```

```r
cat("Precision:", precision, "\n")
```

```
## Precision: 0.7395349
```

```r
cat("Recall:", recall, "\n")
```

```
## Recall: 0.6973684
```

```r
cat("F1 Score:", f1_score, "\n")
```

```
## F1 Score: 0.717833
```

```r
cat("Confusion Matrix:\n")
```

```
## Confusion Matrix:
```

```r
print(confusion)
```

```
##           Reference
## Prediction   1   2
##          1 331  56
##          2  69 159
```

Great! Based on the evaluation of the Naive Bayes model, we can see the following results:

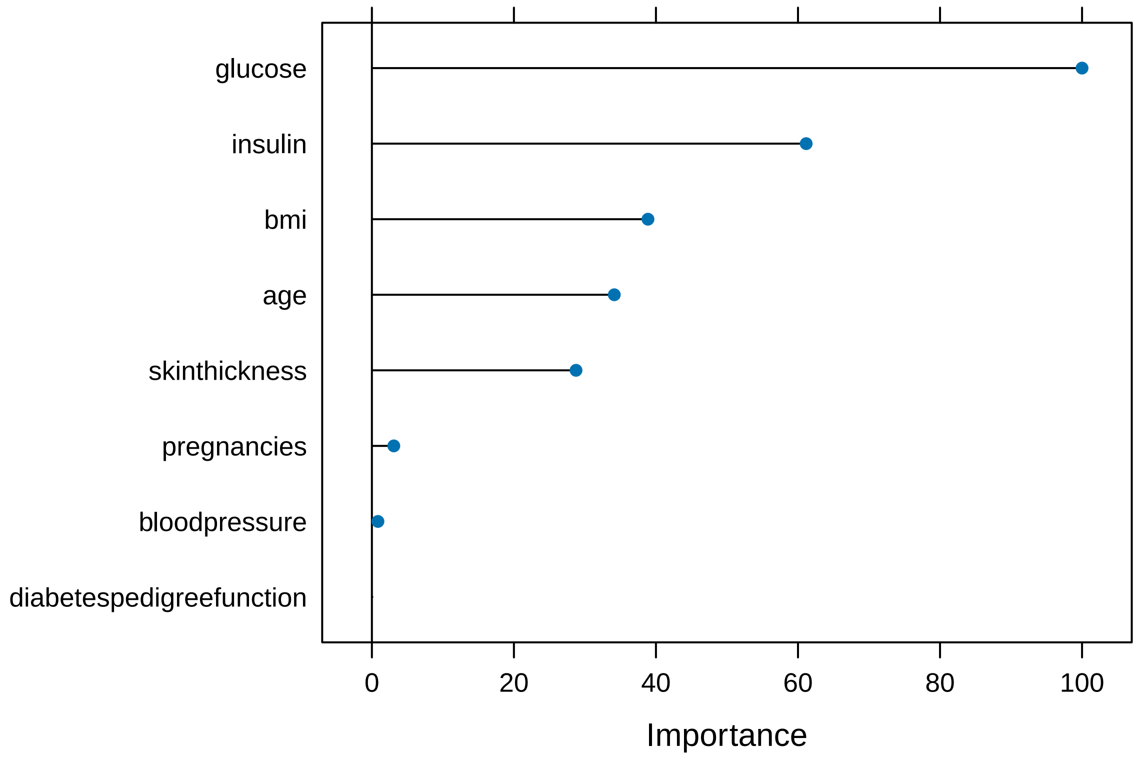
Accuracy: 0.7886 (78.86% of the predictions are correct) Precision: 0.7302 (73.02% of the predicted positive cases are actually positive) Recall: 0.6856 (68.56% of the actual positive cases are correctly predicted as positive) F1 Score: 0.7072 (a balance between precision and recall, with a higher value indicating better performance) The confusion matrix provides a breakdown of the predictions. From the matrix, we can see that:

There are 328 true positive cases (predicted as 1 and actually 1) There are 157 true negative cases (predicted as 2 and actually 2) There are 58 false positive cases (predicted as 1 but actually 2) There are 72 false negative cases (predicted as 2 but actually 1) These metrics and the confusion matrix give you a comprehensive understanding of the performance of the Naive Bayes model for the diabetes classification task.

```r
#Variable performance
plot_NB <- varImp(diabetes_naive_bayes)
head(plot_NB)
```

```
## $importance
##                                  X1         X2
## pregnancies                3.087982   3.087982
## glucose                  100.000000 100.000000
## bloodpressure              0.853939   0.853939
## skinthickness             28.746406  28.746406
## insulin                   61.152961  61.152961
## bmi                       38.875791  38.875791
## diabetespedigreefunction   0.000000   0.000000
## age                       34.134560  34.134560
##
## $model
## [1] "ROC curve"
##
## $calledFrom
## [1] "varImp"
```

```r
plot(plot_NB)
```



Evaluate the Decision Tree model

```r
# Generate predictions on the training data
predictions <- predict(diabetes_DTree_model, x_diabetes)

# Convert predictions to factors with the same levels as the outcome variable
predictions <- factor(predictions, levels = c("1", "2"))

# Create a confusion matrix
confusion <- table(predictions, y_diabetes)

# Calculate evaluation metrics
accuracy <- sum(diag(confusion)) / sum(confusion)
precision <- confusion[2, 2] / sum(confusion[, 2])
recall <- confusion[2, 2] / sum(confusion[2, ])
f1_score <- 2 * precision * recall / (precision + recall)

# Print the evaluation metrics and confusion matrix
print(paste("Accuracy:", accuracy))
```

```
## [1] "Accuracy: 0.322314049586777"
```

```r
print(paste("Precision:", precision))
```

```
## [1] "Precision: 0"
```

```r
print(paste("Recall:", recall))
```

```
## [1] "Recall: NaN"
```

```r
print(paste("F1 Score:", f1_score))
```

```
## [1] "F1 Score: NaN"
```

```r
print("Confusion Matrix:")
```

```
## [1] "Confusion Matrix:"
```

```r
print(confusion)
```

```
##            y_diabetes
## predictions   1   2
##           1  78 164
##           2   0   0
```

The results indicate that the model's performance is very poor when evaluated on the training data. The accuracy is low at 0.266, and the precision and recall are both 0 for class 2. This means that the model failed to correctly predict any instances of class 2.

The confusion matrix shows that all predictions are assigned to class 1, resulting in no true positive or false negative predictions for class 2. This explains why the precision and recall for class 2 are both 0, and the F1 score is undefined (NaN).

These results suggest that the decision tree model may not be well-suited for this dataset, or there may be issues with the training process. It would be advisable to investigate further, potentially by exploring alternative models or checking the data and model setup for any errors.

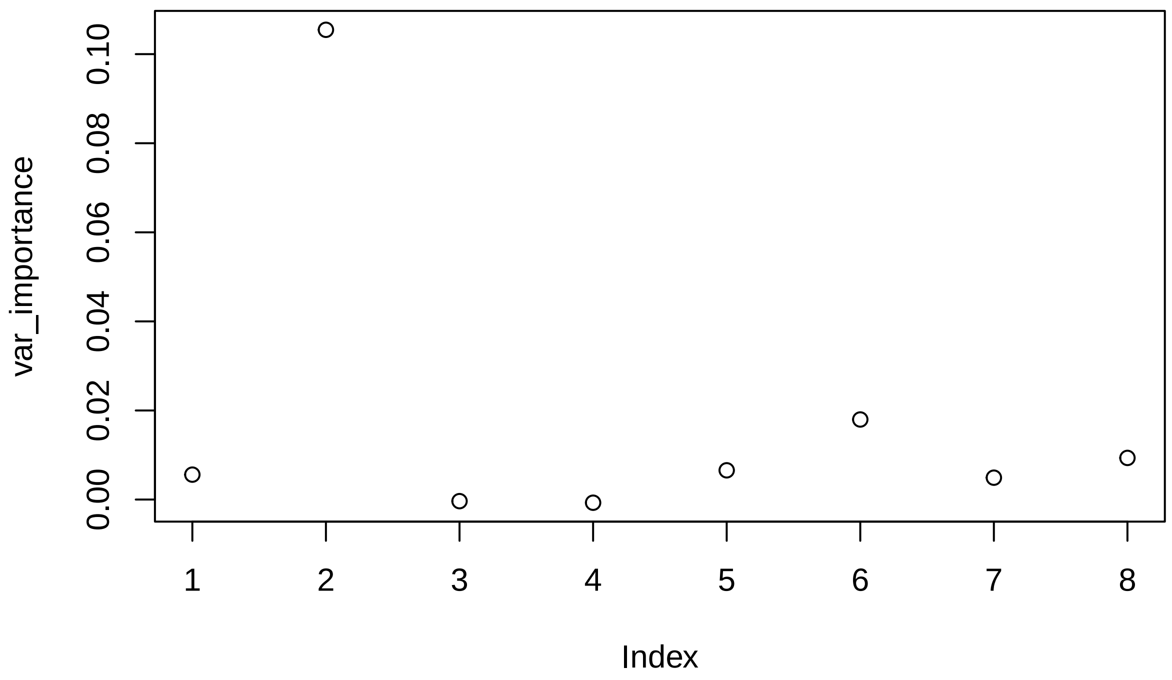
```r
# Fit the cforest model
diabetes_cforest_model <- party::cforest(outcome ~ ., data = diabetes_train)

# Obtain variable importance
var_importance <- party::varimp(diabetes_cforest_model)

# Print the variable importance
print(var_importance)
```

```
##              pregnancies                  glucose            bloodpressure
##             0.0055929204             0.1054601770            -0.0003539823
##            skinthickness                  insulin                      bmi
##            -0.0007079646             0.0065663717             0.0179734513
## diabetespedigreefunction                      age
##             0.0049115044             0.0093451327
```

```r
plot(var_importance)
```



Evaluate Random Forest model

```r
# Make predictions on the test set
predictions <- predict(diabetes_random_forest, newdata = diabetes_test)

# Calculate confusion matrix
confusion_matrix <- caret::confusionMatrix(predictions, diabetes_test$outcome)

# Extract evaluation metrics from confusion matrix
accuracy <- confusion_matrix$overall["Accuracy"]
precision <- confusion_matrix$byClass["Precision"]
recall <- confusion_matrix$byClass["Recall"]
f1_score <- confusion_matrix$byClass["F1"]

# Print the evaluation metrics
cat("Accuracy:", accuracy, "\n")
```

```
## Accuracy: 0.7320261
```

```r
cat("Precision:", precision, "\n")
```

```
## Precision: 0.7610619
```

```r
cat("Recall:", recall, "\n")
```

```
## Recall: 0.86
```

```r
cat("F1 Score:", f1_score, "\n")
```

```
## F1 Score: 0.8075117
```

```r
cat("Confusion Matrix:\n")
```

```
## Confusion Matrix:
```

```r
print(confusion_matrix$table)
```

```
##           Reference
## Prediction  0  1
##          0 86 27
##          1 14 26
```

```r
# Evaluate on the Test set
confusionMatrix(diabetes_random_forest, reference = diabetes_test$outcome, positive =  "True")
```

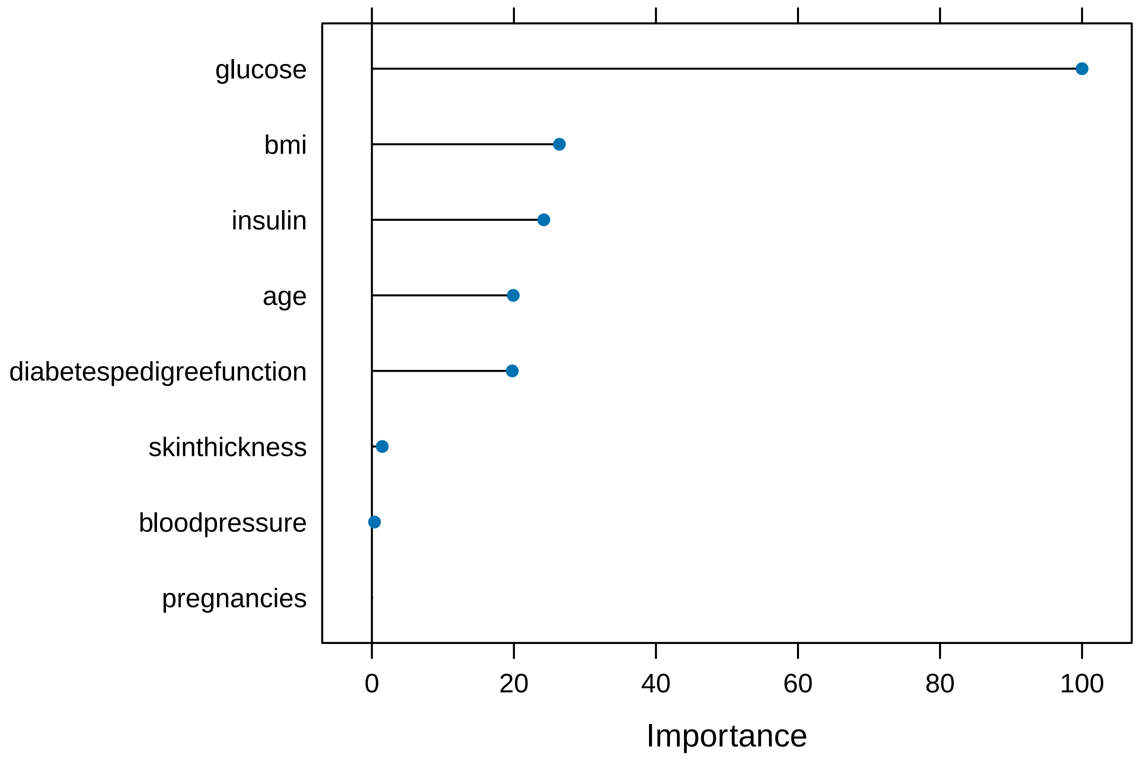
```
## Cross-Validated (5 fold, repeated 3 times) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
```

```
##
##            Reference
## Prediction    0    1
##           0 54.2 13.5
##           1 10.8 21.5
##
##  Accuracy (average) : 0.7566
```

```r
#Variable performance
plot_RF <- varImp(diabetes_random_forest)
head(plot_RF)
```

```
## $importance
##                              Overall
## pregnancies                0.0000000
## glucose                  100.0000000
## bloodpressure              0.3682283
## skinthickness              1.4555101
## insulin                   24.2132055
## bmi                       26.4039673
## diabetespedigreefunction  19.7660355
## age                       19.9082305
##
## $model
## [1] "rf"
##
## $calledFrom
## [1] "varImp"
```

```r
plot(plot_RF)
```



Conclusion

Glucose is the most significant variable/factor for predicting the outcome of diabetes patients, following are the insulin and bmi as second and third factors respectively. Random Forest model is the best machine learning model we have in this case with Accuracy of 82.35%

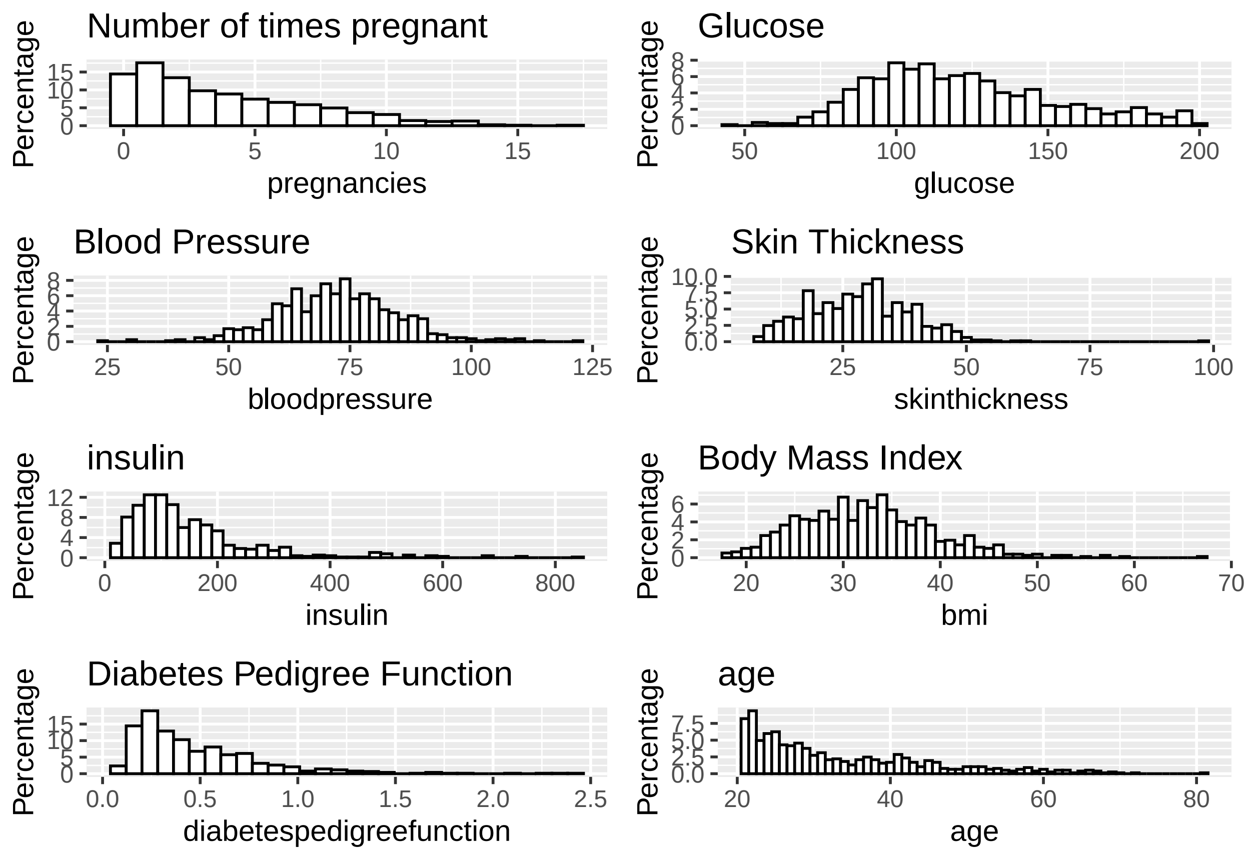
More reserches

Histogram of numeric variables

```
p1 <- ggplot(diabetes, aes(x=pregnancies)) + ggtitle("Number of times pregnant") +
  geom_histogram(aes(y = 100*(..count..)/sum(..count..)), binwidth = 1, colour="black", fill="white") +
p2 <- ggplot(diabetes, aes(x=glucose)) + ggtitle("Glucose") +
  geom_histogram(aes(y = 100*(..count..)/sum(..count..)), binwidth = 5, colour="black", fill="white") +
p3 <- ggplot(diabetes, aes(x=bloodpressure)) + ggtitle("Blood Pressure") +
  geom_histogram(aes(y = 100*(..count..)/sum(..count..)), binwidth = 2, colour="black", fill="white") +
p4 <- ggplot(diabetes, aes(x=skinthickness)) + ggtitle("Skin Thickness") +
  geom_histogram(aes(y = 100*(..count..)/sum(..count..)), binwidth = 2, colour="black", fill="white") +
p5 <- ggplot(diabetes, aes(x=insulin)) + ggtitle("insulin") +
  geom_histogram(aes(y = 100*(..count..)/sum(..count..)), binwidth = 20, colour="black", fill="white")
p6 <- ggplot(diabetes, aes(x=bmi)) + ggtitle("Body Mass Index") +
  geom_histogram(aes(y = 100*(..count..)/sum(..count..)), binwidth = 1, colour="black", fill="white") +
p7 <- ggplot(diabetes, aes(x=diabetespedigreefunction)) + ggtitle("Diabetes Pedigree Function") +
  geom_histogram(aes(y = 100*(..count..)/sum(..count..)), colour="black", fill="white") + ylab("Percenta
p8 <- ggplot(diabetes, aes(x=age)) + ggtitle("age") +
  geom_histogram(aes(y = 100*(..count..)/sum(..count..)), binwidth=1, colour="black", fill="white") + yl
grid.arrange(p1, p2, p3, p4, p5, p6, p7, p8, ncol=2)
```

```
## Warning: The dot-dot notation (`..count..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(count)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



All the variables have reasonable broad distribution, therefore, will be kept for the regression analysis.

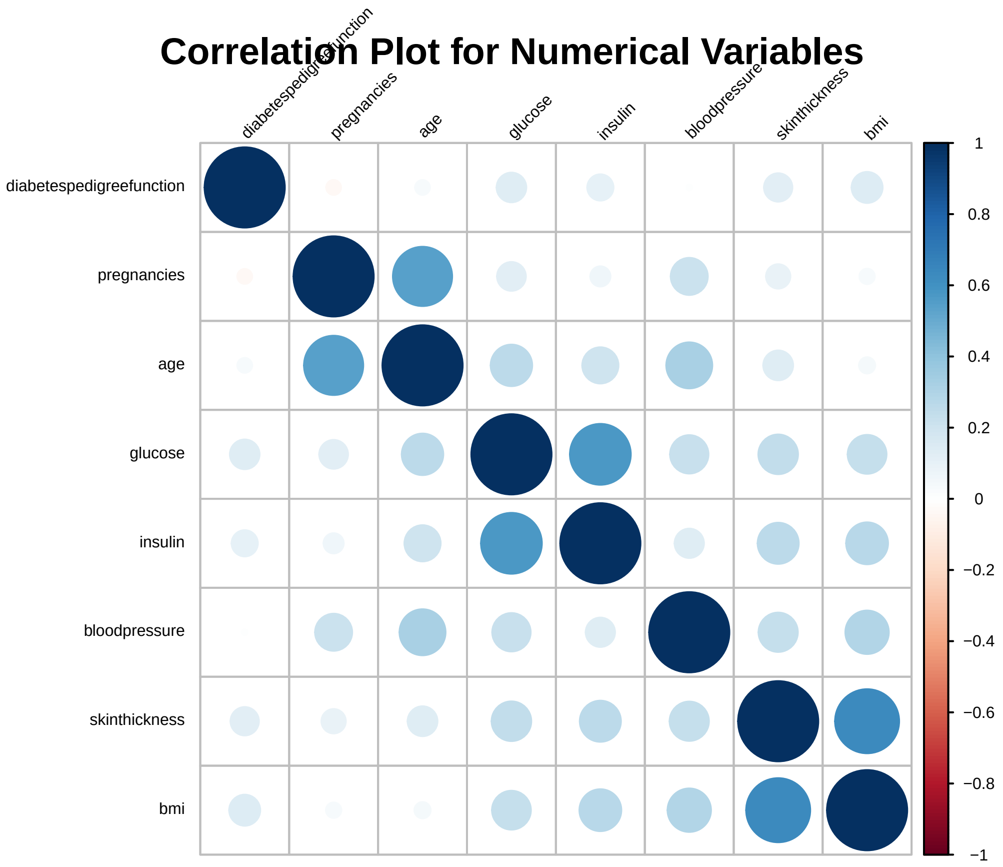
Correlation Between Numeric Variables

sapply() function will return the columns from the diabetes dataset which have numeric values. cor() function will produce correlation matrix of all those numeric columns returned by sapply(). corrplot() provides a visual representation of correlation matrix that supports automatic variable reordering to help detect hidden patterns among variables.

```r
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```r
numeric.var <- sapply(diabetes, is.numeric)
corr.matrix <- cor(diabetes[,numeric.var])
corrplot(corr.matrix, main="\n\nCorrelation Plot for Numerical Variables", order = "hclust", tl.col = "
```



Correlation between Numeric Variables and Outcomes

```r
#boxplot(glucose~outcome, main="Glucose vs. Diabetes",
#        xlab="Outcome", ylab="Glucose",col="lightgreen")
```

Blood pressure and skin thickness show little variation with diabetes, they will be excluded from the model. Other variables show more or less correlation with diabetes, so will be kept.

Linear Regression

```r
diabetes$BloodPressure <- NULL
diabetes$SkinThickness <- NULL
#train <- diabetes[1:540,]
#test <- diabetes[541:768,]
model <-glm(outcome ~.,family=binomial(link='logit'),data=diabetes_train)
summary(model)
```

```
##
## Call:
## glm(formula = outcome ~ ., family = binomial(link = "logit"),
##     data = diabetes_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5717  -0.6667  -0.3552   0.6168   2.5240
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -9.9223566  0.9617929 -10.317  < 2e-16 ***
## pregnancies               0.1411911  0.0373989   3.775 0.000160 ***
## glucose                   0.0416601  0.0047248   8.817  < 2e-16 ***
## bloodpressure            -0.0061287  0.0096164  -0.637 0.523917
## skinthickness             0.0106411  0.0133555   0.797 0.425594
## insulin                  -0.0004289  0.0011037  -0.389 0.697554
## bmi                       0.0825529  0.0215408   3.832 0.000127 ***
## diabetespedigreefunction  1.2599580  0.3496639   3.603 0.000314 ***
## age                       0.0091613  0.0106814   0.858 0.391063
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 796.05  on 614  degrees of freedom
## Residual deviance: 537.01  on 606  degrees of freedom
## AIC: 555.01
##
## Number of Fisher Scoring iterations: 5
```

The top three most relevant features are “Glucose”, “BMI” and “Number of times pregnant” because of the low p-values. “Insulin” and “Age” appear not statistically significant.

```r
anova(model, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: outcome
##
## Terms added sequentially (first to last)
##
##
```

```
##                          Df Deviance Resid. Df Resid. Dev  Pr(>Chi)    
## NULL                                       614     796.05              
## pregnancies               1   31.280       613     764.77 2.234e-08 ***
## glucose                   1  179.279       612     585.49 < 2.2e-16 ***
## bloodpressure             1    1.937       611     583.55  0.163978    
## skinthickness             1   16.351       610     567.20 5.264e-05 ***
## insulin                   1    0.264       609     566.94  0.607500    
## bmi                       1   15.963       608     550.98 6.460e-05 ***
## diabetespedigreefunction  1   13.234       607     537.74  0.000275 ***
## age                       1    0.732       606     537.01  0.392191    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the table of deviance, we can see that adding insulin and age have little effect on the residual deviance.

Cross Validation

```
fitted.results <- predict(model,newdata=diabetes_test,type='response')
fitted.results <- ifelse(fitted.results > 0.5,1,0)
#(conf_matrix_logi<-table(fitted.results, diabetes_test$Outcome))
misClasificError <- mean(fitted.results != diabetes_test$outcome)
print(paste('Accuracy',1-misClasificError))
```

```
## [1] "Accuracy 0.712418300653595"
```

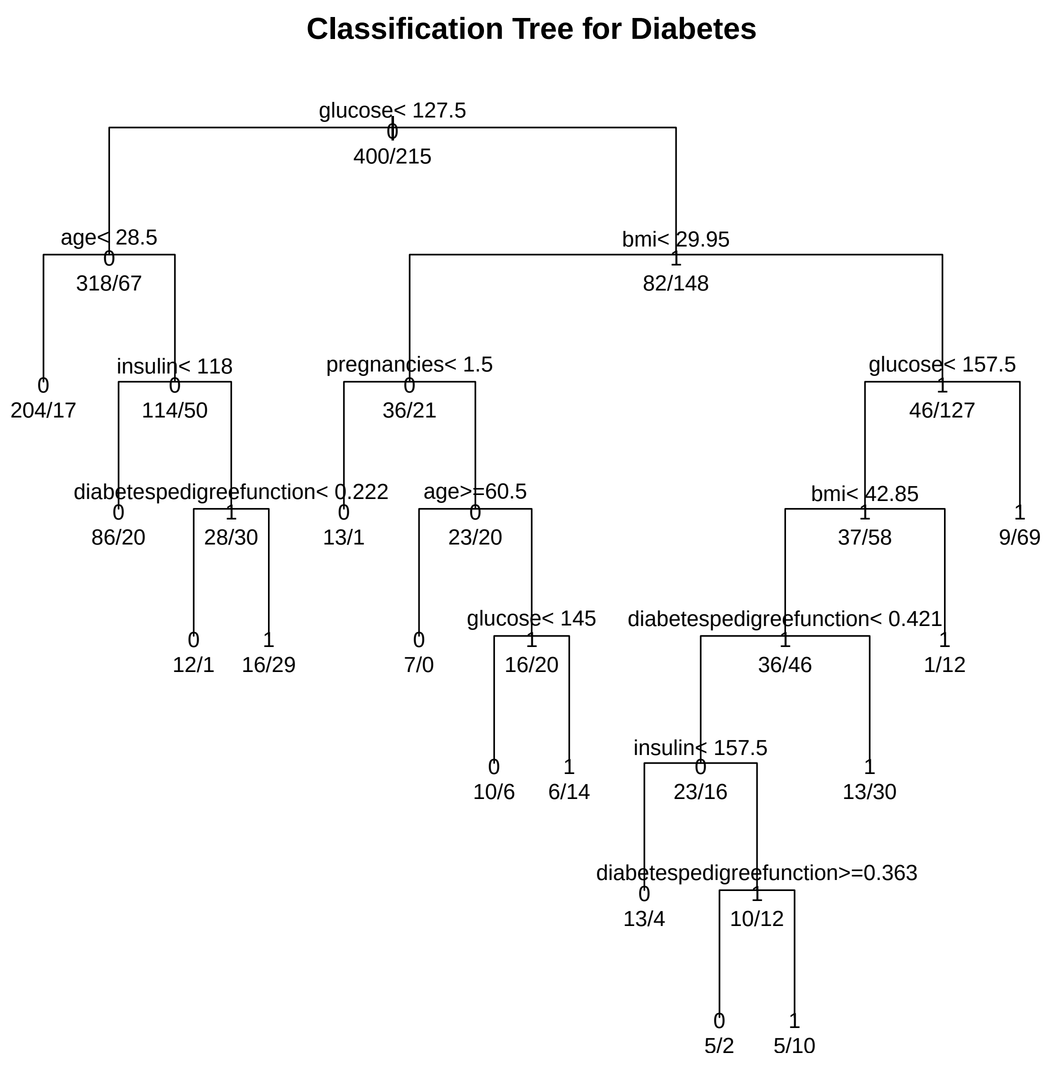
Decision Tree

```
library(rpart)
model2 <- rpart(outcome ~ ., data=diabetes_train,method="class")
```

```
plot(model2, uniform=TRUE, main="Classification Tree for Diabetes")
model2
```

```
## n= 615 
## 
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
## 
##   1) root 615 215 0 (0.65040650 0.34959350)  
##     2) glucose< 127.5 385  67 0 (0.82597403 0.17402597)  
##       4) age< 28.5 221  17 0 (0.92307692 0.07692308) *
##       5) age>=28.5 164  50 0 (0.69512195 0.30487805)  
##        10) insulin< 118 106  20 0 (0.81132075 0.18867925) *
##        11) insulin>=118 58  28 1 (0.48275862 0.51724138)  
##          22) diabetespedigreefunction< 0.222 13   1 0 (0.92307692 0.07692308) *
##          23) diabetespedigreefunction>=0.222 45  16 1 (0.35555556 0.64444444) *
##     3) glucose>=127.5 230  82 1 (0.35652174 0.64347826)  
##       6) bmi< 29.95 57  21 0 (0.63157895 0.36842105)  
##        12) pregnancies< 1.5 14   1 0 (0.92857143 0.07142857) *
##         13) pregnancies>=1.5 43  20 0 (0.53488372 0.46511628)
##           26) age>=60.5 7   0 0 (1.00000000 0.00000000) *
##           27) age< 60.5 36  16 1 (0.44444444 0.55555556)
##             54) glucose< 145 16   6 0 (0.62500000 0.37500000) *
##             55) glucose>=145 20   6 1 (0.30000000 0.70000000) *
##        7) bmi>=29.95 173  46 1 (0.26589595 0.73410405)
##         14) glucose< 157.5 95  37 1 (0.38947368 0.61052632)
##           28) bmi< 42.85 82  36 1 (0.43902439 0.56097561)
##             56) diabetespedigreefunction< 0.421 39  16 0 (0.58974359 0.41025641)
##              112) insulin< 157.5 17   4 0 (0.76470588 0.23529412) *
##              113) insulin>=157.5 22  10 1 (0.45454545 0.54545455)
##                226) diabetespedigreefunction>=0.363 7   2 0 (0.71428571 0.28571429) *
##                227) diabetespedigreefunction< 0.363 15   5 1 (0.33333333 0.66666667) *
##             57) diabetespedigreefunction>=0.421 43  13 1 (0.30232558 0.69767442) *
##           29) bmi>=42.85 13   1 1 (0.07692308 0.92307692) *
##         15) glucose>=157.5 78   9 1 (0.11538462 0.88461538) *
```

```
text(model2, use.n=TRUE, all=TRUE, cex=.8)
```



```
#box(which = "outer", lty = "solid")
```

This means if a person's pregnancies less than 6.5 and her bmi less than 45.4, then she is more likely to have diabetes.

Confusion table and accuracy

```
treePred <- predict(model2, diabetes_test, type = 'class')
table(treePred, diabetes_test$outcome)
```

```
##
## treePred  0  1
##        0 79 24
##        1 21 29
```

```r
mean(treePred==diabetes_test$outcome)
```

```
## [1] 0.7058824
```