A Comprehensive Guide to Naive Bayes in R

Resource

This guide follows this website.

```r
data_file_path <- '../data/diabetes.csv'
```

Introduction

What is Naive Bayes

Naive Bayes is a Supervised Machine Learning algorithm based on Bayes Theorem that is used to solve classification problems by following a probabilistic approach. It is based on the idea that the predictor variables in a Machine Learning model are independent of each other. Meaning that the outcome of a model depends on a set of independent variables that have nothing to do with one another.

But, why is it called `Naive`?

In real world problems, predictor variables are not always independent of each other, there are always some correlations between them. Since Naive Bayes considers each predictor variable to be independent of any other variable in the model, it is called `Naive`.

The Math behind Naive Bayes

The principle behind Naive Bayes is the Bayes Theorem also known as the Bayes Rule. The Bayes theorem is used to calculate the conditional probability, which is nothing but the probability of an event occurring based on information about the events in the past.

$P(A|B) = \frac{(P(B|A)P(A)}{P(B)}$

Formmally, the terminologies of the Bayes Theorem are as follows:

- A is known as the proposition and B is known as the evidence
- P(A) represents the prior probability of the proposition
- P(B) represents the prior probability of the evidence
- P(A|B) is called the posterior
- P(B|A) is called the likelihood

Therefore, Bayes theorem can be summed up as:

Posterior = (Likelihood) * (Proposition prior probability) / Evidence prior probability

Another way to state Bayes Theorem

Given a Hypothesis (H) and Evidence (E), Bayes theorem states that the relationship between the probability of Hypothesis before getting the evidence P(H) and the probability of the hypothesis after getting the evidence P(H|E) is:

$P(H|E) = \frac{P(E|H)P(H)}{P(E)}$

How does Naive Bayes Work?

To get a better understanding of how Naive Bayes works, let's look at an example.

Consider a data set with 1500 observations and the following output classes:

- Cat
- Parrot
- Turtle

The predictor variables are categorical in nature, *i.e.*, they store two values, either `True` or `False`:

- Swim
- Wings
- Green Color
- Sharp Teeth

```
data.frame(
  Type = c('Cat', 'Parrot', 'Turtle'),
  Swim = c('450/500', '50/500', '500/500'),
  Wings = c('0', '500/500', '0'),
  `Green Color` = c('0', '400/500', '100/500'),
  `Sharp Teeth` = c('500/500', '0', '50/500')
)
```

```
##     Type    Swim   Wings Green.Color Sharp.Teeth
## 1    Cat 450/500       0           0     500/500
## 2 Parrot  50/500 500/500     400/500           0
## 3 Turtle 500/500       0     100/500      50/500
```

From the above table, we can summarize that:

The class of type Cat shows that:

- Out of 500, 450 (90%) cats can swim
- 0 number of cats have wings
- 0 number of cats are of Green color
- All 500 cats have sharp teeth

The class of type Parrot shows that:

- 50 (10%) parrots have a true value for swim
- All 500 parrots have wings
- Out of 500, 400 (80%) parrots are green in color
- No parrots have sharp teeth

The class of type Turtle shows that:

- All 500 turtles can swim
- No turtles have wings
- Out of 500, 100 (20%) are green in color
- Out of 500, 50 (10%) have sharp teeth

Now, with the available data, let's classify the following observation into one of the four output classes (Cats, Parrots, Turtles) by using a Naive Bayes Classifier:

```
data.frame(
  ID = 'New Observation',
  Swim = TRUE,
  Wings = FALSE,
  Green = TRUE,
  `Sharp Teeth` = FALSE
)
```

```
##                   ID Swim Wings Green Sharp.Teeth
## 1 New Observation TRUE FALSE  TRUE       FALSE
```

The goal here is to predict whether the animal is a Cat, Parrot, or Turtle based on the defined predictor variables (swim, wings, green color, sharp teeth).

To solve this, we will use the Naive Bayes approach, $P(H|MultipleEvidences) = \frac{P(C_1|H)*P(C_2|H)...P(C_n|H)P(H)}{P(MultipleEvidences)}$

In the observation, the variables Swim and Green are true and the outcome cqn be any one of the animals (Cat, Parrot, Turtle).

To check that the animal is a cat:

$P(Cat|Swim, Green) = P(Swim|Cat) * P(Green|Cat) * P(Cat)/P(Swim, Green)$ $P(Cat|Swim, Green) = 0.9 * 0 * 0.333/P(Swim, Green)$ $P(Cat|Swim, Green) = 0$

To check that the animal is a parrot:

$P(Parrot|Swim, Green) = P(Swim|Parrot) * P(Green|Parrot) * P(Parrot)/P(Swim, Green)$ $P(Parrot|Swim, Green) = 0.1*0.8*0.333/P(Swim, Green)$ $P(Parrot|Swim, Green) = 0.02664/P(Swim, Green)$

To check that the animal is a turtle:

$P(Turtle|Swim, Green) = P(Swim|Turtle)*P(Green|Turtle)*P(Turtle)/P(Swim, Green)$ $P(Turtle|Swim, Green) = 1.0 * 0.2 * 0.333/P(Swim, Green)$ $P(Turtle|Swim, Green) = 0.066/P(Swim, Green)$

For all the above calculations, the denominator is the same, ie. P(Swim, Green). The value of P(Turtle|Swim, Green) is greater than P(Parrot|Swim, Green), therefore we can correctly predict the class of the animal as turtle.

Now, let's see how you can implement Naive Bayes using the R language.

Practical Implementation of Naive Bayes in R

Problem Statement: To study a Diabetes data set and build a Machine Learning model that predicts whether or not a person has Diabetes.

Data Set Description: The given data set contains 100% of observations of patients along with their health details. Here's a list of the predictor variables that will help us classify a patient as either Diabetic or Normal:

- Pregnancies: Number of pregnancies so far.
- Glucose: Plasma glucose concentation
- BloodPressure: Diastolic blood pressure (mm Hg)
- SkinThickness: Triceps skin fold thickness (mm)
- Insulin: 2-Hour serum insulin (mu U/ml)
- BMI: Body mass index (weight in kg/(height in m)^2)
- DiabetesPedigreeFunction: Diabetes pedigree function
- Age: Age (years)

The response variable or the output variable is:

- Outcome: Class variable (0 or 1)

Objective: To build a Naive Bayes model in order to classify patients as either Diabetic or normal by studying their medical records such as Glucose level, age, BMI, etc.

Now that you know the objective of this demo, let's get our brains working and start coding.

Step 1: Import the required packages

Step 2: Import the data set

```
df <- read.csv(data_file_path)
```

Before we study the data set, let's convert the output variable (`"Outcome"`) into a categorical variable. This is necessary because our output will be in the form of 2 classes, `TRUE` or `FALSE`. Where `TRUE` denotes that a patient has diabetes and `FALSE` indicates that the person is diabetes-free.

```
df$Outcome <- factor(df$Outcome, levels = c(0, 1), labels = c('False', 'True'))
```

Step 3: Study the data set

```
str(df)
```

```
## '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 "False","True": 2 1 2 1 2 1 2 1 2 2 ...
```

```
df %>% head(10)
```

```
##    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
## 7            3      78            50            32      88 31.0
## 8           10     115             0             0       0 35.3
## 9            2     197            70            45     543 30.5
## 10           8     125            96             0       0  0.0
##    DiabetesPedigreeFunction Age Outcome
## 1                     0.627  50    True
## 2                     0.351  31   False
## 3                     0.672  32    True
## 4                     0.167  21   False
## 5                     2.288  33    True
## 6                     0.201  30   False
## 7                     0.248  26    True
## 8                     0.134  29   False
## 9                     0.158  53    True
## 10                    0.232  54    True
```

```
describe(df)
```

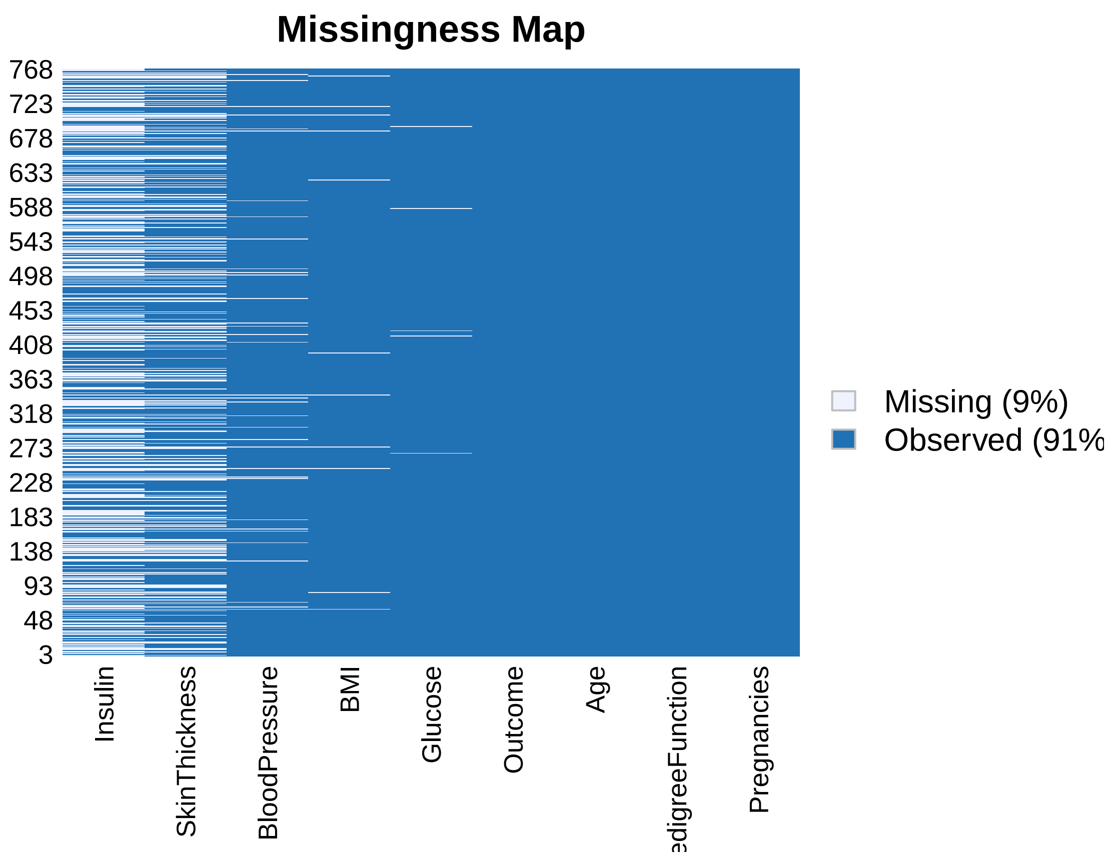
```
##                          vars   n   mean     sd median trimmed   mad   min
## Pregnancies                 1 768   3.85   3.37   3.00    3.46  2.97  0.00
## Glucose                     2 768 120.89  31.97 117.00  119.38 29.65  0.00
## BloodPressure               3 768  69.11  19.36  72.00   71.36 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
```

Step 4: Data cleaning

While analyzing the structure of the data set, we can see that the minimum values for Glucose, BloodPressure, SkinThickness, Insulin, and BMI are all zero. This is not ideal since no one can have a value of zero for Glucose, blood pressure, etc. Therefore, such values are treated as missing observations.

```
df[, 2:7][df[, 2:7] == 0] <- NA
```

```
missmap(df)
```

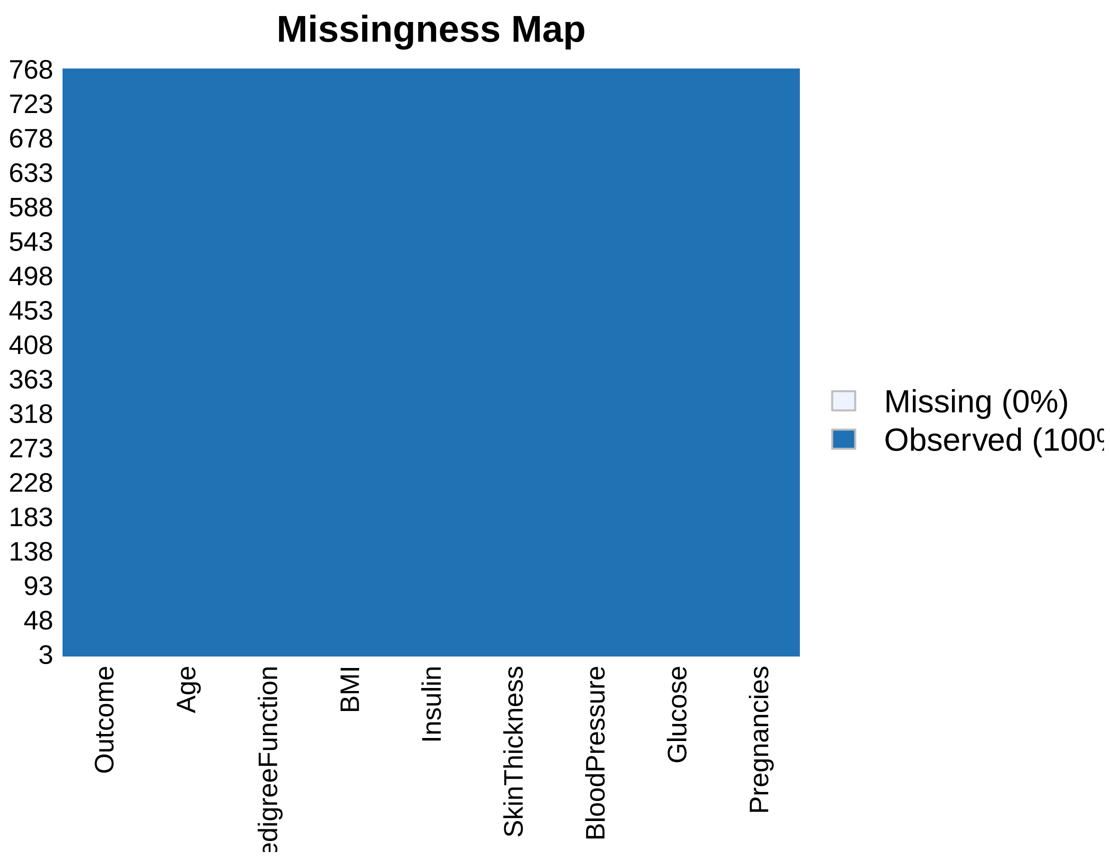


The above plot shows that our data set has plenty of missing values and removing all of them will leave us with an even smaller data set, therefore, we can perform imputations by using the *mice* package in R.

```
df_complete <- df %>%
  mutate(
    Glucose = mice_complete$Glucose,
    BloodPressure = mice_complete$BloodPressure,
    SkinThickness = mice_complete$SkinThickness,
    Insulin = mice_complete$Insulin,
    BMI = mice_complete$BMI)
```

Check to see if there are any missing values:

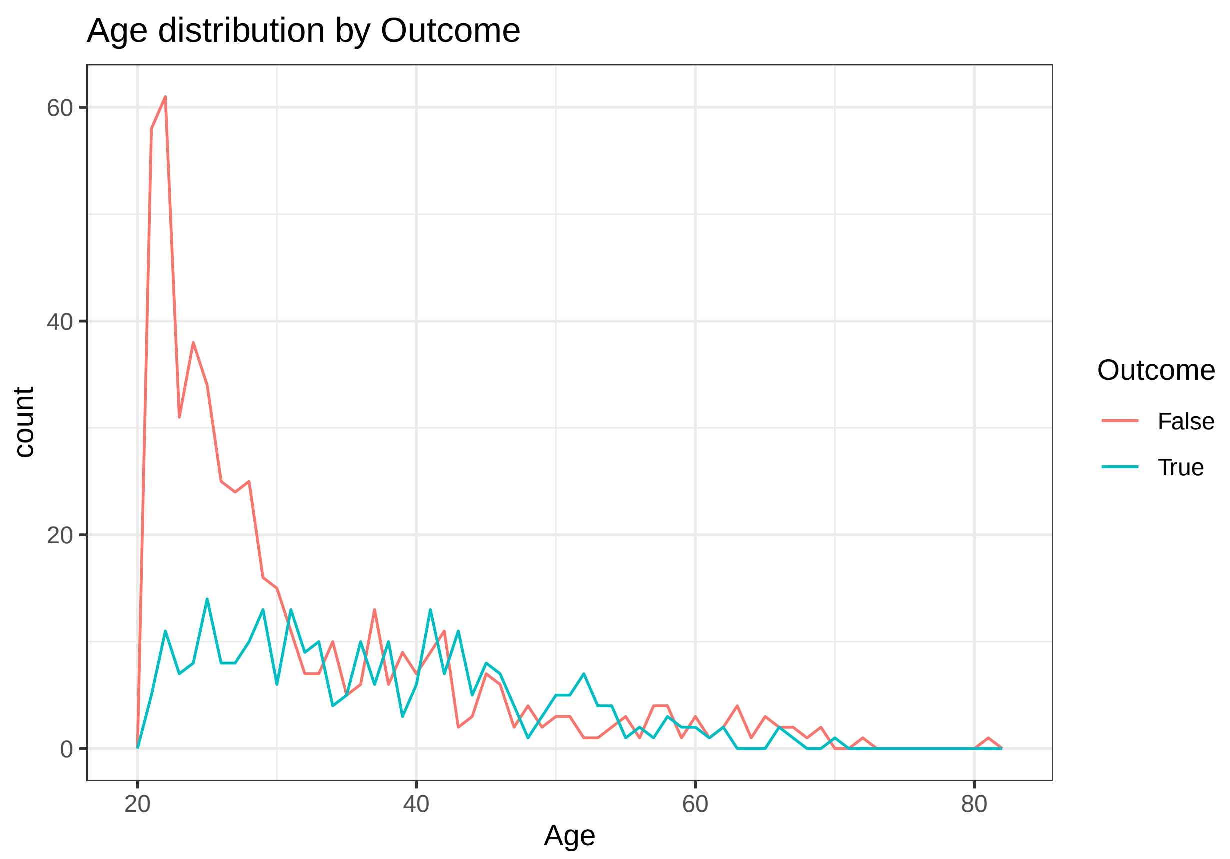
```
missmap(df_complete)
```



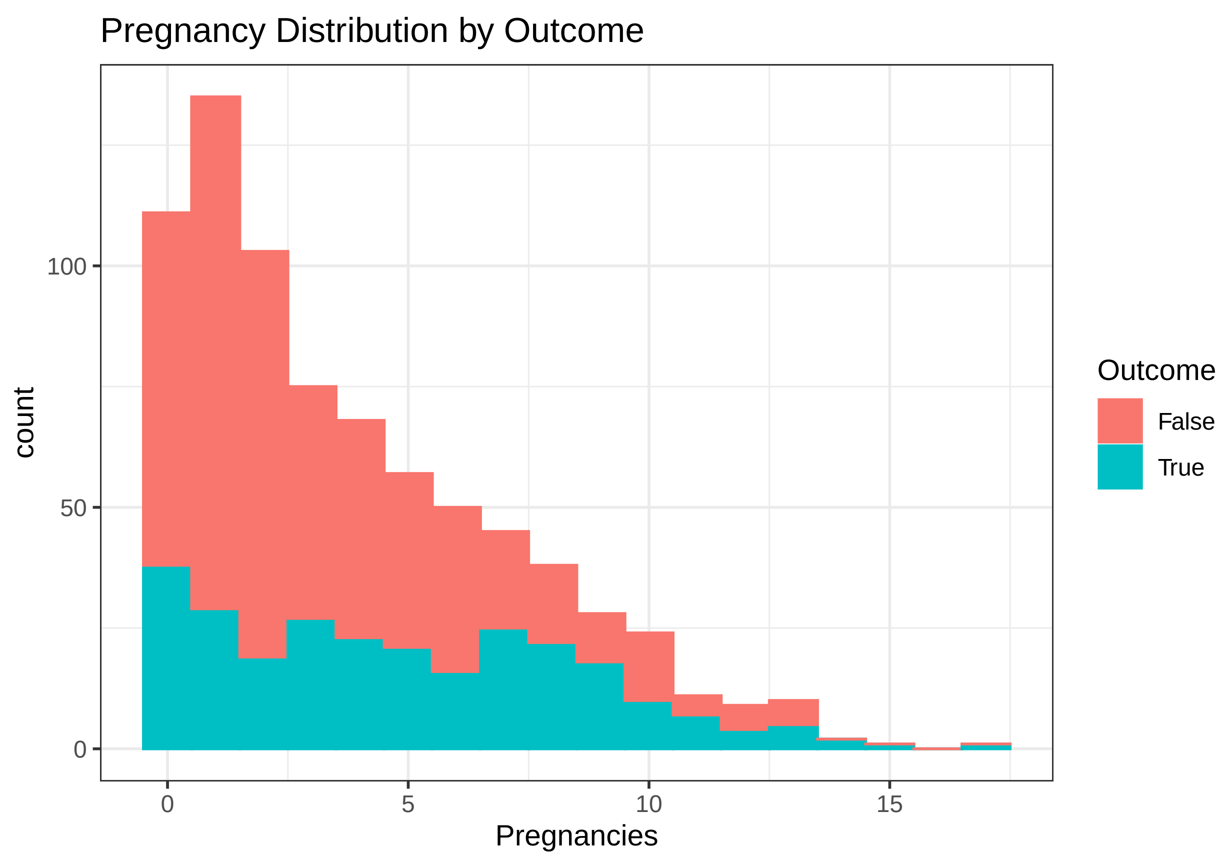
Exploratory Data Analysis

Now, let's perform a couple of visualizations to take a better look at each variable, this stage is essential to understanding the significance of each predictor variable.

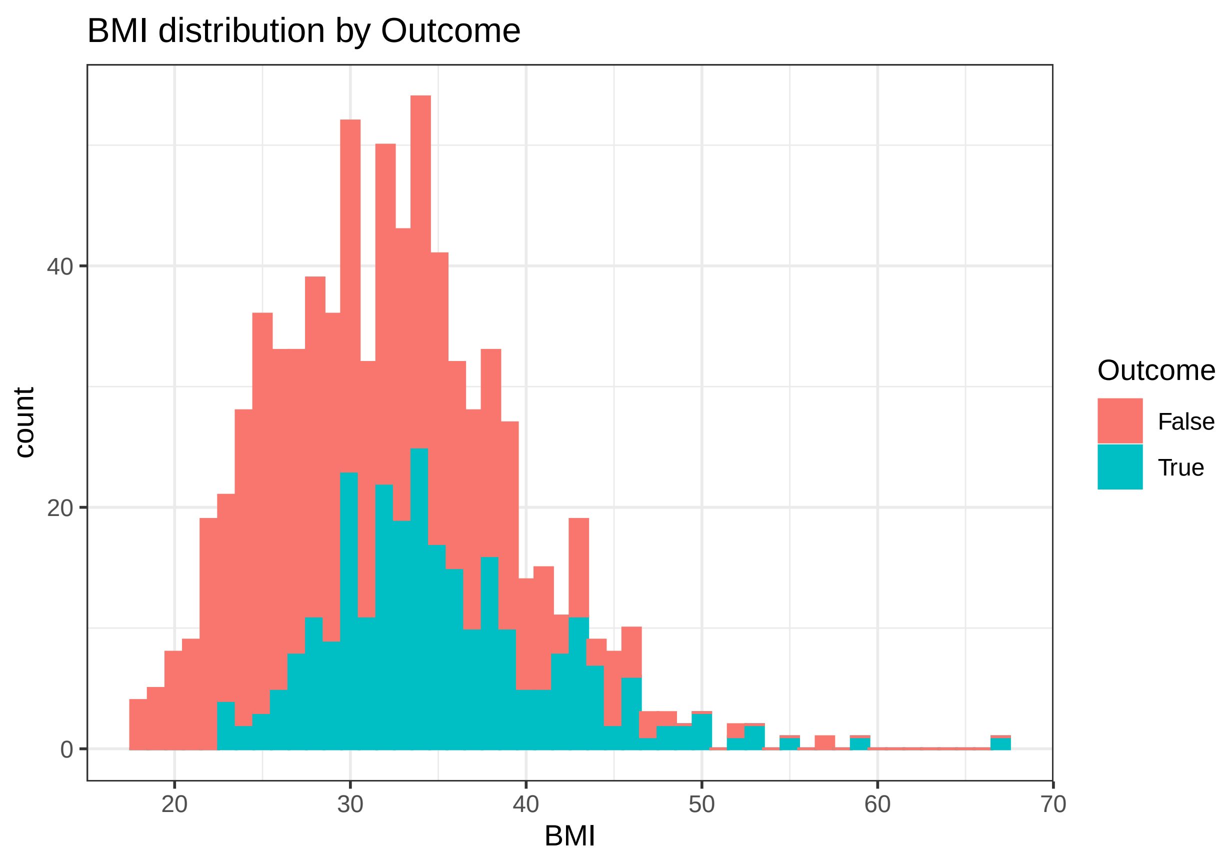
```
ggplot(df_complete, aes(Age, color = Outcome)) +
  geom_freqpoly(binwidth = 1) +
  labs(title = 'Age distribution by Outcome') +
  theme_bw()
```



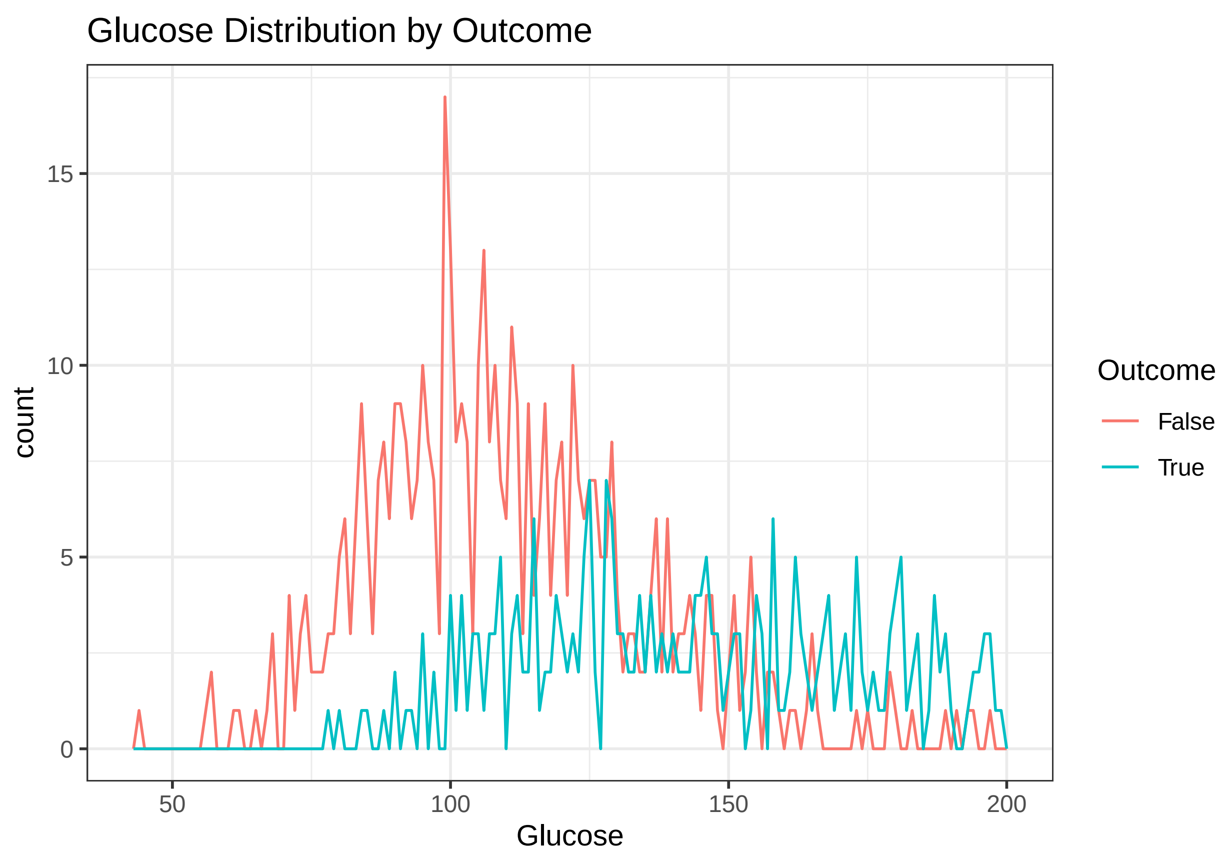
```r
ggplot(df_complete, aes(Pregnancies, fill = Outcome, colour = Outcome)) +
  geom_histogram(binwidth = 1) +
  labs(title = 'Pregnancy Distribution by Outcome') +
  theme_bw()
```



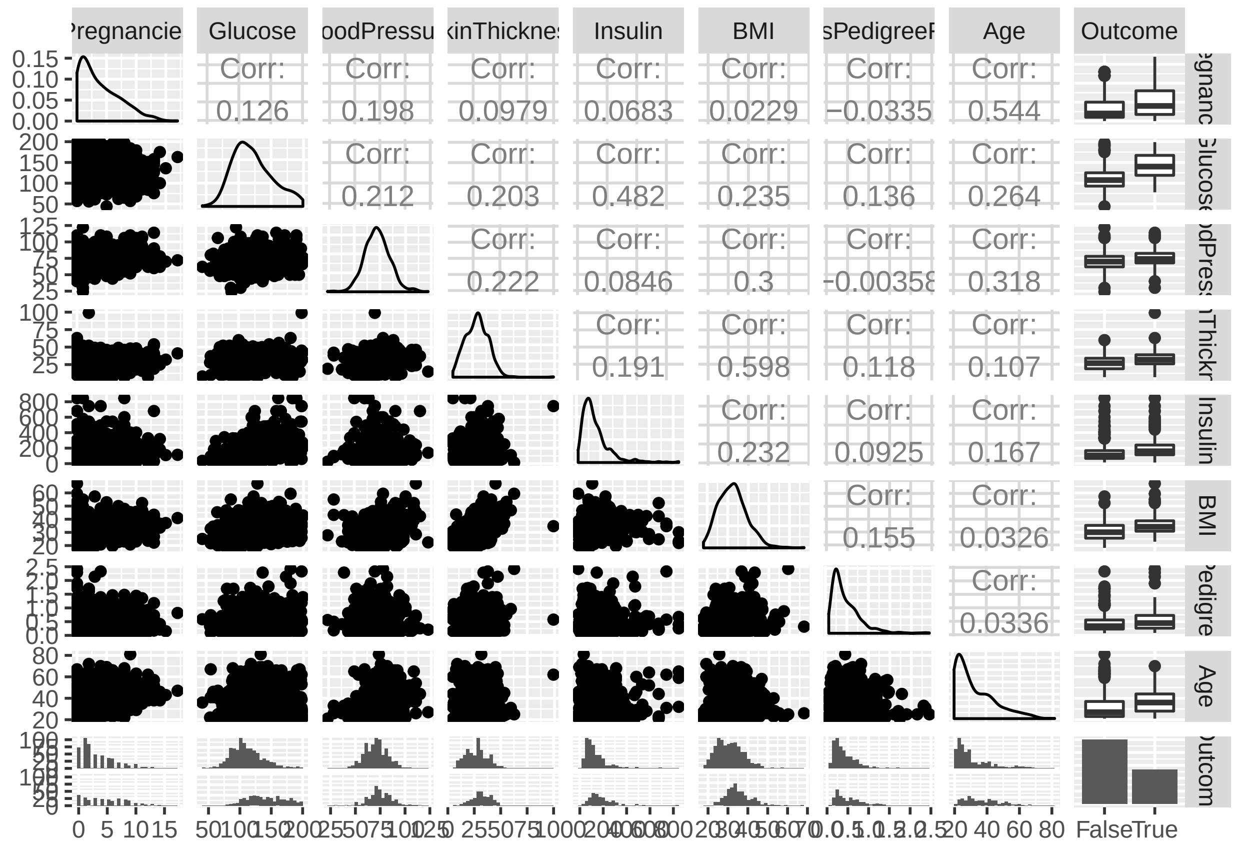
```r
ggplot(df_complete, aes(BMI, fill = Outcome, colour = Outcome)) +
  geom_histogram(binwidth = 1) +
  labs(title = 'BMI distribution by Outcome') +
  theme_bw()
```



```r
ggplot(df_complete, aes(Glucose, colour = Outcome)) +
  geom_freqpoly(binwidth = 1) +
  labs(title = 'Glucose Distribution by Outcome') +
  theme_bw()
```



```r
ggpairs(df_complete)
```



Step 6: Data Modeling

This stage begins with a process called *Data Splicing*, wherein the data set is split into two parts:

- **Training set:** This part of the data set is used to build and train the Machine Learning model.
- **Test set:** This part of the data set is used to evaluate the efficiency of the model.

```
indexTrain <- createDataPartition(df_complete$Outcome, p = 0.75, list = FALSE)

df_train <- df_complete[indexTrain,]
df_test <- df_complete[-indexTrain,]
```

```
prop.table(table(df_complete$Outcome)) * 100
```

```
##
##    False     True
## 65.10417 34.89583
```

```
prop.table(table(df_train$Outcome)) * 100
```

```
##
##    False     True
## 65.10417 34.89583
```

```
prop.table(table(df_test$Outcome)) * 100
```

```
##
##    False     True
## 65.10417 34.89583
```

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

```r
## Create objects `x` which holds the predictor variables and `y` which holds the response variables
x <- df_train[,-9]
y <- df_train$Outcome
```

Now, it's time to load the `e1071` package that holds the Naive Bayes function. This is a built-in function provided by R.

```r
library(e1071)
```

After loading the package, the below script will create Naive Bayes model by using the training data set:

```r
nbm <- train(x, y, 'nb', trControl = trainControl(method = 'cv', number = 10))

nbm
```

```
## Naive Bayes
##
## 576 samples
##   8 predictor
##   2 classes: 'False', 'True'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 518, 519, 518, 517, 518, 518, ...
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy  Kappa
##   FALSE      0.756975  0.4620989
##    TRUE      0.743328  0.4476797
##
## Tuning parameter 'fL' 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 fL = 0, usekernel = FALSE
##  and adjust = 1.
```

We have now created a predictive classification model by using the Naive Bayes Classifier.

Step 7: Model Evaluation

To check the efficiency of the model, we are now going to run the testing data set on the model, after which we will evaluate the accuracy of the model by using a Confusion Matrix.

```r
## Predict testing data set
Predict <- predict(nbm, newdata = df_test)

## Get the confusion matrix to see accuracy value and other parameter values
conf_matrix <- confusionMatrix(Predict, df_test$Outcome)
conf_matrix
```

```
## Confusion Matrix and Statistics
```

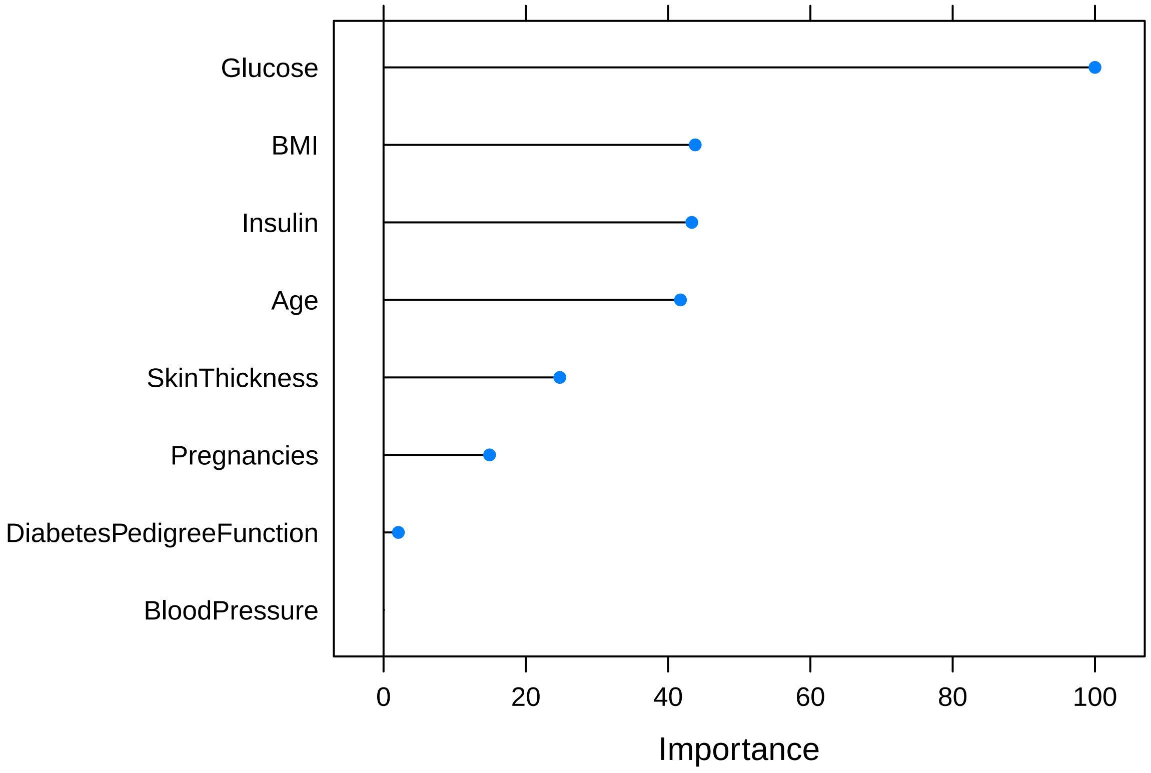
```
##
##            Reference
## Prediction False True
##      False   101   29
##      True     24   38
##
##                Accuracy : 0.724
##                  95% CI : (0.655, 0.7859)
##     No Information Rate : 0.651
##     P-Value [Acc > NIR] : 0.0191
##
##                   Kappa : 0.3818
##
##  Mcnemar's Test P-Value : 0.5827
##
##             Sensitivity : 0.8080
##             Specificity : 0.5672
##          Pos Pred Value : 0.7769
##          Neg Pred Value : 0.6129
##              Prevalence : 0.6510
##          Detection Rate : 0.5260
##    Detection Prevalence : 0.6771
##       Balanced Accuracy : 0.6876
##
##        'Positive' Class : False
##
```

```
cat(paste0('The final output shows that we built a Naive Bayes classifier that can predict whether a pe
```

```
## The final output shows that we built a Naive Bayes
classifier that can predict whether a person is diabetic or
not, with a accuracy of approximately 72%
```

To summarize the demo, let's draw a plot that shows how each predictor variable is independently responsible for predicting the outcome.

```
## Plot variable performance
X <- varImp(nbm)
plot(X)
```



From the above illustration, it is clear that `Glucose` is the most significant variable for predicting the outcome.

Now that you know how Naive Bayes works, I'm sure you're curious to learn more about the various Machine Learning algorithms. Here's a list of blogs on Machine Learning Algorithms:

- Linear Regression
- Logistic Regression
- Support Vector Machine
- Decision Trees
- Random Forest
- K-Means