Maternity Health Risk - Classification System

Ernesto Ferrer Mena

2023-10-25

Introduction

Evaluating the risk during pregnancy aids in anticipating the likelihood of adverse health events in women, empowering healthcare providers to deliver perinatal care tailored to the assessed level of risk offering significant improvements in several birth outcomes metrics used in Quality Departments in health insurance companies. Metrics like prenatal births, C-Sections, infant mortality, etc. reducing liquidated damages for such companies. The application of machine learning algorithms can provide significant advantages for patients, providers and insurance companies.

Overview

Rural areas are the most impacted on several health indicators because of lack of healthcare services when comparing with urban areas within the same country. Prenatal services are not an exception. The dataset selected for the risk classification was *"Maternal Health Risk"* by Marzia Ahmed with data collected in the rural areas of Bangladesh, donated on 8/14/2023.

Dataset citation: Ahmed,Marzia. (2023). Maternal Health Risk. UCI Machine Learning Repository. https://doi.org/10.24432/C5DP5D. Follow the link for more details about the dataset.

The dataset has 7 variables and 1014 observations with important data about the patients. I went into a more detailed description about it in the section Analysis. Three different methods were tested and finally I recommended the one with the best results.

Executive Summary

Analysis

Exploratory Analysis

"Maternity Health Risk" has 1014 observations, six features, and the class column called *"RiskLevel"*. Those features are Age as *"Age"*, Systolic Blood Pressure as *"SystolicBP"*, Diastolic Blood Pressure as *"DiastolicBP"*, Sugar in Blood as *"BS"*, Body Temperature as *"BodyTemp"*, Heart Rate as *"HeartRate"*, and the Risk Level as *"RiskLevel"*. Here is a preview of the dataset:

Age	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate	RiskLevel
25	130	80	15.00	98	86	high risk
35	140	90	13.00	98	70	high risk
29	90	70	8.00	100	80	high risk
35	120	60	6.10	98	76	low risk
23	130	70	7.01	98	78	mid risk

Age	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate	RiskLevel
35	85	60	11.00	102	86	high risk

With *"summay()"* function we can get a first approach of each of the six features in the dataset. We can see the minimum and maximum values, the median and important quartiles:

Age	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate
Min. :10.00	Min. : 70	Min. : 49.00	Min. : 6.000	Min. : 98.00	Min. : 7.00
1st Qu.:19.00	1st Qu.: 98	1st Qu.: 65.00	1st Qu.: 6.900	1st Qu.: 98.00	1st Qu.:70.00
Median :25.00	Median :120	Median : 80.00	Median : 7.500	Median : 98.00	Median :76.00
Mean :29.72	Mean :113	Mean : 76.35	Mean : 8.632	Mean : 98.68	Mean :74.27
3rd Qu.:38.25	3rd Qu.:120	3rd Qu.: 90.00	3rd Qu.: 8.000	3rd Qu.: 98.00	3rd Qu.:80.00
Max. :66.00	Max. :160	Max. :100.00	Max. :19.000	Max. :103.00	Max. :90.00

Observe what happens with each feature once we divide the dataset by *RiskLevel* and summarize them:

Summary for *RiskLevel = "high risk"*:

Age	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate
Min. :12.00	Min. : 83.0	Min. : 60.00	Min. : 6.10	Min. : 98.00	Min. :60.00
1st Qu.:25.00	1st Qu.:120.0	1st Qu.: 75.00	1st Qu.: 7.90	1st Qu.: 98.00	1st Qu.:70.00
Median :35.00	Median :120.0	Median : 90.00	Median :11.00	Median : 98.00	Median :77.00
Mean :35.71	Mean :123.2	Mean : 84.62	Mean :11.86	Mean : 98.93	Mean :76.74
3rd Qu.:48.00	3rd Qu.:140.0	3rd Qu.:100.00	3rd Qu.:15.00	3rd Qu.:100.50	3rd Qu.:86.00
Max. :65.00	Max. :160.0	Max. :100.00	Max. :19.00	Max. :103.00	Max. :90.00

Summary for *RiskLevel = "mid risk"*:

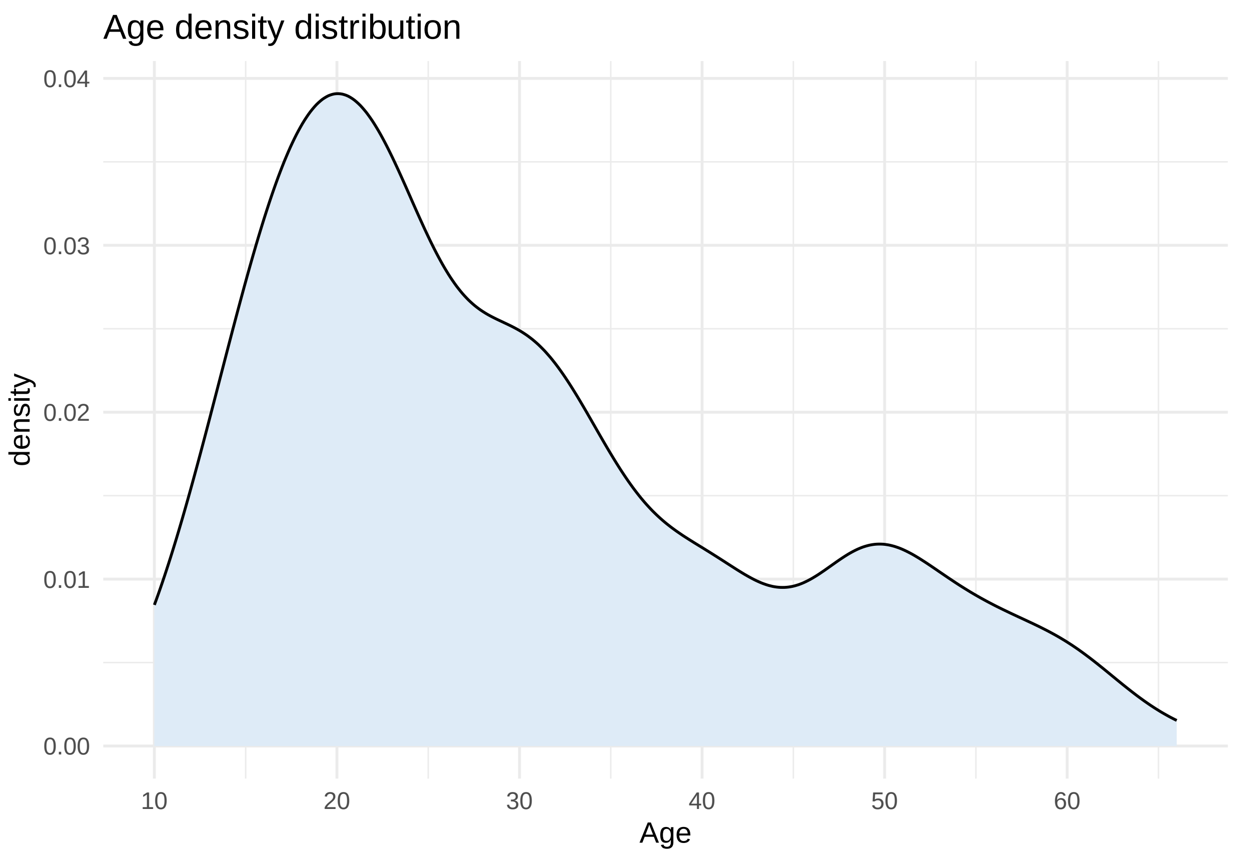
Age	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate
Min. :10.00	Min. : 70.0	Min. : 50.00	Min. : 6.000	Min. : 98.00	Min. :60.00
1st Qu.:19.00	1st Qu.:100.0	1st Qu.: 65.00	1st Qu.: 6.800	1st Qu.: 98.00	1st Qu.:70.00
Median :25.00	Median :120.0	Median : 75.00	Median : 7.000	Median : 98.00	Median :76.00
Mean :28.53	Mean :113.2	Mean : 74.06	Mean : 7.702	Mean : 98.79	Mean :74.13
3rd Qu.:32.00	3rd Qu.:120.0	3rd Qu.: 80.00	3rd Qu.: 7.800	3rd Qu.:100.00	3rd Qu.:78.00
Max. :60.00	Max. :140.0	Max. :100.00	Max. :18.000	Max. :102.00	Max. :88.00

Summary for *RiskLevel = "low risk"*:

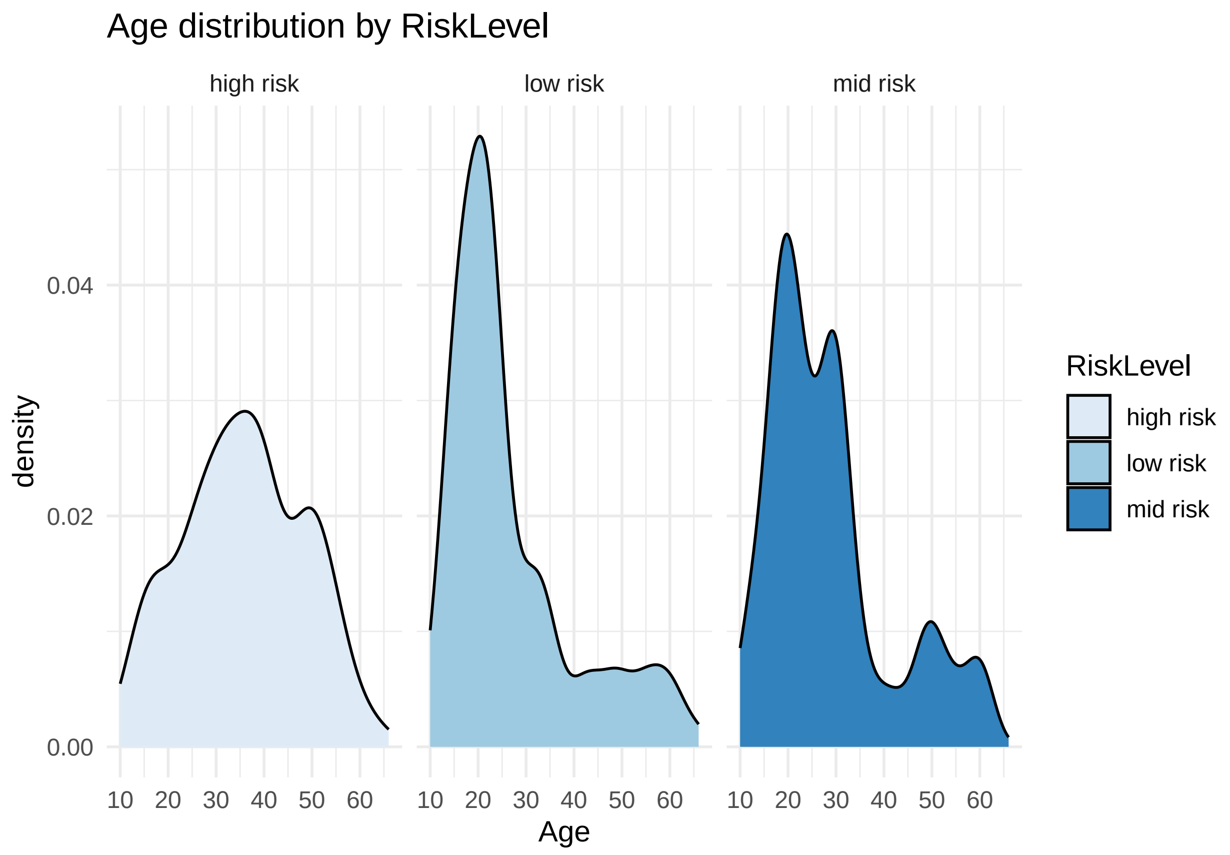
Age	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate
Min. :10.00	Min. : 70	Min. : 49.00	Min. : 6.000	Min. : 98.00	Min. : 7.00
1st Qu.:17.75	1st Qu.: 90	1st Qu.: 60.00	1st Qu.: 6.900	1st Qu.: 98.00	1st Qu.:70.00
Median :22.00	Median :120	Median : 75.00	Median : 7.500	Median : 98.00	Median :70.00
Mean :26.69	Mean :106	Mean : 72.73	Mean : 7.247	Mean : 98.41	Mean :72.74
3rd Qu.:32.00	3rd Qu.:120	3rd Qu.: 80.00	3rd Qu.: 7.500	3rd Qu.: 98.00	3rd Qu.:77.00
Max. :66.00	Max. :129	Max. :100.00	Max. :11.000	Max. :103.00	Max. :88.00

As you can see, the mean decreases for each feature when the risk level also decreases.

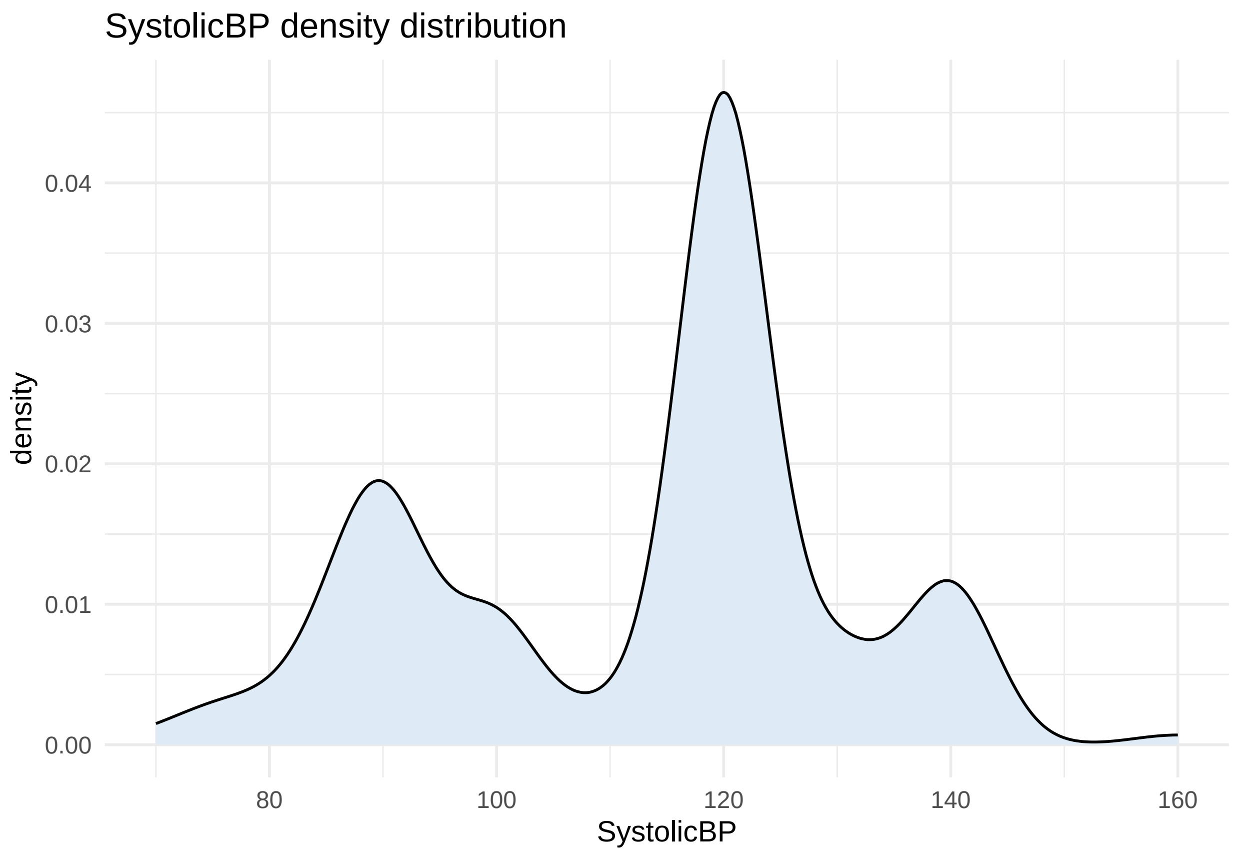
Lets look into the distribution of each feature to understand better the population within the dataset. I am going to plot the general population for each feature and then I will slice it into the three risk levels. In the order they appeared in the summary table we have the following:



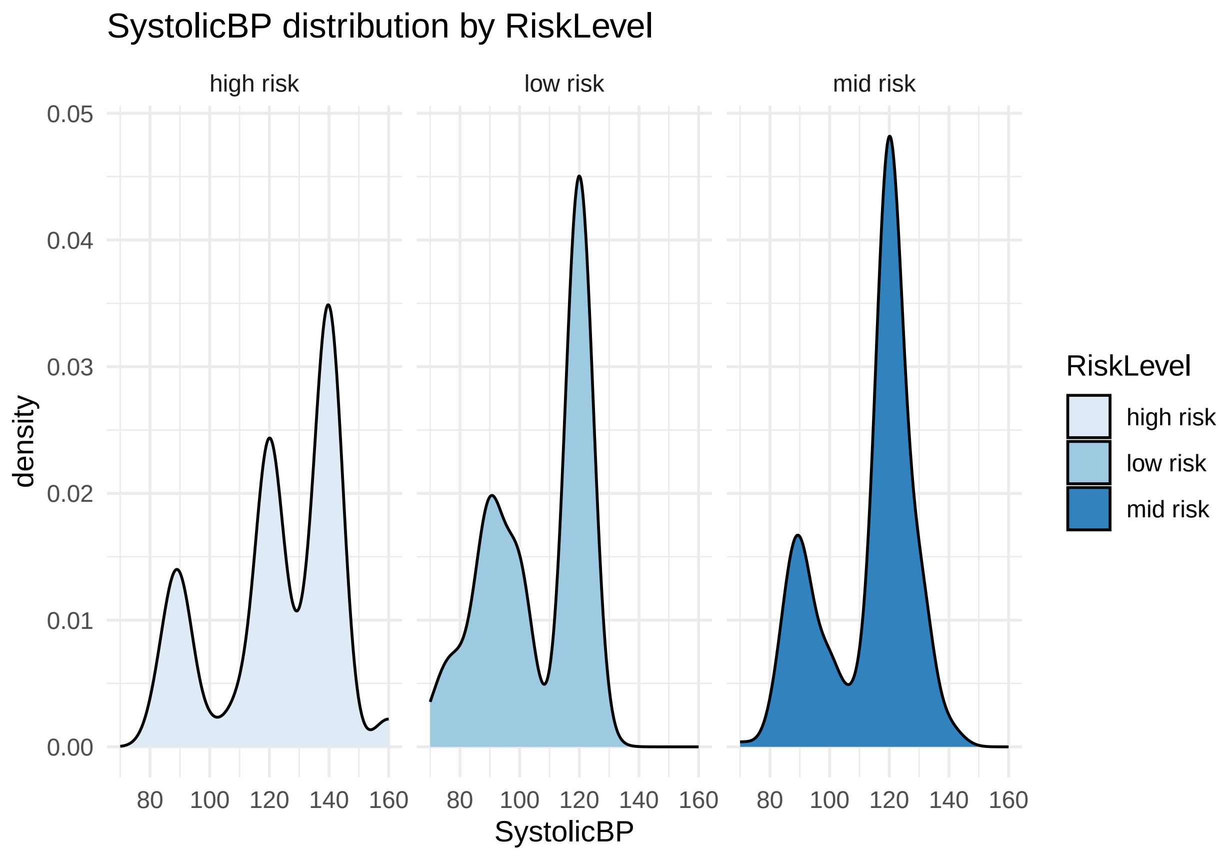
Age



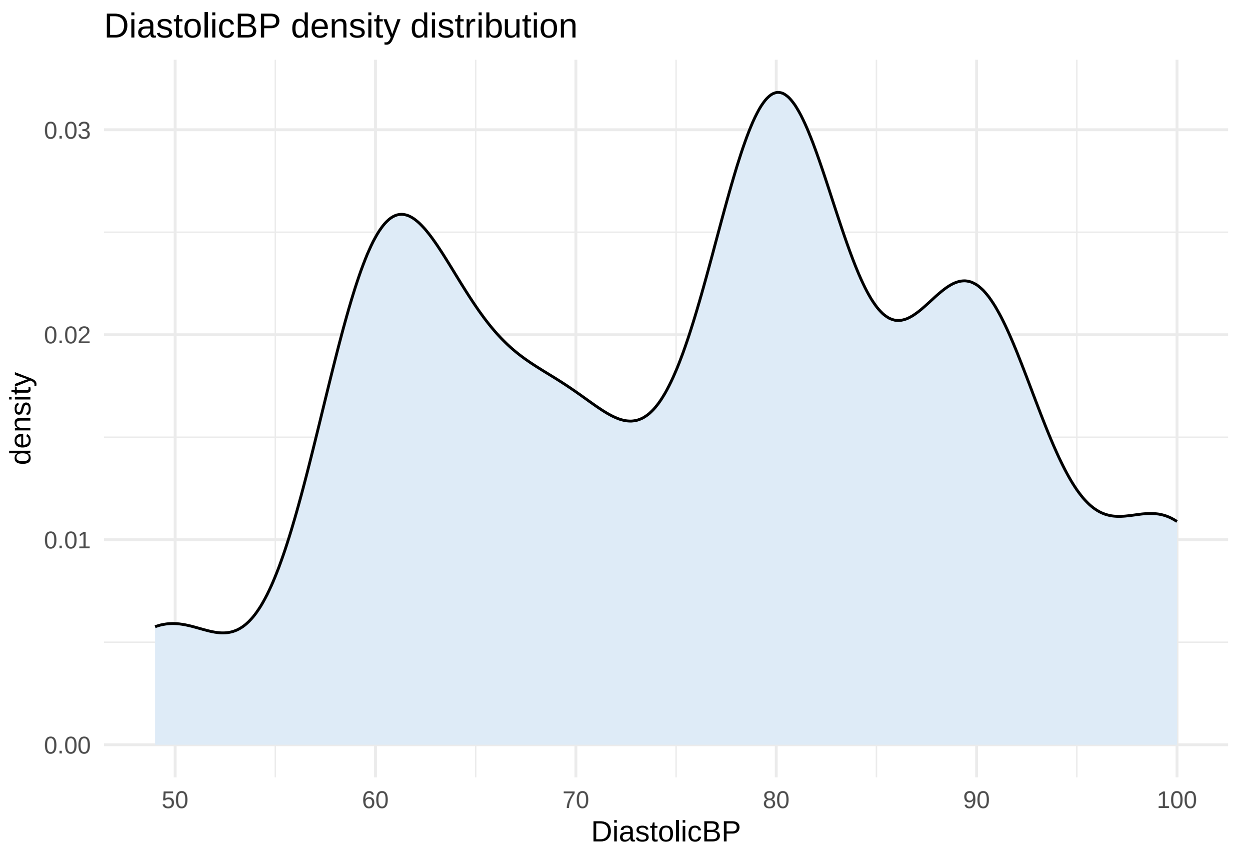
We can see some differences but, the most obvious is *high risk*. Its density tend to be higher in the older population while *low risk* is concentrated more around the 20's and *mid risk* starts on the 20's and almost disappears close to 35.


SystolicBP density distribution
density
0.04
0.03
0.02
0.01
0.00
80
100
120
140
160
SystolicBP

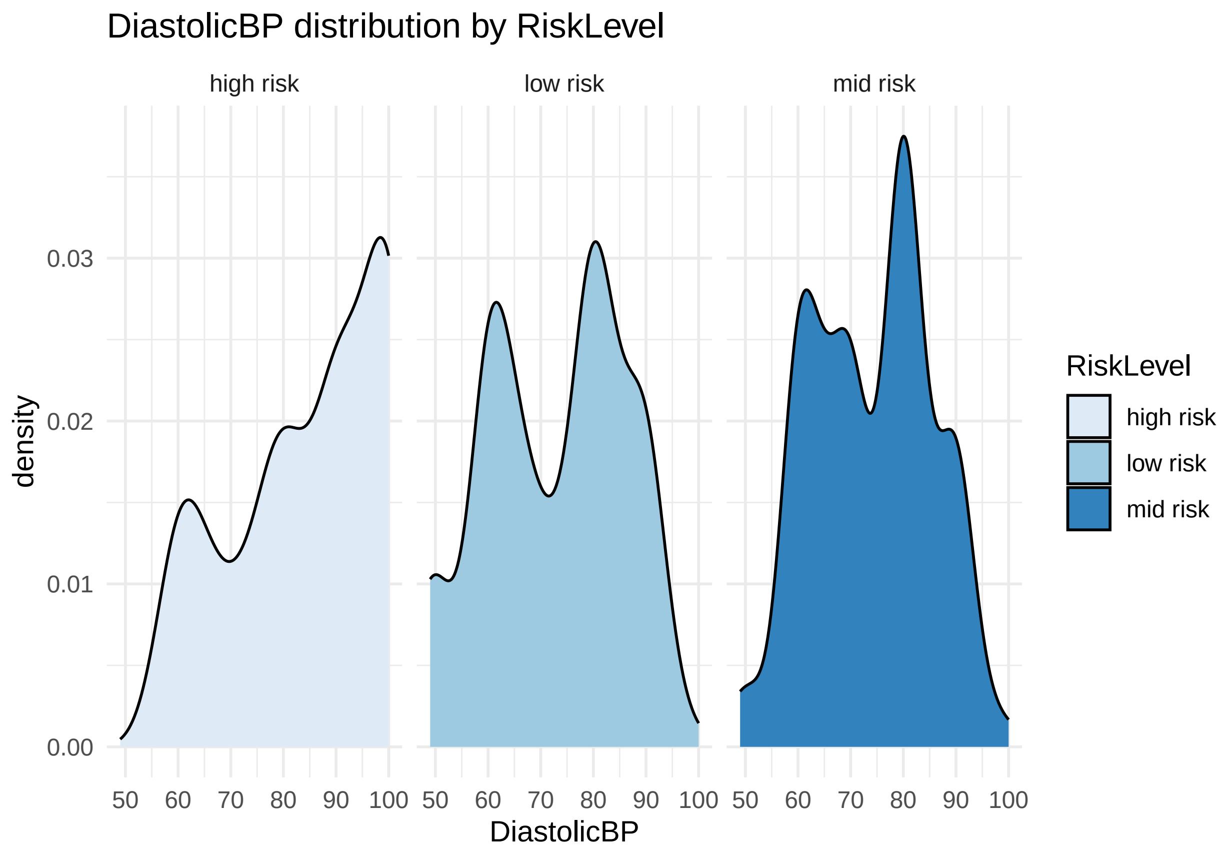
SystolicBP



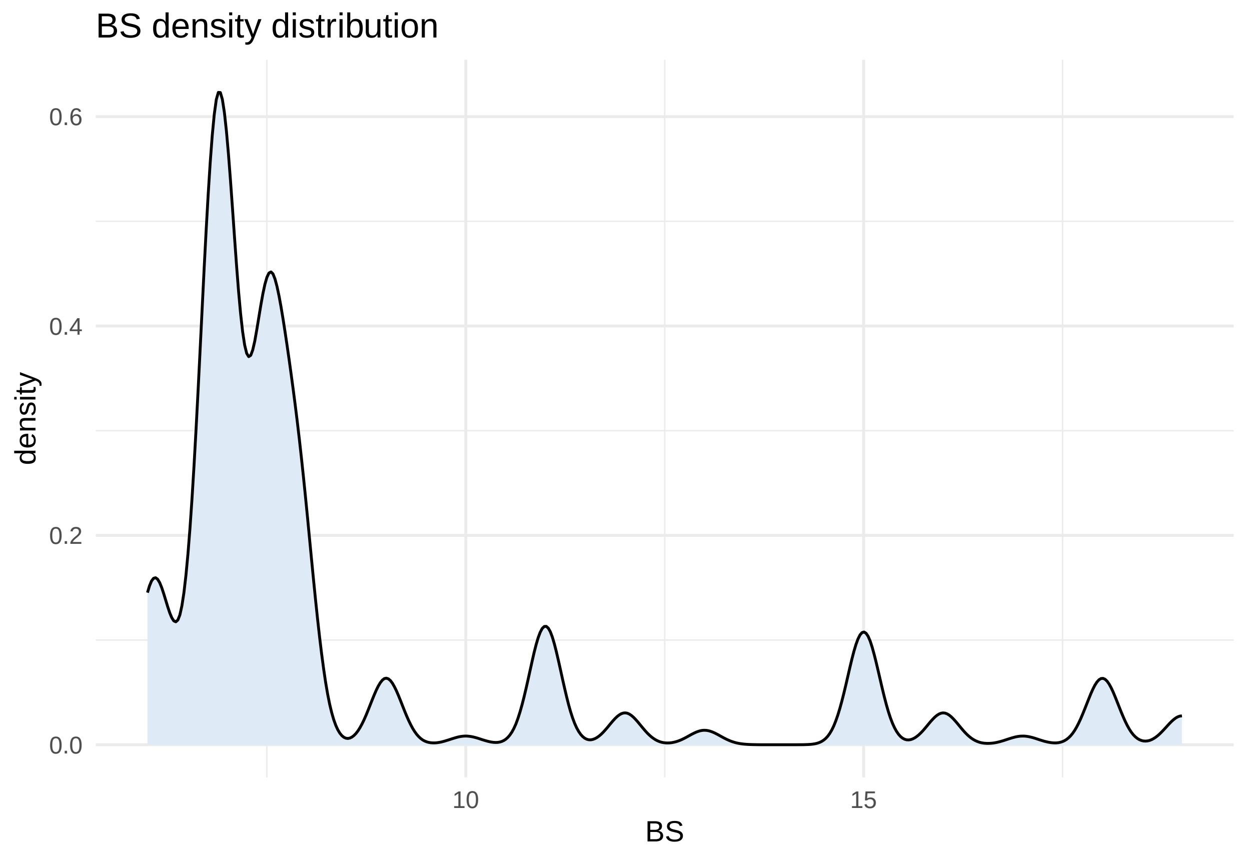
While *low risk* and *mid risk* seems very alike and the first don't reach the 140, the latest almost insignificant at 140 but *high risk* density is the highest at that same value.



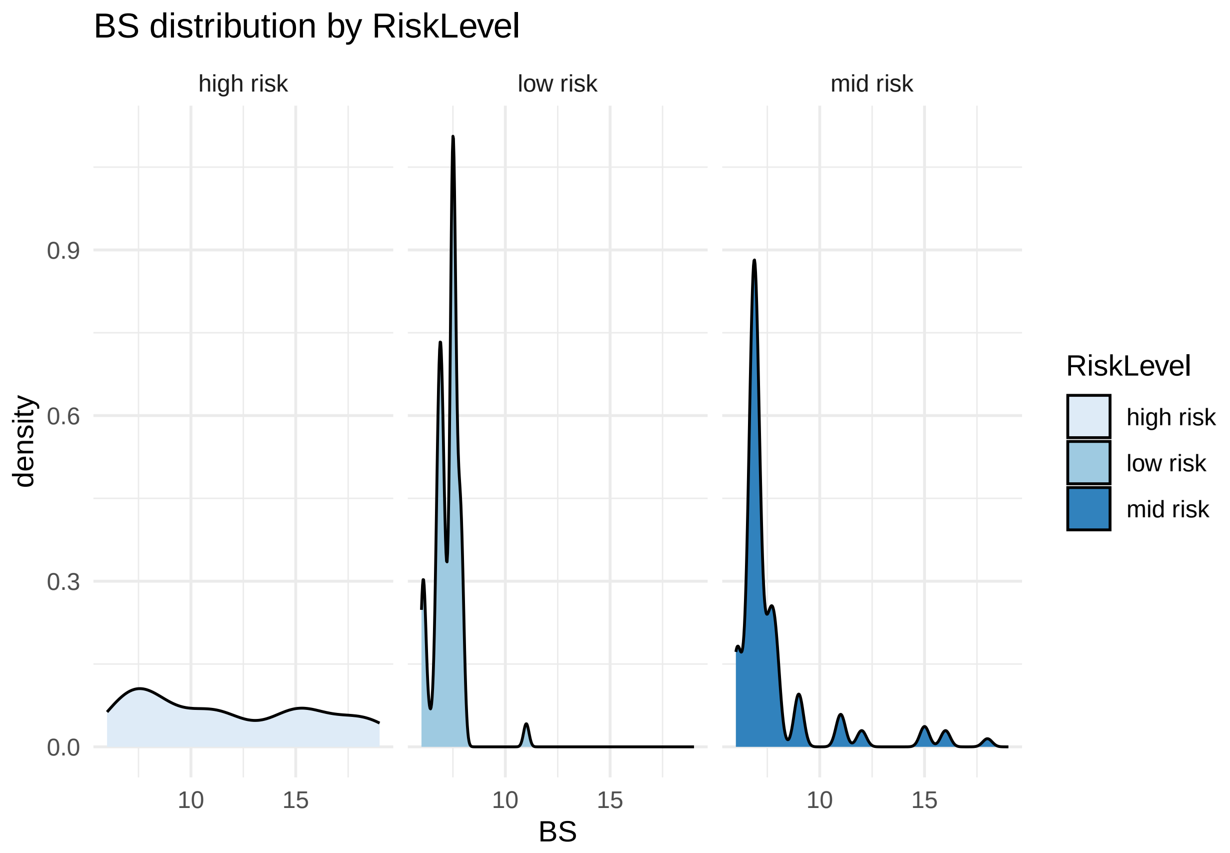
DiastolicBP



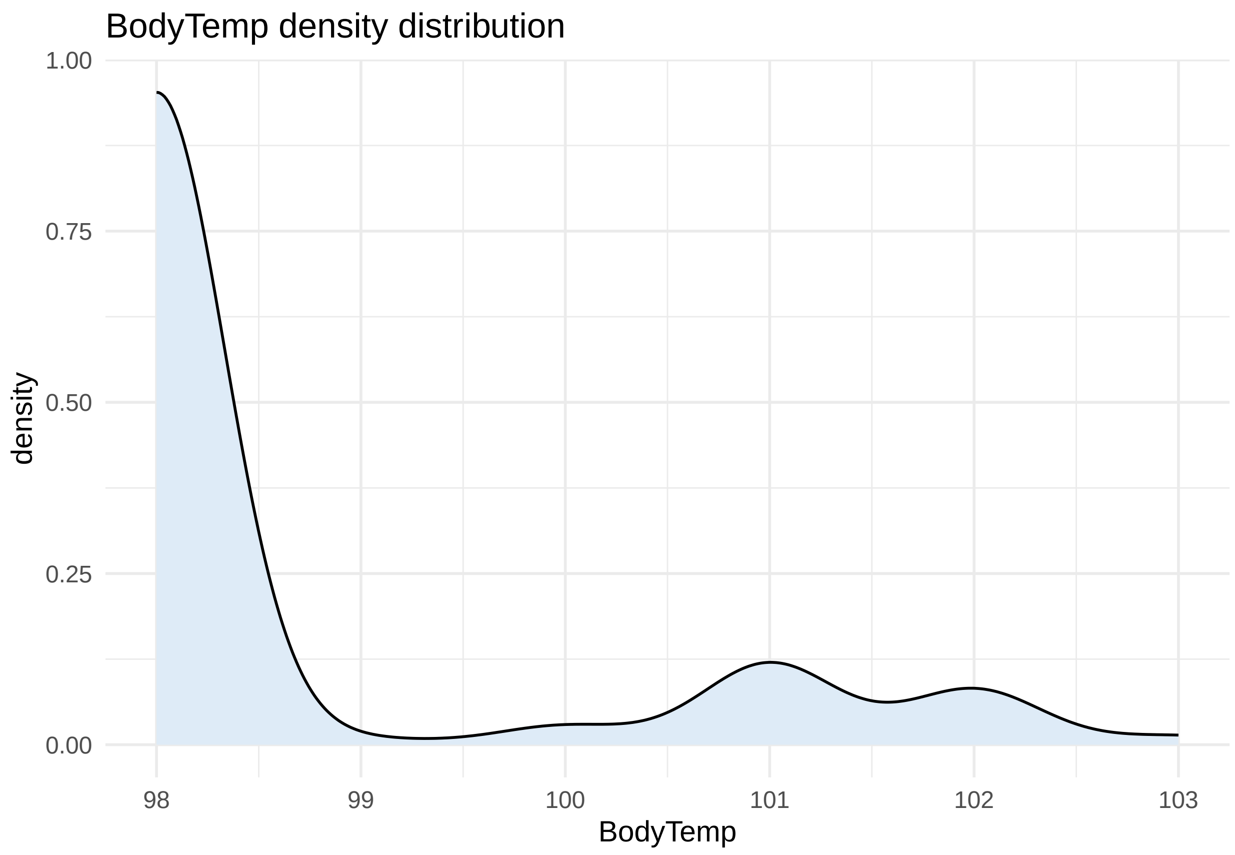
The main observation in the previous plots is that *high risk* density increases when *DiastolicBP's* value also increases. It's evident in the plots that the highest density for *high risk* is when it is closer to 100 while in the other two risk levels decreases when approaching to the same value.



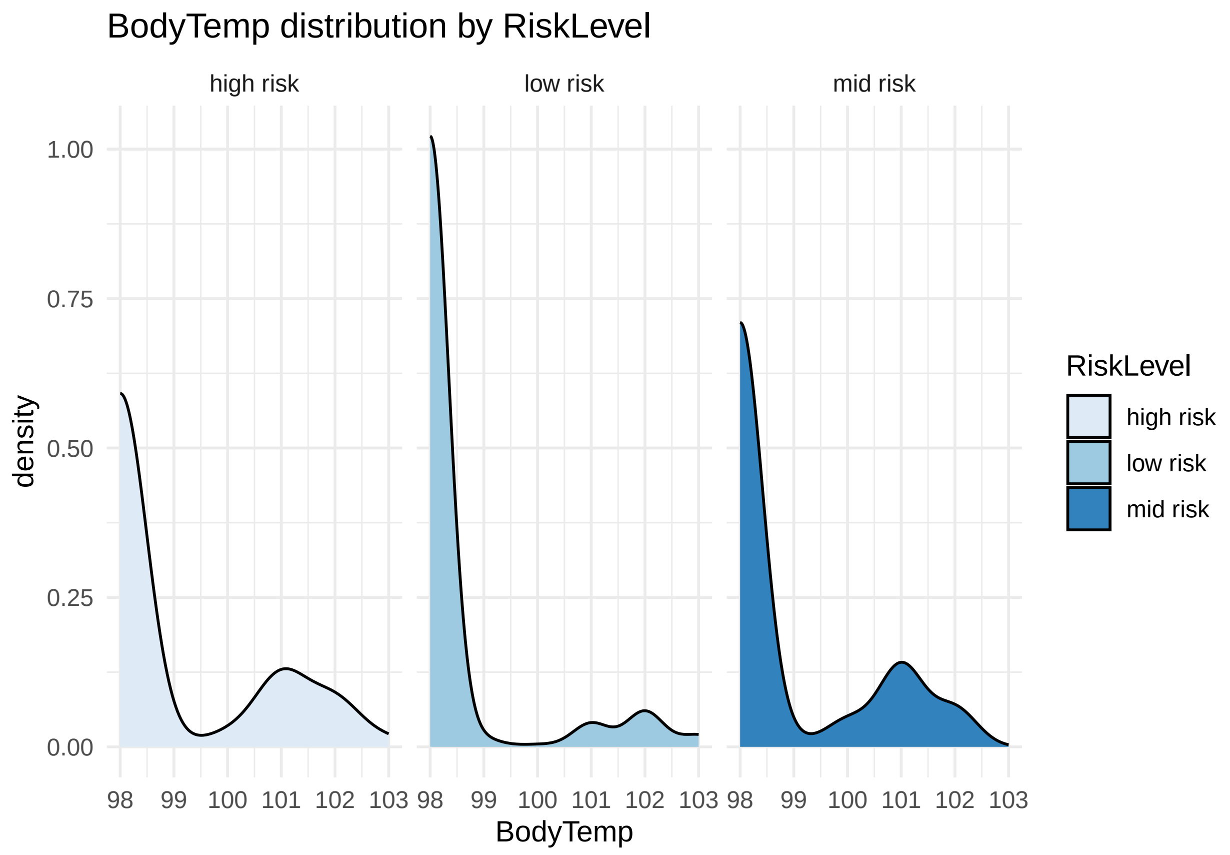
BS



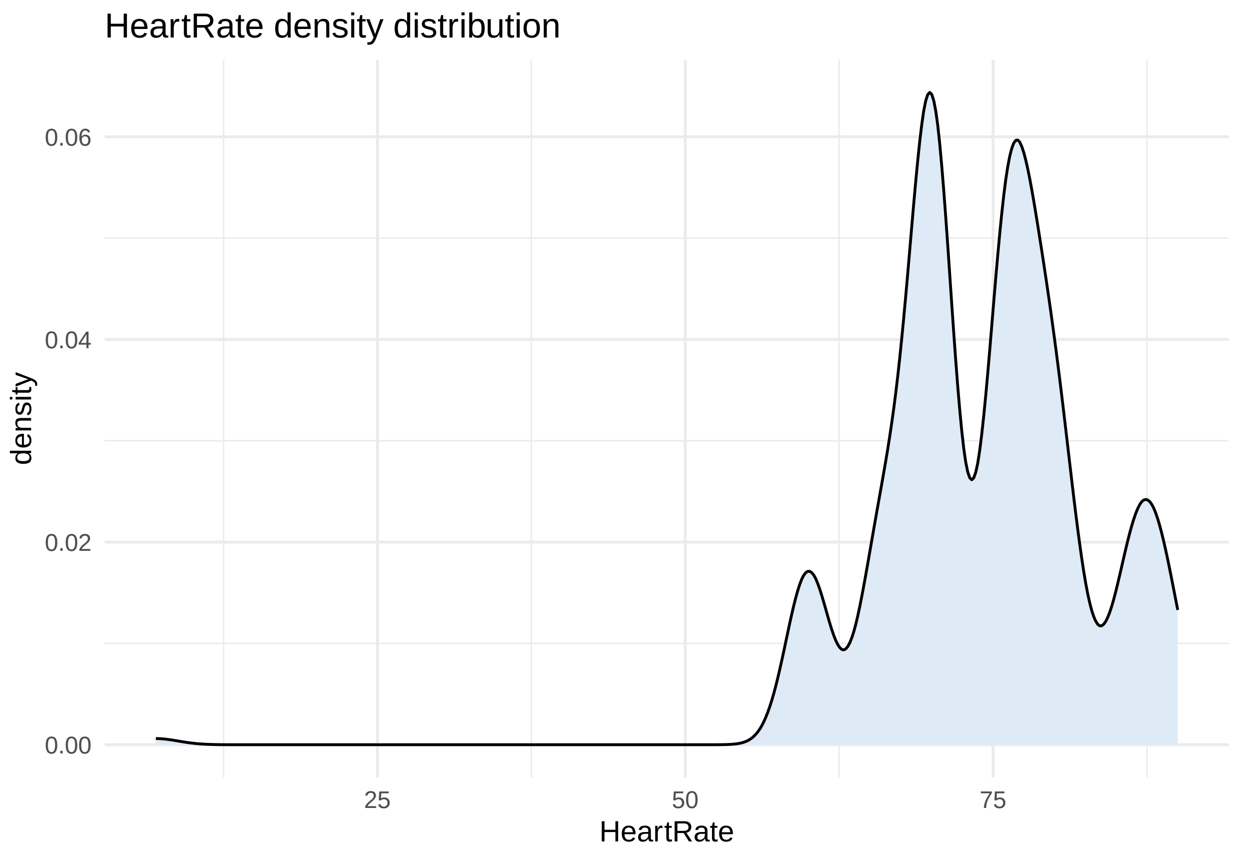
Most of the population has a BS of less than 7 of close to it except *high risk* population that presents a very steady densities values almost regardless the BS.


BodyTemp density distribution
density
1.00
0.75
0.50
0.25
0.00
98
99
100
101
102
103
BodyTemp

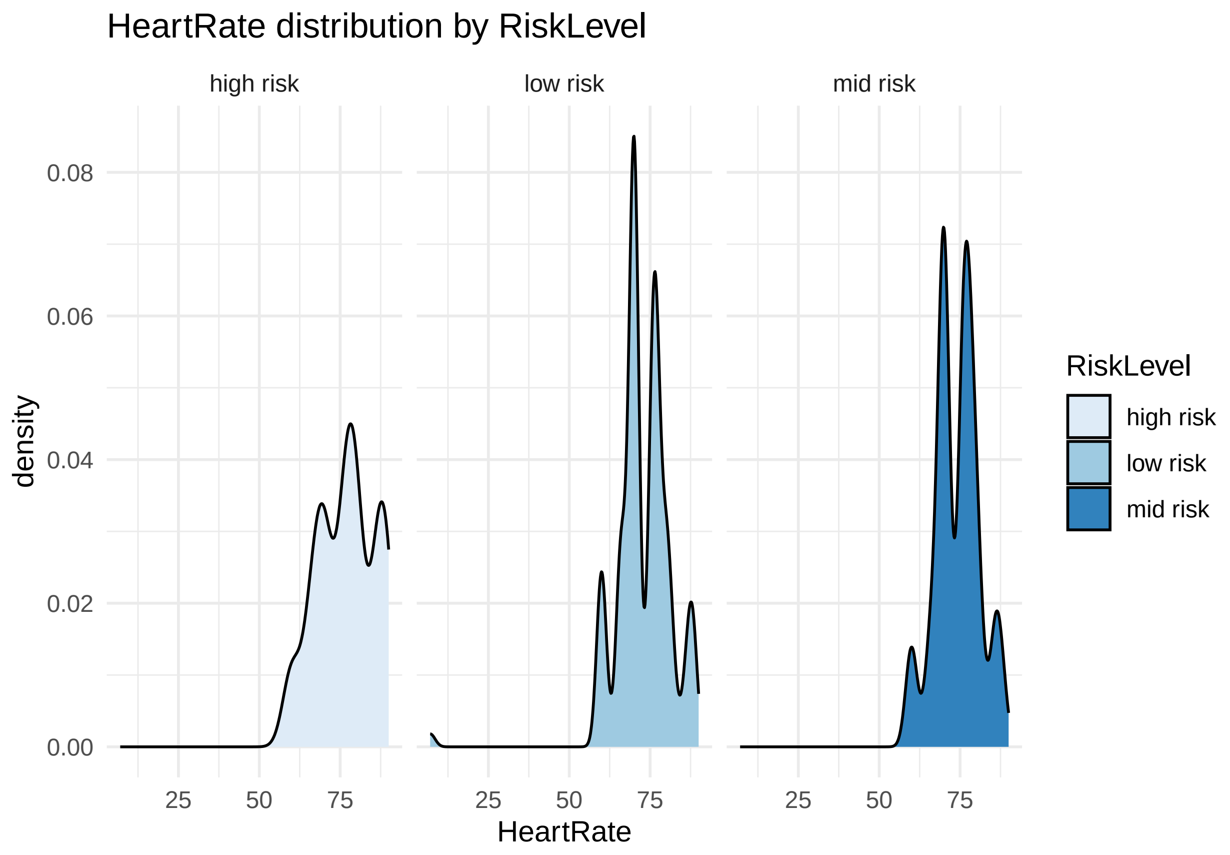
BodyTemp



Around a *BodyTemp* of 98 *low risk* population has the highest density value, close to 1, and some very small differences around a temperature of 101.

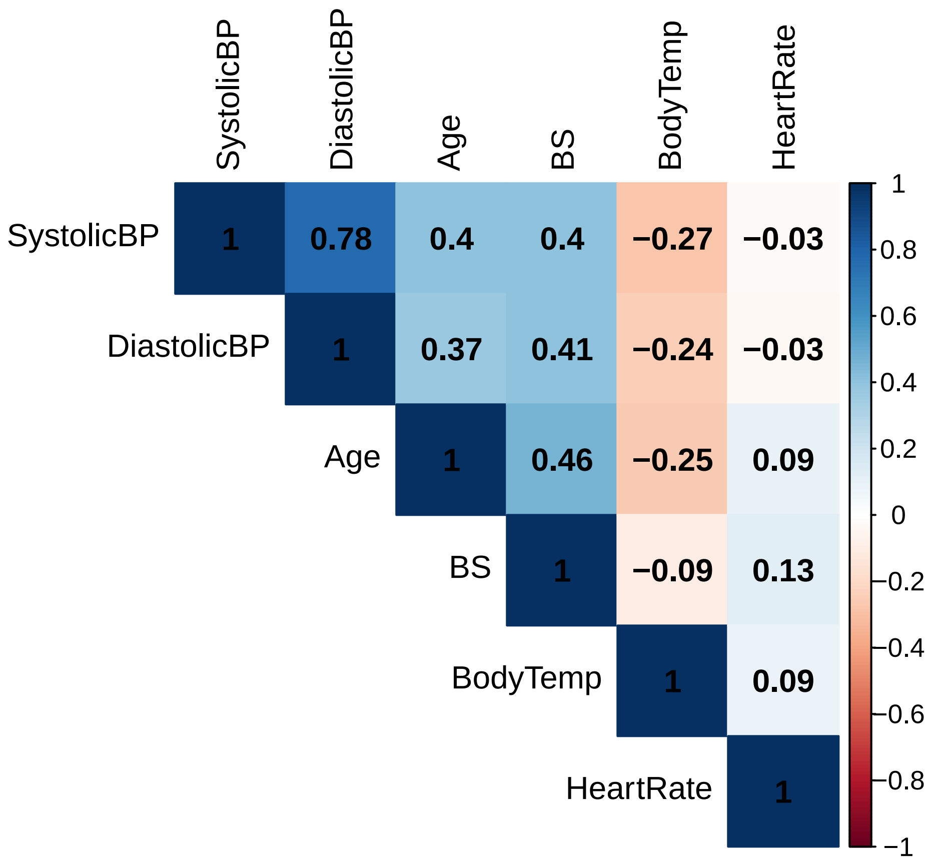

HeartRate density distribution
density
0.06
0.04
0.02
0.00
25
50
75
HeartRate

HeartRate



While HeartRate density distributions for *low risk* and *mid risk* are very alike, and generally speaking the value doesn't go down 50, the distribution for *high risk* stays bellow a density value of less than 0.05 while the others surpasses 0.07.

Correlation matrix heat-map It is time now to analyze any possible correlation between the different factors. With the following heat-map we can see that there are some interesting values between some of them.



There is no surprise about the correlation between both blood pressures but, we car see that some other values (in blue) show that a weak positive correlation exist between factors like Age and BS, and both blood pressures and Age and BS.

Methods

I used three methods for this project and those are Naive Bayes, Random Forest and K-Nearest Neighbors (KNN). To measure the accuracy of each method the dataset was divided in three: *training*, *validation*, and *final_test*. The dataset final_test has 10% of the observation in the original dataset while training has 80% of the remaining and validation the other 20%. Once I have finished training and evaluating each method with training and validation datasets I evaluated the one with the best result using the final_test.

This is the code used to split the original dataset:

```r
#Training set and testing set----

set.seed(1, sample.kind = "Rounding") #if using R 3.6 or later

#set.seed(1) #if using R3.5 or earlier

final_test_index <- createDataPartition(y = ds$RiskLevel, times = 1, p = .1, list = FALSE)

temp_ds <- ds[-final_test_index,]

validation_index <- createDataPartition(y= temp_ds$RiskLevel, times = 1, p = .2, list = FALSE)
```

```r
training <- temp_ds[-validation_index,]

validation <- temp_ds[validation_index,]

final_test <- ds[final_test_index,]
```

Method I - Naive Bayes

Naive Bayes constitutes a group of straightforward probabilistic classifiers that leverage Bayes' theorem while making robust independence assumptions among features. These classifiers find application in tasks like text classification, aiming to capture the input distribution within a specific class or category. Notwithstanding their simple design and oversimplified assumptions, Naive Bayes classifiers demonstrate high scalability and can attain notable accuracy levels, especially when combined with kernel density estimation. Despite their naive characteristics, they have proven effective in addressing various complex real-world situations.

To implement Naive Bayes in this project I used the package e1071 by David Meyer. The code will look like this:

```r
# Create a Naive Bayes model
nb_model <- naiveBayes(RiskLevel ~ BodyTemp + HeartRate + DiastolicBP + SystolicBP + Age + BS, data = t

# Make predictions on the training data
predictions <- predict(nb_model, validation)

# Display the confusion matrix
conf_matrix <- table(predictions, validation$RiskLevel)

# Calculate accuracy
accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
```

Method II - Random Forest

The Random Forest algorithm is employed in both classification and regression tasks within machine learning. As an ensemble learning method, it builds multiple decision trees during training and determines the class chosen by the majority of trees for classification or the mean/average prediction for regression. Random Forests address the tendency of decision trees to overfit their training sets, offering a correction mechanism. Notably scalable, they can effectively manage large datasets characterized by high dimensionality. Random Forests have found application in diverse fields, including remote sensing, bioinformatics, and computer vision.

The package used for Random Forest was randomForest by Andy Liaw and matthew Wiener based on original Fortran code by Leo Breimen and Adele Cutler. Its used will look like this:

```r
# Copy datasets and convert "RiskLevel" to a factor
rf_training <- training
rf_training$RiskLevel <- as.factor(rf_training$RiskLevel)
rf_validation <- validation
rf_validation$RiskLevel <- as.factor(rf_validation$RiskLevel)

# Create a Random Forest model
rf_model <- randomForest(RiskLevel ~ BodyTemp + HeartRate + DiastolicBP + SystolicBP + Age + BS,
                         data = rf_training,
                             ntree = 500,  # Number of trees in the forest
                             importance = TRUE)

# Make predictions on the training data
predictions <- predict(rf_model, newdata = rf_validation)

# confusion matrix
conf_matrix <- table(predictions, rf_validation$RiskLevel)

# Calculate accuracy
accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
```

Method III - K-Nearest Neighbors

The K-Nearest Neighbors (KNN) algorithm, a non-parametric supervised learning approach for classification and regression tasks, relies on the concept that similar data points are proximate in the feature space. This method identifies the k nearest data points in the training set to a given test point and classifies the test point based on the majority class among its k nearest neighbors.

KNN stands out for its simplicity, making it easily understandable and implementable. Its interpretability is high, allowing for straightforward explanations to non-technical stakeholders. However, its computational demands can be significant, especially with large datasets and high-dimensional feature spaces.

The package used for KNN was class. This package has the function knn() used for classification that calculate the Euclidean distance and its code looks like this:

```
# Create a k-NN model
knn_model <- knn(train = rf_training[, c("BodyTemp", "HeartRate", "DiastolicBP", "SystolicBP", "Age", "
                 test = rf_validation[, c("BodyTemp", "HeartRate", "DiastolicBP", "SystolicBP", "Age",
                 cl = rf_training$RiskLevel,
                 k = 3)  # Specify the number of neighbors (k)

# Display the confusion matrix
conf_matrix <- table(knn_model, rf_validation$RiskLevel)

# Calculate accuracy
accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
```

Results

After testing each method, the best result was produced by Random Forest. The following code test the Random Forest method with the Final_test dataset and displays the accuracy of each method along the final result:

```
# Let's prepare the final_test data set by conferting RiskLevel to a factor
final_test$RiskLevel <- as.factor(final_test$RiskLevel)
# Make predictions on the final_test data

predictions <- predict(rf_model, newdata = final_test)
conf_matrix <- table(predictions, final_test$RiskLevel)
results <- bind_rows(results, tibble(method = "Final", Accuracy = sum(diag(conf_matrix)) / sum(conf_mat
kable(results)
```

method	Accuracy
Naive Bayes	0.6393443
Random Forest	0.8142077
KNN	0.7213115
Final	0.8349515

Conclusion

Each method used during the development of the project is commonly used on classification systems. Even when all of them are created for this, in this scenario Random Forest produced the best result. It is also incredible how with so little data this algorithms have the ability to predict with high accuracy. During the development I didn't took into consideration some characteristics specific to healthcare. One of the improvements that could be implemented is increasing sensitivity even when that could mean that patients that aren't high risk are classified as one. When working with live threatening data it is better to be safe.