CYO PROJECT

MATERNAL HEALTH RISK

HarvardX PH125.9x

Data Science: Capstone

SYEDA AQEEL

Contents

1.Introduction

Maternal health risk refers to risks which women are exposed to during pregnancy, childbirth and postnatal period. These risks can involve blood loss, anaemia, obstructed labour or even death in severe cases. In 2020, maternal mortality accounted for 152 deaths per 100,000 live births globally.

Most of these risks associated with maternal health can be prevented with timely management of resources and skills. Therefore, it is critical to expand efforts to study relationship between predictors of risks and maternal injury, disability and mortality so that women and their children can be ensured to reach their full potential for health and well-being.

In this report we document the case of Maternal health risk in context of rural Bangladesh. We'll use Maternal health risk data set, contributed by Daffodil International University in Dhaka, Bangladesh, from *UCI Machine Learning Repository*.

The *data* is collected through Internet of Things (IoT) based risk monitoring system from various hospitals, community clinics and maternal health care facilities from rural regions of Bangladesh. Pregnant women were effectively monitored for risk indicators with Wearable sensing technology. These indicators were then evaluated to identify risk intensity level during pregnancy.

The data set stores values for following predictors of risk:

- **Age**: Age in years when a woman is pregnant
- **SystolicBP**: Upper value of Blood Pressure in mmHg
- **DiastolicBP**: Lower value of Blood Pressure in mmHg
- **BS**: Blood glucose levels in terms of a molar concentration in mmol/L
- **BodyTemp**: body temperature in degrees Fahrenheit (°F).
- **HeartRate**: A normal resting heart rate in beats per minute

And outcome of these predictors as

- **Risk Level**: Predicted Risk Intensity Level during pregnancy considering the previous attribute

1.1. Project Workflow

This project will follow following structure:

i. Import Data set

We'll import Maternal Health Risk data set from University of California Irvine Machine Learning Repository in to RStudio interface.

ii. Clean Data set

Then we'll look in to structure of data set and make sure there are no missing values for any of predictors and outcome. If none are found, we'll divide Maternal Health Risk data set in to two sets: *dat* and *validation set.*

dat set will contain 90% of original data while Validation set will have 10% entries of original set. The latter will only be used to test final algorithm.

Since dat set has to be partitioned further to train and test accuracy of at least five machine learning algorithms with tuning parameters, we chose to have 90:10 data partition rule. We need sufficiently enough data points to train our algorithms and tune their parameters.

iii. Explore and visually analyse Data set

Bar graphs, density plots, correlation plots and other visual and statistical representations of data will aid us in understanding features of data set.

iv. Model Data set

We'll model data set based on insights gained in data exploration. To model, we'll generate two more data sets from *dat* set: Train set and Test set. 90% of *dat* entries will be in Train set while Test set will hold only 10% of *dat* entries. We'll predict Risk Level with Test set and evaluate accuracy of the models.

v. Analyse Results

Results obtained through all the models that will be applied to train data set will be analyzed to find maximum Accuracy. The method of modelling with highest accuracy will then be applied to final set of data (*dat set*) to train and validate.

vi. Report results.

Final results will be reported to form conclusion.

2. Methods & Analysis

2.1. Data Cleaning

We'll begin with installing and loading all required libraries.

```
# Install all required libraries

if(!require(readr)) install.packages("readr")
if(!require(tidyverse)) install.packages("tidyverse")
if(!require(caret)) install.packages("caret")
if(!require(dplyr)) install.packages("dplyr")
if(!require(matrixStats)) install.packages("matrixStats")
if(!require(xgboost)) install.packages("xgboost")
if(!require(Ckmeans.1d.dp)) install.packages("Ckmeans.1d.dp")
if(!require(ggridges)) install.packages("ggridges")
if(!require(patchwork)) install.packages("patchwork")
if(!require(corrplot)) install.packages("corrplot")
if(!require(ggplot2)) install.packages("ggplot2")
if(!require(GGally)) install.packages("GGally")



# load all required libraries

library(readr)
library(tidyverse)
library(caret)
library(dplyr)
library(matrixStats)
library(xgboost)
library(Ckmeans.1d.dp)
library(ggridges)
library(patchwork)
library(corrplot)
library(ggplot2)
library(GGally)
```

Then we'll load Maternal Health Risk data set from UCI Machine Learning Repository.

```
# load Maternal Health Risk data set from UCI Machine Learning Repository

Data <- read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/00639/Maternal%20Health%20Risk%20Data%20Set.csv")
```

```
## Rows: 1014 Columns: 7
## -- Column specification --------------------------------------------------
------
## Delimiter: ","
## chr (1): RiskLevel
## dbl (6): Age, SystolicBP, DiastolicBP, BS, BodyTemp, HeartRate
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this m
essage.
```

The Data set has 1014 rows and 7 columns.

```
# Number of rows and columns

dim(Data)

## [1] 1014    7
```

Let's look in to first few entries of Data set.

```
# view first six entries of Data set
head(Data)

## # A tibble: 6 x 7
##     Age SystolicBP DiastolicBP    BS BodyTemp HeartRate RiskLevel
##   <dbl>      <dbl>       <dbl> <dbl>    <dbl>     <dbl> <chr>
## 1    25        130          80 15          98        86 high risk
## 2    35        140          90 13          98        70 high risk
## 3    29         90          70  8         100        80 high risk
## 4    30        140          85  7          98        70 high risk
## 5    35        120          60  6.1        98        76 low risk
## 6    23        140          80  7.01       98        70 high risk
```

Now we'll look for missing values in Data set.

```
# check missing values in Data set
sapply(Data, function(x) sum(is.na(x)))

##         Age  SystolicBP DiastolicBP          BS    BodyTemp   HeartRate
##           0           0           0           0           0           0
##   RiskLevel
##           0
```

There are no missing values.

We'll split *Data set* in to two sets: dat and validation set. *dat set* will contain 90% of original data while Validation set will have 10% entries of original set. The latter will only be used to test final algorithm.

Since dat set has to be partitioned further to train and test accuracy of at least five machine learning algorithms with tuning parameters, we chose to have 90:10 data partition rule. We need sufficiently enough data points to train our algorithms and tune their parameters.

```
# split the "Data set" in to train set(dat) and test set(Validation)

set.seed(1, sample.kind="Rounding")

## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' s
ampler
## used

Validation_index <- createDataPartition(y = Data$RiskLevel, times = 1, p = 0.
1, list = FALSE)

dat <- Data[-Validation_index,]
dat_x <- dat[,-7]
dat_y <- dat$RiskLevel


Validation <- Data[Validation_index,]
val_x <- Validation[,-7]
val_y <- Validation$RiskLevel
```

We'll generate two more data sets from dat set: Train set and Test set. 90% of dat entries will be in Train set while Test set will hold only 10% of dat entries. We'll predict Risk Level with Test set and evaluate accuracy of the models.

```
# Further split "dat set" in to train set and test set

test_index <- createDataPartition(y = dat_y, times = 1, p = 0.1, list = FALSE
)

train <- dat[-test_index,]
train_x <- train[,-7]
train_y <- train$RiskLevel


test <- dat[test_index,]
```

```
test_x <- test[,-7]
test_y <- test$RiskLevel
```

2.2. Data Exploration

Exploratory analysis will aid in understanding data set's construct. It will help summarize data set's main characteristics. And discover patterns and visualize these patterns.

We'll begin with observing data set's dimension.

```
# Number of rows and columns in dat

dim(dat)

## [1] 911   7
```

There are 911 rows and 7 columns.

All predictors in data set are numeric. These include Age, SystolicBP, DiastolicBP, BS, BodyTemp, and HeartRate. The outcome of these predictors, RiskLevel, is of character class.

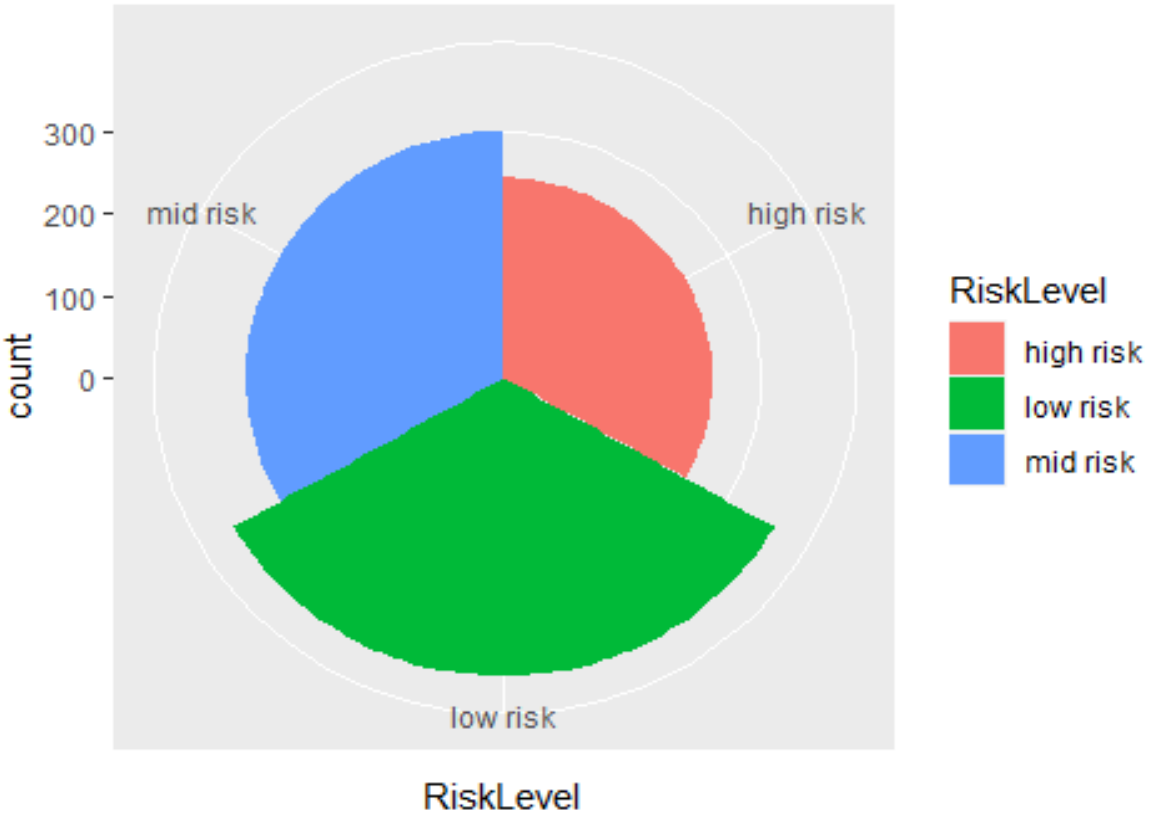
```
# Structure of dat
str(dat)

## tibble [911 x 7] (S3: tbl_df/tbl/data.frame)
##  $ Age        : num [1:911] 25 35 29 35 23 23 35 32 42 23 ...
##  $ SystolicBP : num [1:911] 130 140 90 120 140 130 85 120 130 90 ...
##  $ DiastolicBP: num [1:911] 80 90 70 60 80 70 60 90 80 60 ...
##  $ BS         : num [1:911] 15 13 8 6.1 7.01 7.01 11 6.9 18 7.01 ...
##  $ BodyTemp   : num [1:911] 98 98 100 98 98 98 102 98 98 98 ...
##  $ HeartRate  : num [1:911] 86 70 80 76 70 78 86 70 70 76 ...
##  $ RiskLevel  : chr [1:911] "high risk" "high risk" "high risk" "low risk"
...
```

2.2.1. RiskLevel

We'll begin with looking into RiskLevel.

RiskLevel is classified in to three levels according to intensity of risk. The samples of maternal health are determined to be either high risk, low risk or mid risk.

```r
# type of RiskLevel

levels(factor(dat$RiskLevel))

## [1] "high risk" "low risk"  "mid risk"

# count of each RiskLevel

table(dat$RiskLevel)

##
## high risk  low risk  mid risk
##       244       365       302

# Visualize RisKLevel distribution

dat %>% ggplot(aes(RiskLevel,
                   fill = RiskLevel)) +  geom_bar(width = 1) + coord_polar(th
eta = "x")
```



Most samples are of low-risk patients.

2.2.2. Age

The minimum age of patients in data set is 10 years. This deviation is consistent with literature provided by UNICEF.

UNICEF documents that Bangladesh has the highest prevalence of child marriage in South Asia. It is home to 42 million child brides. Of these, 21 million are married before age 15.

The maximum age in maternal health sample is 70 years. Although only one such case is reported in data set. Medical findings suggest that situations like these are very less likely to occur but are not impossible with the help of modern fertility treatment.

Moreover, high fertility rates in Bangladesh are supported with its low contraceptive prevalence rate. According to annual World Bank report, Bangladesh's contraceptive prevalence for 2020 was just 62.5%.

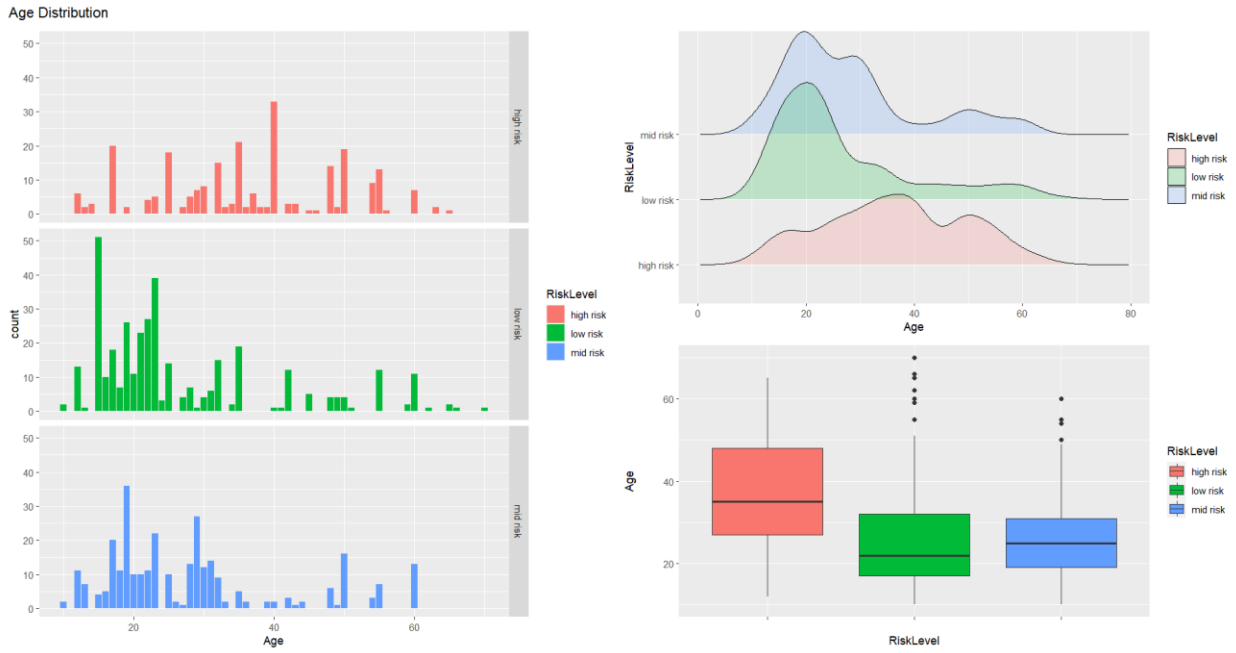
```
# summary statistics of Age Distribution

summary(dat$Age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.00   19.00   25.00   29.78   39.00   70.00
```

Here we represented Age distribution with bar graph, density plots and boxplot for each risk level.

```
# plot Bar Graph

A1 <- dat %>%  ggplot( aes(Age,  fill = RiskLevel)) +
  geom_bar() + facet_grid(RiskLevel~ . )


# plot Density Plot

A2 <- dat %>%  ggplot( aes(Age,  RiskLevel, fill = RiskLevel)) +
  geom_density_ridges(alpha = 0.2)

# plot Boxplot
A3 <- dat %>%  ggplot( aes(RiskLevel, Age,  fill = RiskLevel)) +
  geom_boxplot() +
  theme(axis.text.x = element_blank())


# display these plots together

( A1 | (A2 / A3) ) + plot_annotation(title = "Age Distribution")
```

```
## Picking joint bandwidth of 3.18
```



For low risk and mid risk samples, we observe that distributions are asymmetrical and right-skewed.

The density plot for high-risk samples tend to follow a bell-shaped curve.

The median age for high-risk cases is approximately 35 years. And for mid risk samples it is 25 years.

2.2.3 SystolicBP

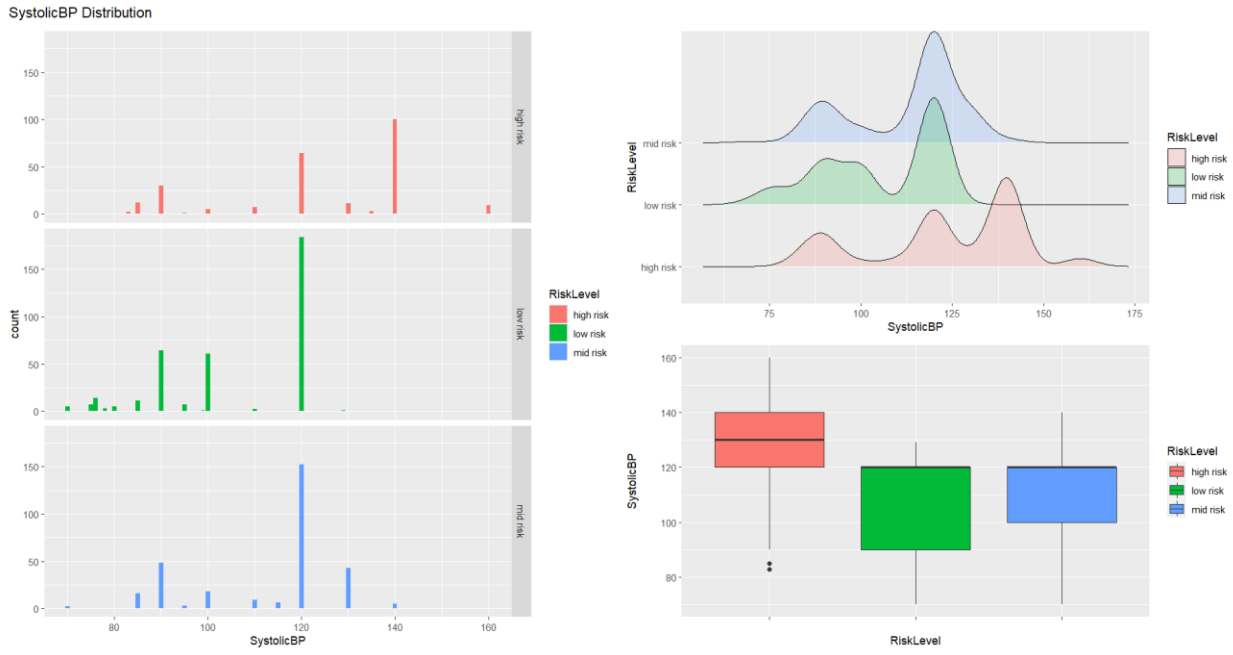
SystolicBP, upper value of Blood Pressure, has a median of 120 mmHg.

It has a trimodal distribution across high-risk samples and bimodal distribution across mid risk samples.

The maximum value recorded for SystolicBP was 160 mmHg which was in the case of high risk samples.

```
# summary statistics of SystolicBP distribution

summary(dat$SystolicBP)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    70.0    95.0   120.0   112.9   120.0   160.0
```

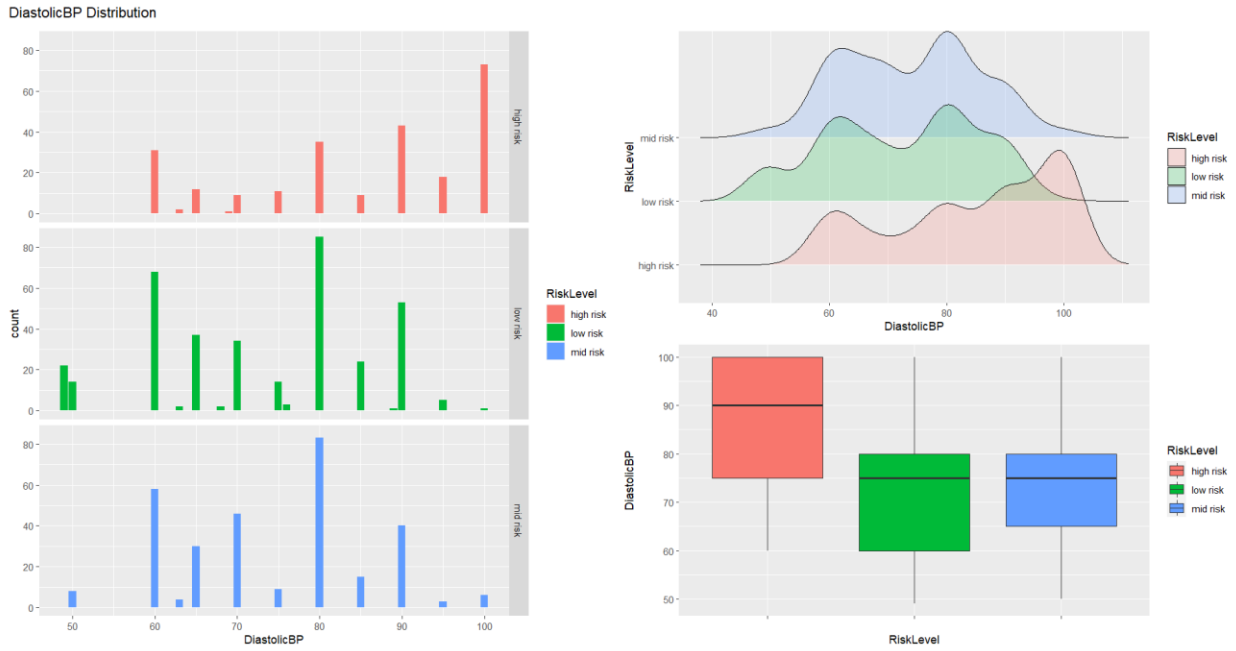
```r
# plot Bar Graph

S1 <- dat %>%  ggplot( aes(SystolicBP,  fill = RiskLevel)) +
  geom_bar() + facet_grid(RiskLevel~ . )


# plot Density Plot

S2 <- dat %>%  ggplot( aes(SystolicBP, RiskLevel, fill = RiskLevel)) +
  geom_density_ridges(alpha = 0.2)

# plot  Boxplot

S3 <- dat %>%  ggplot( aes(RiskLevel, SystolicBP,  fill = RiskLevel)) +
  geom_boxplot() +
  theme(axis.text.x = element_blank())


# display these plots together

(S1 | (S2 / S3)) + plot_annotation(title = "SystolicBP Distribution")

## Picking joint bandwidth of 4.39
```



2.2.4 DiastolicBP

The lower value of Blood Pressure is referred to as DiastolicBP. An average of 76.37 mmHg is recorded in data set. High risk samples tend to score more on this index. DiastolicBP distribution for all risk levels is not symmetrical.

```
# summary statistics of DiastolicBP Distribution

summary(dat$DiastolicBP)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   49.00   65.00   80.00   76.37   90.00  100.00

# plot Bar Graph

D1 <- dat %>%  ggplot( aes(DiastolicBP,  fill = RiskLevel)) +
  geom_bar() + facet_grid(RiskLevel~ . )


# plot Density plot


D2 <- dat %>%  ggplot( aes(DiastolicBP, RiskLevel, fill = RiskLevel)) +
  geom_density_ridges(alpha = 0.2)


# plot Boxplot

D3 <- dat %>%  ggplot( aes(RiskLevel, DiastolicBP,  fill = RiskLevel)) +
  geom_boxplot() +
  theme(axis.text.x = element_blank())


# display these plots together

(D1 | (D2 / D3)) + plot_annotation(title = "DiastolicBP Distribution")

## Picking joint bandwidth of 3.68
```

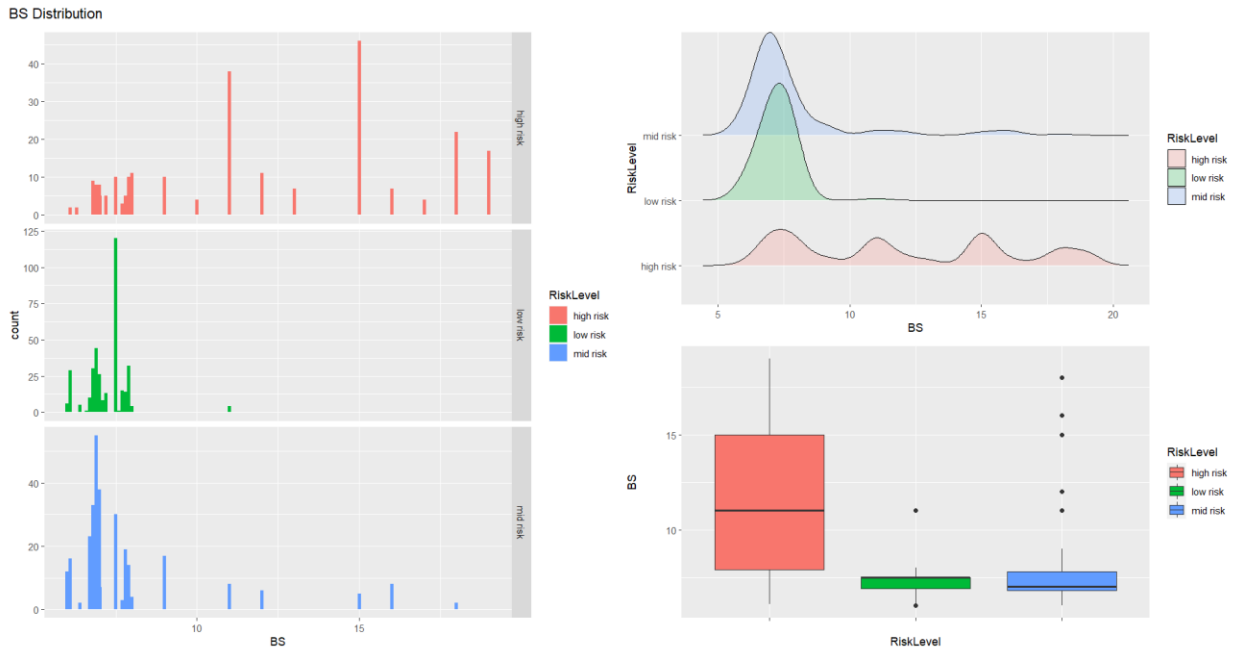


2.2.5. BS

Blood glucose levels in terms of a molar concentration is termed as BS in data set. For low risk and mid risk samples we observe its distribution is right-skewed and asymmetrical. High risk samples compared to other risk levels score more on this index. A median of 7.5 mmol/L is reported for data set.

```
# summary of BS

summary(dat$BS)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   6.000   6.900   7.500   8.693   8.000  19.000

# plot Bar Graph

B1 <- dat %>%  ggplot( aes(BS,  fill = RiskLevel)) +
  geom_bar(width = 0.1) +
  facet_grid(RiskLevel~ ., scales = "free")


# plot Density Plot

B2 <- dat %>%  ggplot( aes(BS, RiskLevel,  fill = RiskLevel)) +
  geom_density_ridges(alpha = 0.2)
```

```
# plot Boxplot

B3 <- dat %>%  ggplot( aes(RiskLevel, BS,  fill = RiskLevel)) +
  geom_boxplot() +
  theme(axis.text.x = element_blank())


# display these plots together

( B1 | (B2 / B3)) + plot_annotation(title = "BS Distribution")
```



2.2.6. BodyTemp

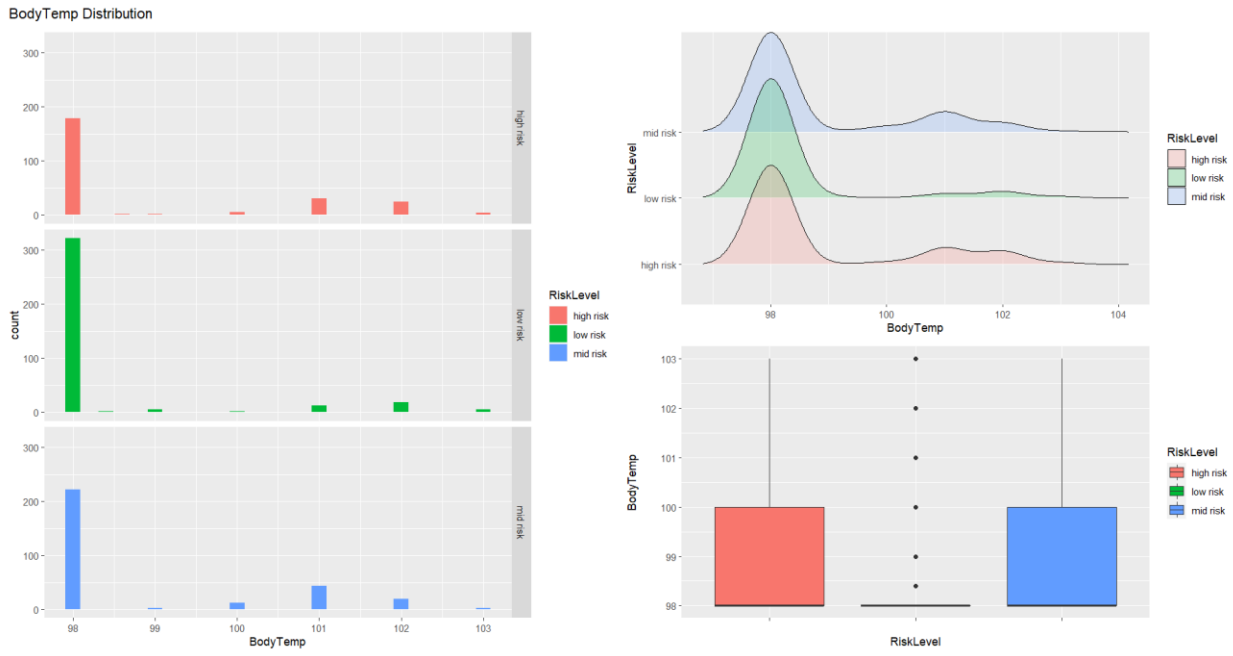
The body temperature of samples were recorded in Fahrenheit. It ranges from 98 F to 103 F with an average of 98.67.

We observe asymmetrical and right-skewed distribution for all risk levels.

```
# summary statistics of BodyTemp Distribution

summary(dat$BodyTemp)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   98.00   98.00   98.00   98.67   98.00  103.00
```

```r
# plot Bar Graph

T1 <- dat %>%  ggplot( aes(BodyTemp,  fill = RiskLevel)) +
  geom_bar() + facet_grid(RiskLevel~ . )


# plot Density Plot

T2 <-  dat %>%  ggplot( aes(BodyTemp, RiskLevel, fill = RiskLevel)) +
  geom_density_ridges(alpha = 0.2)


# plot Boxplot

T3 <- dat %>%  ggplot( aes(RiskLevel, BodyTemp,  fill = RiskLevel)) +
  geom_boxplot() +
  theme(axis.text.x = element_blank())


# display these plots together

(T1 | (T2 / T3)) + plot_annotation(title = "BodyTemp Distribution")
```

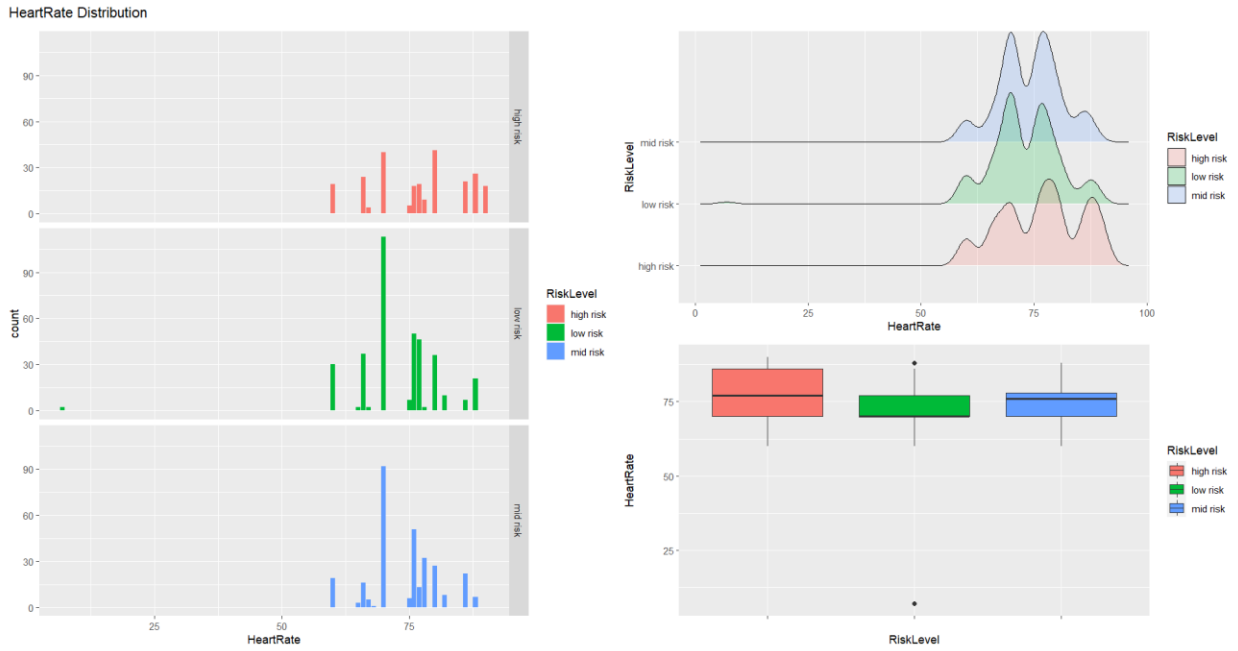
```
## Picking joint bandwidth of 0.391
```



2.2.7. HeartRate

Data set reports normal resting heart rate in beats per minute in HeartRate index. We observe asymmetrical distribution for the index across all risk levels. High risk samples report higher median HeartRate relative to other risk levels.

```
# summary statistics of HeartRate Distribution

summary(dat$HeartRate)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    7.00   70.00   76.00   74.29   80.00   90.00


# plot Bar Graph

H1 <- dat %>%  ggplot( aes(HeartRate,  fill = RiskLevel)) +
  geom_bar() + facet_grid(RiskLevel~ . )


# plot Density Plot

H2 <- dat %>%  ggplot( aes(HeartRate, RiskLevel,  fill = RiskLevel)) +
  geom_density_ridges(alpha = 0.2)



# plot Boxplot

H3 <- dat %>%  ggplot( aes(RiskLevel, HeartRate,  fill = RiskLevel)) +
  geom_boxplot() +
  theme(axis.text.x = element_blank())



# display these plots together


(H1 | (H2 / H3)) + plot_annotation(title = "HeartRate Distribution")

## Picking joint bandwidth of 1.94
```



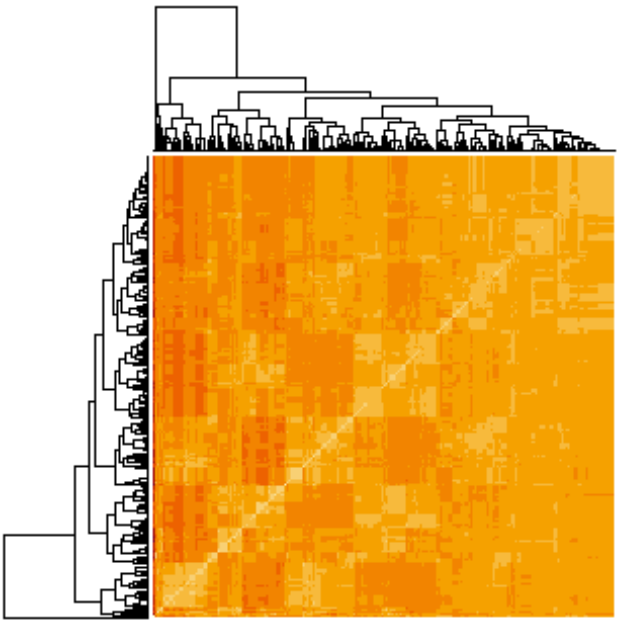
2.3 Data Visualization & Analysis

2.3.1. Heatmap

To discover patterns or clusters in our Maternal Health risk data set, we plot the Heatmap. We centered the observations and then scaled them. The distance of these scaled observations is visually represented in the image.

```
# convert into matrix

m <- as.matrix(dat_x)


# center entries

x_centered <- sweep(m, 2, colMeans(m))


# scale entries

x_scaled <- sweep(x_centered, 2, colSds(x_centered), FUN = "/")


# calculate distance
d <- dist(x_scaled)
```

```
# plot heatmap
heatmap(as.matrix(d), labRow = NA, labCol = NA)
```



2.3.2. Correlation Heatmap and Hierarchical Clustering

To understand how predictors relate to other predictors in data set, we plot Correlation Heatmap. The method of correlation is Spearman. That is because we observed outlier values for predictors.

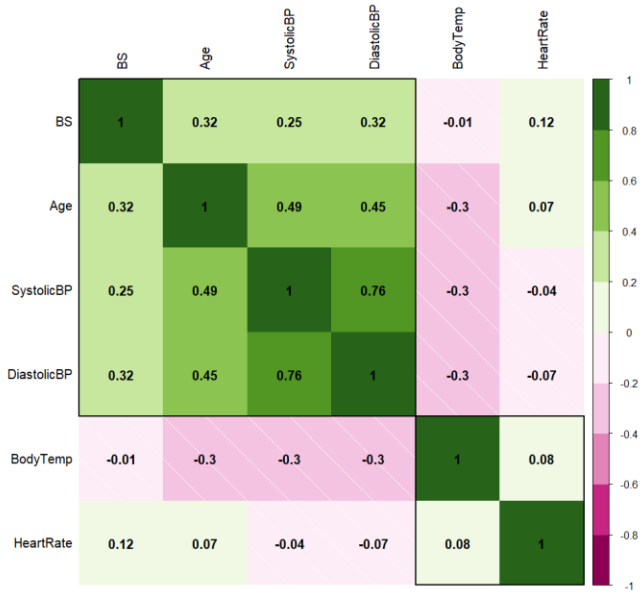
The predictors are ordered with respect to Hierarchical Clustering in to two groups. The average distance between the features determined the clusters.

BS, Age, SystolicBP and DiastolicBP are all positively related to each other. Of these predictors, SystolicBP and DiastolicBP are relatively more correlated.

Features with negative value for correlations are shaded. BodyTemp is negatively correlated to most features except for HeartRate.

```
# determine correlation between predictors

x <- cor(dat_x, method = "spearman")
```

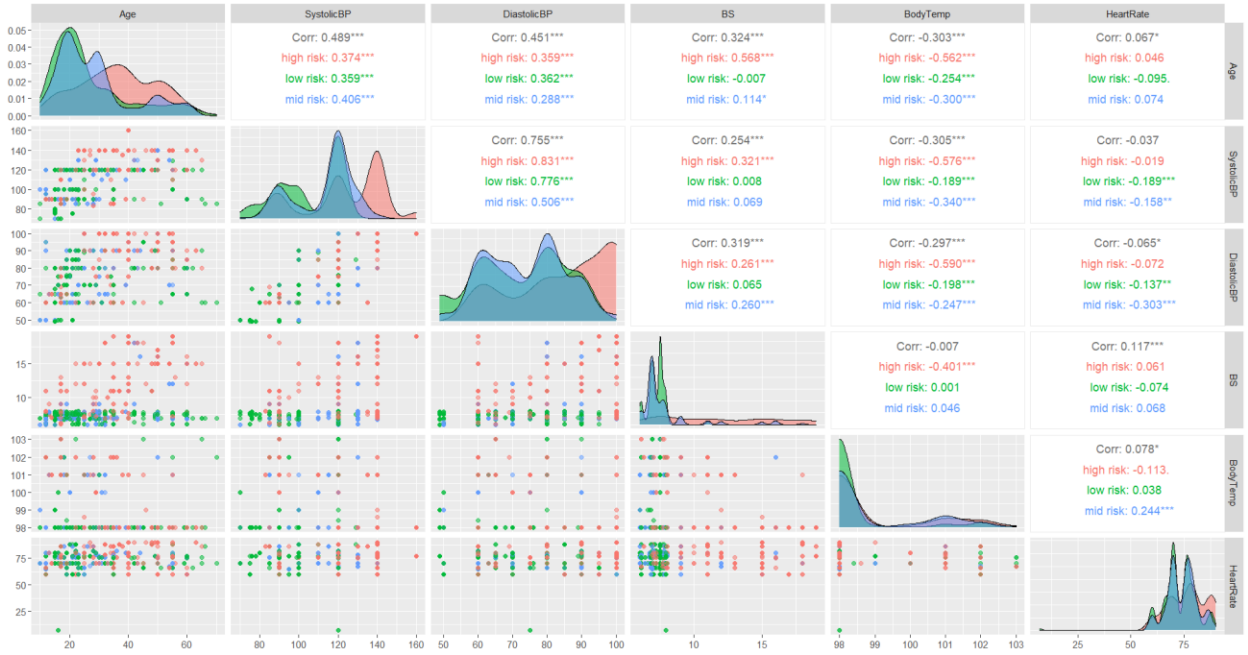
```
# plot correlation matrix
# Hierarchical Clustering is based on average distance of features

corrplot(x, method = 'shade', order = 'hclust', addrect = 2, hclust.method =
"average",
         addCoef.col ='black', col = COL2('PiYG', 10),tl.col = 'black' )
```



2.3.3. Scatter Matrix

The upper pane of Scatter Matrix shows pairwise spearman correlation index for each predictor and also for each risk level. The lower pane of matrix displays scatter plots of these predictors.

```
# plot scatter matrix

ggpairs(dat_x,
        aes(color = dat$RiskLevel ,
            alpha = 0.5), upper = list(continuous = wrap("cor", method= "spea
rman")))
```



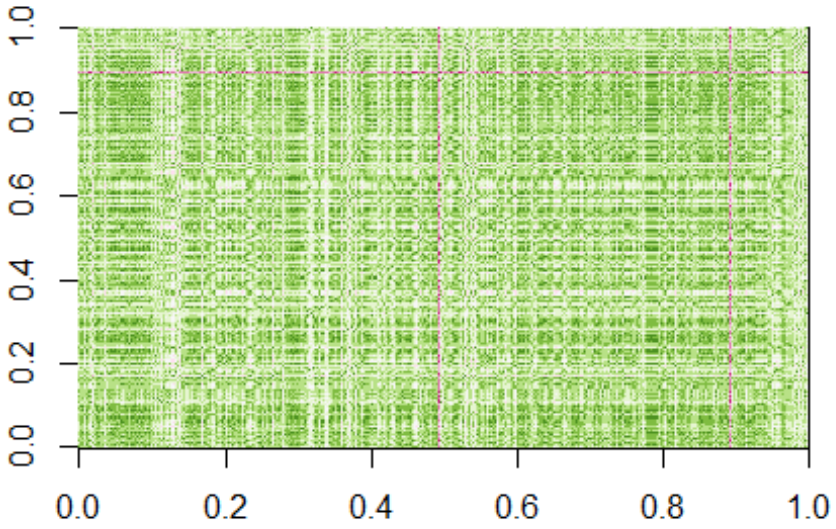
2.3.4. Principal Component Analysis

Principal component analysis (PCA) is used to reduce the dimensionality of large datasets. This data analysis technique increases large datasets interpretability without the loss of the information.

```
x <- as.matrix(dat_x)


# center entries

x_centered <- sweep(x, 2, colMeans(x))


# scale entries

x_scaled <- sweep(x_centered, 2, colSds(x_centered), FUN = "/")


# calculate distance

d <- dist(x_scaled)
```

To visualize distance vector, we plot this image.

```
# visualize distance

image(as.matrix(d), col = rev(RColorBrewer::brewer.pal(9, "PiYG")))
```



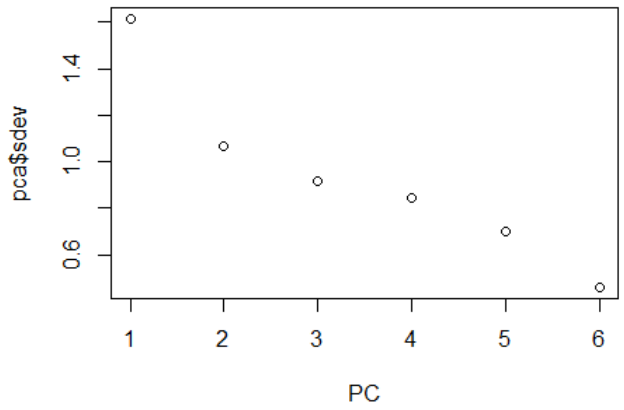

```
# perform principal component analysis

pca <- prcomp(x_scaled)

summary(pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5     PC6
## Standard deviation     1.6141 1.0676 0.9150 0.8471 0.69890 0.46036
## Proportion of Variance 0.4342 0.1900 0.1395 0.1196 0.08141 0.03532
## Cumulative Proportion  0.4342 0.6241 0.7637 0.8833 0.96468 1.00000
```

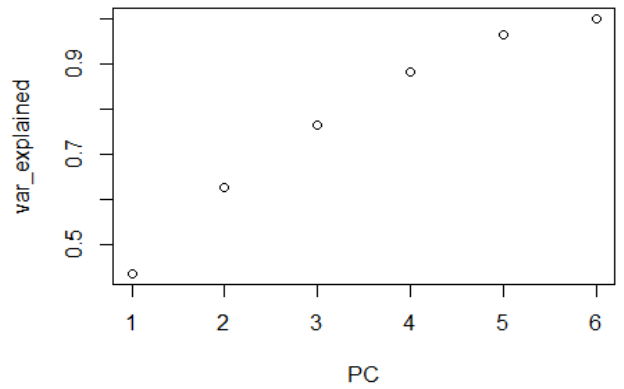
More than 50% of data is explained with just two principal components.

Standard Deviation of Principal Components reduce with each successive component.

```
# plot Standard Deviation

plot(pca$sdev, xlab = "PC")
```

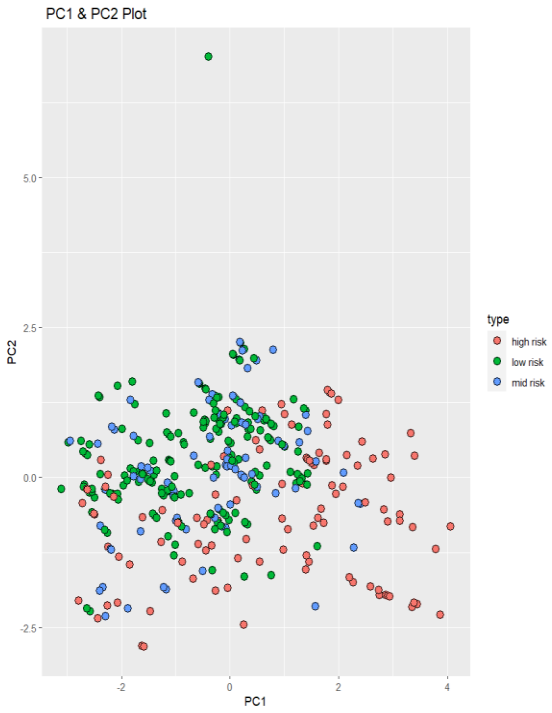



```
# calculate Variance Explained

var_explained <- cumsum(pca$sdev^2/sum(pca$sdev^2))


# plot Varaince Explained

plot(var_explained, xlab = "PC")
```



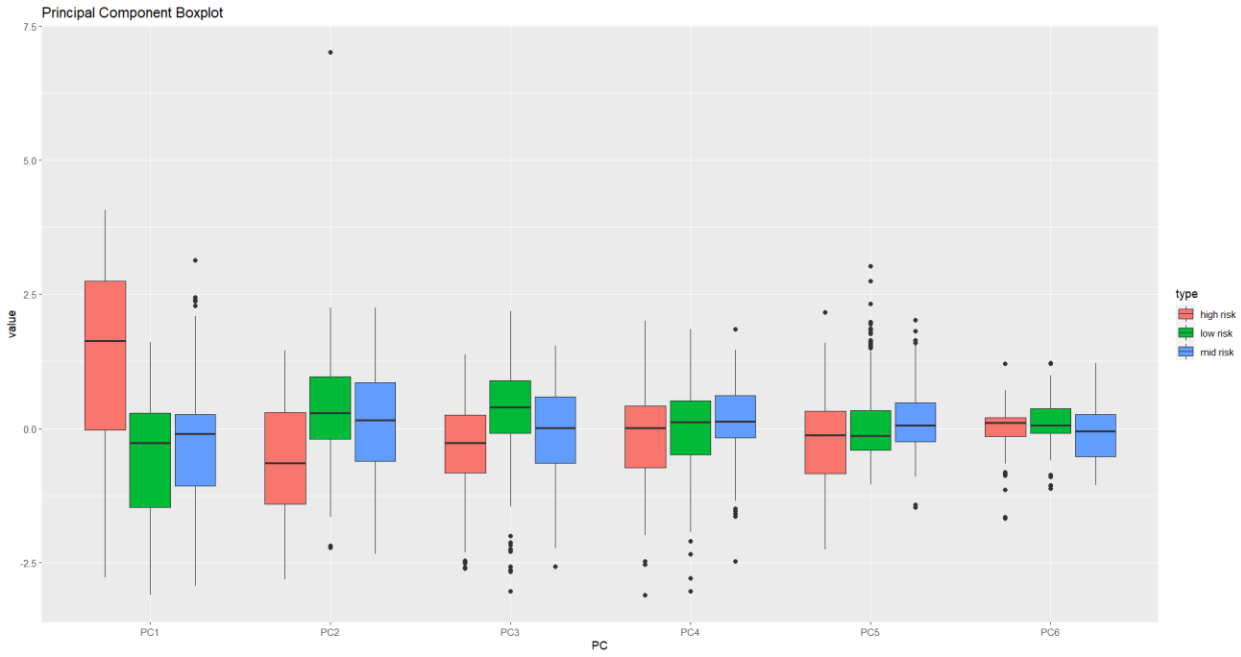
The first two components report 62.4% of total variability in data set.

```
# Plot PC1 & PC2

data.frame(pca$x[,1:2], type = dat_y) %>%
  ggplot(aes(PC1, PC2, fill = type)) +
  geom_point(cex=3, pch=21)+
  ggtitle(" PC1 & PC2 Plot") +
  coord_fixed(ratio = 1)
```



High-risk samples tend to have larger values of PC1 than low-risk and mid-risk samples.

```
# Plot Principal Component Boxplot

data.frame(type = dat_y, pca$x) %>%
      gather(key = "PC", value = "value", -type) %>%
      ggplot(aes(PC, value, fill = type)) +
      geom_boxplot() +
      ggtitle("Principal Component Boxplot ")
```



The inter quartile range of high-risk samples for PC1 is larger than for any other component.

2.4 Data Modeling

This report will use five Machine Learning algorithms. Here is a brief description of the models.

2.4.1. Quadratic Discriminant Analysis (QDA)

A quadratic discriminant analysis is widely used classification technique. It classifies a target variable from a class of independent variables into two or more classes that is multi class classification. This technique generalizes the linear discriminant analysis (LDA) classifier to the case of distinct covariance matrices among classes. A disadvantage of QDA is that it cannot be used as a dimensionality reduction technique.

2.4.2. Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis is a dimensionality reduction technique that is commonly used for supervised classification problems. The approach models the differences among samples assigned to certain groups.

The aim of the method is to maximize the ratio of the between-group variance and the within-group variance. When the value of this ratio is at its maximum, then the samples within each group have the smallest possible scatter and the groups are separated from one another the most. LDA projects the features in higher dimensional space onto a lower-dimensional space in order to avoid the curse of dimensionality.

2.4.3.K Nearest Neighbors

The K-Nearest Neighbors algorithm or k-NN is a supervised learning classifier. It uses proximity to classify or predict the group of an individual data point. The algorithm can be used for both Regression and Classification problems.

K-NN is easy to implement. There are only two parameters required to implement model: value of K and the distance function. As new training samples are added, the algorithm easily adjusts to account for any new data since all training data is stored into memory. However, the model doesn't perform well with high-dimensional data inputs and is prone to overfitting. It takes up more memory and data storage compared to other classifiers.

2.4.4. Random Forrest

Random forest is a supervised learning algorithm that is used widely in Classification and Regression problems. It combines the output of multiple decision trees to reach a single result. The algorithm's three prime parameters are node size, number of trees, and number of features sampled.

Few key advantages of Random Forrest are that it reduces the risk of overfitting and makes it easy to evaluate variable importance in the model. However, algorithm presents some challenges as well. Random Forrest algorithm is a time-consuming process. And with large datasets, it requires more resources to store data. Moreover, since it averages a lot of decision trees, the model loses interpretability.

2.4.5. Extreme Gradient Boosting – (XGBoost)

Extreme Gradient boosting is an extremely popular machine learning algorithm. Whereas random forests build an ensemble of deep independent trees, Extreme Gradient boosting algorithm builds an ensemble of shallow trees in sequence with each tree learning and improving on the previous one. Although shallow trees by themselves are rather weak predictive models, they can be boosted to produce a powerful "committee".

XGBoost is an optimized distributed gradient boosting library. It offers regularization hyperparameters that curtail overfitting. And allows users to define and optimize gradient boosting models using custom objective and evaluation criteria. A user can train an XGBoost model, save the results, and later on return to that model and continue building onto the results. It also implements early stopping so that model assessment is stopped when additional trees offer no improvement.

3. Results

3.1 Quadratic Discriminant Analysis (QDA)

```
# Model 1:  Quadratic Discriminant Analysis (QDA)


# train model

train_qda <- train(train_x, train_y, method = "qda")


# calculate predictions
qda_preds <- predict(train_qda, test_x)


# Confusion Matrix and Statistics

confusionMatrix(data = qda_preds, reference = factor(test_y))
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  high risk low risk mid risk
##   high risk        18        0        4
##   low risk          5       35       18
##   mid risk          2        2        9
##
## Overall Statistics
##
##                Accuracy : 0.6667
##                  95% CI : (0.5613, 0.7611)
##     No Information Rate : 0.3978
##     P-Value [Acc > NIR] : 1.479e-07
##
##                   Kappa : 0.4805
##
##  Mcnemar's Test P-Value : 0.0003524
##
## Statistics by Class:
##
##                      Class: high risk Class: low risk Class: mid risk
## Sensitivity                    0.7200          0.9459         0.29032
## Specificity                    0.9412          0.5893         0.93548
## Pos Pred Value                 0.8182          0.6034         0.69231
## Neg Pred Value                 0.9014          0.9429         0.72500
## Prevalence                     0.2688          0.3978         0.33333
## Detection Rate                 0.1935          0.3763         0.09677
```

```
## Detection Prevalence           0.2366          0.6237          0.13978
## Balanced Accuracy              0.8306          0.7676          0.61290

# calculate accuracy

qda_result <- mean(qda_preds == factor(test_y))


Result <- data.frame(Model = "Quadratic Discriminant Analysis (QDA)", Accurac
y = qda_result )

Result

##                                    Model  Accuracy
## 1 Quadratic Discriminant Analysis (QDA) 0.6666667
```

With Quadratic Discriminant Analysis, accuracy was just 66.6 %. Sensitivity for mid-risk samples was 29.03 %. The model didn't perform well.

3.2. Linear Discriminant Analysis (LDA)

```
# Model 2: Linear Discriminant Analysis (LDA)

# train model

train_lda <- train(train_x, train_y, method = "lda")


# calculate predictions

lda_preds <- predict(train_lda, test_x)

# Confusion Matrix and Statistics

confusionMatrix(data = lda_preds, reference = factor(test_y))

## Confusion Matrix and Statistics
##
##            Reference
## Prediction  high risk low risk mid risk
##   high risk        20        0        5
##   low risk          1       33       16
##   mid risk          4        4       10
##
## Overall Statistics
```

```
##
##                Accuracy : 0.6774
##                  95% CI : (0.5725, 0.7707)
##     No Information Rate : 0.3978
##     P-Value [Acc > NIR] : 4.708e-08
##
##                   Kappa : 0.5032
##
##  Mcnemar's Test P-Value : 0.04
##
## Statistics by Class:
##
##                      Class: high risk Class: low risk Class: mid risk
## Sensitivity                    0.8000          0.8919          0.3226
## Specificity                    0.9265          0.6964          0.8710
## Pos Pred Value                 0.8000          0.6600          0.5556
## Neg Pred Value                 0.9265          0.9070          0.7200
## Prevalence                     0.2688          0.3978          0.3333
## Detection Rate                 0.2151          0.3548          0.1075
## Detection Prevalence           0.2688          0.5376          0.1935
## Balanced Accuracy              0.8632          0.7942          0.5968
```

```
# calculate accuracy

lda_result <- mean(lda_preds == factor(test_y))

Result <- bind_rows(Result, tibble(Model = "Linear Discriminant Analysis (LDA
)",
                                   Accuracy = lda_result))
Result
```

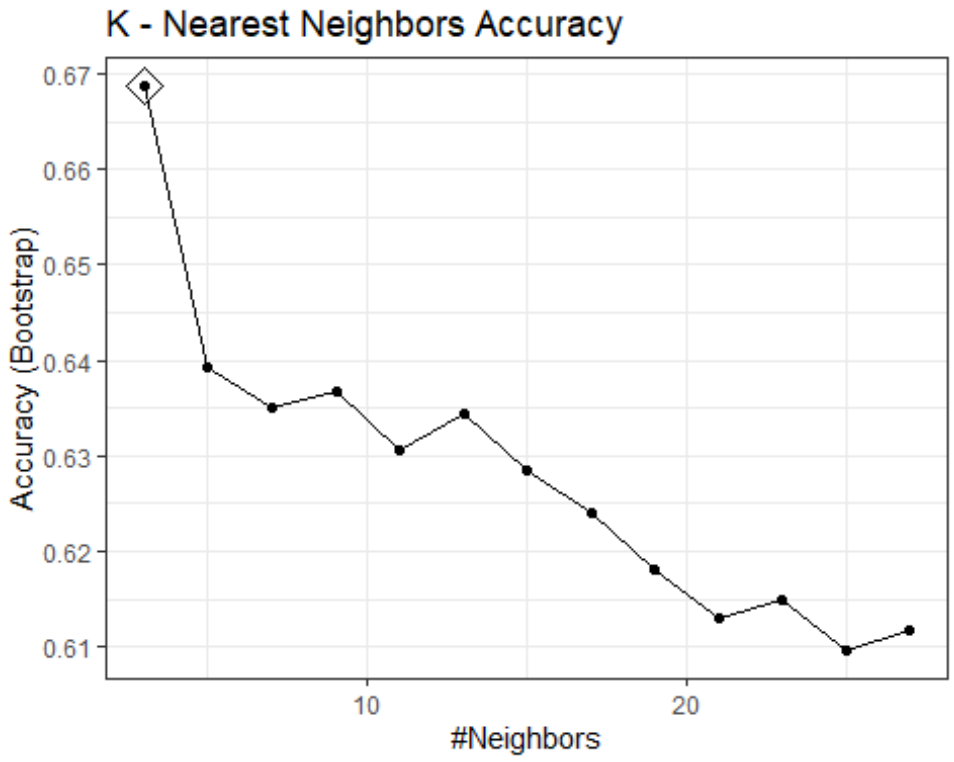
```
##                                    Model  Accuracy
## 1 Quadratic Discriminant Analysis (QDA) 0.6666667
## 2    Linear Discriminant Analysis (LDA) 0.6774194
```

The model didn't improve accuracy significantly. The Specificity for low-risk samples was 69.64% while Sensitivity for mid-risk samples was 32.36%.

3.3. K - Nearest Neighbors

```
# Model 3: K - Nearest Neighbors

# define tuning parameter k
```

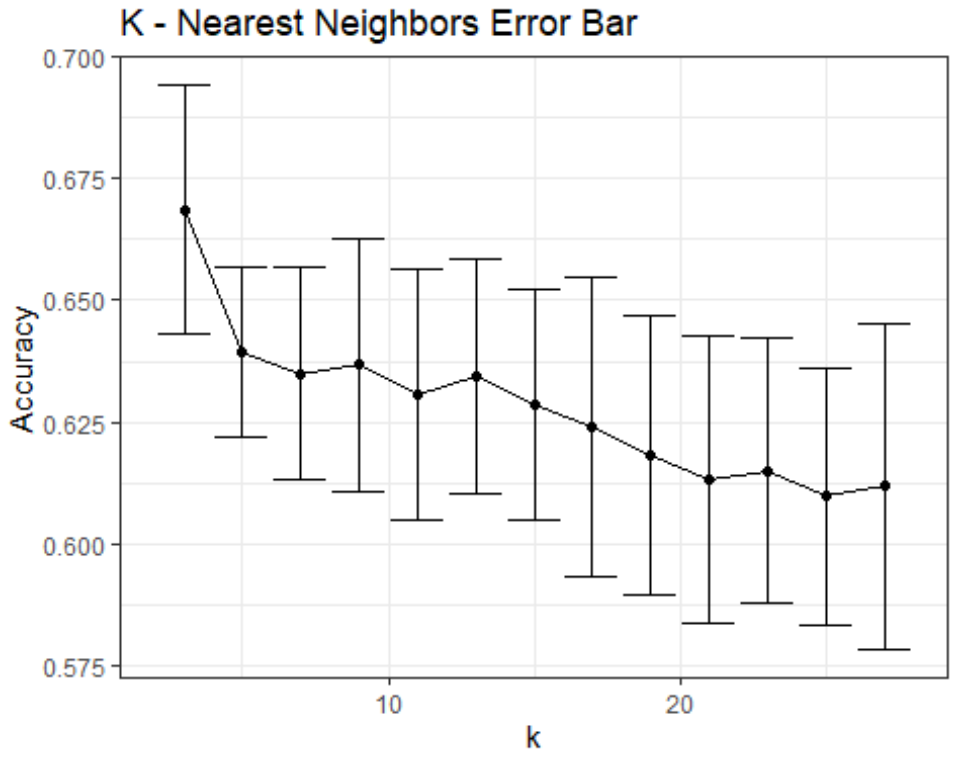
```
tuning <- data.frame(k = seq(3,27,2))

# train model

train_knn <- train(train_x, train_y, method = "knn",
                   tuneGrid = tuning)
# find best tune


train_knn$bestTune

##   k
## 1 3

# plot accuracy for each k


ggplot(train_knn, highlight = TRUE) + ggtitle("K - Nearest Neighbors Accuracy
") + theme_bw()
```



Maximum accuracy is achieved when k is equal to 3. As more neighbors were added, accuracy reduced.

Here we plot Error bars for each K.

```
# plot error bar for each k


train_knn$results %>%
  ggplot(aes(x = k, y = Accuracy)) +
  geom_line() +
  geom_point() +
  geom_errorbar(aes(x = k,
                    ymin = Accuracy - AccuracySD,
                    ymax = Accuracy + AccuracySD)) +
                    ggtitle("K - Nearest Neighbors Error Bar") +
                    theme_bw()
```




```
# training set outcome distribution of final model

train_knn$finalModel
```

```
## 3-nearest neighbor model
## Training set outcome distribution:
##
## high risk  low risk  mid risk
##       219       328       271
```

```r
# calculate predictions

knn_preds <- predict(train_knn, test_x )


# Confusion Matrix and Statistics
confusionMatrix(knn_preds, factor(test_y))
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  high risk low risk mid risk
##   high risk        19        2        3
##   low risk          4       26        5
##   mid risk          2        9       23
##
## Overall Statistics
##
##                Accuracy : 0.7312
##                  95% CI : (0.6292, 0.8179)
##     No Information Rate : 0.3978
##     P-Value [Acc > NIR] : 7.384e-11
##
##                   Kappa : 0.5921
##
##  Mcnemar's Test P-Value : 0.5704
##
## Statistics by Class:
##
##                      Class: high risk Class: low risk Class: mid risk
## Sensitivity                    0.7600          0.7027          0.7419
## Specificity                    0.9265          0.8393          0.8226
## Pos Pred Value                 0.7917          0.7429          0.6765
## Neg Pred Value                 0.9130          0.8103          0.8644
## Prevalence                     0.2688          0.3978          0.3333
## Detection Rate                 0.2043          0.2796          0.2473
## Detection Prevalence           0.2581          0.3763          0.3656
## Balanced Accuracy              0.8432          0.7710          0.7823
```

```r
# calculate accuracy
knn_result <- mean(knn_preds == factor(test_y))

Result <- bind_rows(Result, tibble(Model = "K - Nearest Neighbors",
                                   Accuracy = knn_result))
```

```
Result
```

```
##                                     Model  Accuracy
## 1 Quadratic Discriminant Analysis (QDA) 0.6666667
## 2    Linear Discriminant Analysis (LDA) 0.6774194
## 3                 K - Nearest Neighbors 0.7311828
```

The accuracy improved by approximately 5% from Linear Discriminant Analysis.

3.4. Random Forrest

```
#Model 4: Random Forest

# define tuning parameter - number of variables randomly sampled as candidate
s at each split

tuning <- data.frame(mtry = c(3, 5, 7))

# train model
train_rf <- train(train_x, train_y, method = "rf", tuneGrid = tuning,
              importance = TRUE)


# find best tune

train_rf$bestTune
```

```
##   mtry
## 1    3
```

```
# calculate predictions

rf_preds <- predict(train_rf, test_x)

#Confusion Matrix and Statistics

confusionMatrix(rf_preds, factor(test_y))
```

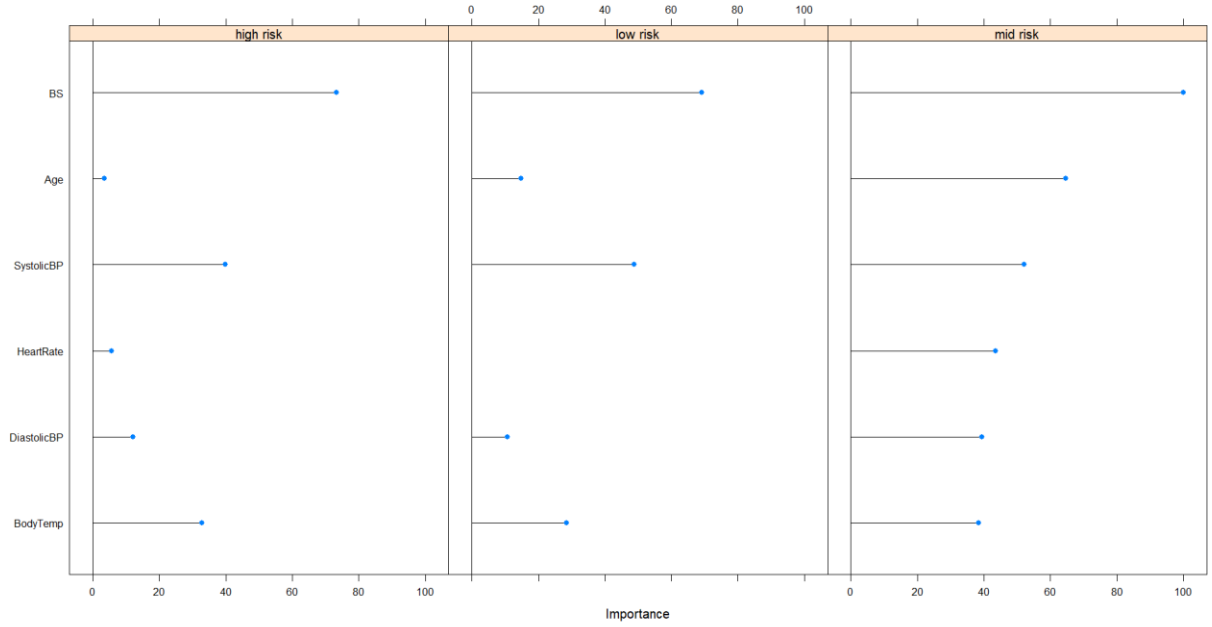
```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  high risk low risk mid risk
##   high risk        23        0        3
##   low risk          0       31        4
##   mid risk          2        6       24
##
## Overall Statistics
##
##                Accuracy : 0.8387
##                  95% CI : (0.748, 0.9068)
##     No Information Rate : 0.3978
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7558
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: high risk Class: low risk Class: mid risk
## Sensitivity                    0.9200          0.8378          0.7742
## Specificity                    0.9559          0.9286          0.8710
## Pos Pred Value                 0.8846          0.8857          0.7500
## Neg Pred Value                 0.9701          0.8966          0.8852
## Prevalence                     0.2688          0.3978          0.3333
## Detection Rate                 0.2473          0.3333          0.2581
## Detection Prevalence           0.2796          0.3763          0.3441
## Balanced Accuracy              0.9379          0.8832          0.8226
```

```
# calculate accuracy


rf_result <- mean(rf_preds == factor(test_y))


# evaluate Variable Importance

varImp(train_rf)
```

```
## rf variable importance
##
##   variables are sorted by maximum importance across the classes
##             high risk low risk mid risk
## BS             73.342    69.26   100.00
## Age             3.400    14.83    64.58
## SystolicBP     39.758    48.86    52.12
## HeartRate       5.722     0.00    43.47
```

```
## DiastolicBP    12.089    10.72    39.41
## BodyTemp       32.896    28.52    38.48
```

```
# plot Variable Importance

plot(varImp(train_rf))
```



Variables are sorted by maximum importance across the risk intensity levels. BS and Age are two most important features in determining risk intensity.

```
Result <- bind_rows(Result, tibble(Model = "Random Forrest",
                                   Accuracy = rf_result))
Result
```

```
##                                    Model  Accuracy
## 1 Quadratic Discriminant Analysis (QDA) 0.6666667
## 2    Linear Discriminant Analysis (LDA) 0.6774194
## 3                 K - Nearest Neighbors 0.7311828
## 4                         Random Forrest 0.8387097
```

So far, highest accuracy is achieved with Random Forrest approach. It made a significant improvement of 10% from accuracy that was estimated with K-NN approach.

3.5. Extreme Gradient Boosting – (XGBoost)

```
# Model 5: Extreme Gradient Boosting (xgboost)


# convert train data in to required class

train_x <- as.matrix(train_x)
train_y <- as.integer(factor(train$RiskLevel)) - 1

# convert test data in to required class

test_x <- as.matrix(test_x)
test_y <- as.integer(factor(test$RiskLevel)) - 1


#  define train data input as dense matrix

xgb_train <- xgb.DMatrix(data = train_x, label = train_y)


# define test data input as dense matrix

xgb_test <- xgb.DMatrix(data = test_x, label = test_y)


# tune parameters
# gbtree - tree is grown one after other and attempts to reduce misclassifica
tion rate in subsequent iterations
# objective is multiclassification using softmax objective which returns pred
icted class probabilities
# evaluate model's accuracy with  multiclass logloss function

xgb_params <- list(
  booster = "gbtree",
  objective = "multi:softprob",
  eval_metric = "mlogloss",
  num_class = length(levels(factor(train$RiskLevel))))

# train model

xgb_model <- xgb.train(
  params = xgb_params,
  data = xgb_train,
  verbose = 1,
  nrounds = 200)
```

```r
# calculate predictions

xgb_preds <- predict(xgb_model, as.matrix(test_x), reshape = TRUE)

xgb_preds <- as.data.frame(xgb_preds)

colnames(xgb_preds) <- levels(factor(test$RiskLevel))


# determine Predicted Risk

Predicted_Risk <- apply(xgb_preds, 1, function(y) colnames(xgb_preds)[which.m
ax(y)])

# determine Actual Risk

Actual_Risk <- levels(factor(test$RiskLevel))[test_y + 1]


# Confusion Matrix and Statistics

confusionMatrix(factor(Predicted_Risk), factor(Actual_Risk))
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  high risk low risk mid risk
##   high risk        23        0        3
##   low risk          0       31        3
##   mid risk          2        6       25
##
## Overall Statistics
##
##                Accuracy : 0.8495
##                  95% CI : (0.7603, 0.9152)
##     No Information Rate : 0.3978
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7723
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: high risk Class: low risk Class: mid risk
## Sensitivity                    0.9200          0.8378          0.8065
## Specificity                    0.9559          0.9464          0.8710
## Pos Pred Value                 0.8846          0.9118          0.7576
```

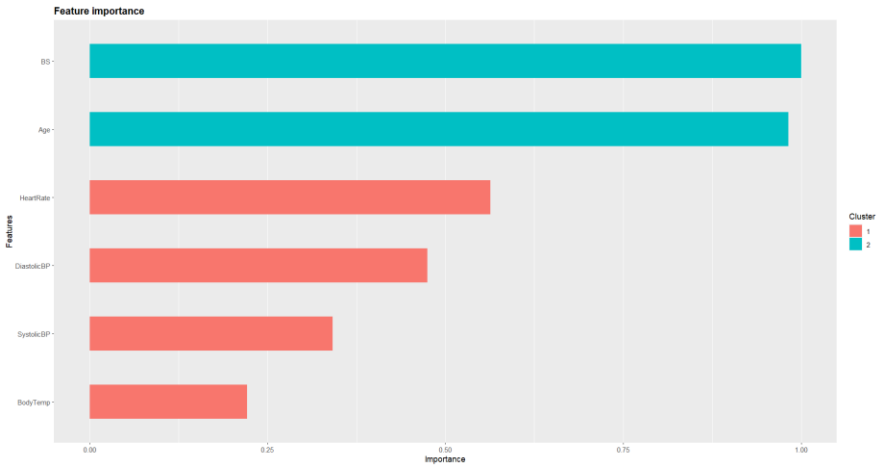
```
## Neg Pred Value                 0.9701          0.8983          0.9000
## Prevalence                     0.2688          0.3978          0.3333
## Detection Rate                 0.2473          0.3333          0.2688
## Detection Prevalence           0.2796          0.3656          0.3548
## Balanced Accuracy              0.9379          0.8921          0.8387
```

```
# calculate accuracy

xgb_result <- mean(Predicted_Risk == Actual_Risk)


# determine feature importance

importance_matrix <- xgb.importance(feature_names = colnames(xgb_train), mode
l = xgb_model)

importance_matrix
```

```
##        Feature       Gain      Cover  Frequency
## 1:          BS 0.44598715 0.31385757 0.27906035
## 2:  SystolicBP 0.20137232 0.14278662 0.09538275
## 3:         Age 0.13640926 0.23031711 0.27409883
## 4:    BodyTemp 0.07298096 0.07422698 0.06176590
## 5: DiastolicBP 0.07258502 0.09748345 0.13254354
## 6:   HeartRate 0.07066529 0.14132828 0.15714864
```

To plot feature importance, we'll order variables in their relative importance from most important predictor.

```
# plot feature importance

xgb.ggplot.importance(importance_matrix,measure = "Frequency",rel_to_first =
TRUE)
```



The plot shows that BS is most important predictor followed by Age. The least important predictor in determining risk level is BodyTemp.

```
Result <- bind_rows(Result, tibble(Model = "Extreme Gradient Boosting (xgboos
t)",
                                   Accuracy = xgb_result))
Result
```

```
##                                    Model  Accuracy
## 1 Quadratic Discriminant Analysis (QDA) 0.6666667
## 2    Linear Discriminant Analysis (LDA) 0.6774194
## 3                 K - Nearest Neighbors 0.7311828
## 4                         Random Forrest 0.8387097
## 5   Extreme Gradient Boosting (xgboost) 0.8494624
```

We estimated Risk levels with several Machine Learning algorithms. The best accuracy is achieved with Extreme Gradient Boosting. 85% of data set was correctly predicted. Therefore, we'll use this approach to train and test our final model.

3.6. Final Model - Extreme Gradient Boosting – (XGBoost)

```
#  Final Model - Extreme Gradient Boosting (xgboost)

# convert dat data in to required class

dat_x <- as.matrix(dat_x)
dat_y <- as.integer(factor(dat$RiskLevel)) - 1



# convert Validation data in to required class

val_x <- as.matrix(val_x)
val_y <- as.integer(factor(Validation$RiskLevel)) - 1


# define dat data input as dense matrix

xgb_train <- xgb.DMatrix(data = dat_x, label = dat_y)

# define Validation data input as dense matrix

xgb_test <- xgb.DMatrix(data = val_x, label = val_y)

# tune parameters
# gbtree - tree is grown one after other and attempts to reduce misclassifica
```

```
tion rate in subsequent iterations
# objective is multiclassification using softmax objective which returns pred
icted class probabilities
# evaluate model's accuracy with  multiclass logloss function

xgb_params <- list(
  booster = "gbtree",
  objective = "multi:softprob",
  eval_metric = "mlogloss",
  num_class = length(levels(factor(dat$RiskLevel))))

# train model

xgb_model <- xgb.train(
  params = xgb_params,
  data = xgb_train,
  verbose = 1,
  nrounds = 200)


# calculate predictions

xgb_preds <- predict(xgb_model, as.matrix(val_x), reshape = TRUE)

xgb_preds <- as.data.frame(xgb_preds)

colnames(xgb_preds) <- levels(factor(Validation$RiskLevel))

# determine Predicted Risk

Predicted_Risk <- apply(xgb_preds, 1, function(y) colnames(xgb_preds)[which.m
ax(y)])

# determine Actual Risk

Actual_Risk <- levels(factor(Validation$RiskLevel))[val_y + 1]


# Confusion Matrix and Statistics

confusionMatrix(factor(Predicted_Risk), factor(Actual_Risk))

## Confusion Matrix and Statistics
##
##            Reference
## Prediction  high risk low risk mid risk
##   high risk        27        0        1
##   low risk          1       36        6
##   mid risk          0        5       27
```

```
##
## Overall Statistics
##
##                Accuracy : 0.8738
##                  95% CI : (0.7938, 0.9311)
##     No Information Rate : 0.3981
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.808
##
##  Mcnemar's Test P-Value : 0.5538
##
## Statistics by Class:
##
##                      Class: high risk Class: low risk Class: mid risk
## Sensitivity                    0.9643          0.8780          0.7941
## Specificity                    0.9867          0.8871          0.9275
## Pos Pred Value                 0.9643          0.8372          0.8438
## Neg Pred Value                 0.9867          0.9167          0.9014
## Prevalence                     0.2718          0.3981          0.3301
## Detection Rate                 0.2621          0.3495          0.2621
## Detection Prevalence           0.2718          0.4175          0.3107
## Balanced Accuracy              0.9755          0.8826          0.8608
```

```
# calculate accuracy

Final_xgb_result <- mean(Predicted_Risk == Actual_Risk)
```

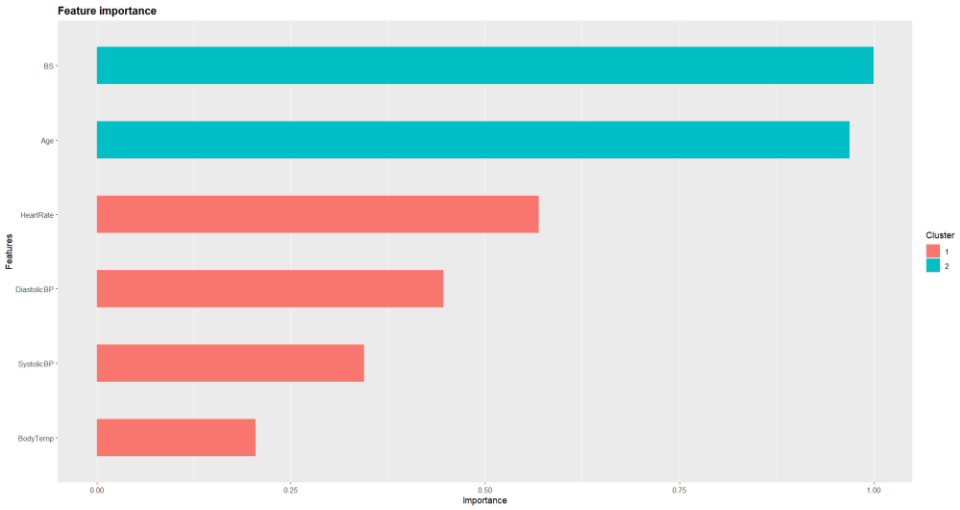
We list features ordered by their importance.

```
# determine feature importance

importance_matrix <- xgb.importance(feature_names = colnames(xgb_train), mode
l = xgb_model)

importance_matrix
```

```
##        Feature       Gain      Cover  Frequency
## 1:          BS 0.44956859 0.32693612 0.28300947
## 2:  SystolicBP 0.19943515 0.13961636 0.09735924
## 3:         Age 0.14555375 0.22607804 0.27433981
## 4:   HeartRate 0.07290708 0.13851684 0.16103637
## 5: DiastolicBP 0.06926252 0.09102514 0.12635775
## 6:    BodyTemp 0.06327291 0.07782749 0.05789736
```

To plot feature importance, we'll order variables in their relative importance from most important predictor.

```
# plot feature importance

xgb.ggplot.importance(importance_matrix,measure = "Frequency",rel_to_first =
TRUE)
```



The plot shows that BS is most important predictor followed by Age. The least important predictor in determining risk level is BodyTemp.

```
Result <- bind_rows(Result, tibble(Model = "Final Model - Extreme Gradient Bo
osting (xgboost)",
                                   Accuracy = Final_xgb_result))
Result
```

```
##                                                Model Accuracy
## 1            Quadratic Discriminant Analysis (QDA) 0.6666667
## 2               Linear Discriminant Analysis (LDA) 0.6774194
## 3                          K - Nearest Neighbors 0.7311828
## 4                                Random Forrest 0.8387097
## 5              Extreme Gradient Boosting (xgboost) 0.8494624
## 6 Final Model - Extreme Gradient Boosting (xgboost) 0.8737864
```

Our final model accomplished an accuracy of 87.37%.

4. Conclusion

The report documented insights from modeling Maternal Health Risk Data with several Machine Learning algorithms. We began with Quadratic Discriminant Analysis. The classification technique achieved just 66.6% accuracy in predicting risk. We then used dimensionality reduction technique – Linear Discriminant Analysis to improve our estimate. With no significant gain in our results, a supervised learning classifier approach, K-NN, was adopted. The approach accomplished a 73% accuracy. In pursuit of achieving best accuracy, risk levels were also predicted with Random Forest approach. Finally, with Extreme Gradient Boosting – (XGBoost) we achieved our best accuracy.