Maternal Health Risk project

Signe Arias

July 2023 - June 2024

Contents

1 Introduction

1.1 Setting the scene

Overall maternal mortality rates in the world are still very high. According to the World Health Organisation[1] most of these occur in low and lower middle-income countries with highest incidence in the Sub-Saharan Africa and Southern Asia. Most women die due to very preventable issues and better access to healthcare would remedy this.

1.2 Aim of the project

The aim of this project is to develop a predictor on the level of maternal mortality using the minimum amount of input data. The less data is required for the predictor, the more women can be reached and saved by freeing up resources.

1.3 Maternal Health Risk dataset[2]

The Maternal Health Risk dataset was collected by a research team for the purpose of monitoring pregnant women in remote areas in order to reduce mortality rates. The data is based on a rural area in Bangladesh and was collected from various sources such as hospitals, community clinics etc. The data was collected from the individual women by giving them wearable sensing enabled technology.

1.4 Data layout

The data contains 1014 entries and no personal information such as names was collected, only the relevant information – age and health markers. The health markers that were collected are: systolic and diastolic blood pressure, blood glucose level, body temperature and heart rate. Then based on these parameters each woman was assigned a risk level – low, medium or hight.

1.5 Key analysis steps

To begin with, I wanted to familiarise myself with the data provided. I did so by looking at the headings and then the summary of the data. The data seems fairly straightforward and contains just seven columns. It contains data such as age, systolic and diastolic blood pressures as well as glucose readings, body temperature and heart rate.

2 Analysis

2.1 Data preview

Firstly, the necessary libraries were loaded and latex installed. Then the csv file containing the data was uploaded.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
```

```
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(knitr)
library(readr)

if (!require(tinytex)) install.packages("tinytex")
```

```
## Loading required package: tinytex
```

```
library(tinytex)

url <- "https://raw.githubusercontent.com/signearias/MaternalHealth/5c76580bcac266c6145c6a7fb30ca301e0a8
data <- read_csv(url)
```

```
## 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 message.
```

I then wanted to have a quick look at the first few rows of the data, check for any missing values and unique values. And then check the summary.

```
head(data, 10)
```

```
## # A tibble: 10 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
##  7    23        130          70  7.01       98        78 mid risk
##  8    35         85          60 11         102        86 high risk
##  9    32        120          90  6.9        98        70 mid risk
## 10    42        130          80 18          98        70 high risk
```

```
sapply(data, function(x) sum(is.na(x)))
```

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

```
sapply(data, function(x) length(unique(x)))
```

```
##          Age  SystolicBP DiastolicBP          BS    BodyTemp   HeartRate
##           50          19          16          29           8          16
##   RiskLevel
##            3
```

```
summary(data)
```

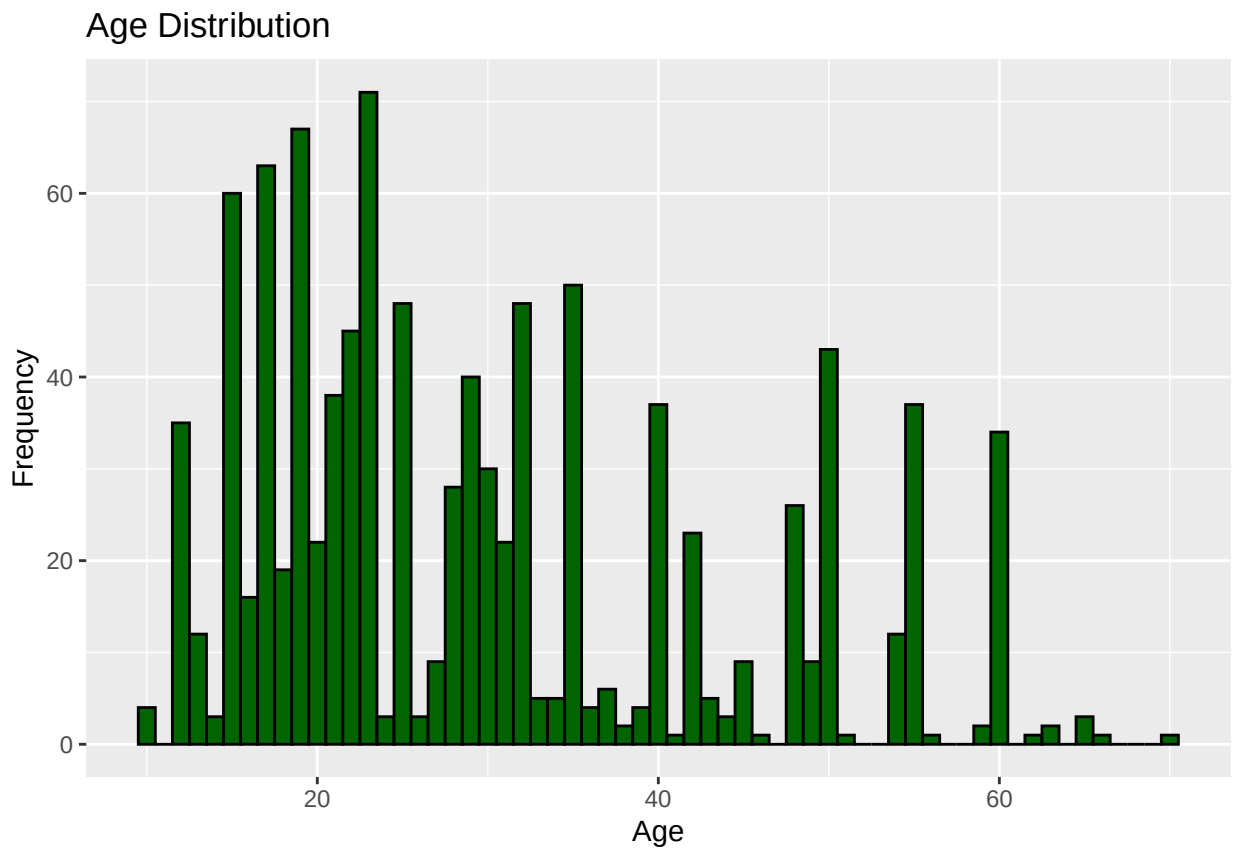
```
##       Age          SystolicBP     DiastolicBP          BS
##  Min.   :10.00   Min.   : 70.0   Min.   : 49.00   Min.   : 6.000
##  1st Qu.:19.00   1st Qu.:100.0   1st Qu.: 65.00   1st Qu.: 6.900
##  Median :26.00   Median :120.0   Median : 80.00   Median : 7.500
##  Mean   :29.87   Mean   :113.2   Mean   : 76.46   Mean   : 8.726
##  3rd Qu.:39.00   3rd Qu.:120.0   3rd Qu.: 90.00   3rd Qu.: 8.000
##  Max.   :70.00   Max.   :160.0   Max.   :100.00   Max.   :19.000
##     BodyTemp        HeartRate     RiskLevel
##  Min.   : 98.00   Min.   : 7.0   Length:1014
##  1st Qu.: 98.00   1st Qu.:70.0   Class :character
##  Median : 98.00   Median :76.0   Mode  :character
##  Mean   : 98.67   Mean   :74.3
##  3rd Qu.: 98.00   3rd Qu.:80.0
##  Max.   :103.00   Max.   :90.0
```

From the above analysis I learned that the data had the expected headings such as age and blood pressure. There are no missing values and the number of unique values in each column makes sense. For example, age has 50 different values and body temperature just eight. From the summary I can see that there are some potential outliers in age and heart rate which I will check for by performing some basic visualisation.

2.2 Basic visualisation

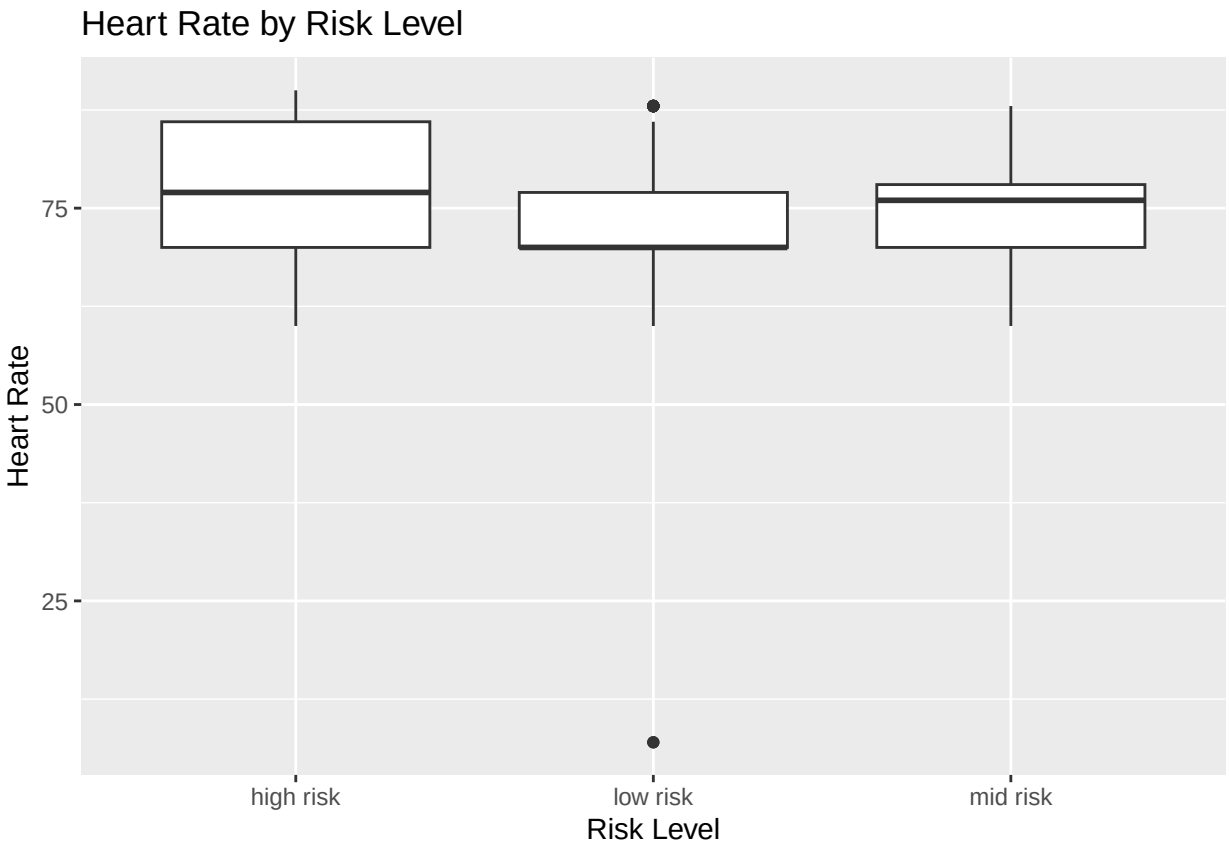
I started by running a histogram to see what the distribution of age is.

```
ggplot(data, aes(x = Age)) +
  geom_histogram(binwidth = 1, fill = "darkgreen", color = "black") +
  labs(title = "Age Distribution", x = "Age", y = "Frequency")
```



Looking at the histogram I can see that there are several values for women aged below 16 and over 60. I will be excluding these values when running my algorithms as I believe these ages come with their own challenges beyond the markers collected in this data set. I then looked at the heart rate.

```
ggplot(data, aes(y = HeartRate, x = RiskLevel)) +
  geom_boxplot() +
  labs(title = "Heart Rate by Risk Level", x = "Risk Level", y = "Heart Rate")
```



As shown in the box plot there are some abnormally low heart rates which will also need to be excluded from the analysis.

Finally, I looked at the distribution of the risk levels to see if the data was skewed towards low or high but it appears to be approximately equally distributed.

2.3 Data cleaning

As mentioned in the basic visualisation section above I will be excluding entries where the age is below 16 and above 60.

```
clean_data <- subset(data, Age >= 16 & Age <= 49)
```

I will also exclude entries where the heart rate is recorded below 50.

```
clean_data <- subset(clean_data, HeartRate >= 50)
```

When initially inspecting the data, I noticed that the Risk Level values are recorded as low, medium and high which will make comparison and visualisation hard and therefore I re-wrote them to be displayed as numeric values where low risk is a value of 1, medium risk is 2 and high risk is 3.

```
clean_data$RiskLevelNumeric <- as.numeric(factor(clean_data$RiskLevel, levels = c("low risk", "mid risk
```

I then ran a summary of my clean data to see that all the outliers were indeed removed and that the Risk Level values were now numeric.

```r
summary(clean_data)
```

```
##       Age          SystolicBP     DiastolicBP          BS
##  Min.   :16.00   Min.   : 70.0   Min.   : 50.0   Min.   : 6.000
##  1st Qu.:21.00   1st Qu.:100.0   1st Qu.: 65.0   1st Qu.: 6.900
##  Median :25.00   Median :120.0   Median : 80.0   Median : 7.500
##  Mean   :27.74   Mean   :114.1   Mean   : 77.1   Mean   : 8.431
##  3rd Qu.:32.25   3rd Qu.:120.0   3rd Qu.: 90.0   3rd Qu.: 7.900
##  Max.   :49.00   Max.   :160.0   Max.   :100.0   Max.   :19.000
##     BodyTemp       HeartRate      RiskLevel         RiskLevelNumeric
##  Min.   : 98.0   Min.   :60.00   Length:760         Min.   :1.00
##  1st Qu.: 98.0   1st Qu.:70.00   Class :character   1st Qu.:1.00
##  Median : 98.0   Median :76.00   Mode  :character   Median :2.00
##  Mean   : 98.7   Mean   :75.04                      Mean   :1.88
##  3rd Qu.: 98.0   3rd Qu.:80.00                      3rd Qu.:3.00
##  Max.   :103.0   Max.   :90.00                      Max.   :3.00
```

2.4 Correlation analysis

In order to choose what health markers to include / exclude from my algorithm I ran a correlation between each marker and the risk level.

```r
numeric_data <- clean_data[, sapply(clean_data, is.numeric)]
correlation_matrix <- cor(numeric_data)
correlation_with_risk <- correlation_matrix[, "RiskLevelNumeric"]

correlation_df <- tibble(Feature = names(correlation_with_risk), Correlation = correlation_with_risk)
correlation_df %>% knitr::kable()
```

Feature	Correlation
Age	0.2636896
SystolicBP	0.3424675
DiastolicBP	0.2700112
BS	0.5209228
BodyTemp	0.1840668
HeartRate	0.2708038
RiskLevelNumeric	1.0000000

The analysis showed that Body Temperature had the least correlation and Blood Sugar the highest. This makes sense as Body Temperature had only eight different values and is a marker that we expect doesn't vary too much from person to person. The opposite is true for Blood Sugar as this can vary greatly from person to person and highly depends on when the measurement was taken – first thing in the morning after a meal etc.

2.5 Create training and validation sets

Now it's time to create the training and validation data sets. Firstly, I set a random seed for reproducibility and then displayed the number of rows I had. When splitting my data into training and validation sets, I wanted to make sure that I had enough data in the validation set.

```r
set.seed(12345)
num_rows <- nrow(clean_data)
num_rows
```

```
## [1] 760
```

I saw that I had 760 rows of data and as having about a 100 in the validation set makes sense, I randomly split my data into 85% / 15% training vs validation.

```r
train_indices <- sample(seq_len(num_rows), size = 0.85 * num_rows)
training_data <- clean_data[train_indices, ]
validation_data <- clean_data[-train_indices, ]
```

I then sense checked the number of rows I had in each set.

```r
cat("Number of rows in clean data:", nrow(clean_data), "\n")
```

```
## Number of rows in clean data: 760
```

```r
cat("Number of rows in training data:", nrow(training_data), "\n")
```

```
## Number of rows in training data: 646
```

```r
cat("Number of rows in validation data:", nrow(validation_data), "\n")
```

```
## Number of rows in validation data: 114
```

To further inspect the validity of my 85/15 split I wanted to make sure that the ratios of risk levels were approximately equal. I therefore installed a package that allowed me to do so.

```r
if (!require(patchwork)) install.packages("patchwork")
```

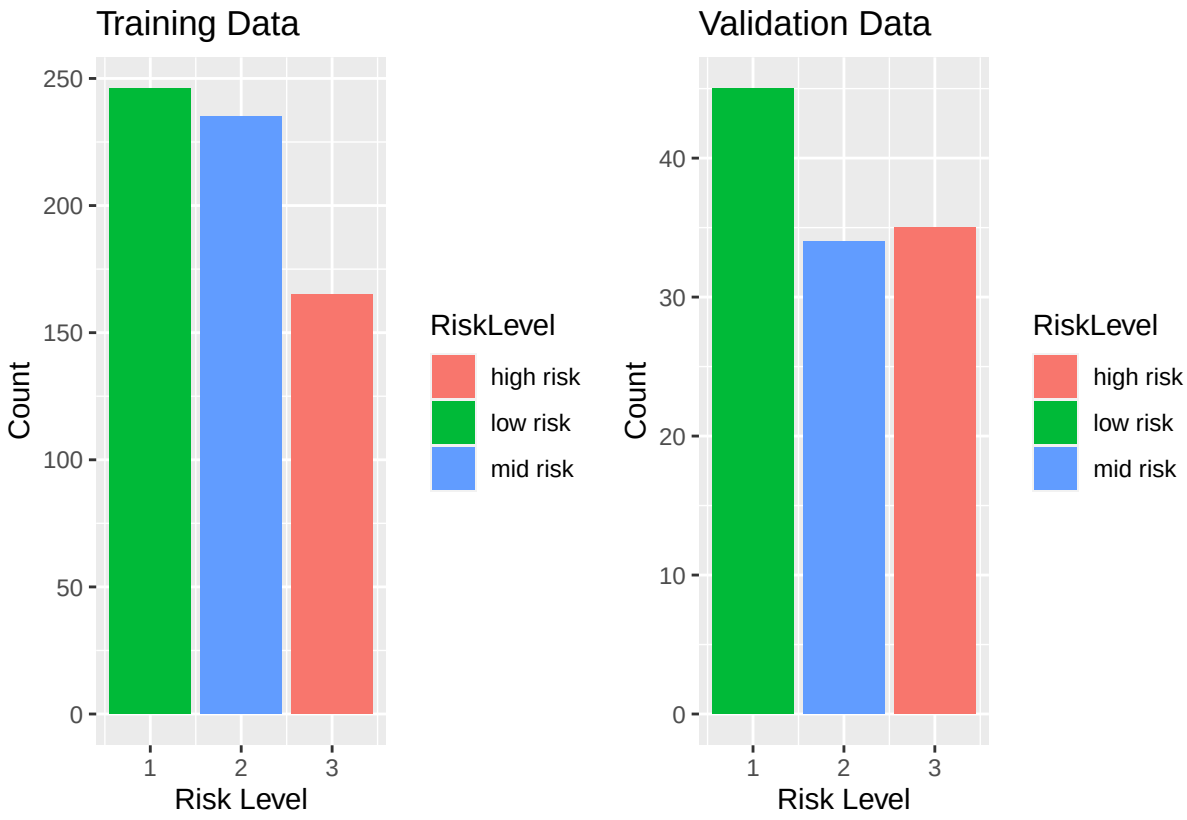
```
## Loading required package: patchwork
```

```
## Warning: package 'patchwork' was built under R version 4.3.3
```

```r
library(patchwork)
```

And then plotted the individual graphs and displayed the combined one.

```r
plot_training <- ggplot(training_data, aes(x = RiskLevelNumeric, fill = RiskLevel)) +
  geom_bar() +
  labs(title = "Training Data", x = "Risk Level", y = "Count")
plot_validation <- ggplot(validation_data, aes(x = RiskLevelNumeric, fill = RiskLevel)) +
  geom_bar() +
  labs(title = "Validation Data", x = "Risk Level", y = "Count")
combined_plot <- plot_training + plot_validation
combined_plot
```



2.6 Choose and test various models

As this problem requires supervised multi-label classification of a small amount (<100k) of labeled data, I decided to use 4 simple classifiers that provided a balance between ease of use and speed. These models to test are: Decision Tree, SVM, Linear Regression and Random Forest.

Once the different models were selected, I installed the required packages and libraries.

```
if (!require(rpart)) install.packages("rpart")
```

```
## Loading required package: rpart
```

```
## Warning: package 'rpart' was built under R version 4.3.3
```

```
if (!require(e1071)) install.packages("e1071")
```

```
## Loading required package: e1071
```

```
## Warning: package 'e1071' was built under R version 4.3.3
```

```
if (!require(randomForest)) install.packages("randomForest")
```

```
## Loading required package: randomForest
```

```
## Warning: package 'randomForest' was built under R version 4.3.2
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(rpart)
library(e1071)
library(randomForest)
```

I then trained each of the models using the training data.

```
decision_tree_model <- rpart(RiskLevelNumeric ~ ., data = training_data, method = "class")
svm_model <- svm(RiskLevelNumeric ~ ., data = training_data, kernel = "linear")
linear_regression_model <- lm(RiskLevelNumeric ~ ., data = training_data)
random_forest_model <- randomForest(RiskLevelNumeric ~ ., data = training_data)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values.  Are you sure you want to do regression?
```

I then generated predictions by running the models against the validation set.

```
dt_predictions <- predict(decision_tree_model, validation_data, type = "class")
svm_predictions <- predict(svm_model, validation_data)
linear_regression_predictions <- round(predict(linear_regression_model, validation_data))
random_forest_predictions <- predict(random_forest_model, validation_data)
```

And finally tested each for accuracy, where accuracy is defined as the proportion of predictions that were correct.

```
dt_accuracy <- sum(dt_predictions == validation_data$RiskLevelNumeric) / nrow(validation_data)
cat("Decision Tree Accuracy:", dt_accuracy, "\n")
```

```
## Decision Tree Accuracy: 1
```

```
svm_accuracy <- sum(svm_predictions == validation_data$RiskLevelNumeric) / nrow(validation_data)
cat("SVM Accuracy:", svm_accuracy, "\n")
```

```
## SVM Accuracy: 0
```

```
linear_regression_accuracy <- sum(linear_regression_predictions == validation_data$RiskLevelNumeric) /
  nrow(validation_data)
cat("Linear Regression Accuracy:", linear_regression_accuracy, "\n")
```

```
## Linear Regression Accuracy: 1
```

```
random_forest_accuracy <- sum(random_forest_predictions == validation_data$RiskLevelNumeric) /
  nrow(validation_data)
cat("Random Forest Accuracy:", random_forest_accuracy, "\n")
```

```
## Random Forest Accuracy: 0.02631579
```

For easy comparison I created a table that displays the four different models and their accuracies.

```
accuracies <- c(dt_accuracy, svm_accuracy, linear_regression_accuracy, random_forest_accuracy)
names(accuracies) <- c("Decision Tree", "SVM", "Linear Regression", "Random Forest")

accuracies_df <- tibble(Model = names(accuracies), Accuracy = accuracies)
accuracies_df %>% knitr::kable()
```

Model	Accuracy
Decision Tree	1.0000000
SVM	0.0000000
Linear Regression	1.0000000
Random Forest	0.0263158

From the table above we can see that both a decision tree and linear regression provide the best accuracy.

2.7 Re-test with fewer health markers

As mentioned in my introduction it would be beneficial to know if less data can be collected to make it easier and quicker and therefore reach more women. As the marker with the least correlation was shown to be Body Temperature, this is the parameter I chose to exclude.

New training and validation sets were created to exclude Body Temperature.

```
training_data_subset <- training_data[, setdiff(names(training_data), "BodyTemp")]
validation_data_subset <- validation_data[, setdiff(names(validation_data), "BodyTemp")]
```

I then decided to use the decision tree model to test my hypothesis. (Linear regression was also shown to lead to high accuracy however course instructions asked us to use something more advanced than linear regression.)

I trained, evaluated and tested the accuracy of the model.

```
decision_tree_model_2 <- rpart(RiskLevelNumeric ~ ., data = training_data_subset, method = "class")
dt_predictions_2 <- predict(decision_tree_model_2, validation_data_subset, type = "class")
dt_accuracy_2 <- sum(dt_predictions_2 == validation_data_subset$RiskLevelNumeric) /
  nrow(validation_data_subset)
cat("Decision Tree Accuracy #2:", dt_accuracy_2, "\n")
```

```
## Decision Tree Accuracy #2: 1
```

```
cat("Compared to full model Accuracy:", dt_accuracy, "\n")
```

```
## Compared to full model Accuracy: 1
```

3 Results

As shown by the analysis section both a decision tree and linear regression provide the best accuracy. Below is the same table for reference.

```
accuracies_df %>% knitr::kable()
```

Model	Accuracy
Decision Tree	1.0000000
SVM	0.0000000
Linear Regression	1.0000000
Random Forest	0.0263158

Only the decision tree model was used for further analysis and this showed that even when Body Temperature parameter is removed high accuracy on the risk level can be achieved.

4 Conclusion

To summarise, various different health markers of pregnant women were used to train a model that shows the mortality risks categorised as low, medium and high. Four different models were used and their accuracies compared. Both decision tree and linear regression models showed high accuracy. Fewer markers were then used to run further analysis in order to establish whether less data could be collected whilst providing the same level of accuracy. This was indeed the case and provides an exciting premise for future work in evaluating the minimum number of markers needed to accurately score pregnant women in low- and middle-income countries where resources are scarce.

However, there are limitations to our approach and data, data entry error being the highest as shown by some entries showing human heart rates being 7, which we know cannot be the case biologically.

Therefore, further future work could include a way to record data more accurately. Also, the time the glucose levels were measured could be added / standardised as this would affect the readings depending on whether measurements were taken close to a meal or not. Preexisting health conditions could be considered as well such as if the women contract any illnesses during the pregnancy in order to further standardise the dataset.

5 References

1. https://www.who.int/news-room/fact-sheets/detail/maternal-mortality
2. Marzia Ahmed, M. A. Kashem, Mostafijur Rahman, S. Khatun. 2020. Review and Analysis of Risk Factor of Maternal Health in Remote Area using the Internet of Things (IoT). Lecture Notes in Electrical Engineering, vol 632. DOI = 10.1007/978-981-15-2317-5_30