Fetal Health Classification Project

Alexia

2025-01

Fetal Health Classification Project

Introduction

Cardiotocography (CTG) is a frequently used tool in obstetrics to evaluate fetal wellbeing. It consists of a prolonged registration of the fetal heart rate with a posterior evaluation of the trace's characteristics which, together, allow us to classify the exam into 3 different categories - **I** (normal), **II** (indeterminate) and **III** (abnormal). The classification above is used when the exam is performed while the patient is in active labour. In category **III**, the patient should be prepared for delivery, whereas category **II** warrants evaluation of the characteristic of the trace (to determine which characteristics are abnormal), intrauterine resuscitative measures, evaluation for factors which might be causing the alteration and frequent reassessment.

The aim of this project is to develop a machine learning model to predict fetal health given characteristics of the CTG. It will be based upon the "Fetal Health Classification" dataset, which contains 2126 records of exams which were classified by obstetricians into 3 classes - normal, suspect and pathological. For our project, we will assume that all exams were performed during labour.

We will begin by analysing our dataset, understanding its characteristics and preparing it for training. Finally, we will train and test different models so as to determine the most appropriate. Given that our outcome is a categorical variable, we will use measures of accuracy, sensitivity, specificity, precision and F1 score for evaluation. We will also plot ROC curves and compare the area underneath them.

Methods/Analysis

First we will install the required packages and import our desired dataset:

```r
# Identifying if packages are already installed and, if not, installing them

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(pROC)) install.packages("pROC", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(pROC)


# Downloading dataset

url <- "https://www.kaggle.com/api/v1/datasets/download/andrewmvd/fetal-health-classification"
dl <- "fetalhealthclassification.zip"
if(!file.exists(dl))
  download.file(url, dl)
```

```
dl <- unzip(dl)
fetalhealthclassification <- read.csv(dl, header = TRUE)
```

Dataset analysis We will begin by analysing the summary of the dataset:

```
summary(fetalhealthclassification)
```

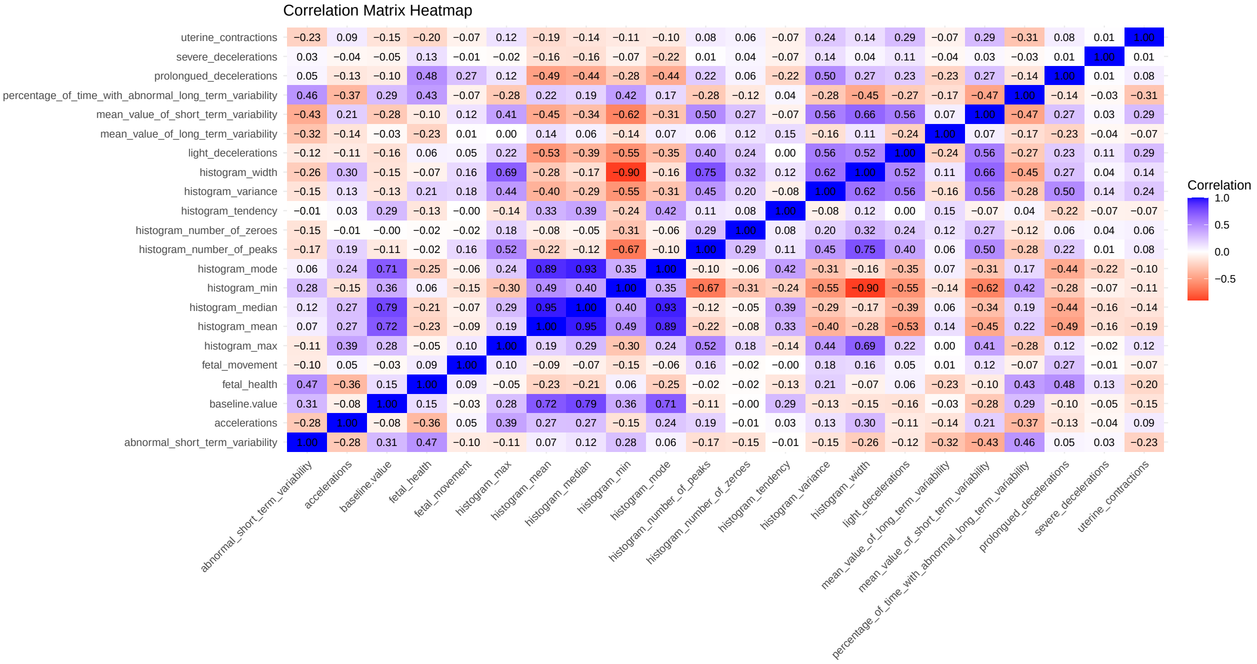
```
##  baseline.value  accelerations      fetal_movement     uterine_contractions
##  Min.   :106.0   Min.   :0.000000   Min.   :0.000000   Min.   :0.000000
##  1st Qu.:126.0   1st Qu.:0.000000   1st Qu.:0.000000   1st Qu.:0.002000
##  Median :133.0   Median :0.002000   Median :0.000000   Median :0.004000
##  Mean   :133.3   Mean   :0.003178   Mean   :0.009481   Mean   :0.004366
##  3rd Qu.:140.0   3rd Qu.:0.006000   3rd Qu.:0.003000   3rd Qu.:0.007000
##  Max.   :160.0   Max.   :0.019000   Max.   :0.481000   Max.   :0.015000
##  light_decelerations severe_decelerations prolongued_decelerations
##  Min.   :0.000000    Min.   :0.000e+00    Min.   :0.0000000
##  1st Qu.:0.000000    1st Qu.:0.000e+00    1st Qu.:0.0000000
##  Median :0.000000    Median :0.000e+00    Median :0.0000000
##  Mean   :0.001889    Mean   :3.293e-06    Mean   :0.0001585
##  3rd Qu.:0.003000    3rd Qu.:0.000e+00    3rd Qu.:0.0000000
##  Max.   :0.015000    Max.   :1.000e-03    Max.   :0.0050000
##  abnormal_short_term_variability mean_value_of_short_term_variability
##  Min.   :12.00                   Min.   :0.200
##  1st Qu.:32.00                   1st Qu.:0.700
##  Median :49.00                   Median :1.200
##  Mean   :46.99                   Mean   :1.333
##  3rd Qu.:61.00                   3rd Qu.:1.700
##  Max.   :87.00                   Max.   :7.000
##  percentage_of_time_with_abnormal_long_term_variability
##  Min.   : 0.000
##  1st Qu.: 0.000
##  Median : 0.000
##  Mean   : 9.847
##  3rd Qu.:11.000
##  Max.   :91.000
##  mean_value_of_long_term_variability histogram_width  histogram_min
##  Min.   : 0.000                      Min.   :  3.00   Min.   : 50.00
##  1st Qu.: 4.600                      1st Qu.: 37.00   1st Qu.: 67.00
##  Median : 7.400                      Median : 67.50   Median : 93.00
##  Mean   : 8.188                      Mean   : 70.45   Mean   : 93.58
##  3rd Qu.:10.800                      3rd Qu.:100.00   3rd Qu.:120.00
##  Max.   :50.700                      Max.   :180.00   Max.   :159.00
##  histogram_max histogram_number_of_peaks histogram_number_of_zeroes
##  Min.   :122   Min.   : 0.000            Min.   : 0.0000
##  1st Qu.:152   1st Qu.: 2.000            1st Qu.: 0.0000
##  Median :162   Median : 3.000            Median : 0.0000
##  Mean   :164   Mean   : 4.068            Mean   : 0.3236
##  3rd Qu.:174   3rd Qu.: 6.000            3rd Qu.: 0.0000
##  Max.   :238   Max.   :18.000            Max.   :10.0000
##  histogram_mode  histogram_mean  histogram_median histogram_variance
##  Min.   : 60.0   Min.   : 73.0   Min.   : 77.0    Min.   :  0.00
##  1st Qu.:129.0   1st Qu.:125.0   1st Qu.:129.0    1st Qu.:  2.00
##  Median :139.0   Median :136.0   Median :139.0    Median :  7.00
##  Mean   :137.5   Mean   :134.6   Mean   :138.1    Mean   : 18.81
##  3rd Qu.:148.0   3rd Qu.:145.0   3rd Qu.:148.0    3rd Qu.: 24.00
##  Max.   :187.0   Max.   :182.0   Max.   :186.0    Max.   :269.00
##  histogram_tendency  fetal_health
##  Min.   :-1.0000    Min.   :1.000
##  1st Qu.: 0.0000    1st Qu.:1.000
##  Median : 0.0000    Median :1.000
##  Mean   : 0.3203    Mean   :1.304
##  3rd Qu.: 1.0000    3rd Qu.:1.000
##  Max.   : 1.0000    Max.   :3.000
```

The dataset consists of 2126 observations of 22 variables, one of which is our desired outcome (fetal_health) and the remaining, possible features for its prediction. Some of the features included are among the characteristics used for the manual classification of the trace, such as baseline value, accelerations, decelerations and short term variability. There is also information on the histograms for the values observed for each individual CTG exam. Information on accelerations, fetal movement, uterine contractions and decelerations is given as a measure per second, possibly to reduce bias due to different test durations.

We can see that there are no missing values in the dataset. One possible limitation identified from the analysis is that there are no values of fetal heart rate baseline over 160. A baseline over 160 is considered fetal tachycardia and can indicate an abnormal trace. While this may not prove relevant for our current analysis, it can be a limitation in the model's use for external datasets.

We will now plot the correlation matrix as a heatmap to identify correlated variables:

```r
# Building the correlation matrix

correlation_matrix <- cor(fetalhealthclassification)

# Converting the matrix into long format for plotting
correlation_df <- as.data.frame(correlation_matrix)
correlation_df$rowname <- rownames(correlation_matrix)
correlation_long <- correlation_df %>%
  pivot_longer(
    cols = -rowname,
    names_to = "variable",
    values_to = "correlation"
  )

# Plotting the heatmap

ggplot(correlation_long, aes(x = rowname, y = variable, fill = correlation)) +
  geom_tile() +
  scale_fill_gradient2(low = "red", mid = "white", high = "blue", midpoint = 0) +
  theme_minimal() +
  geom_text(aes(label = sprintf("%.2f", correlation)), size = 3) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Correlation Matrix Heatmap",
       x = "",
       y = "",
       fill = "Correlation")
```



As expected from previous clinical knowledge, accelerations are correlated with better fetal health, while decelerations (especially prolonged) and short term variability correlated with worse fetal health.

There is a high degree of correlation between baseline value and the histogram median, mean and mode, which also makes sense intuitively (as the baseline value is the mean heart rate over the period excluding accelerations and decelerations). The histogram mode, mean and median are also highly correlated with each other.

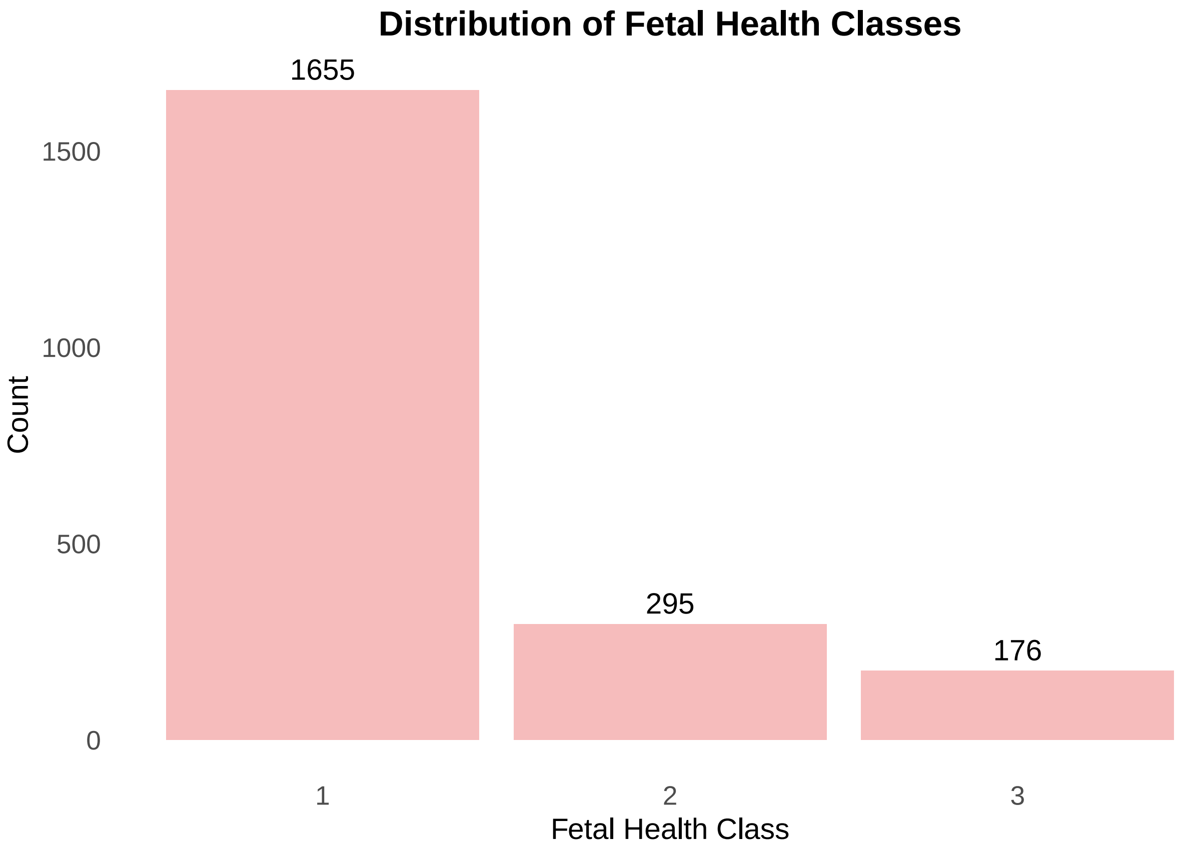
In order to simplify and improve our model, we can remove features that are poorly correlated with the outcome (and therefore usually have little predictive power) and features that are highly correlated with each other (which can make the model less reliable and increase the risk of overfitting).

This can be done with the following code:

```
# Selecting features that will be used for prediction based on the confusion matrix -
# we will remove features poorly correlated with fetal_health (abs(< 0.15)) and
# highly correlated components (> 0.8)

features <- c("baseline.value", "accelerations", "prolongued_decelerations", "uterine_contractions",
"abnormal_short_term_variability", "mean_value_of_long_term_variability",
"percentage_of_time_with_abnormal_long_term_variability", "histogram_variance")
outcome <- "fetal_health"


# Building a new dataset with features that will be utilized

fetalhealthclassification_optimized <- fetalhealthclassification[,c(outcome, features)]
```

We will further analyse the distribution of the outcomes:

```
# Plotting a bar graph to illustrate the frequency of each outcome

fetalhealthclassification_optimized %>%
  group_by(fetal_health) %>%
  ggplot(aes(x = factor(fetal_health))) +
  geom_bar(fill = "#F09090", alpha = 0.6) +
  theme_minimal() +
  labs(
    title = "Distribution of Fetal Health Classes",
    x = "Fetal Health Class",
    y = "Count"
  ) +
  geom_text(
    stat = "count",
    aes(label = after_stat(count)),
    vjust = -0.5,
    color = "black",
    size = 4
  ) +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),
    axis.text = element_text(size = 10),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank()
  )
```



As seen from the graph above, we have an imbalanced dataset, with “1” as the majority class corresponding to 77.8% of the observed values and “2” and “3” as minority classes corresponding to 13.8% and 8.3% of the dataset respectively, indicating a moderate degree of imbalance.

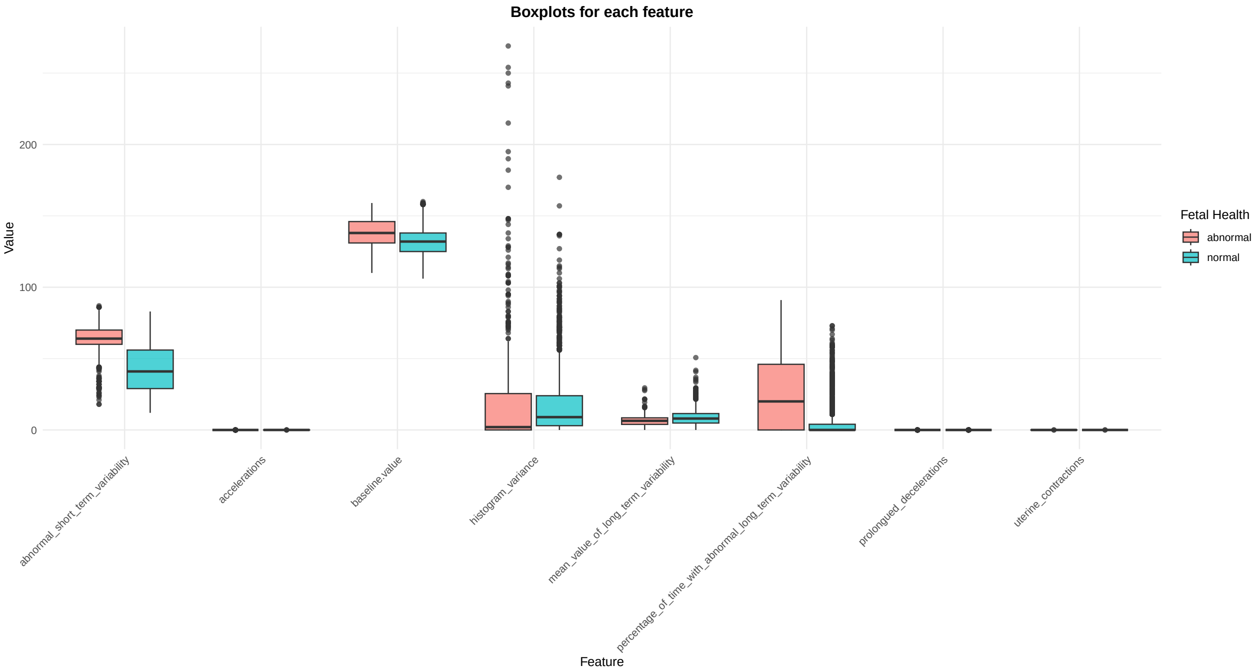
In this particular case, the imbalance can be mitigated by joining fetal health classes 2 and 3 into the same outcome. In the context of this machine learning model, grouping category II and III exams is reasonable as both require intervention by the healthcare professional. As such, we will now consider the exam as either “normal” or “abnormal”.

For our classification algorithm, the fetal_health column must also be a factor. The transformations above can be made using the following code:

```
# Making fetal_health a factor for training the ML models and grouping category 2 and 3 into "abnormal"

fetalhealthclassification_optimized <- fetalhealthclassification_optimized %>%
  mutate(fetal_health = factor(ifelse(fetal_health == 1, "normal", "abnormal")))
```

Finally, we will proceed to an analysis of the remaining features:

```
# Creating a long table to allow us to plot boxplots for each feature in the same plot

fetalhealthclassification_long <- fetalhealthclassification_optimized %>%
pivot_longer(
    cols = seq(2,9), #Columns which correspond to features
    names_to = "variable",
    values_to = "value"
    )


# Plotting the graph

ggplot(fetalhealthclassification_long, aes(x = variable, y = value, fill = fetal_health)) +
    geom_boxplot(alpha = 0.7) +
    labs(
        title = "Boxplots for each feature",
        x = "Feature",
        y = "Value",
        fill = "Fetal Health"
       ) +
    theme_minimal() +
    theme(
        plot.title = element_text(hjust = 0.5, face = "bold"),
        axis.text.x = element_text(angle = 45, hjust = 1)
        )
```

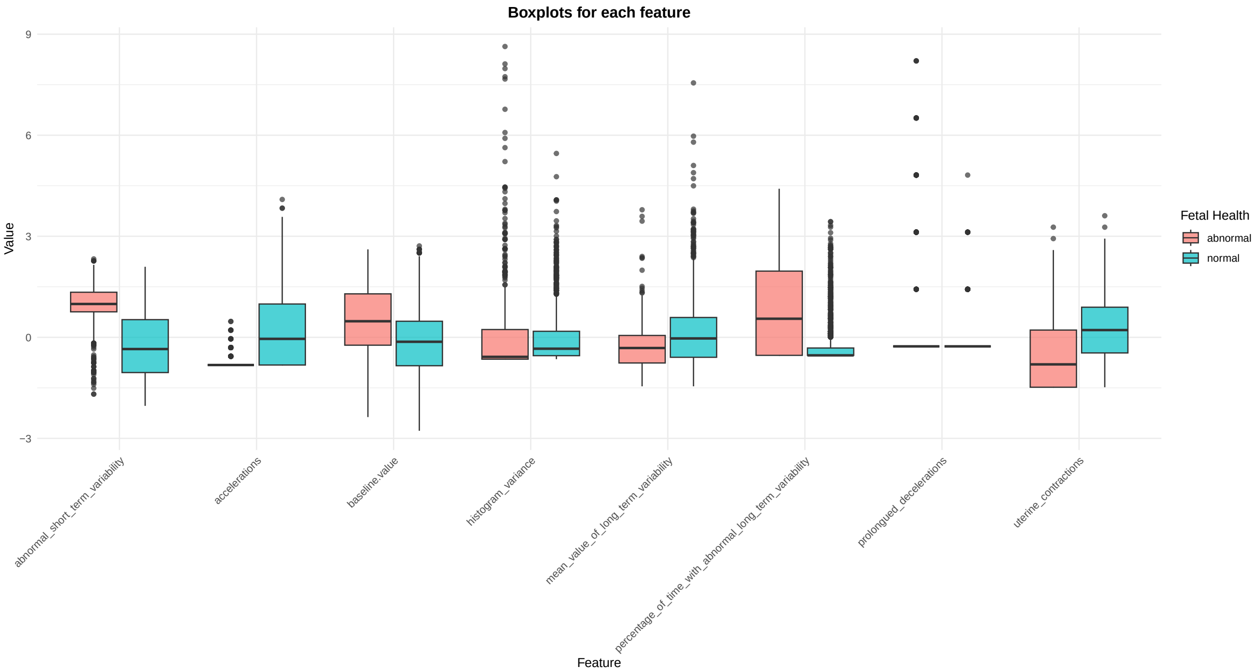


From the boxplots, we can see that the features are measured in different scales. This affects the performance of distance-based algorithms such as K-nearest Neighbours. As such, we will scale the features with the following code:

```
# Scaling the features

fetalhealthclassification_scaled <- fetalhealthclassification_optimized
fetalhealthclassification_scaled[,2:9] <- scale(fetalhealthclassification_scaled[,2:9])
```

The code for the boxplots can be repeated with the scaled dataset to provide an improved graph:

```
# Creating a long table to allow us to plot boxplots for each scaled feature in the same plot

fetalhealthclassification_long <- fetalhealthclassification_scaled %>%
pivot_longer(
    cols = seq(2,9), #Columns which correspond to features
    names_to = "variable",
    values_to = "value"
    )


# Plotting the graph

ggplot(fetalhealthclassification_long, aes(x = variable, y = value, fill = fetal_health)) +
    geom_boxplot(alpha = 0.7) +
    labs(
        title = "Boxplots for each feature",
        x = "Feature",
        y = "Value",
        fill = "Fetal Health"
      ) +
    theme_minimal() +
    theme(
        plot.title = element_text(hjust = 0.5, face = "bold"),
        axis.text.x = element_text(angle = 45, hjust = 1)
        )
```

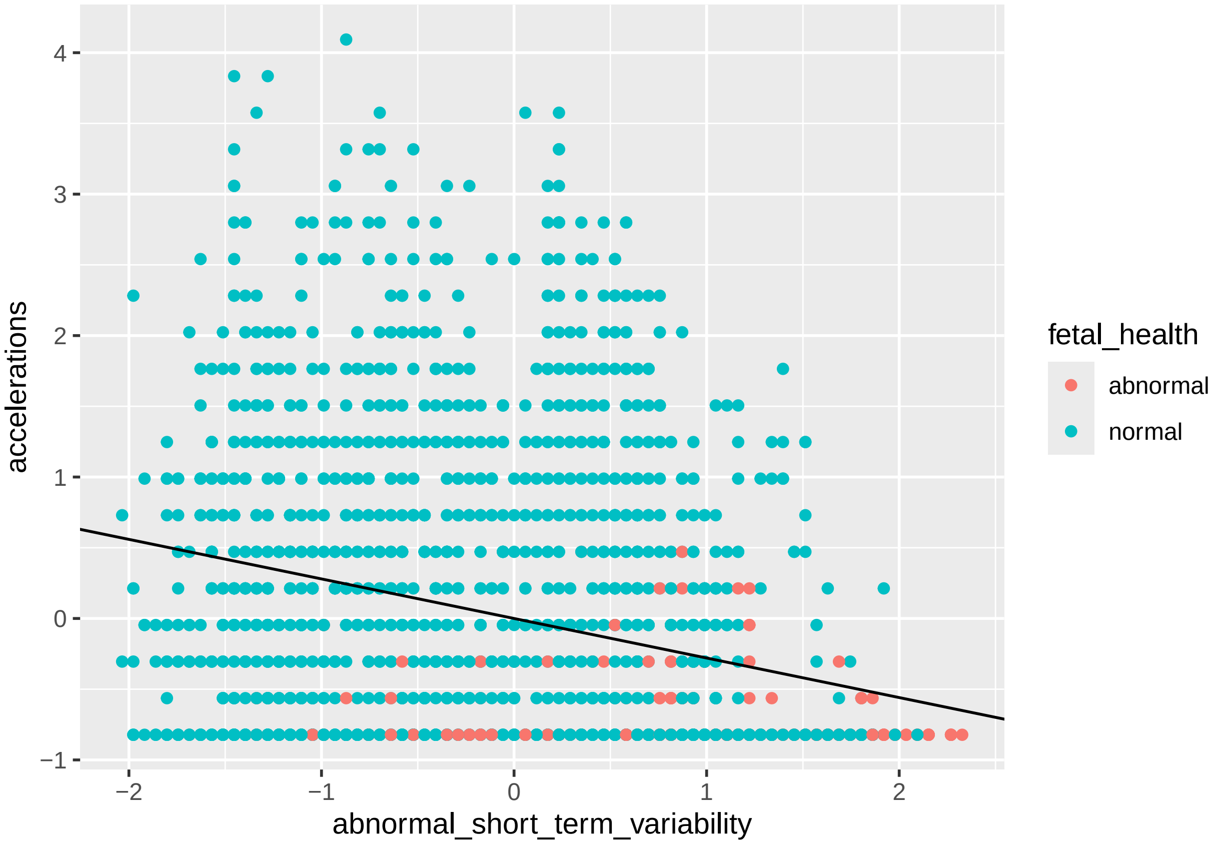


As expected, the boxplots with the largest visual difference between “normal” and “abnormal” fetal health are abnormal_short_term_variability and accelerations. Finally, we can evaluate the relationship between those two variables:

```
# Identifying the intercept and slope for a line representing the correlation between the variables
# abnormal_short_term_variability and accelerations

lm(accelerations ~ abnormal_short_term_variability, data = fetalhealthclassification_scaled)
```

```
##
## Call:
## lm(formula = accelerations ~ abnormal_short_term_variability,
##     data = fetalhealthclassification_scaled)
##
## Coefficients:
##                     (Intercept)  abnormal_short_term_variability
##                       4.461e-15                       -2.796e-01
```

```
# Plotting the graph

fetalhealthclassification_scaled %>%
  ggplot(aes(abnormal_short_term_variability, accelerations)) +
  geom_point(aes(col = fetal_health)) +
  geom_abline(intercept = 4.461e-15, slope = -0.2796, color = "black", linewidth = 0.5)
```



Once again, as expected from both our previous analysis and our clinical knowledge, accelerations tend to be less common in abnormal exams; abnormal short term variability, more common.

Results

We will now begin creating models and comparing their results.

First, we will split the data into training and testing sets, each corresponding respectively to 80% and 20% of our original set.

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

# Creating our training and testing sets

index <- createDataPartition(fetalhealthclassification_scaled$fetal_health, times = 1, p = 0.2, list = FALSE)
train_set <- fetalhealthclassification_scaled[-index,]
test_set <- fetalhealthclassification_scaled[index,]
```

We will also create a variable “control” with our parameters for cross-validation for use in the argument “trControl” in the train function:

```
# Creating our parameters for cross validation
control <- trainControl(method = "cv", number = 10)
```

We will train and test each model and evaluate their results. Finally, we will make a table of all results for all models and discuss the best model and possible improvements.

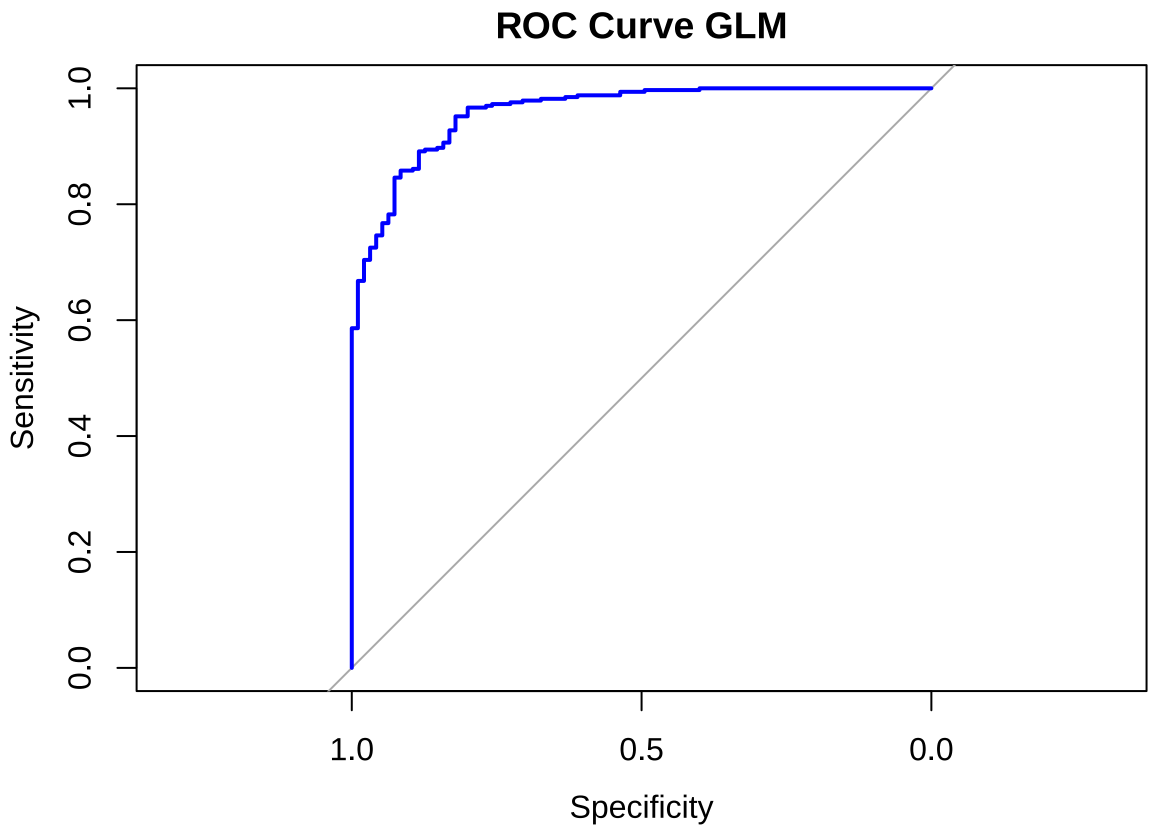
Generalized linear model

The first algorithm we will test is the generalized linear model. The model can be trained and tested with the code below:

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

# Training the model with a glm algorithm

fit_glm <- train(fetal_health ~.,
                 data = train_set,
                 method = "glm",
                 trControl = control)
predictions_glm <- predict(fit_glm, newdata = test_set)

# Obtaining the confusion matrix between the predictions and the reference

conf_matrix_glm <- confusionMatrix(data = predictions_glm, reference = test_set$fetal_health)


# Predicting probabilities for each class

probabilities_glm <- predict(fit_glm, newdata = test_set, type = "prob")[, 2]

# Obtaining the roc curve and the area under the curve

roc_curve_glm <- roc(test_set$fetal_health, probabilities_glm)
auc_glm <- auc(roc_curve_glm)

# Saving metrics to a separate variable for comparison

metrics_glm <- c(
   conf_matrix_glm$byClass["Sensitivity"],
   conf_matrix_glm$byClass["Precision"],
   conf_matrix_glm$byClass["Specificity"],
   conf_matrix_glm$overall["Accuracy"],
   conf_matrix_glm$byClass["F1"],
   auc_glm
)
names(metrics_glm) <- c("sensitivity", "precision", "specificity", "accuracy", "f1_score", "auc")
```

We can now plot our ROC curve and the metrics obtained with the following code:

```
# Plotting ROC curve and printing metrics for evaluation

plot(roc_curve_glm, main = "ROC Curve GLM", col = "blue")
```



```
knitr::kable(metrics_glm, caption = "Evaluation metrics GLM")
```

Table 1: Evaluation metrics GLM

	x
sensitivity	0.8210526
precision	0.8210526
specificity	0.9486405
accuracy	0.9201878
f1_score	0.8210526
auc	0.9595166

The model has adequate accuracy, specificity and area under the curve. However, due to the imbalanced nature of our dataset, it is also important to evaluate precision, sensitivity and the F1 score, which evaluate our minority class. We can strive to improve these values in the next models tested.

Decision tree

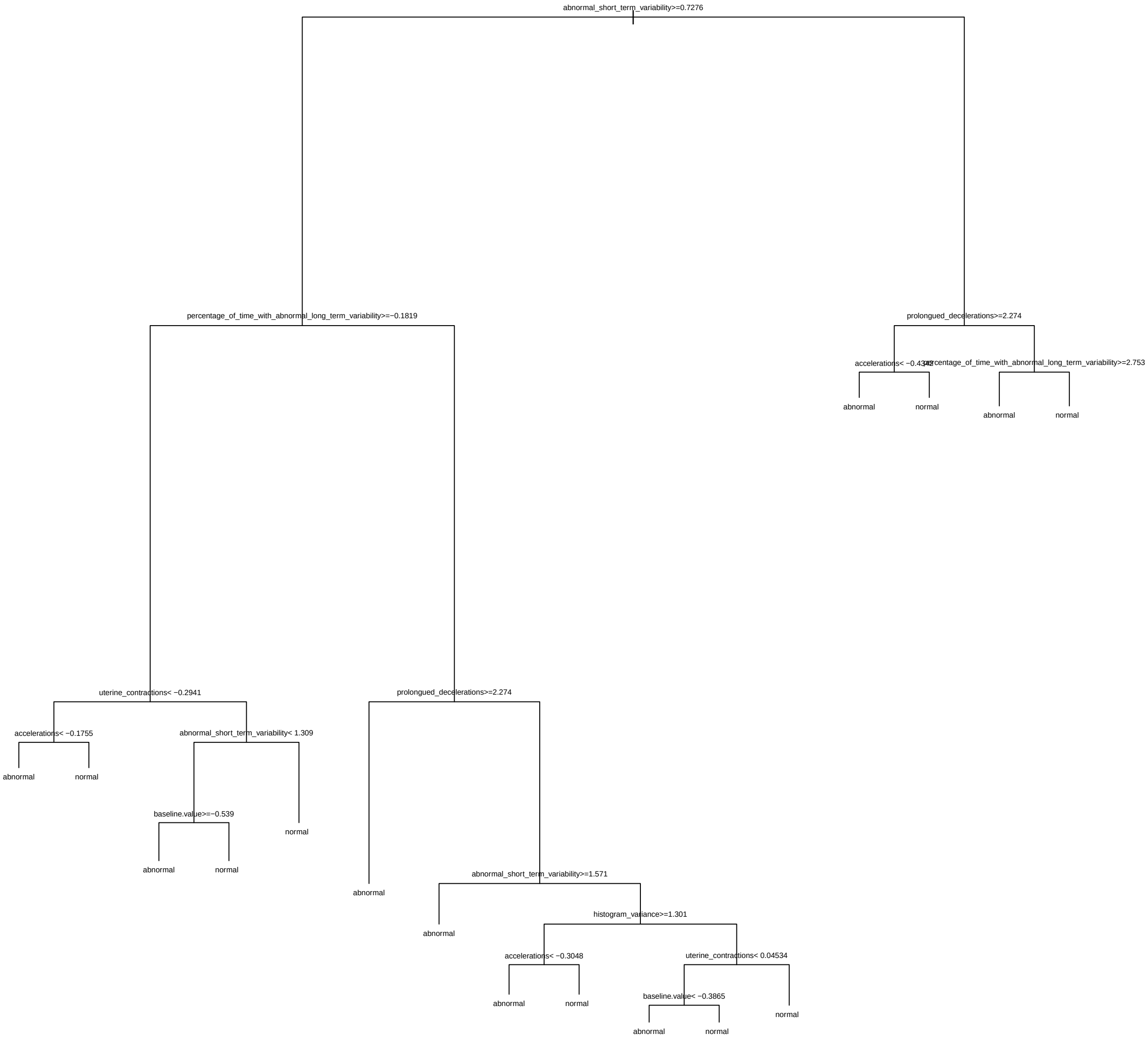
We will now replicate the code above with a few modifications to train a model based on a decision tree algorithm:

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

# Training the model
# The tuneGrid argument defines a data frame of complexity parameters (which control
# the size of the decision tree) for testing
```

```r
fit_dt <- train(fetal_health ~.,
          data = train_set,
          method = "rpart",
          tuneGrid = data.frame(cp = seq(0.0, 0.1, len = 25)),
          trControl = control)
predictions_dt <- predict(fit_dt, newdata = test_set)

# Obtaining the confusion matrix between the predictions and the reference

conf_matrix_dt <- confusionMatrix(data = predictions_dt, reference = test_set$fetal_health)


# Predicting probabilities for each class

probabilities_dt <- predict(fit_dt, newdata = test_set, type = "prob")[, 2]

# Obtaining the ROC curve and AUC

roc_curve_dt <- roc(test_set$fetal_health, probabilities_dt)
auc_dt <- auc(roc_curve_dt)


# Saving metrics to a separate variable

metrics_dt <- c(
  conf_matrix_dt$byClass["Sensitivity"],
  conf_matrix_dt$byClass["Precision"],
  conf_matrix_dt$byClass["Specificity"],
  conf_matrix_dt$overall["Accuracy"],
  conf_matrix_dt$byClass["F1"],
  auc_dt
)
names(metrics_dt) <- c("sensitivity", "precision", "specificity", "accuracy", "f1_score", "auc")
```

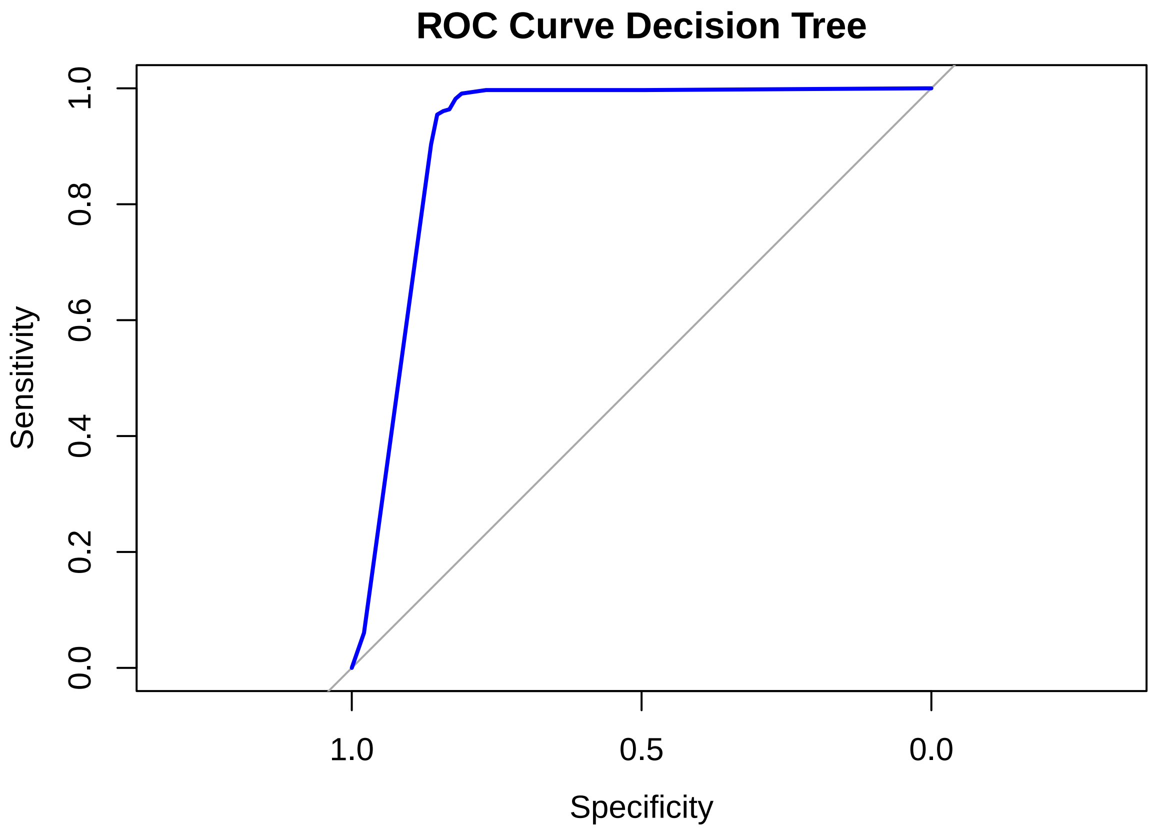
An important advantage of the decision tree algorithm is that is lends itself to good explainability as it allows for a direct evaluation of the decision tree, which in turn allows us to understand the process through which all predictions we made. We can see it with the following code:

```r
# Plotting tree structure

plot(fit_dt$finalModel, margin = 0.05)
text(fit_dt$finalModel, cex = 0.5)
```


abnormal_short_term_variability>=0.7276
percentage_of_time_with_abnormal_long_term_variability>=−0.1819
prolongued_decelerations>=2.274
accelerations< −0.4342
percentage_of_time_with_abnormal_long_term_variability>=2.753
abnormal
normal
abnormal
normal
uterine_contractions< −0.2941
prolongued_decelerations>=2.274
accelerations< −0.1755
abnormal_short_term_variability< 1.309
abnormal
normal
baseline.value>=−0.539
normal
abnormal
normal
abnormal
abnormal_short_term_variability>=1.571
abnormal
histogram_variance>=1.301
accelerations< −0.3048
uterine_contractions< 0.04534
abnormal
normal
baseline.value< −0.3865
normal
abnormal
normal

We can also plot the ROC curve and print the metrics obtained with this algorithm with the following code:

```
# Plotting ROC curve and printing metrics for evaluation
plot(roc_curve_dt, main = "ROC Curve Decision Tree", col = "blue")
```



```
knitr::kable(metrics_dt, caption = "Evaluation metrics DT")
```

Table 2: Evaluation metrics DT

	x
sensitivity	0.8210526
precision	0.9285714
specificity	0.9818731
accuracy	0.9460094
f1_score	0.8715084
auc	0.9157577

When compared to our GLM model, our decision tree-based model has similar sensitivity, but better accuracy, precision, specificity and F1 score. The AUC, however, is smaller for the decision tree model.

K nearest neighbours

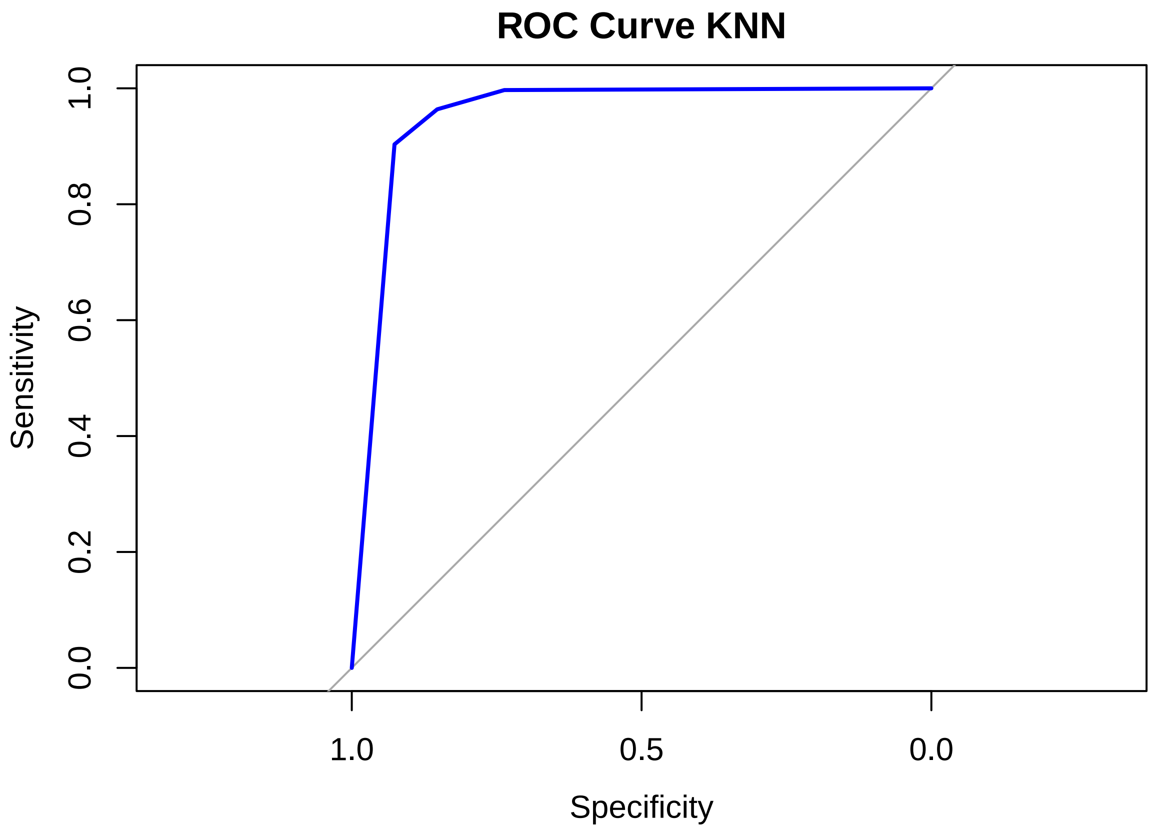
The next algorithm we shall attempt to use is KNN, which performs a prediction based on the outcomes for the K-nearest observations in the neighbourhood - that is, the values surrounding the value we wish to predict, according to vectors based on the features. In our train function, we can test a different number of neighbours and use the best for performing predictions:

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

# Training the model
# The tuneGrid argument defines a data frame of k-neighbours for testing
```

```r
fit_knn <- train(fetal_health ~.,
                 data = train_set,
                 method = "knn",
                 tuneGrid = data.frame(k = seq(1,15,2)),
                 trControl = control)

# Obtaining the confusion matrix between the predictions and the reference

predictions_knn <- predict(fit_knn, newdata = test_set)
conf_matrix_knn <- confusionMatrix(data = predictions_knn, reference = test_set$fetal_health)


# Predicting probabilities for each class

probabilities_knn <- predict(fit_knn, newdata = test_set, type = "prob")[, 2]

# Obtaining the ROC curve and AUC

roc_curve_knn <- roc(test_set$fetal_health, probabilities_knn)
auc_knn <- auc(roc_curve_knn)


# Saving metrics to a separate variable

metrics_knn <- c(
  conf_matrix_knn$byClass["Sensitivity"],
  conf_matrix_knn$byClass["Precision"],
  conf_matrix_knn$byClass["Specificity"],
  conf_matrix_knn$overall["Accuracy"],
  conf_matrix_knn$byClass["F1"],
  auc_knn
)
names(metrics_knn) <- c("sensitivity", "precision", "specificity", "accuracy", "f1_score", "auc")
```

Similarly, we can obtain the ROC curve and the metrics for this algorithm with the code:

```r
# Plotting ROC curve and printing metrics for evaluation

plot(roc_curve_knn, main = "ROC Curve KNN", col = "blue")
```



```
knitr::kable(metrics_knn, caption = "Evaluation metrics KNN")
```

Table 3: Evaluation metrics KNN

	x
sensitivity	0.8526316
precision	0.8709677
specificity	0.9637462
accuracy	0.9389671
f1_score	0.8617021
auc	0.9513118

This model has the highest sensitivity thus far. However, its precision, accuracy, F1 score and specificity are inferior to the decision tree model, and its AUC is inferior to the glm model. This means it correctly identifies the abnormal class better than the previous models, but misidentifies normal exams more often. It could be applied depending on the desired use, but we can still further test new models.

Random Forest

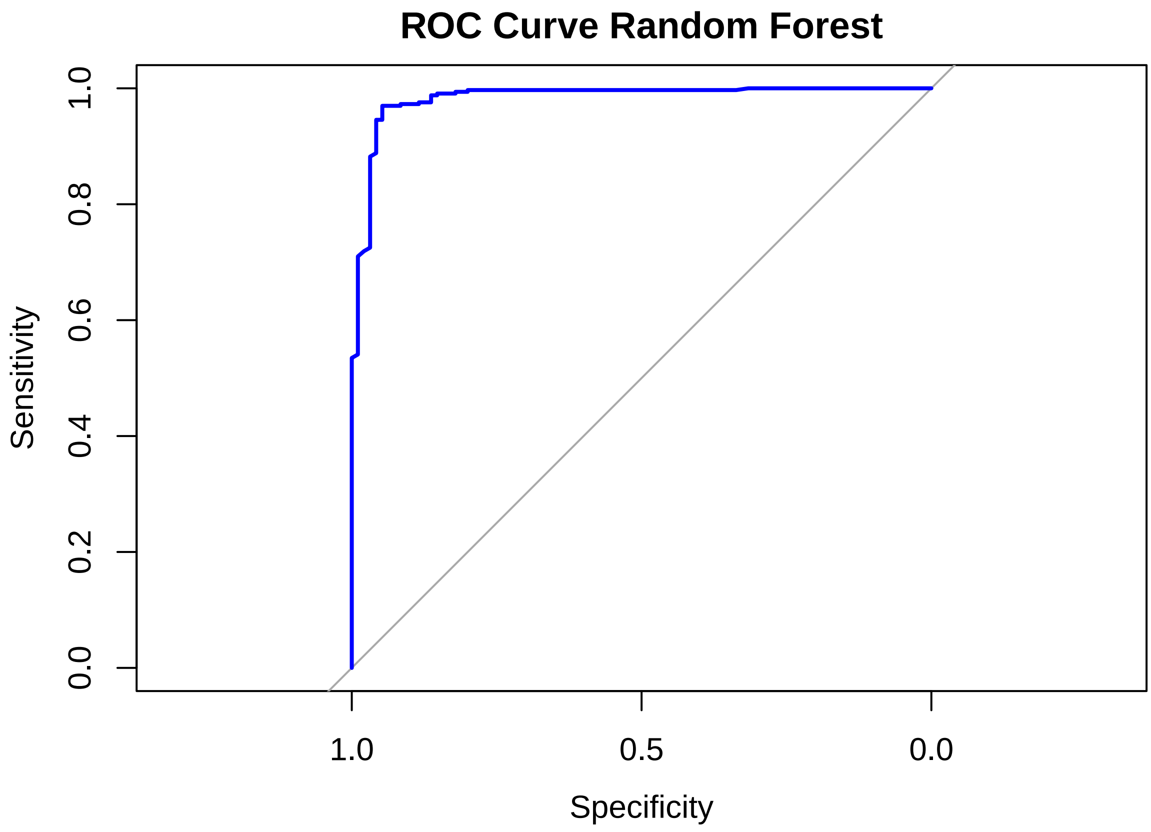
Random forest is an algorithm that builds decision trees and makes predictions by combining their outcomes. We can train our model with the code:

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

# Training the model
# The tuneGrid argument defines a data frame of number of features to consider for
# each split in the tree for testing
```

```
fit_rf <- train(fetal_health ~.,
            data = train_set,
            method = "rf",
            tuneGrid = data.frame(mtry = seq(1, 8)),
            trControl = control)

# Obtaining the confusion matrix between the predictions and the reference

predictions_rf <- predict(fit_rf, newdata = test_set)
conf_matrix_rf <- confusionMatrix(data = predictions_rf, reference = test_set$fetal_health)


# Predicting probabilities for each class

probabilities_rf <- predict(fit_rf, newdata = test_set, type = "prob")[, 2]

# Obtaining the ROC curve and AUC

roc_curve_rf <- roc(test_set$fetal_health, probabilities_rf)
auc_rf <- auc(roc_curve_rf)

# Saving metrics to a separate variable

metrics_rf <- c(
  conf_matrix_rf$byClass["Sensitivity"],
  conf_matrix_rf$byClass["Precision"],
  conf_matrix_rf$byClass["Specificity"],
  conf_matrix_rf$overall["Accuracy"],
  conf_matrix_rf$byClass["F1"],
  auc_rf
)
names(metrics_rf) <- c("sensitivity", "precision", "specificity", "accuracy", "f1_score", "auc")
```

We can now plot the ROC curve and obtain the metrics:

```
# Plotting ROC curve and printing metrics for evaluation

plot(roc_curve_rf, main = "ROC Curve Random Forest", col = "blue")
```



```
knitr::kable(metrics_rf, caption = "Evaluation metrics RF")
```

Table 4: Evaluation metrics RF

	x
sensitivity	0.8631579
precision	0.9534884
specificity	0.9879154
accuracy	0.9600939
f1_score	0.9060773
auc	0.9831293

The Random Forest algorithm appears to be superior in all metrics when compared to the other algorithms.

Comparing the models

```
# Building dataframe with the metrics on each model

model_comparison <- data.frame(
   GLM = metrics_glm,
   DT = metrics_dt,
   KNN = metrics_knn,
   RF = metrics_rf
)

# Plotting the table
```

```
knitr::kable(model_comparison, caption = "Comparison of Model Metrics")
```

Table 5: Comparison of Model Metrics

	GLM	DT	KNN	RF
sensitivity	0.8210526	0.8210526	0.8526316	0.8631579
precision	0.8210526	0.9285714	0.8709677	0.9534884
specificity	0.9486405	0.9818731	0.9637462	0.9879154
accuracy	0.9201878	0.9460094	0.9389671	0.9600939
f1_score	0.8210526	0.8715084	0.8617021	0.9060773
auc	0.9595166	0.9157577	0.9513118	0.9831293

A final evaluation of the models allows us to identify that the Random Forest model performed better according to all of our evaluation metrics. It has the largest AUC, precision and sensitivity and is the only model that achieved an F1 score above 0.9, measures which evaluate the prediction capacity for the minority class, which is important in an imbalanced dataset.

Possible modifications to the model will be based upon its possible clinical use. In our context, the purpose of a machine learning model to classify CTGs would be to possibly "bypass" an evaluation of the exam by a healthcare professional when it is classified as normal (or prioritize evaluation when abnormal), which could be beneficial to the patient (faster result), the healthcare system (reduced cost) and the healthcare professional (reduced workload by evaluating directly exams flagged as abnormal). Taking this practical use into consideration, the risk of missing an abnormal exam is higher than flagging a normal exam for evaluation, as it could lead to a negative neonatal outcome. As such, optimizing sensitivity is paramount.

Adjusting the decision threshold

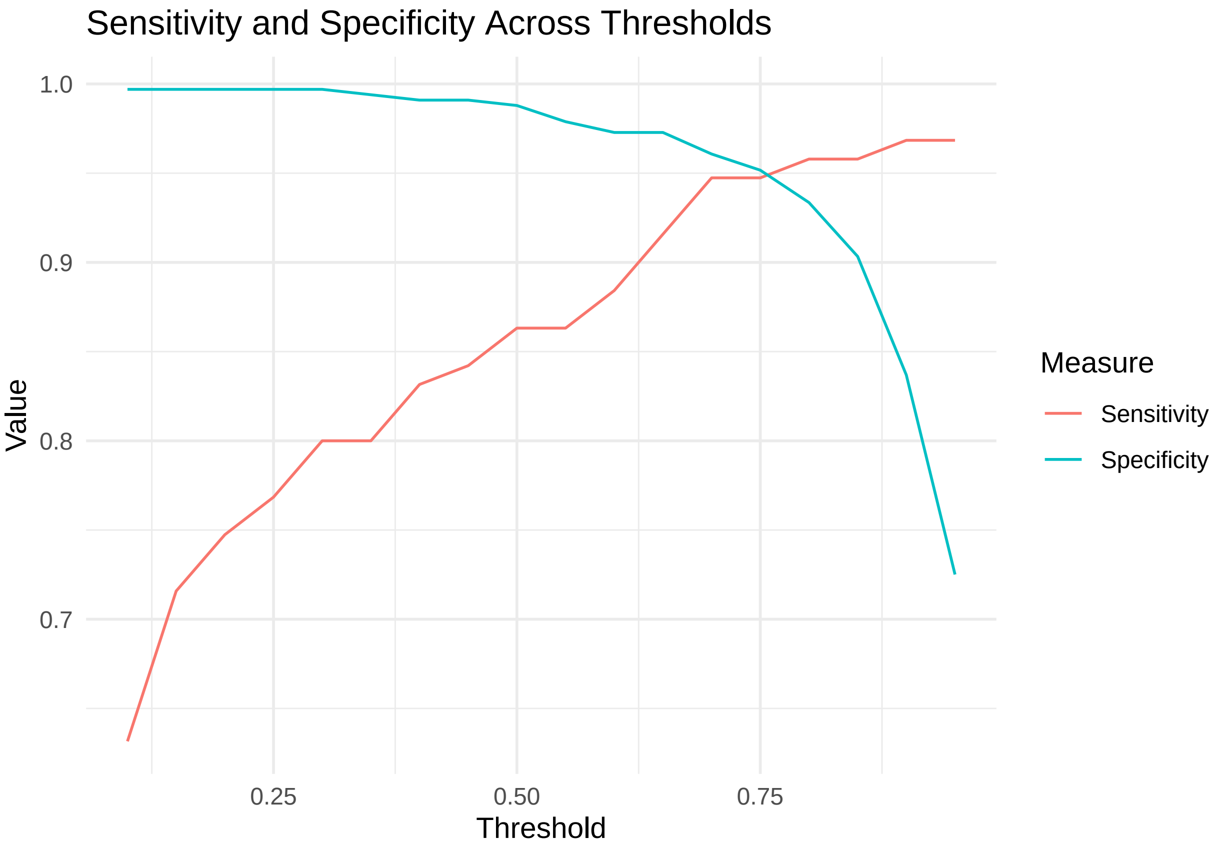
We can increase sensitivity by adjusting the decision threshold, albeit at the cost of decreasing specificity. This strategy might not be applicable to all use cases, but is interesting for us as, as stated before, the risk of missing an abnormal exam exceeds the risk of mislabeling a normal one.

We already have the probability values for each prediction for the test set with our Random Forest model, which we used in the previous section to plot the ROC curve. With this, we can build a function to perform predictions based on different thresholds and give us the evaluation metrics for each (on which we will now focus on sensitivity and specificity):

```
# Creating a variable containing different thresholds for testing

thresholds <- seq(0.1, 0.95, 0.05)

# Creating a function to predict outcomes given the probability values and a threshold,
# generate a confusion matrix and extract our evaluation metrics

ideal_threshold <- function(threshold){
  predicted <- as.factor(ifelse(probabilities_rf > as.numeric(threshold), "normal", "abnormal"))
  conf <- confusionMatrix(data = predicted, reference = test_set$fetal_health)
  metrics <- c(
    conf$byClass["Sensitivity"],
    conf$byClass["Precision"],
    conf$byClass["Specificity"],
    conf$overall["Accuracy"],
    conf$byClass["F1"]
  )
  names(metrics) <- c("sensitivity", "precision", "specificity", "accuracy", "f1_score")
  return(metrics)
}

# Creating a table showing the results for each threshold

threshold_results <- lapply(thresholds, ideal_threshold)
threshold_results <- as.data.frame(do.call(rbind, threshold_results)) %>%
  mutate(threshold = thresholds)
```

We can now plot a graph showing sensitivity and specificity across the different thresholds:

```
# Plotting a graph to illustrate sensitivity and specificity across different thresholds

threshold_results %>%
    ggplot() +
    geom_line(aes(x = threshold, y = sensitivity, color = "Sensitivity")) +
    geom_line(aes(x = threshold, y = specificity, color = "Specificity")) +
    labs(
        title = "Sensitivity and Specificity Across Thresholds",
        y = "Value",
        x = "Threshold",
        color = "Measure"
    ) +
    theme_minimal()
```



We can see that sensitivity increases and specificity decreases as we increase our threshold. Sensitivity increases significantly between 0.5 (the standard threshold used) and 0.7, and incrementally from that point forward. Specificity, on the other hand, decreases exponentially from 0.75. From a visual analysis of the graph, we see that around 0.75 both sensitivity and specificity are around 0.95 - we can choose a value around that number to select our threshold in a way that greatly improves our sensitivity from the original model without compromising our specificity to a large degree.

```
# Identifying thresholds for which sensitivity is greater than 0.94

threshold_results %>% filter(sensitivity > 0.94)
```

```
##   sensitivity precision specificity  accuracy  f1_score threshold
## 1   0.9473684 0.8737864   0.9607251 0.9577465 0.9090909      0.70
## 2   0.9473684 0.8490566   0.9516616 0.9507042 0.8955224      0.75
## 3   0.9578947 0.8053097   0.9335347 0.9389671 0.8750000      0.80
## 4   0.9578947 0.7398374   0.9033233 0.9154930 0.8348624      0.85
## 5   0.9684211 0.6301370   0.8368580 0.8661972 0.7634855      0.90
## 6   0.9684211 0.5027322   0.7250755 0.7793427 0.6618705      0.95
```

We can see from above that sensitivity becomes greater than 0.95 between a threshold of 0.75 and 0.8. We can apply our ideal_threshold function to find the evaluation metrics between these values to further improve our model.

```
# Testing values between 0.75 and 0.8 to identify ideal threshold

thresholds_refined <- seq(0.75, 0.8, 0.01)
threshold_results_refined <- lapply(thresholds_refined, ideal_threshold)
threshold_results_refined <- as.data.frame(do.call(rbind, threshold_results_refined)) %>%
  mutate(threshold = thresholds_refined)

knitr::kable(threshold_results_refined, caption = "Comparison of Thresholds")
```

Table 6: Comparison of Thresholds

sensitivity	precision	specificity	accuracy	f1_score	threshold
0.9473684	0.8490566	0.9516616	0.9507042	0.8955224	0.75
0.9473684	0.8490566	0.9516616	0.9507042	0.8955224	0.76
0.9473684	0.8490566	0.9516616	0.9507042	0.8955224	0.77
0.9578947	0.8348624	0.9456193	0.9483568	0.8921569	0.78
0.9578947	0.8198198	0.9395770	0.9436620	0.8834951	0.79
0.9578947	0.8053097	0.9335347	0.9389671	0.8750000	0.80

At a threshold of 0.78 we have a sensitivity of 0.958 and a specificity of 0.946. Considering the practical application of our model, this value appears to be reasonable in that it increases sensitivity importantly while not compromising specificity (which, if too small, could reduce the other benefits of the model such as reduced manual workload).

Our final confusion table and metrics for the chosen threshold are as follows:

```
# Making our final predictions and confusion matrix which a threshold of 0.78

final_predictions <- as.factor(ifelse(probabilities_rf > 0.78, "normal", "abnormal"))
conf_matrix_final <- confusionMatrix(data = final_predictions, reference = test_set$fetal_health)

metrics_final <- c(
        conf_matrix_final$byClass["Sensitivity"],
        conf_matrix_final$byClass["Precision"],
        conf_matrix_final$byClass["Specificity"],
        conf_matrix_final$overall["Accuracy"],
        conf_matrix_final$byClass["F1"]
     )
names(metrics_final) <- c("sensitivity", "precision", "specificity", "accuracy", "f1_score")
conf_matrix_final$table
```

```
##           Reference
## Prediction abnormal normal
##   abnormal       91     18
##   normal          4    313
```

```
knitr::kable(metrics_final, caption = "Final metrics for our model")
```

Table 7: Final metrics for our model

	x
sensitivity	0.9578947
precision	0.8348624
specificity	0.9456193
accuracy	0.9483568
f1_score	0.8921569

Conclusion

We have trained and tested multiple machine learning models to determine the best for use in classifying normal and abnormal CTGs, a commonly used exam to evaluate fetal wellbeing in the world of obstetrics. The best results were obtained with a Random Forest algorithm. We then adjusted the threshold value for prediction to improve sensitivity, a relevant metric for our use case. This tool could have impact in the health sector by screening which exams must be seen by a healthcare professional with more priority (as they require evaluation and possible intervention). This could decrease healthcare costs, alleviate the burden on healthcare professionals and lead to benefits for the patients such as faster exam evaluations which could lead to more timely intervention. This report has some limitations. First, there is no specific information on gestational age (which can affect some characteristics of the exam) and no examples of fetal heart rate baseline over 160 (a known possible characteristic of an abnormal exam). This last observation in particular limits the model's application to external datasets. Additionally, cardiotocography in itself is an exam with fair sensitivity but poor specificity (Schiermeier et al). As such, even if the model can predict accurately the classification based on the exam, it could be more clinically applicable to use other outcomes which evaluate fetal wellbeing directly such as fetal scalp pH. These measures, however, are far more difficult to obtain for the training and testing of a machine learning model. Furthermore, there is a degree of subjectivity to the evaluation of CTGs, with a fair degree of intra and interobserver variability, meaning there might be bias among the outcomes used for training and testing. Once more, the use of objective outcomes such as the fetal scalp pH cited above could be beneficial.

References

1. Ayres de Campos et al. (2000) SisPorto 2.0 A Program for Automated Analysis of Cardiotocograms. J Matern Fetal Med 5:311-318
2. https://www.kaggle.com/datasets/andrewmvd/fetal-health-classification/data
3. Schiermeier S, Pildner von Steinburg S, Thieme A, Reinhard J, Daumer M, Scholz M, Hatzmann W, Schneider K. Sensitivity and specificity of intrapartum computerised FIGO criteria for cardiotocography and fetal scalp pH during labour: multicentre, observational study. BJOG 2008;115: 1557–1563.