Choose Your Own project: 'FetalHealth'

Saman Musician

06/02/2021

Introduction

Reduction of child mortality is reflected in several of the United Nations' Sustainable Development Goals and is a key indicator of human progress. The UN expects that by 2030, countries end preventable deaths of newborns and children under 5 years of age, with all countries aiming to reduce under 5 mortality to at least as low as 25 per 1,000 live births. Parallel to notion of child mortality is of course maternal mortality, which accounts for 295000 deaths during and following pregnancy and childbirth (as of 2017). The vast majority of these deaths (94%) occurred in low-resource settings, and most could have been prevented. In light of what was mentioned above, Cardiotocograms (CTGs) are a simple and cost accessible option to assess fetal health, allowing healthcare professionals to take action in order to prevent child and maternal mortality. The equipment itself works by sending ultrasound pulses and reading its response, thus shedding light on fetal heart rate (FHR), fetal movements, uterine contractions and more. In this project, we use a data set, Fetal Health, including 2126 CTG results to perform a Machine Learning algorithm to determine the health condition of the fetal by the information provided from CTGs. The CTG results to features including fetal heart rate (baseline value), accelerations, fetal movement, uterine contraction, light, severe and prolonged decelerations (decelerations are an abrupt decrease in the baseline fetal heart rate of greater than 15 bpm for greater than 15 seconds), abnormal short term variability (Short term variation (STV) examines the variability of the fetal heart rate from beat to beat), mean value of short term variability, percentage of time with abnormal long term variability, mean value of long term variability and histogram information. For each CTG result there is a final classification of "fetal health" column in the data set, indicating normal (1), suspect (2) or pathological (3) situations.

1.1. Goal of the project

In this project, we are implementing some algorithms learned in the Machine Learning course and the tools and instructions learned in Data Science program. The main goal is to comprehend the whole concept of a real project and its challenges. This project's specific goal is to find and tune an algorithm to classify the health condition of CTG cases by the features provided with highest accuracy.

1.2. Key Steps

First of all, the websites suggested in project overview was explored and a data set which seemed to be clean and useful for this project purpose was chosen. This data set is included in the Github repository for this project and is being read by the code. The original link to the data set is as follows:

https://www.kaggle.com/andrewmvd/fetal-health-classification/

The next step is the preprocessing step and data wrangling. We should transform the predictors if necessary and delete the columns which will not be used as predictors. Since the data is clean, the features columns which is going to be used will be selected. In this case all columns are being used as predictors. Then, we split the data into a train set which is used to optimize algorithms and a hold-out validation set for final validation of our algorithm. The analysis step is performed on the train set. K-nearest neighbors, decision tree and random forest algorithms are trained on the train set and tested on the test set in order to achieve

the highest overall accuracy for our categorical classification project. The results are then presented and the performance of the models are discussed. Finally the conclusion is made on this project.

```
data <- read.csv("fetal_health.csv")
```

2. Analysis

In this section, I will explain the process and techniques used in order to reach our final algorithm.

2.1. Data Cleaning, Exploration and Visualization

Fortunately, the data used is cleaned by the provider and is ready to be used as inputs to the desired algorithms. Here is the preview of the data:

```
#summary(data)
colnames(data)
```

```
##  [1] "baseline.value"
##  [2] "accelerations"
##  [3] "fetal_movement"
##  [4] "uterine_contractions"
##  [5] "light_decelerations"
##  [6] "severe_decelerations"
##  [7] "prolongued_decelerations"
##  [8] "abnormal_short_term_variability"
##  [9] "mean_value_of_short_term_variability"
## [10] "percentage_of_time_with_abnormal_long_term_variability"
## [11] "mean_value_of_long_term_variability"
## [12] "histogram_width"
## [13] "histogram_min"
## [14] "histogram_max"
## [15] "histogram_number_of_peaks"
## [16] "histogram_number_of_zeroes"
## [17] "histogram_mode"
## [18] "histogram_mean"
## [19] "histogram_median"
## [20] "histogram_variance"
## [21] "histogram_tendency"
## [22] "fetal_health"
```

```
head(data)
```

```
##   baseline.value accelerations fetal_movement uterine_contractions
## 1            120         0.000              0                 0.000
## 2            132         0.006              0                 0.006
## 3            133         0.003              0                 0.008
## 4            134         0.003              0                 0.008
## 5            132         0.007              0                 0.008
## 6            134         0.001              0                 0.010
##   light_decelerations severe_decelerations prolongued_decelerations
## 1               0.000                    0                     0.000
## 2               0.003                    0                     0.000
## 3               0.003                    0                     0.000
## 4               0.003                    0                     0.000
## 5               0.000                    0                     0.000
## 6               0.009                    0                     0.002
```

```
##   abnormal_short_term_variability mean_value_of_short_term_variability
## 1                              73                                   0.5
## 2                              17                                   2.1
## 3                              16                                   2.1
## 4                              16                                   2.4
## 5                              16                                   2.4
## 6                              26                                   5.9
##   percentage_of_time_with_abnormal_long_term_variability
## 1                                                     43
## 2                                                      0
## 3                                                      0
## 4                                                      0
## 5                                                      0
## 6                                                      0
##   mean_value_of_long_term_variability histogram_width histogram_min
## 1                                 2.4              64            62
## 2                                10.4             130            68
## 3                                13.4             130            68
## 4                                23.0             117            53
## 5                                19.9             117            53
## 6                                 0.0             150            50
##   histogram_max histogram_number_of_peaks histogram_number_of_zeroes
## 1           126                         2                          0
## 2           198                         6                          1
## 3           198                         5                          1
## 4           170                        11                          0
## 5           170                         9                          0
## 6           200                         5                          3
##   histogram_mode histogram_mean histogram_median histogram_variance
## 1            120            137              121                73
## 2            141            136              140                12
## 3            141            135              138                13
## 4            137            134              137                13
## 5            137            136              138                11
## 6             76            107              107               170
##   histogram_tendency fetal_health
## 1                  1            2
## 2                  0            1
## 3                  0            1
## 4                  1            1
## 5                  1            1
## 6                  0            3
```

As it is shown, the predictors have multiple scales of variations. All data features are included for training algorithms to classify the fetal health condition of the CTG cases randomly selected for testing. The fetal health classes are either normal (1), suspect (2) or pathological (3). We modify this outcome to be a factor class in our data set. Here is the probability of each fetal health condition in the data set.

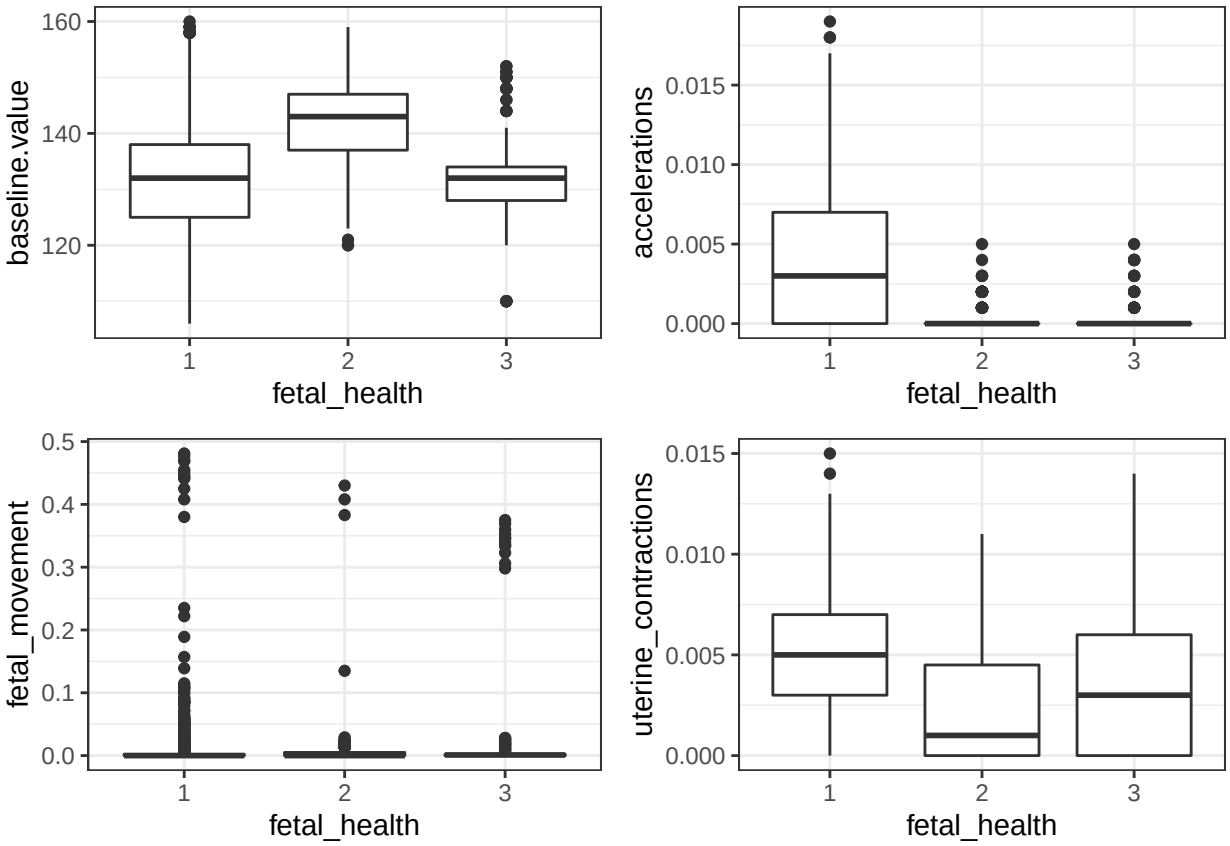
```
## [1] "number of each health condition in CTG results:"
```

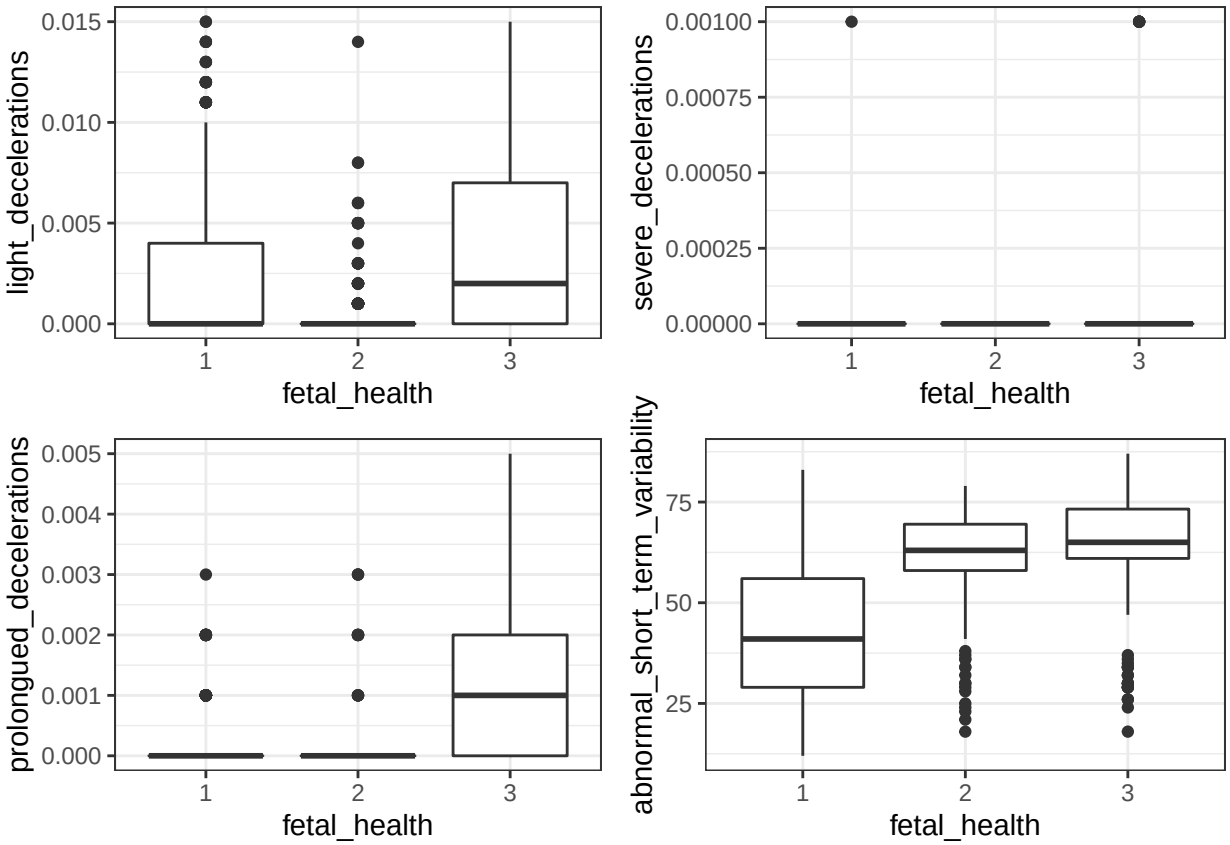
```
##
##    1    2    3
## 1655  295  176
```

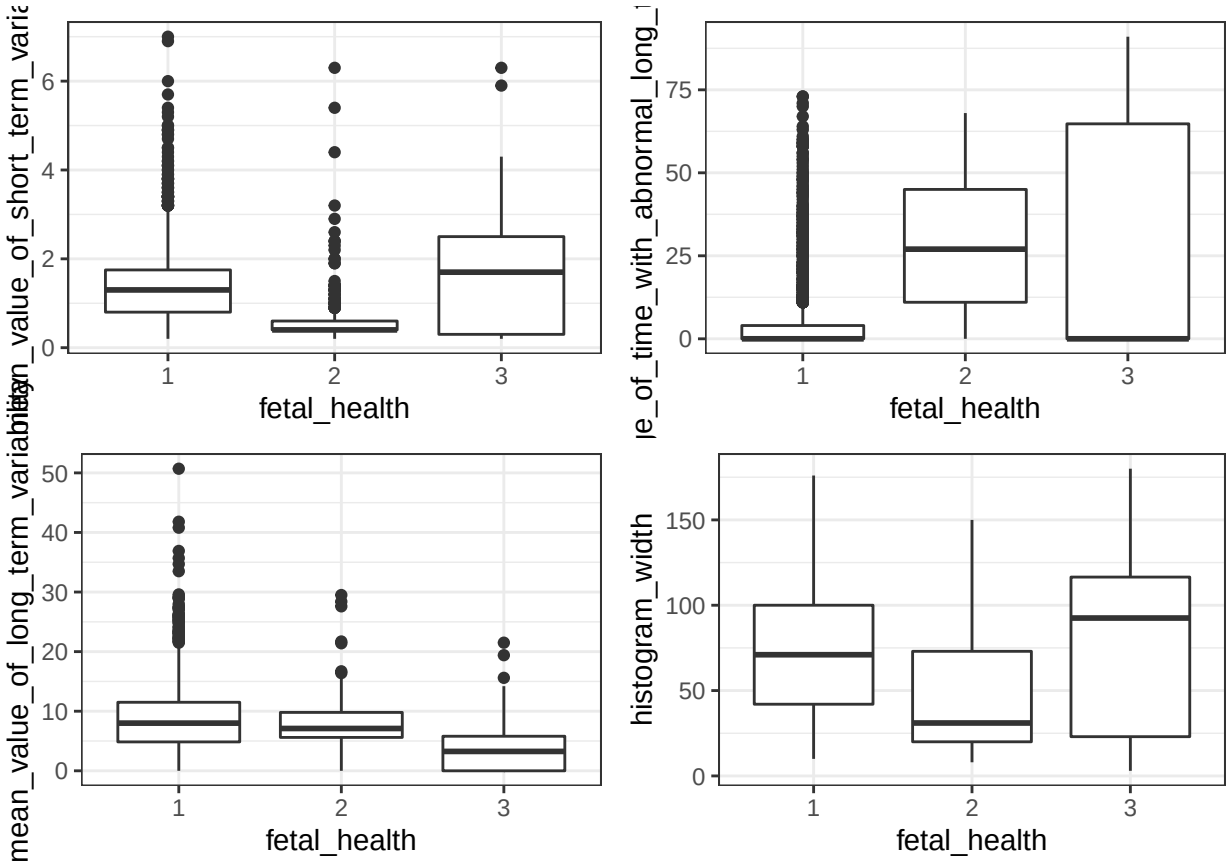
```
## [1] "health conditions Proportions:"
```

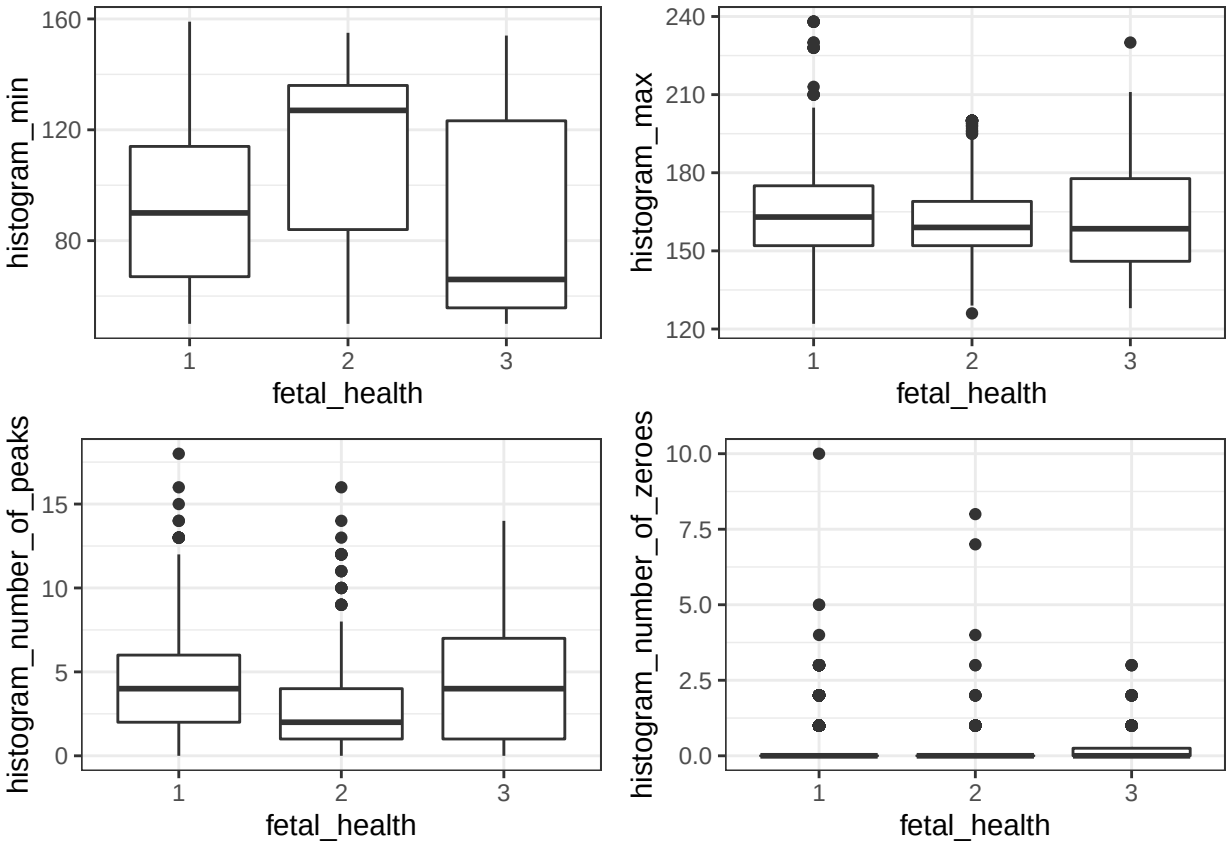
```
##
##           1           2           3
## 0.77845720 0.13875823 0.08278457
```

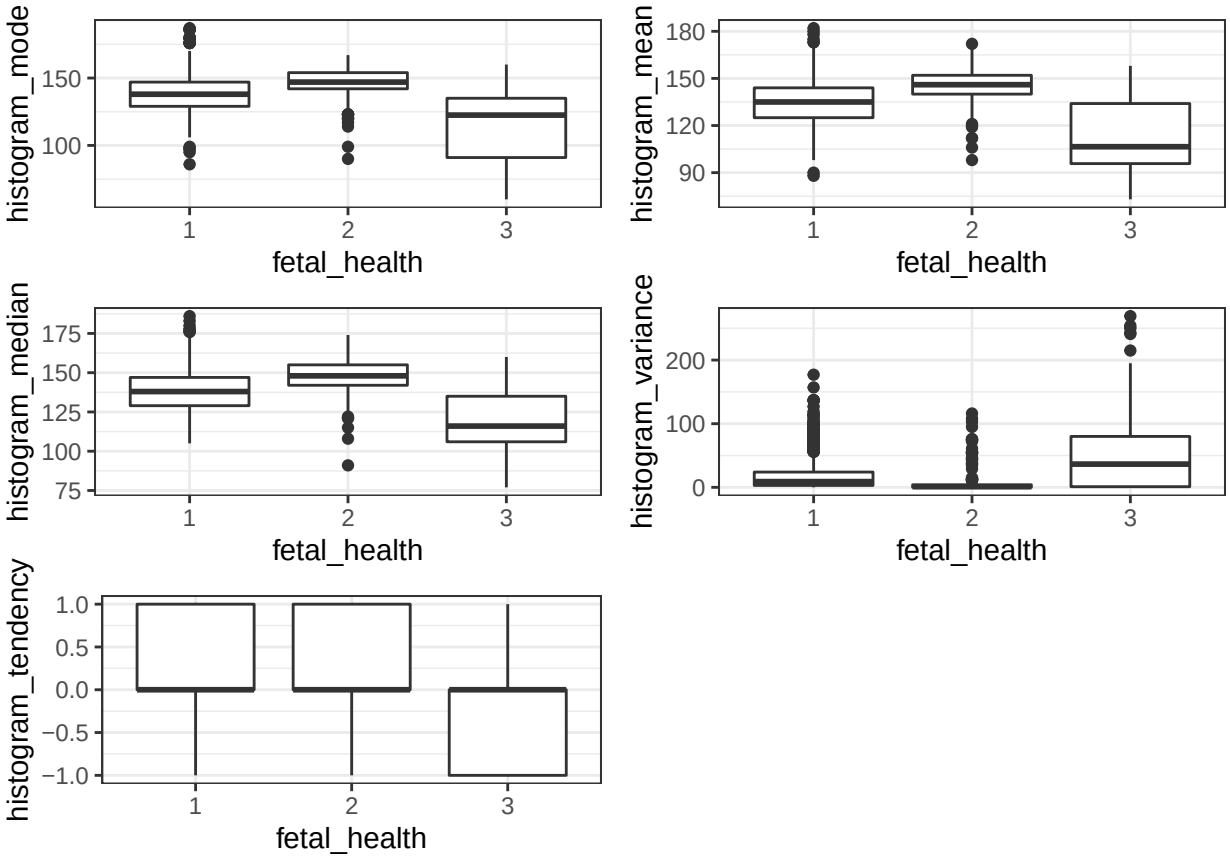
As it is shown 78 percent of the cases are normal, 14 percent are suspect and just about 8 percent are pathological. In order to explore features, we plot each feature for different health conditions.




light_deceleration
0.015
0.010
0.005
0.000
1
2
3
fetal_health
severe_decelerations
0.00100
0.00075
0.00050
0.00025
0.00000
1
2
3
fetal_health
prolongued_decelerations
0.005
0.004
0.003
0.002
0.001
0.000
1
2
3
fetal_health
abnormal_short_term_variability
75
50
25
1
2
3
fetal_health


mean_value_of_short_term_variability
fetal_health
1
2
3
0
2
4
6
mean_value_of_long_term_variability
fetal_health
0
10
20
30
40
50
ge_of_time_with_abnormal_long_t
fetal_health
0
25
50
75
histogram_width
fetal_health
0
50
100
150


histogram_min
160
120
80
1
2
3
fetal_health
histogram_max
240
210
180
150
120
1
2
3
fetal_health
histogram_number_of_peaks
15
10
5
0
1
2
3
fetal_health
histogram_number_of_zeroes
10.0
7.5
5.0
2.5
0.0
1
2
3
fetal_health



Exploring the predictors in this data set, we find out many features have impact on the fetal health but not necessarily in an obvious and distinctive way. For example, accelerations, mean value of long term variability, histogram mode, median and variance looks to have a decisive impact on the fetal health. However, a few features seem to have less impact on the fetal health classification. Although it seems better to remove "histogram number of zeroes" and "severe decelerations" features, I chose to keep all features available, and later evaluate their importance on the accuracy of our algorithms.

To start creating our models, we split a hold out validation set including 10% of the data. We use 90% of the data (called "fetalhealth" set) to train the algorithm and the "validation" set for our final test of the best trained algorithm.

```
set.seed(1, sample.kind = "Rounding" )
test_index <- createDataPartition(y = data$fetal_health, times = 1, p = 0.1, list = FALSE)
fetalhealth <- data[-test_index,]
validation <- data[test_index,]
rm(test_index)
dim(fetalhealth)
```

```
## [1] 1912   22
```

```
dim(validation)
```

```
## [1] 214  22
```

In order to train using only the "fetalhealth" set, we split this data set in to a train set and a test set. The proportion of this train and test set is again 10% and 90%.

```
set.seed(1, sample.kind = "Rounding" )
index <- createDataPartition(fetalhealth$fetal_health, 1, p = 0.1, list = F)
```

```
train_set <- fetalhealth[-index,]
test_set <- fetalhealth[index,]
rm(index)
dim(train_set)
```

```
## [1] 1720   22
```

```
dim(test_set)
```

```
## [1] 192  22
```

3. Modeling Approaches

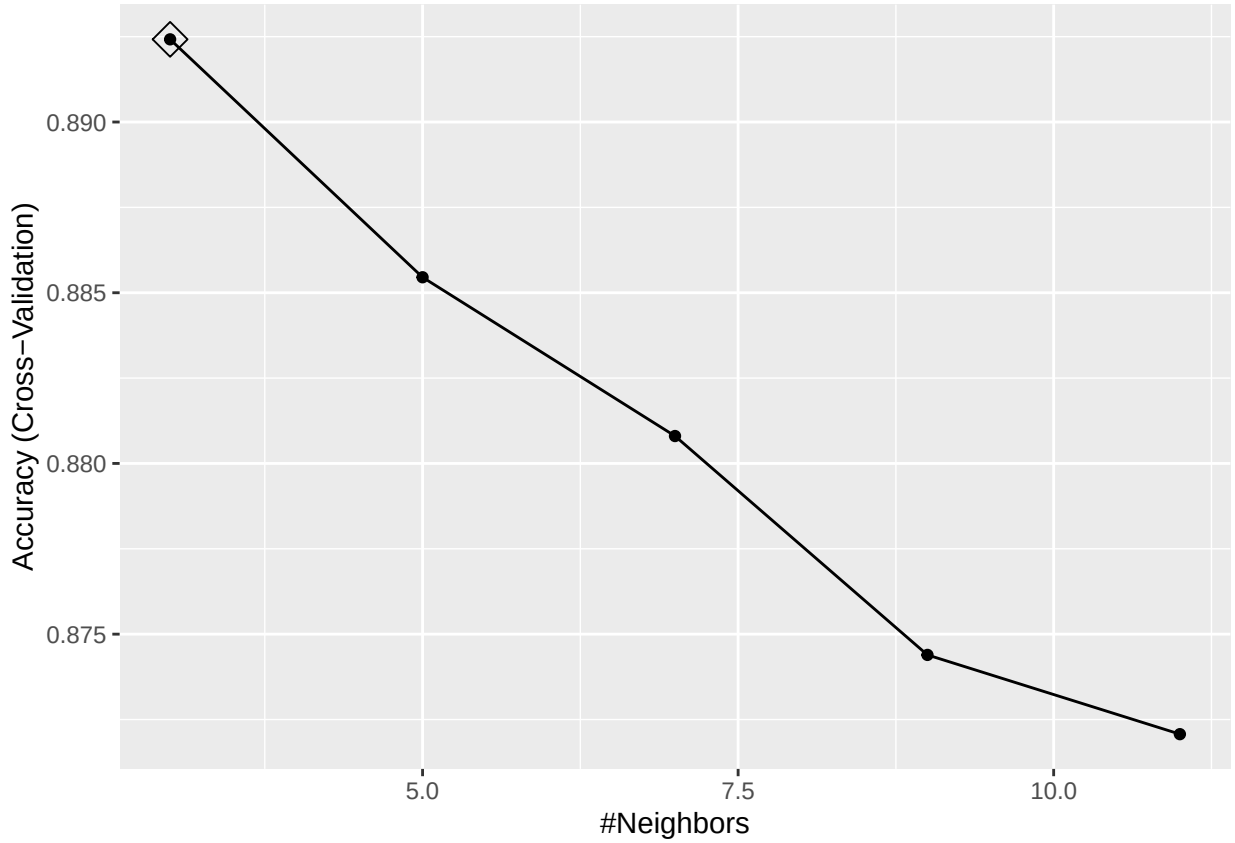
We try k-nearest neighbors, decision trees and random forest algorithms to find the best tune, and finally evaluate our optimized algorithm with the hold-out evaluation set.

3.1. K-Nearest Neighbors

For training a Knn algorithm, lots of calculations need to be done because it is needed to calculate the distance between many observations in train and test sets. Therefore, we use k-fold cross validation to improve the speed. Then we train a knn algorithm and tune the k in k-nearest neighbors. We divide the train set into a set of only predictors (x) and a set of the main label of each CTG case which is the fetal health class only (y). Then we use predict function to predict the fetal health class of the test set and calculate the accuracy of our model.

```
x <- train_set[,-22]    # 22 is the "fetal_health" column number
y <- train_set[,22]

set.seed(1, sample.kind = "Rounding")
control <- trainControl(method = "cv", number = 3, p = .9)
fit_knn <- train(x,y,method = "knn",
                 tuneGrid = data.frame(k=seq(3,11,2)),
                                       trControl = control)
ggplot(fit_knn, highlight = TRUE)
```




```
fit_knn$bestTune
```

```
##   k
## 1 3
```

```
y_hat_knn <- predict(fit_knn, test_set)

confusionMatrix(predict(fit_knn,test_set), test_set$fetal_health)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3
##          1 144   8   0
##          2   2  19   1
##          3   3   0  15
##
## Overall Statistics
##
##                Accuracy : 0.9271
##                  95% CI : (0.8807, 0.9596)
##     No Information Rate : 0.776
##     P-Value [Acc > NIR] : 1.778e-08
##
##                   Kappa : 0.7984
##
##  Mcnemar's Test P-Value : 0.05504
```

```
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.9664  0.70370  0.93750
## Specificity            0.8140  0.98182  0.98295
## Pos Pred Value         0.9474  0.86364  0.83333
## Neg Pred Value         0.8750  0.95294  0.99425
## Prevalence             0.7760  0.14062  0.08333
## Detection Rate         0.7500  0.09896  0.07812
## Detection Prevalence   0.7917  0.11458  0.09375
## Balanced Accuracy      0.8902  0.84276  0.96023
```

```
confusionMatrix(predict(fit_knn,test_set), test_set$fetal_health)$overal["Accuracy"]
```

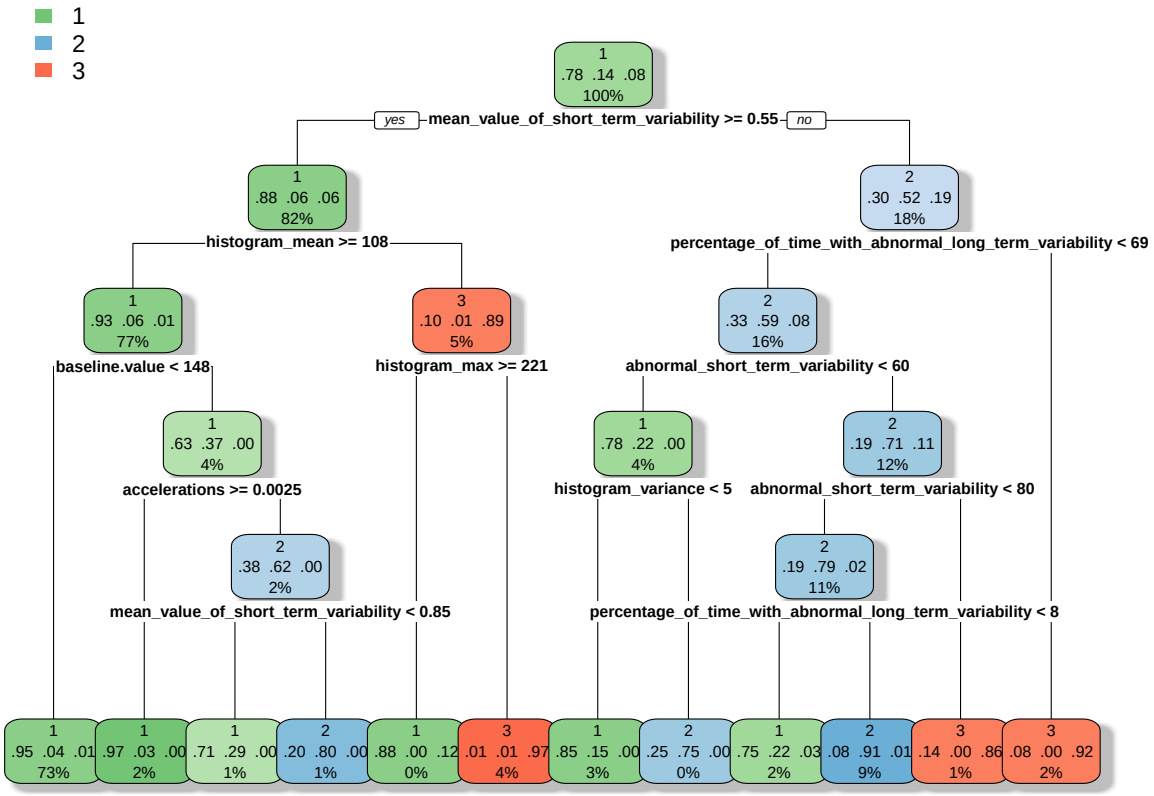
```
##  Accuracy
## 0.9270833
```

As shown, after tuning this model classifies the test set classes with 92.7% accuracy. This accuracy can still get improved

3.2. Decision Tree

We train a decision tree on the train set for classifying the fetal health of each case in the test set. Here I used "rpart.plot" package and function to plot the tree in a clear way.

```
fit_tree <- rpart(fetal_health ~ ., data = train_set)
rpart.plot(fit_tree, tweak = 1.2,
           digits = 2, box.palette = list("Greens","Blues","Reds"),
           shadow.col="gray", nn=F)
```



The 3 different fetal health classes are shown by different colors. By using the predict function, we predict fetal health classes and then compute the accuracy of this decision tree algorithm. We also check the variables importance to verify our first assumptions of some of the features to be less influential.

```
#summary(fit_tree)
varImp(fit_tree)
```

```
##                                                        Overall
## abnormal_short_term_variability                      233.845692
## accelerations                                        114.174287
## baseline.value                                        29.210593
## fetal_movement                                        17.542032
## histogram_max                                         15.178798
## histogram_mean                                       224.685197
## histogram_median                                      87.572723
## histogram_min                                         10.845192
## histogram_mode                                        96.257085
## histogram_number_of_peaks                             11.893271
## histogram_variance                                     5.105051
## histogram_width                                        8.691893
## mean_value_of_long_term_variability                   85.117596
## mean_value_of_short_term_variability                 176.558463
## percentage_of_time_with_abnormal_long_term_variability 227.599669
## prolongued_decelerations                             111.864969
## uterine_contractions                                  46.039693
## light_decelerations                                    0.000000
## severe_decelerations                                   0.000000
```

```
## histogram_number_of_zeroes                                    0.000000
## histogram_tendency                                            0.000000
```

```
y_hat_t <- predict(fit_tree, test_set)
```

```
y_hat_tree <- 0
for (i in 1:length(test_set$fetal_health)){
  y_hat_tree[i] <- which.max(y_hat_t[i,])
}
y_hat_tree <- factor(y_hat_tree)
confusionMatrix(y_hat_tree,test_set$fetal_health)$overal["Accuracy"]
```

```
## Accuracy
##   0.9375
```

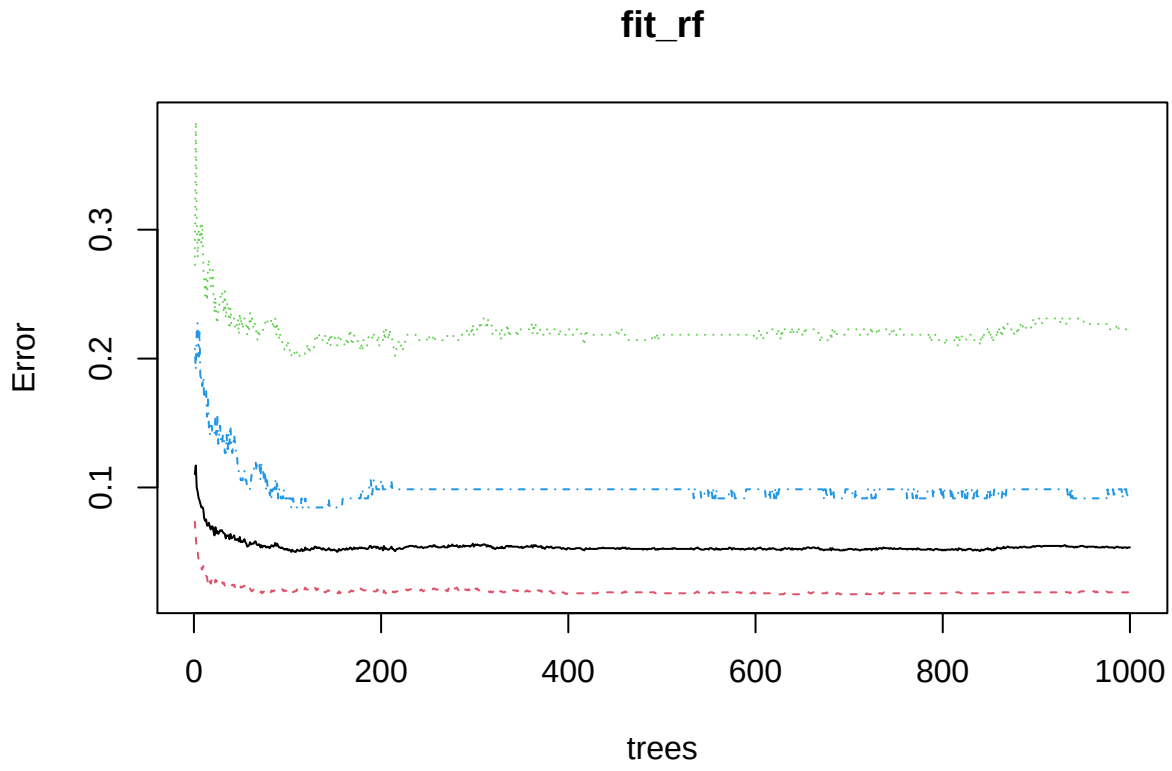
It is clearly shown that 4 predictors have had no importance in training the decision tree model. They could have easily removed during data explorations. This algorithm is highly interpretable and easy to visualize. The accuracy is also higher than knn algorithm. However, we should note that decision trees are not flexible. It is usually better to improve the prediction performance and reduce instability by averaging multiple decision trees.

3.3. Random Forest

After using the decision tree model, it is probable that we can achieve better classification results by using a random forest model. Similar to the Knn model training, we divide the train set into a set of only predictors (x) and only outcomes (y). In order to increase the accuracy we increased the ntree parameter in the randomForest function to 1000.

```
x <- train_set[,-22]    # 22 is the "fetal_health" column number
y <- train_set[,22]

fit_rf <- randomForest(x, y, ntree = 1000)
plot(fit_rf)
```




```
imp <- importance(fit_rf)
imp
```

```
##                                                      MeanDecreaseGini
## baseline.value                                           26.53620384
## accelerations                                            32.52936174
## fetal_movement                                           14.01781289
## uterine_contractions                                     24.30440395
## light_decelerations                                       5.38201223
## severe_decelerations                                      0.08051238
## prolongued_decelerations                                 29.84945116
## abnormal_short_term_variability                          82.36681548
## mean_value_of_short_term_variability                     71.07817037
## percentage_of_time_with_abnormal_long_term_variability   76.30947648
## mean_value_of_long_term_variability                      29.38654697
## histogram_width                                          23.83219646
## histogram_min                                            25.36715779
## histogram_max                                            18.39014966
## histogram_number_of_peaks                                12.17960768
## histogram_number_of_zeroes                                3.13488415
## histogram_mode                                           38.03758584
## histogram_mean                                           58.33409609
## histogram_median                                         31.51859461
## histogram_variance                                       20.99827227
## histogram_tendency                                        4.96127548
```

```r
y_hat_rf <- predict(fit_rf, test_set)
confusionMatrix(y_hat_rf,test_set$fetal_health)$overal["Accuracy"]
```

```
## Accuracy
## 0.953125
```

We have calculated variable importance on the features and realized that by averaging many decision trees the interpretability is not as much as a single decision tree algorithm. However, the accuracy of this model is improved significantly as anticipated. Our final model accuracy is 0.953 on the test set.

4. Results

Now that we optimized our classification model to be a random forest model, we train a similar rainforest algorithm with 1000 trees using the whole "Fetalhealth" data set and test it on the "validation" set which we have held out for validating our final model.

```r
data_x <- fetalhealth[,-22]
data_y <- fetalhealth[,22]

final_rf <- randomForest(data_x, data_y, ntree = 1000)
y_hat_final_rf <- predict(final_rf, validation)
confusionMatrix(y_hat_final_rf,validation$fetal_health)$overal["Accuracy"]
```

```
##  Accuracy
## 0.9392523
```

The final accuracy on the validation set is about 0.94. It is a bit lower than the accuracy we had on the test set. The difference in the total accuracy of this final model and our best tuned model on the train set is maybe due to luck or some overtraining during the procedure.

5. Conclusion

We chose a data set of 21 predictors and an outcome for each entry, being a class of 3 different levels and tried to classify the data by predictors. Some conclusions are made on this project. First, data exploration is a very important part of this data science project. It could be clearly shown that not all the columns of the data are helpful in predictions and classifications. Second, optimizing K-nearest neighbors algorithm, we can overcome the curse of dimensionality by using k-fold cross validation and improve speed. Third, a decision tree algorithm is capable of understandably visualizing the Machine Learning procedure and is very comprehensible. Using random forest algorithm can improve the accuracy of decision trees, although it lowers the interpretability.

Future Work

To better understand the accuracy of trained models on the training set we determine the wrongly classified cases by each of the three algorithms we trained.

```r
which(y_hat_knn != test_set$fetal_health)
```

```
##  [1]  17  19  24  27  32  37  45  60  66  78 122 134 166 190
```

```r
which(y_hat_tree != test_set$fetal_health)
```

```
##  [1]  19  23  24  32  37  60  66 122 127 134 143 190
```

```r
which(y_hat_rf != test_set$fetal_health)
```

```
## [1]  19  23  24  32  60  66 122 127 136
```

By exploring these wrongly classified cases we see some meaningful similarities and uniqueness between algorithms and their predictions. Some improvements can be implemented to aggregate the three algorithms to focus on the differences between them. It is also possible to explore these data set more technically and consult with medical experts to gain more insight on the characteristics of the features of our data set.