Fetal Health Classification

Liz Rightmire

2024-05-13

Introduction

I am interested in how machine learning can be used to predict health outcomes. I chose to focus this project on maternal and fetal health, using the Fetal Health Dataset (https://www.kaggle.com/datasets/andrewmvd/fetal-health-classification).

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

dim(fetalhealth)
```

```
## [1] 2126   22
```

```
colnames(fetalhealth)
```

```
##  [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"
```

This dataset contains 2126 observations from Cardiotocography scans taken during labor. CTG scans are used during pregnancy to monitor fetal heart rate and uterine contractions. Indicators from CTG scans can aid in early detection of fetal distress. There are 21 explanatory variables in this dataset, and the response variable I will predict is “fetal_health” – 1 means normal, 2 means suspect, and 3 means pathological, or of high concern.

I aim to create a model that uses variables in the Cardiotocogram test to predict fetal health score, and determine which factors in the CTG scans are most important in predicting fetal health.

A few variables in the dataset warrant explanation. Baseline value means baseline fetal heart rate – normal range is 110 – 160. Accelerations are short-term rises in heart rate, which are actually a good thing because they indicate adequate oxygen supply. Deceleartions, however, are potential signs of fetal distress. A FHR histogram is a graphical representation of fetal heart rate values over time. By analyzing the shape and characteristics of the histogram, healthcare providers deduce potential signs of distress. Any variable in the dataset that references "histogram" is referring to this FHR histogram.

Exploratory Data Analysis

Let's take a look at class breakdown. We're working with unbalanced data, with the majority class being "normal."

```
fetalhealth |>
  ggplot(aes(x = fetal_health, fill = factor(fetal_health))) +
  geom_bar() +
  xlab("Fetal Health") +
  ylab("Count") +
  ggtitle("Counts of Fetal Health Risk Score") +
  theme_minimal() +
  scale_fill_brewer(palette = "Accent") +
  guides(fill = "none")
```

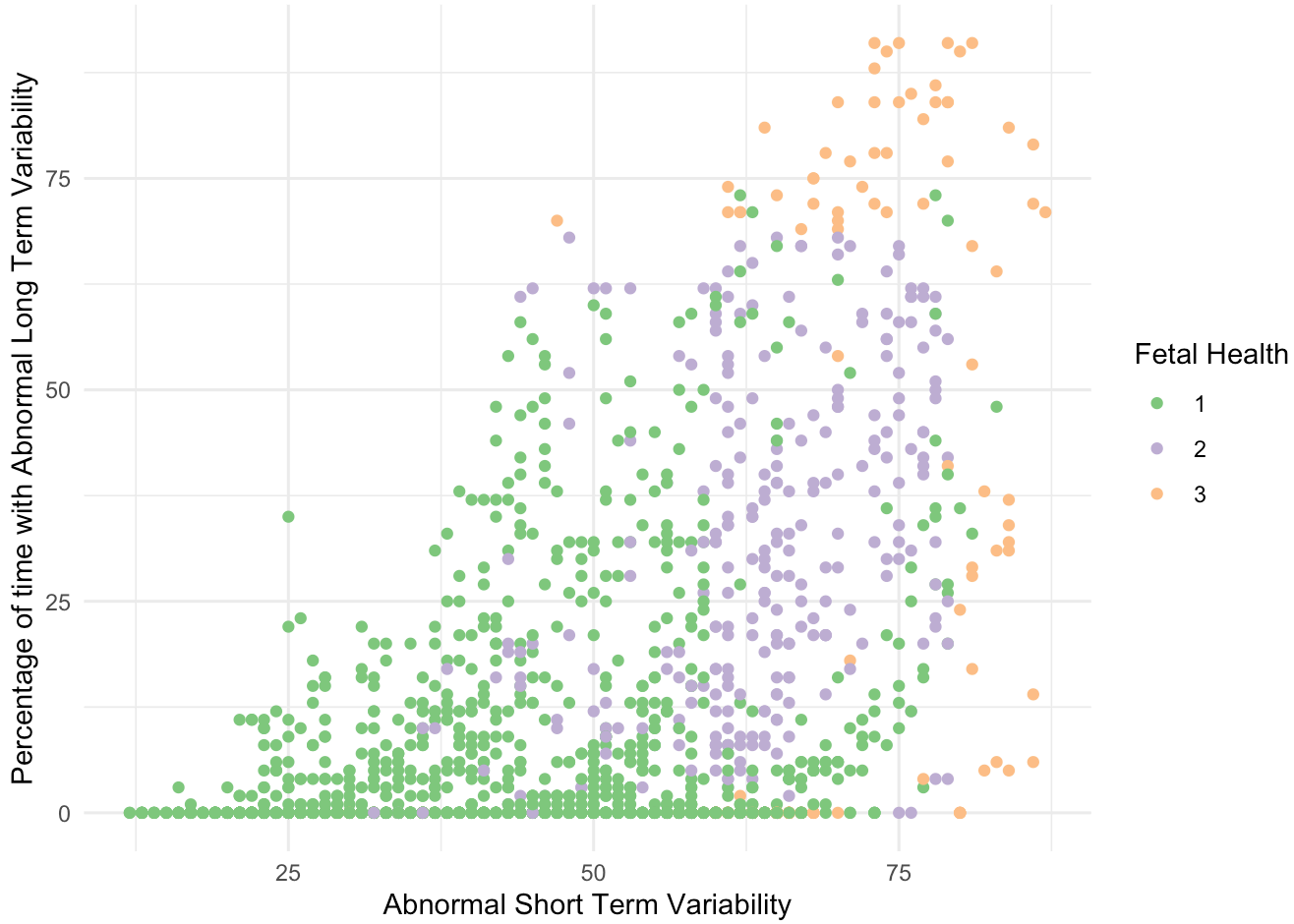
```
# Count occurrences of each class
class_counts <- table(fetalhealth$fetal_health)

# Calculate percentages
class_percentages <- prop.table(class_counts) * 100
print(class_percentages)
```

```
##
##         1         2         3
## 77.845720 13.875823  8.278457
```

The baseline accuracy for our model is 77.8%

Let's make a simple scatter plot, trying features at random. The most obvious trend appears when using Abnormal Short Term Variability and Percentage of Time with Abnormal Long Term Variability. There appears to be enough of a trend in the data that a classification model is feasible.

```
fetalhealth |>
  ggplot() +
  geom_point(aes(y = percentage_of_time_with_abnormal_long_term_variability, x = abnorma
l_short_term_variability, color = factor(fetal_health))) +
  theme_minimal()+
  xlab("Abnormal Short Term Variability") +
  ylab("Percentage of time with Abnormal Long Term Variability") +
  scale_color_brewer(palette = "Accent") +
  labs(color = "Fetal Health")
```



Random Forest Model

Create a train test split

```
set.seed(4) # lucky number
test_rows <- sample(1:nrow(fetalhealth), size = nrow(fetalhealth)/2)
test_data <- fetalhealth[test_rows, ]
train_data <- fetalhealth[-test_rows, ]
```

Let's make a random forest, trained on every feature from the dataset

```
rf <- randomForest(factor(fetal_health) ~ .,
                   data = train_data)

rf
```

```
##
## Call:
##  randomForest(formula = factor(fetal_health) ~ ., data = train_data)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 7.53%
## Confusion matrix:
##     1   2  3 class.error
## 1 799  17  3  0.02442002
## 2  44 101  1  0.30821918
## 3   6   9 83  0.15306122
```

Evaluate our model using Cohen's Kappa.

```
# make predictions on test_data
predictions <- predict(rf, test_data)

# evaluate kappa
kappa_result <- cohen.kappa(table(test_data$fetal_health, predictions))
kappa_result
```

```
## Call: cohen.kappa1(x = x, w = w, n.obs = n.obs, alpha = alpha, levels = levels,
##     w.exp = w.exp)
##
## Cohen Kappa and Weighted Kappa correlation coefficients and confidence boundaries
##                  lower estimate upper
## unweighted kappa  0.80     0.84  0.88
## weighted kappa    0.85     0.89  0.92
##
##  Number of subjects = 1063
```

The random forest model receives a kappa value of 0.80 – this is a strong model.

What features are most important in determining this result?

```
sorted_importance <- rf$importance[order(rf$importance, decreasing = TRUE),]

sorted_importance
```

```
##                       abnormal_short_term_variability
##                                             51.3103879
## percentage_of_time_with_abnormal_long_term_variability
##                                             45.1392031
##                     mean_value_of_short_term_variability
##                                             43.4677426
##                                          histogram_mean
##                                             39.6883619
##                                          histogram_mode
##                                             24.3360514
##                                        histogram_median
##                                             22.7081862
##                               prolongued_decelerations
##                                             21.5506280
##                      mean_value_of_long_term_variability
##                                             19.4311135
##                                       accelerations
##                                             18.8447010
##                                  uterine_contractions
##                                             16.6320490
##                                       histogram_width
##                                             15.6554834
##                                       baseline.value
##                                             15.3936935
##                                        histogram_min
##                                             15.1742379
##                                    histogram_variance
##                                             13.3897464
##                                        histogram_max
##                                             11.7592856
##                                       fetal_movement
##                                             10.4201110
##                               histogram_number_of_peaks
##                                              8.3735453
##                                    light_decelerations
##                                              3.7089357
##                                     histogram_tendency
##                                              3.3072341
##                               histogram_number_of_zeroes
##                                              1.4593406
##                                    severe_decelerations
##                                              0.1238741
```

Supporting what was found in the scatterplot, abnormal_short_term_variability and percentage_of_time_with_abnormal_long_term_variability have the highest impact on random forest model accuracy.

Our Cohen's Kappa value is pretty good, but can we make a better model by applying boosting to our model?

Apply AdaBoost:

```
# Prepare the data
train_data_matrix <- model.matrix(~. - 1, data = train_data)
test_data_matrix <- model.matrix(~. - 1, data = test_data)

label_vector <- as.numeric(train_data$fetal_health) - 1

# Train the model
xgb_model <- xgboost(data = train_data_matrix, label = label_vector, nrounds = 100, verb
ose = FALSE)

# Predict
predictions <- predict(xgb_model, newdata = test_data_matrix)
```

```
test_labels_vector <- as.numeric(test_data$fetal_health) - 1

# Convert predictions to a factor with levels same as test_labels_vector
predictions_factor <- as.factor(round(predictions))
levels(predictions_factor) <- levels(as.factor(test_labels_vector))

# Compute Cohen's Kappa
kappa <- cohen.kappa(data.frame(actual = test_labels_vector, predicted = predictions_fac
tor))

print(kappa)
```

```
## Call: cohen.kappa1(x = x, w = w, n.obs = n.obs, alpha = alpha, levels = levels,
##     w.exp = w.exp)
##
## Cohen Kappa and Weighted Kappa correlation coefficients and confidence boundaries
##                  lower estimate upper
## unweighted kappa     1        1     1
## weighted kappa       1        1     1
##
##  Number of subjects = 1063
```

Yes! We receive a perfect Cohen's kappa score of 1 when using boosting.

Support Vector Machine

I will now try a new algorithm to classify the dataset: Support Vector Machine. SVM works to find the optimal hyperplane to divide classes such that the hyperplane is as far away as possible from the points closest to it. The term "support vector" refers to the points closest to the hyperplane. In a 1d space, the hyperplane is a point. In 2D, it's a line, and in 3D it's a plane, but SVM can be used for very high dimensional data. Only support vectors decide the decision boundary; moving other vectors/points has no impact on moving the decision boundary. SVM utilizes something called a "kernel function" by moving the data to a higher dimension where it is easier to find a way to segregate between the two classes.

First, let's use SVM to predict fetal_health by using 2 features: abnormal short term variability and mean value of short term variability, and a linear kernel function.

```
# Compute class weights
class_freq <- table(train_data$fetal_health)
class_weights <- 1 / class_freq
class_weights <- class_weights / sum(class_weights) # normalize

# initialize svm model
svm_linear = svm(factor(fetal_health) ~ abnormal_short_term_variability + percentage_of_
time_with_abnormal_long_term_variability, data = train_data, kernel = "linear", class.we
ights = class_weights)

# make predictions on test_data
predictions <- predict(svm_linear, test_data)

# Calculate accuracy
accuracy <- mean(predictions == test_data$fetal_health)
print(paste("Accuracy:", accuracy))
```

```
## [1] "Accuracy: 0.61618062088429"
```

We receive an accuracy of 61% – this is below the baseline :(

Cohen Kappa:

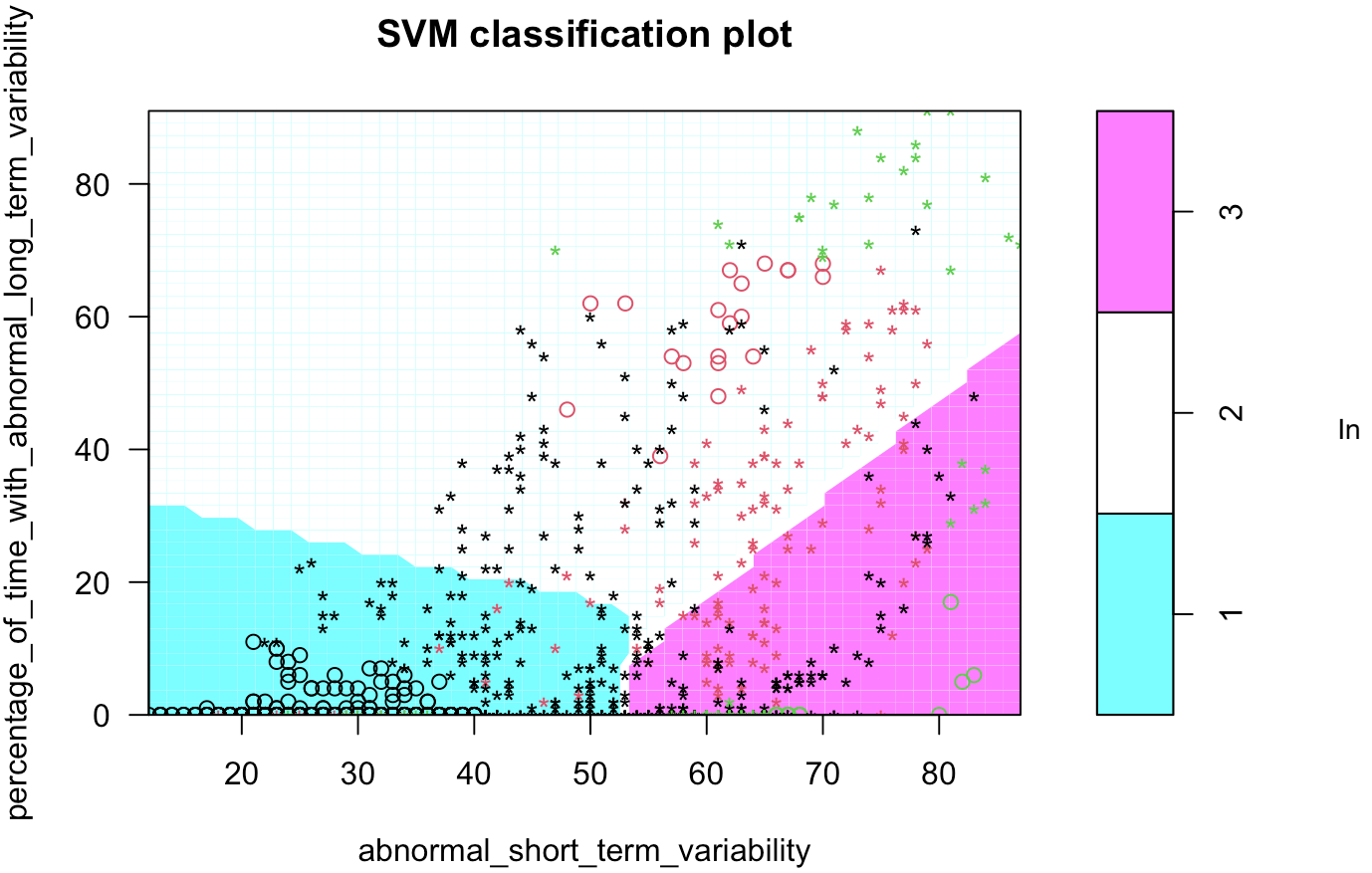
```
svm1_kappa<- cohen.kappa(table(test_data$fetal_health, predictions))
svm1_kappa
```

```
## Call: cohen.kappa1(x = x, w = w, n.obs = n.obs, alpha = alpha, levels = levels,
##     w.exp = w.exp)
##
## Cohen Kappa and Weighted Kappa correlation coefficients and confidence boundaries
##                  lower estimate upper
## unweighted kappa  0.24     0.28  0.32
## weighted kappa    0.19     0.23  0.28
##
##  Number of subjects = 1063
```

A kappa value of 0.24 is quite low!

What's going wrong? Let's visualize the SVM

```
plot(svm_linear, data=train_data, grid = 50, svSymbol = "*", dataSymbol = 1, color.palet
te = cm.colors, percentage_of_time_with_abnormal_long_term_variability ~ abnormal_short_
term_variability)
```



this plot, the support vectors are stars, while the other data points are circles. The colors of the points are the true fetal health classes: 1 is black, 2 is red, and 3 is green. The blue area embodies where SVM classifies as fetal health risk 1, white is 2, and purple is 3. The linear kernel struggles to differentiate between classes.

(note: changing axis labels is not supported in built-in SVM plot function)

What if we use a different kernel?

```
svm_poly = svm(factor(fetal_health) ~ abnormal_short_term_variability + percentage_of_ti
me_with_abnormal_long_term_variability, data = train_data, kernel = "polynomial", class.
weights = class_weights)

# predictions
predictions <- predict(svm_poly, test_data)

#  accuracy
accuracy <- mean(predictions == test_data$fetal_health)
print(paste("Accuracy:", accuracy))
```

```
## [1] "Accuracy: 0.811853245531515"
```

Higher accuracy – a good sign!

Cohen Kappa:

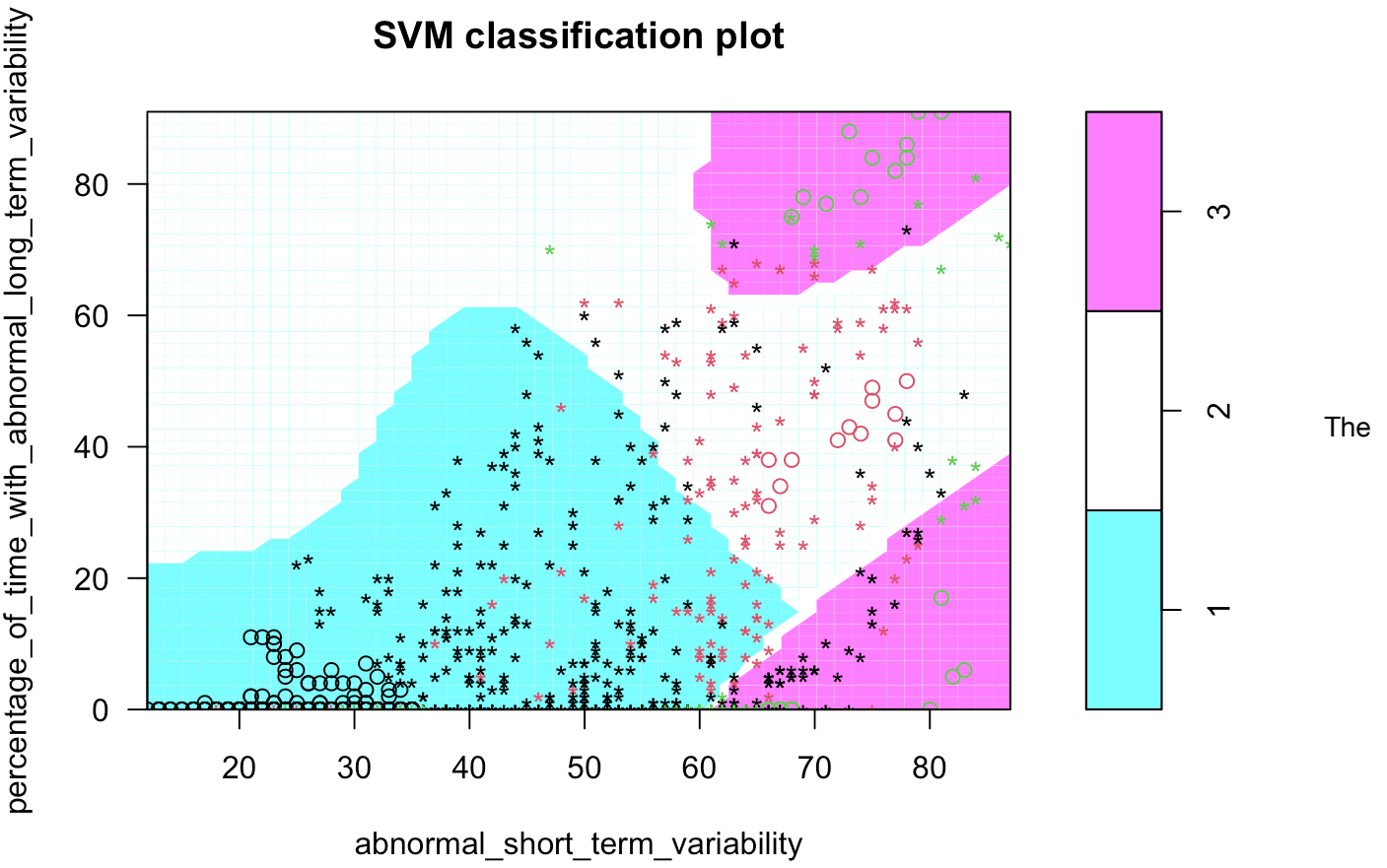
```
svm2_kappa<- cohen.kappa(table(test_data$fetal_health, predictions))
svm2_kappa
```

```
## Call: cohen.kappa1(x = x, w = w, n.obs = n.obs, alpha = alpha, levels = levels,
##     w.exp = w.exp)
##
## Cohen Kappa and Weighted Kappa correlation coefficients and confidence boundaries
##                  lower estimate upper
## unweighted kappa  0.45     0.50  0.56
## weighted kappa    0.36     0.43  0.50
##
##  Number of subjects = 1063
```

Also improvement in Cohen Kappa!

What does SVM with polynomial kernel look like?

```
plot(svm_poly, data=train_data, grid = 50, svSymbol = "*", dataSymbol = 1, color.palette
= cm.colors, percentage_of_time_with_abnormal_long_term_variability ~ abnormal_short_ter
m_variability)
```



The more complex hyperplane implemented by a polynomial kernel is able to pick up on the intricacies of this dataset.

Can we get a higher classification accuracy if we train SVM on every feature and use the polynomial kernel?

```
svm_all = svm(factor(fetal_health) ~ ., data = train_data, kernel = "polynomial")

# predictions
predictions <- predict(svm_all, test_data)

#  accuracy
accuracy <- mean(predictions == test_data$fetal_health)
print(paste("Accuracy:", accuracy))
```

```
## [1] "Accuracy: 0.893697083725306"
```

Intriguing accuracy…

Cohen Kappa:

```
svm_all_kappa <- cohen.kappa(table(test_data$fetal_health, predictions))
svm_all_kappa
```

```
## Call: cohen.kappa1(x = x, w = w, n.obs = n.obs, alpha = alpha, levels = levels,
##     w.exp = w.exp)
##
## Cohen Kappa and Weighted Kappa correlation coefficients and confidence boundaries
##                  lower estimate upper
## unweighted kappa  0.60     0.66  0.71
## weighted kappa    0.73     0.78  0.83
##
##  Number of subjects = 1063
```

A more favorable Cohen Kappa as well!

We can't easily visualize this result because the data is of higher dimension when we use the entire dataset. Additionally, variable importance is more challenging to determine for SVM models so it will be omitted.

Conclusion

I was able to build two effective classification models: a random forest, and a support vector machine. From the variable importance values of the random forest, it appears that abnormal_short_term_variability, percentage_of_time_with_abnormal_long_term_variability, and mean_value_of_short_term_variability are the most important in predicting fetal health. High values of these features are indicators of poor fetal health. In creating these models, I learned more about fetal health predictors and effective classification model implementation.