STAT 337/437 Cleaned Analysis

Esha Ahmad (eahmad1), Michaela Brady (mbrady9), and Emil Cacayan (ecacayan)

04/23/2024

Package Loading

The following packages will be used:

- `readr`: Used for the reading of rectangular data into `R`. This is quicker than the typically used `read.csv()` or `read.table()` methods of loading `.csv` files into `R`, and loads data into `R` as a tibble/dataframe.
- `nnet`: Used for multinomial regression (`multinom()`), and is a package used to fit single-hidden-layer neural networks.
- `car`: Used for `Anova()` statistical test, contains functions to accompany regression tasks.
- `class`: Used to run K nearest neighbors classification.
- `randomForest`: Used for implementation of Random Forest classification method (`randomForest()`).
- `gbm`: Used for implementation of boosting classification method (`gbm()`)
- `tree`: Used to generate decision trees (`tree()`).

```
library(readr)
library(nnet)
library(car)
library(randomForest)
library(gbm)
library(class)
library(tree)
```

Research Question and Introduction to the Dataset

Introduction

Link to Dataset

The dataset linked above titled "Fetal Health Classification" is a classification task classifying the health of a fetus as `Normal`, `Suspect`, or `Pathological` using cardiotocogram exams, which is a continuous recording of the fetal heart rate obtained via an ultrasound transducer placed on the mother's abdomen, which is widely used as a method of assessing fetal well-being, predominantly in pregnancies with increased risk of complications (**note:** this is taken from this paper).

Research Question: Which variables are important in determine the health of a fetus, and which classification method will most accurately classify the health of a fetus (for the population - not just for the sample collected in this analysis)?

Dataset Loading and Cleaning

Let's load this data into R.

```
# Reading the dataset into R.
fetal.health <- read_csv("../Data/fetal_health.csv", show_col_types = FALSE)
```

Now let's see the dimensions of the dataset.

```
# Displaying dimensions of the dataset.
c(`Observations: ` = dim(fetal.health)[1], `Variables: ` = dim(fetal.health)[2])
```

```
## Observations:     Variables:
##           2126             22
```

As we see, we have $n = 2126$ observations and $p = 22$ variables. Let's see the names of these variables.

```
# Displaying names of variables in dataset.
colnames(fetal.health)
```

```
##  [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"
```

So we have the following variables in our dataset:

- `baseline value`: Baseline fetal heart rate (FHR).
- `accelerations`: Number of accelerations per second.
- `fetal_movement`: Number of fetal movements per second.
- `uterine_contractions`: Number of uterine contractions per second.
- `light_decelerations`: Number of LD's per second.
- `severe_decelerations`: Number of SD's per second.
- `prolongued_decelerations`: Number of PD's per second.

- `abnormal_short_term_variability`: Percentage of time with abnormal short term variability.
- `mean_value_of_short_term_variability`: Mean value of short term variability.
- `percentage_of_time_with_abnormal_long_term_variability`: Percentage of time with abnormal long term variability.
- `mean_value_of_long_term_variability`: Mean value of long term variability.
- `histogram_width`: Width of the histogram made using all values from a record.
- `histogram_min`: Histogram minimum value.

- `histogram_max`: Histogram maximum value.
- `histogram_number_of_peaks`: Number of peaks in the exam histogram.
- `histogram_number_of_zeroes`: Number of zeroes in the exam histogram.
- `histogram_mode`: Hist mode.
- `histogram_mean`: Hist mean.
- `histogram_median`: Hist median.
- `histogram_variance`: Hist variance.
- `histogram_tendency`: Histogram trend.
- `fetal_health`: Fetal health with 3 levels:
  - 1 - Normal
  - 2 - Suspect
  - 3 - Pathological

Our response will be `fetal_health`, and the goal We can perform some cleanup of the dataset. While there are no missing data points, we will do two things: * Change the name of the `baseline value` to `baseline_value`. - Many of the functions used in this analysis do not play well with spaces in the variable name. * Change the `fetal_health` variable to a factor variable. - Some of the functions do not automatically cast `fetal_health` (by default a continuous variable) to a factor variable, and since this is a classification problem, the values in `fetal_health` should be discrete classes, not a continuous variable.

```
# Changing the name of baseline value to baseline_value.
colnames(fetal.health)[1] <- "baseline_value"

# Verifying name change.
colnames(fetal.health)[1]
```

```
## [1] "baseline_value"
```

```
# Changing fetal_health to a factor variable.
fetal.health$fetal_health <- as.factor(fetal.health$fetal_health)

# Verifying change.
is.factor(fetal.health$fetal_health)
```

```
## [1] TRUE
```

```
# Seeing levels of fetal_health.
levels(fetal.health$fetal_health)
```

```
## [1] "1" "2" "3"
```

Let's see see the first few data points of the dataset:

```r
# Displaying first 10 data points of the dataset.
head(fetal.health, n = 10)
```

```
## # A tibble: 10 x 22
##    baseline_value accelerations fetal_movement uterine_contractions
##             <dbl>         <dbl>          <dbl>                <dbl>
##  1            120         0                  0                0
##  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                0.01
##  7            134         0.001              0                0.013
##  8            122         0                  0                0
##  9            122         0                  0                0.002
## 10            122         0                  0                0.003
## # i 18 more variables: light_decelerations <dbl>, severe_decelerations <dbl>,
## #   prolongued_decelerations <dbl>, abnormal_short_term_variability <dbl>,
## #   mean_value_of_short_term_variability <dbl>,
## #   percentage_of_time_with_abnormal_long_term_variability <dbl>,
## #   mean_value_of_long_term_variability <dbl>, histogram_width <dbl>,
## #   histogram_min <dbl>, histogram_max <dbl>, histogram_number_of_peaks <dbl>,
## #   histogram_number_of_zeroes <dbl>, histogram_mode <dbl>, ...
```

There are some variables here that have means of 0, or have inputs of very close to 0. As a result, we should be careful if we choose to use regression for this model, as it is highly likely that the regression will either not converge or overfit the data.

For the sake of space, viewing the correlation matrix is not possible. But there are no truly strong or suspicious correlations, and so this data does not show any red flags for collinearity issues.

Analysis

Fitting a Full Multinomial Regression (All Predictors)

Now let's fit the full model using multinomial regression using the `multinom()` function, the `maxit` argument ensures that the regression converges and doesn't stop halfway through:

```r
# Fitting the model.
multinom.mod.full <- multinom(fetal_health ~ ., data = fetal.health,
    maxit = 1000, trace = FALSE)

# Viewing the summary of the model.
summary(multinom.mod.full)
```

```
## Call:
## multinom(formula = fetal_health ~ ., data = fetal.health, maxit = 1000,
##     trace = FALSE)
##
## Coefficients:
##   (Intercept) baseline_value accelerations fetal_movement uterine_contractions
```

```
## 2   -18.18482     -0.1112367     -1012.722       10.02011            -259.7271
## 3   -19.24030      0.2644462     -1036.211       21.54699            -304.1431
##   light_decelerations severe_decelerations prolongued_decelerations
## 2           -1.440745            -78.74683                 2228.491
## 3           19.692999            143.10203                 2334.992
##   abnormal_short_term_variability mean_value_of_short_term_variability
## 2                      0.08660964                           -0.2225313
## 3                      0.18171629                           -1.3422267
##   percentage_of_time_with_abnormal_long_term_variability
## 2                                             0.02284919
## 3                                             0.07581845
##   mean_value_of_long_term_variability histogram_width histogram_min
## 2                       -0.004857624      0.00529124    0.01893658
## 3                        0.066606084      0.02141726    0.03112828
##   histogram_max histogram_number_of_peaks histogram_number_of_zeroes
## 2    0.02422759                 0.1403659                -0.07215815
## 3    0.05254552                -0.4553416                 0.68734821
##   histogram_mode histogram_mean histogram_median histogram_variance
## 2    -0.07136585     0.28431132      -0.05632803         0.04304709
## 3    -0.05184160    -0.06509058      -0.20306017         0.07169165
##   histogram_tendency
## 2          0.2767743
## 3          0.6144064
##
## Std. Errors:
##   (Intercept) baseline_value accelerations fetal_movement uterine_contractions
## 2   1.6224882     0.02693425  0.0032153398      0.3840921          0.003905614
## 3   0.4069886     0.04113179  0.0006750457      0.1241618          0.004087520
##   light_decelerations severe_decelerations prolongued_decelerations
## 2        0.0004148428         4.223382e-05             0.0013203952
## 3        0.0015452002         1.597897e-05             0.0002949339
##   abnormal_short_term_variability mean_value_of_short_term_variability
## 2                      0.00980797                            0.2660468
## 3                      0.01819326                            0.4473236
##   percentage_of_time_with_abnormal_long_term_variability
## 2                                            0.005892841
## 3                                            0.009478830
##   mean_value_of_long_term_variability histogram_width histogram_min
## 2                          0.03189494     0.003950814   0.007291756
## 3                          0.06305003     0.005937446   0.010821403
##   histogram_max histogram_number_of_peaks histogram_number_of_zeroes
## 2   0.009489542                0.05285359                 0.1400570
## 3   0.011870063                0.11209545                 0.2690968
##   histogram_mode histogram_mean histogram_median histogram_variance
## 2     0.02824436     0.05478777       0.06587291        0.007898634
## 3     0.03498425     0.04474168       0.05342962        0.010616550
##   histogram_tendency
## 2          0.2577197
## 3          0.4037197
##
## Residual Deviance: 904.6783
## AIC: 988.6783
```

`multinom()` fits a baseline category logit model (BCL) and each set of $\hat{\beta}$ parameters estimated compares the

log odds with a baseline category (which is category 1, normal).

Odds Ratios

You can skip over this part if you want

The interpretation of a baseline category logit model is usually through the log odds, which is calculated as follows:

$$\text{OR}_i = e^{c\beta_i}$$

For instance, for the parameters estimated for `uterine_contractions`, we get the following:

```
# Calculating odds ratio.
OR_contractions.2 <- exp(coefficients(multinom.mod.full)[9])

# Displaying odds ratio.
OR_contractions.2
```

```
## [1] 1.592047e-113
```

This means that for a 1 unit increase in `uterine_contractions`, the odds of an infant being in the suspect category (vs. the normal category) changes by a factor of $1.5920468 \times 10^{-113}$ times, meaning overall the odds are smaller for each 1 unit increase in `uterine_contractions`. In other words, increases in `uterine_contractions` scales the odds of an infant being in the normal category (vs. the suspect category) upward. You can do this for the remaining variables, but in general, parameter estimates greater than 0 scale the odds upward, while parameter estimates less than 0 scale the odds downward, and odds ratios equal to 0 do not change the odds.

Let's do one more example, this time for `baseline_value`.

```
# Calculating odds ratios.
OR_baseline.2 <- exp(coefficients(multinom.mod.full)[3])
OR_baseline.3 <- exp(coefficients(multinom.mod.full)[4])

# Displaying odds ratios.
(c(`2 vs. 1` = OR_baseline.2, `3 vs. 1` = OR_baseline.3))
```

```
##   2 vs. 1   3 vs. 1
## 0.8947269 1.3027093
```

I'll just put a shorthand interpretation here. The first odds ratio says that keeping all other variables equal, a 1 unit increase in `baseline_value` changes the odds of being in the suspect category (vs. normal) by a factor of 0.8947, and changes the odds of being in the pathological category (vs. normal) by a factor of 1.3027. This means that a faster heart rate will cause the odds of you being suspect (vs. normal) to become smaller, but the odds of you being suspect (vs. normal) to be larger. This implies that at the lower range of the heart rates recorded in this dataset, an increase is generally better, but there must be some cutoff point where increases cause a more negative fetal outcome, but this is speculative as more models need to be fit (perhaps with a different baseline than 1).

Hypothesis Test: Anova

We can run `Anova()` from the `car` package, which tests the following hypotheses:

$$H_0 : \text{All } \beta_j = 0, j = 1 \dots p$$
$$H_a : \text{At least 1 } \beta_j \neq 0, j = 1 \dots p$$

This uses a likelihood ratio test which follows a χ^2 distribution.

```
# Running Anova.
Anova(multinom.mod.full)
```

```
## Analysis of Deviance Table (Type II tests)
##
## Response: fetal_health
##                                                          LR Chisq Df Pr(>Chisq)
## baseline_value                                            110.074  2  < 2.2e-16
## accelerations                                             229.339  2  < 2.2e-16
## fetal_movement                                             95.673  2  < 2.2e-16
## uterine_contractions                                      116.772  2  < 2.2e-16
## light_decelerations                                        81.083  2  < 2.2e-16
## severe_decelerations                                       73.759  2  < 2.2e-16
## prolongued_decelerations                                   77.279  2  < 2.2e-16
## abnormal_short_term_variability                           180.222  2  < 2.2e-16
## mean_value_of_short_term_variability                       70.053  2  6.140e-16
## percentage_of_time_with_abnormal_long_term_variability    127.904  2  < 2.2e-16
## mean_value_of_long_term_variability                        61.938  2  3.550e-14
## histogram_width                                            79.953  2  < 2.2e-16
## histogram_min                                              93.497  2  < 2.2e-16
## histogram_max                                              59.675  2  1.101e-13
## histogram_number_of_peaks                                  82.077  2  < 2.2e-16
## histogram_number_of_zeroes                                 72.399  2  < 2.2e-16
## histogram_mode                                             64.281  2  1.100e-14
## histogram_mean                                             96.061  2  < 2.2e-16
## histogram_median                                          109.991  2  < 2.2e-16
## histogram_variance                                        132.318  2  < 2.2e-16
## histogram_tendency                                         67.111  2  2.673e-15
##
## baseline_value                                           ***
## accelerations                                            ***
## fetal_movement                                           ***
## uterine_contractions                                     ***
## light_decelerations                                      ***
## severe_decelerations                                     ***
## prolongued_decelerations                                 ***
## abnormal_short_term_variability                          ***
## mean_value_of_short_term_variability                     ***
## percentage_of_time_with_abnormal_long_term_variability   ***
## mean_value_of_long_term_variability                      ***
## histogram_width                                          ***
## histogram_min                                            ***
## histogram_max                                            ***
## histogram_number_of_peaks                                ***
```

```
## histogram_number_of_zeroes                                    ***
## histogram_mode                                                ***
## histogram_mean                                                ***
## histogram_median                                              ***
## histogram_variance                                            ***
## histogram_tendency                                            ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As we can see, all variables are considered highly significant according to the p-values with a significance threshold of $\alpha = 0.05$. This is a red flag because this indicates that our model is overfitting the dataset. It is difficult to compare p-values because they are so small, so for ease of interpretations, we will be looking at the largest (relative to the remaining variables) test statistics generated by this test for the "most significant" variables:

```
Anova(multinom.mod.full)[1]
```

```
##                                                          LR Chisq
## baseline_value                                            110.074
## accelerations                                             229.339
## fetal_movement                                             95.673
## uterine_contractions                                      116.772
## light_decelerations                                        81.083
## severe_decelerations                                       73.759
## prolongued_decelerations                                   77.279
## abnormal_short_term_variability                           180.222
## mean_value_of_short_term_variability                       70.053
## percentage_of_time_with_abnormal_long_term_variability    127.904
## mean_value_of_long_term_variability                        61.938
## histogram_width                                            79.953
## histogram_min                                              93.497
## histogram_max                                              59.675
## histogram_number_of_peaks                                  82.077
## histogram_number_of_zeroes                                 72.399
## histogram_mode                                             64.281
## histogram_mean                                             96.061
## histogram_median                                          109.991
## histogram_variance                                        132.318
## histogram_tendency                                         67.111
```

So our variables of note are (using a cutoff of around 100):

- `baseline_value`: `LR Chisq` = 110.074
- `accelerations`: `LR chisq` = 229.339
- `light_decelerations`: `LR Chisq` = 116.772
- `abnormal_short_term_variability`: `LR Chisq` = 180.222
- `percentage_of_time_with_abnormal_long_term_variability` = 127.904
- `histogram_median`: `LR Chisq` = 109.991
- `histogram_variance`: `LR Chisq` = 132.318

Let's fit a new model with only these variables.

```r
# Fitting a new multinomial model.
multinom.mod.reduced <- multinom(fetal_health ~ baseline_value +
    accelerations + light_decelerations + abnormal_short_term_variability +
    percentage_of_time_with_abnormal_long_term_variability +
    histogram_median + histogram_variance, data = fetal.health,
    trace = FALSE, maxiter = 1000)
```

And compare how this model performs against the original model. Here we will use the `anova()` function in base `R`, which is the same test. But in the `Anova()` function, `R` starts with the full model and removes each variable and tests how RSS increases. `anova()` starts with an empty model and shows how RSS decreases as each predictor is added.

```r
anova(multinom.mod.reduced, multinom.mod.full)[, 6:7]
```

```
##   LR stat. Pr(Chi)
## 1       NA      NA
## 2 244.1428       0
```

We get a very large test statistic and a very small p-value, which indicates that we prefer the new model over the original large model. This is the multinomial model we will use to classify the data.

Let's run `Anova()` once more on this new model.

```r
# Running Anova on the reduced model.
Anova(multinom.mod.reduced)
```

```
## Analysis of Deviance Table (Type II tests)
##
## Response: fetal_health
##                                                        LR Chisq Df Pr(>Chisq)
## baseline_value                                           94.595  2  < 2.2e-16
## accelerations                                           164.852  2  < 2.2e-16
## light_decelerations                                      56.308  2  5.928e-13
## abnormal_short_term_variability                         191.879  2  < 2.2e-16
## percentage_of_time_with_abnormal_long_term_variability   71.179  2  3.497e-16
## histogram_median                                        200.220  2  < 2.2e-16
## histogram_variance                                      103.595  2  < 2.2e-16
##
## baseline_value                                         ***
## accelerations                                          ***
## light_decelerations                                    ***
## abnormal_short_term_variability                        ***
## percentage_of_time_with_abnormal_long_term_variability ***
## histogram_median                                       ***
## histogram_variance                                     ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We are suspicious still of overfitting because of the small p-values, and so we would like to reduce this model further a selection criteria such as `AIC()` and compare the performance of this model with other classification methods.

AIC Selection

AIC will help with our overfitting problem by adding a penalty for the complexity of the model. Let's see the AIC for each of the models we fit.

```
# Displaying AIC for each model.
(c(`AIC Full: ` = extractAIC(multinom.mod.full)[[2]], `AIC Reduced: ` = extractAIC(multinom.mod.reduced
```

```
##    AIC Full:  AIC Reduced:
##      988.6783     1180.8211
```

As we can see, reducing our model improved the AIC significantly. Let's see if this AIC can be improved further. We will utilize backwards AIC to select a new model - using hypothesis testing might be somewhat erroneous because of the overfitting problem we are running into.

```
# Performing backwards AIC.
bac.aic.mod <- step(object = multinom.mod.reduced, direction = "backward",
    k = 2, trace = FALSE)
```

```
## trying - baseline_value
## trying - accelerations
## trying - light_decelerations
## trying - abnormal_short_term_variability
## trying - percentage_of_time_with_abnormal_long_term_variability
## trying - histogram_median
## trying - histogram_variance
```

Let's compare the AIC for the models.

```
(c(`Original Model: ` = extractAIC(multinom.mod.reduced)[[2]],
    `AIC Selected Model: ` = extractAIC(bac.aic.mod)[2]))
```

```
##     Original Model:  AIC Selected Model:
##             1180.821             1180.821
```

So as we can see, our backwards AIC selection did not change the model, and so we will keep all the variables for our model. Let's test the prediction accuracy of this model using $K = 20$ cross-validation (LOOCV not computationally realistic). Each time the `for()` loop is run, the reduced model is generated and then undergoes backwards AIC selection.

```
# Setting seed for consistency of results.
set.seed(2022)

# Randomly shuffling the index of the data.
index.random <- sample(1:dim(fetal.health)[1])

# Splitting the data for K = 20 cross-validation.
groups <- cut(1:dim(fetal.health)[1], 20, labels = FALSE)
index.fold <- split(index.random, groups)

# An empty vector to save prediction for fetal_health.
```

```r
health.pred <- c()

# Running cross-validation.
for (index.test in index.fold) {
    # Creating training and test set.
    data.test <- fetal.health[index.test, ]
    data.train <- fetal.health[-index.test, ]

    # Fitting an aic selected model on the training set.
    bac.aic.mod.tmp <- step(object = multinom(fetal_health ~
        baseline_value + accelerations + light_decelerations +
            abnormal_short_term_variability + percentage_of_time_with_abnormal_long_term_variability +
            histogram_median + histogram_variance, data = data.train,
        maxit = 1000, trace = FALSE), direction = "backward",
        k = 2, trace = FALSE)

    # Predicting on the test set.
    pred <- predict(bac.aic.mod.tmp, data.test, type = "class")

    # Saving the predict fetal_health predictions.
    health.pred <- c(health.pred, pred)
}
```

```
## trying - baseline_value
## trying - accelerations
## trying - light_decelerations
## trying - abnormal_short_term_variability
## trying - percentage_of_time_with_abnormal_long_term_variability
## trying - histogram_median
## trying - histogram_variance
## trying - baseline_value
## trying - accelerations
## trying - light_decelerations
## trying - abnormal_short_term_variability
## trying - percentage_of_time_with_abnormal_long_term_variability
## trying - histogram_median
## trying - histogram_variance
## trying - baseline_value
## trying - accelerations
## trying - light_decelerations
## trying - abnormal_short_term_variability
## trying - percentage_of_time_with_abnormal_long_term_variability
## trying - histogram_median
## trying - histogram_variance
## trying - baseline_value
## trying - accelerations
## trying - light_decelerations
## trying - abnormal_short_term_variability
## trying - percentage_of_time_with_abnormal_long_term_variability
## trying - histogram_median
## trying - histogram_variance
## trying - baseline_value
## trying - accelerations
```

```
## trying - light_decelerations
## trying - abnormal_short_term_variability
## trying - percentage_of_time_with_abnormal_long_term_variability
## trying - histogram_median
## trying - histogram_variance
## trying - baseline_value
## trying - accelerations
## trying - light_decelerations
## trying - abnormal_short_term_variability
## trying - percentage_of_time_with_abnormal_long_term_variability
## trying - histogram_median
## trying - histogram_variance
## trying - baseline_value
## trying - accelerations
## trying - light_decelerations
## trying - abnormal_short_term_variability
## trying - percentage_of_time_with_abnormal_long_term_variability
## trying - histogram_median
## trying - histogram_variance
## trying - baseline_value
## trying - accelerations
## trying - light_decelerations
## trying - abnormal_short_term_variability
## trying - percentage_of_time_with_abnormal_long_term_variability
## trying - histogram_median
## trying - histogram_variance
## trying - baseline_value
## trying - accelerations
## trying - light_decelerations
## trying - abnormal_short_term_variability
## trying - percentage_of_time_with_abnormal_long_term_variability
## trying - histogram_median
## trying - histogram_variance
## trying - baseline_value
## trying - accelerations
## trying - light_decelerations
## trying - abnormal_short_term_variability
## trying - percentage_of_time_with_abnormal_long_term_variability
## trying - histogram_median
## trying - histogram_variance
## trying - baseline_value
## trying - accelerations
## trying - light_decelerations
## trying - abnormal_short_term_variability
## trying - percentage_of_time_with_abnormal_long_term_variability
## trying - histogram_median
## trying - histogram_variance
## trying - baseline_value
## trying - accelerations
## trying - light_decelerations
## trying - abnormal_short_term_variability
## trying - percentage_of_time_with_abnormal_long_term_variability
## trying - histogram_median
## trying - histogram_variance
```

```
## trying - baseline_value
## trying - accelerations
## trying - light_decelerations
## trying - abnormal_short_term_variability
## trying - percentage_of_time_with_abnormal_long_term_variability
## trying - histogram_median
## trying - histogram_variance
## trying - baseline_value
## trying - accelerations
## trying - light_decelerations
## trying - abnormal_short_term_variability
## trying - percentage_of_time_with_abnormal_long_term_variability
## trying - histogram_median
## trying - histogram_variance
## trying - baseline_value
## trying - accelerations
## trying - light_decelerations
## trying - abnormal_short_term_variability
## trying - percentage_of_time_with_abnormal_long_term_variability
## trying - histogram_median
## trying - histogram_variance
## trying - baseline_value
## trying - accelerations
## trying - light_decelerations
## trying - abnormal_short_term_variability
## trying - percentage_of_time_with_abnormal_long_term_variability
## trying - histogram_median
## trying - histogram_variance
## trying - baseline_value
## trying - accelerations
## trying - light_decelerations
## trying - abnormal_short_term_variability
## trying - percentage_of_time_with_abnormal_long_term_variability
## trying - histogram_median
## trying - histogram_variance
## trying - baseline_value
## trying - accelerations
## trying - light_decelerations
## trying - abnormal_short_term_variability
## trying - percentage_of_time_with_abnormal_long_term_variability
## trying - histogram_median
## trying - histogram_variance
## trying - baseline_value
## trying - accelerations
## trying - light_decelerations
## trying - abnormal_short_term_variability
## trying - percentage_of_time_with_abnormal_long_term_variability
## trying - histogram_median
## trying - histogram_variance
## trying - baseline_value
## trying - accelerations
## trying - light_decelerations
## trying - abnormal_short_term_variability
## trying - percentage_of_time_with_abnormal_long_term_variability
```

```
## trying - histogram_median
## trying - histogram_variance
```

Now let's see how well the model fitting performed.

```
# Calculating prediction accuracy.
health.truth <- fetal.health$fetal_health[index.random]
pred.acc.aic <- mean(health.pred == health.truth)
(c(`Prediction Accuracy for AIC Model: ` = pred.acc.aic))
```

```
## Prediction Accuracy for AIC Model:
##                           0.8809972
```

```
# Displaying confusion matrix.
table(health.pred, health.truth)
```

```
##            health.truth
## health.pred    1    2    3
##           1 1568  104   15
##           2   71  175   31
##           3   16   16  130
```

Let's compare this to the full model:

```
# Setting seed for consistency of results.
set.seed(2022)

# Randomly shuffling the index of the data.
index.random <- sample(1:dim(fetal.health)[1])

# Splitting the data for K = 20 cross-validation.
groups <- cut(1:dim(fetal.health)[1], 20, labels = FALSE)
index.fold <- split(index.random, groups)

# An empty vector to save prediction for fetal_health.
health.pred <- c()

# Running cross-validation.
for (index.test in index.fold) {
    # Creating training and test set.
    data.test <- fetal.health[index.test, ]
    data.train <- fetal.health[-index.test, ]

    # Fitting a multinomial model based on training set.
    full.mod.tmp <- multinom(fetal_health ~ ., data = data.train,
        trace = FALSE)

    # Predicting on the test set.
    pred <- predict(full.mod.tmp, data.test, type = "class")

    # Saving the predict fetal_health predictions.
    health.pred <- c(health.pred, pred)
}
```

Now let's see how well this model performs:

```
# Calculating prediction accuracy.
health.truth <- fetal.health$fetal_health[index.random]
pred.acc.full <- mean(health.pred == health.truth)
(c(`Prediction Accuracy for Full Model: ` = pred.acc.full))
```

```
## Prediction Accuracy for Full Model:
##                            0.8960489
```

```
# Displaying confusion matrix.
table(health.pred, health.truth)
```

```
##             health.truth
## health.pred    1    2    3
##           1 1579   84   11
##           2   66  188   27
##           3   10   23  138
```

So our full model has a 1.6% incrase in accuracy. In terms of explaining the model, the smaller is superior, but the larger model has better prediction accuracy. But for the computational overhead required to use the full model for regression task, it is questionable whether the 1.6% increase in accuracy is truly worth it - and if some of the variables are so incredibly biased (some of the variables mostly 0 values) that the full model is still overfitting the test data even after cross-validation.

Let's compare how well these performs with other classification methods. All classification methods will also be cross-validated.

K Nearest Neighbors Classification

We will use cross-validation for as well. From previous tests, it seems $K = 4$ produces the highest accuracy without too much overfitting.

```
# Setting seed for consistency of results.
set.seed(2022)

# Randomly shuffling the index of the data.
index.random <- sample(1:dim(fetal.health)[1])

# Splitting the data for K = 20 cross-validation.
groups <- cut(1:dim(fetal.health)[1], 20, labels = FALSE)
index.fold <- split(index.random, groups)

# An empty vector to save prediction for fetal_health.
health.pred <- c()

# Running cross-validation.
for (index.test in index.fold) {
    # Creating training and test set.
    data.test <- fetal.health[index.test, ]
    data.train <- fetal.health[-index.test, ]
```

```
    # Classifying using k = 4 nearest neighbors.
    pred <- knn(train = data.train[, -22], test = data.test[,
        -22], cl = data.train$fetal_health, k = 4)

    # Saving the predict fetal_health predictions.
    health.pred <- c(health.pred, pred)
}
```

Now let's see how well this classifier performs:

```
# Calculating prediction accuracy.
health.truth <- fetal.health$fetal_health[index.random]
pred.acc.knn <- mean(health.pred == health.truth)
(c(`Prediction Accuracy for KNN: ` = pred.acc.knn))
```

```
## Prediction Accuracy for KNN:
##                     0.9016933
```

```
# Displaying confusion matrix.
table(health.pred, health.truth)
```

```
##            health.truth
## health.pred    1    2    3
##           1 1591  102   19
##           2   53  185   16
##           3   11    8  141
```

After being cross-validated, this classification method outperforms both of the multinomial regression models tested earlier.

Decision Tree

```
# Setting seed for consistency of results.
set.seed(2022)

# Randomly shuffling the index of the data.
index.random <- sample(1:dim(fetal.health)[1])

# Splitting the data for K = 20 cross-validation.
groups <- cut(1:dim(fetal.health)[1], 20, labels = FALSE)
index.fold <- split(index.random, groups)

# An empty vector to save prediction for fetal_health.
health.pred <- c()

# Running cross-validation.
for (index.test in index.fold) {
    # Creating training and test set.
    data.test <- fetal.health[index.test, ]
    data.train <- fetal.health[-index.test, ]
```

```
    # Creating trees.
    tree.fetal_health <- tree(fetal_health ~ ., data = data.train)

    # Predicting based off of trees.
    pred <- predict(tree.fetal_health, newdata = data.test, type = "class")

    # Saving the predict fetal_health predictions.
    health.pred <- c(health.pred, pred)
}
```

Now let's see how well this classifier performs:

```
# Calculating prediction accuracy.
health.truth <- fetal.health$fetal_health[index.random]
pred.acc.tree <- mean(health.pred == health.truth)
(c(`Prediction Accuracy for Decision Tree: ` = pred.acc.tree))
```

```
## Prediction Accuracy for Decision Tree:
##                               0.9317968
```

```
# Displaying confusion matrix.
table(health.pred, health.truth)
```

```
##            health.truth
## health.pred    1    2    3
##           1 1619   91    8
##           2   18  198    4
##           3   18    6  164
```

This classification method performs the best out of the methods tested so far, indicating that logistic regression and KNN may not be the best method of testing the dataset.

Bagging, Random Forest, and Boosting

Bagging

The value of `mtry` here is the number of total variables excluding the explanatory variable.

```
# Setting the seed for consistency of results.
set.seed(2022)

# Randomly shuffling the index of the data.
index.random <- sample(1:dim(fetal.health)[1])

# Splitting the data for K = 20 cross-validation.
groups <- cut(1:dim(fetal.health)[1], 20, labels = FALSE)
index.fold <- split(index.random, groups)

# An empty vector to save prediction for fetal_health.
```

```
health.pred <- c()

# Running cross-validation.
for (index.test in index.fold) {
    # Creating training and test set.
    data.test <- fetal.health[index.test, ]
    data.train <- fetal.health[-index.test, ]

    # Performing bagging.
    bagging.fetal_health <- randomForest(fetal_health ~ ., mtry = ncol(fetal.health) -
        1, ntree = 500, importance = TRUE, data = data.train)

    # Predicting based off bagging.
    pred <- predict(bagging.fetal_health, newdata = data.test,
        type = "class")

    # Saving the predict fetal_health predictions.
    health.pred <- c(health.pred, pred)
}
```

Now let's see how well this classifier performs:

```
# Calculating prediction accuracy.
health.truth <- fetal.health$fetal_health[index.random]
pred.acc.bag <- mean(health.pred == health.truth)
(c(`Prediction Accuracy for Bagging: ` = pred.acc.bag))
```

```
## Prediction Accuracy for Bagging:
##                          0.9430856
```

```
# Displaying confusion matrix.
table(health.pred, health.truth)
```

```
##            health.truth
## health.pred    1    2    3
##           1 1614   61   10
##           2   33  231    6
##           3    8    3  160
```

Bagging is a better classification method than decision tree as it uses an ensemble of trees.

Random Forest

The value of `mtry` here is roughtly $\sqrt{p}$.

```
# Setting the seed for consistency of results.
set.seed(2022)

# Randomly shuffling the index of the data.
index.random <- sample(1:dim(fetal.health)[1])
```

```
# Splitting the data for K = 20 cross-validation.
groups <- cut(1:dim(fetal.health)[1], 20, labels = FALSE)
index.fold <- split(index.random, groups)

# An empty vector to save prediction for fetal_health.
health.pred <- c()

# Running cross-validation.
for (index.test in index.fold) {
    # Creating training and test set.
    data.test <- fetal.health[index.test, ]
    data.train <- fetal.health[-index.test, ]

    # Performing random forest.
    rf.fetal_health <- randomForest(fetal_health ~ ., mtry = round(sqrt(ncol(fetal.health) -
        1)), ntree = 500, importance = TRUE, data = data.train)

    # Predicting based off random forest.
    pred <- predict(rf.fetal_health, newdata = data.test, type = "class")

    # Saving the predict fetal_health predictions.
    health.pred <- c(health.pred, pred)
}
```

Now let's see how well this classifier performs:

```
# Calculating prediction accuracy.
health.truth <- fetal.health$fetal_health[index.random]
pred.acc.rf <- mean(health.pred == health.truth)
(c(`Prediction Accuracy for Random Forest: ` = pred.acc.rf))
```

```
## Prediction Accuracy for Random Forest:
##                               0.9515522
```

```
# Displaying confusion matrix.
table(health.pred, health.truth)
```

```
##            health.truth
## health.pred    1    2    3
##           1 1631   58    9
##           2   21  233    8
##           3    3    4  159
```

Boosting is better than bagging for prediction accuracy, which reduces the overfitting/variance of the model.

Boosting

Boosting's implementation for a multinomial response is broken, but we will test it anyway. Normally, this `R` code will throw warnings for this, but these have been supressed.

```r
# Setting the seed for consistency of results.
set.seed(2022)

# Randomly shuffling the index of the data.
index.random <- sample(1:dim(fetal.health)[1])

# Splitting the data for K = 20 cross-validation.
groups <- cut(1:dim(fetal.health)[1], 20, labels = FALSE)
index.fold <- split(index.random, groups)

# An empty vector to save prediction for fetal_health.
health.pred <- c()

# Running cross-validation.
for (index.test in index.fold) {
    # Creating training and test set.
    data.test <- fetal.health[index.test, ]
    data.train <- fetal.health[-index.test, ]

    # Performing boosting.
    boost.fetal_health <- gbm(fetal_health ~ ., n.trees = 500,
        distribution = "multinomial", data = data.train)

    # Predicting based off boosting.
    prob <- predict(boost.fetal_health, newdata = data.test,
        type = "response")
    pred <- max.col(matrix(unlist(prob), ncol = 3, byrow = T),
        "first")

    # Saving the predict fetal_health predictions.
    health.pred <- c(health.pred, pred)
}
```

Now let's see how well this classifier performs:

```r
# Calculating prediction accuracy.
health.truth <- fetal.health$fetal_health[index.random]
pred.acc.boost <- mean(health.pred == health.truth)
(c(`Prediction Accuracy for Boosting: ` = pred.acc.boost))
```

```
## Prediction Accuracy for Boosting:
##                           0.341016
```

```r
# Displaying confusion matrix.
table(health.pred, health.truth)
```

```
##            health.truth
## health.pred   1   2   3
##           1 567 100  56
##           2 537  97  59
##           3 551  98  61
```

The inaccuracy is likely due to the implementation of this classifier still being broken.

Summary

The best classification method seems to be random forest.

The variables with the strongest relationship with the response variable seem to be:

- `baseline_value`: `LR Chisq` = 110.074
- `accelerations`: `LR chisq` = 229.339
- `light_decelerations`: `LR Chisq` = 116.772
- `abnormal_short_term_variability`: `LR Chisq` = 180.222
- `percentage_of_time_with_abnormal_long_term_variability` = 127.904
- `histogram_median`: `LR Chisq` = 109.991
- `histogram_variance`: `LR Chisq` = 132.318

Things to do next:

- Summarize results in a table.
- Explain strengths of using logistic multinomial regression (interpretability).
- Explain strengths of using other classifiers.