Fetal Health Findings

Finn de Lange

2025-07-06

Background

A cardiotocogram is a technique used to monitor fetal heart heart, as well as uterine contractions during pregnancy and labour. The information gathered from these can be used as a indicator of the health of the fetus, enabling clinicians to successfully identify feti in need of further care from those that are healthy.

Our goal is to develop an accurate prediction model using data found through Kaggle, provided by Ayes de Campos et al. (https://onlinelibrary.wiley.com/doi/10.1002/1520-6661(200009/10)9:5%3C311::AID-MFM12%3E3.0.CO;2-9).

Logistic Regression

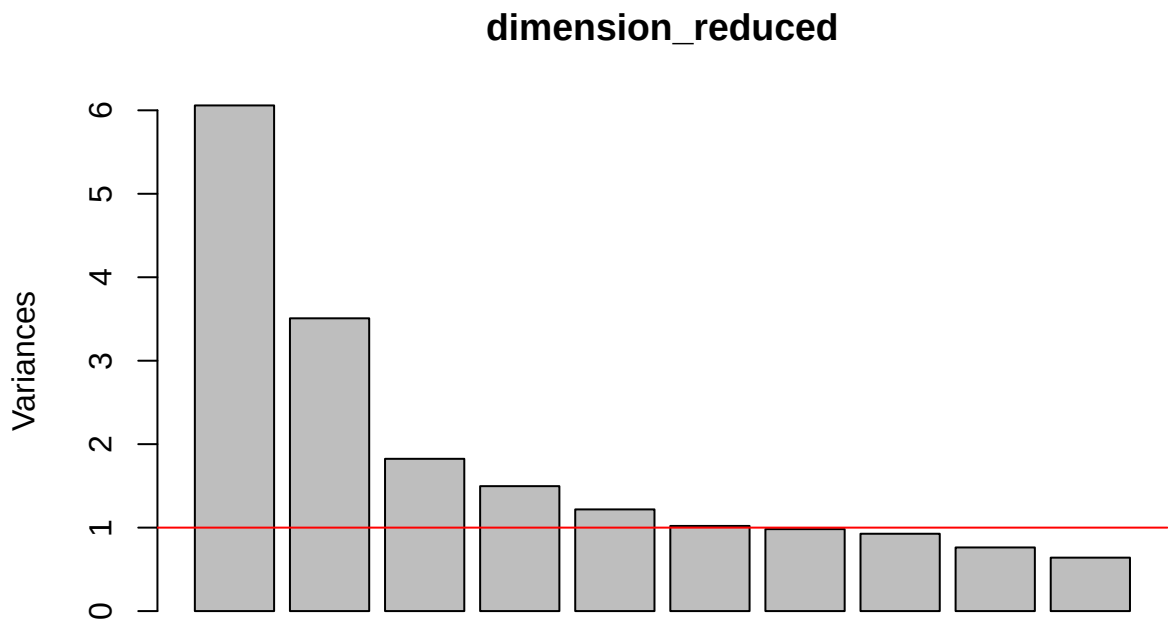
Logistic regression is an excellent option when handling binary cases, such as if someone has cancer, or not. Unfortunately in this case, our data was coded into three groups, those being normal, suspect and pathological.

In order to utilise logistic regression here, we can collapse the suspect and pathological groups into a single group, which we will consider the "at-risk" group. With this, we can now perform logistic regression.

However, before we do, I wanted to try using principal components logistic regression for this data set. This comes with many more challenges than a typical multiple logistic regression model, especially in terms of interpretability and ease of use. However, I think that this is a good decision here, as many of these variables are quite likely to be highly correlated with each other, and principal components allows us to somewhat handle this. Additionally, I hope that the loadings of the principal components may shed insights into the more useful variables, especially when we later consider other multivariate modelling techniques.

Let us first work through the principal components.

```
data <- data %>%
  mutate(
    recoded = ifelse(fetal_health == 1, 0, 1)
  ) %>%
  dplyr::select(-fetal_health)

dimension_reduced <- prcomp(data[,1:21], scale = TRUE, center = TRUE)

screeplot(dimension_reduced)
abline(1, 0, col="red")
```




```
sum(dimension_reduced$sdev[1:4]^2)/sum(dimension_reduced$sdev^2)
```

```
## [1] 0.6136749
```

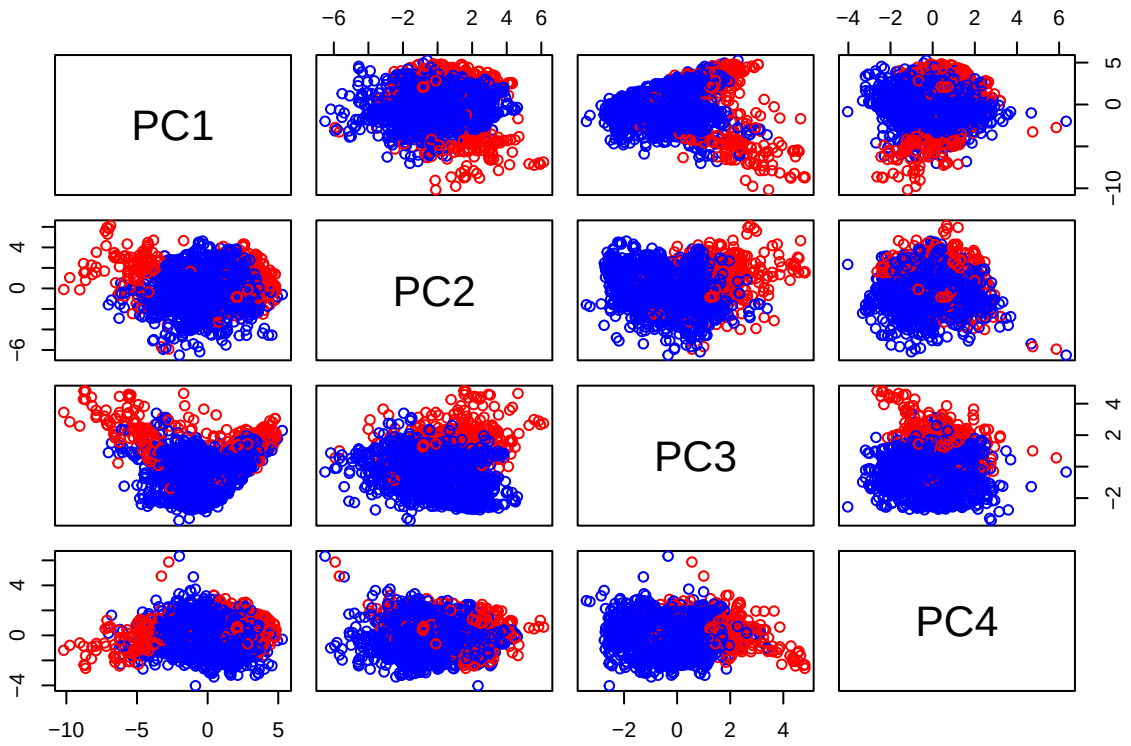
```
round(dimension_reduced$sdev^2 / sum(dimension_reduced$sdev^2), 2)
```

```
##  [1] 0.29 0.17 0.09 0.07 0.06 0.05 0.05 0.04 0.04 0.03 0.03 0.02 0.02 0.02 0.01
## [16] 0.01 0.01 0.01 0.00 0.00 0.00
```

Using the elbow rule would suggest taking two principal components, but this would result in capturing less than half the variation in the data, and the eignevalue/variance greater-than-one rule would select too many components for this to be a practical option. Instead, let us consider the number of components required to encapsulate at least 60% of the data; 4, at 61.4%.

Let us inspect the scatter plots using these as derived axes, colouring by our new groups.

```
colours <- ifelse(data$recoded == 1, 'red', 'blue')
pairs(dimension_reduced$x[,1:4], col=colours)
```



The pairs plot shows that many of these axes result in significant overlapping of the groups, so it is unlikely that one of these pairings would provide a suitable classification method.

It is worth noting that the third principal component seems to do a better job than the others at separating the groups. Later on this report we will return to this observation.

Now that we have the derived variables, we can begin trying to build a logistic regression model on these.

First, we will check if the derived variables may benefit from a transformation using GAM modelling.

```
library(mgcv)

derived_data <- as.data.frame(dimension_reduced$x[,1:4])
derived_data$code <- data$recoded

gam_model <- gam(code ~ s(PC1) + s(PC2) + s(PC3) + s(PC4), family = 'binomial', data = derived_data)

plot(gam_model)
```

Based on these outputs, it is strongly suggested that we transform the first four components. These transformation are large and will only serve to further complicate the model, but at this point we are already well beyond trying to make a "human" model, so we will continue trying to build the best model regardless of complexity from here on, assuming that a computer will be computing the probabilities.

I have chosen not to fit a model using interaction terms, as part of the benefit to principal components is that they are constructed to be orthogonal to each other, so by allowing them to interact, we reintroduce room for mutlicolinearity.

```r
terms <- c(
  paste0("I(PC1^", 8:1, ")"),
  paste0("I(PC2^", 7:1, ")"),
  paste0("I(PC3^", 5:1, ")")
)

formula_string <- paste("code ~", paste(terms, collapse = " + "))


formula_obj <- as.formula(formula_string)
dummy_model <- glm(code ~ poly(PC1, 8) + poly(PC2, 7) + poly(PC3, 5) + PC4, data = derived_data, family

library(MuMIn)

options(na.action = "na.fail")

dredged_model <- dredge(dummy_model)

chosen_model <- get.models(dredged_model, 2)[[1]]

summary(chosen_model)
```

```
##
## Call:
## glm(formula = code ~ PC4 + poly(PC1, 8) + poly(PC3, 5) + 1, family = binomial,
##     data = derived_data)
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -4.25907    0.97467  -4.370 1.24e-05 ***
## PC4              0.91788    0.08361  10.978  < 2e-16 ***
## poly(PC1, 8)1    6.36482   64.74438   0.098  0.92169
## poly(PC1, 8)2  110.64006  134.48145   0.823  0.41067
## poly(PC1, 8)3  -79.62292  179.61872  -0.443  0.65756
## poly(PC1, 8)4   37.95659  171.34032   0.222  0.82468
## poly(PC1, 8)5  -58.87212  138.27397  -0.426  0.67028
## poly(PC1, 8)6   41.80534   92.36646   0.453  0.65083
## poly(PC1, 8)7  -10.80579   48.05283  -0.225  0.82208
## poly(PC1, 8)8  -16.25628   17.71953  -0.917  0.35892
## poly(PC3, 5)1  218.05915   69.20639   3.151  0.00163 **
## poly(PC3, 5)2 -117.71628   66.89799  -1.760  0.07847 .
## poly(PC3, 5)3  116.20401   61.75159   1.882  0.05986 .
## poly(PC3, 5)4  -73.89421   37.40596  -1.975  0.04822 *
## poly(PC3, 5)5   33.67965   20.44234   1.648  0.09945 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2248.69  on 2125  degrees of freedom
## Residual deviance:  958.67  on 2111  degrees of freedom
## AIC: 988.67
##
```

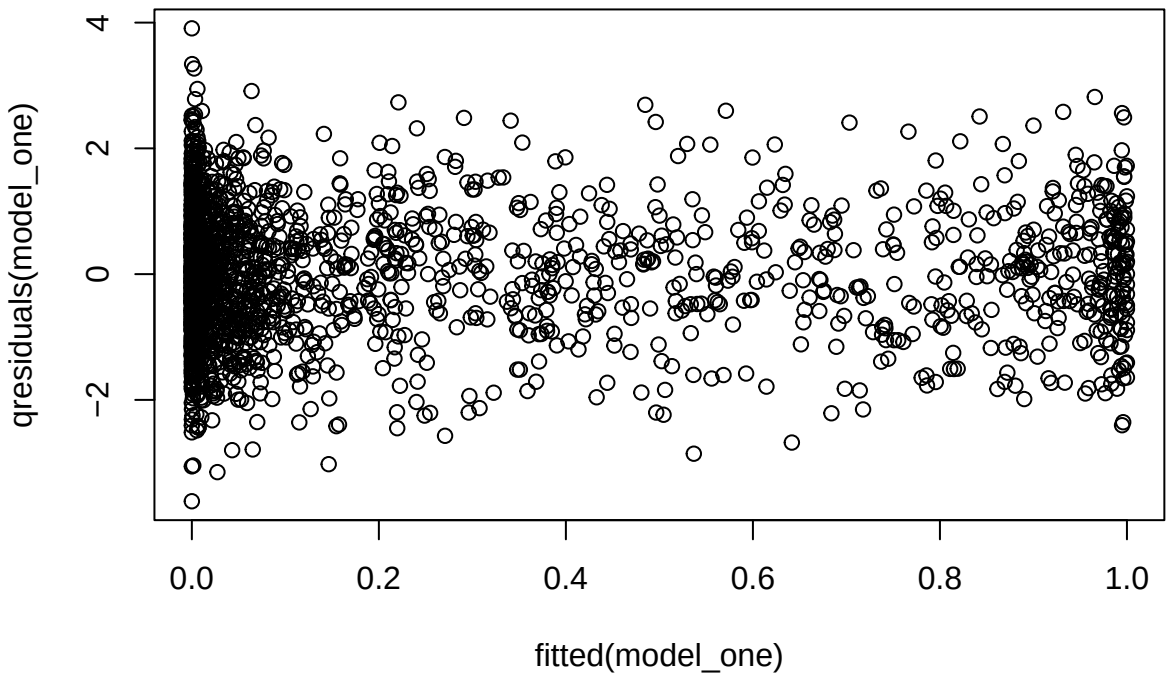
```
## Number of Fisher Scoring iterations: 12
```

```
# the dredging was a failure, attempting manually

model_one <- glm(code ~ poly(PC1, 7) + PC2 + poly(PC3, 4) + PC4, data = derived_data, family = binomial)

summary(model_one)
```

```
##
## Call:
## glm(formula = code ~ poly(PC1, 7) + PC2 + poly(PC3, 4) + PC4,
##     family = binomial, data = derived_data)
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -3.19972    0.47992  -6.667 2.61e-11 ***
## poly(PC1, 7)1  -47.90606   35.20082  -1.361 0.173534
## poly(PC1, 7)2  218.49640   73.97167   2.954 0.003139 **
## poly(PC1, 7)3 -226.45537   95.20119  -2.379 0.017374 *
## poly(PC1, 7)4  191.92973   85.42024   2.247 0.024647 *
## poly(PC1, 7)5 -175.67360   64.31325  -2.732 0.006304 **
## poly(PC1, 7)6  122.04619   38.25787   3.190 0.001422 **
## poly(PC1, 7)7  -60.57593   16.03426  -3.778 0.000158 ***
## PC2              0.37971    0.05334   7.118 1.09e-12 ***
## poly(PC3, 4)1  157.81203   31.13390   5.069 4.00e-07 ***
## poly(PC3, 4)2  -59.48459   29.11927  -2.043 0.041073 *
## poly(PC3, 4)3   50.91725   21.59023   2.358 0.018357 *
## poly(PC3, 4)4  -31.66344   13.48270  -2.348 0.018852 *
## PC4              1.00485    0.09026  11.132  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2248.69  on 2125  degrees of freedom
## Residual deviance:  907.47  on 2112  degrees of freedom
## AIC: 935.47
##
## Number of Fisher Scoring iterations: 11
```

Now that we have our model, we need to consider the validity of the model. Since the model is based on principal component regression, our data is sparse already, so many of the standard techniques for assessing model fit are not plausible. Instead, we will assess model fit using the randomised quantile residuals.

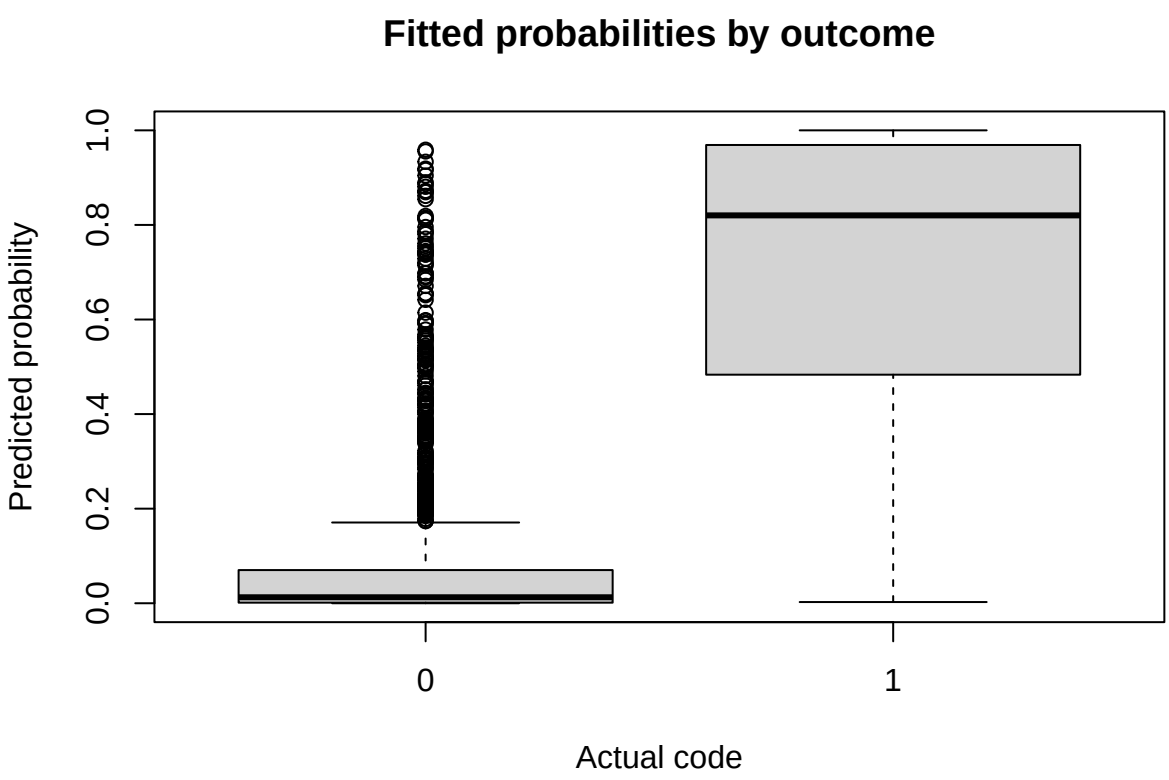
```
library(statmod)
plot(fitted(model_one), qresiduals(model_one))
```



The randomised quantile residuals appear to be randomly distributed through the range with constant variance about mean zero. This suggests that our model is a suitable fit.

We can also inspect the predicted probability of the fetus being at-risk, then compare this by their actual condition.

```
boxplot(fitted(model_one) ~ derived_data$code,
        main = "Fitted probabilities by outcome",
        xlab = "Actual code", ylab = "Predicted probability")
```



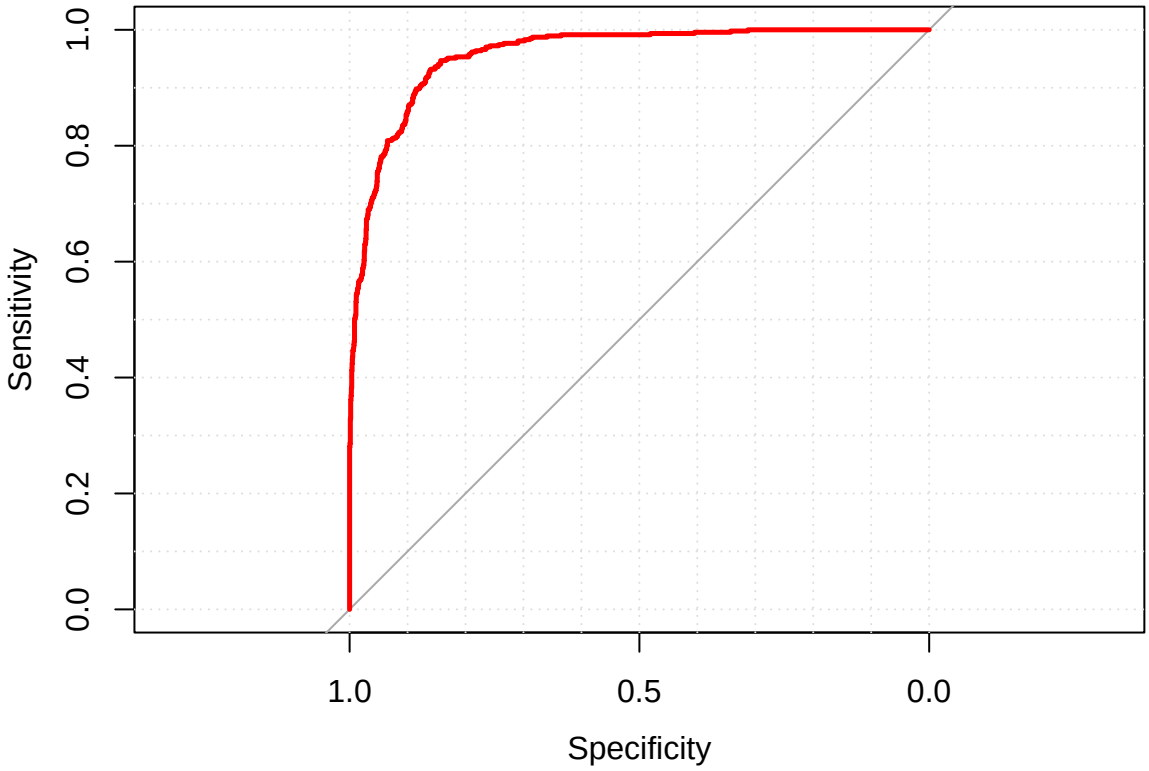
This is a somewhat concerning result. Although very clearly the interquartile ranges do not overlap, we can see that the lower quartile of the at-risk group overlaps completely with the healthy group, dropping well below the 50% predicted probability mark indicating a significant number of at-risk feti will be predicted as being totally healthy, which is obviously, a significant risk.

To handle this, we need to consider the model sensitivity and specificity, as well as how we can can optimise on this. This will be done using Receiver-Operating-Characteristic (ROC) curves and considering the Area-Under-Curve (AUC) for how we can improve these measures.

```
library(pROC)

fetal_roc <- roc(response = data$recoded, predictor = fitted.values(model_one))

coords(fetal_roc, "best", inut="threshold")
```

```
##   threshold specificity sensitivity
## 1  0.195188   0.8598187   0.9320594
```

```
plot(fetal_roc, col="red", grid=TRUE, lwd=2.5)
```



When we opt to maximise both sensitivity and specificity, we can attain 93.2% and 86.0% respectively. However, the cost of missing a positive case is much worse than that of a negative, so we will instead look to maximise the sensitivity, while trying to ensure the specificity doesn't drop too significantly.

```
chosen_threshold <- 1100

fetal_roc$sensitivities[chosen_threshold]
```

```
## [1] 0.9872611
```

```
fetal_roc$specificities[chosen_threshold]
```

```
## [1] 0.6652568
```

```
chosen_threshold <- 1200

fetal_roc$sensitivities[chosen_threshold]
```

```
## [1] 0.9766454
```

```
fetal_roc$specificities[chosen_threshold]
```

```
## [1] 0.7232628
```

We are able to identify two fairly suitable thresholds:

Threshold one - Sensitivity: 98.7% - Specificity: 66.5%

Threshold two - Sensitivity: 97.7% - Specificity: 72.1%

There is a clear trade off with these models, as to if the 1% increase in sensitivity is offset by the 6.4% drop in specificity. This is a decision ultimately to be left to clinicians. While we obviously want to successfully catch as many at-risk feti as possible, the question of what happens *after* this is significant. If further testing or possible treatment poses a risk to the fetus or the mother, we want to minimise the number of false positives, which means we want a higher specificity as well. Additionally, if the cost of further testing/treatment based on this model is high, then we may use hospital resources unnecessarily, creating significant levels of wastage in a crucial service.

This is only made more apparent when considering that in our sample, 0.22% of the 2126 feti are actually at-risk. This means that using threshold one would result in sending about 462 of the 471 at risk feti for further testing/treatment, but also sending about 554 healthy feti for further testing, which is more than the total number of at risk feti, much less than the number successfully identified.

So this model's use is up to the decision of clinicians, but it is somewhat disapointing the level of accuracy that it achieves.

Since this model is extremely complex, maybe we should consider a more simple model. Indeed, let us now answer a much simpler question - Is there a single variable model that can serve as a good "rule of thumb" in classification?

To answer this, we return to our PCA, where we observed that the third component seemed to be a fairly good discriminator. We can perform a varimax rotation to interpret this component more easily to identify which variables it loads most highly on.

```
new_scale <- dimension_reduced$rotation[, 1:5]%*%diag(dimension_reduced$sd[1:5])
class(new_scale)<-"loadings"
round(new_scale[,3], 3)
```

```
##                                          baseline value
##                                                   0.459
##                                           accelerations
##                                                  -0.252
##                                         fetal_movement
##                                                   0.137
##                                    uterine_contractions
##                                                  -0.163
##                                     light_decelerations
##                                                   0.271
##                                    severe_decelerations
##                                                   0.096
##                                 prolongued_decelerations
##                                                   0.455
##                          abnormal_short_term_variability
##                                                   0.664
##                        mean_value_of_short_term_variability
##                                                  -0.101
## percentage_of_time_with_abnormal_long_term_variability
##                                                   0.433
##                         mean_value_of_long_term_variability
##                                                  -0.584
##                                        histogram_width
```

```
##                                             0.086
##                                     histogram_min
##                                             0.039
##                                     histogram_max
##                                             0.252
##                             histogram_number_of_peaks
##                                             0.145
##                           histogram_number_of_zeroes
##                                            -0.005
##                                      histogram_mode
##                                             0.031
##                                     histogram_mean
##                                            -0.030
##                                    histogram_median
##                                             0.089
##                                  histogram_variance
##                                             0.359
##                                  histogram_tendency
##                                            -0.053
```

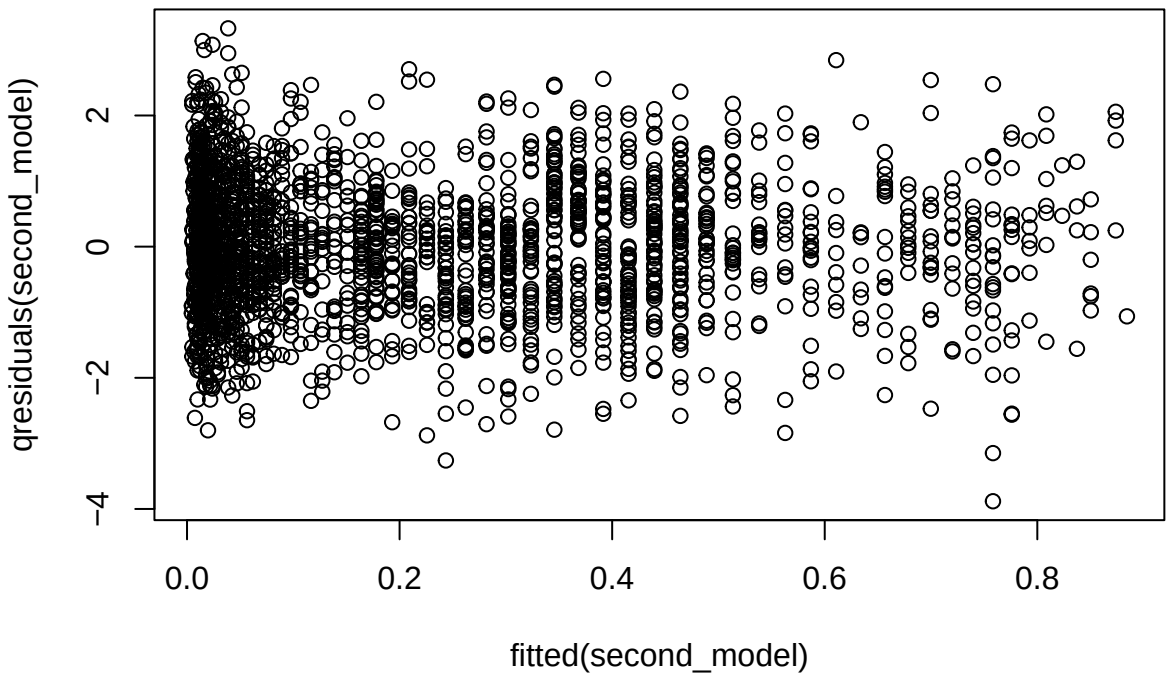
We can see rather clearly here that the highest loading, by a considerable margin, is abnormal short term variability. As such, let us fit a model using *only* this variable, and compare it's performance to our previous, more complicated model.

```
second_model <- glm(recoded ~ abnormal_short_term_variability, data = data, family = binomial)
summary(second_model)
```

```
##
## Call:
## glm(formula = recoded ~ abnormal_short_term_variability, family = binomial,
##     data = data)
##
## Coefficients:
##                                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    -6.577585   0.314157  -20.94   <2e-16 ***
## abnormal_short_term_variability  0.098986   0.005267   18.79   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2248.7  on 2125  degrees of freedom
## Residual deviance: 1640.6  on 2124  degrees of freedom
## AIC: 1644.6
##
## Number of Fisher Scoring iterations: 6
```

Once again, we check the residuals.

```
plot(fitted(second_model),  qresiduals(second_model))
```

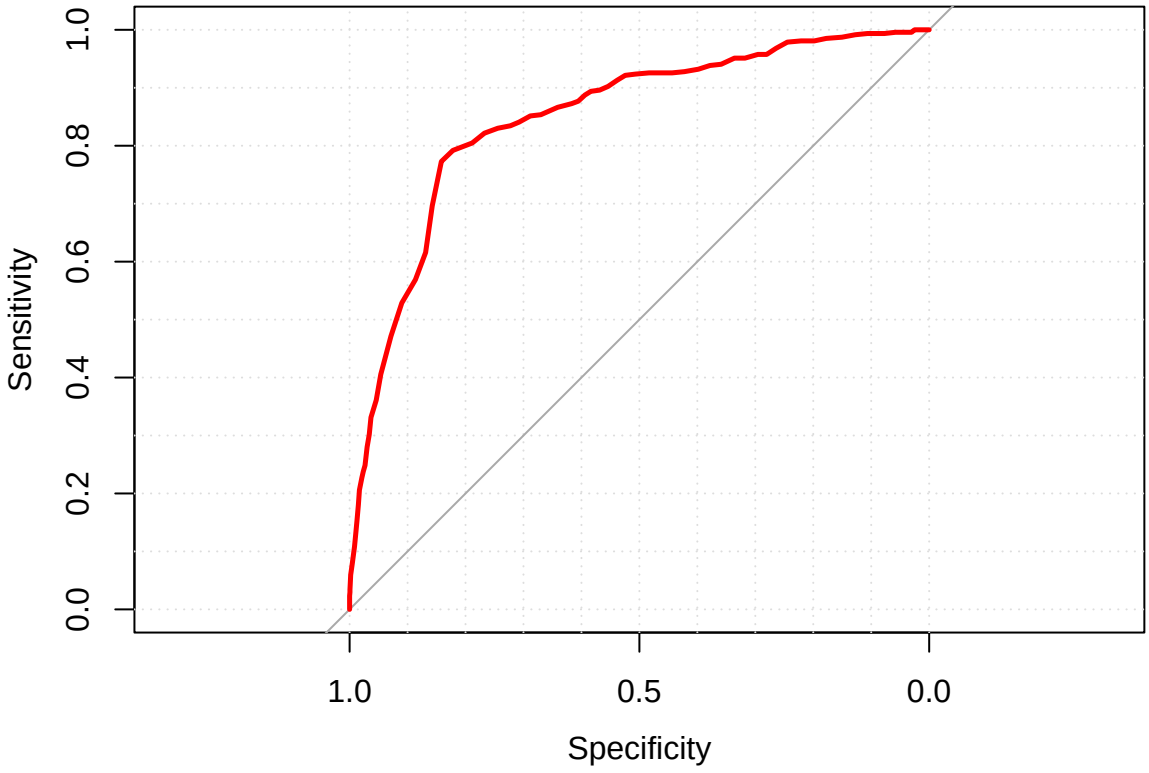


The residuals once again appear more random, with a lot more points clustered towards the bottom of the plot, which indicates the model predicts more of the feti to be healthy, which is consistent with what we know about our sample. We now consider the classification ability.

```
fetal_roc <- roc(response = data$recoded, predictor = fitted.values(second_model))

coords(fetal_roc, "best", inut="threshold")
```

```
##   threshold specificity sensitivity
## 1 0.3345881   0.8416918   0.7728238
```

```
plot(fetal_roc, col="red", grid=TRUE, lwd=2.5)
```



Off the bat, the initial optimum is far below that of the previous model. However, we once again look to optimise the sensitivity preferentially.

```
chosen_threshold <- 40

fetal_roc$sensitivities[chosen_threshold]
```

```
## [1] 0.866242
```

```
fetal_roc$specificities[chosen_threshold]
```

```
## [1] 0.6410876
```

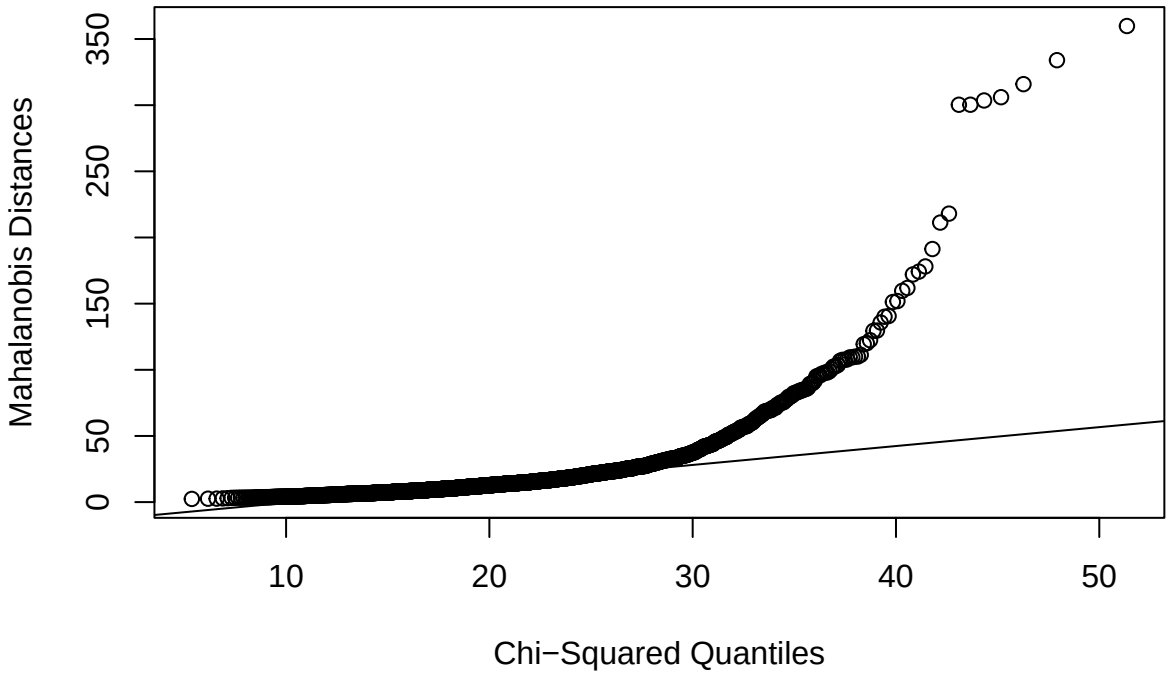
Checking for better thresholds only led to significantly worse results than the previous model. As a rule of thumb, this model could be used to correct spot positive cases about 86.6% of the time, and negative cases about 64.1% of the time. This is an average performance, as it would lead to at least elevated suspicions in the majority of at-risk feti, and the lower specificity is fine provided there are other low-cost low-risk tests/treatments that can be performed. However, I would not recommend using this model as a rigorous indicating of fetal health, as these classification rates are far too low for such a high stake field.

It appears as though the logistic regression methods are suitable in my non-clinical opinion, but the question is, can we achieve even better results? Part of the issue with this approach was that we had to collapse the pathological and suspect groups into a single group. Even if we don't necessarily improve the overall accuracy, it would be beneficial to see if we can achieve at least the same level of accuracy as the first logistic model across all three groups, instead of the two.

In order to solve this problem, the use of multivariate methods regarding discriminant analysis would be ideal. Before we can use LDA or QDA, we first need to check the assumptions of these parametric methods. This means we will need to check the equal covariance assumption for LDA, and multivariate normality assumption for both of them. If both of these fail, then we can still use QDA to an extent, or we can consider the use of more computationally intense, non-parametric methods such as PLS-DA. However, it is worth noting that in practice, almost all of these methods would involve a computer, and that depending on complexity, PLS-DA may be a much slower method.

We will now test both the MVN and equal covariance assumptions. I suspect MVN will be violated as many of these variables are related, especially those from the histogram.

```
# testing for multivariate normality
library(mvabund)
data <- read_csv("fetal_health.csv")
data_matrix <- as.matrix(data[,-22])

dimesion_manova <- manova(data_matrix ~ data$fetal_health)

resid_mat <- residuals(dimesion_manova)
Sigma <- cov(resid_mat)

maha <- mahalanobis(resid_mat, center = colMeans(resid_mat), cov = Sigma)

qqplot(qchisq(ppoints(nrow(data)), df = ncol(resid_mat)), y = maha,
       xlab = "Chi-Squared Quantiles", ylab = "Mahalanobis Distances")
qqline(maha, distribution = function(p) qchisq(p, df = ncol(resid_mat)))
```



```r
library(MVN)
MVN::mvn(data = resid_mat, mvnTest = "hz")$multivariateNormality
```

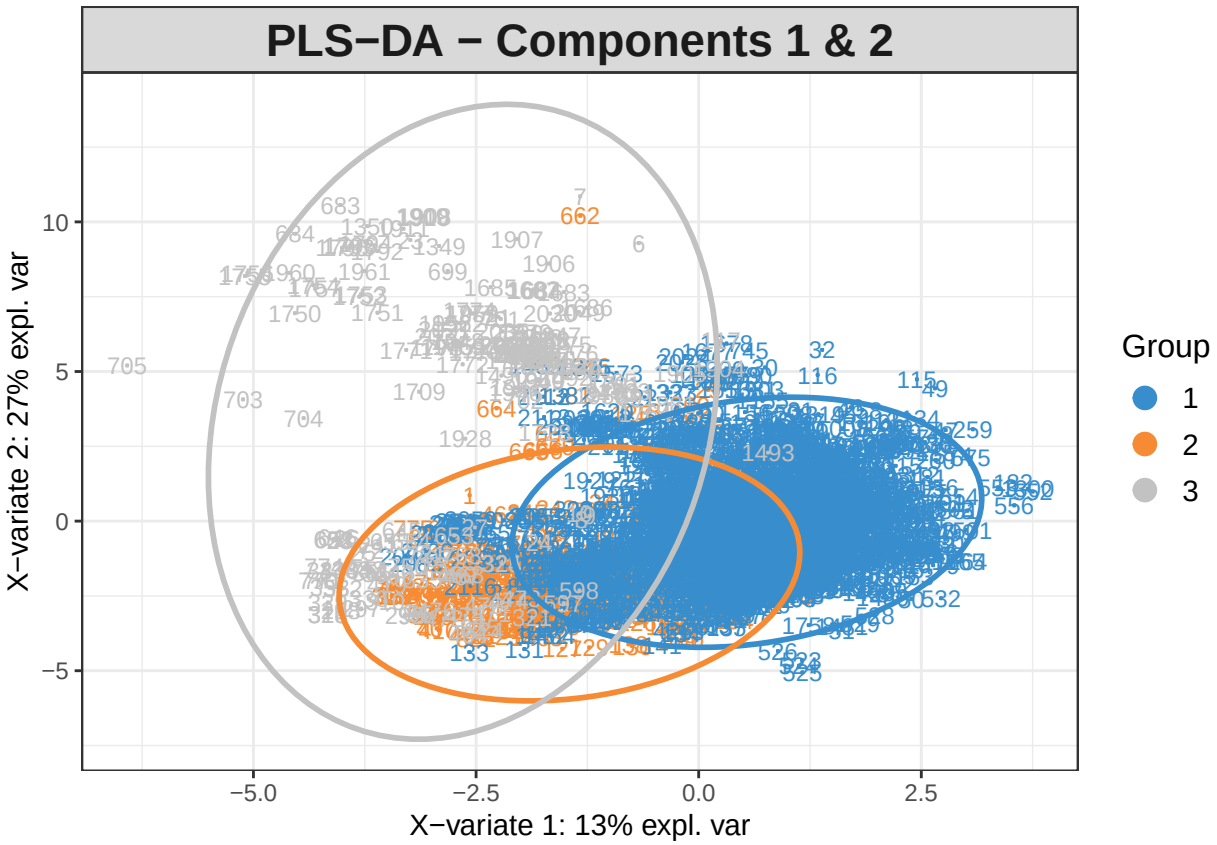
```
##            Test   HZ p value MVN
## 1 Henze-Zirkler 8504       0  NO
```

```r
# testing for equal covariance
abs_resids <- abs(dimesion_manova$resid)

maonva_permutations <- manylm(abs_resids ~ data$fetal_health, cor.type = "R", test = "F")
anova(maonva_permutations, resamp = 'perm.resid')
```

```
## Analysis of Variance Table
##
## Model: manylm(formula = abs_resids ~ data$fetal_health, test = "F",
## Model:     cor.type = "R")
##
## Overall test for all response variables
## Test statistics:
##                   Res.Df Df.diff val(F) Pr(>F)
## (Intercept)         2125
## data$fetal_health   2124       1   4837  0.002 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Arguments:
##  Test statistics calculated assuming unconstrained correlation response
##  P-value calculated using 999 iterations via residual (without replacement) resampling.
```

As we can see, both the equal covariance and multivariate normality assumptions failed. Although this typically means QDA would still yield better/more reliable results than LDA, we will skip over these parametric methods in favour of the non-parametric approach, using PLS-DA. We will attempt this in two forms. One will involve all three original groups, and the other our merged groups to compare it to the logisitic regression method.

```r
library(mixOmics)

X <- data_matrix
Y <- data$fetal_health

plsda_model <- mixOmics::plsda(X, Y, ncomp=2)

plotIndiv(plsda_model,
          comp = c(1, 2),
          group = Y,
          ellipse = TRUE,
          legend = TRUE,
          legend.title = "Group",
          title = "PLS-DA - Components 1 & 2")
```



As we can see, there is significant overlap of all groups, but especially with the second group - suspect - being almost completely engulfed in the other groups. We also do not encapsulate much of the variation of the original data.

However, instead of a visual inspection, we would prefer a robust analytical method, namely using a cross-validation approach.

```
library(caret)

train_index <- createDataPartition(data$fetal_health, p = 0.8, list = FALSE)

train_data <- data[train_index, ]
test_data  <- data[-train_index, ]

plsda_model <- mixOmics::plsda(train_data[-22], train_data$fetal_health, ncomp = 2)

plsda_pred <- predict(plsda_model, newdata = test_data[-22], dist = "max.dist")
pred_classes <- plsda_pred$class$max.dist[, 2]
actual_classes <- test_data$fetal_health
conf_matrix <- confusionMatrix(factor(pred_classes), factor(actual_classes))
conf_matrix
```

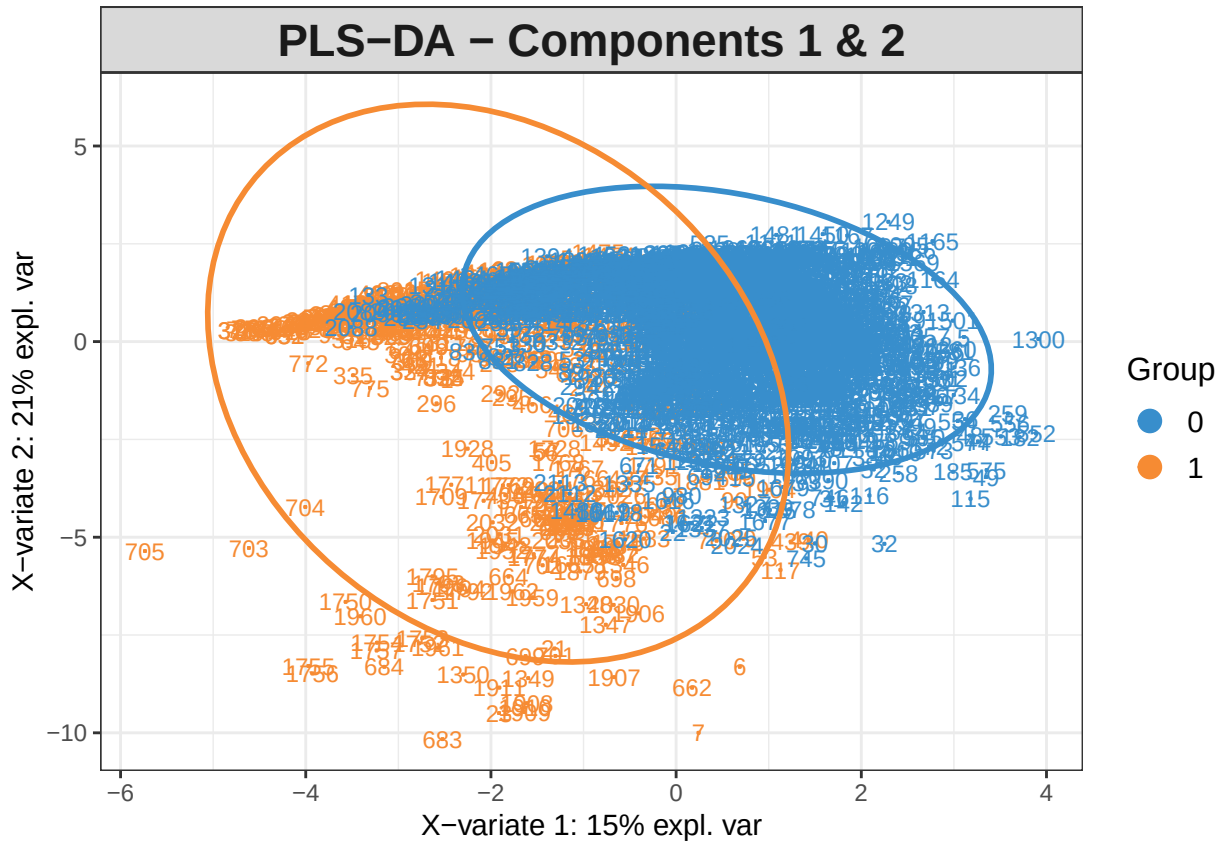
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3
```

```
##          1 316  42   6
##          2   7  14  18
##          3   0   1  21
##
## Overall Statistics
##
##                Accuracy : 0.8259
##                  95% CI : (0.7864, 0.8607)
##     No Information Rate : 0.76
##     P-Value [Acc > NIR] : 0.0006316
##
##                   Kappa : 0.4744
##
##  Mcnemar's Test P-Value : 5.116e-10
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.9783  0.24561  0.46667
## Specificity            0.5294  0.93207  0.99737
## Pos Pred Value         0.8681  0.35897  0.95455
## Neg Pred Value         0.8852  0.88860  0.94045
## Prevalence             0.7600  0.13412  0.10588
## Detection Rate         0.7435  0.03294  0.04941
## Detection Prevalence   0.8565  0.09176  0.05176
## Balanced Accuracy      0.7539  0.58884  0.73202
```

This model achieved an overall 85.7% accuracy rate. In absolute terms, it is very good to see that of all of the critical cases, only four were missed. However, this does represent nearly a fifth of all critical cases, which is *very* bad.

We now consider the two group solution from the same collapsed "at-risk" group used in the logistic regression solution.

```
data <- data %>%
  mutate(
    recoded = ifelse(fetal_health == 1, 0, 1)
  ) %>%
  dplyr::select(-fetal_health)

data_matrix <- as.matrix(data[,-22])

X <- data_matrix
Y <- data$recoded

plsda_model_two <- mixOmics::plsda(X, Y, ncomp=2)

plotIndiv(plsda_model_two,
          comp = c(1, 2),
          group = Y,
          ellipse = TRUE,
          legend = TRUE,
          legend.title = "Group",
          title = "PLS-DA - Components 1 & 2")
```



As we can see, there is still a very large level of overlap in these groups. We now check the classification ability, same as we did for the previous model.

```
train_index <- createDataPartition(data$recoded, p = 0.8, list = FALSE)

train_data <- data[train_index, ]
test_data  <- data[-train_index, ]

plsda_model_two <- mixOmics::plsda(train_data[-22], train_data$recoded, ncomp = 2)

plsda_pred <- predict(plsda_model_two, newdata = test_data[-22], dist = "max.dist")
pred_classes <- plsda_pred$class$max.dist[, 2]
actual_classes <- test_data$recoded
conf_matrix <- confusionMatrix(factor(pred_classes), factor(actual_classes))
conf_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 318  35
##          1  19  53
##
##                Accuracy : 0.8729
##                  95% CI : (0.8375, 0.9031)
##     No Information Rate : 0.7929
```

```
##     P-Value [Acc > NIR] : 1.172e-05
##
##                   Kappa : 0.5852
##
##  Mcnemar's Test P-Value : 0.04123
##
##             Sensitivity : 0.9436
##             Specificity : 0.6023
##          Pos Pred Value : 0.9008
##          Neg Pred Value : 0.7361
##              Prevalence : 0.7929
##          Detection Rate : 0.7482
##    Detection Prevalence : 0.8306
##       Balanced Accuracy : 0.7729
##
##        'Positive' Class : 0
##
```

There is an improvement in the accuracy model, but the classification rate for the at-risk group is scant higher than the critical group, which suggests that in exchange for less granularity in classification, we are hardly improving prediction ability. The confidence intervals of these models' accuracy also overlaps, indicating that we cannot claim one is definitively better at classifying, but as they work on different groups, this is not a fair comparison to begin with.

Note that in both of these models, due to how the data is coded the sensitivity actually refers to the successful classification rate of negative cases - healthy feti, while specificity refers to the classification rate of positive cases. In both cases, we can see that our models do a much better job at classifying the healthy feti than those that are at-risk, suspect or pathological.

Overall, I would not say that this model is an improvement over the logistic regression methods, especially as it performs much better at classifying the healthy feti than those actually at risk.

At this stage, we should try at least one more idea. I considered trying QDA, but given the assumptions are not satisfied, I have instead decided to try a Naive Bayes Classifier.

```
library(e1071)
library(caTools)

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

set.seed(8904)
split <- sample.split(data$fetal_health, SplitRatio = 0.8)
training_data <- subset(data, split == TRUE)
testing_data <- subset(data, split == FALSE)

scaled_training <- scale(training_data)
scaled_testing <- scale(testing_data)

nb_classifier <- naiveBayes(fetal_health ~ ., data = training_data)

nb_preds <- predict(nb_classifier, newdata = testing_data)

cm <- table(testing_data$fetal_health, nb_preds)
confusionMatrix(cm)
```

```
## Confusion Matrix and Statistics
```

```
##
##    nb_preds
##       1   2   3
##   1 277  33  21
##   2   5  48   6
##   3   0  14  21
##
## Overall Statistics
##
##                Accuracy : 0.8141
##                  95% CI : (0.7738, 0.85)
##     No Information Rate : 0.6635
##     P-Value [Acc > NIR] : 3.765e-12
##
##                   Kappa : 0.5803
##
##  Mcnemar's Test P-Value : 1.005e-09
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.9823   0.5053  0.43750
## Specificity            0.6224   0.9667  0.96286
## Pos Pred Value         0.8369   0.8136  0.60000
## Neg Pred Value         0.9468   0.8716  0.93077
## Prevalence             0.6635   0.2235  0.11294
## Detection Rate         0.6518   0.1129  0.04941
## Detection Prevalence   0.7788   0.1388  0.08235
## Balanced Accuracy      0.8023   0.7360  0.70018
```

The Naive Bayes has the worst accuracy of all of the models, although the confidence interval overlaps with the first PLS-DA model, which is a fair comparison as both operate on classification of all three groups. This means that we unfortunately cannot claim that either of these models is better than the other.

Conclusion

None of these models perform to the rigorous standards that healthcare is expected to maintain and as such I cannot recommend their use in any capacity beyond, perhaps, general rules of thumb.

This is however, a pointless recommendation as they are highly computationally complex models that require a large amount of data to operate, except for the second logistic regression model.

The best performing models were the logistic regression models, but these come with the downside of only being able to class feti as being at-risk or healthy; they do not provide any further level of detail, which the other models do.

This study has served as an excellent exercise into the use of multivariate methods, logistic regression, and statistical computing methods. The insights shed on obstetric care from this are extremely valuable, and demonstrate that the quality of care and human insights and intuition that health professionals have is not easily replicated with computers.