STAT432 - Final Project - Fetal Health Classification

Justin Collins (jtc3) , Matt Nicholson (mn12) and Anoushka Berry (ab18)

Project Description and Summary

Our Project is the analysis of the fetal health classification using unsupervised learning and multi-class classification. We will be using k-means, hierarchical clustering and principal component analysis (PCA) to visualize the data and then classifying using RandomForest and SVM. Our goal is to find those variables in the dataset that most affect the fetal_health dependant variable which is the result of the cardiotocogram and one of the three categories of normal, suspect or pathalogical indicating the level of danger of the fetus. We want to determine wether this information correctly predicts the safety levels of the fetus.

Literature Review:

We read 2 relevant papers, one of which was published in 2012 in the Journal of Clinical & Experimental Investigations, titled *The success of cardiotocography in predicting perinatal outcome* which conducted a study to understand fetal health during labor to minimize fetal death due to asphyxia and the neurological sequelae of fetal hypoxia. This study enrolled 101 full-term pregnant women admitted for delivery between October 2009 and February 2010. Women were included if they were aged 18-45 years and within 36-41 weeks of gestation. Women whose ages or gestation status fell outside these ranges, those with a systemic disease, and those who were regular drug users were excluded. The fetus' health is evaluated, in part, by assessing the fetal heart rate (FHR), which is also a part of our dataset under baseline value. Results showed that while the CTG was an important test during labor for labor management, it is insufficient for predicting the perinatal outcome.

A second relevant paper related to cardiotocography and fetal death would be one published in 2021 by Springer Nature Singapore Pte Ltd, called *Cardiotocogram Data Classification Using Random Forest Based Machine Learning Algorithm* where they implement a model based on CTG training data utilizing a supervised Random Forest (RF) model. In their introduction, they talk about cardiotocography as a strategy that is utilized to screen fetal health condition during pregnancy and that it can be utilized to examine the fetus' health condition, normoxia and normal or abnormal fetus acid base status. For the classification model the dataset was split into 80% train and 20% test sets. The data set comprises of 2126 cardiotocograms described by 22 attributes. Each observation in the dataset is labeled with one the three classes, Normal (N), Suspicious (S) and Pathological (P). To classify these three classes they used 10-folded cross validation on a random forest model and the results they found shows good accuracy with different randomly selected predictors. The proposed technique achieves the classification accuracy of 94.8%. Looking at these results, we can see that there is sufficient evidence to suggest that that the CTG can be utilized to examine the fetus' health condition (between normal vs suspicious cs pathological), normoxia and normal or abnormal fetus acid base status, therefore, labor should be evaluated on an individualized basis.

From the first paper, we took into account their dislike for the cardiotocogram and looked into reducing the variables we worked with so that we could get a better and more compact model fit with higher accuracy. From the second paper, we used the RandomForest method for multi-class classification.

Works Cited

Kaban, Alpaslan, et al. "The Success of Cardiotocography in Predicting Perinatal Outcome." Journal of Clinical & Experimental Investigations, vol. 3, no. 2, June 2012, pp. 168–171. EBSCOhost, doi:10.5799/ahinjs.01.2012.02.0137.

Imran Molla M.M., Jui J.J., Bari B.S., Rashid M., Hasan M.J. (2021) Cardiotocogram Data Classification Using Random Forest Based Machine Learning Algorithm. In: Md Zain Z. et al. (eds) Proceedings of the 11th National Technical Seminar on Unmanned System Technology 2019. Lecture Notes in Electrical Engineering, vol 666. Springer, Singapore. https://doi-org.proxy2.library.illinois.edu/10.1007/978-981-15-5281-6_25

Summary Statistics

The dataset is a collection of information about 2126 Cardiotocogram exams (which represent the health of the fetus), which were then classified by three expert obstetritians into 3 classes: Normal, Suspect and Pathological.

```r
library(randomForest)
```

```
## randomForest 4.6-14
```

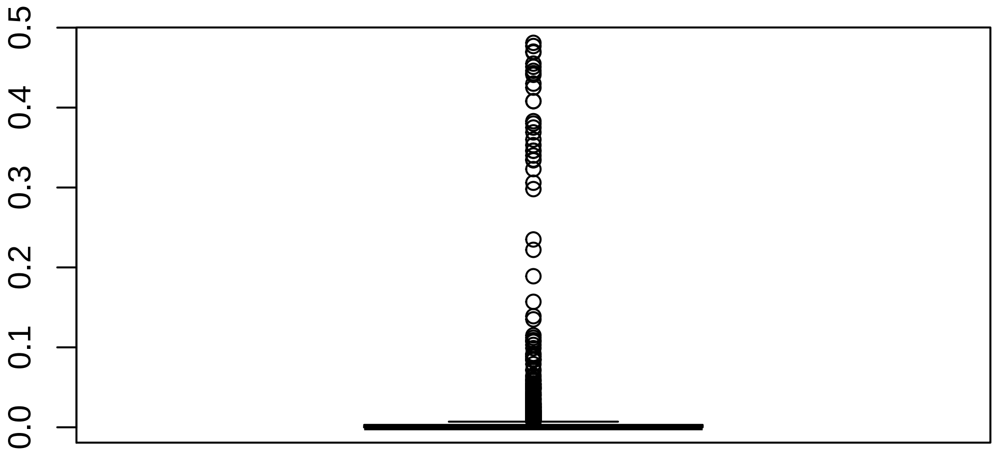
```
## Type rfNews() to see new features/changes/bug fixes.
```

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

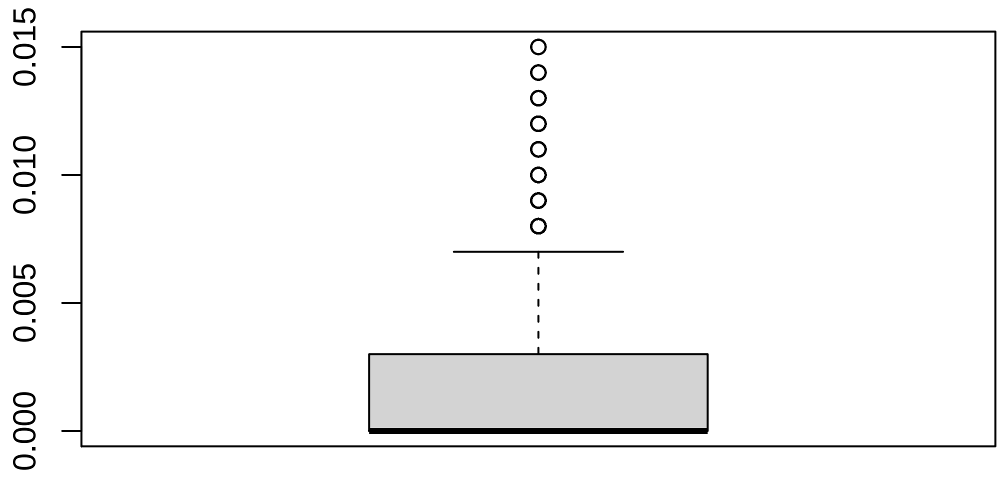
Upon further observation of our variables, we noticed that there are no categorical variables in our data, all of the variables have continuous distributions. Those that have a relatively normal distribution (the mean is relatively close to the median) would be baseline.value, accelerations, uterine_contractions, abnormal_short_term_variability and mean_value_of_short_term_variability. We also found two variables that had very skewed distributions and a lot of outliers. These two variables are fetal_movement and light_decelerations.

We saw some outliers using box plots and decided to not use that variable in our analysis. Running the boxplot for the entire dataset, we see that some variables have a lot out outliers. We see that for example, `fetal_movement` has a lot of outliers.

```r
boxplot(fetal_health$fetal_movement)
```



```r
boxplot(fetal_health$light_decelerations)
```



```r
# Remove unwanted variable
fetal_health <- fetal_health[,-c(3, 5, 12:21)]
set.seed(20)
#Randomly selects 75% data, then sorts them in accending order.
RandomData <- sort( sample(nrow(fetal_health), nrow(fetal_health) * .75))
training <- fetal_health[RandomData,]
testing <- fetal_health[-RandomData,]
fetal_stats <- matrix(0, nrow = 2, ncol=10)
colnames(fetal_stats) <- colnames(fetal_health)
rownames(fetal_stats) <- c("Mean", "Median")
 for (i in 1:10)
 {
  fetal_stats[1, i] <- colMeans(fetal_health)[i]

  fetal_stats[2, i] <- apply(fetal_health,2,median)[i]

  }
```

Unsupervised Learning

Hierarchical clustering

Hierarchical clustering, also known as hierarchical cluster analysis, is an algorithm that groups similar objects into groups called clusters. The endpoint is a set of clusters, where each cluster is distinct from each other cluster, and the objects within each cluster are broadly similar to each other. The clustering results in a dendogram displaying natural hierarchy. A hierarchal representation contains, at the lowest level, each cluster that contains a single observation and at the highest level, a single cluster that contains all the observations.

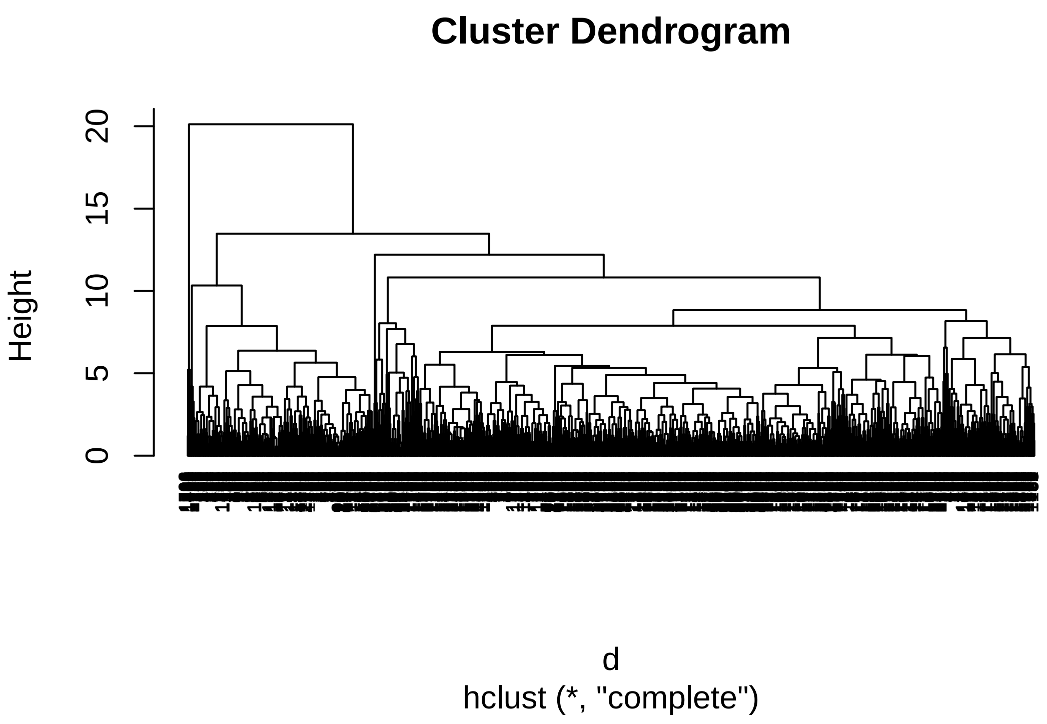
```r
library(cluster)
library(factoextra)
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
## The following object is masked from 'package:randomForest':
##
##     margin
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

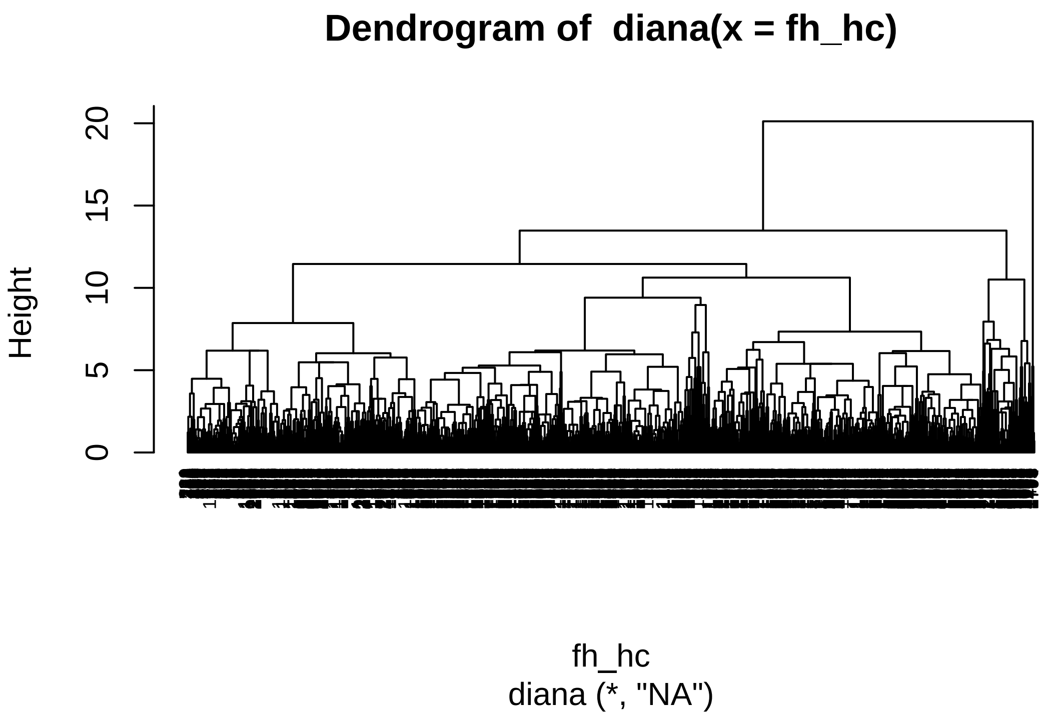
```
fh_hc <- fetal_health
# Sclaes variables, except the outcome variable.
for (i in 1:10)
{
fh_hc[,i] <- scale(fh_hc)[,i]
}
d <- dist(fh_hc, method = "euclidean")
hc1 <- hclust(d, method = "complete")
plot(hc1, cex = 0.6, hang = -1)
```



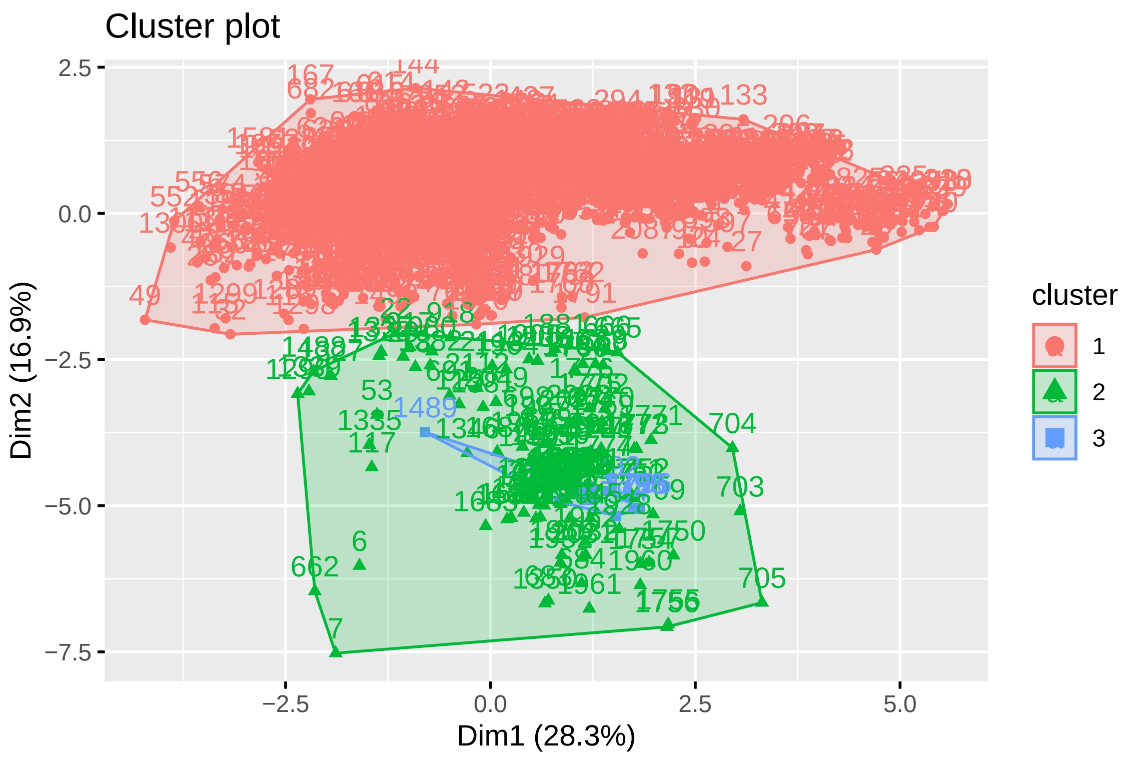
```
hc2 <- diana(fh_hc)
hc2$dc
```

```
## [1] 0.9642396
```

```
pltree(hc2, cex = 0.6, hang = -1,)
```



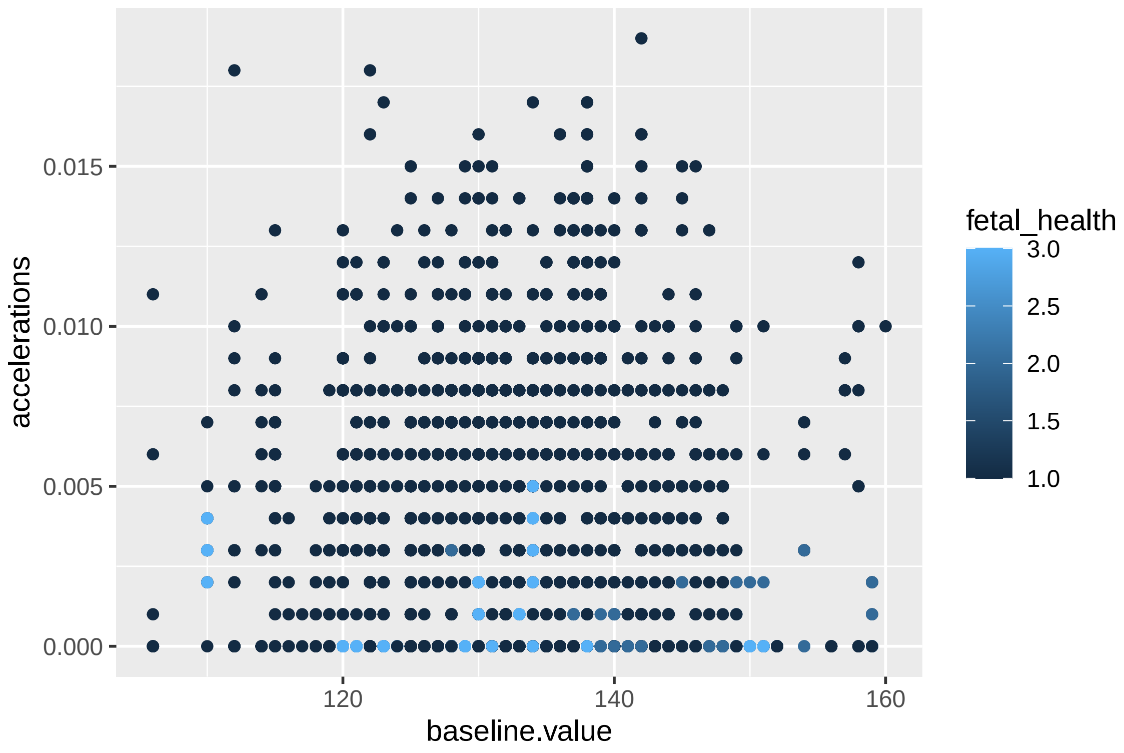
```
fviz_cluster(list(data = fh_hc, cluster = cutree(hc2, k = 3)))
```



K-means clustering:

K-means clustering is an unsupervised machine learning algorithm for partitioning a given data set into a set of k cluster. It classifies objects in cluster, such that objects within the same cluster are as similar as possible, whereas objects from different clusters are as dissimilar as possible. In k-means clustering, each cluster is represented by its center (i.e, centroid) which corresponds to the mean of points assigned to the cluster. We used k-means clustering on the data set, and clustered in respect to the baseline value and accelerations variables. We took 3 clusters, and separated them by the colors green, red and blue.

```
library(ggplot2)
ggplot(fetal_health, aes(baseline.value,accelerations))+geom_point(aes(color=fetal_health))
```



```
set.seed(1)
    # k mean clustering
fetal.kmean <- kmeans(fetal_health[, 1:2], centers = 3, nstart = 20)

    # the center of each class
fetal.kmean$centers
```

```
##   baseline.value accelerations
## 1       145.3378   0.002284281
## 2       120.8303   0.003029520
## 3       132.8621   0.003802231
```

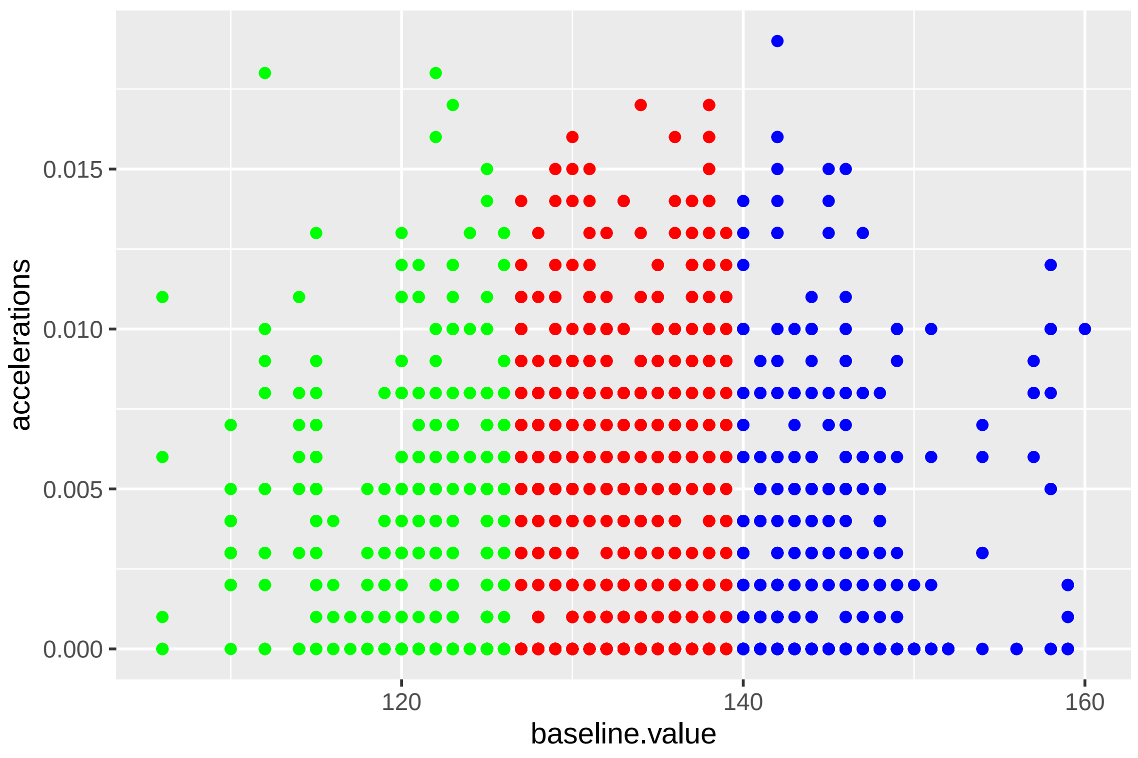
```
fetal.kmean$withinss
```

```
## [1] 12135.77 10156.39 12375.26
```

```
fetal.kmean$betweenss
```

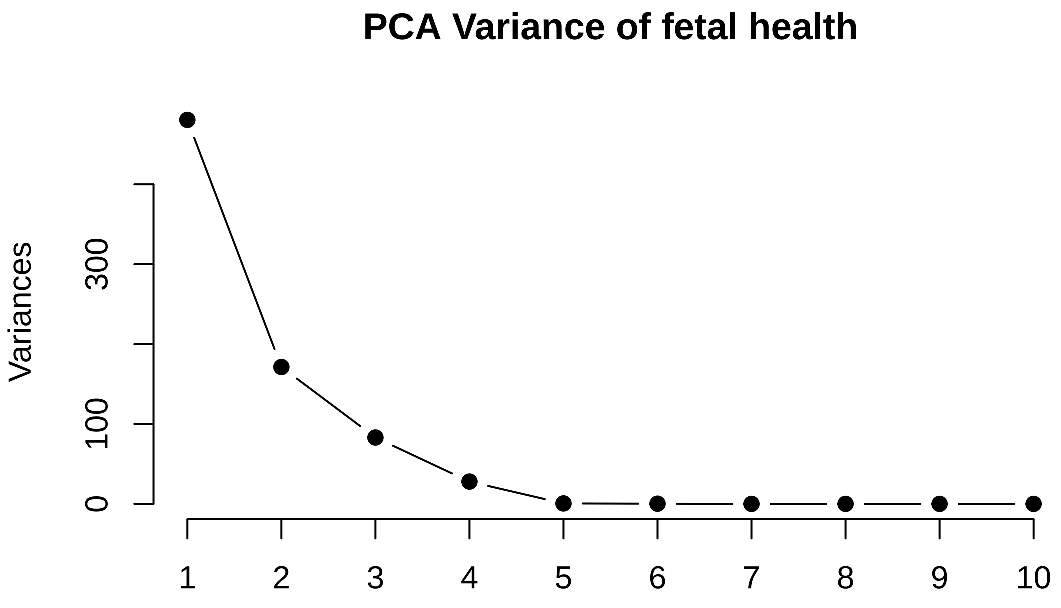
```
## [1] 171122.3
```

```r
fetal.kmean$cluster <- as.factor(fetal.kmean$cluster)
ggplot(fetal_health, aes(baseline.value, accelerations, color = fetal_health)) +
  geom_point(col = c("blue", "green", "red")[fetal.kmean$cluster])
```



PCA

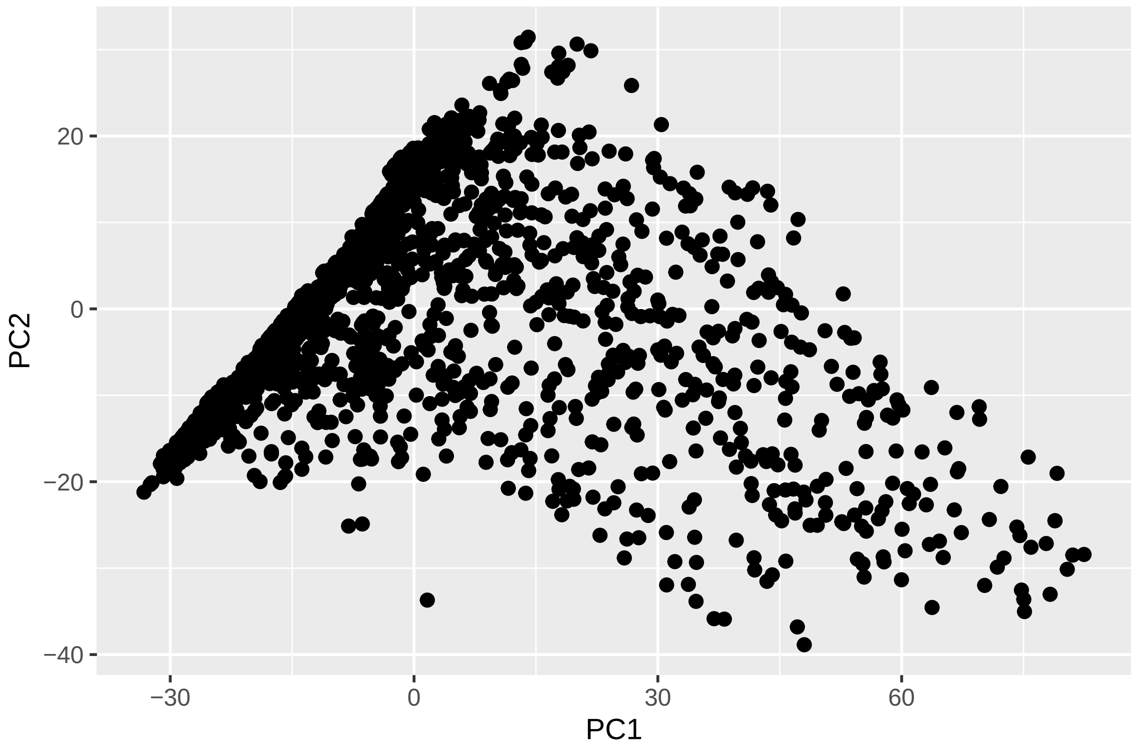
```r
zip_pc <- prcomp(fetal_health)
plot(zip_pc, type = "l", pch = 19, main = "PCA Variance of fetal health")
```



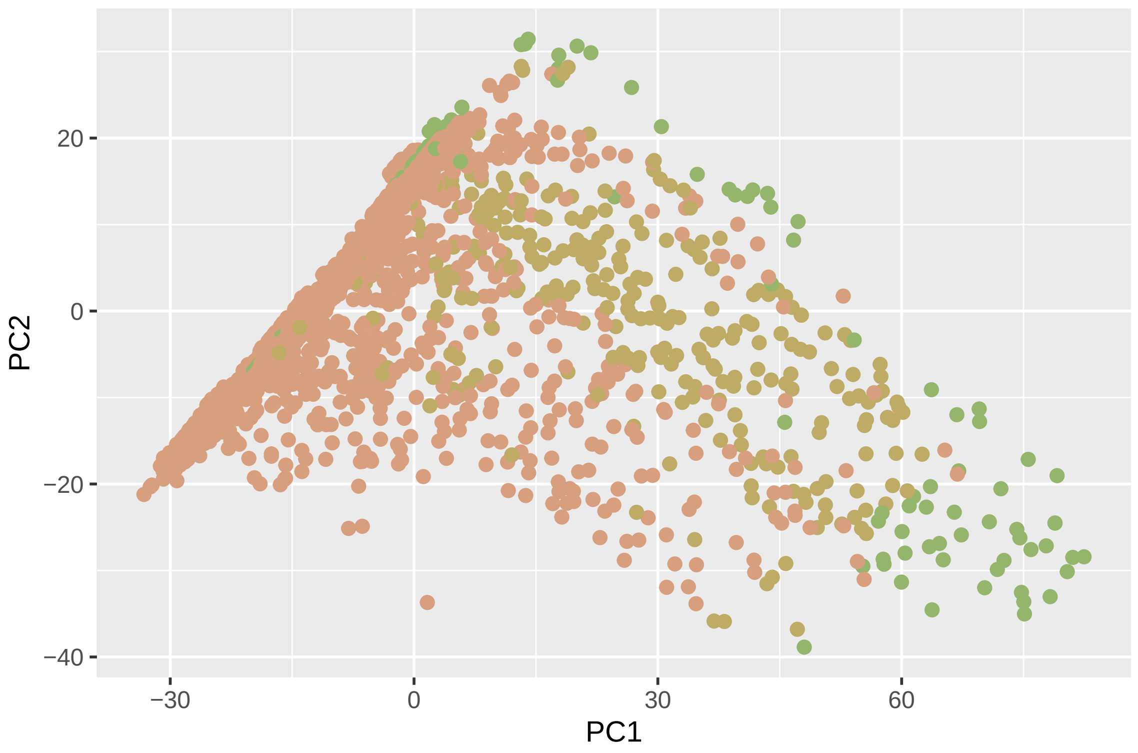
In the results above, we can see that the first four components make up almost 90% of our data. The scree plot shows us that the eigenvalues start to form a straight line after the fourth component. Therefore, it would be fine to say that 4 components is the adequate amount to have in order to have a large amount of variation explained by the data.

```r
zip.sub.truth = as.factor(fetal_health[fetal_health[,10], 1])
```

```r
ggplot(data = data.frame(zip_pc$x), aes(x=PC1, y=PC2)) + geom_point(size = 2)
```



```r
library(colorspace)
ggplot(data = data.frame(prcomp(fetal_health)$x), aes(x=PC1, y=PC2)) +
geom_point(color = rainbow_hcl(10)[fetal_health[,10]+1], size = 2)
```



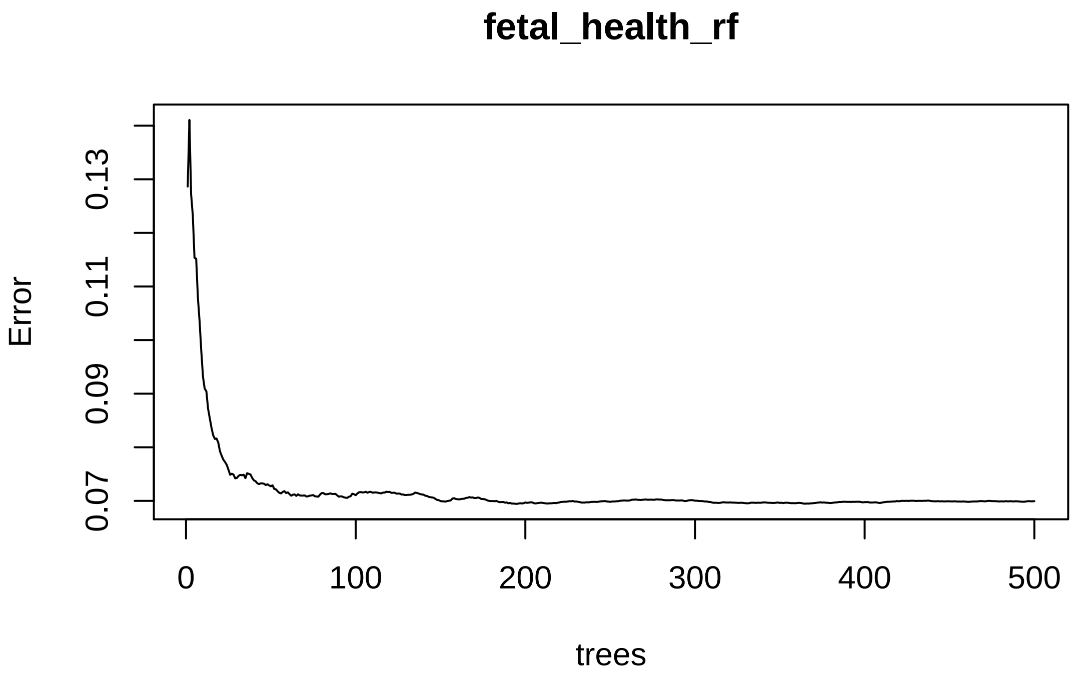
Multi-class classification

Random Forest

```
fetal_health_rf <- randomForest(fetal_health ~. ,data=training, tree=500, mtry = 5, proximity=TRUE)
rf_predictionTable <- table(predict(fetal_health_rf),training$fetal_health)
print(fetal_health_rf)
```

```
##
## Call:
##  randomForest(formula = fetal_health ~ ., data = training, tree = 500,      mtry = 5, proximity = TRU
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##           Mean of squared residuals: 0.06994955
##                     % Var explained: 81.4
```

```r
plot(fetal_health_rf)
```

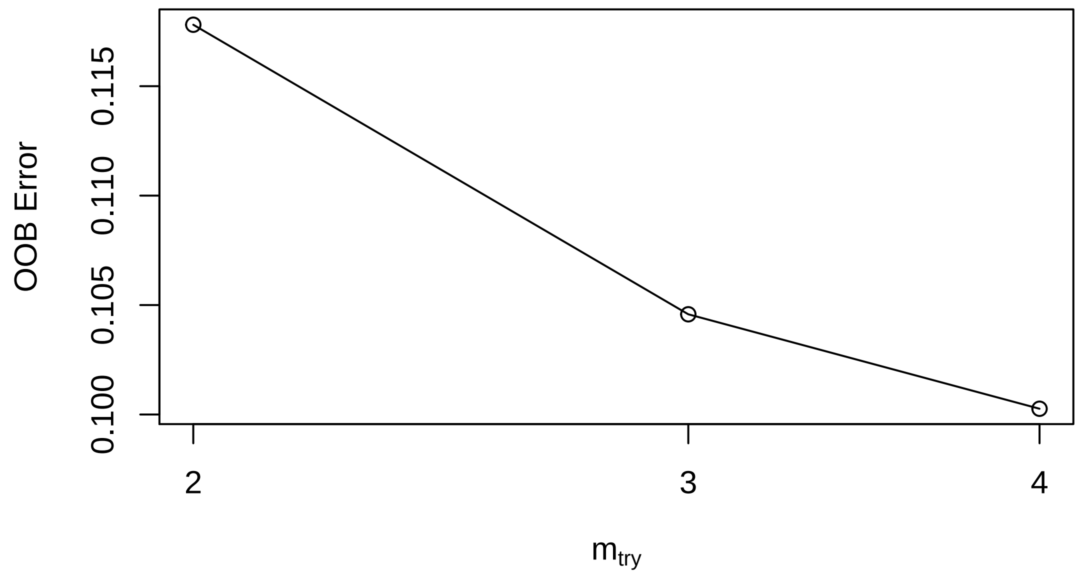


```r
rfpredict <- predict(fetal_health_rf, newdata= testing)
#Rounding values for fetal health. Cutoff is at .5
for(i in 1:nrow(testing))
{
        if(rfpredict[i] < 1.5)
        {
            rfpredict[i] <- 1
        }
        if(rfpredict[i] >= 1.5 & rfpredict[i] < 2.5)
        {
            rfpredict[i] <- 2
        }
        if(rfpredict[i] >= 2.5)
        {
            rfpredict[i] <- 3
        }
}
table(rfpredict, testing$fetal_health)
```

```
##
## rfpredict   1   2   3
##         1 401  13   1
##         2  14  59   7
##         3   0   0  37
```

```r
tune.rf <- tuneRF(testing[, -10], testing[, 10], stepFactor= 1.5)
```

```
## mtry = 3  OOB error = 0.1045791
## Searching left ...
## mtry = 2     OOB error = 0.117807
## -0.1264877 0.05
## Searching right ...
## mtry = 4     OOB error = 0.1002635
## 0.04126636 0.05
```



```
print(tune.rf)
```

```
##   mtry  OOBError
## 2    2 0.1178070
## 3    3 0.1045791
## 4    4 0.1002635
```

SVM

```
library(e1071)
svm <- svm(factor(training$fetal_health) ~ ., data = training, method = "C-classification", kernal = "ra
summary(svm)
```

```
##
## Call:
## svm(formula = factor(training$fetal_health) ~ ., data = training,
##     method = "C-classification", kernal = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  443
```

```
##
##  ( 207 74 162 )
##
##
## Number of Classes:  3
##
## Levels:
##  1 2 3
```

```
#head(svm$SV, 3)
svm_prediction <- predict(svm, testing, decision.values = TRUE)
table(testing$fetal_health, svm_prediction)
```

```
##    svm_prediction
##       1   2   3
##   1 404  10   1
##   2  25  45   2
##   3   6   6  33
```

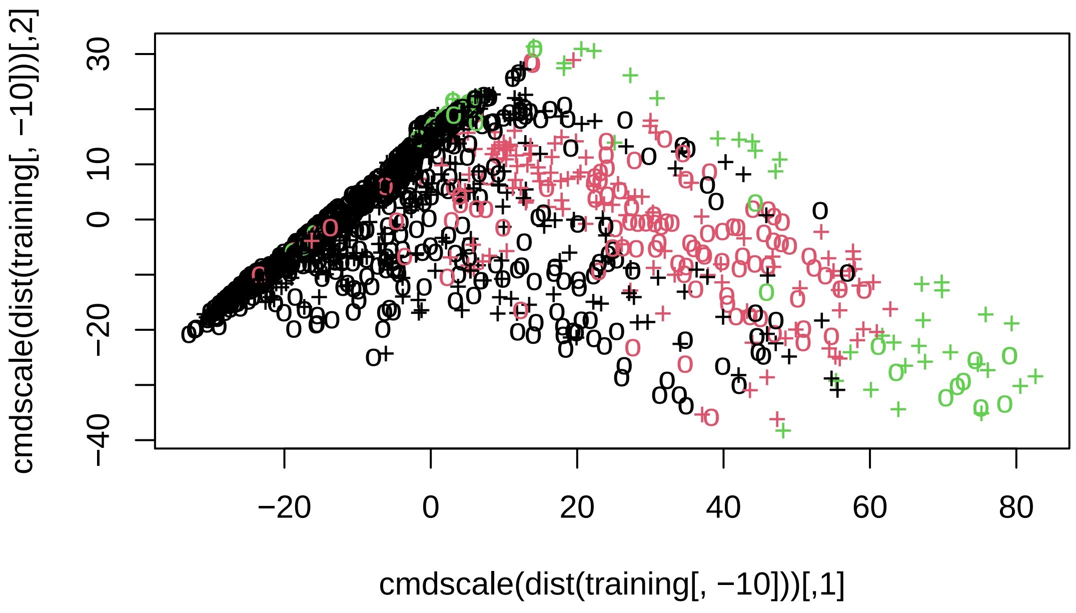
```
attr(svm_prediction, "decision.values")[1:4,]
```

```
##           1/3         1/2        3/2
## 1  -0.1550722 -0.43933045 -0.2972553
## 4   1.2128428  1.09252637 -0.1962091
## 9  -0.2841793  0.01779177  0.6191430
## 11  0.9946660  0.62544795 -0.8134359
```

```
#accuracy of prediction
(404 + 45 + 33) / nrow(testing)
```

```
## [1] 0.906015
```

```
plot(cmdscale(dist(training[,-10])), col = as.integer(training[,10]),
     pch = c("o", "+")[1:443 %in% svm$index + 1])
```


cmdscale(dist(training[, −10]))[,2]
cmdscale(dist(training[, −10]))[,1]