Predicting Fetal Health with Fetal Heart Rate

March 2022

Narin Dhatwalia
Jason Summer
Simon Zhu

Executive Summary

For long, the United States has witnessed skyrocketing maternal mortality and an increase in the rate of unnecessary Cesarean sections. Such unfortunate phenomena arguably originate from lack of consistent and objective interpretation of the pre-delivery CTG data. Therefore, to better facilitate the decision-making process of the clinicians, this project aims to construct an effective supervised learning model that classifies potentially pathological fetuses.

Through analyses of the 22 variables extracted from the CTG fetal heart rate data, several variables stand out as key indicators for fetuses at danger. The top three are the abnormal short-term variability, percentage of time with abnormal long-term variability, and mean value of short-term variability. The optimal model is a Random Forest model that yields an overall accuracy of 0.9443 and Pathological class F1-score of 0.9122.

With consistent and precise fetal heart rate tracking in the third trimester of pregnancy and accurate CTG processing, real-time classifications made with this model could support clinicians in determining fetal risk and subsequent need for Cesarean sections.

Problem Statement

The U.S. has one of the highest rates of maternal mortality in the industrialized world, placing the United States in the unenviable company of Afghanistan, Greece, and El Salvador. Each year in the U.S., 700 to 900 women die from complications in pregnancy or childbirth. These numbers overshadow the more pervasive problem of severe maternal morbidity. Maternal health experts credit massive disparities in race, education, income, rural demographics, access to healthcare/insurance, and other socio-economic factors as prime reasons, accelerated by a greater prevalence of C- sections, obesity, and chronic health issues including hypertension and diabetes.

“In light of what was mentioned above, Cardiotocograms (CTGs) are a simple and cost accessible option to assess fetal health, allowing healthcare professionals to take action in order to prevent child and maternal mortality. The equipment itself works by sending ultrasound pulses and reading its response, thus shedding light on fetal heart rate (fHR), fetal movements, uterine contractions and more. (kaggle)”

Despite routine use of fetal heart rate (fHR) monitoring as the standard of care to assess the well-being of the fetus during labor, a consistent lack of consensus in fHR interpretation, even among experts, exists despite decades of use of fHR as a common tool. In addition, in smaller or more rural facilities, exposure to lower volumes further erodes reliable interpretation and recognition of concerning fHR patterns. The impact of inconsistent fHR interpretation has led to skyrocketing C-section rates and unfortunate delays in delivery with adverse lifelong consequences for the newborn.

Project Objective and Approach

To combat subjectivity of CTG reading and interpretation among clinicians, the goal of this project is to construct an effective supervised learning model to classify fetal health as “normal”, “suspect”, and “pathological”. The latter two classifications (“normal” and “pathological”) indicate increased risk of adverse fetal events or conditions, such as asphyxia. More specifically, several multi-class classification models will be evaluated in their effectiveness in classifying the three categories of fetal health with a heavy focus on accurately identifying “pathological” fetal health observations, which correspond to the highest risk pregnancies. To conduct the predictions, each pregnancy observation is accompanied by features derived from fHR and uterine contraction (UC) CTG readings. More information about the features considered can be seen in the Data Landscape section below.

Data Landscape

The data studied in this effort were obtained via https://www.kaggle.com/andrewmvd/fetal-health-classification, but originally sourced from in the University of California Irvine Machine Learning Repository. The dataset contains 2,126 observations, each of which corresponds to a third trimester pregnancy and includes features extracted from Cardiotocogram exams. The fetal CTGs were generated from SisPorto 2.0 software, a program intended for automated analysis of CTG (ncbi). Each observation was classified by three expert obstetricians as either “normal” (78%), “suspect (14%)”, or “pathological” (8%).

When evaluating CTG fetal heart rate and uterine contractions, clinicians will take note of a variety of characteristics of the signals in deducing risk and health of the fetus. Generally speaking, these characteristics aim to gauge one of the following areas: fHR baseline rate, variability, accelerations, decelerations, and uterine contraction frequency, intensity, and duration. Furthermore, each one of these metrics has subjectively agreed upon normal ranges, suggesting that periods below or above these ranges correspond to fetal risk. Many of these measures were captured from CTG processing and listed below as considerations for modeling features. Each row corresponds to a single pregnancy.

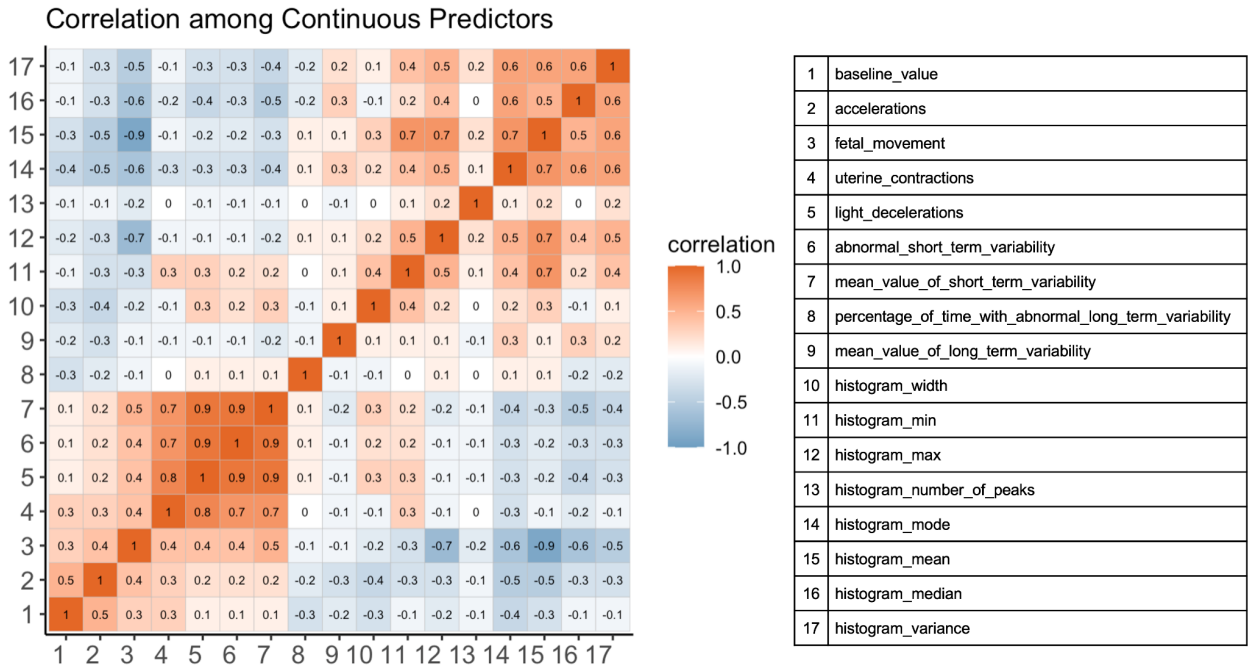
Attribute	**Description**
baseline value	Average beats per minute captured across large portions of fHR
accelerations	Number of accelerations per second in fHR
fetal_movement	Number of fetal movements per second in fHR
uterine_contractions	Number of uterine contractions per second in fHR
light_decelerations	Number of light decelerations per second in fHR
severe_decelerations	Number of severe decelerations per second in fHR
prolongued_decelerations	Number of prolonged decelerations per second in fHR
abnormal_short_term_variability	Percentage of fHR with abnormal short term variability
mean_value_of_short_term_variability	Mean value of short term variability in fHR
percentage_of_time_with_abnormal_long_term_variability	Percentage of fHR with abnormal long term

	variability
mean_value_of_long_term_variability	Mean value of long term variability in fHR
histogram_width	Range of observed fHR values
histogram_min	Smallest observed fHR value
histogram_max	Largest observed fHR value
histogram_number_of_peaks	Modality of fHR
histogram_number_of_zeroes	Number of null/absent readings in fHR
histogram_mode	Most frequently observed fHR value
histogram_mean	Average of observed fHR values
histogram_median	50th percentile of observed fHR values
histogram_variance	Variance of observed fHR values
histogram_tendency	Skewness measure of observed fHR values (left, right, symmetric)

Data Exploration and Considerations

Predictor Evaluation

A large portion of the predictors in consideration are intended to characterize the variability and central tendency of the fHR signal. As a result, a degree of correlation and subsequently multicollinearity among predictors is expected, as shown in the below view. Multicollinearity will be considered further in the subsequent section.



1	baseline_value
2	accelerations
3	fetal_movement
4	uterine_contractions
5	light_decelerations
6	abnormal_short_term_variability
7	mean_value_of_short_term_variability
8	percentage_of_time_with_abnormal_long_term_variability
9	mean_value_of_long_term_variability
10	histogram_width
11	histogram_min
12	histogram_max
13	histogram_number_of_peaks
14	histogram_mode
15	histogram_mean
16	histogram_median
17	histogram_variance

Figure #: Pairwise Pearon's correlation among predictors

As an initial gauge of predictive ability, values among the continuous variables were compared based on each observation's fetal health classification. Several variables highlight an immediate distinction between a "pathological" classification and "normal", such as accelerations, light_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_mode, histogram_mean, histogram_median, and histogram_variance.

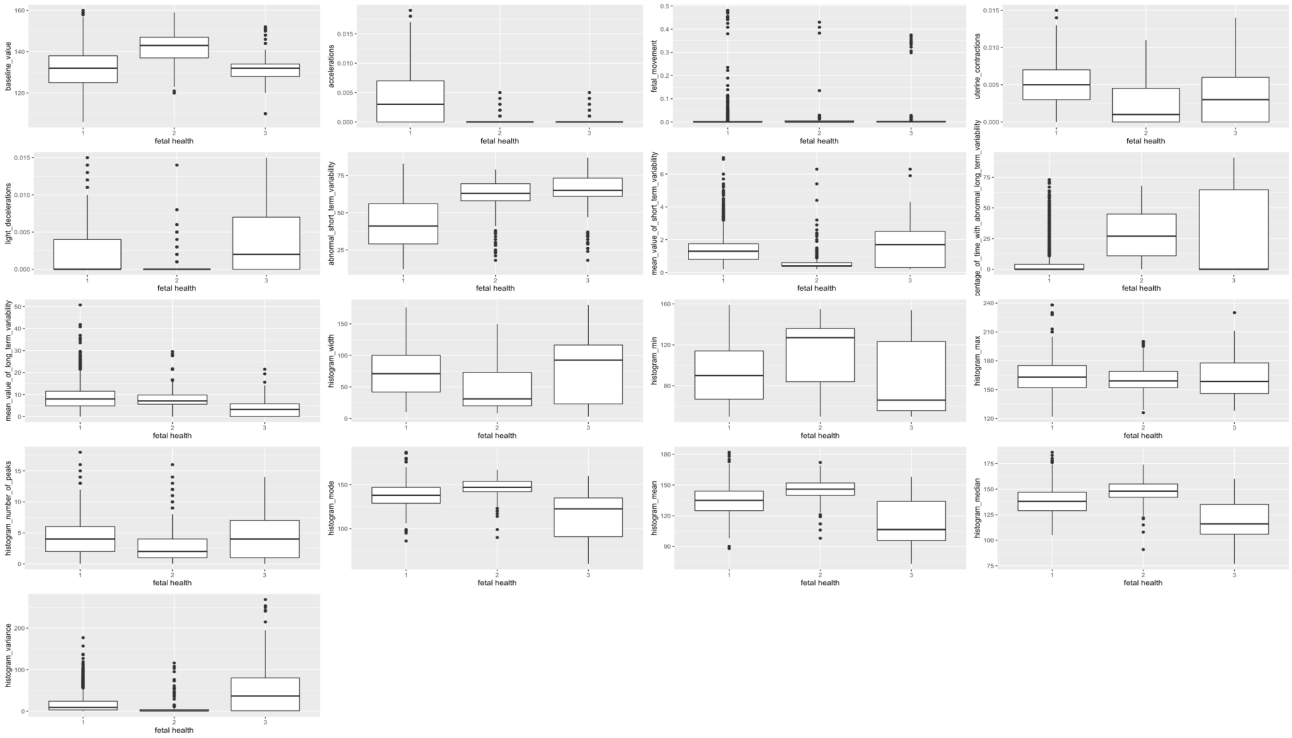


Figure #: Fetal heart rate and uterine contraction feature values among fetal health classifications

Fetal Health Exploration and Distribution of Predictors

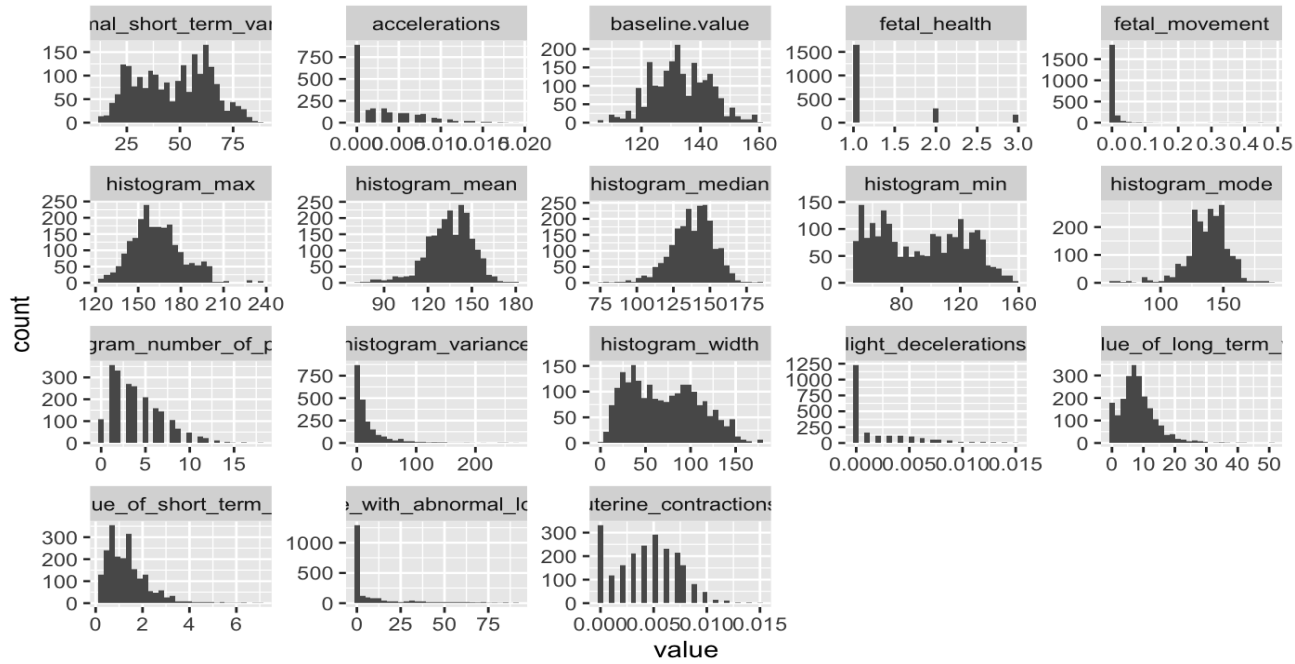


Figure #: Histograms of continuous predictors as well as the categorical response (fetal_health)

The fetal health classes, “normal” ,”suspect”, and “pathological”, were set to values 1, 2, and 3, respectively. As seen in the histogram above, the distribution of the response variable

(fetal_health) is as follows:

- fetal_health = 1; 78% of observations (Normal)
- fetal_health = 2; 14% of observations (Suspect)
- fetal_health = 3; 08% of observations (Pathological)

This clearly implies that the response variable has a much higher proportion of “Normal” observations as compared to “Suspect” and “Pathological” observations. To address this issue of class imbalance, we may need to consider oversampling techniques that aim to balance the data by creating roughly equal number of observations corresponding to each category in the response variable.

Some other notable observations about the predictor distributions are as follows:

- The predictor “histogram mode” provides the most frequently observed fHR value corresponding to a single mother. As seen in the visualization above, most mothers tend to have “histogram mode” between 120 and 150 beats per minute.
- The “histogram width” appears to have a bimodal distribution with one mode at a width of 45, and another mode at a width of 90. This is primarily due to large variability in the “histogram min” predictor since the “histogram width” is defined as the difference between “histogram max” and “histogram min”. This implies that the fHR readings are prone to sudden drops in the signal (either due to physiological reasons or due to equipment malfunction). Note that “histogram max” has much smaller variance, and therefore, does not contribute much to the large variability observed in “histogram width”.

Key Modeling Considerations

Multicollinearity

There are certain notable linear relationships between some of the predictors in the dataset. The most important ones have been mentioned below:

- “Histogram min” and “Histogram width”; correlation = -0.89851
- “Histogram mean” and “Histogram mode”; correlation = 0.89341
- “Histogram median” and “Histogram mode”; correlation = 0.93339
- “Histogram mean” and “Histogram median”; correlation = 0.94825

These highly correlated variables can have unintended effects in the model fitting process. For example, while fitting a logistic regression model, the matrix of predictors becomes singular which leads to reduced precision of the estimated coefficients, thus weakening the statistical power of the model.

Class Imbalance

As previously mentioned, “pathological” and “suspect” fetal health classifications are outnumbered by “normal” classifications at a rate of 9-to-1 and 5-to-1, respectively. Several models under consideration are sensitive to such class imbalances. To mitigate the imbalance, “pathological” and “suspect” observations were upsampled, e.g. replicated, based on the above ratios. Furthermore, upsampling was performed within each cross-validation fold to avoid any leakage of training observations into validation sets.

Cross Validation

To gauge future prediction performance and to provide a fair comparison among models, 10-fold cross validation was used. Furthermore, each model’s cross-validation was executed five times to ensure reliability of metrics.

Modeling Approaches and Performance Evaluation

A number of classification metrics were considered in comparing models. However, the largest focus was placed on correctly classifying truly “pathological” observations. Thus, precision, recall, and the harmonic mean of the two (F1-scores) were prioritized as the most critical metrics when comparing performance using balanced class data.

Generalized Linear Model (Poisson Regression)

This model serves as a baseline against which other more complex models can be benchmarked. The “glm” package in R was leveraged to train the Poisson regression model. Poisson regression assumes the response variable *Y* has a Poisson distribution, and assumes the logarithm of its expected value can be modeled by a linear combination of unknown parameters. Thus the following model is considered:

$$log(\lambda_i) = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_k x_{ik}$$

In order to avoid the problems associated with multicollinearity, the predictor variables in the final model were determined using a backward stepwise regression approach. Further, *K*-fold Cross Validation (with *K* = 10 and multiple replicates) was utilized to avoid overfitting. The final prediction metrics pertaining to the “pathological” class are as follows:

Pathological Class F1-Score	0.2076
Pathological Class Precision	1.0000
Pathological Class Recall	0.1159

Overall Accuracy	0.8062

Support Vector Machine (SVM)

As one of the most common supervised learning algorithms for classification, Support Vector Machine applies well to the data. Various kernels and corresponding parameters such as cost (regularization parameter) and gamma were tested in a grid search. Among the 25 different combinations, the SVM model with radial basis function (RBF) kernel, cost of 10, and gamma of 0.047 yielded the highest F1 score and overall accuracy for the “pathological” class. The exact evaluation metrics are as follows:

Pathological Class F1-Score	0.8834
Pathological Class Precision	0.9169
Pathological Class Recall	0.8523
Overall Accuracy	0.9308

In terms of feature importance, the transformations conducted by the RBF kernel are implicit and hard to retrieve. Therefore, the respective weights on the features remain unknown.

Neural Network

Single layer neural networks were evaluated with varying degrees of regularization and number of hidden nodes. As with other methods, five replicates of 10-fold cross-validation were used to provide comprehensive and accurate performance metrics. Ultimately, a hidden node size of 30 and decay (regularization parameter) of 0.11 provided optimal cross-validated performance, as shown below.

Pathological Class F1-Score	0.88152
Pathological Class Precision	0.90648
Pathological Class Recall	0.85795
Overall Accuracy	0.93603

Unfortunately, due to the nature of multi-class classification with a neural network, individual feature importance evaluations are very limited, thus restricting the interpretative use of such modeling.

Random Forest

Random Forest models were evaluated with varying combinations of hyperparameters, including *mtry* (number of variables randomly sampled as candidates at each split) and *nodesize* (minimum size of terminal nodes). The optimal model has *mtry* of 6 and *nodesize* of 5, and yields the following evaluation metrics.

Pathological Class F1-Score	0.9122
Pathological Class Precision	0.9379
Pathological Class Recall	0.8886
Overall Accuracy	0.9443

Based on the mean decrease in accuracy yielded from all the features, the top three important features are *abnormal_short_term_variability, percentage_of_time_with_abnormal_long_term_variability, and mean_value_of_short_term_variability.* However, the mean decrease in accuracy does not differ much for these variables.

The partial dependency plots of the features show their effects on the probability of being classified as pathological. For instance, when the abnormal short-term variability increases, the probability of pathological fetus dramatically increases when the scaled variability is 0.5. On the other hand, when the mean value of short-term variability increases, the probability drops drastically.

Model Selection

Ten-fold cross validation with five replicates was conducted on all of the models for comparison. As shown from the results above, the Random Forest model outperforms all the other models in terms of all the metrics except for the pathological class precision rate. However, although the Poisson regression model yields a pathological class precision rate of 1.0, its other metrics are far lower than those of random forest models. Therefore, the Random Forest model is the optimal choice based on model performance.

The Random Forest model also exhibits higher model interpretability, which is commonly required in healthcare data analyses. Although tools such as Partial Dependency plots help demonstrate main effects of various variables, the random forest model and the Poisson regression are the only two models that could provide insights into the relative feature importance.

In brief, Random Forest is the optimal model with regards to both model performance and model interpretability.

Key Takeaways

The selected random forest model renders an optimal overall accuracy of 0.9443 and pathological classification F1-score of 0.9122, calculated as the harmonic mean of the class' recall and precision. Such optimal performance has significant implications in the fight to reduce maternal mortality, fetal risk, and unnecessary Cesarean sections. With consistent and precise fetal heart rate tracking in the third trimester of pregnancy and accurate CTG processing (to derive critical features), real-time predictions can be made to support clinicians in determining fetal risk and subsequent need for Cesarean sections. While it is worth mentioning at this juncture that accurate CTG processing to derive fetal heart rate variability, accelerations, decelerations, and other key features is not trivial, there are a variety of signal processing tools available to do so. Furthermore, the optimal model shown above accentuates the impact of conducting such research.

Sources

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

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6822315/

Appendix - EDA and Modeling (R Code)

Exploratory Data Analysis

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

```
library(readr)
library(dplyr)
library(ggplot2)
library(ggthemes)
library(ggcorrplot)
library(expss)
library(here)
library(epiDisplay)
library(dplyr)
library(ggplot2)
library(nnet)
library(caret)
library(e1071)
library(splitTools)
library(randomForest)
```

```
heart <- read_csv("../data/fetal_health.csv")
names(heart)[1] = "baseline_value"
```

EDA (Continous Variables)

```
cols_discrete <- c("severe_decelerations","prolongued_decelerations",
                   "histogram_number_of_zeroes",
                   "histogram_tendency", "fetal_health")
cols_continuous <- names(heart %>% select(-cols_discrete))

corr <- round(heart %>% select(cols_continuous) %>% cor(),1)
ggcorrplot(corr, hc.order=TRUE,
           ggtheme=theme_classic,
           colors = c("#6D9EC1", "white", "#E46726"),
           lab=TRUE,
           lab_size=2,
           title="Correlation among Continuous Predictors",
           legend.title="correlation") +
  scale_x_discrete(labels=seq(1:length(cols_continuous))) +
  scale_y_discrete(labels=seq(1:length(cols_continuous))) +
  theme(axis.text.x = element_text(angle=0))

col_lookup <- data.frame(seq(1:length(cols_continuous)),cols_continuous )
print(col_lookup)
```

```
fetal_health_boxplot <- function(col_name) {
  x <- unlist(heart[,c(col_name)], use.names = F)
```

```
  data <- data.frame(heart$fetal_health,x)
  p <- ggplot(data, aes(x=factor(heart.fetal_health), y=x)) +
    geom_boxplot() + xlab("fetal health") + ylab(col_name)
    scale_x_discrete(labels=c("normal","suspect",""))
  p
}

p<-c()
p1 <- fetal_health_boxplot(cols_continuous[1])
p2 <- fetal_health_boxplot(cols_continuous[2])
p3 <- fetal_health_boxplot(cols_continuous[3])
p4 <- fetal_health_boxplot(cols_continuous[4])
p5 <- fetal_health_boxplot(cols_continuous[5])
p6 <- fetal_health_boxplot(cols_continuous[6])
p7 <- fetal_health_boxplot(cols_continuous[7])
p8 <- fetal_health_boxplot(cols_continuous[8])
p9 <- fetal_health_boxplot(cols_continuous[9])
p10 <- fetal_health_boxplot(cols_continuous[10])
p11 <- fetal_health_boxplot(cols_continuous[11])
p12 <- fetal_health_boxplot(cols_continuous[12])
p13 <- fetal_health_boxplot(cols_continuous[13])
p14 <- fetal_health_boxplot(cols_continuous[14])
p15 <- fetal_health_boxplot(cols_continuous[15])
p16 <- fetal_health_boxplot(cols_continuous[16])
p17 <- fetal_health_boxplot(cols_continuous[17])

grid.arrange(p1, p2, p3, p4, p5, p6, p7, p8, p9,
             p10, p11, p12, p13, p14, p15, p16, p17)
```

```
create_upsampled_folds <- function(df, k) {
  #' Create Newly Upsampled Folds
  #'
  #' Generates new list of folds wherein indices corresponding to minority labels of 3 and 2
  #' are upsampled 9x and 5x, respectively.
  #'
  #' @param df dataframe. fetal health records should be used here
  #' @param k integer. number of folds

  p <- rep((nrow(df)/k)/nrow(df),k)
  folds <- partition(df$fetal_health, p=p, type=c("basic"), shuffle=TRUE)
  new_folds <- c()
  for (fold_i in 1:length(folds)) {
    upsampled_fold <- c()
    for (i in folds[[fold_i]]) { # Check label and upsample based on entire data balance
      label <- unlist(df[i,22],use.names = F)
      if (label == 3) {
        upsampled_fold <- append(upsampled_fold, rep(i,9))
      } else if (label == 2) {
        upsampled_fold <- append(upsampled_fold, rep(i,5))
      } else {
        upsampled_fold <- append(upsampled_fold, i)
      }
    }
    new_folds[[fold_i]] <- upsampled_fold
```

```
  }
  new_folds
}
```

```
data = read_csv("../data/fetal_health.csv")
```

EDA of discrete variables and response variable

```
cols_discrete = c("severe_decelerations","prolongued_decelerations",
                  "histogram_number_of_zeroes",
                  "histogram_tendency")

# Histogram for discrete variables
data %>% dplyr::select(cols_discrete) %>%
  tidyr::pivot_longer(cols_discrete) %>%
  ggplot(aes(x=value)) +
  geom_histogram()+
  scale_y_continuous(trans="log",labels=function(x) sprintf("%.2f", x))+
  facet_wrap(~name,scale="free")+
  xlab("Variable")+
  ylab("log(count)")

# cross tab
for (i in 1:4){
  for (j in i:4){
    if (i!=j){
      t = table(data[,cols_discrete[i]],
                          data[,cols_discrete[j]])
      chisq = chisq.test(t)
      if (chisq$p.value <=0.05){
        print(paste0(cols_discrete[i], " and ", cols_discrete[j], " are associated."))
        print(paste0("With a p-value of: ",round(chisq$p.value,4)))
      }
    }
  }
}
```

EDA of response variable

```
# frequency table and distribution graphs of the fetal_health
tab1(data$fetal_health, sort.group = "decreasing",
     graph=FALSE, cum.percent = TRUE,
     bar.values="percent",main = "Distribution of fetal health classes")

# relationship with other categorical variables
for (c in cols_discrete){
  print(paste0("---------------- cross table between ",c," and fetal_health ----------------"))
  print(prop.table(table(data[,c],data[,"fetal_health"])),digits=3)
}
```

Histograms of the Continuous Predictors

```
df <- read.csv('../data/fetal_health.csv')
cols_discrete = c("severe_decelerations","prolongued_decelerations",
                  "histogram_number_of_zeroes",
                  "histogram_tendency")

df <- df[, -which(names(df) %in% cols_discrete)]

df_transpose <- as.data.frame(t(as.matrix(df)))
df_transpose <- cbind(rownames(df_transpose), data.frame(df_transpose, row.names=NULL))

library(tidyverse)

df_transpose %>%
  pivot_longer(cols = -`rownames(df_transpose)`) %>%
  ggplot(aes(value)) +
  facet_wrap(~ `rownames(df_transpose)`, scales = "free") +
  geom_histogram(bins = 30)
```

Corellation Matrix

```
test <- cor(df)
as.data.frame(apply(test, 2, function(x) ifelse (abs(x) >=0.85,x,"NA")))
```

Modeling

```
library("readr")
library("dplyr")

create_upsampled_folds <- function(df, k) {
  #' Create Newly Upsampled Folds
  #'
  #' Generates new list of folds wherein indices corresponding to minority labels of 3 and 2
  #' are upsampled 9x and 5x, respectively.
  #'
  #' @param df dataframe. fetal health records should be used here
  #' @param k integer. number of folds

  p <- rep((nrow(df)/k)/nrow(df),k)
  folds <- partition(df$fetal_health, p=p, type=c("basic"), shuffle=TRUE)
  new_folds <- c()
  for (fold_i in 1:length(folds)) {
    upsampled_fold <- c()
    for (i in folds[[fold_i]]) { # Check label and upsample based on entire data balance
      label <- unlist(df[i,22],use.names = F)
      if (label == 3) {
        upsampled_fold <- append(upsampled_fold, rep(i,9))
      } else if (label == 2) {
        upsampled_fold <- append(upsampled_fold, rep(i,5))
      } else {
        upsampled_fold <- append(upsampled_fold, i)
```

```
      }
    }
    new_folds[[fold_i]] <- upsampled_fold
  }
  new_folds
}
```

Data Import

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

```
heart <- read_csv("../data/fetal_health.csv")
names(heart)[1] = "baseline_value"
```

Multinomial Neural Network Classification

```
heart_std <- heart
heart_std[1:21] <-  sapply(heart_std[1:21], function(x) (x-mean(x))/sd(x)) #standardize predictors
heart_std$fetal_health <- factor(heart_std$fetal_health, ordered = FALSE) # set response to factor

set.seed(42)
##Now use multiple reps of CV to compare Neural Nets and logistic reg models###
Nrep<-5 #number of replicates of CV
K<-10  #K-fold CV on each replicate
n.models = 5 #number of different models to fit
n=nrow(heart_std)
y<-heart_std[[22]]
yhat=matrix(0,n,n.models)
CV.overall_accuracy <- matrix(0,Nrep,n.models)
CV.precision_class_3 <- matrix(0,Nrep,n.models)
CV.recall_class_3 <- matrix(0,Nrep,n.models)
CV.F1_class_3 <- matrix(0,Nrep,n.models)
CV.F1_class_2 <- matrix(0,Nrep,n.models)
CV.F1_class_1 <- matrix(0,Nrep,n.models)
CV.test <- matrix(0,Nrep,n.models)
for (j in 1:Nrep) {
  Ind<-create_upsampled_folds(heart_std,K)
  for (k in 1:K) {
    out<-nnet(fetal_health~.,heart_std[-Ind[[k]],],linout=FALSE,skip=FALSE,size=2,decay=.1, maxit=1000,
    phat<-predict(out,heart_std[Ind[[k]],], 'class');  yhat[Ind[[k]],1] <- phat
    out<-nnet(fetal_health~.,heart_std[-Ind[[k]],],linout=FALSE,skip=FALSE,size=5,decay=.1, maxit=1000,
    phat<-predict(out,heart_std[Ind[[k]],], 'class');  yhat[Ind[[k]],2] <- phat
    out<-nnet(fetal_health~.,heart_std[-Ind[[k]],],linout=FALSE,skip=FALSE,size=7,decay=.1, maxit=1000,
    phat<-predict(out,heart_std[Ind[[k]],], 'class');  yhat[Ind[[k]],3] <- phat
    out<-nnet(fetal_health~.,heart_std[-Ind[[k]],],linout=FALSE,skip=FALSE,size=9,decay=.1, maxit=1000,
    phat<-predict(out,heart_std[Ind[[k]],], 'class');  yhat[Ind[[k]],4] <- phat
    out<-nnet(fetal_health~.,heart_std[-Ind[[k]],],linout=FALSE,skip=FALSE,size=13,decay=.1, maxit=1000
    phat<-predict(out,heart_std[Ind[[k]],], 'class');  yhat[Ind[[k]],5] <- phat

  } #end of k loop
  #CV.rate[j,]=apply(yhat,2,function(x) sum(y != x)/n) # mis-class rate

  CV.overall_accuracy[j,] = apply(yhat,2,
                      function(x) confusionMatrix(reference=heart_std$fetal_health,
```

```
                                                    as.factor(x))$overall["Accuracy"])
  CV.precision_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[3,"Precision"])
  CV.recall_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[3,"Recall"])
  CV.F1_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[3,"F1"])
  CV.F1_class_2[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[2,"F1"])
  CV.F1_class_1[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[1,"F1"])

} #end of j loop
CV.overall_accuracy_Ave<- apply(CV.overall_accuracy,2,mean) #averaged CV misclass rate
CV.precision_class_3Ave<-apply(CV.precision_class_3,2,mean)
CV.recall_class_3Ave<-apply(CV.recall_class_3,2,mean)
CV.F1_class_3Ave<-apply(CV.F1_class_3,2,mean)
CV.F1_class_2Ave<-apply(CV.F1_class_2,2,mean)
CV.F1_class_1Ave<-apply(CV.F1_class_1,2,mean)
```

A hidden node size of 9 provides an optimal initial balance of recall and precision for class 3 (pathological fetal health) while maintaining adequate F1 scores for classes 1 and 2. The F1 for class 1 is 0.8683865 Below, a more narrow tuning will be used for node size and lambda.

```
set.seed(42)
##Now use multiple reps of CV to compare Neural Nets and logistic reg models###
Nrep<-5 #number of replicates of CV
K<-10  #K-fold CV on each replicate
n.models = 5 #number of different models to fit
n=nrow(heart_std)
y<-heart_std[[22]]
yhat=matrix(0,n,n.models)
CV.overall_accuracy <- matrix(0,Nrep,n.models)
CV.precision_class_3 <- matrix(0,Nrep,n.models)
CV.recall_class_3 <- matrix(0,Nrep,n.models)
CV.F1_class_3 <- matrix(0,Nrep,n.models)
CV.F1_class_2 <- matrix(0,Nrep,n.models)
CV.F1_class_1 <- matrix(0,Nrep,n.models)
CV.test <- matrix(0,Nrep,n.models)
for (j in 1:Nrep) {
  Ind<-create_upsampled_folds(heart_std,K)
  for (k in 1:K) {
    out<-nnet(fetal_health~.,heart_std[-Ind[[k]],],linout=FALSE,skip=FALSE,size=8,decay=.1, maxit=1000,
    phat<-predict(out,heart_std[Ind[[k]],], 'class');  yhat[Ind[[k]],1] <- phat
    out<-nnet(fetal_health~.,heart_std[-Ind[[k]],],linout=FALSE,skip=FALSE,size=9,decay=.1, maxit=1000,
    phat<-predict(out,heart_std[Ind[[k]],], 'class');  yhat[Ind[[k]],2] <- phat
    out<-nnet(fetal_health~.,heart_std[-Ind[[k]],],linout=FALSE,skip=FALSE,size=10,decay=.1, maxit=1000
    phat<-predict(out,heart_std[Ind[[k]],], 'class');  yhat[Ind[[k]],3] <- phat
    out<-nnet(fetal_health~.,heart_std[-Ind[[k]],],linout=FALSE,skip=FALSE,size=11,decay=.1, maxit=1000
```

```
    phat<-predict(out,heart_std[Ind[[k]],], 'class');  yhat[Ind[[k]],4] <- phat
    out<-nnet(fetal_health~.,heart_std[-Ind[[k]],],linout=FALSE,skip=FALSE,size=12,decay=.1, maxit=1000
    phat<-predict(out,heart_std[Ind[[k]],], 'class');  yhat[Ind[[k]],5] <- phat

  } #end of k loop
  #CV.rate[j,]=apply(yhat,2,function(x) sum(y != x)/n) # mis-class rate

  CV.overall_accuracy[j,] = apply(yhat,2,
                      function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                  as.factor(x))$overall["Accuracy"])
  CV.precision_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[3,"Precision"])
  CV.recall_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[3,"Recall"])
  CV.F1_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[3,"F1"])
  CV.F1_class_2[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[2,"F1"])
  CV.F1_class_1[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[1,"F1"])

} #end of j loop
CV.overall_accuracy_Ave<- apply(CV.overall_accuracy,2,mean) #averaged CV misclass rate
CV.precision_class_3Ave<-apply(CV.precision_class_3,2,mean)
CV.recall_class_3Ave<-apply(CV.recall_class_3,2,mean)
CV.F1_class_3Ave<-apply(CV.F1_class_3,2,mean); CV.F1_class_3Ave
CV.F1_class_2Ave<-apply(CV.F1_class_2,2,mean)
CV.F1_class_1Ave<-apply(CV.F1_class_1,2,mean)
```

Node size of 13 provides a class 3 F1 score of 0.8699674.

```
set.seed(42)
##Now use multiple reps of CV to compare Neural Nets and logistic reg models###
Nrep<-5 #number of replicates of CV
K<-10  #K-fold CV on each replicate
n.models = 3 #number of different models to fit
n=nrow(heart_std)
y<-heart_std[[22]]
yhat=matrix(0,n,n.models)
CV.overall_accuracy <- matrix(0,Nrep,n.models)
CV.precision_class_3 <- matrix(0,Nrep,n.models)
CV.recall_class_3 <- matrix(0,Nrep,n.models)
CV.F1_class_3 <- matrix(0,Nrep,n.models)
CV.F1_class_2 <- matrix(0,Nrep,n.models)
CV.F1_class_1 <- matrix(0,Nrep,n.models)
CV.test <- matrix(0,Nrep,n.models)
for (j in 1:Nrep) {
  Ind<-create_upsampled_folds(heart_std,K)
  for (k in 1:K) {
```

```
    out<-nnet(fetal_health~.,heart_std[-Ind[[k]],],linout=FALSE,skip=FALSE,size=12,decay=.1, maxit=1000
    phat<-predict(out,heart_std[Ind[[k]],], 'class');  yhat[Ind[[k]],1] <- phat
    out<-nnet(fetal_health~.,heart_std[-Ind[[k]],],linout=FALSE,skip=FALSE,size=15,decay=.1, maxit=1000
    phat<-predict(out,heart_std[Ind[[k]],], 'class');  yhat[Ind[[k]],2] <- phat
    out<-nnet(fetal_health~.,heart_std[-Ind[[k]],],linout=FALSE,skip=FALSE,size=20,decay=.1, maxit=1000
    phat<-predict(out,heart_std[Ind[[k]],], 'class');  yhat[Ind[[k]],3] <- phat

  } #end of k loop
  #CV.rate[j,]=apply(yhat,2,function(x) sum(y != x)/n) # mis-class rate

  CV.overall_accuracy[j,] = apply(yhat,2,
                      function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                  as.factor(x))$overall["Accuracy"])
  CV.precision_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[3,"Precision"])
  CV.recall_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[3,"Recall"])
  CV.F1_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[3,"F1"])
  CV.F1_class_2[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[2,"F1"])
  CV.F1_class_1[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[1,"F1"])

} #end of j loop
CV.overall_accuracy_Ave<- apply(CV.overall_accuracy,2,mean) #averaged CV misclass rate
CV.precision_class_3Ave<-apply(CV.precision_class_3,2,mean)
CV.recall_class_3Ave<-apply(CV.recall_class_3,2,mean)
CV.F1_class_3Ave<-apply(CV.F1_class_3,2,mean); CV.F1_class_3Ave
CV.F1_class_2Ave<-apply(CV.F1_class_2,2,mean)
CV.F1_class_1Ave<-apply(CV.F1_class_1,2,mean)
```

Node size of 20 provides a class 3 F1 score of 0.8717784.

```
set.seed(42)
##Now use multiple reps of CV to compare Neural Nets and logistic reg models###
Nrep<-5 #number of replicates of CV
K<-10  #K-fold CV on each replicate
n.models = 1 #number of different models to fit
n=nrow(heart_std)
y<-heart_std[[22]]
yhat=matrix(0,n,n.models)
CV.overall_accuracy <- matrix(0,Nrep,n.models)
CV.precision_class_3 <- matrix(0,Nrep,n.models)
CV.recall_class_3 <- matrix(0,Nrep,n.models)
CV.F1_class_3 <- matrix(0,Nrep,n.models)
CV.F1_class_2 <- matrix(0,Nrep,n.models)
CV.F1_class_1 <- matrix(0,Nrep,n.models)
CV.test <- matrix(0,Nrep,n.models)
```

```
for (j in 1:Nrep) {
  Ind<-create_upsampled_folds(heart_std,K)
  for (k in 1:K) {
    out<-nnet(fetal_health~.,heart_std[-Ind[[k]],],linout=FALSE,skip=FALSE,size=30,decay=.1, maxit=1000
    phat<-predict(out,heart_std[Ind[[k]],], 'class');  yhat[Ind[[k]],1] <- phat

  } #end of k loop
  #CV.rate[j,]=apply(yhat,2,function(x) sum(y != x)/n) # mis-class rate

  CV.overall_accuracy[j,] = apply(yhat,2,
                      function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                  as.factor(x))$overall["Accuracy"])
  CV.precision_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[3,"Precision"])
  CV.recall_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[3,"Recall"])
  CV.F1_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[3,"F1"])
  CV.F1_class_2[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[2,"F1"])
  CV.F1_class_1[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[1,"F1"])

} #end of j loop
CV.overall_accuracy_Ave<- apply(CV.overall_accuracy,2,mean) #averaged CV misclass rate
CV.precision_class_3Ave<-apply(CV.precision_class_3,2,mean)
CV.recall_class_3Ave<-apply(CV.recall_class_3,2,mean)
CV.F1_class_3Ave<-apply(CV.F1_class_3,2,mean); CV.F1_class_3Ave
CV.F1_class_2Ave<-apply(CV.F1_class_2,2,mean)
CV.F1_class_1Ave<-apply(CV.F1_class_1,2,mean)
```

Node size of 30 and decay of 0.1 provides class 3 F1 score of 0.8803023.

```
set.seed(42)
##Now use multiple reps of CV to compare Neural Nets and logistic reg models###
Nrep<-5 #number of replicates of CV
K<-10  #K-fold CV on each replicate
n.models = 3 #number of different models to fit
n=nrow(heart_std)
y<-heart_std[[22]]
yhat=matrix(0,n,n.models)
CV.overall_accuracy <- matrix(0,Nrep,n.models)
CV.precision_class_3 <- matrix(0,Nrep,n.models)
CV.recall_class_3 <- matrix(0,Nrep,n.models)
CV.F1_class_3 <- matrix(0,Nrep,n.models)
CV.F1_class_2 <- matrix(0,Nrep,n.models)
CV.F1_class_1 <- matrix(0,Nrep,n.models)
CV.test <- matrix(0,Nrep,n.models)
for (j in 1:Nrep) {
```

```
  Ind<-create_upsampled_folds(heart_std,K)
  for (k in 1:K) {
    out<-nnet(fetal_health~.,heart_std[-Ind[[k]],],linout=FALSE,skip=FALSE,size=10,decay=.05, maxit=1000
    phat<-predict(out,heart_std[Ind[[k]],], 'class');  yhat[Ind[[k]],1] <- phat
    out<-nnet(fetal_health~.,heart_std[-Ind[[k]],],linout=FALSE,skip=FALSE,size=15,decay=.05, maxit=1000
    phat<-predict(out,heart_std[Ind[[k]],], 'class');  yhat[Ind[[k]],2] <- phat
    out<-nnet(fetal_health~.,heart_std[-Ind[[k]],],linout=FALSE,skip=FALSE,size=20,decay=.05, maxit=1000
    phat<-predict(out,heart_std[Ind[[k]],], 'class');  yhat[Ind[[k]],3] <- phat

  } #end of k loop
  #CV.rate[j,]=apply(yhat,2,function(x) sum(y != x)/n) # mis-class rate

  CV.overall_accuracy[j,] = apply(yhat,2,
                      function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                  as.factor(x))$overall["Accuracy"])
  CV.precision_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[3,"Precision"])
  CV.recall_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[3,"Recall"])
  CV.F1_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[3,"F1"])
  CV.F1_class_2[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[2,"F1"])
  CV.F1_class_1[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[1,"F1"])

} #end of j loop
CV.overall_accuracy_Ave<- apply(CV.overall_accuracy,2,mean) #averaged CV misclass rate
CV.precision_class_3Ave<-apply(CV.precision_class_3,2,mean)
CV.recall_class_3Ave<-apply(CV.recall_class_3,2,mean)
CV.F1_class_3Ave<-apply(CV.F1_class_3,2,mean); CV.F1_class_3Ave
CV.F1_class_2Ave<-apply(CV.F1_class_2,2,mean)
CV.F1_class_1Ave<-apply(CV.F1_class_1,2,mean)
```

```
set.seed(42)
##Now use multiple reps of CV to compare Neural Nets and logistic reg models###
Nrep<-5 #number of replicates of CV
K<-10  #K-fold CV on each replicate
n.models = 1 #number of different models to fit
n=nrow(heart_std)
y<-heart_std[[22]]
yhat=matrix(0,n,n.models)
CV.overall_accuracy <- matrix(0,Nrep,n.models)
CV.precision_class_3 <- matrix(0,Nrep,n.models)
CV.recall_class_3 <- matrix(0,Nrep,n.models)
CV.F1_class_3 <- matrix(0,Nrep,n.models)
CV.F1_class_2 <- matrix(0,Nrep,n.models)
CV.F1_class_1 <- matrix(0,Nrep,n.models)
CV.test <- matrix(0,Nrep,n.models)
```

```
for (j in 1:Nrep) {
  Ind<-create_upsampled_folds(heart_std,K)
  for (k in 1:K) {
    out<-nnet(fetal_health~.,heart_std[-Ind[[k]],],linout=FALSE,skip=FALSE,size=30,decay=.05, maxit=1000
    phat<-predict(out,heart_std[Ind[[k]],], 'class');  yhat[Ind[[k]],1] <- phat

  } #end of k loop
  #CV.rate[j,]=apply(yhat,2,function(x) sum(y != x)/n) # mis-class rate

  CV.overall_accuracy[j,] = apply(yhat,2,
                      function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                  as.factor(x))$overall["Accuracy"])
  CV.precision_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[3,"Precision"])
  CV.recall_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[3,"Recall"])
  CV.F1_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[3,"F1"])
  CV.F1_class_2[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[2,"F1"])
  CV.F1_class_1[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[1,"F1"])

} #end of j loop
CV.overall_accuracy_Ave<- apply(CV.overall_accuracy,2,mean) #averaged CV misclass rate
CV.precision_class_3Ave<-apply(CV.precision_class_3,2,mean)
CV.recall_class_3Ave<-apply(CV.recall_class_3,2,mean)
CV.F1_class_3Ave<-apply(CV.F1_class_3,2,mean); CV.F1_class_3Ave
CV.F1_class_2Ave<-apply(CV.F1_class_2,2,mean)
CV.F1_class_1Ave<-apply(CV.F1_class_1,2,mean)
```

```
set.seed(42)
##Now use multiple reps of CV to compare Neural Nets and logistic reg models###
Nrep<-5 #number of replicates of CV
K<-10  #K-fold CV on each replicate
n.models = 4 #number of different models to fit
n=nrow(heart_std)
y<-heart_std[[22]]
yhat=matrix(0,n,n.models)
CV.overall_accuracy <- matrix(0,Nrep,n.models)
CV.precision_class_3 <- matrix(0,Nrep,n.models)
CV.recall_class_3 <- matrix(0,Nrep,n.models)
CV.F1_class_3 <- matrix(0,Nrep,n.models)
CV.F1_class_2 <- matrix(0,Nrep,n.models)
CV.F1_class_1 <- matrix(0,Nrep,n.models)
CV.test <- matrix(0,Nrep,n.models)
for (j in 1:Nrep) {
  Ind<-create_upsampled_folds(heart_std,K)
  for (k in 1:K) {
```

```
    out<-nnet(fetal_health~.,heart_std[-Ind[[k]],],linout=FALSE,skip=FALSE,size=30,decay=.09, maxit=1000
    phat<-predict(out,heart_std[Ind[[k]],], 'class');  yhat[Ind[[k]],1] <- phat
    out<-nnet(fetal_health~.,heart_std[-Ind[[k]],],linout=FALSE,skip=FALSE,size=30,decay=.1, maxit=1000
    phat<-predict(out,heart_std[Ind[[k]],], 'class');  yhat[Ind[[k]],2] <- phat
    out<-nnet(fetal_health~.,heart_std[-Ind[[k]],],linout=FALSE,skip=FALSE,size=30,decay=.11, maxit=1000
    phat<-predict(out,heart_std[Ind[[k]],], 'class');  yhat[Ind[[k]],3] <- phat
    out<-nnet(fetal_health~.,heart_std[-Ind[[k]],],linout=FALSE,skip=FALSE,size=30,decay=.12, maxit=1000
    phat<-predict(out,heart_std[Ind[[k]],], 'class');  yhat[Ind[[k]],4] <- phat

  } #end of k loop
  #CV.rate[j,]=apply(yhat,2,function(x) sum(y != x)/n) # mis-class rate

  CV.overall_accuracy[j,] = apply(yhat,2,
                      function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                  as.factor(x))$overall["Accuracy"])
  CV.precision_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[3,"Precision"])
  CV.recall_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[3,"Recall"])
  CV.F1_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[3,"F1"])
  CV.F1_class_2[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[2,"F1"])
  CV.F1_class_1[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=heart_std$fetal_health,
                                                       as.factor(x))$byClass[1,"F1"])

} #end of j loop
CV.overall_accuracy_Ave<- apply(CV.overall_accuracy,2,mean) #averaged CV misclass rate
CV.precision_class_3Ave<-apply(CV.precision_class_3,2,mean)
CV.recall_class_3Ave<-apply(CV.recall_class_3,2,mean)
CV.F1_class_3Ave<-apply(CV.F1_class_3,2,mean); CV.F1_class_3Ave
CV.F1_class_2Ave<-apply(CV.F1_class_2,2,mean)
CV.F1_class_1Ave<-apply(CV.F1_class_1,2,mean)
```

Optimal node size is 30 and lambda/decay of 0.11.

```
# scale features

data_scale = data %>%
  mutate(fetal_health = as.factor(fetal_health)) %>%
  mutate_if(is.double,function(x) as.numeric(scale(x)))
```

SVM

rbf model

```
set.seed(42)
##Now use multiple reps of CV to compare svm models###
Nrep<-5 #number of replicates of CV
```

```r
K<-10  #K-fold CV on each replicate
n.models = 25 #number of different models to fit
n=nrow(data_scale)
y<-data_scale$fetal_health
yhat=matrix(0,n,n.models)
CV.overall_accuracy <- matrix(0,Nrep,n.models)
CV.precision_class_3 <- matrix(0,Nrep,n.models)
CV.recall_class_3 <- matrix(0,Nrep,n.models)
CV.F1_class_3 <- matrix(0,Nrep,n.models)
CV.F1_class_1 <- matrix(0,Nrep,n.models)
CV.F1_class_2 <- matrix(0,Nrep,n.models)

cost = c(0.01,0.1,1,10,100)
gamma = c(1/50,1/40,1/21,0.3,0.75)
grid = expand.grid(cost,gamma)
colnames(grid) = c("cost","gamma")

for (j in 1:Nrep) {
  Ind<-create_upsampled_folds(data_scale,K)
  for (k in 1:K) {
    for (l in 1:n.models){
      out = svm(fetal_health~.,data=data_scale[-Ind[[k]],],
                kernel="radial", cost=grid[l,1], gamma = grid[l,2])
      yhat[Ind[[k]],l] = predict(out,data_scale[Ind[[k]],],type="class")
    }
  } #end of k loop

  CV.overall_accuracy[j,] = apply(yhat,2,
                      function(x) confusionMatrix(reference=data_scale$fetal_health,
                                                  as.factor(x))$overall["Accuracy"])
  CV.precision_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=data_scale$fetal_health,
                                                       as.factor(x))$byClass[3,"Precision"])
  CV.recall_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=data_scale$fetal_health,
                                                       as.factor(x))$byClass[3,"Recall"])
  CV.F1_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=data_scale$fetal_health,
                                                       as.factor(x))$byClass[3,"F1"])
  CV.F1_class_1[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=data_scale$fetal_health,
                                                       as.factor(x))$byClass[1,"F1"])
  CV.F1_class_2[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=data_scale$fetal_health,
                                                       as.factor(x))$byClass[2,"F1"])
} #end of j loop
CV.overall_accuracy_Ave<- apply(CV.overall_accuracy,2,mean) #averaged CV misclass rate
CV.precision_class_3Ave<-apply(CV.precision_class_3,2,mean)
CV.recall_class_3Ave<-apply(CV.recall_class_3,2,mean)
CV.F1_class_3Ave<-apply(CV.F1_class_3,2,mean)
CV.F1_class_1Ave<-apply(CV.F1_class_1,2,mean)
CV.F1_class_2Ave<-apply(CV.F1_class_2,2,mean)
```

```
grid %>% mutate(CV.acc = CV.overall_accuracy_Ave,
                CV.prec_3 = CV.precision_class_3Ave,
                CV.recall_3 = CV.recall_class_3Ave,
                CV.F1_class1 = CV.F1_class_1Ave,
                CV.F1_class2 = CV.F1_class_2Ave,
                CV.F1_class3 = CV.F1_class_3Ave) %>%
  arrange(-CV.F1_class3)
```

Linear Model

```
set.seed(42)
##Now use multiple reps of CV to compare svm models###
Nrep<-5 #number of replicates of CV
K<-10  #K-fold CV on each replicate
n.models = 5 #number of different models to fit
n=nrow(data_scale)
y<-data_scale$fetal_health
yhat=matrix(0,n,n.models)
CV.overall_accuracy <- matrix(0,Nrep,n.models)
CV.precision_class_3 <- matrix(0,Nrep,n.models)
CV.recall_class_3 <- matrix(0,Nrep,n.models)
CV.F1_class_3 <- matrix(0,Nrep,n.models)
CV.F1_class_1 <- matrix(0,Nrep,n.models)
CV.F1_class_2 <- matrix(0,Nrep,n.models)

cost = c(0.01,0.1,1,10,100)
grid = expand.grid(cost)
colnames(grid) = c("cost")

for (j in 1:Nrep) {
  Ind<-create_upsampled_folds(data_scale,K)
  for (k in 1:K) {
    for (l in 1:n.models){
      out = svm(fetal_health~.,data=data_scale[-Ind[[k]],],
                kernel="linear", cost=grid[l,1])
      yhat[Ind[[k]],l] = predict(out,data_scale[Ind[[k]],],type="class")
    }
  } #end of k loop

  CV.overall_accuracy[j,] = apply(yhat,2,
                      function(x) confusionMatrix(reference=data_scale$fetal_health,
                                                  as.factor(x))$overall["Accuracy"])
  CV.precision_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=data_scale$fetal_health,
                                                       as.factor(x))$byClass[3,"Precision"])
  CV.recall_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=data_scale$fetal_health,
                                                       as.factor(x))$byClass[3,"Recall"])
  CV.F1_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=data_scale$fetal_health,
                                                       as.factor(x))$byClass[3,"F1"])
  CV.F1_class_1[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=data_scale$fetal_health,
```

```
                                                    as.factor(x))$byClass[1,"F1"])
  CV.F1_class_2[j,] = apply(yhat,2,
                            function(x) confusionMatrix(reference=data_scale$fetal_health,
                                                    as.factor(x))$byClass[2,"F1"])
} #end of j loop
CV.overall_accuracy_Ave<- apply(CV.overall_accuracy,2,mean) #averaged CV misclass rate
CV.precision_class_3Ave<-apply(CV.precision_class_3,2,mean)
CV.recall_class_3Ave<-apply(CV.recall_class_3,2,mean)
CV.F1_class_3Ave<-apply(CV.F1_class_3,2,mean)
CV.F1_class_1Ave<-apply(CV.F1_class_1,2,mean)
CV.F1_class_2Ave<-apply(CV.F1_class_2,2,mean)
```

```
grid %>% mutate(CV.acc = CV.overall_accuracy_Ave,
                CV.prec_3 = CV.precision_class_3Ave,
                CV.recall_3 = CV.recall_class_3Ave,
                CV.F1_class1 = CV.F1_class_1Ave,
                CV.F1_class2 = CV.F1_class_2Ave,
                CV.F1_class3 = CV.F1_class_3Ave) %>%
  arrange(-CV.F1_class3)
```

Random Forest

```
set.seed(42)
##Now use multiple reps of CV to compare svm models###
Nrep<-5 #number of replicates of CV
K<-10  #K-fold CV on each replicate
n.models = 15 #number of different models to fit
n=nrow(data_scale)
y<-data_scale$fetal_health
yhat=matrix(0,n,n.models)
CV.overall_accuracy <- matrix(0,Nrep,n.models)
CV.precision_class_3 <- matrix(0,Nrep,n.models)
CV.recall_class_3 <- matrix(0,Nrep,n.models)
CV.F1_class_3 <- matrix(0,Nrep,n.models)
CV.F1_class_1 <- matrix(0,Nrep,n.models)
CV.F1_class_2 <- matrix(0,Nrep,n.models)

mtry = c(2,4,6,9,12)
nodesize = c(1,3,5)
grid = expand.grid(mtry,nodesize)
colnames(grid) = c("mtry","nodesize")

for (j in 1:Nrep) {
  Ind<-create_upsampled_folds(data_scale,K)
  #print("Replica = ", j)
  for (k in 1:K) {
    for (l in 1:n.models){
      rf1 = randomForest(fetal_health~.,data_scale[-Ind[[k]],], mtry=grid[j,1],
                         ntree=500,nodesize=grid[j,2],importance=TRUE)
      yhat[Ind[[k]],l] = predict(rf1,data_scale[Ind[[k]],],type="class")
    }
  } #end of k loop
```

```
  CV.overall_accuracy[j,] = apply(yhat,2,
                      function(x) confusionMatrix(reference=data_scale$fetal_health,
                                                  as.factor(x))$overall["Accuracy"])
  CV.precision_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=data_scale$fetal_health,
                                                       as.factor(x))$byClass[3,"Precision"])
  CV.recall_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=data_scale$fetal_health,
                                                       as.factor(x))$byClass[3,"Recall"])
  CV.F1_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=data_scale$fetal_health,
                                                       as.factor(x))$byClass[3,"F1"])
  CV.F1_class_1[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=data_scale$fetal_health,
                                                       as.factor(x))$byClass[1,"F1"])
  CV.F1_class_2[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=data_scale$fetal_health,
                                                       as.factor(x))$byClass[2,"F1"])
} #end of j loop
CV.overall_accuracy_Ave<- apply(CV.overall_accuracy,2,mean) #averaged CV misclass rate
CV.precision_class_3Ave<-apply(CV.precision_class_3,2,mean)
CV.recall_class_3Ave<-apply(CV.recall_class_3,2,mean)
CV.F1_class_3Ave<-apply(CV.F1_class_3,2,mean)
CV.F1_class_1Ave<-apply(CV.F1_class_1,2,mean)
CV.F1_class_2Ave<-apply(CV.F1_class_2,2,mean)
```

```
####### SHOW MODEL PERFORMANCES
grid %>% mutate(CV.acc = CV.overall_accuracy_Ave,
                CV.prec_3 = CV.precision_class_3Ave,
                CV.recall_3 = CV.recall_class_3Ave,
                CV.F1_class1 = CV.F1_class_1Ave,
                CV.F1_class2 = CV.F1_class_2Ave,
                CV.F1_class3 = CV.F1_class_3Ave) %>%
  arrange(-CV.F1_class3)
```

```
############ RUN THE BEST MODEL
rf1 = randomForest(fetal_health~.,data_scale,mtry=6,
                       ntree=500,nodesize=1,importance=TRUE)

########## FEATURE IMPORTANCE
as.data.frame(rf1$importance) %>%
  arrange(-MeanDecreaseAccuracy)

######### PARTIAL DEPENDENCY PLOTS
imp <- importance(rf1)
impvar <- rownames(imp)[order(imp[, 4], decreasing=TRUE)]
op <- par(mfrow=c(1, 3))
for (i in 1:3) {
    partialPlot(rf1, data_scale, impvar[i], xlab=impvar[i],
                main = c("Patial Dependence on",impvar[i]),
                which.class = "3")
}
par(op)
```

Data Split

```
library(tidyverse)
library(dplyr)
library(caret)

df <- read.csv('fetal_health.csv')
```

Step-wise Model using glm (with poisson family and 'log' link)

```
full_mod <- glm(fetal_health ~ ., data=df, family=poisson(link = 'log'))
backwards <- step(full_mod,trace=0) #would suppress step by step output.
formula(backwards)
```

```
#install.packages("roxygen2")
#install.packages("docstring")
library(roxygen2)
library(docstring)
create_upsampled_folds <- function(df, k) {
  #' Create Newly Upsampled Folds
  #'
  #' Generates new list of folds wherein indices corresponding to minority labels of 3 and 2
  #' are upsampled 9x and 5x, respectively.
  #'
  #' @param df dataframe. fetal health records should be used here
  #' @param k integer. number of folds

  p <- rep((nrow(df)/k)/nrow(df),k)
  folds <- partition(df$fetal_health, p=p, type=c("basic"), shuffle=TRUE)
  new_folds <- c()
  for (fold_i in 1:length(folds)) {
    upsampled_fold <- c()
    for (i in folds[[fold_i]]) { # Check label and upsample based on entire data balance
      label <- unlist(df[i,22],use.names = F)
      if (label == 3) {
        upsampled_fold <- append(upsampled_fold, rep(i,9))
      } else if (label == 2) {
        upsampled_fold <- append(upsampled_fold, rep(i,5))
      } else {
        upsampled_fold <- append(upsampled_fold, i)
      }
    }
    new_folds[[fold_i]] <- upsampled_fold
  }
 new_folds
}
```

```
set.seed(42)
library(splitTools)
##Now use multiple reps of CV to compare Neural Nets and logistic reg models###
Nrep<-5 #number of replicates of CV
K<-10  #K-fold CV on each replicate
n.models = 1 #number of different models to fit
n=nrow(df)
```

```
y<-df$fetal_health
yhat=matrix(0,n,n.models)
CV.overall_accuracy <- matrix(0,Nrep,n.models)
CV.precision_class_3 <- matrix(0,Nrep,n.models)
CV.recall_class_3 <- matrix(0,Nrep,n.models)
CV.F1_class_3 <- matrix(0,Nrep,n.models)
CV.test <- matrix(0,Nrep,n.models)
for (j in 1:Nrep) {
  Ind<-create_upsampled_folds(df,K)
  for (k in 1:K) {
    out = glm(formula(backwards), data=df[-Ind[[k]],], family=poisson(link = 'log'))
    Y.prob.1 <- predict(out, newdata=df[Ind[[k]],],response)
    lambdas <- Y.prob.1

    probs_1 <- (lambdas^1)*exp(-1*lambdas)/factorial(1)
    probs_2 <- (lambdas^2)*exp(-1*lambdas)/factorial(2)
    probs_3 <- (lambdas^3)*exp(-1*lambdas)/factorial(3)

    prob_matrix <- cbind(probs_1, probs_2, probs_3)

    n <- nrow(df[Ind[[k]],])
    Y.hat.1 = rep(0,n);

    for (i in 1:nrow(prob_matrix)){
      Y.hat.1[i] <- which.max(prob_matrix[i,])
    }

    yhat[Ind[[k]],1] <- Y.hat.1

  }

  CV.overall_accuracy[j,] = apply(yhat,2,
                      function(x) confusionMatrix(reference=as.factor(df$fetal_health),
                                                  as.factor(x))$overall["Accuracy"])
  CV.precision_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=as.factor(df$fetal_health),
                                                       as.factor(x))$byClass[3,"Precision"])
  CV.recall_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=as.factor(df$fetal_health),
                                                       as.factor(x))$byClass[3,"Recall"])
  CV.F1_class_3[j,] = apply(yhat,2,
                           function(x) confusionMatrix(reference=as.factor(df$fetal_health),
                                                       as.factor(x))$byClass[3,"F1"])
} #end of j loop
CV.overall_accuracy_Ave<- apply(CV.overall_accuracy,2,mean) #averaged CV misclass rate
CV.precision_class_3Ave<-apply(CV.precision_class_3,2,mean)
CV.recall_class_3Ave<-apply(CV.recall_class_3,2,mean)
CV.F1_class_3Ave<-apply(CV.F1_class_3,2,mean)

CV.overall_accuracy_Ave
CV.precision_class_3Ave
CV.recall_class_3Ave
CV.F1_class_3Ave
```

```
table(yhat, df$fetal_health)
summary(out)
```