Untitled

2024-04-12

# Introduction

Advanced analytical techniques are becoming more important for guiding cardiotocographic decisions in healthcare. Practitioners can use strong prediction models to enhance treatment efficacy and diagnostic accuracy while navigating complex patient care. However, because medical problems are complex and there is a large amount of data, it is difficult to distinguish the signal from noise. The problem is particularly relevant to cardiotocography (CTG), a key obstetric care treatment. Regression analysis, a fundamental statistical tool, allows you to discover complicated correlations hidden in CTG data. The CTG analysis aimed to clarify the regression analysis in this study. Using the data generated by the cardiotocographic recorder. Extreme gradient boosting (XGBoost), a machine learning technique noted for its prediction skills, uses a dataset. This study aims to create a prediction model that can differentiate between suspicious, pathogenic, and normal CTG traces to improve baby outcomes and allow for prompt treatment interventions. Through a careful combination of theoretical insights and practical implementations, I hope to provide data scientists and medical professionals with the knowledge and skills they need to traverse the complexities of regression analysis in healthcare successfully.

# Materil and Method

## Load the data set

Before model building process, first load the data set in R and save the data set as a df. The top five rows and structure of the data set was checked by using the head and str function of R. While to better understand the data set, the descriptive statistics was accessed by using the summary function.

library(readr)  
df <- read\_csv("cardiotocographic.csv")

## Rows: 2126 Columns: 22  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## dbl (22): LB, AC, FM, UC, DL, DS, DP, ASTV, MSTV, ALTV, MLTV, Width, Min, Ma...  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

head(df)

## # A tibble: 6 × 22  
## LB AC FM UC DL DS DP ASTV MSTV ALTV MLTV  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 120 0 0 0 0 0 0 73 0.5 43 2.4  
## 2 132 0.00638 0 0.00638 0.00319 0 0 17 2.1 0 10.4  
## 3 133 0.00332 0 0.00831 0.00332 0 0 16 2.1 0 13.4  
## 4 134 0.00256 0 0.00768 0.00256 0 0 16 2.4 0 23   
## 5 132 0.00651 0 0.00814 0 0 0 16 2.4 0 19.9  
## 6 134 0.00105 0 0.0105 0.00944 0 0.00210 26 5.9 0 0   
## # ℹ 11 more variables: Width <dbl>, Min <dbl>, Max <dbl>, Nmax <dbl>,  
## # Nzeros <dbl>, Mode <dbl>, Mean <dbl>, Median <dbl>, Variance <dbl>,  
## # Tendency <dbl>, NSP <dbl>

str(df)

## spc\_tbl\_ [2,126 × 22] (S3: spec\_tbl\_df/tbl\_df/tbl/data.frame)  
## $ LB : num [1:2126] 120 132 133 134 132 134 134 122 122 122 ...  
## $ AC : num [1:2126] 0 0.00638 0.00332 0.00256 0.00651 ...  
## $ FM : num [1:2126] 0 0 0 0 0 0 0 0 0 0 ...  
## $ UC : num [1:2126] 0 0.00638 0.00831 0.00768 0.00814 ...  
## $ DL : num [1:2126] 0 0.00319 0.00332 0.00256 0 ...  
## $ DS : num [1:2126] 0 0 0 0 0 0 0 0 0 0 ...  
## $ DP : num [1:2126] 0 0 0 0 0 ...  
## $ ASTV : num [1:2126] 73 17 16 16 16 26 29 83 84 86 ...  
## $ MSTV : num [1:2126] 0.5 2.1 2.1 2.4 2.4 5.9 6.3 0.5 0.5 0.3 ...  
## $ ALTV : num [1:2126] 43 0 0 0 0 0 0 6 5 6 ...  
## $ MLTV : num [1:2126] 2.4 10.4 13.4 23 19.9 0 0 15.6 13.6 10.6 ...  
## $ Width : num [1:2126] 64 130 130 117 117 150 150 68 68 68 ...  
## $ Min : num [1:2126] 62 68 68 53 53 50 50 62 62 62 ...  
## $ Max : num [1:2126] 126 198 198 170 170 200 200 130 130 130 ...  
## $ Nmax : num [1:2126] 2 6 5 11 9 5 6 0 0 1 ...  
## $ Nzeros : num [1:2126] 0 1 1 0 0 3 3 0 0 0 ...  
## $ Mode : num [1:2126] 120 141 141 137 137 76 71 122 122 122 ...  
## $ Mean : num [1:2126] 137 136 135 134 136 107 107 122 122 122 ...  
## $ Median : num [1:2126] 121 140 138 137 138 107 106 123 123 123 ...  
## $ Variance: num [1:2126] 73 12 13 13 11 170 215 3 3 1 ...  
## $ Tendency: num [1:2126] 1 0 0 1 1 0 0 1 1 1 ...  
## $ NSP : num [1:2126] 2 1 1 1 1 3 3 3 3 3 ...  
## - attr(\*, "spec")=  
## .. cols(  
## .. LB = col\_double(),  
## .. AC = col\_double(),  
## .. FM = col\_double(),  
## .. UC = col\_double(),  
## .. DL = col\_double(),  
## .. DS = col\_double(),  
## .. DP = col\_double(),  
## .. ASTV = col\_double(),  
## .. MSTV = col\_double(),  
## .. ALTV = col\_double(),  
## .. MLTV = col\_double(),  
## .. Width = col\_double(),  
## .. Min = col\_double(),  
## .. Max = col\_double(),  
## .. Nmax = col\_double(),  
## .. Nzeros = col\_double(),  
## .. Mode = col\_double(),  
## .. Mean = col\_double(),  
## .. Median = col\_double(),  
## .. Variance = col\_double(),  
## .. Tendency = col\_double(),  
## .. NSP = col\_double()  
## .. )  
## - attr(\*, "problems")=<externalptr>

summary(df)

## LB AC FM UC   
## Min. :106.0 Min. :0.000000 Min. :0.000000 Min. :0.000000   
## 1st Qu.:126.0 1st Qu.:0.000000 1st Qu.:0.000000 1st Qu.:0.001876   
## Median :133.0 Median :0.001630 Median :0.000000 Median :0.004482   
## Mean :133.3 Mean :0.003170 Mean :0.009474 Mean :0.004357   
## 3rd Qu.:140.0 3rd Qu.:0.005631 3rd Qu.:0.002512 3rd Qu.:0.006525   
## Max. :160.0 Max. :0.019284 Max. :0.480634 Max. :0.014925   
## DL DS DP ASTV   
## Min. :0.000000 Min. :0.000e+00 Min. :0.0000000 Min. :12.00   
## 1st Qu.:0.000000 1st Qu.:0.000e+00 1st Qu.:0.0000000 1st Qu.:32.00   
## Median :0.000000 Median :0.000e+00 Median :0.0000000 Median :49.00   
## Mean :0.001885 Mean :3.585e-06 Mean :0.0001566 Mean :46.99   
## 3rd Qu.:0.003264 3rd Qu.:0.000e+00 3rd Qu.:0.0000000 3rd Qu.:61.00   
## Max. :0.015385 Max. :1.353e-03 Max. :0.0053476 Max. :87.00   
## MSTV ALTV MLTV Width   
## Min. :0.200 Min. : 0.000 Min. : 0.000 Min. : 3.00   
## 1st Qu.:0.700 1st Qu.: 0.000 1st Qu.: 4.600 1st Qu.: 37.00   
## Median :1.200 Median : 0.000 Median : 7.400 Median : 67.50   
## Mean :1.333 Mean : 9.847 Mean : 8.188 Mean : 70.45   
## 3rd Qu.:1.700 3rd Qu.:11.000 3rd Qu.:10.800 3rd Qu.:100.00   
## Max. :7.000 Max. :91.000 Max. :50.700 Max. :180.00   
## Min Max Nmax Nzeros   
## Min. : 50.00 Min. :122 Min. : 0.000 Min. : 0.0000   
## 1st Qu.: 67.00 1st Qu.:152 1st Qu.: 2.000 1st Qu.: 0.0000   
## Median : 93.00 Median :162 Median : 3.000 Median : 0.0000   
## Mean : 93.58 Mean :164 Mean : 4.068 Mean : 0.3236   
## 3rd Qu.:120.00 3rd Qu.:174 3rd Qu.: 6.000 3rd Qu.: 0.0000   
## Max. :159.00 Max. :238 Max. :18.000 Max. :10.0000   
## Mode Mean Median Variance   
## Min. : 60.0 Min. : 73.0 Min. : 77.0 Min. : 0.00   
## 1st Qu.:129.0 1st Qu.:125.0 1st Qu.:129.0 1st Qu.: 2.00   
## Median :139.0 Median :136.0 Median :139.0 Median : 7.00   
## Mean :137.5 Mean :134.6 Mean :138.1 Mean : 18.81   
## 3rd Qu.:148.0 3rd Qu.:145.0 3rd Qu.:148.0 3rd Qu.: 24.00   
## Max. :187.0 Max. :182.0 Max. :186.0 Max. :269.00   
## Tendency NSP   
## Min. :-1.0000 Min. :1.000   
## 1st Qu.: 0.0000 1st Qu.:1.000   
## Median : 0.0000 Median :1.000   
## Mean : 0.3203 Mean :1.304   
## 3rd Qu.: 1.0000 3rd Qu.:1.000   
## Max. : 1.0000 Max. :3.000

## Split the data

The data set was splitted into the 20 and 80 percent. Our train data set have 1700 observations while test data have 426 observations with 22 columns.

# Set seed and split the data into training and testing sets (80:20)  
set.seed(123)  
train\_index <- sample(nrow(df), 0.8 \* nrow(df))  
train\_data <- df[train\_index, ]  
test\_data <- df[-train\_index, ]  
dim(train\_data)

## [1] 1700 22

dim(test\_data)

## [1] 426 22

## Prepare the Data for model

In this process first create a model matrix for train and test data set and our dependent variable was saved separately into y train and test data.

X\_train <- as.matrix(train\_data[, -ncol(train\_data)])  
y\_train <- train\_data$NSP  
X\_test <- as.matrix(test\_data[, -ncol(test\_data)])  
y\_test <- test\_data$NSP

# XGBoost model

After training on three classes and a multi-class classification goal, the XGBoost model underwent one hundred iterations to refine its parameters and improve its prediction capability. The multi-class logarithmic loss metric to assess the model’s performance during training. This metric evaluates the degree to which predicted probability conforms with actual class labels. When the model started to perform less than optimally, it showed significant loss numbers. However, as the iteration continued, the loss progressively dropped, demonstrating the model’s increasing ability to make precise predictions. The logarithmic loss significantly decreased to an exceptionally low value at the 100th iteration, indicating that the model has successfully and precisely recognized occurrences within the dataset. The incremental increase shows how well the XGBoost approach enhances model performance and how well it can handle challenging predictive modeling jobs.

# Load necessary libraries  
library(xgboost)  
# Train the XGBoost model  
xgb\_model <- xgboost(data = X\_train, label = (y\_train) - 1, nrounds = 100, objective = "multi:softmax", num\_class = 3)

## [1] train-mlogloss:0.749250   
## [2] train-mlogloss:0.544720   
## [3] train-mlogloss:0.411360   
## [4] train-mlogloss:0.316362   
## [5] train-mlogloss:0.248717   
## [6] train-mlogloss:0.198651   
## [7] train-mlogloss:0.160796   
## [8] train-mlogloss:0.131517   
## [9] train-mlogloss:0.109109   
## [10] train-mlogloss:0.092007   
## [11] train-mlogloss:0.078787   
## [12] train-mlogloss:0.067756   
## [13] train-mlogloss:0.059858   
## [14] train-mlogloss:0.052471   
## [15] train-mlogloss:0.047231   
## [16] train-mlogloss:0.043225   
## [17] train-mlogloss:0.039278   
## [18] train-mlogloss:0.036447   
## [19] train-mlogloss:0.033627   
## [20] train-mlogloss:0.031555   
## [21] train-mlogloss:0.029619   
## [22] train-mlogloss:0.027094   
## [23] train-mlogloss:0.025245   
## [24] train-mlogloss:0.023958   
## [25] train-mlogloss:0.022035   
## [26] train-mlogloss:0.020653   
## [27] train-mlogloss:0.019022   
## [28] train-mlogloss:0.017971   
## [29] train-mlogloss:0.016839   
## [30] train-mlogloss:0.015957   
## [31] train-mlogloss:0.014989   
## [32] train-mlogloss:0.014288   
## [33] train-mlogloss:0.013628   
## [34] train-mlogloss:0.013109   
## [35] train-mlogloss:0.012439   
## [36] train-mlogloss:0.011884   
## [37] train-mlogloss:0.011403   
## [38] train-mlogloss:0.010977   
## [39] train-mlogloss:0.010586   
## [40] train-mlogloss:0.010236   
## [41] train-mlogloss:0.009868   
## [42] train-mlogloss:0.009526   
## [43] train-mlogloss:0.009280   
## [44] train-mlogloss:0.008983   
## [45] train-mlogloss:0.008699   
## [46] train-mlogloss:0.008445   
## [47] train-mlogloss:0.008237   
## [48] train-mlogloss:0.008042   
## [49] train-mlogloss:0.007816   
## [50] train-mlogloss:0.007624   
## [51] train-mlogloss:0.007461   
## [52] train-mlogloss:0.007307   
## [53] train-mlogloss:0.007145   
## [54] train-mlogloss:0.006998   
## [55] train-mlogloss:0.006857   
## [56] train-mlogloss:0.006734   
## [57] train-mlogloss:0.006614   
## [58] train-mlogloss:0.006497   
## [59] train-mlogloss:0.006402   
## [60] train-mlogloss:0.006286   
## [61] train-mlogloss:0.006188   
## [62] train-mlogloss:0.006072   
## [63] train-mlogloss:0.005989   
## [64] train-mlogloss:0.005901   
## [65] train-mlogloss:0.005823   
## [66] train-mlogloss:0.005735   
## [67] train-mlogloss:0.005667   
## [68] train-mlogloss:0.005586   
## [69] train-mlogloss:0.005513   
## [70] train-mlogloss:0.005450   
## [71] train-mlogloss:0.005391   
## [72] train-mlogloss:0.005329   
## [73] train-mlogloss:0.005269   
## [74] train-mlogloss:0.005218   
## [75] train-mlogloss:0.005153   
## [76] train-mlogloss:0.005101   
## [77] train-mlogloss:0.005039   
## [78] train-mlogloss:0.004994   
## [79] train-mlogloss:0.004943   
## [80] train-mlogloss:0.004886   
## [81] train-mlogloss:0.004845   
## [82] train-mlogloss:0.004796   
## [83] train-mlogloss:0.004747   
## [84] train-mlogloss:0.004710   
## [85] train-mlogloss:0.004672   
## [86] train-mlogloss:0.004630   
## [87] train-mlogloss:0.004591   
## [88] train-mlogloss:0.004548   
## [89] train-mlogloss:0.004510   
## [90] train-mlogloss:0.004469   
## [91] train-mlogloss:0.004438   
## [92] train-mlogloss:0.004409   
## [93] train-mlogloss:0.004379   
## [94] train-mlogloss:0.004343   
## [95] train-mlogloss:0.004318   
## [96] train-mlogloss:0.004286   
## [97] train-mlogloss:0.004259   
## [98] train-mlogloss:0.004231   
## [99] train-mlogloss:0.004207   
## [100] train-mlogloss:0.004182

xgb\_model

## ##### xgb.Booster  
## raw: 432.4 Kb   
## call:  
## xgb.train(params = params, data = dtrain, nrounds = nrounds,   
## watchlist = watchlist, verbose = verbose, print\_every\_n = print\_every\_n,   
## early\_stopping\_rounds = early\_stopping\_rounds, maximize = maximize,   
## save\_period = save\_period, save\_name = save\_name, xgb\_model = xgb\_model,   
## callbacks = callbacks, objective = "multi:softmax", num\_class = 3)  
## params (as set within xgb.train):  
## objective = "multi:softmax", num\_class = "3", validate\_parameters = "TRUE"  
## xgb.attributes:  
## niter  
## callbacks:  
## cb.print.evaluation(period = print\_every\_n)  
## cb.evaluation.log()  
## # of features: 21   
## niter: 100  
## nfeatures : 21   
## evaluation\_log:  
## iter train\_mlogloss  
## <num> <num>  
## 1 0.749249818  
## 2 0.544720274  
## ---   
## 99 0.004206591  
## 100 0.004181796

## Model Prediction and Performance

The confusion matrix provides information on the model’s classification overall performance by showing the wide variety of real high-quality, false high-quality, and fake terrible predictions for every elegance. The accuracy of the model changed to around 95.54%. Class 1 (suspect) has an extremely decreased sensitivity of 79.66% compared to Class Zero, with the highest sensitivity of 98.46% and 95.24 % for Class 2 (pathologic). The model’s specificity ratings of 88% and 98% demonstrate how well it recognizes occurrences that do not fall into either Class 1 or Class 2. The real negative rate for each class is known as specificity.

For every class, the positive predictive value ranges from 90% to 96% and shows the percentage that separates the model’s actual positive forecasts from its predictions.

Among the model’s negative predictions or negative predictive value, 94%-99% are actual negative forecasts.

The most prevalent class is 0, with 76.29% of all occurrences in the dataset.

Prevalence displays the proportion of occurrences in a particular class; Class 0 (419.45) has the highest detection rate, and Class 2 (405.71) is followed by Class 1 (339.36).

The average number of successfully identified instances per class is known as the detection rate; class 0 has the highest detection prevalence, at 77.93%.

The balanced accuracy calculates the average sensitivity and specificity across classes as a final indicator of the model’s overall efficacy and accuracy at 93.27%. Combined, these metrics provide a comprehensive evaluation of the model’s classification performance and accuracy in predicting the various fetal state categories in the dataset.

# Make predictions  
predictions <- predict(xgb\_model, X\_test)  
# Evaluate the model  
confusion\_matrix <- table(Actual = as.numeric(y\_test) - 1, Predicted = predictions)  
confusion\_matrix

## Predicted  
## Actual 0 1 2  
## 0 320 4 1  
## 1 11 47 1  
## 2 1 1 40

# Calculate statistics  
n <- sum(confusion\_matrix)  
n\_correct <- sum(diag(confusion\_matrix))  
n\_classes <- nrow(confusion\_matrix)  
# Calculate accuracy  
accuracy <- n\_correct / n  
accuracy

## [1] 0.9553991

# Calculate sensitivity for each class  
sensitivity <- diag(confusion\_matrix) / rowSums(confusion\_matrix, na.rm = TRUE)  
sensitivity

## 0 1 2   
## 0.9846154 0.7966102 0.9523810

# Calculate specificity for each class  
specificity <- rep(NA, n\_classes)  
for (i in 1:n\_classes) {  
 specificity[i] <- sum(confusion\_matrix[-i, -i]) / sum(confusion\_matrix[-i, ])  
}  
specificity

## [1] 0.8811881 0.9863760 0.9947917

# Calculate positive predictive value (PPV) for each class  
ppv <- diag(confusion\_matrix) / colSums(confusion\_matrix, na.rm = TRUE)  
ppv

## 0 1 2   
## 0.9638554 0.9038462 0.9523810

# Calculate negative predictive value (NPV) for each class  
npv <- rep(NA, n\_classes)  
for (i in 1:n\_classes) {  
 npv[i] <- sum(confusion\_matrix[-i, -i]) / sum(confusion\_matrix[, -i])  
}  
npv

## [1] 0.9468085 0.9679144 0.9947917

# Calculate prevalence for each class  
prevalence <- rowSums(confusion\_matrix) / n  
prevalence

## 0 1 2   
## 0.76291080 0.13849765 0.09859155

# Calculate detection rate (sensitivity) for each class  
detection\_rate <- diag(confusion\_matrix) / prevalence  
detection\_rate

## 0 1 2   
## 419.4462 339.3559 405.7143

# Calculate detection prevalence for each class  
detection\_prevalence <- colSums(confusion\_matrix) / n  
detection\_prevalence

## 0 1 2   
## 0.77934272 0.12206573 0.09859155

# Calculate balanced accuracy  
balanced\_accuracy <- mean(c(sensitivity, specificity), na.rm = TRUE)  
balanced\_accuracy

## [1] 0.9326604

## Importance

The features of the XGBoost model provide valuable information about the factors influencing the model’s prediction accuracy. The three most crucial elements to consider when assessing any feature are gain, coverage, and frequency. The gain displays the average contribution of each feature to raising the accuracy of the model throughout training. The rise in MSTV (Mean Short Term Variability) of around 16.85% is the main factor. It suggests that the modifications made to the MSTV have a significant impact on the scenario identification model’s accuracy. Following with an improvement of over 16.00% and 14.53%, respectively, ASTV (percentage of time with aberrant short-term variability) and mean value show their important contribution to model performance. Based on the observational coverage of the data set, cover determines the relative significance of each characteristic. The fact that ASTV has drawn the most attention demonstrates how frequent and significant it is to fully represent a dataset’s variety. Each attribute’s frequency denotes how often the file use it to build the decision tree. They are important because of the ubiquity of the umbilical cord (UC) and anomalous long-term variability (ALTV) in the differentiation of prenatal diseases. Our findings emphasize the significance of differentiating between hazardous, suspicious, and normal fetal states based on certain CTG traits; this categorization may help with clinical decision-making in obstetrics. Healthcare practitioners may enhance maternal health and pregnancy outcomes by selecting treatments and interventions according to the relative importance of these parameters.

importance <- xgb.importance (feature\_names = colnames(X\_test),model = xgb\_model)  
importance

## Feature Gain Cover Frequency  
## <char> <num> <num> <num>  
## 1: MSTV 0.168526549 0.052535578 0.033196240  
## 2: ASTV 0.160011332 0.127248827 0.113689777  
## 3: Mean 0.145273455 0.078577768 0.053760282  
## 4: ALTV 0.139719376 0.126368892 0.082549941  
## 5: DP 0.069394202 0.104684824 0.032608696  
## 6: AC 0.066252840 0.090638957 0.057579318  
## 7: UC 0.053051302 0.044671956 0.084312573  
## 8: Max 0.028551532 0.031901916 0.060810811  
## 9: Mode 0.025288618 0.047810703 0.037309048  
## 10: Median 0.021902518 0.035106203 0.045534665  
## 11: FM 0.021413403 0.050076818 0.051997650  
## 12: LB 0.020766462 0.037257711 0.068448884  
## 13: Min 0.016870703 0.034696290 0.045828437  
## 14: Nmax 0.016170907 0.022513024 0.036721504  
## 15: Width 0.012326434 0.029924948 0.057285546  
## 16: Variance 0.011733134 0.018529101 0.039659224  
## 17: MLTV 0.010113407 0.014738971 0.056698002  
## 18: DL 0.007017714 0.038560910 0.022032902  
## 19: Nzeros 0.002834161 0.008804575 0.005581669  
## 20: Tendency 0.002781951 0.005352028 0.014394830  
## Feature Gain Cover Frequency

xgb.plot.importance (importance\_matrix = importance)

