Cardiotocography Project Report

Rajashekar Vasanth

5th January 2020

Introduction

1. Introduction to Cardiotocography

Cardiotocography (CTG) is a technical means of recording the fetal heartbeat and the uterine contractions during pregnancy. The machine used to perform the monitoring is called a cardiotocograph, more commonly known as an electronic fetal monitor (EFM).

Fetal monitoring was invented by Doctors Alan Bradfield, Orvan Hess and Edward Hon. A refined (antepartal, non-invasive, beat-to-beat) version (cardiotocograph) was later developed for Hewlett Packard by Konrad Hammacher.

The basic concepts of CTG

Cardiotocography (CTG) is a simultaneous recording of Fetal Heart Rate (FHR) and Uterine Contractions (UC) and it is one of the most common diagnostic techniques to evaluate maternal and fetal well-being during pregnancy and before delivery. By observing the Cardiotocography trace patterns doctors can understand the state of the fetus. There are several signal processing and computer programming based techniques for interpreting a typical CTG data. Even a few decades after the introduction of Cardiotocography in clinical practice, the predictive capacity of these methods remains controversial and still inaccurate. FHR patterns are observed manually by obstetricians during the process of CTG analyses. For the last three decades, great interest has been paid to the fetal heart rate baseline and its frequency analysis, Fetal Heart Rate (FHR) monitoring remains as a widely used method for detecting changes in fetal oxygenation that can occur during labor.

Yet deaths and long-term disablement from intrapartum hypoxia remain an important cause of suffering for parents and families, even in industrialized countries. Confidential inquiries have highlighted that as much as 50% of these deaths could have been avoided because they were caused by non-recognition of abnormal FHR patterns, poor communication between staff, or delay in taking appropriate action. Computation and other data mining techniques can be used to analyze and classify the CTG data to avoid human mistakes and to assist doctors to take a decision.

Dataset Information

2126 fetal cardiotocograms (CTGs) were automatically processed and the respective diagnostic features measured. The CTGs were also classified by three expert obstetricians and a consensus classification label assigned to each of them. Classification was both with respect to the fetal state (N, S, P). Therefore the dataset can be used for 3-class experiments.

Classification Variables

Updated 2015 FIGO(https://www.figo.org/about-us) Intrapartum Fetal Monitoring Guidelines - these are the metrics that are currently used by obstetricians to evaluate and monitor. Let us see how accurately we can use algorithms to learn the same and detect abnormalities.

FIGO has recently modified the guidelines on intrapartum fetal monitoring, proposing following interpretation:

1. Normal No hypoxia/acidosis, no intervention necessary to improve fetal oxygenation state:

- Baseline 110-160 bpm
- Variability 5-25 bpm
- No repetitive decelerations (decelerations are defined as repetitive when associated with > 50% contractions)

2. Suspicious Low probability of hypoxia/acidosis, warrants action to correct reversible causes if identified, close monitoring or adjunctive methods:

- Lacking at least one characteristic of normality, but with no pathological features.

3. Pathological High probability of hypoxia/acidosis, requires immediate action to correct reversible causes, adjunctive methods, or if this is not possible expedite delivery. In acute situations immediate delivery should be accomplished

- Baseline <100 bpm
- Reduced or increased variability or sinusoidal pattern
- Repetitive late or prolonged decelerations for > 30 min, or > 20 min if reduced variability (decelerations are defined as repetitive when associated with > 50% contractions)
- Deceleration > 5 min

Importing the required libraries

Writing a function to install and import libraries.

```
#Writing a function to install and import libraries.
fn_install_packages <- function(library_name) {
#Check if the library is present, if not install it
   if(!require(library_name,character.only=TRUE))
     install.packages(library_name,character.only=TRUE,
                      repos = "http://cran.us.r-project.org")

#Import the library
   library(library_name,character.only=TRUE)
}
```

Creating a list of libraries required and importing all required libraries

```
#Defining a list of required libraries
library_list <- c("ggplot2","reshape2","elasticnet","brnn","pls",
                  "ipred","plyr","e1071","xgboost","knitr","reshape",
                  "gam","kernlab","randomForest","tidyverse","caret","data.table",
                 "matrixStats","corrplot","rpart","nnet","htmltools","klaR",
                 "mda","RSNNS","adabag")

#Installing and importing the libraries
for(lib in library_list){
  fn_install_packages(lib)
  }
```

Importing the data file from GIT and looking the structure of the imported data. We see that the data consists of 2126 rows and 25 columns. All the columns are either of the type INT or NUM. This makes cleansing slightly easier. We see that the first column is an exact copy of the second. Let us confirm that by conducting a test.

```
#Reading the datafile from GITHUB
ctg <- read.table(
  "https://raw.githubusercontent.com/rajashekarv95/cardiotocography_edx_capstone/master/ctg_new.csv",
  header = TRUE,sep = ",")

#Examining the structure of the datafile
str(ctg)
```

```
## 'data.frame':    2126 obs. of  25 variables:
##  $ ï..LBE  : int  120 132 133 134 132 134 134 122 122 122 ...
##  $ LB      : int  120 132 133 134 132 134 134 122 122 122 ...
##  $ AC      : int  0 4 2 2 4 1 1 0 0 0 ...
##  $ FM      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ UC      : int  0 4 5 6 5 10 9 0 1 3 ...
##  $ ASTV    : int  73 17 16 16 16 26 29 83 84 86 ...
##  $ MSTV    : num  0.5 2.1 2.1 2.4 2.4 5.9 6.3 0.5 0.5 0.3 ...
##  $ ALTV    : int  43 0 0 0 0 0 0 6 5 6 ...
##  $ MLTV    : num  2.4 10.4 13.4 23 19.9 0 0 15.6 13.6 10.6 ...
##  $ DL      : int  0 2 2 2 0 9 6 0 0 0 ...
##  $ DS      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ DP      : int  0 0 0 0 0 2 2 0 0 0 ...
##  $ DR      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Width   : int  64 130 130 117 117 150 150 68 68 68 ...
##  $ Min     : int  62 68 68 53 53 50 50 62 62 62 ...
##  $ Max     : int  126 198 198 170 170 200 200 130 130 130 ...
##  $ Nmax    : int  2 6 5 11 9 5 6 0 0 1 ...
##  $ Nzeros  : int  0 1 1 0 0 3 3 0 0 0 ...
##  $ Mode    : int  120 141 141 137 137 76 71 122 122 122 ...
##  $ Mean    : int  137 136 135 134 136 107 107 122 122 122 ...
##  $ Median  : int  121 140 138 137 138 107 106 123 123 123 ...
##  $ Variance: int  73 12 13 13 11 170 215 3 3 1 ...
##  $ Tendency: int  1 0 0 1 1 0 0 1 1 1 ...
##  $ CLASS   : int  9 6 6 6 2 8 8 9 9 9 ...
##  $ NSP     : int  2 1 1 1 1 3 3 3 3 3 ...
```

Checking if the first and second columns contain the same values. If yes, we can eliminate one of them.

By writing the below code, we can be sure that the first two columns are indeed the same. We will be removing one of these later when we look to cleanse the data.

```
#Checking if the first and second columns contain the same values.
sum(ctg$LB != ctg$ï..LBE)
```

```
## [1] 0
```

Let us download the description file to see what each column means. We see below the descriptions of each columns. We see that there are two classification columns - CLASS and NSP. For this project, we will be using NSP which categorizes the fetuses as Normal, Suspect or Pathologic.

```r
#Reading the description file - provides description of each columns
ctg_description <- read.table(
  "https://raw.githubusercontent.com/rajashekarv95/cardiotocography_edx_capstone/master/ctg_description
  header = TRUE,sep = ",")

#Examining the description file
kable(ctg_description)
```

ï..Column.Name	What.they.stand.for.
LB	FHR baseline (beats per minute)
AC	Number of accelerations per second
FM	Number of fetal movements per second
UC	Number of uterine contractions per second
DL	Number of light decelerations per second
DS	Number of severe decelerations per second
DP	Number of prolongued decelerations per second
ASTV	Percentage of time with abnormal short term variability
MSTV	Mean value of short term variability
ALTV	Percentage of time with abnormal long term variability
MLTV	Mean value of long term variability
Width	Width of FHR histogram
Min	Minimum of FHR histogram
Max	Maximum of FHR histogram
Nmax	Number of histogram peaks
Nzeros	Number of histogram zeros
Mode	Histogram mode
Mean	Histogram mean
Median	Histogram median
Variance	Histogram variance
Tendency	Histogram tendency
CLASS	FHR pattern class code (1 to 10)
NSP	Fetal state class code (N=normal; S=suspect; P=pathologic)

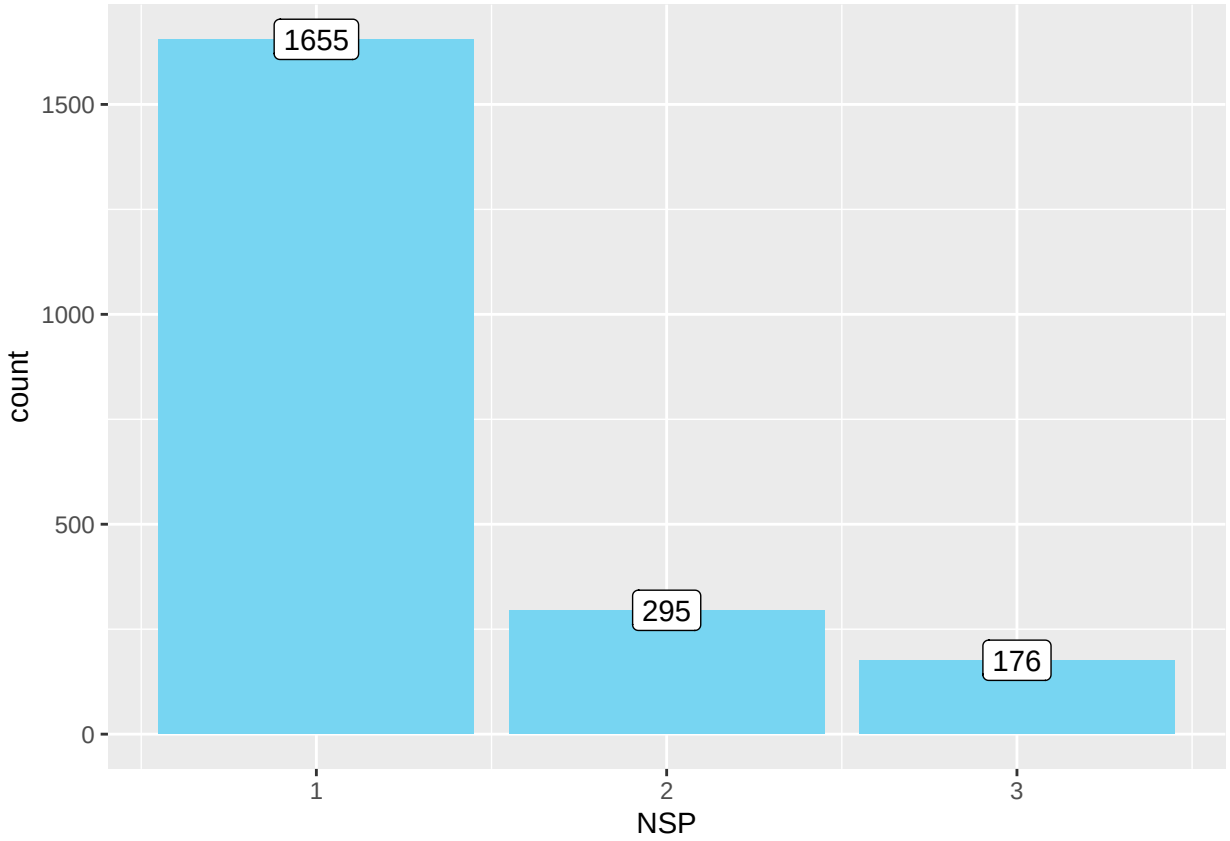
This data set can be used for a 10 class as well as a 3 class classification problem. Since the volume is very less to perform a 10 class classification, I am here considering only the 3 class variable.

The column is called NSP and it contains 3 values.

1 - Normal(N) 2 - Suspect(S) 3 - Pathologic(P)

Let us see how the data is distributed across these three values. We see from the plot below that there is an imbalance. A greater percentage of the fetuses are normal with a few of them either suspect/pathologic.

```r
ctg %>%
  group_by(NSP) %>% summarize(count = n()) %>%
  ggplot(aes(NSP,count,label=count)) +
  geom_bar(stat = "identity", position = "dodge",fill="#77d5f2") +
  geom_label()
```



Looking at the percentage of data for each of these values, we see that almost 80% of the fetuses under consideration are normal.

As this is an imbalanced dataset, we cannot use accuracy as a metric of evaluation. We would be looking at Confusion Matrix, Precision, Recall and F1 score to evaluate the performance of the models we develop here.

```
num_of_datapoints <- length(ctg$LB)
ctg %>% group_by(NSP) %>%
  summarize(count = n(), percentage = n()/num_of_datapoints)
```

```
## # A tibble: 3 x 3
##     NSP count percentage
##   <int> <int>      <dbl>
## 1     1  1655     0.778
## 2     2   295     0.139
## 3     3   176     0.0828
```

Data Cleansing

Removing unwanted columns and separating the feature columns and prediction columns.

1. We remove ï..LBE as discussed previously that this is a duplicated column.
2. We remove DR as this column is 0 for all rows.
3. We remove CLASS as this is a 10 classification column and we will not be using it.

Post this, the dataset is split into two dataframes - one containing the feature columns and the other prediction column.

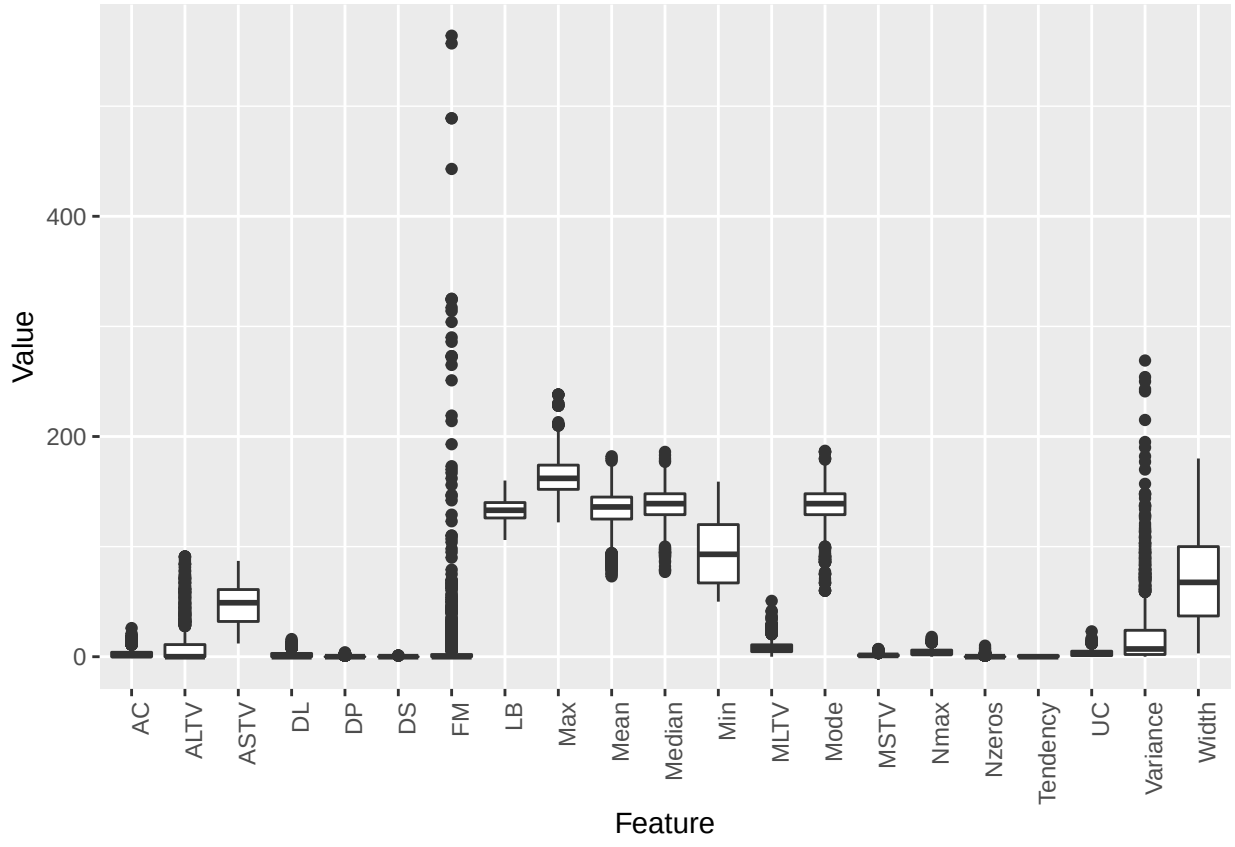
```
#Removing unwanted columns
ctg <- ctg %>% dplyr :: select(-ï..LBE)
ctg <- ctg %>% dplyr :: select(-DR)
ctg <- ctg %>% dplyr :: select(-CLASS)

#Splitting the feature and decision columns
y <- ctg %>% dplyr :: select(NSP)
ctg <- ctg %>% dplyr :: select(-NSP)
```

Converting the feature dataframe matrix and checking its dimension.

```
#Converting to matrix

ctg <- as.matrix(ctg)
dim(ctg)
```

```
## [1] 2126   21
```

```
ctg_melted_1 <- reshape :: melt(ctg)
ggplot(data = ctg_melted_1, aes(x= X2,y = value)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(y="Value", x = "Feature")
```



As each feature is of a different scale, we perform standardization so that they are centered around 0. Scaling the feature variables and displaying the resulting dataframe

```
#Scaling the feature variables
x_centered <- sweep(ctg, 2, colMeans(ctg))
x_scaled <- sweep(x_centered, 2, matrixStats :: colSds(ctg), FUN = "/")

summary(x_scaled)
```

```
##        LB                 AC                FM                UC
##  Min.   :-2.77454   Min.   :-0.7646   Min.   :-0.1951   Min.   :-1.2855
##  1st Qu.:-0.74220   1st Qu.:-0.7646   1st Qu.:-0.1951   1st Qu.:-0.9343
##  Median :-0.03088   Median :-0.4837   Median :-0.1951   Median :-0.2318
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.68044   3rd Qu.: 0.3588   3rd Qu.:-0.1412   3rd Qu.: 0.4707
##  Max.   : 2.71279   Max.   : 6.5371   Max.   :14.9967   Max.   : 6.7929
##       ASTV               MSTV              ALTV               MLTV
##  Min.   :-2.0352   Min.   :-1.2825   Min.   :-0.53524   Min.   :-1.4547
##  1st Qu.:-0.8719   1st Qu.:-0.7164   1st Qu.:-0.53524   1st Qu.:-0.6374
##  Median : 0.1169   Median :-0.1503   Median :-0.53524   Median :-0.1399
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000
##  3rd Qu.: 0.8149   3rd Qu.: 0.4158   3rd Qu.: 0.06269   3rd Qu.: 0.4642
##  Max.   : 2.3271   Max.   : 6.4164   Max.   : 4.41126   Max.   : 7.5534
##        DL                 DS                DP              Width
##  Min.   :-0.6282   Min.   :-0.05746   Min.   :-0.2715   Min.   :-1.73135
##  1st Qu.:-0.6282   1st Qu.:-0.05746   1st Qu.:-0.2715   1st Qu.:-0.85856
##  Median :-0.6282   Median :-0.05746   Median :-0.2715   Median :-0.07562
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.00000
##  3rd Qu.: 0.5721   3rd Qu.:-0.05746   3rd Qu.:-0.2715   3rd Qu.: 0.75866
##  Max.   : 5.7737   Max.   :17.39459   Max.   : 8.3425   Max.   : 2.81227
##       Min                Max               Nmax             Nzeros
##  Min.   :-1.4743   Min.   :-2.3420   Min.   :-1.3793   Min.   :-0.4583
##  1st Qu.:-0.8992   1st Qu.:-0.6702   1st Qu.:-0.7012   1st Qu.:-0.4583
##  Median :-0.0196   Median :-0.1129   Median :-0.3622   Median :-0.4583
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.8938   3rd Qu.: 0.5559   3rd Qu.: 0.6550   3rd Qu.:-0.4583
##  Max.   : 2.2131   Max.   : 4.1225   Max.   : 4.7236   Max.   :13.7048
##       Mode               Mean              Median            Variance
##  Min.   :-4.7281   Min.   :-3.95101   Min.   :-4.22286   Min.   :-0.6491
##  1st Qu.:-0.5160   1st Qu.:-0.61631   1st Qu.:-0.62837   1st Qu.:-0.5800
##  Median : 0.0945   Median : 0.08911   Median : 0.06288   Median :-0.4075
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.0000
##  3rd Qu.: 0.6439   3rd Qu.: 0.66626   3rd Qu.: 0.68500   3rd Qu.: 0.1792
##  Max.   : 3.0247   Max.   : 3.03903   Max.   : 3.31175   Max.   : 8.6340
##     Tendency
##  Min.   :-2.1615
##  1st Qu.:-0.5244
##  Median :-0.5244
##  Mean   : 0.0000
##  3rd Qu.: 1.1127
##  Max.   : 1.1127
```

Create Train and Test sets

We now create train and test datasets. Training set will have 80% of the data and testing the rest.

```
#Create train and validation dataset
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
test_index <- createDataPartition(y = y$NSP, times = 1, p = 0.2, list = FALSE)
x_train <- x_scaled[-test_index,]
x_test <- x_scaled[test_index,]
y_train <- y$NSP[-test_index]
y_test <- y$NSP[test_index]

#Converting the result variables into alphabetical from numeric
y_train_alpha <- if_else(y_train == 1,"N",if_else(y_train==2,"S","P"))
y_test_alpha <- if_else(y_test == 1,"N",if_else(y_test ==2,"S","P"))
```

Examine the train and test data created to ensure the same proportion

```
#Check if the split has occured proportionally
table(y_train_alpha)/length(y_train_alpha)
```

```
## y_train_alpha
##          N          P          S
## 0.77941176 0.08352941 0.13705882
```

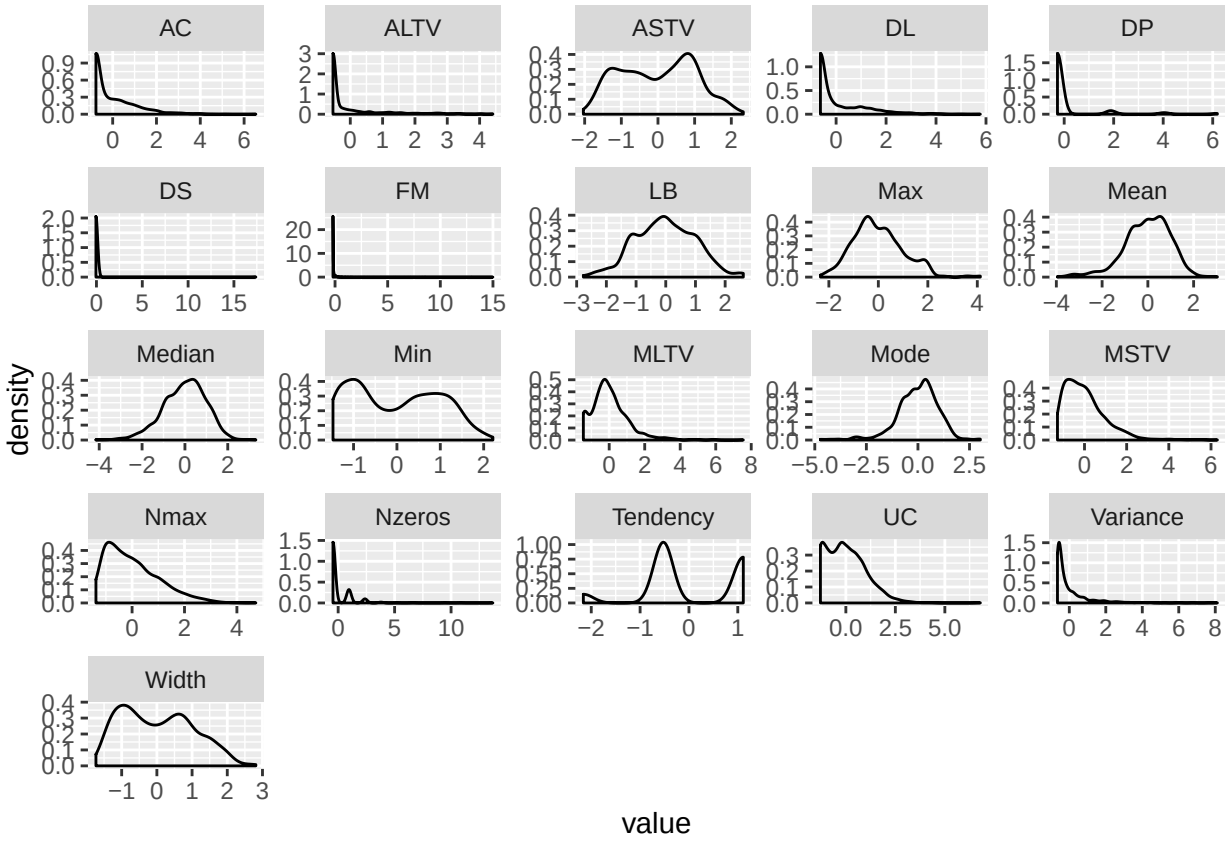
```
table(y_test_alpha)/length(y_test_alpha)
```

```
## y_test_alpha
##          N          P          S
## 0.77464789 0.07981221 0.14553991
```

Exploratory Data Analysis

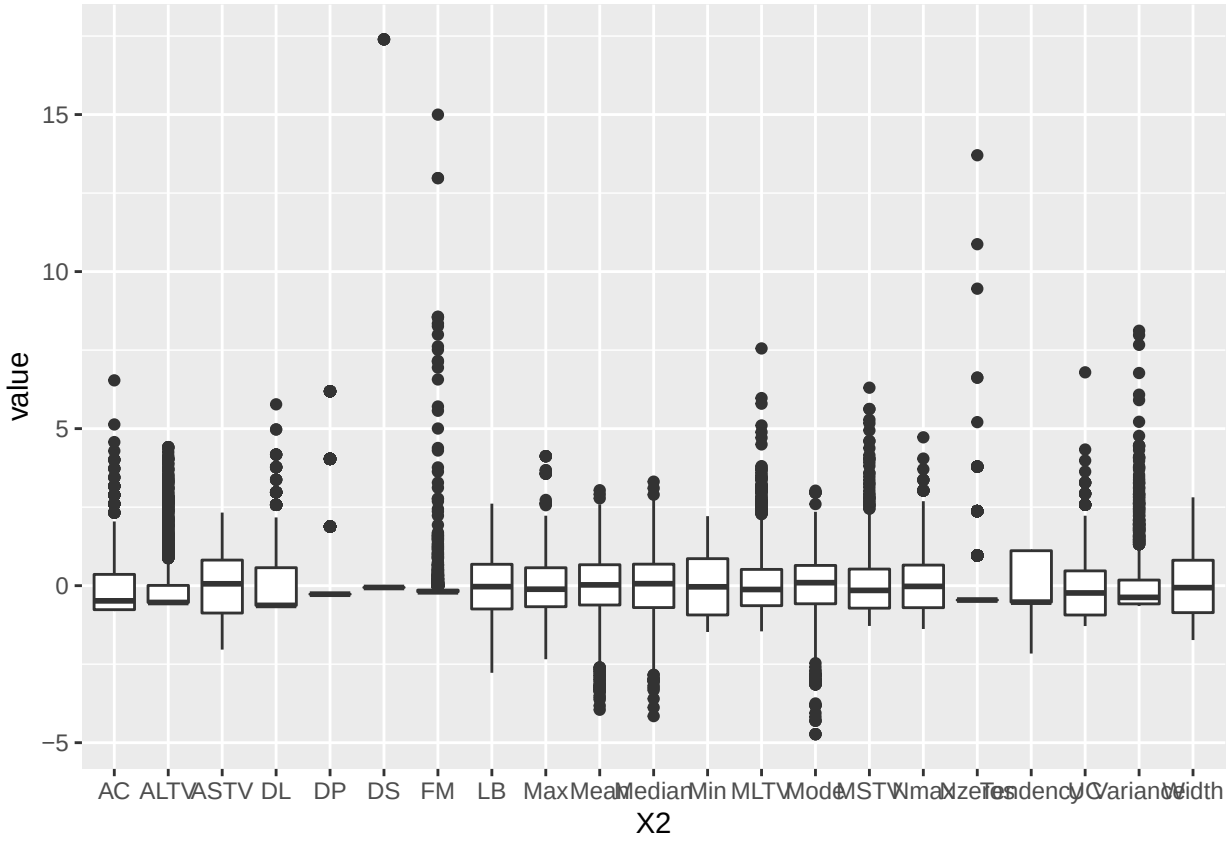
Examine the dsitributions of different feature variables.

```
#Melting the dataframe before plotting the distributions.
ctg_melted <- reshape::melt(x_train)

ggplot(data = ctg_melted, aes(x = value)) +
  geom_density() +
  facet_wrap(~X2, scales = "free")
```



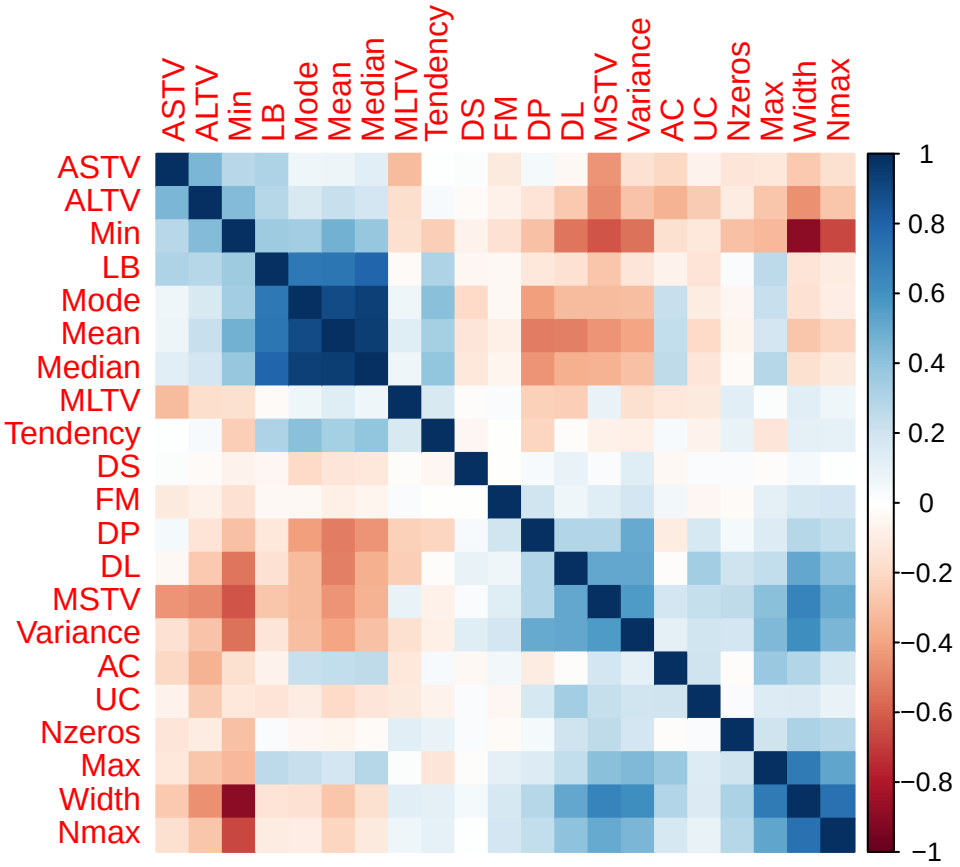
Let us look at the same on a boxplot.

```
ggplot(data = ctg_melted, aes(x= X2,y = value)) + geom_boxplot()
```



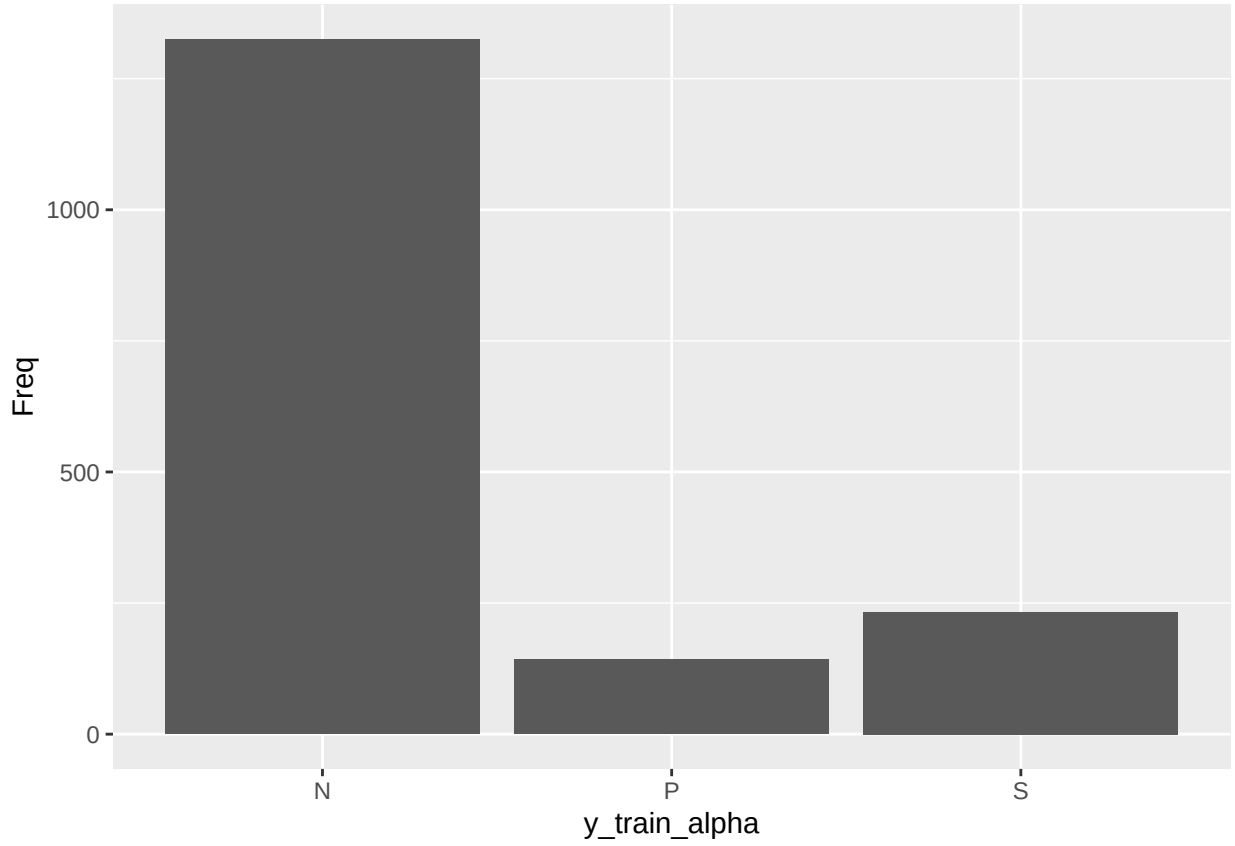
Let us now find the correlations between all the different feature valriables and plot them as a heatmap. I am using the corrplot to plot the correlations.

```
# calculate a correlation matrix for numeric variables
correlations <- cor(x_train)
library(corrplot)
corrplot(correlations, order = "hclust",method = "color")
```



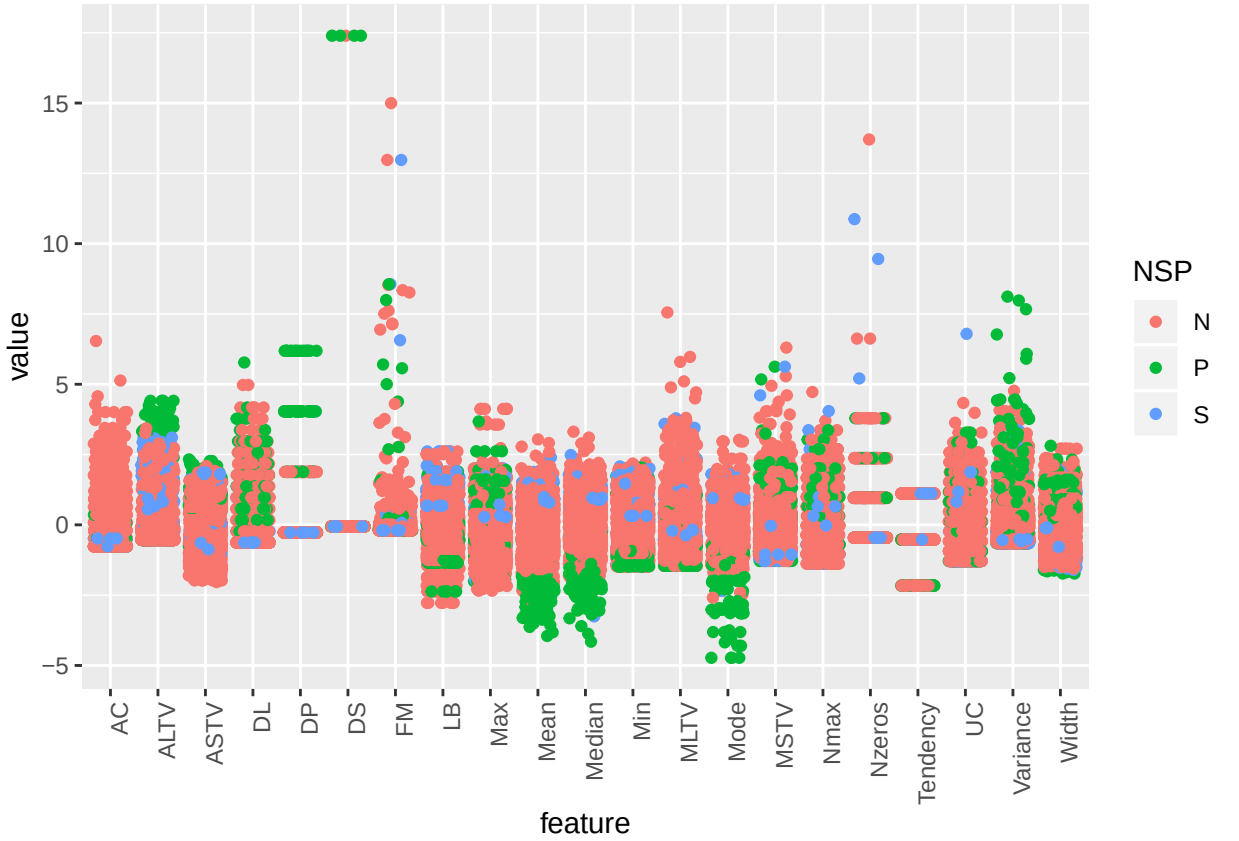
Looking at distribution

```
as.data.frame(table(y_train_alpha)) %>%
  ggplot(aes(x= y_train_alpha, y = Freq)) +
  geom_bar(stat = "identity")
```



Looking at jitter plot of the distribution of all features. We can see there is somewhat clear demarcation between the different classification variables. Mean, mode, median, ALTV, MSTV all have a clear demarcation. Let us see these are the variables which are of most importance to classification algorithms.

```
df_jitter <- as.data.frame(x_train)
df_jitter$NSP <- y_train_alpha
df_jitter <- tidyr::gather(df_jitter,"feature","value",1:21)

df_jitter %>% ggplot(aes(x=feature,y=value,color=NSP)) +
  geom_jitter(position=position_jitter(0.35)) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



Let us now define a list of models to train the dataset on and train it for each of them. We look at the training accuracies and decide which model to use for prediction.

We earlier decided to use metrics(Sensitivity, Specificity, Precision and Recall) other than accuracy to choose which model we want to finalize on. There is no default function currently which does this on the training set. We just get the trained accuracy from the caret trained model and not the other metrics we need. In order to find the other metrics and ensure that we don't use the test dataset for this, we further divide the train dataset into 80:20, 80% of which we use for training and the rest 20% to calculate the abovesaid metrics.

We further split the train data to calculate the metrics described above.

```
#Create interim train and test set using the train set to calculate the metrics required.
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
test_index_new <- createDataPartition(y = y_train_alpha, times = 1, p = 0.2, list = FALSE)

x_train_new <- x_train[-test_index_new,]
x_test_new <- x_train[test_index_new,]
y_train_new <- y_train_alpha[-test_index_new]
y_test_new <- y_train_alpha[test_index_new]
```

Model Training

We train the models on the new train data we have created, predict using the interim test data and record the necessary metrics.

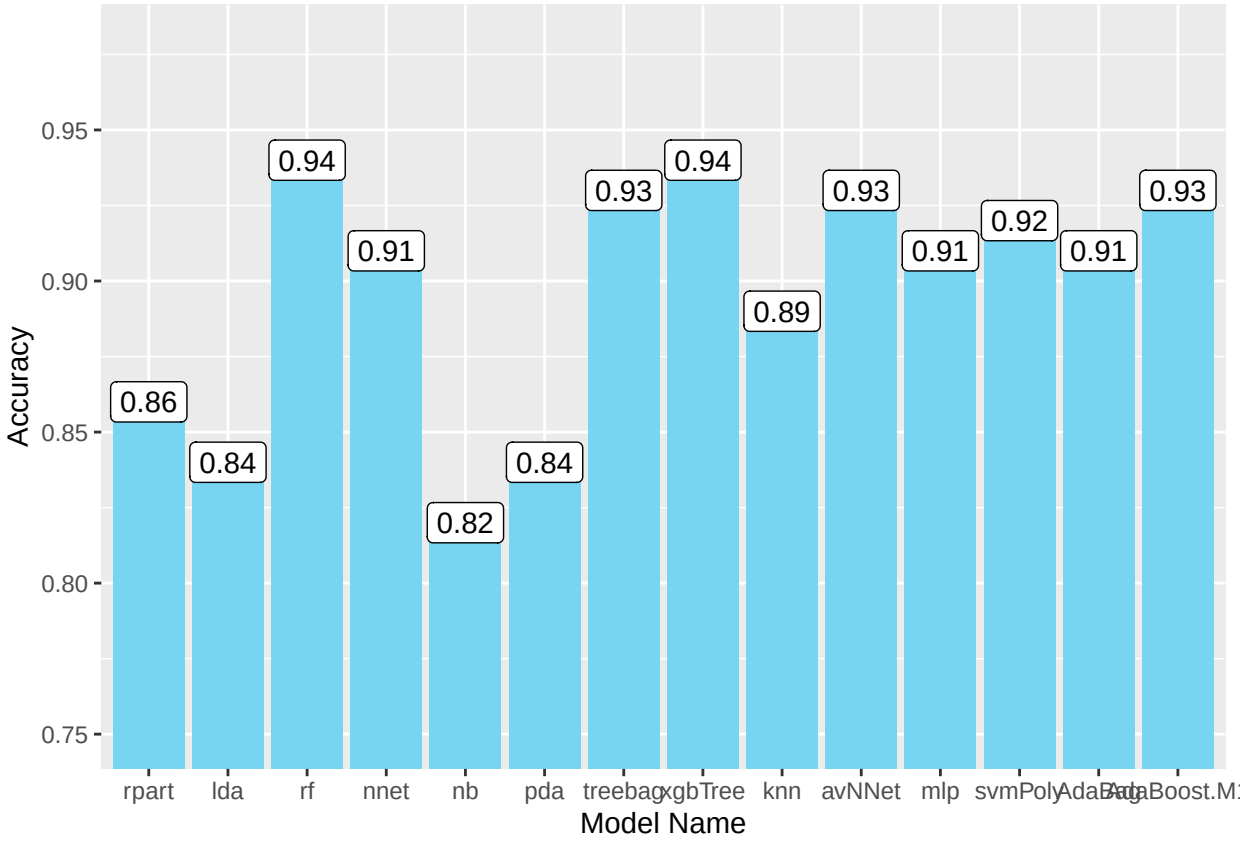
```
#Define model list

#model_list <- c("rpart")
#model_list <- c("rf","rpart","knn","nb","pda")
model_list <- c("rpart","lda","rf","nnet","nb",
                "pda","treebag","xgbTree","knn","avNNet","mlp",
               "svmPoly","AdaBag","AdaBoost.M1")

#Define a data frame to store the models and its evaluation metrics
df_models <- data.frame(model_name=character(),
                        train_accuracy=double(),accuracy=double(),F1_A=double(),
                        F1_B=double(),F1_C=double(),
                        precision_N=double(),precision_S=double(),precision_S=double(),
                        recall_N=double(),recall_S=double(),recall_P=double(),
                        specificity_N=double(),specificity_S=double(),specificity_P=double(),
                        sensitivity_N=double(),sensitivity_S=double(),sensitivity_P=double(),
                        stringsAsFactors=FALSE)

pred_list <- vector()
i <- 1
for(model in model_list){
    train_control <- trainControl(method="cv")
    model_train <- caret :: train(x_train_new,as.factor(y_train_new),
                         method = model, trControl = train_control)
    train_accuracy <- max(model_train$results$Accuracy)
    pred <- predict(model_train, x_test_new)
    pred_list[i] <- list(pred)
    i <- i + 1
    print(model)
    conf_mat <- caret :: confusionMatrix(data = pred, reference = as.factor(y_test_new))

    accuracy <- conf_mat$overall["Accuracy"]

    F1_N <- conf_mat$byClass["Class: N","F1"]
    F1_S <- conf_mat$byClass["Class: S","F1"]
    F1_P <- conf_mat$byClass["Class: P","F1"]

    precision_N <- conf_mat$byClass["Class: N","Precision"]
    precision_S <- conf_mat$byClass["Class: S","Precision"]
    precision_P <- conf_mat$byClass["Class: P","Precision"]

    recall_N <- conf_mat$byClass["Class: N","Recall"]
    recall_S <- conf_mat$byClass["Class: S","Recall"]
    recall_P <- conf_mat$byClass["Class: P","Recall"]

    specificity_N <- conf_mat$byClass["Class: N","Specificity"]
    specificity_S <- conf_mat$byClass["Class: S","Specificity"]
    specificity_P <- conf_mat$byClass["Class: P","Specificity"]
```

```
    sensitivity_N <- conf_mat$byClass["Class: N","Sensitivity"]
    sensitivity_S <- conf_mat$byClass["Class: S","Sensitivity"]
    sensitivity_P <- conf_mat$byClass["Class: P","Sensitivity"]

    new_row <- data.frame(model_name = model, train_accuracy = train_accuracy,accuracy = accuracy,
                          F1_N = F1_N,F1_S = F1_S,F1_P = F1_P,
                          precision_N = precision_N,precision_S = precision_S,
                          precision_P = precision_P,
                          recall_N = recall_N,recall_S = recall_S,recall_P = recall_P,
                          specificity_N = specificity_N ,specificity_S = specificity_S ,
                          specificity_P = specificity_P ,
                          sensitivity_N = sensitivity_N ,sensitivity_S = sensitivity_S ,
                          sensitivity_P = sensitivity_P
    )
    df_models <- rbind(df_models,new_row)
}
```

Checking the accuracy on the intermediate test set.

```
  melt(df_models) %>% filter(variable == "accuracy") %>%
  ggplot(aes(model_name,round(value,2),label=round(value,2))) +
  geom_bar(stat = "identity", position = "dodge",fill="#77d5f2") +
  geom_label()+
  labs(y="Accuracy", x = "Model Name")+
  coord_cartesian(ylim = c(0.75, 0.98))
```

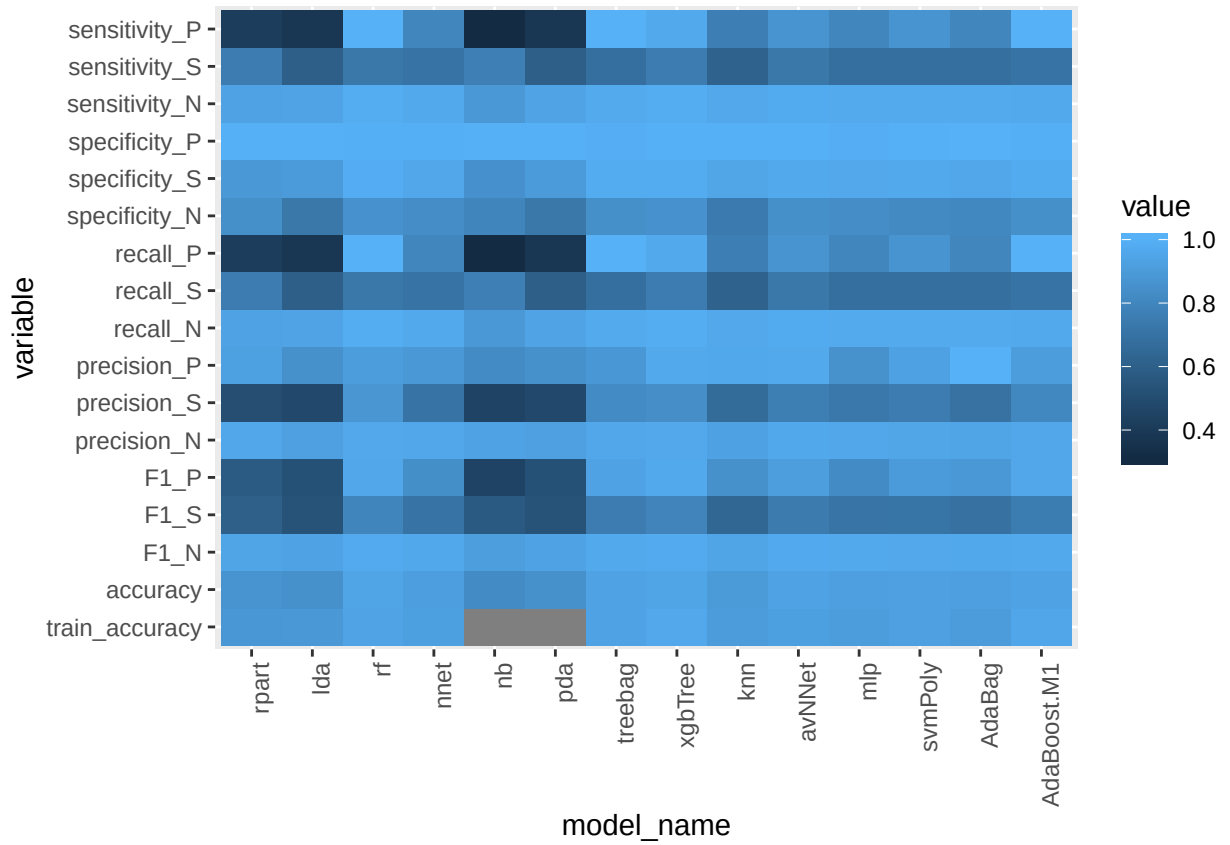
```
## Warning in melt(df_models): The melt generic in data.table has been passed
## a data.frame and will attempt to redirect to the relevant reshape2 method;
## please note that reshape2 is deprecated, and this redirection is now
## deprecated as well. To continue using melt methods from reshape2 while both
## libraries are attached, e.g. melt.list, you can prepend the namespace like
## reshape2::melt(df_models). In the next version, this warning will become an
## error.

## Using model_name as id variables
```



Examining the other metrics based on the intermediate test set(Note that this doesn't use the final test set. The final test set remains unknown).

```
melt(df_models) %>% ggplot(aes(model_name,variable)) +
 geom_tile(aes(fill =value))+
 theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

```
## Warning in melt(df_models): The melt generic in data.table has been passed
## a data.frame and will attempt to redirect to the relevant reshape2 method;
## please note that reshape2 is deprecated, and this redirection is now
## deprecated as well. To continue using melt methods from reshape2 while both
## libraries are attached, e.g. melt.list, you can prepend the namespace like
## reshape2::melt(df_models). In the next version, this warning will become an
## error.

## Using model_name as id variables
```



From the above tile chart we see that 'AdaBoost.M1' has the highest sensitivity for pathologic and suspect classification, meaning this most identifies the pathologic fetuses. It also has a high value of specificity which shows that this model also correctly identifies the healthy fetuses.

Conclusion and Results

Let us now combine the intermediate train and test sets and train the model again. We now run the prediction only for this model and on the unseen test set.

```
train_control <- trainControl(method="cv")
model_train <- caret :: train(x_train,as.factor(y_train_alpha),
                      method = "AdaBoost.M1", trControl = train_control)
```

Using the above model, let us predict the cases for the unseen test set and see how the confusion matrix looks like. We achieve a final accuracy of 92.49% on the test set. On observing the confusion matrix, we see that 3 of the pathologic fetuses are predicted as normal. Although this number seems to tbe very small, this is an area of concern and a base for further improvements.This model accurately detects 30 out of the 34 pathologic fetuses as pathologic which is a good measure to evaluate this algorithm.

Out of the 330 normal fetuses, this model detects 329 of them as normal or suspects and only 1 to be pathologic. This ensures that normal fetuses are not unnecessarily and incorrectly detected as pathologic.

These two evaluation metrics take into account for the bias in the distribution of the classification variable.

```
pred <- predict(model_train, x_test)
conf_mat <- caret :: confusionMatrix(data = pred, reference = as.factor(y_test_alpha))
conf_mat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   N   P   S
##          N 320   4  18
##          P   1  28   0
##          S   9   2  44
##
## Overall Statistics
##
##                Accuracy : 0.9202
##                  95% CI : (0.8903, 0.9441)
##     No Information Rate : 0.7746
##     P-Value [Acc > NIR] : 1.046e-15
##
##                   Kappa : 0.7745
##
##  Mcnemar's Test P-Value : 0.07855
##
## Statistics by Class:
##
##                      Class: N Class: P Class: S
## Sensitivity            0.9697  0.82353   0.7097
## Specificity            0.7708  0.99745   0.9698
## Pos Pred Value         0.9357  0.96552   0.8000
## Neg Pred Value         0.8810  0.98489   0.9515
## Prevalence             0.7746  0.07981   0.1455
## Detection Rate         0.7512  0.06573   0.1033
## Detection Prevalence   0.8028  0.06808   0.1291
## Balanced Accuracy      0.8703  0.91049   0.8397
```

Further Improvements

As a part of this project, I have tried to implement PCA as there were some correlated dimensions. That, however, did not yield acceptable results. The reason for this is still not clear. I would like to try out some more feature reduction techniques and compare the results obtained. I would also like to go deeper into working of PCA to determine why it did not work for this dataset.