# Cardiotocography and Machine Learning

Jason Holland

5/12/2020

## Introduction

Cardiotocography (CTG) measures several aspects of heart rate prior to and during birth. The CTG is most often employed when a pregnancy is at risk due to circumstances such as infection, bleeding, twins, or several other issues. A physician may use a CTG reading to make the decision to deliver a baby early based on signs of distress. For more information on CTG see
https://en.wikipedia.org/wiki/Cardiotocography.

### Data for Analysis

The data used in this paper was obtained from the UCI Machine Learning Repository at
https://archive.ics.uci.edu/ml/datasets/Cardiotocography. The raw data consists of 21 predictors and 2 possible responses. For this paper, the response variable studied is *NSP*. In the raw data *NSP* has three values; 1 for normal, 2 for suspect, and 3 for pathological. For our analysis, we use 0 for one class (normal) and 1 for the other class which is suspected abnormalities. In this way, we are using the data for a binary classification problem. Our task will then be to develop a model that can predict the class based on measurements taken from a CTG reading.

The predictor variables available include such measurements as beats per minute ($LB$), uterine contractions per second ($UC$), abnormal short term variability ($ASTV$) and several others. We include a table of each of these predictors.

```
##    Predictor              Description
## 1         LB           Beats per Min.
## 2         AC          Accel. per sec.
## 3         FM   Fetal Movement per sec.
## 4         UC Uterine contrac. per sec.
## 5         DL     Light Decel. per sec.
## 6         DS    Severe Decel. per sec.
## 7         DP Prolonged Decel. per sec.
## 8       ASTV      Abn. Short Term Var.
## 9       MSTV      Mean Short Term Var.
## 10      ALTV       Abn. Long Term Var.
## 11      MLTV       Mean Long Term Var.
## 12     Width               Hist. Width
## 13       Min                 Hist. Min
## 14       Max                 Hist. Max
## 15      Nmax              Hist. Peaks
## 16    Nzeros              Hist. Zeros
## 17      Mode               Hist. Mode
## 18      Mean               Hist. Mean
## 19    Median             Hist. Median
## 20  Variance               Hist. Var.
## 21  Tendency           Hist. Tendency
```

The goal of this analysis is to accurately predict the class based on the predictor variables using two common machine learning algorithms. The algorithms we employ are Logistic Regression and K-Nearest Neighbors or KNN. We closely follow techniques for developing models and computing model statistics found in *Introduction to Data Science* by Irizarry which can be found at https://rafalab.github.io/dsbook/. The key steps for achieving our goal will be to

1. use R to prepare the data for analysis,

2. use R to summarize and visualize the predictors,

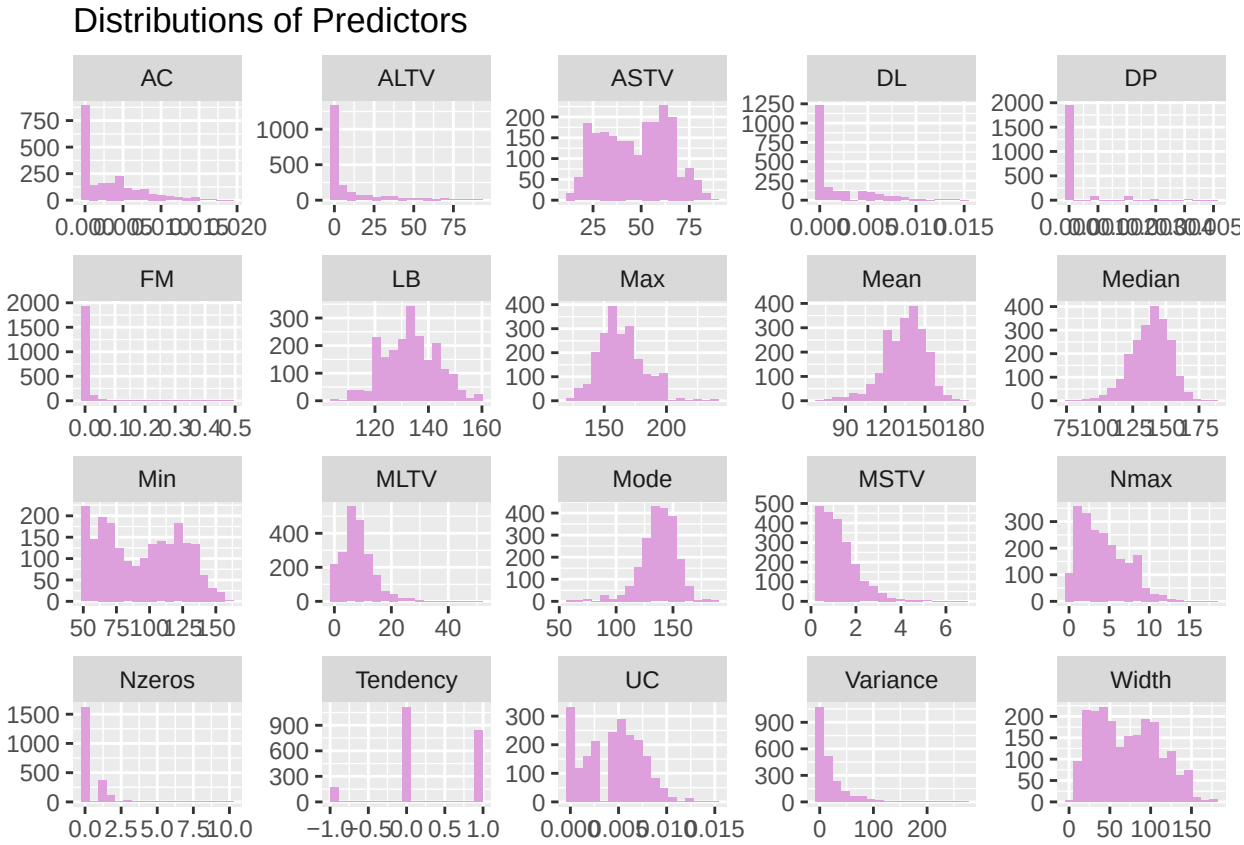
3. use R to run the models, and

4. use R to examine the results and quantify effectiveness.

## Analysis

### Data Preparation

A statistical summary of the data set is provided in the appendix. After loading the data set and all required packages into R, the data is read into R using the *readr* package and stored as a data frame. We remove three rows with missing data and the result is a data set with 2,126 rows containing individual CTG data. There are 22 columns with 21 predictors and the response variable *NSP*. We look at the distributions of 20 of the predictors.




We note that there are several predictors with good variability which gives us hope that we can accurately predict the response variable *NSP*. We remove the variables that have very little variability and we scale the columns by subtracting the mean and dividing by the standard deviation. This helps to remove bias caused

by one or more variables with values that are larger than other predictors. We illustrate by looking at seven columns and the first six rows of the resulting data frame.

```
##          Max       Nmax        Mode        Mean       Median   Variance NSP
## 1 -2.1190934 -0.7012319 -1.06536319  0.15323366 -1.181364219  1.8701287   1
## 2  1.8933489  0.6549828  0.21658719  0.08910477  0.132006901 -0.2349429   0
## 3  1.8933489  0.3159291  0.21658719  0.02497588 -0.006242691 -0.2004335   0
## 4  0.3329547  2.3502511 -0.02759383 -0.03915301 -0.075367487 -0.2004335   0
## 5  0.3329547  1.6721438 -0.02759383  0.08910477 -0.006242691 -0.2694523   0
## 6  2.0048057  0.3159291 -3.75135446 -1.77063300 -2.149111359  5.2175377   1
```

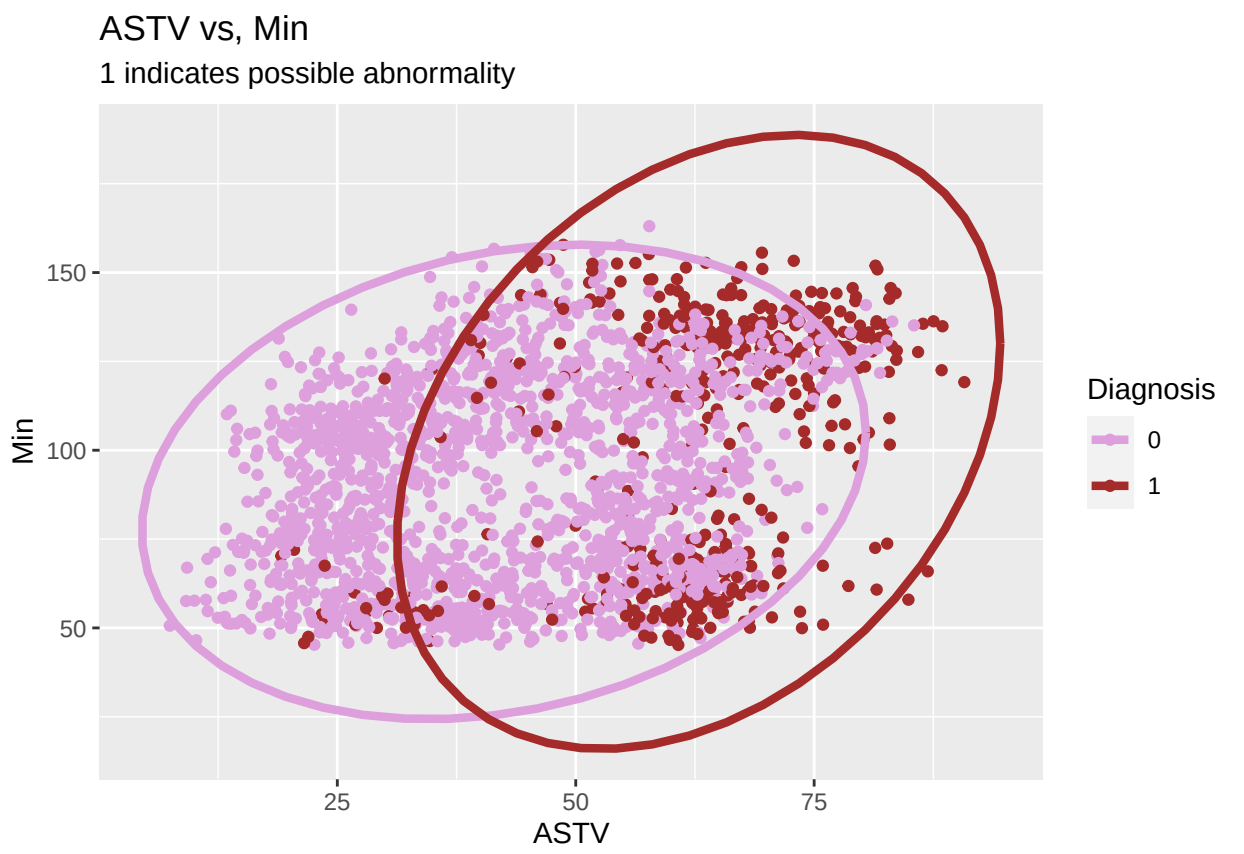
One question we have before looking at predictors is what is the prevalence of possible abnormalities ($\Pr(NSP) = 1$)? We see with the following code that it is about 22%.

```
prev <- mean(ctg$NSP == 1) # Compute the probability of NSP = 1 in the entire data set.
paste("Prevalence of possible abnormalities is ",100*round(prev,2),"%.")
```

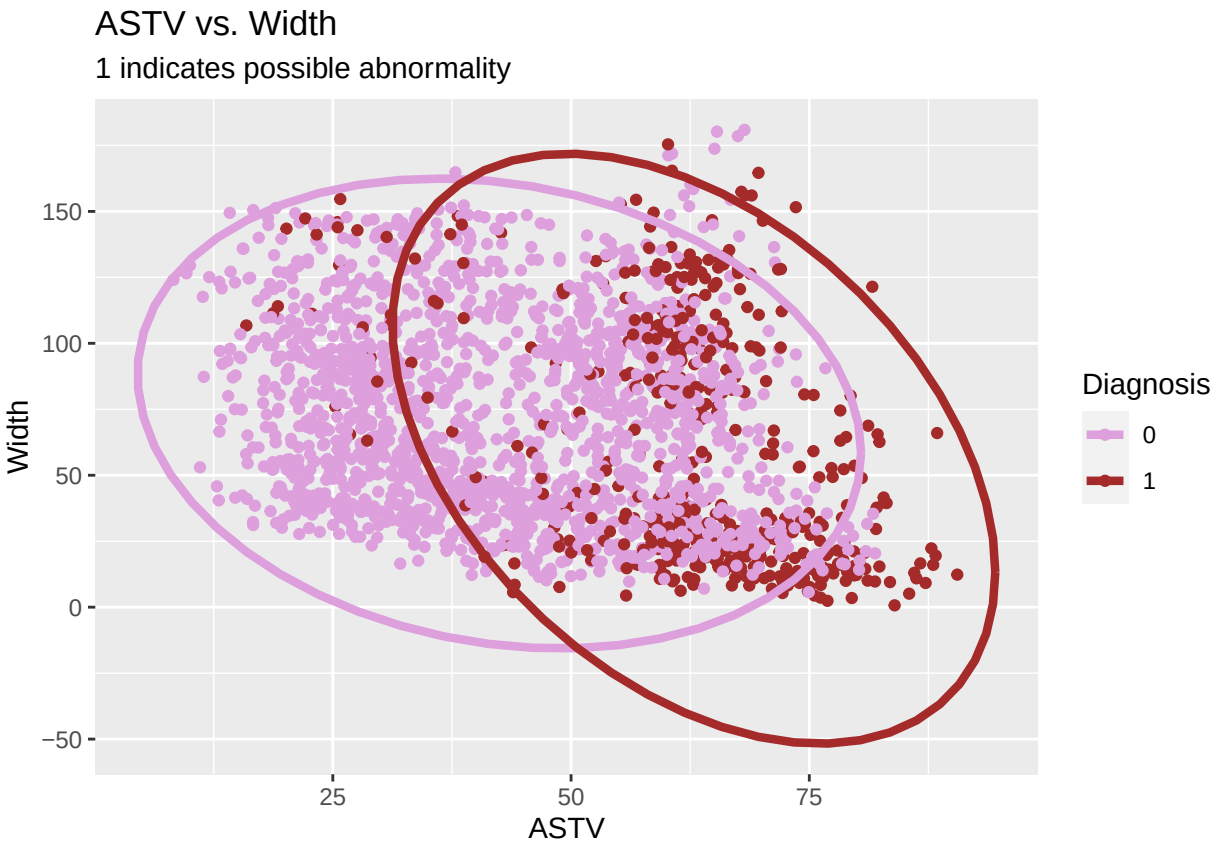
```
## [1] "Prevalence of possible abnormalities is  22 %."
```

This forces us to be careful in evaluating our models that we develop. We know that if we predict that every CTG reading in our data set can be classified as $NSP = 0$, we will be accurate 78% of the time. However, this is useless information to the doctor. We need to be able to predict when $NSP = 1$ as often as possible in order to have a useful model.

In analyzing the data, we examined many scatter plots, two predictors at a time colored by the response variable, to look for separations of the *NSP* variable. We include two examples in this paper. The histograms inform us about which variables might pair well together for the purpose of separating the response. One such pair is *ASTV* and *Min*. We use the original values of the predictor variables (before scaling) for these plots.



Another pair that had reasonable success separating *NSP* was *ASTV* and *Width*.




In looking at predictors with good variability and pairwise separability of the response variable, we gain insight into what predictors might contribute to making accurate predictions.

### Modeling Approach

We first divide the data into training and test sets. The training set will comprise 80% of the data and we will test our models on the remaining 20% of the data. For this task we use the partitioning functions in the widely used *caret* package of R. We include the code for developing the training and test sets at the end of this document in the appendix.

Our approach to developing prediction algorithms will be to start with a baseline model using Logistic Regression. Logistic Regression is a good model for binary classification. Given our exploration of variables that separate *NSP*, we opt for a first model using Logistic Regression with *ASTV* and *Min*. We will then investigate whether a Logistic Regression model with all predictors would improve on the first model. For a third and final model, we will use KNN with 5-fold cross validation *on the training set* as well as tuning the parameter k. We choose 5-fold cross validation based on the size of the training set. Our folds will have approximately 340 rows each which gives a reasonable potential number of *NSP's* with value equal to 1. Given the low prevalence of possible abnormalities, we will look at three metrics together; Sensitivity, Specificity, and Accuracy. We include the code for each model at the end of this document in the appendix.

---

## Results

### Model 1: Logistic Regression with Predictors *ASTV* and *Min*

For our first model, we establish a baseline using Logistic regression with the predictors *ASTV* and *Min*. The results of the three metrics Sensitivity, Specificity, and Accuracy are reported.

```
##                Result
## Sensitivity 0.9470405
## Specificity 0.3142857
## Accuracy    0.7910798
```

We see that Specificity is rather low which means we are unable to predict possible abnormalities when they are present. Accuracy is high due to the prevalence of $NSP = 0$. Hopefully we can improve on these results with models two and three.

### Model 2: Logistic Regression with all Predictors.

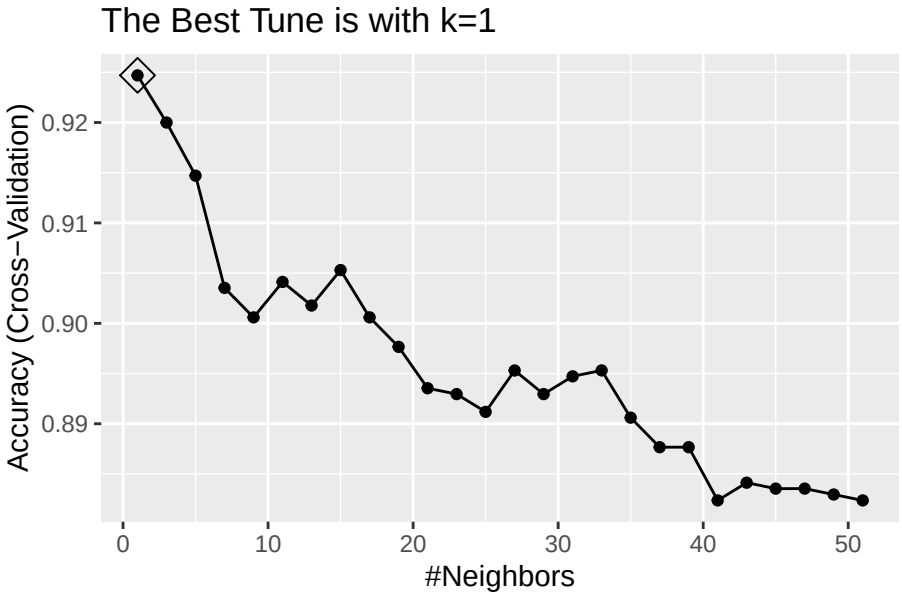
For Model 2, we run the same model using all predictors and look at the improvement in our three metrics.

```
##                Result
## Sensitivity 0.9345794
## Specificity 0.7428571
## Accuracy    0.8873239
```

We get improvement in accuracy and specificity with a decline in sensitivity. In spite of the decline in sensitivity, this is a better model than our first try. We have increased accuracy by 9% and specificity has increased by *more than 40%*! In hopes of advancing our metrics further, we try the knn model so that we can take advantage of cross validation and tuning features.

### Model 3: KNN with 5-fold Cross Validation and Tuning Parameter k

The following graph shows how overall accuracy of the knn model changes with the number of neighbors.



We achieve best results when the number of neighbors is equal to 1.

```
##                Result
## Sensitivity 0.9781931
## Specificity 0.7714286
## Accuracy    0.9272300
```

All three metrics of model 3 are higher than any of the previous models. We experimented with dropping out variables but were unable to improve our metrics. We choose to use model 3 to make predictions based on CTG readings.

## Conclusion

We summarize our three models below.

```
##             Model 1 Model 2 Model 3
## Sensitivity    0.95    0.93    0.98
## Specificity    0.31    0.74    0.77
## Accuracy       0.79    0.89    0.93
```

In our final model, knn with k = 1, Sensitivity is 98%. This means that if the patient has no possible abnormalities, then the model predicts no possible abnormalities 98% of the time. With Specificity equal to 77%, false positives (predicting possible abnormalities when there are none) will occur at a rate of about 23%. This may be an acceptable number since we are concerned with being "better safe than sorry." Ultimately, accuracy of the prediction is 93%.

We are limited by having only 2126 observations in our data set and then only 80% of those being used to develop our knn model. Given that it is expensive to obtain data related to health care, especially data involved with giving birth, this may be a large data set for the situation. If we had much more data, we could likely develop a more accurate model.

Future directions for this data set could include investigating decision tree models and random forest. These models might produce better metrics than those provided by the knn model.

---

## Appendix

### Statistical Summary of the Data Set

```
##        LB              AC                  UC                  DL
##  Min.   :106.0   Min.   :0.000000   Min.   :0.000000   Min.   :0.000000
##  1st Qu.:126.0   1st Qu.:0.000000   1st Qu.:0.002000   1st Qu.:0.000000
##  Median :133.0   Median :0.002000   Median :0.004000   Median :0.000000
##  Mean   :133.3   Mean   :0.003178   Mean   :0.004366   Mean   :0.001889
##  3rd Qu.:140.0   3rd Qu.:0.006000   3rd Qu.:0.007000   3rd Qu.:0.003000
##  Max.   :160.0   Max.   :0.019000   Max.   :0.015000   Max.   :0.015000
##       ASTV            MSTV             ALTV             MLTV
##  Min.   :12.00   Min.   :0.200   Min.   : 0.000   Min.   : 0.000
##  1st Qu.:32.00   1st Qu.:0.700   1st Qu.: 0.000   1st Qu.: 4.600
##  Median :49.00   Median :1.200   Median : 0.000   Median : 7.400
##  Mean   :46.99   Mean   :1.333   Mean   : 9.847   Mean   : 8.188
##  3rd Qu.:61.00   3rd Qu.:1.700   3rd Qu.:11.000   3rd Qu.:10.800
##  Max.   :87.00   Max.   :7.000   Max.   :91.000   Max.   :50.700
##      Width             Min              Max           Nmax
##  Min.   :  3.00   Min.   : 50.00   Min.   :122   Min.   : 0.000
##  1st Qu.: 37.00   1st Qu.: 67.00   1st Qu.:152   1st Qu.: 2.000
##  Median : 67.50   Median : 93.00   Median :162   Median : 3.000
##  Mean   : 70.45   Mean   : 93.58   Mean   :164   Mean   : 4.068
##  3rd Qu.:100.00   3rd Qu.:120.00   3rd Qu.:174   3rd Qu.: 6.000
##  Max.   :180.00   Max.   :159.00   Max.   :238   Max.   :18.000
##       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
##       NSP
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.2215
##  3rd Qu.:0.0000
##  Max.   :1.0000
```

### Code for Creating Train and Test Sets

```
# We remove variables with very little variance.
ctg <- ctg %>% select(-DP,-DS,-FM,-Nzeros,-Tendency)

################### We scale the columns
ctgs <- ctg[,1:16] # subset columns to be scaled
ctgs <- as.data.frame(scale(ctgs)) # convert to data frame and scale
ctgs$NSP <- ctg$NSP                # add back the response variable
head(ctgs[,11:17])
# Before modeling, we split the data into test and train sets.
set.seed(1965,sample.kind = "Rounding") # need to be able to reproduce results.
testIndex <- createDataPartition(ctgs$NSP,times = 1,p=.2,list = FALSE)
ctgTrain <- ctgs %>% slice(-testIndex)  # data for training
ctgTest <- ctgs %>% slice(testIndex)    # data for testing
```

```
#####################################
```

### Code for Model 1

```
glm_fit <- ctgTrain %>% # Logistic Regression with 2 predictors
  glm(NSP ~ ASTV + Min, data =.,family = "binomial")

p_hat_glm <- predict(glm_fit,ctgTest, type = "response")
y_hat_glm <- factor(ifelse(p_hat_glm > .5,1,0),levels = c("0","1"))
c1 <-confusionMatrix(y_hat_glm, factor(ctgTest$NSP),positive = "0")$byClass[1:2]
c2<- confusionMatrix(y_hat_glm, factor(ctgTest$NSP),positive = "0")$overall["Accuracy"]
df <- data.frame(Result=c(c1[1],c1[2],c2))


# Produces 79% accuracy; 95% specificity, 31% sensitivity
```

### Code for Model 2

```
##################################### Try all predictors.
glm_fit2 <- ctgTrain %>% # Logistic regression with all predictors
  glm(NSP ~ ., data =.,family = "binomial")

p_hat_glm2 <- predict(glm_fit2,ctgTest, type = "response")
y_hat_glm2 <- factor(ifelse(p_hat_glm2 > .5,1,0),levels = c("0","1"))
c1 <-confusionMatrix(y_hat_glm2, factor(ctgTest$NSP),positive = "0")$byClass[1:2]
c2<- confusionMatrix(y_hat_glm2, factor(ctgTest$NSP),positive = "0")$overall["Accuracy"]
df <- data.frame(Result=c(c1[1],c1[2],c2))
df
# Best model so far: accuracy = .89, spec = .74, sens = .93
```

### code for Model 3

```
# We try knn with all predictors. We tune the parameter k.
# We use 5 fold cross validation.
control <- trainControl(method = "cv", number = 5, p = .8)
knn_fit_all <- train(as.factor(NSP) ~ .,
                     method = "knn",
                     data = ctgTrain,
                     trControl = control,
                     tuneGrid = data.frame(k = seq(1,51,2)))
ggplot(knn_fit_all,highlight = TRUE) +
  labs(title = "The Best Tune is with k=1")  # Make plot k vs accuracy

y_hat_knn_all <- predict(knn_fit_all,ctgTest) # Predictions
c1 <-confusionMatrix(y_hat_knn_all, factor(ctgTest$NSP),positive = "0")$byClass[1:2]
c2<- confusionMatrix(y_hat_knn_all, factor(ctgTest$NSP),positive = "0")$overall["Accuracy"]
df <- data.frame(Result=c(c1[1],c1[2],c2))
df                                          # output three metrics

# Accuracy .93, spec = .77, sens = .98
```