

Classification

Code ▾

Classification Notebook

Importing the libraries

Hide

```
library(vioplot)
library(e1071)
library(caret)
library(ggplot2)
library(dplyr)
library(broom)
library(ggpubr)
```

Loading Data

Read in the Dataset

Retrieved the data from a kaggle database measuring Heart Arrhythmia in patients

Hide

```
data <- read.csv("C:/Users/Eric/repo/CS-4375-Machine-Learning/Component-2/INCART 2-lead Arrhythmia Database.csv", na.strings="NA", header=TRUE)
```

Make a copy of the data to reduce re-reading the data set

Hide

```
df <- data
```

Printing a basic the Summary of the data set

Hide

```
summary(df)
```

record	type	X0_pre.RR	X0_post.RR	X0_pPeak	X0
_tPeak					
Length:175729	Length:175729	Min. : 49.0	Min. : 71.0	Min. : -4.208601	Min.
: -7.9595					
Class :character	Class :character	1st Qu.:153.0	1st Qu.:153.0	1st Qu.: -0.068575	1st Q
u.: -0.2381					
Mode :character	Mode :character	Median :188.0	Median :188.0	Median : -0.000857	Media
n : -0.0397					
		Mean :197.2	Mean :197.2	Mean : 0.039047	Mean
: 0.1177					
		3rd Qu.:235.0	3rd Qu.:235.0	3rd Qu.: 0.108743	3rd Q
u.: 0.2234					
		Max. :506.0	Max. :506.0	Max. :10.565904	Max.
: 4.6332					
X0_rPeak	X0_sPeak	X0_qPeak	X0_qrs_interval	X0_pq_interval	X0_qt
_interval					
Min. : -7.1132	Min. : -7.9713	Min. : -7.11320	Min. : 0.00	Min. : 1.000	Min.
: 3.00					
1st Qu.: 0.3298	1st Qu.: -0.9104	1st Qu.: -0.17626	1st Qu.: 14.00	1st Qu.: 4.000	1st Q
u.: 25.00					
Median : 0.9055	Median : -0.6347	Median : -0.10549	Median : 17.00	Median : 6.000	Media
n : 30.00					
Mean : 0.8567	Mean : -0.5868	Mean : -0.17421	Mean : 17.62	Mean : 8.714	Mean
: 37.46					
3rd Qu.: 1.4286	3rd Qu.: -0.3504	3rd Qu.: -0.05458	3rd Qu.: 22.00	3rd Qu.: 12.000	3rd Q
u.: 44.00					
Max. : 4.5983	Max. : 3.8944	Max. : 3.12903	Max. :131.00	Max. :113.000	Max.
:247.00					
X0_st_interval	X0_qrs_morph0	X0_qrs_morph1	X0_qrs_morph2	X0_qrs_morph3	X0_q
rs_morph4					
Min. : 1.00	Min. : -7.11320	Min. : -7.11320	Min. : -7.51305	Min. : -7.7958	Min.
: -7.9404					
1st Qu.: 6.00	1st Qu.: -0.17626	1st Qu.: -0.12004	1st Qu.: -0.05432	1st Qu.: 0.1241	1st
Qu.: -0.1029					
Median : 6.00	Median : -0.10549	Median : -0.02621	Median : 0.23982	Median : 0.6414	Medi
an : 0.1432					
Mean :11.13	Mean : -0.17421	Mean : -0.08403	Mean : 0.27836	Mean : 0.6118	Mean
: 0.1665					
3rd Qu.: 8.00	3rd Qu.: -0.05458	3rd Qu.: 0.07186	3rd Qu.: 0.69674	3rd Qu.: 1.1380	3rd
Qu.: 0.4625					
Max. :89.00	Max. : 3.12903	Max. : 3.46521	Max. : 4.08571	Max. : 4.4986	Max.
: 4.3042					
X1_pre.RR	X1_post.RR	X1_pPeak	X1_tPeak	X1_rPeak	X1_s
Peak					
Min. : 49.0	Min. : 71.0	Min. : -1.10302	Min. : -2.0218	Min. : -2.39371	Min.
: -3.2696					
1st Qu.:153.0	1st Qu.:153.0	1st Qu.: 0.01095	1st Qu.: 0.2044	1st Qu.: -0.32062	1st Q
u.: -0.9343					
Median :188.0	Median :188.0	Median : 0.04446	Median : 0.3190	Median : -0.08507	Median
: -0.7451					
Mean :197.2	Mean :197.2	Mean : 0.06245	Mean : 0.3664	Mean : -0.11384	Mean
: -0.7524					

3rd Qu.:235.0	3rd Qu.:235.0	3rd Qu.: 0.08618	3rd Qu.: 0.4519	3rd Qu.: 0.10980	3rd Q
u.: -0.5643					
Max. :506.0	Max. :506.0	Max. : 7.63177	Max. : 3.6454	Max. : 4.06481	Max.
: 2.8227					
X1_qPeak	X1_qrs_interval	X1_pq_interval	X1_qt_interval	X1_st_interval	X1_qrs_m
orph0					
Min. : -6.46175	Min. : 0.00	Min. : 1.000	Min. : 4.00	Min. : 1.00	Min. :
-6.46175					
1st Qu.: -0.32406	1st Qu.: 4.00	1st Qu.: 3.000	1st Qu.: 18.00	1st Qu.: 8.00	1st Qu.:
-0.32406					
Median : -0.12467	Median : 5.00	Median : 5.000	Median : 24.00	Median : 9.00	Median :
-0.12467					
Mean : -0.21606	Mean : 10.59	Mean : 7.107	Mean : 28.05	Mean : 10.35	Mean :
-0.21606					
3rd Qu.: -0.02073	3rd Qu.: 15.00	3rd Qu.: 8.000	3rd Qu.: 33.00	3rd Qu.: 12.00	3rd Qu.:
-0.02073					
Max. : 1.38021	Max. : 128.00	Max. : 121.000	Max. : 290.00	Max. : 86.00	Max. :
1.38021					
X1_qrs_morph1	X1_qrs_morph2	X1_qrs_morph3	X1_qrs_morph4		
Min. : -4.91330	Min. : -3.98796	Min. : -2.96139	Min. : -3.1321		
1st Qu.: -0.36821	1st Qu.: -0.53684	1st Qu.: -0.67904	1st Qu.: -0.8133		
Median : -0.18547	Median : -0.33999	Median : -0.47761	Median : -0.5922		
Mean : -0.24779	Mean : -0.31871	Mean : -0.38666	Mean : -0.5396		
3rd Qu.: -0.01321	3rd Qu.: 0.02637	3rd Qu.: 0.05749	3rd Qu.: -0.1722		
Max. : 2.78418	Max. : 4.04009	Max. : 2.77139	Max. : 2.7649		

Filtering the Data

Remove the unnecessary columns

We remove the record column since we won't need to differentiate patients

[Hide](#)

```
df <- df[, !names(df) %in% c("record")]
```

Looking at the number and type of different arrhythmia we can see that most people have a normal rhythm, but there is a few people with abnormal rhythms.

[Hide](#)

```
uniqueTypes <- unique(df$type)
for (type in uniqueTypes) {
  print(paste("The number of rows of type ", type, " is ", nrow(df[df$type == type, ])))
}
```

```
[1] "The number of rows of type N is 153546"  
[1] "The number of rows of type VEB is 20000"  
[1] "The number of rows of type SVEB is 1958"  
[1] "The number of rows of type F is 219"  
[1] "The number of rows of type Q is 6"
```

Remove rows of type SVEB, F, and type Q since there is not enough data to support classifying these types. Want to focus on the difference between normal and VEB

[Hide](#)

```
df <- df[df$type == "N" | df$type == "VEB",]  
summary(df)
```

type	X0_pre.RR	X0_post.RR	X0_pPeak	X0_tPeak	X0_rPeak
Length:173546 :-7.1132	Min. : 49.0	Min. : 71.0	Min. :-4.208601	Min. :-7.95946	Min. :-7.1132
Class :character u.: 0.3236	1st Qu.:154.0	1st Qu.:152.0	1st Qu.: -0.068890	1st Qu.: -0.23754	1st Qu.: -0.068890
Mode :character n : 0.9016	Median :188.0	Median :187.0	Median :-0.001037	Median :-0.03608	Median :-0.001037
	Mean :197.8	Mean :196.3	Mean : 0.038994	Mean : 0.12001	Mean : 0.038994
	3rd Qu.:236.0	3rd Qu.:234.0	3rd Qu.: 0.108408	3rd Qu.: 0.22579	3rd Qu.: 0.108408
u.: 1.4279	Max. :506.0	Max. :506.0	Max. :10.565904	Max. : 4.63317	Max. :10.565904
: 4.5983					
X0_sPeak	X0_qPeak	X0_qrs_interval	X0_pq_interval	X0_qt_interval	X0_st_interval
Min. :-7.9713	Min. :-7.11320	Min. : 0.00	Min. : 1.000	Min. : 3.00	Min. : 1.00
1st Qu.: -0.9140	1st Qu.: -0.17678	1st Qu.: 14.00	1st Qu.: 4.000	1st Qu.: 25.00	1st Qu.: 6.00
Median : -0.6376	Median : -0.10570	Median : 17.00	Median : 6.000	Median : 30.00	Median : 6.00
Mean : -0.5871	Mean : -0.17450	Mean : 17.54	Mean : 8.696	Mean : 37.41	Mean : 11.17
3rd Qu.: -0.3487	3rd Qu.: -0.05497	3rd Qu.: 22.00	3rd Qu.: 12.000	3rd Qu.: 44.00	3rd Qu.: 8.00
Max. : 3.8944	Max. : 3.12903	Max. :108.00	Max. :113.000	Max. :247.00	Max. :89.00
X0_qrs_morph0	X0_qrs_morph1	X0_qrs_morph2	X0_qrs_morph3	X0_qrs_morph4	
X1_pre.RR					
Min. :-7.11320	Min. :-7.11320	Min. :-7.51305	Min. :-7.7958	Min. :-7.9404	Min. :-7.11320
1st Qu.: -0.17678	1st Qu.: -0.12030	1st Qu.: -0.05453	1st Qu.: 0.1248	1st Qu.: -0.1048	1st Qu.: -0.17678
Median : -0.10570	Median : -0.02640	Median : 0.23927	Median : 0.6392	Median : 0.1405	Median : -0.10570
Mean : -0.17450	Mean : -0.08426	Mean : 0.27796	Mean : 0.6110	Mean : 0.1640	Mean : -0.17450
3rd Qu.: -0.05497	3rd Qu.: 0.07163	3rd Qu.: 0.69544	3rd Qu.: 1.1374	3rd Qu.: 0.4578	3rd Qu.: -0.05497
Max. : 3.12903	Max. : 3.46521	Max. : 4.08571	Max. : 4.4986	Max. : 4.3042	Max. : 3.12903
X1_post.RR	X1_pPeak	X1_tPeak	X1_rPeak	X1_sPeak	X1_qPeak
Min. : 71.0	Min. :-1.10302	Min. :-2.0218	Min. :-2.19031	Min. :-3.2696	Min. : 71.0
1st Qu.:152.0	1st Qu.: 0.01163	1st Qu.: 0.2037	1st Qu.: -0.31569	1st Qu.: -0.9322	1st Qu.:152.0
Median :187.0	Median : 0.04498	Median : 0.3188	Median :-0.08121	Median :-0.7418	Median :187.0
Mean :196.3	Mean : 0.06310	Mean : 0.3661	Mean :-0.11006	Mean :-0.7497	Mean :196.3

```

3rd Qu.:234.0    3rd Qu.: 0.08657    3rd Qu.: 0.4520    3rd Qu.: 0.11129    3rd Qu.: -0.5626    3rd Q
u.: -0.02008
Max.    :506.0    Max.    : 7.63177    Max.    : 3.6454    Max.    : 4.06481    Max.    : 2.8227    Max.
: 1.38021
X1_qrs_interval X1_pq_interval X1_qt_interval X1_st_interval X1_qrs_morph0 X1_qrs_mo
rph1
Min.    : 0.00    Min.    : 1.000    Min.    : 4.00    Min.    : 1.0    Min.    : -6.46175    Min.    : -
4.91330
1st Qu.: 4.00    1st Qu.: 3.000    1st Qu.: 18.00    1st Qu.: 8.0    1st Qu.: -0.31907    1st Qu.: -
0.36390
Median : 5.00    Median : 5.000    Median : 24.00    Median : 9.0    Median : -0.12229    Median : -
0.18194
Mean   : 10.66    Mean   : 7.124    Mean   : 28.09    Mean   : 10.3    Mean   : -0.21316    Mean   : -
0.24490
3rd Qu.: 15.00    3rd Qu.: 8.000    3rd Qu.: 33.00    3rd Qu.: 12.0    3rd Qu.: -0.02008    3rd Qu.: -
0.01235
Max.    :128.00    Max.    :121.000    Max.    :290.00    Max.    :86.0    Max.    : 1.38021    Max.    :
2.78418
X1_qrs_morph2    X1_qrs_morph3    X1_qrs_morph4
Min.    : -3.98796    Min.    : -2.96139    Min.    : -3.1321
1st Qu.: -0.53146    1st Qu.: -0.67509    1st Qu.: -0.8096
Median : -0.33564    Median : -0.47286    Median : -0.5863
Mean   : -0.31488    Mean   : -0.38230    Mean   : -0.5354
3rd Qu.: 0.02809    3rd Qu.: 0.05914    3rd Qu.: -0.1682
Max.    : 4.04009    Max.    : 2.77139    Max.    : 2.7649

```

Factor each of the 2 types to ensure that each has its own value. This will help us ensure that naive bayes recognizes each of these as a number instead a string

Hide

```

df$type <- factor(df$type, levels=c("N", "VEB"))
contrasts(df$type)

```

```

      VEB
N       0
VEB     1

```

Using the Data

Splitting the data into train/test

We split the data into 2 separate data frames depending on a random sample. 80 percent of the data is going to training and the other 20 percent is going to testing and evaluation.

Hide

```
dt = sort(sample(nrow(df), nrow(df)*.8))  
train <- df[dt,]  
test <- df[-dt,]
```

Data Exploration Functions

Data exploration Function 1 - We can see that there are no NA's inside of the data frame. This is good for us since all data can be used inside our linear models and we don't need to worry about replacing any values. For the dim function we can see the number of observation times the number of columns

[Hide](#)

```
sum(is.na(train))
```

```
[1] 0
```

[Hide](#)

```
dim(train)
```

```
[1] 138836    33
```

Data Exploration Function 2 - gets the coefficients of all numerical values in data and looks for the those variables in the upper diagonal that are most highly correlated. From this table we can see that we have 6 variables variables that have a coefficient of at least .9 and many more with .8. Using this information we can see several opportunities to look for correlations among variables

[Hide](#)

```
cor <- cor(df[2:33], use="complete")  
cor[cor < .5 & cor > -.5] <- 0  
cor[upper.tri(cor, diag = TRUE)] <- 0  
table(cor)
```

cor					
-0.681552879721901	-0.680452238026283	-0.613787819747585	-0.57719425529799	-0.543725476969976	-
0.532249558133321					
2	1	1	1	1	
2					
-0.512725069534083	-0.507848330225606	-0.5011677148133	0	0.551826345143866	
0.557453050090752					
1	1	1	952	2	
2					
0.569976100807781	0.591670461519874	0.596196766041581	0.599707309780323	0.600555090682228	
0.608382978601611					
1	1	1	1	1	1
1					
0.609816558106336	0.614250894000366	0.615474343462091	0.617395597336139	0.634660909137918	
0.636073413766473					
2	1	2	1	1	
1					
0.637998375145257	0.656407935313161	0.668954992015978	0.685589309537155	0.687264505644391	
0.694347066833005					
1	1	1	1	2	
1					
0.708205695359803	0.713006221051641	0.737740642456596	0.753248412246809	0.764116598110986	
0.768268176124287					
1	2	1	1	2	
1					
0.778110120011753	0.778149876122119	0.784260201927876	0.792273479475908	0.814000721851494	
0.840852045732589					
1	1	1	1	2	
1					
0.844236520057343	0.845760374105322	0.863353608495156	0.872253699074859	0.873143316095882	
0.884628792760613					
1	2	1	1	1	
1					
0.886280462017853	0.89714939570761	0.903030549958228	0.928607577815106	0.947014688865174	
0.949507110672762					
1	1	1	1	2	
1					
0.960574156794689	0.968017171366627	1			
1	2	4			

Data Exploration Function 3 - Using the graph on the left we can see there is a very strong imbalance between the number of N's and VEB's. Using the graph on the right we can see that even among highly correlated values we may not be able to differentiate between the two types of rhythms. The bottom graph is another example with semi-strong correlation

[Hide](#)

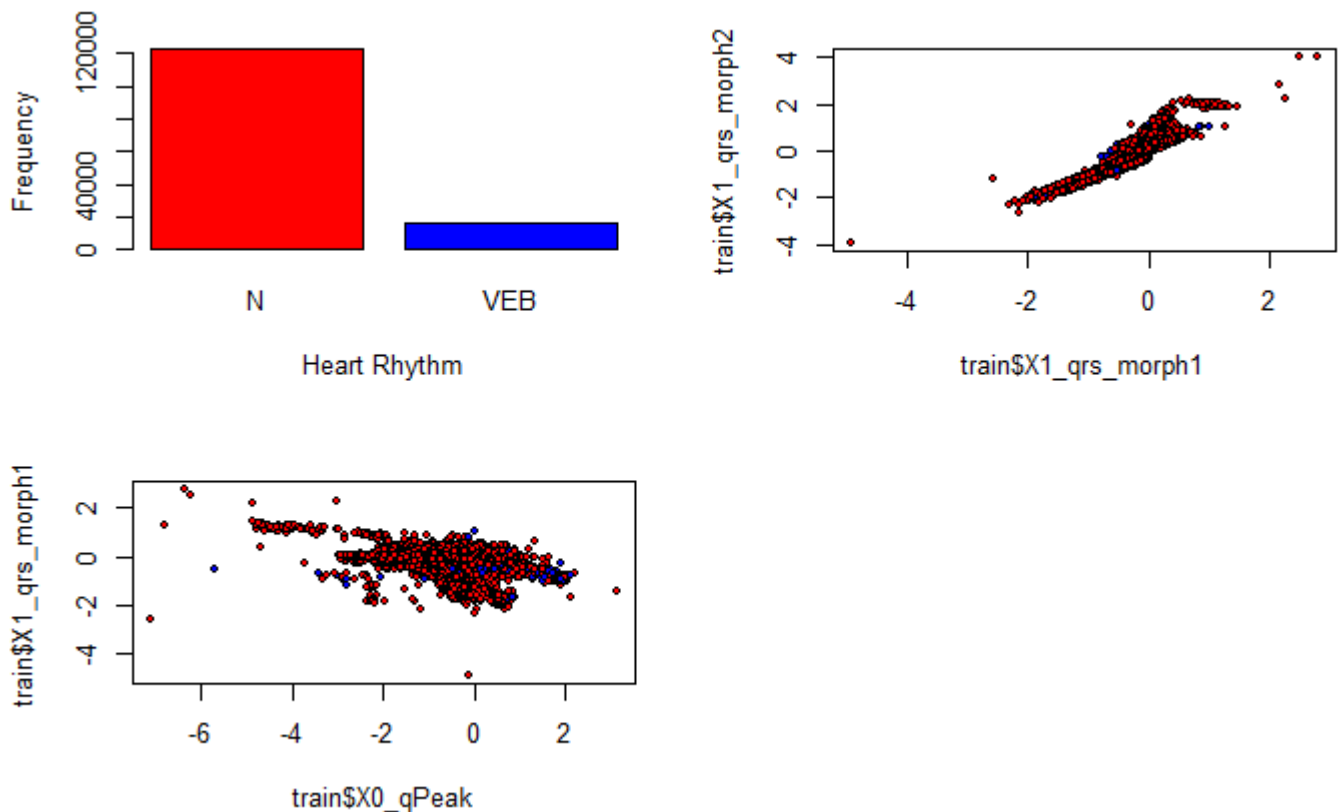

```
par(mfrow=c(2,2))

counts <- table(train$type)
barplot(counts, xlab="Heart Rhythm", ylab="Frequency", col=c("red","blue","green"))

plot(train$X1_qrs_morph1, train$X1_qrs_morph2, pch=21, cex=0.75, bg=c("red", "blue", "green")[unclass(df$type)],)
```

Hide

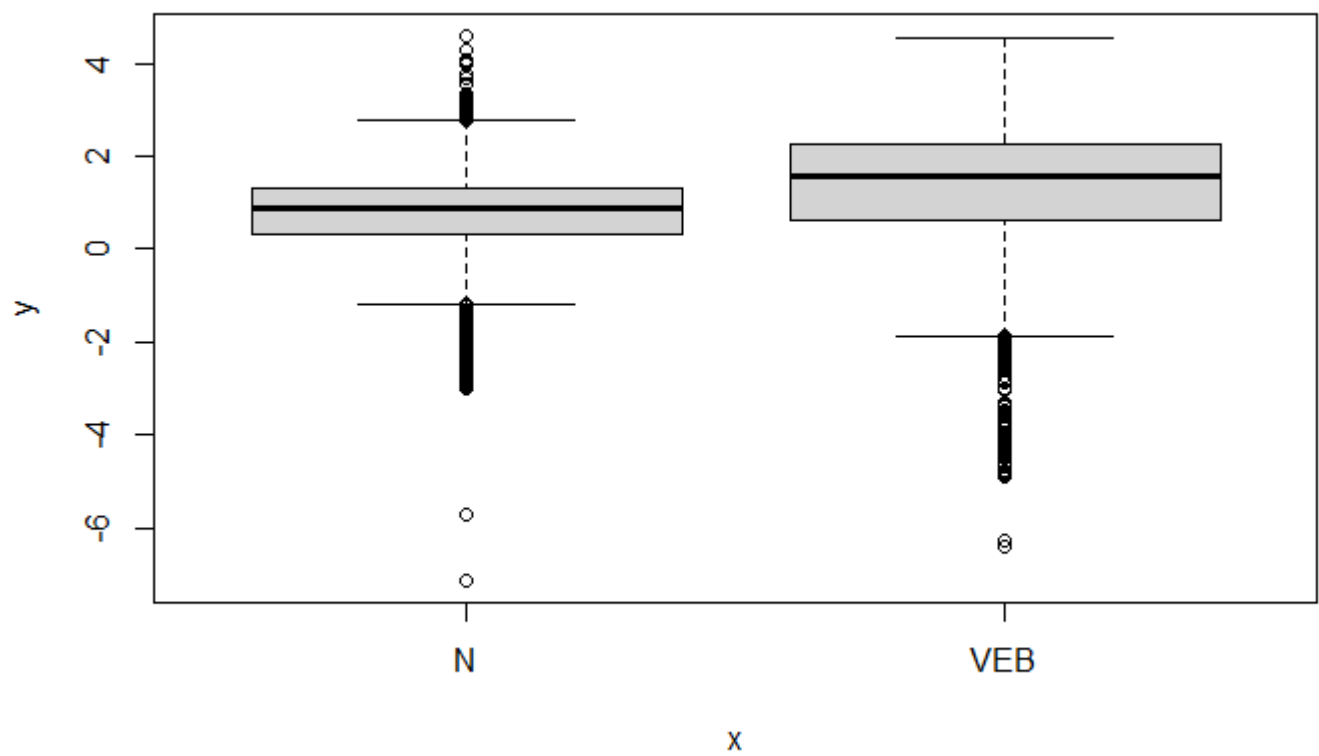
```
plot(train$X0_qPeak, train$X1_qrs_morph1, pch=21, cex=0.75, bg=c("red", "blue", "green")[unclass(df$type)])
```



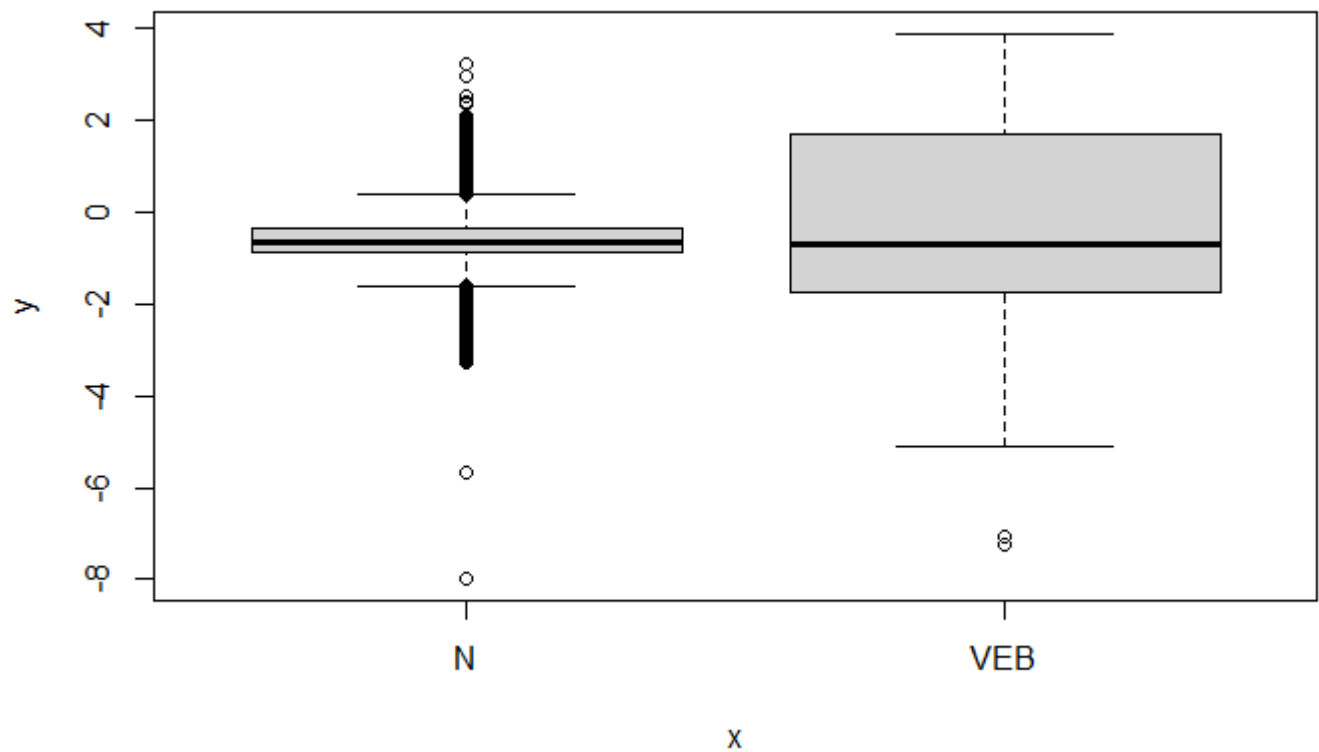
Some more basic plot functions for other varied visual confirmation of some correlation.

Hide

```
plot(train$type, train$X0_rPeak)
```

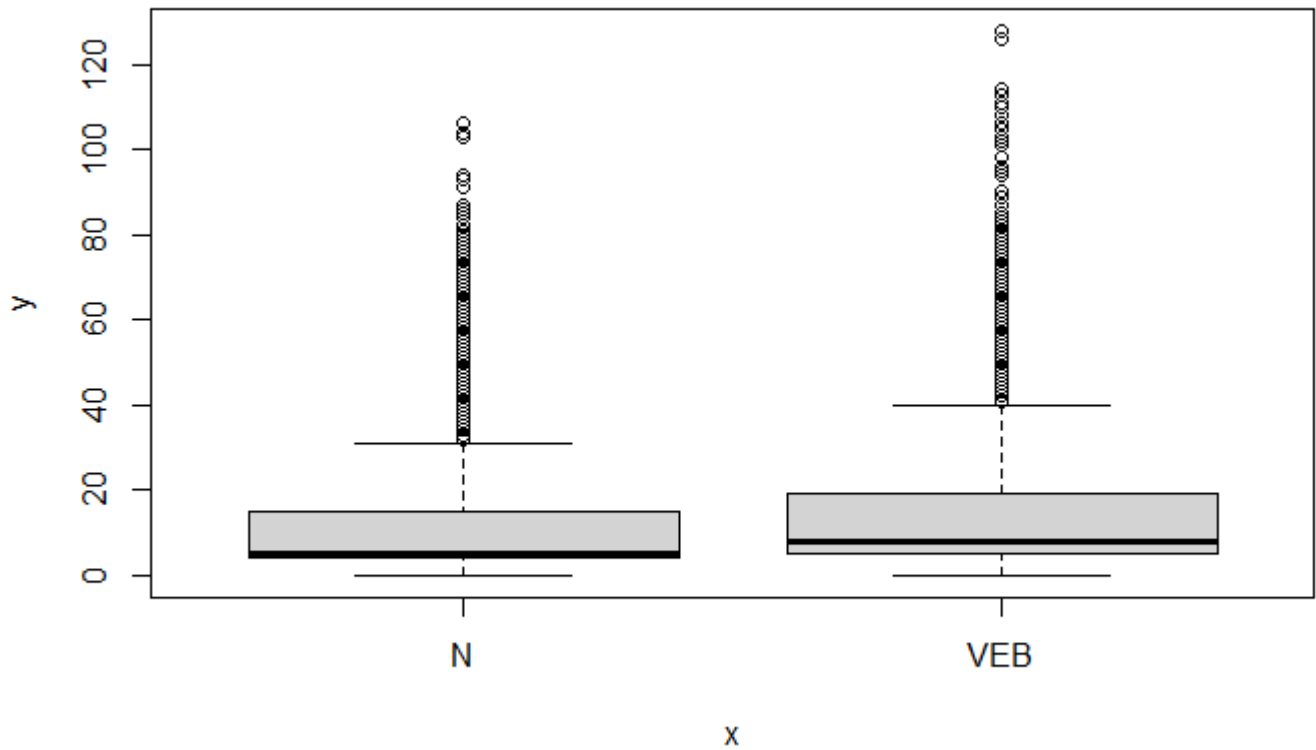
[Hide](#)

```
plot(train$type, train$sPeak)
```

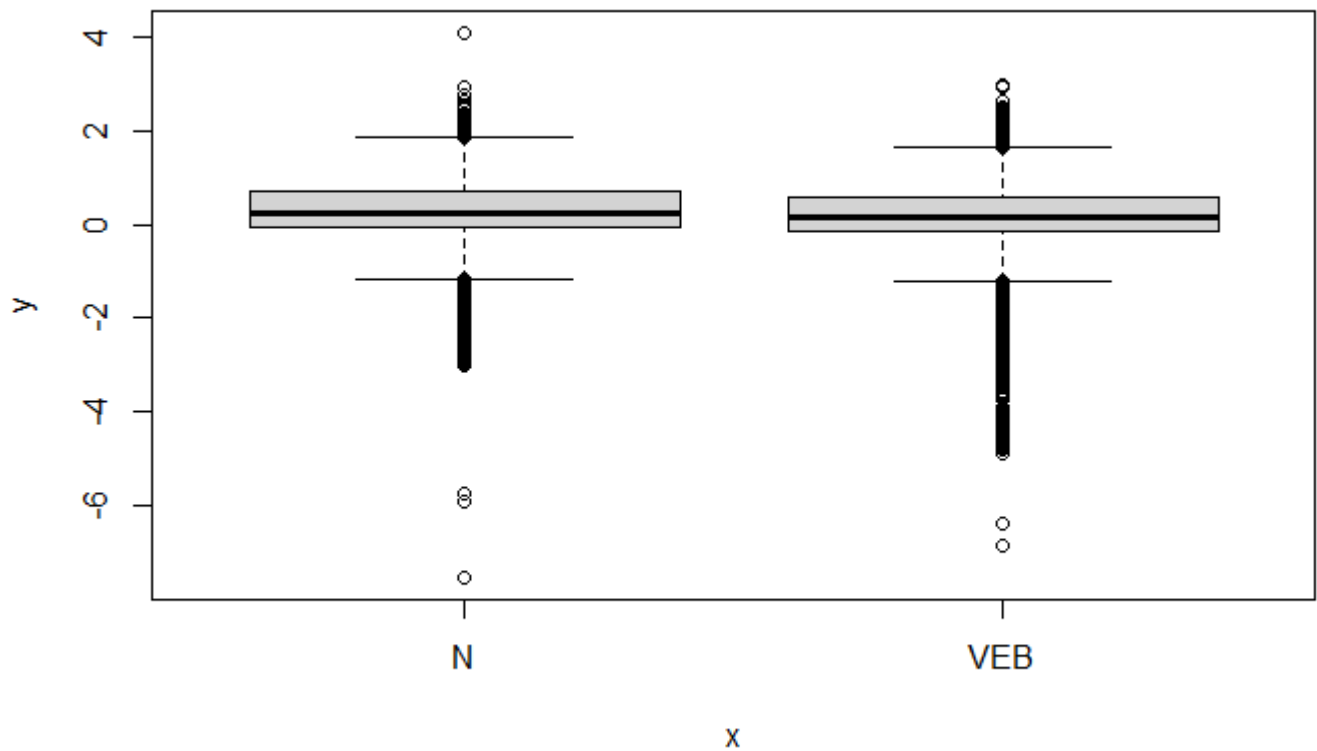


[Hide](#)

```
plot(train$type, train$X1_qrs_interval)
```

[Hide](#)

```
plot(train$type, train$X0_qrs_morph2)
```



Data Exploration Function 5

`cov(trainX1rPeak, trainX0rPeak)` - shows not a lot of similarity between different X values even in a single patient
`cov(trainX0qrs_interval, trainX0qPeak)` - shows high covariance between the intervals length and peaks

[Hide](#)

```
cov(train$X1_rPeak, train$X0_rPeak)
```

```
[1] -0.1699659
```

[Hide](#)

```
cov(train$X0_qrs_interval, train$X0_qPeak)
```

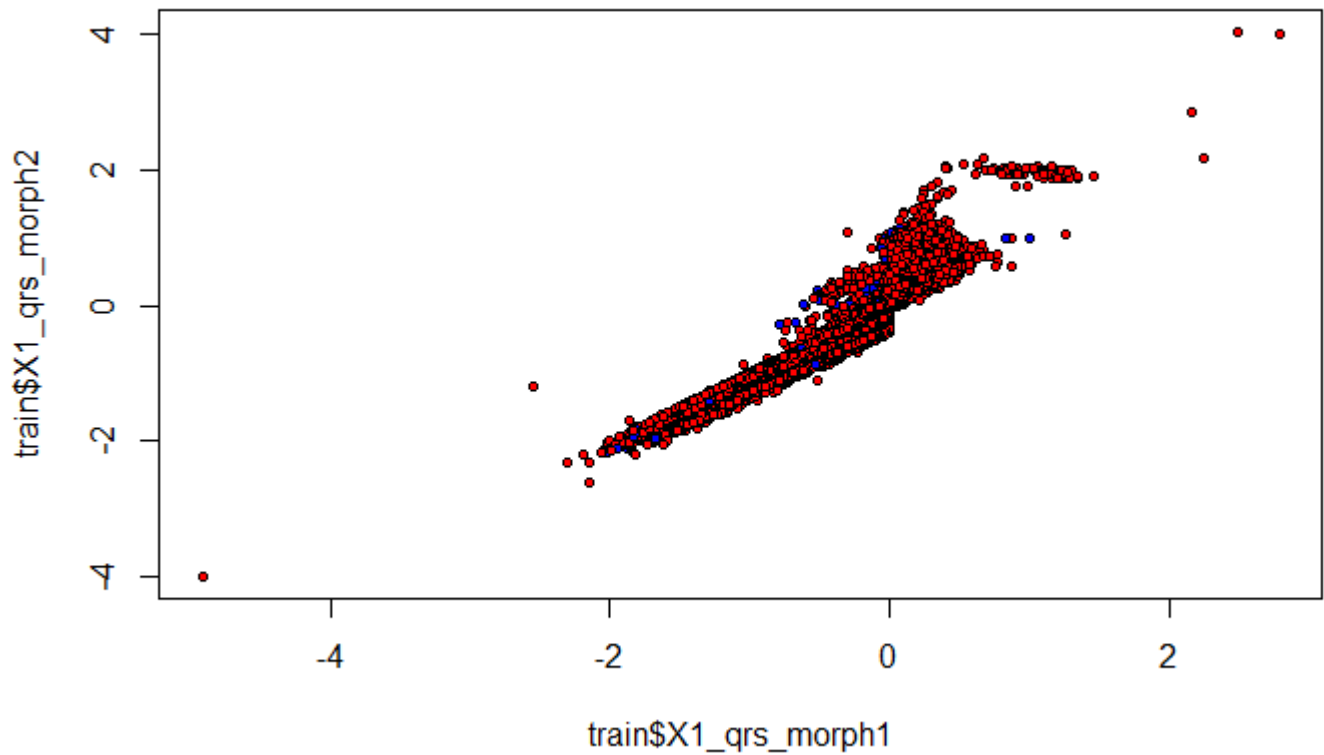
```
[1] 0.7559834
```

Informative Graphs

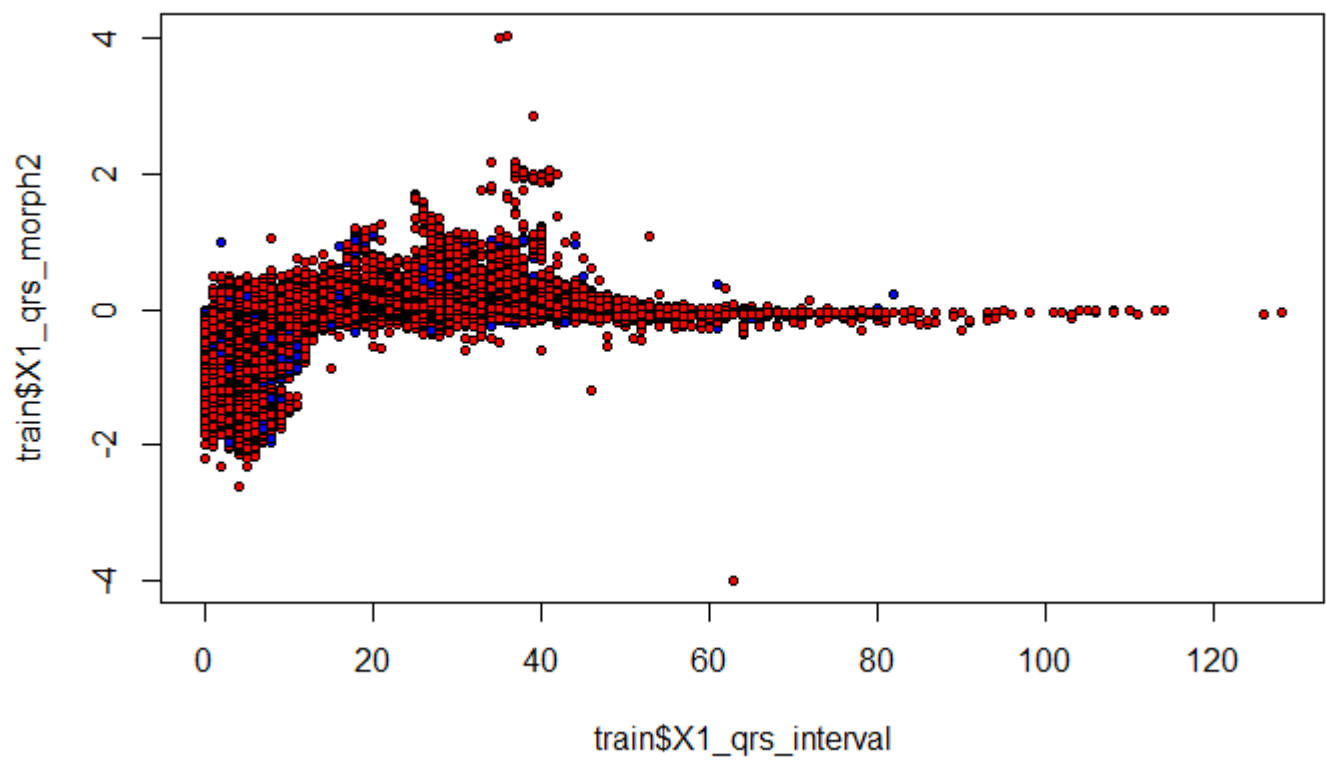
Several plots that show the relationship between different variables while also showing the different circles depending on which type of rhythm the patient had

[Hide](#)

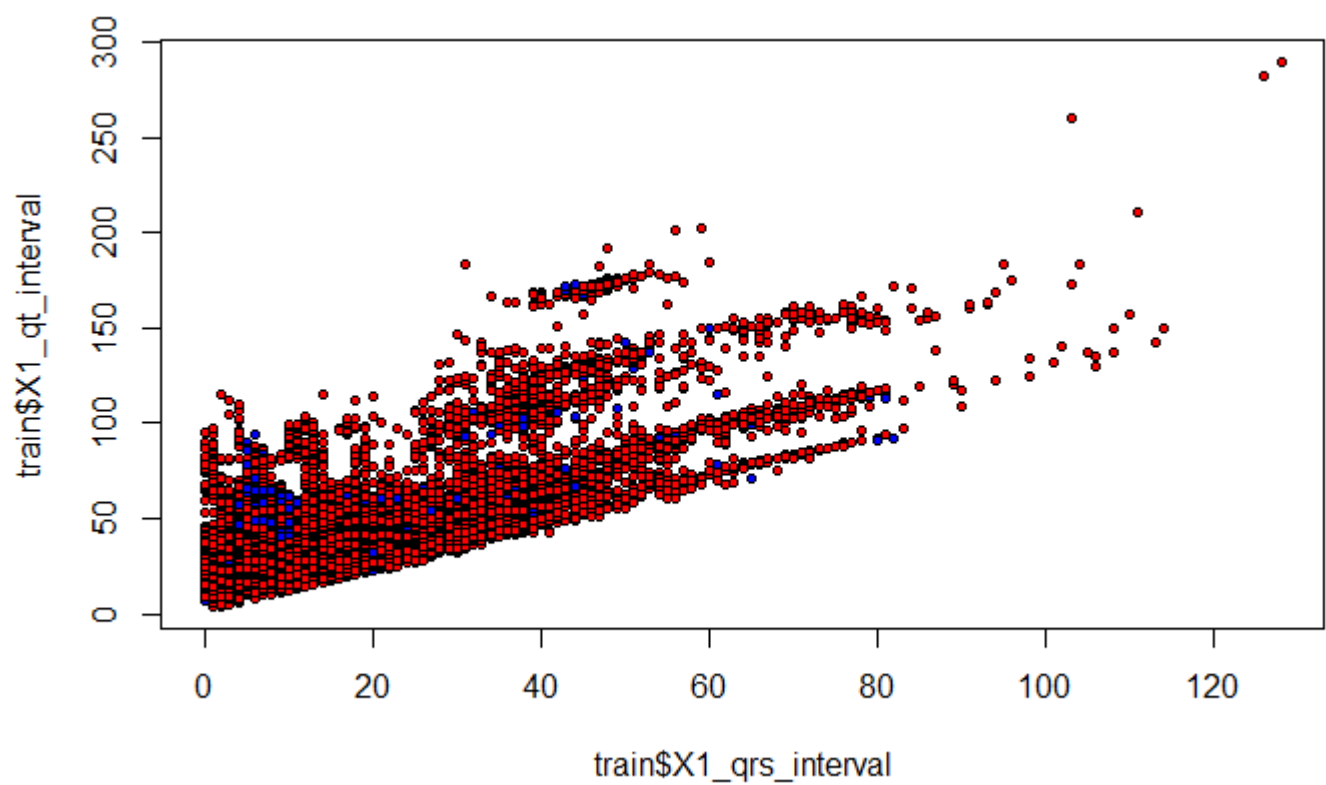
```
plot(train$X1_qrs_morph1, train$X1_qrs_morph2, pch=21, cex=0.75, bg=c("red", "blue")[unclass(df$type)])
```

[Hide](#)

```
plot(train$X1_qrs_interval, train$X1_qrs_morph2, pch=21, cex=0.75, bg=c("red", "blue")[unclass(df$type)])
```

[Hide](#)

```
plot(train$X1_qrs_interval, train$X1_qt_interval, pch=21, cex=0.75, bg=c("red", "blue")[unclass(df$type)])
```



Making Models

Hide

```
func <- function(df){
  dt <- sort(sample(nrow(df), nrow(df)*.8))
  train <- df[dt,]
  test <- df[-dt,]
  glm1 <- glm(type~X0_qPeak, data=train, family="binomial")
  probs <- predict(glm1, newdata=test)
  pred <- ifelse(probs>0.5, 1, 0)
  acc <- mean(pred==test$type)
  print(paste("accuracy = ", acc))
  table(pred, test$type)
}
```

Makes a logistical model that compares the peak to the interval plus the type of rhythm the patient had. We first split the data into its different classes and then run it through the function for the logistical creation. It then evaluates the data on the test data and shows that the model is perfectly seperable into the two types. Even giving a warning of not converging. This means that the classes were easily separable given the predictor of X0_qPeak.

Hide

```
n_data <- df[df$type == "N",]
veb_data <- df[df$type == "VEB",]

func(n_data)
```

Warning: glm.fit: algorithm did not converge

```
[1] "accuracy = 0"
```

pred	N	VEB
0	30710	0

Hide

```
func(veb_data)
```

Warning: glm.fit: algorithm did not converge

```
[1] "accuracy = 0"
```

pred	N	VEB
0	0	4000

Makes a naive bayes model that attempts to find the type of rhythm depending on the peak value of a patient. Uses the training data while training. Prints a basic summary of the naive bayes model Naive Byes - shows a table with the conditional probabilities of a certain rhythm happening. We can see that in general once X gets below a

certain value there is a much higher chance of the rhythm being VEB

[Hide](#)

```
nb1 <- naiveBayes(type~X0_qPeak, data=train)
nb1
```

Naive Bayes Classifier for Discrete Predictors

Call:

```
naiveBayes.default(x = X, y = Y, laplace = laplace)
```

A-priori probabilities:

Y		N	VEB
	0.8846337	0.1153663	

Conditional probabilities:

		X0_qPeak	
Y		[,1]	[,2]
N	-0.1619998	0.3295289	
VEB	-0.2629795	0.5671563	

Predicting on train data for Naive Bayes

Uses the predict module to predict for naive Bayes module. This predict uses the test data set to test how effective our model was in predicting the correct result

[Hide](#)

```
## Predicting the linear model
resultNaiveBayes <- predict(nb1, newdata = test, type="class")
```

Evaluate on test data for Naive Bayes

The prediction on the naive bayes algorithm shows that we got an accuracy of about 88 percent. The low kappa value shows that there is a high prevalence on the number of N's. If we were to do this again we would want to ensure that the model took more of VEB into consideration. This is further shown with a high sensitivity of 98%. We have a high positive predicted value which makes sense since it predicts the class with the highest number most of the time.

[Hide](#)

```
mean(resultNaiveBayes==test$type)
```

```
[1] 0.8810717
```

[Hide](#)


```
confusionMatrix(resultNaiveBayes, test$type)
```

Confusion Matrix and Statistics

	Reference	
Prediction	N	VEB
N	30173	3574
VEB	554	409

Accuracy : 0.8811

95% CI : (0.8776, 0.8845)

No Information Rate : 0.8852

P-Value [Acc > NIR] : 0.9927

Kappa : 0.1263

McNemar's Test P-Value : <2e-16

Sensitivity : 0.9820

Specificity : 0.1027

Pos Pred Value : 0.8941

Neg Pred Value : 0.4247

Prevalence : 0.8852

Detection Rate : 0.8693

Detection Prevalence : 0.9723

Balanced Accuracy : 0.5423

'Positive' Class : N

Comparing Naive Bayes to Logistical Regression

For both models they output a classification result saying which class it thinks the specific class is in. In this case Logistical Regression won out because it found a line that perfectly separated the two classes giving a 100 percent success rate in finding the difference between Normal rhythms and VEB rhythms. In addition we had a data set with an extremely high number of observations meaning we logistical models would have done better even if there wasn't a perfect dividing line between the classes.

Strengths and Weaknesses

Strengths and Weaknesses of Naive Bayes

The strength of Naive Bayes lies in its ability to quickly train on rather small data sets and since it the output of the model is just a single factor it is easy for us to understand the results of the given data. The bias of a given model is also much higher given that the eventual probabilities of the model are based off the population sizes of the the classes in the data. A general weakness is that there is little growth in the abilities of model as data size increases. This is due to the fact that even as more data comes in if the probabilities are the same for the data then the eventual result of the model will not end up changing.

Strengths and Weaknesses of Logistic Regression

The strength of Logistic Regression is great for data that is easily split by a line. For example if there is a data point that easily divides two or more classes. Then it defines being above that line as one class and below as another. Additionally it is very fast since it is just forming a linear model that splits the two classes. Additionally gives a nice output of predicted classes. The main weakness is that it really struggles if there is no clear boundary between any two predictors. Have to find a predictor that splits the two well down the middle or use a different model