Mathematical Methods for Artificial Intelligence Lab 3

Vytautas Kraujalis

2021-10-30

Contents

1 Reading Data

```
set.seed(123)

data_original <- read.csv("Arrhythmia_Dataset.csv")
```

2 Required packages

```r
library(SmartEDA)
library(dplyr)
library(ggplot2)
library(caret)
library(rattle)

# Function for 3 class accuracy from confusion matrix
classAcc <- function(confusionMatrix) {
  class0 <- round(confusionMatrix$table[1, 1] / sum(confusionMatrix$table[, 1]) * 100, 1)
  class1 <- round(confusionMatrix$table[2, 2] / sum(confusionMatrix$table[, 2]) * 100, 1)
  class2 <- round(confusionMatrix$table[3, 3] / sum(confusionMatrix$table[, 3]) * 100, 1)
  acc <- c(class0, class1, class2 )
  names(acc) <- colnames(confusionMatrix$table)
  return(acc)
}
```

3 Parallel processing

```r
library(parallel)
no_cores <- detectCores() - 1
library(doParallel)
cl <- makePSOCKcluster(no_cores)
registerDoParallel(cl)
```

4 EDA, first look at the dataset

```r
ExpData(data_original,type=1)
```

```
##                                               Descriptions     Value
## 1                                       Sample size (nrow)    175729
## 2                                  No. of variables (ncol)        34
## 3                        No. of numeric/interger variables        32
## 4                                  No. of factor variables         0
## 5                                    No. of text variables         2
## 6                                 No. of logical variables         0
## 7                              No. of identifier variables         0
## 8                                    No. of date variables         0
## 9                 No. of zero variance variables (uniform)         0
## 10                  %. of variables having complete cases 100% (34)
## 11   %. of variables having >0% and <50% missing cases    0% (0)
## 12 %. of variables having >=50% and <90% missing cases    0% (0)
## 13          %. of variables having >=90% missing cases    0% (0)
```

We have a dataset of 175729 observations with 34 variables, of which are 2 text variables and 32 - numerical. All variables have no missing values.

Let's look at the text variables:

```r
data_original %>%
   group_by(record) %>%
   summarise(n = n()) %>%
   summary()
```

```
##     record                n
##  Length:75          Min.   :1452
##  Class :character   1st Qu.:1957
##  Mode  :character   Median :2323
##                     Mean   :2343
##                     3rd Qu.:2659
##                     Max.   :3909
```

Our first text variable record has 75 unique values (unique patients). Each patient has on average 2343 observations.

```r
data_original %>%
   group_by(type) %>%
   summarise(n = n()) %>%
   mutate(n_prop = round(n / sum(n) * 100, 2))
```

```
## # A tibble: 5 x 3
##   type       n n_prop
##   <chr>  <int>  <dbl>
## 1 F        219   0.12
## 2 N     153546  87.4
## 3 Q          6   0
## 4 SVEB    1958   1.11
## 5 VEB    20000  11.4
```

The next text variable type is our target variable. It has 5 classes, but there's a huge class imbalance. 2 out of 5 classes takes up to 98.76% of all observations, 1 class has just 6 observations. We are planning to combine F, Q, SVEB classes into one class, as those 3 classes combined only takes up to 1.23% of observations.

N (Normal) - Normal beat.

SVEB (Supraventricular ectopic beat) - Supraventricular Ectopic Beats indicates atrial irritability. Isolated Supraventricular Ectopic Beats are generally not significant in nature but a high frequency can represent more risk. An increasing trend in Supraventricular Ectopic Beats may be an indicator or sign for atrial fibrillation. Atrial Fibrillation is considered to be significant as it can lead to heart attack or stroke.

VEB (Ventricular ectopic beat) - Ventricular ectopics are a type of arrhythmia or abnormal heart rhythm. It is caused by the electric signals in the heart starting in a different place and travelling a different way through the heart. If it happens occasionally, it should not cause any problems but if it happens a lot, you will need to have treatment.

F (Fusion beat) - A fusion beat occurs when a supraventricular and a ventricular impulse coincide to produce a hybrid complex. It indicates that there are two foci of pacemaker cells firing simultaneously: a supraventricular pacemaker (e.g. the sinus node) and a competing ventricular pacemaker (source of ventricular ectopics).

Q (Unknown beat) - Unknown beat.

```r
data <- data_original %>%
   mutate(type = case_when(
   type %in% c("F", "Q", "SVEB") ~ "F_Q_SVEB",
   TRUE ~ type
   ))

data %>%
   group_by(type) %>%
   summarise(n = n()) %>%
   mutate(n_prop = round(n / sum(n) * 100, 2))
```

```
## # A tibble: 3 x 3
##   type          n n_prop
##   <chr>     <int>  <dbl>
## 1 F_Q_SVEB   2183   1.24
## 2 N        153546  87.4
## 3 VEB       20000  11.4
```

We combined the 3 classes into 1 class, which has only 2183 observations (1.24%) which is still low, but still better for our basic models to classify.

We are not interested in the patient record variable, so we'll remove it:

```r
data <- data %>%
   select(-record)
```

Let's look at descriptive statistics of each variable:

```r
ExpNumStat(data,by ="A",round= 2, gp = "type") %>%
  select(Vname, min, max, mean, median, SD)
```

```
##              Vname   min    max   mean median    SD
## 2       X0_post.RR 71.00 506.00 197.20 188.00 61.72
## 3         X0_pPeak -4.21  10.57   0.04   0.00  0.19
## 9   X0_pq_interval  1.00 113.00   8.71   6.00  7.89
## 1        X0_pre.RR 49.00 506.00 197.18 188.00 61.70
## 7         X0_qPeak -7.11   3.13  -0.17  -0.11  0.37
## 8  X0_qrs_interval  0.00 131.00  17.62  17.00 10.62
## 12   X0_qrs_morph0 -7.11   3.13  -0.17  -0.11  0.37
## 13   X0_qrs_morph1 -7.11   3.47  -0.08  -0.03  0.40
## 14   X0_qrs_morph2 -7.51   4.09   0.28   0.24  0.64
## 15   X0_qrs_morph3 -7.80   4.50   0.61   0.64  0.85
## 16   X0_qrs_morph4 -7.94   4.30   0.17   0.14  0.68
## 10  X0_qt_interval  3.00 247.00  37.46  30.00 23.64
## 5         X0_rPeak -7.11   4.60   0.86   0.91  0.87
## 6         X0_sPeak -7.97   3.89  -0.59  -0.63  0.80
## 11  X0_st_interval  1.00  89.00  11.13   6.00 14.30
## 4         X0_tPeak -7.96   4.63   0.12  -0.04  0.64
## 18      X1_post.RR 71.00 506.00 197.20 188.00 61.72
## 19        X1_pPeak -1.10   7.63   0.06   0.04  0.11
## 25  X1_pq_interval  1.00 121.00   7.11   5.00  7.65
## 17       X1_pre.RR 49.00 506.00 197.18 188.00 61.70
```

```
## 23         X1_qPeak -6.46   1.38  -0.22  -0.12  0.28
## 24 X1_qrs_interval  0.00 128.00  10.59   5.00  9.92
## 28   X1_qrs_morph0 -6.46   1.38  -0.22  -0.12  0.28
## 29   X1_qrs_morph1 -4.91   2.78  -0.25  -0.19  0.31
## 30   X1_qrs_morph2 -3.99   4.04  -0.32  -0.34  0.41
## 31   X1_qrs_morph3 -2.96   2.77  -0.39  -0.48  0.49
## 32   X1_qrs_morph4 -3.13   2.76  -0.54  -0.59  0.49
## 26  X1_qt_interval  4.00 290.00  28.05  24.00 14.83
## 21         X1_rPeak -2.39   4.06  -0.11  -0.09  0.40
## 22         X1_sPeak -3.27   2.82  -0.75  -0.75  0.42
## 27  X1_st_interval  1.00  86.00  10.35   9.00  4.83
## 20         X1_tPeak -2.02   3.65   0.37   0.32  0.28
```

```
variables_of_further_interest <- c("X0_post.RR", "X0_pq_interval", "X0_pre.RR", "X0_qrs_interval", "X0_q
```

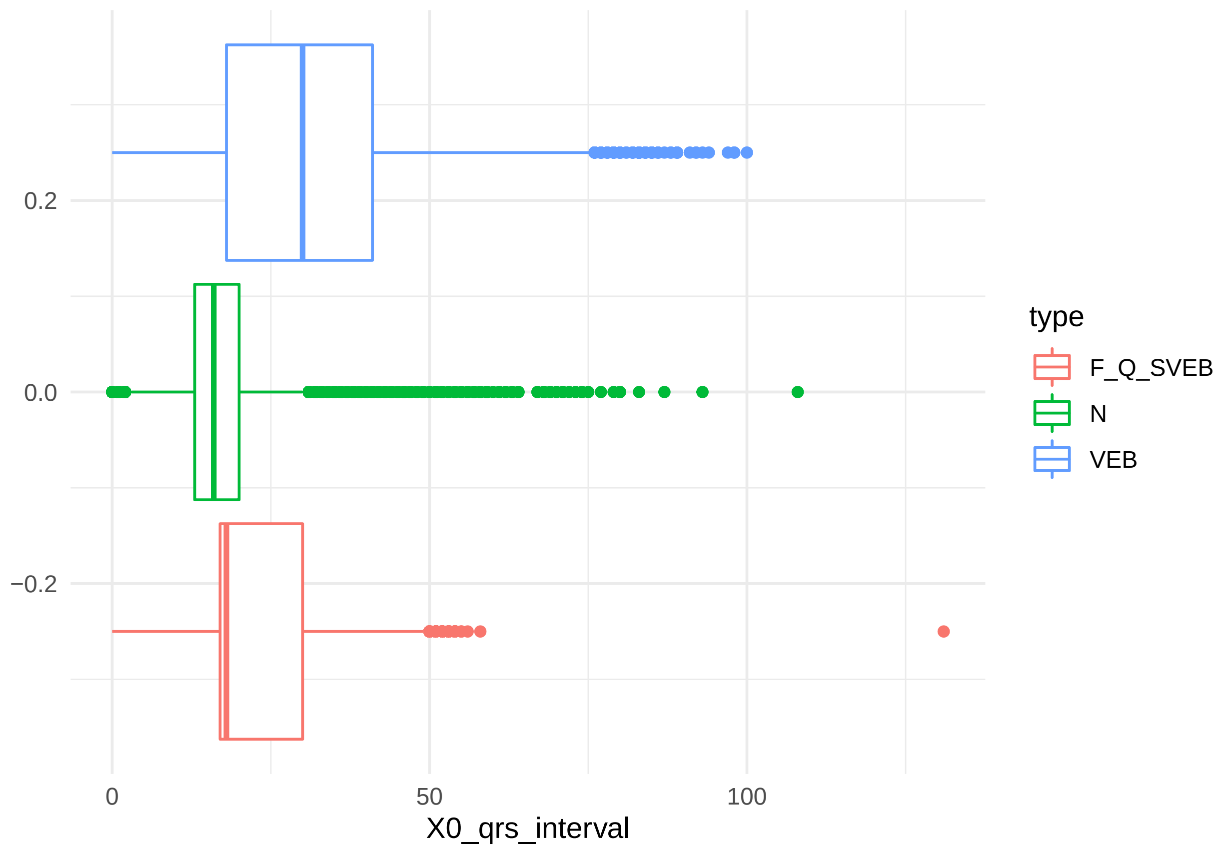
We noticed some variables which should require further analysis, those variables are:

```
variables_of_further_interest
```

```
## [1] "X0_post.RR"      "X0_pq_interval"  "X0_pre.RR"       "X0_qrs_interval"
## [5] "X0_qt_interval"  "X1_post.RR"      "X1_pre.RR"       "X1_qrs_interval"
## [9] "X1_qt_interval"
```

We will look through these variables more closely and report any irregularities.

```
ggplot(data, aes(x = X0_qrs_interval, color = type)) +
   geom_boxplot() +
   theme_minimal()
```



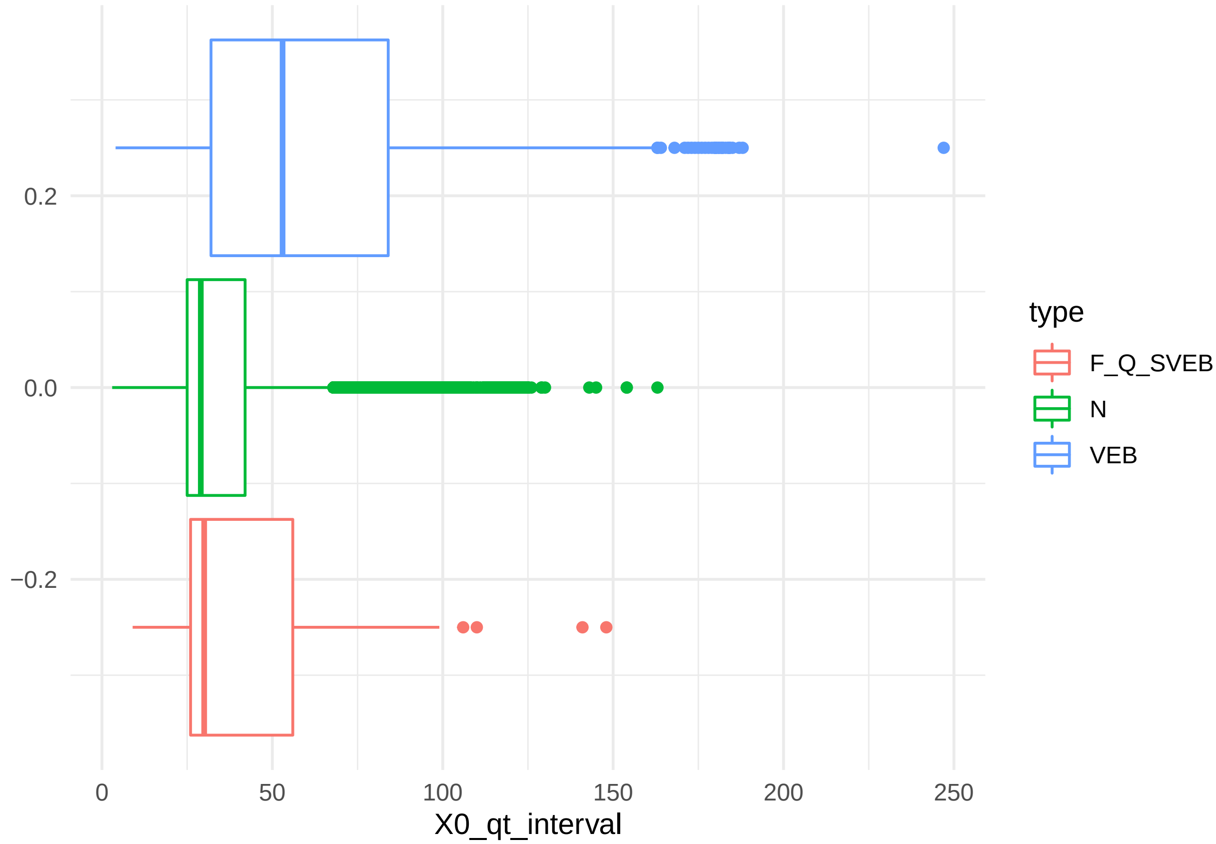

```r
data_original %>%
  filter(X0_qrs_interval > 125)
```

```
##   record type X0_pre.RR X0_post.RR   X0_pPeak X0_tPeak X0_rPeak X0_sPeak
## 1    I43    F       335        115 -0.1446686 1.394598 1.178978 1.178978
##     X0_qPeak X0_qrs_interval X0_pq_interval X0_qt_interval X0_st_interval
## 1 -0.1450307             131              2            135              2
##   X0_qrs_morph0 X0_qrs_morph1 X0_qrs_morph2 X0_qrs_morph3 X0_qrs_morph4
## 1    -0.1450307   -0.09808338   -0.04288349  -0.004493709    0.01901518
##   X1_pre.RR X1_post.RR  X1_pPeak X1_tPeak  X1_rPeak  X1_sPeak  X1_qPeak
## 1       335        115 0.1182595 1.325323 -2.114941 -2.323128 -2.114941
##   X1_qrs_interval X1_pq_interval X1_qt_interval X1_st_interval X1_qrs_morph0
## 1               2              8             26             16     -2.114941
##   X1_qrs_morph1 X1_qrs_morph2 X1_qrs_morph3 X1_qrs_morph4
## 1     -2.114941     -2.114941     -2.306017     -2.306017
```

There's quite an unusual observation in X0_qrs_interval variable, where our new class F_Q_SVEB has a value of > 125. We also see from the original dataset that this observation was classified as F. We are going to classify this observation as an outlier and remove it.

```r
data <- data %>%
  filter(X0_qrs_interval < 125)
```

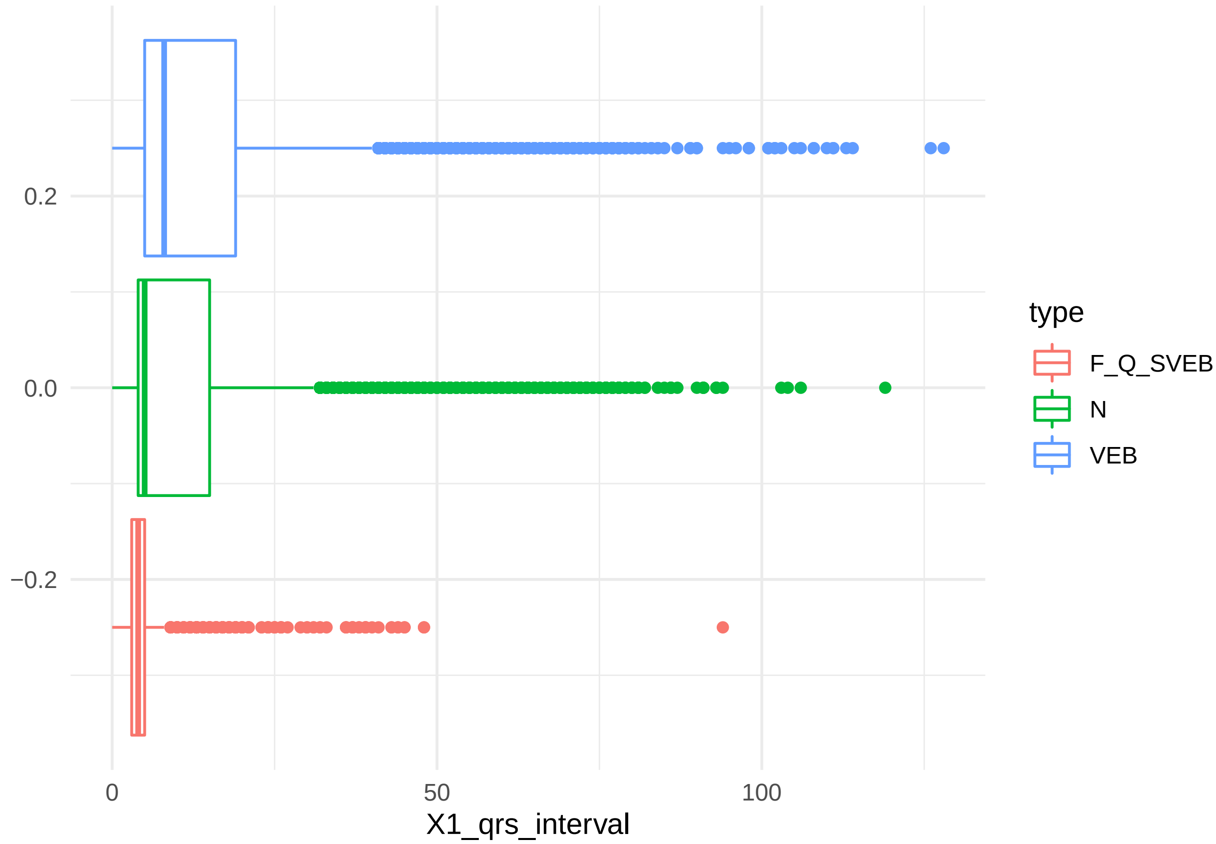
```r
ggplot(data, aes(x = X0_qt_interval, color = type)) +
  geom_boxplot() +
  theme_minimal()
```



We noticed another outlier, for the X0_qt_interval variable, when the type is VEB, the value is close to 250 which is quite unusual in this dataset. We'll remove this observation.

```r
data <- data %>%
  filter(X0_qt_interval < 225)
```

```r
ggplot(data, aes(x = X1_qrs_interval, color = type)) +
  geom_boxplot() +
  theme_minimal()
```




```
data_original %>%
  filter(X1_qrs_interval > 75, type %in% c("F", "Q", "SVEB"))
```

```
##   record type X0_pre.RR X0_post.RR   X0_pPeak  X0_tPeak   X0_rPeak   X0_sPeak
## 1    I05    F       177        367 0.02885405 0.4625619 -0.1868355 -0.2210357
##     X0_qPeak X0_qrs_interval X0_pq_interval X0_qt_interval X0_st_interval
## 1 -0.1868355               2             12             27             13
##   X0_qrs_morph0 X0_qrs_morph1 X0_qrs_morph2 X0_qrs_morph3 X0_qrs_morph4
## 1    -0.1868355    -0.1868355    -0.1868355    -0.2103347    -0.2103347
##   X1_pre.RR X1_post.RR   X1_pPeak  X1_tPeak  X1_rPeak   X1_sPeak   X1_qPeak
## 1       177        367 0.08331537 0.3157339 0.1065414 -0.1558735 -0.1257657
##   X1_qrs_interval X1_pq_interval X1_qt_interval X1_st_interval X1_qrs_morph0
## 1              94             22            123              7    -0.1257657
##   X1_qrs_morph1 X1_qrs_morph2 X1_qrs_morph3 X1_qrs_morph4
## 1   -0.09340949   -0.04265345   -0.01324232   0.006886856
```

Seems like another outlier was detected in X1_qrs_interval variable, where the type is F_Q_SVEB and the value is close to 100. From the original dataset, this observation had F type. We'll remove it.

```
data %>%
  mutate(row = row_number()) %>%
  filter(X1_qrs_interval > 75, type == "F_Q_SVEB") %>%
  select(row)
```

```
##     row
## 1 10934
```

```r
data <- data %>%
   filter(row_number() != 10934)
```

We should look at the correlation between variables

```r
# Correlation

corr_simple <- function(df,sig=0.5){
  corr <- cor(df)
  #prepare to drop duplicates and correlations of 1
  corr[lower.tri(corr,diag=TRUE)] <- NA
  #drop perfect correlations
  corr[corr == 1] <- NA
  #turn into a 3-column table
  corr <- as.data.frame(as.table(corr))
  #remove the NA values from above
  corr <- na.omit(corr)
  #select significant values
  corr <- subset(corr, abs(Freq) > sig)
  #sort by highest correlation
  corr <- corr[order(-abs(corr$Freq)),]
  return(corr)
}

correlation_matrix = cor(data %>% select(-type))

length(findCorrelation(correlation_matrix, cutoff = 0.99))
```

```
## [1] 4
```

```r
length(findCorrelation(correlation_matrix, cutoff = 0.95))
```

```
## [1] 6
```

```r
length(findCorrelation(correlation_matrix, cutoff = 0.9))
```

```
## [1] 8
```

We have 4 variables with higher than .99 correlation. We have 6 variables with higher than .95 correlation. We have 8 variables with higher than .9 correlation.

5 Pre-process

5.1 Removal of correlated variables

We've found some highly correlated variables. We don't have a lot of variables (32) and because of that, we'll only remove the variables with >.99 correlation.

```
data <- data %>%
  select(-findCorrelation(correlation_matrix, cutoff = 0.99))
```

6 Fitting models

6.1 Simple LDA

```
fitControl = trainControl(## 5-fold CV
                          method = "cv",
                          number = 5,
                          ## Estimate class probabilities
                          classProbs = TRUE,
                          ## Evaluate performance using
                          ## the following function
                          summaryFunction = multiClassSummary)
lda.fit = train(type ~ ., data = data,
                method = "lda",
                preProcess=c("center","scale"),
                trControl = fitControl)

print(lda.fit$results %>% t())
```

```
##                          1
## parameter                "none"
## logLoss                  "0.1569349"
## AUC                      "0.9660774"
## prAUC                    "0.8503791"
## Accuracy                 "0.9682119"
## Kappa                    "0.8479544"
## Mean_F1                  "0.8095788"
## Mean_Sensitivity         "0.7998741"
## Mean_Specificity         "0.9366059"
## Mean_Pos_Pred_Value      "0.8271019"
## Mean_Neg_Pred_Value      "0.9781875"
## Mean_Precision           "0.8271019"
## Mean_Recall              "0.7998741"
## Mean_Detection_Rate      "0.3227373"
## Mean_Balanced_Accuracy   "0.86824"
## logLossSD                "0.003783519"
## AUCSD                    "0.002409343"
## prAUCSD                  "0.006933635"
## AccuracySD               "0.0004901414"
## KappaSD                  "0.002839515"
## Mean_F1SD                "0.003479908"
## Mean_SensitivitySD       "0.002722242"
## Mean_SpecificitySD       "0.002274744"
## Mean_Pos_Pred_ValueSD    "0.005635765"
## Mean_Neg_Pred_ValueSD    "0.0006510096"
## Mean_PrecisionSD         "0.005635765"
## Mean_RecallSD            "0.002722242"
```

```
## Mean_Detection_RateSD    "0.0001633805"
## Mean_Balanced_AccuracySD "0.00183603"
```

AUC is 0.96, which is pretty good, the Mean F1 is 0.8 which also indicated quite a good model. The variance for AUC is 0.002, for Mean F1 - 0.003, this indicated a stable model.

```
predictions <- lda.fit %>% predict(data)
confusion_matrix <- confusionMatrix(predictions, as.factor(data$type))

confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction F_Q_SVEB      N    VEB
##   F_Q_SVEB     1328    323    846
##   N             778 152896   3230
##   VEB            75    327  15923
##
## Overall Statistics
##
##                Accuracy : 0.9683
##                  95% CI : (0.9674, 0.9691)
##     No Information Rate : 0.8738
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8481
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: F_Q_SVEB Class: N Class: VEB
## Sensitivity                 0.608895   0.9958    0.79619
## Specificity                 0.993264   0.8193    0.99742
## Pos Pred Value              0.531838   0.9745    0.97538
## Neg Pred Value              0.995076   0.9655    0.97443
## Prevalence                  0.012411   0.8738    0.11381
## Detection Rate              0.007557   0.8701    0.09061
## Detection Prevalence        0.014210   0.8929    0.09290
## Balanced Accuracy           0.801079   0.9075    0.89680
```

```
classAcc(confusion_matrix)
```

```
## F_Q_SVEB        N      VEB
##     60.9     99.6     79.6
```

The overall accuracy is pretty good (96.8%), the accuracies for each class are: F_Q_SVEB = 60.9%, N = 99.6%, VEB = 79.6%.

6.2 Simple QDA

```r
fitControl <- trainControl(# 5-fold CV
                           method = "cv",
                           number = 5,
                           # Estimate class probabilities
                           classProbs = TRUE,
                           # Evaluate performance using
                           # the following function
                           summaryFunction = multiClassSummary,
                           verboseIter = TRUE)
# qda.fit <- train(type ~ ., data = data,
#                  method = "qda",
#                  preProcess=c("center","scale"),
#                  trControl = fitControl)
```

Can't fit a QDA model because of our small class F_Q_SVEB.

6.3 Simple RDA

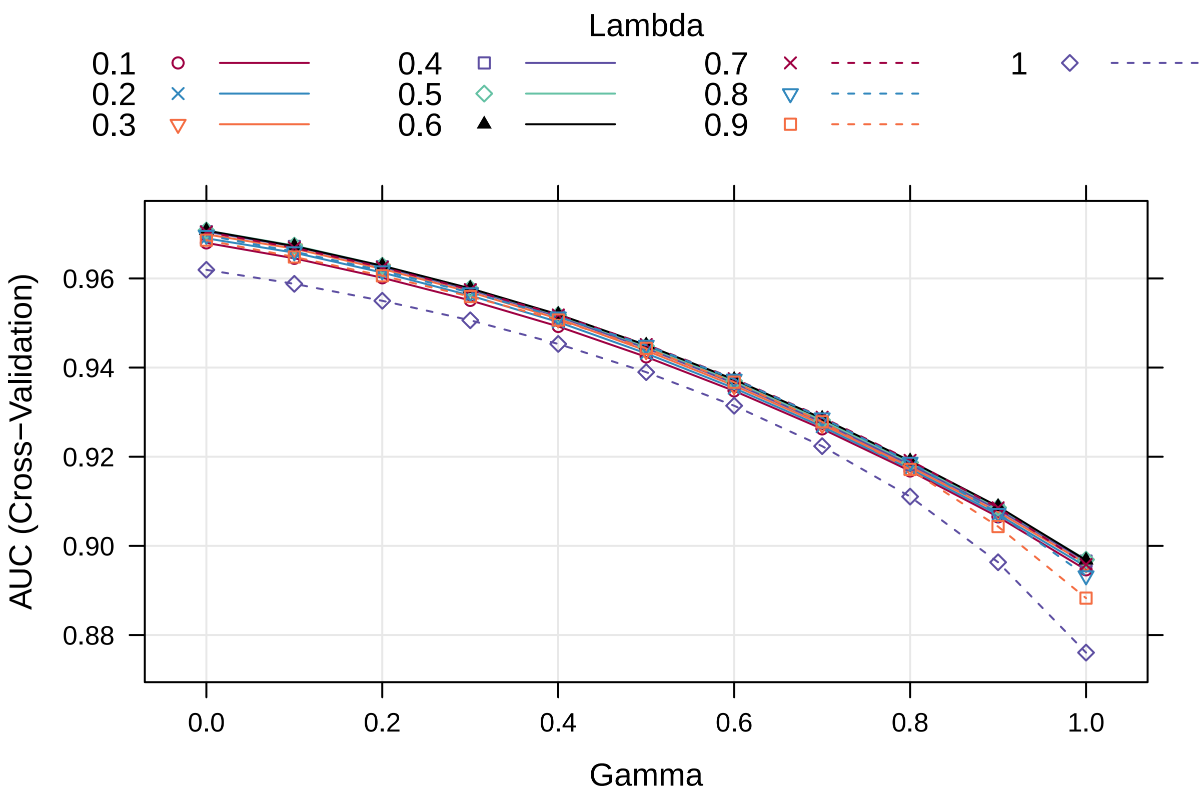
```r
library(tictoc)
tic()
fitControl <- trainControl(## 5-fold CV
                           method = "cv",
                           number = 5,
                           ## Estimate class probabilities
                           classProbs = TRUE,
                           ## Evaluate performance using
                           ## the following function
                           summaryFunction = multiClassSummary,
                           savePredictions="all",
                           verboseIter = TRUE)
rdaGrid =  expand.grid(lambda = seq(0.1,1,0.1),
                        gamma = seq(0,1,0.1))
rda.fit <- train(type ~ ., data = data,
                 method = "rda",
                 trControl = fitControl,
                 preProcess=c("center", "scale","pca"),
                 ## Now specify the exact models
                 ## to evaluate:
                 tuneGrid = rdaGrid)
```

```
## Aggregating results
## Selecting tuning parameters
## Fitting gamma = 0, lambda = 0.9 on full training set
```

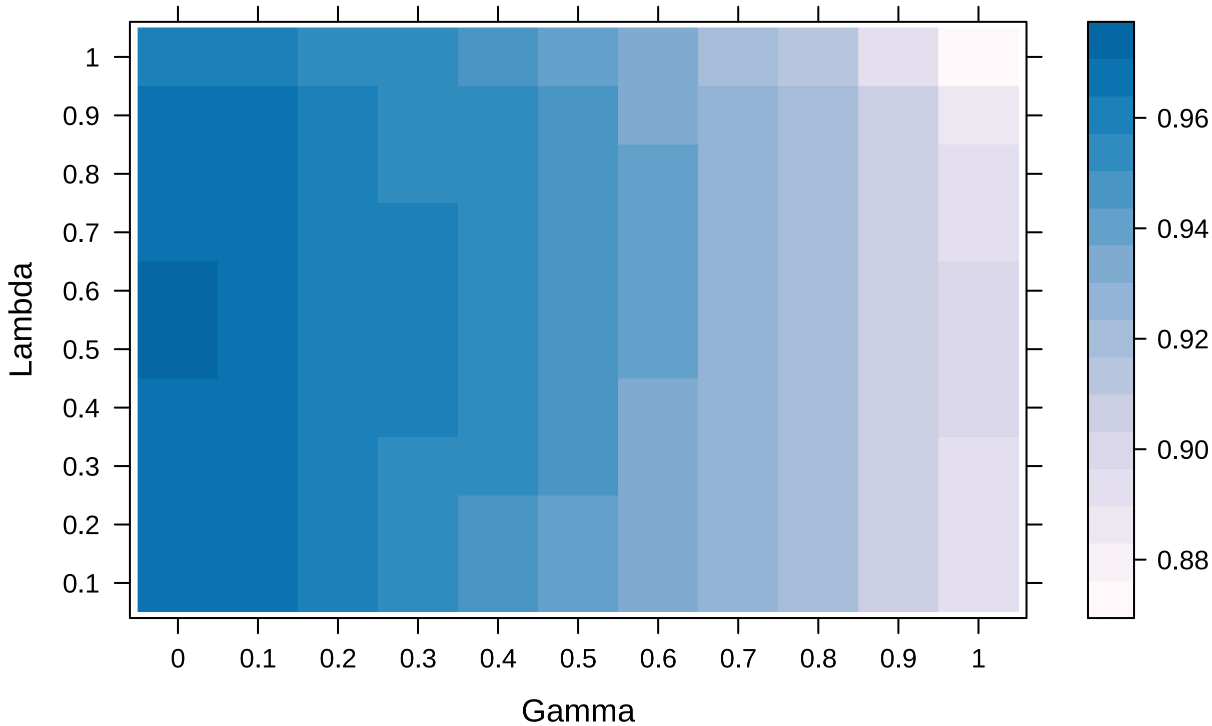
```r
toc()
```

```
## 1807.66 sec elapsed
```

```
trellis.par.set(caretTheme())
plot(rda.fit, metric = "AUC")
```




```
plot(rda.fit, metric = "AUC", plotType = "level")
```



AUC (Cross−Validation)

From those 2 graphs, we can see that the best cross-validation AUC is achieved using gamma - 0 and Lambda 0.5 or 0.6.

```
resamp = rda.fit$pred %>%
    filter(gamma == 0 & lambda == 0.6)
confusion_matrix <- confusionMatrix(resamp$pred, resamp$obs)
confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction F_Q_SVEB      N    VEB
##   F_Q_SVEB     1309    329    932
##   N             725 150230   2912
##   VEB           147   2987  16155
##
## Overall Statistics
##
##                Accuracy : 0.9543
##                  95% CI : (0.9533, 0.9553)
##     No Information Rate : 0.8738
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7943
##
##  Mcnemar's Test P-Value : < 2.2e-16
```

```
##
## Statistics by Class:
##
##                      Class: F_Q_SVEB Class: N Class: VEB
## Sensitivity                 0.600183   0.9784    0.80779
## Specificity                 0.992734   0.8360    0.97988
## Pos Pred Value              0.509339   0.9764    0.83752
## Neg Pred Value              0.994964   0.8483    0.97543
## Prevalence                  0.012411   0.8738    0.11381
## Detection Rate              0.007449   0.8549    0.09193
## Detection Prevalence        0.014625   0.8756    0.10977
## Balanced Accuracy           0.796459   0.9072    0.89383
```

```
classAcc(confusion_matrix)
```

```
## F_Q_SVEB        N      VEB
##     60.0     97.8     80.8
```

Overall accuracy using gamma - 0 and lambda - 0.6 is 95.4%, while accuracies for each class are: F_Q_SVEB = 60.1%, N = 97.8% and VEB = 80.8%.

6.4 CART

```
fitControl <- trainControl(## 5-fold CV
                           method = "cv",
                           number = 5,
                           ## Estimate class probabilities
                           classProbs = TRUE,
                           ## Evaluate performance using
                           ## the following function
                           summaryFunction = multiClassSummary,
                           savePredictions="all",
                           verboseIter = TRUE)
cart.fit <- train(type ~ ., data = data,
                  method = "rpart",
                  trControl = fitControl,
                  preProcess=c("center", "scale"))
```

```
## Aggregating results
## Selecting tuning parameters
## Fitting cp = 0.0357 on full training set
```

```
cart.fit
```

```
## CART
##
## 175726 samples
##     28 predictor
##      3 classes: 'F_Q_SVEB', 'N', 'VEB'
##
```

```
## Pre-processing: centered (28), scaled (28)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 140579, 140581, 140582, 140581, 140581
## Resampling results across tuning parameters:
##
##   cp          logLoss    AUC        prAUC       Accuracy   Kappa      Mean_F1
##   0.03566276  0.1786449  0.8252275  0.30794981  0.9605920  0.8052525  NaN
##   0.04332732  0.1976976  0.8122371  0.17183438  0.9546681  0.7696368  NaN
##   0.31169973  0.2915470  0.6851202  0.05012975  0.9208718  0.4535553  NaN
##   Mean_Sensitivity  Mean_Specificity  Mean_Pos_Pred_Value  Mean_Neg_Pred_Value
##   0.5990333         0.9166634         NaN                  0.9726729
##   0.5799575         0.8991634         NaN                  0.9714899
##   0.4774082         0.8029081         NaN                  0.9696886
##   Mean_Precision  Mean_Recall  Mean_Detection_Rate  Mean_Balanced_Accuracy
##   NaN             0.5990333    0.3201973            0.7578484
##   NaN             0.5799575    0.3182227            0.7395605
##   NaN             0.4774082    0.3069573            0.6401581
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.03566276.
```

```
predictions <- cart.fit %>% predict(data)
confusion_matrix <- confusionMatrix(predictions, as.factor(data$type))

confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction F_Q_SVEB      N    VEB
##   F_Q_SVEB        0      0      0
##   N            1416 152877   4542
##   VEB           765    669  15457
##
## Overall Statistics
##
##                Accuracy : 0.9579
##                  95% CI : (0.957, 0.9589)
##     No Information Rate : 0.8738
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7889
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: F_Q_SVEB Class: N Class: VEB
## Sensitivity                  0.00000   0.9956    0.77289
## Specificity                  1.00000   0.7314    0.99079
## Pos Pred Value                   NaN   0.9625    0.91510
## Neg Pred Value               0.98759   0.9604    0.97140
## Prevalence                   0.01241   0.8738    0.11381
## Detection Rate               0.00000   0.8700    0.08796
```

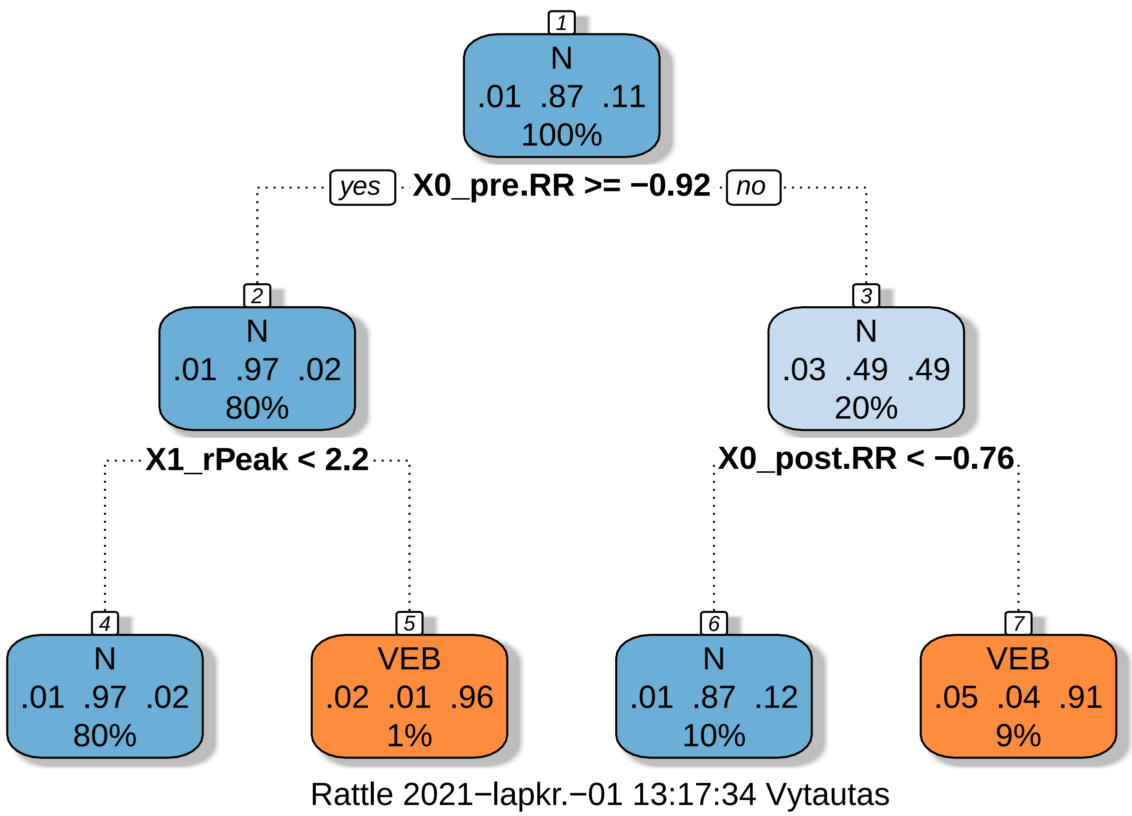
```
## Detection Prevalence          0.00000   0.9039    0.09612
## Balanced Accuracy             0.50000   0.8635    0.88184
```

```
classAcc(confusion_matrix)
```

```
## F_Q_SVEB        N      VEB
##      0.0    99.6     77.3
```

Our overall accuracy is 95.8%, but if we would look at each class accuracies: F_Q_SVEB = 0%, N = 99.6% and VEB = 77.3% we could see that our newly created class has 0% accuracy.

```
fancyRpartPlot(cart.fit$finalModel)
```



We can see that our simple decision tree is too simple for our data.

7 Comparison of models

```
resamps = resamples(list(LDA = lda.fit, RDA = rda.fit, CART = cart.fit))
summary(resamps)
```

```
##
## Call:
```

```
## summary.resamples(object = resamps)
##
## Models: LDA, RDA, CART
## Number of resamples: 5
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LDA  0.9676777 0.9678484 0.9681311 0.9682119 0.9685304 0.9688718    0
## RDA  0.9617606 0.9623844 0.9626394 0.9626179 0.9630673 0.9632380    0
## CART 0.9569770 0.9585432 0.9616446 0.9605920 0.9627564 0.9630388    0
##
## AUC
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LDA  0.9635871 0.9636174 0.9668114 0.9660774 0.9673088 0.9690626    0
## RDA  0.9665374 0.9671713 0.9692470 0.9685768 0.9695154 0.9704131    0
## CART 0.8157842 0.8190693 0.8234511 0.8252275 0.8307674 0.8370653    0
##
## Kappa
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LDA  0.8451663 0.8456669 0.8472608 0.8479544 0.8498086 0.8518692    0
## RDA  0.8173951 0.8201225 0.8212765 0.8213531 0.8236918 0.8242795    0
## CART 0.7839078 0.7919512 0.8121432 0.8052525 0.8174476 0.8208126    0
##
## logLoss
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LDA  0.1512249 0.1562215 0.1564785 0.1569349 0.1600049 0.1607446    0
## RDA  0.1518779 0.1528204 0.1537549 0.1575828 0.1646560 0.1648047    0
## CART 0.1713400 0.1737416 0.1759465 0.1786449 0.1841291 0.1880673    0
##
## Mean_Balanced_Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LDA  0.8656041 0.8670207 0.8692831 0.8682400 0.8695879 0.8697044    0
## RDA  0.8529640 0.8548076 0.8567340 0.8577915 0.8608292 0.8636224    0
## CART 0.7467136 0.7494044 0.7620201 0.7578484 0.7635739 0.7675297    0
##
## Mean_Detection_Rate
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LDA  0.3225592 0.3226161 0.3227104 0.3227373 0.3228435 0.3229573    0
## RDA  0.3205869 0.3207948 0.3208798 0.3208726 0.3210224 0.3210793    0
## CART 0.3189923 0.3195144 0.3205482 0.3201973 0.3209188 0.3210129    0
##
## Mean_F1
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LDA  0.8066834 0.8069488 0.8076621 0.8095788 0.8123586 0.8142412    0
## RDA  0.7869788 0.7875537 0.7951355 0.7953770 0.8028126 0.8044042    0
## CART        NA        NA        NA       NaN        NA        NA    5
##
## Mean_Neg_Pred_Value
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LDA  0.9778328 0.9778440 0.9778449 0.9781875 0.9780785 0.9793374    0
## RDA  0.9678969 0.9684508 0.9686151 0.9688364 0.9695881 0.9696311    0
## CART 0.9712291 0.9717812 0.9724677 0.9726729 0.9736701 0.9742164    0
##
## Mean_Pos_Pred_Value
```

```
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LDA  0.8211892 0.8218708 0.8268752 0.8271019 0.8326699 0.8329045    0
## RDA  0.8014779 0.8057200 0.8123176 0.8116519 0.8192844 0.8194593    0
## CART        NA        NA        NA       NaN        NA        NA    5
##
## Mean_Precision
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LDA  0.8211892 0.8218708 0.8268752 0.8271019 0.8326699 0.8329045    0
## RDA  0.8014779 0.8057200 0.8123176 0.8116519 0.8192844 0.8194593    0
## CART        NA        NA        NA       NaN        NA        NA    5
##
## Mean_Recall
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LDA  0.7957643 0.7997502 0.7998441 0.7998741 0.8006556 0.8033562    0
## RDA  0.7755548 0.7813015 0.7850337 0.7866291 0.7934752 0.7977805    0
## CART 0.5879250 0.5905707 0.6028431 0.5990333 0.6054517 0.6083760    0
##
## Mean_Sensitivity
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LDA  0.7957643 0.7997502 0.7998441 0.7998741 0.8006556 0.8033562    0
## RDA  0.7755548 0.7813015 0.7850337 0.7866291 0.7934752 0.7977805    0
## CART 0.5879250 0.5905707 0.6028431 0.5990333 0.6054517 0.6083760    0
##
## Mean_Specificity
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LDA  0.9342911 0.9352099 0.9354438 0.9366059 0.9387532 0.9393316    0
## RDA  0.9281833 0.9283138 0.9284344 0.9289538 0.9294643 0.9303733    0
## CART 0.9055023 0.9082380 0.9211972 0.9166634 0.9216961 0.9266835    0
##
## prAUC
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LDA  0.8429418 0.8447308 0.8498277 0.8503791 0.8547689 0.8596263    0
## RDA  0.8392596 0.8413105 0.8433615 0.8479929 0.8523596 0.8613577    2
## CART 0.3052808 0.3057780 0.3065798 0.3079498 0.3098714 0.3122391    0
```

Both LDA and RDA models gave very similar results, resulting in ~60% accuracy for F_Q_SVEB, ~99% for N and ~80% for VEB classes. Based on Mean F1, simple LDA model is slightly better than RDA, where LDA had 0.81 mean F1 and RDA had 0.79.

Decision tree performed worse, the model could not predict F_Q_SVEB class at all.