Comp 3 ex. 3

Group 12

2022-11-16

```r
# Load all necessary libraries
library('readr')
library('forecast')
library('lubridate')
library('tsfeatures')
library('ggplot2')
library('dplyr')
library('caret')
library('tsfeatures')
library('mltools')
library('ggpubr')
```

a)

Start by loading the data:

```r
# Load training data
train = read_csv(
  "/Users/martinbergsholmnesse/Documents/NMBU/DAT320/comp3/hs_train.csv")
```

Print the dimension of the data.

```r
# Dimensions of data
dim(train)
```

```
## [1] 10915   189
```

We see that we have 10915 observations and 189 features, which is as expected. Due to the large amount of features, we choose to include the summary of only the first three features.

```r
# Summary of data of the first three features
summary(train[1])
```

```
##        V1
##  Min.   :0.6414
##  1st Qu.:0.9588
##  Median :1.0000
##  Mean   :0.9765
##  3rd Qu.:1.0000
##  Max.   :1.0000
```

```r
summary(train[2])
```

```
##        V2
##  Min.   :0.0000
##  1st Qu.:0.5869
##  Median :0.7430
##  Mean   :0.7239
##  3rd Qu.:0.8842
##  Max.   :1.0000
```

```r
summary(train[3])
```

```
##        V3
##  Min.   :0.0000
##  1st Qu.:0.2137
##  Median :0.3720
##  Mean   :0.4046
##  3rd Qu.:0.5596
##  Max.   :1.0000
```

From the summaries we see that the minimum values vary, but the maximum is the same. We also print the unique classes to check that it is in fact only two classes.

```r
# Unique classes
unique(train$class)
```

```
## [1] "normal"   "abnormal"
```

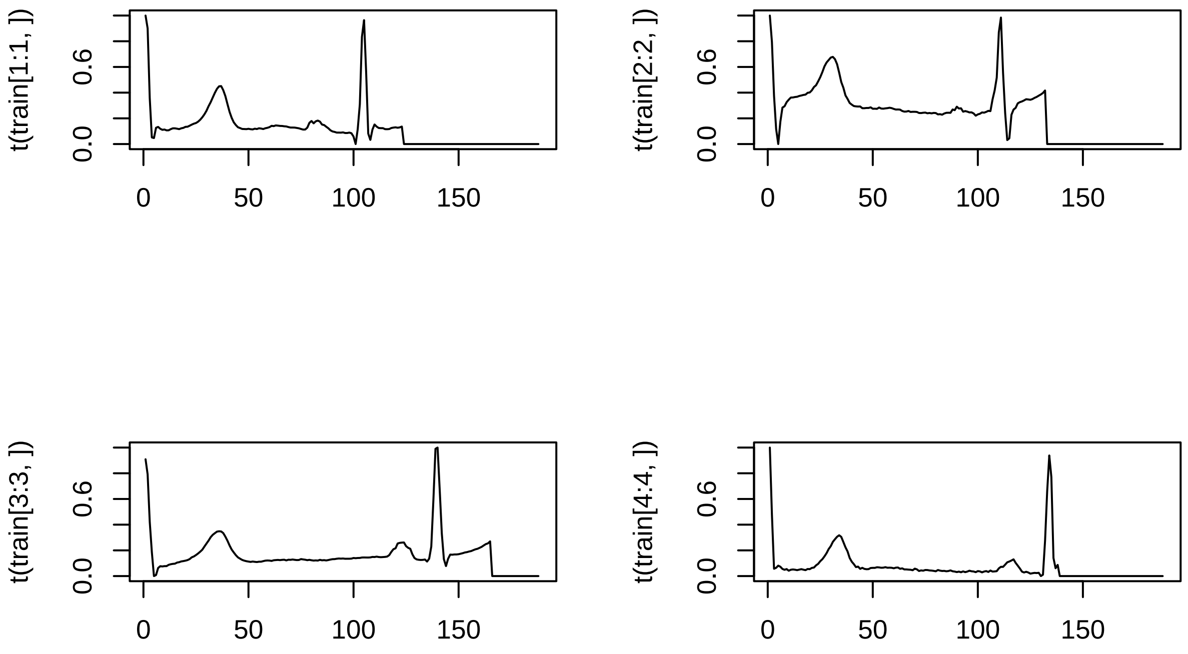
As expected, the two classes are “normal” and “abnormal”. We also check for missing values.

```r
# check for missing values (none missing)
sum(is.na(train))
```

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

Find that no values are missing.

```r
# Plot first four rows
par(mfrow=c(2,2))
matplot(t(train[1:1, ]),type="l")
matplot(t(train[2:2, ]),type="l")
matplot(t(train[3:3, ]),type="l")
matplot(t(train[4:4, ]),type="l")
```

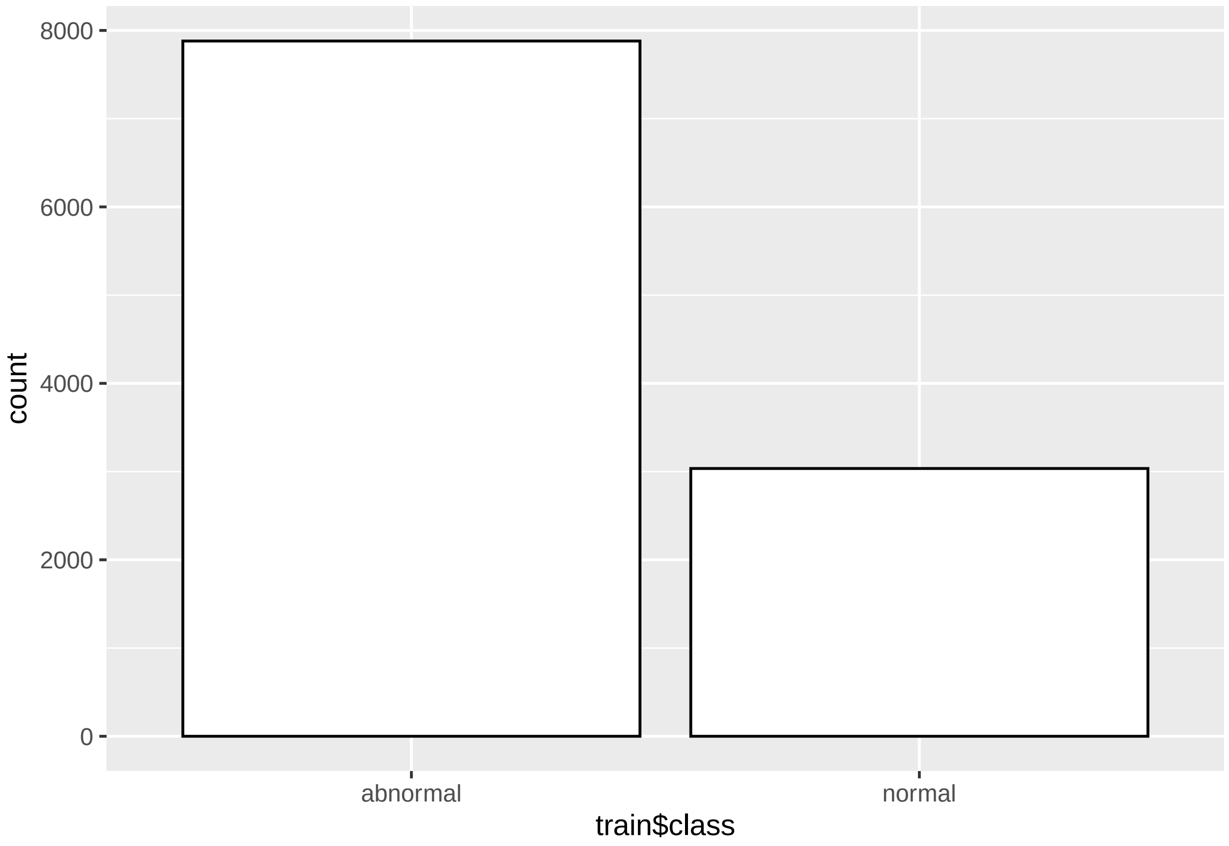


The

plot shows the first four observations, we see that they are similar, but not identical.

We also visualize the distribution of normal and abnormal heart sequences.

```r
# Plot distribution of abnormal/normal
train %>% ggplot(aes(x = train$class)) +
  geom_bar(fill = "white", color = "black")
```



From the plot we see that the data is not evenly distributed. A dummy-classifier predicting everything as abnormal should be able to achieve an accuracy of about 70%. Meaning that our models should perform better than this.

Due to the high number of features we did not include test for trend in every time series or ACF/PACF.

b)

Start by removing the id-column since it provides no additional information. Also remove the class-column.

```r
# Remove id-column
train$id <- NULL
# Remove class column and convert to matrix
train_mat <- train[-c(1, ncol(train))] %>% as.matrix() # cast to matrix
```

Perform the feature extraction.

```r
# feature extraction
feat.mat <- tsfeatures(t(train_mat)) %>% as.data.frame()
feat.mat <- feat.mat %>%
  select_if(~n_distinct(.) > 1)
```

```
colnames(feat.mat)
```

```
##  [1] "trend"       "spike"       "linearity"   "curvature"   "e_acf1"
##  [6] "e_acf10"     "entropy"     "x_acf1"      "x_acf10"     "diff1_acf1"
## [11] "diff1_acf10" "diff2_acf1"  "diff2_acf10"
```

From the feature extraction we extract 13 different features, which are printed out above.

c)

We standardize the input data, which may improve the results of the logistic regression.

```
# Standardize data [0,1]
# Improved results for the Logistic Regression
feat.mat <- scale(feat.mat)
```

Set up a 10-fold cross validation, which repeats three times. The cross-validation also performs random hyperparameter tuning, which gave better results than grid tuning.

```
# 10 fold cross validation with random hyperparameter search
# Random hyperparameter got better results than other option, grid.
fitControl <- trainControl(
  method = "repeatedcv",
  number = 10,
  repeats = 3, # not too high due to poor computer
  search = "random")
```

Start training the random forest, using standardized input, cross-validation and hyperparameter tuning.

```
# Train random forest
rf <- train(as.data.frame(feat.mat), as.vector(train$class),
            method = "rf",
            trControl = fitControl,
            tunelength = 5,
            verbose = FALSE)
```

```
# Results Random Forest
rf$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, tunelength = 5,      verbose = FALSE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.29%
## Confusion matrix:
##          abnormal normal class.error
## abnormal     7868     12 0.001522843
## normal         20   3015 0.006589786
```

From the cross-validation the validation accuracy is very high. Almost all of the predictions are correct.

Then we train the logistic regression.

```
# Train Logistic Regression
lr <- train(as.data.frame(feat.mat), as.vector(train$class),
              method = "glmnet",
```

```r
              trControl = fitControl,
              tunelength = 30,
              verbose = FALSE)
```

We did not include a model summary for the logistic regression due to messy output.

Load the test data.

```r
# Load test data
test = read_csv(
  "/Users/martinbergsholmnesse/Documents/NMBU/DAT320/comp3/hs_test.csv")
```

Extracting features from the test data as input for the trained models.

```r
# Preprocess input data
test_mat <- test[-c(1, ncol(test))] %>% as.matrix() # cast to matrix
feat.mat_test <- tsfeatures(t(test_mat)) %>% as.data.frame()
feat.mat_test <- feat.mat_test %>%
  select_if(~n_distinct(.) > 1)
```

Perform predictions with random forest and logistic regression

```r
# Perform predictions
rf.pred <- predict(rf, newdata = feat.mat_test) # Random forest predictions
lr.pred <- predict(lr, newdata = feat.mat_test) # Logistic predictions
```

Need to convert the data to same format as the predictions in order to compare results.

```r
# Convert test data to same type as predictions
test$class = as.factor(test$class)
```

Confusion matrix from the random forest.

```r
# Random Forest: confusion matrix with accuracy
caret::confusionMatrix(rf.pred, test$class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction abnormal normal
##   abnormal     1670      3
##   normal        956   1008
##
##                Accuracy : 0.7363
##                  95% CI : (0.7217, 0.7506)
##     No Information Rate : 0.722
##     P-Value [Acc > NIR] : 0.02784
##
##                   Kappa : 0.4907
##
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 0.6359
##             Specificity : 0.9970
##          Pos Pred Value : 0.9982
##          Neg Pred Value : 0.5132
##              Prevalence : 0.7220
##          Detection Rate : 0.4592
##    Detection Prevalence : 0.4600
##       Balanced Accuracy : 0.8165
##
##        'Positive' Class : abnormal
##
```

From the confusion matrix we see that the random forest model classifies a lot of the abnormal heart sequences as normal, but rarely the other way around. The high number of misclassifications results in a poor accuracy barely better than a dummy-classifier.

```
# Random forest F1
caret::F_meas(rf.pred, test$class)
```

```
## [1] 0.7769249
```

The random forest classifier achieves a F1-score of 0.81.

Then we construct a confusion matrix for the logistic regression.

```
# Logistic Regression: confusion matrix with accuracy
caret::confusionMatrix(lr.pred, test$class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction abnormal normal
##   abnormal     2626   1006
##   normal          0      5
##
##                Accuracy : 0.7234
##                  95% CI : (0.7085, 0.7379)
##     No Information Rate : 0.722
##     P-Value [Acc > NIR] : 0.4349
##
##                   Kappa : 0.0071
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.000000
##             Specificity : 0.004946
##          Pos Pred Value : 0.723018
##          Neg Pred Value : 1.000000
##              Prevalence : 0.722024
##          Detection Rate : 0.722024
##    Detection Prevalence : 0.998625
##       Balanced Accuracy : 0.502473
##
##        'Positive' Class : abnormal
##
```

The results from the logistic regression are much better than for the random forest. This may be due to the tune length which is 30, compared to 5 for the random forest. Due to old computer we could not increase the tune length for the random forest. The tune length is the number of different values to try for each hyperparameter. For logistic regression we tuned using 30 different values, compared to 5 for the random forest.

The accuracy for the logistic regression is 98%, but it classified a lot more normal heart sequence as abnormal, than the other way around.

```
# Logistic regression F1
caret::F_meas(lr.pred, test$class)
```

```
## [1] 0.8392458
```

As a result the F1 for the logistic regression is 98.65%, which is a lot higher than for the random forest.

d)

Print the importance of features from the random forest

```
# evaluate feature importances random forest
rf$finalModel$importance
```

```
##             MeanDecreaseGini
## trend               4.616166
## spike               5.481525
## linearity          10.684573
## curvature           8.094780
## e_acf1           1183.983774
## e_acf10           211.449550
## entropy             7.510628
## x_acf1             13.041488
## x_acf10             8.060752
## diff1_acf1        422.899991
## diff1_acf10      2355.345916
## diff2_acf1         41.301038
## diff2_acf10       109.029761
```

From the results above we see that the diff1_acf10, which is the ACF of lag = ten, first differenced, is most important. Followed by the sum of the first ten squared autocorrelation coefficients. Third, the ACF of the first differenced with lag = 1.

```
# evaluate feature importances logistic regression
varImp(lr, scale = FALSE)
```

```
## glmnet variable importance
##
##             Overall
## e_acf1       0.8095
## diff1_acf10  0.7835
## curvature    0.0000
## diff2_acf1   0.0000
## x_acf10      0.0000
## e_acf10      0.0000
## diff1_acf1   0.0000
## diff2_acf10  0.0000
## linearity    0.0000
## spike        0.0000
## entropy      0.0000
## x_acf1       0.0000
## trend        0.0000
```

The most important feautures for the logistic regression are the same as for random forest, but e_acf1 is more important than diff1_acf10. For the random forest diff1_acf10 is more important than e_acf1. diff1_acf1 is the fourth most important feature for the logistic regression, while it is the third most important for the random forest. The third most important feature for the logistic regression is x_acf1.