Case 1

Damage detection of the Valdemar platform model

Kasper Juul Jensen, Lars Roed Ingerslev, Maya Katrin Gussmann

March 21, 2016

Please upload a file `yhat.txt` containing predictions $\{0, 1, 2, 3\}$ as one long vector with newline separating the observations. The corresponding order is that given in the Cas1_tst (Xt), and a small report with a brief introduction/abstract, pre-processing, modeling and model assessment, plus your evaluation of the actual dimensions needed to describe the data/problem.

Please note that the deadline is mid-day.

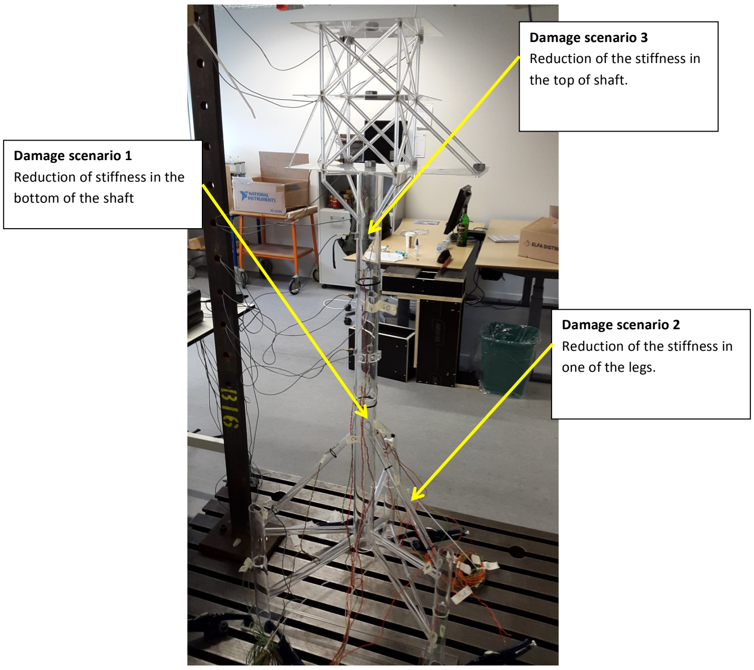
1. Introduction

There are three sensors on an offshore platform, that record data for detecting damage to the platform. Damage can occur at three different sites (see table 1.1) and for three different intensities $(5\%, 10\%, 15\%)$. The recorded data is given in the form of Frequency Response Functions (FRFs).

Class	Description
0	undamaged
1	damage in the bottom of the shaft
2	damage in one of the legs
3	damage in the top of the shaft

Table 1.1.: Damage classes.

In this case, 4092 samples of the three FRFs are given, together with their respective damage classes. The goal is to form a model to predict damage and damage class.



2. Pre-processing

2.1. Cross validation

2.2. PLS

3. Modeling

In the following sections we will look into different models. The final damage class will then be chosen based on a majority vote of the models.

3.1. KNN

3.2. Logistic regression

3.3. SVM

3.4. CART

3.5. Boosting

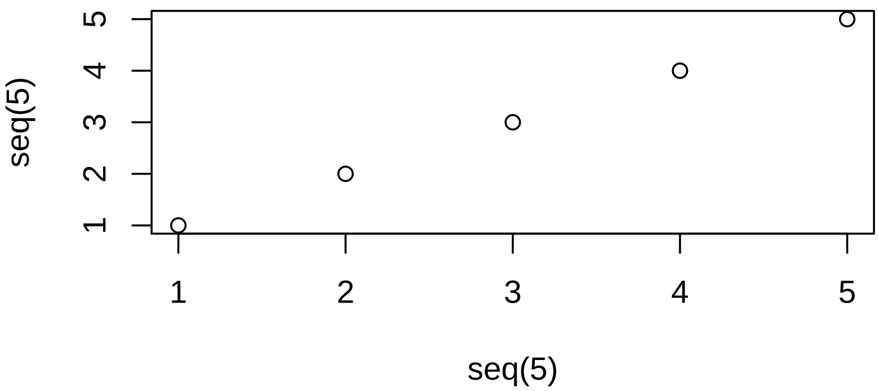
3.6. Random Forest

3.7. Majority Vote

4. Dimensions

5. Test

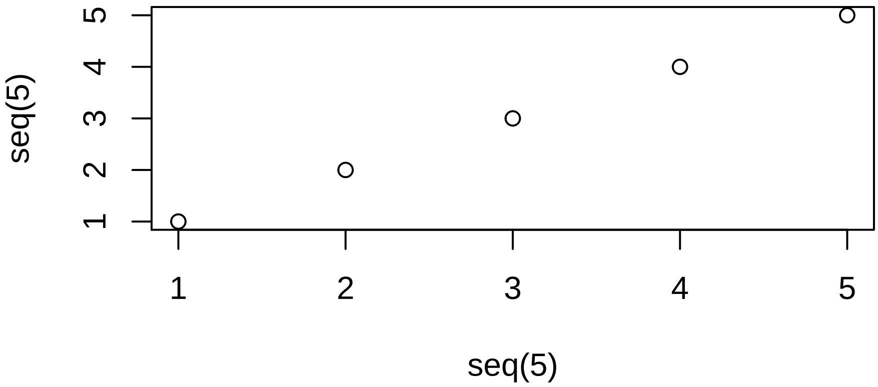
```
[1] 2
```




```
1 + 1

[1] 2

plot(seq(5),seq(5))
```




```
# you have to start a chunk with ## ----
# and then add the name of the chunk, if you want to have it in another file
# then the file has to be read by read_chunk in the beginning
```

Appendices

A. Pre-processing

```
library(reshape2)
library(data.table)
library(cvTools)
library(class)
library(nnet)
library(e1071)
library(pls)
library(rpart)


load("Case1.RData")

setup <- data.table(
    sensor = rep(rep(c("S1", "S2", "S3"), each = 4092), 2),
    freq = rep(1:4092, 6),
    part = rep(c("real", "imaginary"), each = 4092*3)
)
setup[, idx:=seq_len(nrow(setup))]


set.seed(1)
nObs <- nrow(Xtr)
nFolds <- 10
folds <- cvFolds(nObs, nFolds)

subsetData <- function(i, x, y, folds)
{
    ind <- folds$which == i
    xTrain <- x[!ind, ]
    yTrain <- y[!ind]
    xTest <- x[ind, ]
    yTest <- y[ind]
    list(xTrain = xTrain,
         yTrain = yTrain,
         xTest = xTest,
         yTest = yTest)
```

```r
}


rotateData <- function(xTrain, yTrain, xTest, yTest)
{
    # Center, but do no scale x
    xTrain <- scale(xTrain, scale = FALSE)
    xTest <- scale(xTest, center = attr(xTrain, "scaled:center"), scale = FALSE)

    # Reshape y to be a 4 column matrix and do PLS
    y <- as.data.frame(lapply(levels(yTrain), function(x) as.integer(yTrain == x)))
    pls <- plsr(as.matrix(y) ~ xTrain)

    # shuffle xTrain, do PLS again and find the maximum explained variance by the shuffled
    xShuf <- apply(xTrain, 2, function(x)x[sample(length(x))])
    plsShuf <- plsr(as.matrix(y) ~ xShuf)
    cutOff <- max(explvar(plsShuf))

    nComp <- max(which(explvar(pls) > cutOff))
    pls2 <- plsr(as.matrix(y) ~ xTrain, ncomp = nComp)
    # Select only components with a higher explained variance
    xTrainRot <- pls2$scores
    xTestRot <- predict(pls2, xTest, type = "scores")

    list(xTrain = xTrainRot,
         yTrain = yTrain,
         xTest = xTestRot,
         yTest = yTest)
}

makeSkeleton <- function(i, x, y, folds)
{
    data <- subsetData(i, x, y, folds)
    list(
        i = i,
        data = with(data, rotateData(xTrain, yTrain, xTest, yTest)),
        predictions = data.frame(matrix(nrow = length(data$yTest), ncol = 0)),
        errorRate = numeric()
    )
}

dataList <- lapply(seq_len(nFolds), makeSkeleton, Xtr, class_tr, folds)
save(dataList, folds, file = "preprocessed.RData")
```