Functional Data with `mlr`

Florian Pfisterer & Xudong Sun & Laura Beggel

June 28, 2018

LMU Munich
Working Group Computational Statistics

Table of contents

Introduction

Functional Data

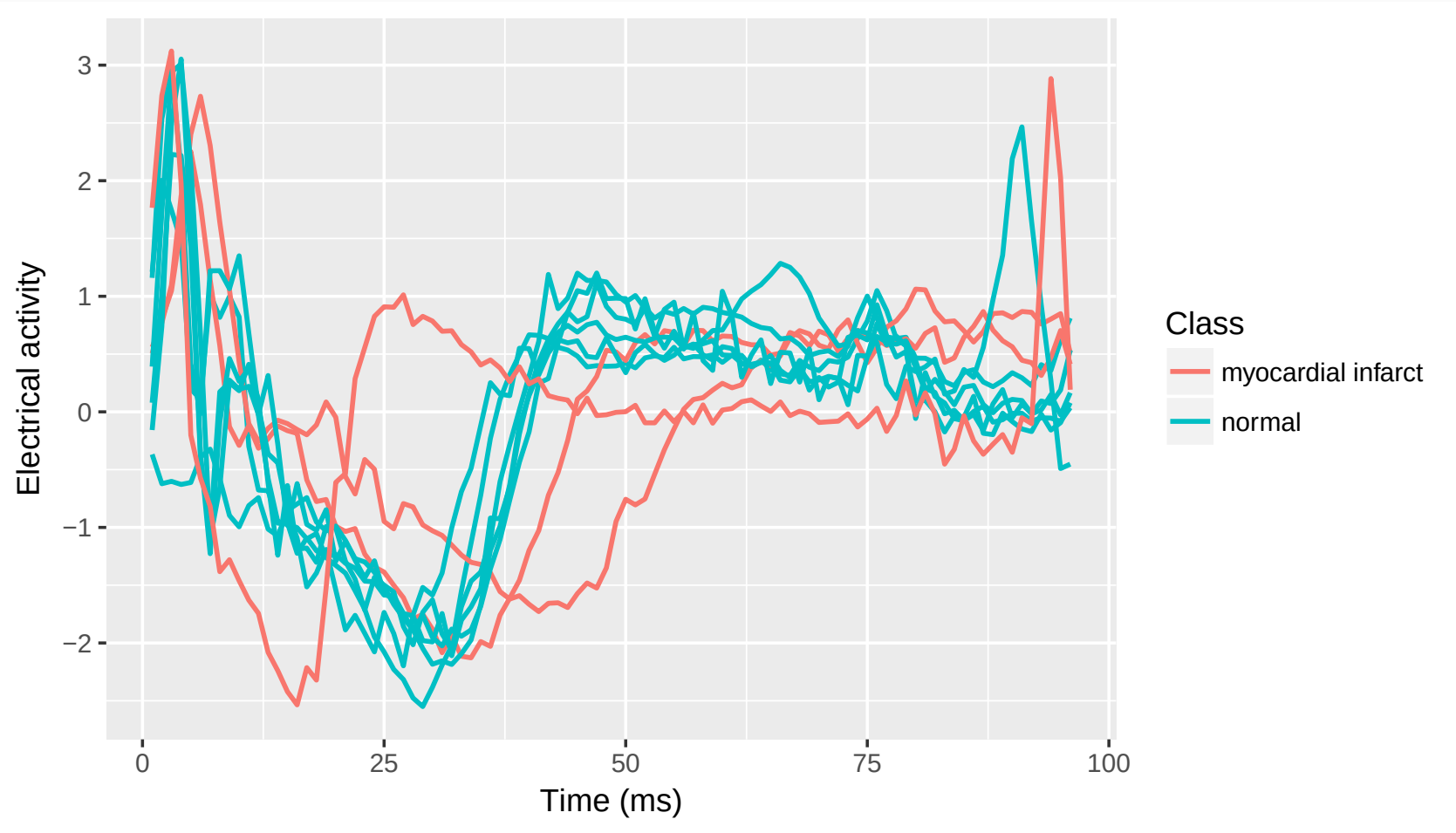


Figure 1: Electro-Cardiogram measurements during a heartbeat (Olszewski, 2001)

Functional Data in mlr

Example: Fuelsubset data from FDboost package

```
data(fuelSubset, package = "FDboost")
str(fuelSubset)
```

```
## List of 7
##  $ heatan      : num [1:129] 26.8 27.5 23.8 18.2 17.5 ...
##  $ h2o         : num [1:129] 2.3 3 2 1.85 2.39 ...
##  $ nir.lambda  : num [1:231] 800 803 805 808 810 ...
##  $ NIR         : num [1:129, 1:231] 0.2818 0.2916 -0.0042 -0.034 -0.1804 ...
##  $ uvvis.lambda: num [1:134] 250 256 261 267 273 ...
##  $ UVVIS       : num [1:129, 1:134] 0.145 -1.584 -0.814 -1.311 -1.373 ...
##  $ h2o.fit     : num [1:129] 2.58 3.43 1.83 2.03 3.07 ...
```

```
df = data.frame(fuelSubset[c("heatan", "h2o", "UVVIS", "NIR")])
dim(df)
```

```
## [1] 129 367
```

Functional Data in mlr

How can we use this structure in mlr?

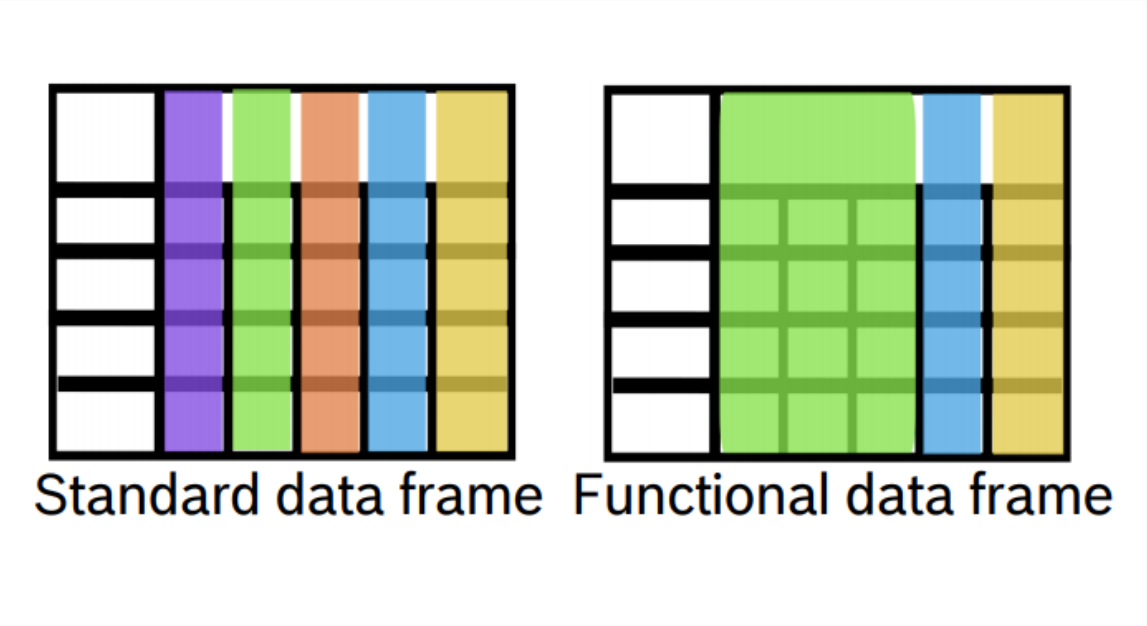


Figure 2: Data structure for functional data

Create a regression task from the dataset

```
fdf = makeFunctionalData(df,fd.features = list("UVVIS" = 3:136,"NIR" = 137:367))
fuelsubset.task = makeRegrTask("fuelSubset", data = fdf, target = "heatan")
fuelsubset.task
```

```
## Supervised task: fuelSubset
## Type: regr
## Target: heatan
## Observations: 129
## Features:
##    numerics     factors     ordered functionals
##           1           0           0           2
## Missings: FALSE
## Has weights: FALSE
## Has blocking: FALSE
## Has coordinates: FALSE
```

Regression & Classification

Create a learner

```
# List available learners
listLearners(obj = fuelsubset.task, properties = "functionals")[,1:3]
```

```
##               class
## 1     regr.FDboost
## 2 regr.featureless
##                                          name  short.name
## 1 Functional linear array regression boosting     FDboost
## 2                      Featureless regression featureless
```

```
# Create the learner
lrn = makeLearner("regr.FDboost")
```

Train on a regression task

```
# Train the learner on a subset of our fuelsubset data
model = train(learner = lrn, task = subsetTask(fuelsubset.task, subset = 1:80))
# Predict on held out data
p = predict(model, subsetTask(fuelsubset.task, subset = 81:129))
# Compute the performance
performance(p, list(rmse, rsq))
```

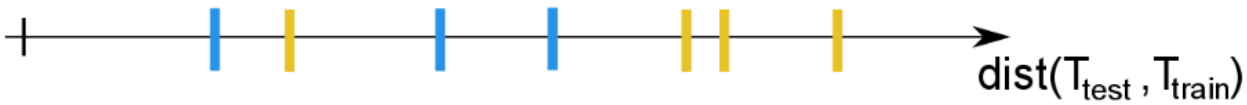
```
##      rmse       rsq
## 2.7218511 0.8042802
```

Example: Functional KNN

- Calculate distance between a pair of functional data
- Distance measures are “functional data specific” such as Lp-Norm

$$\|f(t) - g(t)\|^p = \left(\frac{1}{\int_a^b w(t)dt} \int_a^b |f(t) - g(t)|^p \, w(t)dt \right)^{1/p}$$

- Classification itself via k nearest neighbours:



Classification

Use learners that exploit the functional nature:

```
listLearners(obj = gunpoint.task, properties = "single.functional")[,1:3]
```

```
##                   class                                   name
## 1 classif.fdausc.kernel        Kernel classification on FDA
## 2    classif.fdausc.knn                             fdausc.knn
## 3     classif.fdausc.np Nonparametric classification on FDA
##      short.name
## 1 fdausc.kernel
## 2    fdausc.knn
## 3     fdausc.np
```

```
lrn = makeLearner("classif.fdausc.knn")
resample(lrn, gunpoint.task, cv3)
```

```
## Resample Result
## Task: gp.fdf
## Learner: classif.fdausc.knn
## Aggr perf: mmce.test.mean=0.0501281
## Runtime: 5.01326
```

Classification

Or we can completely disregard the functional nature:

```
listLearners(obj = gunpoint.task)[1:3, 1:2]
```

```
##                 class
## 1        classif.ada
## 2   classif.binomial
## 3 classif.blackboost
##                                     name
## 1                            ada Boosting
## 2                     Binomial Regression
## 3 Gradient Boosting With Regression Trees
```

```
lrn = makeLearner("classif.randomForest")
resample(lrn, gunpoint.task, cv3)
```

```
## Resample Result
## Task: gp.fdf
## Learner: classif.randomForest
## Aggr perf: mmce.test.mean=0.0250264
## Runtime: 0.930977
```

Feature Extraction

Extracting non-functional features

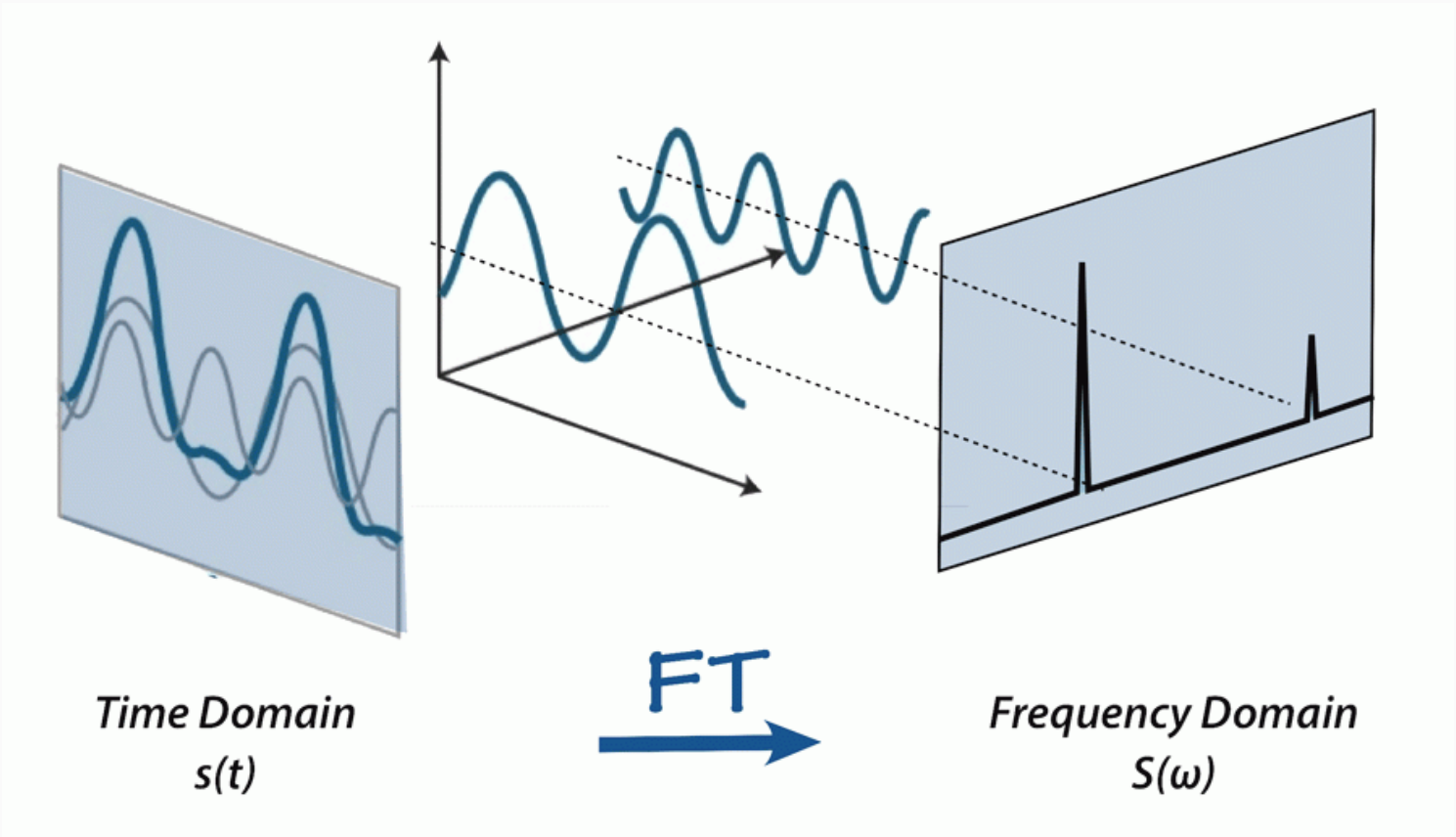


Figure 3: Source: https://aavos.eu/glossary/fourier-transform/

Extracting features

Currently implemented

- Fourier Transformation
- Functional Principal Components
- Wavelets
- Spline Coefficients
- ...

```
# Define what to extract from which feature
feat.methods = list("UVVIS" = extractFDAFourier(), "NIR" = extractFDAFPCA())
extracted.task = extractFDAFeatures(fuelsubset.task, feat.methods = feat.methods)
```

Wrappers

Feature extraction can be used in conjunction with standard ML learners:

```
# Wrap the feature extraction arround a learner
lrn = makeLearner(id = "xgb", cl = "regr.xgboost") %>%
  makeExtractFDAFeatsWrapper(feat.methods = feat.methods)
train(lrn, fuelsubset.task)


## Model for learner.id=xgb.extracted; learner.class=extractFDAFeatsWrapper
## Trained on: task.id = fuelSubset; obs = 129; features = 3
## Hyperparameters: nrounds=1,verbose=0
```

Tuning the wrapper

```
# Define the param space to search over
ps = makeParamSet(
  makeNumericParam("eta", lower = 0.003, upper = 0.25),
  makeNumericParam("alpha", lower = 0.001, upper = 5),
  makeNumericParam("pve", lower = 0.95, upper = 0.99),
  makeDiscreteParam("trafo.coeff", values = c("amplitude", "phase"))
  )
# Define how we tune
ctrl = makeTuneControlRandom(maxit = 5)
# Create the learner and the wrapper(s)
lrn = makeLearner("regr.xgboost") %>%
  makeExtractFDAFeatsWrapper(feat.methods = feat.methods) %>%
  makeTuneWrapper(hout, rmse, ps, ctrl)
res = resample(lrn, fuelsubset.task, hout, rmse)
```

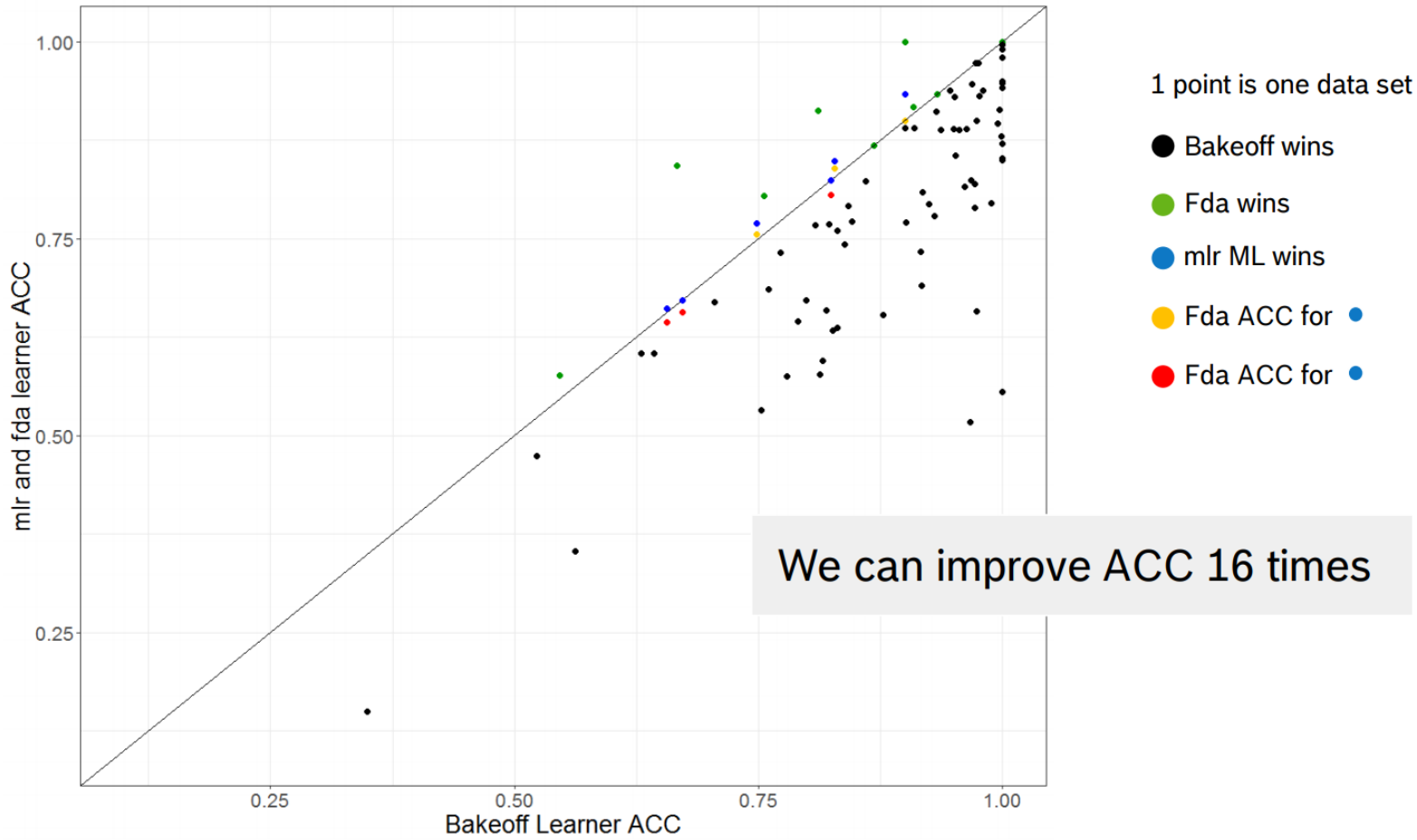
Benchmark

Benchmark Setup

- Compare to UCR Time Series Classification Repository Bakeoff
- 83 classification data sets (univariate time series)
- Fixed train-test splits
- tuning via random search (100iters) and MBO (100iters)
- Algorithms:
  - Functional Data Algorithms (`fda.usc`, `FDboost`, `refund`)
  - Feature Extraction (`wavelets`, `refund`, `mboost`, `FDboost`, `rucrdtw`, `...`)

Benchmark Results

Overview ACC



Outlook

Status of the project

Figure 4: Source: https://twitter.com/thewippod

Future extensions

- Include full FDA preprocessing pipeline (curve registration etc.)
- Clustering & Anomaly Detection
- More learners & feature extractors

Contributions and suggestions are welcome!

Questions?

Packages & Bibliography i

Manuel Febrero-Bande and Manuel Oviedo de la Fuente

Statistical Computing in Functional Data Analysis: The R Package fda.usc.

R package

Brockhaus, S. and Ruegamer, D.

FDboost: Boosting Functional Regression Models,.

R package

Bischl B, Lang M, Kotthoff L, Schiffner J, Richter J, Studerus E, Casalicchio G and Jones Z (2016)

mlr: Machine Learning in R.

Journal of Machine Learning Research, 17(170):1–5, 2016

Packages & Bibliography ii

Jeff Goldsmith and Fabian Scheipl and Lei Huang and Julia Wrobel and Jonathan Gellar and Jaroslaw Harezlak and Mathew W. McLean and Bruce Swihart and Luo Xiao and Ciprian Crainiceanu and Philip T. Reiss

refund: Regression with Functional Data

R package

Eric Aldrich

wavelets: A package of functions for computing wavelet filters, wavelet transforms and multiresolution analyses.

R package

And many, many more such as `classiFunc`, `rucrdtw`, `tsfeatures`, ...

FuelSubset Data

Spectral Data Of Fossil Fuels

For 129 laboratory samples of fossil fuels the heat value and the humidity were determined together with two spectra. One spectrum is ultraviolet-visible (UV-VIS), measured at 1335 wavelengths in the range of 250.4 to 878.4 nanometer (nm), the other a near infrared spectrum (NIR) measured at 2307 wavelengths in the range of 800.4 to 2779.0 nm. fuelSubset is a subset of the original dataset containing only 10% of the original measures of the spectra, resulting in 231 measures of the NIR spectrum and 134 measures of the UVVIS spectrum.

Gunpoint Data

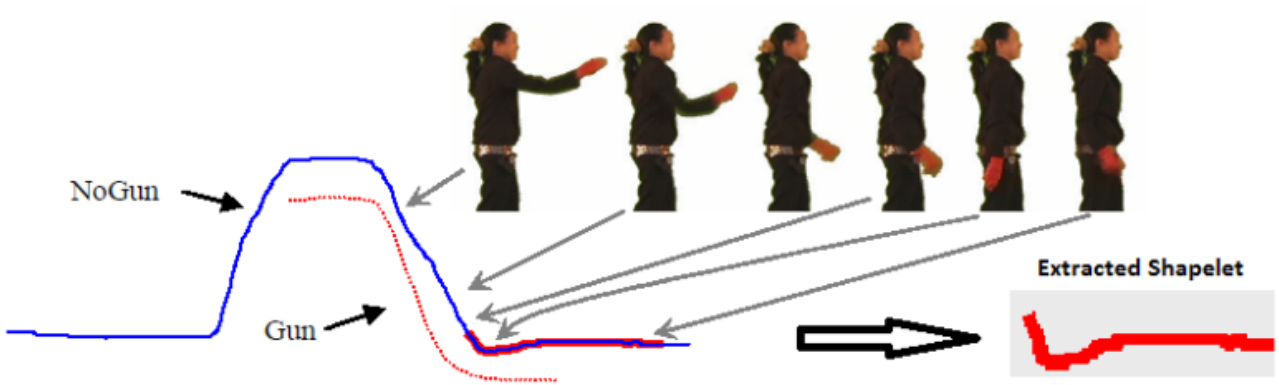
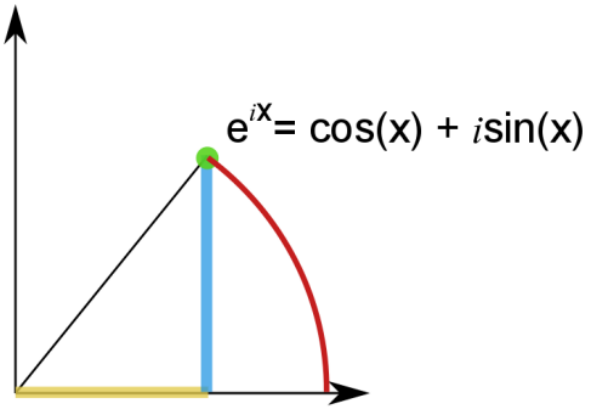


Fig. 2: Shapelets from gunpoint data set (Ye and Keogh, 2009)

Fourier Transform II

Key idea: each signal (over time) can be filtered into combinations of circular paths, i.e.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i 2\pi k n / N}$$



Signal over time → Signal over frequencies

functional → non-functional

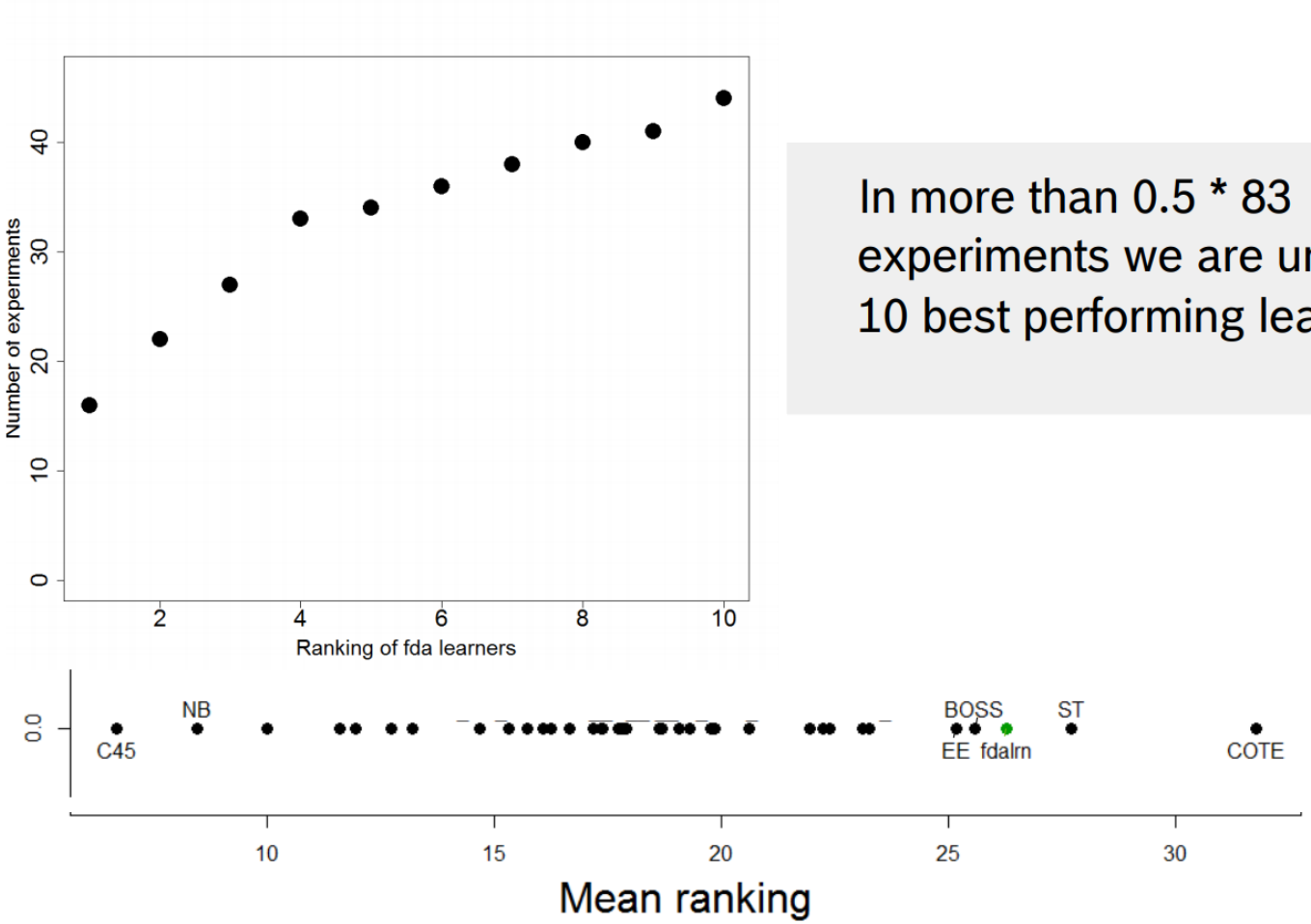
Result of tuning xgboost and feature extraction

```
res


## Resample Result
## Task: fuelSubset
## Learner: regr.xgboost.extracted.tuned
## Aggr perf: rmse.test.rmse=23.0243984
## Runtime: 39.9007
```

Benchmark II

Ranking



In more than 0.5 * 83 experiments we are under the 10 best performing learners

Benchmark III

Individual datasets