Time-series Machine Learning

Student workshop at the 6^{th} Symposium of the International Engineering Science Symposium, Auckland, 08.09.2025

Assoc Prof Andreas W. Kempa-Liehr

Department of Engineering Science and Biomedical Engineering

UNIVERSITY OF AUCKLAND
Waipapa Taumata Rau
NEW ZEALAND

ENGINEERING

08.09.2025 – StW IESC

Outline

A personal timeline of time-series analytics

1998–1999 Masters thesis (Dipl.-Phys.)

1999–2004 PhD thesis (Dr. rer. nat.) about the dynamics of self-organisation phenomena (Institute of Applied Physics, University of Münster, Germany)

2004–2009 Head of Service Group *Scientific Information Processing* at Freiburg Materials Research Center of the University of Freiburg, Germany

2009–2014 Professional Analyst at EnBW Baden-Württemberg AG (Karlsruhe, Germany)

2014–2016 Senior Data Scientist at Blue Yonder GmbH (Karlsruhe)

2016–2021 Senior Lecturer at Department of Engineering Science, UoA

since 2022 Associate Professor at Department of Engineering Science and Biomedical Engineering, UoA

Engineering Applications of Artificial Intelligence (AI)

Decision Automation

Data Science Which information do we need?

Operations Research What should we do?

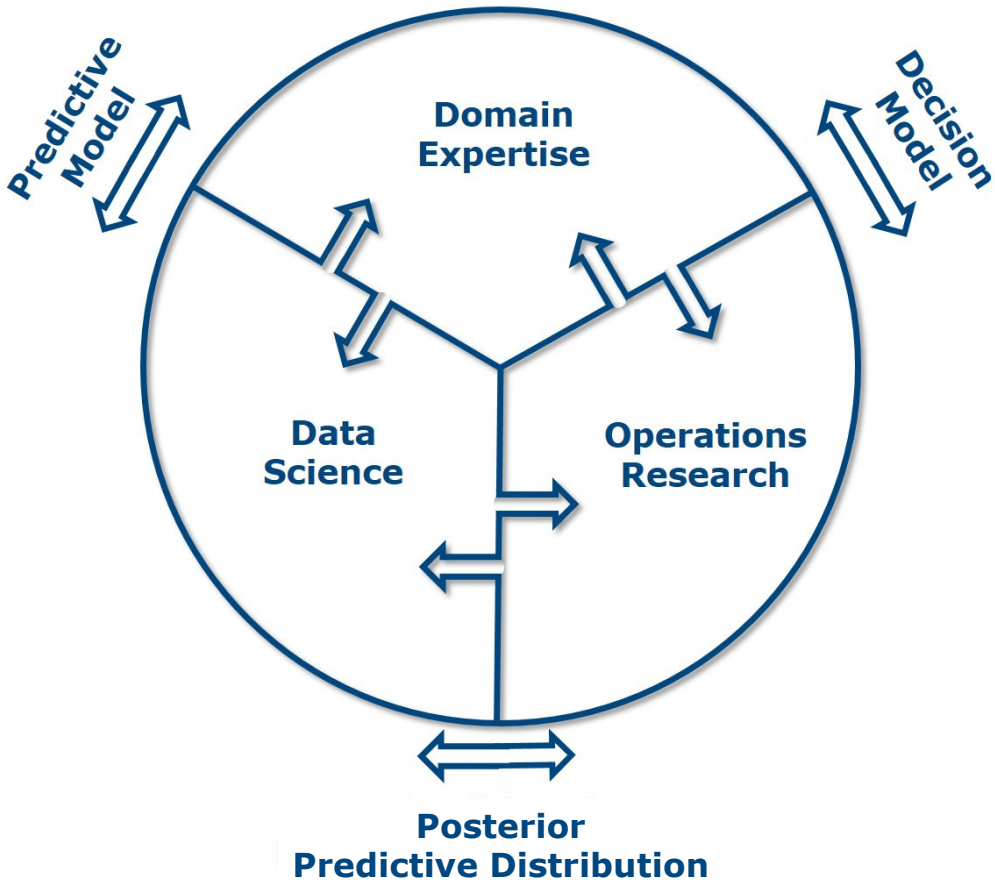
Deployment of New Sensor System

Data Science

- How can we identify defective sensors, when we have not been able to catalog all error types?

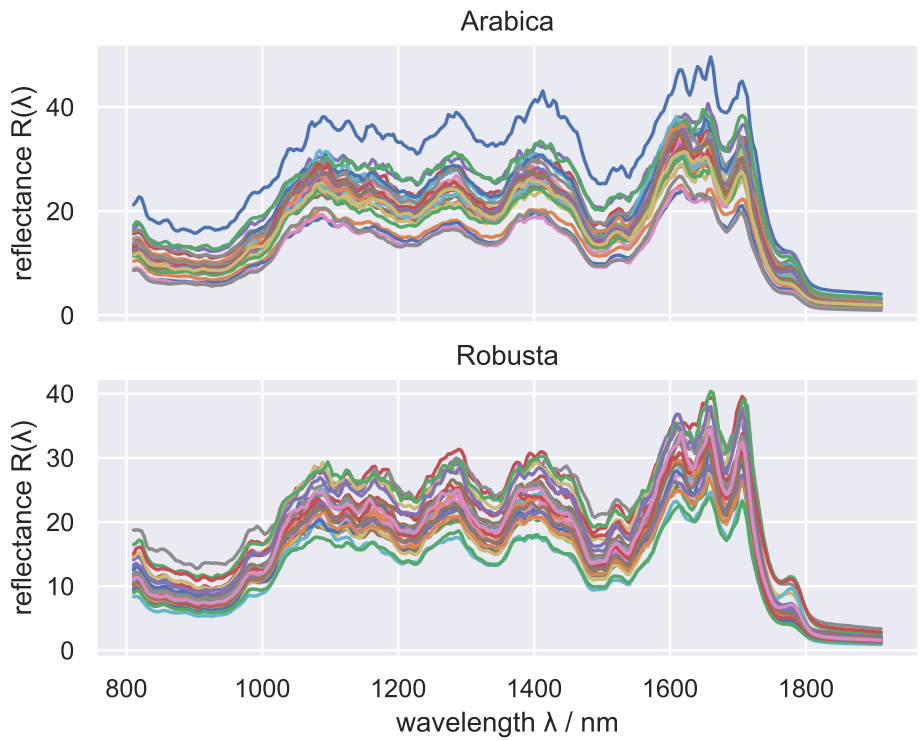
Operations Research

- What are the costs for missing that a sensor has become defective?



Welcome to CoffeeAI (a students' assignment)

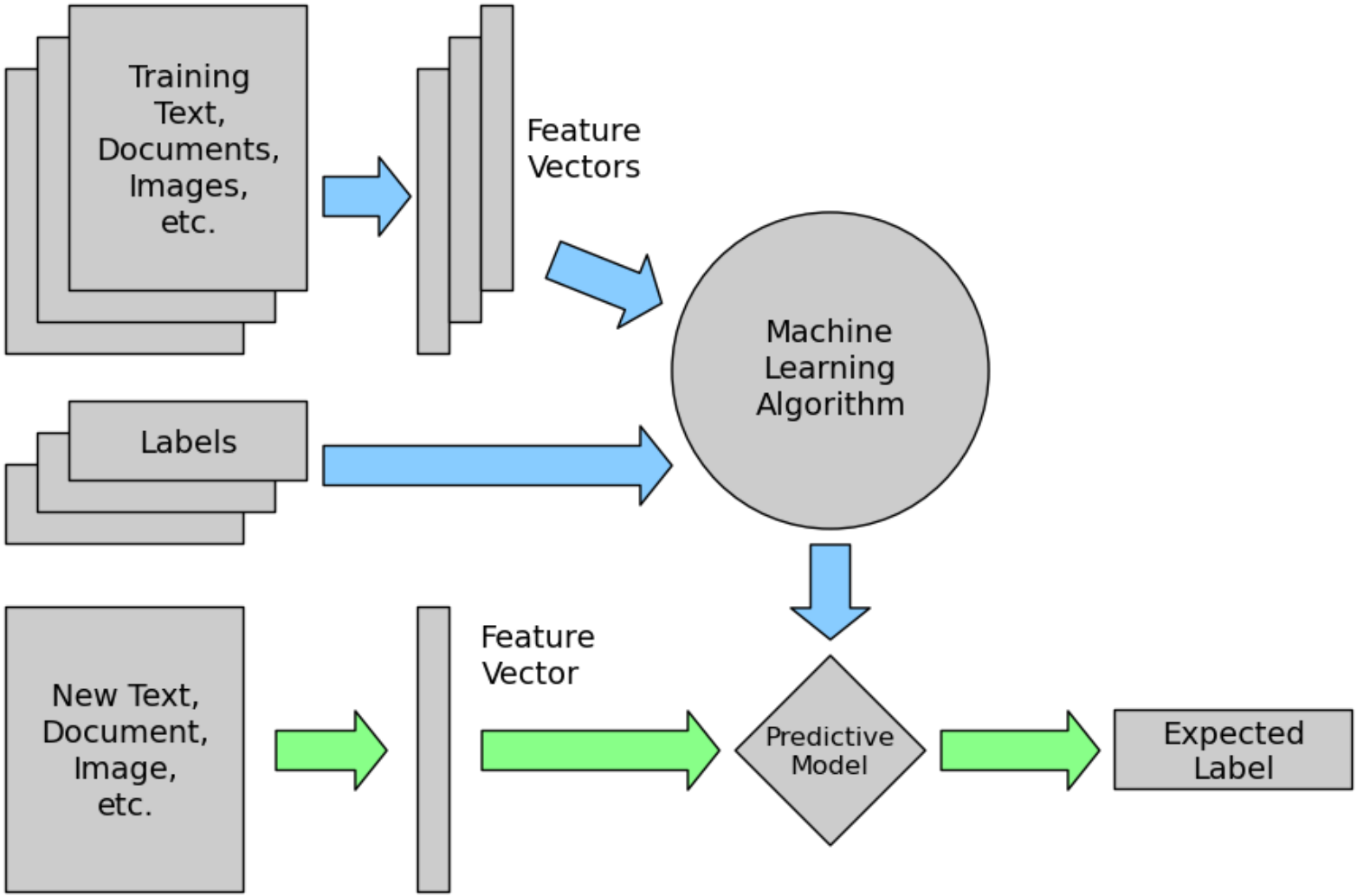
You have decided to join the Auckland based technology startup CoffeeAI, which develops an *intelligent* coffee machine named *eBarista*. This coffee machine is supposed to automatically tune its grinding and brewing process according to the coffee beans provided.
The heart of each *eBarista* is a Fourier-Transform Infrared Spectrometer, which measures the infrared spectrum of absorption of the coffee beans.
Your task is to use computational signal processing to generates features, which can be used for discriminating Arabica and Robusta beans.



Data source: Romain Briandet, E. Katherine Kemsley, and Reginald H. Wilson. "Discrimination of Arabica and Robusta in Instant Coffee by Fourier Transform Infrared Spectroscopy and Chemometrics". In: *Journal of Agriculture and Food Chemistry* 44.1 (1996), pp. 170–174. DOI: 10.1021/jf950305a

In order to evaluate your features, you're going to fit a logistic regression classifier and perform an in-sample analysis.

Flowchart of supervised learning



Jakob VanderPlas. *Astronomy with scikit-learn*. Release Scipy2012. Seattle: University of Washington, 2012. URL: http://astroML.github.com/sklearn_tutorial

Naive feature engineering for CSP

Naive Time-Series Feature Matrix

The feature matrix X, which results from naive feature engineering in computational signal processing, is characterized as follows:

- Rows of feature matrix X represent specific time-series $\vec{x}_i = (x_i(t_1), \ldots, x_i(t_j), \ldots, x_i(t_T))$. In machine learning, these rows are also called feature vectors.
- Columns of feature matrix X represent measurements at a specific point in time t_j. In machine learning, these columns are called feature.

$$X = \begin{pmatrix} z_1(t_1) & \ldots & z_1(t_j) & \ldots & z_1(t_T) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ z_i(t_1) & \ldots & z_i(t_j) & \ldots & z_i(t_T) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ z_N(t_i) & \ldots & z_N(t_j) & \ldots & z_N(t_T) \end{pmatrix} = \begin{pmatrix} z_{1,1} & \ldots & z_{1,j} & \ldots & z_{1,T} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ z_{i,1} & \ldots & z_{i,j} & \ldots & z_{i,T} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ z_{N,1} & \ldots & z_{N,j} & \ldots & z_{N,T} \end{pmatrix} \in \mathbb{R}$$

Classical approaches

Functional Data Analysis

- Find continuous representation of dataset
- Solve Functional Classification Problem

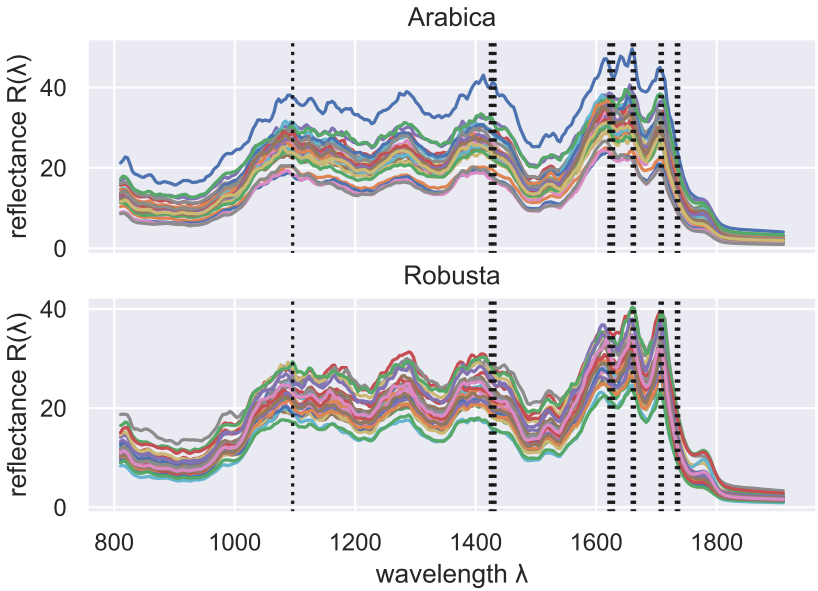
Machine Learning

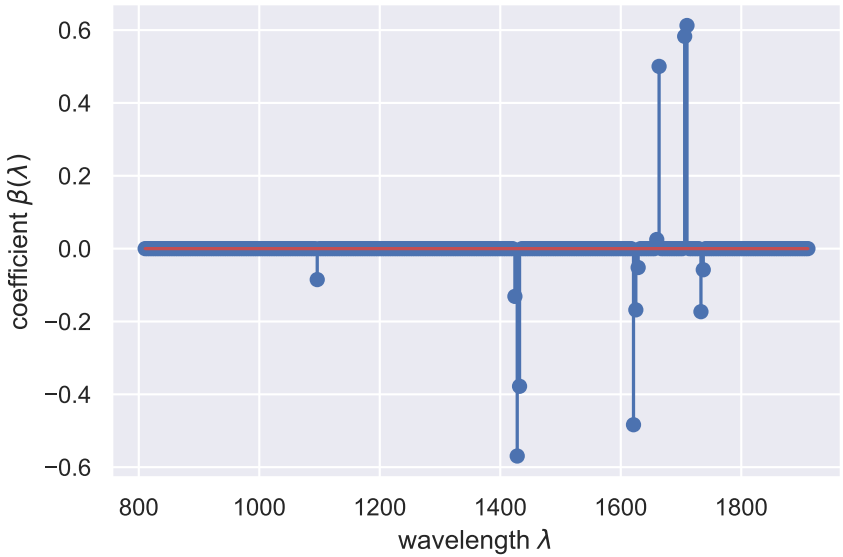
- Trivial feature engineering (use raw data as features)
- Project onto Principal Components
- Fit classifier
- Cross-validate machine learning pipeline

Signal processing

- Karhunen-Loève transformation (PCA)
- Design discriminating signal processor

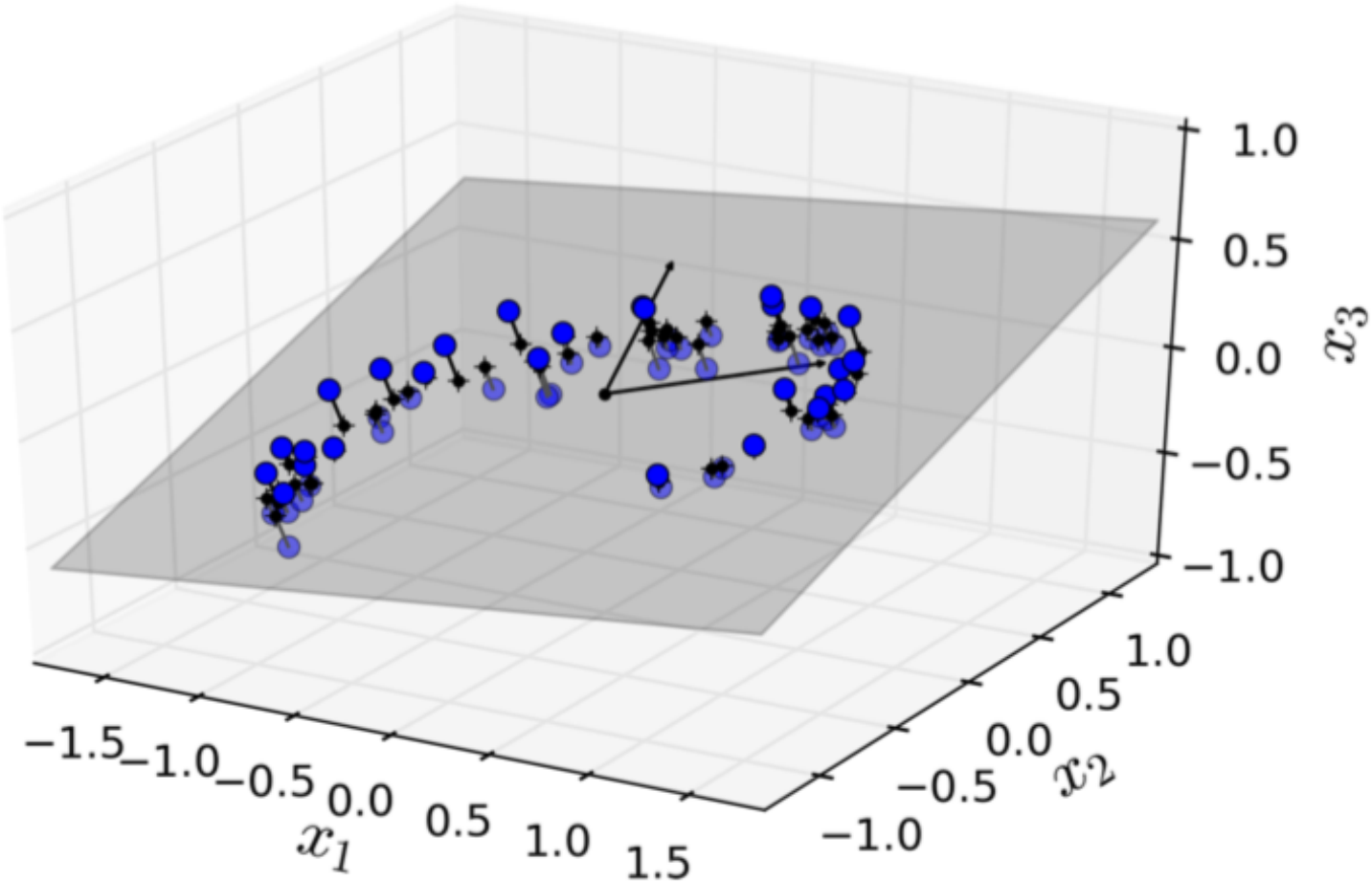
Simple Logistic Regression with L1 norm





A 3D dataset lying close to a 2D subspace

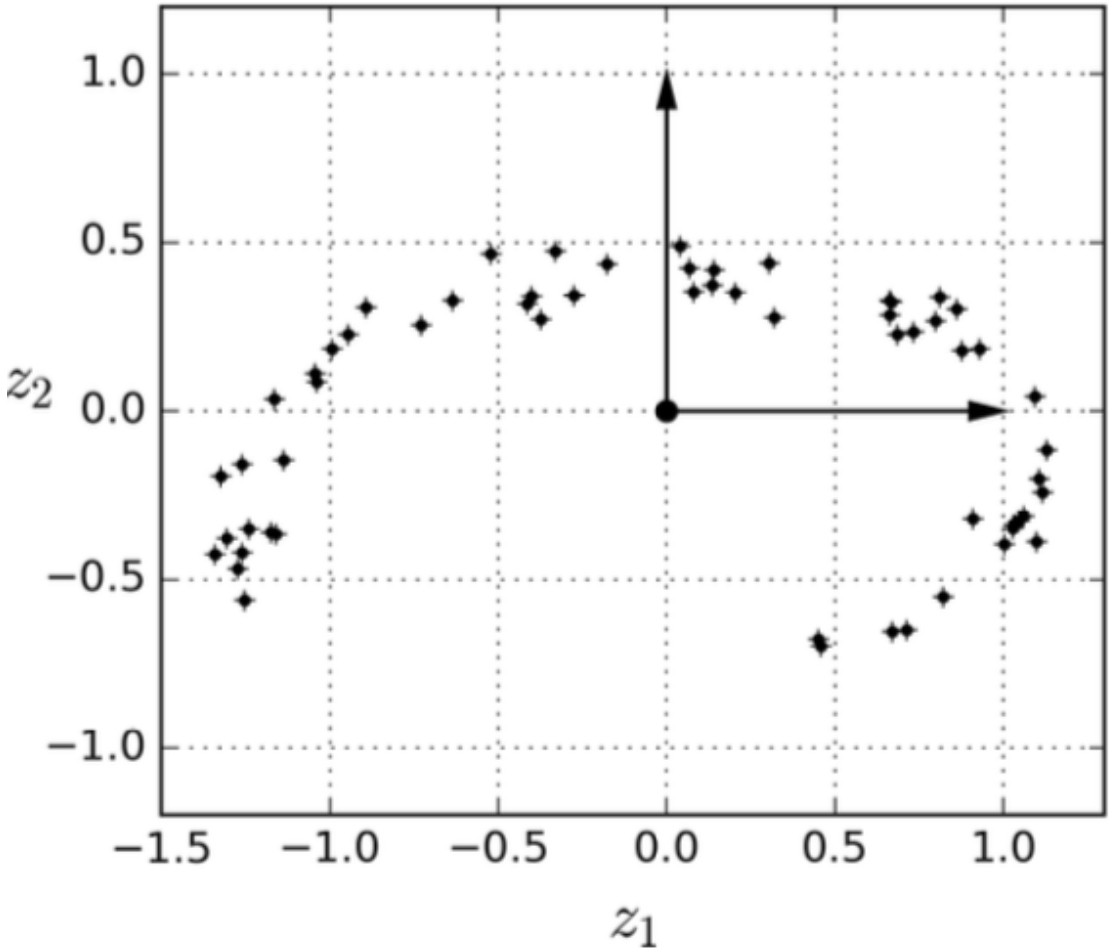
All features x_1, x_2, x_3 lie close to a plane: this is a lower-dimensional (2D) subspace of the high-dimensional (3D) space.



Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. 4th release. Sebastopol, CA, United States: O'Reilly Media, 2017, Fig. 8.2

Projection

If we project every (x_1, x_2, x_3) tupple perpendicularly onto this subspace (as represented by the short lines connecting the instances to the plane), we get the new 2D dataset.



Principal Component Analysis

Principal Component Analysis (PCA) is by far the most popular dimensionality reduction algorithm. For naive feature engineering (use raw signal values as features of machine learning), PCA is equivalent to the Karhunen-Loève Transformation.

- PCA identifies the hyperplane that lies closest to the data, and then
- it projects the data onto it.
- The hyperplane is given by the eigenvectors of the covariance matrix.
- The corresponding eigenvalues are proportional to the explained variance.

Assumptions

PCA assumes that each column of the feature matrix has zero mean.

- Subtract empirical mean of each column from the respective column.

Scaling

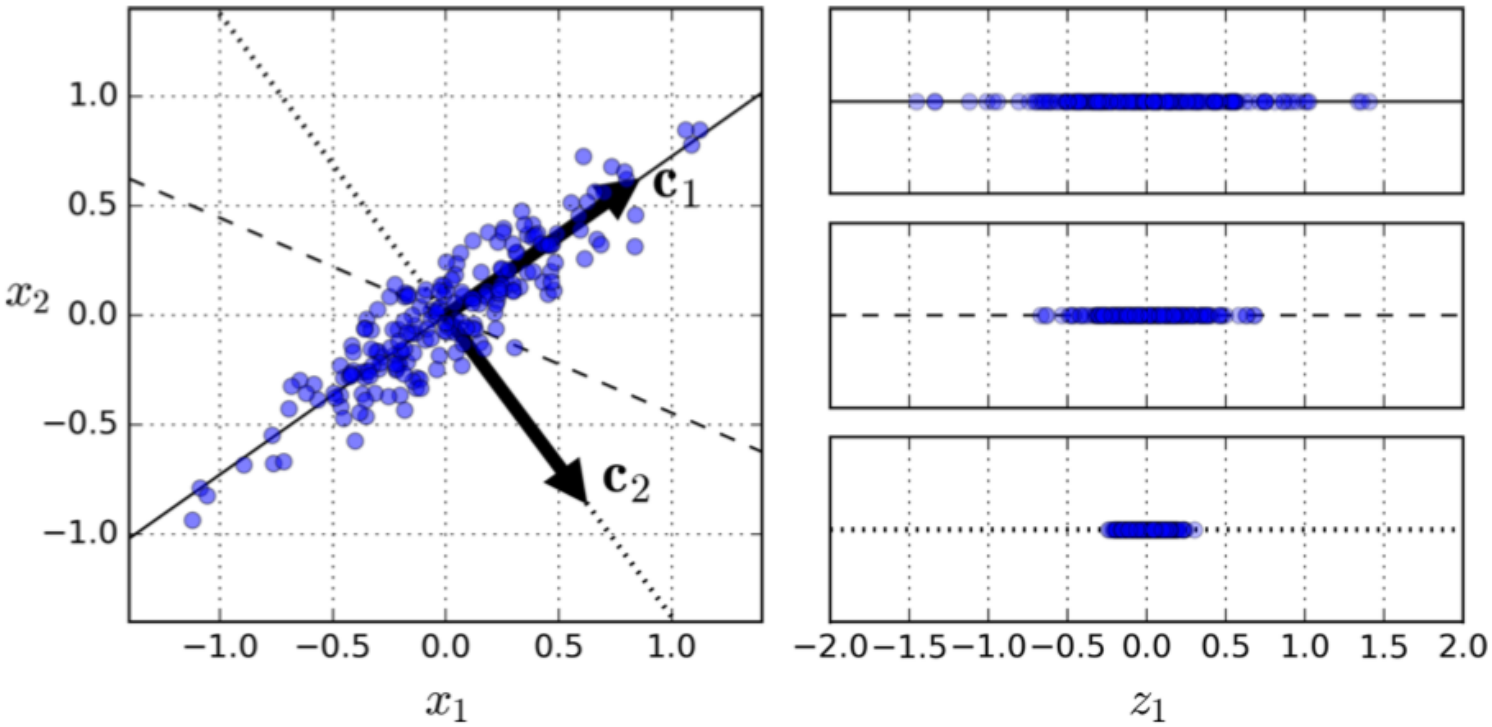
Machine Learning Rule-Of-Thumb

Variables measured on different scales or on a common scale with widely differing ranges are often standardized.

- After subtracting the empirical mean, divide each column by its empirical standard deviation.

Selecting the subspace

- It seems reasonable to select the axis that preserves the maximum amount of variance, as it will most likely lose less information than the other projections.
- Another way to justify this choice is that it is the axis that minimizes the mean squared distance between the original dataset and its projection onto that axis. This is the rather simple idea behind PCA.



Naive feature engineering for CSP

Naive Time-Series Feature Matrix

The feature matrix X, which results from naive feature engineering in computational signal processing, is characterized as follows:

- Rows of feature matrix X represent specific time-series $\vec{x}_i = (x_i(t_1), \ldots, x_i(t_j), \ldots, x_i(t_T))$. In machine learning, these rows are also called feature vectors.
- Columns of feature matrix X represent measurements at a specific point in time t_j. In machine learning, these columns are called feature.

$$X = \begin{pmatrix} z_1(t_1) & \ldots & z_1(t_j) & \ldots & z_1(t_T) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ z_i(t_1) & \ldots & z_i(t_j) & \ldots & z_i(t_T) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ z_N(t_i) & \ldots & z_N(t_j) & \ldots & z_N(t_T) \end{pmatrix} = \begin{pmatrix} z_{1,1} & \ldots & z_{1,j} & \ldots & z_{1,T} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ z_{i,1} & \ldots & z_{i,j} & \ldots & z_{i,T} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ z_{N,1} & \ldots & z_{N,j} & \ldots & z_{N,T} \end{pmatrix} \in \mathbb{R}$$

Uncorrelated variables/features/columns $z[j]$

- PCA assumes that the columns of the feature matrix are centered, such that every column has a mean of zero.

Definition (Centered feature matrix X_c)

The centered feature matrix X_c of signals $x_1[\cdot], \ldots, x_N[\cdot]$ is computed by subtracting all columns $j = 1, \ldots, T$ by the respective column mean $\bar{z}[j]$:

$$X_c = \begin{pmatrix} z_1[1] - \bar{z}[1] & \ldots & z_1[j] - \bar{z}[j] & \ldots & z_1[T] - \bar{z}[T] \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ z_i[1] - \bar{z}[1] & \ldots & z_i[j] - \bar{z}[j] & \ldots & z_i[T] - \bar{z}[T] \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ z_N[1] - \bar{z}[1] & \ldots & z_N[j] - \bar{z}[j] & \ldots & z_N[T] - \bar{z}[T] \end{pmatrix}$$

Covariance

- Empirical variance

$$\sigma_j^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(z_i[j]-\bar{z}[j]\right)^2$$

$$= \frac{1}{N-1}\sum_{i=1}^{N}\left(z_i[j]-\bar{z}[j]\right)\left(z_i[j]-\bar{z}[j]\right)$$

- Empirical covariance

$$\sigma_{j,k}^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(z_i[j]-\bar{z}[j]\right)\left(z_i[k]-\bar{z}[k]\right)$$

- Empirical covariance matrix

$$K_{zz} = \frac{1}{N-1}X_c^{\mathrm{T}}X_c$$

The six steps of PCA

- Centering the data.
- Normalizing the data, if units of columns (and/or the range of the column values) are different.
- Computing the cross-correlation matrix.
- Computing the eigenpairs of the cross-correlation matrix.
- Computing the cumulative sum of the eigenvalues and identify the number of principal components (eigenvectors of the cross-correlation matrix), which should be used for representing the data.
- Projecting the data onto these principal components.

Problem for more general applications

- Time series are similar, but in reality length of time series might be different for each sample
- There might be more than one time series per observation
- Often samples are further characterized by univariate features
- We do not have continuously sampled time series, but event sequences (which might even be incomplete)

Approach

1. Project into well-defined feature space
2. Use univariate hypothesis testing for identifying relevant features
3. Convert event sequences to time series

Pattern recognition

Systematic Feature Engineering for Time-Series and Other Types of Data With Internal Ordering

- Engineering Applications of Artificial Intelligence using
  - sensor signals (time-series, spectra) or
  - event sequences (processes)

Time Series FeatuRe Extraction on basis of Scalable Hypothesis Tests

`https://github.com/blue-yonder/tsfresh`

ACML Workshop on Learning on Big Data WLBD:1–17, 2016 Hamilton (NZ), 16th November 2016

Distributed and parallel time series feature extraction for industrial big data applications

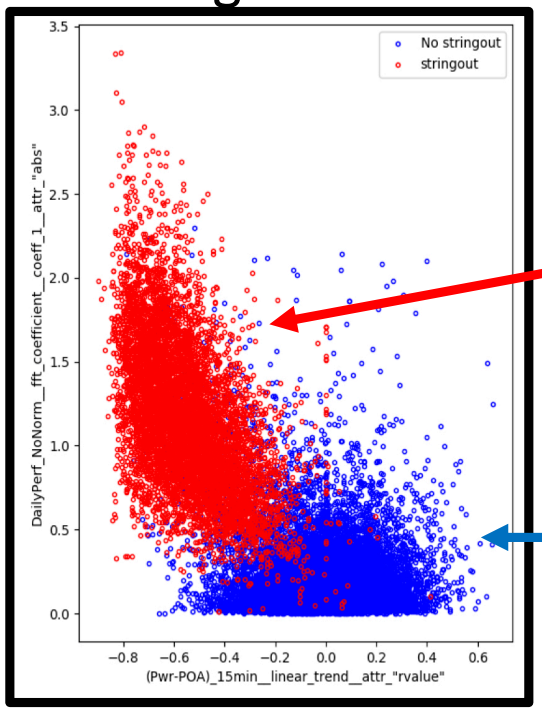
Maximilian Christ MAXIMILIAN.CHRIST@BLUE-YONDER.COM
Blue Yonder GmbH, Karlsruhe, Germany

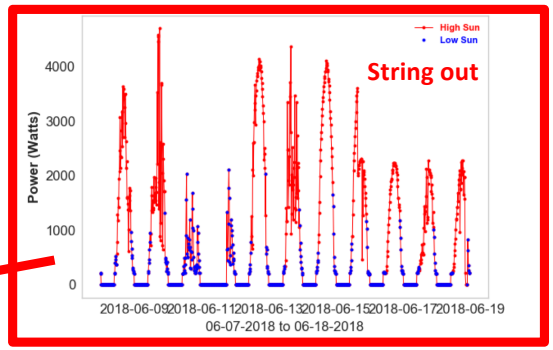
Andreas W. Kempa-Liehr KEMPA-LIEHR@FMF.UNI-FREIBURG.DE
Freiburg Materials Research Center, University of Freiburg, Freiburg, Germany
Blue Yonder GmbH, Karlsruhe, Germany

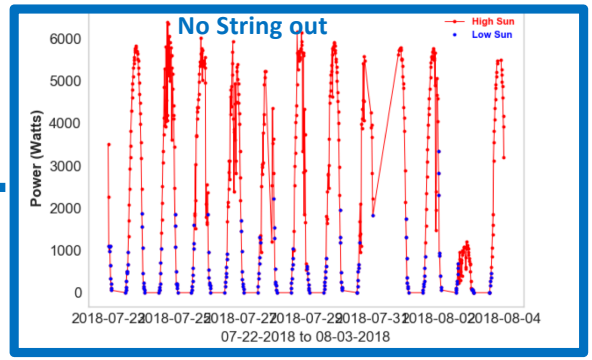
Michael Feindt MICHAEL.FEINDT@BLUE-YONDER.COM
on leave of absence from Karlsruhe Institute of Technology
Blue Yonder GmbH, Karlsruhe, Germany

tsfresh application at TESLA

TSFresh for time series classification to diagnose electrical issues in Residential PV systems

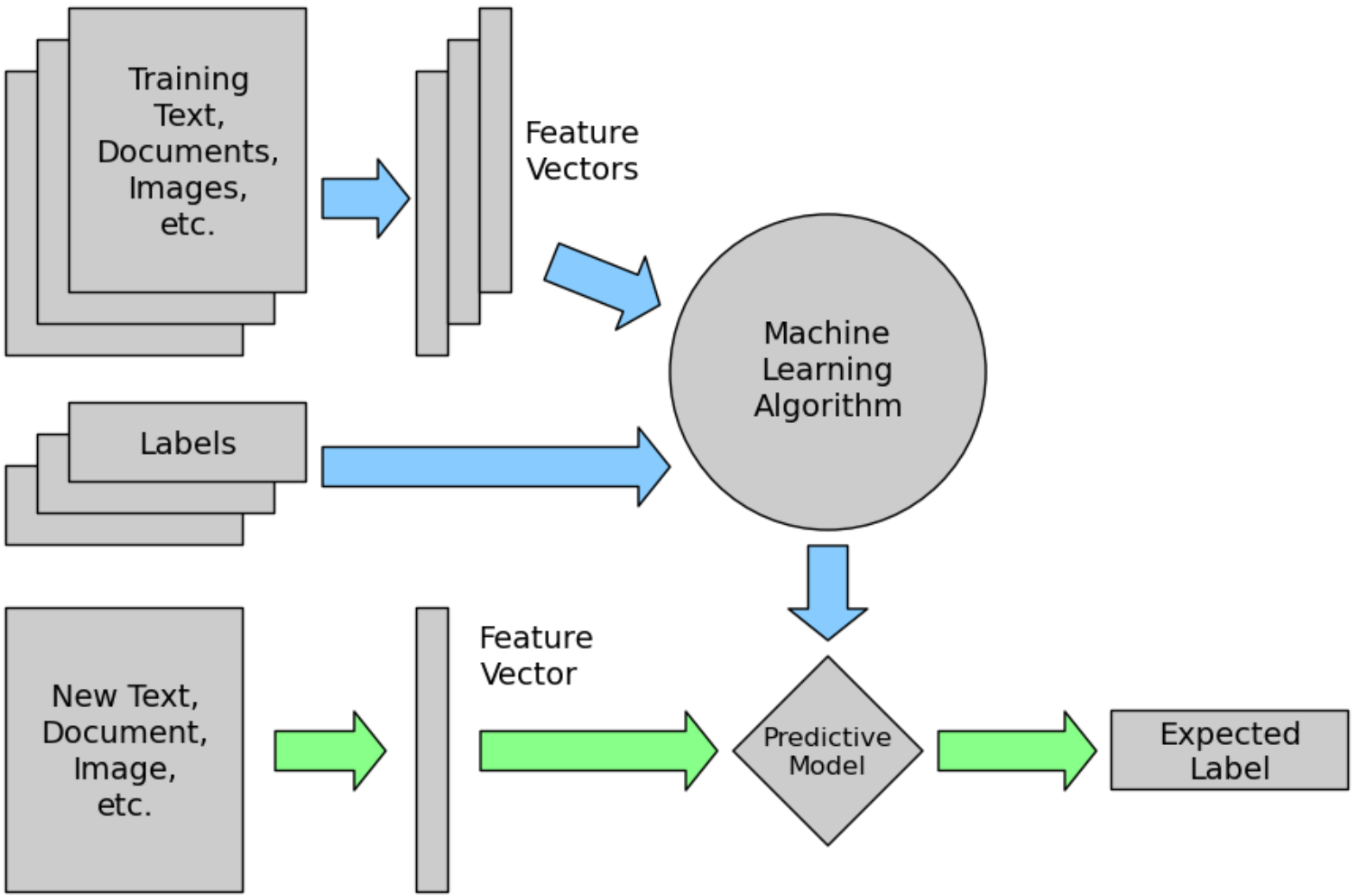






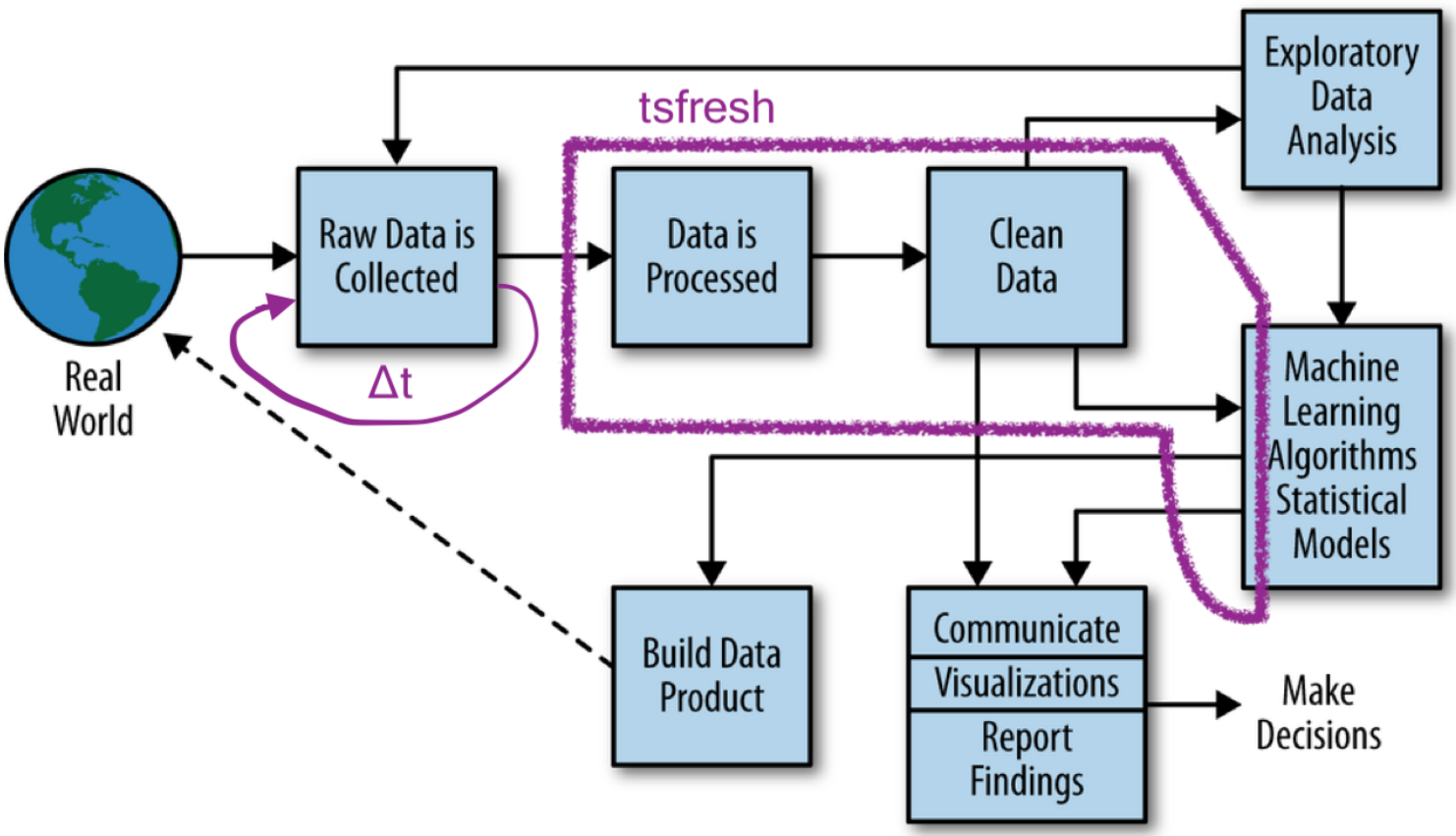
Credit:
E. Victor Garcia, Ph.D.
Data Scientist @Tesla,
eugenio.v.garcia@gmail.com

Flowchart of supervised learning



VanderPlas, *Astronomy with scikit-learn*

Data Science Process for Time-series



Adapted from Cathy O'Neil and Rachel Schutt. *Doing Data Science. Straight Talk from the Frontline.* Sebastopol, CA, United States: O'Reilly Media, 2013, Fig. 2.2

https://github.com/blue-yonder/tsfresh

Maximilian Christ et al. "Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package)". In: *Neurocomputing* 307 (2018), pp. 72–77. DOI: 10.1016/j.neucom.2018.03.067

Naive feature engineering

Naive Time-Series Feature Matrix

The feature matrix X, which results from naive feature engineering in computational signal processing, is characterized as follows:

- Rows of feature matrix X represent specific time-series $\vec{x}_i = (x_i(t_1), \ldots, x_i(t_j), \ldots, x_i(t_T))$. In machine learning, these rows are also called feature vectors.
- Columns of feature matrix X represent measurements at a specific point in time t_j. In machine learning, these columns are called feature.

$$X = \begin{pmatrix} x_1(t_1) & \ldots & x_1(t_j) & \ldots & x_1(t_T) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_i(t_1) & \ldots & x_i(t_j) & \ldots & x_i(t_T) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_N(t_i) & \ldots & x_N(t_j) & \ldots & x_N(t_T) \end{pmatrix} = \begin{pmatrix} x_1[1] & \ldots & x_1[j] & \ldots & x_1[T] \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_i[1] & \ldots & x_i[j] & \ldots & x_i[T] \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_N[1] & \ldots & x_N[j] & \ldots & x_N[T] \end{pmatrix} \in \mathbb{R}$$

Feature matrix X_ϕ

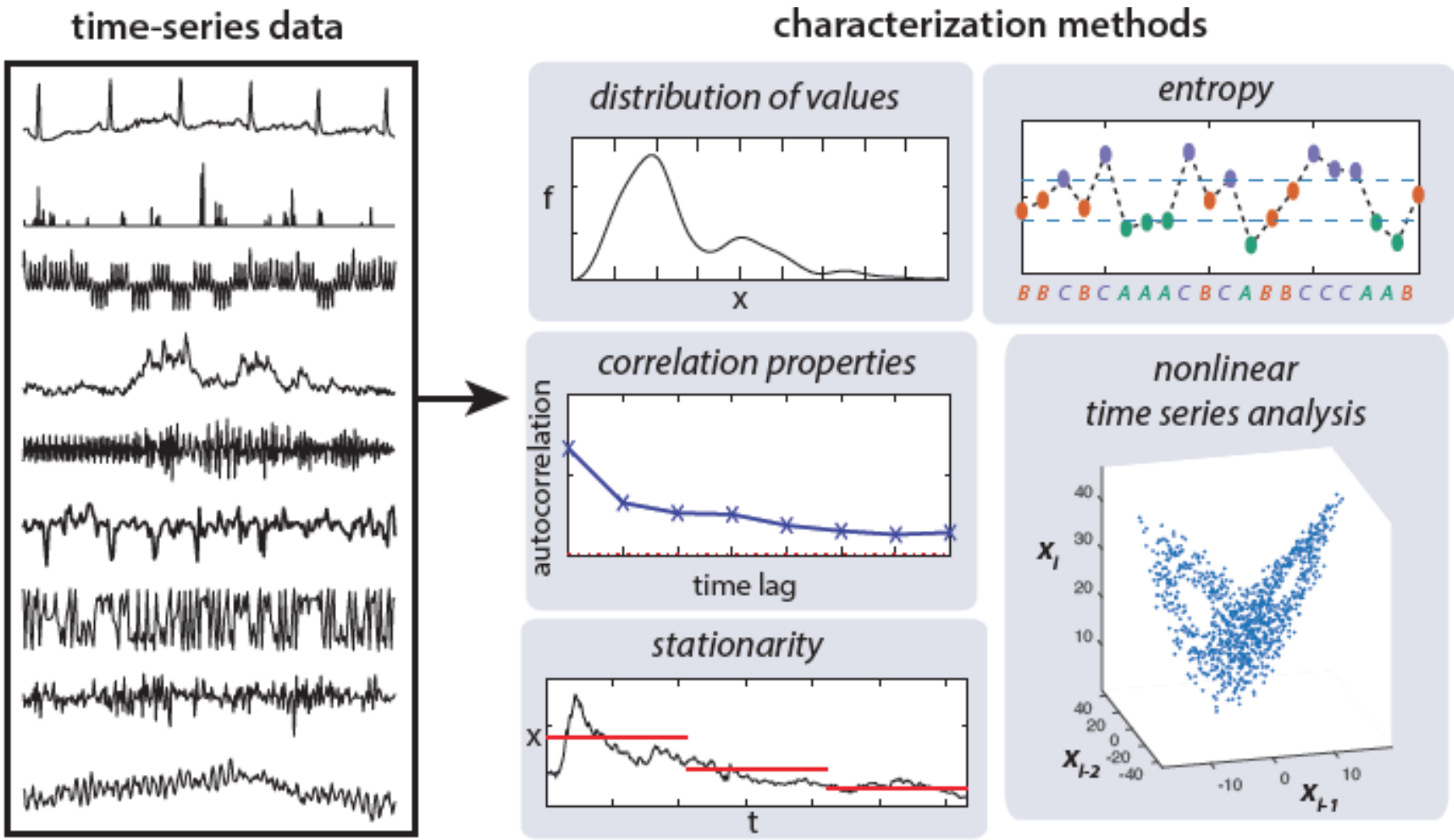
Engineered Time-Series Feature Matrix

The feature matrix X_ϕ, which results from systematic feature engineering in computational signal processing, is characterized as follows:

- Rows of feature matrix X_ϕ represent feature vectors from at least one time-series $\vec{x}_i = (x_i(t_1), \ldots, x_i(t_j), \ldots, x_i(t_T))$.
- Columns of feature matrix X_ϕ quantify a specific characteristic of a time-series, which are computed by functions $\phi_1(\vec{x}), \ldots, \phi_k(\vec{x}), \ldots, \phi_K(\vec{x})$. These columns are called feature.

$$X_\phi = \begin{pmatrix} \phi_1(\vec{x}_1) & \ldots & \phi_k(\vec{x}_1) & \ldots & \phi_K(\vec{x}_1) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \phi_1(\vec{x}_i) & \ldots & \phi_k(\vec{x}_i) & \ldots & \phi_K(\vec{x}_i) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \phi_1(\vec{x}_N) & \ldots & \phi_k(\vec{x}_N) & \ldots & \phi_K(\vec{x}_N) \end{pmatrix} = \begin{pmatrix} \hat{x}_{1,1} & \ldots & \hat{x}_{1,k} & \ldots & \hat{x}_{1,K} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \hat{x}_{i,1} & \ldots & \hat{x}_{i,k} & \ldots & \hat{x}_{i,K} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \hat{x}_{N,1} & \ldots & \hat{x}_{N,k} & \ldots & \hat{x}_{N,K} \end{pmatrix} \in \mathbb{R}$$

Time-series features $\phi(\vec{x})$



Ben D. Fulcher. "Feature-based time-series analysis". In: *Feature engineering for machine learning and data analytics*. Ed. by Guozhu Dong and Huan Liu. Boca Raton, FL: Taylor & Francis, 2018, pp. 87–116

Systematic Time-series Feature Engineering

Given a specific classification or regression task:

- Collect labelled time series and relevant univariate variables
- Extract statisticall significant time series features, which have been found to be useful in some other context (FRESH algorithm)
- Develop machine learning model and identify relevant features
  - Undersample imbalanced problems
- If concept drift can be neglected, restrict feature extraction to relevant features
- Deploy model and monitor performance

Example: Human Activity Recognition (HAR)

Definition (Human Activity Recognition)

HAR is an active research area within the field of *ubiquitous sensing*, which has applications in

- medicine (monitoring exercise routines),
- military (decision making in combat and training scenarios),
- sport (monitoring the potential for injuries and enhance athletes performance).

Challenges

- selection of the attributes to be measured,
- design of feature extraction and inference methods,
- flexibility to support new users without the need of re-training the system,

IMeasureU BlueThunder sensor

accelerometer range $\pm$ 16g

accelerometer resolution 16bit

gyroscope range $\pm$ 2000°/s

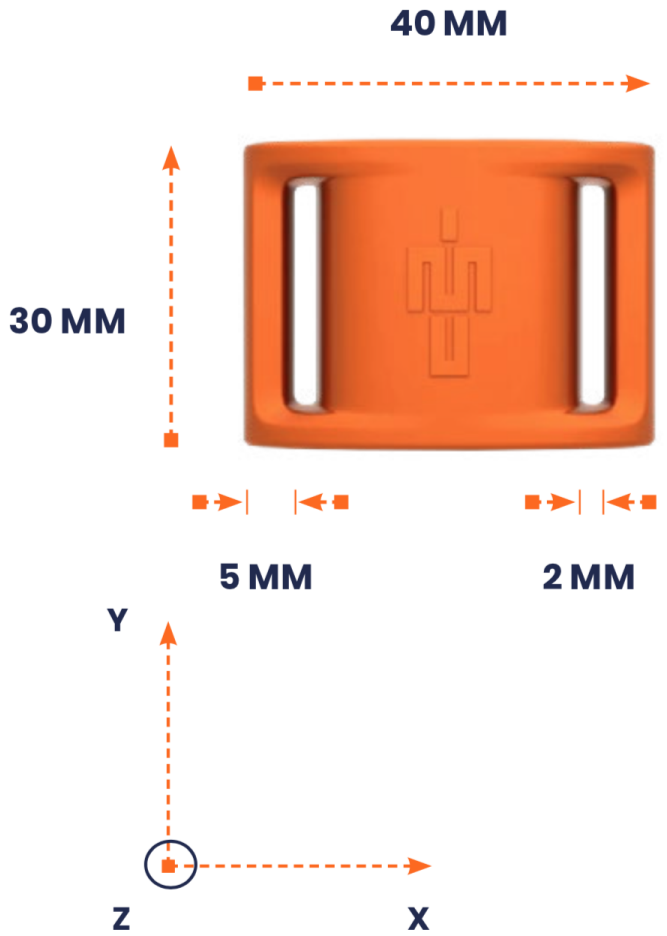
gyroscope resolution 16bit

compass range $\pm$ 1200 µT

compass resolution 13bit

data logging 500Hz

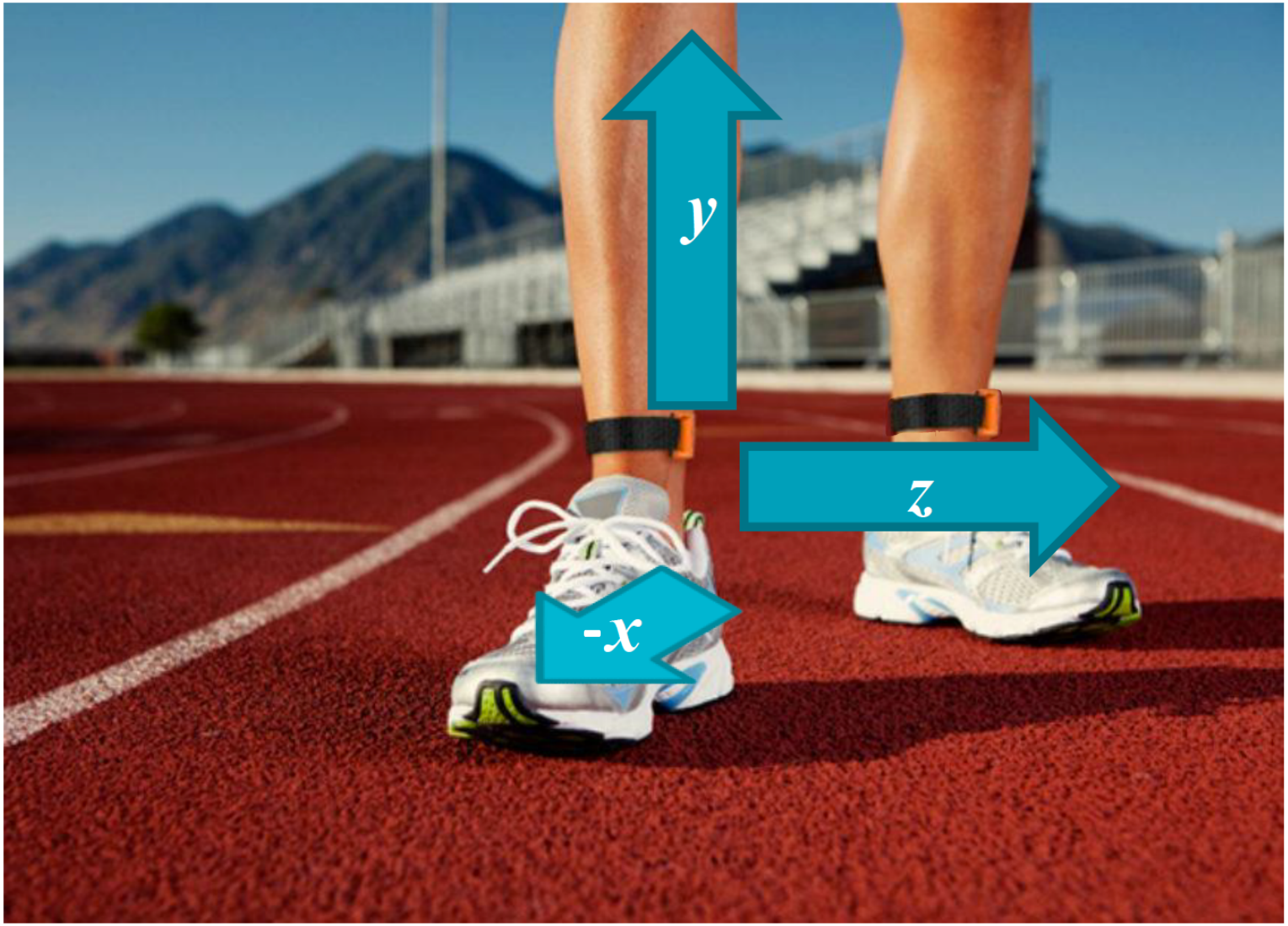
weight 12g



Andrew Wong and Rakesh Vallabh. *IMeasureU BlueThunder sensor*. Sensor Specification 1.5. Auckland: Vicon IMeasureU Limited, 2018. URL: https://imeasureu.com/wp-content/uploads/2018/05/Sensor_Specification_v1.5.pdf

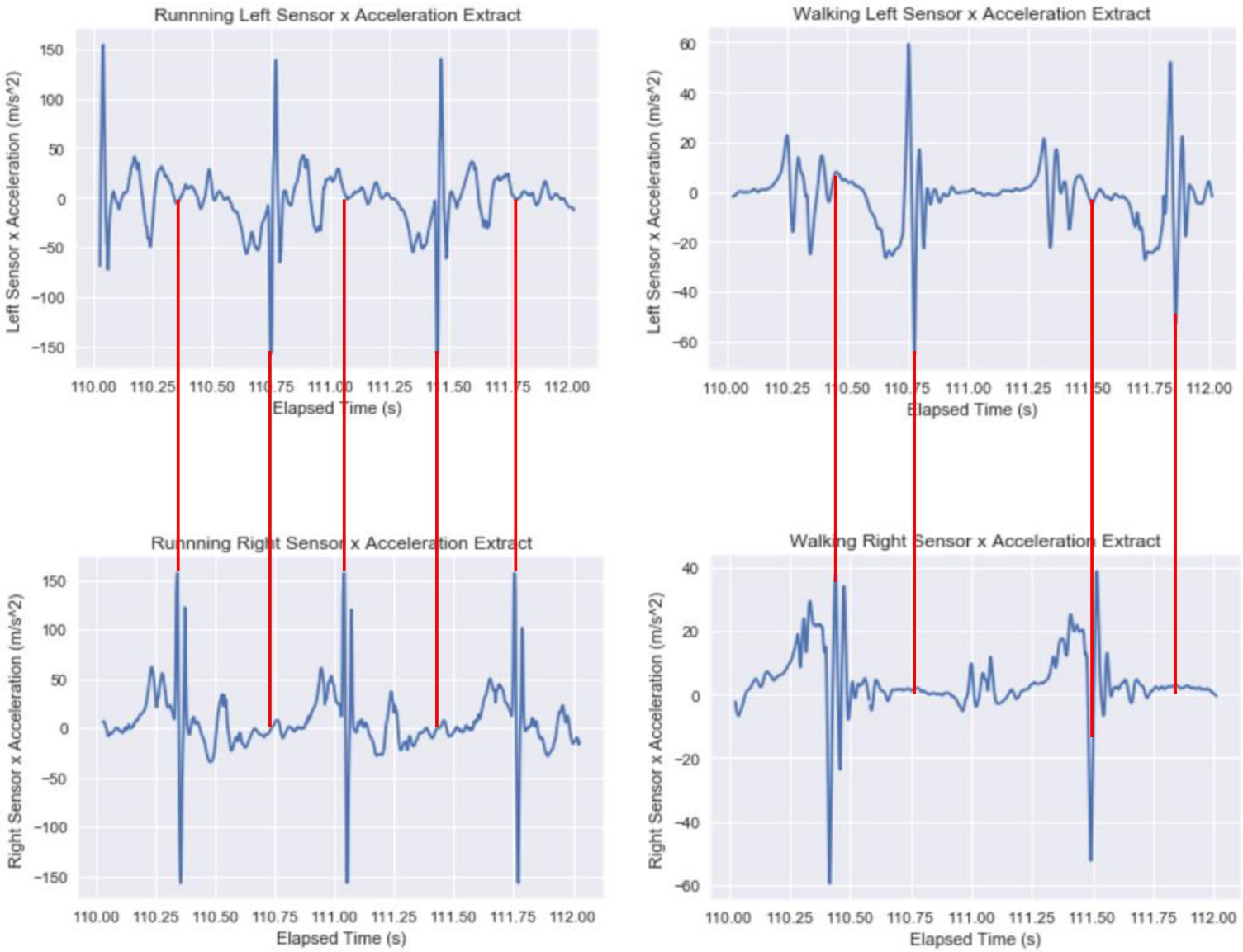
Synchronized sensors

Two sensor units: 18 time series + 9 paired time series



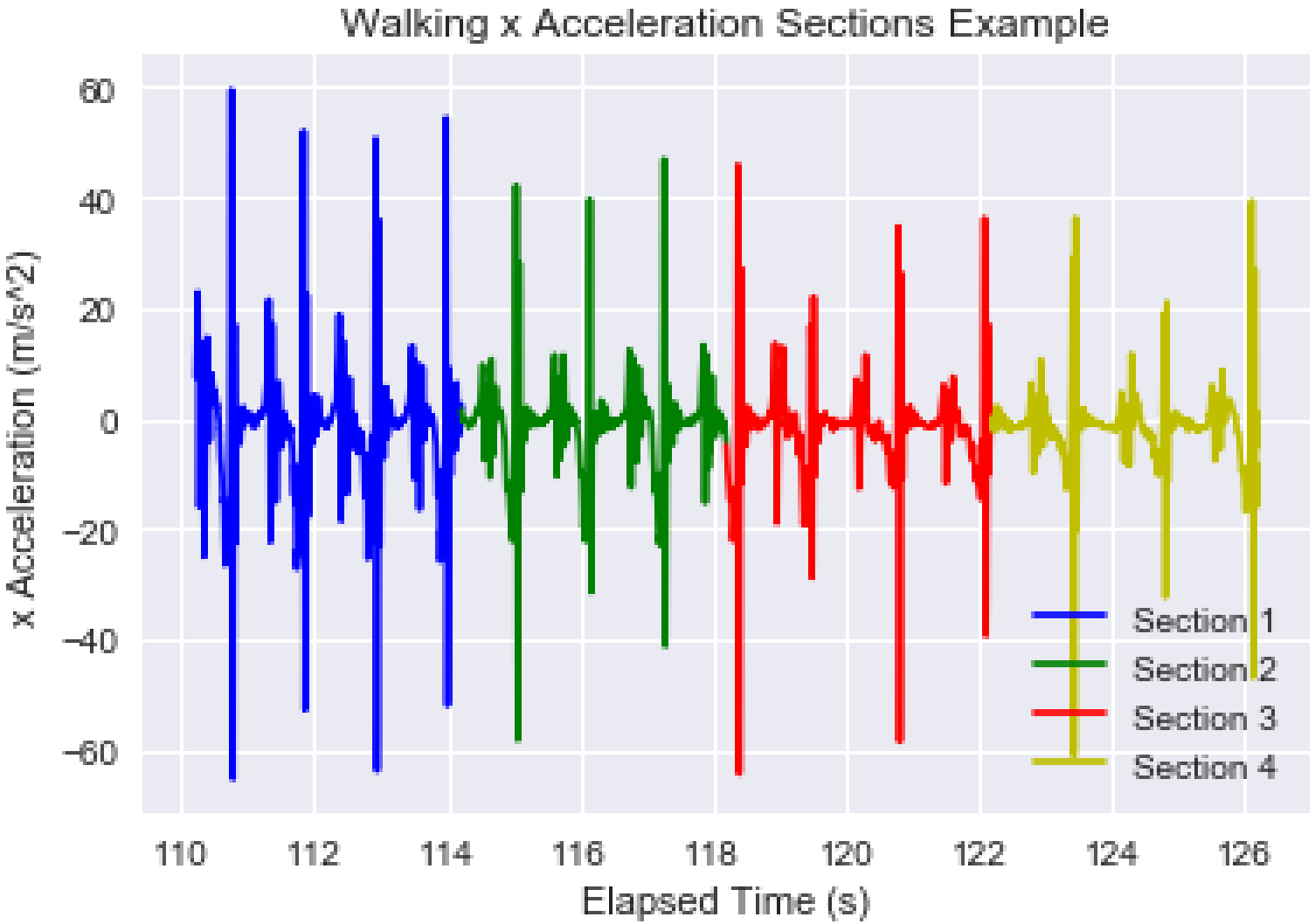
Andreas W. Kempa-Liehr, Jonty Oram, and Thor Besier. "Automating Time Series Feature Engineering for Activity Recognition from Synchronized Inertial Measurement Units". In: *European Conference on Data Analysis (ECDA)*. presentation. Paderborn, Germany, 2018, p. 6

Left-right and running-walking comparison



Kempa-Liehr, Oram, and Besier, "Automating Time Series Feature Engineering for Activity Recognition from Synchronized Inertial Measurement Units", p. 7

Windowing / framing (example)



Jonty Oram. "Classification of athletes' motion from sensor data". BE(Hons) thesis. The University of Auckland: Department of Engineering Science, 2017, Fig. 10

Windowing / framing (algorithm)

```
window_idx = np.asarray(np.floor(jumpNrun.reset_index().index / 100).values,
                        np.int)
window_idx
```

```
array([ 0,  0,  0, ..., 79, 79, 79])
```

```
jumpNrun['window_idx'] = ['j{:02}'.format(idx) for idx in window_idx]
```

```
jumpNrun.head()
```

time	ax	ay	az	|a|	delta_t	window_idx
2020-09-10 14:15:16.540	14.47	0.73	11.34	18.39	00:00:03.500000	j00
2020-09-10 14:15:16.548	9.63	-0.34	12.48	15.76	00:00:03.508000	j00
2020-09-10 14:15:16.558	4.68	-1.88	12.29	13.28	00:00:03.518000	j00
2020-09-10 14:15:16.568	0.75	-2.98	10.50	10.94	00:00:03.528000	j00
2020-09-10 14:15:16.578	-1.79	-4.12	9.75	10.73	00:00:03.538000	j00

General automatic feature engineering (example)

```
from tsfresh.feature_extraction import extract_features

X = extract_features(ts, column_id='window_idx', column_sort='delta_t')
```

```
Feature Extraction: 100%|██████████| 48/48 [00:04<00:00, 10.13it/s]
```

```
X.head()
```

variable id	ax__abs_energy	ax__absolute_sum_of_changes	ax__agg_autocorrelation__f_agg_"mean"__maxlag_40
j00	5106.7276	135.58	-0.062407
j01	7802.6890	251.27	-0.024009
j02	8337.9200	224.01	-0.001983
j03	5619.6522	208.97	0.006786
j04	6439.2676	203.05	-0.062307

5 rows × 3036 columns

Feature Engineering

Feature transformation is about constructing new features from existing features; this is often achieved using mathematical mappings.

Feature generation is about generating new features that are often not the result of feature transformations (e.g. using domain-specific ways)

Feature selection is about selecting a small set of features from a very large pool of features (often included in feature analysis and evaluation).

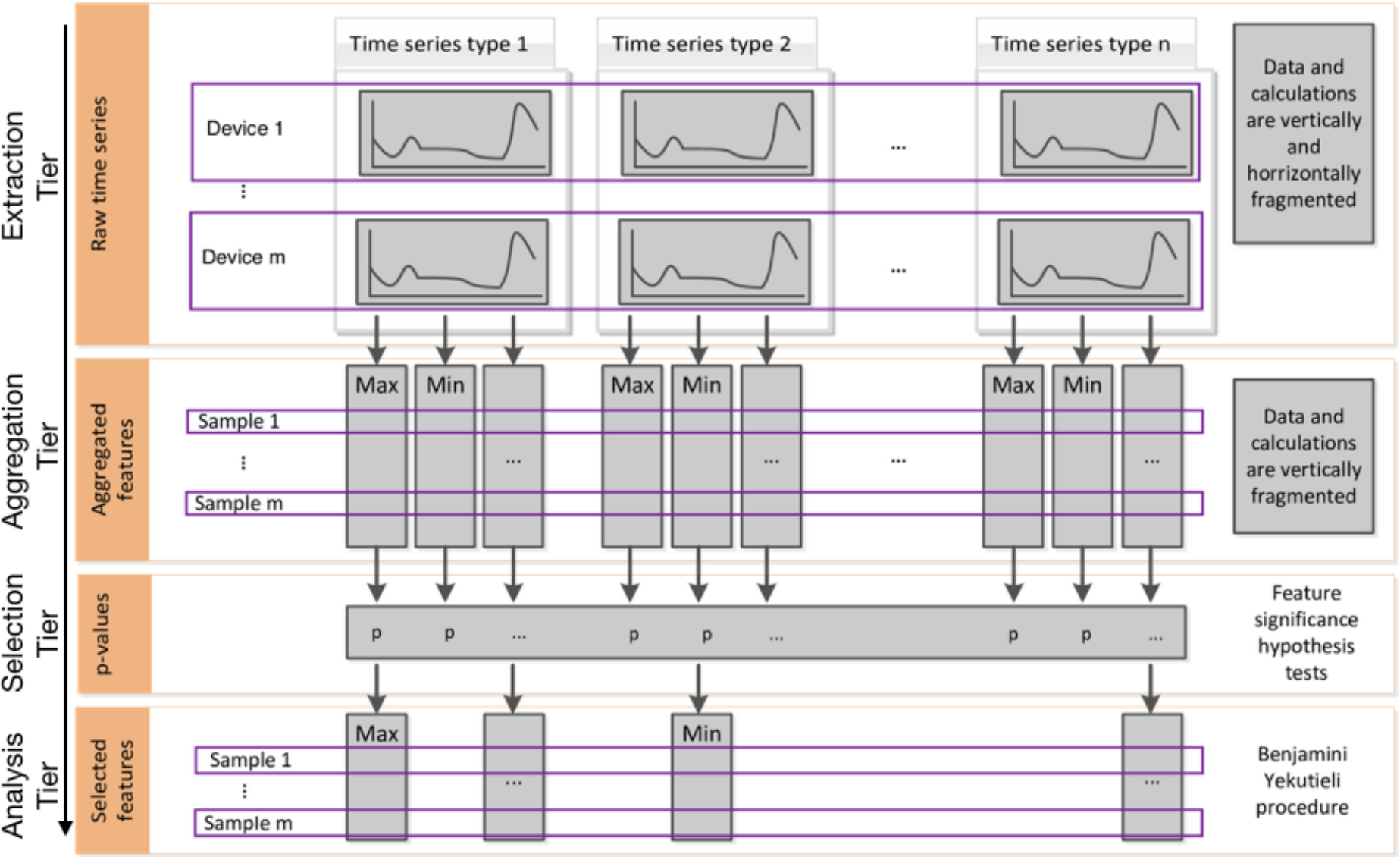
Feature analysis and evaluation is about concepts, methods, and measures for evaluating the usefulness of features and feature sets.

General automatic feature engineering methodology is about generic approaches for automatically generating a large number of features and selecting an effective subset of the generated features.

Feature engineering applications involve feature engineering but the focus is to solve some other data analytic tasks in specific contexts.

Guozhu Dong and Huan Liu, eds. *Feature engineering for machine learning and data analytics*. Boca Raton, FL: Taylor & Francis, 2018

Distributed General automatic feature engineering



Maximilian Christ, Andreas W. Kempa-Liehr, and Michael Feindt. *Distributed and parallel time series feature extraction for industrial big data applications.* Learning 1610.07717v1. Asian Conference on

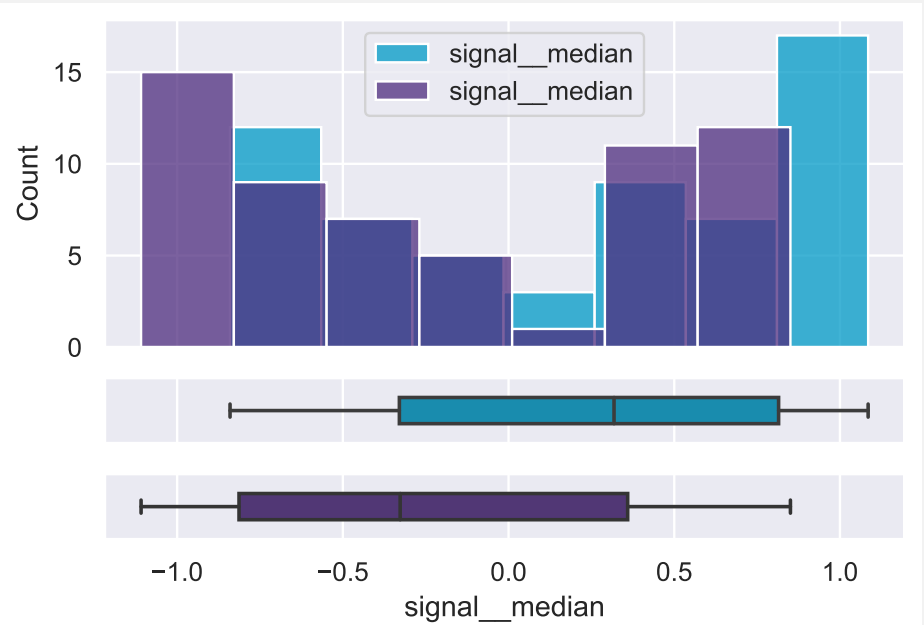
FRESH algorithm

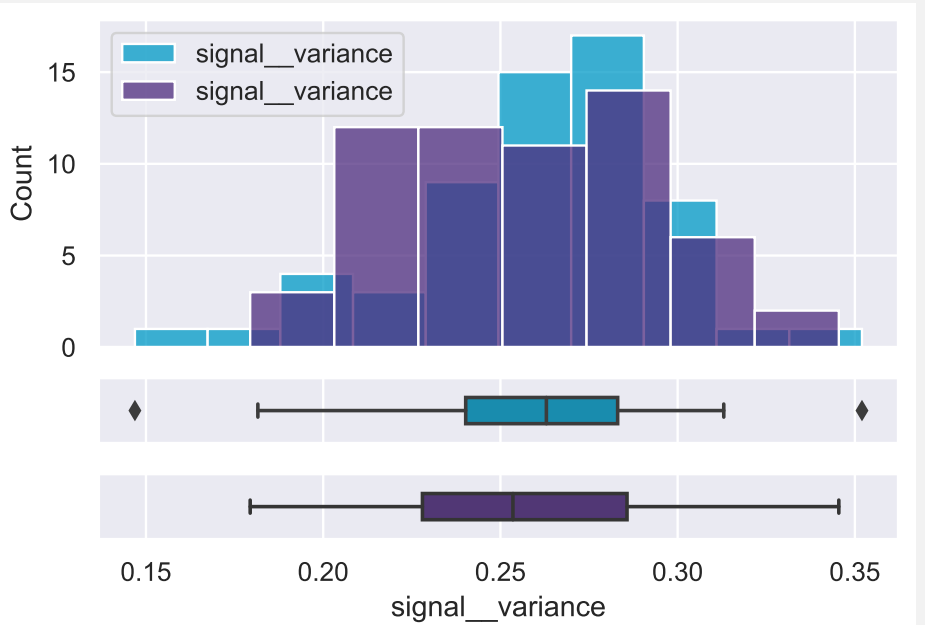
Feature extRaction on basis of Scalable Hypothesis testing (FRESH)
Data: Labelled samples comprising different time series
Result: Relevant time series features
for *all predefined feature extraction algorithms* **do**
  for *all time series* **do**
    for *all samples* **do**
      Apply feature extraction algorithm to time series sample and compute time series feature;
    end
    Test statistical significance of feature for predicting the label;
  end
end
Select significant features while preserving false discovery rate;

Feature selection (Example)

A feature X_ϕ is *relevant* or *meaningful* for the prediction of Y if and only if X_ϕ and Y are not statistically independent.
$\exists\ y_1, y_2$ with $f_Y(y_1) > 0, f_Y(y_2) > 0\ :\ f_{X|Y=y_1} \neq f_{X|Y=y_2}$

Example from FE05





Feature selection (Hypothesis testing)

A feature X_ϕ is *relevant* or *meaningful* for the prediction of Y if and only if X_ϕ and Y are not statistically independent.
$\exists\ y_1, y_2$ with $f_Y(y_1) > 0, f_Y(y_2) > 0\ :\ f_{X|Y=y_1} \neq f_{X|Y=y_2}$

Null hypothesis

$$H_0^\phi = \{X_\phi \text{ is irrelevant for predicting } Y\},$$
$$H_1^\phi = \{X_\phi \text{ is relevant for predicting } Y\}.$$

Binary target

binary feature Fisher's exact test

non binary feature Mann-Whitney U test

General automatic feature engineering (example)

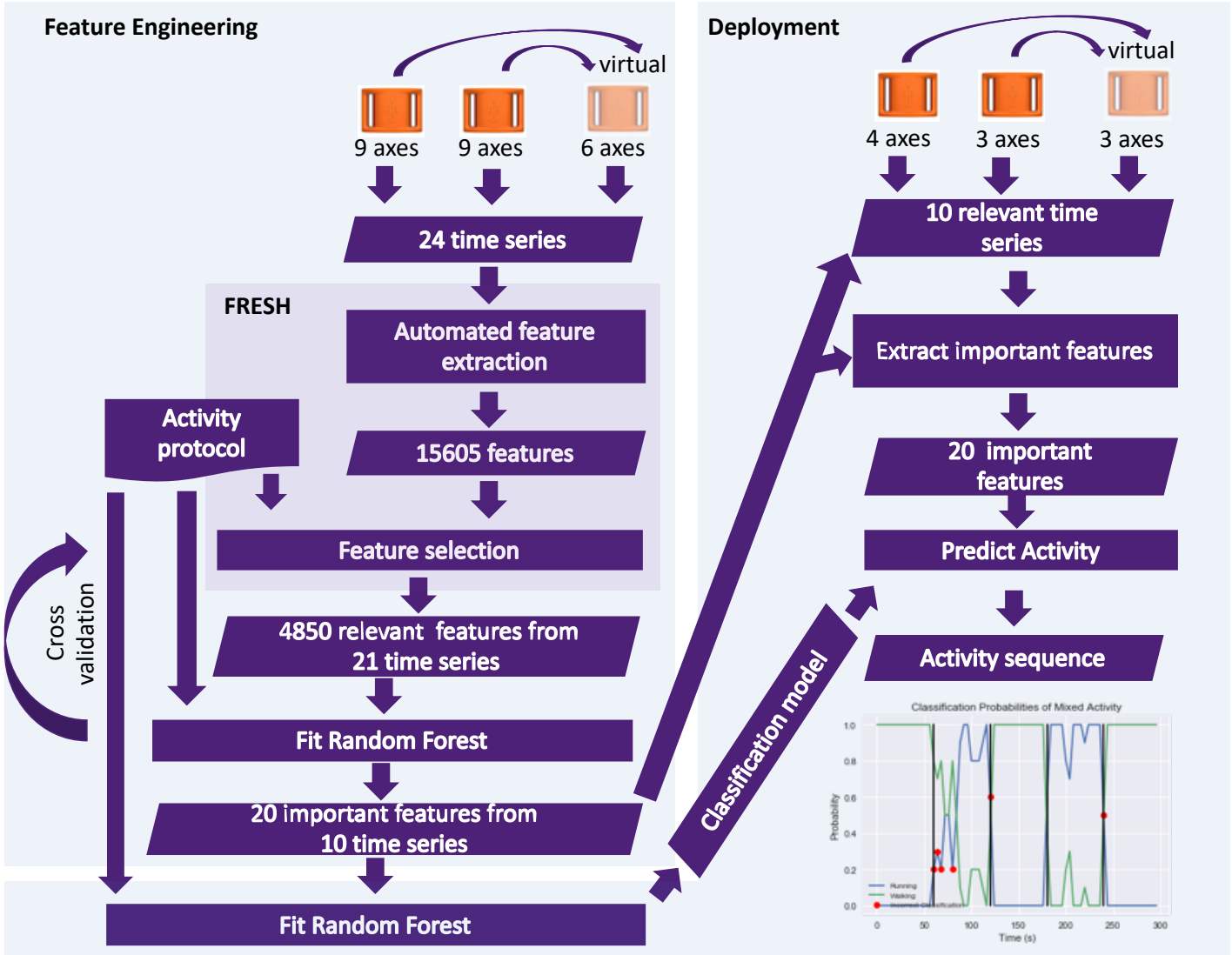
```
from tsfresh.transformers import FeatureSelector
from tsfresh.utilities.dataframe_functions import impute

select = FeatureSelector()
select.fit(X.dropna(axis=1), y)
```

```
FeatureSelector(chunksize=None, fdr_level=0.05, hypotheses_independent=False,
                ml_task='auto', n_jobs=10,
                test_for_binary_target_binary_feature='fisher',
                test_for_binary_target_real_feature='mann',
                test_for_real_target_binary_feature='mann',
                test_for_real_target_real_feature='kendall')
```

```
p_values = pd.Series(select.p_values,
                     index = select.features)
p_values.head()
```

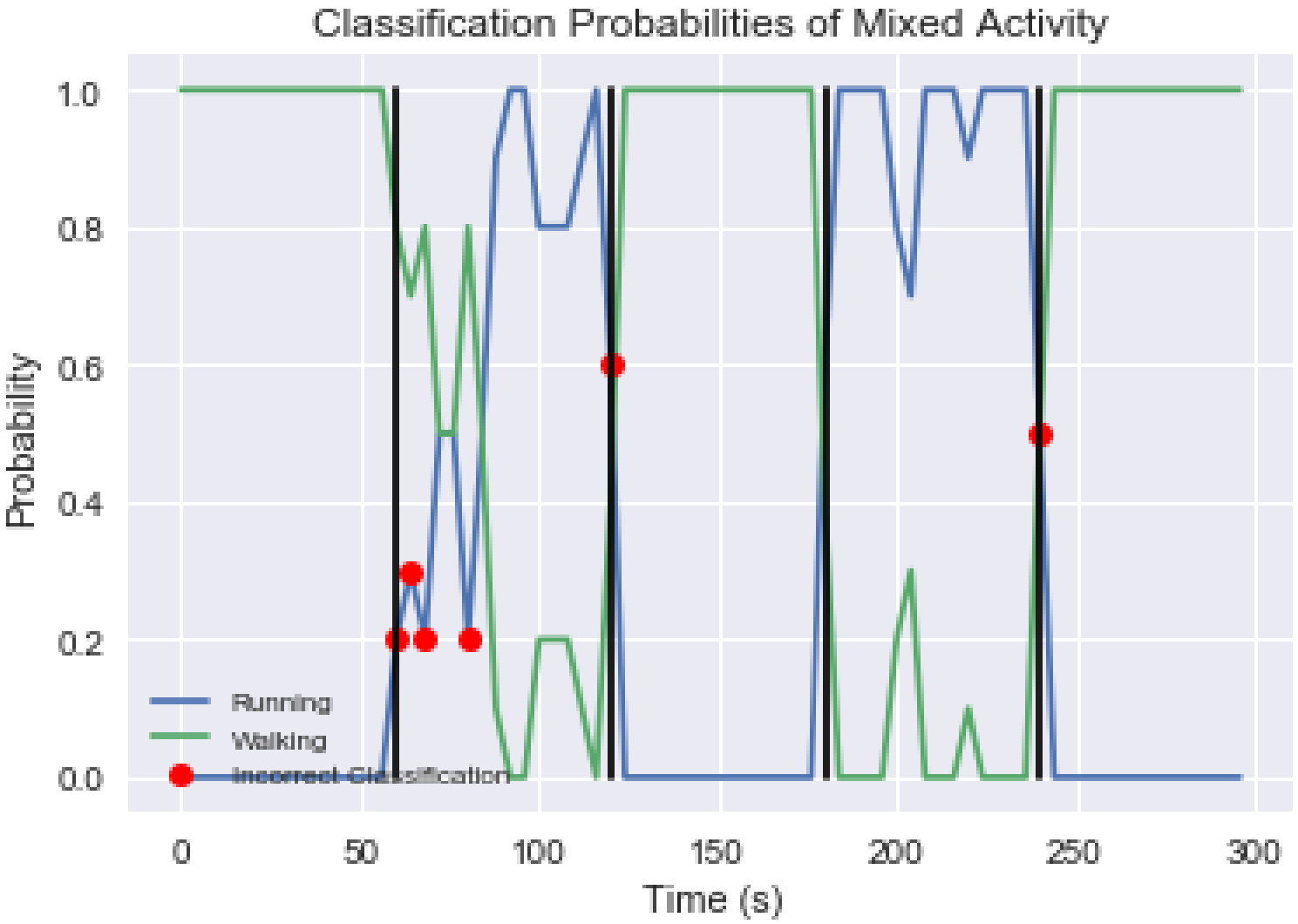
```
ax__range_count__max_1__min_-1    9.485052e-29
ax__maximum                       1.051853e-28
|a|__quantile__q_0.2              1.052111e-28
|a|__quantile__q_0.8              1.052197e-28
|a|__median                       1.052282e-28
dtype: float64
```

Feature engineering worfklow with virtual sensors



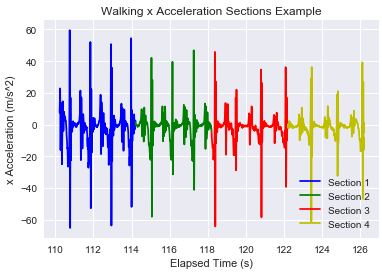
Andreas W. Kempa-Liehr et al. "Feature engineering workflow for activity recognition from synchronized inertial measurement units". In: *Pattern Recognition. ACPR 2019*. Ed. by Michael Cree et al. Vol. 1180. Communications in Computer and Information Science (CCIS). Singapore: Springer, 2020, pp. 223–231. DOI: 10.1007/978-981-15-3651-9_20, Fig. 2a

Validation with separate run



Kempa-Liehr et al., "Feature engineering workflow for activity recognition from synchronized inertial measurement units", Fig. 3

Classification Running vs. Walking



Oram, "Classification of athletes' motion from sensor data", Fig. 10

- 560 seconds of mixed running and walking
- 280000 measurements for each of the 18 sensors (plus 9 paired measurements)
- 140 sections of 4s length (82 walking, 58 running)
- 16283 features in total
- 5212 statistically significant features (false discovery rate 50%)
- Random Forest achieves 100% accuracy under 10-fold cross-validation

Interpretable machine learning models

Naming scheme supports interpretability

`kind` name of the time series the feature is based on,

`calculator` name of the function, which has been used to extract the feature,

`parameter_value` pairs configuration of respective feature calculator

`[kind]__[calculator]__[parameterA]_[valueA]__[parameterB]_[valueB]`

Most important features

Ranked by mean feature importance of 100k random forests.

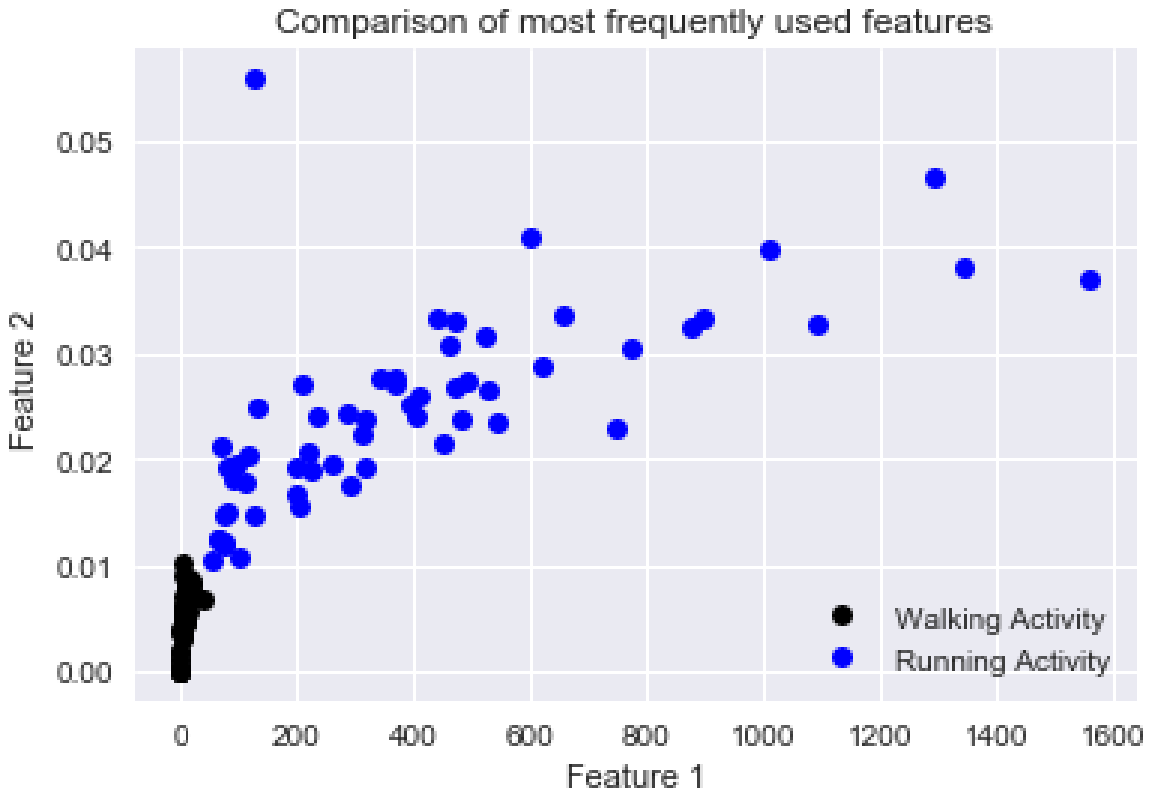
1. `gyro_z_l__change_quantiles__f_agg_"var"__isabs_False__qh_0.6__ql_0.4`

2. `accel_z_l__agg_linear_trend__f_agg_"min"__chunk_len_10__attr_"stder r"`

3. `gyro_y_diff__change_quantiles__f_agg_"var"__isabs_False__qh_1.0__ql _0.4`

Scatter plot of most important features

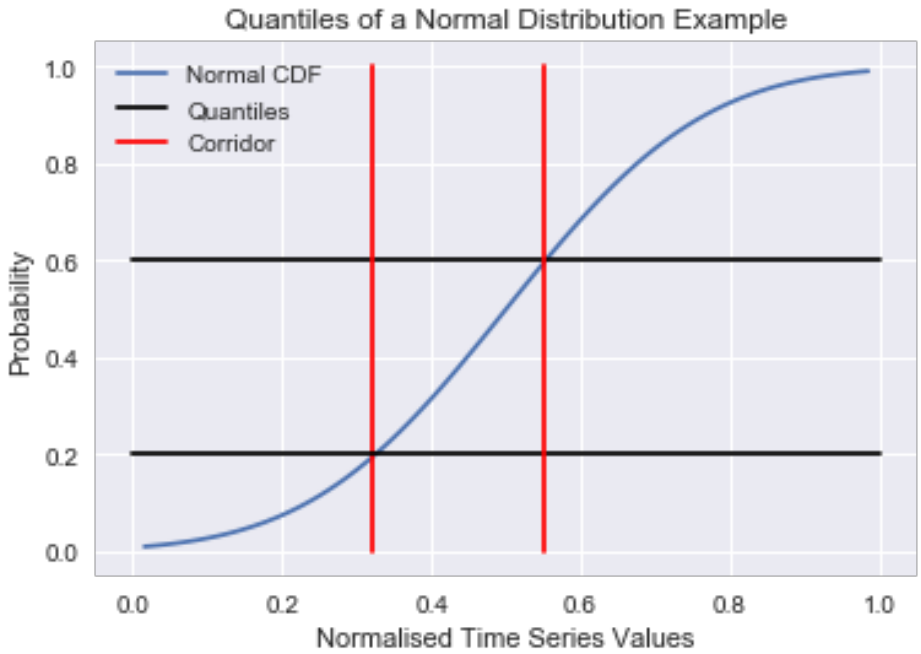
```
Feature 1:
  gyro_z_l__change_quantiles__f_agg_"var"__isabs_False__qh_0.6__ql_0.4
Feature 2:
  accel_z_l__agg_linear_trend__f_agg_"min"__chunk_len_10__attr_"stderr"
```

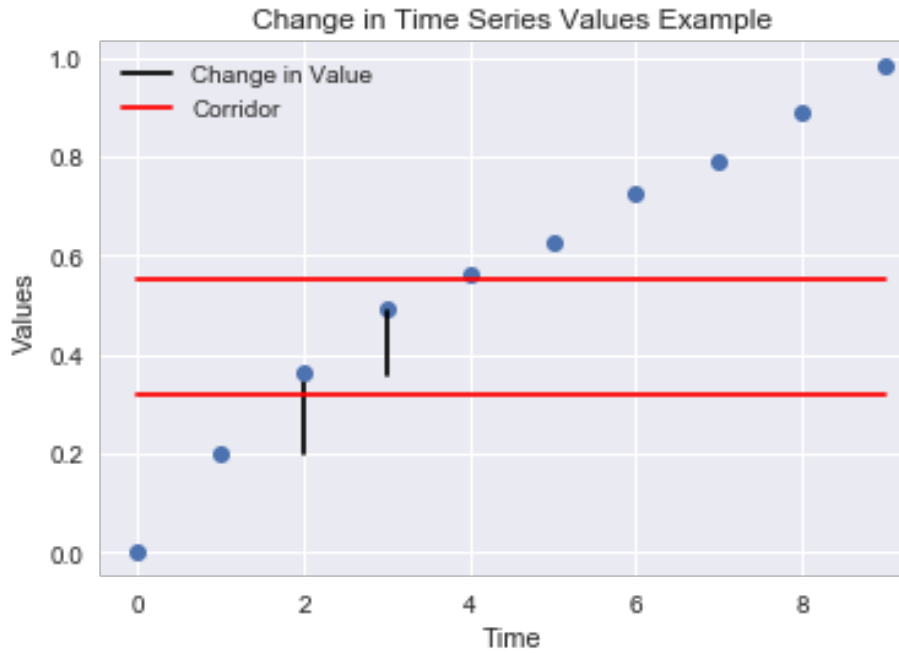


Kempa-Liehr, Oram, and Besier, "Automating Time Series Feature Engineering for Activity Recognition from Synchronized Inertial Measurement Units", p. 15

Feature 1 – Change quantiles

`gyro_z_l__change_quantiles__f_agg_"var"__isabs_False__qh_0.6__ql_0.2`

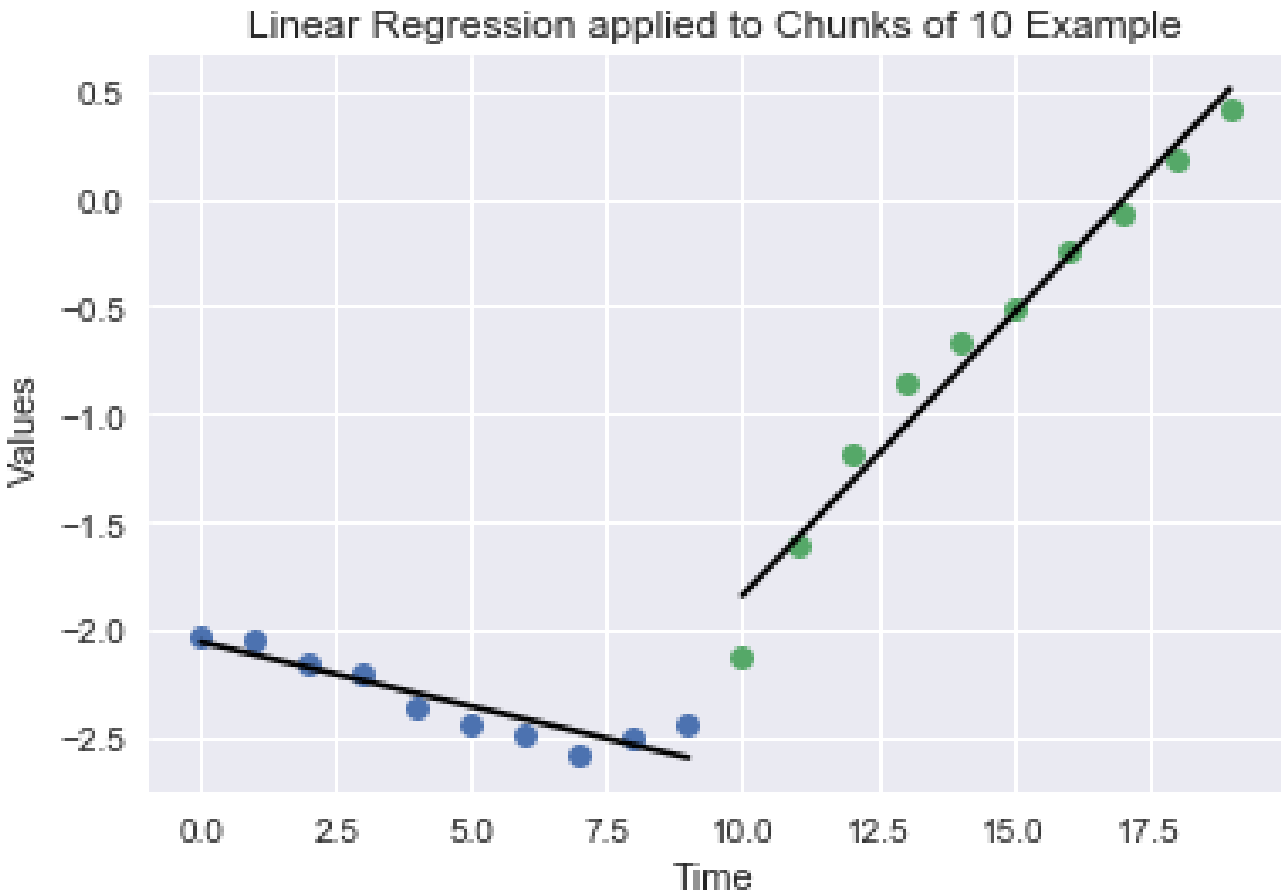




Kempa-Liehr, Oram, and Besier, "Automating Time Series Feature Engineering for Activity Recognition from Synchroniz Inertial Measurement Units", p. 16

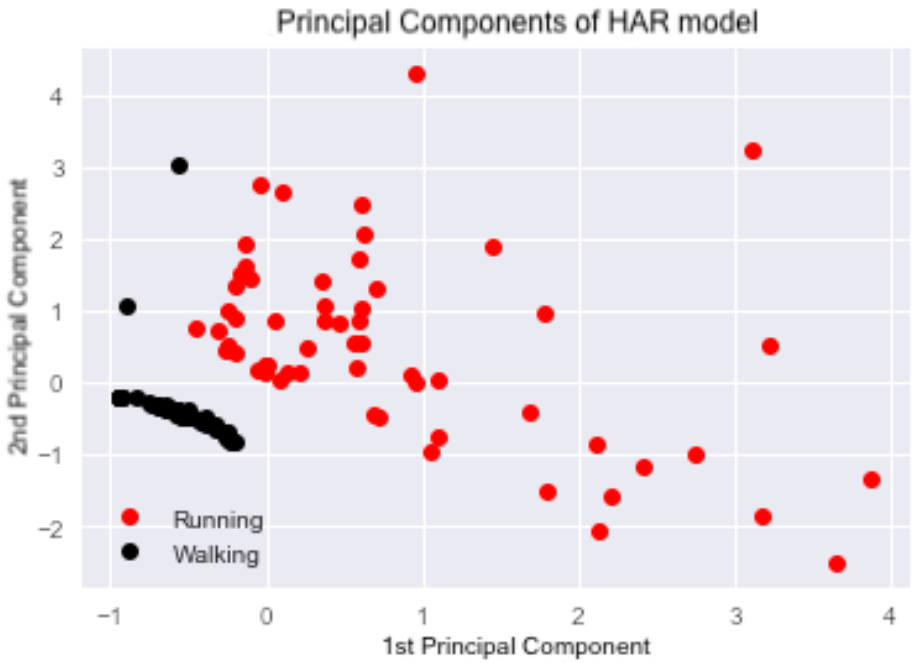
Figure 2 – Agg linear trend feature

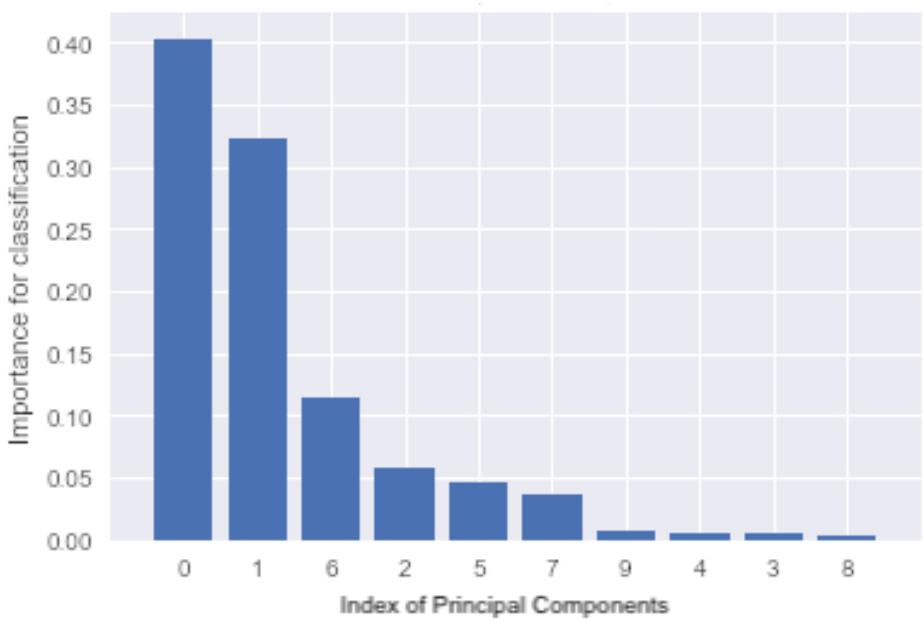
`accel_z_l__agg_linear_trend__f_agg_"min"__chunk_len_10__attr_"stderr"`



Kempa-Liehr, Oram, and Besier, "Automating Time Series Feature Engineering for Activity Recognition from Synchronized Inertial Measurement Units", p. 17

Principal components

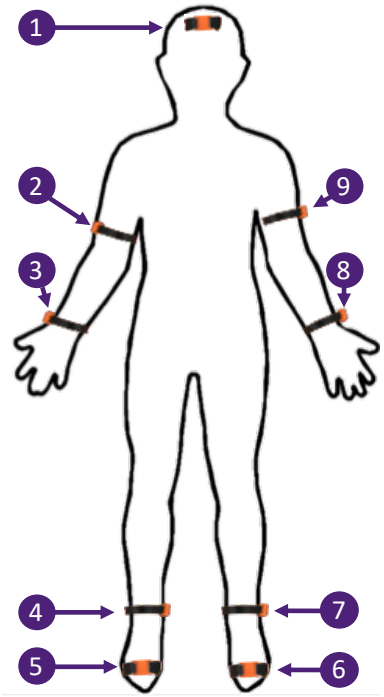


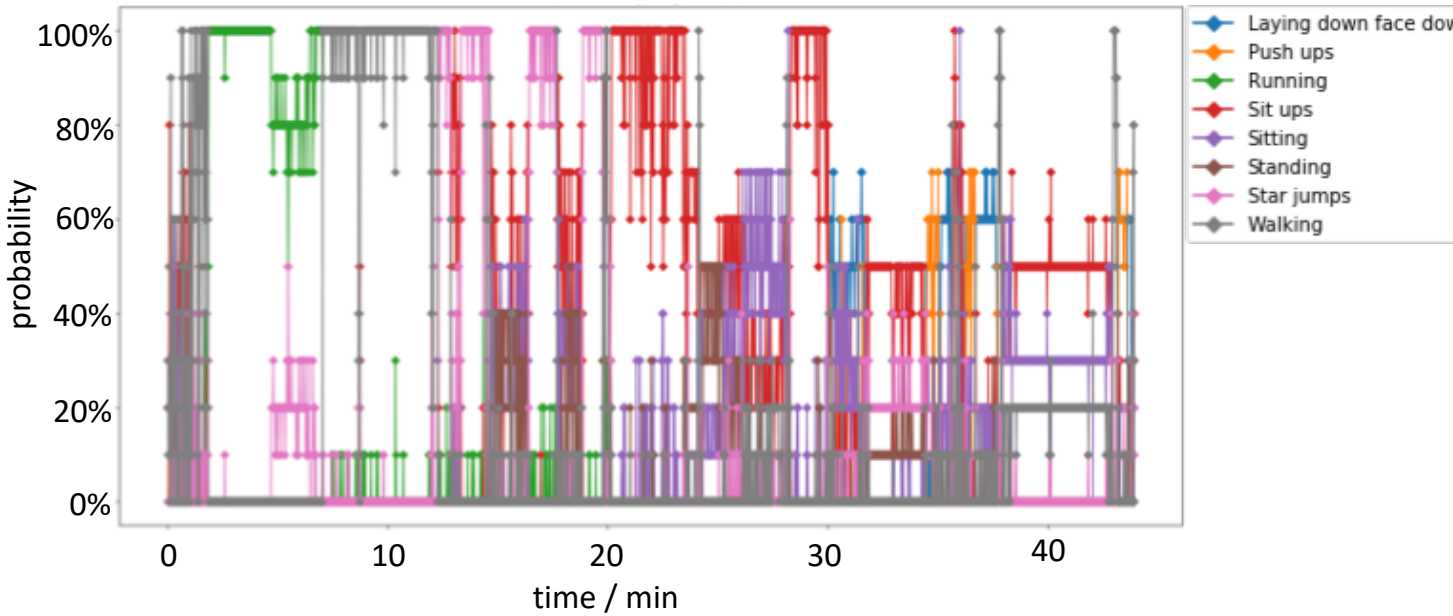


Adapted from Figs 17a and 17b of Oram, "Classification of athletes' motion from sensor data"

Multi-activity classification

The best combination of sensors was top right arm (2) and tip of right foot (5).

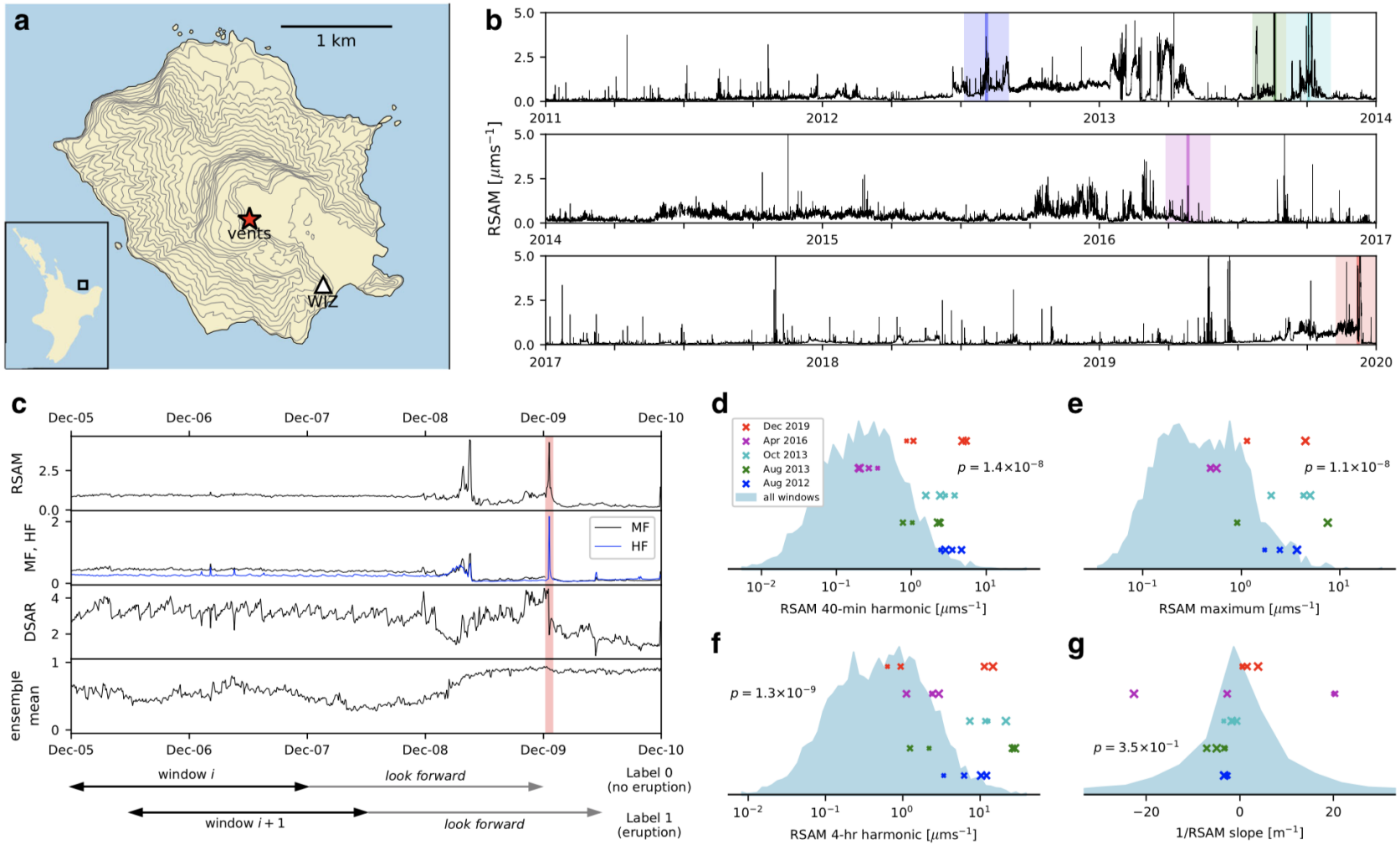




Kempa-Liehr et al., "Feature engineering workflow for activity recognition from synchronized inertial measurement units" Figs 1a & 4

Volcanic eruptions

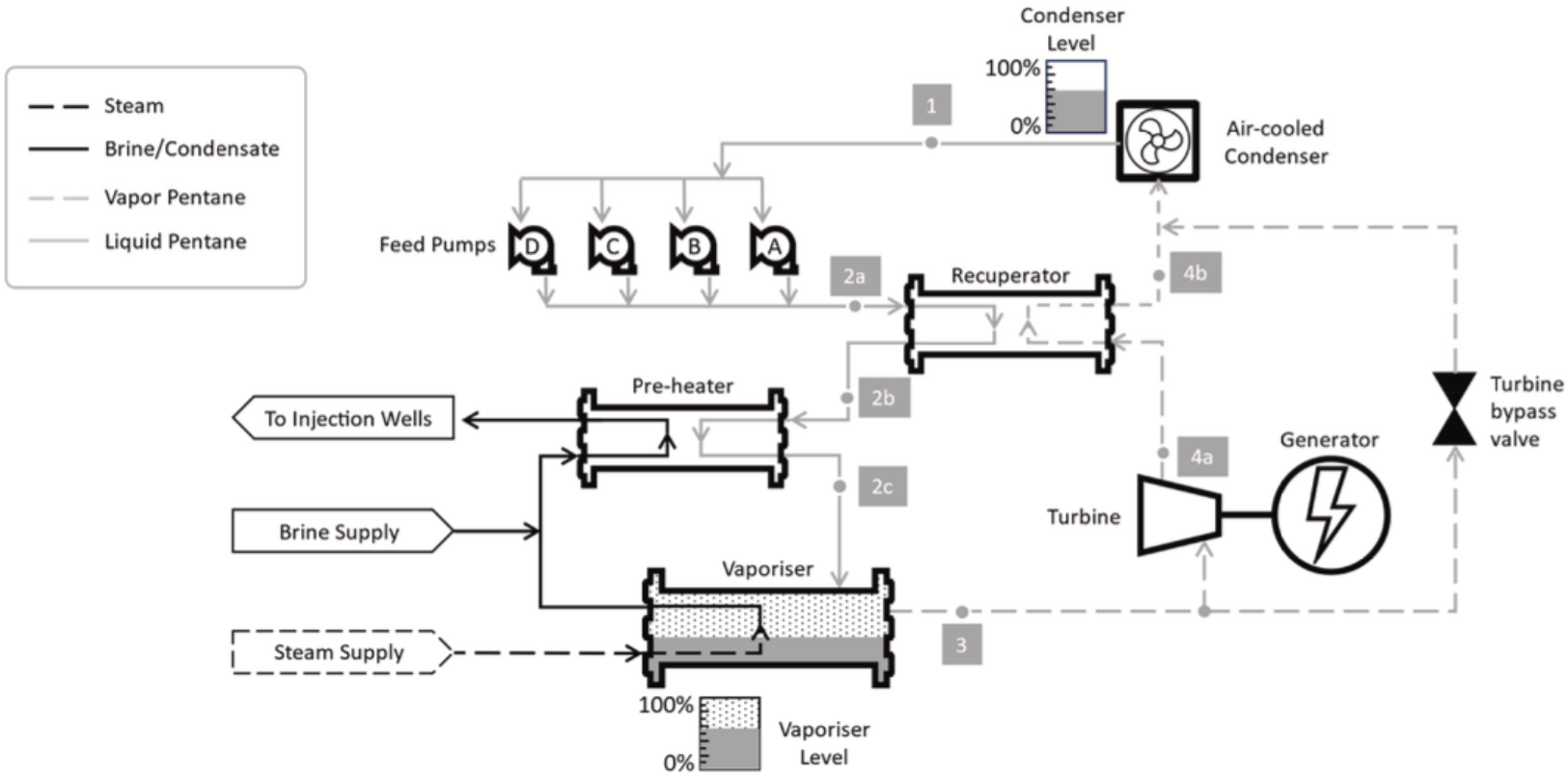
Models on imbalanced data sets can be effectively fitted using undersampling.



D. E. Dempsey et al. "Automatic precursor recognition and real-time forecasting of sudden explosive volcanic eruptions at Whakaari, New Zealand". In: *Nature Communications* 11.3562 (2020), pp. 1–8. DOI: 10.1038/s41467-020-17375-2

Te Huka Powerplant in New Zealand

- 209 GWh
- working fluid is n-pentane



Paul Michael B. Abrasaldo et al. "Detection of abnormal operation in geothermal binary plant feed pumps using time-series analytics". In: *Expert Systems With Applications* 247.123305 (2024), pp. 1–17. DOI: 10.1016/j.eswa.2024.123305, Fig 1

Abnormal operating conditions

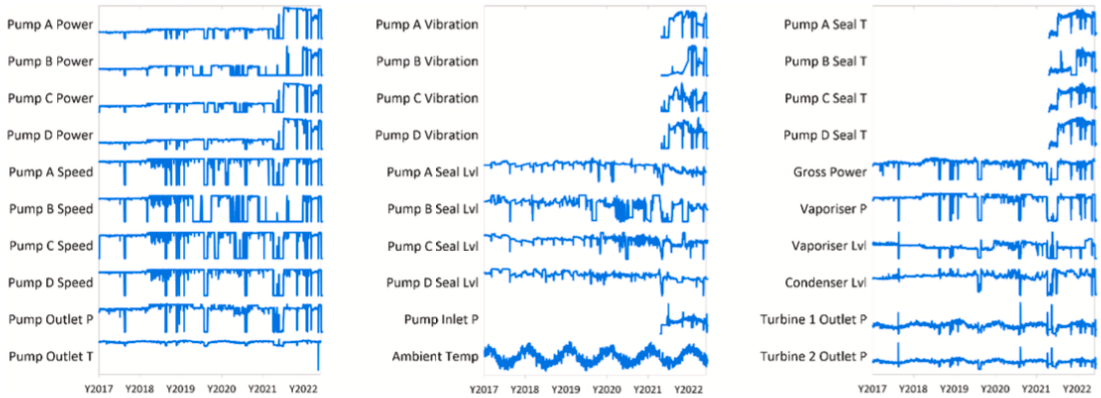


Fig. 3. Sparklines of the monitored system parameters showing changes in relative magnitude across the covered period. Significant changes in magnitude observed in the pump power measurements were due to a change in measurement units and had to be corrected during data pre-processing.

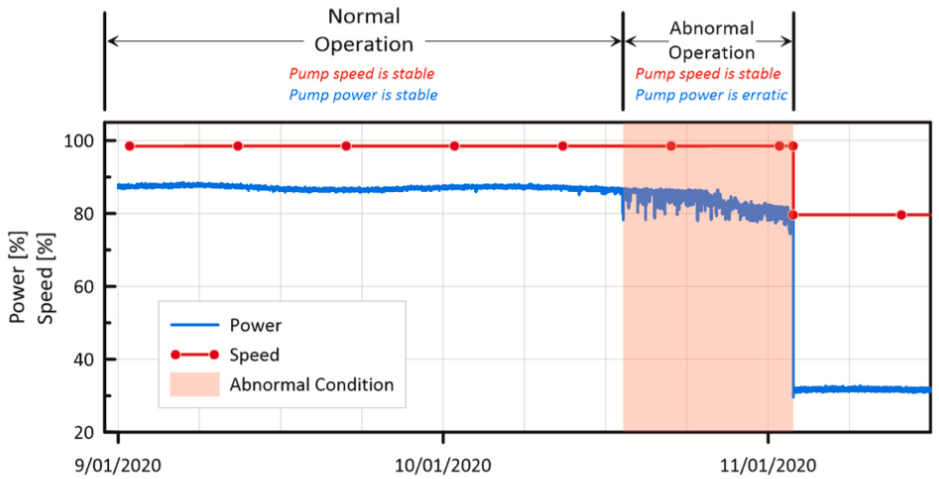
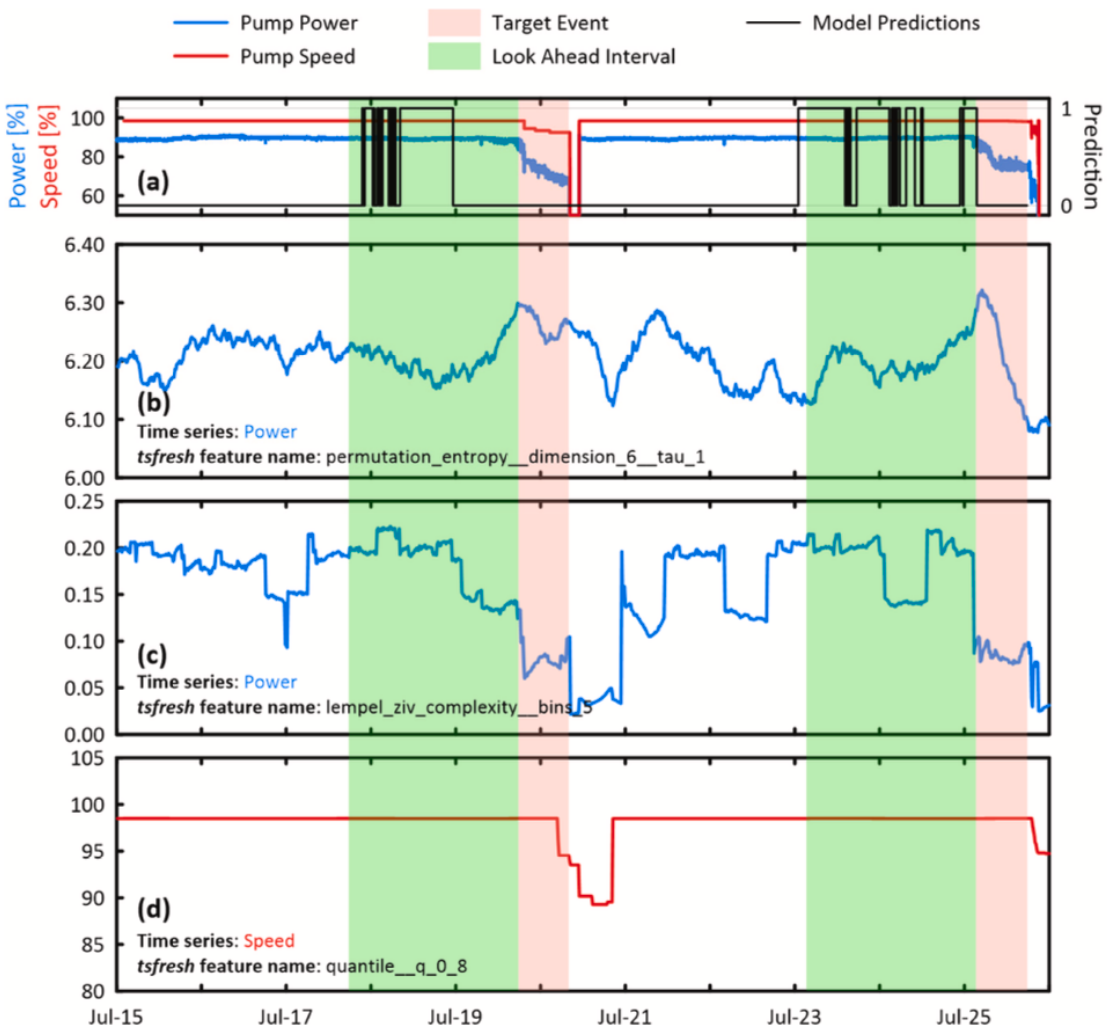


Fig. 4. Pump power and speed time series showing normal and abnormal operating conditions. The target events under investigation are characterised by an erratic, decreasing trend in the pump power time series while the pump speed remains constant.

Abrasaldo et al., "Detection of abnormal operation in geothermal binary plant feed pumps using time-series analytics", Figs 3 & 4

Explainable machine learning

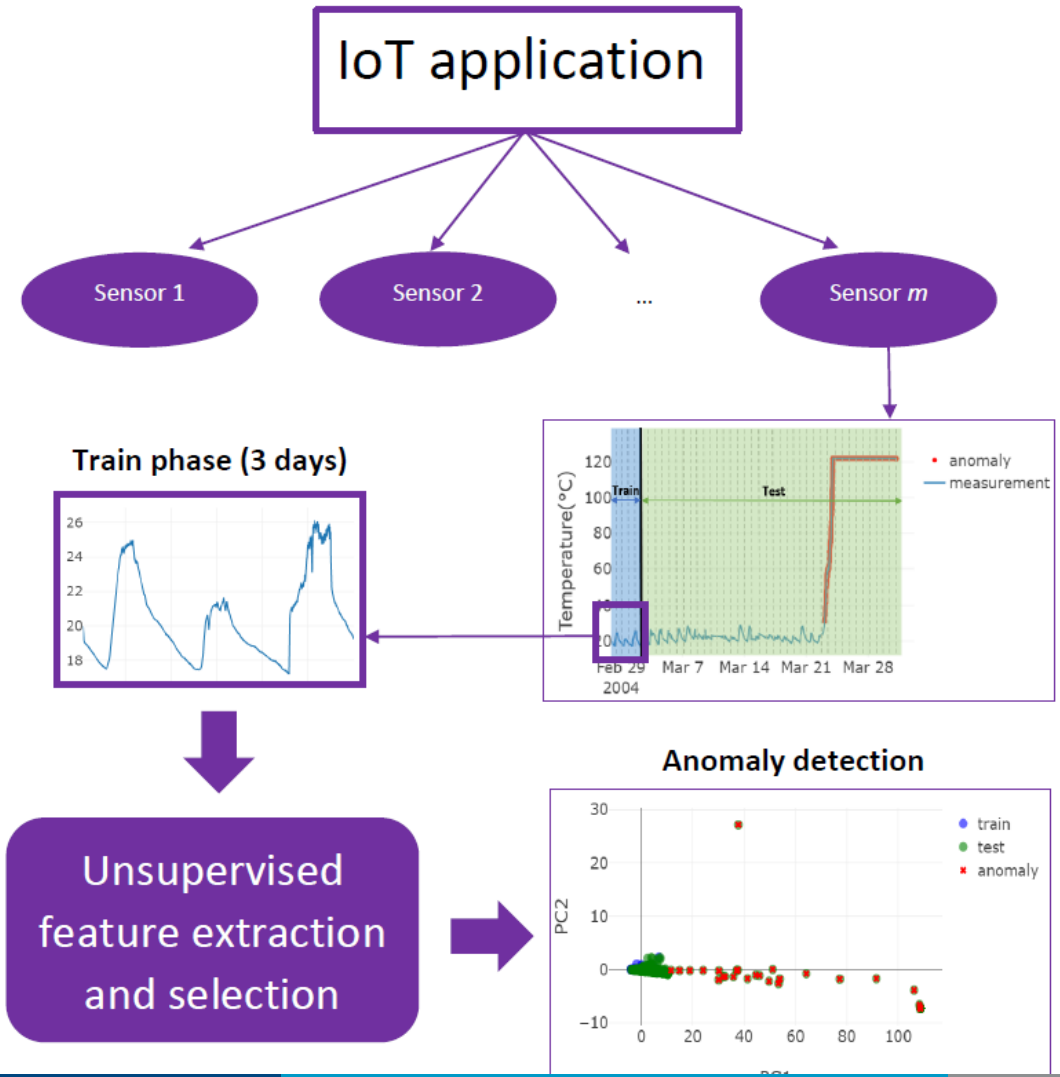
- Abnormal events were 22% more likely if 4 pumps were running.



Abrasaldo et al., "Detection of abnormal operation in geothermal binary plant feed pumps using time-series analytics", Fig 15

Identifying anomalies without examples

Learning normality models from short calibration phase



Yie Teh, Kevin I-Kai Wang, and Andreas W. Kempa-Liehr. "Expect Unexpected: Unsupervised feature selection for automated sensor maly detection". In: *IEEE Sensors Journal* 15.16 (2021), 18033–18046. DOI: 10.1109/JSEN.2021.3084970

Using *normality* as baseline for feature selection

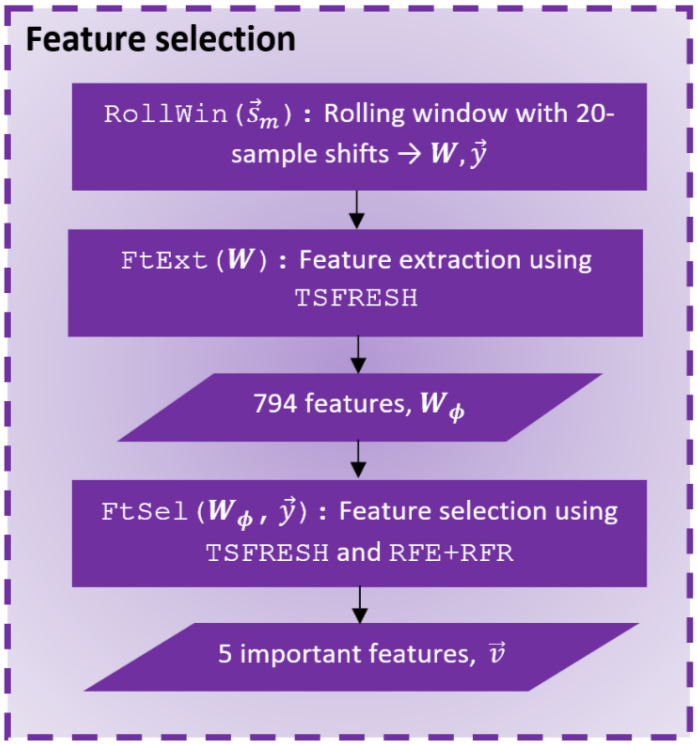


Fig. 2. Flowchart of the proposed unsupervised feature selection for sensor data with generation of rolling windows (RollWin), feature extraction and standardization (FtExt), and feature selection (FtSel) combining univariate hypothesis testing (TSFRESH) with a random forest regressor (RFE) and recursive feature elimination (RFE). The algorithm is agnostic with respect to labelled anomalies but uses statistics $\vec{y}$ of future measurements for the feature selection.

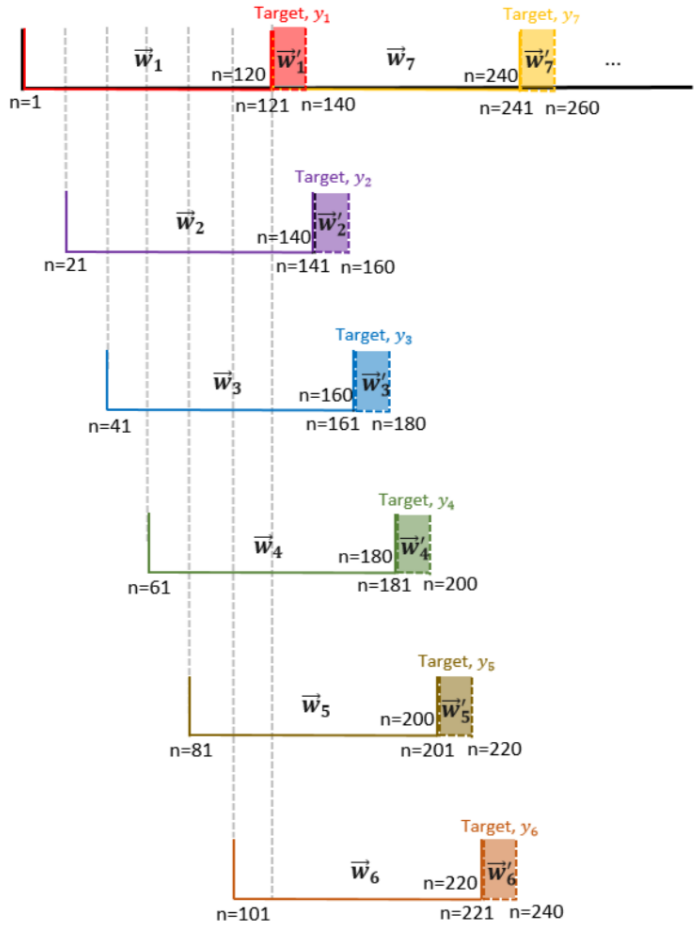
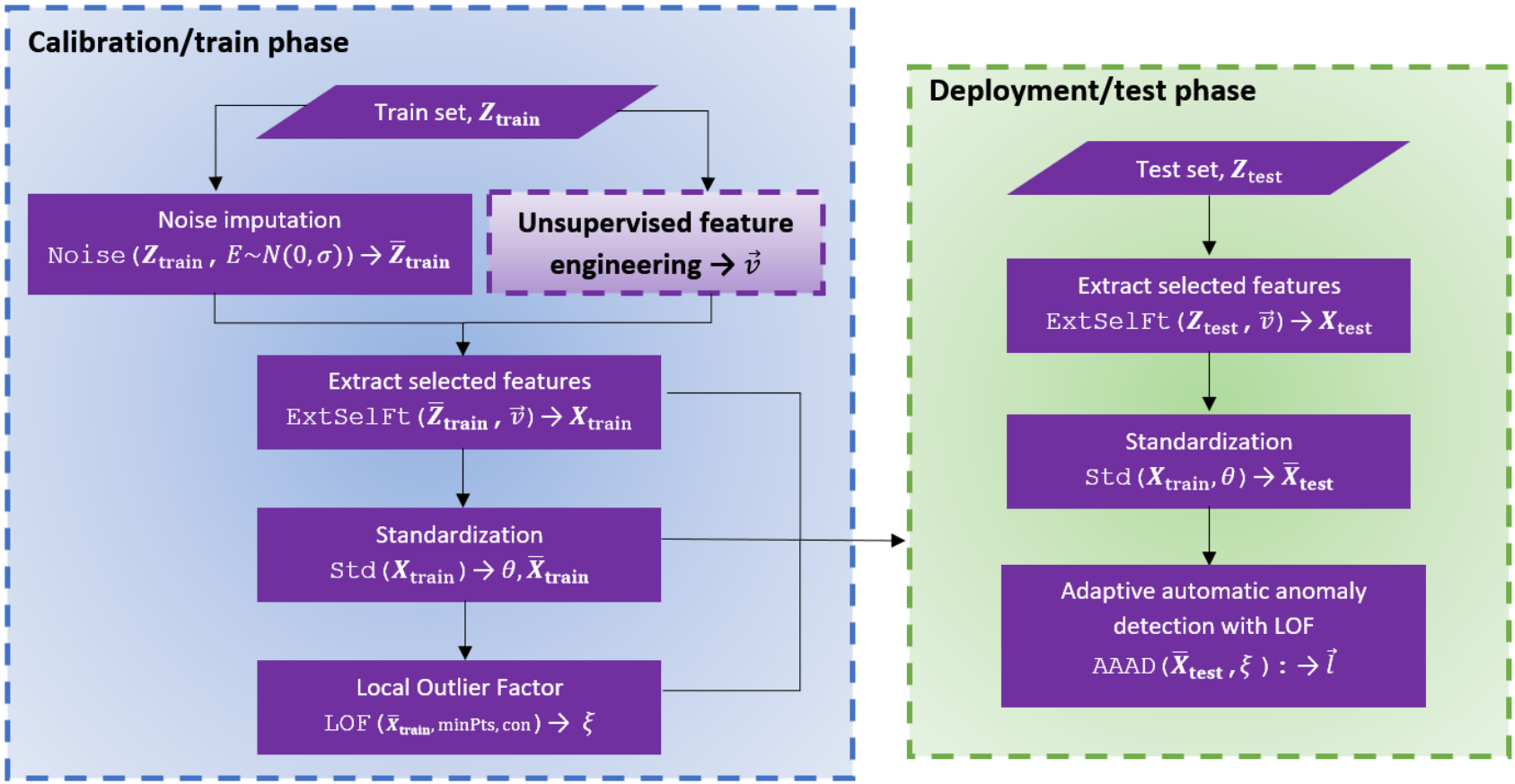


Fig. 3. Rolling window applied of calibration period. Every window is shifted by 20 samples, e.g. the first window, $\vec{w}_1$ starts at sample $n = 1$, and shifts by 20 samples in the next rolling window iteration, starting at $n = 21$, and so on.

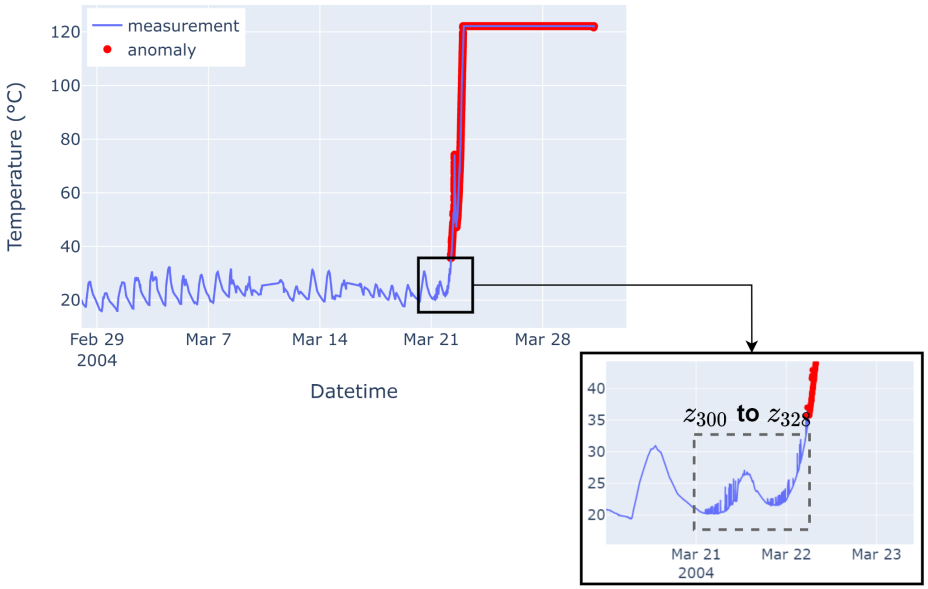
Teh, Wang, and Kempa-Liehr, "Expect the Unexpected: Unsupervised feature selection for automated sensor anomaly detection"

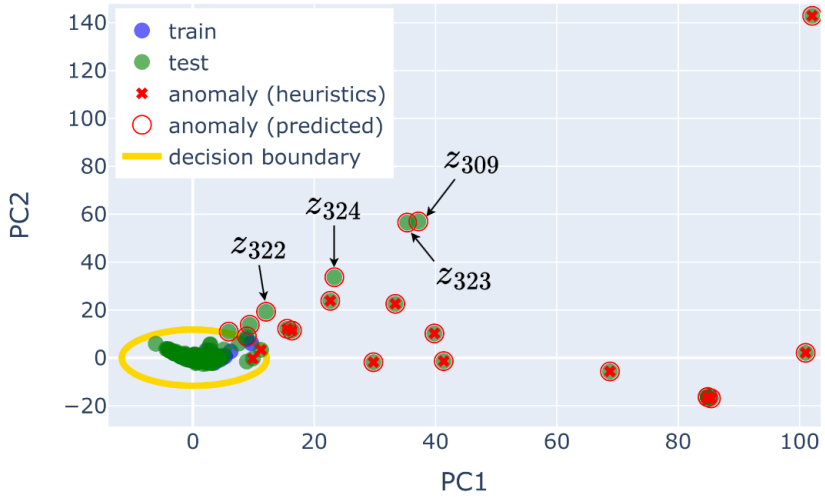
Learning normality models



Hui Yie Teh, Kevin I-Kai Wang, and Andreas W. Kempa-Liehr. "Feature-based normality models for anomaly detection". In: *Sensors* 25.4757 (2025), pp. 1–25. DOI: 10.3390/s25154757, Fig 3

Anomaly detection





Time-series feature engineering ⇒ Engineering of time-series

The FRESH algorithm and the machine library `tsfresh` automate the engineering of time-series features.

Python – `tsfresh`

Christ, Kempa-Liehr, and Feindt, *Distributed and parallel time series feature extraction for industrial big data applications*

Christ et al., "Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package)"

Time series engineering

`tsfresh`	⇒	Time-series feature engineering for free
time series engineering	⇒	Which signals/time series could be useful?

Fault prediction on powerlines

- 2904 measurements of powerline voltage (**insulated**) in different locations
- **Predictive maintenance:** Presence of Partial Discharge (PD) indicates a future fault
- Partial Discharge can be easily mistaken for external background noise (EBN)
- This makes detecting future faults difficult!

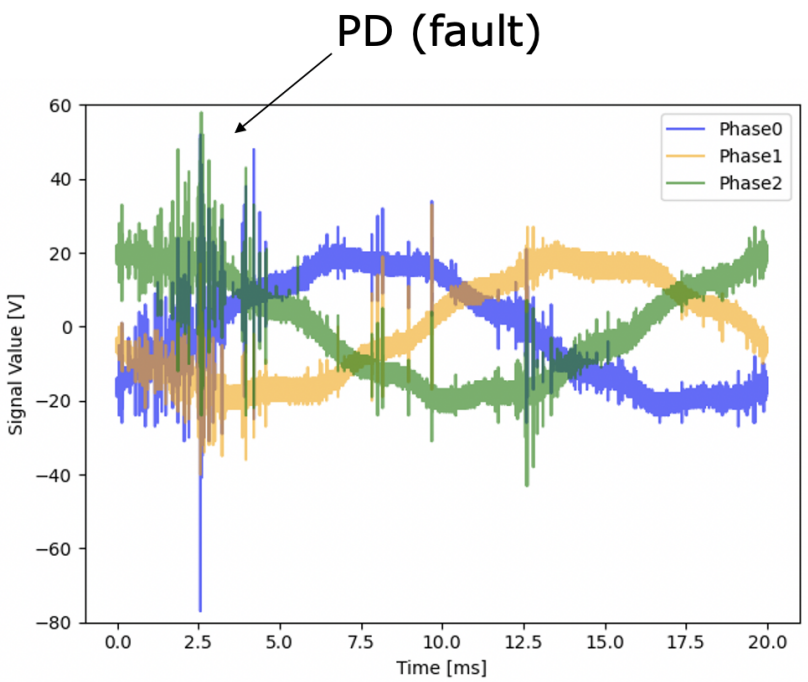
Sourced from
https://earth-planets-space.springeropen.com/articles/10.1186/s40623-018-0948-8

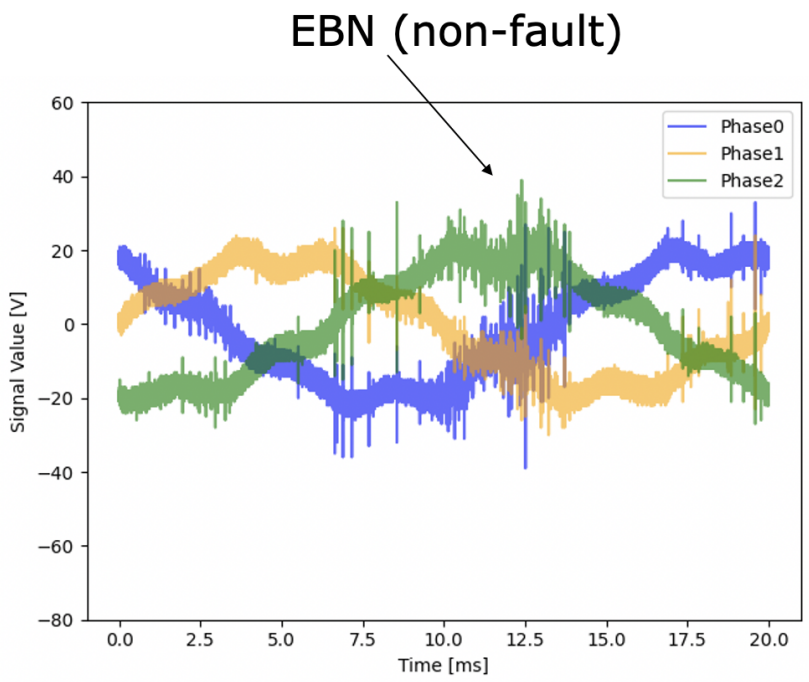
Scott Simmons & Louis Jarvis, BHons presentation 2021

Partial Discharge is hard to detect

PD Partial Discharge

EBN External Background Noise

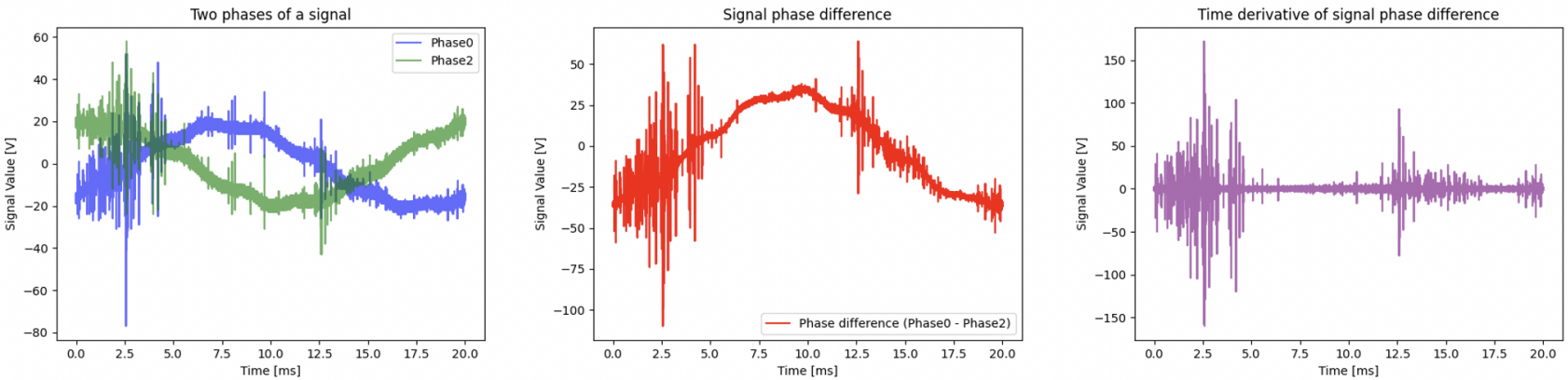




Scott Simmons et al. "Data Mining on Extremely Long Time-Series". In: *2021 International Conference on Data Mining Workshops (ICDMW)*. IEEE. Los Alamitos, 2021, pp. 1057–1066. DOI: 10.1109/ICDMW53433.2021.00137

Time-series engineering

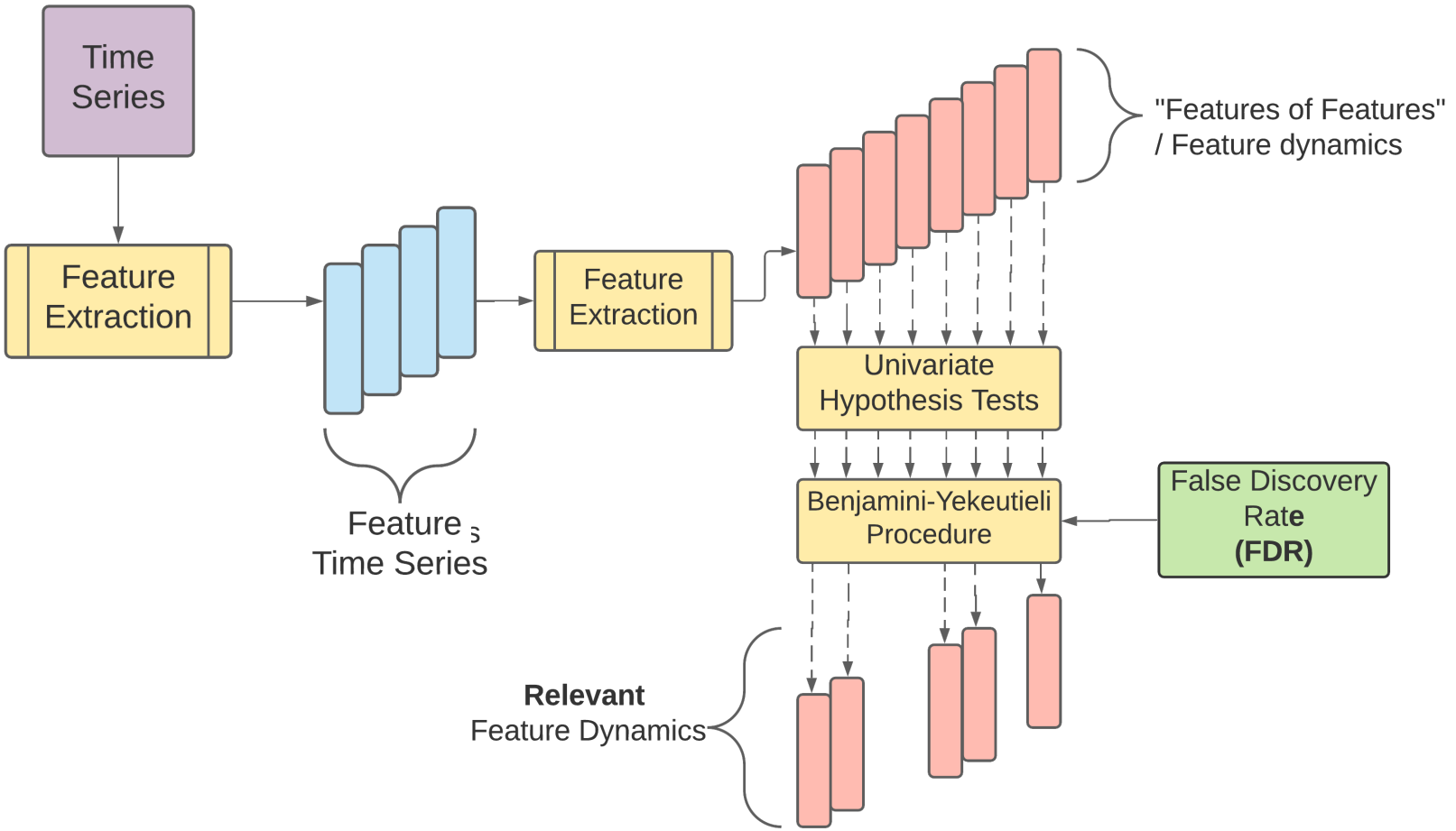
Step 1: Engineer Input Time Series



Raw time series → **Compute phase difference** → **Compute time derivative**

Simmons et al., "Data Mining on Extremely Long Time-Series"

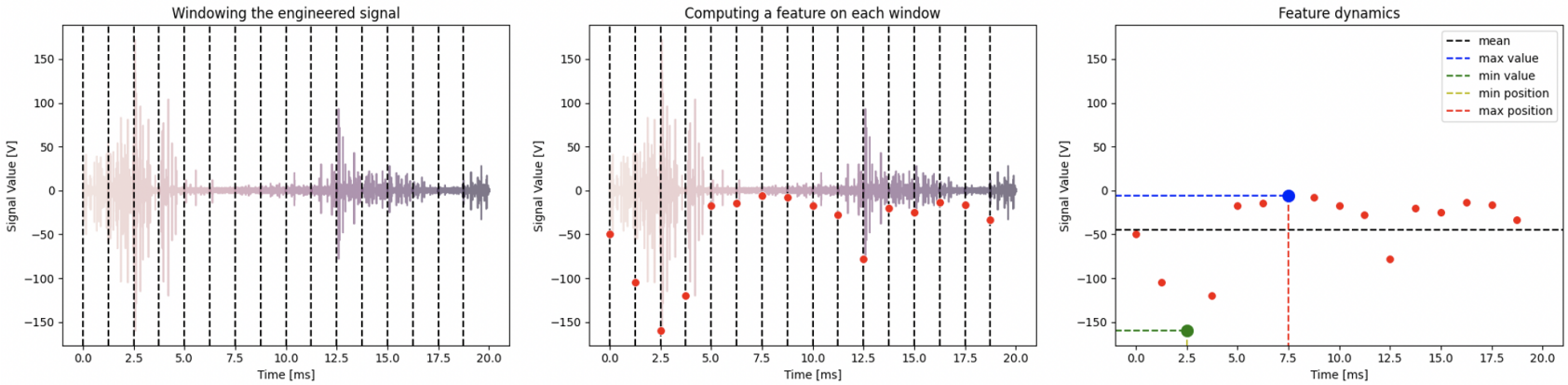
Extracting time-series feature dynamics



Scott Simmons & Louis Jarvis, BHons presentation 2021

Feature Dynamics

Step 2: Extract Feature Dynamics



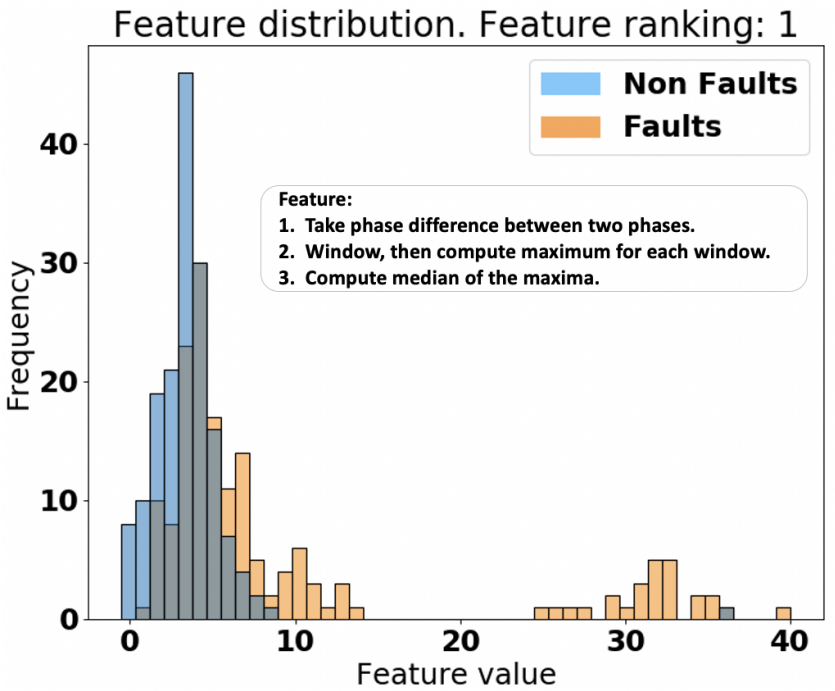
Window input time series → **Compute feature time series** → **Compute feature dynamics**

Simmons et al., "Data Mining on Extremely Long Time-Series"

Example Feature

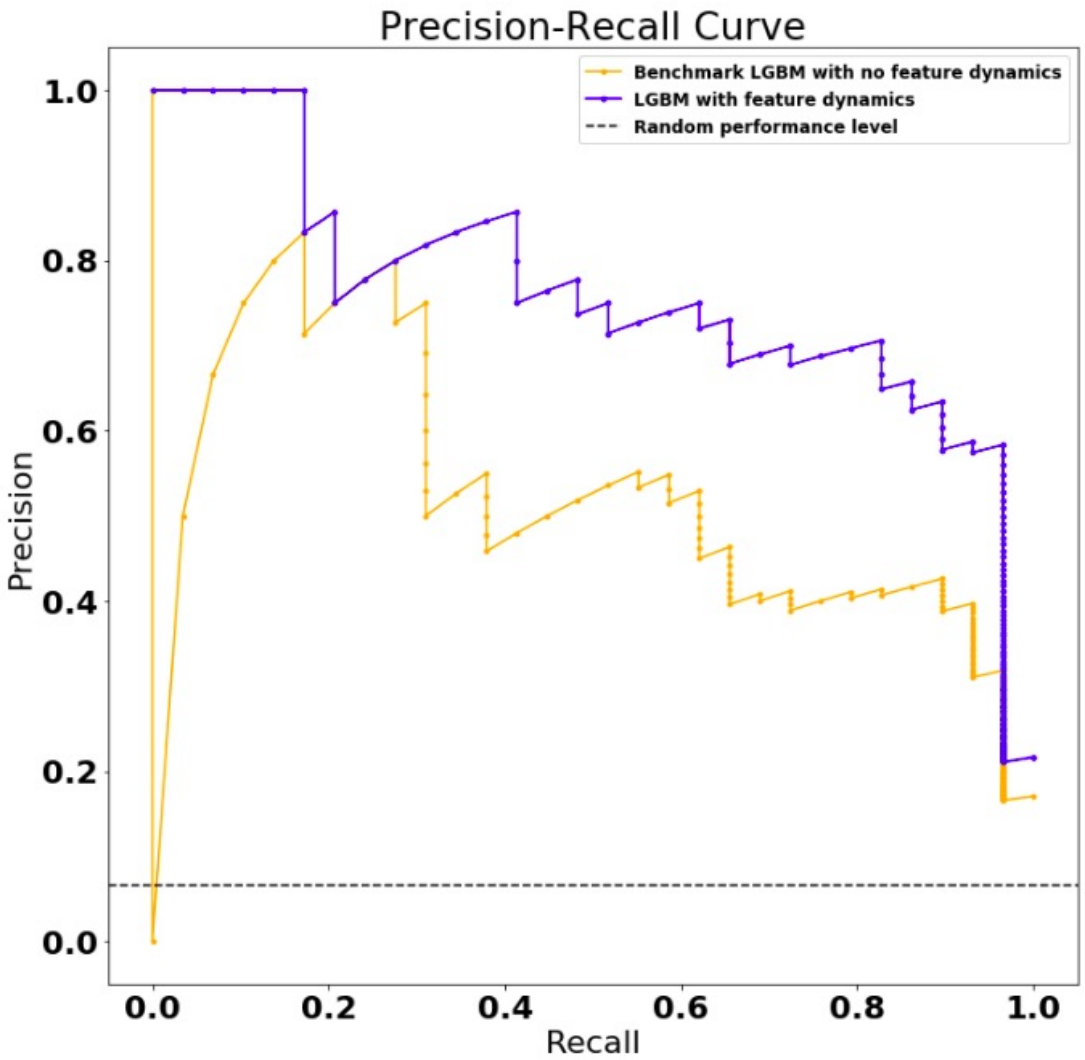
Best Feature Dynamics

- **Small p-values**: 2.51×10^{-32}
- **Interpretable**
- **More informative** than benchmark features.
- **295 Feature Dynamics** after FDR tuning.



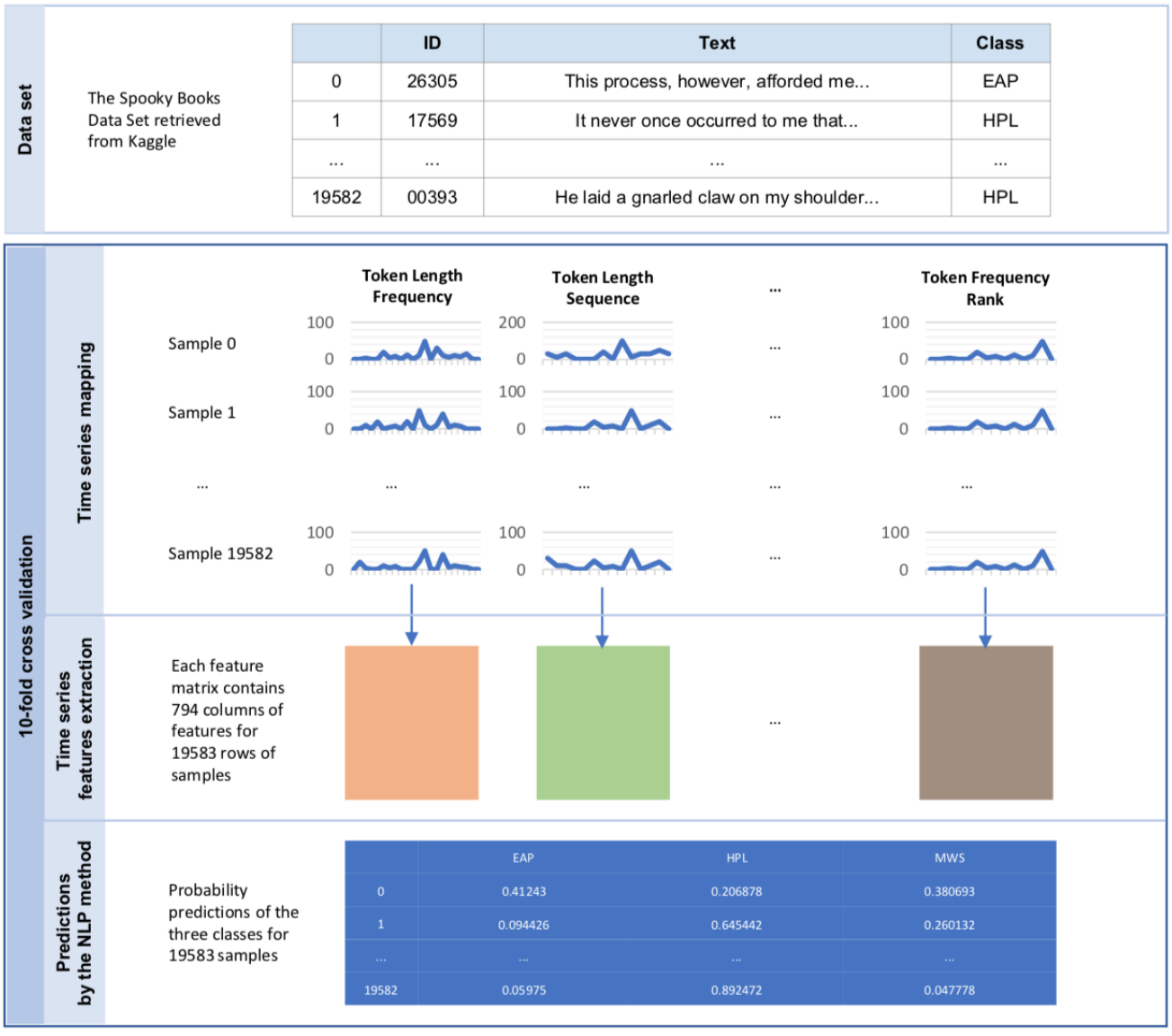
Simmons et al., "Data Mining on Extremely Long Time-Series"

Evaluation



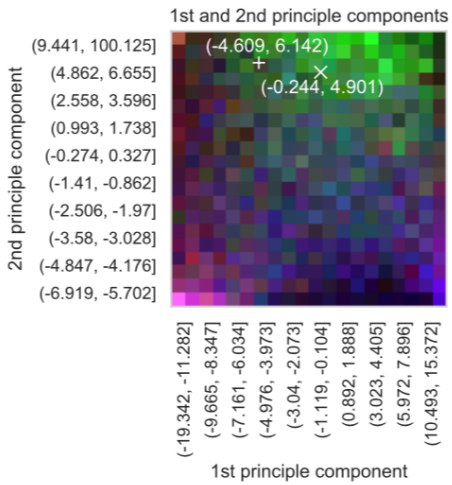
Scott Simmons & Louis Jarvis, BHons presentation 2021

Natural Language Processing – Language Time Series

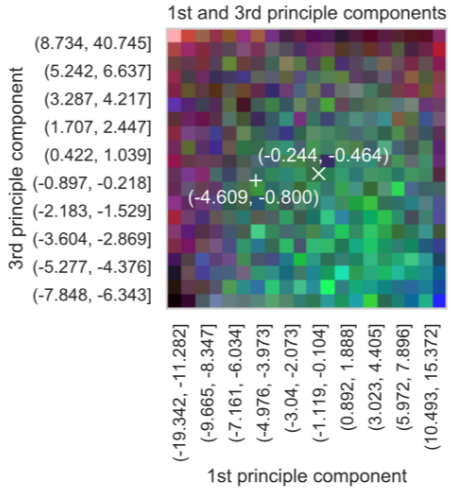


Yichen Tang, Kelly Blincoe, and Andreas W. Kempa-Liehr. "Enriching Feature Engineering for Short Text Samples by Language Time Series Analysis". In: *EPJ Data Science* 9.26 (2020), pp. 1–59. DOI: 10.1140/epjds/s13688-020-00244-9, Fig. 3

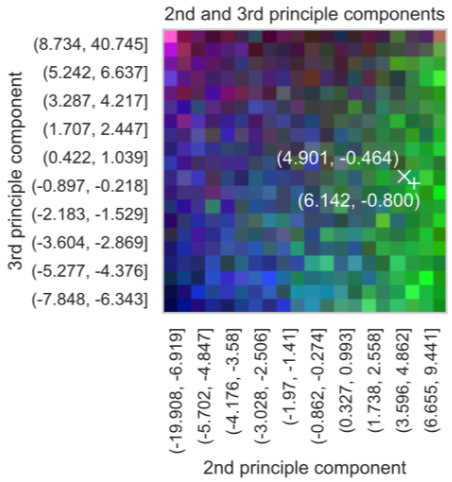
Natural Language Processing – Authorship Attribution



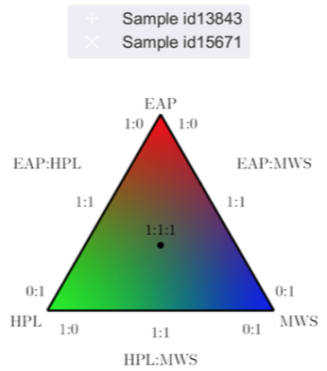
(a) 1st PC against 2nd PC



(b) 1st PC against 3rd PC



(c) 2nd PC against 3rd PC

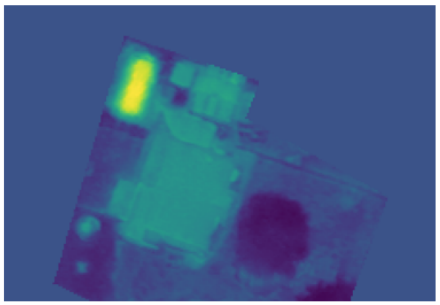


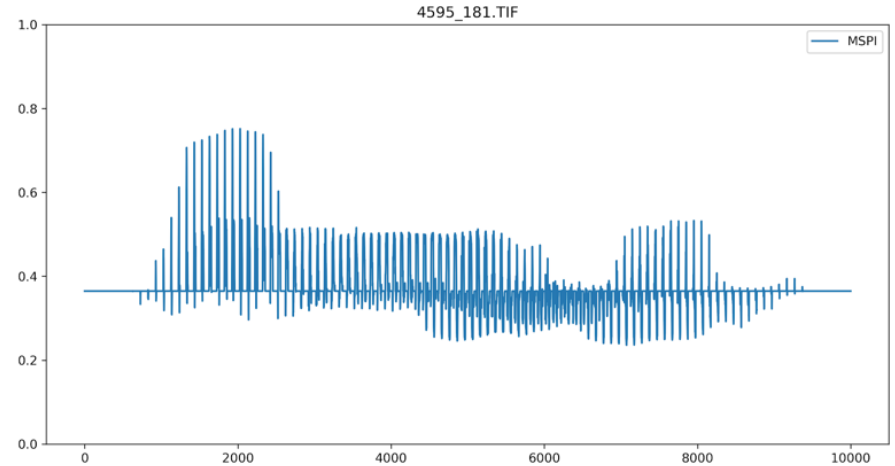
(d) Legends

Tang, Blincoe, and Kempa-Liehr, "Enriching Feature Engineering for Short Text Samples by Language Time Series Analysis", Fig. 2

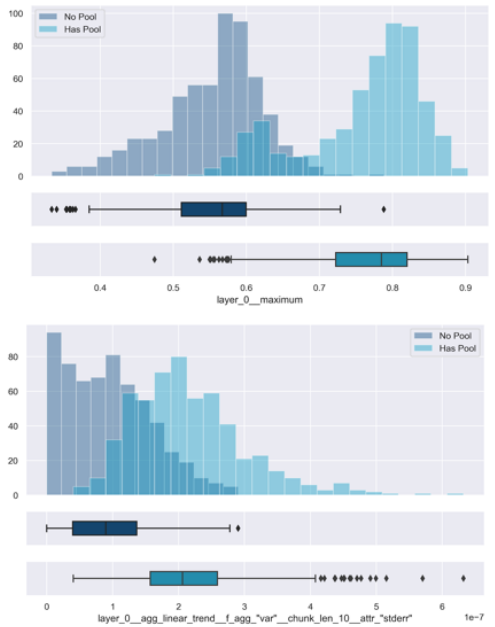
Image Processing – Pool Detection for Gisborne District

F1-score 0.95 (10-fold cross-validation)





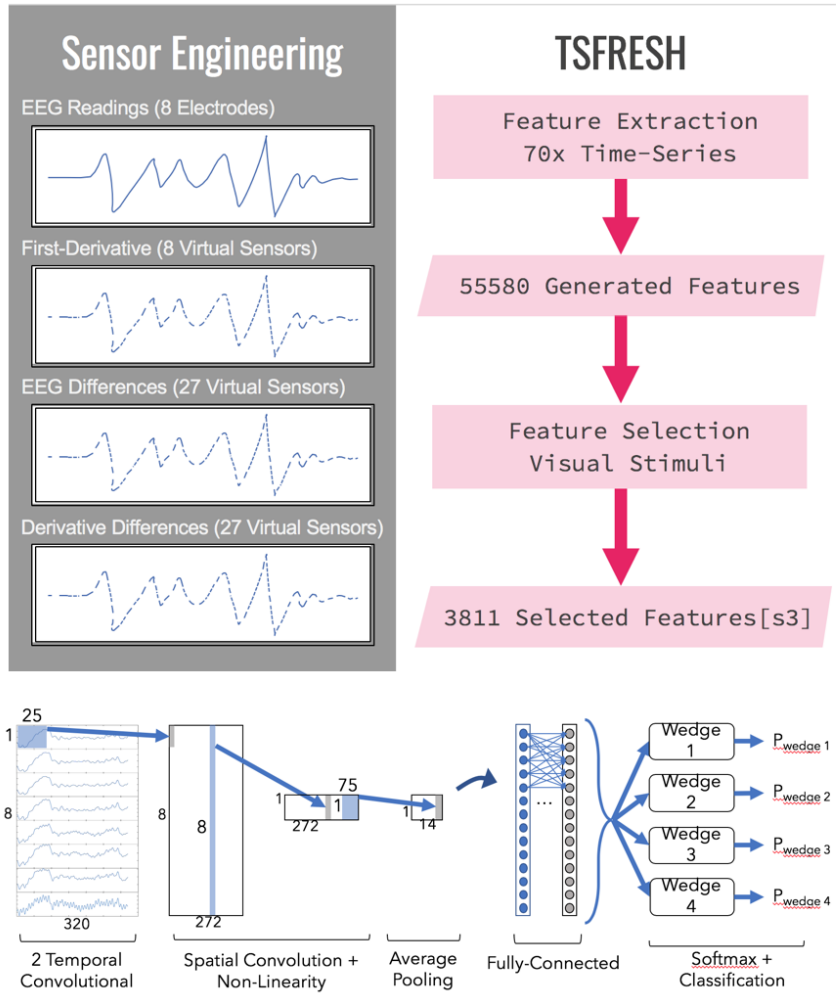
$$MSPI = \frac{B + (255 - NIR)}{2}$$

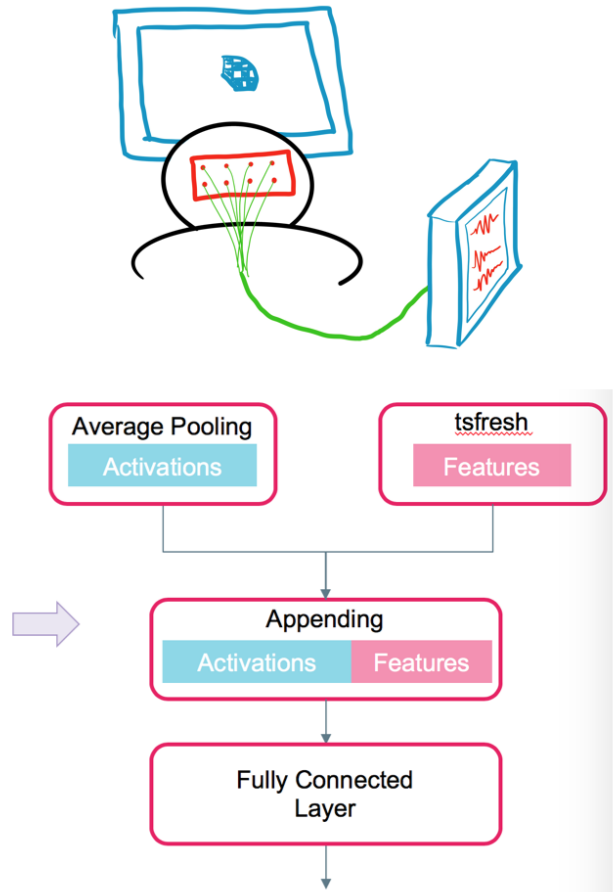


Niklas Pechan. "Machine Learning Approaches to Automated Swimming Pool Detection: Signal Processing Approach". BE(Hons) thesis. University of Auckland: Department of Engineering Science, 2019

EEG - Combining time series features with CNN

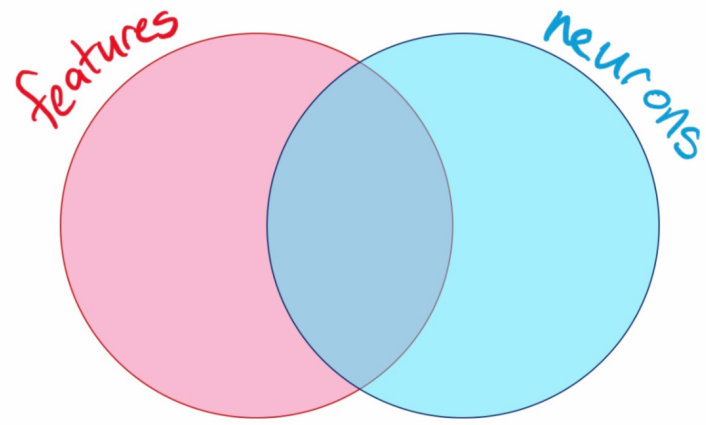
94% accuracy (10 times repeated 10-fold cross-validation)



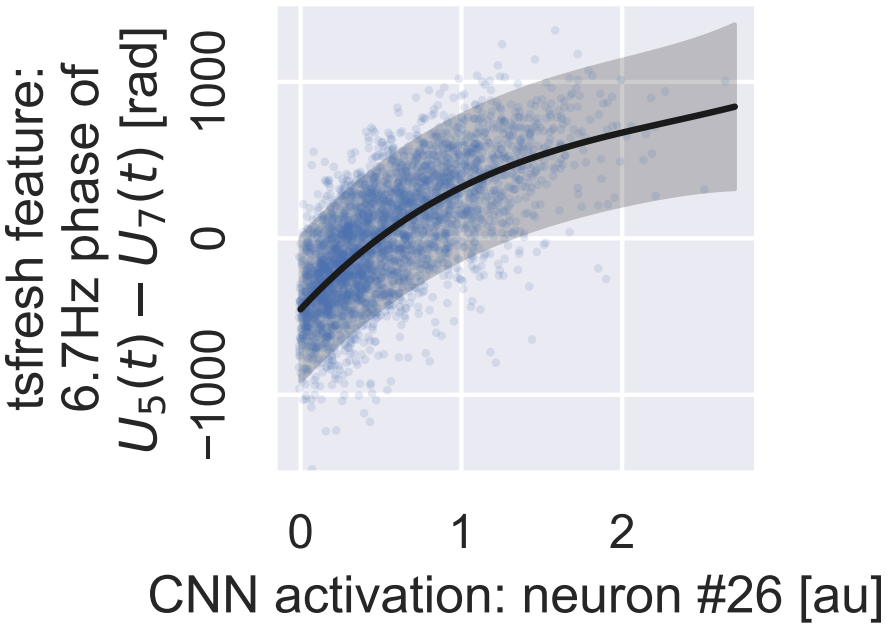


Stephen Orenia and Andrew Tan

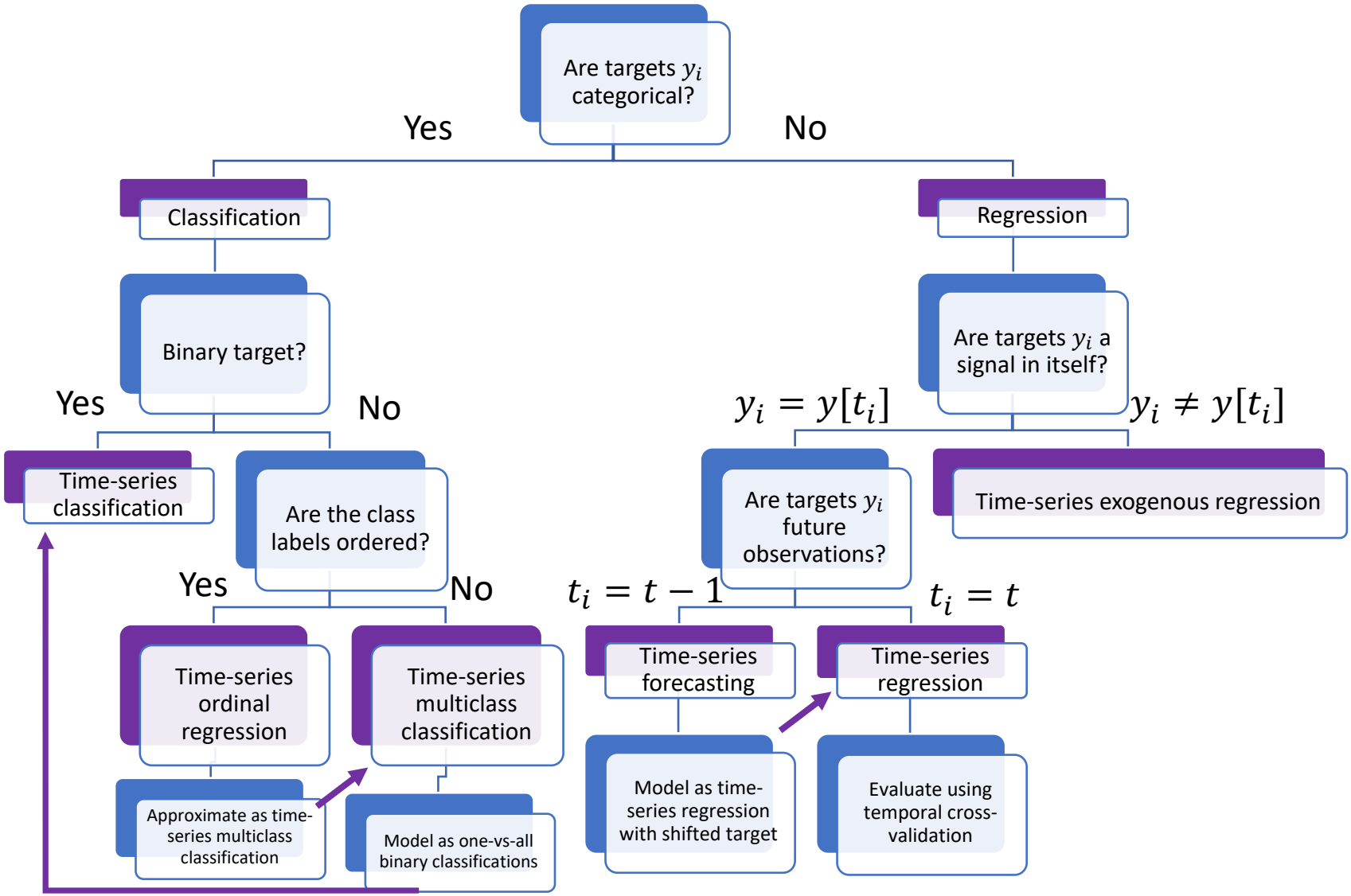
EEG - Interpreting CNNs



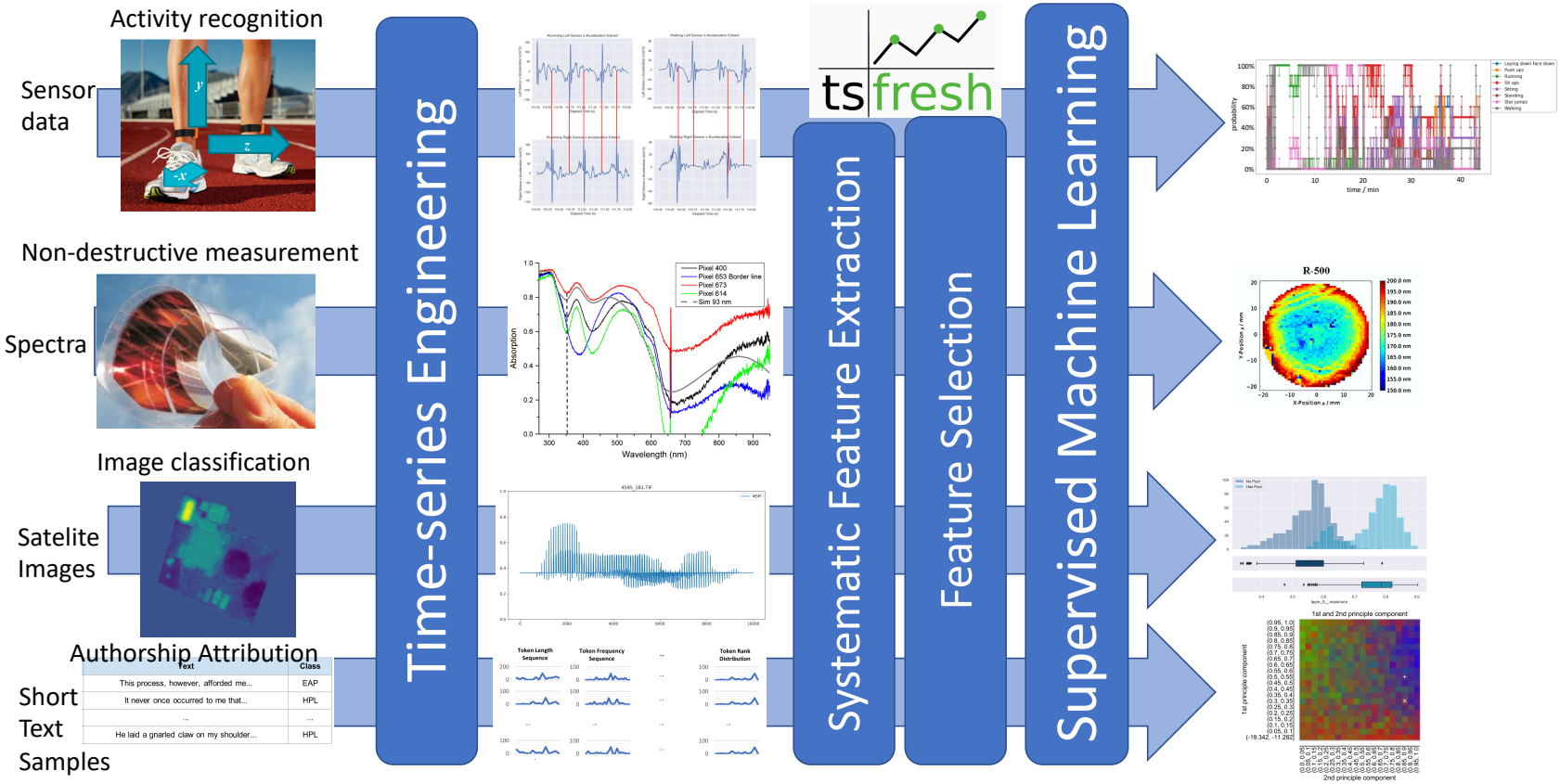
Stephen Lee Orenia. "Step Towards Classification of Electroencephalogram Signals from the Visual Cortex: Time-series Engineering Workflow and Hybrid Classification". BE(Hons) thesis. University of Auckland: Department of Engineering Science, 2019, Fig. 15



Time series analytics



Breadth of tsfresh applications



Summary

Time-series feature engineering

- Machine learning libraries for systematic time series features extraction complement the machine learning toolbox.
- This capability shifts the focus from time series feature engineering to the *engineering of time series*.
- The automation of feature engineering speeds up the experimental validation and operationalization of new sensor systems.

Other applications of systematic time-series feature engineering (UoA)

Engineering Healthcare, predictive maintenance, remote sensing, orthopaedics

Science Volcanic eruption forecasting, detection of exoplanets

Machine learning Authorship attribution