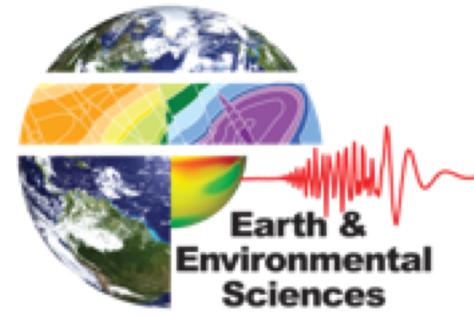


# Feature Engineering & Feature Selection



Maruti Kumar Mudunuru,

Staff Scientist,

Computational Earth Science Group,

Earth and Environmental Sciences,

Los Alamos National Laboratory

Date: April-23-2019, SSA ML Workshop, LA-UR-19-23588.

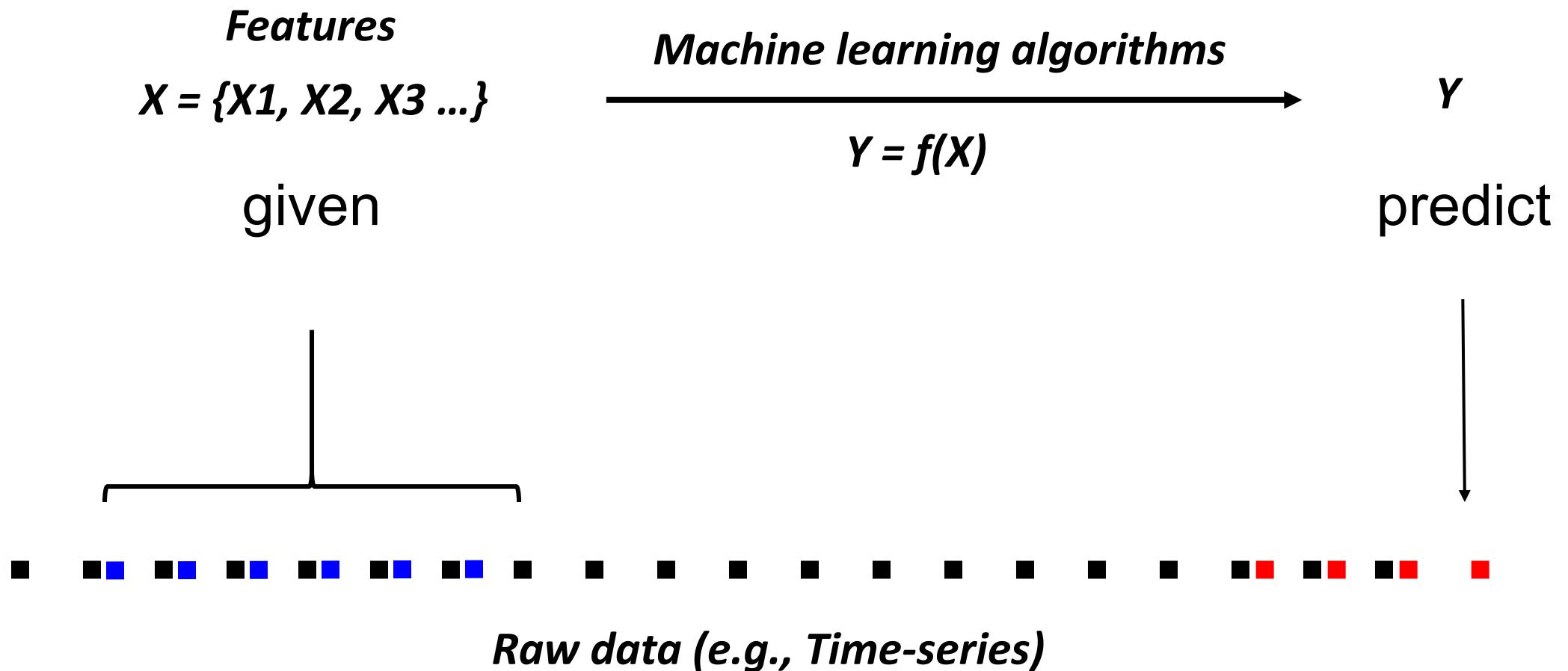
# Outline

- Feature engineering
- Feature extraction
- Feature selection
  - Feature significance testing
  - Multiple test procedure
- Data formats
- Parallelization
- Basics of time-series forecasting

# Basics of feature engineering

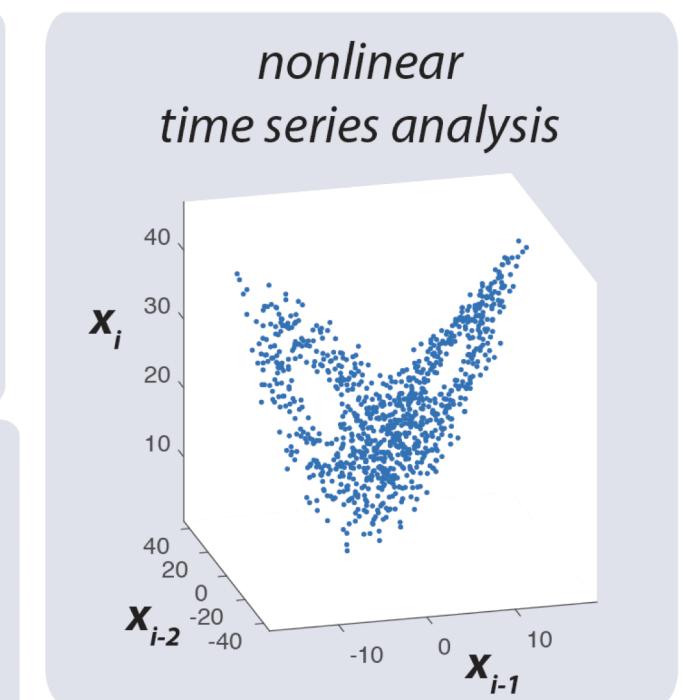
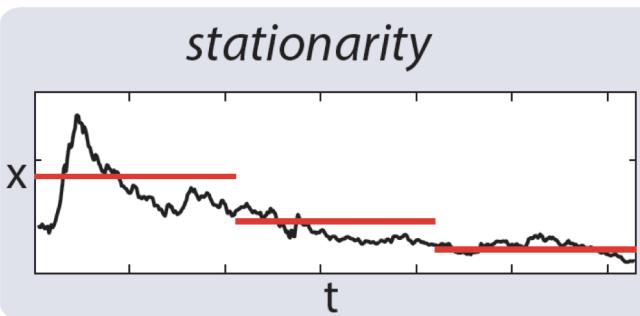
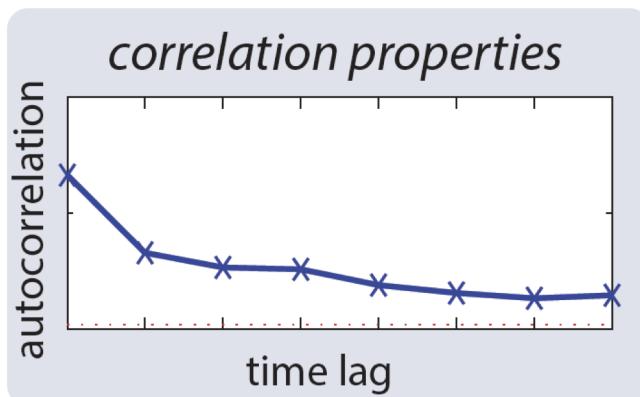
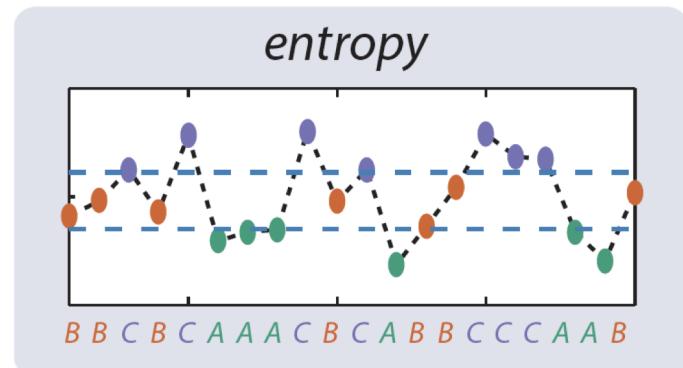
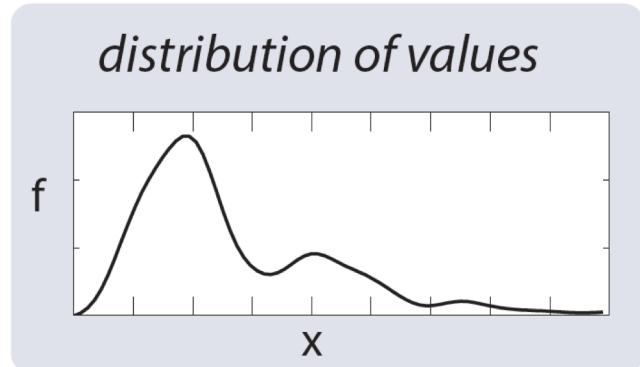
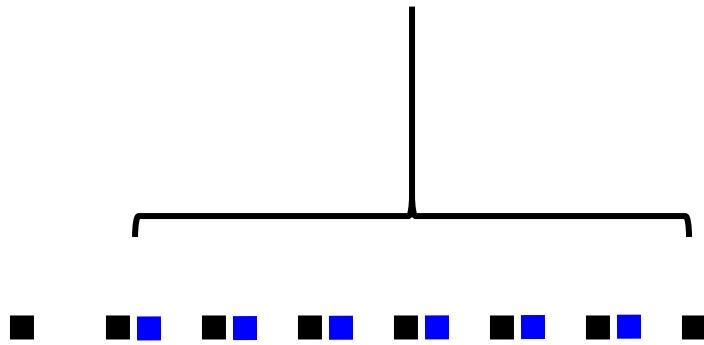
- Feature engineering is the process of transforming **raw data** into **features** that make machine learning algorithms work.
- The goal is to better represent **the underlying problem to the predictive models**.
- **Intelligent** feature engineering results in improved **model accuracy** on **unseen data**.
- Most of the time is spent on engineering or creating features.
  - Deciding what features to create.
  - Checking how the created features work with your ML-model
  - Based on model performance, improving features (if needed)

# Predict future value given current values



# ML-approach: Feature extraction

given



# Engineering or creating features

- A **feature** is an individual, **measurable attribute**, or characteristic of a phenomenon being observed.
- **Features** can be numeric, strings, graphs etc.
- A **feature** is typically a specific representation on top of **raw data**.
- It is depicted by a column in a dataset.
- For example, in a two-dimensional dataset, each *observation* is depicted by a *row* and each *feature* by a *column*
- Note that **features** are **very much dependent** on the underlying problem.

The diagram illustrates a 2D dataset with 5 observations (rows) and 3 features (columns). The columns are labeled X1, X2, and X3. The rows are labeled 0, 1, 2, 3, and 4. A black arrow points downwards from the text "Columns (Features)" to the column labels X1, X2, and X3. A black arrow points to the right from the text "Rows (Observations)" to the row labels 0, 1, 2, 3, and 4.

	Columns (Features)		
	X1	X2	X3
0	45	49	49
1	60	62	63
2	80	82	83
3	80	100	123
4	39	52	43

# Feature Engineering packages (Python, R, and MATLAB)

- hctsa -- MATLAB
- pyopy
- **tsfresh**
  - <https://github.com/blue-yonder/tsfresh>
- tscompdata and tsfeatures -- R
- Khiva
- pyunicorn
- FATS
- seglearn
- cesium-ml
- tslearn

# Advantages of tsfresh

- Field tested
- Unit tested
- The filtering process is statistically/mathematically correct
- Comprehensive documentation
- Compatible with sklearn, pandas and numpy
- Allows anyone to easily add their favorite features
- Runs on your local machine or even on a cluster
- <https://github.com/blue-yonder/tsfresh>

# tsfresh – Time-series feature engineering

- tsfresh is used to extract characteristics from time series
- tsfresh provides a library of functions to calculate features
- Automatic extraction of 100s of features
- If you want to install on your local machine:
  - pip install tsfresh
    - [https://tsfresh.readthedocs.io/en/latest/text/quick\\_start.html](https://tsfresh.readthedocs.io/en/latest/text/quick_start.html)
  - conda install -c conda-forge tsfresh
    - <https://anaconda.org/conda-forge/tsfresh>
- At the top level the following are the three most important submodules of tsfresh:
  - **extract\_features**
  - **select\_features**
  - **extract\_relevant\_features**

# tsfresh – Feature extraction (extract\_features)

- from tsfresh import extract\_features
- X = extract\_features(df, column\_id='id', column\_sort='time')
  - [https://tsfresh.readthedocs.io/en/latest/api/tsfresh.feature\\_extraction.html](https://tsfresh.readthedocs.io/en/latest/api/tsfresh.feature_extraction.html)
- What does it do?
  - Extract features from a [pandas.DataFrame](#) containing the different time series
  - Returns a [pandas.DataFrame](#) with the calculated features will be returned.
  - *default\_fc\_parameters*
    - [MinimalFCParameters](#): includes only a handful of features and can be used for quick tests
    - [EfficientFCParameters](#): features which are marked with the “high\_comp\_cost” are not calculated
    - [ComprehensiveFCParameters](#): all features with parameters, each with different parameter combinations are calculated. This is the default for *extract\_features* if you do not hand in a *default\_fc\_parameters* at all.

```
>>> from tsfresh.examples import load_robot_execution_failures
>>> from tsfresh import extract_features
>>> df, _ = load_robot_execution_failures()
>>> X = extract_features(df, column_id='id', column_sort='time')
```

# tsfresh – Feature extraction settings (extract\_features)

- MinimalFCParameters

```
>>> from tsfresh.feature_extraction import extract_features, MinimalFCParameters  
>>> extract_features(df, default_fc_parameters=MinimalFCParameters())
```

- EfficientFCParameters

```
>>> from tsfresh.feature_extraction import extract_features, EfficientFCParameters  
>>> extract_features(df, default_fc_parameters=EfficientFCParameters())
```

- ComprehensiveFCParameters

```
>>> from tsfresh.feature_extraction import extract_features, ComprehensiveFCParameters  
>>> extract_features(df, default_fc_parameters=ComprehensiveFCParameters())
```

# tsfresh – Feature calculators

- Tsfresh module contains the feature calculators that take time-series as input and calculate the values of the feature.
- There are two types of features:
  - **Scalar-valued:** feature calculators which calculate a single number (simple)
  - **Vector-valued:** feature calculators which calculate a bunch of features for a list of parameters at once (combiner). They return a list of (key, value) pairs for each input parameter
- Examples of **scalar-valued** features:
  - Absolute energy, absolute sum of changes, autocorrelation, binned entropy, standard deviation, skewness, kurtosis, mean etc.
- Examples of **vector-valued** features:
  - Aggregated auto correlation, autoregressive coefficients, CWT coefficients, FFT coefficient, ARIMA coefficients, PSD using Welch's method
- [https://tsfresh.readthedocs.io/en/latest/text/list\\_of\\_features.html](https://tsfresh.readthedocs.io/en/latest/text/list_of_features.html)

# tsfresh – Feature extraction (extract\_relevant\_features)

- from tsfresh import extract\_relevant\_features
- X = extract\_relevant\_features(df, y, column\_id='id', column\_sort='time')
  - <https://tsfresh.readthedocs.io/en/latest/api/tsfresh.convenience.html>
- What does it do?
  - **High level convenience function** to extract time series features from *timeseries\_container*
  - *timeseries\_container* – The pandas.DataFrame with the time series to compute the features for, or a dictionary of pandas.DataFrames
  - ‘df’ is an example of *timeseries\_container*
  - It returns feature matrix ‘X’ possibly augmented with relevant time-series features with respect to target vector ‘y’.

```
>>> from tsfresh.examples import load_robot_execution_failures
>>> from tsfresh import extract_relevant_features
>>> df, y = load_robot_execution_failures()
>>> X = extract_relevant_features(df, y, column_id='id', column_sort='time')
```

# tsfresh – Feature selection basics (select\_features)

- Time series often contain noise, redundancies or irrelevant information.
- As a result most of the extracted features will not be useful for the machine learning task at hand.
- To avoid extracting irrelevant features, the *tsfresh* package has a built-in filtering procedure.
- This filtering procedure evaluates the explaining power and importance of each characteristic for the regression or classification tasks at hand.
- It is based on the well developed theory of hypothesis testing and uses a multiple test procedure.
- As a result the filtering process mathematically controls the percentage of irrelevant extracted features.
- **from tsfresh import select\_features**

```
>>> from tsfresh import extract_features, select_features  
>>> X_extracted = extract_features(df, column_id='id', column_sort='time')  
>>> X_selected = select_features(X_extracted, y)
```

# tsfresh – Feature selection (null hypothesis)

- Each feature vector is evaluated individually and independently based on its importance for predicting the target or class label.
- We are testing
  - $H_0$  = the Feature is not relevant and should not be added
- against the following
  - $H_1$  = the Feature is relevant and should be kept
- Or in other words
  - $H_0$  = Target and Feature are independent / the Feature has no influence on the target
  - $H_1$  = Target and Feature are associated / dependent

# tsfresh – Feature selection (p-value)

- When you perform a hypothesis test in statistics, a **p-value** helps you determine the significance of your results.
- The **p-value** is a number between 0 and 1 and interpreted in the following way:
  - A **small p-value** (typically  $\leq 0.05$ ) indicates strong evidence against the null hypothesis that the feature is not relevant and should not be added
  - Meaning that, we reject the null hypothesis and the feature should be kept.
  - A **large p-value** ( $>0.05$ ) indicates weak evidence against the null hypothesis, so we fail to reject the null hypothesis.
- Always report the **p-value** so your readers can draw their own conclusions.

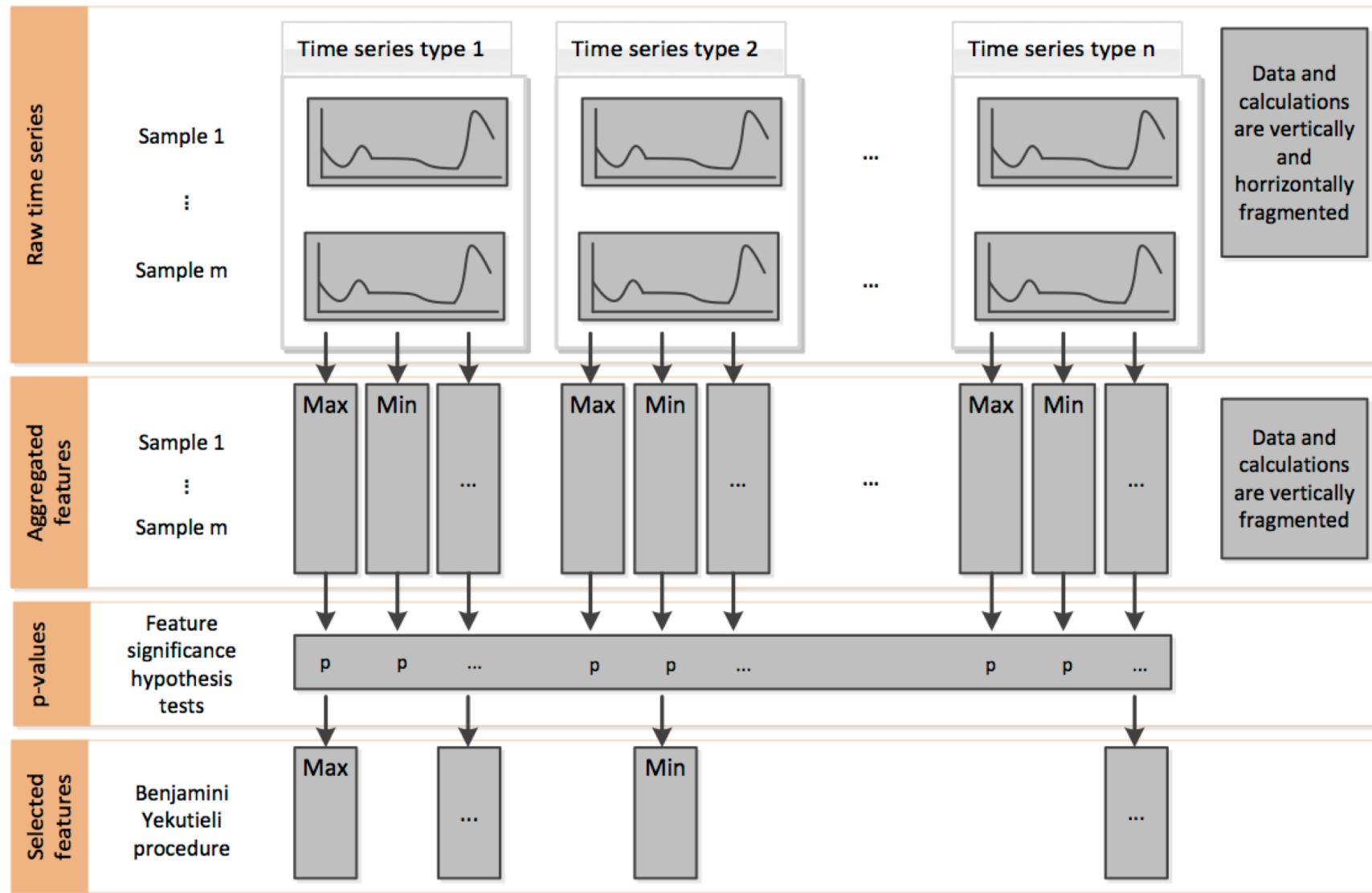
# tsfresh – Feature selection (testing procedure)

- **Benjamini Hochberg procedure**

- It is a multiple testing procedure decides which features to keep and which to cut off (solely based on the p-values)
- Determines if the null hypothesis for a given feature can be rejected.
- For this the test regards the features' p-values and controls the global false discovery rate
- Global false discovery rate (FDR) is the ratio of false rejections by all rejections:

$$FDR = \mathbb{E} \left[ \frac{|\text{false rejections}|}{|\text{all rejections}|} \right]$$

# tsfresh – Feature selection summary



# tsfresh – Data Formats

- Input format – pandas.DataFrame
  - <http://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html#pandas.DataFrame>
  - flat DataFrame
  - Stacked DataFrame
  - Dictionary of flat DataFrames
  - [https://tsfresh.readthedocs.io/en/latest/text/data\\_formats.html](https://tsfresh.readthedocs.io/en/latest/text/data_formats.html)

# tsfresh – Data Formats (Flat DataFrame Example)

- Input Data Format:

- Imagine you record the values of time-series x and y for different objects A and B for three different times t1, t2 and t3.
- Now you want to calculate some feature with tsfresh.
- Your resulting DataFrame may look like this
- You would pass
- `column_id="id", column_sort="time", column_kind=None, column_value=None`
- to extract functions, to extract features separately for all ids and
- separately for the x and y values.

id	time	x	y
A	t1	x(A, t1)	y(A, t1)
A	t2	x(A, t2)	y(A, t2)
A	t3	x(A, t3)	y(A, t3)
B	t1	x(B, t1)	y(B, t1)
B	t2	x(B, t2)	y(B, t2)
B	t3	x(B, t3)	y(B, t3)

- Output Format of Features:

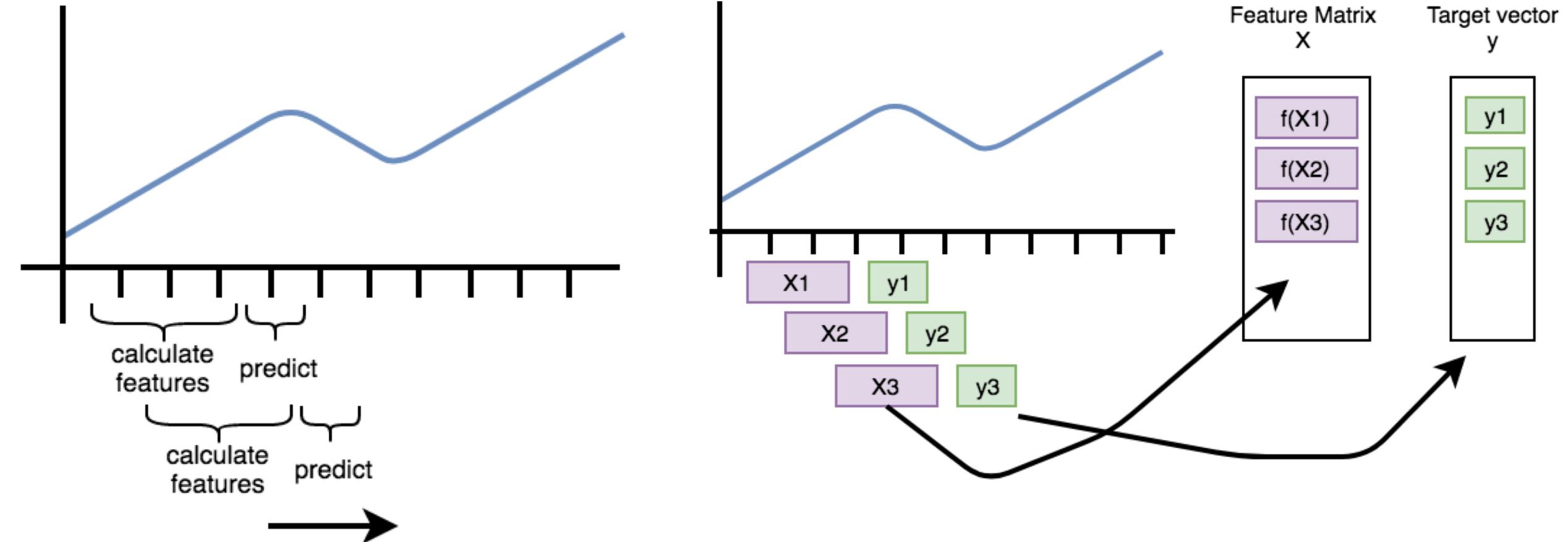
- The resulting feature matrix for all three input DataFrame options will be the same.
- It will always be a [pandas.DataFrame](#) with the following layout

id	x_feature_1	...	x_feature_N	y_feature_1	...	y_feature_N
A	...	...	...	...	...	...
B	...	...	...	...	...	...

# tsfresh – Parallelization

- Parallelization is based on multiprocessing.Pool
- For the **feature extraction** and ***feature selection*** parallelization parameters include
  - *n\_jobs* and *chunksize* in *extract\_features* and *select\_features*
  - *n\_jobs* corresponds to number of processors on the current system
- On instantiation we set the Pool's number of worker processes to *n\_jobs*
- Recommend setting it to the maximum number of available (and otherwise idle) processors
- *chunksize* is the number of chunks that are submitted as one task to one worker process
- E.g, if you set the chunksize to 10, then it means that one worker task corresponds to calculate all features for 10 id/kind time series combinations
- To do performance studies and profiling, turn off parallelization.
- This can be setting the parameter *n\_jobs* to 0 and *chunksize = None*.

# tsfresh – Basics of time-series forecasting



# tsfresh – Summary of time-series forecasting

- Features are calculated by *rolling* over the data
- A rolling mechanism will give you the sub time series to construct the features.
- So, we move the time-series window that extract the features and then predict the next time step (which was not used to extract features) forward
- Both tsfresh and pandas provide methods for rolling mechanisms
  - `tsfresh.utilities.dataframe_functions.roll_time_series`
    - [https://tsfresh.readthedocs.io/en/latest/api/tsfresh.utilities.html#tsfresh.utilities.dataframe\\_functions.roll\\_time\\_series](https://tsfresh.readthedocs.io/en/latest/api/tsfresh.utilities.html#tsfresh.utilities.dataframe_functions.roll_time_series)
  - `DataFrame.rolling`
    - <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.rolling.html>

# Conclusions – Things to remember

- Feature scaling is done to Normalize data so that priority is not given to a particular feature.
  - <https://scikit-learn.org/stable/modules/preprocessing.html>
- Role of scaling is mostly important in algorithms that are distance based (e.g, Euclidean metric).
  - E.g., k-means clustering, Support Vector Machines, etc.
- ASIDE -- Random Forest (RF) is a tree-based model and hence **does not require** feature scaling.
- RF algorithm is based on partitioning, even if you apply Normalization then also the result would be the same.

# References & Useful Resources

- <https://github.com/benfulcher/hctsa>
- <https://tsfresh.readthedocs.io/en/latest/>
- <https://github.com/blue-yonder/tsfresh>
- <https://towardsdatascience.com/how-not-to-use-machine-learning-for-time-series-forecasting-avoiding-the-pitfalls-19f9d7adf424>
- <https://towardsdatascience.com/how-to-use-machine-learning-for-anomaly-detection-and-condition-monitoring-6742f82900d7>
- <https://towardsdatascience.com/understanding-feature-engineering-part-1-continuous-numeric-data-da4e47099a7b>
- <https://towardsdatascience.com/feature-engineering-what-powers-machine-learning-93ab191bcc2d>
- <https://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/>
- <https://developers.google.com/machine-learning/crash-course/representation/feature-engineering>
- Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists – Alice Zheng & Amanda Casari, O'Reilly, 2018.
- Feature Engineering Made Easy, Sinan Ozdemir, Divya Susarla, Packt, 2018.
- Feature Extraction: Foundations and Applications, by Isabelle Guyon, Steve Gunn, Masoud Nikravesh, Lofti A. Zadeh, Springer 2006.
- Feature Extraction, Construction and Selection: A Data Mining Perspective, Huan Liu, Hiroshi Motoda, Springer, 1998.

# Questions?