sktime: What, Why & Welcome

sktime tutorial

January 2020

UCL, London, UK

Markus Löning & Franz Király

Thank you to all our contributors

- API design: Anthony Bagnall, Sajaysurya Ganesh, Viktor Kazakov, Franz Király, Jason Lines, Markus Löning, Patrick Rockenschaub, Mathew Smith, @big-o
- Coding: Anthony Bagnall, Sajaysurya Ganesh, Viktor Kazakov, Jason Lines, Markus Löning, Patrick Rockenschaub, Amaia Abanda Elustondo, Aaron Bostrom, Saurabh Dasgupta, David Guijo-Rubio, James Large, Matthew Middlehurst, George Oastler, Piotr Oleskiewicz, Mathew Smith, and Jeremy Sellier, Patrick Rockenschaub, Angus Dempster, Thach Le Nguyen, Claudia Sanches, @abdkg, @Kludex, @simone-pignotti

Learning objectives

- Become familiar with ML time series setting
- Understand the need for toolboxes and sktime in particular
- Find out how to contribute
- Play around with sktime

Agenda

1. Introduction to sktime

2. Interactive tutorial & discussion

Intro to sktime

AI for medical data

„Typical“ Nature/Science paper on AI for medicine

		outcome *(binary)*	**time stamp** *(date)*	**lab value** *(continuous)*
Patient ID *(unique ID)*	***1***	*cured*	*Nov 4, 2019*	*100*
	1	*cured*	*Jan 12, 2019*	*120*
	2	*died*	*Aug 18, 2017*	*42*

„There were 10 observations per day, for over a year, of 30 patients,
resulting in a BIG DATA set with 120.000 samples
on which gradient boosting was trained to predict the outcome
resulting in 92% accuracy on a hold-out test set of 40.000 samples“

(confidence intervals are negligible due to large size of dataset)

Question: *are there any (technical) problems you can spot?*

… are there really 120.000 samples?

… is it a problem if we feed this table to sklearn?

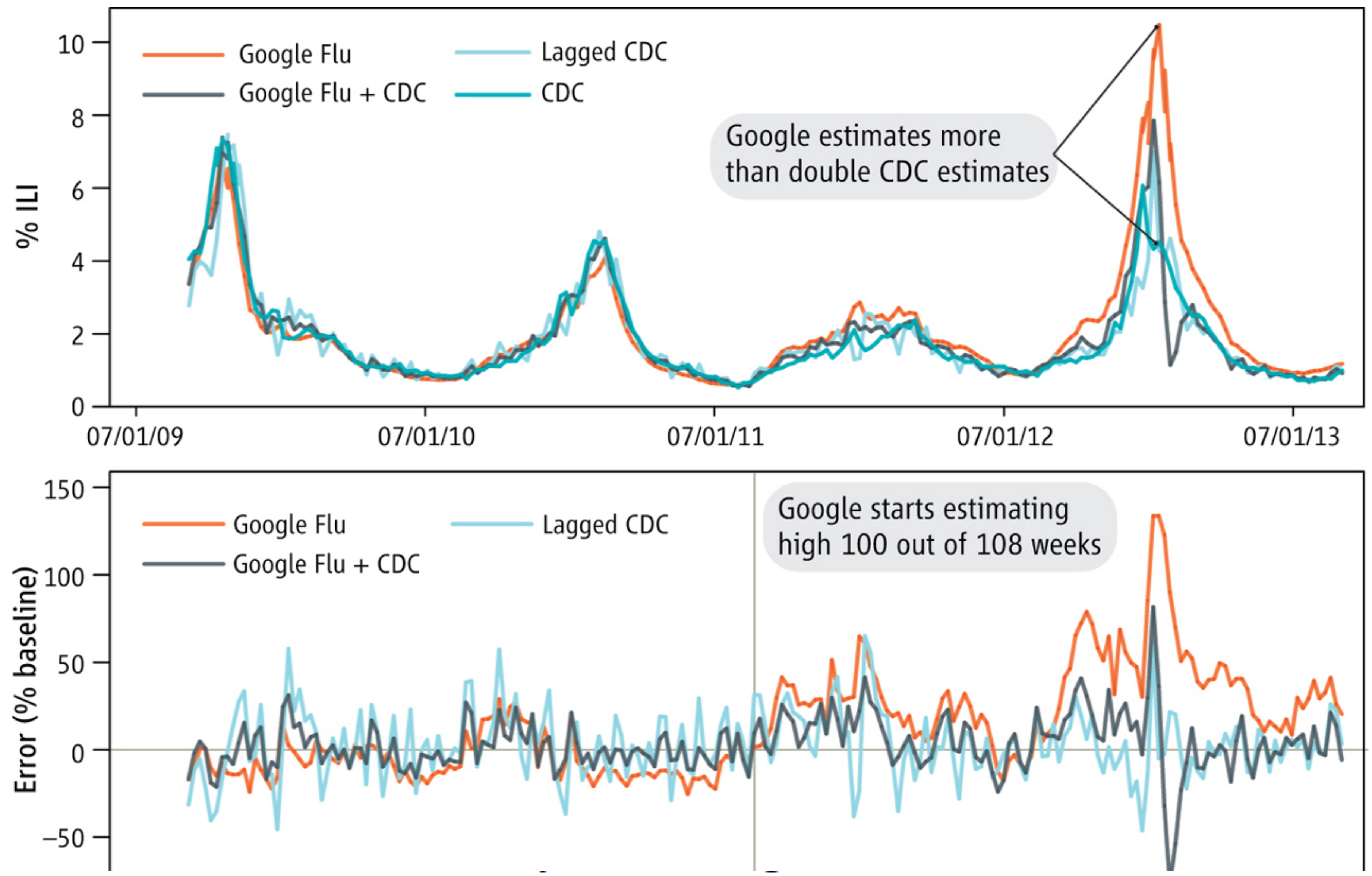
BIG DATA

The Parable of Google Flu: Traps in Big Data Analysis

Large errors in flu prediction were largely avoidable, which offers lessons for the use of big data.

David Lazer,[1,2*] Ryan Kennedy,[1,3,4] Gary King,[3] Alessandro Vespignani[5,6,3]

In February 2013, Google Flu Trends (GFT) made headlines but not for a reason that Google executives or the creators of the flu tracking system would have hoped. *Nature* reported that GFT was predicting more than double the proportion of doctor visits for influ-



model:

GLM on aggregate tabular extract of search histories, with lagged features

performs poorly in regions 6 and 10. Nevertheless, in almost every case, GFT is outperformed by the basic time series predictions and the combined model. Although not discussed in the Policy

But can't I still use sklearn?

for example, I could convert to aggregated long format

Patient ID (unique ID)	outcome (binary)	time stamp (date)	lab value (continuous)
1	cured	Nov 4, 2019	100
1	cured	Jan 12, 2019	120
2	died	Aug 18, 2017	42

→

Patient ID (unique ID)	outcome (binary)	Lab 2017-01 (date)	Lab 2017-02
1	cured	100	N/A
2	cured	N/A	N/A
3	died	50	55

true, it creates a feature table with lots of columns

true, it creats a table in which a lot of entries are missing

true, there are a number of modelling choices I have to make

such as aggregation bin width; aggregation mode; aggregation periods

the choice what I do with the NAs that I created

the choice whether I want calendar date or a date offset defining the column

But then I can use sklearn or keras, which really is the main thing

… can I?

after all, it‘s the hammers I have…

modelling choices made outside sklearn aren't „real“

George Box, 1976:

(Science and Statistics)

The maladies which result may be called *cookbookery* and *mathematistry*. The symptoms of the former are a tendency to force all problems into the molds of one or two routine techniques, insufficient thought being given to the real objectives of the investigation or to the relevance of the assumptions implied by the imposed methods.

State your purpose

crucial to carefully state „what one wants to do“

appropriateness of methods and workflows depend on this

Relational data model: what is the semantic data/index format?

e.g., instance/time hierarchy in the patients example

e.g., instance/time/space hierarchy in the flu example

Statistical data model: which (in)dependence relations exist?

What statistical sampling assumptions are reasonable?

e.g., independence across instances; dependence across time

Modelling goal: what to predict? Paramter inference? Causality?

Hidden „cookbookery“: always assuming supervised prediction

Success control: how do we know a solution is „good“?

appropriate choice of evaluation/assessment points and workflow

These choices must inform toolbox usage and interfaces!

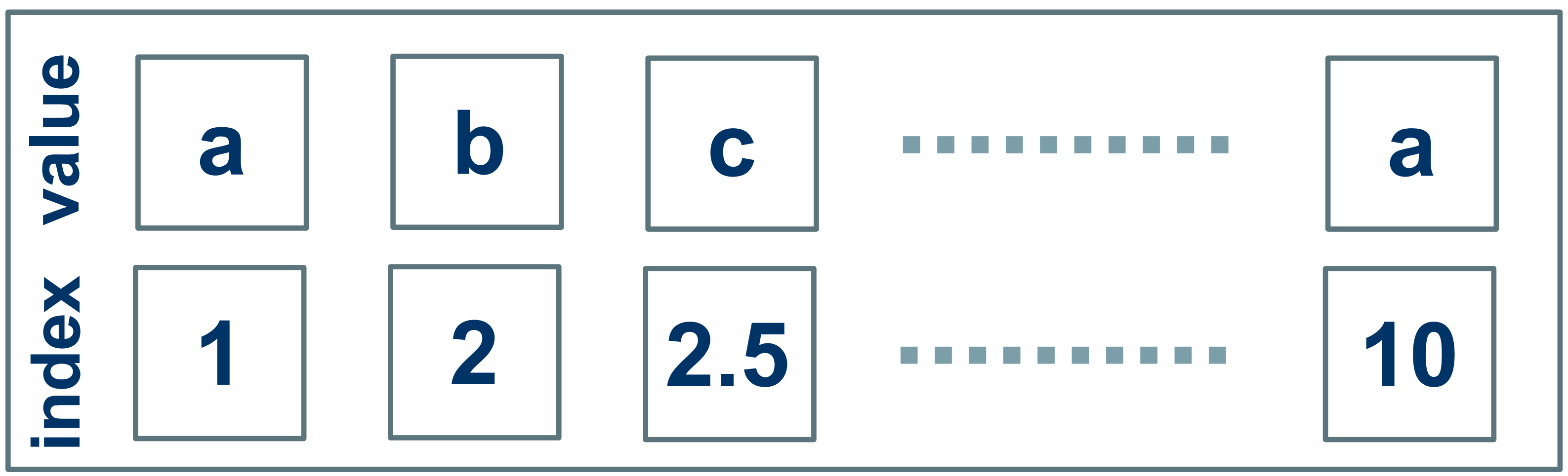
Data models for temporal data

„sequence“: type-homogenous ordered tuple

`pandas.series`

(index-free)

„time series“: type-hom., time-indexed, ordered tuple



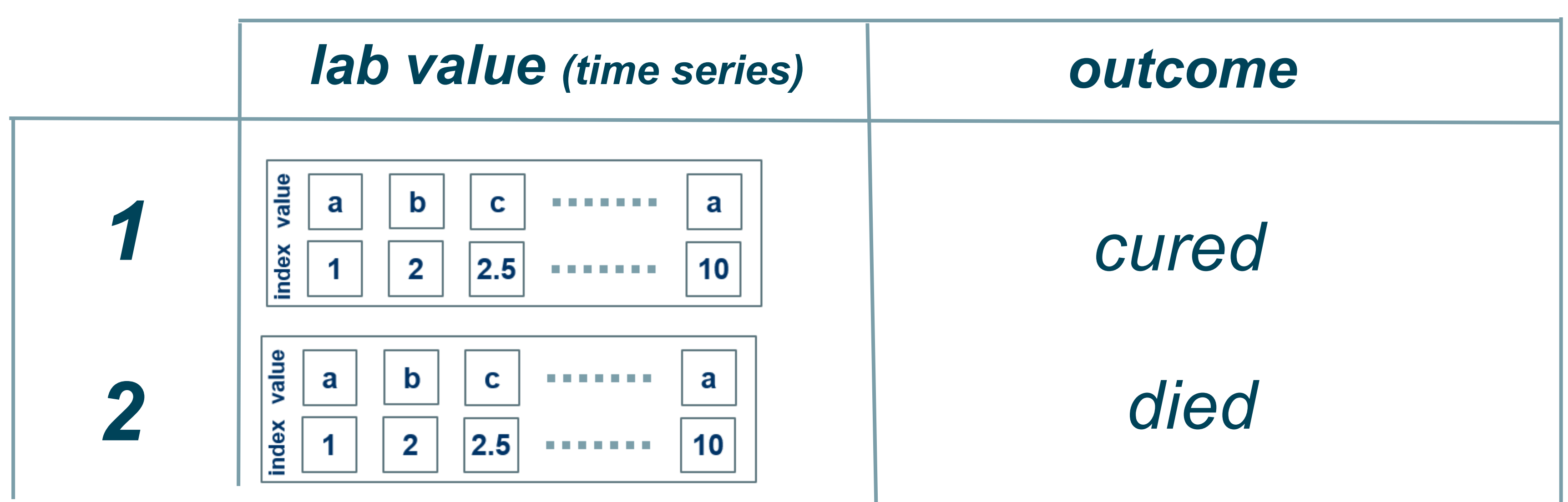
`pandas.series`

with time index

`DataFrame`

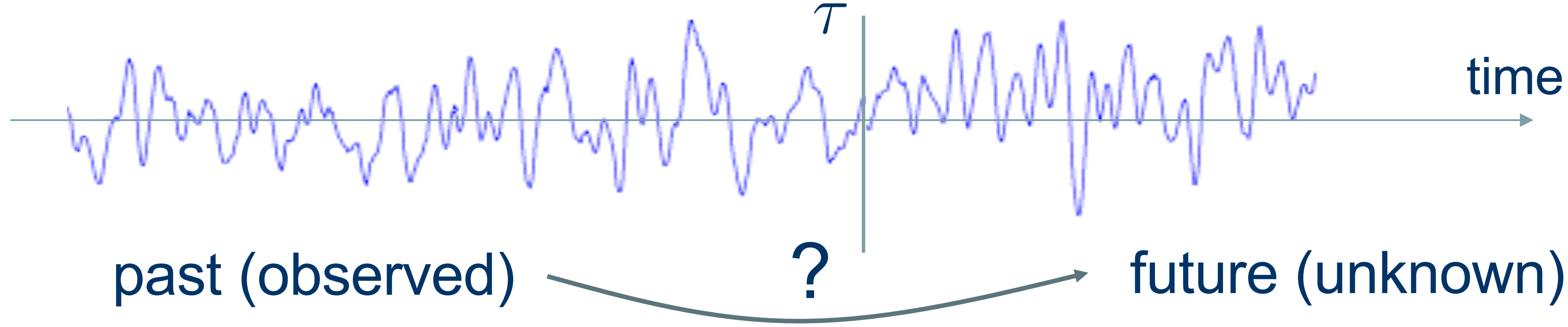
„panel data“:

Data frame w. series cols



Key temporal statistical data models

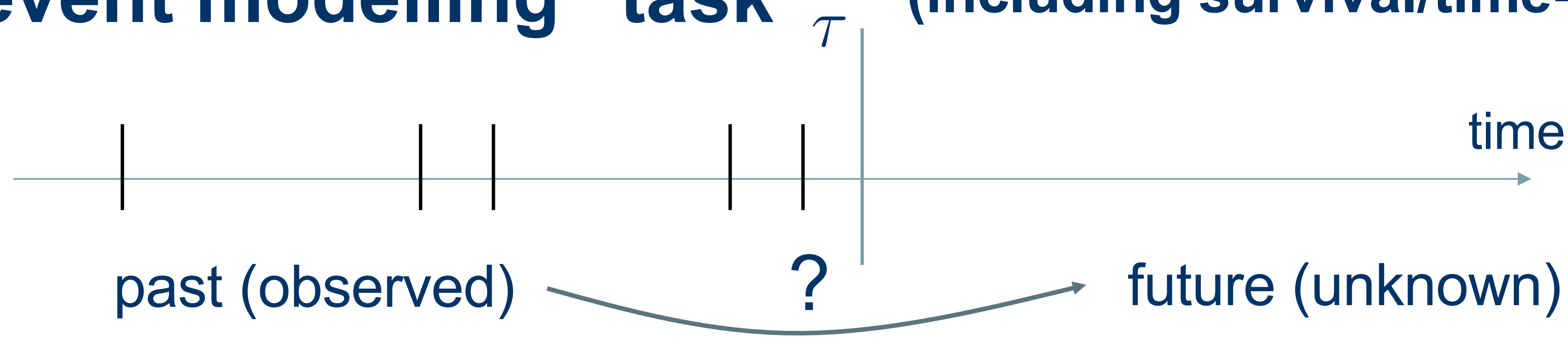
Classical „forecasting“ task



Common statistical model: $X = (X_t\,; t \in \mathcal{T})$ with $\mathcal{T} \subseteq \mathbb{R}$

predict $X_{>\tau} := (X_t\,;\, t \in \mathcal{T}, t > \tau)$ given $X_{\leq\tau} := (X_t\,;\, t \in \mathcal{T}, t \leq \tau)$

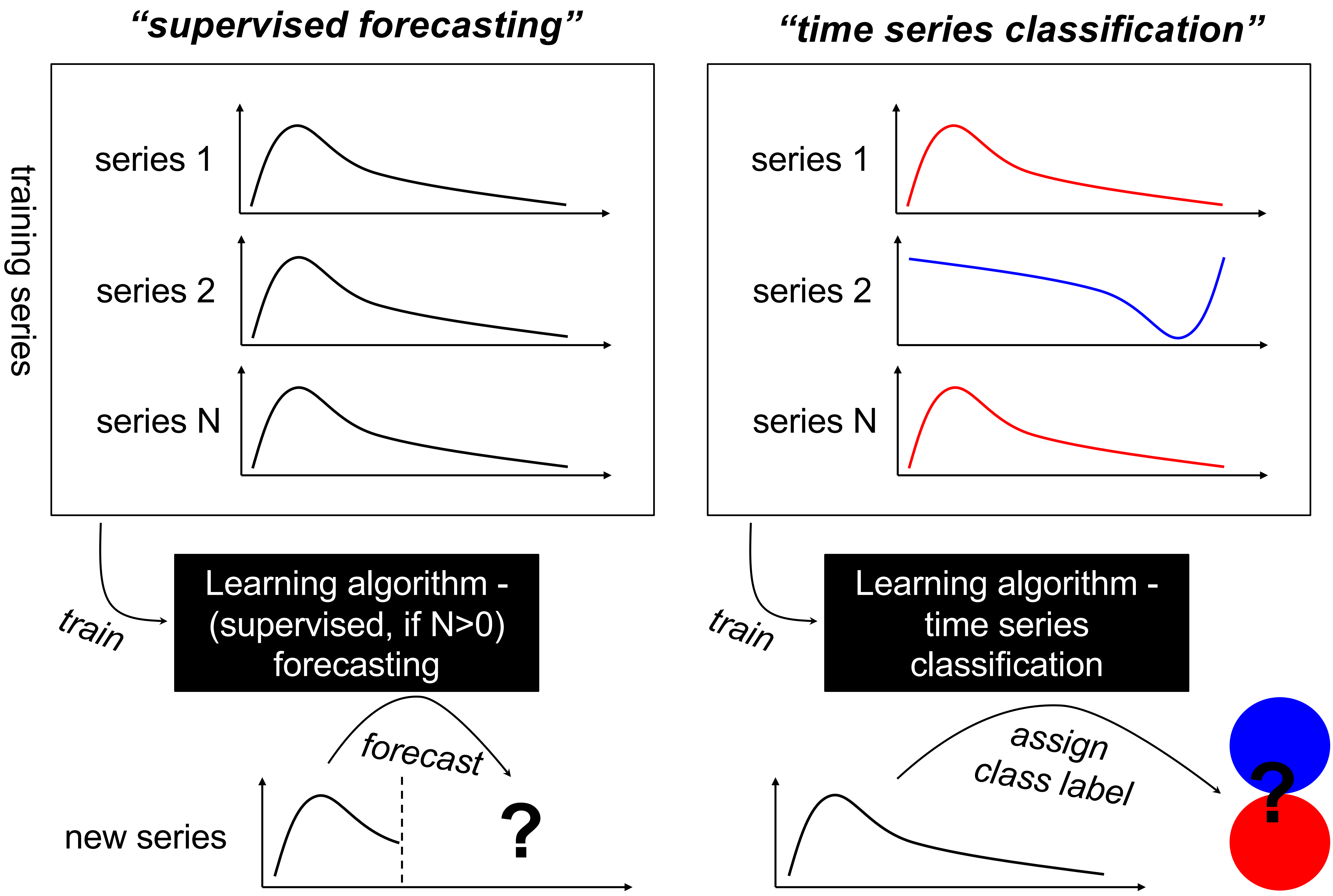
„event modelling“ task (including survival/time-to-event)



Common statistical model: $X = (T_1, \dots, T_N)$ with T_i r.v.in $\mathbb{R}$, r.v. N t.v.in $\mathbb{N}$

predict/state *generative model of* $X_{>\tau}$ given $X_{\leq\tau} := (T_i,$ s.t. $T_i \leq \tau)$

Panel data related tasks



The i.i.d. assumption divide

*Crucial distinction: are samples **statistically independent**?*

(A) i.i.d. (panel) samples are available

series-as-features, or panel modelling tasks

It is *crucial* to make use of the i.i.d.-ness assumption!

models also trained on other time series will be better

„independent samples of time series“, panel tasks

(B) no i.i.d. assumption can be made

in essence: one object, observed at subsequent time points

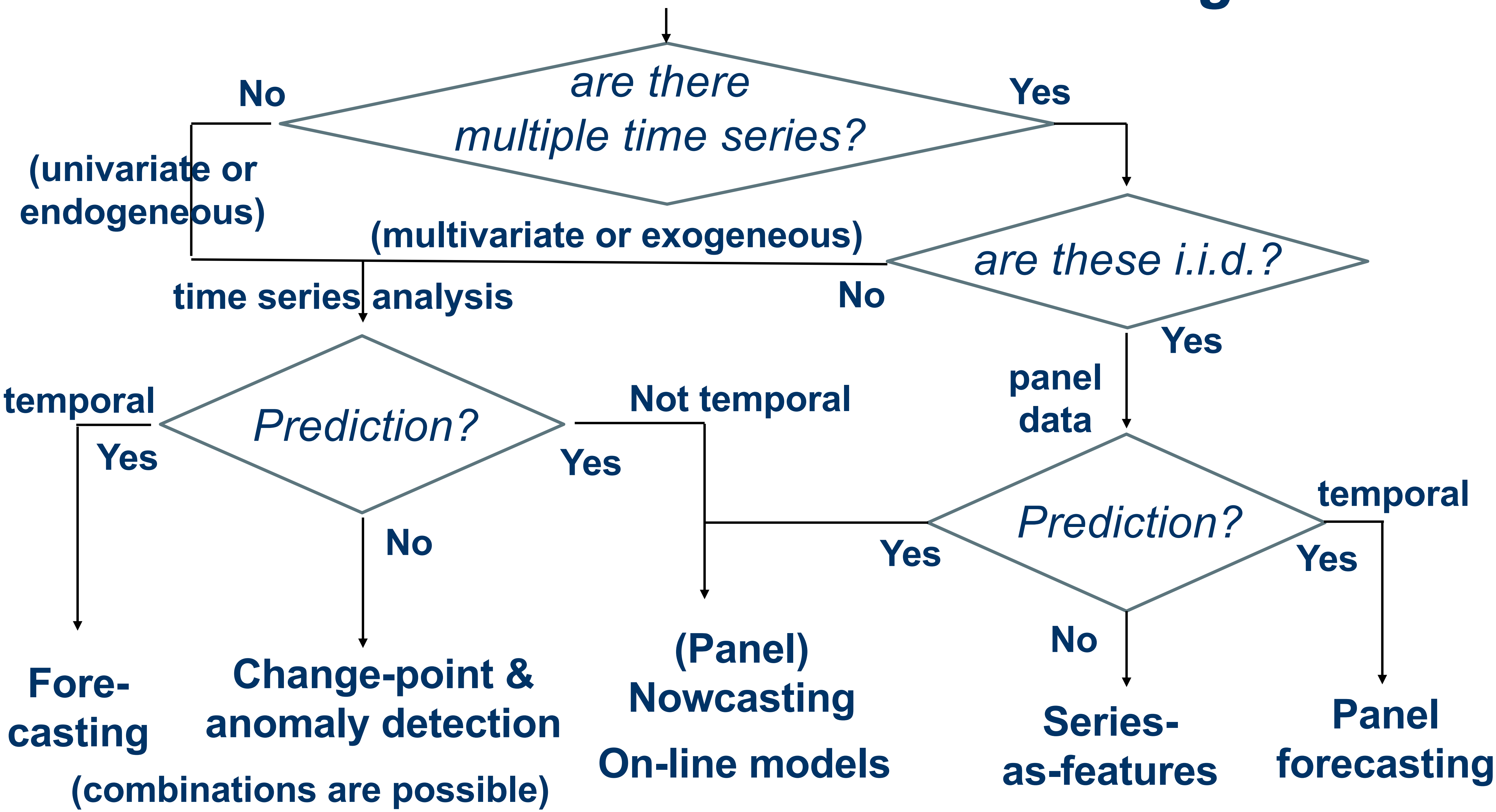
Alternative assumptions lead to *difficulties*, much is open

models using i.i.d. strategies (CV tuning etc) perform badly

„multivariate time series“, forecasting/annotation tasks

Time series related modelling tasks

Crucial distinction: what is the scientific goal?



Applicable models depend on the task/setting!

Reduction: moving between tasks

= *solving learning task A by a solution for learning task B*

usually (not necessarily) task A is more difficult

The „act of reduction" is an algorithm in itself!

Toy example: regression to classification (see Longley)

Task A: predict a number, learn functional $f : \mathcal{X} \to \mathbb{R}$

Task B: predict yes/no, learn functional $f : \mathcal{X} \to \{\text{yes}, \text{no}\}$

Example reduction algorithm (silly thresholder)

hyper-parameters: Increasing cut-offs $a_1, a_2, \dots, a_k \in \mathbb{R}$

fitting: convert training data $(X_1, Y_1), \dots, (X_N, Y_N)$

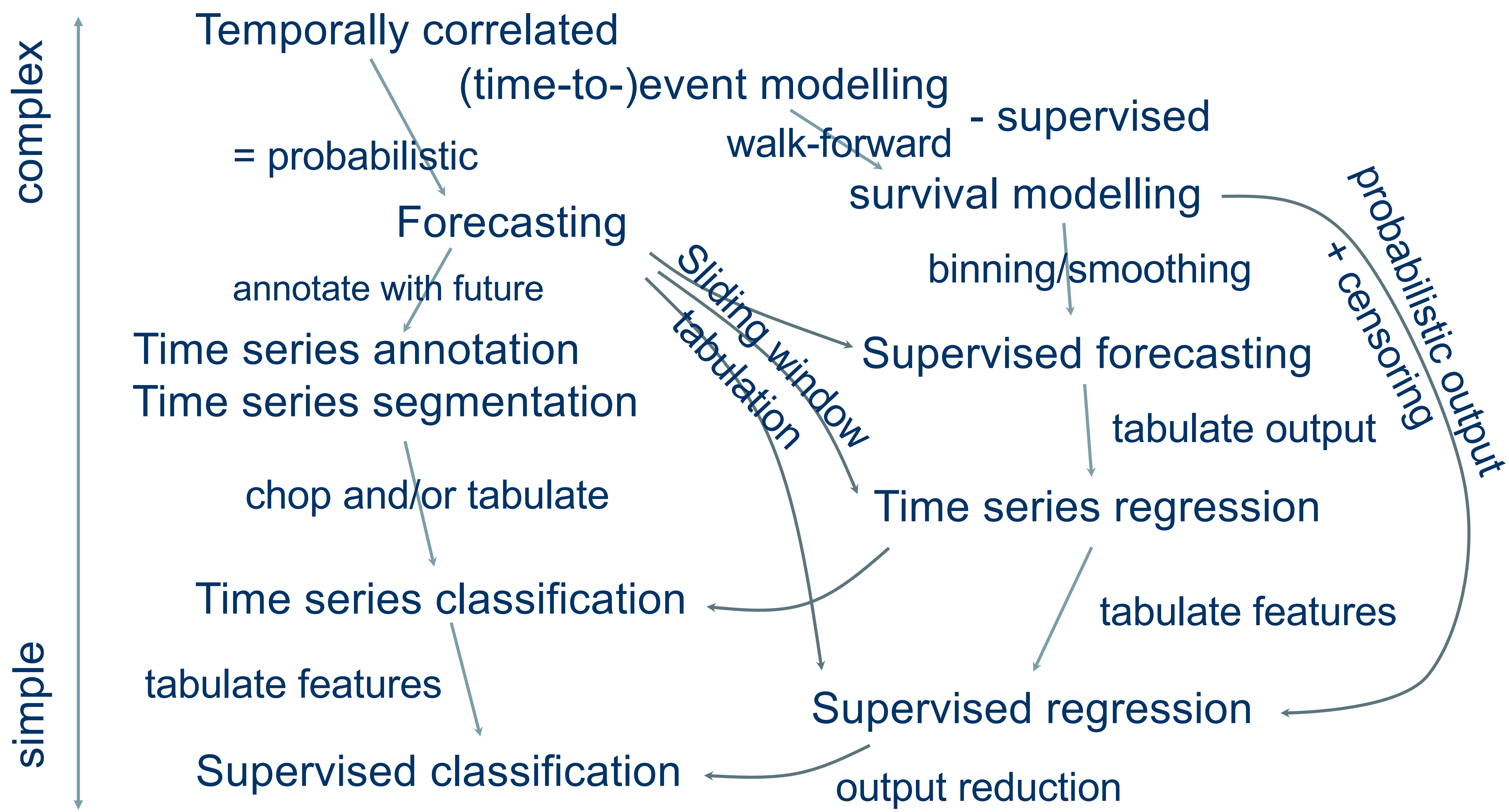
to $(X_1, Z_{11}, \dots, Z_{1k}), \dots, (X_N, Z_{N1}, \dots, Z_{Nk})$ wh. $Z_{ij} := \mathbb{1}(Y_i \le a_j)$

train $f_j : \mathcal{X} \to \{\text{yes}, \text{no}\}$ s.t. $f_j(X_i) \approx Z_{ij}$

prediction: for $X_* \in \mathcal{X}$ aggregate $f_j(X_*)$ to a number in $\mathbb{R}$

reduction strategy is a model composition strategy!

Time series task reduction schema



Reduction strategies listed are common examples (there are more)

ML toolboxes

Why ML toolboxes?

Standardized modelling & templating workflows

Learners/estimators and components follow standard API

choices (e.g., parameters) exposed through core interfaces points

enables collaboration, debugging, extension, deployment

Transparent external evaluation and inspection

Unified, interoperable interfaces allow interfacing meta-layers, e.g.,

large-scale benchmarking studies; model interpretability methods

enables scientific/commercial success control, monitoring in-use

Rapid experimentation and deployment cycles

ML toolbox as abstraction framework for practical use

easy user access to off-shelf methodology, standard workflows

enables progress and innovation in research and applications

Key features of (sklearn-like) ML toolboxes

Model interface: provide access to a wide class of models

e.g., OLS, support vector machines, neural networks, etc

Fitting and prediction: simple interface given train/test data

Settable hyper-parameters: easily accessible and changeable

This should be similar for all classes and kinds of models

Model tuning & composition: grid-tuning, ensembling, pipelines

Specification of tuning/composition parameter same for all models

Exposing meta-model parameters as hyper-parameters of result

Model validation & evaluation: estimation of generalization loss

for standard loss metrics and re-sampling based validation schemes

Running of benchmarking experiments including all the above

User/workflow interaction: experiment set-up and reporting

All enablers of reproducibility and scientific transparency!

Incomplete list of toolboxes for time series

tsfresh	Feature extraction
ts-gluon	Probabilistic forecasting and anomaly detection with deep learning (Amazon)
statsmodels	Traditional forecasting models like ARIMA, Exponential Smoothing, etc. (currently not maintained)
prophet	Forecasting with multiple seasonalities (Facebook)
tslearn	Time series classification
Featuretools	Feature extraction
pmdarima	Auto-ARIMA in Python

For a more complete list, see https://github.com/alan-turing-institute/sktime/wiki/Related-software

Exhaustive list of ML toolboxes for time series:

(i.e., with sklearn-like interface and features as on previous slide)

2018

2020

Computer Science > Machine Learning

sktime: A Unified Interface for Machine Learning with Time Series

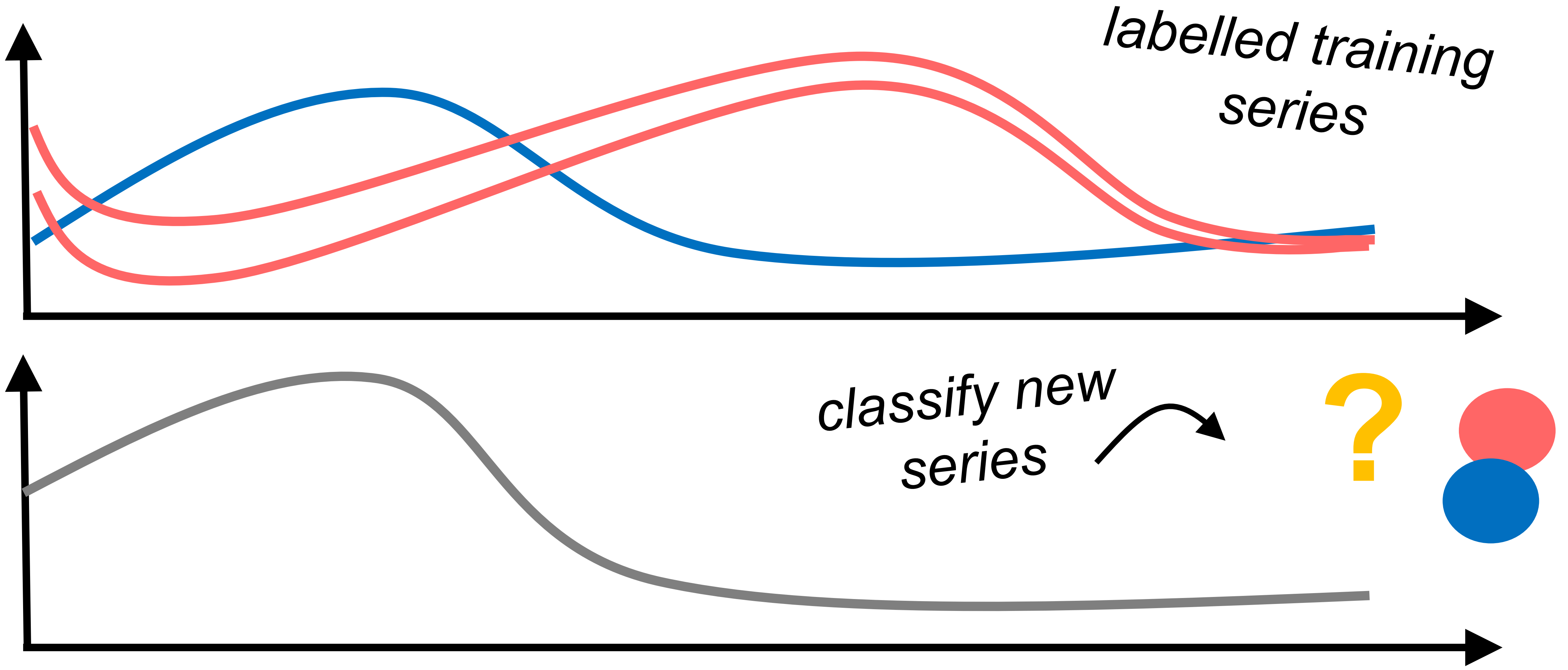
Markus Löning, Anthony Bagnall, Sajaysurya Ganesh, Viktor Kazakov, Jason Lines, Franz J. Király

(Submitted on 17 Sep 2019)

We present sktime -- a new scikit-learn compatible Python library with a unified interface for machine learning with time series. Time series data gives rise to various distinct but closely related learning tasks, such as forecasting and time series classification, many of which can be solved by reducing them to related simpler

sktime in action

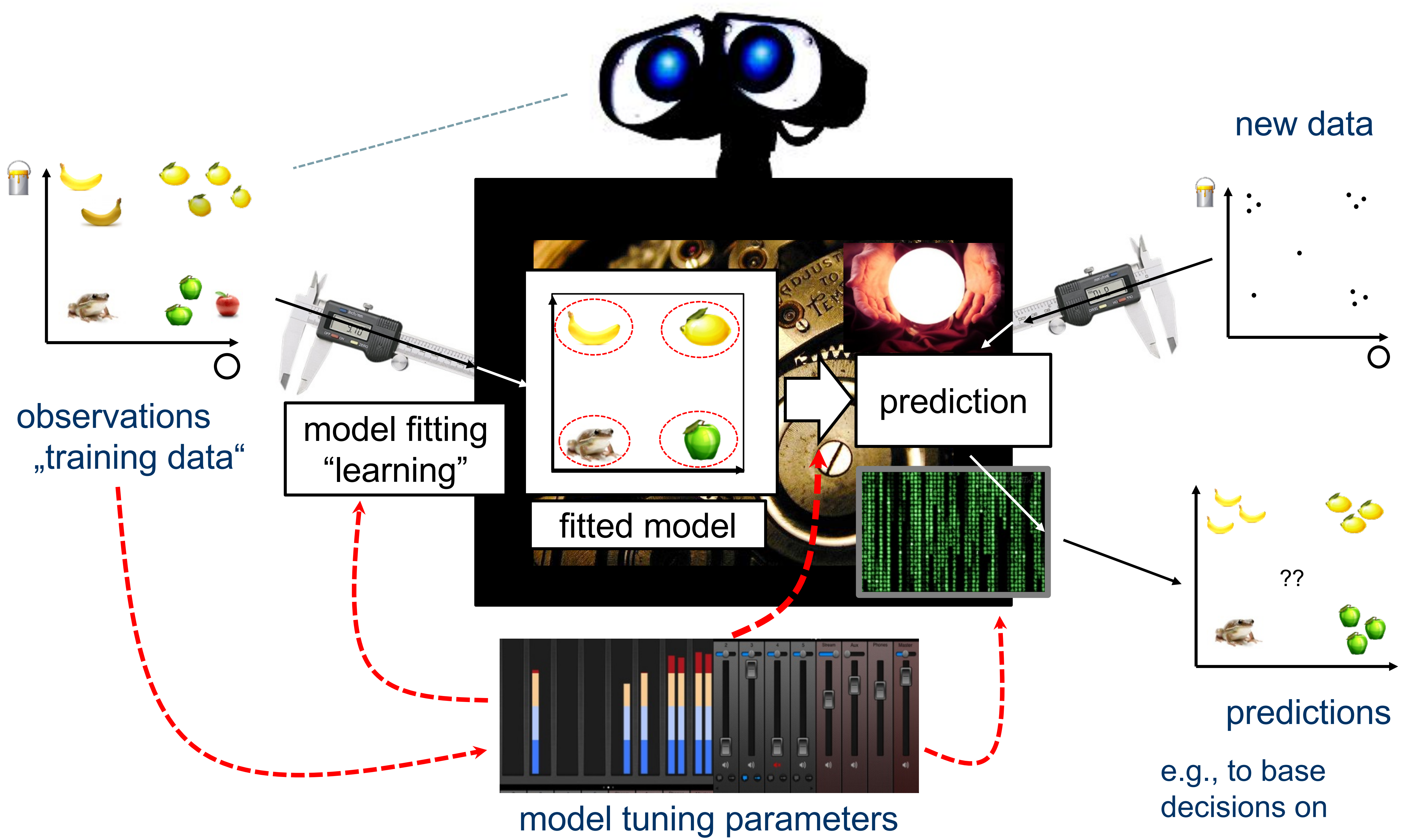
Time series classification



Time series classification API

```
clf = TimeSeriesForestClassifier(n_estimators=100)

# fit/predict interface
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

# common hyper-parameter interface
clf.get_params()
```

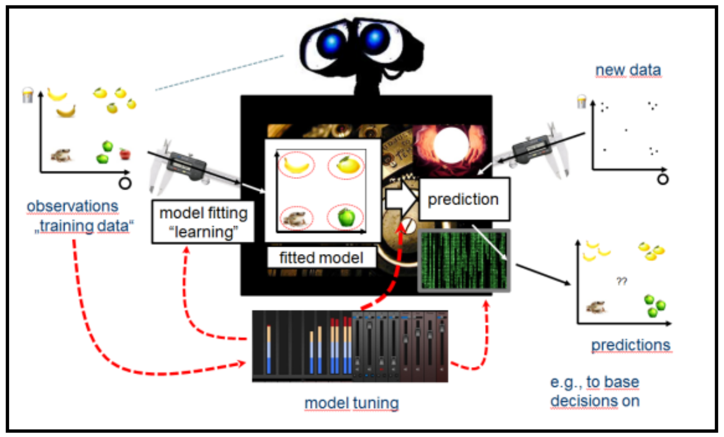
API design: learners as classes/objects



API design: ML toolboxes

as found in the `R/mlr` or `scikit-learn` packages

Leading principles: encapsulation, modularization



modular structure

object orientation

„`learning machine`“ object

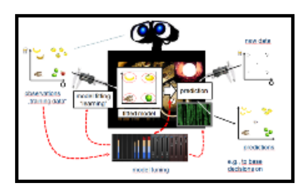
```
TimeSeriesForest
fit(X_train)
predict(X_test)
```

plus metadata & model info

Unified interface for parts of the ML workflow

„pipelining“ *„model building“*

data access → Pre-processing, feature extraction → model interface



model validation, benchmarking

„success control“

„estimating the generalization error“

(cross-validation also used here)

Meta-modelling, automated modeling

e.g., by cross-validation, Bayesian optimization

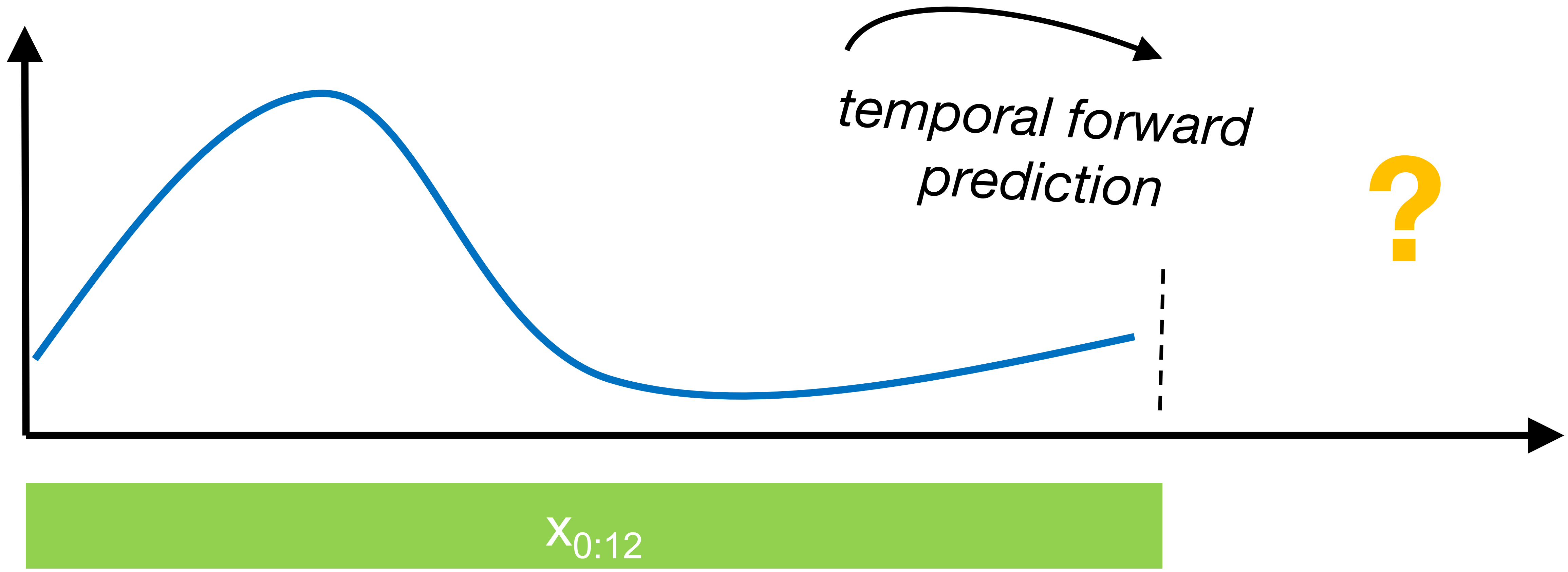
„tuning“ *„model selection“*

Current state-of-the-art classifiers[1] in sktime

- **Interval based:** time series forest, RISE
- **Distance based:** Elastic Ensemble, Proximity Forest, KNN, kernels
- **Shapelet based:** Shapelet transform, Shapelet Forest
- **Dictionary based:** SAX, SFA, BOP, BOSS
- **Deep learning:** https://github.com/sktime/sktime-dl

[1] See Bagnall, Anthony, et al. (2017) "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances." and Fawaz, Hassan Ismail, et al. (2019) "Deep learning for time series classification: a review."

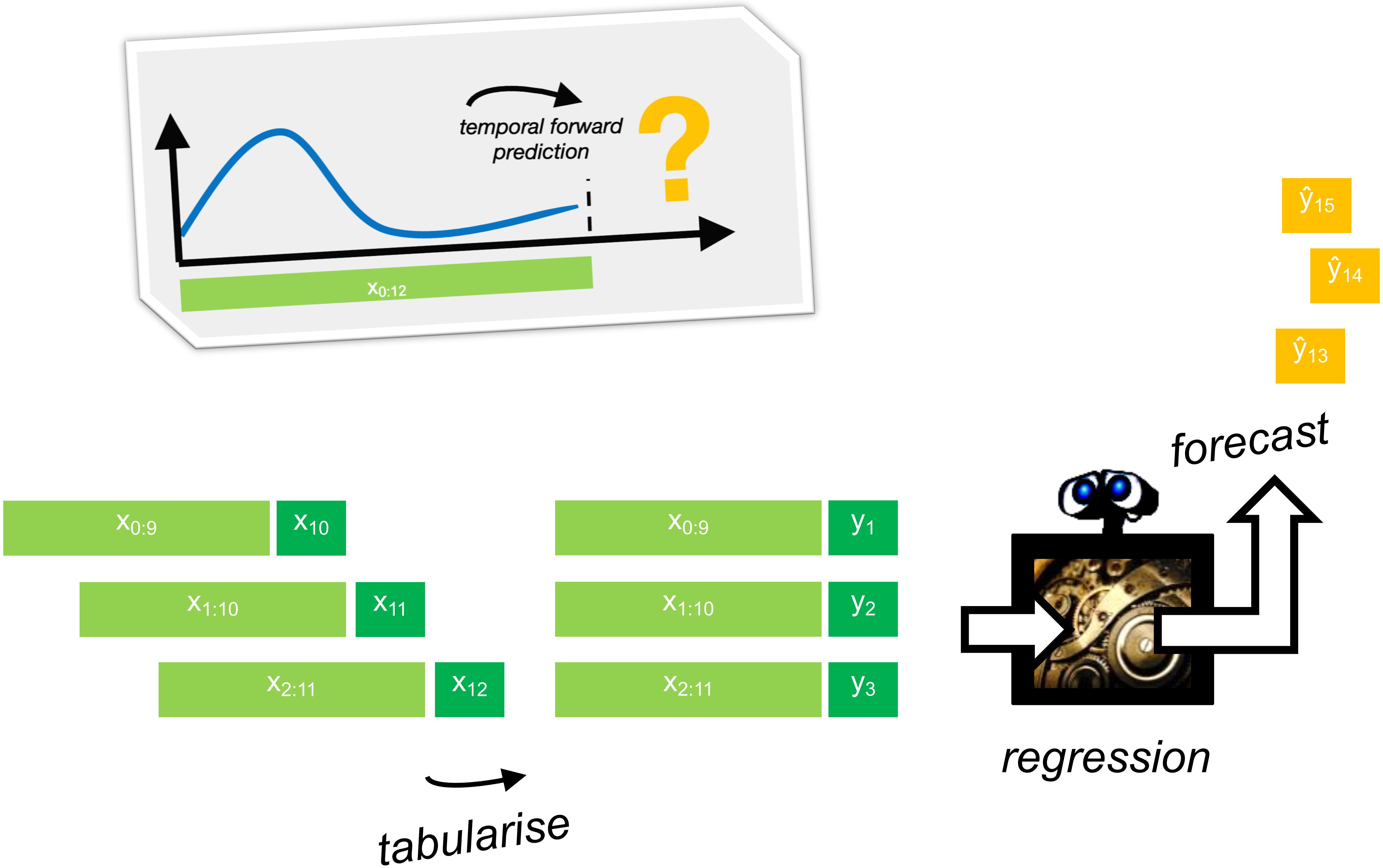
Forecasting



Forecasting API (experimental)

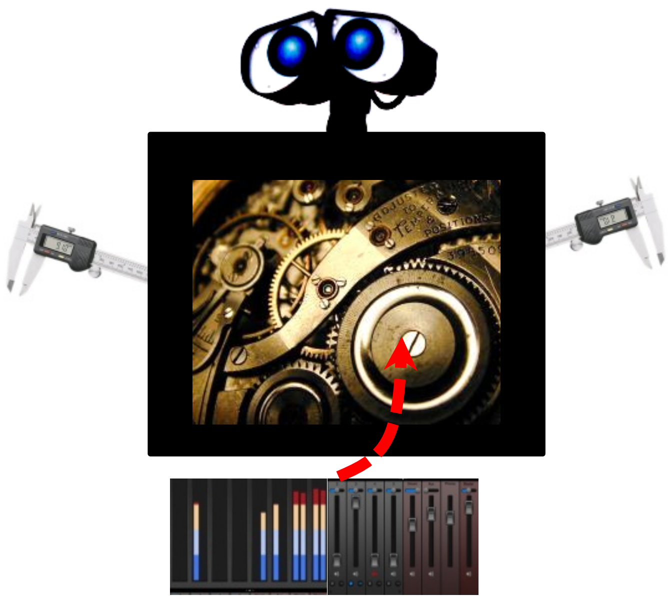
```
f = ARIMAForecaster()
f.fit(y_train)

fh = np.arrange(1, 4) #forecasting horizon
y_pred = f.predict(fh)
```

Solving forecasting by regression?

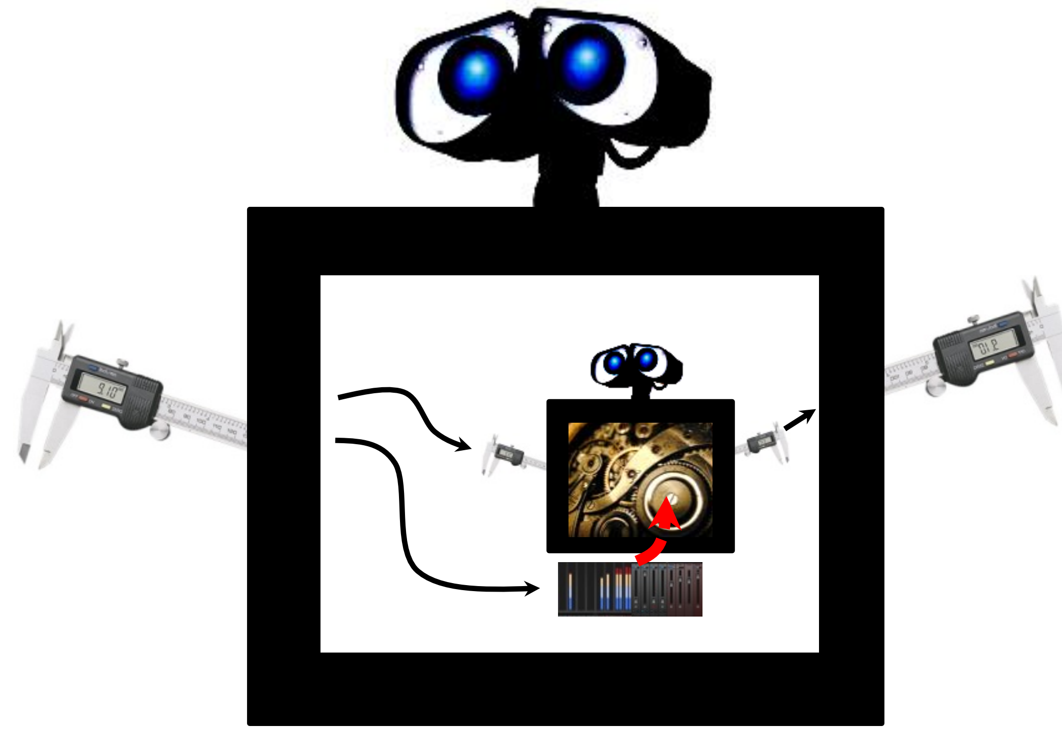


Meta-estimators

Tuning

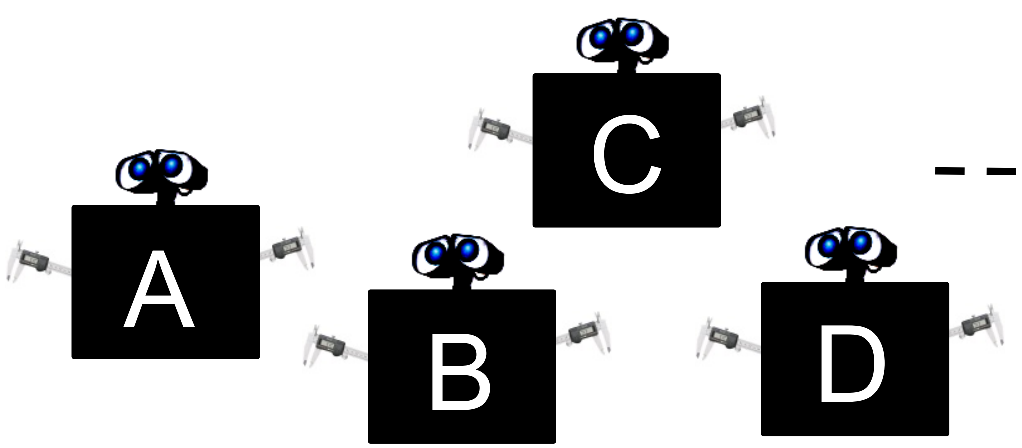


Model with tuning parameters

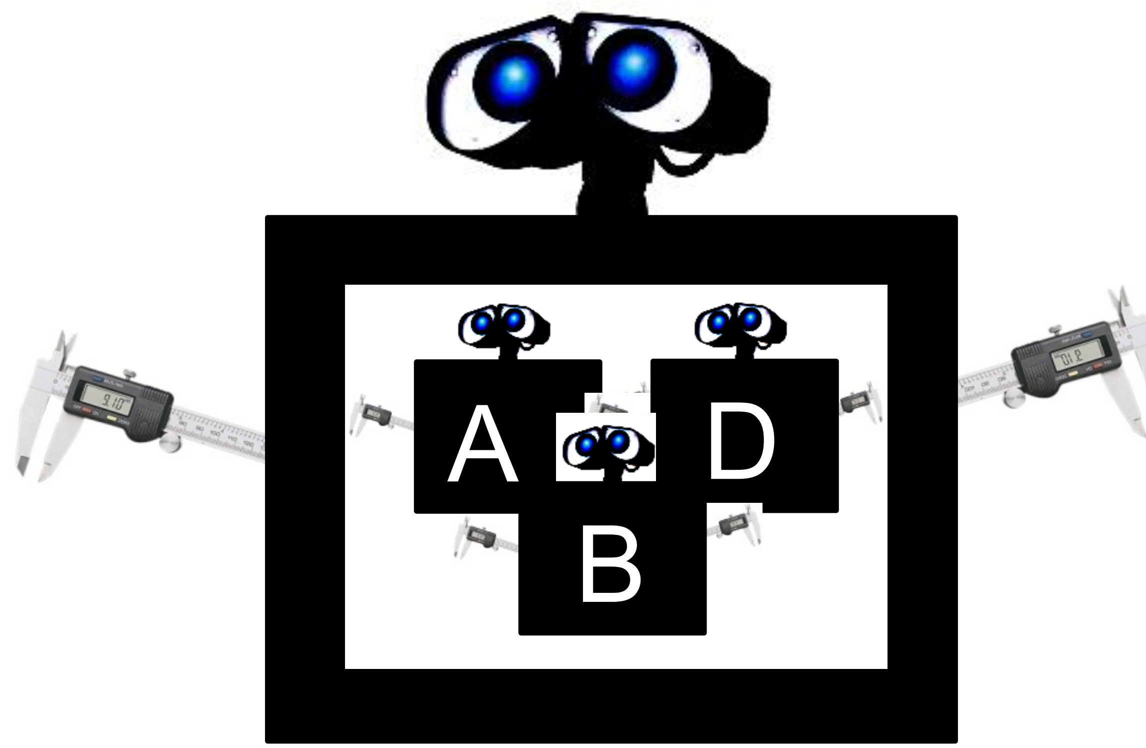


Best tuning parameters are determined using data-driven tuning algorithm

Ensembling

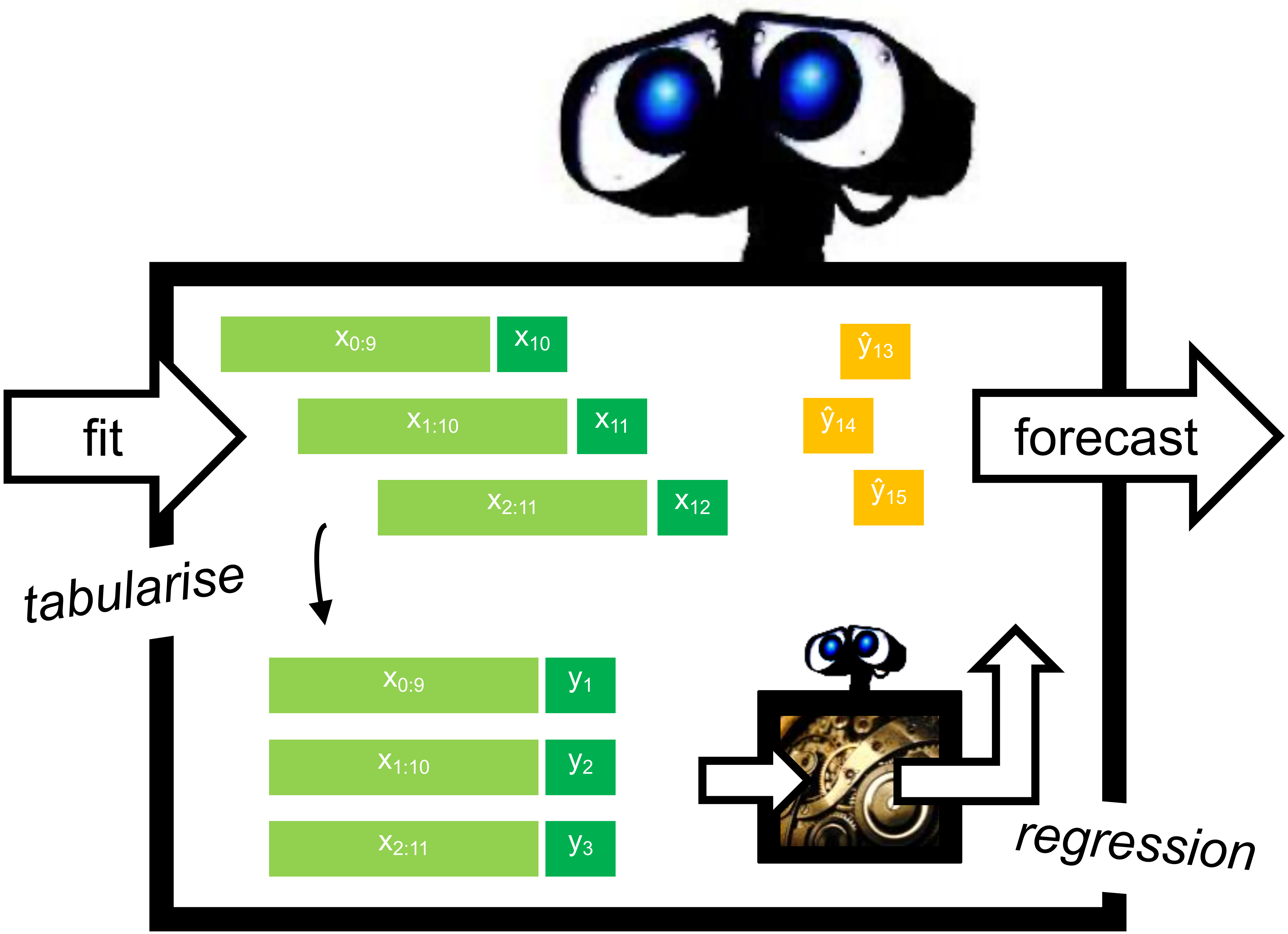


a number of (possibly „weak“) models



„strong“ ensemble model

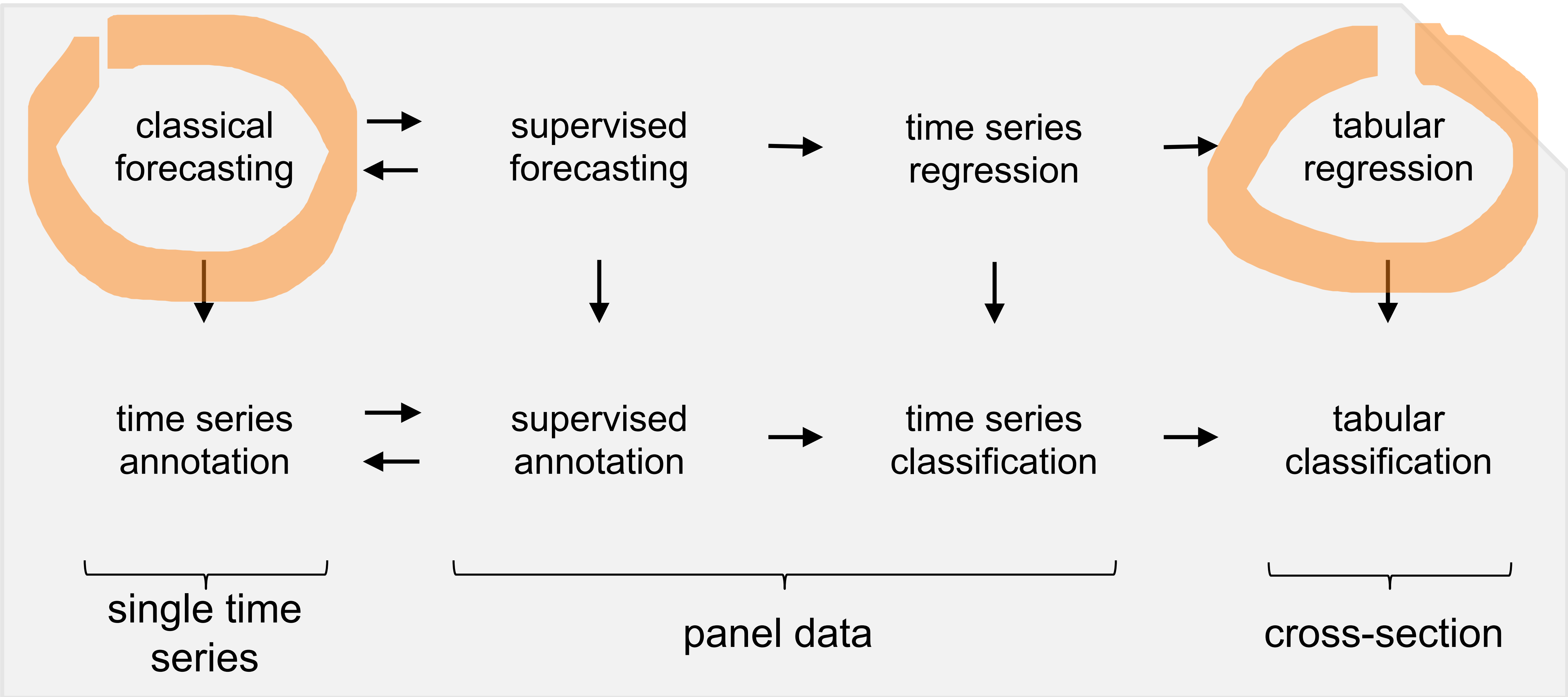
Reduction as a meta-strategy



Forecasting API (experimental)

```
regressor = RandomForestRegressor()

# reduction as meta-estimator
f = ReducedRegressionForecaster(
        regressor=regressor,
        look_back=10
)

f.fit(y_train)
y_pred = f.predict(fh)
```

More reduction relations



Detrending as meta-estimator

```python
f = ARIMAForecaster()

# detrender takes in forecaster
d = Detrender(forecaster=f)

yt = d.fit_transform(y)
```

```python
f = TransformedTargetForecaster(
       estimator=ARIMAForecaster(),
       transformer=Detrender()
)
f.fit(y_train)
y_pred = f.predict(fh)
```

The future and how you can help us get there

Development roadmap

- **Time series regression** (refactoring available classifiers into regressors)
- **Time series annotation:** segmentation and outlier detection
- **Supervised/panel forecasting** based on pysf[2]
- **Unsupervised methods:** time series clustering
- **Data container** (better representation and more efficient handling of time series data in modelling workflows)
- **Probabilistic temporal modelling:** survival and point process models based on skpro[3]

2 https://github.com/alan-turing-institute/pysf

3 https://github.com/alan-turing-institute/skpro

How to contribute

- API design
- Implementation (estimators, routines, interfaces to other toolboxes)
- Applied projects
- Documentation (tutorial notebooks, cheat-sheets, online documentation)
- Dev ops (testing, builds, CI, website)

Good first issues

5 Open ✓ 0 Closed Author Label Projects Milest

- Add feature importance graph to time series forest `enhancement` `good first issue` `implementation: algorithms`
  #214 opened 16 minutes ago by mloning
- Refactor time series classifiers into regressors `enhancement` `good first issue` `implementation: algorithms`
  #212 opened yesterday by mloning
- Add tsfresh transformer `enhancement` `good first issue` `interfacing algorithms`
  #81 opened on 18 Jun 2019 by mloning
- Implement forecasting algorithms `good first issue` `help wanted` `implementation: algorithms`
  #67 opened on 20 May 2019 by mloning
- Implement important series-to-series transformers `good first issue` `implementation: algorithms` `implementation: framework` `must - high priority`
  #6 opened on 4 Jan 2019 by fkiraly

Thank you for listening!

- GitHub: https://github.com/alan-turing-institute/sktime
- DL extension: https://github.com/sktime/sktime-dl
- Chat: https://gitter.im/sktime/community

- The first phase of development for sktime was done jointly between researchers at the University of East Anglia (UEA), University College London (UCL) and The Alan Turing Institute as part of the UK Research Innovation (UKRI) Strategic Priorities Fund, particularly the “Tools, Practices and Systems” theme with that grant, to develop tools for data science and artificial intelligence [EPSRC grant: EP/T001569/1]
- Markus Löning’s contribution was supported by the Economic and Social Research Council (ESRC) [grant: ES/P000592/1], the Consumer Data Research Centre (CDRC) [ESRC grant: ES/L011840/1], and The Alan Turing Institute [EPSRC grant: EP/N510129/1]

UEA University of East Anglia

UCL

The Alan Turing Institute

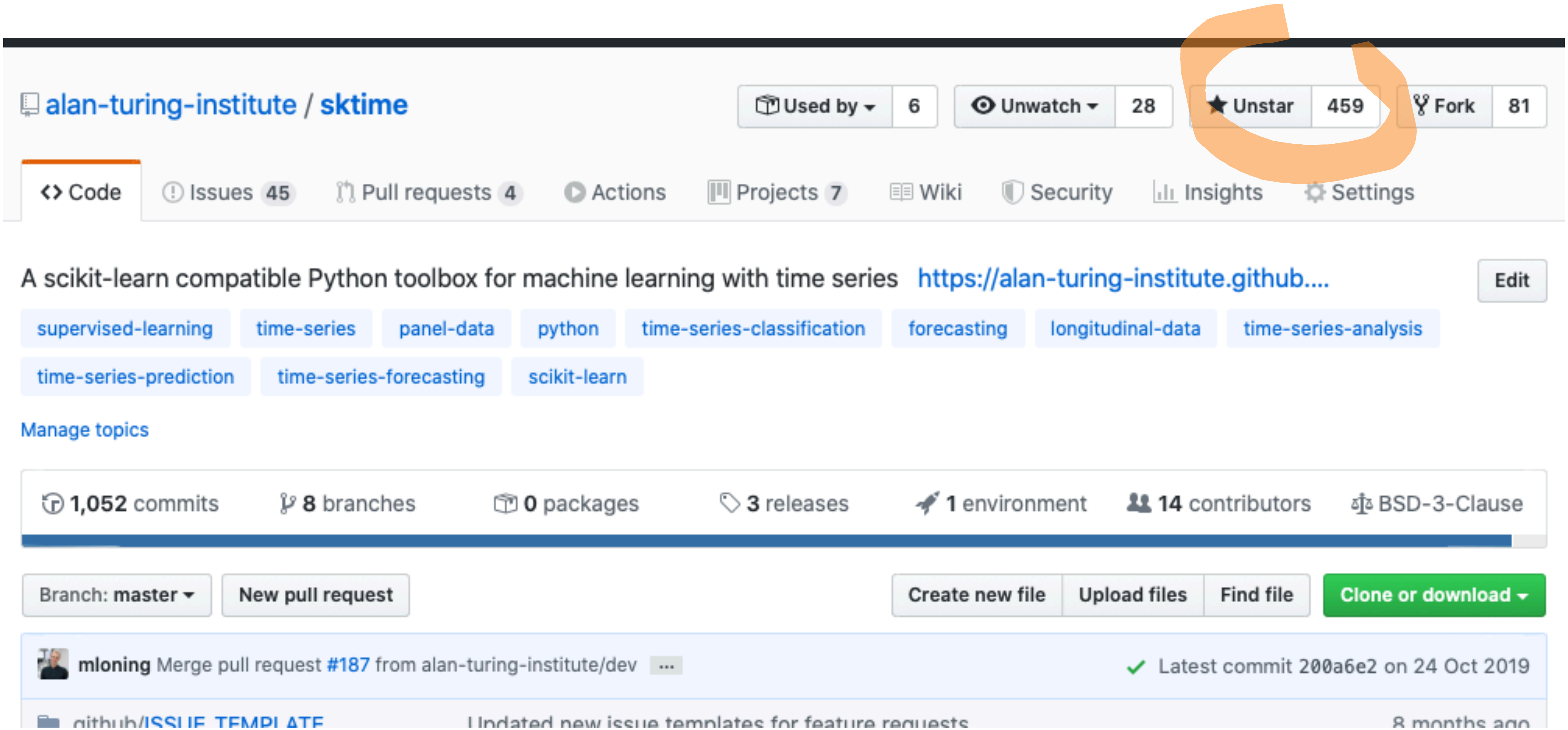
Hands-on tutorial

1. Go to https://github.com/alan-turing-institute/sktime

2. Launch binder launch binder
   Or pip install sktime and download repo to run notebooks locally

- Raise an issue on GitHub
- Chat on Gitter
- Give us feedback

Hands-on tutorial



Feedback

- Which functionality of sktime did you find most useful?
- Which part of the sktime was most confusing?
- Which new features would you like to see most?
- Are you planning to use sktime in your future projects? If yes, why? If no, what would convince you to use sktime in the future?