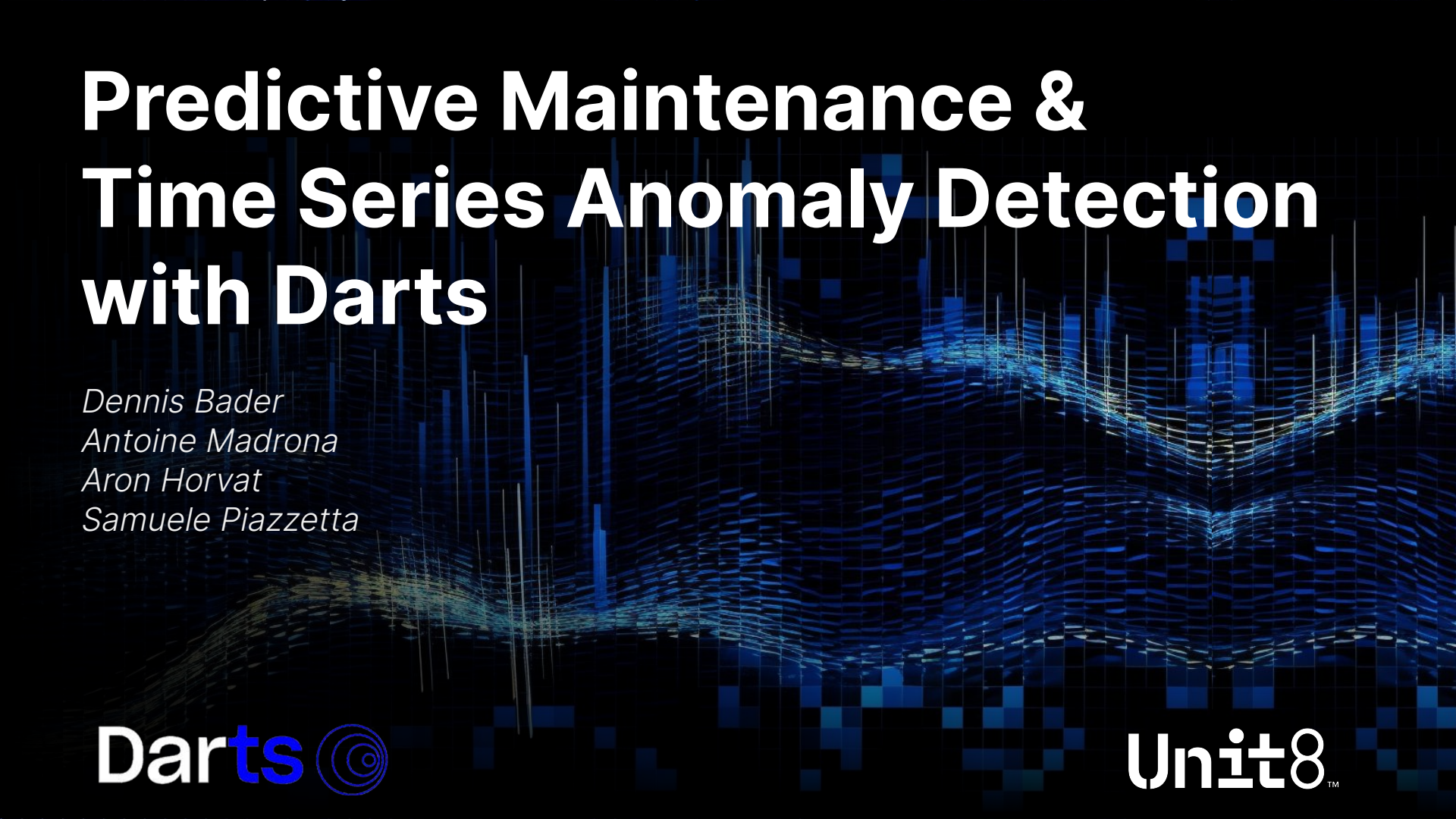
Predictive Maintenance & Time Series Anomaly Detection with Darts

Dennis Bader
Antoine Madrona
Aron Horvat
Samuele Piazzetta



Your Unit8 team today

Dennis Bader

Data Scientist &
Darts Lead Developer

MSc ETH
Mechanical Engineering

Antoine Madrona

Data Scientist &
Darts Developer

MSc EPFL
Life Science Engineering

Aron Horvath

Data Scientist &
Data Engineer

PostDoc University Zurich
Clinical Data Science

Samuele Giuliano Piazzetta

Data Scientist &
Darts Developer

MSc ETH
Computer Science

Unit8 in a nutshell

Turning data into value

Unit8 is a leading Swiss data services company with a mission to help non-digital native companies transform data into value with a mix of data science, analytics, and AI.

Selected Clients

FT FINANCIAL TIMES 1000 EUROPE'S FASTEST GROWING COMPANIES 2023 statista

DAIMLER MERCK Firmenich Swiss Re

helvetia UBV TECAN.

Unit8

unit8.com

Key facts:

- Established in 2017
- 110+ employees and growing
- 5 locations: Zürich, Lausanne, Bern (Switzerland), Frankfurt (Germany) and Krakow, Wroclaw (Poland)
- 160+ data & AI projects completed

We operate at the intersection of technology and business to accompany our customers at every step of their data & AI journey

Services portfolio:

- Data & AI Consulting
- Data & AI Solutions
- Data & AI Platforms
- Data & AI Operations
- Data & AI Academy

Key Partners

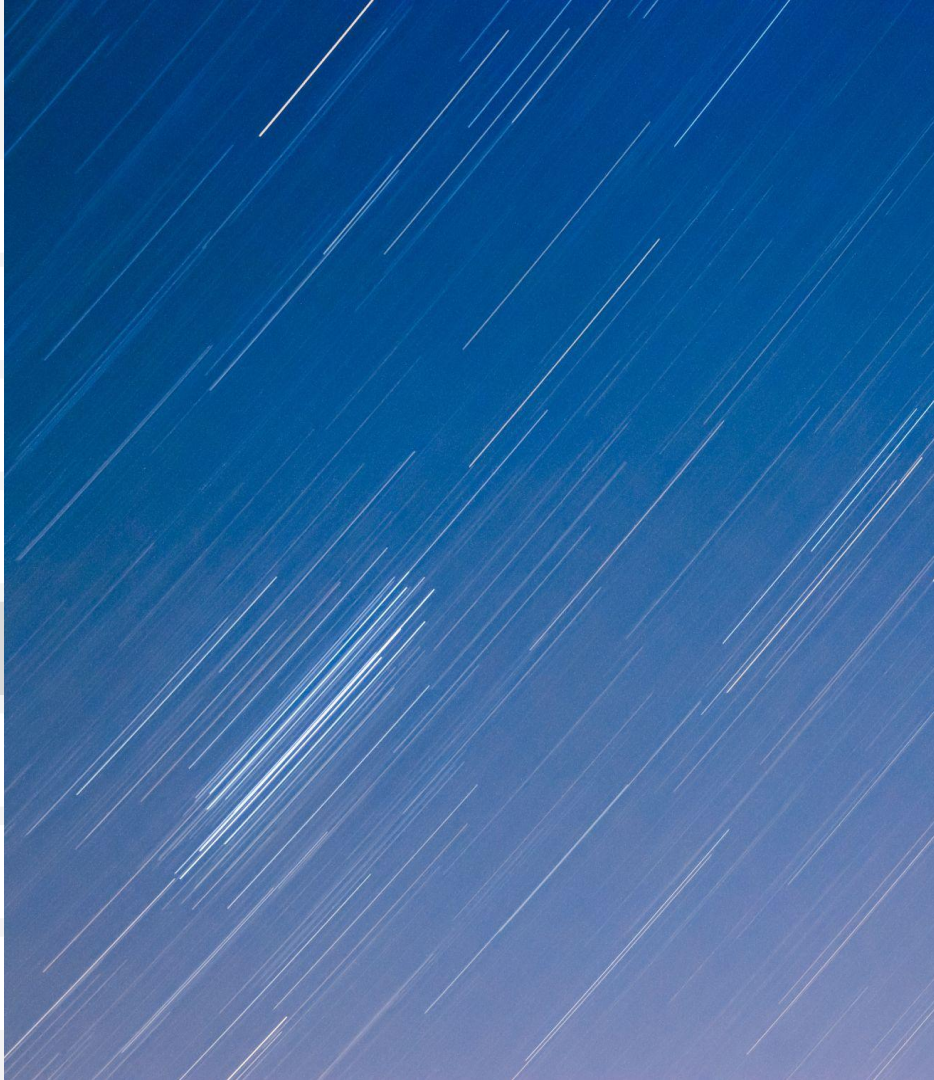
Microsoft aws Palantir data iku

digitalswitzerland doi data innovation alliance

digital natives

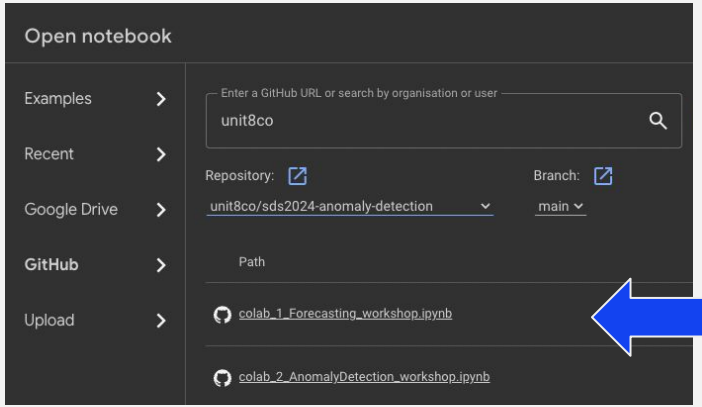
Agenda Today

13:40 - 14:10	Intro to Forecasting with Darts
14:10 - 14:30	*<Example Notebook>*
14:30 - 14:40	Intro to Anomaly Detection
14:40 - 15:00	*<Example Notebook>*
15:00 - 15:30	Break
15:30 - 15:45	Intro to Predictive Maintenance
15:45 - 16:45	*<Example Notebook>*
16:45 - 17:00	Outro

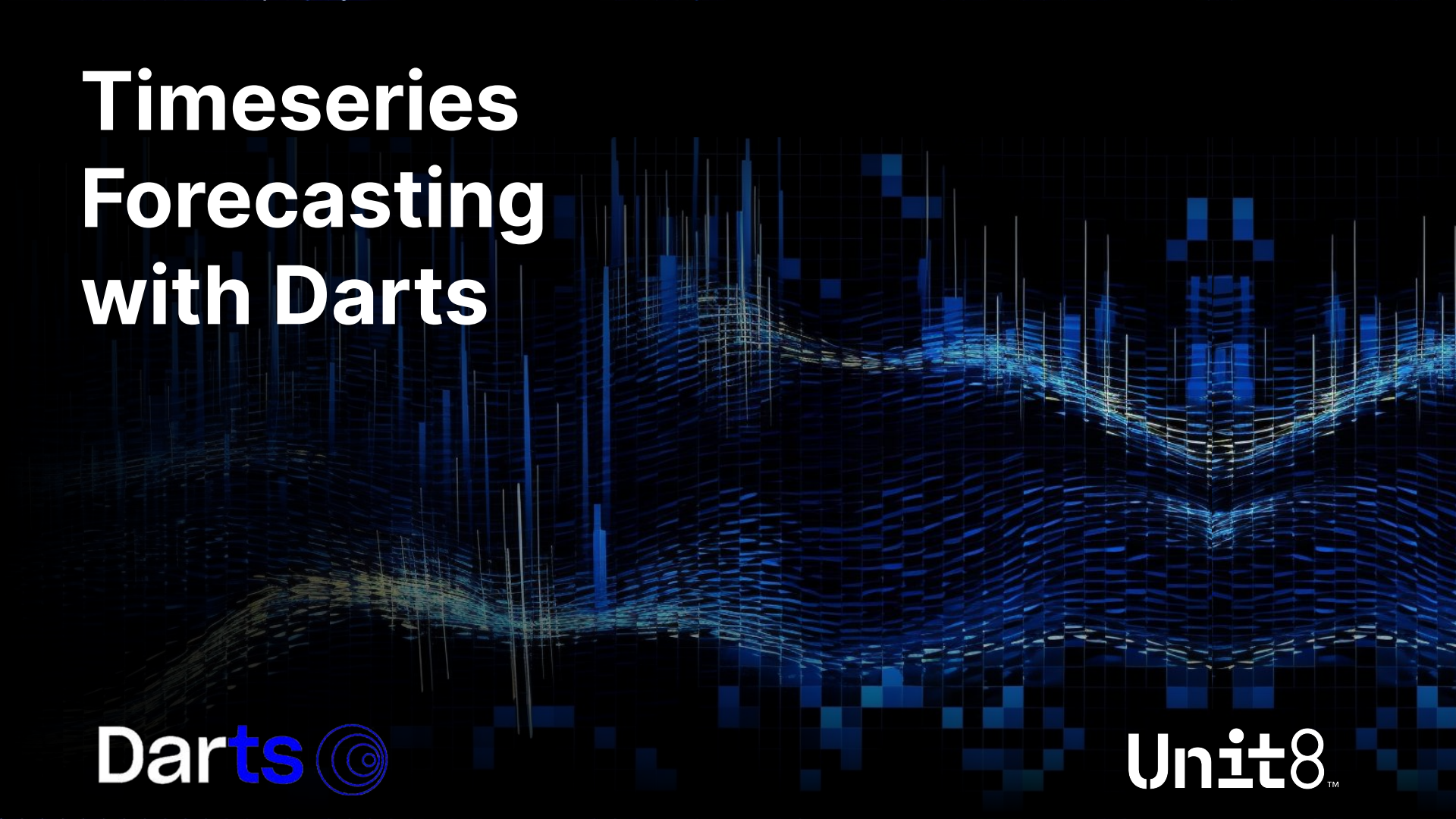


About The Example Notebooks

- **3 example notebooks** to work on with **Google Colab**
  - Open: https://colab.google/
  - Click on “Open Colab”
  - On the left sidebar click GitHub
    - GitHub URL: “unit8co”
    - Repository: “unit8co/sds2024-anomaly-detection”
    - The three notebooks to work on are:
      - “colab_1_Forecasting_workshop.ipynb”
      - “colab_2_AnomalyDetection_workshop.ipynb”
      - “colab_3_PredictiveMaintenance_workshop.ipynb”
  - **Run up to 3 notebooks simultaneously** in different tabs
  - **Run the notebooks** until you receive the first error (first cells install dependencies and data)
- We also provide a **solution** for each example with prefix “colab_solution*”



Timeseries Forecasting with Darts

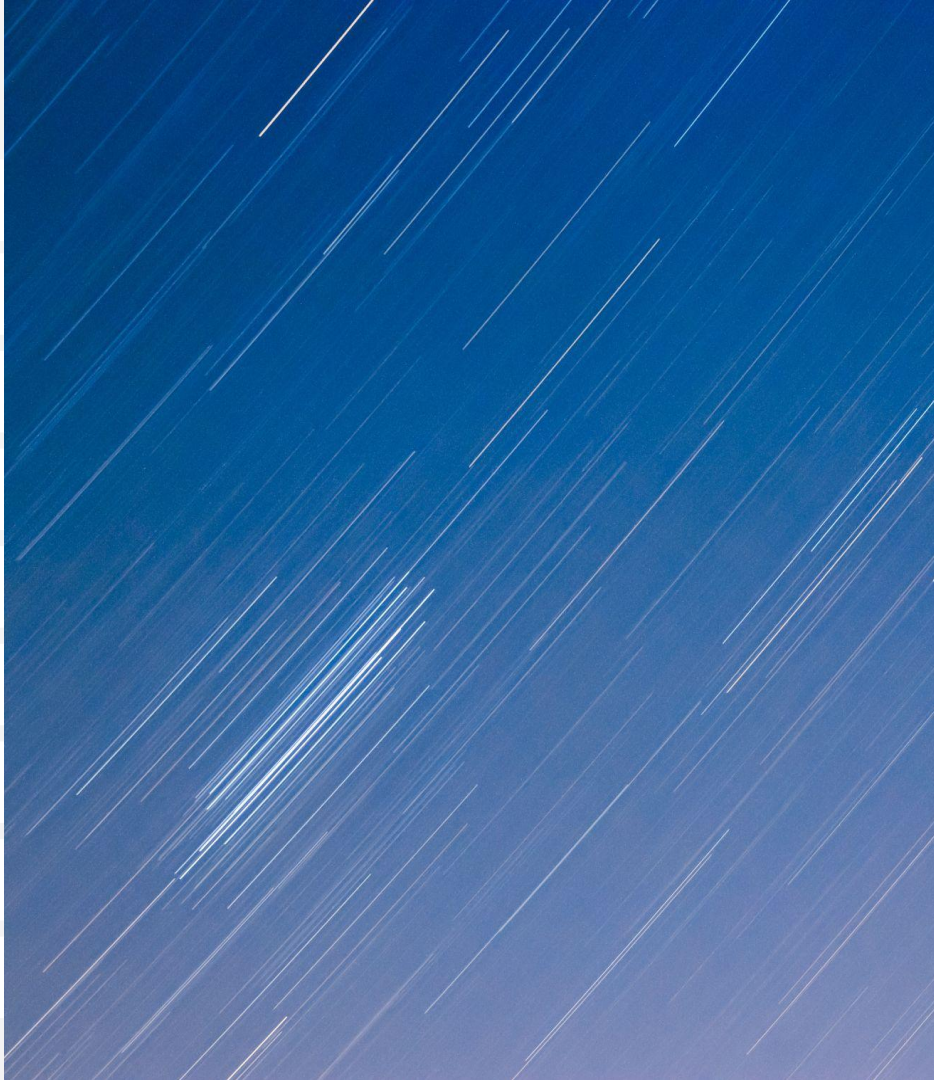


Darts

Unit8

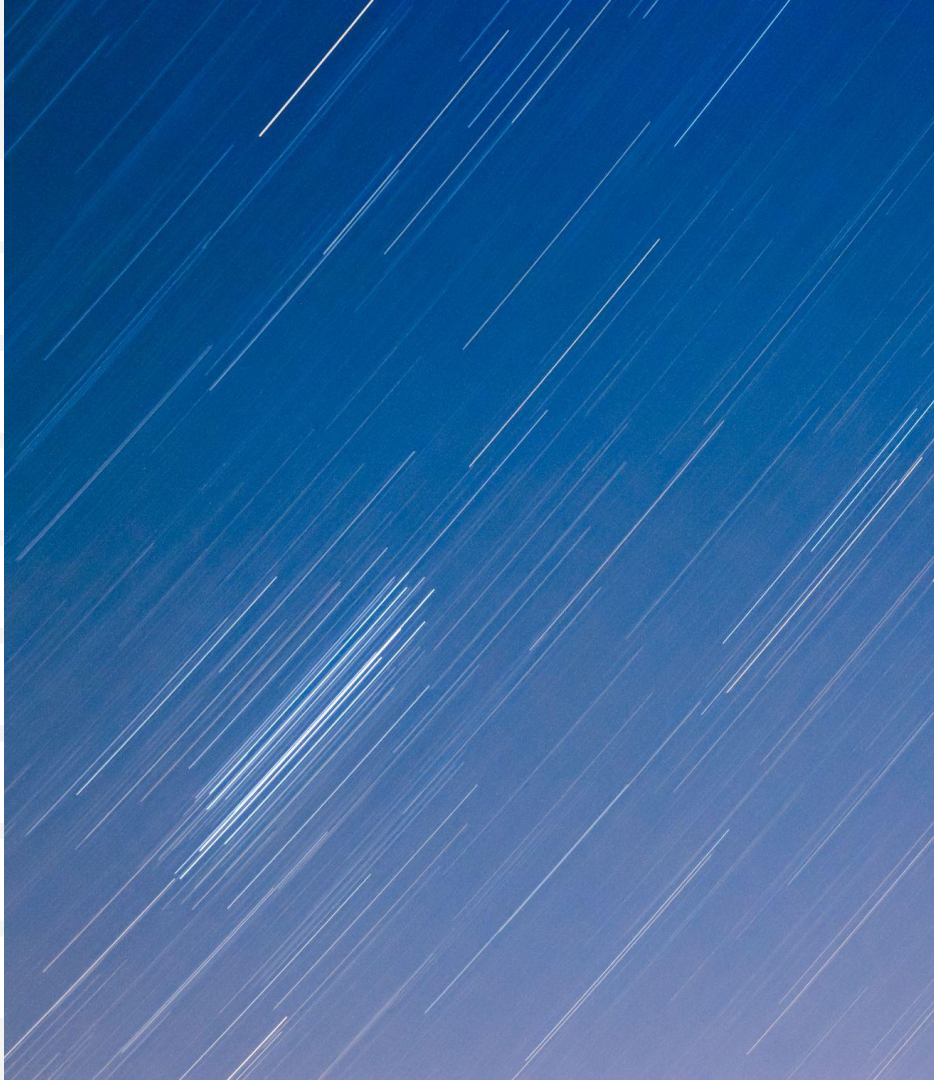
Agenda

1. Intro to Darts
2. TimeSeries Class
3. Forecasting models
4. Simple forecasting example
5. Historical forecasting and backtesting
6. Covariates
7. Probabilistic Forecasting
8. Training / predicting with multiple series
9. Summary and what’s coming next



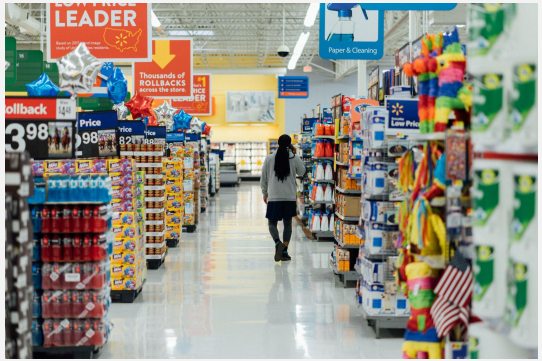
Agenda

1. **Intro to Darts**
2. TimeSeries Class
3. Forecasting models
4. Simple forecasting example
5. Historical forecasting and backtesting
6. Covariates
7. Probabilistic Forecasting
8. Training / predicting with multiple series
9. Summary and what's coming next



Why Forecasting / Anomaly Detection?

Retail



How can we ensure that the consumer demand for every product is met without overspending and creating waste?

Energy

How much energy should we produce?

How much energy can we produce?

What will the electricity price be tomorrow?

Predictive Maintenance

How can we detect anomalous device behavior before serious damage occurs?

Unit8

Darts 

Darts is Unit8's Python library for
easy time series manipulation,
forecasting and anomaly detection.

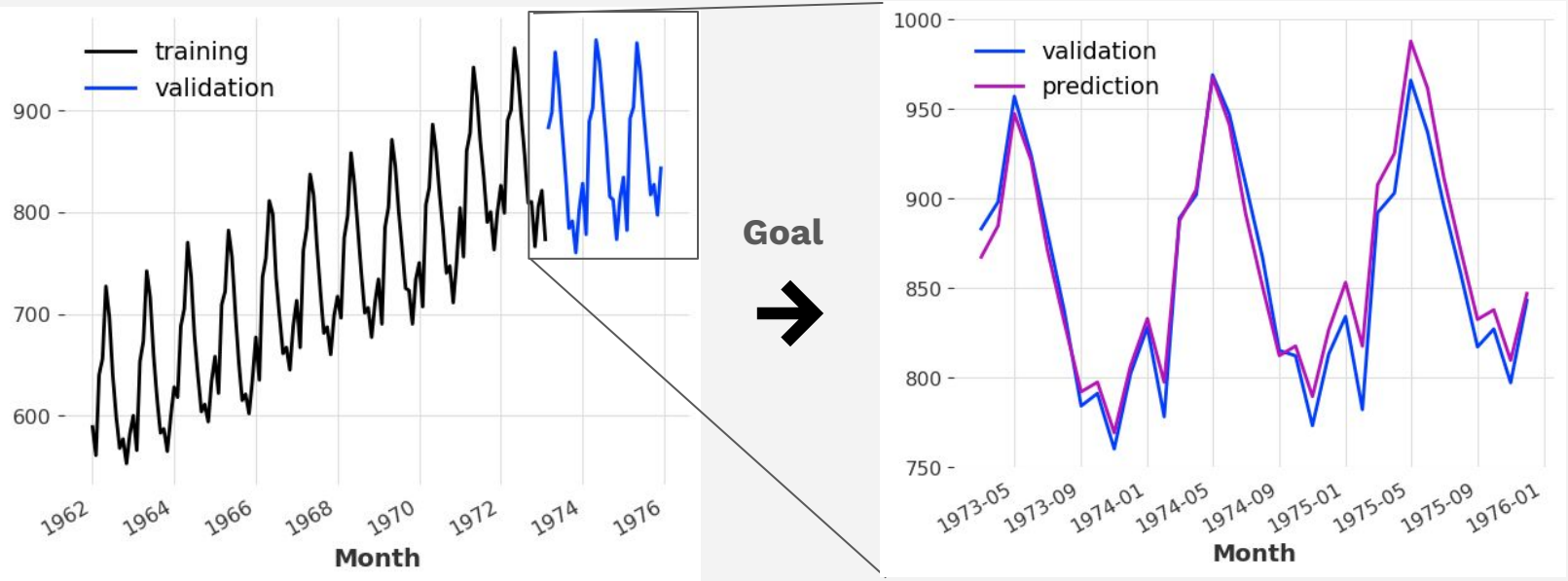
- Immutable `TimeSeries` class as basic building block.
- Unified `fit()`, `predict()` interface.
- 40+ classical models & state-of-the-art ML approaches for forecasting and anomaly detection
- User-friendliness: intuitive API and reasonable defaults.

Unit8

https://github.com/unit8co/darts

Darts

Try out 40+ models in only a few lines of code



Unit8

https://github.com/unit8co/darts

Agenda

1 Intro to Darts

2 TimeSeries Class

3 Forecasting models

4 Simple forecasting example

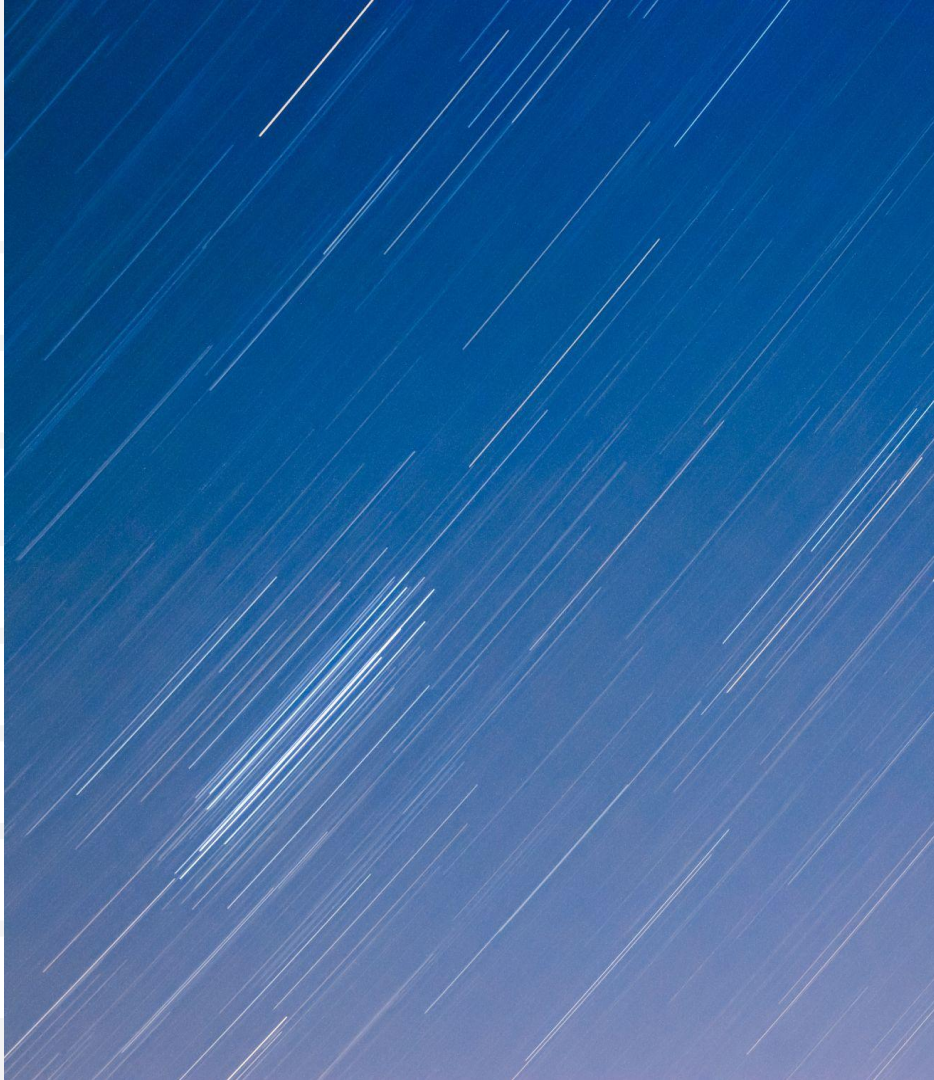
5 Historical forecasting and backtesting

6 Covariates

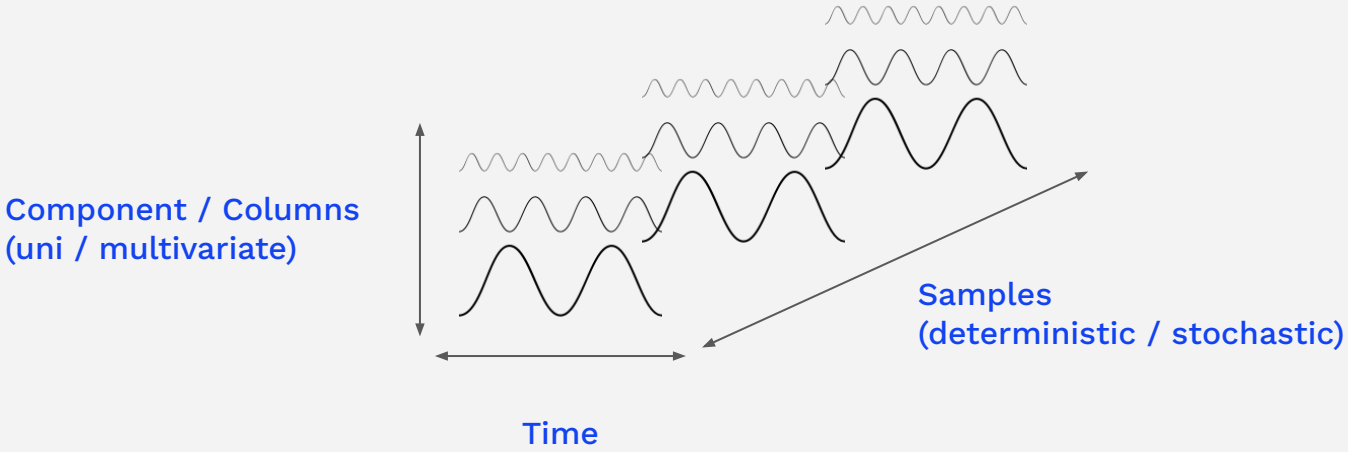
7 Probabilistic Forecasting

8 Training / predicting with multiple series

9 Summary and what's coming next



TimeSeries: Darts' Container for storing and manipulating time series data



Unit8

The TimeSeries object

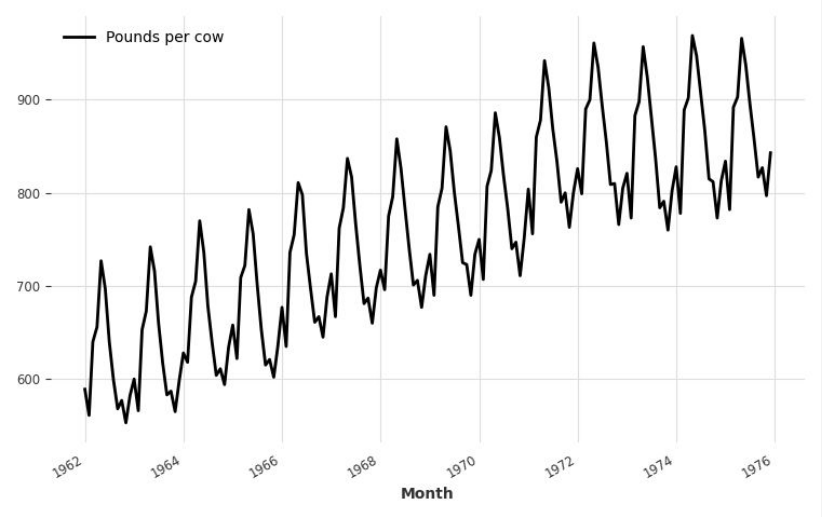
```python
from darts import TimeSeries

# create time series
series = TimeSeries.from_csv('milk.csv', time_col='Month')
series.plot()
```

Factory Method Support

pandas NumPy

xarray

& other file formats



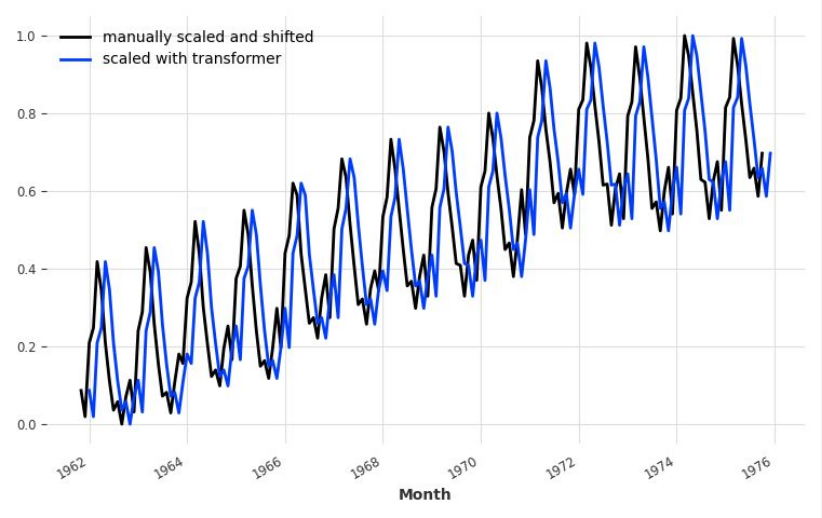
TimeSeries Manipulation

- **Methods and Arithmetic Operations**

```
# manually scale and shift
((series - 553.0) / 416.0).shift(-2).plot()
```

- **Data processing module**
  - Transformers
  - Missing value handling
  - Windowing

```
from darts.dataprocessing.transformers import Scaler

# scale with a data transformer
min_max_scaler = Scaler()
min_max_scaler.fit_transform(series).plot()
```



Agenda

1 Intro to Darts

2 TimeSeries Class

3 Forecasting models

4 Simple forecasting example

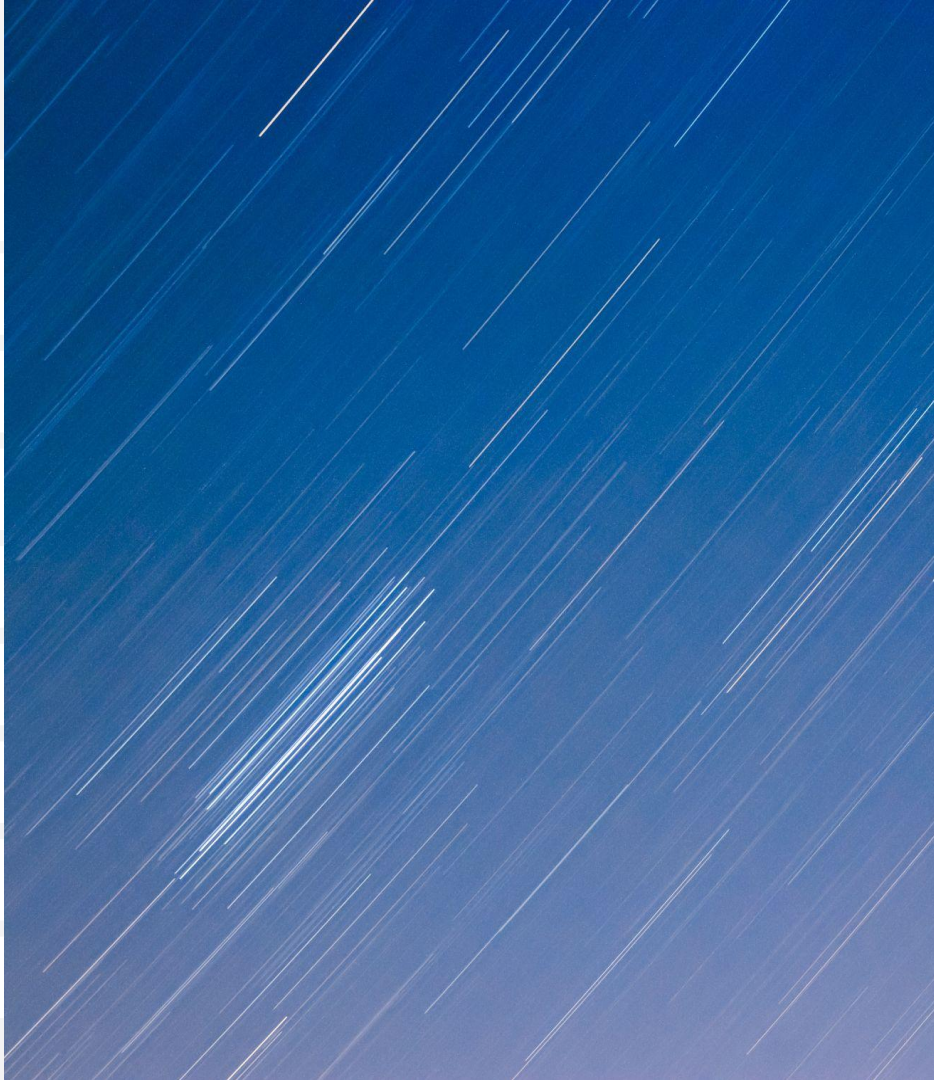
5 Historical forecasting and backtesting

6 Covariates

7 Probabilistic Forecasting

8 Training / predicting with multiple series

9 Summary and what's coming next



Darts ForecastingModels

Darts has a large collection of forecasting models; from statistical to deep learning models.

All models support:

- Univariate forecasting
- Unified sklearn-like fit()/predict() API

Additional support depending on the model:

- Multivariate forecasting
- Training on multiple series
- Using covariates (external data) as input
- Probabilistic forecasting
- Neural-network related features

Model	Sources	Target Series Support: Univariate/ Multivariate	Covariates Support: Past-observed/ Future-known/ Static	Probabilistic Forecasting: Sampled/ Distribution Parameters	Training & Forecasting on Multiple Series
Baseline Models (LocalForecastingModel)					
NaiveMean		🟩🟥	🟥🟥🟥	🟥🟥	🟥
NaiveSeasonal		🟩🟥	🟥🟥🟥	🟥🟥	🟥
NaiveDrift		🟩🟥	🟥🟥🟥	🟥🟥	🟥
NaiveMovingAverage		🟩🟥	🟥🟥🟥	🟥🟥	🟥
Statistical / Classic Models (LocalForecastingModel)					
ARIMA		🟩🟥	🟥🟩🟥	🟩🟥	🟥
VARIMA		🟥🟩	🟥🟩🟥	🟥🟥	🟥
AutoARIMA		🟩🟥	🟥🟩🟥	🟥🟥	🟥
StatsForecastAutoArima (faster AutoARIMA)	Nixtla's statsforecast	🟩🟥	🟥🟩🟥	🟩🟥	🟥

First 8 out of 40+ Forecasting Models, see https://unit8co.github.io/darts/

Agenda

1 Intro to Darts

2 TimeSeries Class

3 Forecasting models

4 Simple forecasting example

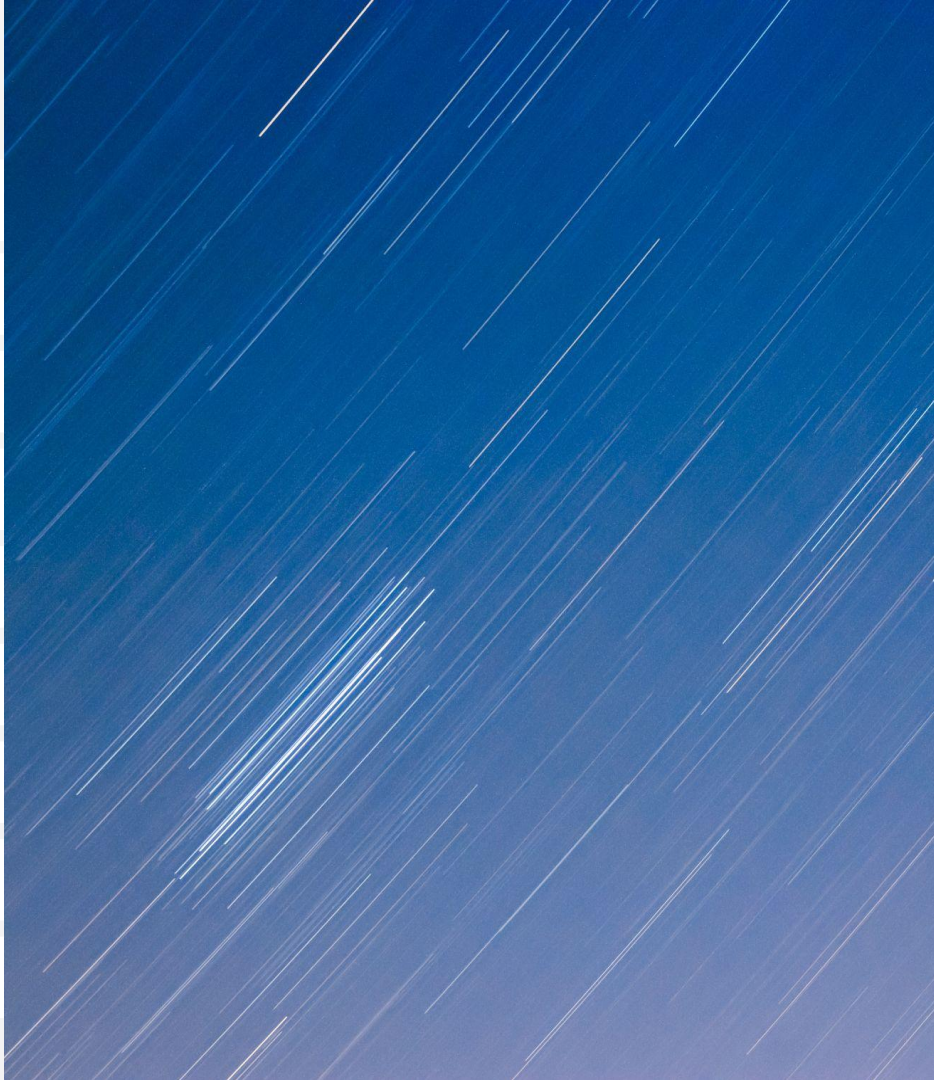
5 Historical forecasting and backtesting

6 Covariates

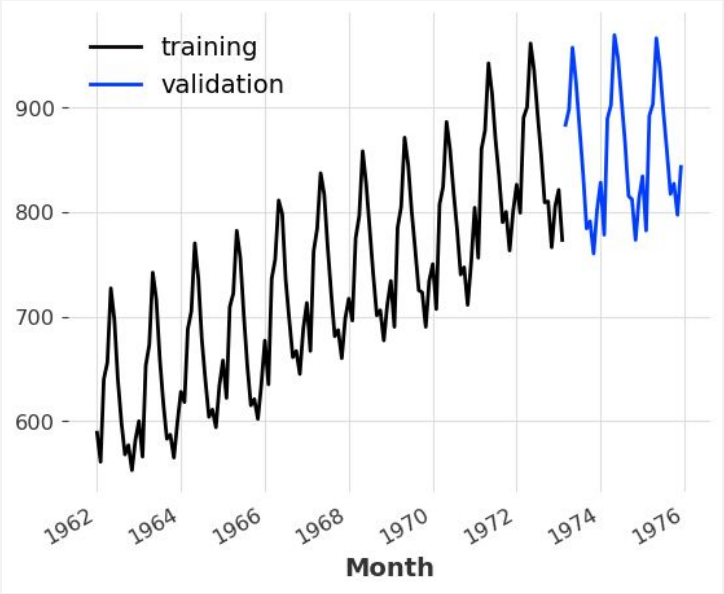
7 Probabilistic Forecasting

8 Training / predicting with multiple series

9 Summary and what's coming next

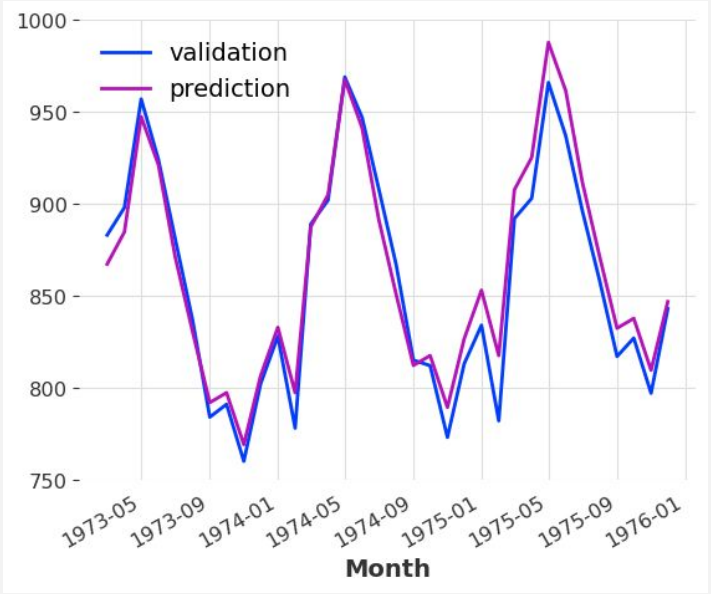


Goal



Accurate Forecasts

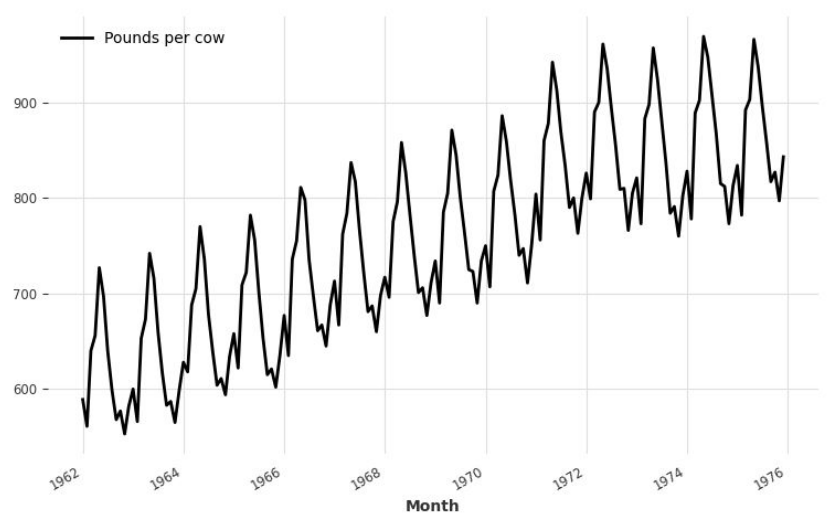
→



TimeSeries | Processing | Forecasting | Evaluating

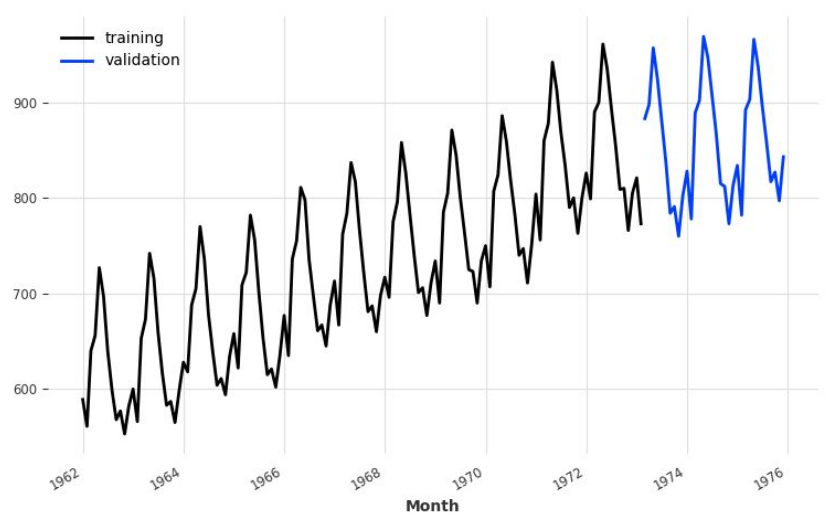
The TimeSeries object

```python
from darts import TimeSeries

# create time series
series = TimeSeries.from_csv('milk.csv', time_col='Month')
series.plot()
```



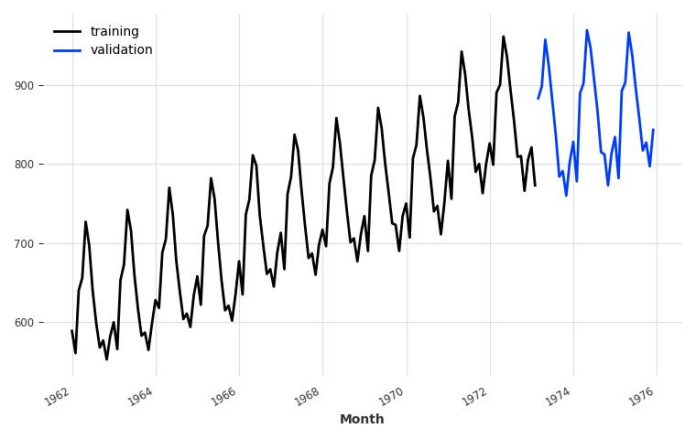
Training / validation split

```python
from darts import TimeSeries

# create time series
series = TimeSeries.from_csv('milk.csv', time_col='Month')

# split train/val
train, test = series.split_after(0.8)
```

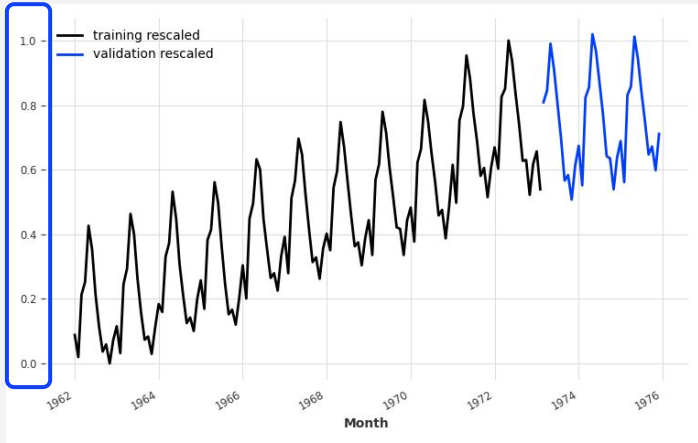


Normalizing TimeSeries

```python
from darts.dataprocessing.transformers import Scaler

scaler = Scaler()
scaler.fit(series)  # learn min and max
rescaled = scaler.transform(series)
```



Normalizing

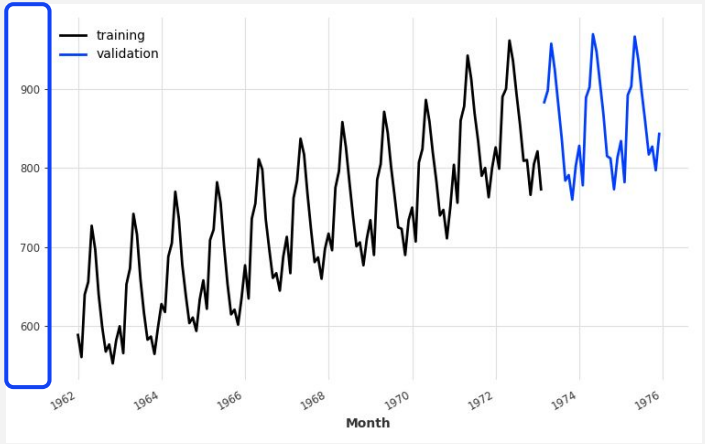


Unit8

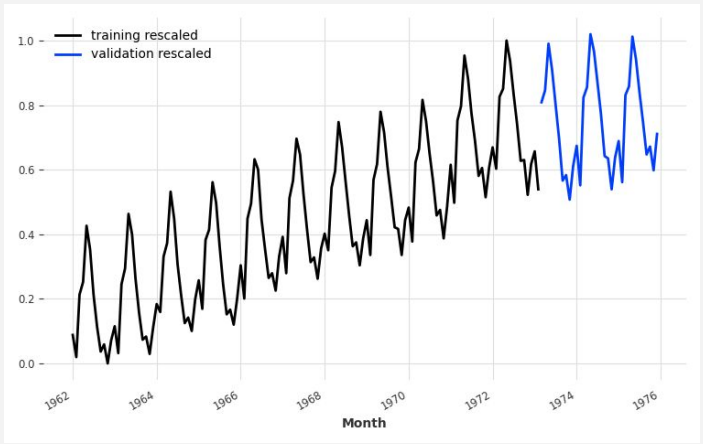
TimeSeries | **Processing** | Forecasting | Evaluating

Normalizing TimeSeries

```
original_series = scaler.inverse_transform(rescaled)
```



Re-scaling back

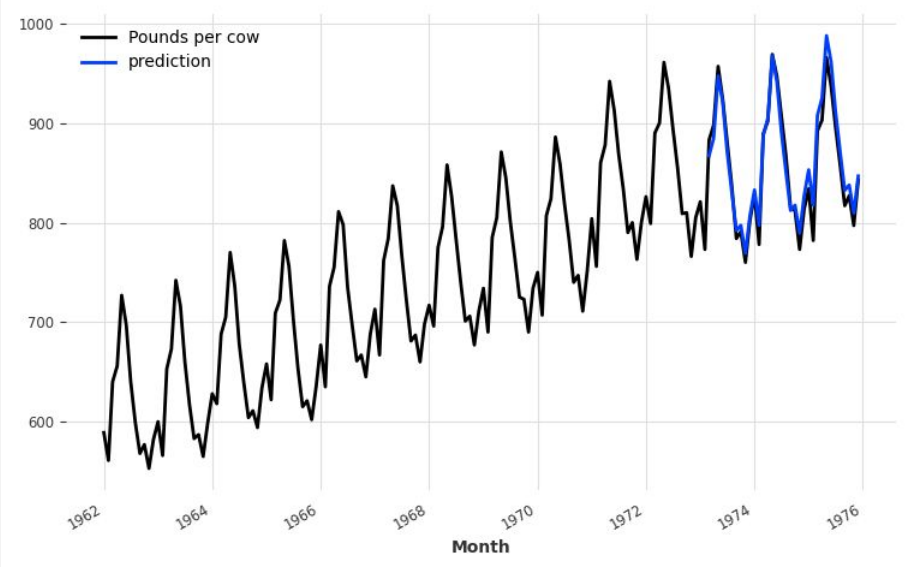


Unit8

TimeSeries | **Processing** | Forecasting | Evaluating

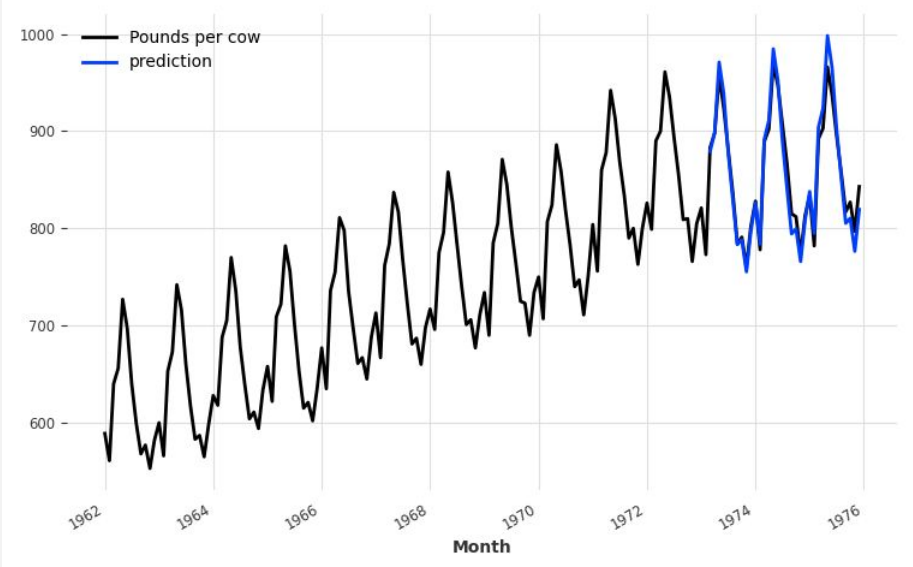
Forecasting – Exponential Smoothing

```python
from darts.models import ExponentialSmoothing

model = ExponentialSmoothing()
model.fit(train)
pred = model.predict(n=len(test))
```



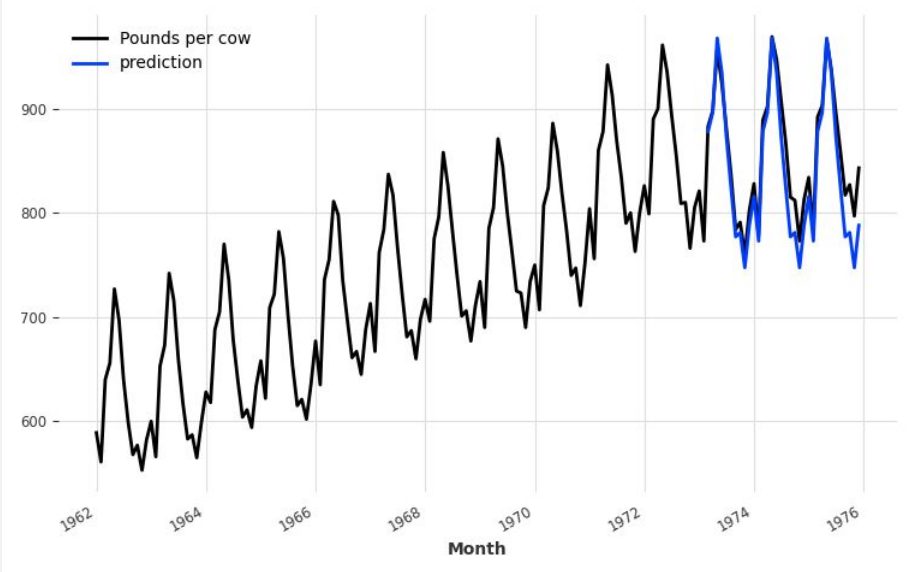
Forecasting – Theta

```
from darts.models import Theta

model = Theta()
model.fit(train)
pred = model.predict(n=len(test))
```



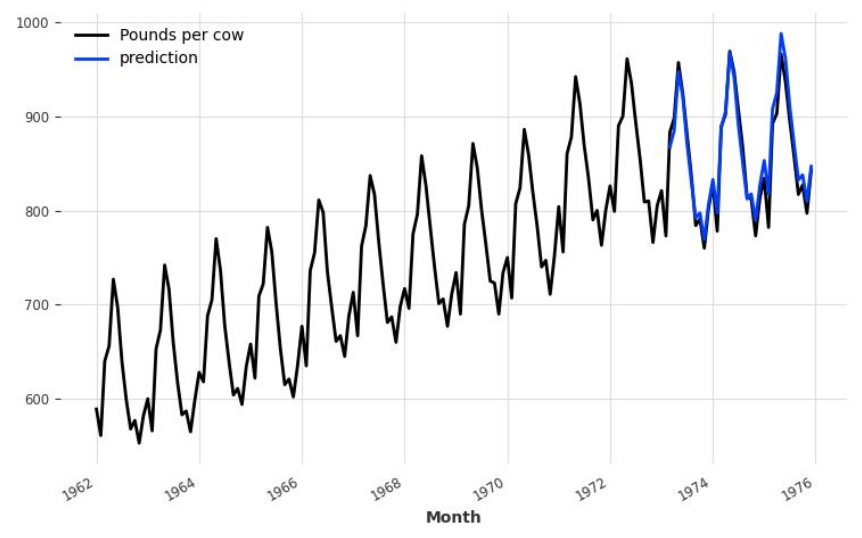
Specifying model parameters

```
from darts.models import Theta

model = Theta(theta=1)
model.fit(train)
pred = model.predict(n=len(test))
```

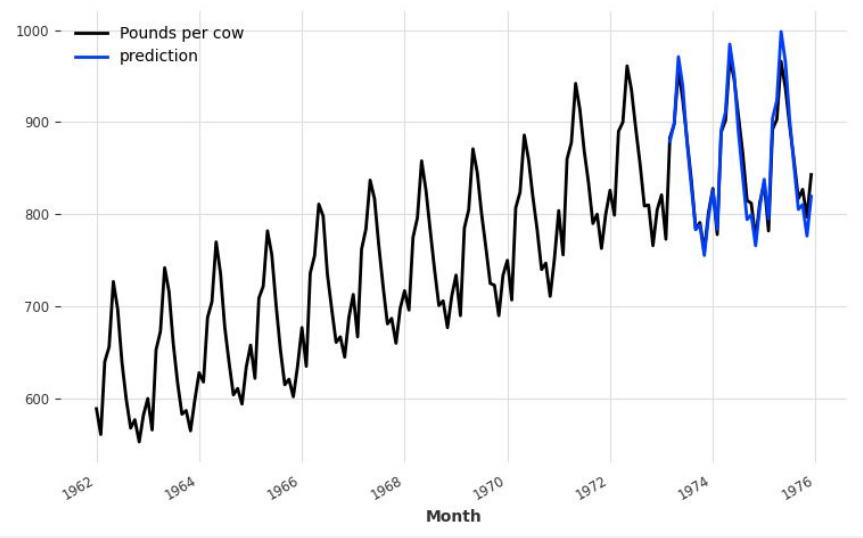


Evaluating predictions – Which one is better?

Exponential Smoothing



Theta



TimeSeries | Processing | Forecasting | **Evaluating**

Metrics

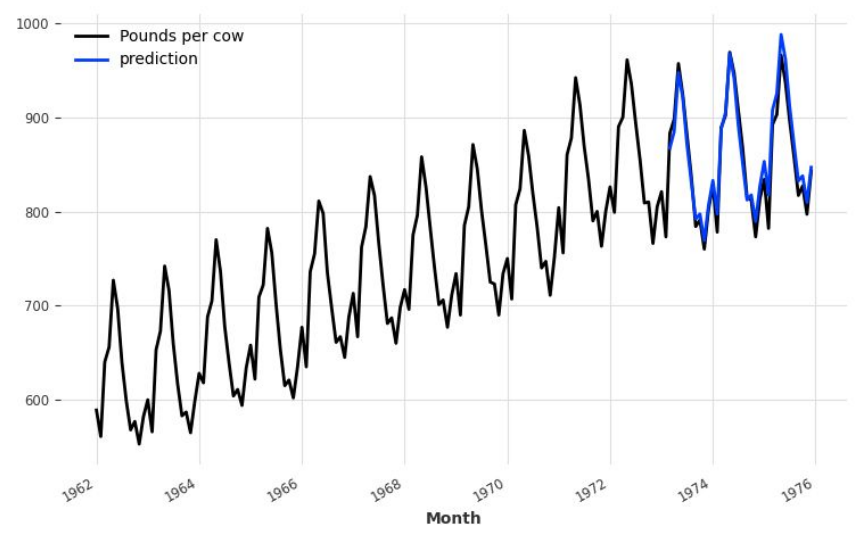
Many different scores can be computed – Darts lets you import the ones you need.

```python
from darts.metrics import mape

score_mape = mape(test, pred)
```

```python
from darts.metrics import smape

score_smape = smape(test, pred)
```

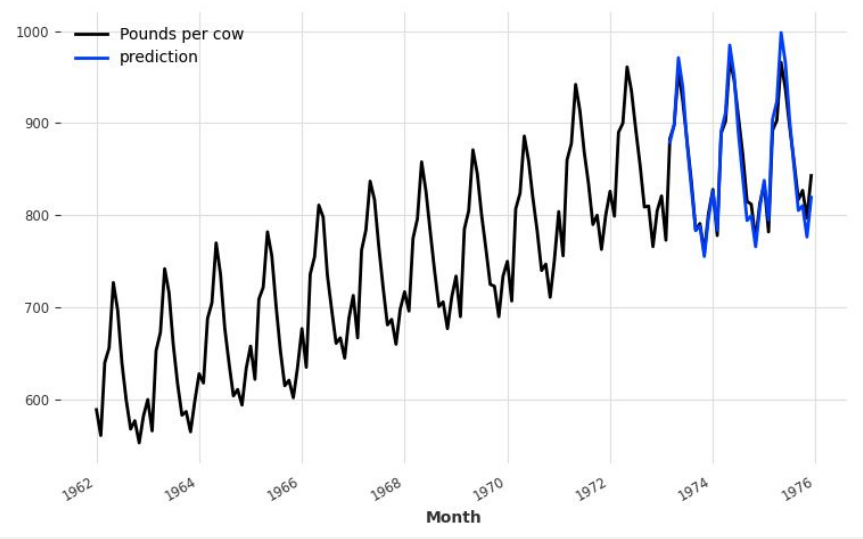
Which one is better?

Exponential Smoothing



sMAPE: ~1.39%

Theta



sMAPE: ~1.28%

TimeSeries → Processing → Forecasting → Evaluating

Recap

End-to-end example

```python
from darts import TimeSeries
from darts.dataprocessing.transformers import Scaler
from darts.metrics import smape
from darts.models import Theta

horizon = 12

# read searies, split train/val/test/...
series = TimeSeries.from_xxx(...)
train, test = series[:-horizon], series[-horizon:]

# scale train/val/...
scaler = Scaler()
train = scaler.fit_transform(train)
val = scaler.transform(test)

# build model with some parameters
model = Theta(...)

# train model
model.fit(train)

# get a prediction
pred: TimeSeries = model.predict(n=horizon)

# compute metrics
smape_test = smape(test, pred)
```

Unit8

Agenda

1 Intro to Darts

2 TimeSeries Class

3 Forecasting models

4 Simple forecasting example

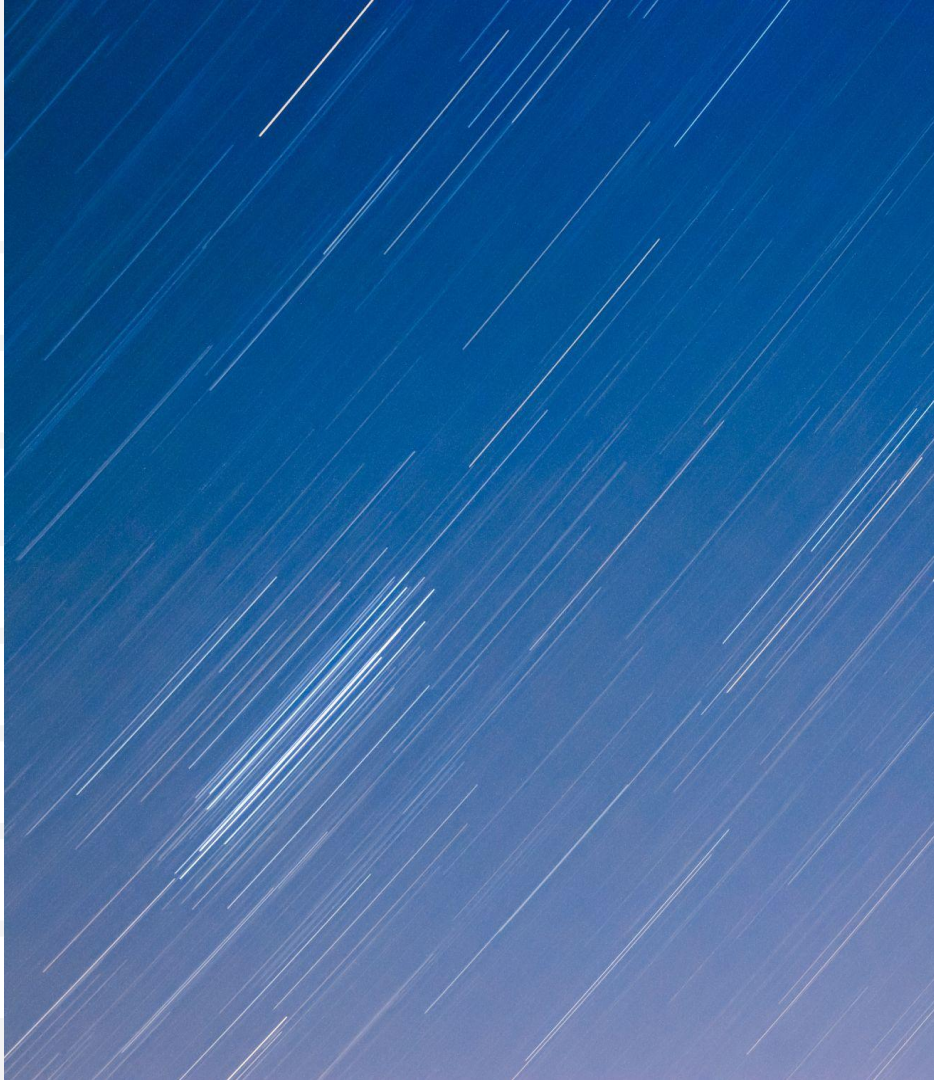
5 Historical forecasting and backtesting

6 Training / predicting with multiple series

7 Covariates

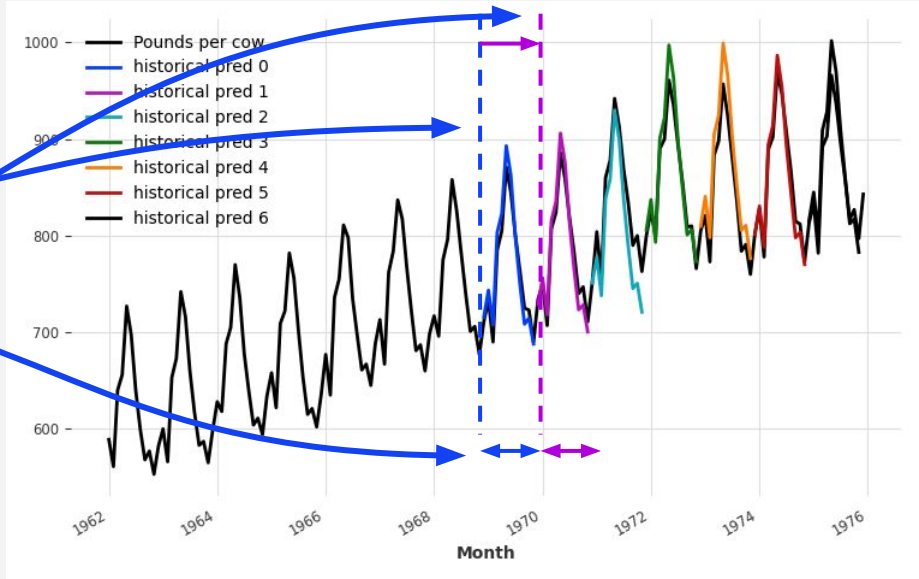
8 Probabilistic Forecasting

9 Summary and what's coming next



Historical simulations using Darts

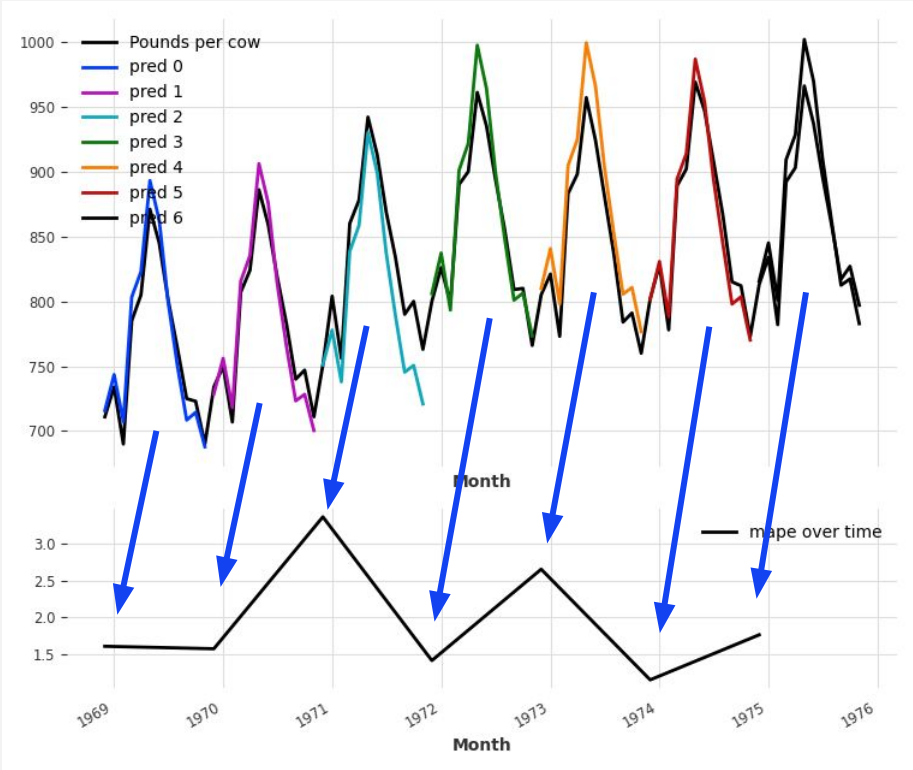
```
forecasts = model.historical_forecasts(
    series=series,
    start=0.5,
    forecast_horizon=12,
    stride=12,
    last_points_only=False,
)
```



Unit8

Backtesting models on historical data

```
backtest_errors = model.backtest(
    historical_forecasts=forecasts,
    series=series,
    metric=mape,
    reduction=None,  # np.mean, ...
)
```



Unit8

Agenda

1 Intro to Darts

2 TimeSeries Class

3 Forecasting models

4 Simple forecasting example

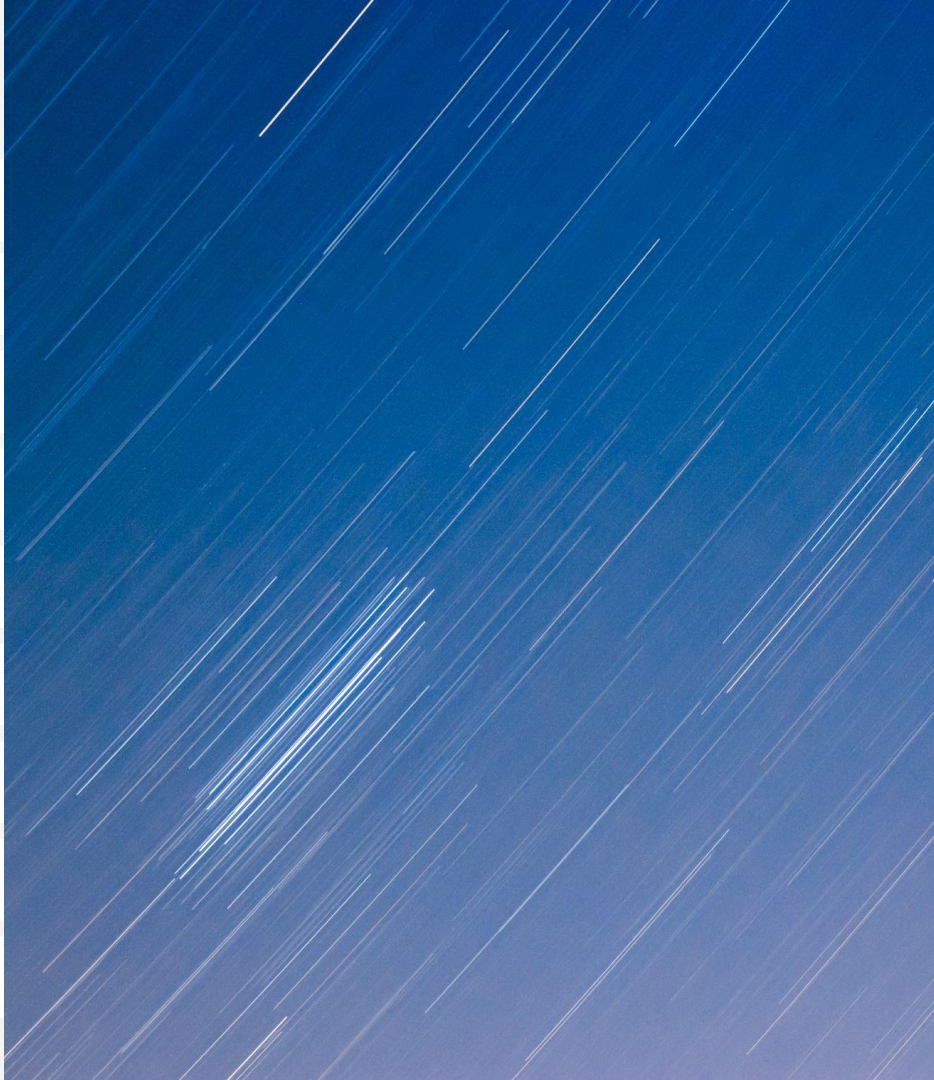
5 Historical forecasting and backtesting

6 Training / predicting with multiple series

7 Covariates

8 Probabilistic Forecasting

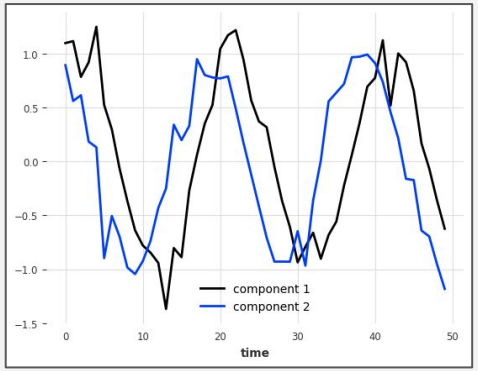
9 Summary and what's coming next



Multiple series means several uni- or multivariate series

In Darts, one `TimeSeries` can be:

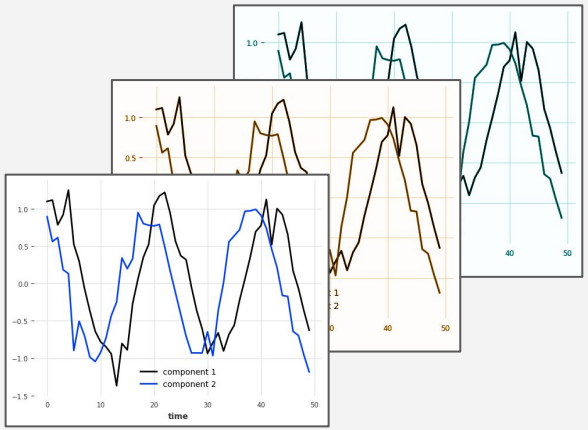
- Univariate: one column
- Multivariate: multiple columns



← one multivariate series with two columns

Multiple series are several `TimeSeries` that:

- must have the same number of columns
- can have different time axes, scales, trends, ...



← multiple multivariate series

Unit8

Back to the forecasting techniques

Local Forecasting Models

ARIMA/ VARIMA	Exponential Smoothing	Fast Fourier Transform
Theta	BATS/TBATS	Croston
Prophet	Baseline Models (Moving Average, Drift, Seasonal, ...)	...

- ➔ Can only be trained on a **single** target series
- ➔ Tend to be simpler statistical or naive models
- ➔ Can predict n -> infinity

Global Forecasting Models

Baseline Models
(Fixed & Moving Aggregate/Mean, Drift, Seasonal)

Regression Models
(scikit-learn, XGBoost, LightGBM, Catboost)

Neural Networks
(RNN, NBEATS/NHiTS, TCN, Transformer, TFT, NLinear/DLinear, TiDE)

Ensemble Models*
(Naive, Regression-based)

- ➔ Can be trained on **single** or **multiple** target series
- ➔ ML based-, and ensemble models
- ➔ Train on fixed-length sub-samples of input data

* depending on which forecasting models are ensembled

Unit8

In Practice

Fitting and forecasting on collections of series

Single Series

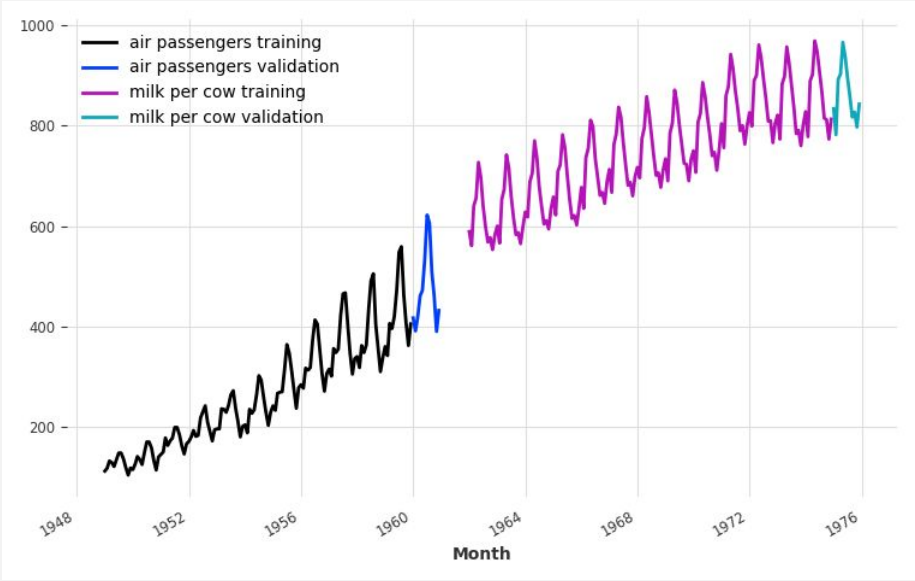
```python
from darts.models import SomeForecastingModel as Model

# create a series
series: TimeSeries = series1

model = Model(...)
model.fit(series)

# generate forecasts for a single series
preds: TimeSeries = model.predict(
    n=horizon,
)
```

Multiple Series

```python
from darts.models import SomeGlobalModel as Model

# create a sequence of series
series: Sequence[TimeSeries] = [series1, series2]

model = Model(...)
model.fit(series)

# generate forecasts for each `series`
preds: List[TimeSeries] = model.predict(
    n=horizon,
    series=series,  # or a completely new series
)
```

Looking at an example with Multiple Series

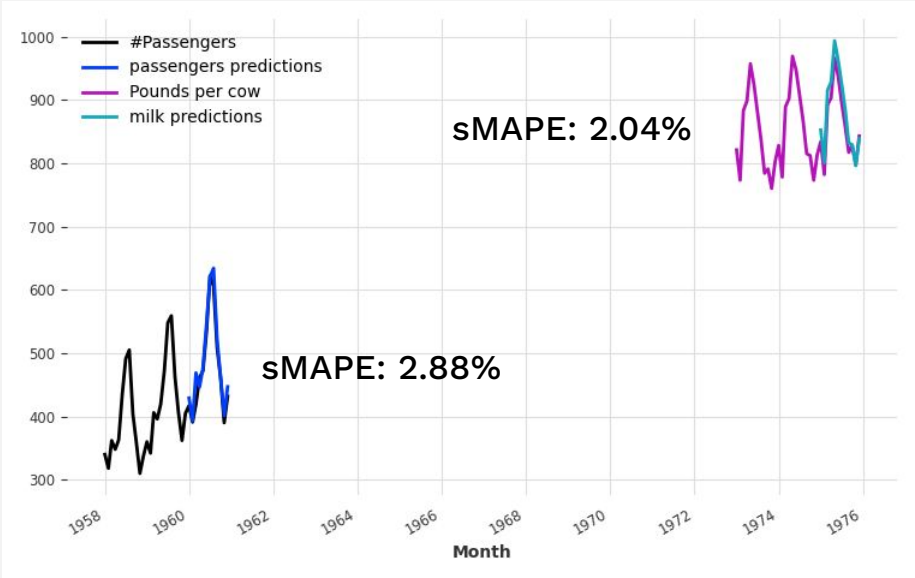
```python
from darts.datasets import (
  MonthlyMilkDataset,
  AirPassengersDataset
)

# load some of Darts datasets for experimentation
milk = MonthlyMilkDataset().load()
psng = AirPassengersDataset().load()

# create train and test sets with multiple series
horizon = 12
psng_train, psng_test = psng[:-horizon], psng[-horizon:]
milk_train, milk_test = milk[:-horizon], milk[-horizon:]

train = [psng_train, milk_train]
test = [psng_test, milk_test]
```



Unit8

Training and forecasting on Multiple Series with a global model

```python
# this is a neural network model
model = TiDEModel(**tide_params)
model.fit(train)

# predict after the end of each series in `train`
preds = model.predict(n=horizon, series=train)

# compute metrics for each forecast
score = smape(test, preds)
```

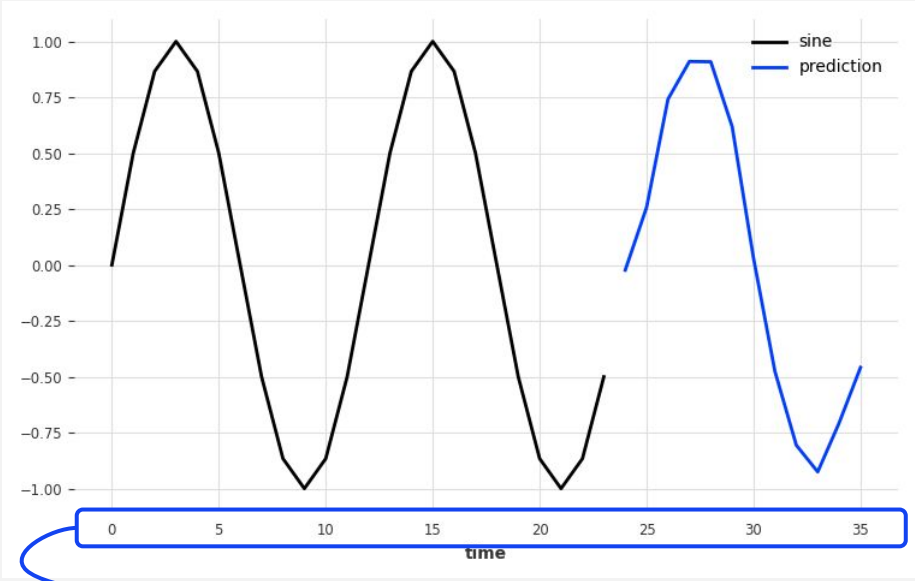



```python
tide_params = {
    "input_chunk_length": 24,  # length of historic input per training sample
    "output_chunk_length": 12,  # length of future output per training sample
    "random_state": 0,  # for reproducibility
    "use_reversible_instance_norm": True,
}
```

Unit8

Using the trained model to forecast an unobserved series

```python
from darts.utils.timeseries_generation import sine_timeseries

# generate a new sine curve series with similar seasonality
new_series = sine_timeseries(
  value_frequency=1/horizon,
  length=24,
  start=0  # give it an integer index
)
preds = model.predict(n=horizon, series=new_series)
```



The time index and frequency doesn't have to match the one it was trained on (before: monthly time index, now: integer index)

Unit8

Agenda

1 Intro to Darts

2 TimeSeries Class

3 Forecasting models

4 Simple forecasting example

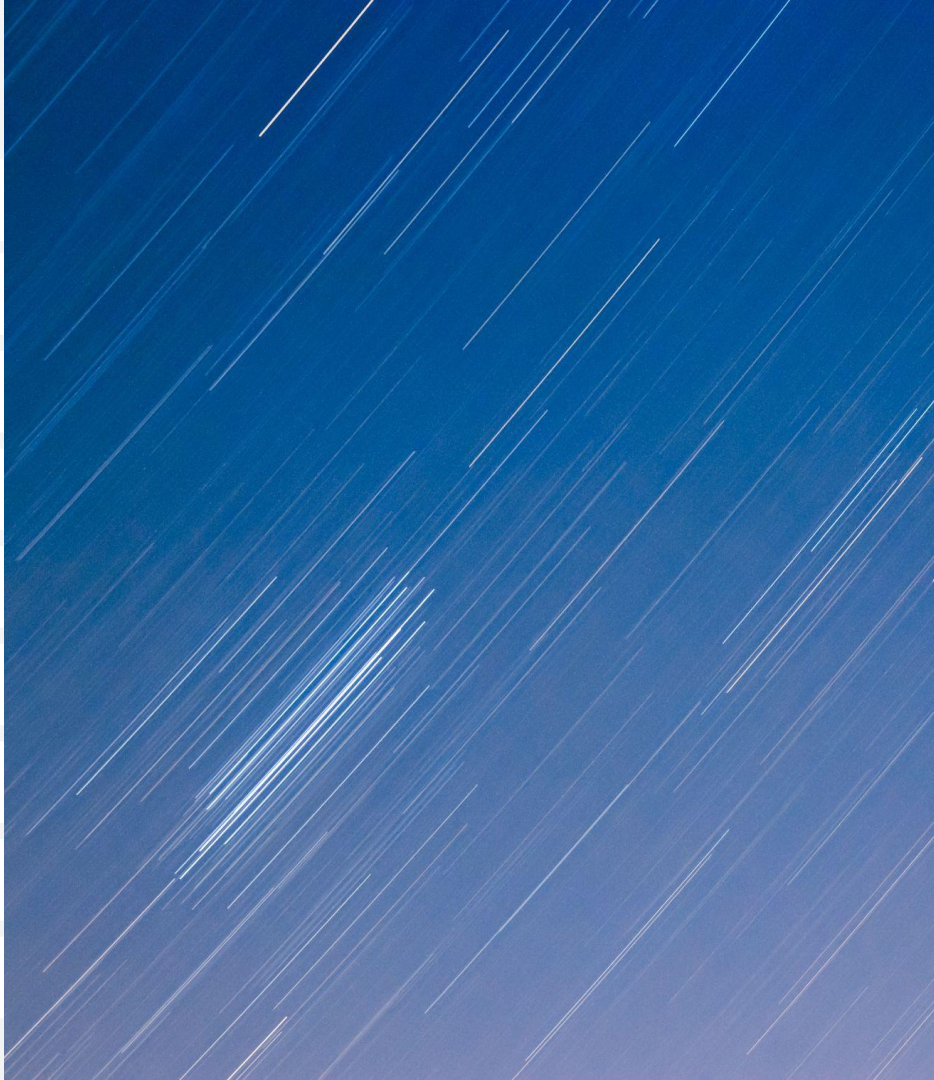
5 Historical forecasting and backtesting

6 Training / predicting with multiple series

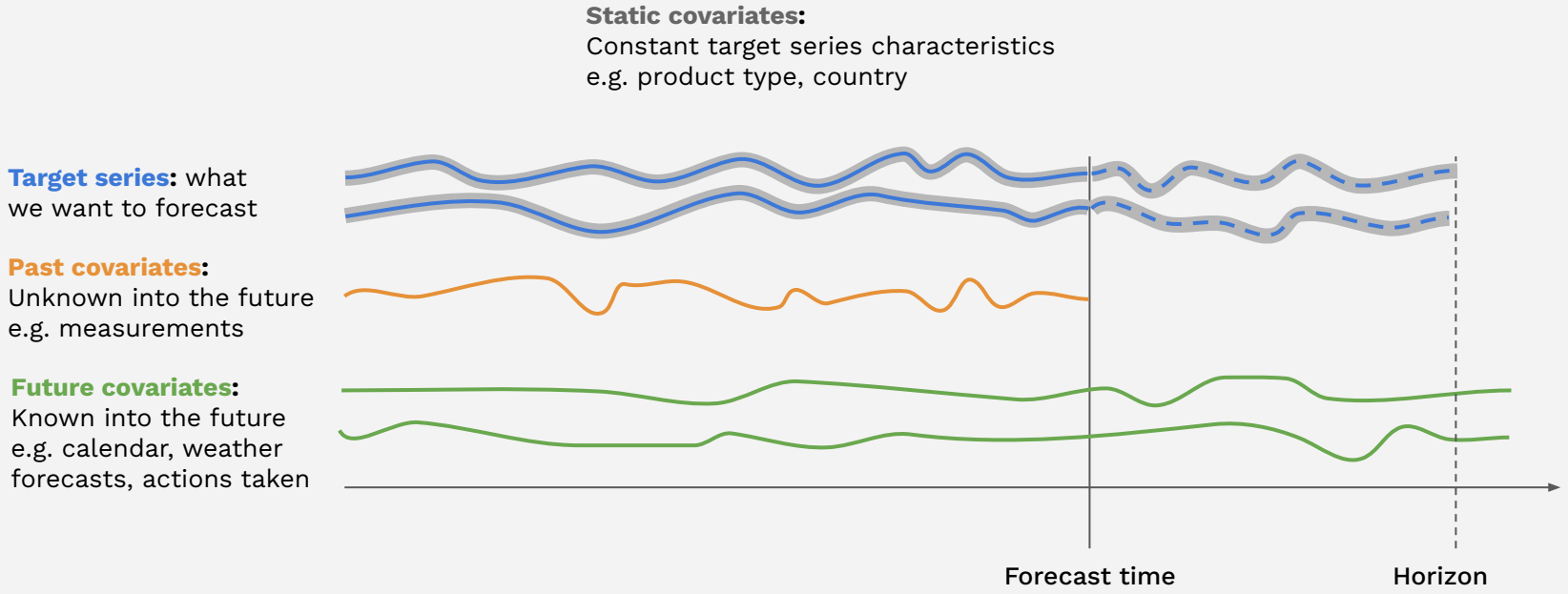
7 Covariates

8 Probabilistic Forecasting

9 Summary and what's coming next



Including Past, Future & Static External Data

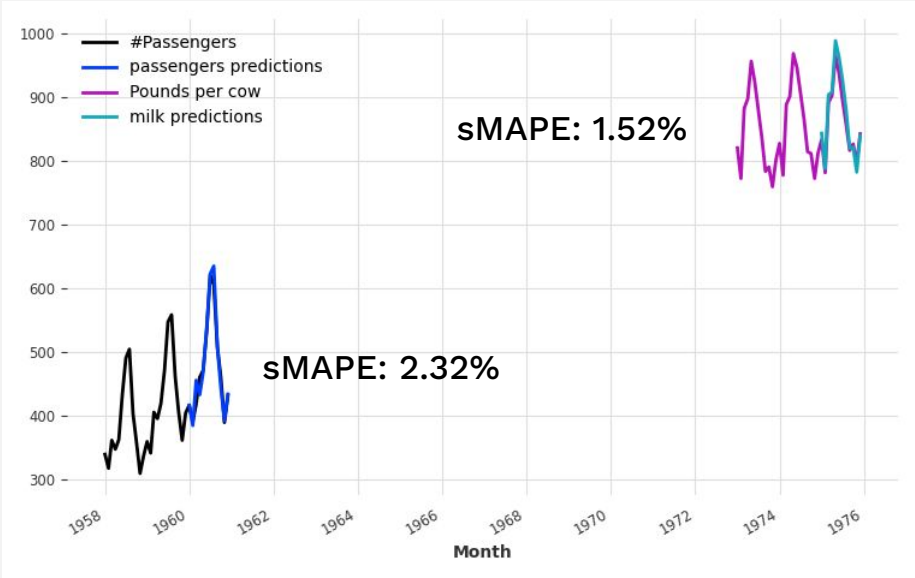


- `fit()` and `predict()` can accept `past_covariates` and/or `future_covariates`, depending on model
- `Static covariates` are embedded in the target series and can be used by some models
- Covariates can be uni- or multivariate
- Using **multiple target** series requires the same number of **past** / **future** **covariate series**
- Alignment of covariates with target is automatic

Unit8

Now we train and forecast using the covariates

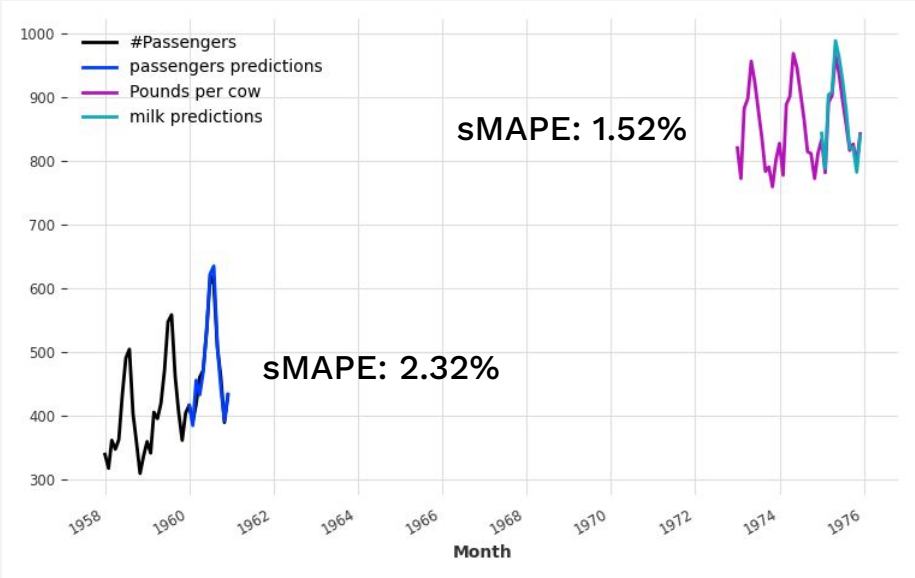
```python
# let the model use static information
model = TiDEModel(
    **tide_params,
    use_static_covariates=True
)
# pass the covariates to fit() & predict()
model.fit(
    train,
    future_covariates=covariates,
)

preds = model.predict(
    n=horizon,
    series=train,
    future_covariates=covariates,
)
```



Unit8

Or just let the model generate these time axis covariates

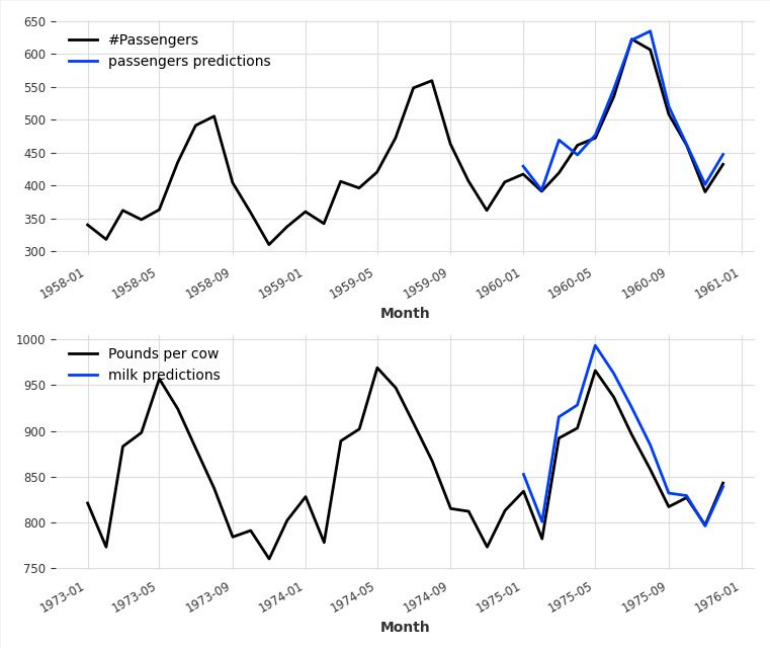
```python
# let the model generate the covariates
model = TiDEModel(
    **tide_params,
    use_static_covariates=True,
    add_encoders={"cyclic": {"future": ["month"]}}
)

model.fit(train)
preds = model.predict(n=horizon, series=train)
```



Unit8

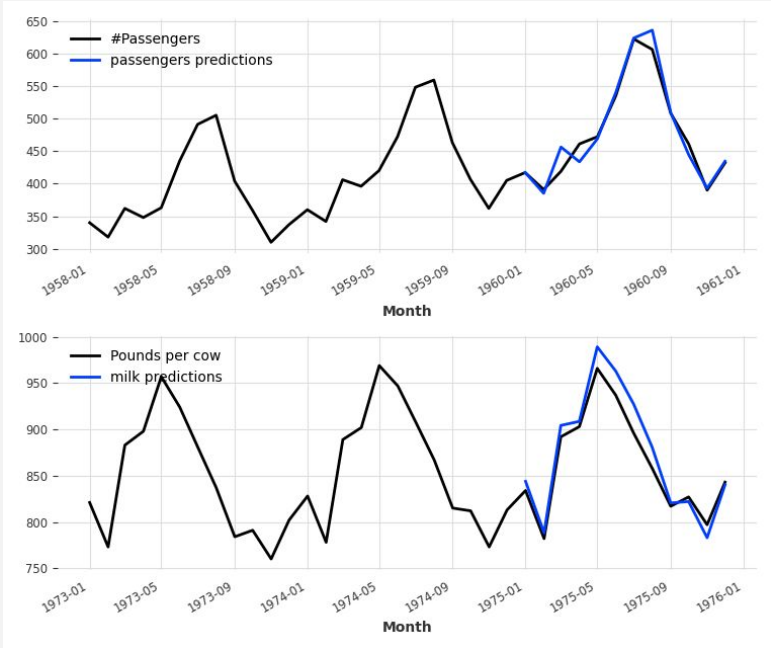
Which one is better?

TiDE - target only



mean sMAPE: 2.46%

TiDE - with covariates



meansMAPE: 1.92%

Agenda

1 Intro to Darts

2 TimeSeries Class

3 Forecasting models

4 Simple forecasting example

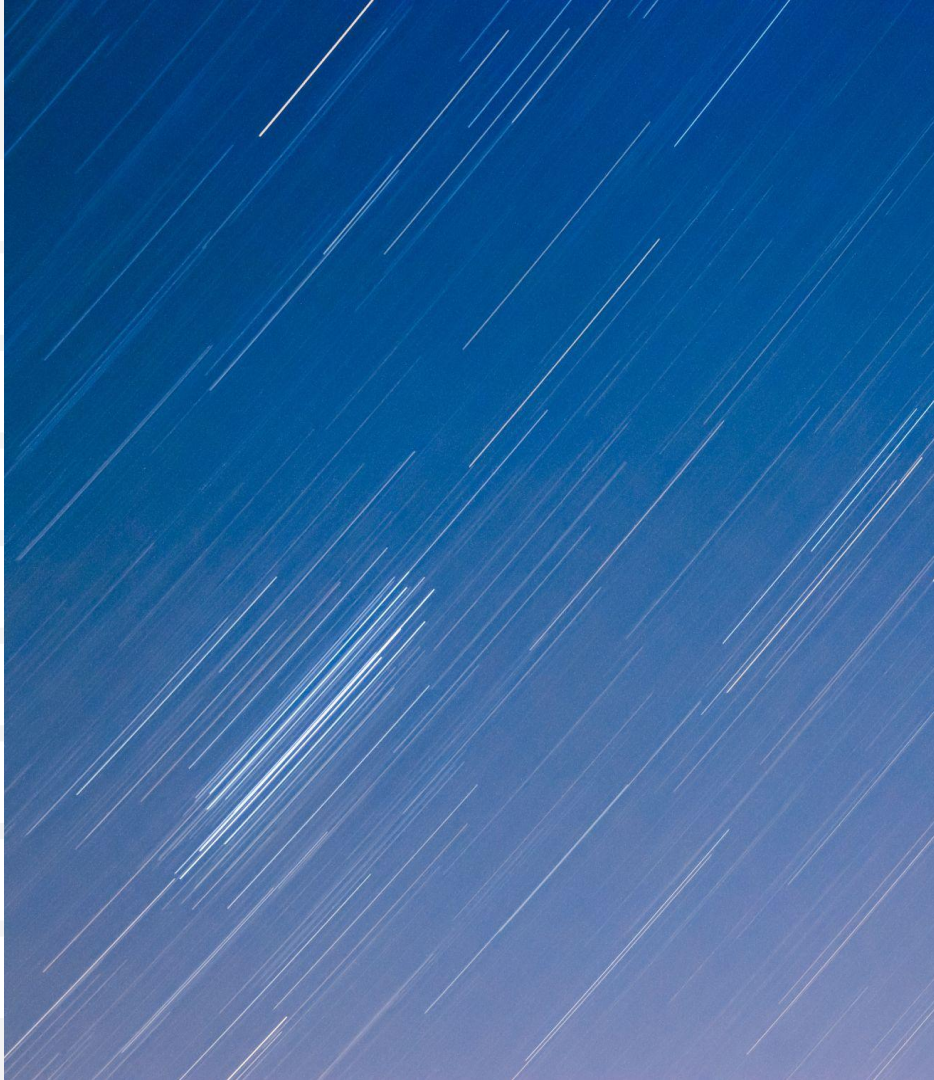
5 Historical forecasting and backtesting

6 Training / predicting with multiple series

7 Covariates

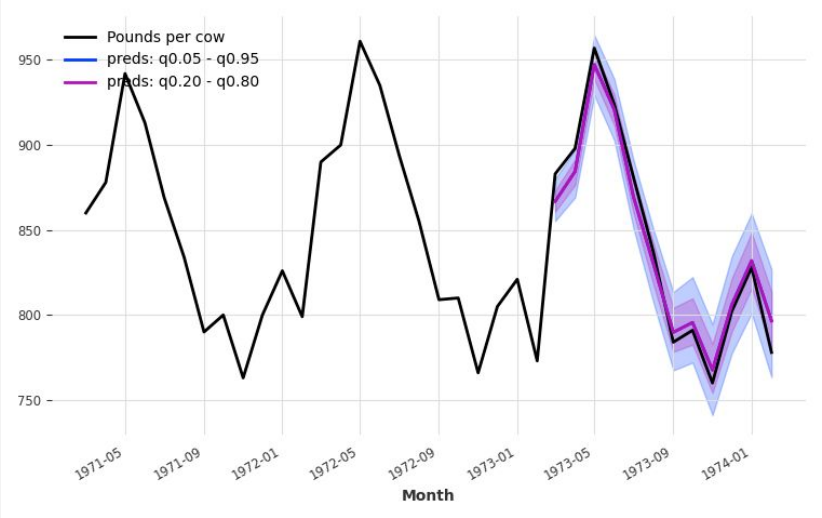
8 Probabilistic Forecasting

9 Summary and what's coming next



Probabilistic Forecasting with a Local Forecasting Model

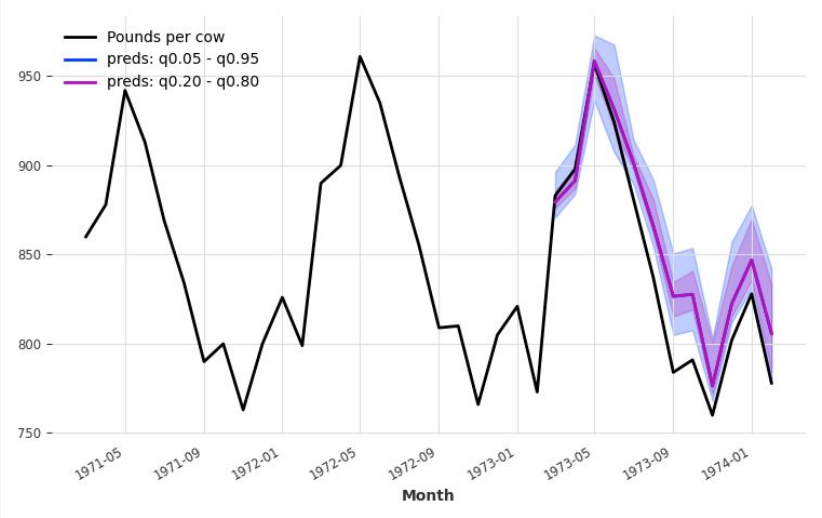
```python
model = ExponentialSmoothing()
model.fit(train)

# make probabilistic forecasts with num_samples >= 1
preds = model.predict(
    n=12,
    num_samples=500,
)

# plotting offers quantile ranges
preds.plot(label="preds: q0.05 - q0.95")
preds.plot(
    label="preds: q0.20 - q0.80",
    low_quantile=0.2,
    high_quantile=0.8
)
```



Unit8

Probabilistic Forecasting with a Global Forecasting Model

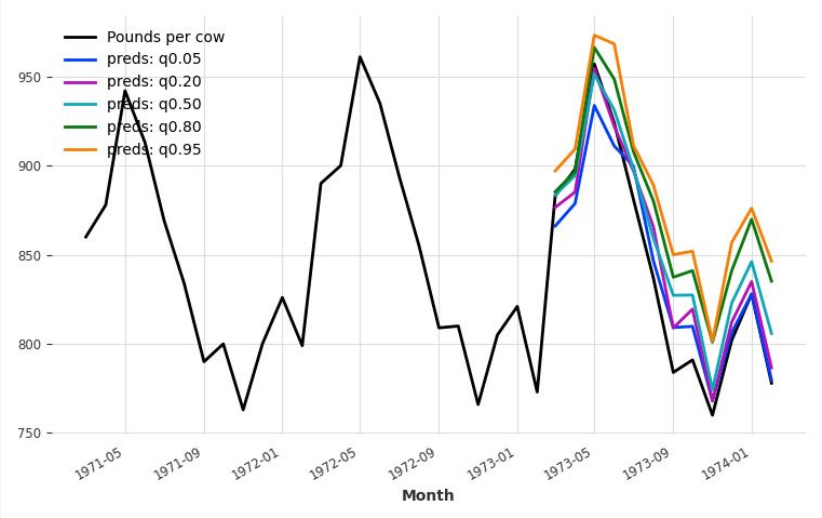
```python
from darts.utils.likelihood_models import QuantileRegression

# make global models probabilistic by adding a `likelihood`
model = TiDEModel(..., likelihood=QuantileRegression())
model.fit(train)

# make probabilistic forecasts with num_samples >= 1
preds = model.predict(
    n=12,
    num_samples=500,
)

# plotting offers quantile ranges
preds.plot(label="preds: q0.05 - q0.95")
preds.plot(
    label="preds: q0.20 - q0.80",
    low_quantile=0.2,
    high_quantile=0.8
)
```



Unit8

Likelihood Parameter Prediction with a Global Forecasting Model

```python
from darts.utils.likelihood_models import QuantileRegression

# make global models probabilistic by adding a `likelihood`
model = TiDEModel(..., likelihood=QuantileRegression())
model.fit(train)

# make probabilistic forecasts with num_samples >= 1
preds = model.predict(
    n=12,
    predict_likelihood_parameters=True,
)

# parameters are stored as columns -> plot them directly
preds.plot()
```



Unit8

Darts' Likelihood Models

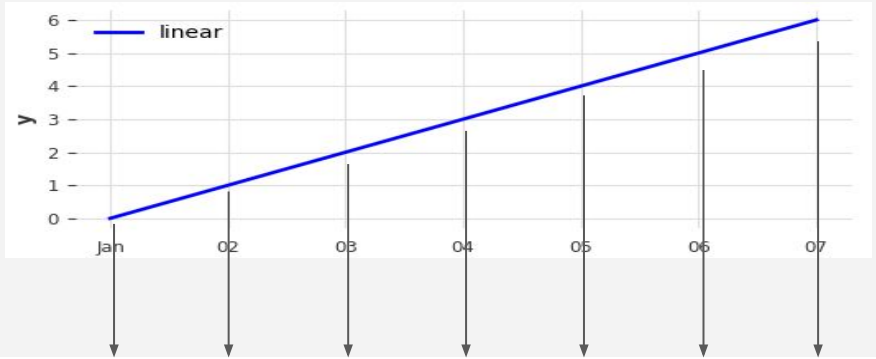
```
from darts.utils.likelihood_models import (
    BernoulliLikelihood,
    BetaLikelihood,
    CauchyLikelihood,
    ContinuousBernoulliLikelihood,
    DirichletLikelihood,
    ExponentialLikelihood,
    GammaLikelihood,
    GaussianLikelihood,
    GeometricLikelihood,
    HalfNormalLikelihood,
    LaplaceLikelihood,
    LogNormalLikelihood,
    NegativeBinomialLikelihood,
    PoissonLikelihood,
    QuantileRegression,
    WeibullLikelihood,
)
```

\+ Priors on distributions' parameters

Unit8

Bonus - Regression Models and Lagged Features

```python
from darts import RegressionModel

# create time series
model = RegressionModel(lags=4, output_chunk_length=1)
```



input series	t_i	t_i+1	t_i+2	t_i+3	t_i+4	t_i+5	t_i+6
	0	1	2	3	4	5	5

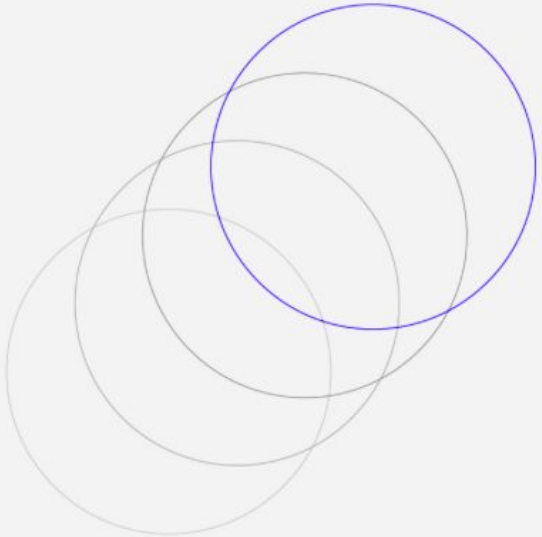
sample extraction							
0	0	1	2	3	4	5	6
1	0	1	2	3	4	5	6
2	0	1	2	3	4	5	6

resulting arrays	X				y
	0	1	2	3	4
	1	2	3	4	5
	2	3	4	5	6

X y

What else exists?

- Anomaly detection
- Explainability (shap for regression models, dedicated explainers for selected neural networks)
- Hierarchical reconciliation
- Filtering
- Dynamic Time Warping
- PyTorch Lightning (multi GPU support, TPU support, callbacks, checkpointing, monitoring, optimizer selection, learning rate scheduling, custom loss functions, ...)
- ...



Agenda

1 Intro to Darts

2 TimeSeries Class

3 Forecasting models

4 Simple forecasting example

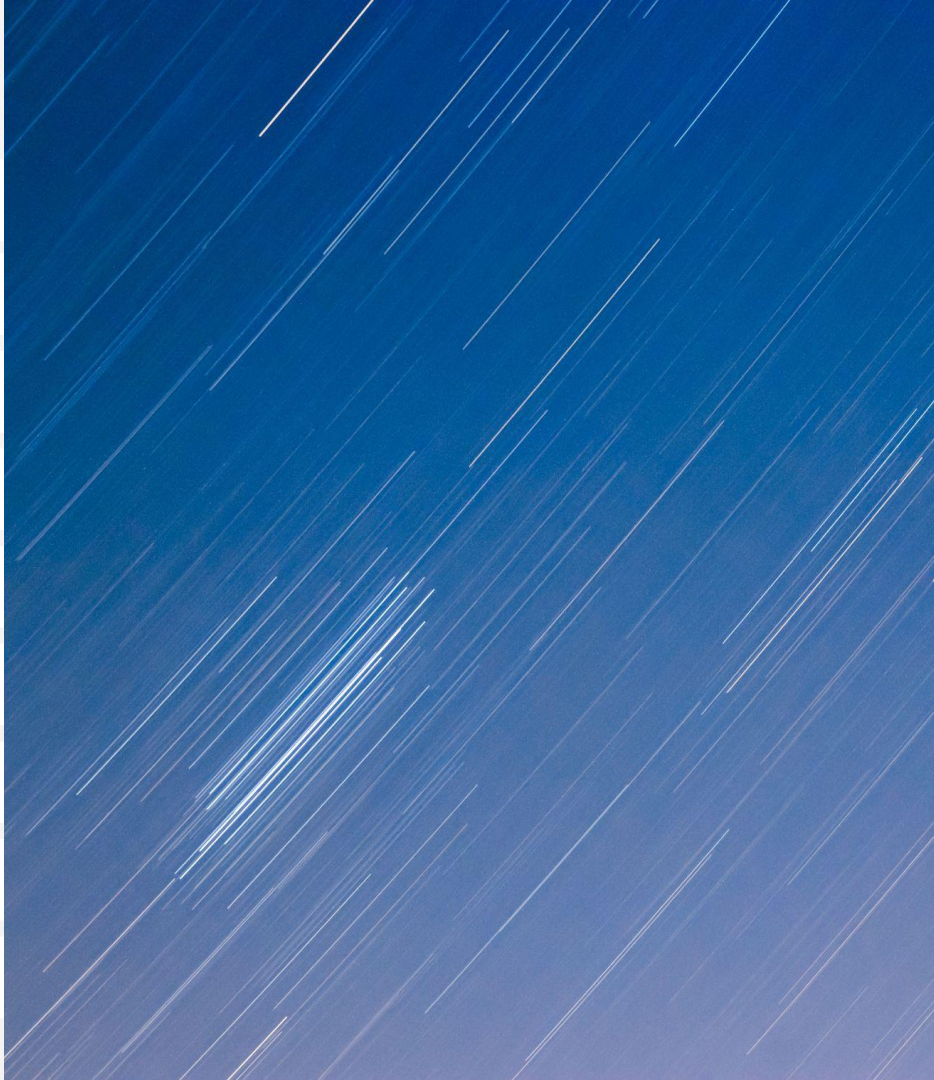
5 Historical forecasting and backtesting

6 Training / predicting with multiple series

7 Covariates

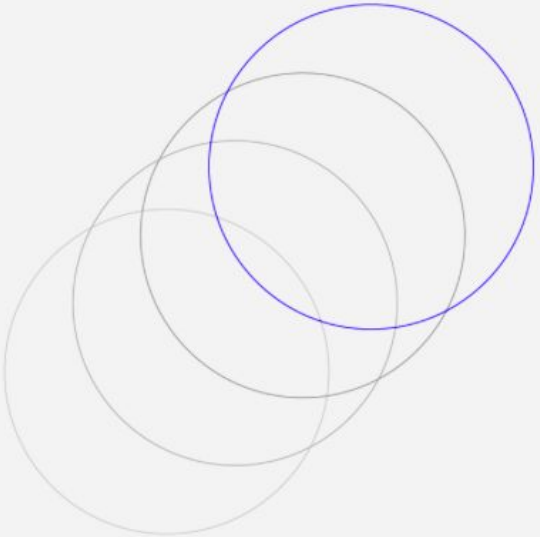
8 Probabilistic Forecasting

9 Summary and what's coming next



Take home messages

- Darts is a tool that simplifies time series manipulation and forecasting
- In just a few lines of code, we can use and compare different models
- The unified API allows to use neural network based models in the same way as simpler models
- External data can help to significantly improve forecasts
- There is more than one way of how to forecast (deterministic, probabilistic, global learning on multiple series (the future?))



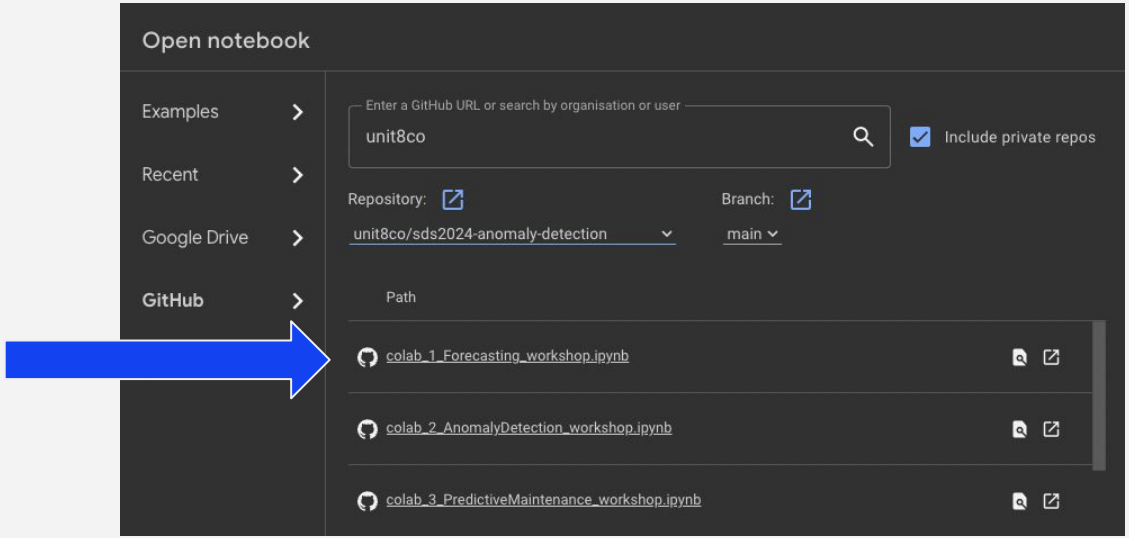
What’s planned in the near future?

- Conformal Prediction: distribution-free probabilistic forecasting
- Time series Classification

Time To Work

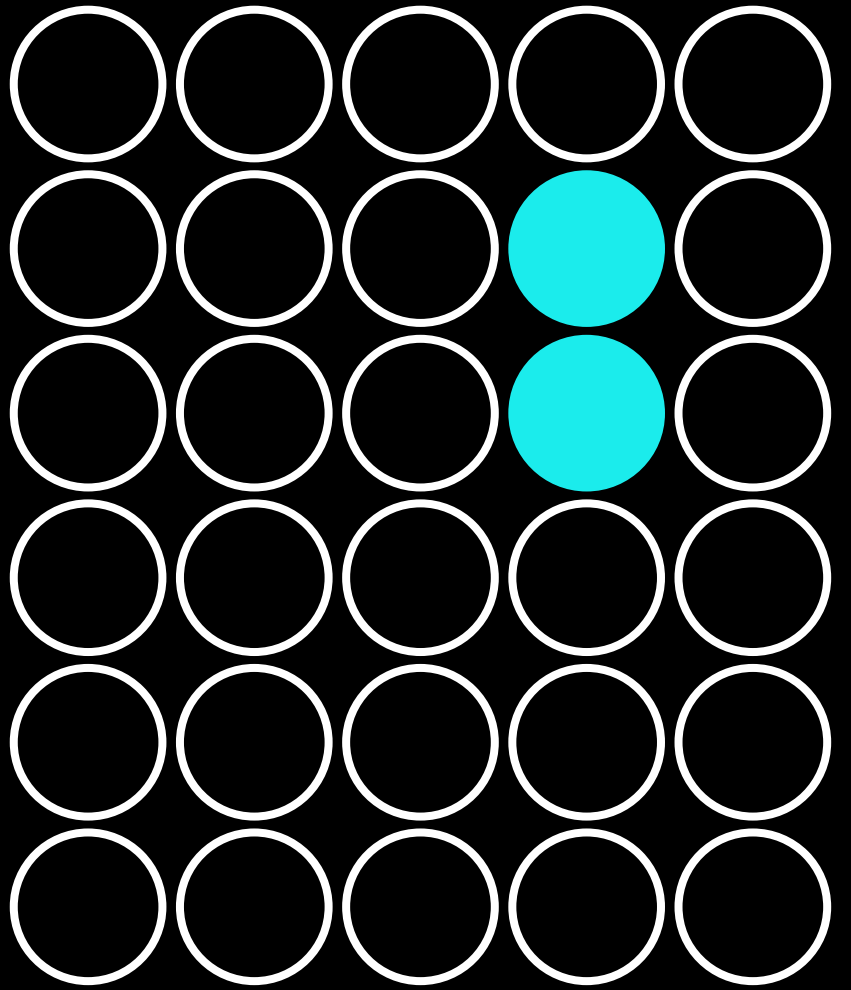
Darts

- Open: https://colab.google/
- Click on “Open Colab”
- On the left sidebar click GitHub
  - Enter GitHub URL: “unit8co”
  - Select repository: “unit8co/sds2024-anomaly-detection”
  - Select notebook: “colab_1_Forecasting_workshop.ipynb”



Unit8

unit8.com



thank you!