Unit8™

Darts

Unifying time series forecasting models from ARIMA to Deep Learning

digital natives

unit8.co

Francesco

- Data Scientist @ Unit8
- One of the main contributors to Darts.

Gaël

- Data Scientist @ Unit8
- Experience working with time series in various industries such as telecom, manufacturing and energy

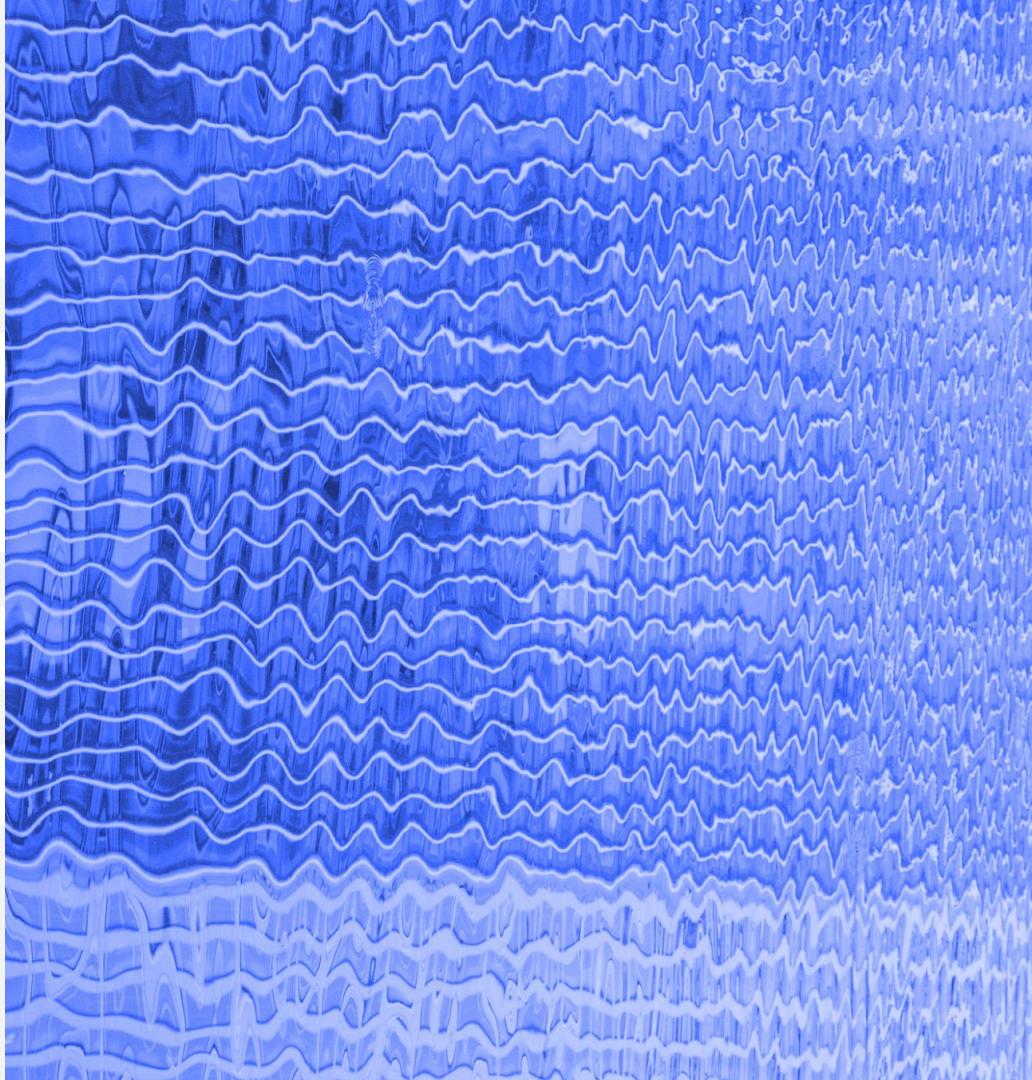
1 **Intro to Forecasting & Darts**

2 Forecasting using Darts

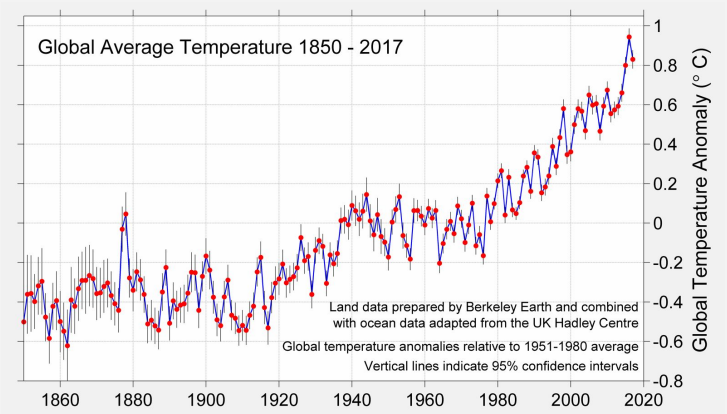
3 Training on multiple time-series

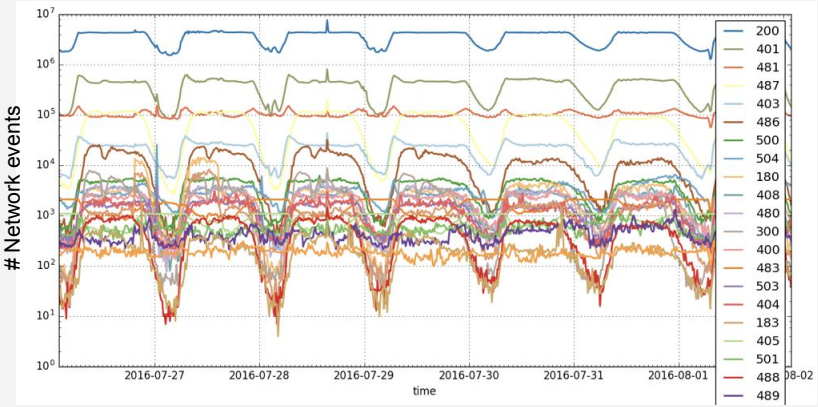
4 Probabilistic forecasting

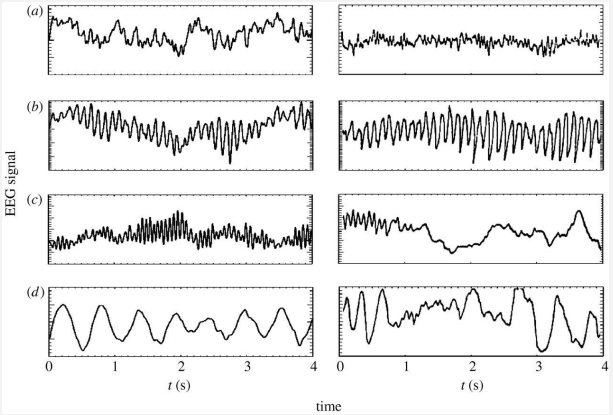
5 Try Darts!

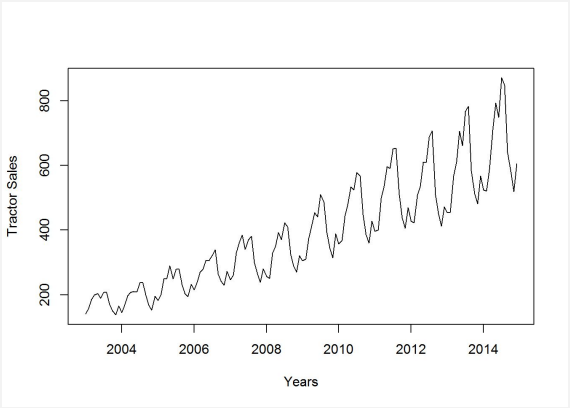


Time series are everywhere!

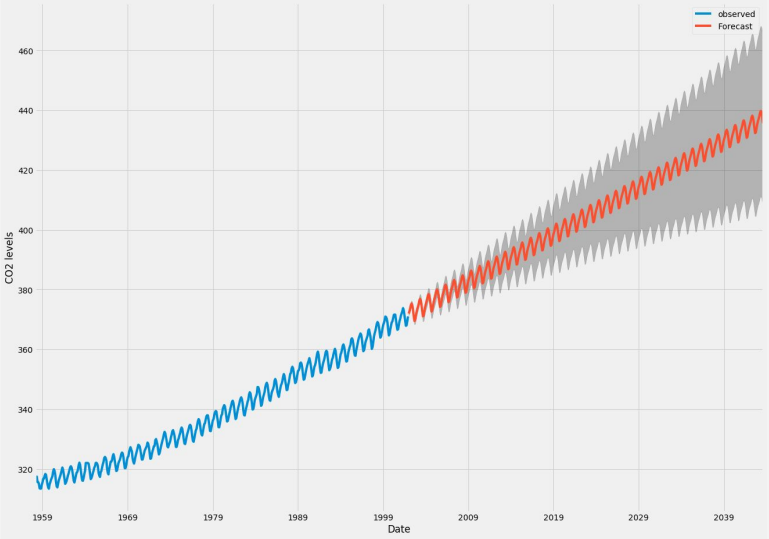


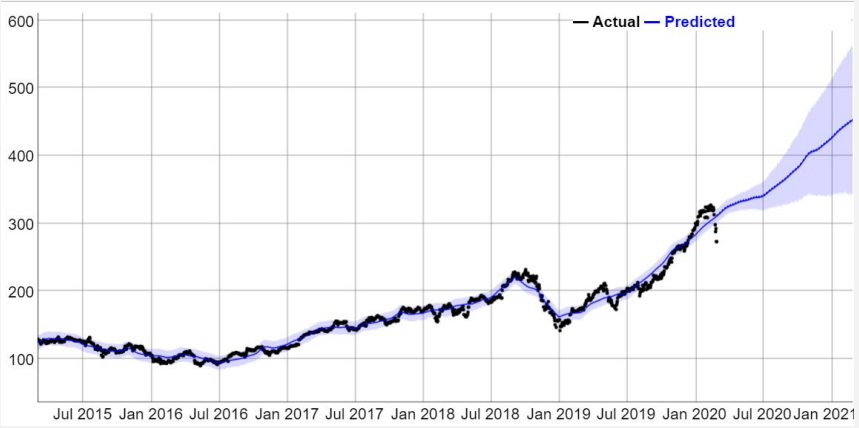






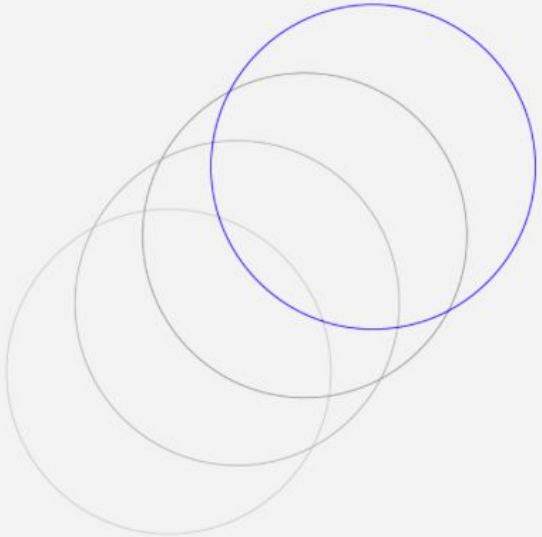
What if we could anticipate the future?



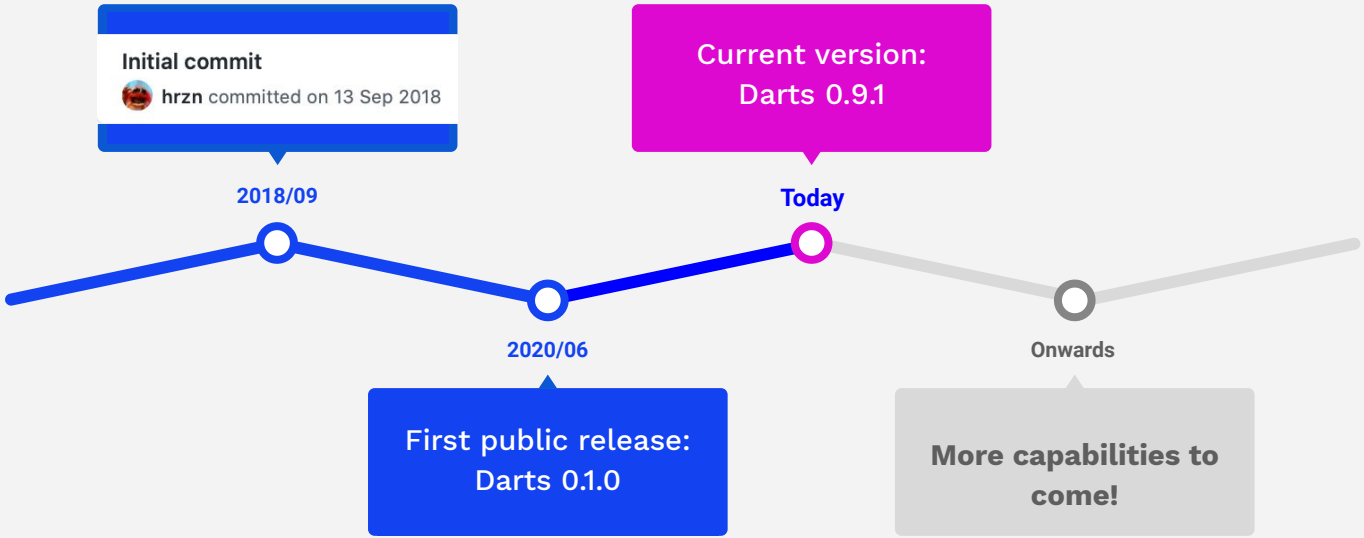


Why Darts?

- Lack of unified library in Python for time series forecasting
- To create a useful tool for ourselves



How did Darts come about?



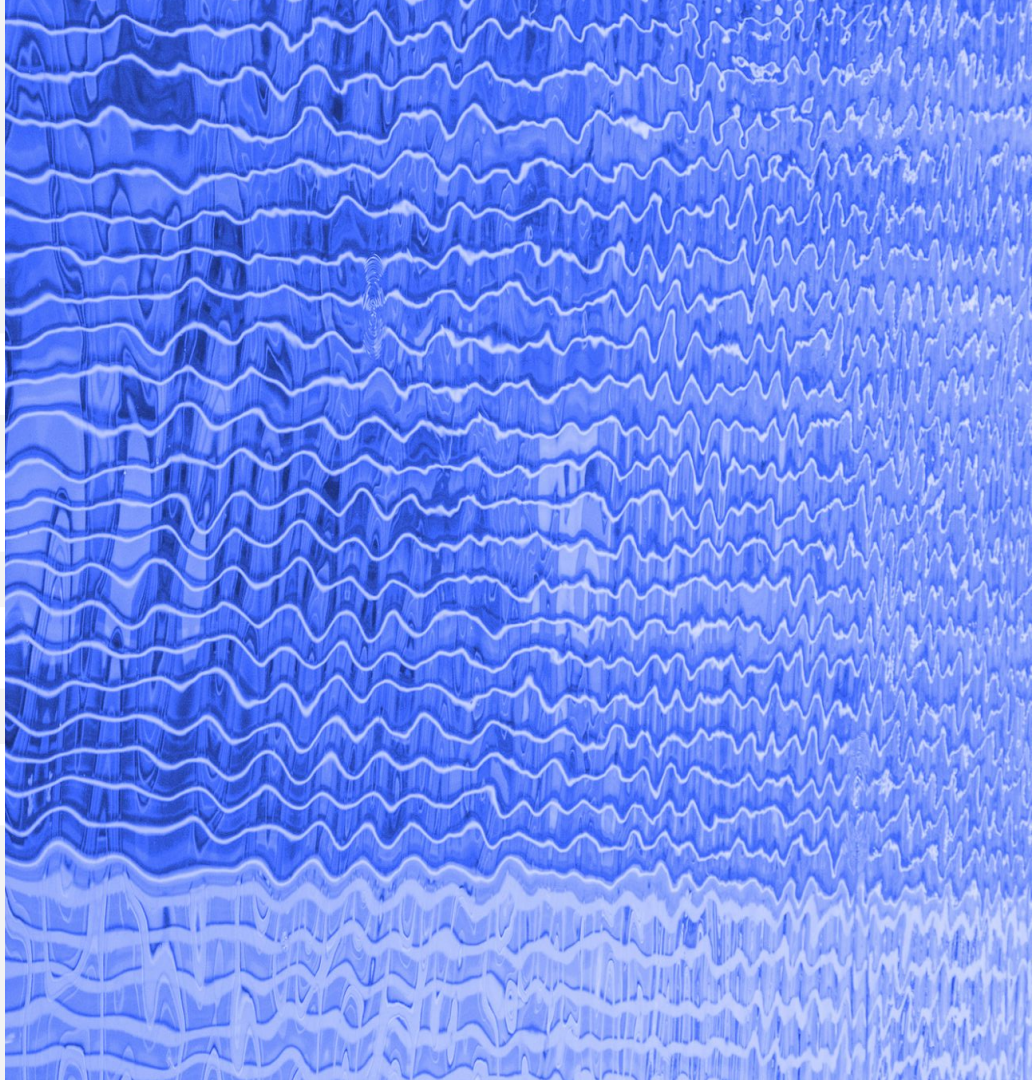
1 Intro to Forecasting & Darts

2 Forecasting using Darts

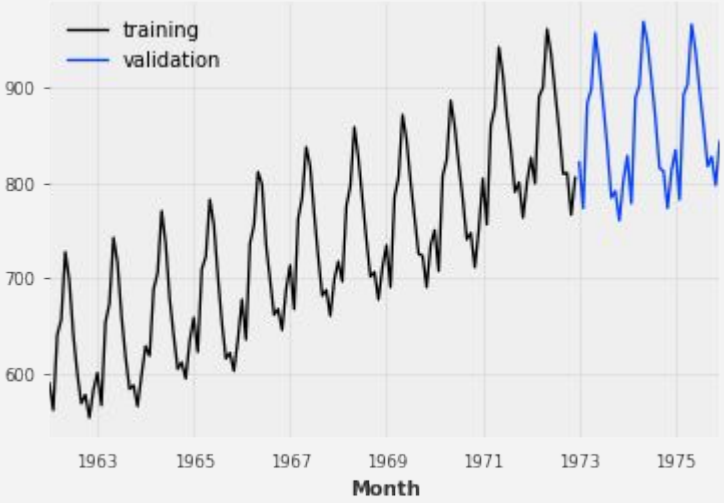
3 Training on multiple time-series

4 Probabilistic forecasting

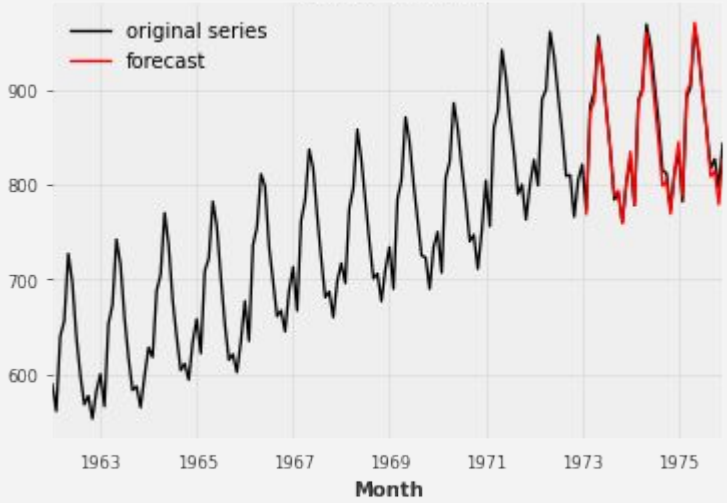
5 Try Darts!



Darts Overview



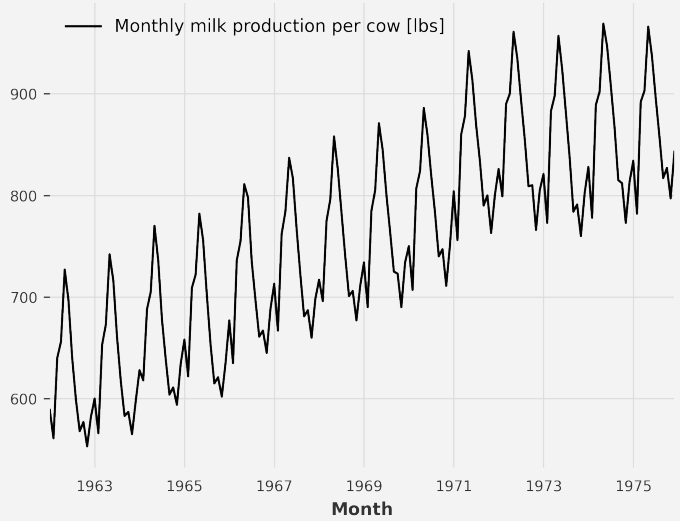
Goal



TimeSeries → Forecasting → Evaluating → Tuning

The TimeSeries object

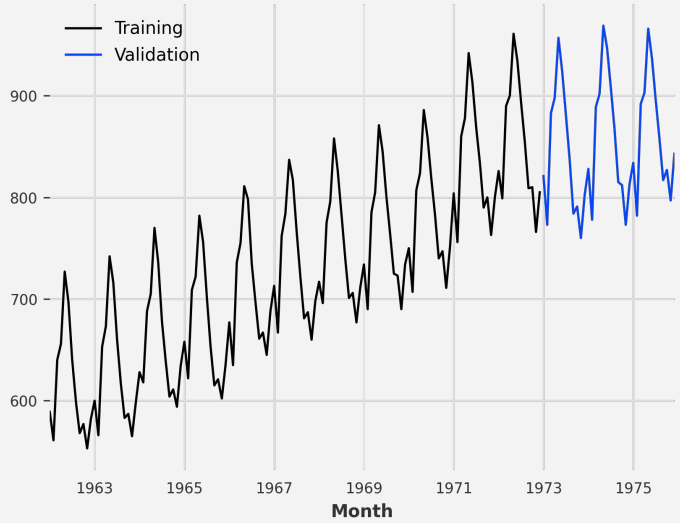
```python
from darts import TimeSeries

df = pd.read_csv("monthly-milk.csv")
series = TimeSeries.from_dataframe(
    df,
    'Month',
    value_cols=["Pounds per cow"]
)
```



TimeSeries | Forecasting | Evaluating | Tuning

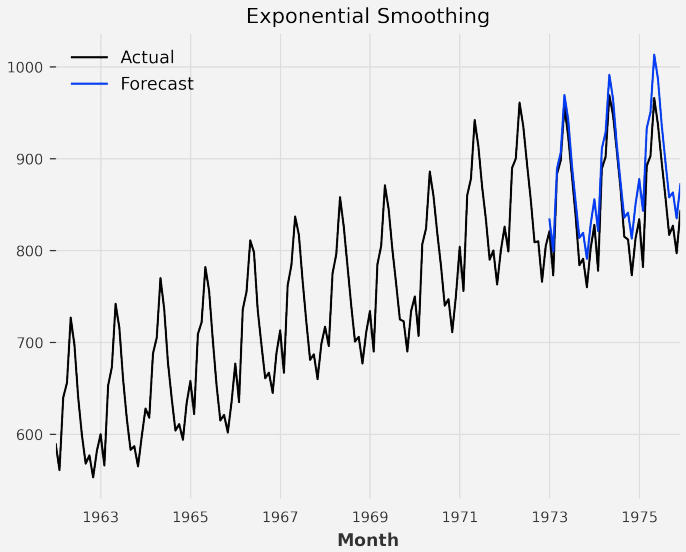
Training / validation split

```python
training, validation = (
    series
    .split_before(pd.Timestamp('1973-01-01'))
)
```



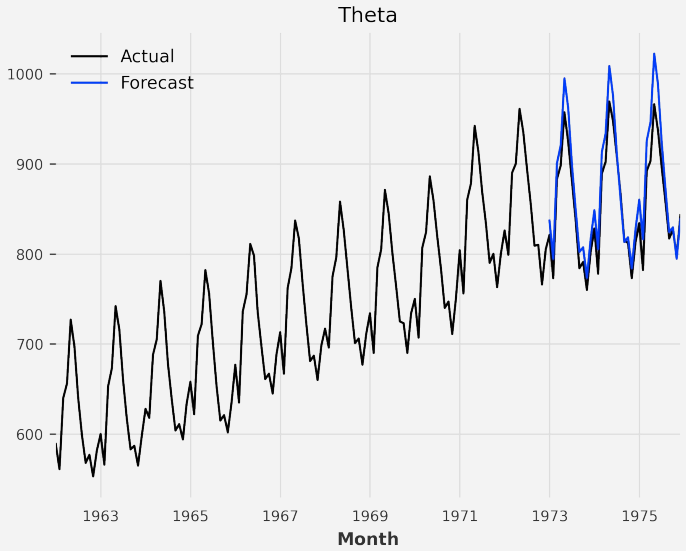
Forecasting – Exponential Smoothing

```python
from darts.models import ExponentialSmoothing

model = ExponentialSmoothing()
model.fit(training)
forecast = model.predict(len(validation))
```



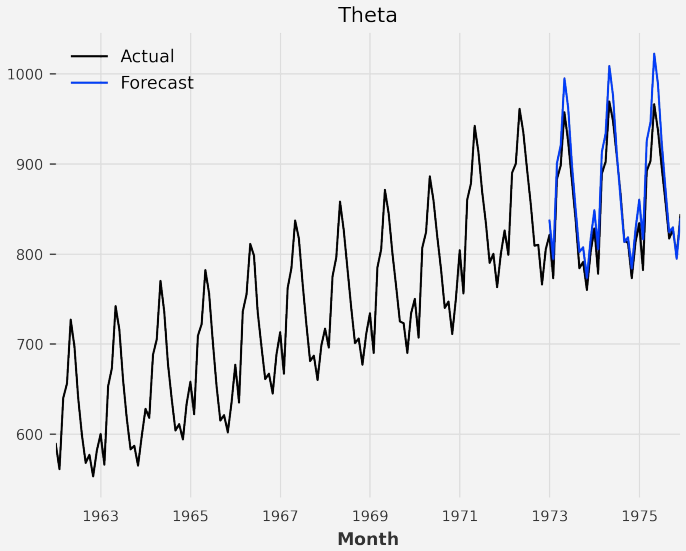
Forecasting – Theta

```python
from darts.models import Theta

model = Theta()
model.fit(training)
forecast = model.predict(len(validation))
```

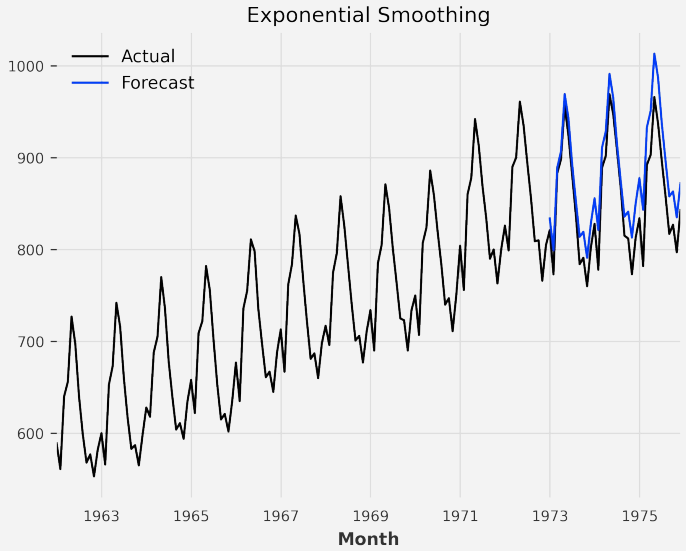


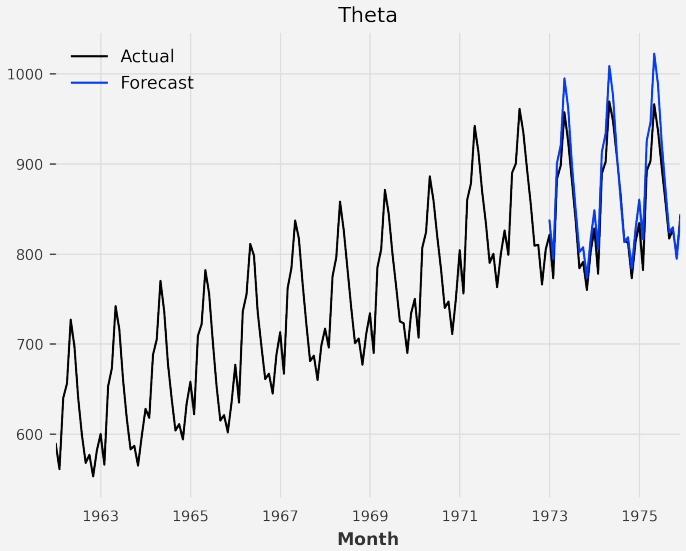
Specifying parameters

```python
from darts.models import Theta
from darts import SeasonalityMode

model = Theta(theta=2,
              seasonality_period=12,
              season_mode=SeasonalityMode.MULTIPLICATIVE)

model.fit(training)
forecast = model.predict(len(validation))
```



Evaluating predictions – Which one is better?





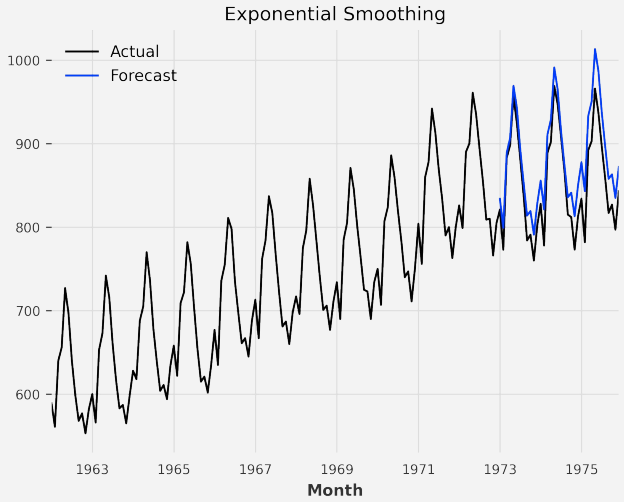
Metrics

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

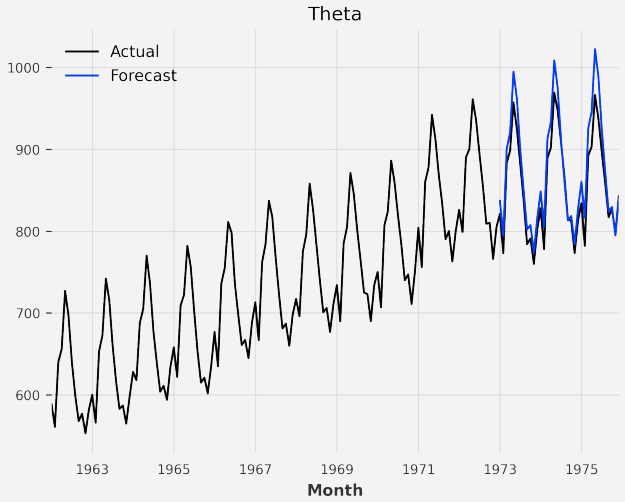
```python
from darts.metrics import mape

score = mape(validation, forecast)
```

```python
from darts.metrics import mase

score = mase(validation, forecast, training)
```

Which one is better?



MAPE: ~3.44%



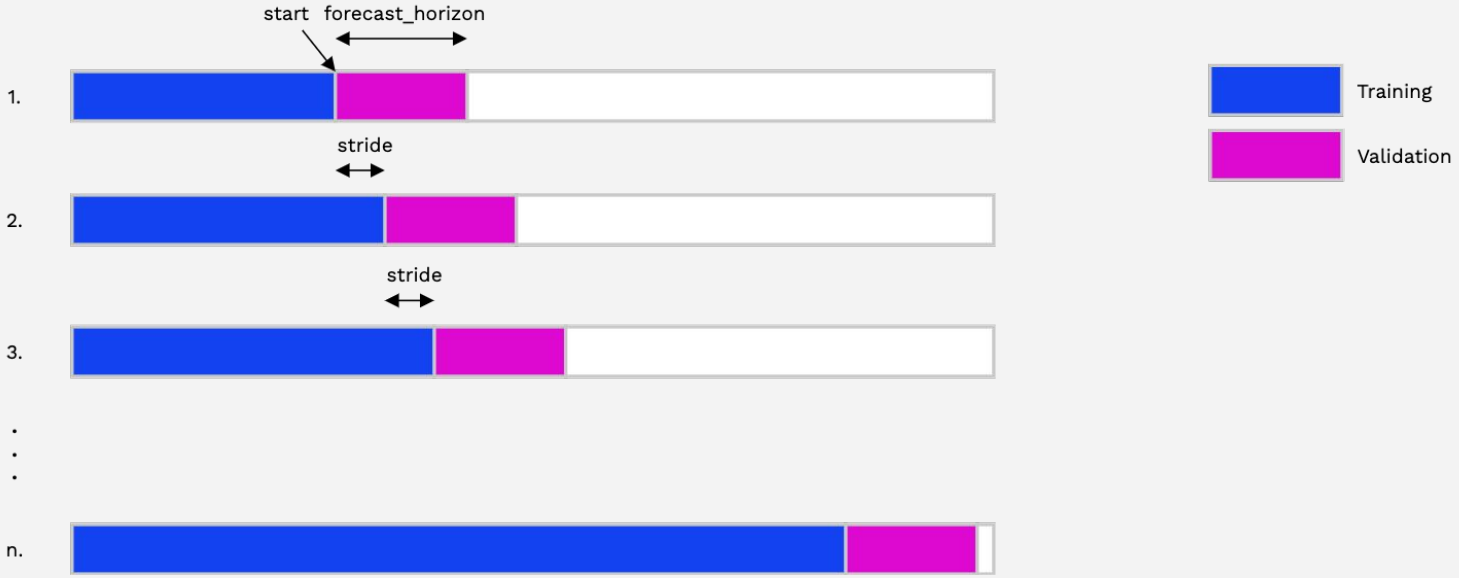
MAPE: ~2.42%

Evaluating model performance

`historical_forecasts()` and `backtest()`

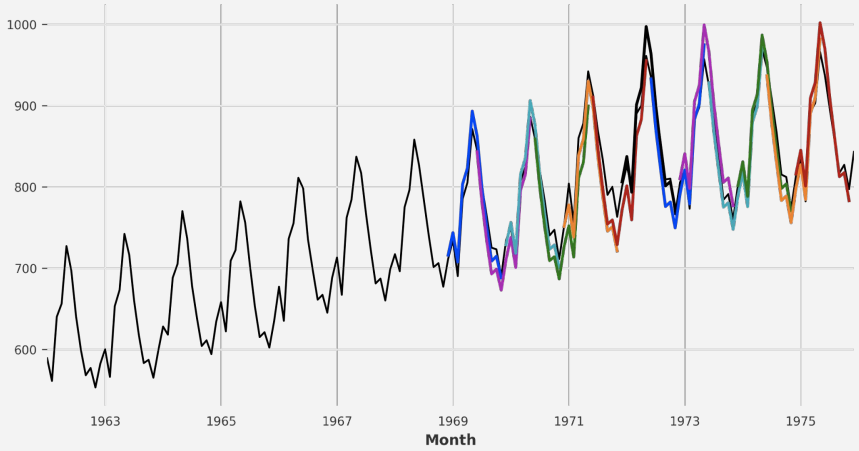
Simulate how a model **would have performed**
if it had been historically used to forecast a time series.

Predicting historical forecasts



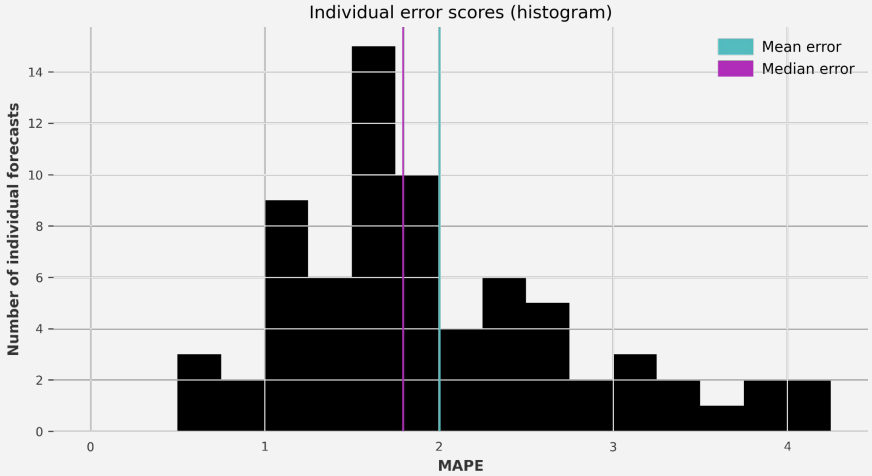
Historical forecasts

```python
forecasts = model.historical_forecasts(
    series=series,
    start=0.5,
    forecast_horizon=12,
    stride=6,
    last_points_only=False
)
```



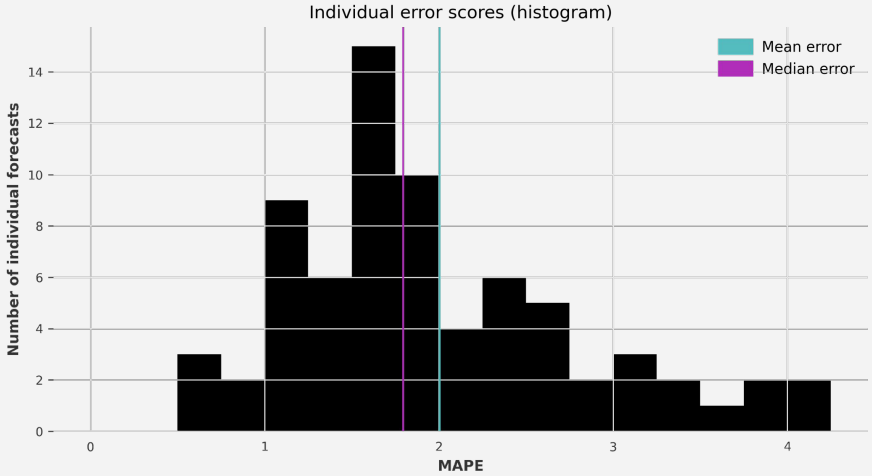
Backtesting

```
backtest_errors = model.backtest(
    series=series,
    start=0.5,
    forecast_horizon=12,
    stride=6,
    last_points_only=False,
    metric=mape,
    reduction=None
)
```



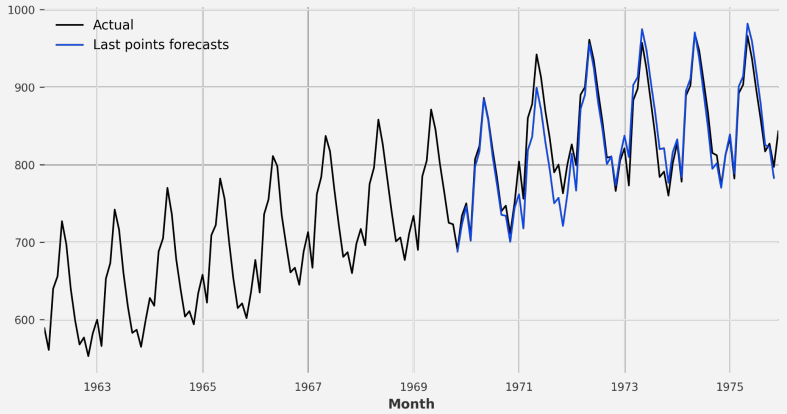
Backtesting

```python
import numpy as np

backtest_errors = model_es.backtest(
    series=series,
    start=0.5,
    forecast_horizon=12,
    stride=6,
    last_points_only=False,
    metric=mape,
    reduction=np.mean
)
```



The `last_points_only` parameter

```python
last_points = model.historical_forecasts(training_series=series,
                                         start=0.5,
                                         forecast_horizon=12,
                                         stride=1,
                                         last_points_only=True)
```



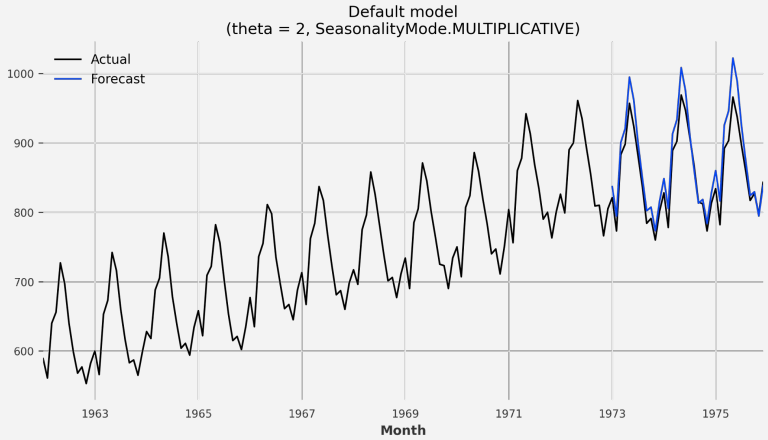
From evaluating to optimizing

How can we find the best hyperparameters to maximize accuracy ?

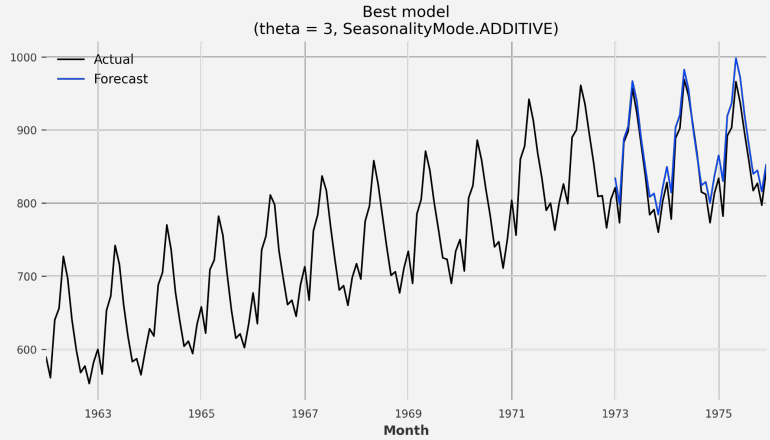
Gridsearch

```
parameters = {
    "theta": [0.5, 1, 1.5, 2, 2.5, 3],
    "season_mode": [SeasonalityMode.MULTIPLICATIVE, SeasonalityMode.ADDITIVE]
}

best_model = Theta.gridsearch(parameters=parameters,
                              training_series=training,
                              forecast_horizon=12,
                              start=0.5,
                              last_points_only=False,
                              metric=mape,
                              reduction=np.mean)

best_model.fit(training)
best_model_forecast = best_model.predict(len(validation))
```

Gridsearch



MAPE: ~2.42%



MAPE: ~2.32%

TimeSeries → Forecasting → Evaluating → **Tuning**

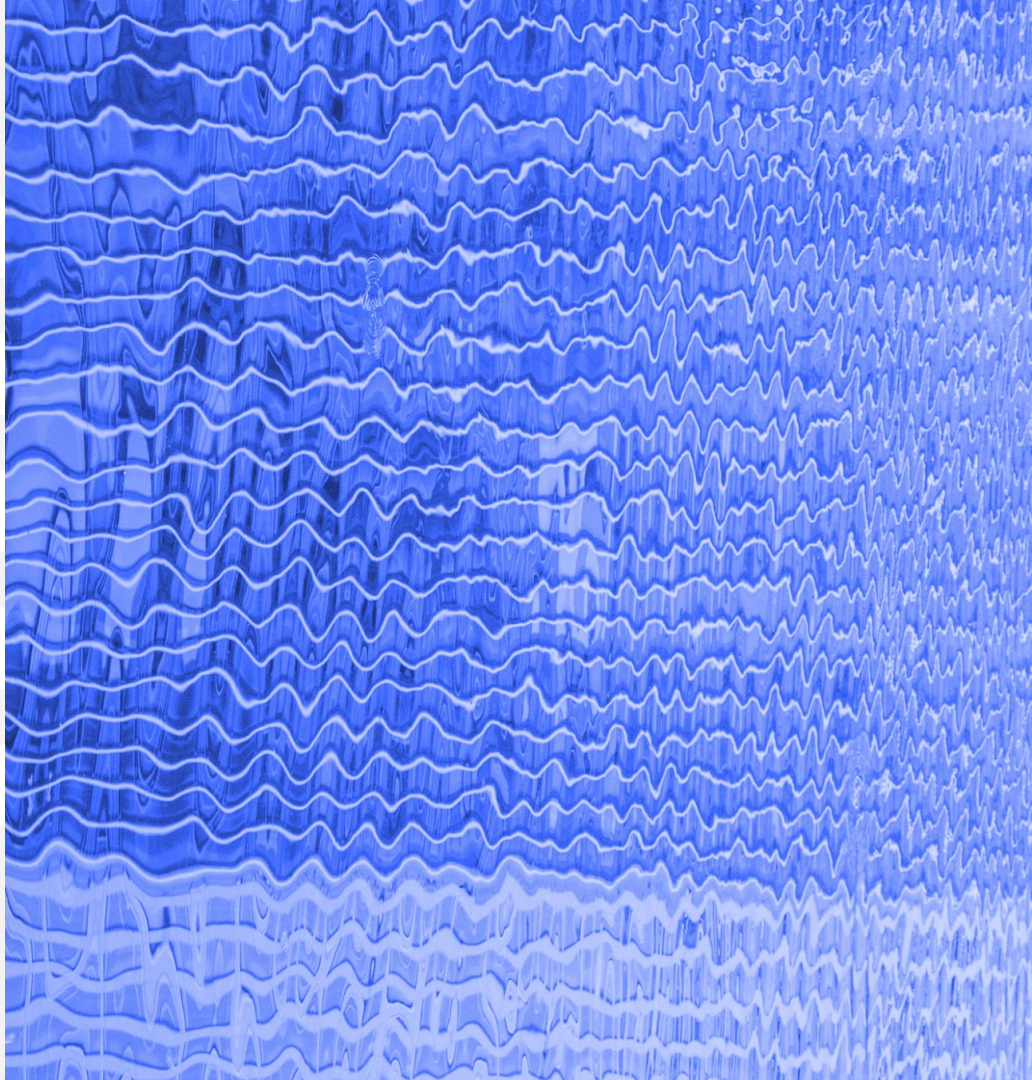
1 Intro to Forecasting & Darts

2 Forecasting using Darts

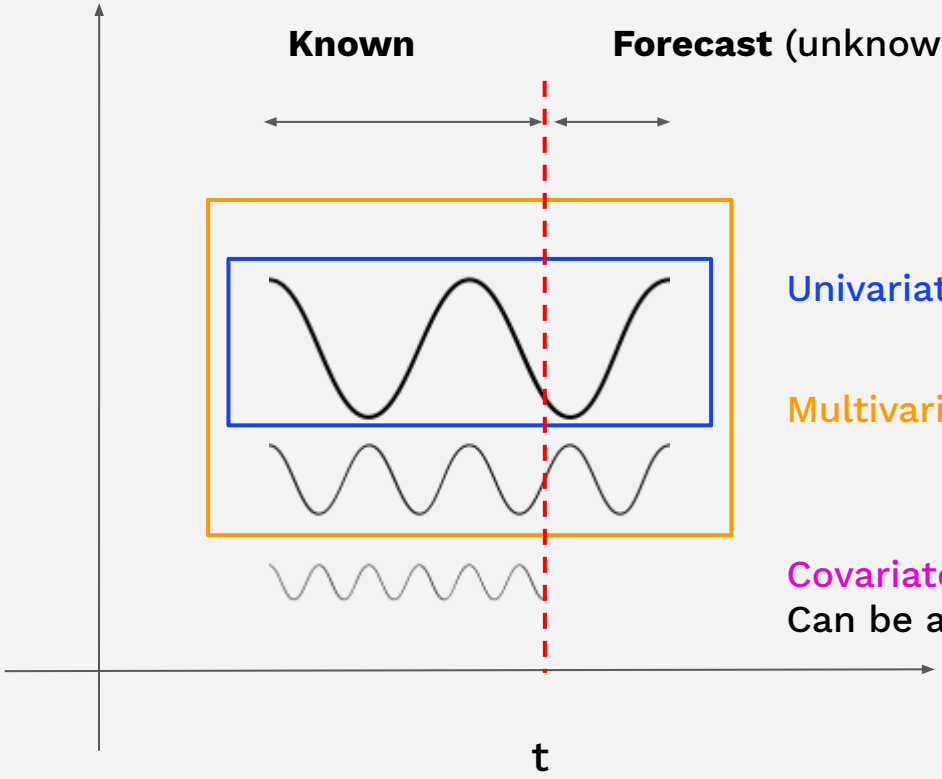
3 Training on multiple time-series

4 Probabilistic forecasting

5 Try Darts!



Supported Data Types

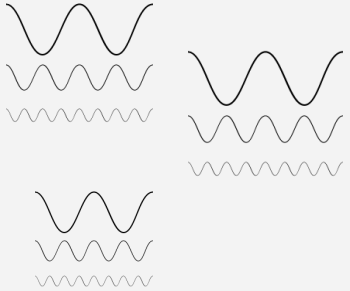


Univariate ts

Multivariate ts

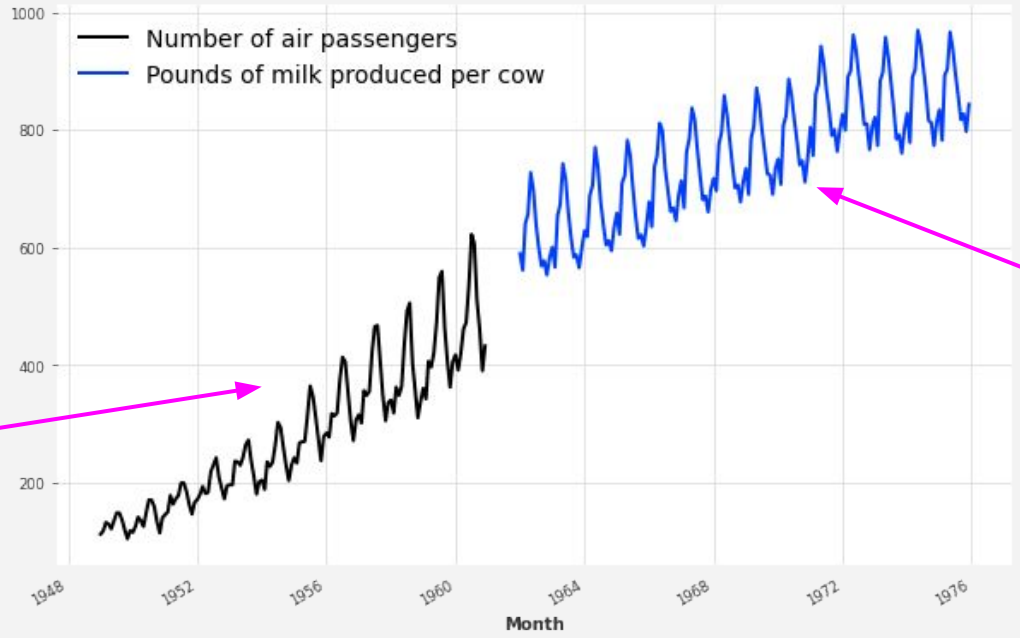
Covariate ts
Can be also known in the future (weekday etc)

Meta learning



Unit8

Meta-learning on multiple time series



want to predict

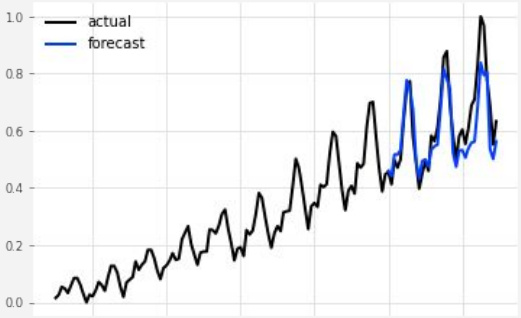
Could this help?

Unit8

Meta-learning on multiple time series

Train on air traffic data only

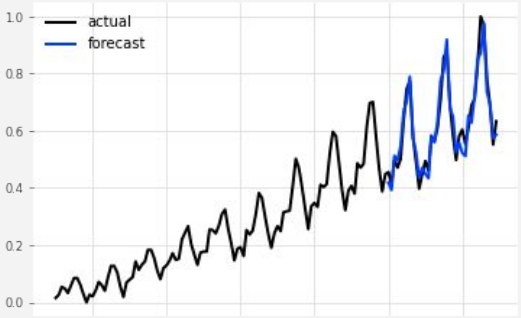
```
model_air = NBEATSModel(**kwargs)
model_air.fit(train_air)
pred = model_air.predict(n)
```



MAPE: 8.9%

Train on air traffic and milk production data

```
model_air_milk = NBEATSModel(**kwargs)
model_air_milk.fit([train_air, train_milk])
pred = model_air_milk.predict(n, series=train_air)
```



MAPE: 5.5%

Unit8

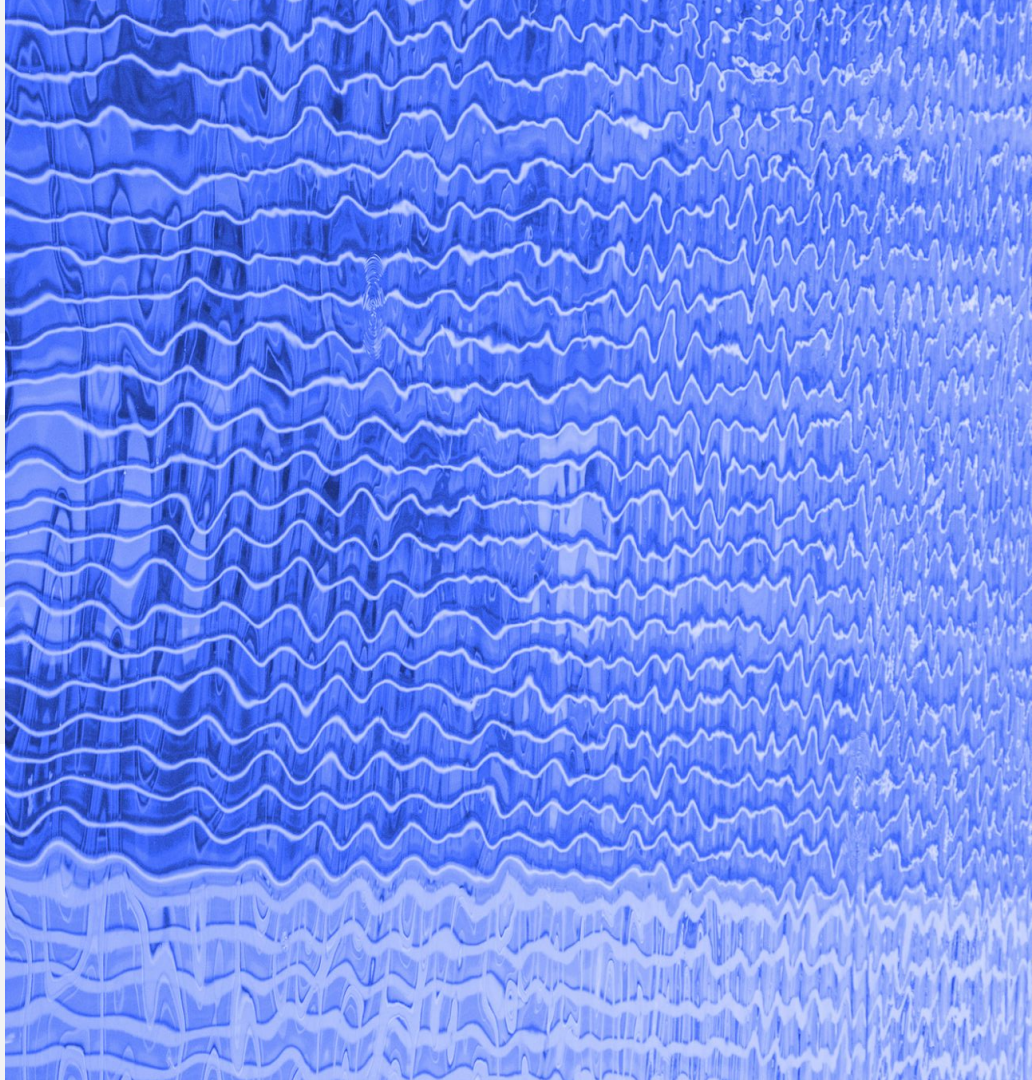
1 Intro to Forecasting & Darts

2 Forecasting using Darts

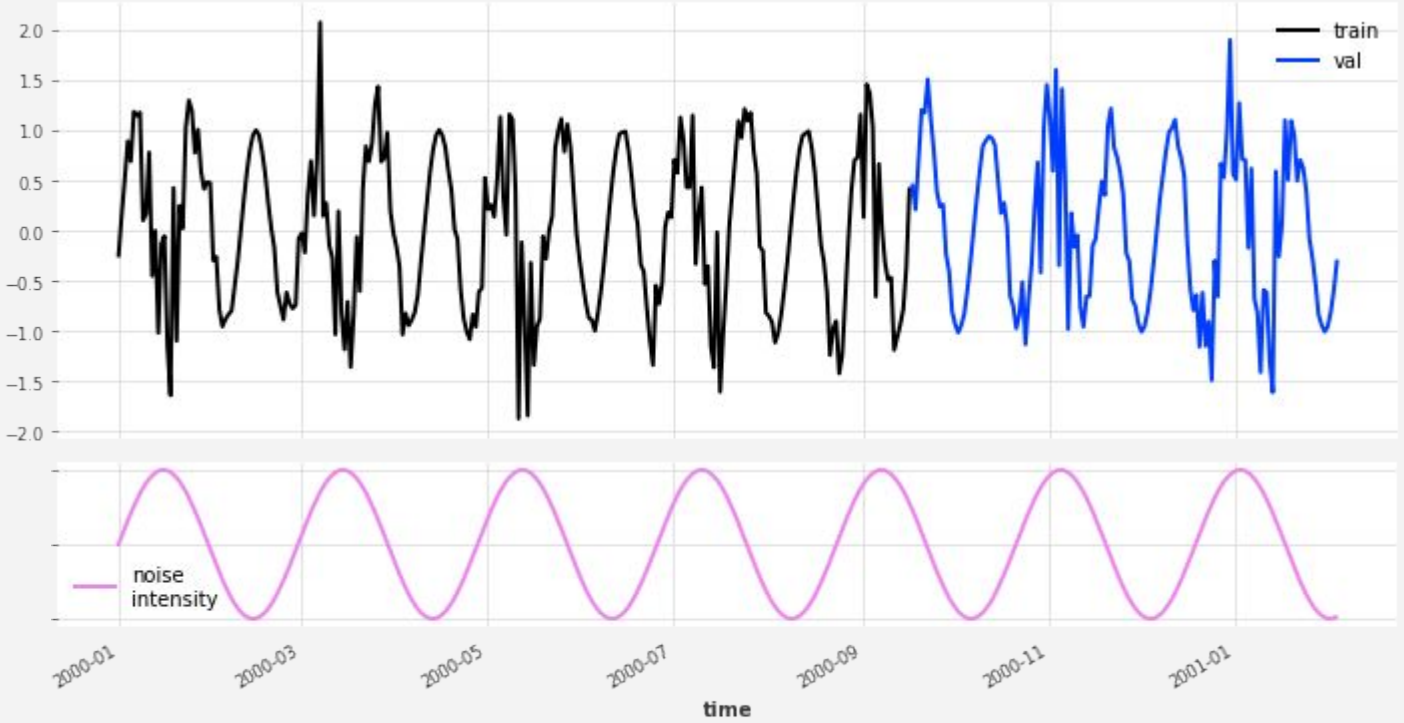
3 Training on multiple time-series

4 Probabilistic forecasting

5 Try Darts!



Unpredictable components in time series

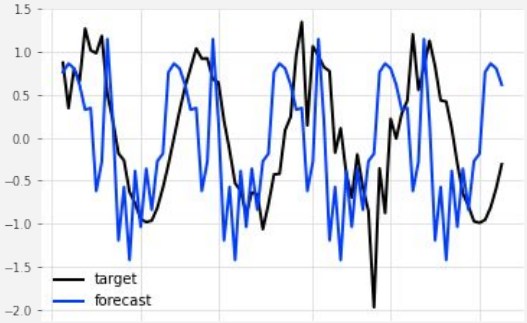


Unit8

Unpredictable components in time series

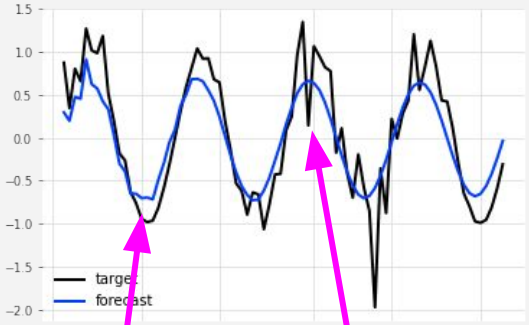
Attempt 1

```
model = NaiveSeasonal(seasonal_period)
model.fit(train)
pred = model.predict(n)
```



Attempt 2

```
model = ARIMA(seasonal_period, 0, 0)
model.fit(train)
pred =  model.predict(n)
```

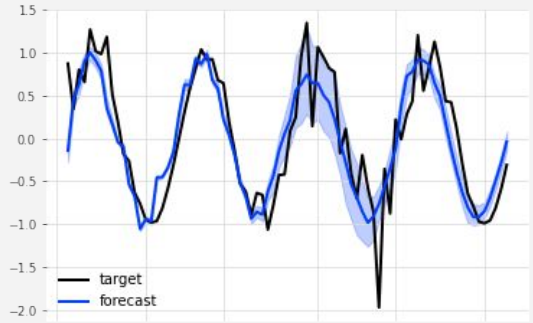


safe forecast

uncertain forecast

Attempt 3

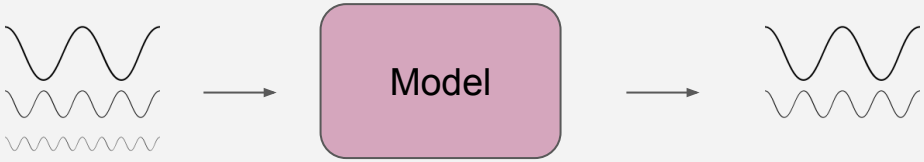
```
model = TCNModel(likelihood=GaussianLikelihoodModel(),
                 **kwargs)
model.fit(train, covariates)
pred = model.predict(n, covariates=covariates,
                     num_samples=100)
```



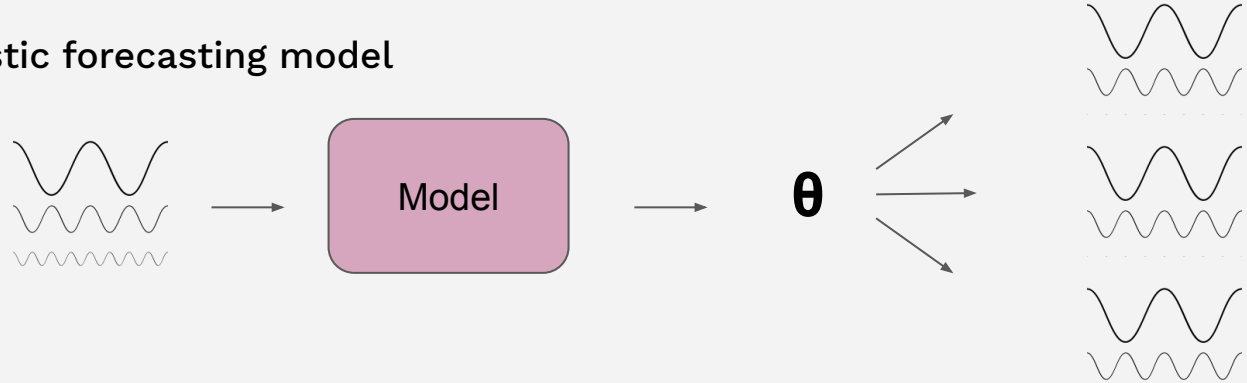
Unit8

Probabilistic forecasts

Deterministic forecasting model

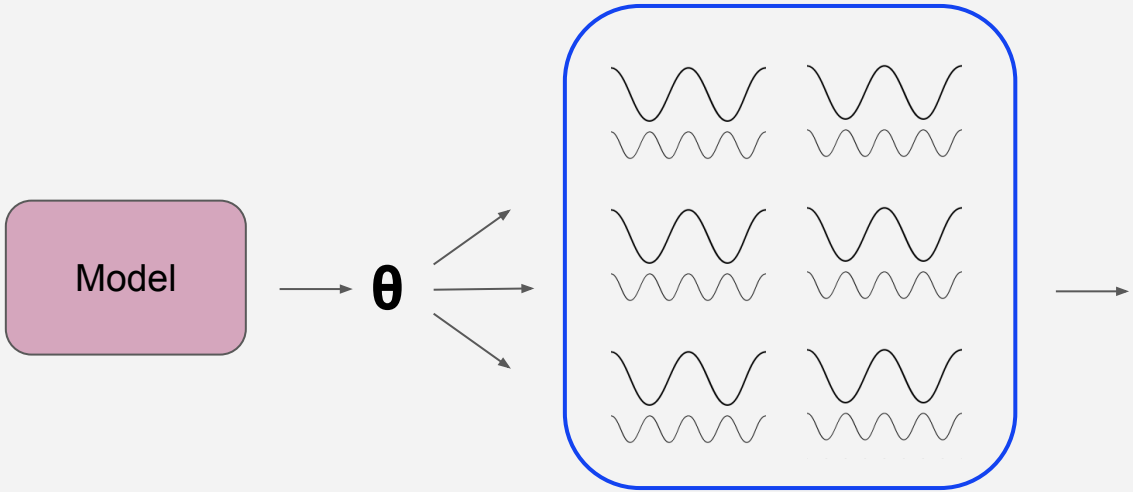


Probabilistic forecasting model

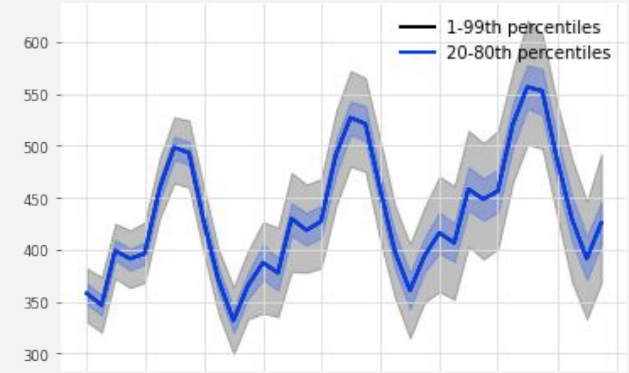


Unit8

Probabilistic forecasts



Probabilistic time series
(distribution-agnostic)

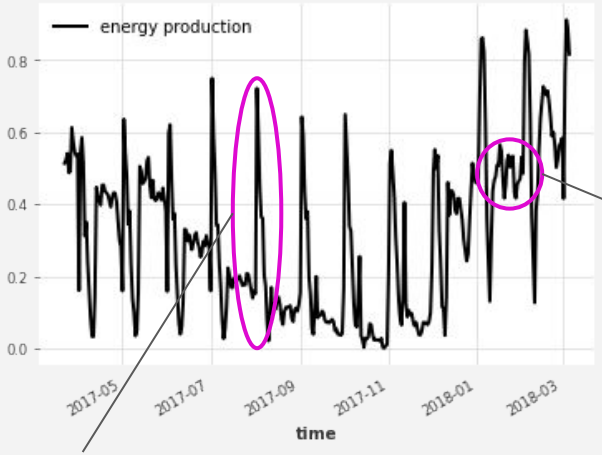



```
forecast.plot(low_quantile=0.01, high_quantile=0.99)
forecast.plot(low_quantile=0.2, high_quantile=0.8)
```

Confidence intervals

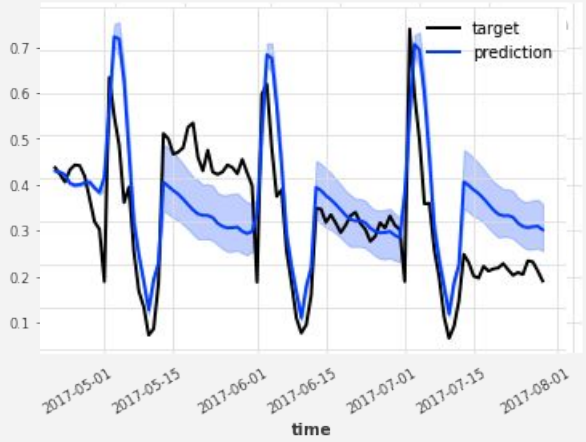
Unit8

Real-world probabilistic forecasting example - energy production



Monthly spikes with predictable shapes

Less predictable values in-between




```
model = RNNModel(likelihood=GaussianLikelihoodModel(), **kwargs)
model.fit(energy_train, covariates=day_of_month)
energy_forecast = model.predict(n, num_samples=100)
```

Unit8

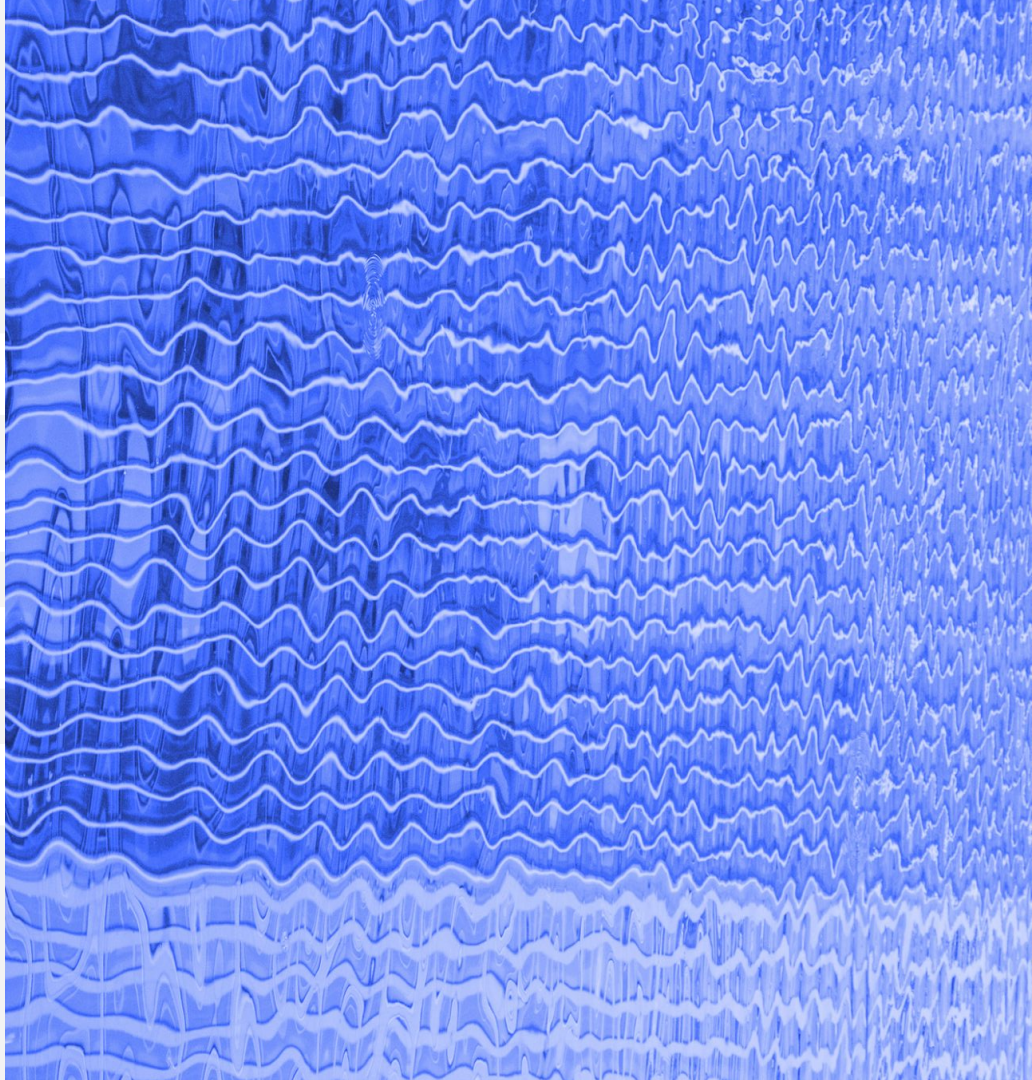
1 Intro to Forecasting & Darts

2 Forecasting using Darts

3 Training on multiple time-series

4 Probabilistic forecasting

5 Try Darts!



Darts

Discovery

- Statistical analysis
- Visualizations

Preprocessing

- Missing value interp.
- Normalizing, scaling
- Seas./trend removal

Forecasting

- Statistical models
- Deep-learning models
- Probabilistic forecasting
- Ensembling
- Multiple TS support (meta learning)
- Multivariate and covariate support

Model Evaluation and Selection

- Historical forecasting / backtesting
- Residual analysis
- Metrics
- Grid search

Unit8

If you want to try darts, here are some steps!

Check out the library yourself! As easy as: `pip install darts`

Look through one of our tutorial notebooks or intro blog post

- https://github.com/unit8co/darts/
- https://medium.com/unit8-machine-learning-publication/darts-time-series-made-easy-in-python-5ac2947a8878

Contacting us directly on github or via: info@unit8.co. We're always happy to answer questions or discuss time series problems!

Unit8

unit8.co

Francesco
francesco.laessig@unit8.com
–
Gaël
gael.grosch@unit8.com

thank you