Forecasting GDP with machine learning methods

SUMMER 🌞 series presentation

28 July 2022

Introduction 👋

Collaborators

We have a team of people that consists of statisticians, econometricians and data scientists / engineers.

- **Dawie van Lill** (analysis) 🗣
- François Kamper (analysis)
- Hylton Hollander (data team supervision)
- Ruan Erasmus (data team)
- Jannes Reddig (data team)

Research question(s)

There are two parts to our research question. One part relates to performance, the other to ease of use.

> Do machine learning (and deep learning) methods contribute to forecasting performance over and above the traditional autoregressive model? Are these methods easy to implement for practitioners?

Managing expectations

This is **not** a technical talk on the methods used.

The talk is more about our results and the way that our experience translates to practitioners.

Please reach out

Contact us after the presentation to discuss model details. We would love to know how we can improve our models!

Contribution

ARIMA forecasting benchmark versus ML and DL models.

ML models	DL models
1. SVR (U & M)	1. RNN-LSTM (M)
2. RF (U & M)	2. RNN-GRU (M)
3. GBM (U & M)	3. N-BEATS (U & M)
	4. N-HiTS (U & M)
	5. TCN (M)

U = Univariate and M = Multivariate

Useful libraries

scikit learn

LightGBM

PyTorch

Darts by Unit8

Data 📈

Data

The **target variable** is the seasonally adjusted annualised quarter-on-quarter growth rate of GDP covering the period from 1993-06-31 to 2022-03-31.

Quarterly and monthly macroeconomic variables are collected using the EconData platform from Codera Analytics.

Target variable is often used in machine learning, while dependent variable is more common in econometrics.

Data preparation and cleaning

Remove variables with large proportion of missing data.

- Percentage of entire series
- Percentage of most recent observations

Outliers removed (beyond 1.5 times the interquartile range)

Remove variables that are scalar multiples.

Data transformations

ADF test used to determine whether a unit root (or roots) is (are) present for each variable.

- `I(0)` variables: level
- `I(1)` variables: **growth rate**
- `I(2)` variables: **rate of growth**

All variables are standardized following transformation.

Ignore compounding effects when annualizing growth rate and rate of growth.

Data transformations (cont.)

Suggestions

Space for improvement possible here. Suggestions are welcome.

Create new variables from the values of the 1st, 2nd, and 3rd month of each quarter.

Alternative: Monthly variables aggregated to a quarterly level using the mean of the monthly values (after which transformations and standardizations are applied)

▶ Code

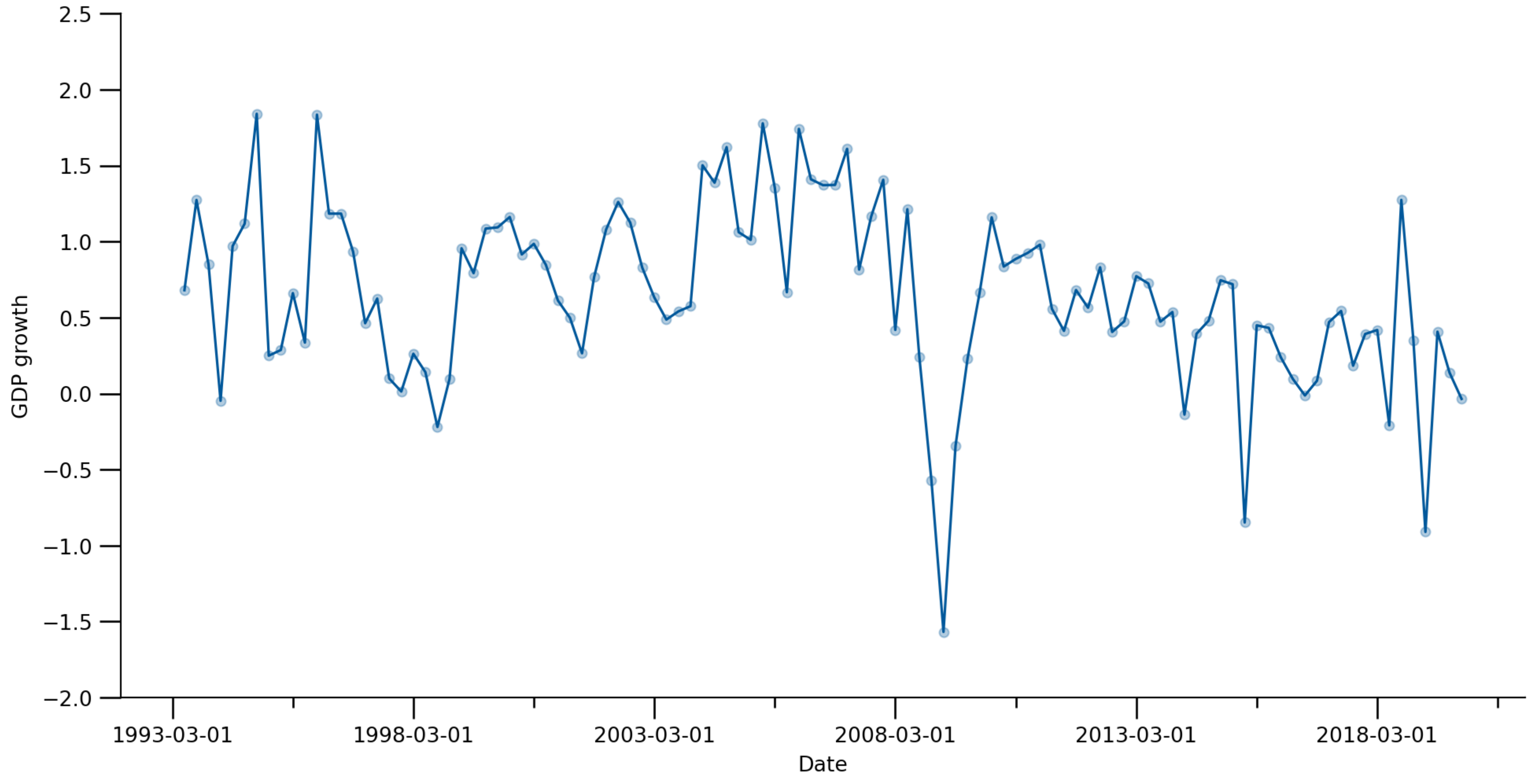

Figure 1: GDP Pre-COVID

▶ Code

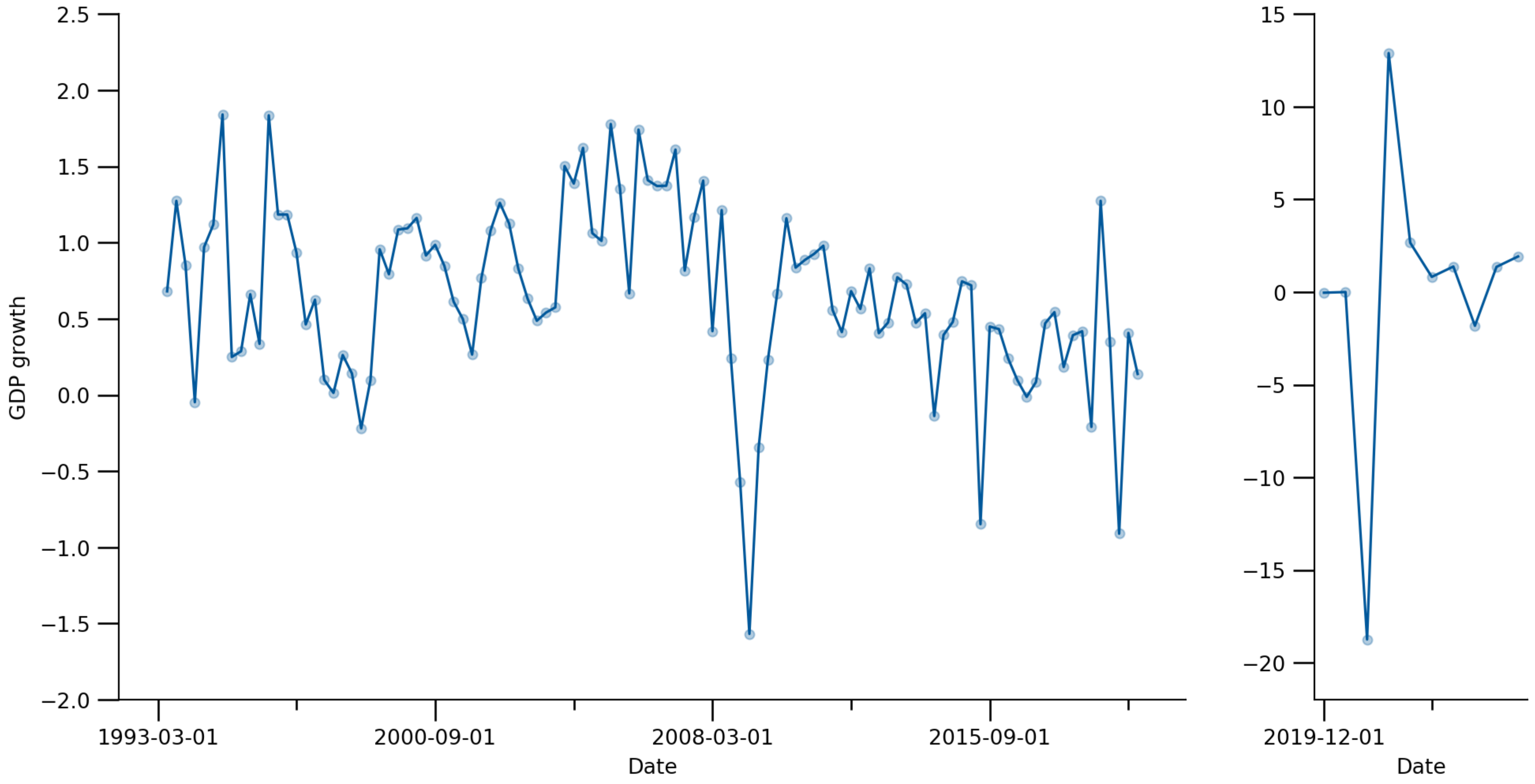

Figure 2: Full GDP Series

▶ Code

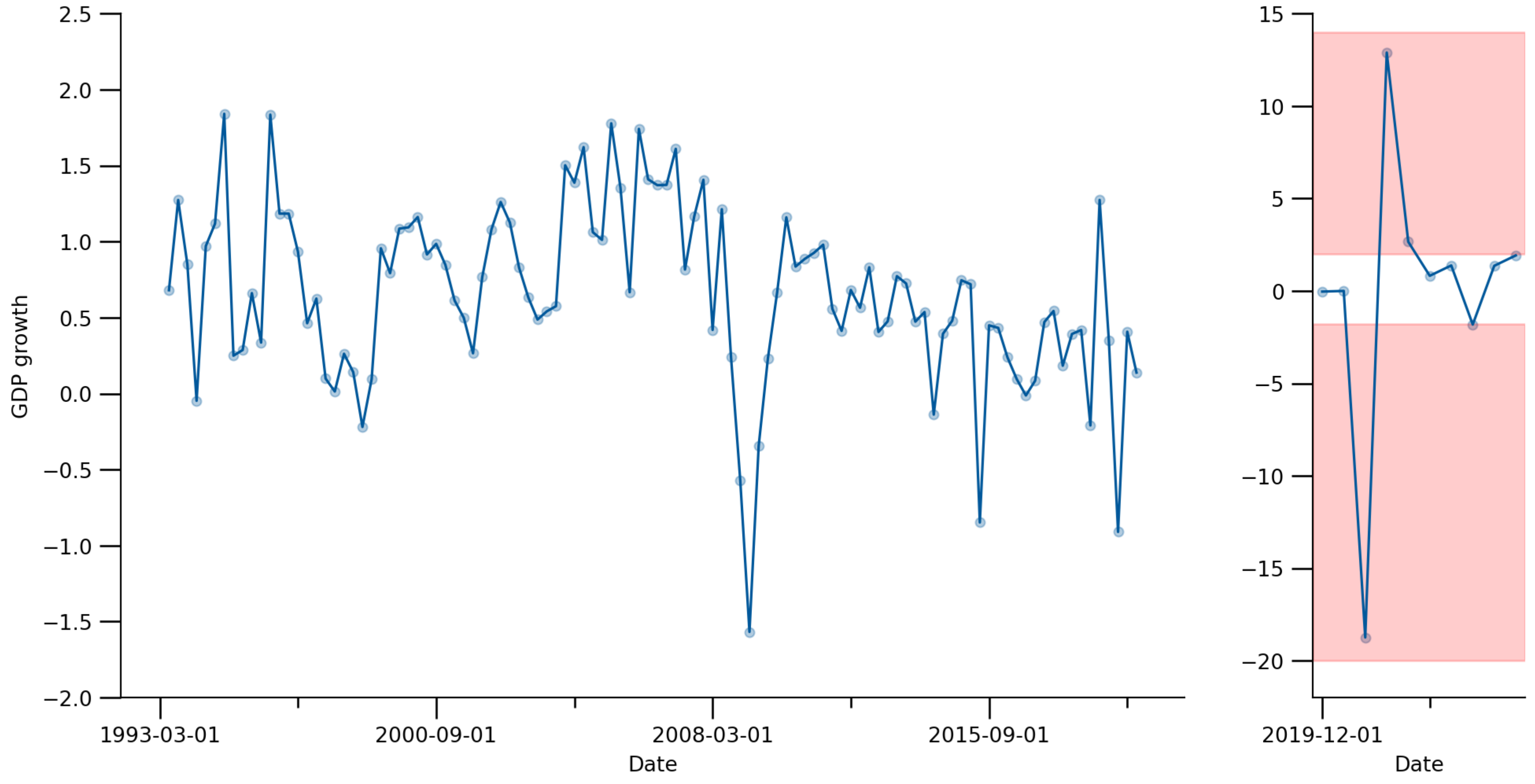

Figure 3: Full GDP Series

Training the model 🧘🏾

▶ Code

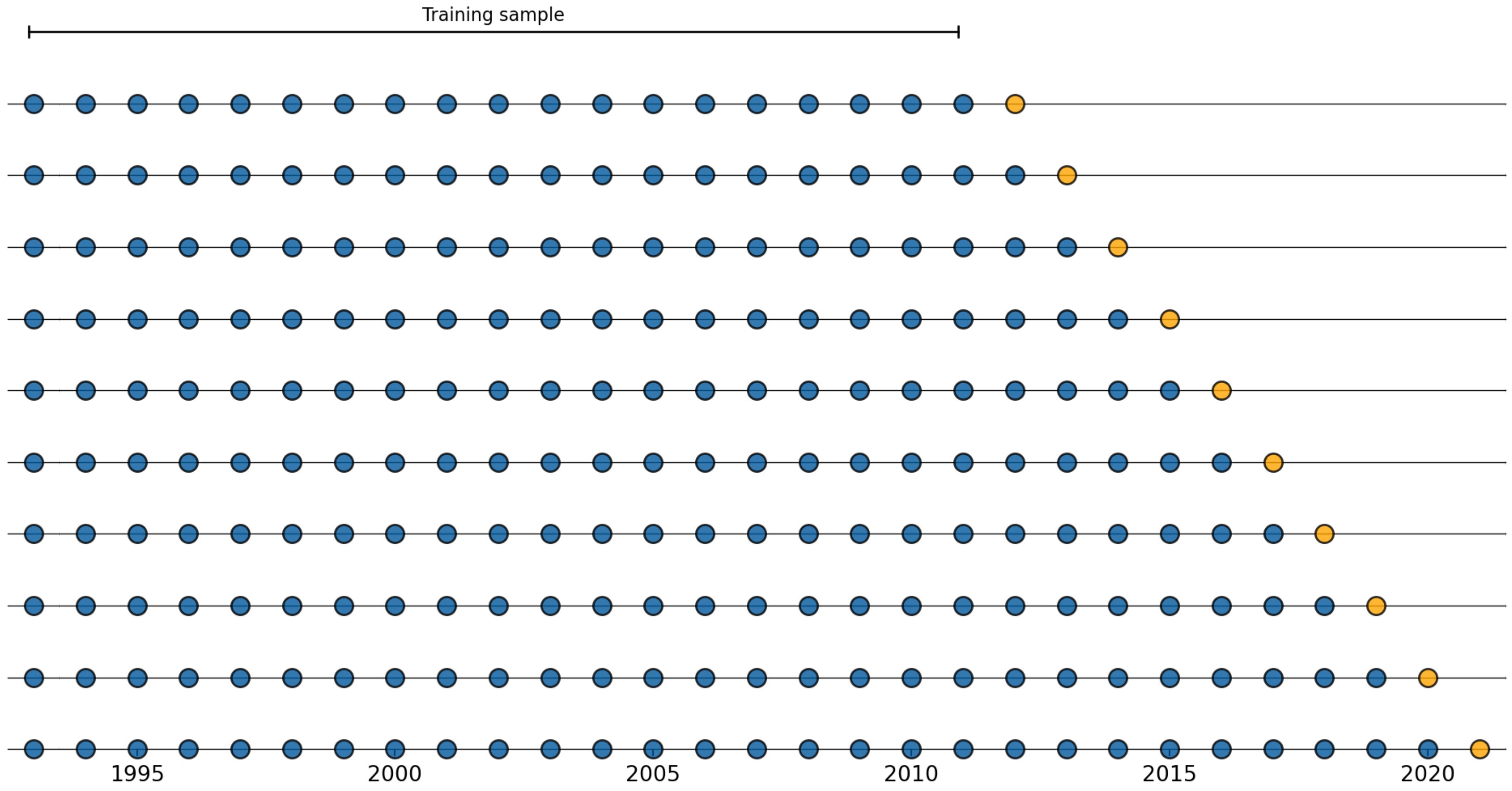

Figure 4: Time series cross validation

▶ Code

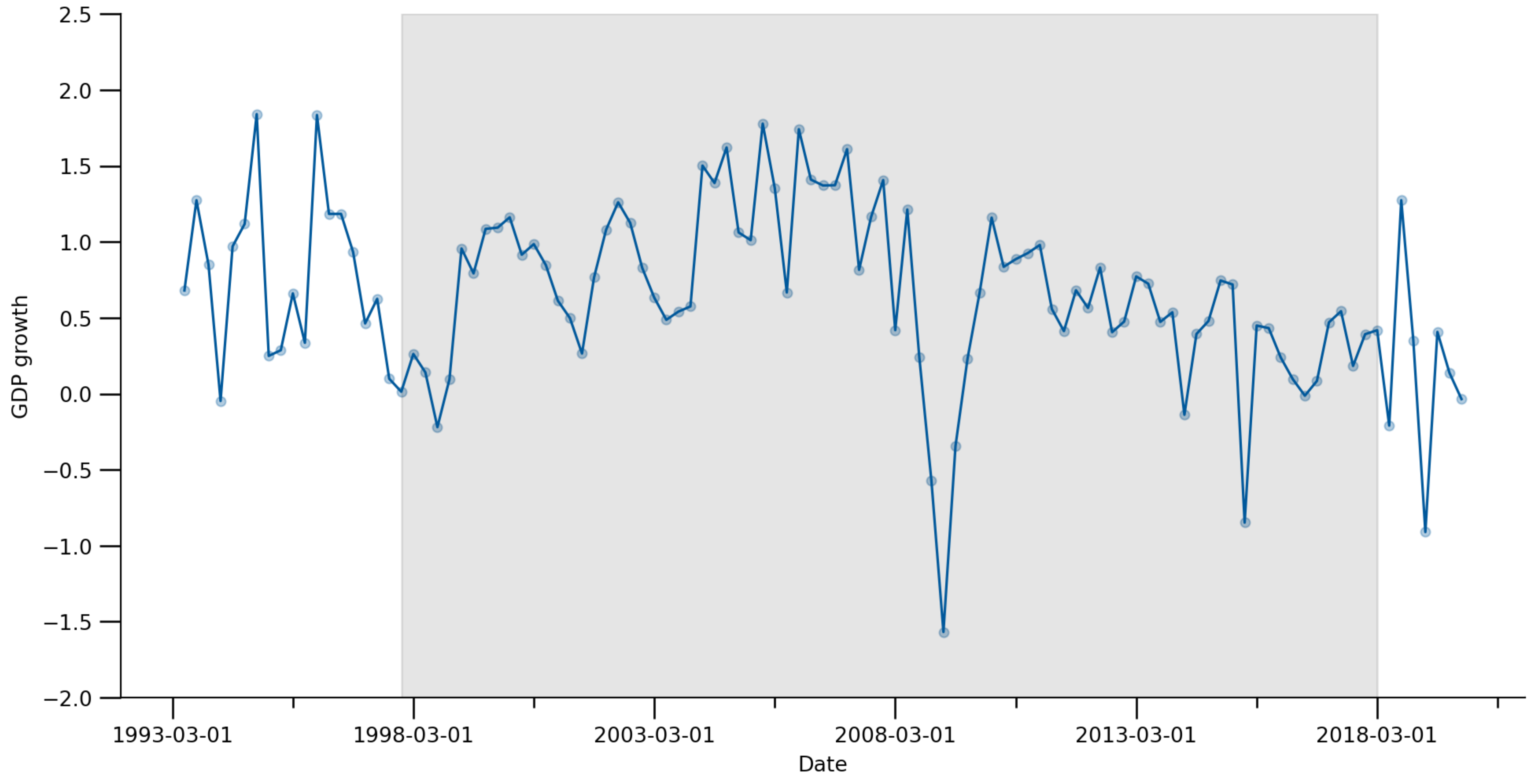

Figure 5: Hyperparameter tuning sample period

Results

Results

We display *two types of results* for each model of interest.

> n-period ahead forecast

The first result compares the forecast against the out of sample data point (**test set**).

> Historical one-period ahead forecast (backtest)

The second result is a hypothetical scenario where we compare against historical data.

Results

Reminder of the types of models we are interested in.

ML models

1. SVR (U & M)
2. RF (U & M)
3. GBM (U & M)

DL models

1. RNN-LSTM (M)
2. RNN-GRU (M)
3. N-BEATS (U & M)
4. N-HiTS (U & M)
5. TCN (M)

U = Univariate and M = Multivariate

Benchmark model

▶ Code

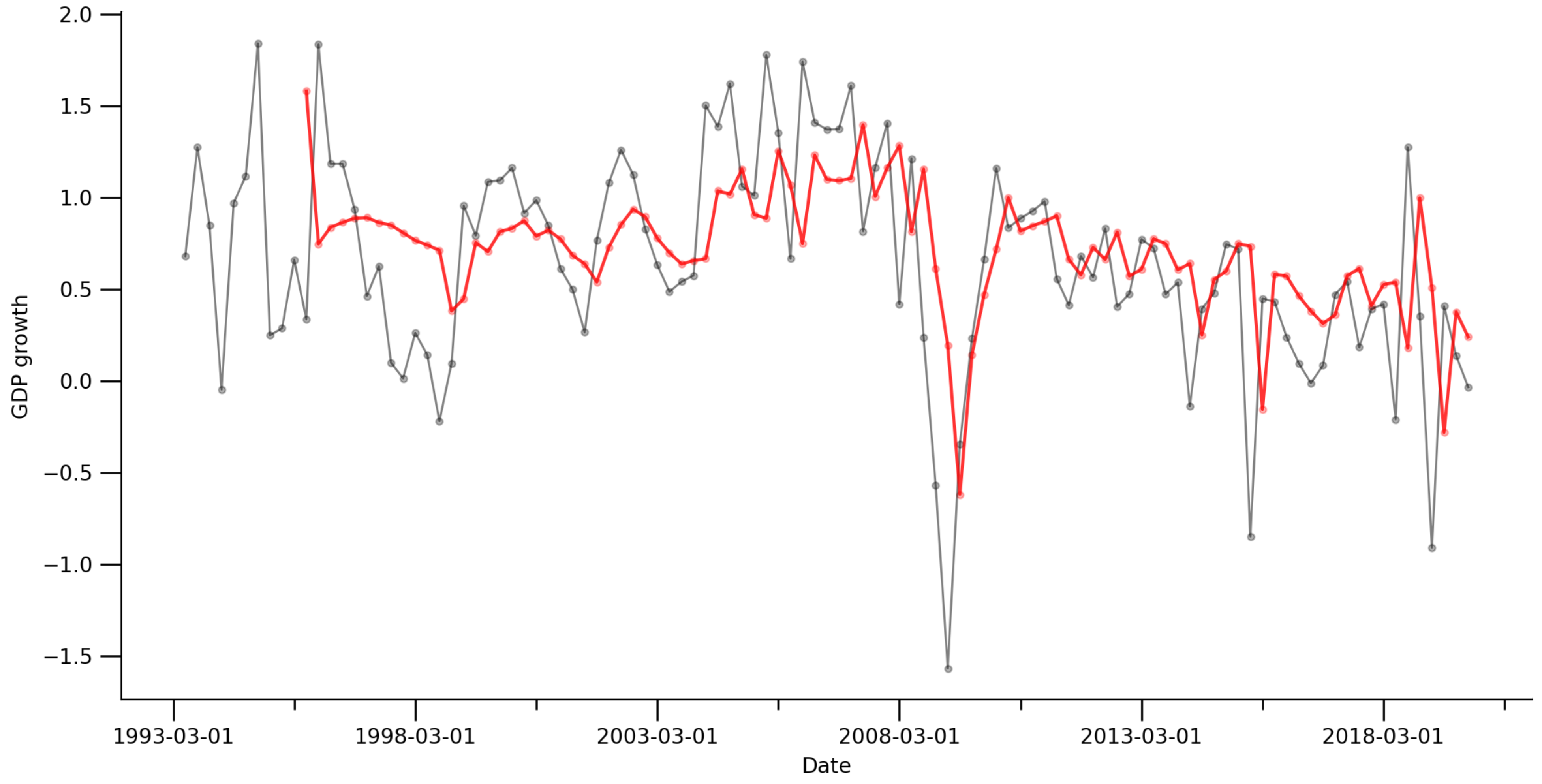

Figure 6: AR(1) model

Benchmark model

Quick **demo** of how to run the ARIMA model with Darts.

```python
from darts import TimeSeries
from darts.models import AutoARIMA
from darts.metrics import rmse

data = pd.read_csv("../000_data/transformed_gdp.csv")
series = TimeSeries.from_series(data)
train, val = series.split_before(0.9)

model_ar = AutoARIMA()
model_ar.fit(train)
prediction_ar = model_ar.predict(len(val))

historical_fcast_ar = model_ar.historical_forecasts(
    series, start = 0.1, forecast_horizon = 1, verbose = True)
```

Machine learning 🤖

Machine learning

Support vector regression (SVR), random forests (RF) and gradient boosted machines (GBM) are machine learning algorithms designed for **tabular data**.

Temporal data needs to be converted into tabular form.

This can be done by using lags of the target series and features as the tabular covariates.

Support vector regression

SVR is trained using the ε-insensitve loss function.

This loss function leads to the concept of support vectors, where other data points do not contribute to the fit.

We use the **Scikit-Learn** implementation of SVR.

▶ Code

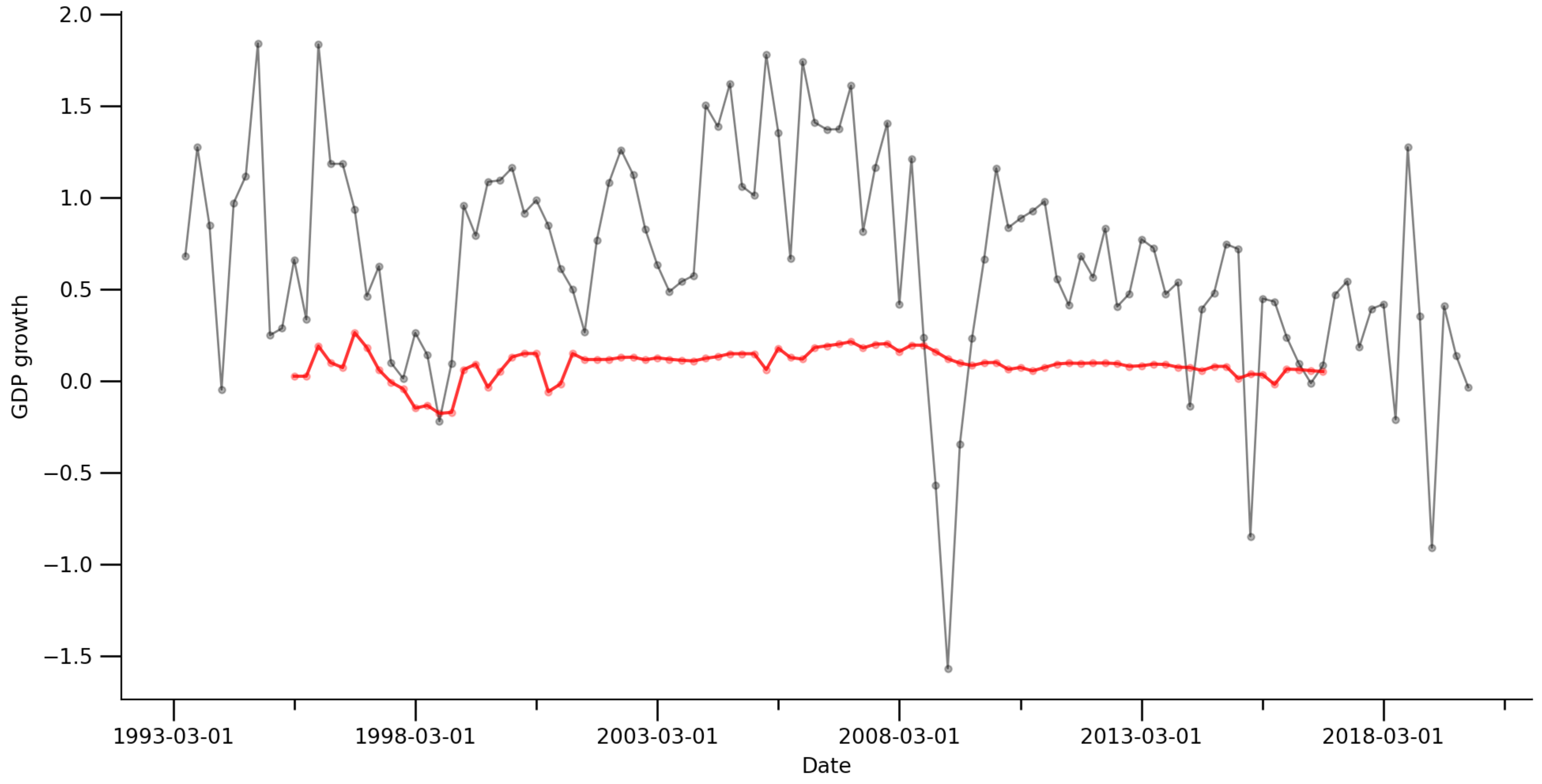

Figure 7: Support vector regression

Random forests

Random forests fits an ensemble of regression trees, where each regression tree is fitted to a bootstraped sample from the original data.

The aim of a random forest is to reduce the high variance associated with single regression trees.

We use the **Scikit-Learn** implementation of random forests.

Gradient boosted machines

Similar to random forests, a gradient boosted machine (GBM) is an ensemble of regression trees.

The difference between random forests and GBMs is that regression trees are added sequentially to the emsemble.

In the case of regression, regression trees are fitted to the residuals produced by the current ensemble.

We use the **LightGBM** implementation.

LightGBM

Deep learning 🧠

Deep learning

Traditional feedforward networks cannot be used in time series forecasting.

Time structure is not recognized.

RNN-LSTM

RNN models use a mechanism to retain historic information to forecast values.

RNN-LSTM: Recurrent neural network - long short-term memory

▶ Code

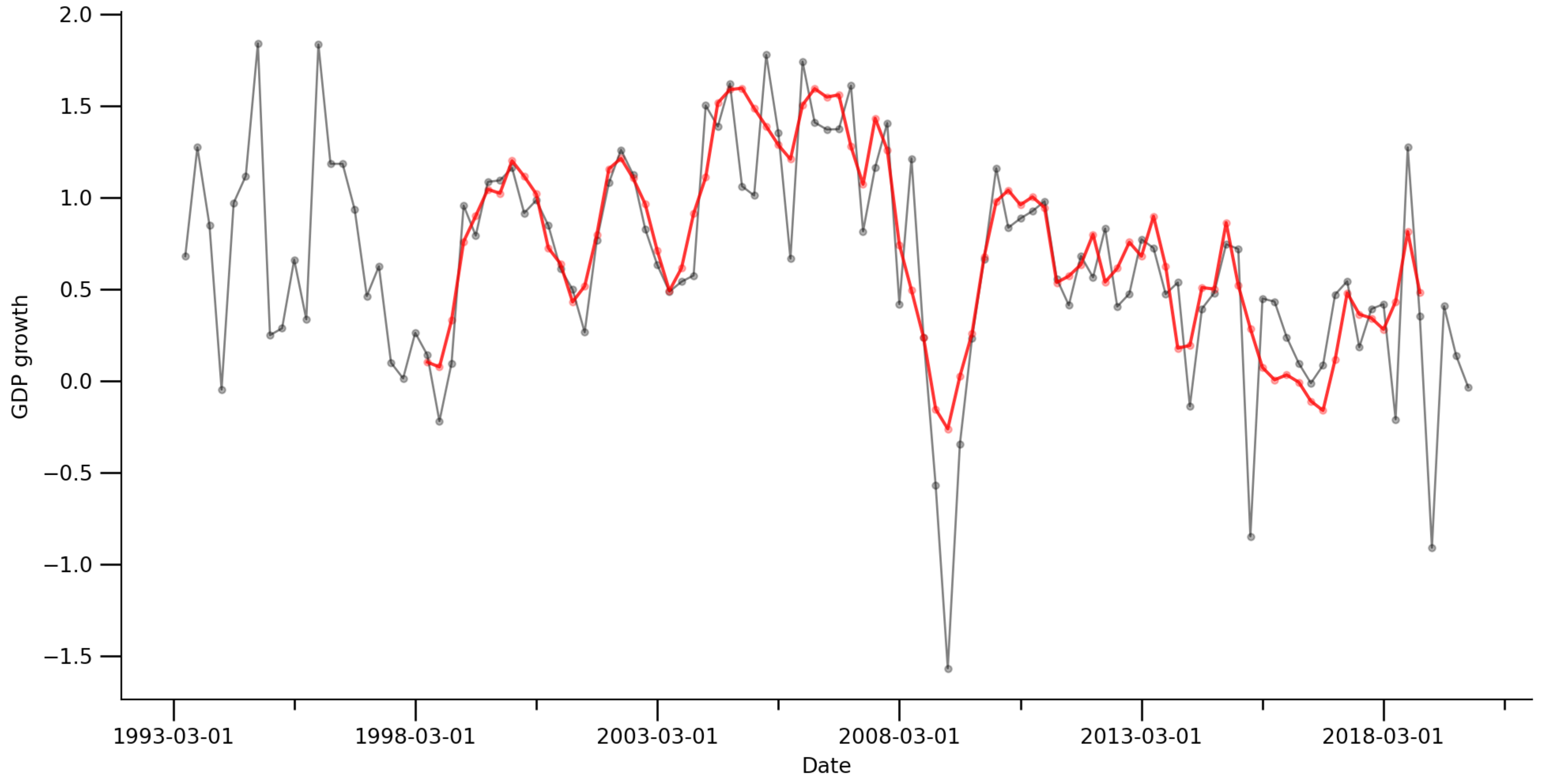

Figure 8: RNN-LSTM model

▶ Code

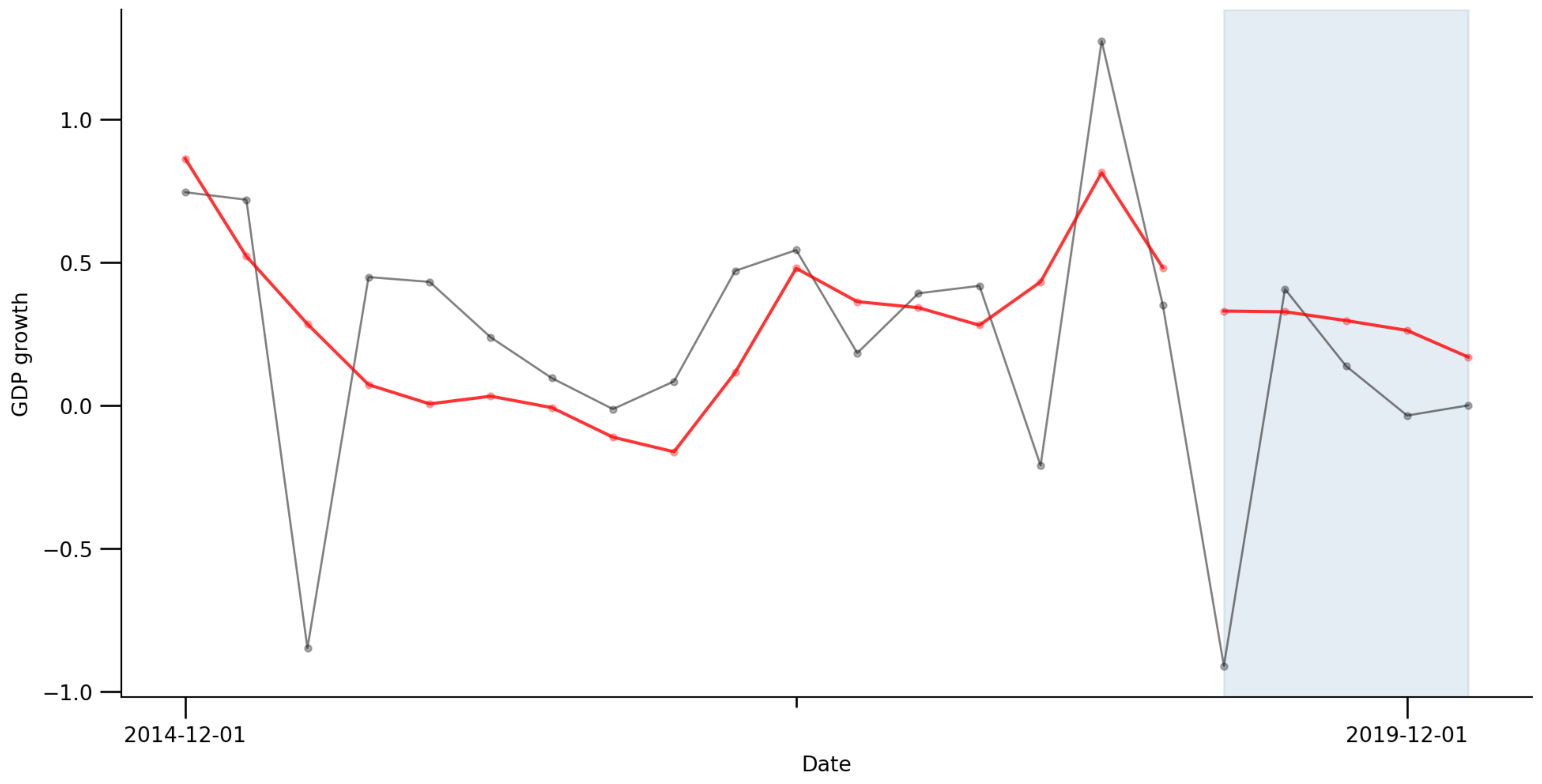

Figure 9: RNN-LSTM model forecast (5-periods ahead)

RNN-GRU

RNN-GRU: Recurrent neural network - gated recurrent unit

▶ Code

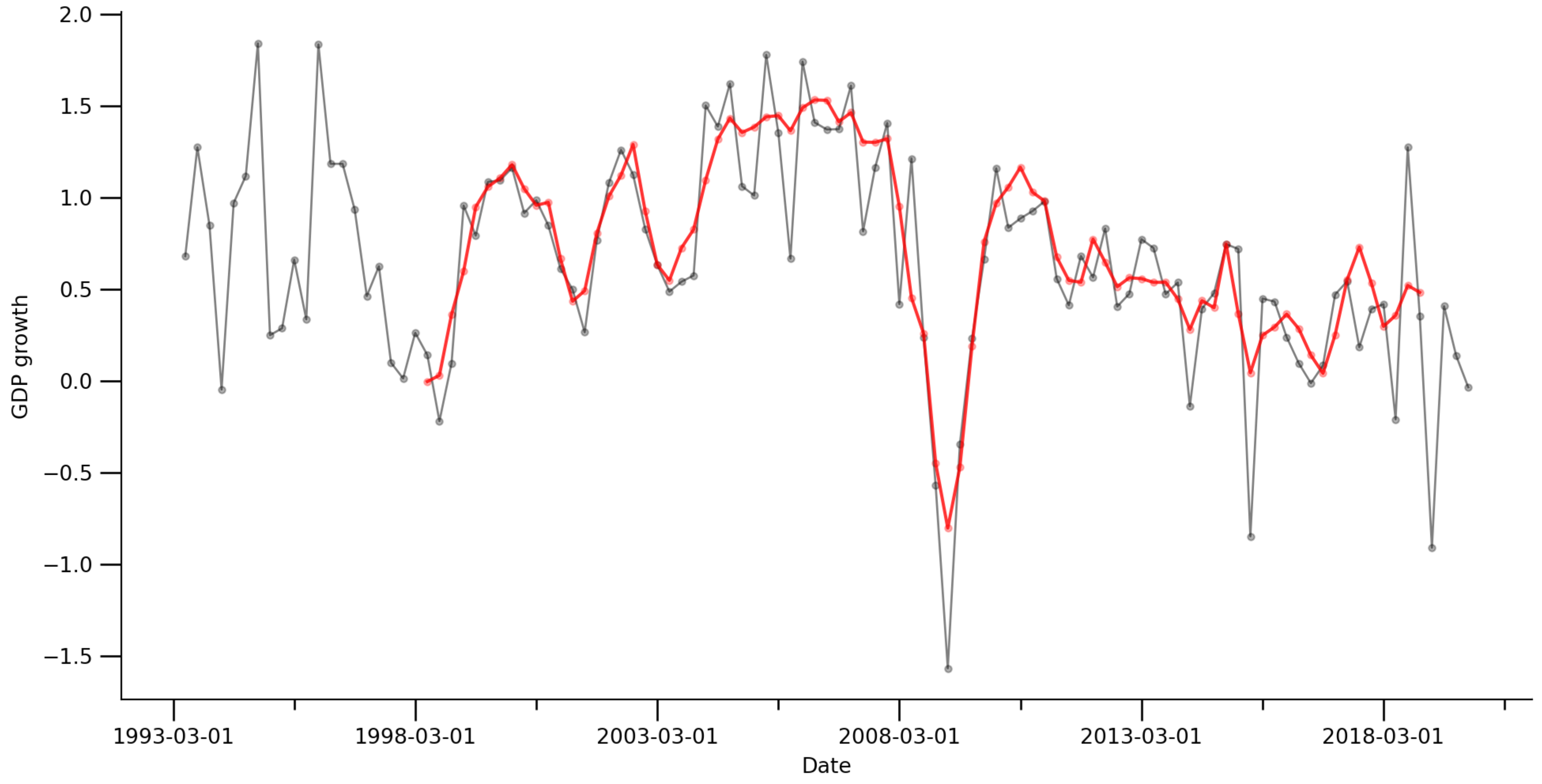

Figure 10: RNN-GRU model

▶ Code

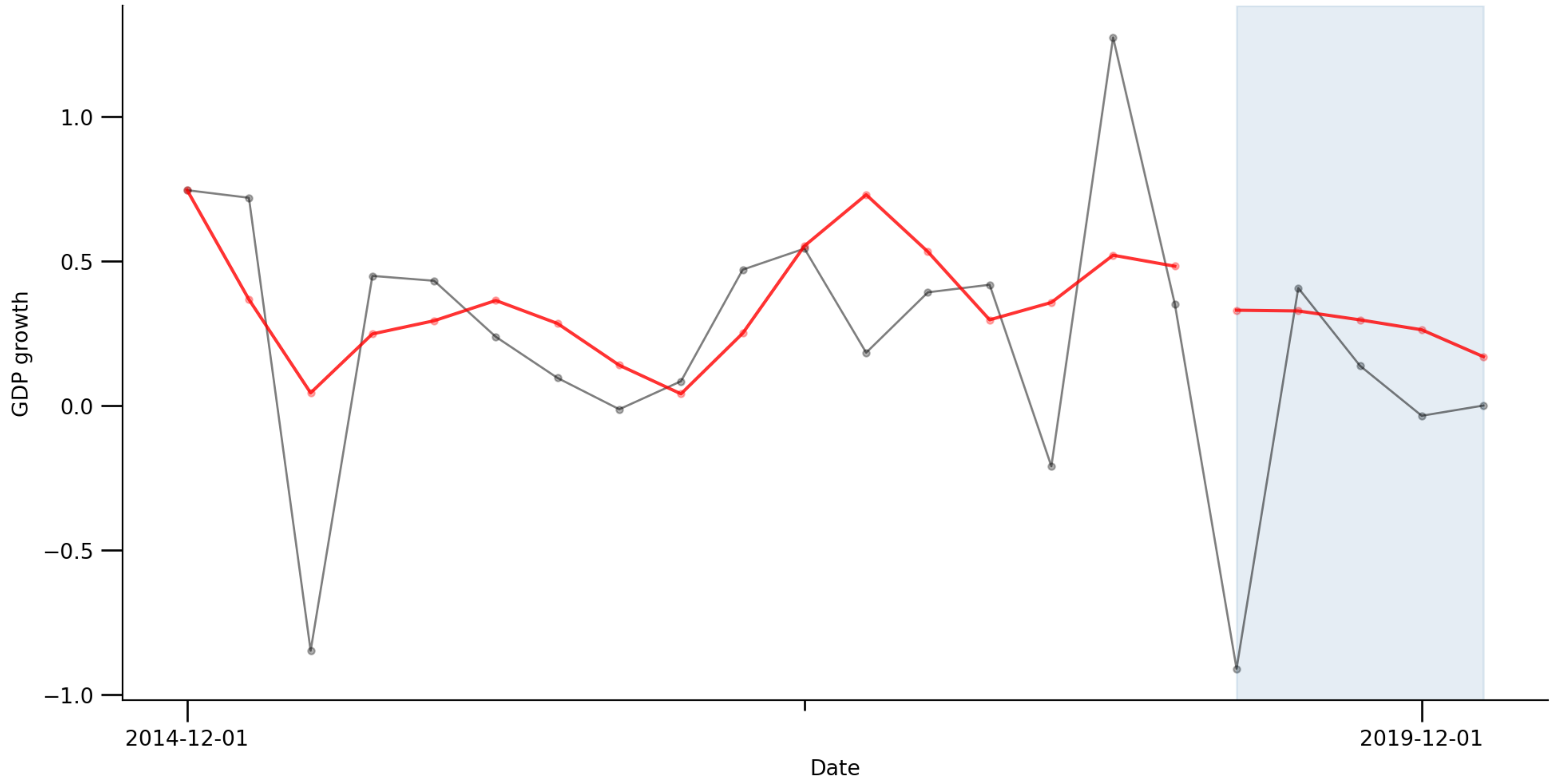

Figure 11: RNN-GRU model forecast (5-periods ahead)

N-BEATS

Intended for univariate time series point forecasting.

Outperformed all other univariate methods in the recent M4 forecasting competition.

N-BEATS: Neural basis expansion analysis for interpretable time-series forecasting

▶ Code

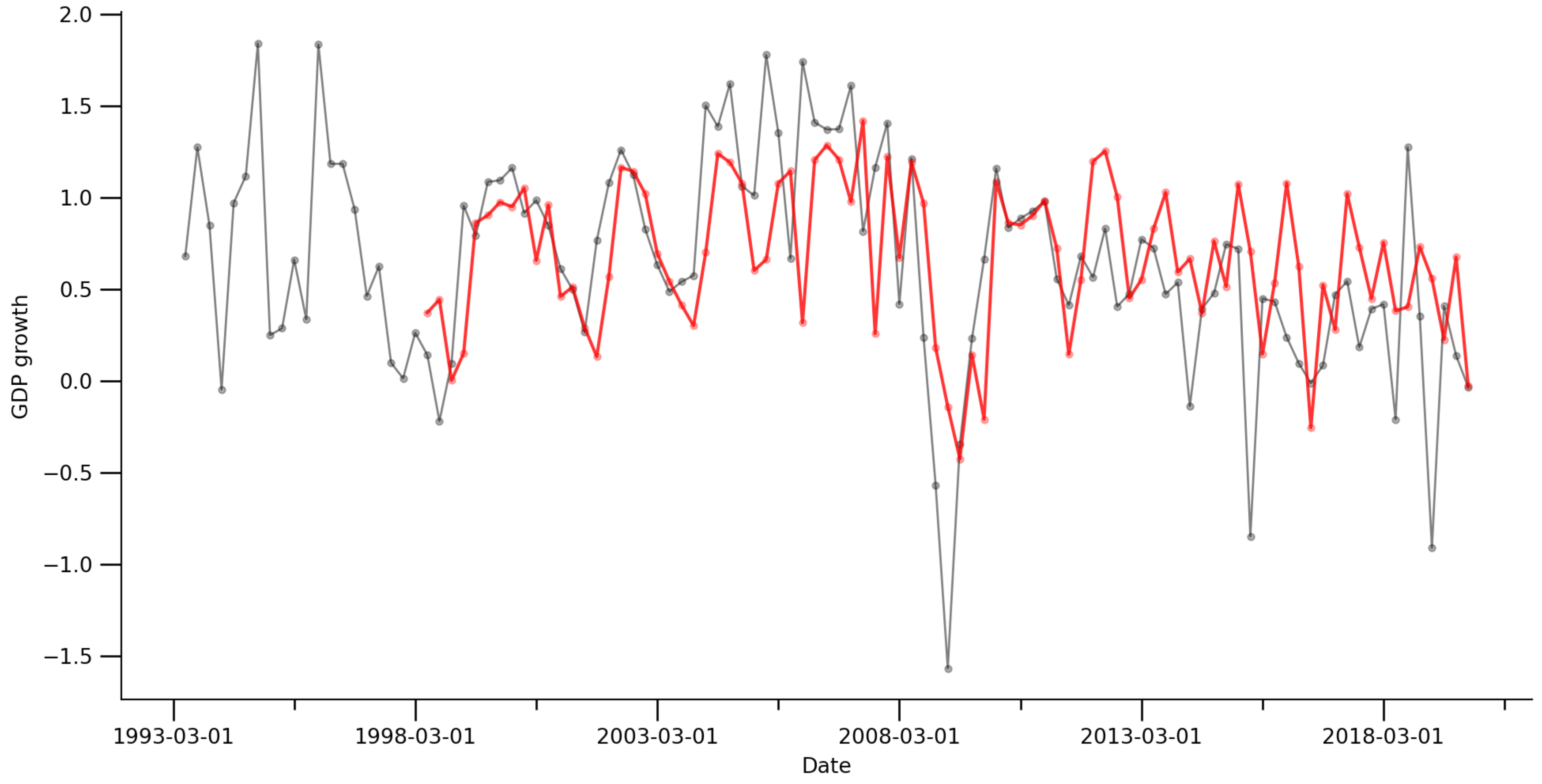

Figure 12: N-BEATS model (univariate)

▶ Code

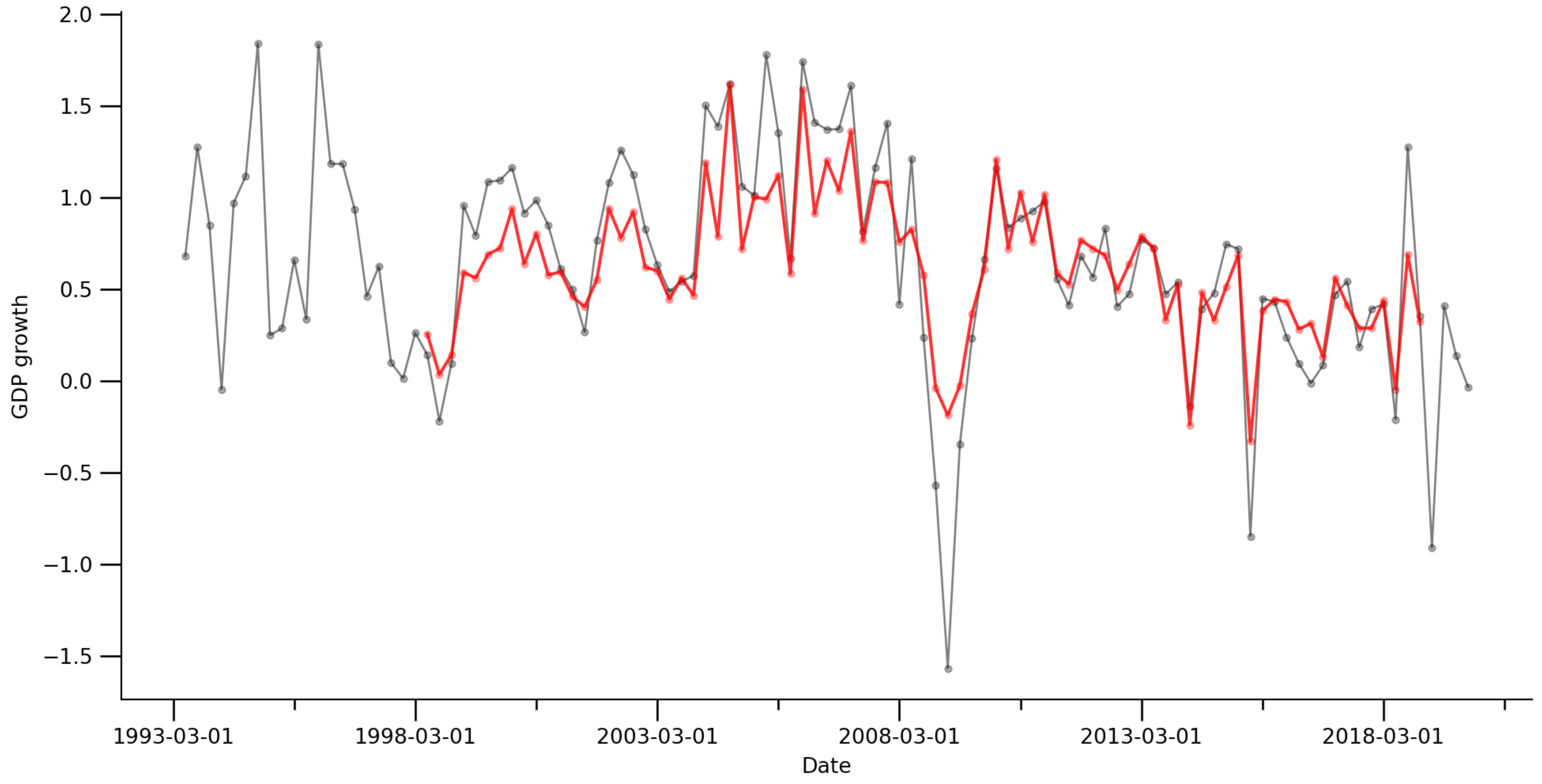

Figure 13: N-BEATSx model (multivariate)

N-BEATSx

N-BEATSx model includes covariates.

```python
model_nbeats = NBEATSModel(
    input_chunk_length=12,
    output_chunk_length=1,
    num_stacks=1,
    num_blocks=2,
    num_layers=3,
    layer_widths=256,
    n_epochs=100,
    random_state = 42,
    add_encoders=encoders,
    dropout=0.3,
    loss_fn=RMSELoss(),
    pl_trainer_kwargs={"accelerator": "gpu",
                       "gpus": 1,
                       "auto_select_gpus": True,
                       "callbacks": [my_stopper]})
```

Would a forecasting practitioner be able to do this easily?

Model involves intensive **hyperparameter tuning.**

In this case the tuning took roughly **22 hours.**

▶ Code

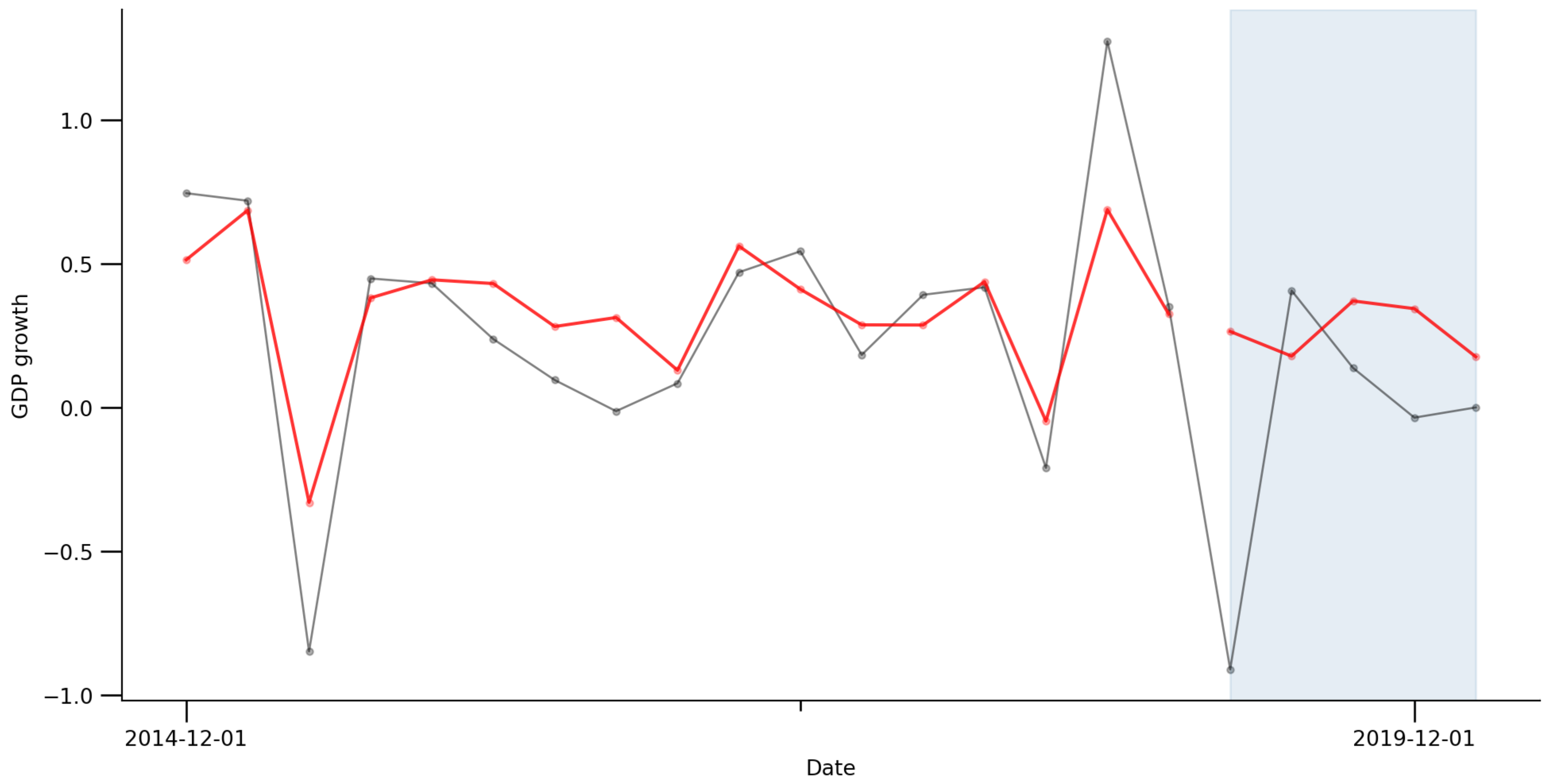

Figure 14: N-BEATSx model forecast (5-periods ahead)

N-HiTS

N-HiTS model has recently shown that it can consistently outperform the N-BEATS model.

▶ Code

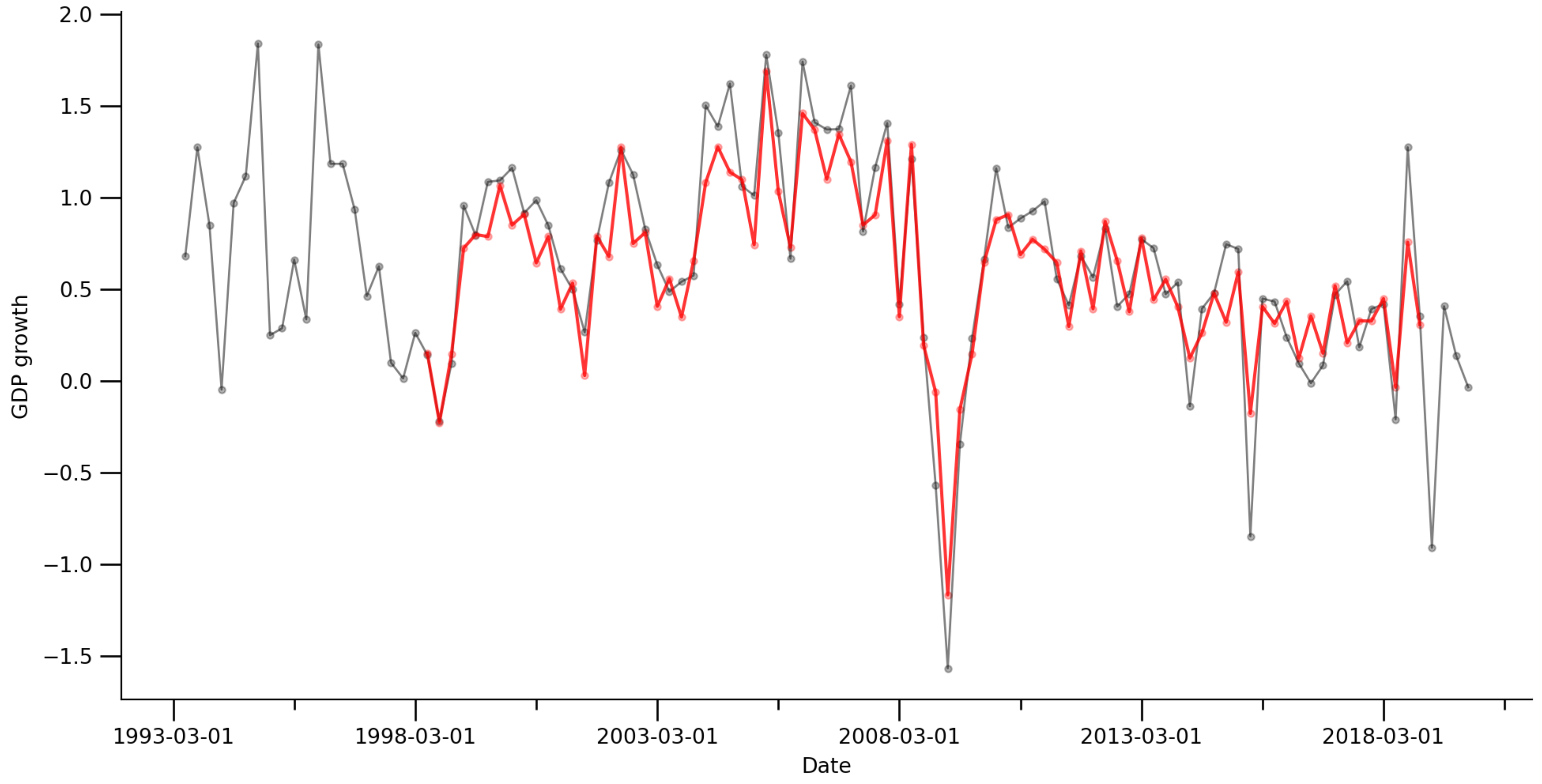

Figure 15: N-HiTS model (multivariate)

▶ Code

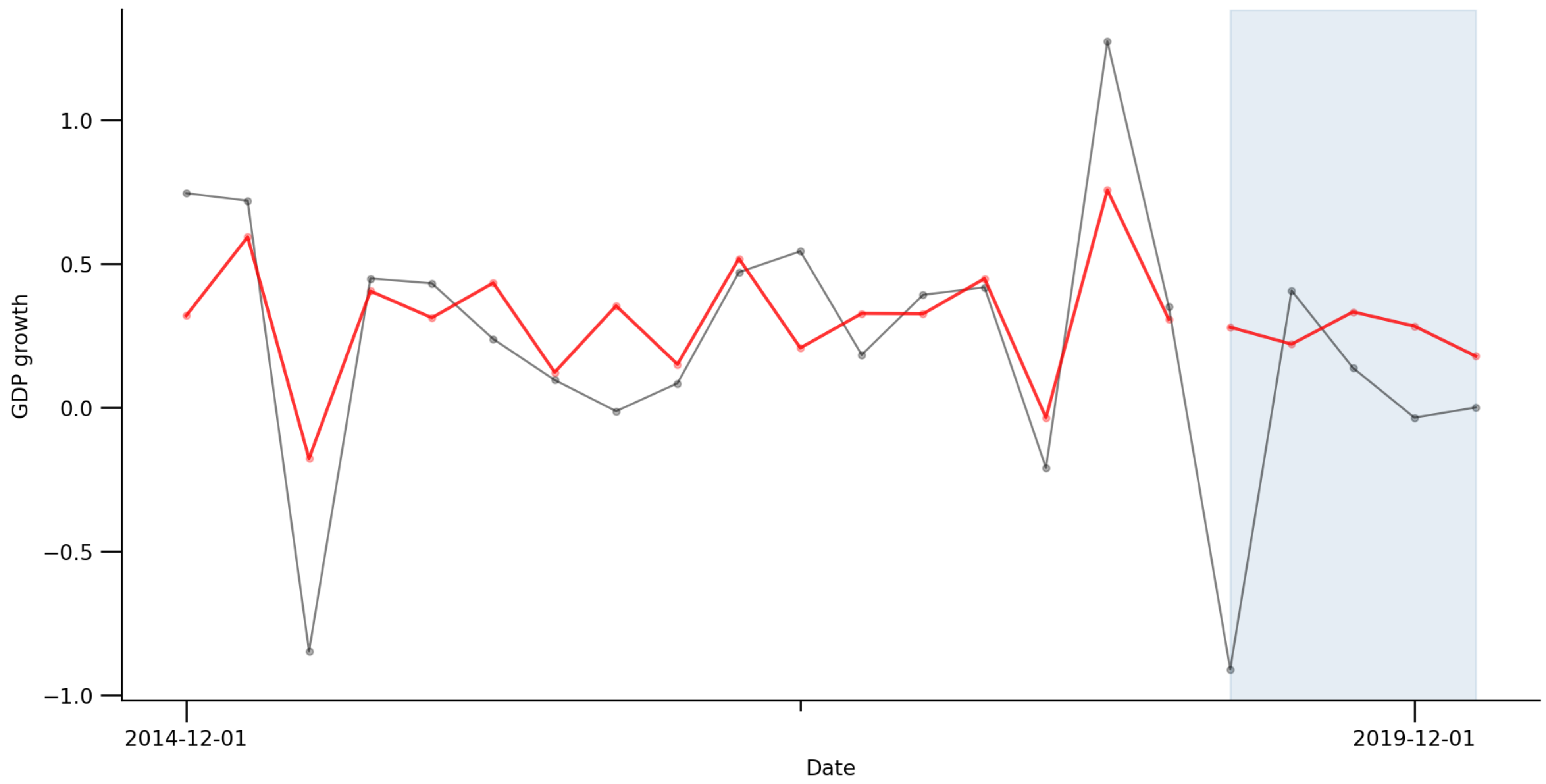

Figure 16: N-HiTS model forecast (5-periods ahead)

TCN

TCN: Temporal convolutional network

▶ Code

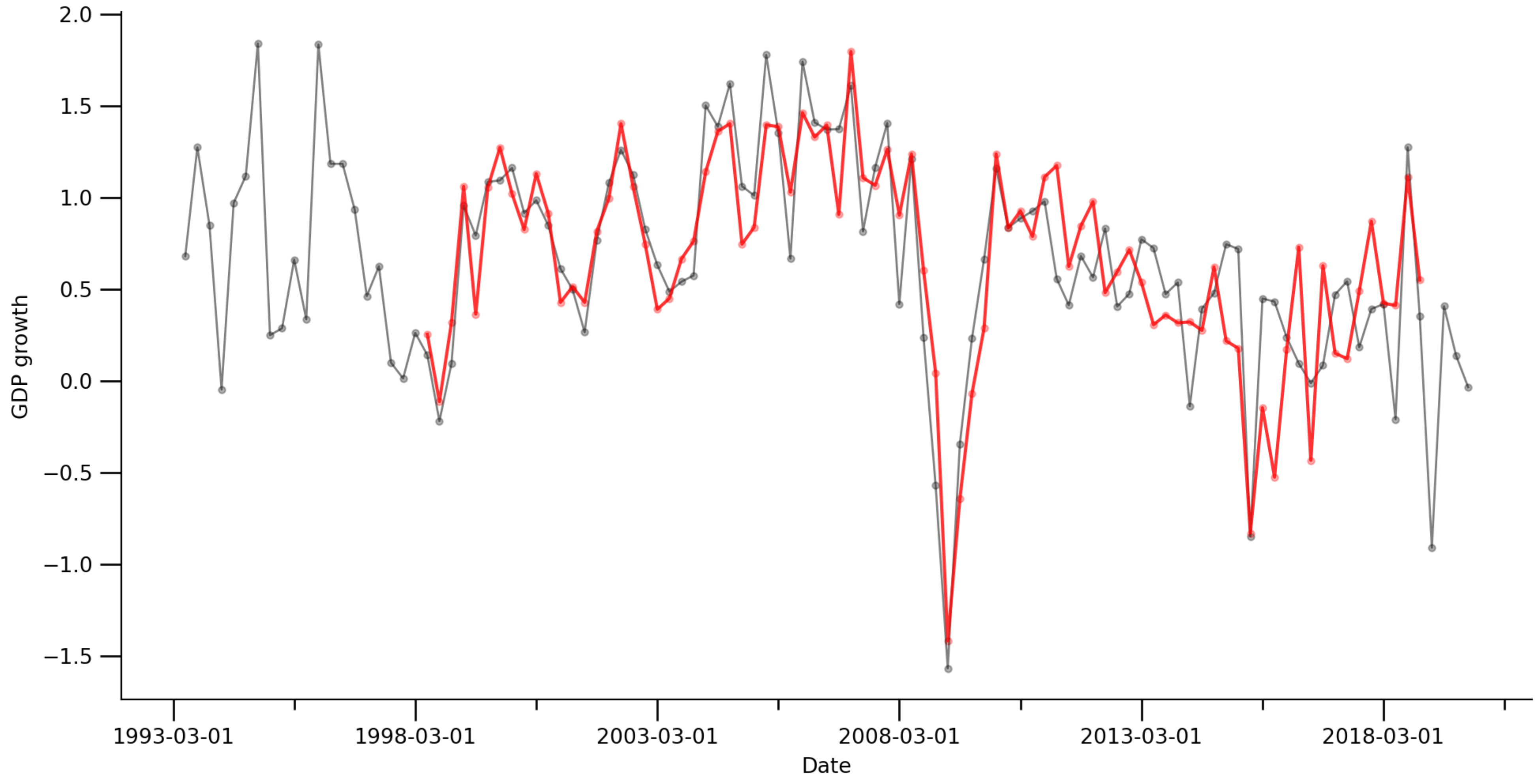

Figure 17: TCN model

▶ Code

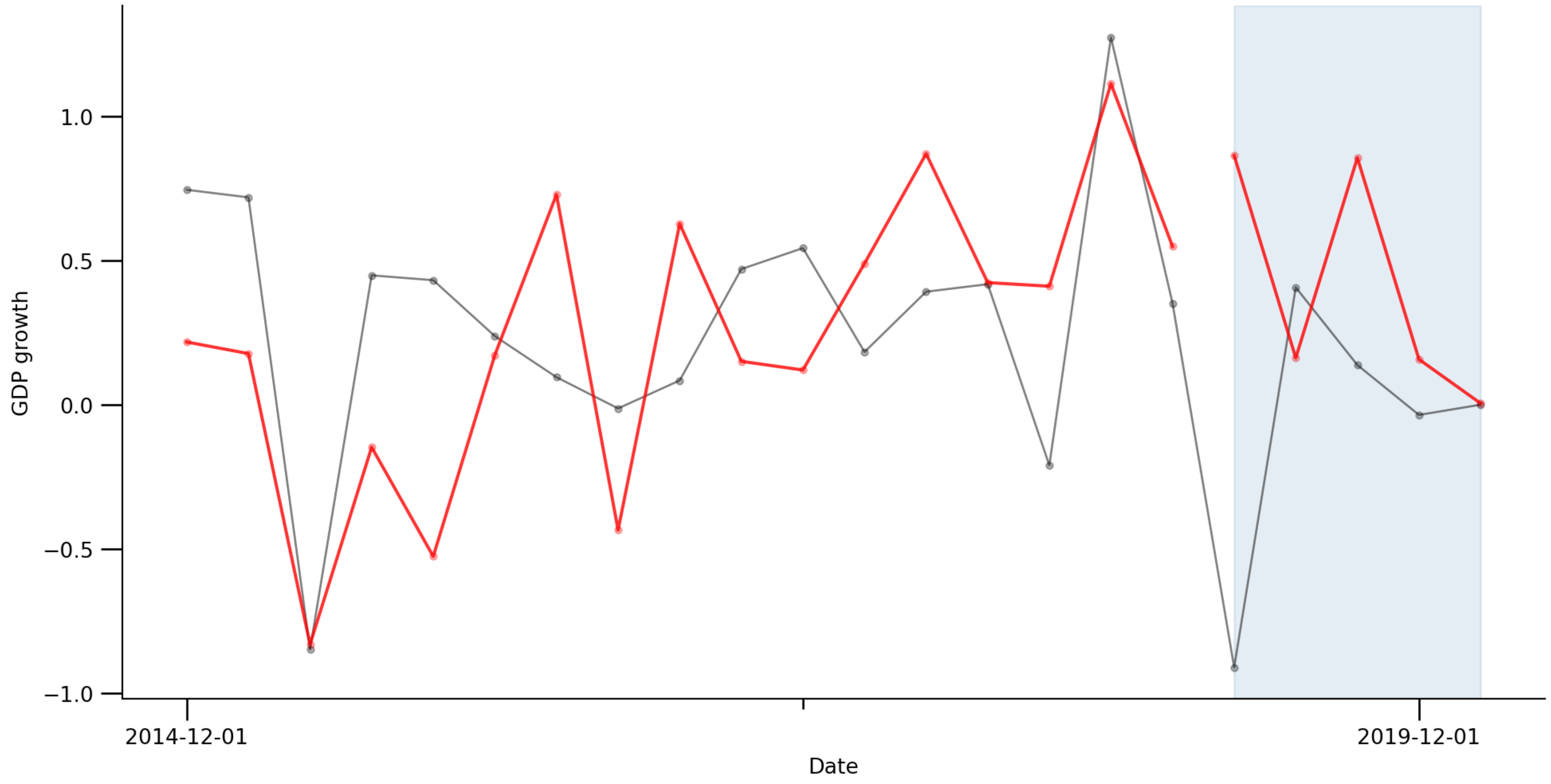

Figure 18: TCN model forecast (5-periods ahead)

Model comparison

▶ Code

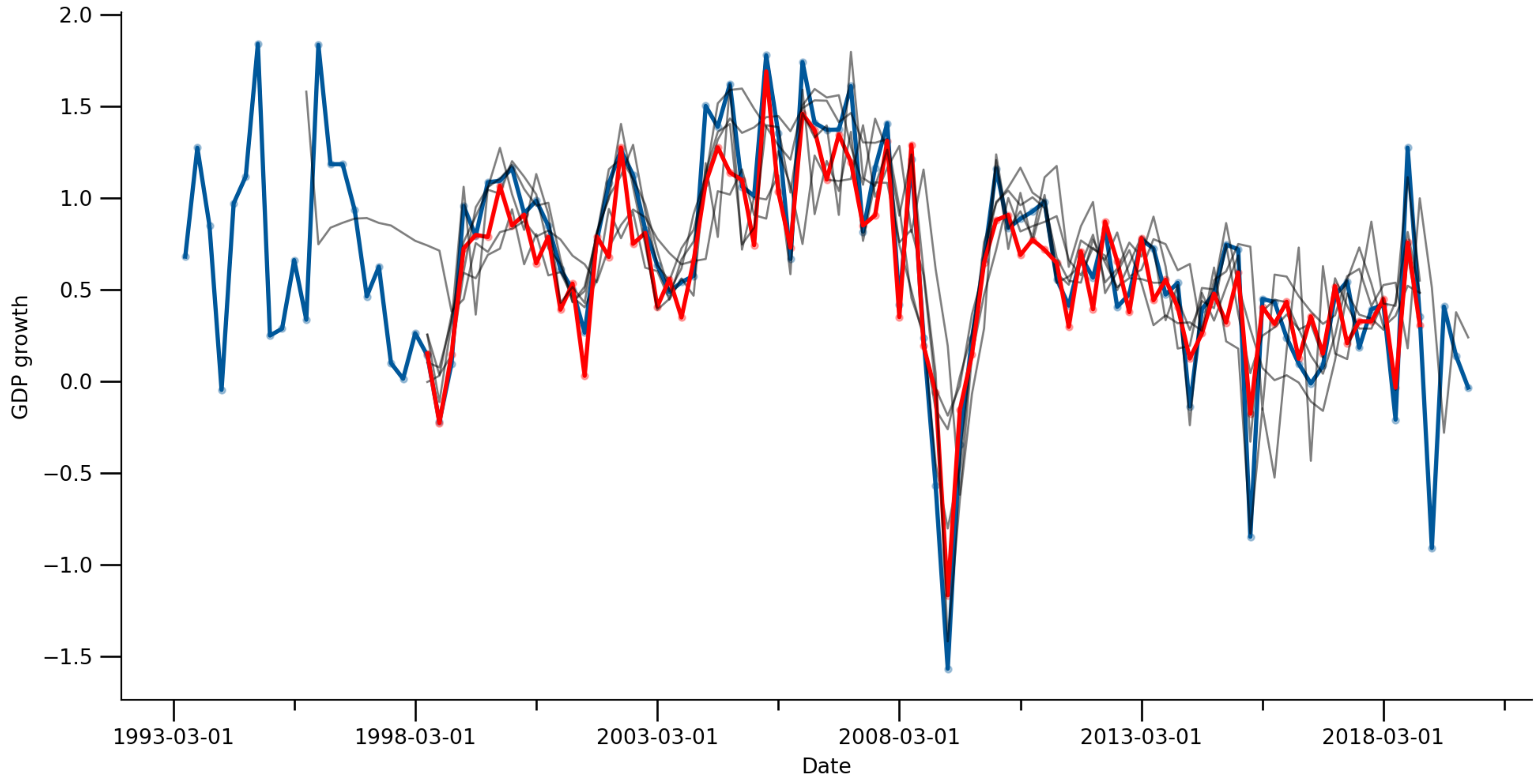

Figure 19: Historical forecast comparison

Model comparison

Here we compare the root mean squared error (RMSE) of the different models.

Lower RMSE is better.

The best performer in terms of this metric on the training data is the **gradient boosting model.**

While the model SVR performs best in terms of the test RMSE.

Model	Train RMSE	Test RMSE
AR(1)	0.53	
SVR	0.27	0.25
RF	0.24	0.26
GBM	**0.23**	0.27
RNN-LSTM	0.33	
RNN-GRU	0.27	
N-BEATS	0.51	
N-BEATSx	0.29	
N-HiTS	0.24	
TCN	0.31	

Life after COVID? 😷

Post-COVID backtest

Post-COVID forecast

Conclusion 🤯

Conclusion

ML and DL models provide improved performance in terms of historical forecasts and future prediction.

Possible to automate components of the process, but expert opinion is required.

Hyperparameter selection is **difficult**

- This process is more art than science
- Can be computationally expensive
- Potential for overfitting

Future plans 🗺

In progress 🚧

Mixed data strategies.

Post-COVID period forecast.

Solution for missing data.

Finer grids for hyperparameter search.

In future 🔮

Impact of including alternative data.

Probabilistic models.

Temporal fusion transformers.

DFM and MF-VAR models.

Real-time data.

In progress 🚧

Mixed data strategies.

Post-COVID period forecast.

Solution for missing data.

Finer grids for hyperparameter search.

In future 🔮

Impact of including alternative data.

Probabilistic models.

Temporal fusion transformers.

DFM and MF-VAR models.

Real-time data.

Questions? 💡