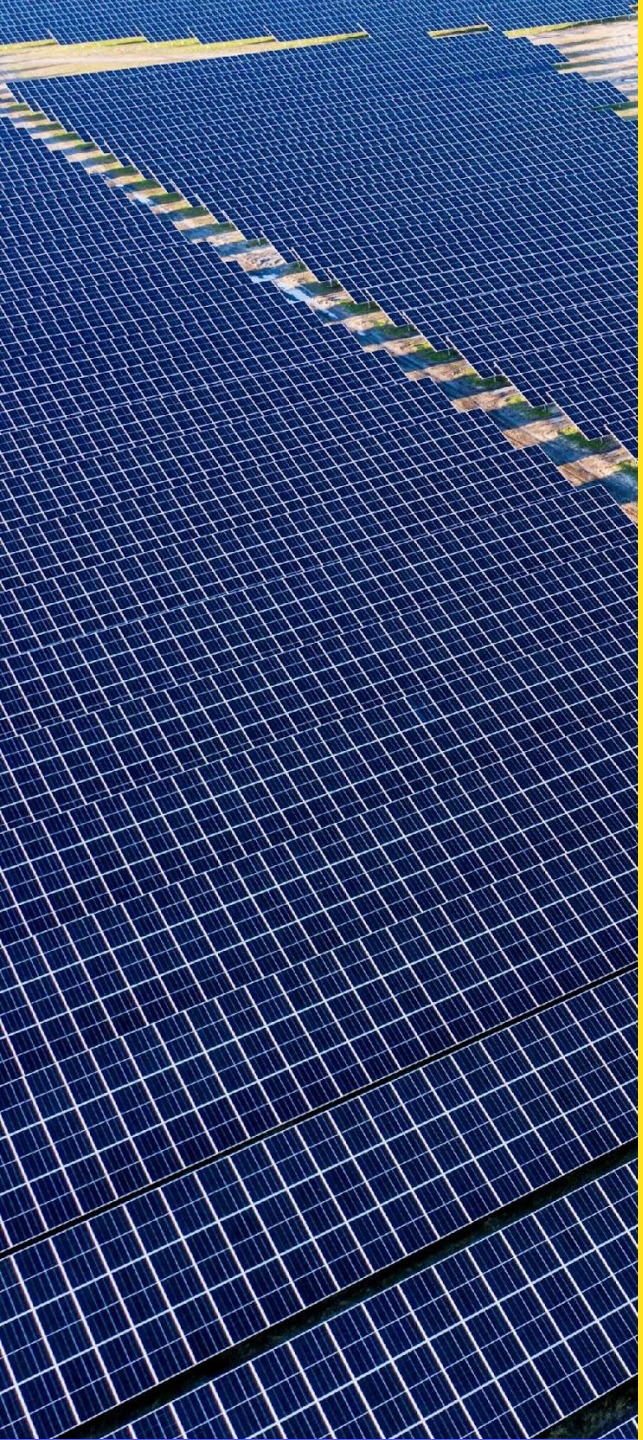


Learning from Light: Advanced Time Series Analysis for German Solar Energy Forecasting

Motivation:

- (cheap) energy supply crucial for industries
- more and more energy from renewable, weather dependant sources
- rising volatility of supply and pricing

Solar power:

- high seasonality: day/night, yearly
- upwards trend: growing expansion
- sensitive to weather phenomena

Motivation:

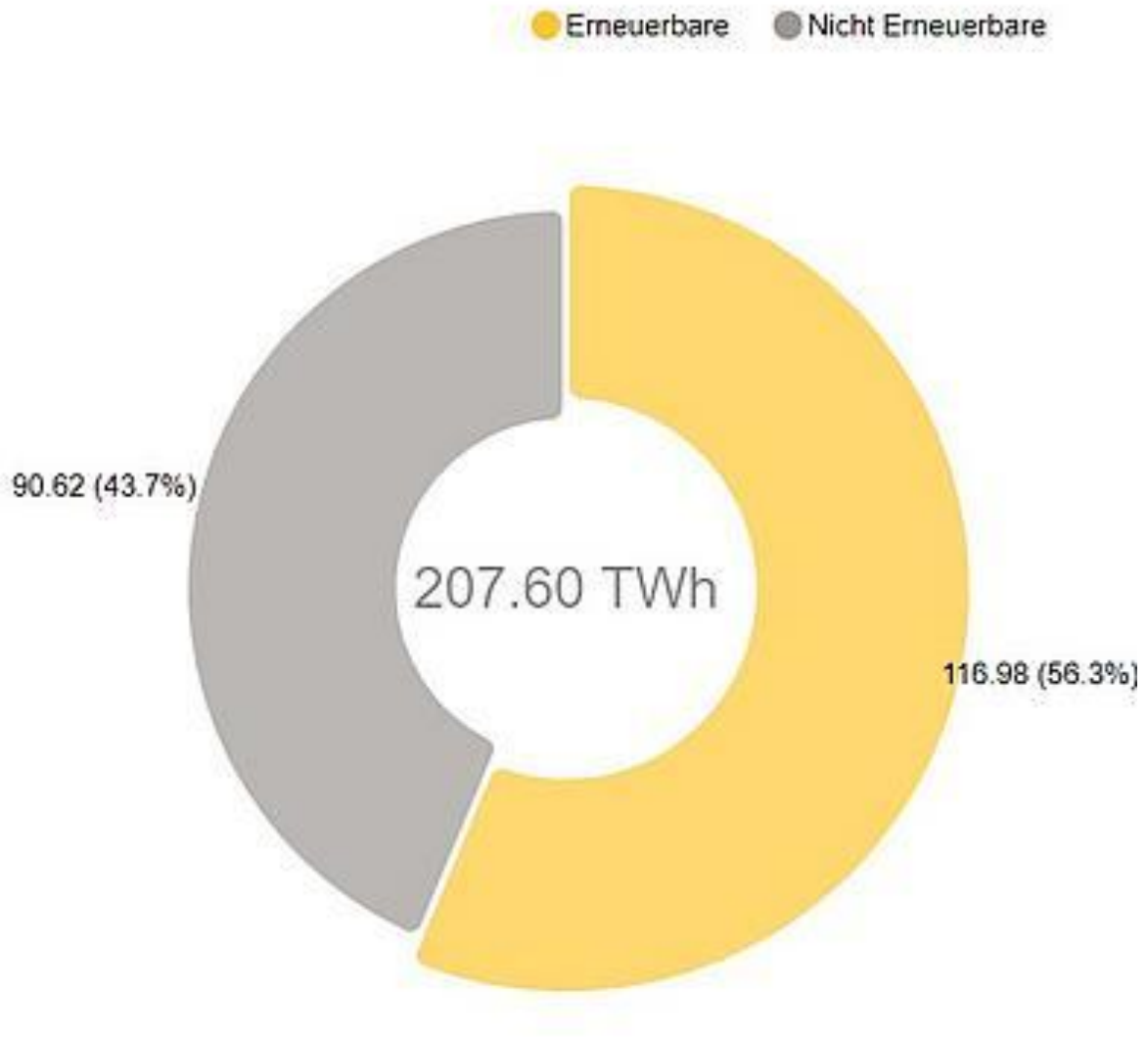
- (cheap) energy supply crucial for industries
- more and more energy from renewable, weather dependant sources
- rising volatility of supply and pricing

Solar power:

- high seasonality: day/night, yearly
- upwards trend: growing expansion
- sensitive to weather phenomena

Nettoerzeugung von Kraftwerken zur öffentlichen Stromversorgung.
Datenquelle: 50 Hertz, Amprion, Tennet, TransnetBW, Destatis, EEX
letztes Update: 02 Jun 2020 17:18

Net Electricity Generation Germany (2020)



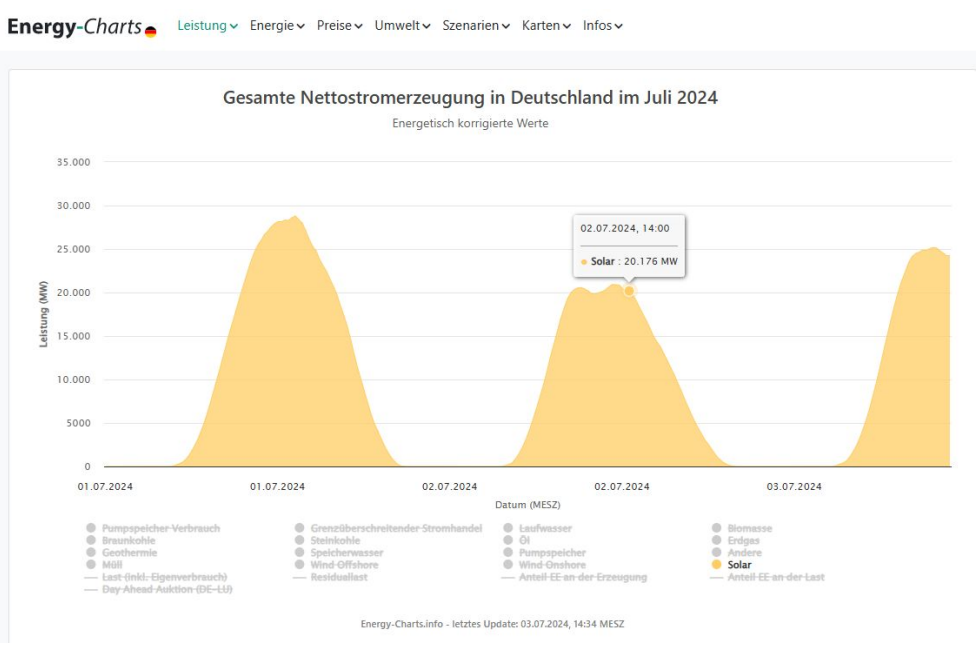
Data Sources:

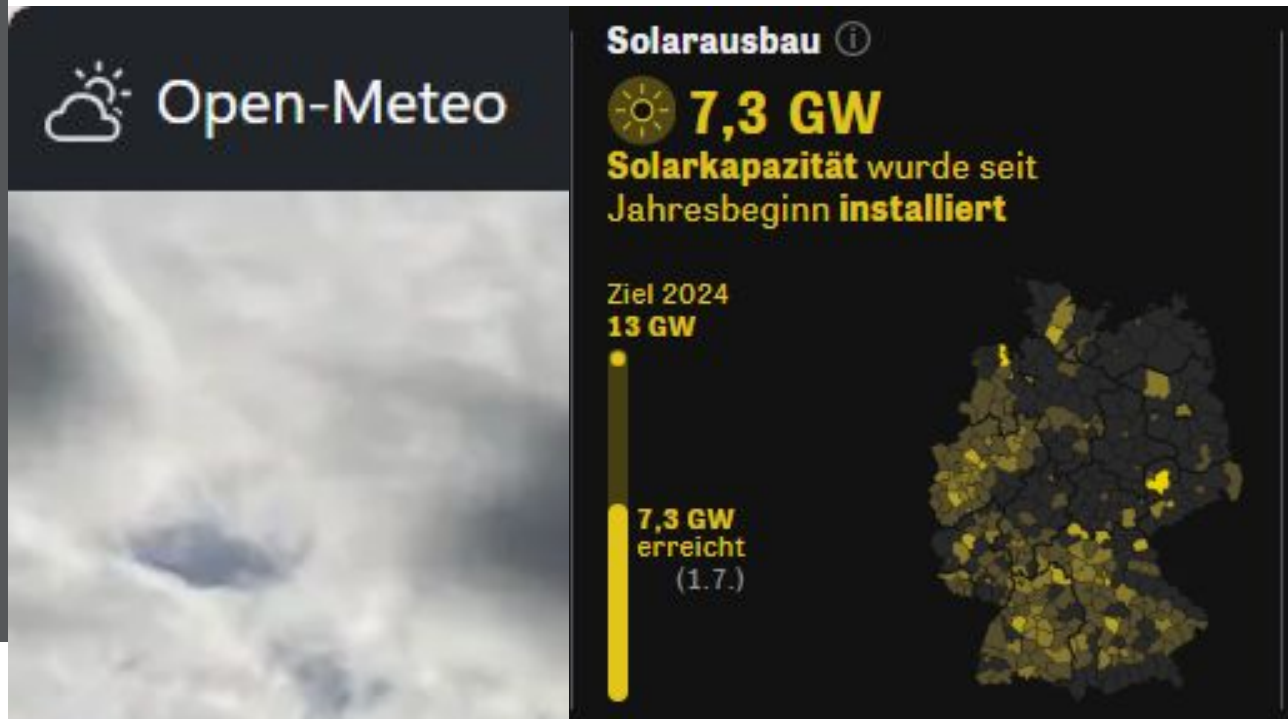
Main time series: Stromproduktion | Energy-Charts

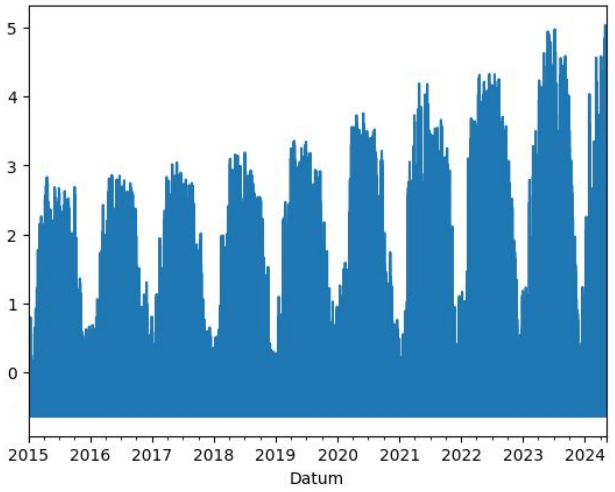
Exogene weather data: 🏛️ Historical Weather API | Open-Meteo.com

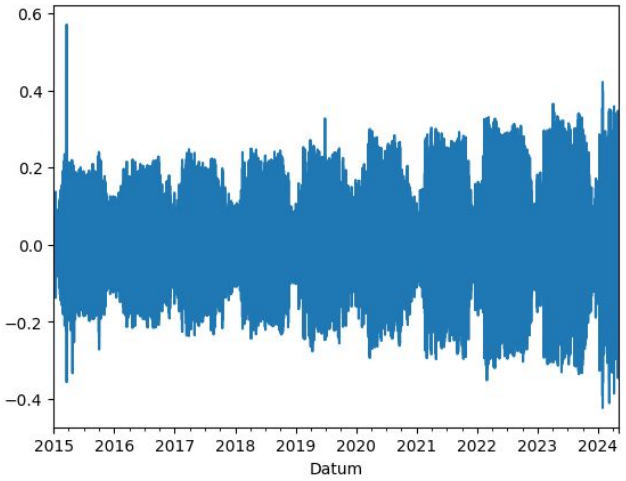
Selection of 6 cities as representatives of the German weather, chosen for their proximity to important solar hubs according to Energiemonitor: Die wichtigsten Daten zur Energieversorgung – täglich aktualisiert | ZEIT ONLINE

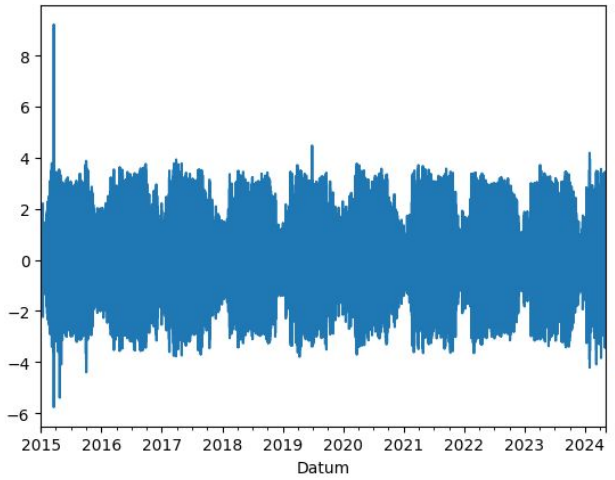
- Templin
- Kastellaun
- Gütersloh
- Ingolstadt
- Erfurt
- Neumünster

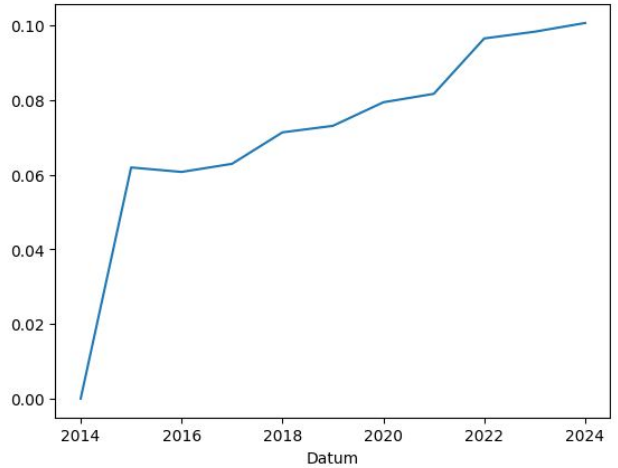








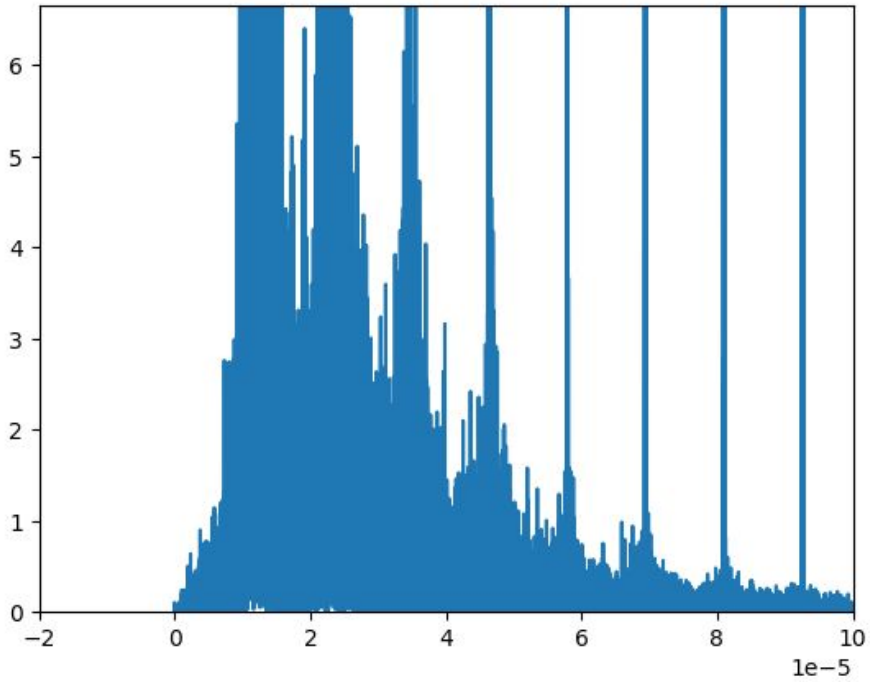


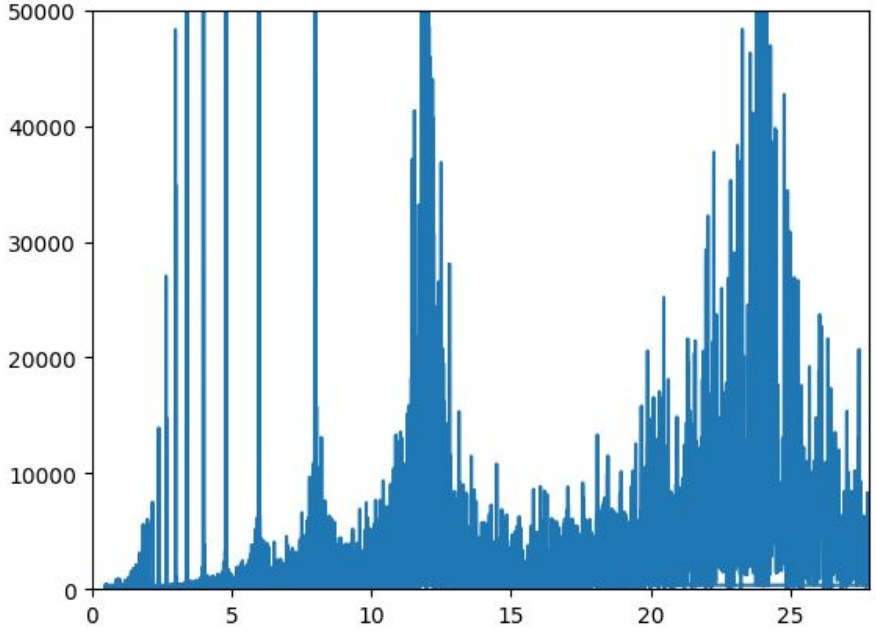


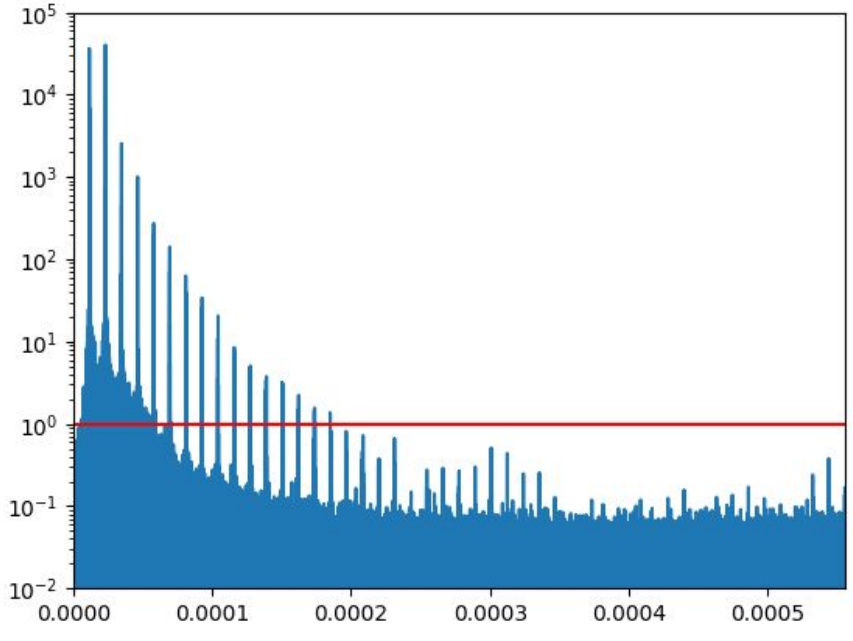
Preprocessing: Achieve stationarity

* standardize
* differentiate (eliminate trend)
* divide by annual volatility (eliminate increase in volatility)

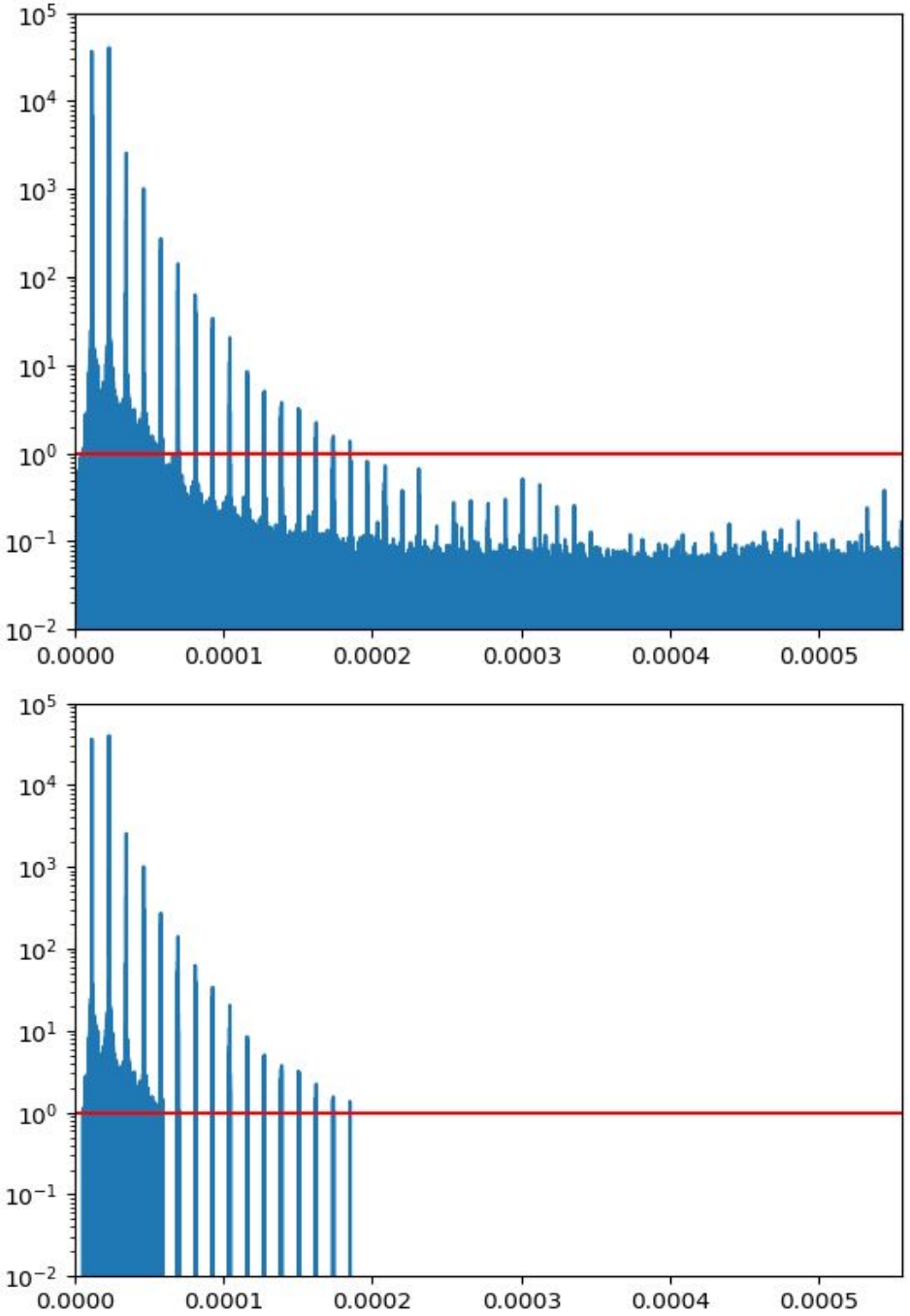
Preprocessing: Fourier analysis

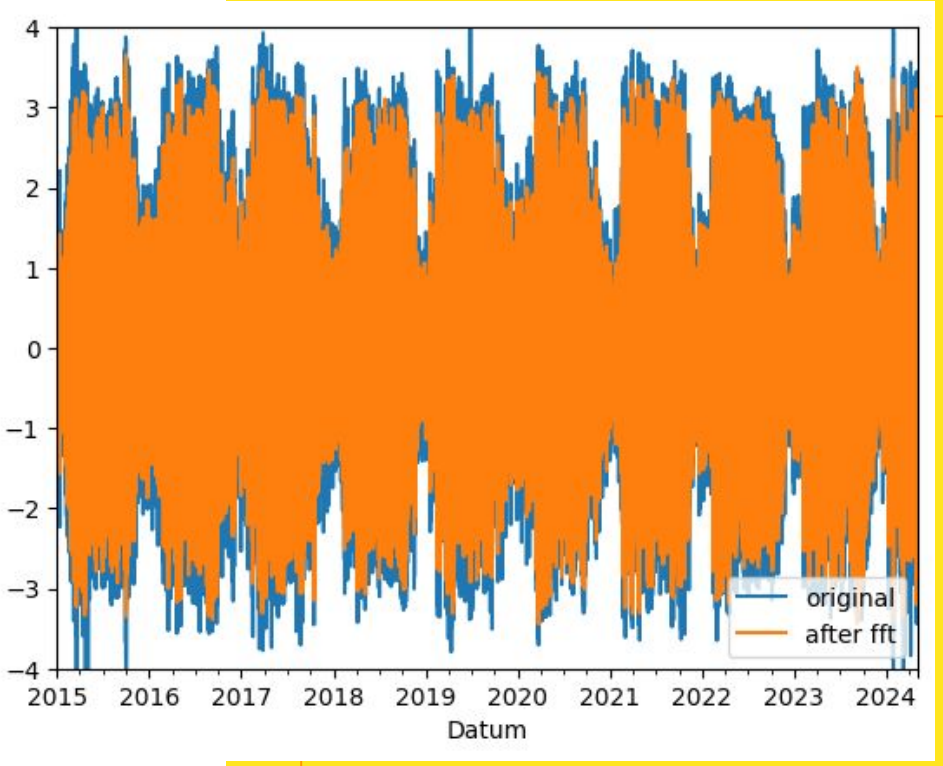






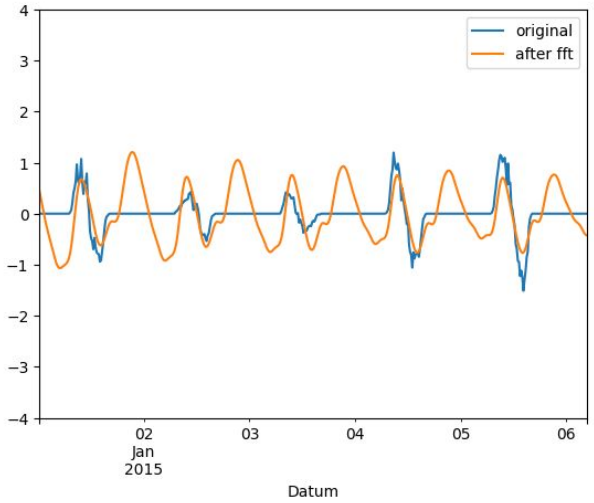
Preprocessing: Fourier analysis



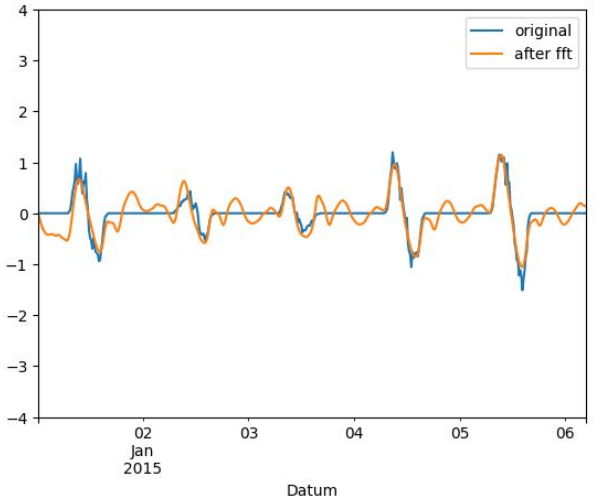


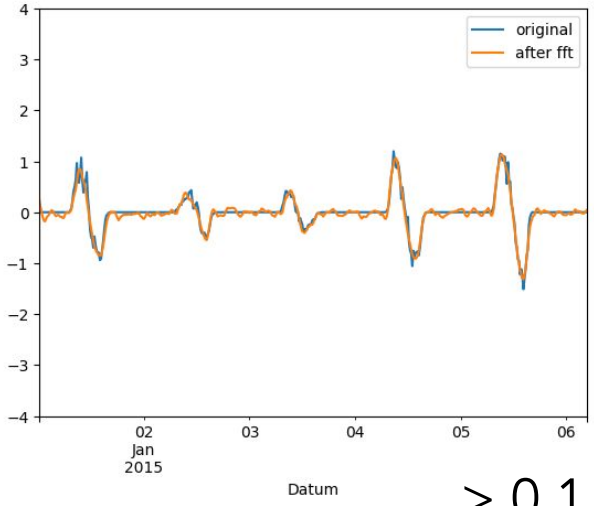
Preprocessing: Fourier analysis

> 10



> 1



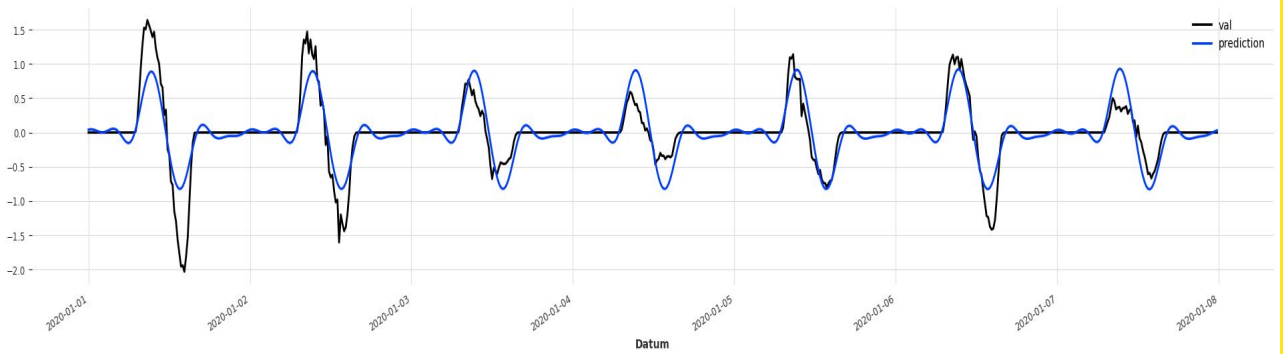


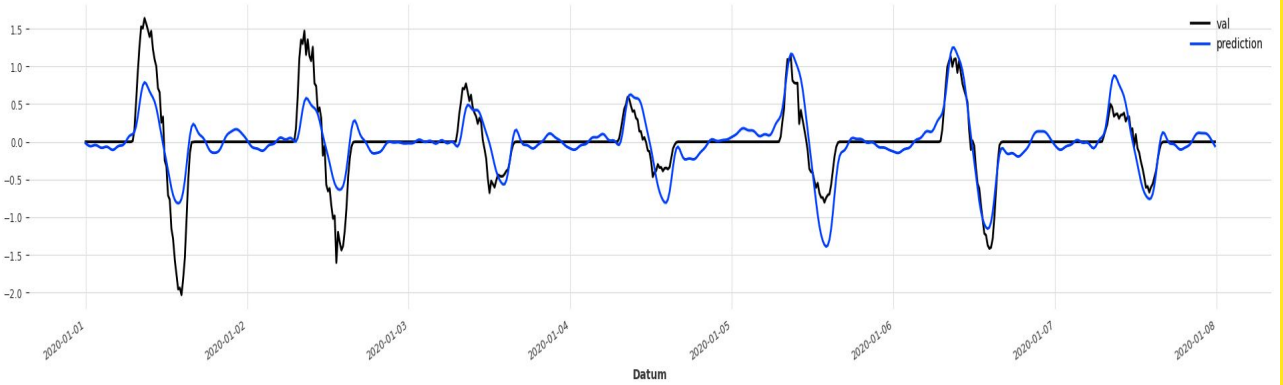
> 0.1

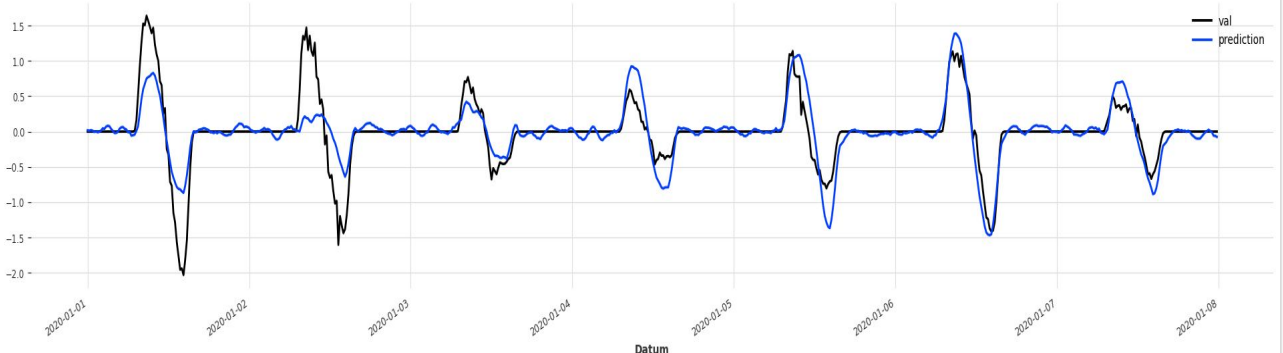
power cutoff and # surviving frequencies:

keep > 10	~ 1700
keep > 1	~ 9000
keep > 0.1	~ 13000

performance, quality of fit

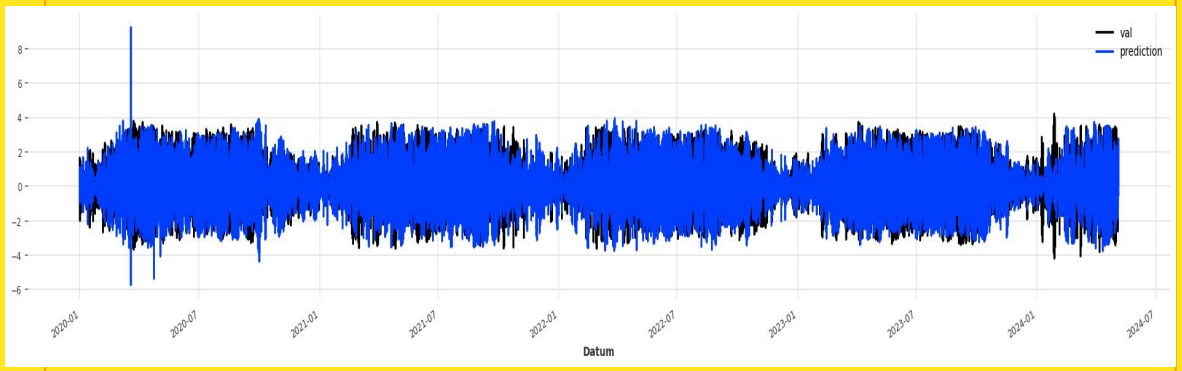




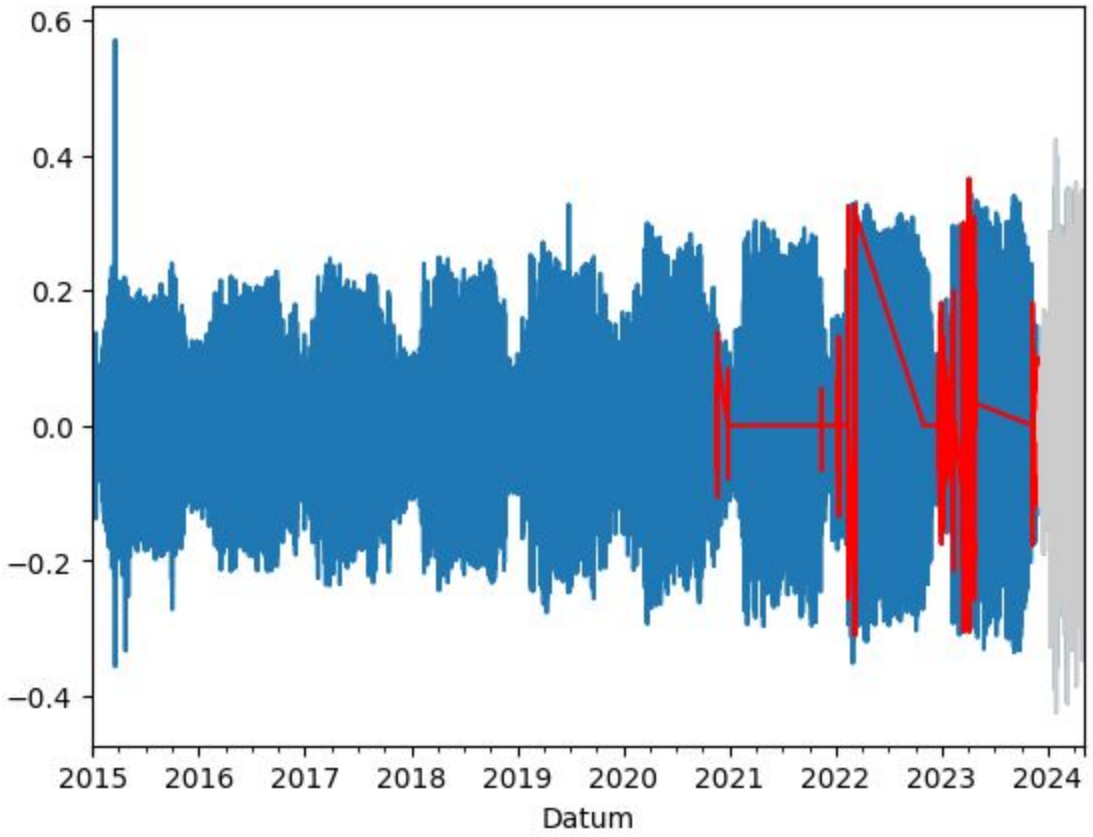


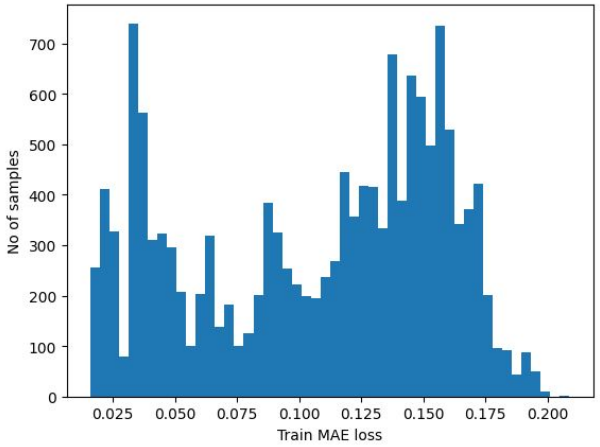
Univariate Models: Fourier analysis

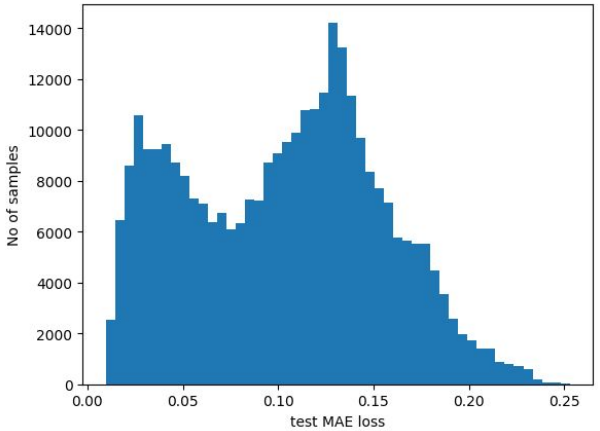
keep
* 50 frequencies
* 5000 frequencies
* 15000 frequencies




```
[56] import darts.metrics.metrics

     model = FFT()
     model.gridsearch(
      parameters={
           "nr_freqs_to_keep": [10, 50, 100, 500, 1000, 2000, 5000, 10000, 15000, 20000]
       },
       series=darts_ts_train,
       val_series=darts_ts_test,
       verbose=True,
       metric = darts.metrics.metrics.mae
     )

     100%|██████████| 10/10 [04:31<00:00, 27.12s/it]
     (FFT(nr_freqs_to_keep=100, required_matches=None, trend=None, trend_poly_degree=3),
      {'nr_freqs_to_keep': 100},
      0.21703434599224633)
```



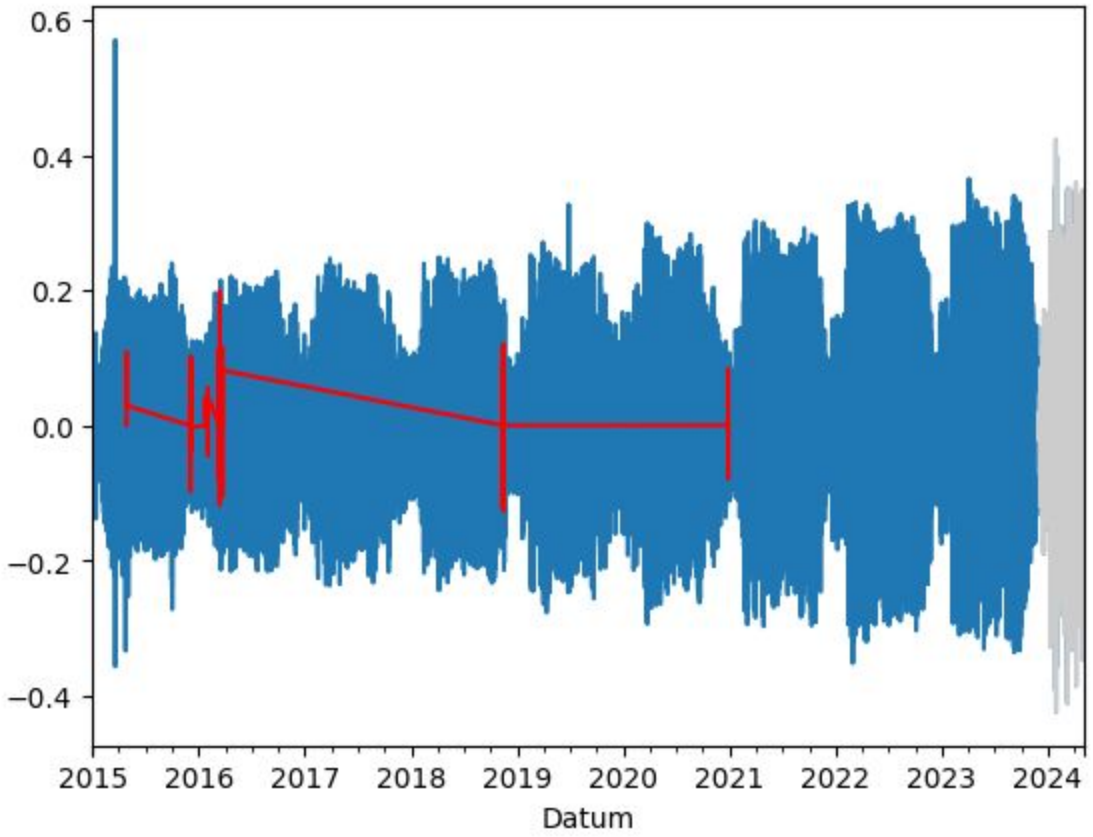


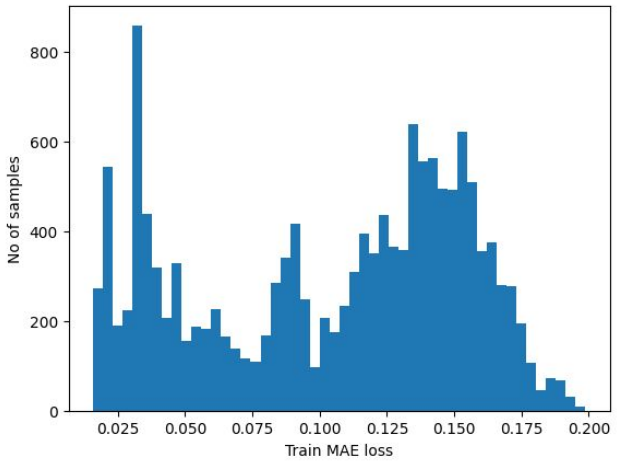


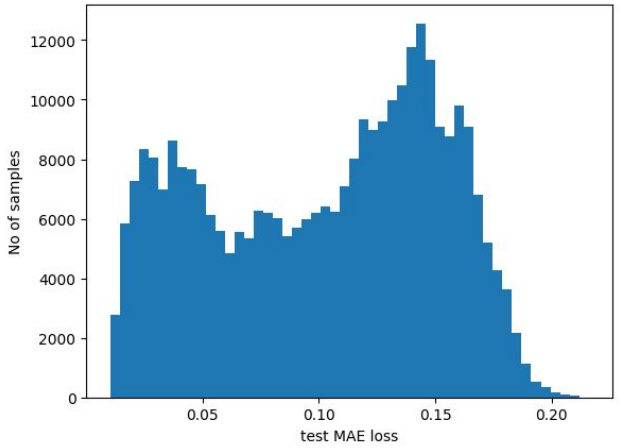
Preprocessing: Anomaly Detection

* slice data into smaller vectors (288 elements), rolling slices
* train autoencoder
* determine cutoff based on reconstruction error histogram (here: 90% of max. reconstruction error)
* classify sample as anomalous if in >90% anomalous slices

autoencoder: simple CNN, 5000 training samples



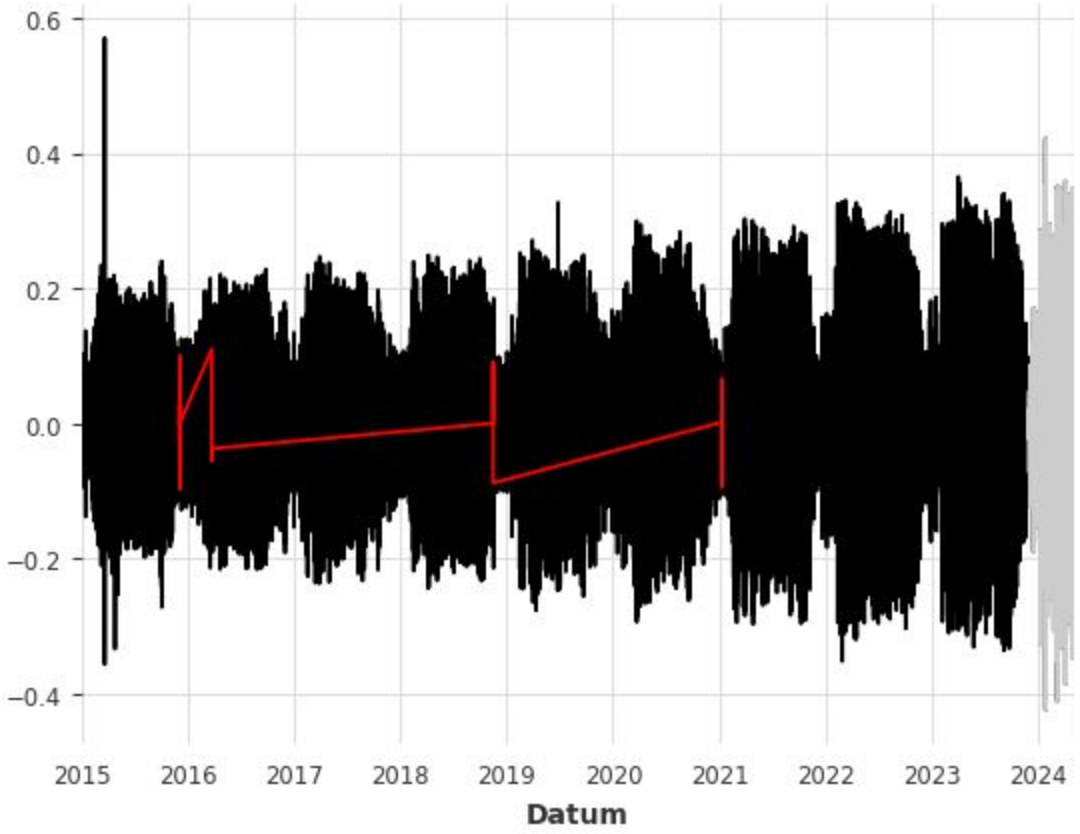


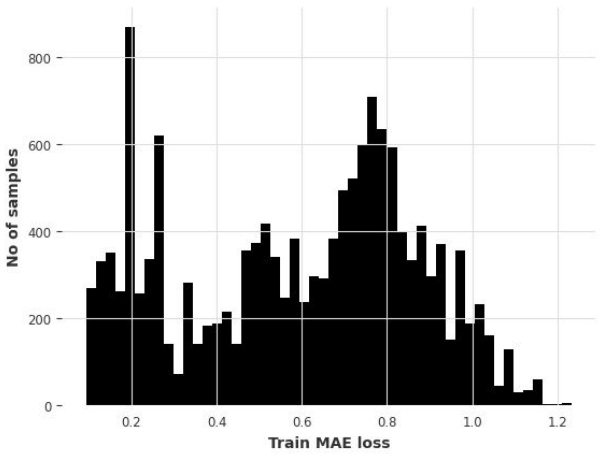


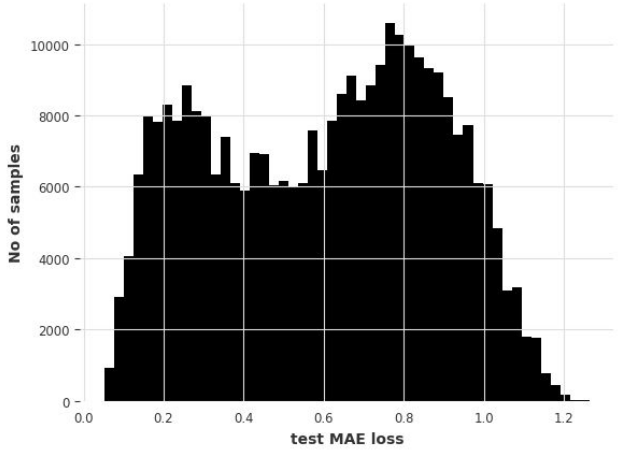
Preprocessing: Anomaly Detection

* slice data into smaller vectors (288 elements), rolling slices
* train autoencoder
* determine cutoff based on reconstruction error histogram (here: 90% of max. reconstruction error)
* classify sample as anomalous if in >90% anomalous slices

autoencoder: simple CNN, 15000 training samples



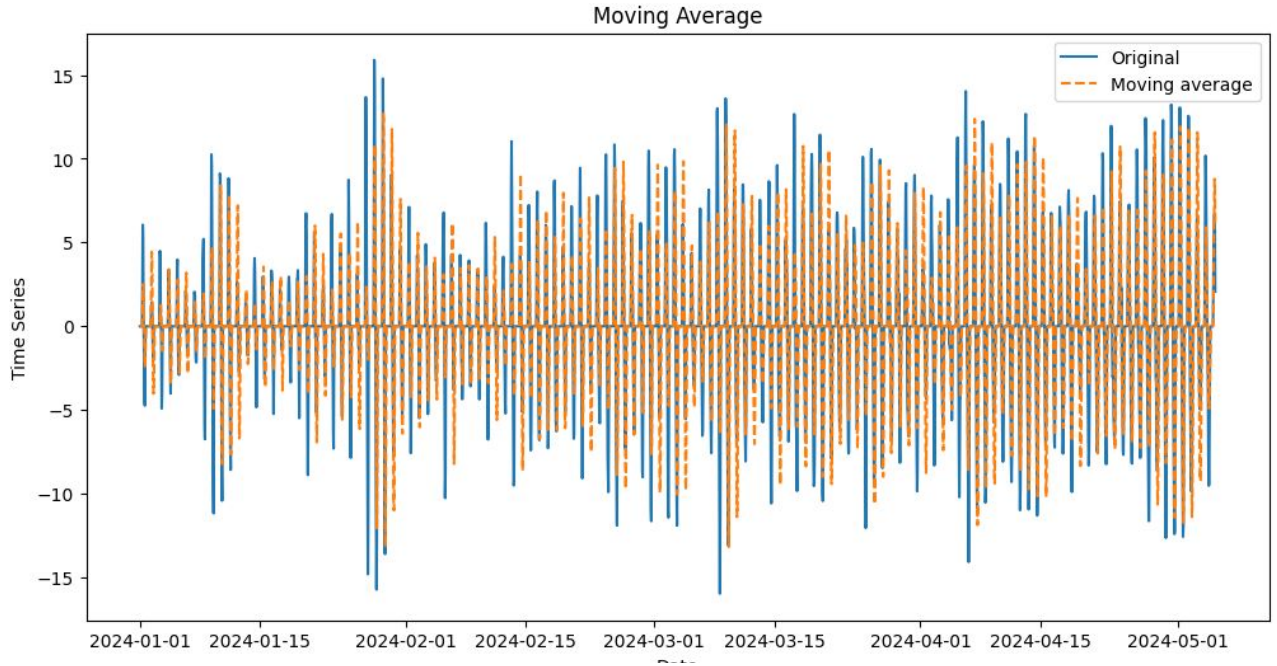


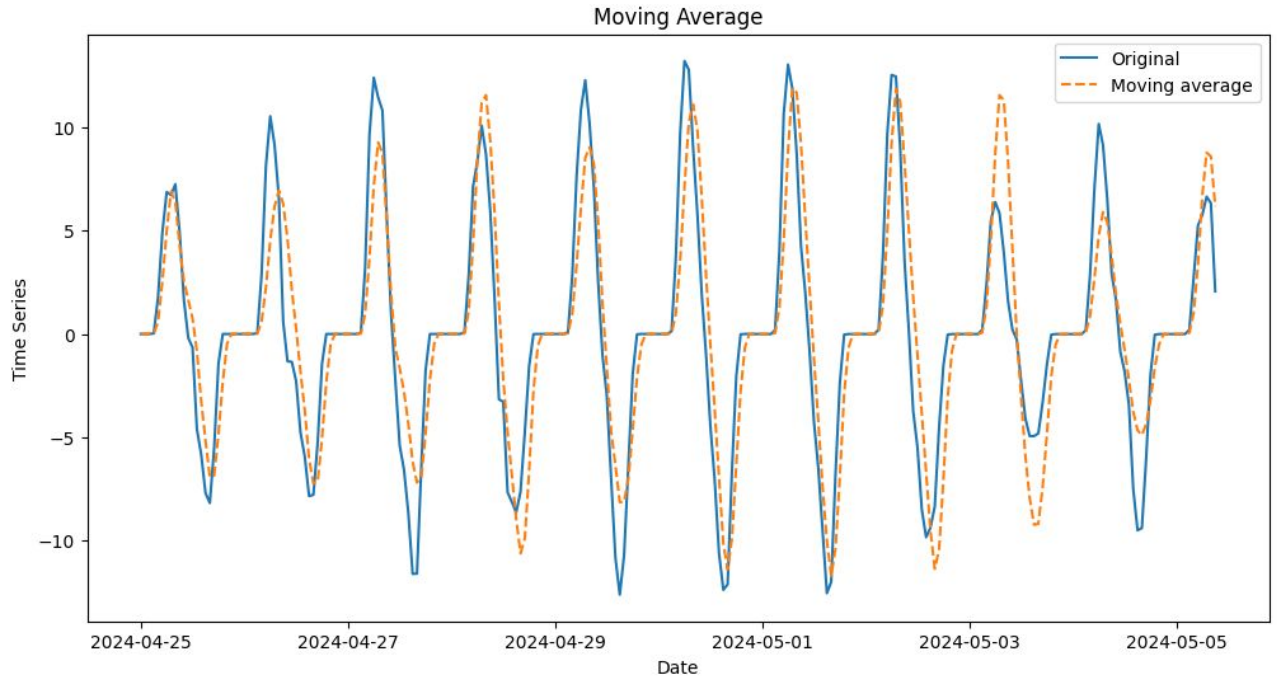


Preprocessing: Anomaly Detection

* slice data into smaller vectors (288 elements), rolling slices
* train autoencoder
* determine cutoff based on reconstruction error histogram (here: 90% of max. reconstruction error)
* classify sample as anomalous if in >90% anomalous slices

autoencoder: LSTM, 15000 training samples





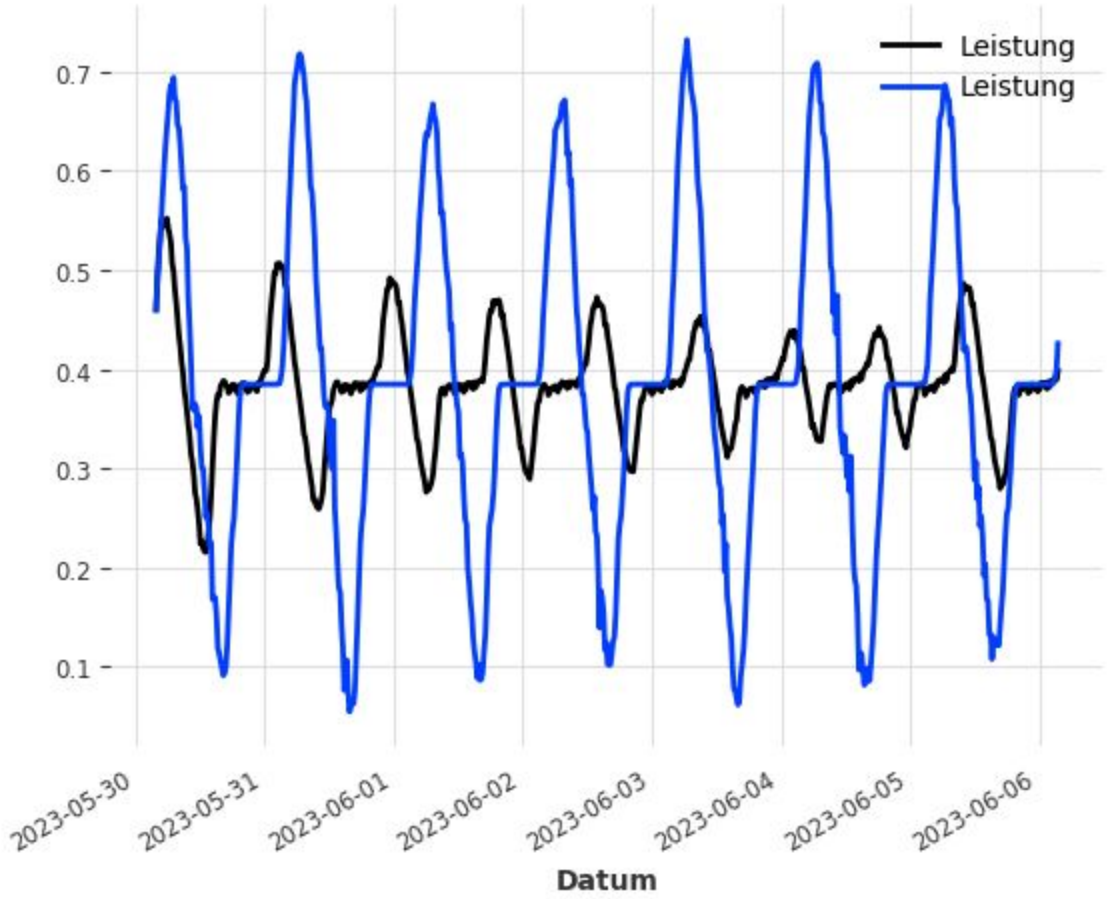
Univariate Models - MOVING AVERAGE

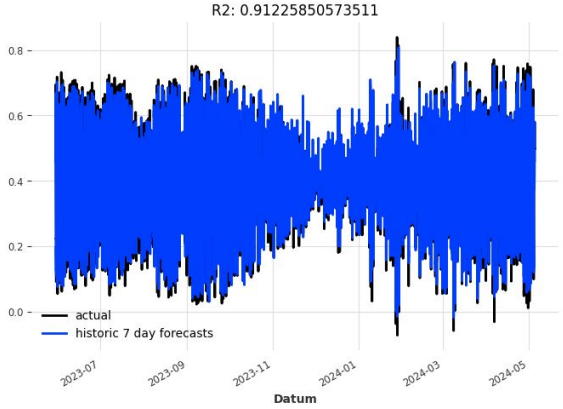
```python
# Moving average model
def moving_average(data: pd.DataFrame, window_size: int=3, shift_size: int=24):
    moving_avg = data.rolling(window=window_size).mean()
    shifted_moving_avg = moving_avg.shift(shift_size)
    return(shifted_moving_avg)
```

Base Model: Moving Average

Mean of yesterday's past 3 hours

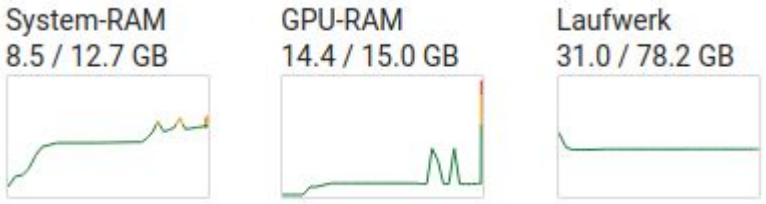
Model: Naive Moving Average
Mean absolute error: 1.480194236184758
Mean absolute percentage error: 181213660399480.06
Mean squared error: 5.952003477077344
r2_score: 0.6150573857393982
Root mean squared error: 2.439672821727812





Univariate Models - NBEATS

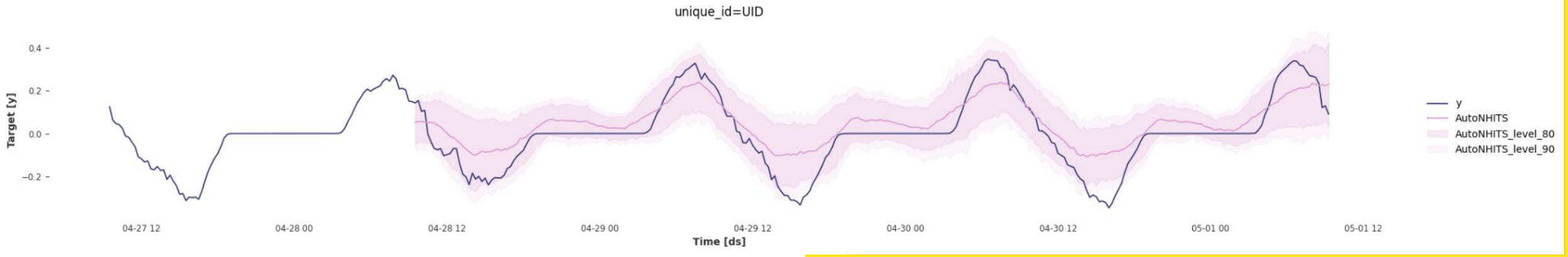
NBEATS

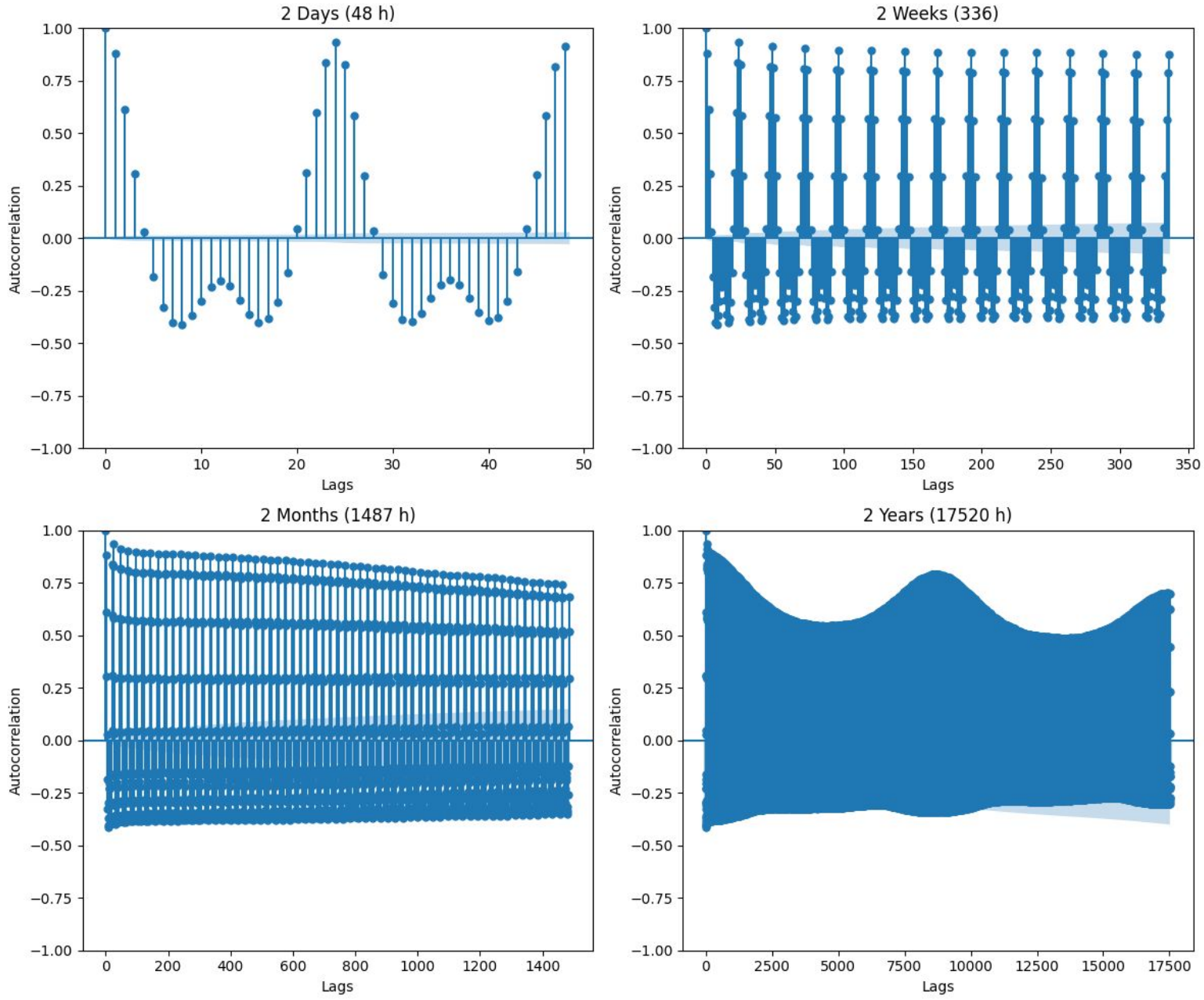


Univariate Models - NHITS

NHITS

* actual series is within the confidence interval
* high GPU RAM usage ⇒ unclear behavior on longer series

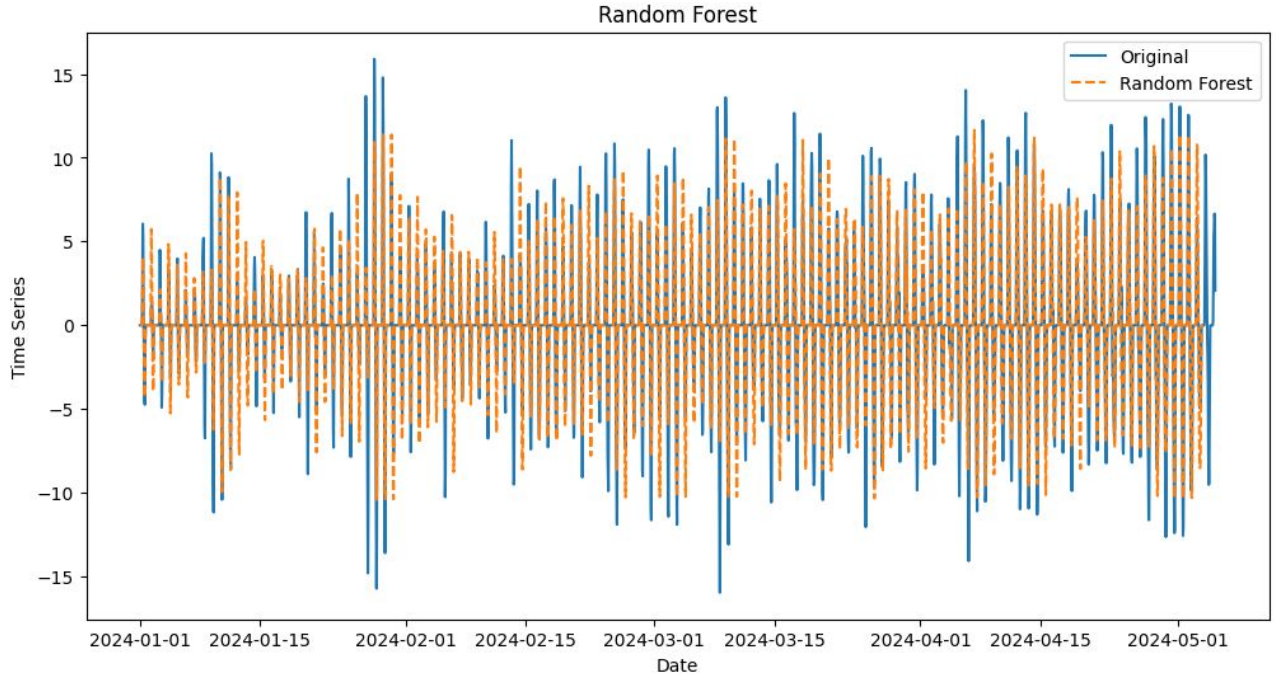


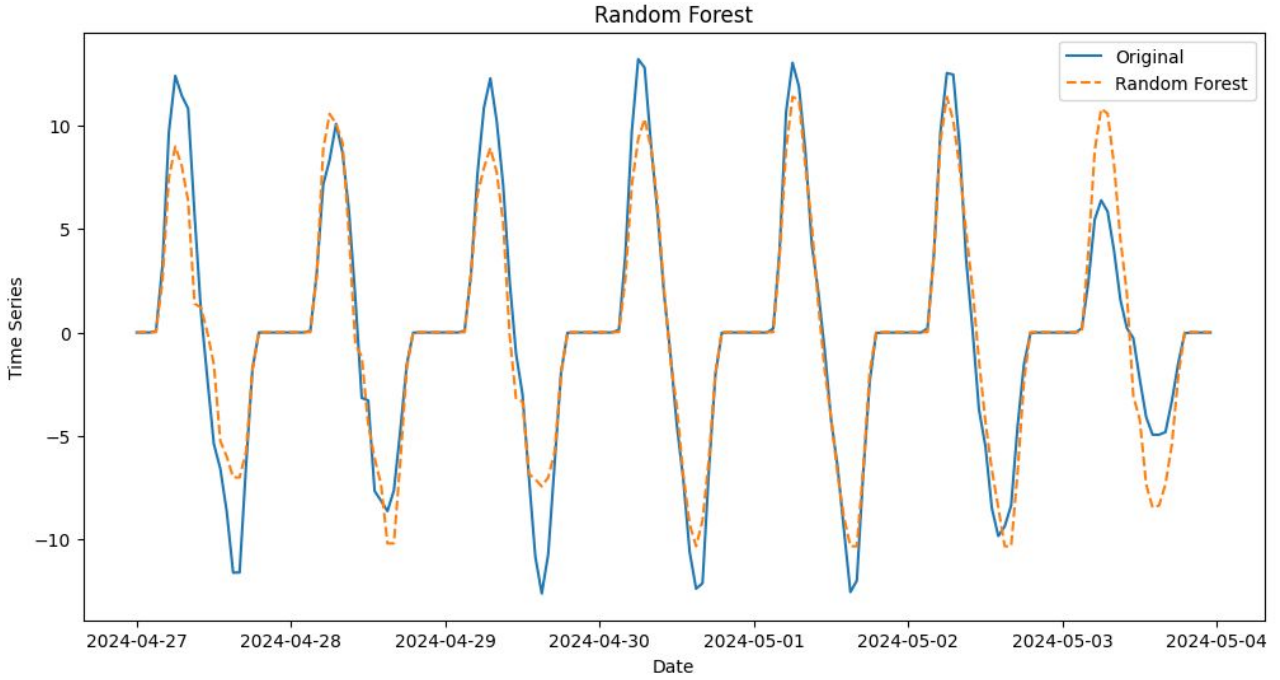


Univariate Models - AUTOCORRELATION

Feature engineering by adding lags

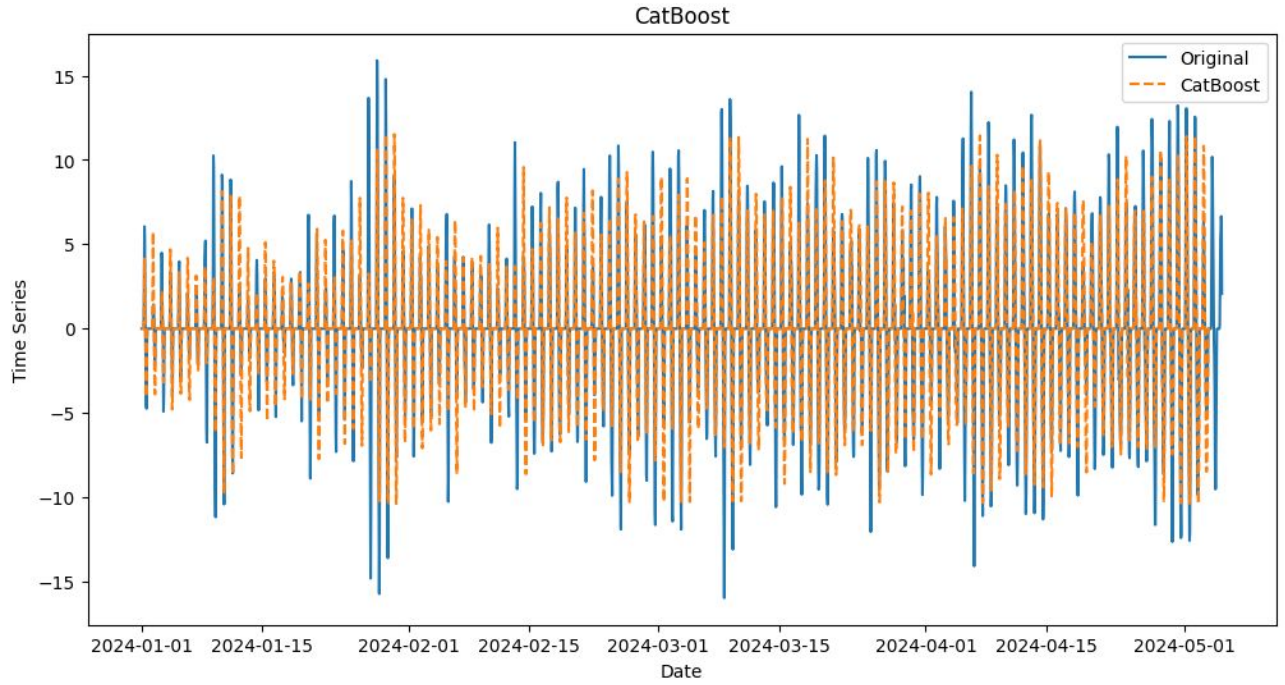
Top 5 lags with highest combined autocorrelation values:
Lag 24: 0.9344
Lag 48: 0.9138
Lag 72: 0.9025
Lag 96: 0.8965
Lag 120: 0.8936

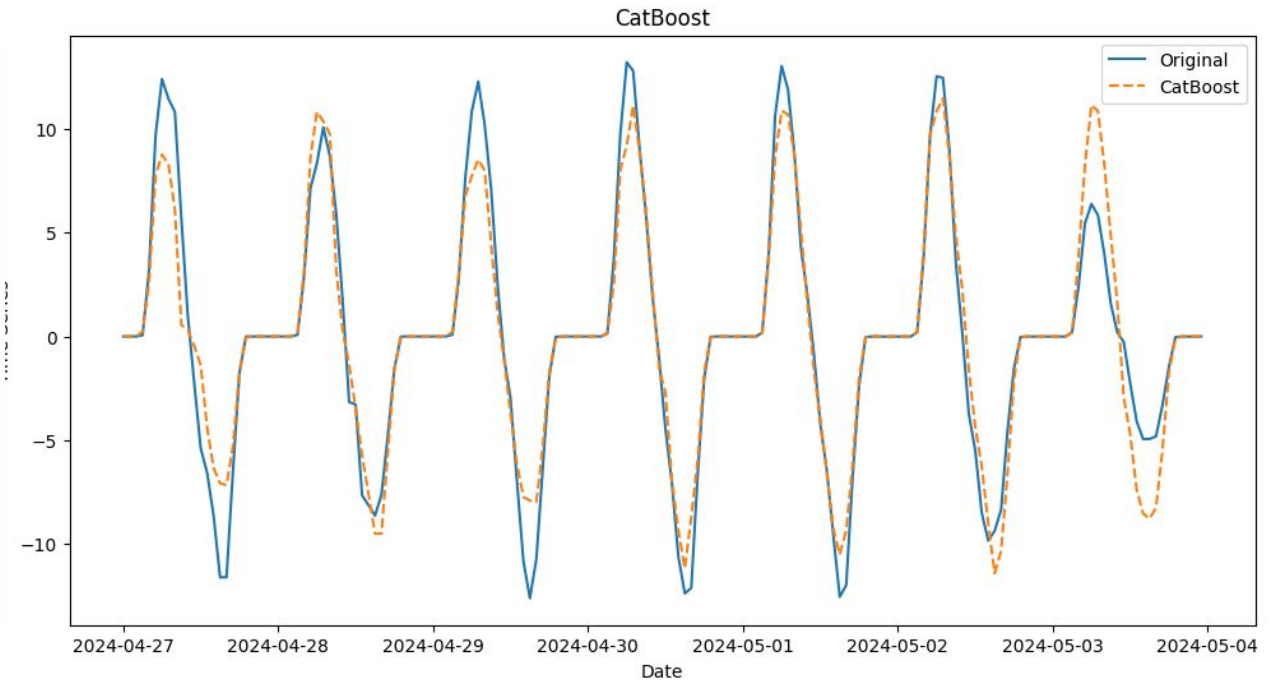




Univariate Models- RANDOM FOREST

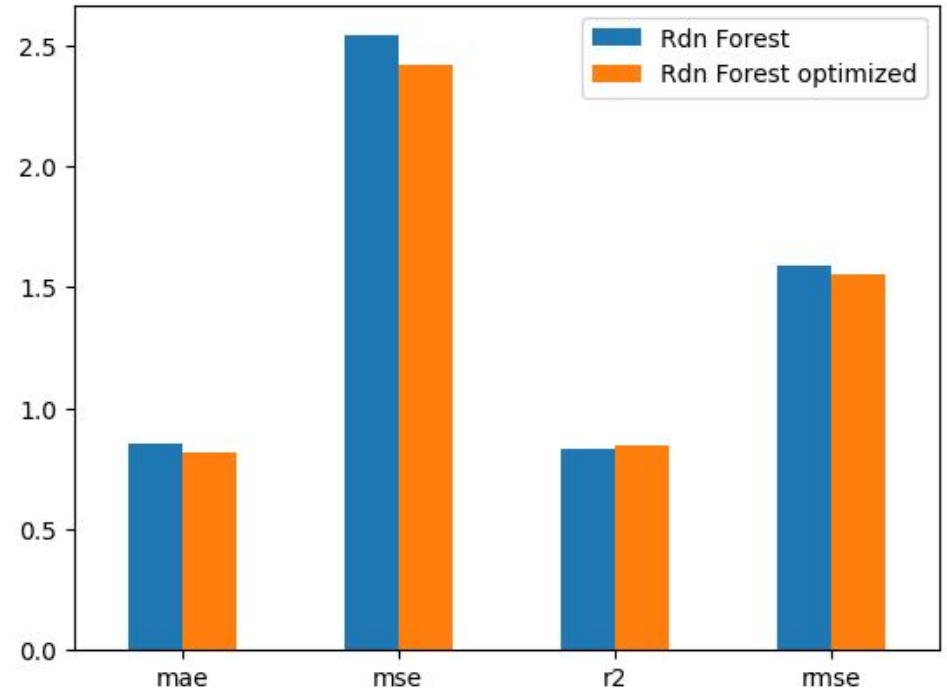
Model: Random Forest
Mean absolute error: 0.8548562992575485
Mean absolute percentage error: 26444413823836.824
Mean squared error: 2.5470251741228482
r2_score: 0.8344775735480734
Root mean squared error: 1.5959402163373315

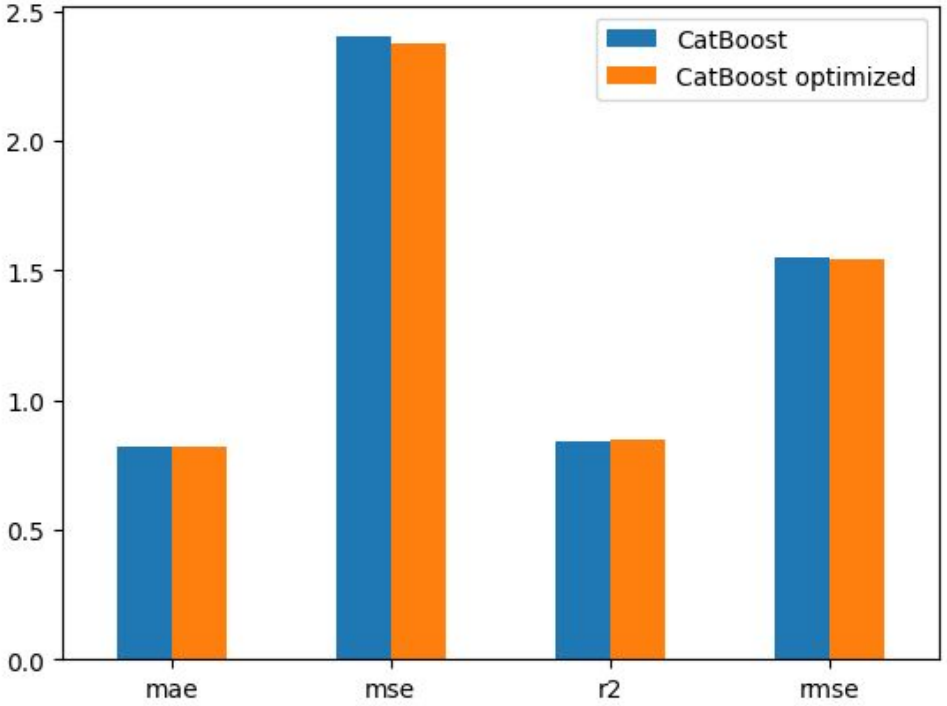




Univariate Models - CATBOOST

Model: CatBoost
Mean absolute error: 0.8233339211975625
Mean absolute percentage error: 6790307298664.622
Mean squared error: 2.404778706670556
r2_score: 0.8437216833771123
Root mean squared error: 1.5507348924527868





Univariate Models - optimization

Hyperparameter tuning and results

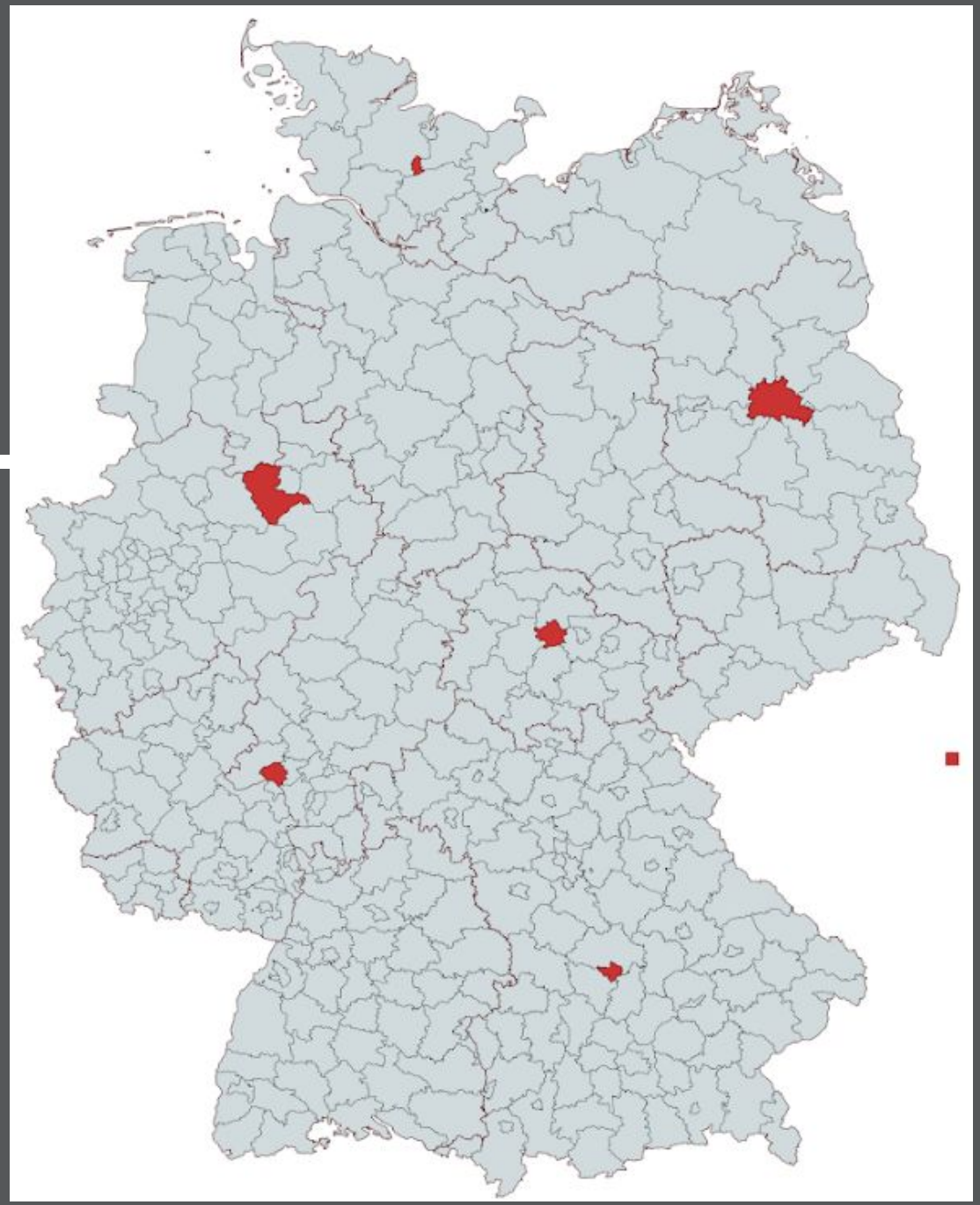
```
# Define parameter grid
param_grid = {
    'iterations': [5, 10, 20],
    'learning_rate': [0.25, 0.5, 0.75],
    'depth': [5, 10, 16]}

# Example scoring function
def my_scoring_function(true, pred):
    return -mean_squared_error(true, pred)

# Perform random search optimization
best_params, best_score = random_search_optimization(
    CatBoostRegressor,
    param_grid,
    X_train= X_train,
    y_train= y_train,
    X_val = X_test,
    y_val= y_test,
    n_iter=20,
    scoring_function=my_scoring_function
)
```

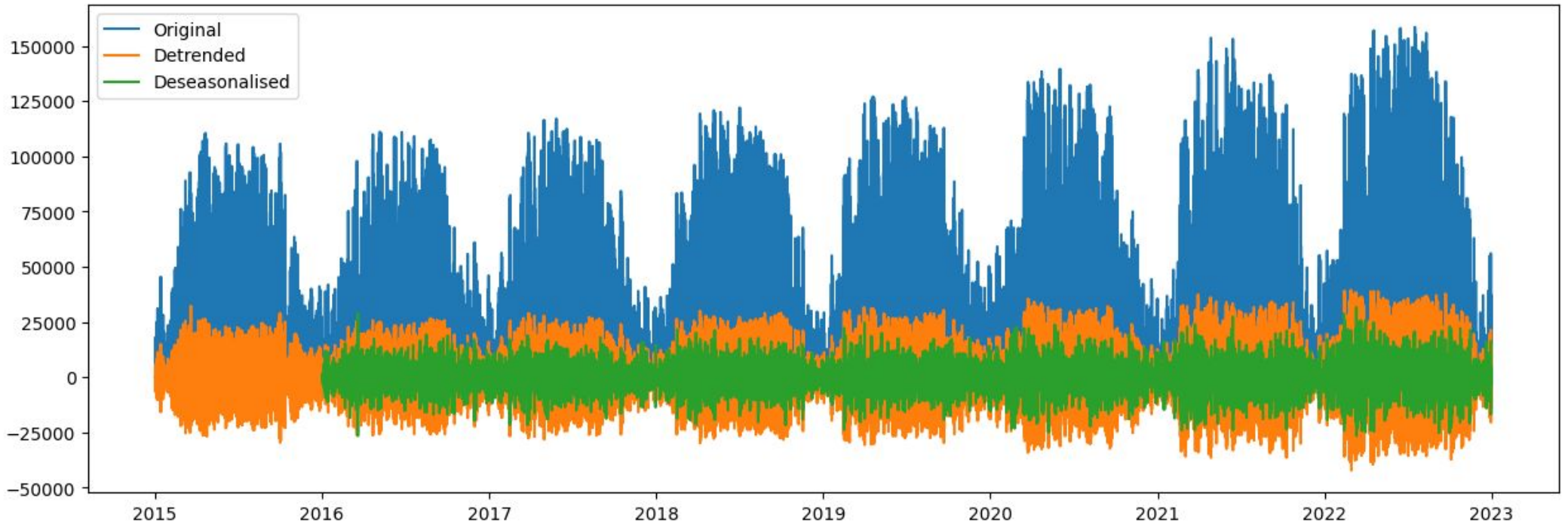
Multivariate Models - Data

OpenMeteo API:

- Six Locations:
  - Templin, Kastellaun, Gütersloh, Ingolstadt, Erfurt, Neumünster

- Six Features:
  - Power, Temperature, Cloud Cover, Shortwave Radiation, Diffuse Radiation, Direct Normal Irradiance





Multivariate Models - Preprocessing

- Remove Trend (Diff)
- Remove Seasonality (Year)
- Z-score Normalisation

Multivariate Models - Features

Feature Engineering

- Lagged Features (up to 1 year)
- Rolling Features (mean and std, up to 1 year)
- Datetime Features (hour cos, hour sin, month cos, month sin, …)

Feature Selection/ Dimensionality Reduction:

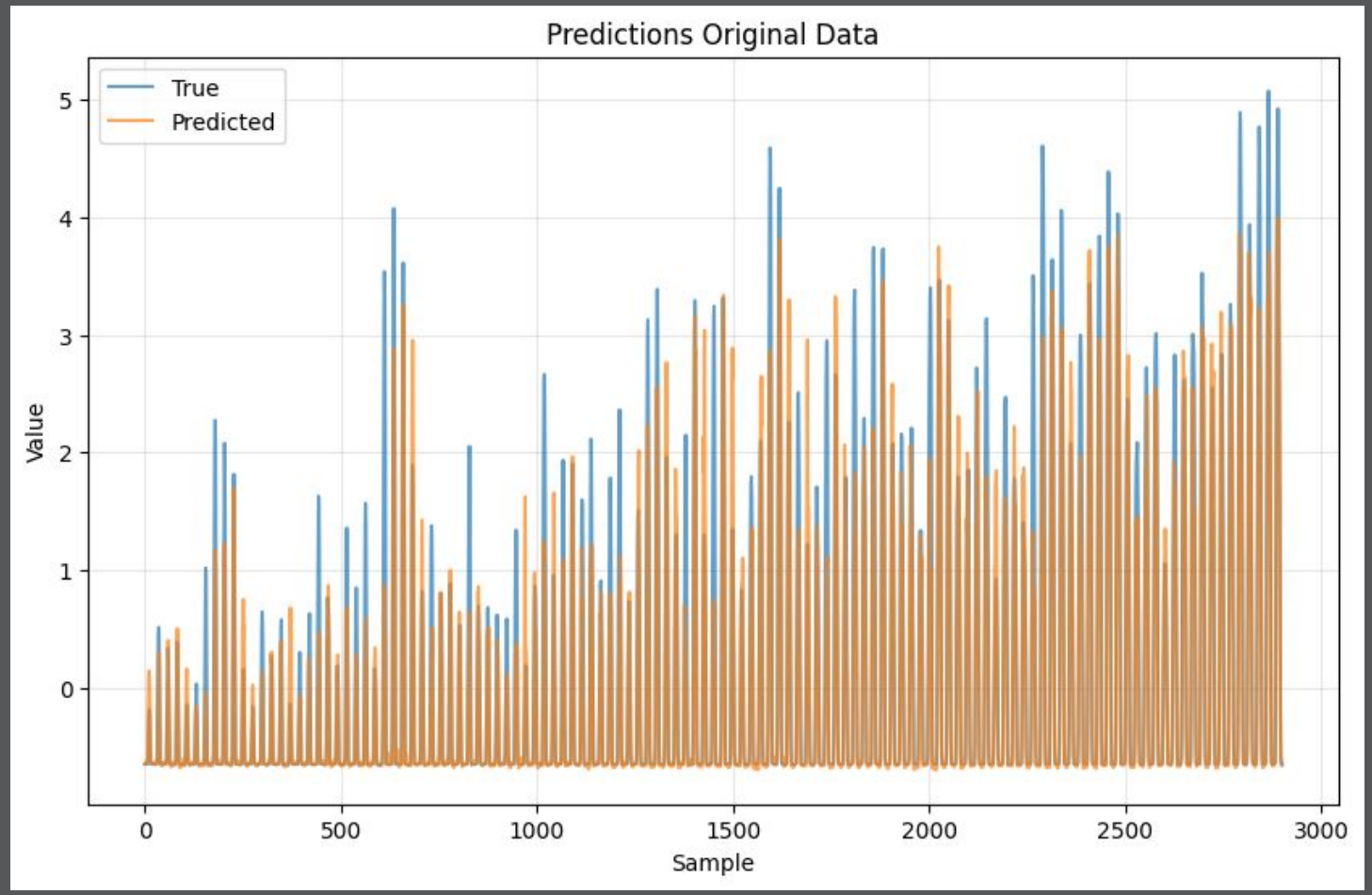
- Random Forest → 50 Features
- PCA → 616 Features to explain 95% of the Variance

GP JOULE
TRUST YOUR ENERGY.

Multivariate Models - LSTM

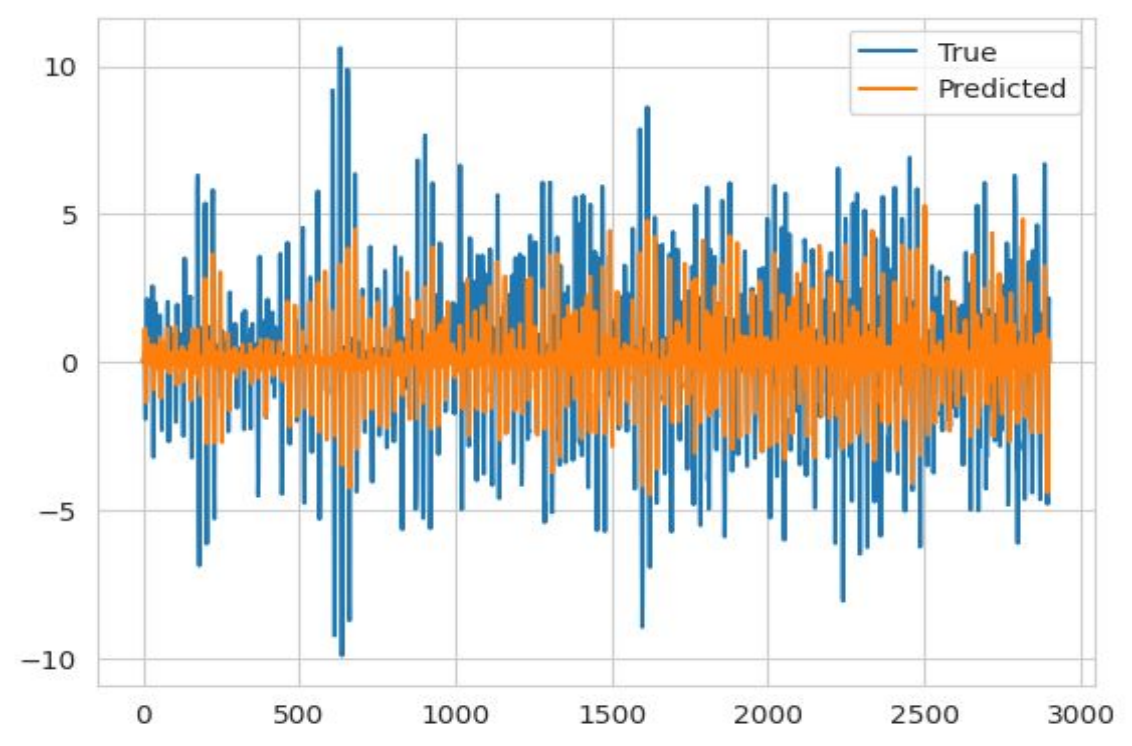
Model Specifications:

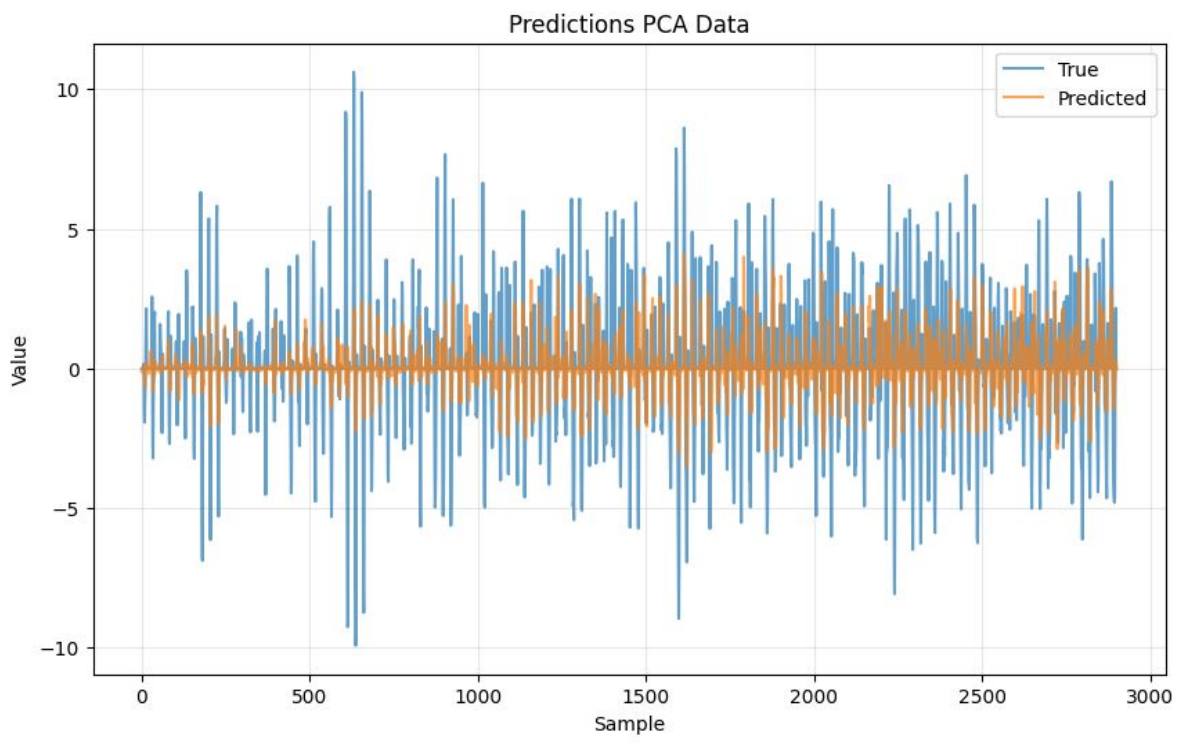
- Loss: MSE
- Optimiser: Adam
- Learning Rate: .001
- Dropout: .1
- Batch Size: 8
- Epochs: 10



GP JOULE
TRUST YOUR ENERGY.

Predictions



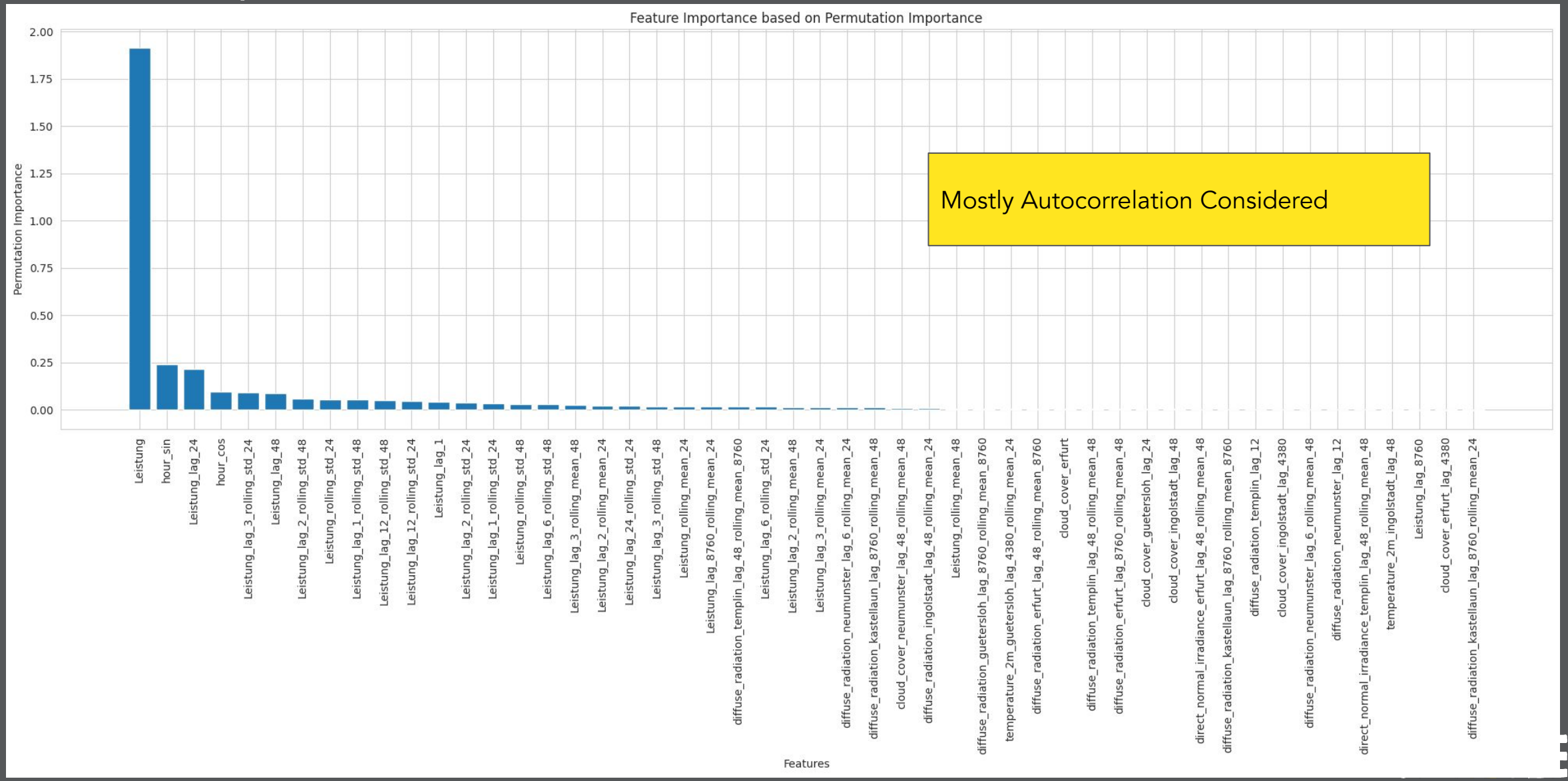


	MAE	MSE	RMSE	R^2
LSTM (Orig)	.2031	.1739	.0417	.8826
LSTM (FE)	1.0324	2.5386	1.5371	.4499
LSTM (PCA)	1.0554	3.0770	1.7541	.2834

Potential Reasons:
- One Year less of data
- PCA is linear
- Too many irrelevant features

Potential fixes:
- Fit PCA on Original Data without Feature Engineering
- Kernel PCA

Feature Importance



Models - Performance Metrics

Type	Model	MAE	MSE	RMSE	R^2
Univariate	Moving Av.	1.4802	5.9520	2.4397	.6150
Univariate	Rdn. Forest	.81939	2.4157	1.5542	.8430
Univariate	CatBoost	.8177	2.3737	1.5407	.8457
Multivariate	LSTM (Orig)	.2031	.1739	.0417	.8826
Multivariate	LSTM (FE)	1.0324	2.5386	1.5933	.4499
Multivariate	LSTM (PCA)	1.0554	3.0770	1.7541	.2834

Challenges

- Running out of GPU
- Gaussian Processes: out-of date python libraries
- High mean absolute percentage errors due to zero values

  → Use MASE instead

- Coordination of tasks, avoiding redundancy
- Using the same preprocessing steps for comparability and reproducibility

Future Outlook

- Further optimization of chosen models
- Align preprocessing, metrics and test period
- Fit Transformer on different datasubsets
- Comprehensive Overview of Preprocessing steps and Models used

Thank You!
Any Questions?

GP JOULE
TRUST YOUR ENERGY.