04.06.2025

Justus Purat & Alexander Kammeyer
Software Project Distributed Systems

Consumption Data Forecast for HPC Systems

Sprint 2
Data Handling & Machine Learning Model

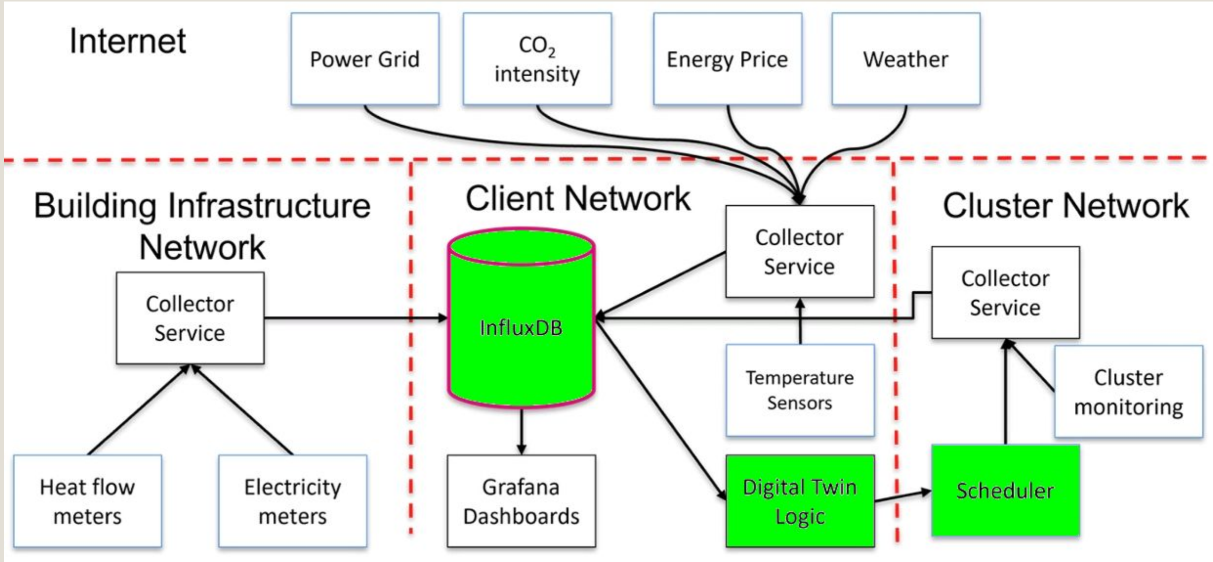
M. Ch
M. Karn
A. Er
Y. Kaya
A. Huth

Institute for Computer Science, FU Berlin

Forecasting

Problem : Limited budget of money, energy, and CO_2 emission

Task : Forecast energy prices and CO_2 footprint for effective scheduling



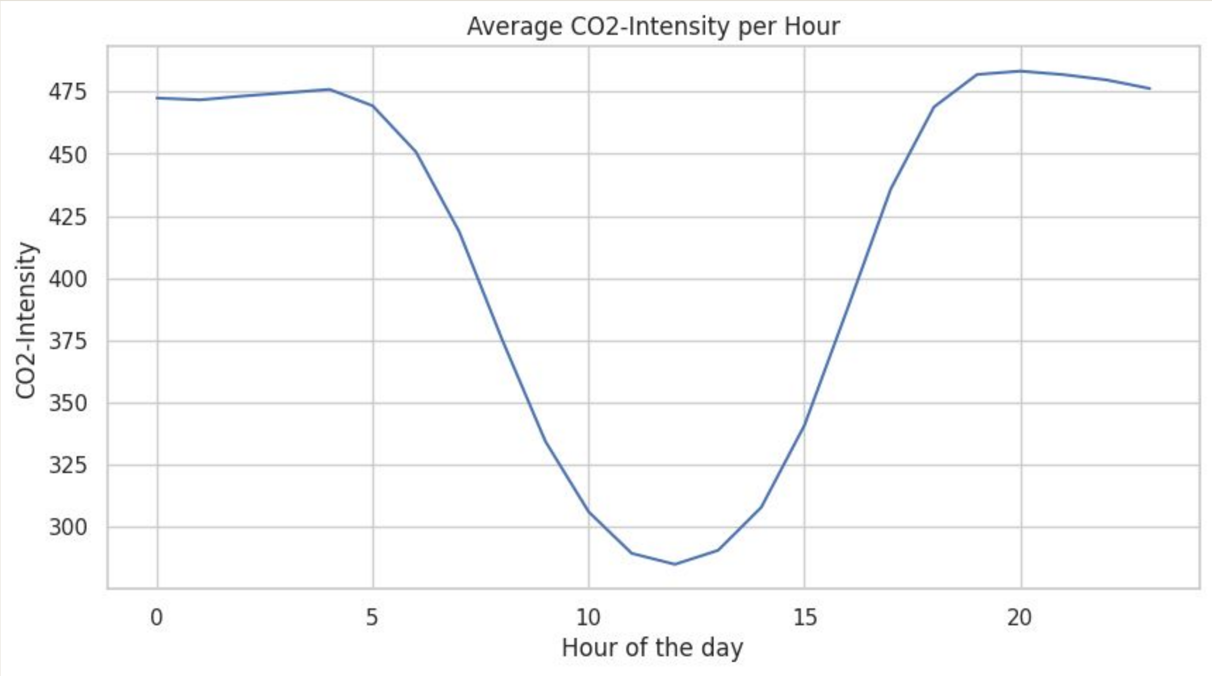
What has been done

- Data Handling & Processing
  - Replacing Null Values
  - Feature Engineering
- Identifying Correlation
- Selection Machine Learning Model
  - Random Forest
  - Prophet
  - Lag-Llama

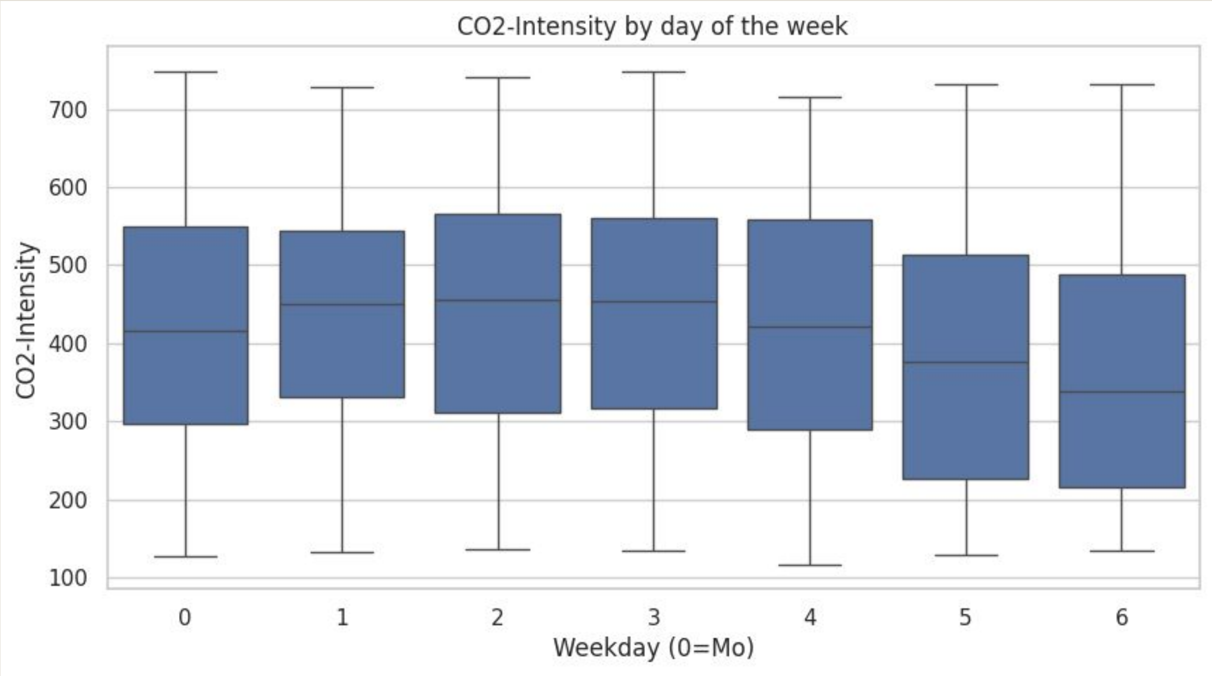
Correlation

- Goal: forecasting the carbon intensity and price
- Timing is important
- Looking for patterns over the year
- Find out how the price correlates with the carbon intensity

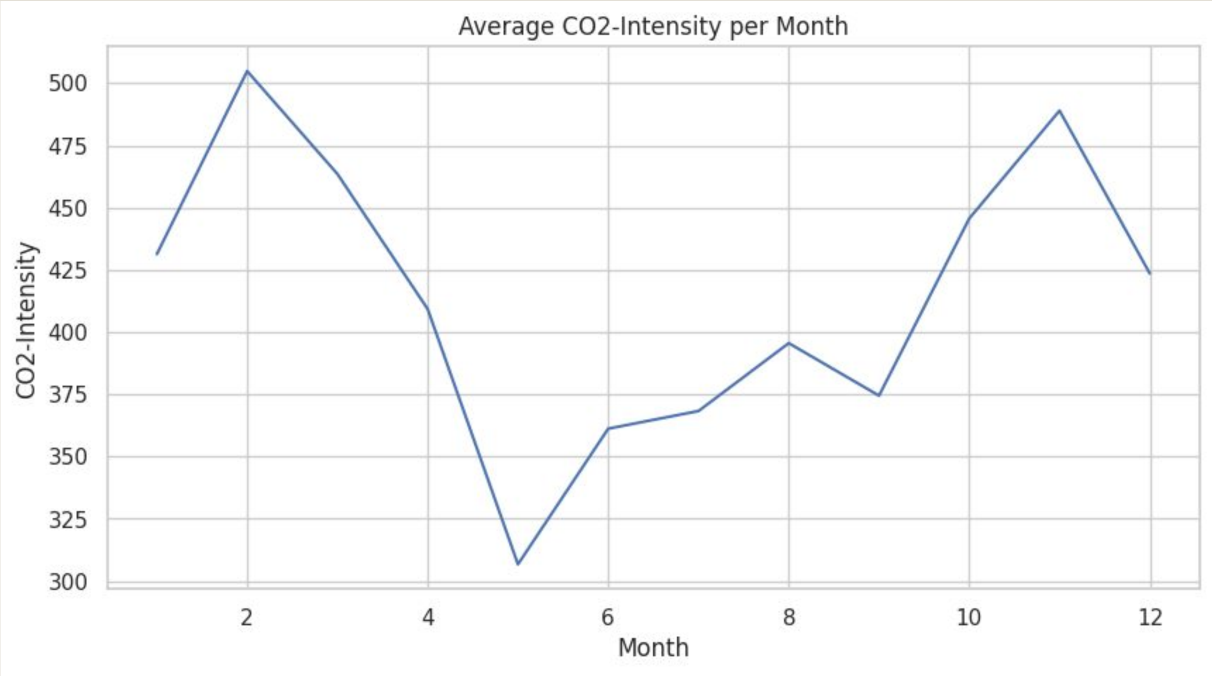
Correlation



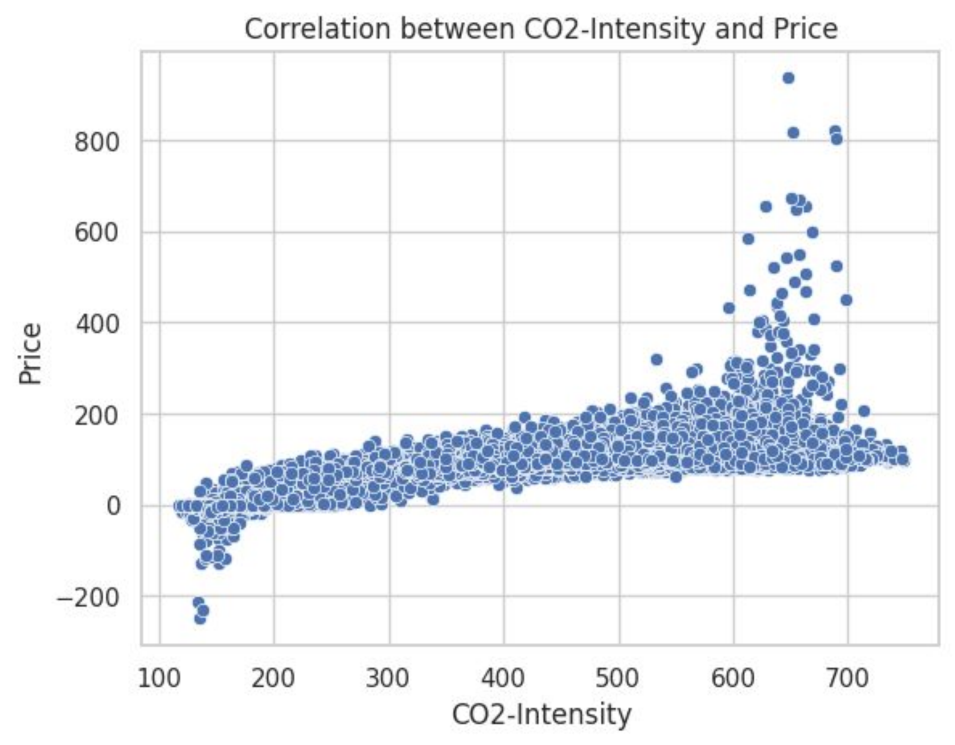
Correlation



Correlation

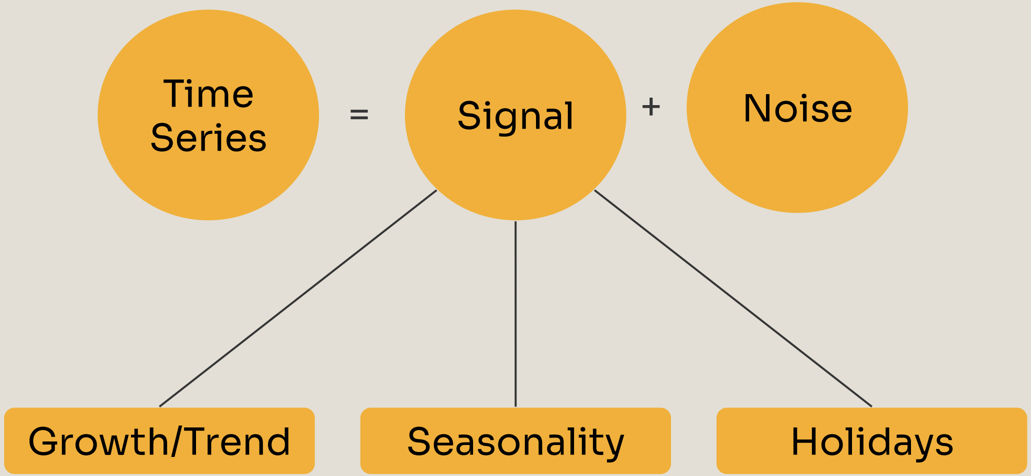


Correlation



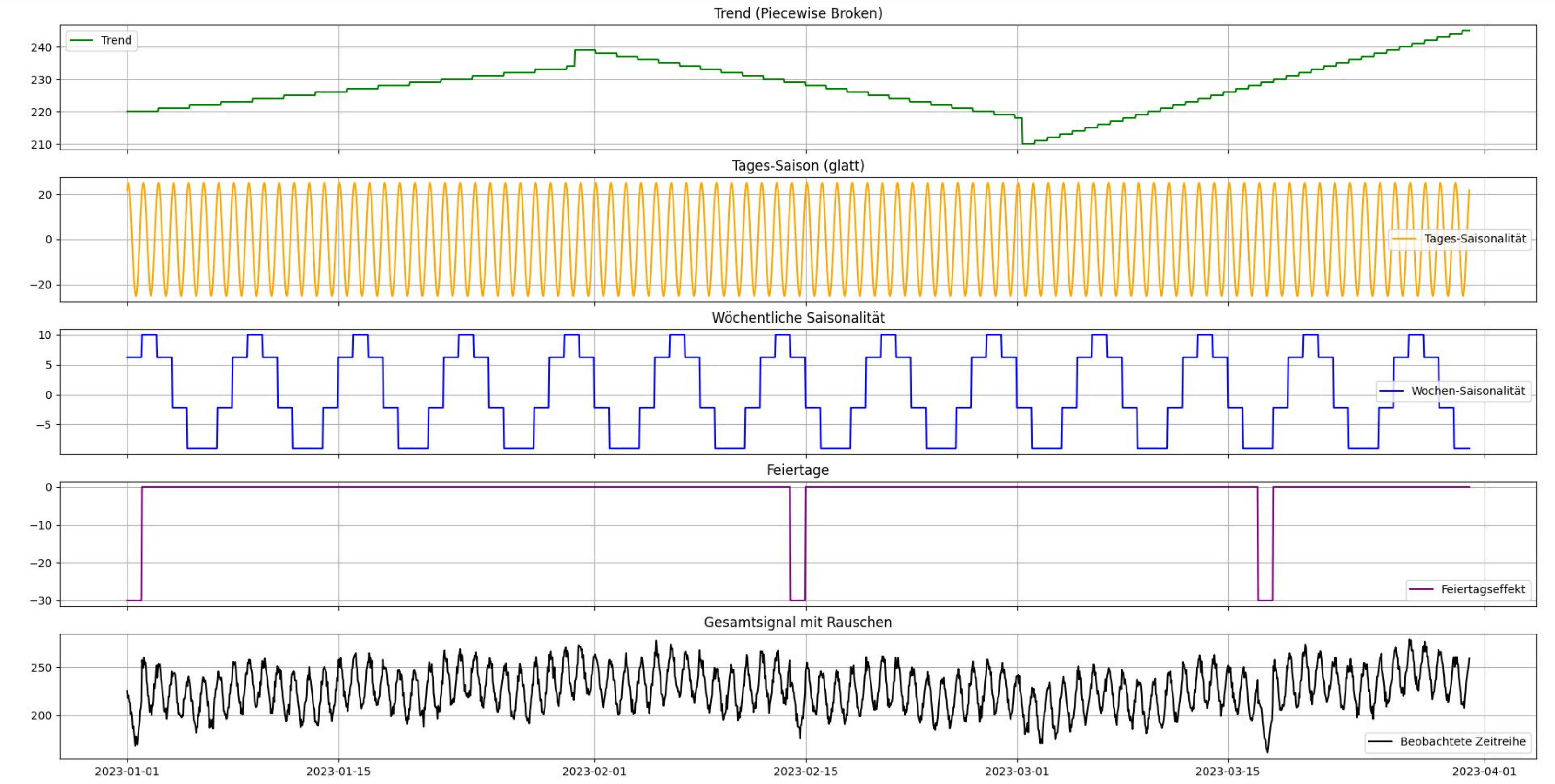
Prophet

- ➢ time series forecasting model
- ➢ decomposes time series into 4 pieces
- ➢ designed to handle data with strong seasonality and trends
- ➢ particularly useful if you have missing data or outliers

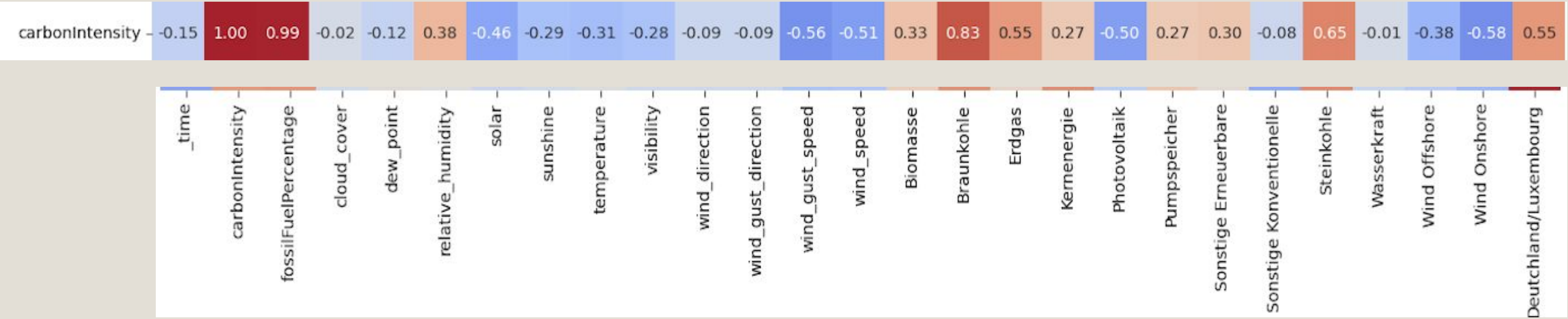


$$y(t) = g(t) + s(t) + h(t) + \epsilon_t.$$

Ideal Input for Prophet



Correlations w.r.t. Carbon Intensity



Random Forest Regressor

- Only interested in correlations with the weather fields
- Future weather data is available so forecasting other fields using that data is logical
- Solar, Relative Humidity, Wind Gust Speed, Wind Speed, Pressure Msl, Sunshine, and Temperature

Feature Engineering

Time based features:

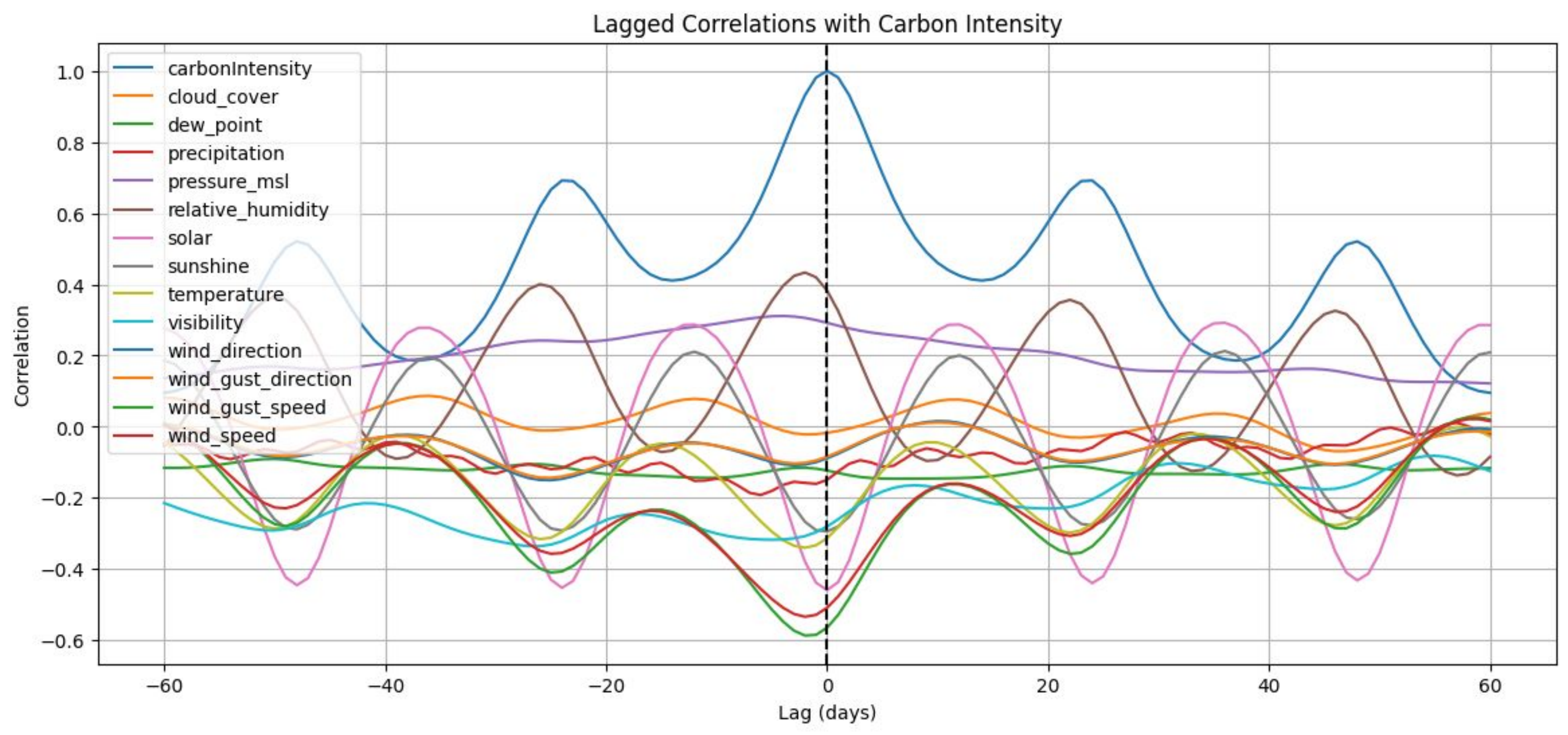
- Hour
- Month
- Day of Week
- Day of Year
- Year

Lagged features:

- how much do the past values of these features explain the current value of target

Exponential weighted:

- a statistical technique that assigns weights to observations in a time series

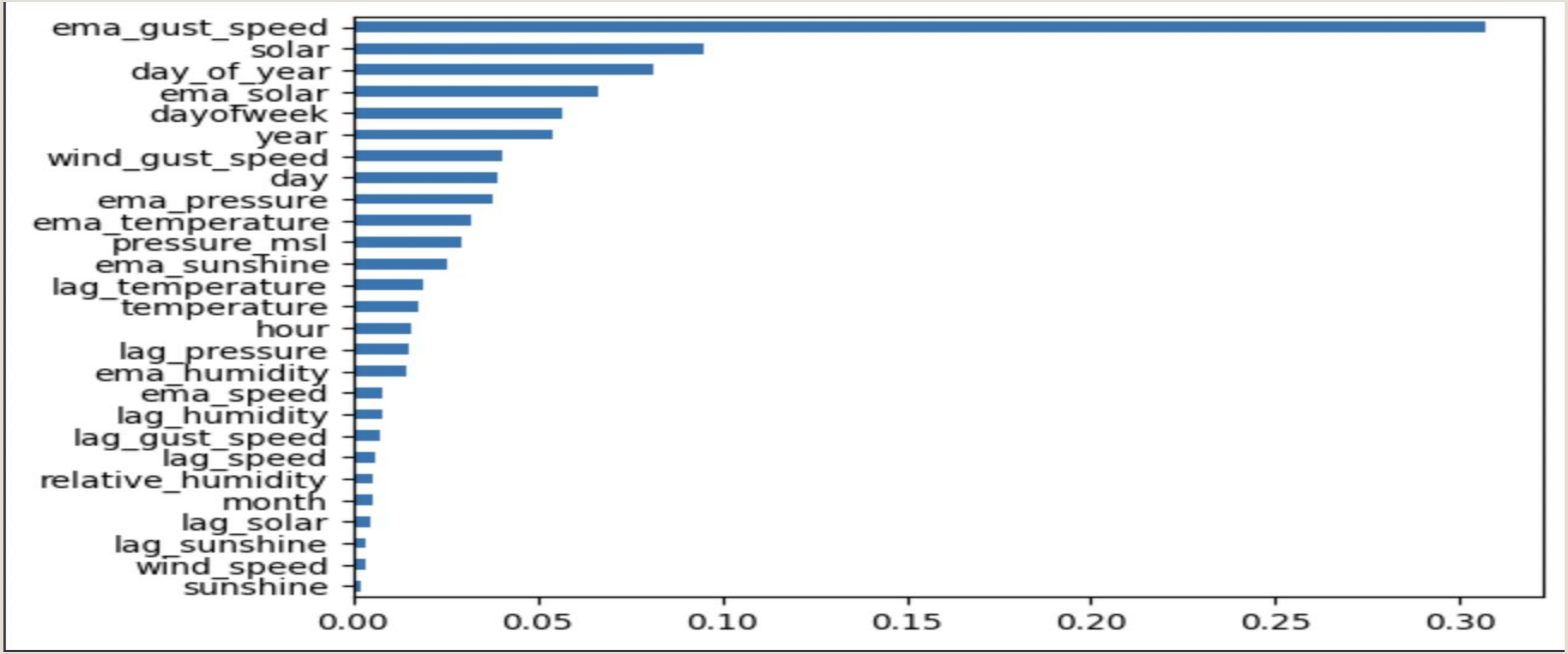


Lagged Correlations with Carbon Intensity
carbonIntensity
cloud_cover
dew_point
precipitation
pressure_msl
relative_humidity
solar
sunshine
temperature
visibility
wind_direction
wind_gust_direction
wind_gust_speed
wind_speed
Correlation
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−60
−40
−20
0
20
40
60
Lag (days)

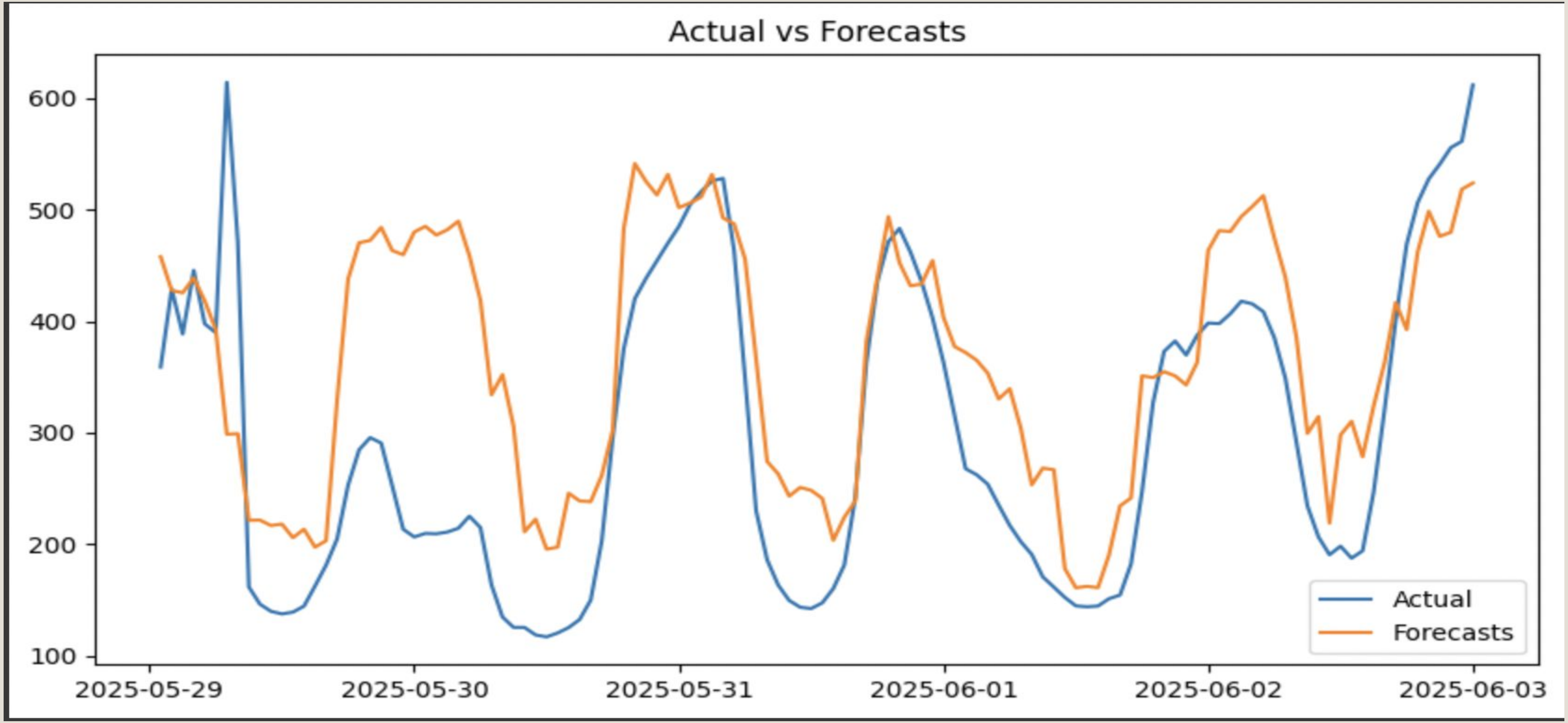
Training

- Since weather data is only available after hourly intervals, we tried to forecast the mean of Carbon Intensity values per hour in the future
- 3 years data of weather was used for training
- 2022-06-03 23:00:00+00:00 to 2025-05-29 00:00:00+00:00

Feature Importance



Forecasts



Actual vs Forecasts
600
500
400
300
200
100
2025-05-29
2025-05-30
2025-05-31
2025-06-01
2025-06-02
2025-06-03
Actual
Forecasts

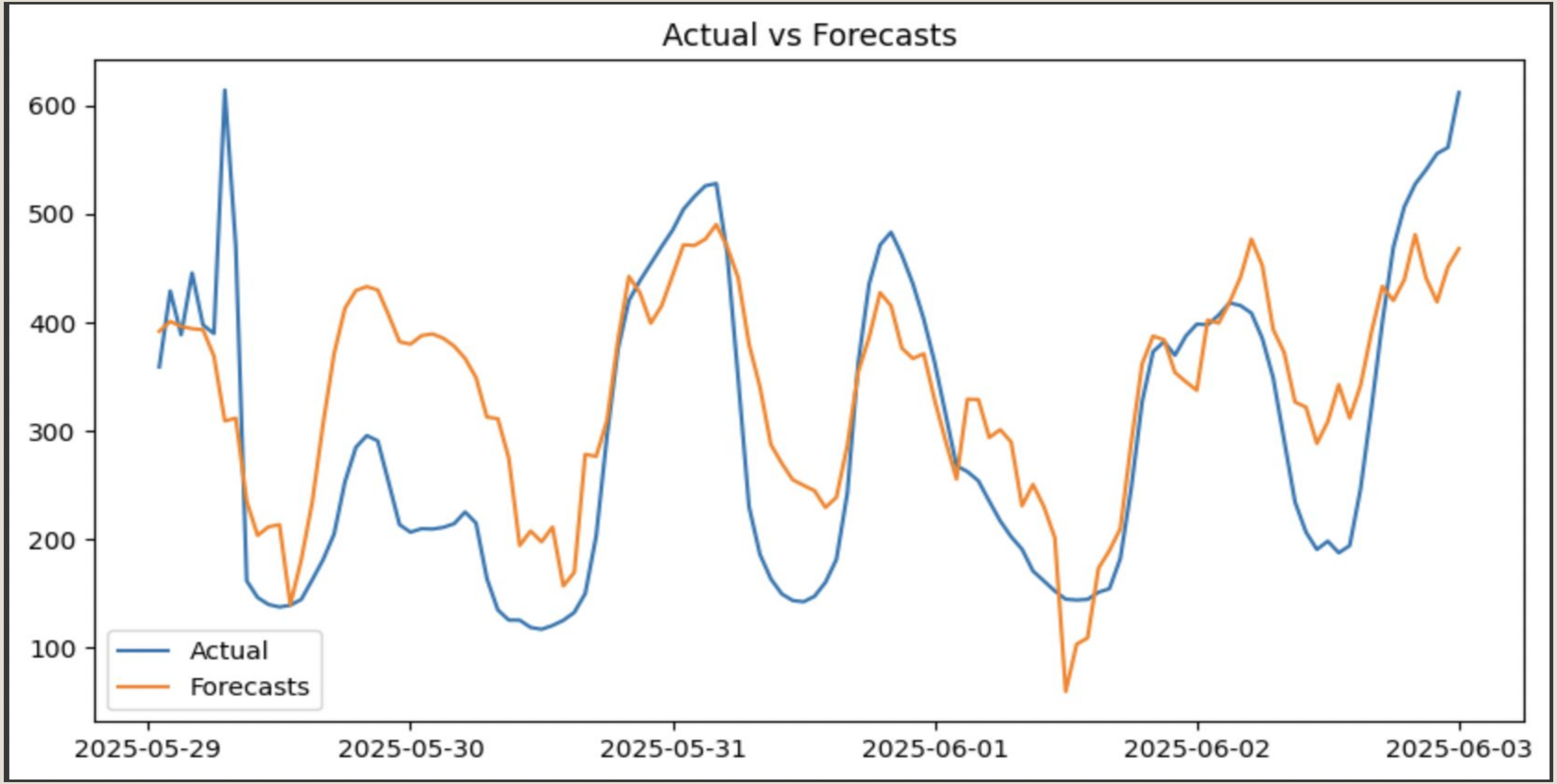
Prophet Time Series

```
[26] from prophet import Prophet

     X_prop = combined_df[['ema_gust_speed', 'solar', 'day_of_year', 'ema_solar', 'dayofweek', 'year',
                          'wind_gust_speed', 'day', 'ema_pressure', 'ema_temperature', 'pressure_msl']].copy()

     X_prop = scaler.transform(X_prop)
     ds = combined_df['_time']

     df = pd.DataFrame({
         'ds': ds.dt.tz_localize(None).values,
         'y': y.values,
         'ema_gust_speed': X_prop[:, 0],
         'solar': X_prop[:, 1],  # correlated variable
         'day_of_year': X_prop[:, 2], # correlated variable
         'ema_solar': X_prop[:, 3],
         'dayofweek': X_prop[:, 4],
         'year': X_prop[:, 5],
         'wind_gust_speed': X_prop[:, 6],
         'day': X_prop[:, 7],
         'ema_pressure': X_prop[:, 8],
         'ema_temperature': X_prop[:, 9],
         'pressure_msl': X_prop[:, 10]
     })
```

Forecasting Features

```
[28] X_fut = future_weather_df[['ema_gust_speed', 'solar', 'day_of_year', 'ema_solar', 'dayofweek', 'year',
                               'wind_gust_speed', 'day', 'ema_pressure', 'ema_temperature', 'pressure_msl']].copy()

     X_fut = scaler.transform(X_fut)

     future = pd.DataFrame({
         'ds': pd.date_range(start=pd.to_datetime('2025-05-29T01:00:00'), periods=120, freq='h'),
         'ema_gust_speed': X_fut[:, 0],
         'solar': X_fut[:, 1],  # correlated variable
         'day_of_year': X_fut[:, 2], # correlated variable
         'ema_solar': X_fut[:, 3],
         'dayofweek': X_fut[:, 4],
         'year': X_fut[:, 5],
         'wind_gust_speed': X_fut[:, 6],
         'day': X_fut[:, 7],
         'ema_pressure': X_fut[:, 8],
         'ema_temperature': X_fut[:, 9],
         'pressure_msl': X_fut[:, 10]
     })

     forecast = model.predict(future)
     forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']]
```

Forecasts



Actual vs Forecasts
600
500
400
300
200
100
Actual
Forecasts
2025-05-29
2025-05-30
2025-05-31
2025-06-01
2025-06-02
2025-06-03

LSTM Architecture

- Used all correlations from all the databases for Carbon Intensity
- Used last 24 hours of days to predict the next hour’s value of Carbon Intensity

```
# Prepare sequences for LSTM training
X_seq, y_seq = create_lstm_training_sequences(X, y, input_steps=24, forecast_horizon=1)

print("X_train shape:", X_seq.shape)  # (samples, 24, features)
print("y_train shape:", y_seq.shape)  # (samples, 1)
```

```
X_train shape: (18871, 24, 15)
y_train shape: (18871, 1)
```

Model

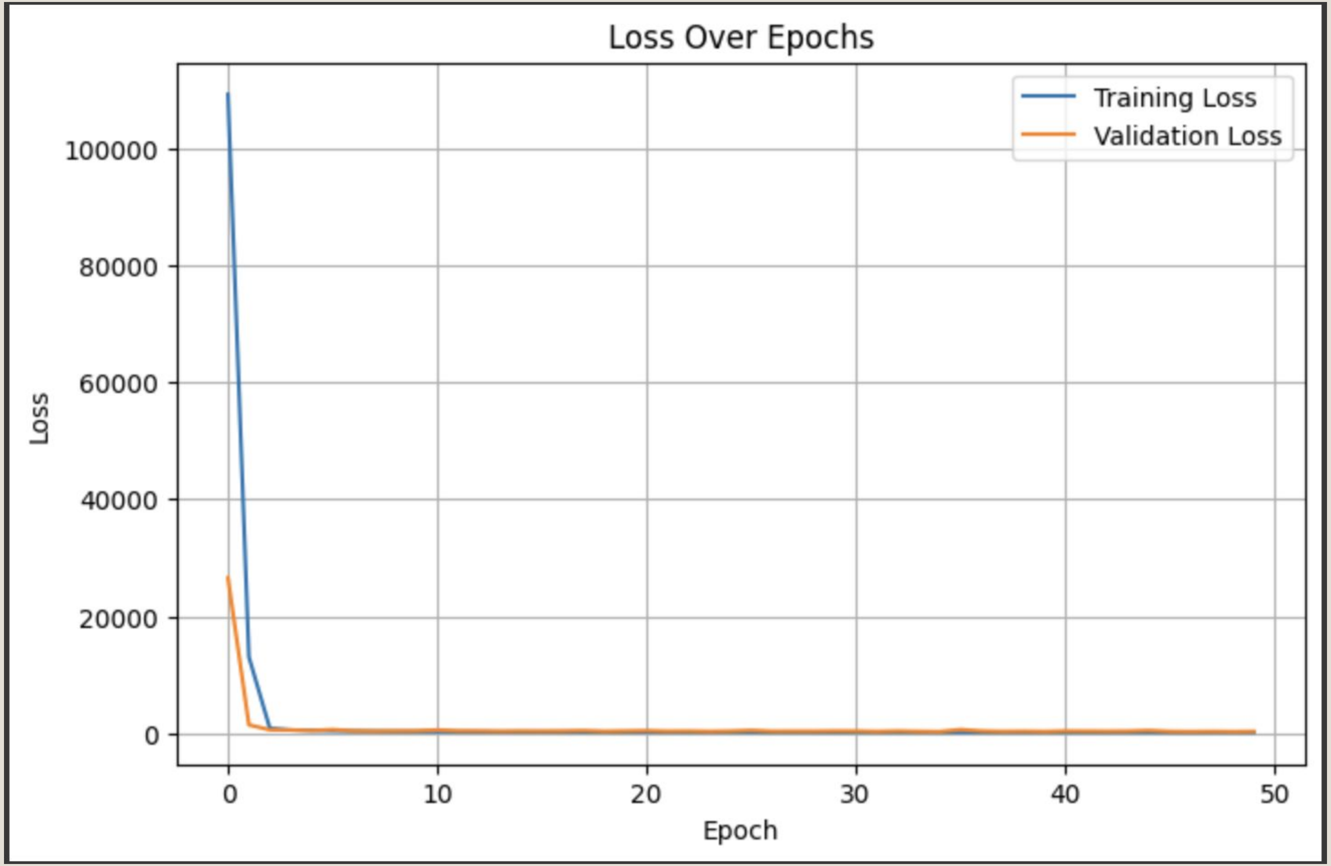
```
model = Sequential()

model.add(Conv1D(32, 3, activation='relu', padding='same', input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Conv1D(64, 3, activation='relu', padding='same'))
model.add(LSTM(32, activation='tanh', recurrent_activation='sigmoid', dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(16, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1))  # Output layer predicts 1 hours ahead

model.compile(optimizer='adam', loss='mse', metrics=['mae'])
model.summary()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pas
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

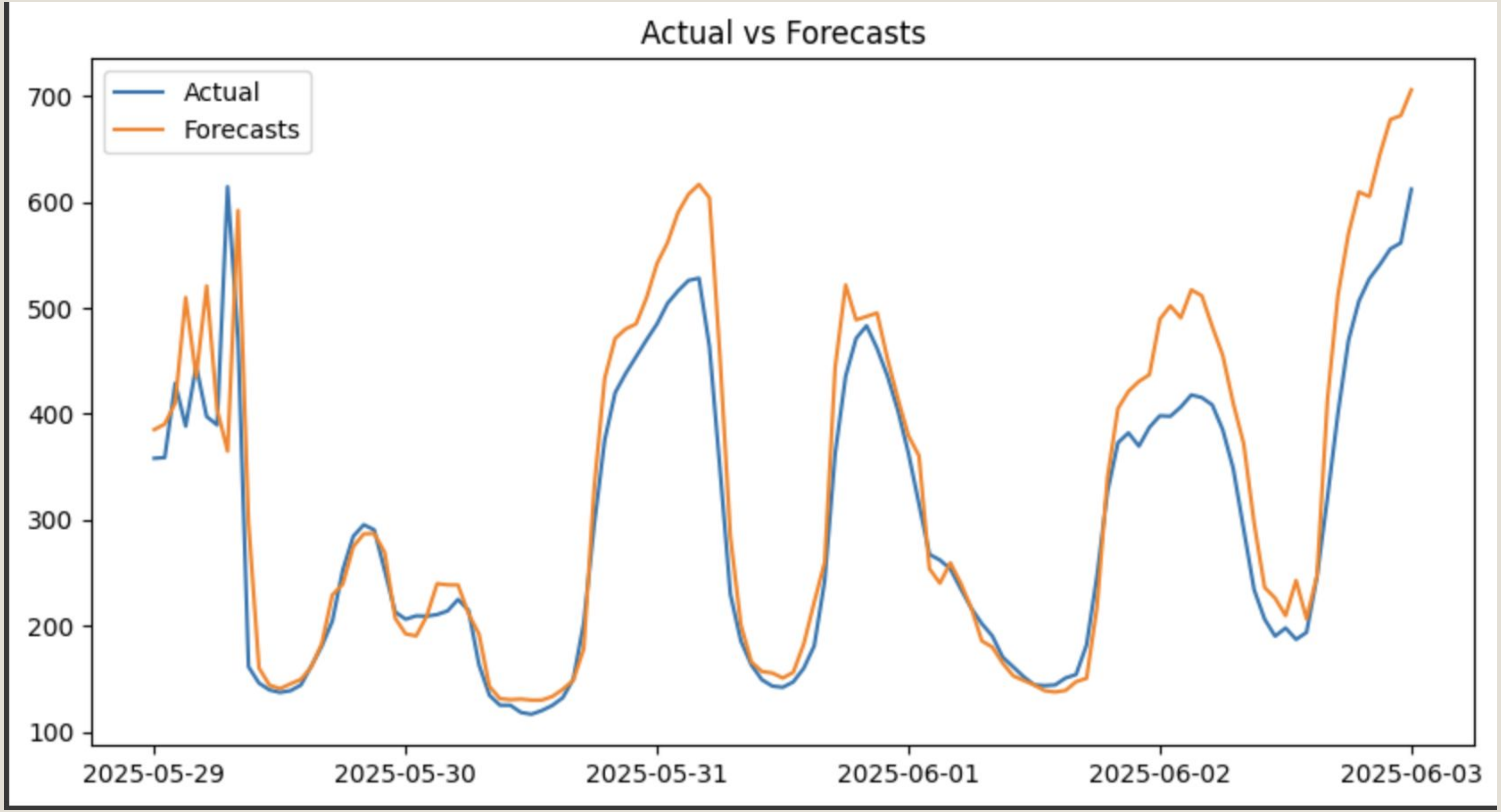
Model: "sequential"

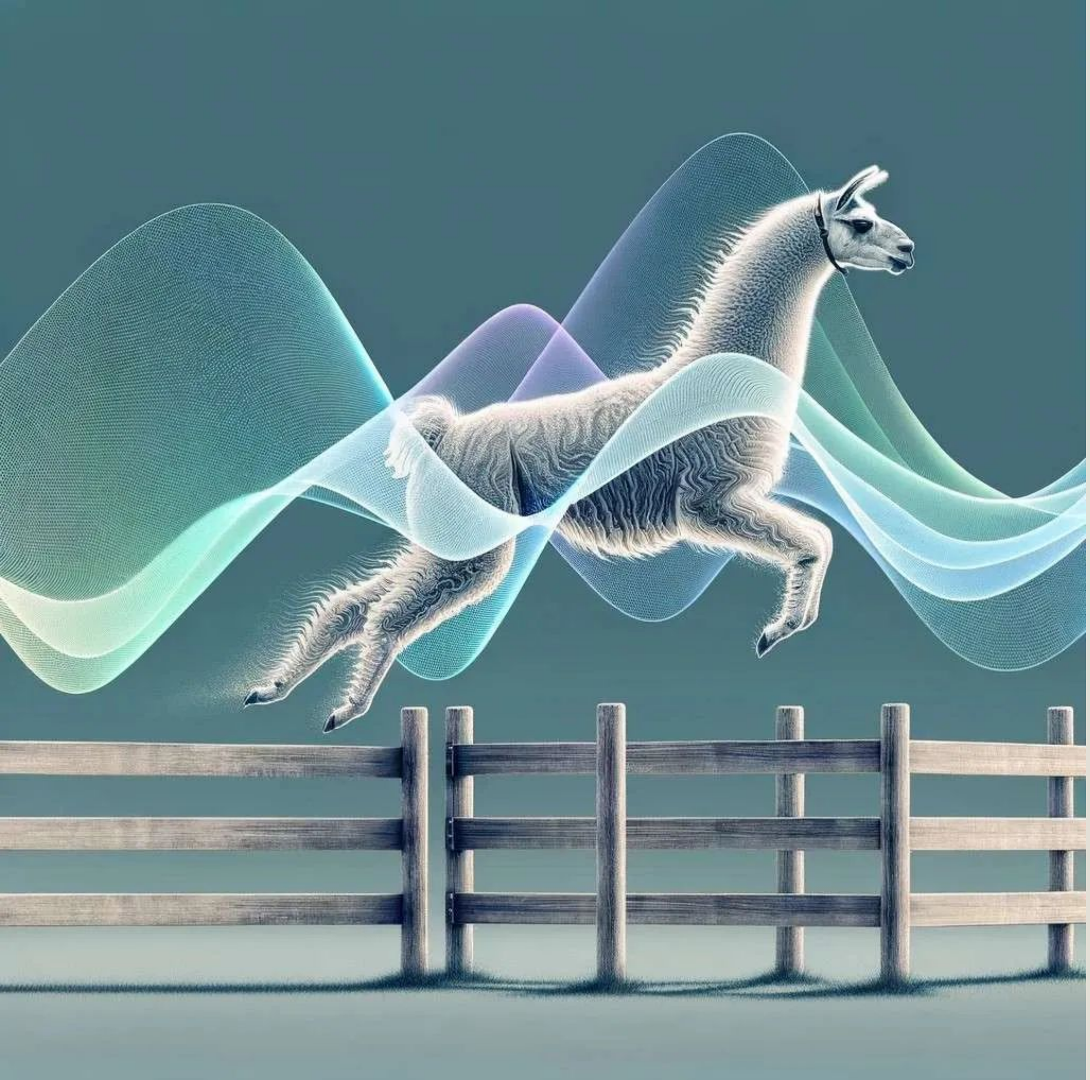
Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 24, 32)	1,472
conv1d_1 (Conv1D)	(None, 24, 64)	6,208
lstm (LSTM)	(None, 32)	12,416
dense (Dense)	(None, 16)	528
dense_1 (Dense)	(None, 8)	136
dense_2 (Dense)	(None, 1)	9

Training History



Forecasts


Actual vs Forecasts
Actual
Forecasts
700
600
500
400
300
200
100
2025-05-29
2025-05-30
2025-05-31
2025-06-01
2025-06-02
2025-06-03



Forecasting with Foundation Models - LagLlama

Lag-Llama

- First open-source foundation model specifically designed for univariate probabilistic time series forecasting.
- Pretrained on a diverse corpus of roughly 8 000 univariate time series spanning six domains (energy, transportation, economics, nature, air quality, cloud operations), totaling about 352 million tokens.
- Without any fine tuning, Lag-Llama demonstrates strong zero-shot forecasting on datasets of arbitrary frequency and prediction length—users can load their data directly and start predicting with minimal setup.
- The code, pretrained weights, and tutorials (zero-shot and finetuning Colab demos) are fully open-source under an Apache-2.0 license, promoting community contributions and reproducibility.

Current Status

Data Pre-processing - Done

Test Train Validate Split - Done

Training Parameter - Identified

Model Training - In progress

Prediction - Not started

Optimization - Not started

[31]

```python
        # Data augmentation
        aug_prob=0.1,

        # Hardware acceleration and training epochs
        trainer_kwargs={
            "accelerator": "gpu" if torch.cuda.is_available() else "cpu",
            "devices": 1,
            "precision": 16 if torch.cuda.is_available() else 32,  # Mixed precision for GPU
            "max_epochs": 20, # Moved 'epochs' here and renamed to 'max_epochs'
            # Early stopping callback
            "callbacks": [
                EarlyStopping(monitor="val_loss", patience=5, mode="min") # <--- Moved 'patience' here
            ]
        }
    )

    print("LagLlama model setup complete.")
    print(f"Context length: {CONTEXT_LENGTH} hours")
    print(f"Prediction length: {PREDICTION_LENGTH} hours")
    print(f"Target series: {len(target_columns)}")

    # Continue with the training cell (CELL 6)
```

```
Setting up LagLlama model...
LagLlama model setup complete.
Context length: 168 hours
Prediction length: 24 hours
Target series: 6
```

Previous Timeline

	Energy Price Group	CO2 Group	Dates
Sprint 1	Data prep & EDA; benchmark ARIMA	Data prep & EDA; energy-mix analysis	30 April – 14 May
Sprint 2	Train ML models (Prophet, Random Forest Regressor); feature engineering	Train ML models; incorporate external regressors	14 May – 28 May
Sprint 3	Develop LSTM; hyperparameter tuning	Time-series cross-validation; tune ML and simple RNN models	28 May – 11 June

Outlook

- Further testing the models

- Data handling
  - Dealing with data gaps
  - Cleaning outliers

- Making decisions about models
  - Continue with Random Forests? Exclude LSTMs?
  - Other models?

Thank You

References

- Forecasting at scale [PeerJ Preprints]
- What is the Prophet Model
- Lag-Llama Arxiv Paper
- Stock Price Forecasting with LagLlama
- Lag-Llama Github