25.06.2025

Justus Purat & Alexander Kammeyer

Software Project Distributed Systems

# Consumption Data Forecast for HPC Systems

Sprint 4

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

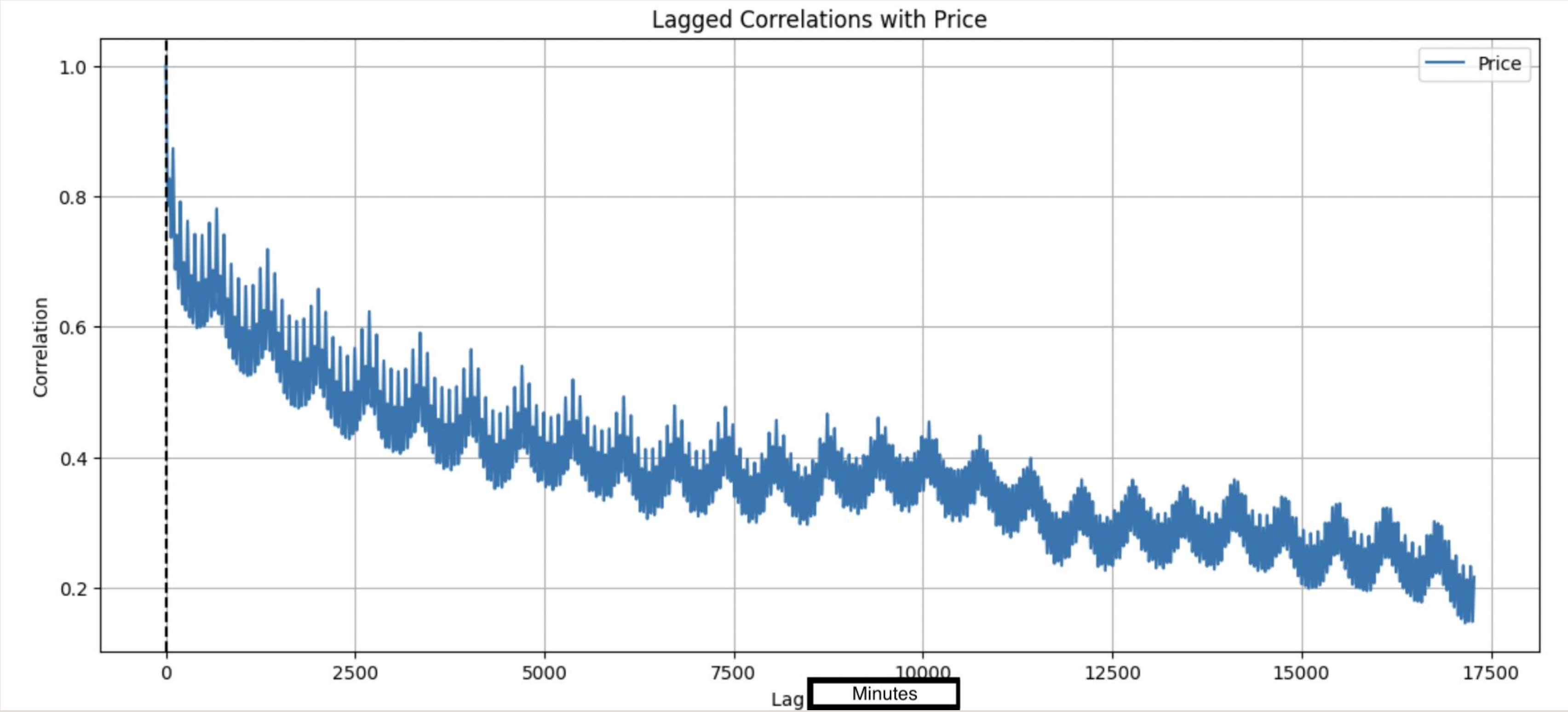
Institute for Computer Science, FU Berlin

# Sprint 4 updates

- Data Preprocessing ✅
  - Feeding the auto-analyzed and correspondingly pre-processed data into the prediction models
- Improve the Models ✅
  - Include Max Pooling Layers in LSTM
  - Predict price values multiple hours into the future
- Making decisions about models ✅
  - Moving forward with LSTM and Lag Llama models
- Start to write the Report ✅

# Univariate LSTM Model

The model relied only  on engineered lagged features and rolling mean features derived from the target variables.



```python
#engineering lagged features based on target

data['lagged_1'] = data['Deutchland/Luxembourg'].shift(1)
data['lagged_2'] = data['Deutchland/Luxembourg'].shift(2)
data['lagged_3'] = data['Deutchland/Luxembourg'].shift(3)
data['lagged_4'] = data['Deutchland/Luxembourg'].shift(4) #1 hour
data['lagged_8'] = data['Deutchland/Luxembourg'].shift(8) #2 hours
data['lagged_12'] = data['Deutchland/Luxembourg'].shift(12) #3 hours
data['lagged_16'] = data['Deutchland/Luxembourg'].shift(16) #4 hours
data['lagged_96'] = data['Deutchland/Luxembourg'].shift(96) # 1 day
data['lagged_192'] = data['Deutchland/Luxembourg'].shift(192) # 2 days
data['lagged_672'] = data['Deutchland/Luxembourg'].shift(672) # 7 days
```

```python
#adding rolling means
data['rolling_mean_1h'] = data['Deutchland/Luxembourg'].shift(1).rolling(4).mean()
data['rolling_mean_2h'] = data['Deutchland/Luxembourg'].shift(1).rolling(8).mean()
data['rolling_mean_3h'] = data['Deutchland/Luxembourg'].shift(1).rolling(12).mean()
data['rolling_mean_6h'] = data['Deutchland/Luxembourg'].shift(1).rolling(24).mean()
data['rolling_mean_12h'] = data['Deutchland/Luxembourg'].shift(1).rolling(48).mean()
data['rolling_mean_24h'] = data['Deutchland/Luxembourg'].shift(1).rolling(96).mean()
data['rolling_mean_48h'] = data['Deutchland/Luxembourg'].shift(1).rolling(192).mean()
data['rolling_mean_168h'] = data['Deutchland/Luxembourg'].shift(1).rolling(672).mean()
```

# Correlation Table

| | olling_mean_168h | rolling_mean_48h | rolling_mean_24h | rolling_mean_12h | rolling_mean_6h | rolling_mean_3h | rolling_mean_2h | rolling_mean_1h | lagged_672 | lagged_192 | lagged_96 | lagged_16 | lagged_12 | lagged_8 | lagged_4 | lagged_3 | lagged_2 | lagged_1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| _time | -0.01 | -0.01 | -0.01 | -0.01 | -0.00 | -0.00 | -0.00 | -0.00 | -0.00 | -0.00 | -0.00 | -0.00 | -0.00 | -0.00 | -0.00 | -0.00 | -0.00 | -0.00 |
| Deutchland/Luxembourg | 0.82 | 0.86 | 0.89 | 0.90 | 0.92 | 0.96 | 0.98 | 0.99 | 0.78 | 0.79 | 0.87 | 0.85 | 0.89 | 0.94 | 0.98 | 0.98 | 0.99 | 0.99 |

# Time Based Features

Cosine and Sine transformations to handle the cyclical nature of time based features

```
#engineering time based features

data['_time'] = pd.to_datetime(data['_time'])
data['hour'] = data['_time'].dt.hour
data['day'] = data['_time'].dt.day
data['month'] = data['_time'].dt.month
data['day_of_week'] = data['_time'].dt.dayofweek
data['day_of_year'] = data['_time'].dt.dayofyear

# Hour of Day (0-23, max_value=24)
data['hour_sin'] = np.sin(2 * np.pi * data['hour'] / 24)
data['hour_cos'] = np.cos(2 * np.pi * data['hour'] / 24)

# Day of Week (0-6, max_value=7)
data['day_of_week_sin'] = np.sin(2 * np.pi * data['day_of_week'] / 7)
data['day_of_week_cos'] = np.cos(2 * np.pi * data['day_of_week'] / 7)

# Month of Year (1-12, max_value=12)
data['month_sin'] = np.sin(2 * np.pi * data['month'] / 12)
data['month_cos'] = np.cos(2 * np.pi * data['month'] / 12)

# Day of Year (1-365.25, max_value=365.25)
data['day_of_year_sin'] = np.sin(2 * np.pi * data['day_of_year'] / 365.25)
data['day_of_year_cos'] = np.cos(2 * np.pi * data['day_of_year'] / 365.25)
```

# Creation of Sequences

```
# Prepare sequences for LSTM training
X_seq, y_targets = create_lstm_training_sequences(X, y, input_steps=192, forecast_horizon=8)

print("X seq shape:", X_seq.shape)  # (samples, 192, features)
print("y targets shape:", y_targets.shape)  # (samples, 8)
```

```
X seq shape: (173153, 192, 26)
y targets shape: (173153, 8)
```

# Data Preprocessing

Training and Testing splits

Applied scalers to both features and target variables to normalize the values

```
Training set size (X_train): (155837, 192, 26)
Testing set size (X_test): (17316, 192, 26)
Training target size (y_train): (155837, 8)
Testing target size (y_test): (17316, 8)
```

# Model Creation and Hyper Parameter Optimization

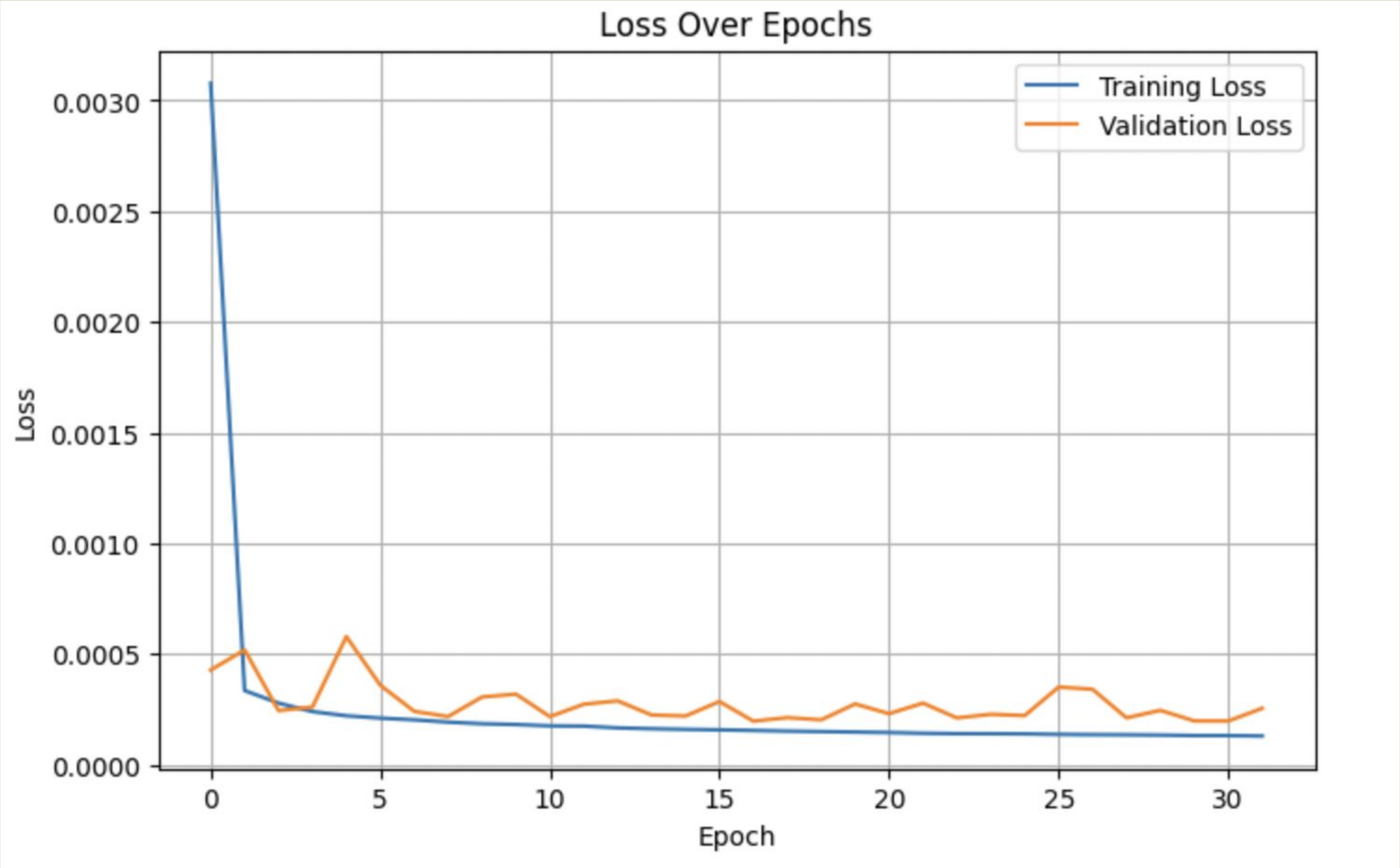
```python
import tensorflow as tf
from tensorflow import keras
from keras.layers import Dense, LSTM, Conv1D, MaxPooling1D, BatchNormalization
from keras.models import Sequential

model = Sequential()

model.add(Conv1D(32, 2, activation='relu', padding='same', input_shape=(X_train_scaled.shape[1], X_train_scaled.shape[2])))
model.add(BatchNormalization())
model.add(MaxPooling1D(pool_size=2, padding='valid'))
model.add(Conv1D(64, 2, activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling1D(pool_size=2, padding='valid'))
model.add(LSTM(64, activation='tanh', recurrent_activation='sigmoid', dropout=0.2, recurrent_dropout=0.2, return_sequences=True))
model.add(LSTM(32, activation='tanh', recurrent_activation='sigmoid', dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(32, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(8))  # Output layer predicts 2 hours ahead

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

# Training Process

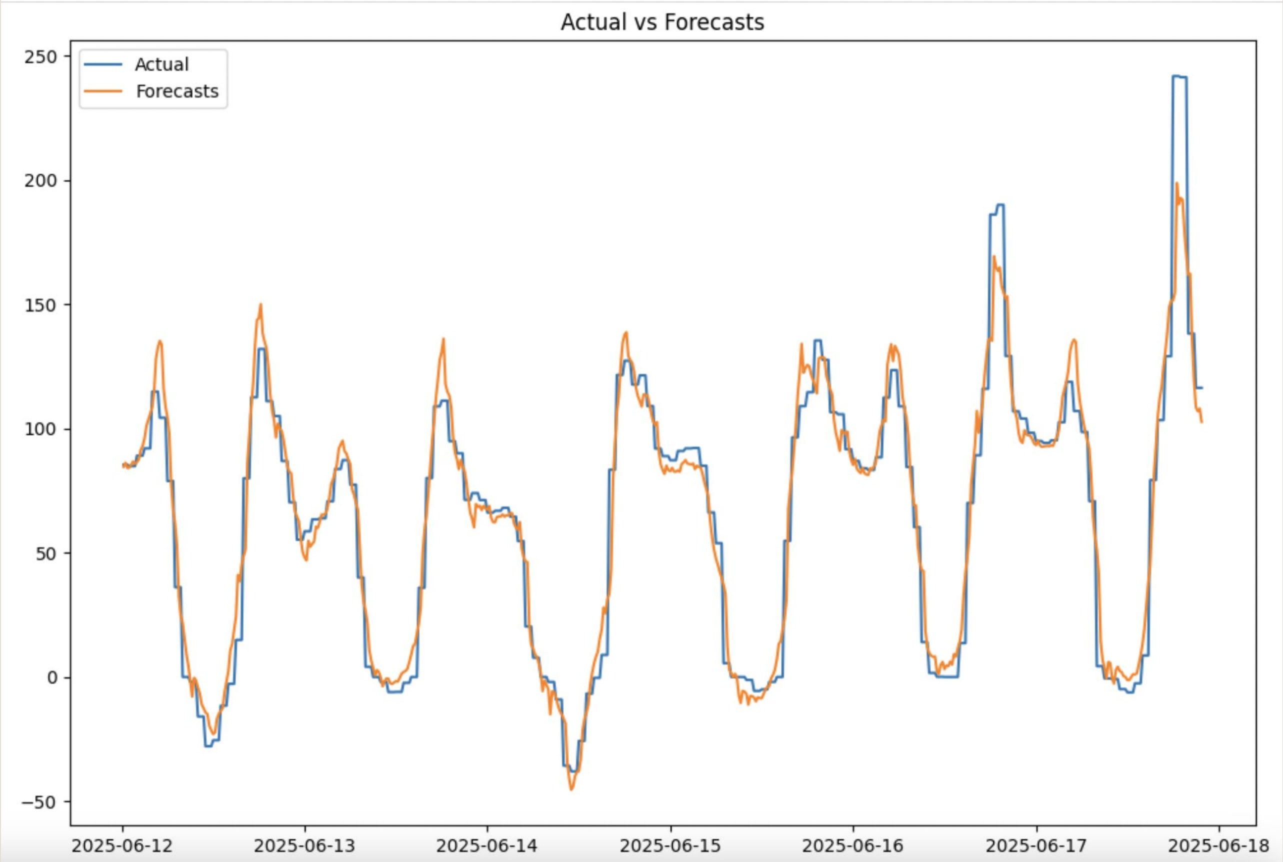
```
from keras.callbacks import EarlyStopping
early_stop = EarlyStopping(
    monitor='val_loss',   # watch validation loss
    patience=15,          # wait 15 epochs before stopping
    restore_best_weights=True  # restore model weights from the best epoch
)
```

```
history = model.fit(
    X_train_scaled, y_train_scaled,
    epochs=100,
    batch_size=64,
    validation_data=(X_test_scaled, y_test_scaled),
    callbacks=[early_stop]
)
```

```
loss: 1.5737e-04 - mae: 0.0084 - val_loss: 1.9682e-04 - val_mae: 0.0081
```


Loss Over Epochs
Training Loss
Validation Loss
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
Loss
0
5
10
15
20
25
30
Epoch

# Forecasts



Actual vs Forecasts
Actual
Forecasts
250
200
150
100
50
0
−50
2025-06-12
2025-06-13
2025-06-14
2025-06-15
2025-06-16
2025-06-17
2025-06-18

```python
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error

mse = mean_squared_error(actual_values['Deutchland/Luxembourg'], f)
mae = mean_absolute_error(actual_values['Deutchland/Luxembourg'], f)
print("Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)
```

```
Mean Squared Error: 201.3466678407244
Mean Absolute Error: 9.437749400675507
```

# Report

Freie Universität Berlin, Department for Mathemetics and Computer Science, Institut for Computer Sience

## Software Project: Distributed Systems S25

## Consumption Data Forecast for HPC Systems

by

Muneeb Ch, Mandeep Karn, Ahmet Er
Michael Zent, Yasin Kaya, Aron Huth

25. Juni 2025

Supervised by
Justus Purat and Alexander Kammeyer

### 1 Motivation

not sure, but my suggestion

- what is machine learning?
- for what it is useful - for what not?

### 2 Problem description

- What is the task of the physikalisch technische Bundesanstalt?
- Konzept of the digital Twin
- Why we need to forcast and where does the date come from?

### 3 Related Work

#### 3.1 Overview of the Papers from A. Kammeyer

#### 3.2 Results form Group from 2024

- Which approach did they had
- What Results did they had

### 4 Progress of Work

### 5 Interface Documentaion

### 6 Presentation of Results

### 7 Evaluation

### 8 Conclusion

# Final Sprint Outlook

- Moving forward with LSTM & try to improve it further

- Prepare for the final presentation

- continue working on the Report

# Thank You