Aalto University
CS-C3240 - Machine Learning
Rafin Jahan

Day-Ahead Electricity Price Forecasting in Finland using ML

Returned: November 28, 2025

1 Introduction

Electricity price forecasting is vital in the Nordic power market, helping producers and consumers plan their operations. In the Nord Pool day-ahead market, 24 hourly prices are cleared daily based on demand, renewable generation, and interconnections. This project applies machine-learning methods to predict Finnish day-ahead prices.

Stage 2 extends the baseline Linear Regression model by adding time-based features and comparing it with a non-linear Random Forest Regressor.

2 Problem Formulation

The data set for this project is obtained from Nord Pool Market Data [1], which provides historical hourly day-ahead prices for the Finnish bidding zone. We used data from 2020–2024, where each data point corresponds to one hour with its clearing price in euros per megawatt hour (€/MWh).

The task is formulated as a supervised regression problem. The target variable is the day-ahead electricity price, while the features are time-based and lagged variables derived directly from the price series. These include **hour of day** (0-23), **day of week** (0–6), and **month/season**, which capture recurring patterns. We also include lagged prices: **lag-1** (previous hour), **lag-24** (the same hour the previous day), and **lag-168** (the same hour last week).

This formulation follows prior research emphasizing seasonality and lagged values. Mohsenian-Rad and Leon-Garcia [2] showed that weighted averages of past prices can provide effective short-term forecasts, while Weron [3] highlighted the strong daily and weekly cycles in electricity prices.

3 Methods

3.1 Dataset and preprocessing

The dataset contains approximately $5 \times 365 \times 24 \approx 43{,}800$ data points for the years 2020–2024. Price spikes above 1000 €/MWh were treated as outliers and were removed. Missing hours are handled by forward-fill interpolation. From the timestamp column we derive categorical features (hour, weekday, month), and construct lagged variables for the previous hour, previous day, and previous week.

3.2 Feature selection

We restrict Stage 1 to simple calendar and lagged features, which capture the main daily, weekly, and seasonal cycles. This minimal feature set allows us to establish a clean baseline before incorporating more complex features (e.g., demand forecasts, wind power) in Stage 2.

For Stage 2, the feature set was extended with two additional types of variables that capture periodicity and short-term trends:

- **Cyclical encodings:** Sine and cosine transformations of hour, weekday, and month were introduced to represent the repeating nature of time. This allows the model to recognize that, for example, 23:00 and 00:00 occur close to each other in behaviour even though numerically distant.

- **Rolling means:** Three-hour and 24-hour moving averages (each shifted by one hour to prevent data leakage) were added to represent short-term and daily price trends, smoothing temporary spikes.

These additional features are simple to compute and improve the model's ability to handle recurring and trending behavior in electricity prices, which are known to exhibit strong seasonality and temporal dependence [3].

3.3 Models

We choose a Linear Regression model with mean squared error (MSE) loss. Linear Regression is simple, interpretable, and efficient, and provides a benchmark for evaluating whether basic features already capture useful structure. Weron [3] noted that regression-based methods are effective baselines, while Mohsenian-Rad and Leon Garcia [2] demonstrated the usefulness of lag-based predictors.

In Stage 2, a **Random Forest Regressor** was added as a non-linear comparison model, implemented using scikit-learn. Random Forests consist of multiple decision trees trained on random subsets of data and features. By averaging their outputs, they can capture complex, non-linear interactions while reducing variance [3]. The model was trained using 120 trees with a maximum depth of 16 and a minimum leaf size of 2, providing a balance between accuracy and computational cost. Since both models are scale-invariant, no feature normalization was required [3].

Both models were evaluated using Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) in €/MWh. Linear Regression directly minimizes Mean Squared Error (MSE) directly, while the same metrics are used for Random Forest to ensure comparability.

3.4 Validation strategy

Because this is a time-series task, random splits are not appropriate. We train on data from 2020–2023 and validate on 2024. This ensures the validation set represents unseen future data, better reflecting real-world forecasting. The Stage 2 training period covers 2020–2023, validation spans

January–September 2024, and testing uses October–December 2024 to evaluate generalization on unseen future data.

4 Results

Table 1 summarizes the validation and test errors for the naive lag-24 baseline, linear regression, and random forest models. Both trained models clearly outperform the naive predictor. Random Forest achieved the lowest validation MAE (9.18 €/MWh) and test MAE (9.32 €/MWh), improving over the naive baseline by roughly 77%. Linear Regression remained competitive with a test MAE of 10.3 €/MWh.

Model	**Val MAE**	**Val RMSE**	**Test MAE**	**Test RMSE**	**Improvement vs Naive (%)**
Naive lag-24	28.58	57.59	40.47	69.04	–
Linear Regression	9.65	21.35	10.33	19.22	+74.5
Random Forest	**9.18**	**25.23**	**9.32**	**19.12**	**+77.0**

Table 1: Validation and test performance (€/MWh).

Figure 1 illustrates the model predictions during the first week of the 2024 test period (October). Both Linear Regression and Random Forest capture the overall daily price patterns and seasonal trends, although Random forest tracks overall the actual price with greater precision. The visible deviations of approximately ±10 €/MWh align with the test MAE values reported in Table 1, demonstrating the consistent performance of the models across different time periods.

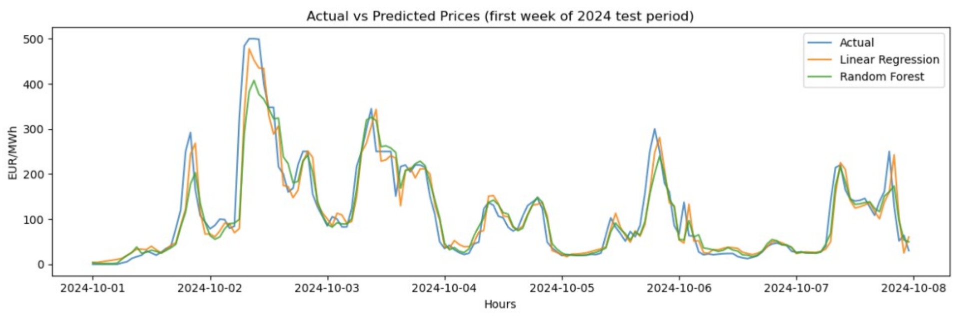

Figure 1: Actual vs predicted electricity prices for the first week of the 2024 validation period.

5 Conclusion

The comparison between Linear Regression and Random Forest demonstrates the benefit of non-linear modeling for electricity price dynamics. Both models produced accurate short-term forecasts using only historical price data. Random Forest slightly outperformed Linear Regression, suggesting that non-linear interactions among lagged and calendar features improve predictive accuracy. Nevertheless, Linear Regression remains a strong, interpretable baseline that requires minimal computation. Future work could integrate external variables such as forecasted demand, renewable-generation levels, or weather conditions to further enhance model performance and explainability.

Use of AI Tools

OpenAI ChatGPT was used during Stage 2 to assist with writing the feature-engineering section, refining the report structure, and generating the Python appendix template for model training and evaluation. All data analysis, parameter selection, and interpretation of results were conducted independently by the authors. AI support was limited to improving clarity, grammar, and implementation guidance for the more complex Stage 2 components.

References

[1] Nord Pool Market Data Public Available: `https://data.nordpoolgroup.com/auction/day-ahead/prices`

[2] A.-H. Mohsenian-Rad and A. Leon-Garcia, "Optimal residential load control with price prediction in real-time electricity pricing environments," *IEEE Transactions on Smart Grid*, vol. 1, no. 2, pp. 120–133, 2010.

[3] R. Weron, "Electricity price forecasting: A review of the state-of-the-art with a look into the future," *International Journal of Forecasting*, vol. 30, no. 4, pp. 1030–1081, 2014.

[4] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, 2006.

Appendix: Implementation Code

The complete implementation code for the electricity price forecasting model:

```
import pandas as pd
import numpy as np
from glob import glob
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error,
                                mean_squared_error
import matplotlib.pyplot as plt


df = pd.concat([pd.read_csv(f, sep=";") for f in files],
                                ignore_index=True)
df["DateTime"] = pd.to_datetime(df["Delivery Start (EET)"],
                                format="%d.%m.%Y %H:%M:%S",
                                errors="coerce")

df = df.set_index("DateTime").sort_index()
df = df[["FI Price (EUR)"]].rename(columns={"FI Price (EUR)":"
                                Price_EUR_MWh"})
df = df[df["Price_EUR_MWh"] < 1000]
df["hour"] = df.index.hour
df["weekday"] = df.index.weekday
df["month"] = df.index.month
df["sin_hour"] = np.sin(2*np.pi*df["hour"]/24)
df["cos_hour"] = np.cos(2*np.pi*df["hour"]/24)
df["sin_wday"] = np.sin(2*np.pi*df["weekday"]/7)
df["cos_wday"] = np.cos(2*np.pi*df["weekday"]/7)
df["sin_month"] = np.sin(2*np.pi*df["month"]/12)
df["cos_month"] = np.cos(2*np.pi*df["month"]/12)
df["lag_1"] = df["Price_EUR_MWh"].shift(1)
df["lag_24"] = df["Price_EUR_MWh"].shift(24)
df["lag_168"] = df["Price_EUR_MWh"].shift(168)
df["mean_3h"] = df["Price_EUR_MWh"].rolling(3).mean().shift(1)
df["mean_24h"] = df["Price_EUR_MWh"].rolling(24).mean().shift(1
                                )
df = df.dropna()
train = df.loc["2020-01-01":"2023-12-31"]
val   = df.loc["2024-01-01":"2024-09-30"]
test  = df.loc["2024-10-01":"2024-12-31"]
features = ["hour","weekday","month",
            "sin_hour","cos_hour","sin_wday","cos_wday",
            "sin_month","cos_month",
            "lag_1","lag_24","lag_168","mean_3h","mean_24h"]

def evaluate(y_true, y_pred):
    mae  = mean_absolute_error(y_true, y_pred)
    rmse = mean_squared_error(y_true, y_pred, squared=False)
    return mae, rmse
lr = LinearRegression().fit(train[features],train["
                                Price_EUR_MWh"])
```

```
rf = RandomForestRegressor(
    n_estimators=120,
    max_depth=16,
    min_samples_leaf=2,
    max_features="sqrt",
    random_state=42,
    n_jobs=-1
).fit(train[features], train["Price_EUR_MWh"])
val_naive = val["lag_24"]
test_naive = test["lag_24"]
results = pd.DataFrame([
    ["Naive lag-24", *evaluate(val["Price_EUR_MWh"], val_naive)
                                                    ,
     *evaluate(test["Price_EUR_MWh"], test_naive)],
    ["Linear Regression", *evaluate(val["Price_EUR_MWh"],
     lr.predict(val[features])),
     *evaluate(test["Price_EUR_MWh"], lr.predict(test[features]
                                                    ))],
    ["Random Forest", *evaluate(val["Price_EUR_MWh"],
     rf.predict(val[features])),
     *evaluate(test["Price_EUR_MWh"], rf.predict(test[features]
                                                    ))]
], columns=["Model","Val MAE","Val RMSE","Test MAE","Test RMSE"
                                               ])
print(results)
sample = test.iloc[:24*7]
plt.figure(figsize=(12,4))
plt.plot(sample.index, sample["Price_EUR_MWh"], label="Actual",
                                 alpha=0.8)
plt.plot(sample.index, lr.predict(sample[features]),
         label="Linear Regression", alpha=0.8)
plt.plot(sample.index, rf.predict(sample[features]),
         label="Random Forest", alpha=0.8)
plt.legend()
plt.title("Actual vs Predicted Prices (first week of 2024 test
                                period)")
plt.ylabel("EUR/MWh")
plt.xlabel("Hours")
plt.tight_layout()
plt.show()
```