Quant Dev Assessment - 2 hours, all tools allowed

> Create a private GitHub repo and share it with fw@nordicpowertrading.dk and bw@nordicpowertrading.dk. Commit frequently with meaningful, small commit messages.

If all phases are done well, you end up with a **predictive feature set** based on **cleaned alternative data from public APIs**. Focus on **clarity, correctness, and scalability**.

In some cases, the task descriptions include the exact methods you are expected to implement, but many best practices and principles are still up to you to consider and adhere to. At some cases the choice of method is also up to you.

Deliverables

When 2 hours has passed then you are welcome to do your last commit and push to github and also zip the entire project and send by email to us. Then we will prepare for the meeting.

1. `README.md` explaining what you did, how to run it, and a short summary of your feature findings.
2. A simple, reproducible **Python** environment (`requirements.txt` or `pyproject.toml`).
3. Code in `src/` and potentially also some code in `test/`
4. Save raw pulls to `data/raw/` (JSON/CSV exactly as received) and cleaned/engineered tables to `data/processed/` and `data/features`.
5. A jupyter notebook, ideally with 2–3 quick plots, that help you reason about feature importance.

> The goal is to find features that are predictive for the daily changes in **Germany front month price (€/MWh)**.

Suggestions for your approach

As this is an exercise, the setting is very artificial, and we are not looking for actual alpha. The data and features probably lack predictive power, and the target is itself sub-optimal.

So my advice is that even though it is an unsophisticated artificial setting, then strive to implement the methods with high standards. That means cleaning data properly, avoiding look-ahead bias, and tracking the experiments. To mention a few best practices. You might not have the time to adhere to and implement all best practices. Then you can note down in `REAMDME.md`, what you did not implement due to time constraints. You can provide a backlog.

Phase 1 — Data fetch

Goal: get **raw** data and store it locally. Keep it programmatic and reproducible.

Data you should fetch (minimal, public sources):

1. Front month base load price for Germany (this is for the target)

   I downloaded it manually from https://www.investing.com/commodities/german-power-baseload-electricity-m-futures and attached it as it is.

2. Historic Day-Ahead(DA) electricity prices for Germany(daily)

   You can use this token-less public API: https://api.energy-charts.info/

   (Optional): Also fetch price data for **France**.

3. Public power production historic (daily) for Germany

   You can use the same API: https://api.energy-charts.info/

   Use at least wind_onshore and solar. Note that it is not the forecasts we are looking for.

Phase 2 — Feature engineering

Goal: build a information-timely feature set that helps predict the **daily absolute change** in the **German front-month baseload future**.

1. Basic cleaning, if needed - Have a look at the raw data with some EDA method and conclude if it needs cleaning. If so, please clean the data accordingly.

2. **Construct target**

   From your attached CSV of the **front-month (FM) daily prices**:

   - Let F_t be the **front-month close** on trading day t.
   - **Primary label**:

     price_change_t = F_t − F_{t−1}

   Save the label as `price_change`.

3. Front-month–derived features

   1. **Momentum in changes**

      - `fm_d1` = ΔF_{t−1}
      - `fm_d5` = ΔF_{t−5}
      - `fm_trend_5` = rolling mean of ΔF over t−5, …, t−1

      Why: front-month moves often **cluster** (order-flow and risk-management persistence). Lags of signed changes capture that short-horizon momentum.

   2. **Stretch / z-score**

      - `fm_z_20` = (F_{t−1}−mean_{20}(F))/std_{20}(F)

Why: captures distance from a local equilibrium; extreme levels can precede larger adjustments.

4. Day-ahead(DA) price based features

   Note that day-ahead auction results for **delivery day T+1** are known from ~**12:57 CET** on T (orderbook closes **12:00 CET**). So DA-based and RES-forecast features can legitimately explain same-day futures moves.

   Compute:

   1. **Level & trend**

      - `da_de_roll7` = 7-day rolling **mean** of DE DA base
      - `da_de_roll28` = 28-day rolling **mean** of DE DA base
      - `da_de_trend_xover` = `da_de_roll7 - da_de_roll28`
      - `da_de_trend_xover_ramp` = `da_de_trend_xover_t - da_de_trend_xover_{t-1}`

      Why: the front-month equals the **expected average DA spot** over the front month; large moves in the **nearest spot (tomorrow's DA)** update that expectation and can move the future.

   2. **(Optional) Level & trend for France**

      - You can compute the same features for fr.

      Why: there is a lot of electricity flow between france and germany that effects the prices. So surprisingly low fr spot prices can reduce the expected spot prices for the next month in Germany.

5. **Renewables features**:

   Compute ramps (how today differs from yesterday)

   - `wind_ramp` = wind_onshore MWh(t)−wind_onshore MWh(t-1)
   - `solar_ramp` = solar MWh(t)−solar MWh(t-1)
   - `total_ramp = total(t) - total(t-1)`=`solar_ramp`+`wind_ramp` (total is the sum of wind and solar)

   Why: Changes in expected production are what reprice the nearby futures contract.

   Optional: If you want to (most likely) improve the predictive power, then go back to data fetching and get the production forecasts instead of historic productions. Then you can compute ramps based on how tomorrow differs from today. I don't know if it is possible to get a great timeseries of forecasts with the supplied api.

6. **Calendar features**

   - Day of week (one-hot), month (one-hot). Keep it minimal.

Why: power markets are structurally seasonal (weekday/holiday profiles, solar geometry by month). Even simple dummies often help stabilize a model.

Put your final feature table(with all features in one table) in data/features with columns: date, price_change (target), and the engineered features.

Phase 3 — Feature selection

Goal: find a **subset** with **predictive power** for `price_change`.

Implement these **three** complementary methods in a jupyter notebook:

1. **Filtering**: Rank by **Spearman** correlation and **mutual information** (`sklearn.feature_selection.mutual_info_regression`).
2. **Embedded**: **LassoCV** (L1) on standardized features. Report non-zero coefficients and their magnitudes.
3. **Model-agnostic**: **Permutation importance** using a small **RandomForestRegressor**. Report top contributors.