Time Copilot Prototype: Two-Page Business and Modeling Report

Executive Summary

This prototype tests whether a Time Copilot workflow can reliably select high-performing time-series models on open market data, with outputs that are actionable for trading and operations decisions. The result is yes: auto-selected champions consistently beat naive baselines and stay competitive against strong public baselines.

Key outcomes: - Forecast accuracy improved materially versus naive in both markets (PJM: `-34.6%` sMAPE, NP: `-15.3%` sMAPE). - Rally-risk prediction quality improved sharply versus naive (PJM: `2.31x` PR-AUC, NP: `2.43x` PR-AUC). - Winners differed by market and task, validating the need for automatic per-task/per-market model selection instead of one global model.

1) Business Problem (Why This Matters)

Commodity-facing teams make repeated decisions under uncertainty: - **Level decisions**: what is the most likely near-term value (price, spread, throughput proxy)? - **Risk decisions**: what is the probability of an adverse or opportunity event (rally/spike/outage-like condition)?

Traditional setups usually optimize one model for one metric and then reuse it broadly. This misses two realities: - Different markets have different dynamics. - Different decision types require different objective functions.

This prototype addresses both by benchmarking model families across two tasks and automatically selecting the best model by task and market.

2) Data Lineage and Governance

Data lineage was intentionally kept explicit and reproducible.

Step	Artifact	What Happens	Leakage / Repro Control
Source	EPF open datasets on Zenodo	Raw market data loaded for `PJM` and `NP`	Source is public and version-addressable
Cache	`datasets/PJM.csv`, `datasets/NP.csv`	Data cached locally for deterministic reruns	Same cached input reused across runs

Step	Artifact	What Happens	Leakage / Repro Control
Normalize	`normalize_epf_frame`	Canonical schema: `timestamp`, `price`	Timestamp parsing and schema validation
Feature build	`build_features`	Lag/rolling/calendar features (`lag_1`, `lag_24`, etc.)	Rolling stats are shifted to avoid look-ahead
Labeling	`build_rally_labels`, `label_future_rally`	Rally event target and future-horizon probability target	Future target is explicit and horizon-controlled
Split	`_train_test_split`	Chronological split (`80/20`)	Time-ordered split prevents train/test overlap
Evaluation	`forecast_metrics`, `classification_metrics`	Forecast: MAE/RMSE/sMAPE; Rally: PR-AUC/ROC-AUC/F1/Brier	Shared metric contracts across all models
Selection	`pick_forecast_champion`, `pick_rally_champion`	Auto-pick best model by task	Rule-based, deterministic tie-breakers

Primary output artifacts: - `artifacts/dual_market/<market>/forecast_benchmark.csv` - `artifacts/dual_market/<market>/rally_benchmark.csv` - `artifacts/dual_market/champion_summary.`

3) Benchmark Setup and Model Set

Two benchmark tasks: - **Forecast task**: next-step price prediction (optimize lower sMAPE). - **Rally task**: probability of rally within the next 24-hour horizon (optimize higher PR-AUC).

Model families: - **Reference**: `naive` - **Strong public baselines**: `lear` (regularized linear autoregressive), `logreg` (balanced logistic regression) - **Nonlinear learners**: `gbdt_reg`, `gbdt_cls` - **Neural**: `dnn_reg`, `dnn_cls`

Auto-pick rules: - Forecast champion = minimum `sMAPE` (tie-break: `MAE`) - Rally champion = maximum `PR-AUC` (tie-break: minimum `Brier`)

4) Performance Results (Including Strong Baselines)

Champion outcomes by market/task

Market	Task	Champion	Primary Metric	Champion Value
PJM	Forecast	gbdt_reg	sMAPE (lower is better)	12.01
PJM	Rally	gbdt_cls	PR-AUC (higher is better)	0.870
NP	Forecast	dnn_reg	sMAPE (lower is better)	5.54
NP	Rally	dnn_cls	PR-AUC (higher is better)	0.789

Lift versus naive and versus strongest non-champion baseline

Market	Task	Lift vs Naive	Strong Baseline Comparator	Lift vs Strong Baseline
PJM	Forecast	34.6% lower sMAPE (18.36 -> 12.01)	dnn_reg (13.20)	9.0% lower sMAPE
NP	Forecast	15.3% lower sMAPE (6.54 -> 5.54)	gbdt_reg (5.62)	1.4% lower sMAPE
PJM	Rally	+0.493 PR-AUC (2.31x)	dnn_cls (0.853)	+0.017 PR-AUC (+2.0%)
NP	Rally	+0.464 PR-AUC (2.43x)	logreg (0.787)	+0.002 PR-AUC (+0.26%)

Interpretation: - Gains over naive are large and consistent. - Against strong baselines, gains are market/task dependent but still positive; in NP rally, improvement is marginal, which is exactly the kind of nuance an auto-pick system should surface.

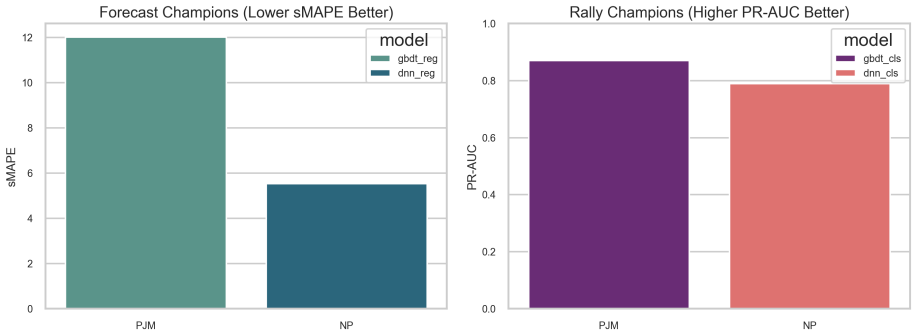

Figure 1: Champion scorecard

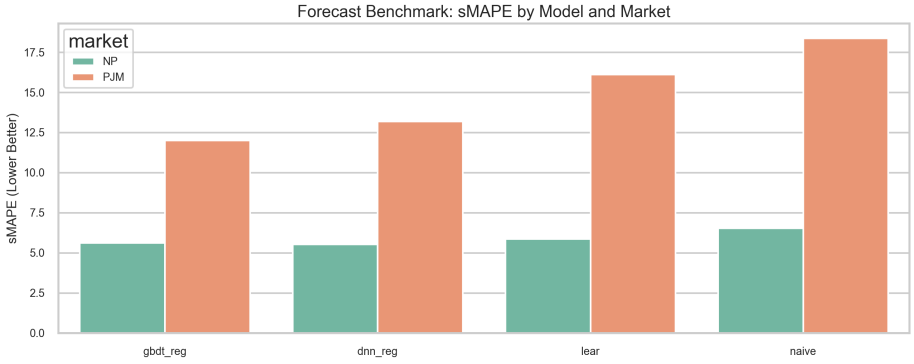

Figure 2: Forecast sMAPE by model

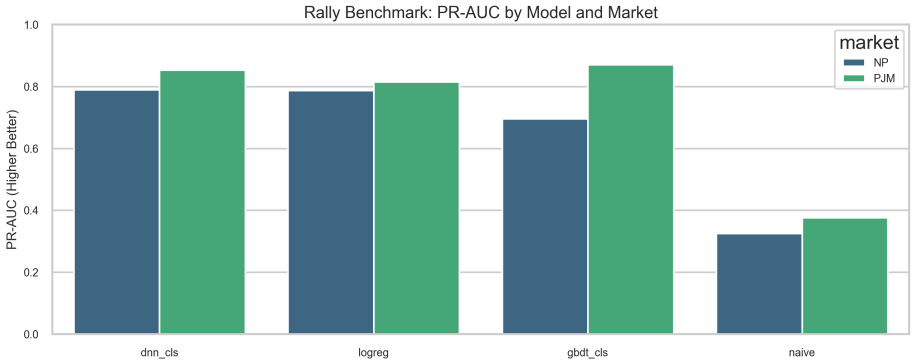

Figure 3: Rally PR-AUC by model

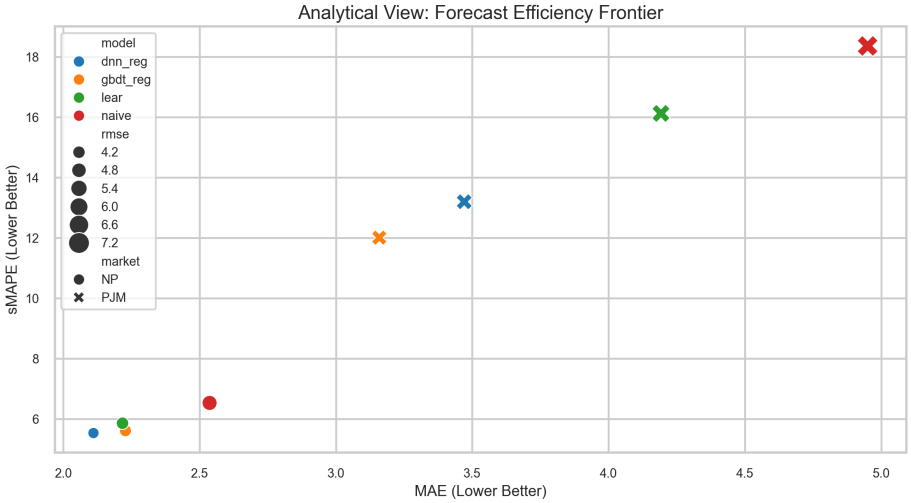

Figure 4: Forecast efficiency frontier

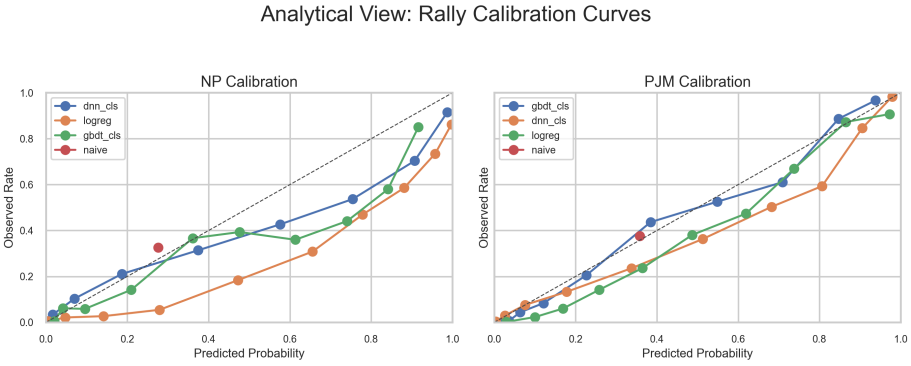

Figure 5: Rally calibration curves

5) Chart Evidence (Executive + Analytical)

Champion scorecard

Forecast benchmark (sMAPE)

Rally benchmark (PR-AUC)

Forecast efficiency frontier (error vs RMSE)

Rally calibration (top model per market)

6) Why This Changes Time-Series Modeling Practice

This prototype is important less because one model wins, and more because it changes **how** modeling is operationalized: - Moves from single-model forecasting to **portfolio benchmarking**. - Separates **level prediction** from **event-risk prediction**, matching real decision needs. - Treats model selection as a governed, repeatable output (`champions.json`), not ad hoc analyst choice. - Makes model swaps cheap: adding/removing models does not change the evaluation contract. - Produces decision-ready artifacts (tables + charts) that non-ML stakeholders can review quickly.

In short, this is a shift from "fit one time-series model" to "run a model selection system for time-series decisions."

7) Limits and Next Validation Gate

Current limits: - Public electricity data is a proxy, not your internal production domain. - Single chronological holdout (`80/20`) is useful for prototype speed but not the final governance standard. - Neural models run, but further hyperparameter tuning could improve stability and consistency.

Next gate before internal rollout: 1. Re-run the exact framework on internal targets with the same contracts and metrics. 2. Add rolling backtests (multiple time windows) to validate stability. 3. Compare against the desk's current heuristic or incumbent model as the operational baseline.