Final Submission Walkthrough & Competency Review

This document explains, step-by-step, how the final solution for the Restaurant Turnover prediction hackathon was built, why each action was taken, and what we expected to observe at each stage. It concludes with a candid review of strengths, gaps, and recommendations for growth based on the questions asked and the assistance needed during execution.

Team: Section 9 | Metric: RMSE | Public Leaderboard: Top-5 finish.

- Core approach: robust preprocessing, careful handling of categorical variables, two complementary gradient-boosted trees (CatBoost and LightGBM), and a log-space convex blend selected by out-of-fold (OOF) validation.
- Submission format: exactly two columns — Registration Number, Annual Turnover — with 500 test rows and a header.

1. Data, Folders, and Reproducibility

We organized the project under a fixed PROJECT_ROOT with subfolders for raw data, processed data, models, and submissions. Google Drive was mounted so the notebook could read/write stable paths. Parquet caches were created for faster reloads and to preserve dtypes.

A schema guardrail verified that 'Registration Number' existed in both train and test, and 'Annual Turnover' existed only in train. Uniqueness of IDs in train/test was asserted to prevent double counting or leakage. These checks make downstream errors easier to diagnose.

- Raw → Processed caching (CSV → Parquet) for speed and dtype stability.
- Assertions on ID presence, target presence, and ID uniqueness.
- Non-destructive branching: train_raw/test_raw ➜ train_clean/test_clean.

2. EDA Objectives and Findings

Before modeling, we inspected types, missingness, cardinality, and potential drift. A constant-mean baseline established a reference RMSE (~21.5M). Key observations:

- Target was right-skewed; log1p transformed target was used for LGBM training to stabilize loss.
- Date column ('Opening Day of Restaurant') required day-first parsing; derived features included Age_days (recency), Open_month (seasonality), and Open_dow (day of week).
- Several ratings had non-trivial missingness. We created informative missingness flags and used median imputation for the rating values themselves.
- Categoricals showed mixed cardinality. We one-hot encoded low-cardinality columns and reserved high-cardinality columns for specialized encodings.

3. Data Cleaning and Feature Construction

We standardized column names where needed, corrected a known typo ('Endoresed By' → 'Endorsed By'), and converted City sentinel '-1' to the explicit category 'Unknown'. Date parsing used dayfirst=True to match the provided format.

For rating-type features, we added __isna indicators (to capture the information that a value is missing) and applied median imputation (robust to skew).

- City: '-1' → 'Unknown' to avoid accidental numeric treatment or silent dropping.
- Date: day-first parsing; derived Age_days, Open_month, Open_dow.
- Ratings: __isna flags + median impute; indicator + value work together for trees.

4. Categorical Encoding Strategy

Categoricals were split by cardinality. Low-cardinality columns were one-hot encoded (with alignment across train/test). High-cardinality columns were handled with target encoding (TE) using K-Fold to avoid leakage.

- City__te: K-Fold TE with smoothing≈200. Built OOF encodings for train; applied pooled statistics to test.
- Restaurant Theme__te: repeated the same K-Fold TE approach.
- Restaurant Type__te: evaluated; impact was negligible on CV, so not retained as a focus feature.
- Frequency encoding was used as a light-touch alternative when appropriate.

5. LightGBM Model (Log-Target)

LightGBM (LGBMRegressor) was trained on the log1p-transformed target to reduce the penalty of extreme outliers. Early stopping prevented overfitting and suggested an effective number of trees.

We sanitized feature names and collapsed duplicates to avoid LightGBM parser issues with JSON-unsafe characters or repeated column names (common after one-hot alignment).

- CV: 5-fold KFold with shuffle and fixed seed for reproducibility.
- Typical settings: n_estimators large with early stopping, learning_rate≈0.02, num_leaves≈31, min_child_samples≈40, subsample/colsample≈0.8, reg_lambda≈1.0.
- Outcome: LGBM with City__te + Theme__te achieved about ~20.06M CV RMSE.

6. CatBoost Model (Raw Target)

CatBoost natively handles string categoricals and missing values using ordered statistics. We trained it on the raw (un-logged) target since CatBoost is generally robust to skew and the metric is RMSE on the original scale.

- CV: 5-fold KFold, early stopping (od_type='Iter'), depth≈6–8, learning_rate≈0.03–0.05, l2_leaf_reg≈3–6, iterations high with OD to stop early.
- Outcome: CatBoost CV around ~19.83–19.88M RMSE, consistently stronger than a single LGBM run.

7. Blending in Log-Space

Because the target is right-skewed, we blended model predictions in log space: log1p(pred) were averaged with a convex weight w for CatBoost and (1−w) for LGBM, then expm1 was applied to return to the original scale. We swept w on OOF predictions and chose the weight minimizing OOF RMSE.

- Best OOF weight: ~0.75 on CatBoost (0.25 on LGBM).
- OOF blend RMSE ≈ 19.97M (better than either model alone).
- We also tested a linear ridge blend. Coefficients were unstable and introduced intercept bias; we preferred convex blends for stability and leaderboard behavior.

8. Optional Calibration and Robust Clipping

We optionally calibrated the log-space blend by fitting a 1D linear map on OOF: y_log ≈ a * oof_log + b. This can correct mild over/underestimation bias between models and the target distribution. We also applied light percentile clipping (0.5%–99.5%) to tame a few extreme predictions that disproportionately affect RMSE.

Public-leaderboard feedback indicated the plain log-space blend was at least as strong as the calibrated variant; the final chosen submission was the non-calibrated log-space blend.

- Calibration parameters observed: a≈1.1153, b≈−2.0594 (used only in the calibrated submission).
- Final choice: non-calibrated log-space convex blend at w≈0.75, saved as submission_blend_convex_logspace.csv.

9. Validation Discipline & Leakage Avoidance

All encodings that could leak target information used K-Fold OOF construction. Target encoding for City/Theme was computed within folds for train and with pooled statistics for test. Sanity checks ensured that target was never present in test and that ID columns were excluded from features. Early stopping was based only on in-fold validation splits.

- OOF construction central to weight selection and calibration.
- No GroupKFold was ultimately necessary for the final blend; simple KFold generalized best.
- Right-tail handling (log-space training and/or blending) reduced variance and improved public leaderboard performance.

10. Results, Reproducibility, and Packaging

The final entry achieved a Top-5 leaderboard placement. To make results reproducible, the final notebook was duplicated and run clean top-to-bottom; the exact winning CSV was fingerprinted with SHA256 and copied into a final/ folder alongside a short run_report.json, requirements.txt, and README.md. This captures code, data assumptions, and environment for auditability.

- Winning file: FINAL_submission_blend_convex_logspace.csv (copy of the best-scoring CSV).
- Artifacts: run_report.json (metrics, weights, versions), README.md (one-page summary), requirements.txt (pip freeze), and the final notebook.
- This packaging makes it straightforward to reproduce or review the work later.

11. Key Pitfalls We Encountered (and How We Resolve

A few issues appeared repeatedly during development. Addressing them in a structured way prevented time loss later and improved stability.

- LightGBM column-name errors (JSON-unsafe characters, duplicates): solved by sanitizing names and collapsing duplicate columns after one-hot alignment.
- Early-stopping API differences: used callback-based early_stopping() and log_evaluation() to be version-agnostic.
- Unstable blend coefficients from unconstrained linear regression: switched to convex OOF weight sweep and, optionally, log-space calibration.
- Small TE improvements: City/Theme helped; Restaurant Type had negligible impact and was de-prioritized.

12. Competency Review — Strengths, Gaps, and Caree

The following observations synthesize the questions asked and the support requested throughout the project. They aim to highlight current strengths, areas for deliberate practice, and how these map to real-world data science and MLOps work.

- Strengths observed:
- — High perseverance and willingness to instrument each step (kept the pipeline clean and reproducible).
- — Solid intuition for leakage risks and evaluation (used OOF CV, checked schema alignment, and validated fold variance).
- — Healthy skepticism of metrics vs. leaderboard (compared OOF to public LB; chose conservative, robust steps).
- Skill gaps / growth edges:
- — Feature engineering patterns: build a small library of go-to transforms (date cyclic, on-demand interactions, binning heuristics).
- — Categorical handling taxonomy: deepen intuition on one-hot vs. TE vs. CatBoost-only and when to prefer each.
- — Model selection discipline: run ablations with a compact grid (num_leaves/min_child_samples/smoothing) and log them systematically.
- — Tool ergonomics: comfort with callbacks, early stopping, and version-safe APIs to avoid time lost on minor errors.
- Career fit & trajectory:
- — Likely to excel in roles mixing rigor and communication: analytics engineering, MLOps for tabular modeling, applied DS for operations.
- — Methodical packaging (SHA, run report, README) mirrors production review culture—good signal for reliability.
- — Watch-out: over-extending into broad hyperparameter searches; keep experiments surgical and hypothesis-led.
- Concrete next steps:
- — Build a tiny 'tabular template' repo and reuse it on 3–5 public datasets.
- — Practice TE smoothing sweeps and keep a results log.
- — Add lightweight experiment tracking (CSV log or MLflow).
- — Give a 5–8 min lightning talk on 'Why log-space blends help with skew'.

13. Closing Notes

The final solution emphasizes correctness, leakage safety, and reproducibility. The blend is intentionally simple and robust, which tends to translate well to hidden test sets. Future work could explore learned stacking with additional base models, improved high-card encoders (e.g., exporting CatBoost encodings for LGBM), and richer date/city interaction features.

Congratulations on the Top-5 result.