Predicting Delivery Duration

Developing Third Party Delivery ETA Prediction Tools with Horizon Models and Meta-Stacking

Why Delivery-Time Prediction Matters

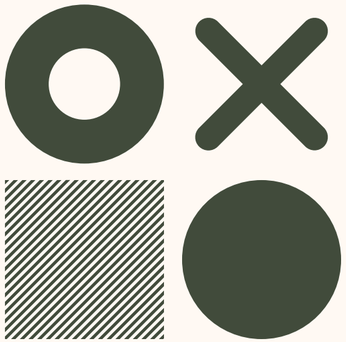
Introduction

Accurate ETAs play a critical role in customer trust and operational efficiency, and even small errors can meaningfully impact user experience.

- Accurate ETAs are essential for customer trust and operational efficiency, and even small errors can significantly impact the delivery experience.
- Under-prediction leads to late deliveries, customer credits/refunds, and a degraded user experience.
- Persistent ETA inaccuracies erode customer trust, increase churn, and negatively impact revenue.
- Delivery conditions shift constantly; restaurant prep speed, dasher supply, traffic, and market congestion all introduce variability.

Objective

Develop a robust, data-driven model that produces reliable, customer-facing delivery-time predictions across highly dynamic operating conditions.



DOORDASH

DoorDash Fulfillment Data

Dataset Summary

- 197,428 delivery records
- Mix of order metadata, store attributes, and real-time marketplace telemetry
- Two timestamp fields used to compute the target:
  - created_at
  - actual_delivery_time
- Target variable: target_delivery_seconds

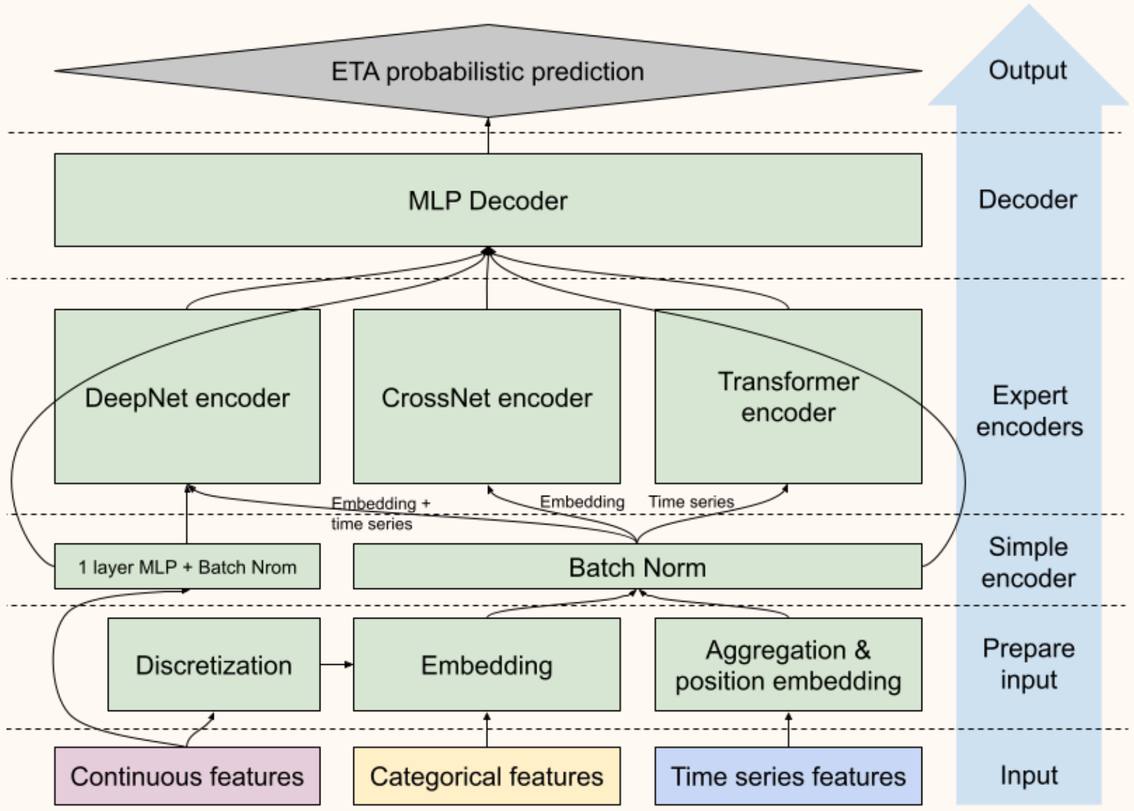
Key Feature Groups

- Order Details
  - total_items, num_distinct_items, subtotal, min_item_price, max_item_price
- Store & Order Metadata
  - store_id, store_primary_category, order_protocol
- Marketplace Telemetry
  - total_onshift_dashers, total_busy_dashers, total_outstanding_orders
- Estimated Durations
  - estimated_order_place_duration, estimated_store_to_consumer_driving_duration

Modeling Approach

The DoorDash Approach

- ETA errors come from many shifting factors like prep time, dasher availability, market congestion, and traffic.
- Their production model uses a Mixture-of-Experts deep learning architecture that includes DeepNet, CrossNet, and a Transformer-based encoder.
- They rely on higher-cardinality features, embeddings, and real-time temporal signals that are not present in the public dataset.
- The system predicts a full delivery-time distribution using interval regression and a Weibull likelihood to manage long-tailed uncertainty.



Preprocessing and Feature Engineering

Preprocessing Steps

- Time zone normalization for all timestamps
- Hierarchical missing-value imputation
- Outlier filtering using IQR rules
- Computation of target_delivery_seconds
- Leakage-safe handling of all temporal and market features

Key Engineered Features

- Temporal features: hour, day, month, weekend, half-hour buckets
- Marketplace congestion features: busy ratio, demand-supply ratio
- Rolling and lagged telemetry signals
- Velocity indicators based on short-term vs medium-term changes
- Store-level performance statistics
- Order structure features: buckets, ranges, average item price
- Categorical Encodings and Interaction Features

Baseline Model and Horizon Models

Baseline LightGBM

- LightGBM model trained on all engineered features
- Strong single-model benchmark for comparison

Horizon Models (Specialized Subsets)

- Short-Horizon (SH): item mix, prep duration, early-stage signals
- Medium-Horizon (MH): marketplace load, supply-demand ratios
- Long-Horizon (LH): store identity, structural duration effects

Why Horizon Modeling?

- Different parts of the delivery process behave differently
- Specialization reduces variance and improves model stability
- Inspired by DoorDash’s multi-encoder, multi-expert structure

Stacking Strategy and Final Model

Stacking Approach

- RidgeCV meta-model
- Inputs: OOF predictions from SH, MH, LH, and the Baseline
- Produces a single blended ETA estimate

Advantages

- Stable linear weighting
- Resistant to multicollinearity
- Improves MAE across most order durations
- Smooths extreme behavior from individual horizon models

Model Evaluation Results

Ridge Stacking Model

- Selected alpha: 50.0
- Intercept: −231.50

Model Weights

- Baseline: 0.891
- LH: 0.080
- SH: 0.072
- MH: 0.039

Validation Performance

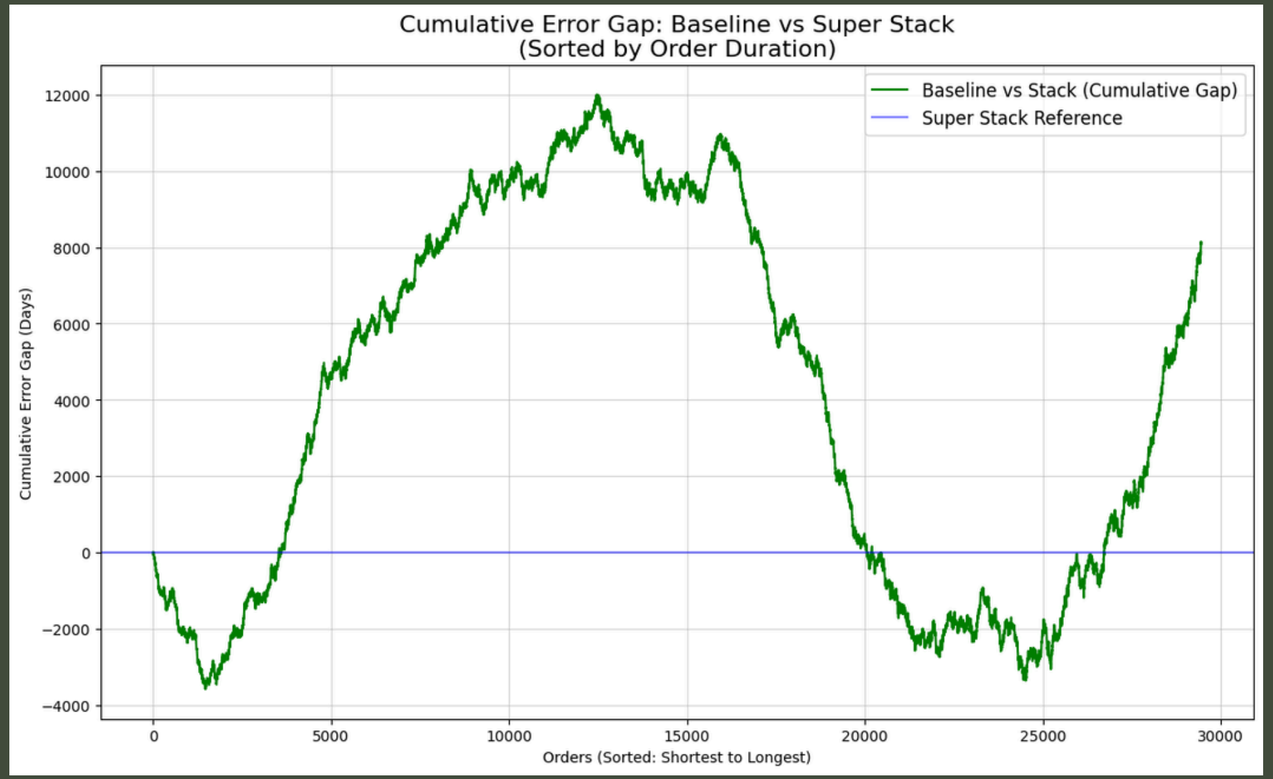
- MAE: 569.10 sec
- RMSE: 758.66 sec

Test Performance

- MAE: 601.90 sec
- RMSE: 805.64 sec

Comparison

- Baseline Test MAE: 602.53 sec
- Improvement: ≈0.63 MAE points



Where a More Advanced Model Could Improve

Opportunities Identified

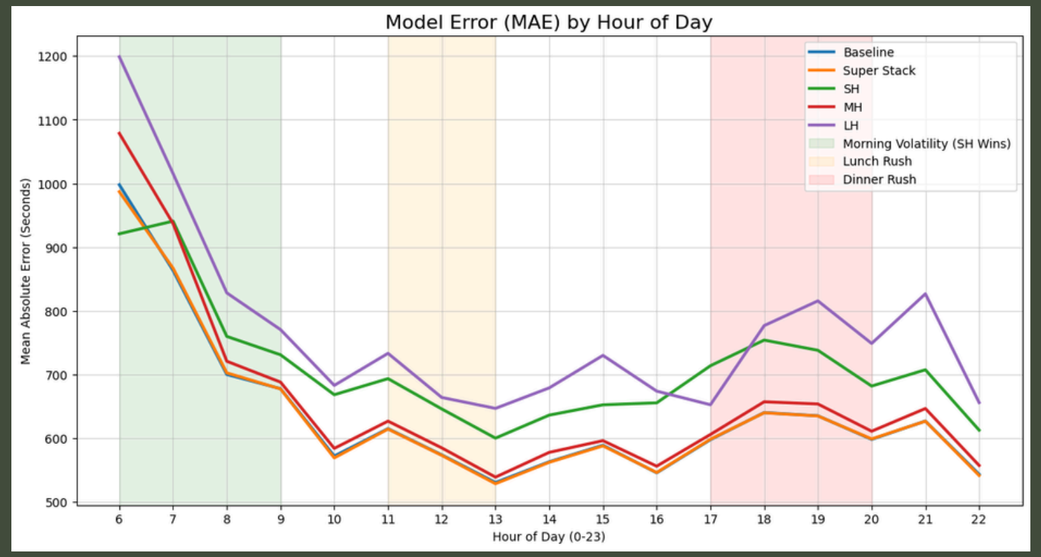
- Horizon models outperform the baseline and stack in specific conditions
- SH model excels early morning during cold-start volatility
- SH and LH models outperform the stack in the early deciles on the lateness metric

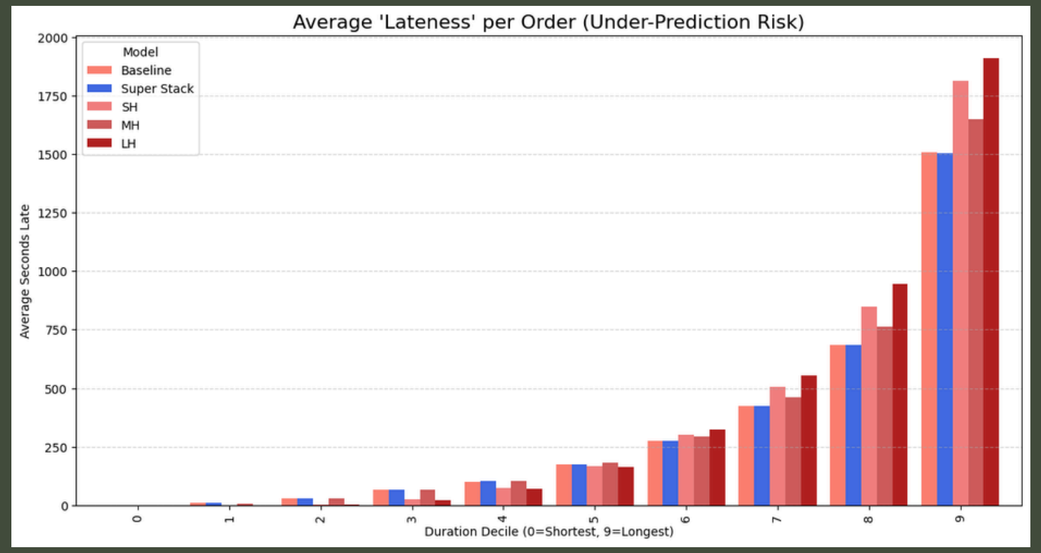
Limitations of Linear Stacking

- Fixed weights cannot adapt to time-of-day or market conditions
- Interaction structure between features and horizons remains unused
- No non-linear blending or gating mechanism

Potential Improvements

- Tree-based or neural meta-learner for dynamic weighting
- Time-aware or market-aware interaction features
- Incorporation of sequential telemetry patterns (mini Transformer-like encoder)





Conclusion, Limitations, and Future Work

Key Conclusions

- Baseline model is strong, but horizon models provide complementary strengths.
- Ridge stacking delivers small but consistent improvements over a single-model approach.
- Visual analyses reveal clear opportunities for context-dependent weighting.

Limitations

- Linear stacker uses fixed global weights that cannot adapt to time-varying conditions.
- No mechanism to emphasize SH or LH when they outperform the stack in specific deciles.
- Dataset is smaller and less feature-rich than DoorDash's internal telemetry pipeline.

Future Work

- Replace ridge stacker with a tree-based or neural meta-learner for dynamic weighting.
- Engineer richer time- and market-aware interaction features.
- Incorporate sequence patterns with light Transformer-style encoders.

Thank You!