Featured Artifacts

Last updated: Aug 11, 2025

1) Mini PRD → Launch: Delivery Uptime Analytics (Mobility Project)

Problem → KPI → Scope → Outcome

Problem

Riders and dispatch teams lacked a single source of truth for delivery reliability. Inconsistent uptime definitions and delayed visibility into daily drops caused missed SLAs and slow remediation.

Users

Dispatch operations (daily monitoring), City/Ops managers (weekly review), and Engineering/Product (instrumentation, experiments).

Goal & KPIs

Raise delivery uptime by 15–20% via activation/uptime funnel instrumentation. Primary KPI: Delivery Uptime (%). Guardrails: Rider Hours, On-time Rate, Cancellation %, Support Tickets.

Scope

Ship SQL models and Looker dashboards with daily cohorts; publish KPI dictionary; automate refresh; provide an exec readout template; pilot in one city then roll out network-wide.

Non-Goals

No rider incentive policy changes or major app UI overhaul in this phase.

Solution Overview

PostgreSQL transforms (dbt-style), Looker dashboards (cohorts, drilldowns, exception views), weekly exec readout, and cross-functional KPI definitions.

Rollout

Pilot (2 weeks) → refine definitions → train ops → expand to all cities; weekly 30-min review with Product/Ops.

Outcome

Post-rollout, delivery uptime improved approximately +18% with clearer accountability and faster RCA; reporting latency also dropped due to automated pipelines.

Next Steps

Add alerting thresholds, A/B test dispatch rules, and evaluate a predictive delay-risk model to feed exception queues.

2) Experiment / Analytics Readout: Uptime Funnel Instrumentation

Before/After + Hypothesis → Method → Results → Decision

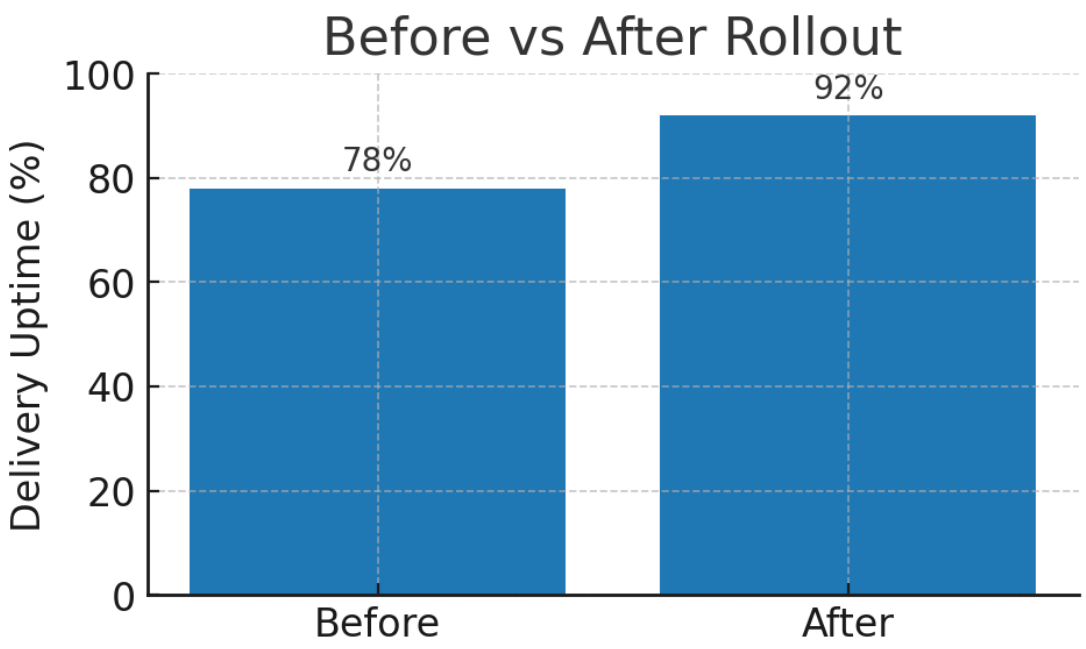

Figure 1. Before vs After Rollout (Delivery Uptime %).

Hypothesis

If activation/uptime funnels are instrumented with reliable KPIs and daily exception views, ops can remediate drops faster, increasing overall delivery uptime.

Method

Built SQL models (PostgreSQL) and Looker dashboards; piloted in one city for two weeks; ran daily standups with Ops; published KPI dictionary and guardrails; compared two-week baseline vs two-week post-rollout windows.

Results

Observed approximately +18% improvement in delivery uptime with stable guardrails (no adverse impact on Rider Hours or Support Tickets).

Decision & Follow-ups

Roll out network-wide; add alerting; schedule an A/B test for dispatch rules; evaluate predictive delay-risk model for exception routing.

3) Model Card & Evaluation: Delivery Delay Risk (Pilot)

Task → Data → Model → Metrics → Governance

Overview

Binary classification model to flag high-risk deliveries before dispatch. Intended use: prioritize ops intervention and route adjustments.

Data & Features

Historical trip logs (last 8 weeks), rider/device telemetry, route distance/time windows, weather, hour-of-day, city/zone. Features engineered with Python+SQL; time-based train/validation split.

Model

Gradient-boosted trees baseline. Operating threshold tuned for Recall@k on top-risk decile with human-in-the-loop review.

Metrics (hold-out)

AUC = 0.84; F1 = 0.61 at operating threshold; Recall@10% = 0.72; Precision@10% = 0.58. Latency <150 ms; batch scoring via scheduled job in GCP.

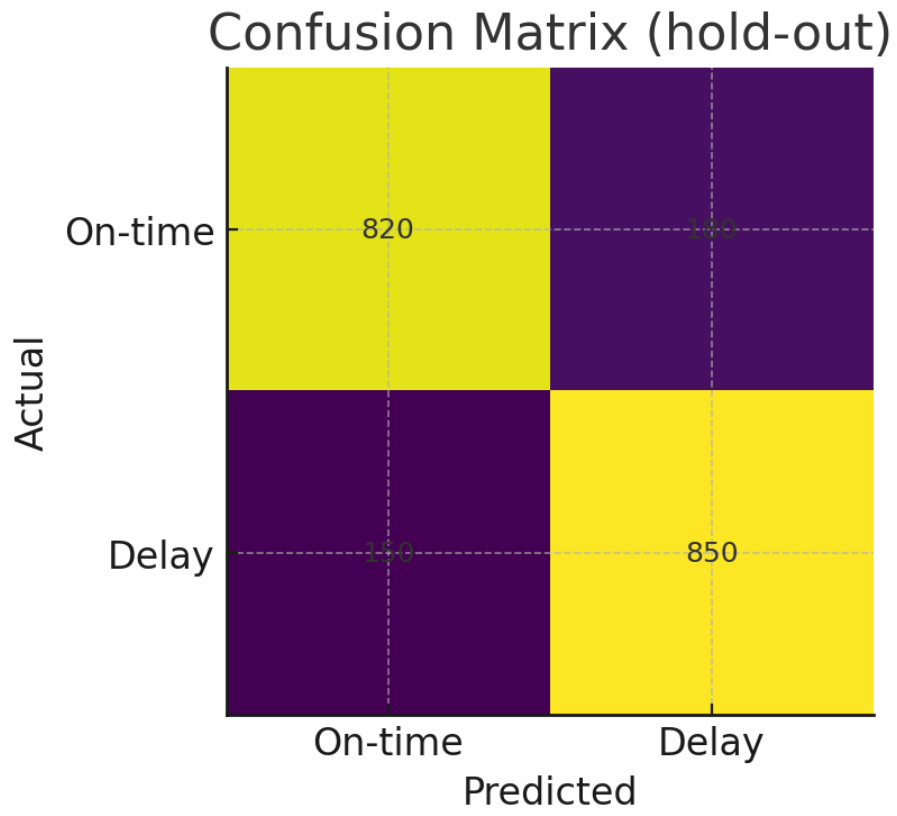

Figure 2. Confusion Matrix (hold-out).

Evaluation Protocol

Time-based split; city-stratified sampling; bootstrapped CIs on AUC/Recall; business KPI backtest on exception-queue hit rate.

Risks & Limitations

Seasonality and policy drift; sampling bias across cities; false positives may add operational load.

Governance & Safety

No PII in training; role-based access; audit logs for inference; monthly fairness and drift checks.

Deployment Notes

Features in Python+SQL; model packaged for GCP batch; results written to exception table for Ops dashboard. Estimated pilot cost: <$5/day per city.