# HubBalance: Intelligent Bluebikes Demand Forecasting & Rebalancing

**Vaishali Singh**
CS5100
Final Report

December 3, 2025

## 1 Introduction

### Problem Statement

Urban bike-sharing systems like Bluebikes (Boston) face a critical "last-mile" inefficiency: **imbalance**. Commute peaks and special events frequently leave popular stations either empty (preventing pickups) or full (preventing returns), degrading user trust and system reliability. Manual rebalancing by dispatchers is often reactive rather than proactive.

### Project Goal

**HubBalance** addresses this by building an end-to-end predictive system. It forecasts station-level demand (pickups and drop-offs) and stockout risk 1–24 hours in advance. Unlike "black box" forecasters, HubBalance provides interpretable natural-language rationales (e.g., *"High stockout risk due to evening peak + impending rain"*), empowering dispatchers to make confident, data-driven rebalancing decisions.

## 2 Method

The system is implemented as a self-contained Python pipeline comprising three core stages:

### 2.1 Synthetic Data Simulation (ETL)

To ensure reproducibility and test the pipeline's robustness without external API dependencies, I implemented a sophisticated synthetic data generator:

- `make_city_context()`: Simulates hourly city-wide signals, including seasonality (temperature/wind via sine waves), MBTA transit headways (rush hour variance), and event pulses (random spikes in activity).
- `simulate_station_timeseries()`: Models station usage as a Poisson process. Demand rates ($\lambda$) are modulated by station base rates, weather favorability, transit availability, and day-of-week logic. It tracks inventory state (`available_bikes`) dynamically, flagging "stockout" events (0–1 bikes/docks) for classification.

### 2.2 Feature Engineering

The raw timeseries is transformed into a rich feature set (56 total features) to capture complex urban dynamics:

- **Temporal:** Cyclic encodings (Sine/Cosine of Hour) and Weekend flags to capture commute patterns.
- **Environmental:** Weather context (Precipitation Probability, Temperature) and Civic context (311 Service Calls, Event Scores).
- **Momentum (Lags):** Rolling windows of historical usage (Lags 1h, 3h, 6h) for pickups, drop-offs, and availability.
- **Spatial:** One-hot encodings for Station ID and Neighborhood to learn localized demand profiles.

### 2.3 Modeling & Algorithms

We employ **LightGBM (Gradient Boosted Trees)** for its efficiency and ability to handle non-linear feature interactions.

1. **Regression:** Two separate `LGBMRegressor` models predict `pickups` and `dropoffs` counts to estimate net flow.
2. **Classification:** An `LGBMClassifier` (with class weighting) predicts the binary `stockout` probability to flag critical stations.
3. **Explainability:** We use **SHAP (SHapley Additive exPlanations)** to compute feature attribution scores, translating complex model weights into human-readable "rationales" for each prediction.

## 3 Complexity

The project demonstrates significant technical depth appropriate for the course timeline:

- **Multi-Source Fusion:** The pipeline successfully integrates disparate signal types (temporal, spatial, meteorological, and lag-based) into a unified tabular structure.
- **End-to-End Architecture:** Rather than just fitting a model, the system implements a full lifecycle: Data Generation $\rightarrow$ Feature Engineering $\rightarrow$ Time-Series Split $\rightarrow$ Modeling $\rightarrow$ Decision Support.
- **Interpretable AI:** The inclusion of SHAP values elevates the project from simple prediction to actionable business intelligence, bridging the gap between raw metrics and operational utility.

## 4 Results and Discussion

### 4.1 Model Performance

Evaluated on a time-aware holdout set (final 20% of the timeline), the gradient-boosted approach significantly outperformed baselines:

Table 1: Regression Metrics (MAE = Mean Absolute Error, lower is better)

| Metric | Persistence Baseline | Seasonal Baseline | HubBalance (LightGBM) |
|---|---|---|---|
| Pickups MAE | 3.31 | 2.59 | **1.98** (*∼24% Improvement*) |
| Drop-offs MAE | 2.09 | 1.61 | **1.43** (*∼11% Improvement*) |

### 4.2 Classification & Decision Support

The Stockout Risk classifier achieved strong separation between safe and critical states:

- **ROC-AUC:** 0.701

- **AUPRC (Precision-Recall):** 0.923

This high Average Precision indicates the model is highly reliable when it flags a station as "at risk," minimizing false alarms for dispatchers.

### 4.3 Actionable Insights (Rebalancing)

The system outputs a prioritized plan. The model successfully learned to prioritize stations with low momentum and adverse weather conditions.

**Example Dispatch Plan Output**

**Station:** JP Centre
**Action:** **Deliver 7 bikes**
**Rationale:** *Recent low availability raises risk (+0.43); Lagged drop-offs indicate incoming deficit.*

## 5 Communication

- **Visuals:** The analysis includes feature importance beeswarm plots (via SHAP) and time-series comparisons of Actual vs. Predicted demand, clearly illustrating the model's fit.
- **Documentation:** The code is modularized into clear functions (`add_time_features`, `add_lags`) with docstrings. The notebook serves as a reproducible artifact.

## 6 Deliverables & Links

- **Codebase:** A fully documented `README.md` and the `hubbalance_offline_demo.ipynb` notebook are provided at github.com/vaishalisingh04/hubbalance , ensuring the project is runnable offline.
- **Slides:** A concise 10-slide deck (`hubBalance.pptx`) summarizes the problem, method, and impact for non-technical stakeholders.