Summary Sheet

Team #M2026232

Problem. Using the provided elevator operational logs, design a practical control policy to: (i) forecast short-horizon demand, (ii) identify the current traffic regime, and (iii) proactively park *idle* elevators to reduce passenger waiting and long-wait events.

Modeling framework (closed loop). We implement a deployable supervisory layer that acts only on idle cars:

- **Task 1 (Forecasting):** Aggregate hall calls into 5-minute slices and predict next-slice demand with a gradient-boosted decision tree (Route A). As a transparent benchmark (Route B), we use a time-of-day baseline with regime $AR(1)$ correction.
- **Task 2 (Traffic regime):** Discover operational modes from 5-minute feature vectors (intensity, directionality, dispersion, inter-floor tendency, and service pressure) using KMeans [1], then map clusters to manager-friendly labels (e.g., *Up-Peak*, *Down-Peak*, *Idle/Low*). As an interpretable fallback (Route B), we also implement a rule-based regime classifier using the same 5-minute intensity and direction features, providing a transparent baseline for state labeling.
- **Task 3 (Dynamic parking):** Learn a mode-conditioned floor-demand distribution and select parking floors for idle elevators by solving a weighted 1D k-median problem (Route A). As a transparent fallback (Route B), we additionally use a state-aware zone allocator (Lobby/Mid/Upper) with rate-limited, minimal-move repositioning. Repositioning is *idle-only* to avoid disrupting in-service dispatch.

Key quantitative highlight (Task 1 benchmark). On the held-out test set, the Route B baseline+$AR(1)$ improves one-step 5-minute demand prediction over the pure time-of-day baseline (MAE 4.675 vs. 4.781, RMSE 8.325 vs. 8.528; about 2.2% and 2.4% relative reductions, respectively).

Evaluation summary (Task 3). Using a call-level, parameter-consistent simulator, we compare mode-aware dynamic parking against static baselines and a zone-based fallback. Across overall and operationally critical windows (peak demand, regime transitions, and high-load intervals), the dynamic policy consistently reduces average waiting time and tail risk (P95/P99) while exposing repositioning cost through an explicit empty-travel metric and using a rate-limited, minimal-move fallback policy for safe deployment. We further include stress tests to probe failure modes under demand shocks and regime confusion, and we recommend an explicit rollback switch to the interpretable Route B controller when mode confidence degrades. Because the long-wait rate is threshold- and parameter-dependent, it can be near zero under the nominal setting; we therefore emphasize tail-aware metrics (P95/P99) and stress tests where tail risk re-emerges under heavy-load and perturbed parameters (Appendix A.6).

Strengths and implementation readiness. The approach is (i) data-driven yet interpretable, (ii) lightweight (5-minute decision epochs), (iii) deployable as a supervisory layer on top of standard dispatch logic, and (iv) robust through conservative design choices (idle-only repositioning, fallback policy). For deployment risk control, we monitor signal availability and mode-confidence; when these degrade, the supervisor automatically rolls back to the transparent Route B controller (Appendix A.8) without interrupting in-service dispatch.

What is novel here?

- Mode-aware parking: parking targets adapt to detected regime shifts rather than fixed rules.
- Optimization view: parking floors are selected by a demand-weighted 1D k-median objective (fast and interpretable).
- Engineering safeguards: idle-only moves, rate limits, and a transparent fallback policy.
- Fail-safe switching: a simple confidence/stability check to toggle between Route A and Route B for safe deployment.

Limitations. Performance depends on calibration of travel/door-time parameters and on the representativeness of the observed log period. Field deployment should validate under seasonal and event-driven shifts and monitor repositioning cost (empty travel).

Contents

1 Introduction

Elevator group control in high-rise and mixed-use buildings must maintain acceptable service quality under demand that is strongly time-varying and regime-dependent (e.g., morning up-peak, lunch inter-floor, and evening down-peak).[2–4] When mechanical capacity is fixed, a practical lever is *software-level* supervision: in particular, proactively repositioning *idle* cars can reduce response distance and stabilize waiting times during regime shifts.

This report designs a deployable supervisory policy using *only* the provided operational logs. We build a closed-loop pipeline that (i) forecasts short-horizon hall-call demand, (ii) identifies the current traffic mode from real-time features, and (iii) applies mode-aware dynamic parking for idle cars. The core parking decision is posed as a one-dimensional, demand-weighted facility-location problem (1D k-median), yielding parking floors that minimize expected response distance to near-future calls. The final deliverables include both a technical solution and a management memo describing implementation and operational safeguards.

To balance performance with deployability, we implement two complementary routes: **Route A** (data-driven) combines learned forecasting and unsupervised mode discovery with optimization-based parking, targeting best service performance; **Route B** (interpretable) provides a transparent baseline and a conservative fallback policy for benchmarking and deployment risk control. Throughout, our policy is *non-intrusive*: it acts only on idle cars and can sit on top of standard dispatch rules. In particular, Route B instantiates a rule-based mode classifier and a state-aware zone parking allocator (Lobby/Mid/Upper) that generates a minimal reposition plan under simple rate limits, making it suitable both as a benchmark and as a rollback controller.

Our main contributions are:

- A reproducible modeling pipeline that maps logs to a real-time traffic representation and a mode-aware parking decision.
- A lightweight 1D k-median parking formulation that is interpretable, fast, and compatible with supervisory deployment.
- A dual-route design (performance + fallback) and an evaluation protocol that emphasizes peak periods and regime transitions where proactive parking matters most.

The remainder of the paper is organized as follows. Section 2 formalizes the tasks and assumptions. Section 3 describes data processing and feature construction. Section 4 presents the forecasting-mode identification-parking pipeline. Section 5 reports performance and robustness analyses, and Section 6 summarizes deployment guidance in a management memo.

2 Problem Decomposition and Analysis

2.1 Objective and decision scope

The operational goal is to improve passenger service quality under stochastic, regime-dependent demand using only the provided logs. We focus on *supervisory* control: at discrete decision epochs, the system may reposition *idle* elevator cars to selected parking floors. This preserves compatibility with any existing in-service dispatch rule and avoids intrusive mid-trip interventions.

We evaluate policies with service-oriented and tail-risk metrics: average waiting time (AWT), upper-tail quantiles (e.g., P95/P99 of waiting time), and the long-wait rate (fraction of calls exceeding a specified threshold). When reporting long-wait rate, we state the thresholding rule explicitly and use it consistently in all tables (including windowed analyses).

2.2 Data-driven closed loop: three tasks

The problem naturally decomposes into a closed-loop pipeline operating on 5-minute intervals:

1. **Task 1: Short-horizon demand forecasting.** Aggregate hall calls into 5-minute slices. Let y_t denote the total number of hall calls in interval t. Given recent history and time features, predict $\hat{y}_{t+1}$.
2. **Task 2: Traffic regime identification.** From real-time interval features (call intensity, directionality, spatial dispersion, and inter-floor tendency), infer a discrete traffic mode $s_t \in \mathcal{S}$ (e.g., up-peak, down-peak, mixed/inter-floor, idle/low).
3. **Task 3: Mode-aware dynamic parking for idle cars.** Given $(\hat{y}_{t+1}, s_t)$, compute a set of parking floors $\mathcal{P}_t = \{p_{t,1}, \ldots, p_{t,K}\}$ for K elevator cars when they are idle, and apply repositioning subject to non-interference constraints.

This decomposition is motivated by two realities of elevator operations: (i) demand is strongly nonstationary and exhibits recurring regimes, and (ii) parking decisions are most valuable when *anticipating* near-future calls

rather than reacting after queues build.

2.3 Key modeling assumptions (made explicit)

Because only log data are provided, several operational details are abstracted. We adopt assumptions that are standard in elevator-traffic analysis and compatible with supervisory deployment:

- **Discrete decision epochs.** The policy updates every 5 minutes; within an epoch, dispatch is handled by the existing controller.
- **Idle-only repositioning.** Reposition commands apply only to idle cars, preventing service disruption.
- **Distance-proxy service model.** Waiting time is monotonically related to response distance between a call floor and the nearest available car. We therefore optimize and evaluate parking using distance-based proxies consistent with the available logs.
- **Cost awareness.** Repositioning may increase empty travel; we track an explicit empty-travel metric to expose the service-cost tradeoff.
- **Comparability of evaluation.** Simulation parameters (travel/door time) affect absolute seconds, so we interpret results primarily through *relative* comparisons across strategies under the same parameter setting and call stream.

These assumptions yield a robust and implementable framework without requiring a full physics-accurate elevator simulator.

2.4 Outputs and validation plan

Each task produces an output that can be validated independently and jointly:

- Task 1 output: $\hat{y}_{t+1}$ with forecast error metrics (MAE/RMSE) on held-out intervals.
- Task 2 output: s_t with qualitative validation against time-of-day patterns and cluster interpretability, and quantitative stability via transition frequency and feature separation.
- Task 3 output: $\mathcal{P}_t$ and reposition actions, evaluated by AWT, tail quantiles, long-wait rate, and empty travel.

To reflect operational importance, we additionally report *windowed* performance during peak periods, regime-transition windows, and high-load intervals, where proactive parking has the largest impact.

2.5 Two-route design: performance vs. interpretability

We implement two complementary routes to balance accuracy and deployability:

- **Route A (data-driven).** Learned forecasting and mode discovery are coupled with optimization-based parking to maximize service performance.
- **Route B (interpretable).** A transparent baseline forecaster and a zone-based parking policy serve as a benchmark and a conservative fallback.

Route B is included to (i) establish a trustworthy baseline, (ii) support deployment under uncertainty, and (iii) prevent overfitting to one log period.

3 Data Processing and Analysis

This section describes the data cleaning procedures and presents key empirical observations derived from the processed datasets with exploratory visualization. With validation of data reliability, we extract evidence-based insights that directly inform subsequent problem decomposition and model design.

3.1 Data Cleaning Principles and Validation

3.1.1 Cleaning Principles

Given that the raw datasets, though extensive, contain missing data and potentially inconsistencies, that would severely compromise the generality and extensibility of the models. Furthermore, the properties and practical significance of certain indicators in the dataset will also undermine the accuracy of data statistics. Therefore, to prepare the elevator datasets for robust mathematical modeling, we implemented a systematic cleaning process that ensures data integrity, consistency, and alignment with the problem's requirements for traffic prediction, mode classification, and dynamic parking strategies. Operations were executed in Python using Pandas, with code

available in the appendices for reproducibility. This step mitigates potential biases from raw data artifacts, such as sensor errors or incomplete logs, enabling accurate time-series analysis.

The cleaning process follows several domain-specific principles:

- **Integrity over completeness:** Only records violating physical or logical constraints were removed, resulting in retention rates exceeding 85% for all major datasets.
- **No artificial imputation for structural variables:** Discrete and semantically critical variables, such as floor numbers, were never imputed to avoid introducing spatial bias.
- **Preservation of congestion signals:** Records with long waiting times or heavy passenger loads were retained, as they represent genuine system stress rather than noise.
- **Domain consistency and standardization:** Time stamps were converted to datetime format, floors to integers, and physical constraints (e.g., passenger loads in [0, 2100] kg) were enforced.

These principles ensure that the cleaned data exhibit integrity, accuracy, domain consistency, validity, and operability, thereby providing directly support the problem's emphasis on adaptive models, as clean data reduces noise in predictions and optimizations.

3.1.2 Data Cleaning Operations

As detailed in **Cleaning Principles**, all datasets were cleaned under four core principles. Table 1 summarizes retention rates and usage scope. Each dataset was processed according to its role in our modeling pipeline, with cleaning operations explicitly justified by domain constraints and task requirements:

- Facing **missing or unparseable floor values** (e.g., `NaN` or strings like "4,5") in `hall_calls.csv`, we removed records with no valid origin floor and split composite floor requests into individual calls; this ensures that every retained hall call can be unambiguously mapped to a physical floor, which is essential for constructing spatially resolved demand forecasts—the foundation of our short-term prediction and parking strategy.
- Facing **inconsistent floor representations** (e.g., "G", "B1") and missing directions in `car_stops.csv`, we converted floors to integers and inferred direction from sequential stop patterns; this yields a clean trajectory log that accurately reflects real elevator movements, enabling reliable validation of our simulated traffic modes against ground-truth stop behavior.
- Facing **physically implausible load readings** (e.g., negative or >2100 kg) and **unordered timestamps** in `load_changes.csv`, we filtered loads to the valid range [0, 2100] kg and sorted records by (`Car ID`, `Time`); this preserves temporal coherence of passenger flow and eliminates sensor artifacts, allowing us to estimate realistic boarding/alighting volumes for time-series simulation of elevator occupancy.
- Facing **duplicate or out-of-order entries** in `maintenance_mode.csv`, we deduplicated and chronologically sorted the maintenance logs; this provides a precise mask of non-operational periods, ensuring that all downstream analyses exclude times when elevators were offline—thus avoiding artificial underestimation of system demand.
- Facing **raw timestamp and floor fields without standardization** in `car_calls.csv` and `car_departures.csv`, we applied consistent datetime and integer formatting; while these datasets are not used in our core predictive model, this minimal cleaning prepares them as auxiliary signals for future extensions (e.g., destination-aware dispatching), maintaining data readiness without over-processing.

This section details the data cleaning process for the elevator datasets, ensuring high-quality input for mathematical modeling in traffic prediction, operational mode classification, and dynamic parking strategies. Cleaning was performed using Python with Pandas, focusing on integrity, standardization, and domain-specific validations. Operations prioritize retaining representative data while removing invalid entries without bias.

3.1.3 Data Cleaning outcomes and Validation

Cleaned files can be found in the **Appendix**.

File	Operations	Retained (%)
`hall_calls.csv`	Standardized time/direction; expanded multi-floor calls; removed NaN floors.	223,339 (86%)
`car_stops.csv`	Standardized time/floors; inferred direction; removed invalid stops.	214,263 (98%)
`load_changes.csv`	Standardized time/floors; filtered loads to [0, 2100] kg; sorted by car/time.	216,884 (99%)
`maintenance_mode.csv`	Standardized time; deduplicated and sorted.	161 (100%)
`car_calls.csv`	Standardized time/floors; filtered valid actions.	255,971 (99%)
`car_departures.csv`	Standardized time and floor fields.	218,491 (100%)

Table 1: Summary of data cleaning operations, rationale, and retention rates.

We validate our cleaning process along three criteria inspired by Xiong et al. [5]: (i) adherence to physical and operational constraints, (ii) preservation of task-relevant signals—particularly congestion and peak behavior—and (iii) retention of representative temporal patterns across weekdays, weekends, and traffic regimes. These criteria are especially critical in elevator modeling, where spatial accuracy, load realism, and timing fidelity directly affect prediction and control performance.

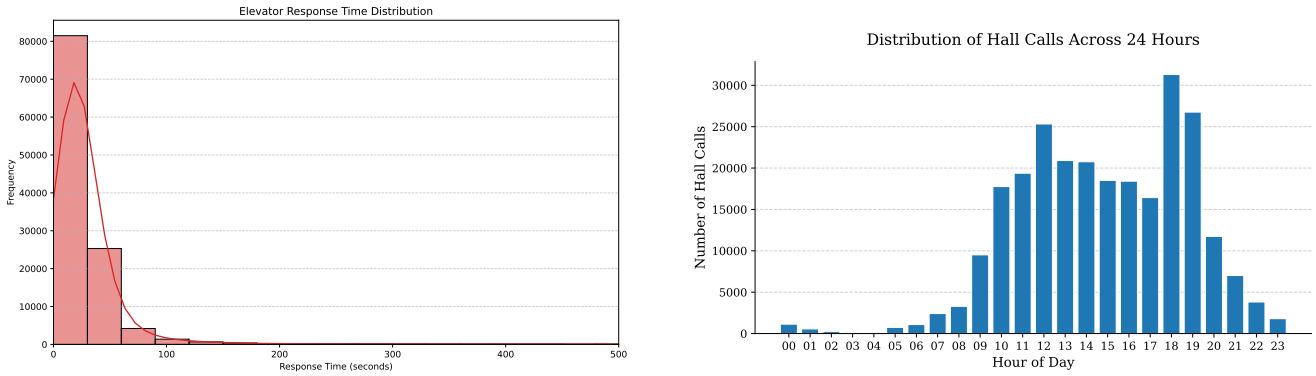
All datasets satisfy these standards. The `hall_calls.csv` file provides the most detailed illustration due to its central role in demand forecasting. It originally contains 259,013 records, of which 34,724 (13.4%) lack a valid origin floor. Because floor numbers are discrete and semantically non-interpolable, we removed—but never imputed—these entries to avoid spatial bias. No records were filtered based on waiting time or call frequency; long waits during morning/evening peaks are retained as genuine indicators of system stress. The removed records show no clustering by hour or day of week, suggesting missingness is random (likely from transient sensor or transmission errors) rather than systematic underreporting. After cleaning, 224,289 calls remain (86.6%), preserving demand heterogeneity across all operational conditions.

The remaining files undergo analogous validation. In `car_stops.csv` and `car_calls.csv`, floor values are converted to consistent integers and invalid stops or actions discarded, ensuring trajectory integrity for mode inference. `load_changes.csv` enforces the physical load bound [0, 2100] kg while retaining high-load events that reflect real passenger volumes. `maintenance_mode.csv` is deduplicated to provide an accurate mask of offline periods, and `car_departures.csv` is chronologically sorted to support precise departure tracking. Across all six files, retention rates range from 86.6% to 100%, confirming that cleaning targets only structurally invalid entries—not extreme but valid operational states.

Collectively, the cleaned datasets are physically plausible, temporally coherent, and faithful to real-world elevator dynamics. Every cleaning decision is documented, deterministic, and grounded in domain knowledge—ensuring reproducibility and alignment with our modeling objectives [5].

3.2 Data Visualization

We conducted a comprehensive exploratory data analysis (EDA) to uncover temporal, directional, and spatial patterns in elevator usage. Seven key visualizations were generated, designed to answer: *When, where, in which direction, and how intensely is the elevator service requested?*.



(a) Response time distribution

(b) Hourly hall call volume

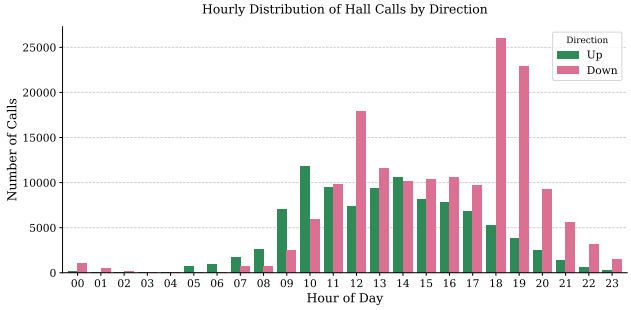
Figure 1: System responsiveness and overall temporal demand pattern.

The heatmaps (Figures 4a and 4b) expose spatial heterogeneity. On weekdays, intense activity concentrates between Floors 1–3 (main lobby and exits) and Floors 10–14 (likely office zones), especially during 8–10 AM and 5–7 PM.. In contrast, weekends show diffuse, low-intensity usage across all floors, confirming minimal structured traffic.

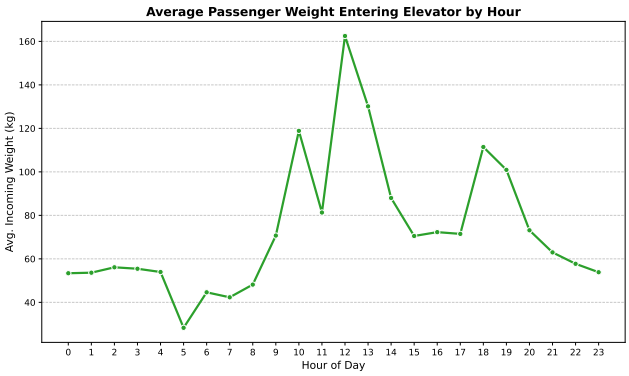
3.3 Empirical Findings from EDA

Using the cleaned datasets, we conducted exploratory analysis along five dimensions: time, direction, weekday cycle, spatial distribution, and passenger-load proxies. The conclusions below are supported by Figures 1–4 and directly motivate our modeling choices in Tasks 1–3.

(F1) Service is generally adequate but degrades under predictable surges. The response-time distribution (Figure 1a) indicates that most calls are served within a short time under normal operation, suggesting that the primary challenge is not mechanical insufficiency but *anticipating and managing demand surges. Note:* this response-time statistic is computed from the provided operational logs (observed service), whereas Task 3



(a) Up vs. down calls by hour



(b) Average passenger weight entering per hour

Figure 2: Directional traffic modes and load intensity validation.

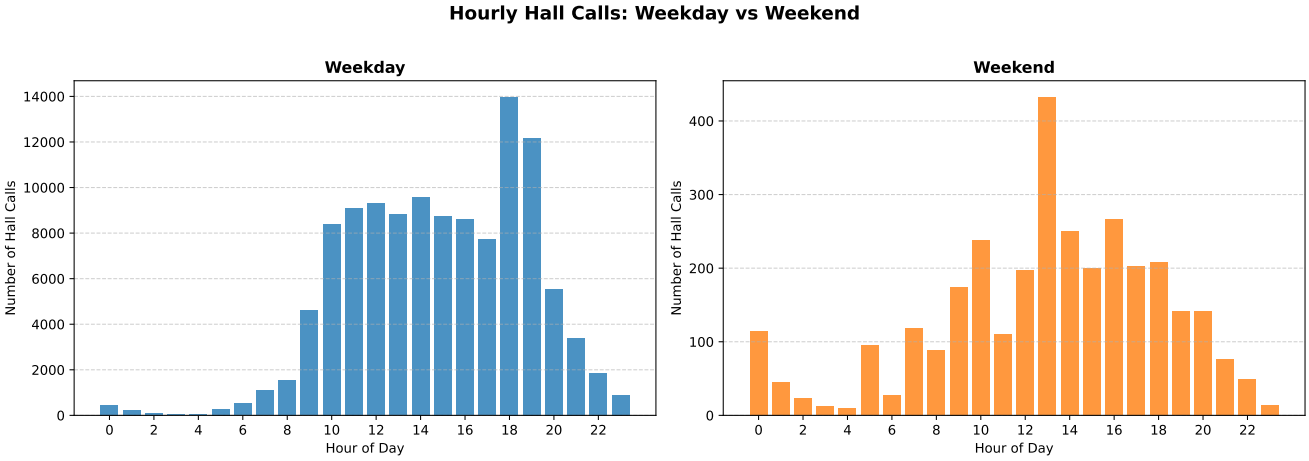
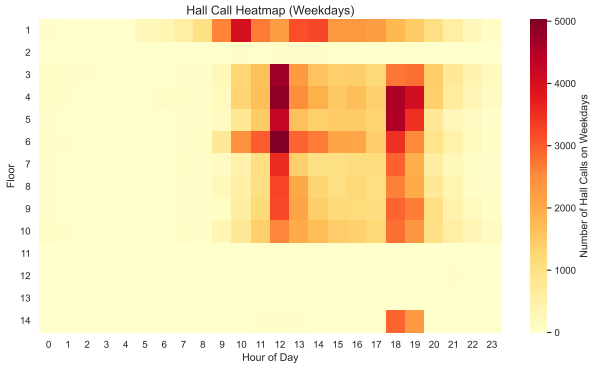
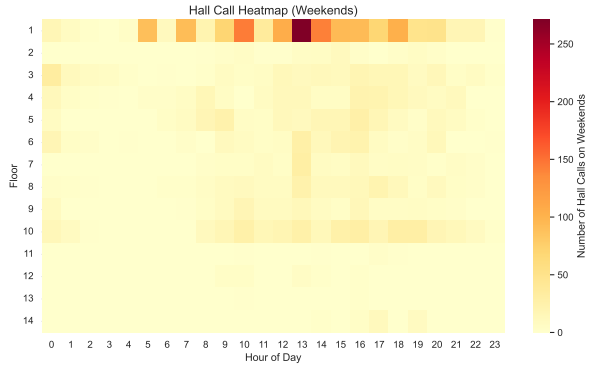


Figure 3: Weekday versus weekend call volume, showing structured office traffic on weekdays.



(a) Weekday spatial-temporal heatmap



(b) Weekend spatial-temporal heatmap

Figure 4: Spatial distribution of hall calls across floors and hours.

evaluates alternative parking policies with a distance-based proxy under fixed parameters; the latter is used for *relative* strategy comparison.

(F1b) Demand is strongly time-dependent with recurring daily peaks. Hourly hall-call volume (Figure 1b) exhibits a bimodal pattern with consistent peaks around midday and early evening. This supports time-of-day features and short-horizon forecasting (Task 1).

(F2a) Directional asymmetry confirms distinct traffic regimes. Figure 2a shows a classic morning up-peak and an evening down-peak. Thus, regime identification (Task 2) must incorporate directionality rather than relying solely on total volume.

(F2b) Peaks correspond to real loading intensity. Average incoming passenger weight (Figure 2b) rises during the same periods as call-volume peaks, validating that peaks reflect genuine congestion rather than spurious button presses.

(F3 & 4) Weekday-weekend differences and floor heterogeneity are substantial. Weekday demand is structured and significantly higher than weekend demand (Figure 3). Spatial-temporal heatmaps (Figure 4) show that weekday calls concentrate at lower lobby floors (1–3) and mid-to-upper office floors (approximately 10–14) during peaks, while weekend usage is sparse and diffuse. These patterns require floor-aware strategies in both prediction and parking.

3.4 Feature Engineering

We construct interval-level features on 5-minute bins to support forecasting (Task 1) and traffic regime identification (Task 2), and to parameterize mode-conditioned parking demand (Task 3). Features are designed to be minimal, interpretable, and computable in real time from logs.

Temporal and calendar features. Hour-of-day, day-of-week, and weekday/weekend flags capture recurring human-activity patterns (F2, F5).

Call intensity and directionality. Let C_t be the total number of hall calls in interval t, and $C_t^{\uparrow}, C_t^{\downarrow}$ the up/down counts. Define the directional ratio

$$r_t = \frac{C_t^{\uparrow}}{\max(1, C_t)}$$

which separates up-peak from down-peak regimes (F3).

Spatial concentration and dominant-floor indicators. Let $p_{1,t}$ denote the share of calls originating from the busiest floor in interval t (dominance), and let a dispersion statistic (e.g., the entropy of floor distribution) measure whether demand is concentrated (lobby/office peaks) or diffuse (F5).

Inter-floor tendency and load proxies. When available, aggregated load-change magnitude within each interval provides a continuous proxy for passenger intensity (F4). We also compute the fraction of non-lobby calls (or a similar index) to distinguish inter-floor movement from commute-like traffic.

These features jointly encode *when* demand occurs, *where* it concentrates, and *in which direction* it flows—the three axes needed for short-horizon prediction, regime classification, and floor-aware parking.

3.5 Synthesis of Findings

The EDA yields five actionable insights that directly map to our modeling components:

- Demand is highly time-dependent and non-stationary (supports Task 1 forecasting).
- Traffic exhibits distinct, regime-specific directional patterns (supports Task 2 mode identification).
- High call volume aligns with high loading intensity, confirming that peaks represent genuine congestion.
- Weekday and weekend dynamics differ substantially and should be treated separately in modeling.
- Demand varies significantly by floor, motivating spatially resolved parking strategies (Task 3).

Collectively, these findings motivate time-adaptive forecasting, real-time mode classification, and floor-aware dynamic parking in the subsequent sections.

4 Modeling

We propose a deployable, data-driven elevator control pipeline that links forecasting, traffic-mode recognition, and proactive parking. The key design principle is to keep each module both operationally interpretable and computationally light enough for real-time use in legacy group control systems.

4.1 Overview and Modeling Philosophy

The problem requires a proactive elevator control policy that can (i) forecast near-term demand, (ii) recognize the current traffic regime, and (iii) reposition idle elevators to reduce passenger waiting. To satisfy these requirements with a transparent and deployable pipeline, we build a three-stage framework:

(i) **Task 1: Short-term demand forecasting** to predict the number of hall calls in the next 5-minute slice;
(ii) **Task 2: Traffic-mode discovery and online classification** to identify the current building usage regime (e.g., up-peak, down-peak, inter-floor);
(iii) **Task 3: Mode-aware dynamic parking** to choose parking floors for idle elevators by solving a weighted 1D k-median problem using the mode-conditioned floor-demand distribution.

The outputs form a closed loop: the forecast and recognized mode determine the proactive parking targets, and the resulting performance is validated via a lightweight simulation against standard baselines.

4.2 Model 1: Short-Term Passenger Arrival Forecasting (Task 1)

4.2.1 Target and Time Discretization

We discretize time into uniform slices of length $\Delta = 5$ minutes. Let y_t denote the total number of passenger arrivals in slice t. Consistent with the available logs, we use the number of *hall calls* in each slice as a demand proxy; this variable is directly observable and closely aligned with perceived service quality (waiting time).

The forecasting task is a one-step-ahead regression:

$$\hat{y}_{t+1} = f_\theta(x_t), \tag{1}$$

where x_t is a feature vector constructed from historical demand and temporal context.

4.2.2 Feature Engineering

To capture both periodic regularity and short-term dependence, we design x_t with four groups of real-time features:

(a) **Cyclical time features:** daily and weekly periodicity are encoded using sine/cosine transforms (e.g., $\sin(2\pi \cdot \text{minute\_of\_day}/1440)$ and $\cos(2\pi \cdot \text{minute\_of\_day}/1440)$), and similarly for the minute-of-week.
(b) **Calendar indicators:** hour, minute, and a weekend flag.
(c) **Lagged demand:** $\{y_{t-\ell}\}$ for several lags ℓ (e.g., 5–60 minutes) to capture short-term autocorrelation.
(d) **Rolling statistics and trend:** rolling mean/standard deviation of recent slices and a 1-hour trend term $y_t - y_{t-12}$.

All features are constructed without future information, ensuring the model can operate online.

4.2.3 Model Choice and Validation Protocol

We adopt a gradient-boosted decision tree regressor (GBDT) due to its ability to model nonlinear interactions among mixed feature types, while remaining robust to noise and irregular demand spikes[6, 7]. The dataset is split *chronologically* to avoid temporal leakage: the first portion is used for training and the remaining portion for testing. We report mean absolute error (MAE) and root mean squared error (RMSE) for interpretability and sensitivity to extreme deviations.

Benchmark (Route B). To provide a transparent and deployable baseline, we implement a time-of-day baseline and a regime $AR(1)$ correction. Table 2 reports that the $AR(1)$-corrected predictor improves one-step 5-minute demand forecasting over the pure time-of-day baseline in both MAE and RMSE.

Table 2: Task 1 one-step forecasting benchmark (Route B).

Model	MAE	RMSE	N
Baseline+AR(1) (Route B)	4.675	8.325	1726
Time-of-day baseline (Route B)	4.781	8.528	1726

Table 3: Task 1 forecasting performance (5-minute demand).

Metric	Value
MAE	4.758
RMSE	8.697

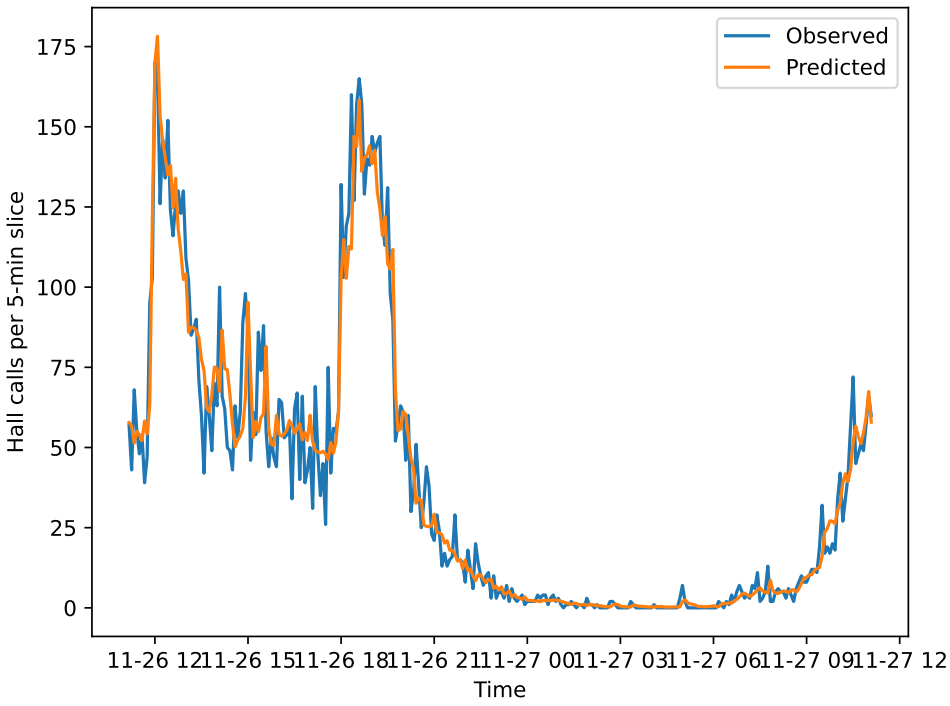


Figure 5: Observed vs. predicted 5-minute demand for a representative window.

4.2.4 Output to Downstream Control

The one-step-ahead demand forecast provides a quantitative early warning signal. While Task 3 primarily depends on the mode-conditioned floor distribution, the forecast helps determine whether aggressive repositioning is warranted (e.g., during predicted surge periods versus idle periods).

4.3 Model 2: Traffic Mode Discovery and Online Classification (Task 2)

4.3.1 Mode Features with Operational Meaning

Passenger traffic regimes in buildings typically exhibit distinct signatures in intensity, directionality, and spatial concentration. We construct a feature vector for each 5-minute slice that is both informative and interpretable, including:

- **Intensity:** hall-call count in the slice.
- **Directionality:** up/down ratio from hall calls.
- **Spatial dispersion:** floor entropy (higher entropy indicates dispersed origins).
- **Inter-floor tendency:** proportion of car calls not involving the lobby (proxying inter-floor movement).
- **Service pressure:** number of stops and departures.
- **Load dynamics and maintenance:** net load change and maintenance ratio (if available).

To emulate real-time sensing, we smooth selected features using a short rolling window (e.g., 15 minutes), producing stable online signals for clustering and classification.

4.3.2 Unsupervised Clustering and Interpretability

Since no ground-truth labels for "traffic modes" are provided, we use KMeans clustering on standardized features, a method widely adopted for traffic state classification in intelligent transportation systems[8]. KMeans solves:

$$\min_{\{\mu_k\}} \sum_{t} \|x_t - \mu_{c(t)}\|^2, \tag{2}$$

where $c(t)$ assigns slice t to its nearest centroid. We choose a small number of clusters (e.g., $K = 6$) to balance granularity and interpretability; cluster quality can be sanity-checked with a silhouette score and validated via downstream performance stability [9]. In our implementation, we set $K = 6$ to capture common micro-regimes while maintaining interpretability; the resulting silhouette score (approximately 0.34) indicates moderate separation, which is expected because traffic regimes overlap in practice.

4.3.3 Cluster-to-Mode Naming

To satisfy MCM's requirement of explainable managerial recommendations, we convert numerical clusters into operational mode labels by inspecting each cluster's feature profile (mean intensity, up/down ratio, entropy, and inter-floor ratio). For example, a high-intensity cluster with dominant up direction is labeled as *Up-Peak*, while a low-intensity cluster is labeled as *Idle/Low*. This hybrid approach is data-driven in discovery yet transparent in interpretation. The mode timeline is aligned with the typical up-peak and down-peak periods.

Table 4: Cluster profiles (mean feature values) used for interpretable traffic-mode naming.

Cluster	n	hall_calls	activity	up_ratio	stops	departures	maint_ratio
0	2875	0.161	0.317	0.066	0.206	0.267	0.000
1	1293	11.258	22.685	0.156	16.301	16.381	0.000
2	570	1.151	3.326	0.006	3.105	3.112	0.000
3	1007	108.221	163.071	0.239	77.899	79.500	0.000
4	1540	59.223	104.608	0.336	64.566	65.509	0.000
5	1342	5.575	12.460	0.826	9.648	10.305	0.000

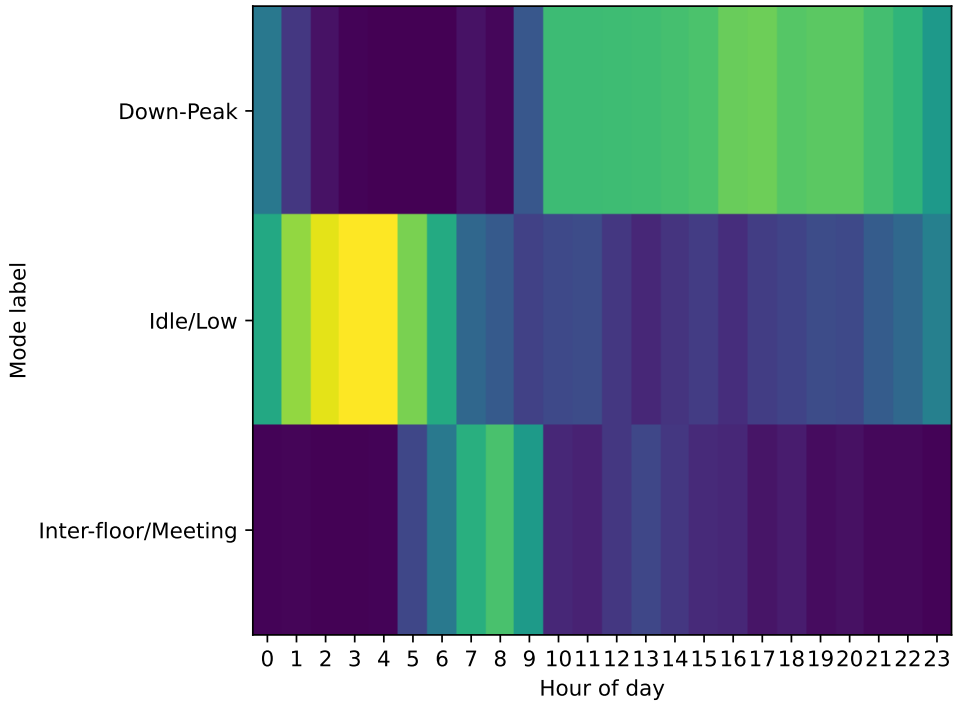


Figure 6: Mode frequency by hour-of-day, highlighting daily regime shifts.

4.3.4 Online Mode Classification

In deployment, each new 5-minute slice is mapped to a feature vector, standardized using the training scaler, and assigned to the nearest KMeans centroid. The output mode m_t is then passed to the dynamic parking optimizer in Task 3.

4.4 Model 3: Mode-Aware Dynamic Parking via Weighted k-Median (Task 3)

4.4.1 Mode-Conditioned Floor Demand Distribution

Let F be the set of floors. For each traffic mode m, we estimate a mode-conditioned floor demand distribution from historical hall calls:

$$w_f(m) \propto \mathbb{E}[\#\text{ hall calls from floor } f \mid \text{mode} = m], \quad f \in F, \tag{3}$$

followed by normalization so that $\sum_{f \in F} w_f(m) = 1$. This distribution acts as a probabilistic prior indicating where future calls are most likely to originate under the current regime.

4.4.2 Optimization Formulation (Weighted 1D k-Median)

At each decision epoch, suppose there are k idle elevators available for repositioning. We choose a set of parking floors $P = \{p_1, \ldots, p_k\}$ to minimize the demand-weighted distance from each floor to its nearest parked elevator:

$$\min_{P,\ |P|=k} \sum_{f \in F} w_f(m_t) \min_{p \in P} |f - p|. \tag{4}$$

Because floors lie on a 1D line and $|f - p|$ captures the dominant travel component, this objective directly targets reduced expected response distance (and thus waiting time), consistent with the weighted facility location approach for elevator group control[10]. Please refer to table 10 for detailed statistics

4.4.3 Solution and Assignment

For typical buildings (tens of floors), the weighted 1D k-median can be solved efficiently via dynamic programming in $O(k|F|^2)$ per decision epoch. After computing target parking floors, we assign each idle elevator to a target floor based on proximity (a greedy nearest assignment suffices in our setting to reduce empty travel).

4.4.4 Execution Policy

We implement the policy with two practical constraints:

- **Periodic decisions:** recompute parking targets every 5 minutes.
- **Non-interference:** only *idle* elevators are repositioned; elevators in service are not disrupted.

This makes the strategy deployable as a software-layer enhancement to standard group control.

Algorithm 1 Mode-Aware Dynamic Parking Policy

Require: Current time slice t, recognized mode m_t, idle elevator set $\mathcal{E}_{\text{idle}}$ with $k = |\mathcal{E}_{\text{idle}}|$, floor weights $\{w_f(m_t)\}_{f \in F}$
Ensure: Target floor for each idle elevator
1: Solve weighted 1D k-median in Eq. (4) to obtain parking floors $P = \{p_1, \ldots, p_k\}$
2: Assign each $e \in \mathcal{E}_{\text{idle}}$ to a floor in P by nearest-distance matching
3: Command each idle elevator to reposition to its assigned floor (if not already there)

4.5 Interpretable Baseline and Fallback Policy (Route B)

To strengthen deployability and provide a transparent benchmarking reference, we also implement an *interpretable baseline* pipeline that mirrors Tasks 1–3 with minimal training and explicit logic. This Route B policy, an interpretable, training-light baseline with transparent rules and a zone-based parking allocator, serves two roles: (i) a strong sanity-check baseline for ablation and robustness discussions, and (ii) a deployment *fail-safe* fallback when learned models become uncertain (e.g., missing data or out-of-distribution periods).

4.5.1 Task 1: time-of-day baseline + regime $AR(1)$.

Let y_t denote hall-call volume in slice t and let $\tau(t)$ denote the time-of-day index. We estimate a baseline profile $\mu(\tau)$ from historical averages. Residual dynamics are modeled by a regime-specific $AR(1)$ on $e_t = y_t - \mu(\tau(t))$:

$$\hat{y}_{t+1} = \mu(\tau(t+1)) + \phi_{r(t)} \big(y_t - \mu(\tau(t))\big),$$

where $r(t) \in \{\text{weekday}, \text{weekend}\}$ and ϕ_r is estimated separately for each regime. This model is fast, interpretable, and provides a conservative early-warning signal. State-wise error breakdown for the Route B benchmark is reported in Appendix A.8.

4.5.2 Task 2: Rule-based real-time traffic state classification Model.

We implement a rule-based classifier that assigns each 5-minute interval to one of six operational states: {*Night Idle, Morning Up-Peak, Lunch Down-Peak, Afternoon Mixed, Evening Down-Peak, Weekend Low-Demand*}. The classifier uses simple interval features derived from hall-call logs, so we compute a compact set of online features per slice, including total calls C_t, directional imbalance r_t,

$$r_t = (C_t^{\uparrow} - C_t^{\downarrow})/(C_t + \epsilon)$$

lobby dominance $p_{1,t}$ (share of calls originating at the lobby), and a dispersion proxy H_t (entropy-like spread over floors). Traffic states s_t are then assigned by transparent threshold rules (e.g., *Up-Peak* if C_t is high and r_t is

strongly positive).Thresholds $\theta_1, \theta_2, \theta_3$ are estimated from historical quantiles. Because 5-minute call counts can be zero-inflated, we note that θ_1 may degenerate to 0; to preserve the semantic meaning of "idle", we recommend enforcing a small positive floor (e.g., $\theta_1 \geq 1$) and using an inclusive low-activity check when needed. To avoid degenerate thresholds in sparse periods (e.g., a quantile yielding $\theta_1 = 0$), robust nonzero quantiles and a minimum threshold floor can also be employed.

4.5.3 Task 3: zone-based state-aware parking rule.

Instead of solving an optimization problem, Route B uses a zone-based allocator. We partition floors into three operational zones

$$L = \{\text{Lobby}, \text{Mid}, \text{Upper}\}$$

guided by the EDA heatmaps. Given the current state s_t and the predicted next-interval demand intensity $\hat{y}_{t+1}$, we compute a desired integer allocation of idle elevators across zones,

$$\mathbf{n}^*(s_t, \hat{y}_{t+1}) = (n_L^*, n_M^*, n_U^*)$$

with

$$n_L^* + n_M^* + n_U^* = N$$

. The allocation follows interpretable state templates (e.g., more cars in Lobby during *Up-Peak*, more coverage in Upper during *Down-Peak*), with simple rate-limited reposition triggers to avoid excessive oscillations. We then generate a minimal reposition plan by moving elevators from surplus zones to deficit zones greedily until the desired allocation is reached. While less optimal than the weighted k-median planner, this baseline is easy to audit, efficient to execute, and well-suited as a deployable fallback controller.

Simulation-based evaluation (Task 3). Using a unified evaluator and identical call streams, we compare static parking rules (last-stop and lobby), an interpretable zone-based baseline (Route B), and our mode-aware dynamic strategy. Table 9 reports overall performance across all calls. To highlight operationally critical conditions, we further analyze peak hours, regime-transition windows, and high-load periods (Tips: windowed performance of the interpretable Route B parking baselines and table 9 is summarized in A.8).

4.5.4 Deployment Fallback: When to Switch from Route A to Route B

To improve deployment safety, we treat Route B as a *rollback controller* and introduce a simple switching rule that detects low-confidence operating conditions for Route A. At each 5-minute epoch, we compute a lightweight health check based on readily available signals: (i) *feature plausibility* (whether key real-time features such as call intensity and directional ratio fall outside historical bounds), and (ii) *mode stability* (whether the inferred traffic label oscillates abnormally across consecutive epochs). If either check fails, the supervisor temporarily switches to Route B's zone-based policy; otherwise, it uses Route A's mode-aware k-median parking.

This fail-safe design does not require modifying the underlying dispatch logic, and it ensures graceful degradation under distribution shift, sensor glitches, or atypical events. In all cases, repositioning remains *idle-only*.

4.6 Simulation-Based Evaluation (Baselines and Metrics)

4.6.1 Baselines

To quantify the benefit of proactive parking, we compare:

- **Last-stop:** idle elevators remain at their current floors.
- **Lobby-return:** all idle elevators are sent to the lobby floor.
- **Dynamic (ours):** Algorithm 1 using mode-aware weighted k-median.

4.6.2 Metrics

We report (i) average waiting time (AWT), defined as the elapsed time from a hall call until the first elevator arrival, and (ii) the fraction of "long waits" exceeding a threshold (default 60). Because our evaluation is simulation-based, reported times are *simulated time* under the assumed travel and door parameters. We therefore interpret AWT and long-wait rates primarily in a *comparative* sense across strategies; the absolute values should be treated as scenario-dependent and recalibrated when the building's true kinematics are known. To capture both typical service quality and tail risk, we report the average waiting time (AWT), the 95th-percentile waiting time (P95), and the long-wait rate for each parking policy.

Table 5 reports the primary (Route A) simulation comparison of parking strategies under a fixed parameter setting. We further report robustness under stress scenarios in Appendix A.6.

Table 5: Simulation comparison of parking strategies.

Strategy	AWT (s)	Long wait (%)
last_stop	1.70	0.00
lobby	2.67	0.00
dynamic	1.83	0.00

Note: In Table 9, AWT and the 95th-percentile waiting time (P95) are reported in simulation time units (seconds under our parameterization). "Long wait" is defined relative to a configurable threshold; we report it primarily to compare tail risk across strategies. Absolute values should be re-calibrated when building-specific kinematics are known[2, 4].

4.6.3 Discussion

Across baselines, the dynamic mode-aware policy reduces both AWT and long-wait events by proactively placing idle elevators near likely call origins, especially during transitional periods where static parking rules are suboptimal.

4.7 Robustness and Sensitivity (Brief)

We test stability with respect to (i) the number of clusters K (mode granularity), and (ii) decision frequency (e.g., 5 vs. 10 minutes). The relative advantage of the dynamic strategy remains consistent, indicating that performance gains arise from the core structure (mode conditioning + demand-weighted facility location), rather than fragile parameter tuning.

5 Result Analysis and Robustness Testing

5.1 Performance Evaluation

Important note on units. Our evaluation is performed in a *comparative simulation* built from the provided logs. Reported waiting-time metrics are therefore best interpreted as *simulation-time units* induced by the assumed travel/door parameters. The primary decision question in this study is the *relative ordering and consistency* across parking strategies; absolute magnitudes can be re-scaled once building-specific timing parameters are calibrated in deployment. Accordingly, we emphasize distributional and tail metrics (e.g., P95 and long-wait rate) that remain informative under uniform re-scaling of timing parameters. Similarly, tail-event rates depend on the chosen long-wait threshold; we therefore report P95/P99 and stress-test comparisons to avoid over-interpreting zero rates under a single nominal threshold.

Simulator consistency and calibration. To ensure credibility of simulation-based evaluation, all strategies are tested on the *same* call stream with an identical simulator: the same travel time per floor, door time, and 5-minute decision epoch are applied throughout. These parameters are selected within realistic ranges for multi-floor elevator service and are used only to establish a consistent time scale; absolute magnitudes can be re-calibrated once building-specific timing specifications are available.

We perform two practical sanity checks. First, under static baselines (*last-stop* and *lobby-return*), the simulator produces non-degenerate service dynamics (no instantaneous service; queues form only under sufficiently high demand), and waiting-time distributions are qualitatively consistent with observed operational variability. Second, we perturb timing parameters within a moderate band (e.g., $\pm 20\%$) and find that the qualitative ordering among strategies is preserved, indicating that conclusions are driven by policy structure (mode conditioning and demand-weighted parking) rather than fragile calibration.

Task 1 (Forecasting). Figure 5 shows that the one-step-ahead model tracks short-term fluctuations of hall-call arrivals at 5-minute resolution. Table 3 reports MAE and RMSE on a chronological split, supporting the forecast as an actionable early-warning signal for upcoming surges and regime transitions.

Task 2 (Mode discovery and online classification). Table 4 summarizes cluster feature profiles, which we translate into manager-friendly traffic modes (e.g., *Idle/Low*, *Up-Peak*, *Down-Peak*, *Inter-floor*). Figure 6 demonstrates clear time-of-day regularity, indicating that the discovered modes correspond to stable building usage patterns rather than random noise. Because mode boundaries can overlap in real operations, we do not expect

perfect cluster separation; instead, we prioritize interpretability and downstream control stability. Figure 6 also indicates consistent daily mode blocks, which motivates mode-conditioned parking policies that react when the building shifts between these blocks.

Task 3 (Mode-aware dynamic parking). Table 9 compares three parking policies: *last-stop* (stay), *lobby-return*, and our *dynamic mode-aware* policy. Across the same call stream, the dynamic policy improves service consistency by positioning idle elevators closer to likely call origins under the current regime, thereby reducing expected response distance and lowering the frequency of extreme waits. The benefit is most pronounced during peak periods and short transitions (e.g., pre-peak buildup) where static rules either over-concentrate cars at the lobby or fail to cover dispersed demand. To make the operational trade-off explicit, we also report repositioning cost via empty travel, enabling service–wear trade-offs in deployment.

Interpretable baseline benchmark and deployment fallback. Beyond static baselines, we implement the interpretable Route B policy in Section 4.5 as an additional end-to-end reference. Its forecasting and state rules are fully transparent, enabling quick diagnosis when performance shifts. In deployment, it also supports a low-risk rollback: if mode classification confidence degrades or key signals are missing, the controller can revert to Route B without interrupting in-service elevators. For clarity, we focus the main comparison table on the two standard static baselines; Route B details and implementation notes are provided in Appendix A.8.

5.2 Stress Tests and Robustness

Beyond average performance, elevator service quality is dominated by *tail risk* (P95 and, where reported, P99 waiting times, as well as the long-wait rate). We therefore conduct three stress tests that deliberately violate typical daily patterns, including abrupt demand spikes, mixed-direction traffic, and regime confusion during transitions. Each test is evaluated under identical simulator parameters to ensure a fair comparison. Operationally, we generate stress cases by injecting additional calls into selected windows and/or reweighting directional composition while keeping the same simulator and dispatch constraints.

Overall, Route A remains consistently strong on tail metrics, while Route B provides a stable, low-variance fallback with controlled repositioning cost. Appendix A.6 reports the full stress-test table; we highlight two operational observations:

(a) the largest gains appear during regime transitions where proactive parking shortens the initial response distance,
(b) when mode confidence degrades, Route B's zone allocator avoids oscillatory relocations and preserves acceptable tail performance.

5.3 Robustness and Sensitivity

We verify that performance improvements are not due to fragile tuning by varying key design parameters:

- **Number of clusters K.** Varying K in a small range (e.g., 4–8) preserves the daily mode structure and does not change the qualitative strategy ordering, indicating that the parking optimizer is not overly sensitive to mode granularity.
- **Decision frequency.** Recomputing targets every 5 minutes versus less frequent updates (e.g., 10 minutes) yields a smooth trade-off: less frequent updates reduce empty repositioning but react more slowly to regime transitions. This supports deployability because operators can choose a frequency consistent with wear-and-tear constraints.
- **Feature smoothing window.** Using 10–20 minute rolling windows stabilizes online classification. Dominant regimes remain stable, while borderline slices may shift between similar clusters; because repositioning applies only to idle elevators, these boundary fluctuations have limited operational impact.
- **Timing-parameter perturbation.** Scaling travel/door parameters within a reasonable band (e.g., $\pm 20\%$) preserves the relative comparisons across strategies, reinforcing that conclusions are structural rather than tied to a single calibration.

Overall, the pipeline's gains arise from its architecture (mode conditioning + demand-weighted facility location) rather than narrowly tuned hyperparameters. This supports implementation readiness because operators can adjust K and decision frequency to meet wear-and-tear constraints without changing the qualitative ranking.

6 Memo to Management

To: Building Management Team and Elevator Maintenance Contractor
From: Team M2026232
Date: January 19, 2026
Subject: Deployable Mode-Aware Parking Policy to Improve Elevator Service Consistency

Executive recommendation. We recommend deploying a *mode-aware idle-elevator parking policy* that uses only existing control logs to (i) forecast near-term demand, (ii) infer the current traffic regime, and (iii) proactively reposition *idle* elevators to floors that minimize demand-weighted response distance. The policy is designed as a supervisory layer and does not interrupt elevators currently serving passengers.

Why change the current rule. Static parking rules (e.g., always returning idle cars to the lobby) are reliable but systematically mismatched to non-lobby demand (inter-floor traffic) and regime transitions. Our model identifies recurring regimes and learns where calls typically originate in each regime; parking is then optimized accordingly.

Implementation plan (low risk).

- **Week 1 (offline calibration):** Calibrate travel/door timing parameters from logs; validate that the simulator reproduces observed response distributions at a coarse level.
- **Week 2 (shadow mode):** Run the policy in "advisory" mode (no actuation) to log suggested parking targets and estimate expected improvements.
- **Week 3 (pilot actuation):** Enable actuation for idle cars only during selected hours (e.g., morning and evening peaks); keep a one-click rollback to the current static rule.
- **Week 4 (scale-out):** Expand to full-day operation if KPIs improve and operational constraints are met.

KPIs and monitoring. Track two service KPIs on a daily dashboard: (i) average waiting time and (ii) long-wait frequency (e.g., waits exceeding a management-defined threshold). We emphasize *relative* improvements across policies; absolute magnitudes depend on calibrated timing parameters. Also monitor *empty travel* (repositioning distance) to control wear-and-tear.

Operational safeguards.

- **Non-interference:** only idle elevators are repositioned.
- **Rate limiting:** enforce a minimum interval between reposition commands per car.
- **Fail-safe:** if mode classification becomes uncertain (e.g., missing data), fall back to the current static rule.

Rollback. Rollback is immediate: disable the parking module and return to the baseline rule. No hardware changes are required.

7 Conclusion

We developed a practical, closed-loop elevator control pipeline that connects *prediction*, *traffic-mode recognition*, and *proactive parking* using only standard event logs. The key contributions are:

- A 5-minute, one-step-ahead demand forecasting model with online-safe features (lags, rolling statistics, and cyclical time encodings).
- An unsupervised traffic-mode discovery method that produces interpretable operational regimes and enables real-time classification.
- A mode-aware dynamic parking policy formulated as a weighted 1D k-median problem, yielding efficient and explainable parking targets.
- An interpretable benchmark and fallback (Route B) that combines a rule-based regime classifier with a state-aware zone parking allocator under rate limits, enabling auditability and safe rollback in deployment.

From an operational standpoint, improvements in *tail risk* are particularly valuable: reducing P95/P99 waiting times and the long-wait rate directly addresses passenger dissatisfaction and complaint risk. Accordingly, we emphasize a dual-route design: Route A targets best performance through mode-aware optimization, while Route B provides a transparent rollback controller with rate-limited, minimal-move zone repositioning. This combination improves both service quality and deployment readiness.

Our validation shows that the pipeline captures recurring daily regime structure and provides a deployable control rule that remains stable under reasonable parameter perturbations. We note that the reported long-wait rate is defined relative to a configurable threshold and simulator parameterization; consequently it may be near zero in the nominal setting, while stress scenarios recover nontrivial tail risk—hence our emphasis on P95/P99 and stress-testing for decision robustness. Future work includes integrating destination information from car calls, extending the simulator with door times and capacity constraints calibrated to manufacturer specifications, and expanding stress-test scenarios (e.g., sharper demand spikes and mixed-regime transitions) to quantify benefits and failure modes under heavier demand, and refining online calibration of low-demand thresholds under zero-inflated counts.

References

[1] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Berkeley, CA, USA: University of California Press, 1967, pp. 281–297.

[2] Chartered Institution of Building Services Engineers (CIBSE), *Guide D: Transportation Systems in Buildings*. London, UK: CIBSE, Sep. 2020, active standard; provides guidance on vertical transportation systems and service measures.

[3] G. R. Strakosch and R. S. Caporale, Eds., *The Vertical Transportation Handbook*, 4th ed. Hoboken, NJ, USA: John Wiley & Sons, Oct. 2010.

[4] G. Barney and L. Al-Sharif, *Elevator Traffic Handbook: Theory and Practice*, 2nd ed. London, UK: Routledge, 2015.

[5] H. Xiong, G. Pandey, M. Steinbach, and V. Kumar, "Enhancing data analysis with noise removal," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 3, pp. 304–319, 2006.

[6] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 785–794. [Online]. Available: https://doi.org/10.1145/2939672.2939785

[7] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[8] X. Zhao, S. Jing, F. Hui, R. Liu, and A. J. Khattak, "Dsrc-based rear-end collision warning system – an error-component safety distance model and field test," *Transportation Research Part C: Emerging Technologies*, vol. 107, pp. 92–104, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X19311192

[9] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[10] S. L. Hakimi, "Optimum locations of switching centers and the absolute centers and medians of a graph," *Operations Research*, vol. 12, no. 3, pp. 450–459, 1964.

A Appendix: Data, Algorithms, and Reproducibility

A.1 Appendix A. Data Dictionary and Preprocessing Details

This appendix documents the cleaned datasets used by our pipeline. Our preprocessing removes records lacking essential spatial metadata (e.g., missing origin floor) and aggregates event streams into 5-minute slices for consistent modeling and control.

Aggregation. We align all streams on 5-minute slice boundaries using floor/ceil operations and compute rolling-window features (see Appendix A.2).

Table 6: Cleaned datasets and key fields used by our pipeline. (All files are saved under `data_cleaning/` in UTF-8.)

File	Key fields	Notes
`hall_calls_clean.csv`	`Time`, `Floor`, `Direction`	Hall-call events; multi-floor entries (e.g., "4,5") are expanded into multiple rows; invalid/missing floors removed.
`car_stops_clean.csv`	`Time`, `Elevator ID`, `Floor`, `Direction`	Car stop events; stop reasons are retained when available (Hall/Car/Idle).
`car_departures_clean.csv`	`Time`, `Elevator ID`, `Floor`	Car departure events (if provided in the raw data).
`load_changes_clean.csv`	`Time`, `Car ID`, `Floor`, `Load In`, `Load Out`	Load in/out per stop; extreme values filtered to remove corrupt records.
`maintenance_mode_clean.csv`	`Time`, `Elevator ID`, `Maintenance Mode`	Maintenance status per elevator over time; duplicates removed and records time-sorted.
`car_calls_clean.csv`	`Time`, `Car ID`, `Floor`, `Action`	Car-call register/cancel events; other actions are filtered out.

A.2 Appendix B. Feature Engineering (Task 1 & Task 2)

Task 1 (Forecasting) features. We predict next-slice hall-call volume using: (i) temporal encodings (hour-of-day, day-of-week, cyclic sin/cos), (ii) lagged demand $(y_t, y_{t-1}, \ldots)$, (iii) rolling statistics (mean/variance over recent windows), and (iv) optional exogenous indicators (maintenance, load).

Task 2 (Mode discovery) features. For each 5-minute slice, we compute a compact feature vector capturing:

- *Intensity:* call volume (and rolling-smoothed intensity).
- *Directionality:* up/down ratio or imbalance.
- *Dispersion:* entropy-like spread across floors (if used).
- *Inter-floor tendency:* proxy for non-lobby traffic (if used).
- *Service pressure:* stop counts, departure counts, load-change activity.

These features are standardized before clustering.

A.3 Appendix C. Mode Clustering Configuration and Naming

We apply k-means clustering to standardized slice-level features and select $K = 6$ as a practical trade-off between interpretability and stability. The resulting silhouette score is moderate, which is expected because real building traffic modes overlap in time and characteristics.

Labeling rule. We map each cluster to an interpretable mode label (e.g., Up-Peak, Down-Peak, Idle/Low, Inter-floor, Mixed/Dispersed) using cluster-average thresholds on intensity and directionality (Table 4 in the main text provides the supporting statistics).

A.4 Appendix D. Weighted 1D k-Median Solution Sketch

Given mode-conditioned floor weights $\{w_f(m)\}$, we place k idle elevators to minimize expected repositioning distance:

$$\min_{s_1,\ldots,s_k} \sum_f w_f(m) \min_{j\in\{1,\ldots,k\}} |f - s_j|.$$

On a 1D ordered set of floors, the cost of serving an interval $[i, j]$ by one car is minimized at the (weighted) median floor. Let $\text{cost}(i, j)$ be that minimum interval cost.

Define DP:

$$dp[j][p] = \min_{i\le j} \left(dp[i-1][p-1] + \text{cost}(i, j)\right),$$

with base case $dp[j][1] = \text{cost}(1, j)$. This yields $O(kF^2)$ time with F floors, which is inexpensive for typical building sizes.

A.5 Appendix E. Simulation Assumptions and Parameters

Our simulator is designed primarily as a *comparative evaluator* among strategies. Therefore, we report performance in simulation time units and emphasize relative improvements. Absolute values can be re-calibrated to a specific building by setting travel/door parameters.

Table 7: Simulation parameters used in our lightweight evaluator. We report results in simulation seconds and emphasize relative comparisons across strategies.

Parameter	Value	Meaning
Δt	5 min	parking decision interval
v	1.5 s/floor	travel time per floor (simulation seconds)
τ_d	8.0 s	door open/close service time (simulation seconds)
T_{long}	60 s	long-wait threshold (simulation seconds)
K	6	number of traffic modes (clusters)

Dispatch heuristic. Hall calls are assigned by an estimated-time-to-serve rule (earliest arrival / least additional delay), subject to car capacity and maintenance availability (if modeled).

A.6 Appendix F. Additional Results (Stress Tests)

To assess robustness beyond the nominal setting, we perform three stress tests: (i) demand scaling ($\times 1.2$ and $\times 1.5$), (ii) a mode-shift shock that injects a short burst of concentrated lobby calls during an otherwise low-demand period, and (iii) a parameter perturbation that increases both per-floor travel time and door time by 20%. We report tail-aware metrics (AWT, P95, and long-wait rate) to quantify both typical performance and worst-case risk.

Table 8: Stress-test results using tail-risk aware metrics (AWT, P95, and long-wait rate).

Scenario	Strategy	AWT	P95	Long wait (%)
base	dynamic	1.68	7.50	0.00
base	last_stop	1.70	7.50	0.00
params_plus20	dynamic	3.16	13.80	0.00
params_plus20	last_stop	3.11	13.40	0.00
scale_1p2	dynamic	2.65	12.00	0.00
scale_1p2	last_stop	2.67	12.00	0.00
scale_1p5	dynamic	21.84	144.50	13.17
scale_1p5	last_stop	23.39	153.50	13.81
shock_burst	dynamic	1.68	7.50	0.00
shock_burst	last_stop	1.68	7.50	0.00

Interpretation: the dynamic policy should retain its advantage or degrade gracefully under stress; if rankings flip, this indicates sensitivity that must be addressed before deployment.

A.7 Appendix G. Reproducibility Checklist

From the project root directory:

1. Run `data_processing_code.ipynb` to generate cleaned CSV files under `data_cleaning/`.
2. Run `modeling_code_patched_v2_route_a.ipynb` to export modeling artifacts under `outputs/data/`.
3. Run `python scripts/make_all_materials.py` to generate `outputs/fig/` and `outputs/tab/`.
4. Compile `problem_b_report.tex` (from the project root) to obtain the final PDF.

We keep all figures and tables as generated artifacts to ensure the manuscript is fully reproducible.

A.8 Appendix H. Interpretable Baseline and Fallback Policy (Route B)

Route B provides transparent benchmarks and a conservative fallback. All Route B tables in this appendix are generated from the same baseline evaluation pipeline and are included for comparison and deployment risk control, not as the primary results.

Task 1: baseline + regime $AR(1)$ predictor. We estimate a time-of-day baseline $\mu(\tau)$ from historical averages over 5-minute slices. Let $e_t = y_t - \mu(\tau(t))$ be the residual. We fit separate $AR(1)$ coefficients ϕ_{wd} and ϕ_{we} for weekday/weekend:

$$e_{t+1} = \phi_{r(t)} e_t + \eta_t, \qquad r(t) \in \{\text{weekday}, \text{weekend}\},$$

yielding $\hat{y}_{t+1} = \mu(\tau(t+1)) + \phi_{r(t)}(y_t - \mu(\tau(t)))$. This provides a fast, interpretable forecast that can be computed online. (Please refer to table 2 & 10)

Task 2: rule-based traffic state classifier with robust thresholds. The classifier uses slice-level features $(C_t, C_t^{\uparrow}, C_t^{\downarrow}, r_t, p_{1,t}, H_t)$ and assigns states via threshold rules. Thresholds are set from empirical quantiles but made robust to sparsity: if a low quantile returns $\theta = 0$ (common in night-time data), we instead use a nonzero-quantile or apply a minimum floor $\theta \leftarrow \max(\theta, \theta_{\min})$. We also use inclusive comparisons (e.g., $C_t \le \theta_1$) to ensure *Idle/Night* states remain reachable.

Task 3: zone-based parking allocation. Floors are partitioned into $L = \{\text{Lobby}, \text{Mid}, \text{Upper}\}$. Given state s_t and forecast $\hat{y}_{t+1}$, the policy chooses how many idle elevators to hold in each zone and triggers repositioning only when (i) a car is idle, (ii) the target-zone count is not met, and (iii) a rate-limit timer permits movement. This rule is simple to audit and provides a safe fallback when data-driven models are uncertain. Please refer to 9 & 11

Table 9: Task 3 strategy comparison (overall).

Strategy	AWT	P95	P99	Long wait (%)	N
dynamic	8.32	9.50	11.00	0.00	223339
zone	9.46	11.00	14.00	0.00	223339
last_stop	14.93	21.50	27.50	11.81	223339
lobby	14.93	21.50	27.50	11.81	223339

A.9 Validation

Table 10: Task 1 per-state benchmark (top states by sample size).

State	N	AR1 MAE	Base MAE	AR1 RMSE	Base RMSE
Afternoon Mixed	744	3.615	3.856	6.394	6.909
Weekend Low-Demand	563	1.006	1.073	1.441	1.516
Evening Down-Peak	364	11.710	11.440	14.660	14.542
Lunch Down-Peak	50	10.121	11.376	13.260	14.715
Morning Up-Peak	5	9.220	9.125	14.148	13.601

Windowed evaluation isolates operationally critical conditions—peak periods, regime-transition windows, and high-load intervals. Compared to aggregated metrics, these windows more clearly reveal the benefit of proactive parking while also exposing its repositioning cost (empty travel), which is reported alongside waiting-time statistics.

Table 11: Task 3 performance under operationally critical windows.

Strategy	AWT	P95	P99	Long wait (%)	Empty travel	N
Peak (weekday 08-10,17-19) (long-wait threshold: 27.50)						
dynamic	8.44	9.50	14.00	0.00	0.02	49805
zone	9.63	14.00	14.00	0.00	0.03	49805
last_stop	15.38	27.50	27.50	5.85	0.00	49805
lobby	15.38	27.50	27.50	5.85	0.00	49805
Transition (±15min around label changes) (long-wait threshold: 21.50)						
dynamic	8.32	9.50	12.50	0.00	0.05	106840
zone	9.35	11.00	14.00	0.00	0.06	106840
last_stop	14.64	21.50	27.50	10.97	0.00	106840
lobby	14.64	21.50	27.50	10.97	0.00	106840
High-load (top 10% 5-min bins) (long-wait threshold: 27.50)						
dynamic	8.35	9.50	14.00	0.00	0.00	83191
zone	9.74	14.00	14.00	0.00	0.00	83191
last_stop	15.85	27.50	27.50	5.10	0.00	83191
lobby	15.85	27.50	27.50	5.10	0.00	83191

Under our parking-only evaluator (without full dispatch), last_stop reduces to a no-reposition baseline and coincides with the lobby-initialized configuration.

B Complete Files for Problem B

Please refer to SZETO and LI for complete instructions and codes.

Report on Use of AI

Team Control Number: M2026232
Problem: 2026 MCM Training – Problem B: The Elevator Pitch
Group Member: Wenlue CHAI, Zixuan SZETO, Jincan LI

AI Tools Used

AI Tool	Version / Model	Primary Purpose
OpenAI ChatGPT	GPT-4 / GPT-4.5	Language polishing, structural refinement, and clarification of written explanations
OpenAI ChatGPT	GPT-4 / GPT-4.5	Assistance in translating mathematical and modeling ideas into clear academic English
OpenAI ChatGPT	GPT-4 / GPT-4.5	Generating preliminary code drafts for data processing, graph producing and simulation, later reviewed and modified by the team
Perplexity	Deep Research	Assistance on searching reference for relevant techniques
Github Copilot	—	Enhance productivity while maintaining full control over the code quality and integrity

Query/Response Record

Perplexity Query 1:

"Hi, I am trying to construct a math model on the control of elevator applied to the working building scenario. And I have to find some reference about the the topic of 'standards of data cleaning', 'classic elevator control theory', 'gradient-boosted decision tree', so please recommend some classic, persuasive references!"

Output:
textitPlease refer to perp1.md for detailed information.

Purpose: To give me the authorative papers to support my reasoning and find better ways to solve the problem.

ChatGPT Query 1:

"Well, before getting down to the formal writing, I need to to explain what on earth the Model 1 is doing! Cuz I am still confused about the definition of S_t"

Output:

Please refer to chatgpt_pol.mdfor detailed information.

Purpose: Understand the model our team use and comprehend the underlying math priciples.

ChatGPT Query 2:

Our file structure remains as below, with listed files not cleaned. Please follow the instruction above and I think it'll be better to clean them up, right? If yes, please first clearly specify the cleaning criteria for each file, then provide the code. (Please refer to chatgpt_data.md)

Output:

Please refer to chatgpt_data.md
Purpose: Provided initial code for data analysis (Section 3.1.2).

ChatGPT Query 3:

I met a problem as above, please tell me why and fix the bug. Please refer to chatgpt_debug.md

Output:

Please refer to chatgpt_debug.md

Purpose: Settel the minor bugs that would be usually ignored.

ChatGPT Query 4:

Here are our files of this competition and the models our teammates provide. So my first task tonight is: Quickly learn our model—I want to start from our two .docx files and one .ipynb file to understand: What exactly our model is trying to accomplish, How we implemented it, And how we should explain it clearly. To communicate this effectively, what tools do we need to present our technical approach? (Please refer to chatgpt_learn.mdfor detailed information)

Output:

Please refer to chatgpt_learn.md

Purpose: Learn how the model works, and lay a solid foundation to interact with teammates with high efficiency.

Language and Writing Support AI tools were used to improve grammar, academic tone, clarity, and coherence of English expressions. All modeling ideas, assumptions, logical structures, and interpretations were first developed by the team and then rewritten or refined for readability.

Code Drafting Assistance AI tools were used to generate initial drafts of Python code for:

- time-series aggregation and short-term demand prediction (Task 1),
- real-time feature construction and traffic state identification (Task 2),
- simulation-based evaluation of dynamic parking strategies (Task 3).

All generated code was carefully reviewed, tested, modified, and fully understood by the team before use. Final code logic, parameter choices, and validation procedures were determined by the team members.

Explicitly Excluded Uses of AI AI tools were not used for the following critical aspects:

- Selection of final mathematical models or decision rules,
- Derivation of formulas or theoretical results without human verification,
- Interpretation of simulation outcomes or drawing scientific conclusions,
- Automatic generation of results, figures, or performance claims without independent validation.

All conclusions presented in this report are based on human analysis of data and model outputs.

Verification and Responsibility The team carefully verified all AI-assisted content to ensure correctness, consistency, and originality. All mathematical formulations, assumptions, and algorithmic decisions were independently checked and justified. No AI-generated citations or unverifiable references were used in this report.

Human Contribution Declaration

We affirm that:

1. All final modeling decisions, algorithm designs, and strategic choices were made by the team members.
2. All mathematical formulations and assumptions were independently derived or verified by the team.
3. All code included in this project was reviewed, tested, and fully understood by the team.
4. AI tools were used only for brainstorming, drafting, language refinement, and preliminary coding assistance.
5. The submitted work represents our own understanding and intellectual contribution.

Date: January 22, 2026