Confidential Internship Report

Data-Driven Forecasting and Automation in Amazon's European Transportation Network

Zakaria Akil

Master's Degree in Data Science - M2DS
zakaria.akil.2002@gmail.com

from 07/04/2025 to 03/10/2025

Amazon EU
38 avenue John F. Kennedy - L-1855 - Luxembourg
Internship-Agreements-RS@amazon.com
aboutamazon.com

School Tutor:
Olivier Fercoq
olivier.fercoq@telecom-paris.fr

Examiner:
Olivier Fercoq
olivier.fercoq@telecom-paris.fr

Company Tutor:
Yueqing Jia
yqjia@amazon.com

Acknowledgments

I would like to express my deepest gratitude to my manager, *Raj Linganna*, whose trust, steady guidance, and thoughtful feedback shaped every stage of this internship. A special thank you goes to my mentor, *Yueqing Jia*, who supervised my projects and consistently supported my personal development and career growth. Her ability to challenge my thinking, encourage independence, and make time for honest and constructive discussions helped me gain confidence and overcome obstacles throughout this experience.

I am also sincerely grateful to *Ambre Bourdier*, *Paulina Daciuk*, *Peter Keszthelyi*, *Chetan Prakash*, *Carlos Gonzalez*, and *Jorge Soldevilla Artajona* for their constant assistance and for so generously sharing their expertise.

My appreciation extends to *Télécom Paris* and the *M2DS Master's program* at *École Polytechnique* for providing the rigorous training that prepared me for this experience. In particular, I am grateful to *Olivier Fercoq* for serving as my academic supervisor. The program's high standards and culture of excellence have given me the theoretical foundations and scientific mindset needed to approach real-world problems with clarity and discipline.

A special thanks goes to my friend *Amon Gieseke*, with whom I shared the internship in the team, and to *Imane Bennaji*, an old friend who was also an intern in Paris within the same organization. Our regular exchanges of knowledge, ideas, and encouragement were invaluable to the success of my internship.

Finally, my heartfelt thanks go to my parents, whose unwavering support and faith in me made this journey possible, and to the friends I met along the way during my studies in France. Your encouragement, patience, and presence helped me stay focused, resilient, and determined. I am profoundly grateful for everything you have done to help me reach this milestone.

Abstract

Amazon's European transportation network operates at a scale and complexity that demand continuous improvement in both analytical and operational systems. This report presents two complementary projects developed within the *Transportation Network Configuration* (TNC) team, each designed to enhance network reliability through data-driven forecasting and cloud-native automation.

The first project, *ANCHOR* (*Analysis of Network Closures and Holiday Operating Requirements*), focuses on forecasting warehouse closures across the five major European countries (FR, DE, IT, ES, and GB) to prevent configuration defects caused by holiday or unplanned shutdowns. Leveraging historical closure data, temporal patterns, and operational attributes, the system predicts short-term closure risks using probabilistic and hazard-based models. The resulting early-warning mechanism strengthens configuration validation workflows in the *Open Request Change System* (*TORCH*), enabling proactive adjustments before deployment. The proof of concept achieved $\mathbf{PR-AUC = 0.90}$ and $\mathbf{F_1 = 0.84}$ for next-day risk prediction, and $\mathbf{PR-AUC = 0.99}$ and $\mathbf{F_1 = 0.97}$ for the early alerting (next-week risk) task, demonstrating strong potential for operational integration.

The second project addresses the automation of metric computation, reporting, and monitoring within the TNC team's weekly performance review framework, known as *Page 0*. The manual process was replaced by a fully automated, AWS-based pipeline built with the *AWS Cloud Development Kit* (*CDK*), leveraging *AWS Lambda*, *AWS Step Functions*, *Amazon QuickSight*, and complementary services. This serverless architecture established a single source of truth, reduced reporting latency from several days to hours, and introduced accountability through the automatic creation and tracking of Release Candidate (RC) rejection tasks via *Asana* and *Slack* bot integrations.

Together, these projects illustrate a unified effort to enhance operational efficiency, data integrity, and predictive capability within Amazon's European logistics ecosystem. They demonstrate how scalable serverless automation and machine learning can jointly improve the resilience, transparency, and reliability of large-scale transportation networks.

Contents

Part I
Project 1: Analysis and Forecasting of Warehouse Closures in Amazon's European Transportation Network

1 Introduction

1.1 Context

Amazon's *Global Transportation Services* (GTS) department is responsible for designing, operating, and continuously optimizing one of the world's most complex logistics networks. Within GTS, the *Transportation Network Optimization* team focuses on improving the efficiency and reliability of the European transportation network through data-driven methods, optimization models, and automation pipelines.

During this internship, I joined the *Network Configuration* scope, an analytics and data engineering sub-team that develops automation tools, data pipelines, optimization algorithms, GenAI applications, and analytical dashboards. These products collectively enhance and streamline the transportation configuration workflow by facilitating collaboration between business stakeholders and technology teams.

The European transportation network can be abstracted as a set of *nodes*—including fulfillment centers (FC), sort centers (SC), delivery stations (DS), and third-party logistics sites (3PL)—and a set of *lanes* connecting them through different transportation legs (air, road, or sea). Each lane has associated configuration parameters such as truck capacity limits, *Critical Pull Time* (CPT), and *Critical Injection Time* (CIT). These configurations are managed through a centralized change management system called the *Transportation Open Request Change System* (*TORCH*), which supports operations such as *Add Lane*, *Remove Lane*, *CPT Change*, or *Transit Time Update*.

Configuration updates are deployed through a controlled process known as *Release Candidates* (RCS), which consolidate validated configuration changes before production deployment. Multiple RC types coexist to address different operational needs: Normal RCs handle standard daily updates, while Emergency RCs (ERCS) and Fast RCs allow expedited deployments for urgent issues such as capacity or timing adjustments. On-Demand RCs (ORCS) are used to anticipate potential configuration risks ahead of scheduled RC cutoffs. Each RC undergoes a multi-stage validation process via systems such as *CIMS* (Configuration Information Management System) and *SIMS*, with final manual approval and deployment through *ATROPS*. The integrity and accuracy of these configurations are essential to ensure smooth network operation and to uphold Amazon's delivery promises to customers.

1.2 Problem Statement

Despite this extensive configuration framework, managing network configurations during public holidays and exceptional events remains a major operational challenge. Warehouse closures—either planned (e.g., national holidays) or unplanned (e.g., strikes, infrastructure maintenance)—can disrupt transportation routes, delay shipments, and invalidate existing lane configurations. A misaligned configuration, such as a lane with a valid CPT passing through a closed node, can lead to significant service degradation and emergency reconfigurations.

In practice, *holiday-related closures are usually reported and preconfigured* in *TORCH* to ensure that affected warehouses are properly excluded or adjusted during configuration changes. However, when holidays are not accurately available ahead of configuration deadlines, or when the reconfiguration process fails to correctly account for them, defects can occur. These defects may propagate across the network, leading to incorrect lane activations, invalid timings, or even service-level incidents.

This motivates the need for a proactive forecasting mechanism. By leveraging historical closure data, it

becomes possible to detect temporal patterns and anticipate upcoming holiday closures before official entries appear in *TORCH*. Such a system could serve as an early-warning or *alarm mechanism*, automatically flagging inconsistencies between forecasted closure periods and ongoing configuration activities. In this way, forecasting not only enhances network resilience but also prevents late reconfigurations and operational disruptions.

A concrete illustration of this need occurred in December 2024, when a manual input error during the reinstatement of node LBA9 resulted in a high severity incident that endangered approximately 150,000 packages. This event underlined the importance of having an automated and predictive solution capable of identifying closure risks in advance.

1.3 Scientific Objective and Scope

This internship focuses on developing the forecasting component of **ANCHOR** — *Analysis of Network Closures and Holiday Operating Requirements*. ANCHOR aims to predict warehouse (node) closures across five major European countries (EU5: France, Germany, Italy, Spain, and the United Kingdom) using historical closure data, temporal patterns, and operational attributes. The system seeks to reduce late or missed configurations by anticipating discrepancies between forecasted closures and existing TORCH entries.

From a scientific perspective, this work falls within the domain of **time series forecasting** and **event prediction**. It leverages statistical analysis, feature engineering, and machine learning techniques to model closure probabilities over time. The project also addresses the challenges of temporal data sparsity, irregular event sequences, and country-specific seasonality related to public holidays. Ultimately, the model will help automate part of the configuration validation workflow by providing early closure predictions that can feed into TORCH validation checks and RC deployment decisions.

2 Related Work

Forecasting warehouse closures intersects several strands of research, notably calendar-aware time series forecasting, event-time modeling through hazard and survival analysis, probabilistic forecasting, and hierarchical reconciliation. This section reviews the most relevant approaches and positions the modeling choices adopted for the *ANCHOR* system.

2.1 Calendar- and Holiday-Aware Time Series Forecasting

Classical approaches to operational time series modeling emphasize the decomposition of temporal signals into trend, seasonality, and irregular components, with explicit handling of calendar effects. *Prophet* (1) is widely adopted in industry for its additive structure with yearly and weekly seasonality, piecewise linear trend, and user-defined holiday regressors. This makes it well suited for incorporating public-holiday calendars per country. For more complex seasonal patterns—such as overlapping daily, weekly, and yearly periodicities—TBATS provides an exponential-smoothing state-space formulation with trigonometric seasonality (2).

These models offer interpretable inclusion of holiday dummies and country–holiday interactions, which is relevant to closure forecasting. However, their main limitation lies in the assumption of regularly spaced signals and sufficient event density. In the *ANCHOR* context, closure events are sparse and often binary (open/closed), which constrains the expressiveness of purely calendar-based regressions.

2.2 Probabilistic Forecasting and Uncertainty Quantification

Operational decision-making, especially around configuration cutoffs, benefits from calibrated uncertainty estimates. Bayesian Structural Time Series (BSTS) (3) provides a principled state-space formulation with spike-and-slab regression for exogenous inputs and coherent posterior intervals. Deep probabilistic

forecasters such as DeepAR (4) and Temporal Fusion Transformers (TFT) (5) extend this paradigm to data-rich, multi-series regimes, learning shared temporal representations across nodes. These models generate predictive distributions that can be interpreted as "risk-aware" forecasts, estimating probabilities of closure around specific dates.

Nevertheless, such models typically require long historical series per node and consistent feature availability, which were not guaranteed in the present dataset. For this reason, they informed the conceptual design of *ANCHOR* (e.g., calibrated uncertainty outputs) but were not directly implemented.

2.3 Event-Time Forecasting: Point Processes and Hazard Models

Warehouse closures can be naturally viewed as discrete events in continuous time. Self-exciting point processes (Hawkes processes) capture temporal clustering, where one event increases the short-term propensity for subsequent ones (e.g., rolling holidays or strike waves) (6). Alternatively, survival and hazard-based models such as the Cox proportional hazards model (7) estimate the instantaneous risk of an event conditioned on covariates—such as holiday proximity, node type, or country.

This family of models aligns closely with the objectives of *ANCHOR*: predicting the probability of closure within a time horizon rather than forecasting a numeric target. In practice, hazard-based modeling proved especially effective for sparse, binary data while remaining interpretable and easy to integrate with operational systems like *TORCH*.

2.4 Hierarchical and Grouped Forecasting

The EU5 network exhibits a natural hierarchical structure—country → region → node type → warehouse. Hierarchical reconciliation methods jointly forecast across levels and enforce aggregate consistency (8; 9). This property is attractive in the closure-forecasting context: it preserves country-level holiday structure while generating warehouse-level forecasts that remain coherent across the hierarchy. Although hierarchical reconciliation was not explicitly implemented in *ANCHOR*, its principles inspired the design of the evaluation pipeline by ensuring consistency between node-level and country-level metrics.

2.5 Robustness and Concept Drift

Operational systems inevitably evolve under regime changes—policy updates, new release workflows, or external disruptions. Concept drift detection techniques (10), such as ADWIN (11), monitor changes in the underlying data distribution and trigger model retraining or recalibration when significant shifts occur. Integrating such mechanisms into forecasting pipelines helps ensure sustained model reliability in dynamic environments.

In the case of *ANCHOR*, drift detection remains a prospective extension for long-term maintenance once the model is deployed in production. The proposed infrastructure is well suited for online learning tasks, even though, in our context, environmental or business-related shifts are relatively slow—typically evolving over several years. As will be discussed later, closure distributions show slight variations across years, partly driven by business adjustments and data quality inconsistencies in historical records. Nevertheless, data streaming paradigms and adaptive drift detectors, such as adaptive sliding-window methods, represent promising directions for future scalability and real-time adaptation.

2.6 Positioning and Summary

In summary, prior work offers multiple methodological avenues for closure forecasting. Calendar-aware models provide interpretability but struggle with sparsity; deep probabilistic forecasters achieve high accuracy in dense multi-series regimes but require large, homogeneous datasets; and hazard-based formulations strike a balance between interpretability, flexibility, and suitability for sparse, event-based data.

Consequently, the *ANCHOR* system adopts a hybrid approach: hazard-based modeling to estimate closure risks, complemented by probabilistic forecasting principles to quantify uncertainty. This design aligns

with operational requirements for explainability, scalability, and integration with existing configuration workflows.

3 Datasets and Preprocessing

The development of the *ANCHOR* forecasting component relies on the integration of heterogeneous data sources: historical warehouse closure records, network metadata, and contextual information such as public holidays. This chapter presents the data foundation of the project, the procedures used to construct the working dataset, and the preprocessing steps that enabled subsequent analysis and modeling.

3.1 Dataset Overview

The analysis focuses on the European transportation network, with particular emphasis on five countries—FR, DE, ES, IT, and GB (EU5). Historical closure data are sourced from the configuration management warehouse, specifically the `cims_warehouse_closures` table, which records event-level closures over multiple years. Each entry provides the node identifier (`warehouse_id`), the closure interval (`start_date`/`start_time` to `end_date`/`end_time`), and a free-text description (`closure_description`) that occasionally specifies the reason for closure. As is typical for operational logs, the description field is inconsistently populated and not suitable as a primary label.

To enrich the training dataset, additional registry and panel tables were joined to attach static and slowly varying attributes to each node, including `gis_longitude`, `gis_latitude`, `zip_code`, `rentable_area`, and organizational fields such as `org` (country), `node_type` (FC, SC, DS, 3PL), and reporting groups. Public holiday data were incorporated from `eu_config.holidays` to capture dependencies between closure patterns and bank holidays. This dimension is central to the study: the objective is to forecast holiday-driven closures—the most systematic and predictable component of network inactivity—while acknowledging that exceptional shutdowns are inherently harder to anticipate given the available signals. The combination of these sources supports both temporal and spatial analyses of network unavailability across the EU5 region.

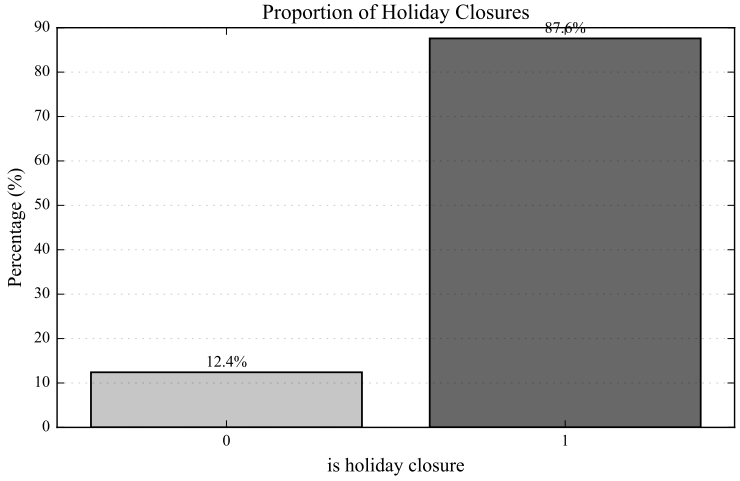


Figure 1: Share of operational closures by category in the working dataset. Bars report the proportion of events linked to public holidays versus other shutdowns derived from `cims_warehouse_closures` joined to `eu_config.holidays` (association rule described in Section 3). Figure values reflect post-processing filters (FR, DE, ES, IT, GB; active nodes; 2021–2025).

Historical Closures Dataset Construction

The core dataset is assembled from *CIMS* and *TORCH*, which log warehouse-level closure and reopening events. Records from `cims_warehouse_closures` are joined to the node registry (`cims_domain_warehouse_info`) using `warehouse_id` as the primary key and filtered to active EU5 nodes over 2021–2025. To contextualize each event, the table is joined to `eu_config.holidays` to indicate whether a closure coincides with a national or regional holiday (`is_holiday_closure`).

The extraction logic (Listing 1) computes `closure_duration_hours` directly from timestamps and applies a $\pm$1-day association window around the closure interval to account for overnight shifts and time-zone boundaries. The output is a unified closure-event table with operational fields (`warehouse_id`, interval, duration), contextual fields (`org`, `node_type`), and holiday linkage (`holiday_name`, `holiday_date`, `is_holiday_closure`). When multiple holidays fall within the window, the closest date is retained for descriptive purposes; model features use binary or count encodings and are not sensitive to the holiday label's textual form.

Preprocessing and Feature Engineering

Preprocessing enforces consistency across sources and prepares learning targets. Timestamps are parsed to a canonical time basis, with explicit handling of daylight-saving transitions. Erroneous durations (e.g., negative or missing values) are recalculated from the raw interval; duplicate or overlapping records per `warehouse_id` are consolidated to a single interval when they arise from upstream reprocessing. Free-text fields are normalized and kept only for audit.

The event table is enriched with static attributes from the registry to attach geospatial and organizational context. Feature derivation then produces the inputs required by the forecasting tasks described later: calendar encodings (day-of-week, month, `weekofyear`, `is_weekend`), proximity to holidays (e.g., `days_to_next_holiday`, `days_since_prev_holiday`), and short-horizon activity summaries (`days_since_last_closure`, rolling counts and means over 7/14/30 days). These features are computed on a regular daily grid obtained by crossing each active `warehouse_id` with the study calendar and marking days intersecting an event as closed, which allows learning on a dense target even when raw events are sparse.

As discussed in Section 3.2, geolocation fields contain occasional missing or erroneous coordinates (e.g., $(0, 0)$). Such cases are flagged and excluded from purely spatial analyses, while their operational records remain in scope for temporal modeling. Together, these steps yield a coherent, multi-granular dataset suitable for exploratory analysis and for the hazard-based forecasting models deployed in *ANCHOR*.

3.2 Exploratory Data Analysis

This section summarizes the main exploratory data analysis (EDA) insights that guided the modeling design of *ANCHOR* forecasting. The analyses focus on the spatial and temporal patterns of warehouse closures across the EU5 region and their relationships with public holidays and node types.

3.2.1 Geographical Overview of Closure Hours

Because each warehouse is geolocated, the dataset supports spatial exploration of closure intensity. Figures 2 and 3 map total closure hours per site in 2024 and 2025.

A clear spatial concentration emerges: the same subset of warehouses tends to appear repeatedly across years, indicating a structural pattern rather than random variation. However, the total volume of closure hours fluctuates over time, suggesting that while some nodes systematically close, the precise duration of their inactivity varies year to year—likely reflecting operational or contractual differences.

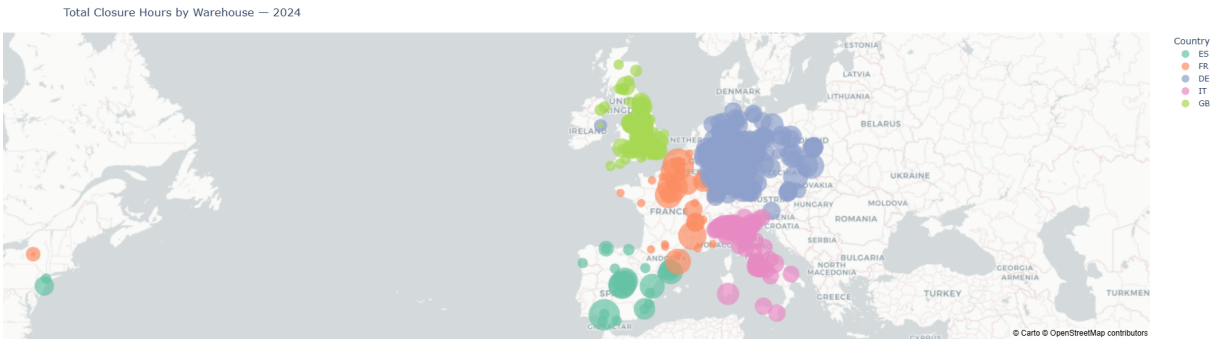


Figure 2: Geographical distribution of cumulative closure hours by warehouse in 2024. Bubble size represents total closed hours per node.

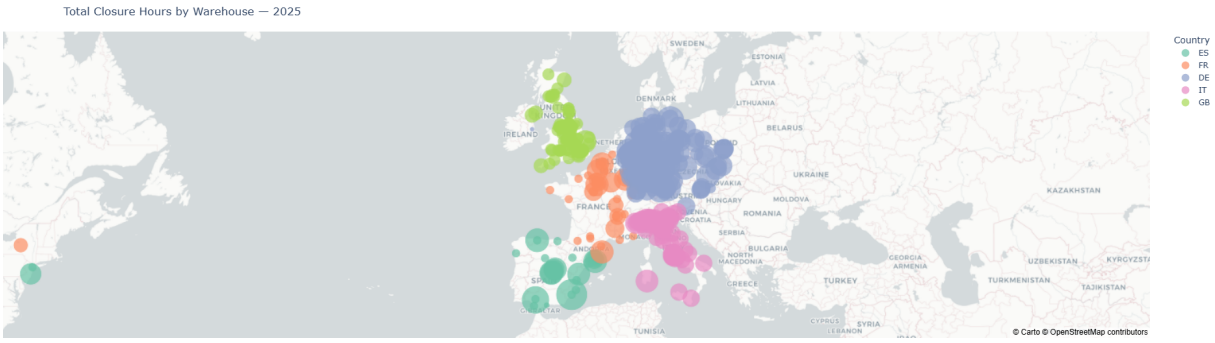


Figure 3: Geographical distribution of cumulative closure hours in 2025. Spatial patterns largely overlap with 2024, though total closure durations differ.

Closure intensity also varies across countries. Germany, for instance, displays a higher frequency of closures relative to Spain or Italy, as shown in Figure 4. These asymmetries reflect differences in national holiday calendars, local labor regulations, and operational practices, all of which influence the probability of node unavailability.

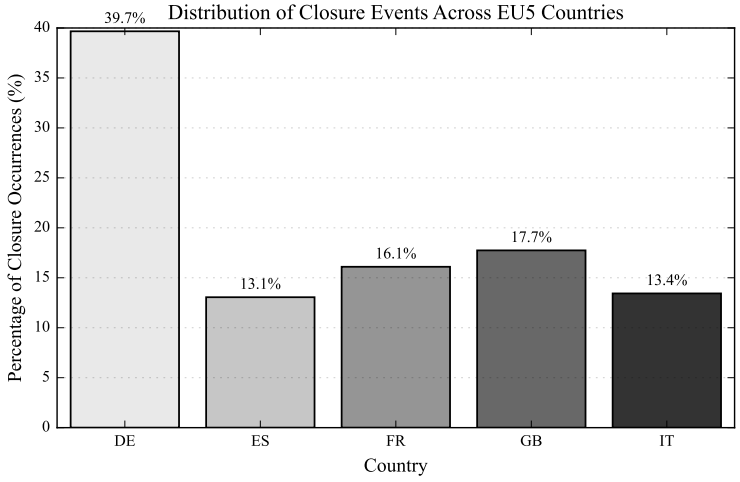


Figure 4: Share of recorded closure events by country across the EU5 region (2021–2025).

It is worth noting that the geospatial information is partially incomplete: several nodes have coordinates set to $(0, 0)$ or missing values, as illustrated in Figure 5. These artifacts stem from legacy entries in the

warehouse registry and limit the accuracy of purely spatial analyses. Affected records were retained for temporal modeling but excluded from maps and distance-based computations.

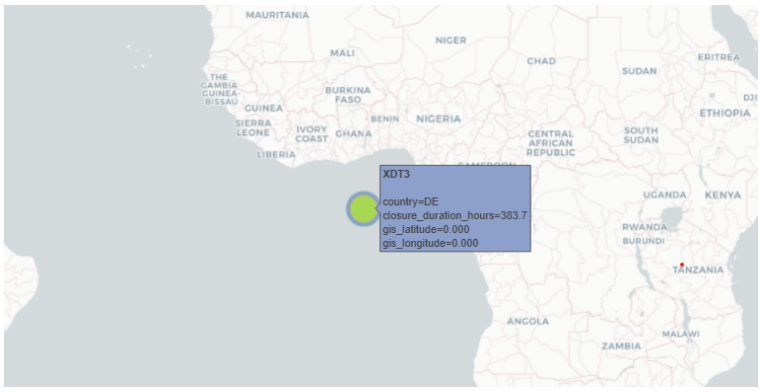
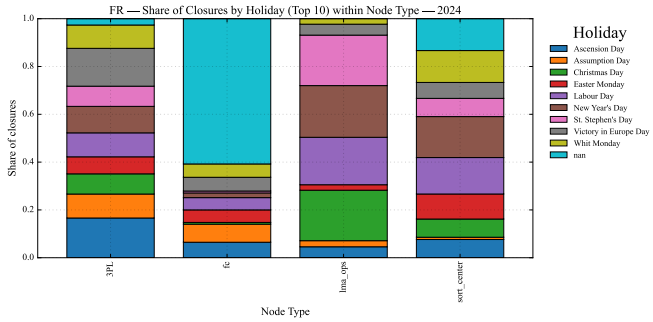


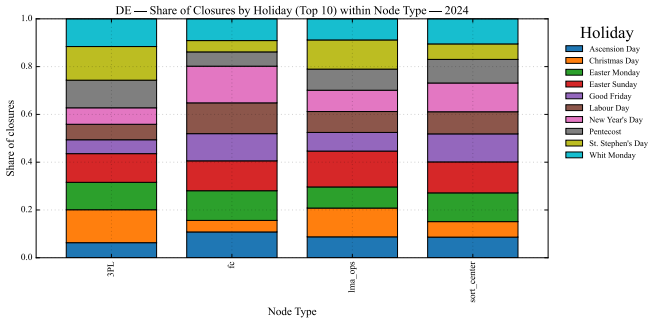
Figure 5: Example of erroneous warehouse coordinates $(0, 0)$, leading to missing spatial localization.

3.2.2 Holiday-Driven Closures by Node Type

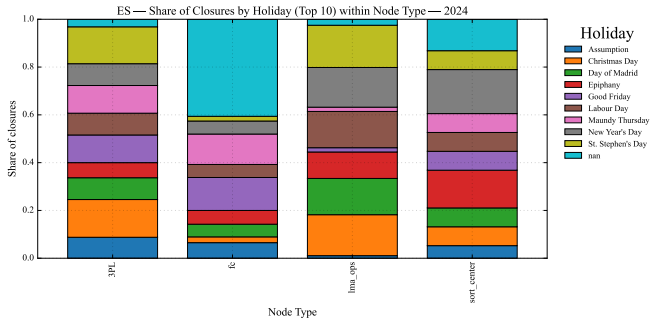
A core hypothesis behind *ANCHOR* is that warehouse closures follow structured, calendar-driven patterns aligned with public holidays, and that this alignment depends on node type. Figure 6 presents, for each EU5 country, the share of 2024 closures associated with the ten most frequent holidays, disaggregated by node type (FC, SC, DS, 3PL). Entries labeled `NaN` correspond to shutdowns not linked to a specific holiday (e.g., maintenance, audits, or exceptional disruptions).



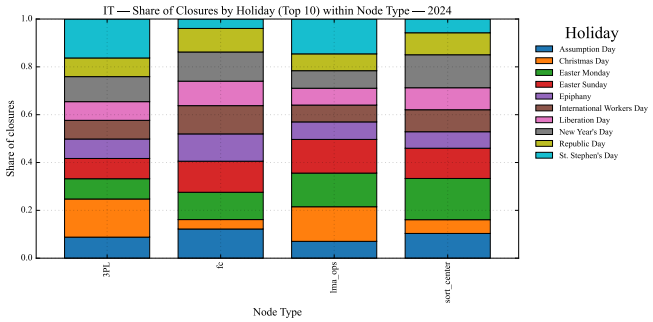
(a) France (FR)



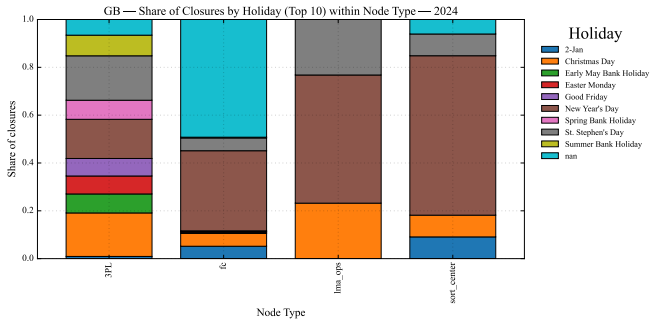
(b) Germany (DE)



(c) Spain (ES)



(d) Italy (IT)



(e) United Kingdom (GB)

Figure 6: Share of closure events by holiday (Top-10) and node type across EU5 countries in 2024. Bars represent the proportion of node-level closures associated with each holiday or shutdown category.

Interpretation. The distribution of closures across holidays and node types confirms the strong role of the calendar in operational dynamics. Fulfillment centers and sort centers exhibit the highest synchronization with holidays such as Christmas, Easter, and Labour Day, while last-mile and third-party nodes show more heterogeneous patterns. Non-holiday (`NaN`) events correspond primarily to maintenance or infrastructure work, which remain irregular and less predictable.

For modeling purposes, both the country context and the node type are retained as key covariates. All closures are included during training to capture potential cross-effects, but evaluation focuses on holiday-driven closures—the principal operational use case of *ANCHOR*.

3.2.3 Holiday Closure Traces (Binary Daily)

To visualize temporal alignment among warehouses, Figure 7 plots binary daily closure traces for French nodes in 2024. Each row corresponds to a warehouse, and black cells mark days overlapping with holiday closures. Clusters of simultaneous closures emerge around major holidays, particularly Christmas and Easter, indicating strong inter-node synchronization—a feature that temporal models can exploit.

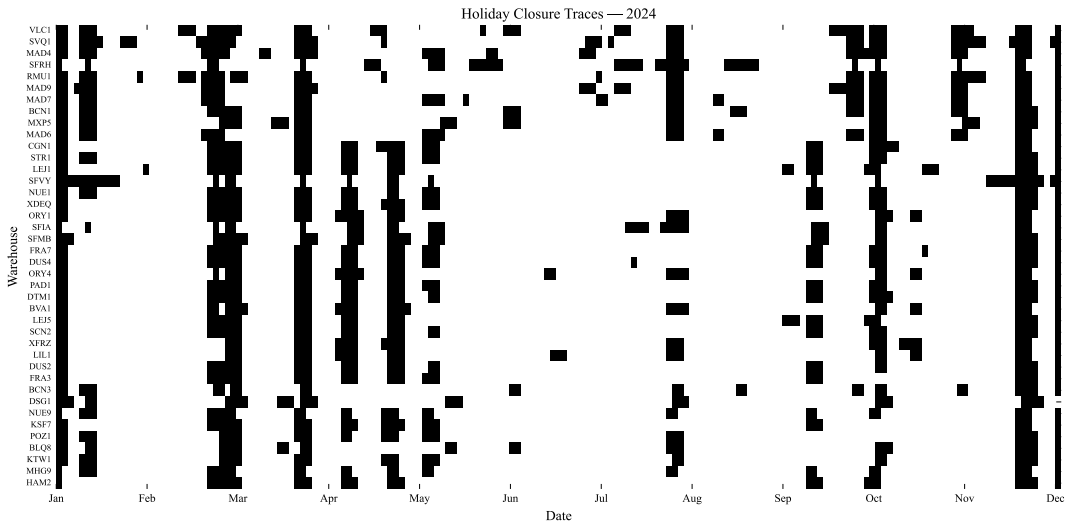


Figure 7: Binary daily closure traces for France in 2024. Each row represents a warehouse; black cells denote holiday-related closures.

3.2.4 Distributions of Closure Durations

Not all closures span the entire holiday period, and duration varies widely across nodes and countries. Figures 8 illustrate the empirical distributions of closure durations between 2021 and 2025. The distributions are strongly right-skewed: most closures last fewer than ten hours, while a thin upper tail extends to multi-day events.

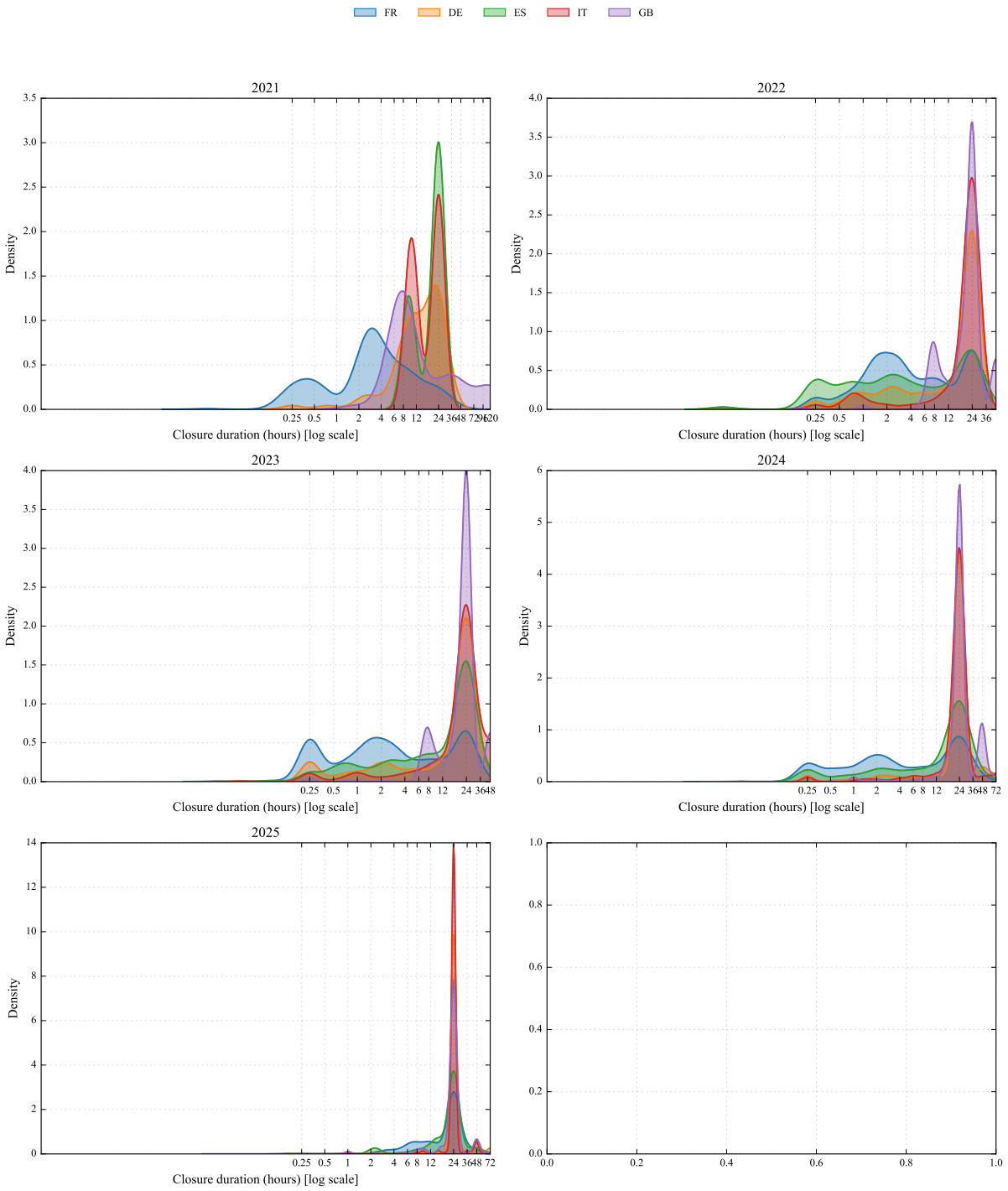


Figure 8: Log-scale kernel density estimates of closure durations (in hours) across years, aggregated by country.

Long-duration events are typically associated with maintenance or exceptional disruptions rather than holidays. While the overall shape of the distribution remains stable across years, a mild broadening is observed after 2023, which may reflect both operational variability and gradual improvements in data capture. This relative stationarity of duration distributions supports the assumption of weak concept drift discussed in Section 3, reinforcing the suitability of long-horizon forecasting models for closure prediction.

Data Quality Notes

As with most large-scale operational datasets, several quality issues were identified and systematically mitigated during preprocessing. First, geospatial attributes such as `gis_latitude` and `gis_longitude` contain missing or erroneous coordinates (e.g., $(0, 0)$), mainly due to legacy records or incomplete synchronization with the node registry. Affected entries were excluded from spatial visualizations but retained for temporal analyses to avoid bias in closure frequency estimates.

Second, the textual field `closure_description` in `cims_warehouse_closures` is inconsistently populated and lacks standardized taxonomy. While this limits the feasibility of direct natural-language categorization, manual sampling confirmed that the majority of missing descriptions correspond to predictable holiday closures or routine maintenance. Consequently, the modeling approach relies primarily on structured calendar and node-level features rather than free-text signals.

Third, cross-source temporal alignment required careful handling. Timestamp granularity and time-zone conventions differ slightly between *CIMS* and *TORCH*, occasionally producing offset intervals or duplicate closure records. The cleaning pipeline standardized all timestamps to UTC and deduplicated overlapping intervals per `warehouse_id` and date boundary, yielding a consistent daily-resolution target.

Finally, the integration of public-holiday data introduced potential labeling ambiguities near holiday boundaries. To mitigate false negatives and positives, the ± 1-day association rule was applied when linking closures to holidays (see Section 3). This conservative matching strategy ensures that overnight or extended closures are not missed due to time-zone effects.

Despite these imperfections, the resulting dataset is internally consistent and sufficiently robust for the forecasting objectives of *ANCHOR*. Residual noise, such as occasional misclassifications between holiday and non-holiday events, is absorbed by the probabilistic and hazard-based modeling frameworks discussed in the next section, which are designed to tolerate sparse and mildly noisy signals.

4 Technical Design

4.1 Solution Overview

The *ANCHOR* system adopts a modular, cloud-native architecture designed for reliability, scalability, and maintainability. It ingests operational and contextual data stored in *Amazon S3*(12)—with optional integration of other input sources such as *Amazon Redshift*(13)—and applies standardized preprocessing pipelines before reinjecting validated datasets back into S3. The feature module enriches this data with additional temporal, categorical, and spatial attributes required for both training and inference. Machine learning models are trained and deployed through *Amazon SageMaker*(14), leveraging its managed environments for reproducible, containerized workflows. Predictions and evaluation outputs are stored in S3 and subsequently processed through dedicated monitoring and validation components.

Conceptually, *ANCHOR* serves as a proactive early-warning mechanism that forecasts upcoming warehouse closures and cross-references them with planned *TORCH* deployments. This predictive layer supports two main operational objectives:

- *Enhanced configuration reliability:* by identifying closure risks ahead of configuration cutoffs, the system increases the confidence level of planning decisions;
- *Proactive defect prevention:* by surfacing inconsistencies between predicted closures and active configurations, it reduces oversight risks and prevents operational incidents.

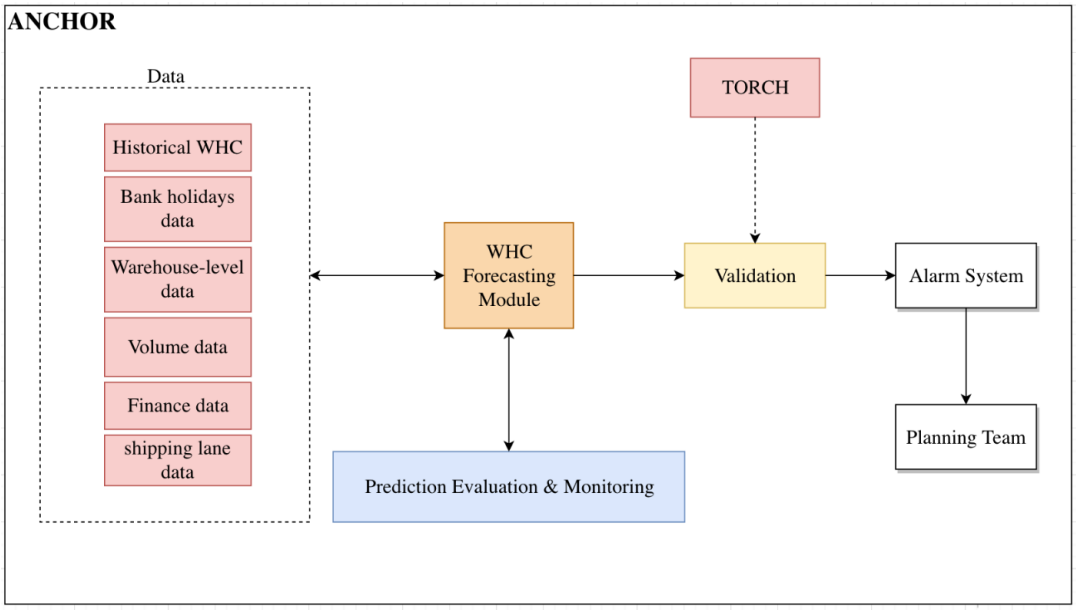


Figure 9: High-level architecture of the *ANCHOR* forecasting system.

4.2 Infrastructure and Pipeline Design

The implementation of *ANCHOR* follows a layered, serverless design built on AWS-managed services. This design provides elasticity, fault tolerance, and traceability while minimizing operational overhead.

Storage and data management. *Amazon S3* acts as the unified storage layer hosting both raw and processed datasets, including historical closure records, regional holiday calendars, and node-level metadata. Data are organized with partitioned directory structures and explicit schema contracts to ensure consistency across retraining and inference cycles. Each dataset version is timestamped and versioned, forming a reproducible and auditable lineage of inputs and outputs.

Preprocessing and feature engineering. Data transformation runs via *SageMaker Processing Jobs*, which provide managed, ephemeral compute for scalable and reproducible preprocessing. These jobs perform data validation, cleaning, normalization, and domain-specific feature generation—*e.g.* `days_since_last_closure`, `days_to_next_holiday`, and country-level seasonality indicators. Resulting feature sets are written to S3 as the input channels for training.

Model training. Training uses *SageMaker Training Jobs*, which provision instances, execute containerized training scripts, and persist artifacts to S3. Each job is configured with declared input channels, hyperparameters, and evaluation metrics, enabling reproducible experimentation and optional hyperparameter tuning. Trained artifacts and metadata are recorded in a model registry for downstream deployment.

Inference. Forecasts are produced via two complementary modes:

- *Batch Transform Jobs* on a weekly cadence to generate forward-looking closure risks for planning windows;
- *Asynchronous Endpoints* for ad-hoc scoring and scenario testing.

Each inference output is stored in S3 with its model version, input snapshot, and timestamp to ensure traceability.

Validation and alerting. Forecasted closures are cross-referenced with *TORCH* configuration data through a validation module. Discrepancies—such as a high predicted closure probability for a node not marked as closed—are flagged and escalated to the planning team via internal alerting channels.

Monitoring and evaluation. A continuous monitoring layer tracks predictive performance, data drift, forecast stability, and latency. Metrics are logged and visualized in dashboards to guide retraining or recalibration when operational patterns deviate from historical norms.

System properties. The workflow is designed to be idempotent and loosely coupled. Components communicate via S3 rather than direct service dependencies, which enables:

- *Independent scalability* of each module according to workload;
- *Fault isolation* and simple retries without cascading failures;
- *Maintainability* via clear input–output contracts;
- *Reproducibility* through versioned datasets and model artifacts.

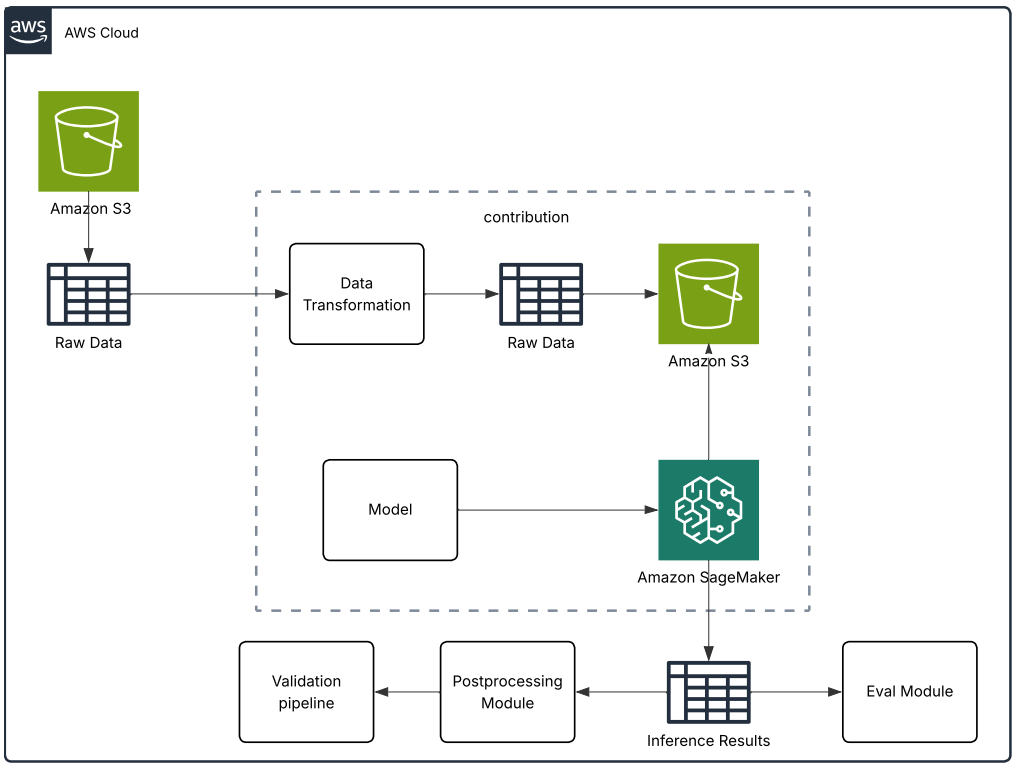


Figure 10: Low-level training and inference workflow for *ANCHOR*.

Summary. This serverless, AWS-based architecture turns *ANCHOR* into a production-ready forecasting system fit for Amazon's European logistics ecosystem. By combining managed storage (*Amazon S3*), elastic compute (*SageMaker Processing* and *Training Jobs*), and automated monitoring, the platform delivers reliability, traceability, scalability, and cost efficiency while remaining easy to extend with new features or models.

5 Scientific Contribution

ANCHOR Forecasting is a project that does not contain strict ML requirements. The ultimate goal is to propose a warning mechanism that is scalable with maintainable code and compatible with the team data infrastructure. the proposed solution contain a forecasting module where we implement our solution.

We are in front of a forecasting problem, that presents several challenges. The goal is to leverage historical warehouse closures and some additional warehouse-level features to anticipate the future closures. We have multiple warehouses per country. Rigorously, given a warehouse w, a continuous time $t \in \mathbb{R}$, and $\theta \in \Theta$ the set of parameters, we consider the function:

$$C_w \ : \ \mathbb{R}^+ \times \Theta \to \{0,1\}$$
$$(t,\theta) \mapsto C_w(t,\theta)$$

$C_w(t,\theta)$ represents the closure state function for given time and parameter set. The perfect goal would be to estimate this function.

This ML problem admits multiple modelling strategies, from deep learning–based classification of multidimensional feature vectors (e.g., MLP, LSTM) to survival and hazard-based risk modelling. For the proof of concept, we adopt the latter approach, as it delivers interpretable risk scores over time and can be directly integrated into decision-making workflows.

5.1 Daily hazard classification

We formulate warehouse-closure forecasting as a discrete-time hazard prediction task. At each node–day (w,t), the target is the probability that *at least one* closure will occur in the forward window $(t,\, t+H]$. This reparameterization turns an irregular event process into a dense supervised problem on a daily panel while preserving causal ordering and enabling calibrated, horizon-specific risk scores that directly inform *TORCH* validation. Compared to sequence models requiring long homogeneous histories, the discrete hazard view is robust to sparsity, naturally integrates structured calendar signals (holidays), and supports clear operating points for alerting.

Problem statement and target. Let $\mathcal{W}$ be the set of warehouses (nodes) and $\mathcal{T}$ the set of observation days. For each $w \in \mathcal{W}$ and $t \in \mathcal{T}$, define the binary label

$$Y_{w,t}^{(H)} = \begin{cases} 1, & \text{if at least one closure occurs in } (t,\, t+H], \\ 0, & \text{otherwise.} \end{cases}$$

We learn the *discrete-H-day hazard*

$$\hat{p}_{w,t}^{(H)} \ = \ \mathbb{P}\Big(Y_{w,t}^{(H)} = 1 \,\Big|\, X_{w,t};\, \theta\Big)\,,$$

where $X_{w,t}$ denotes features available at day t and θ are model parameters. The parameter H controls risk granularity. Operationally, H=1 provides next-day triage, while larger horizons ($H \in \{3,5,7\}$) give earlier planning signals. In what follows we study $H \in \{1,3,5,7\}$ to characterize these regimes under a unified design.

Holiday-aware panel and features. We build a node–day panel by expanding each closure interval to covered dates, aggregating to daily indicators and durations, crossing all active `node_id` with the study calendar, and injecting holiday structure via a country/date join with the internal holidays source. This yields explicit holiday covariates—`is_holiday_today`, `holiday_type`, `days_since_prev_holiday`, `days_to_next_holiday`—that act as temporal anchors. Short-term excitation and local intensity are summarized by `roll_close_count_7d/14d/30d`, `days_since_last_closure`, and `roll_mean_dur_14d`. Structural heterogeneity is captured by `country_from_panel`, `node_type`, and `time_zone_final`. All fields comply with the dataset conventions in Section 3.

Models and tuning protocol. We compare three tree-based classifiers suitable for heterogeneous tabular data: Random Forests (15), XGBoost (16), and LightGBM (17). All models return probabilities suitable for calibrated hazard scores. We first tune each model on a *single fixed horizon* and then *freeze hyperparameters* to examine multi-horizon behaviour ($H \in \{1,3,5,7\}$) under a constant capacity budget. This design isolates the effect of H from confounding changes in model size or regularization.

Metrics and losses. Closures are rare; evaluation therefore emphasizes ranking quality on the positive class. Our main discrimination metric is the *average precision* (AP), i.e. the area under the precision–recall curve (18; 19),

$$\mathrm{AP} \;=\; \sum_{k=1}^{n-1} \big(r_{k+1} - r_k\big)\, p_{k+1}, \tag{1}$$

where p_k and r_k denote precision and recall at operating point k. Probabilistic calibration is assessed with the *Brier score* (20),

$$B \;=\; \frac{1}{N} \sum_{i=1}^{N} \big(y_i - \hat{p}_i\big)^2, \tag{2}$$

with $y_i \in \{0, 1\}$ and $\hat{p}_i$ the predicted probability. We also report decision-threshold metrics: the F_1 score,

$$F_1(\tau) \;=\; \frac{2\,P(\tau)\,R(\tau)}{P(\tau) + R(\tau)}, \tag{3}$$

evaluated at the optimal threshold $\tau^* = \arg\max_\tau F_1(\tau)$, and the *precision at top-1%* alerts, which reflects the constrained manual-review regime used in practice.

Optimization losses. For boosted trees (LightGBM, XGBoost), training minimizes the *logistic loss*

$$\mathcal{L}_{\log} = -\frac{1}{N} \sum_{i=1}^{N} \Big[y_i \log \hat{p}_i + (1 - y_i) \log\big(1 - \hat{p}_i\big) \Big]. \tag{4}$$

Random Forests grow trees by minimizing node impurity (Gini/entropy), and posterior probabilities are produced by averaging class votes across trees.

Evaluation design and leakage control. We enforce temporal integrity with a chronological split and an H-day blank zone between train and validation to prevent label peeking. The training window ends on 2024-12-30; validation begins on 2025-01-01; the blank zone spans (2024-12-31, . . . , 2025-01-H). Prevalence is 1.47% in train and 3.01% in validation (Section 3). All metrics are computed on the validation window.

Fixed-horizon reference: discrimination and calibration at H=7. At the tuned fixed horizon, LightGBM attains high discriminative power with a distinct best-F_1 operating point. The PR curve remains in the upper envelope across recalls (Figure 11), indicating robust ranking under class imbalance. Score densities for positives and negatives are well separated (Figure 12), which enables conservative thresholds without sacrificing coverage. The reliability curve tracks the diagonal closely (Figure 13), so predicted risks are numerically meaningful as probabilities, a prerequisite for thresholding policies and downstream business rules.

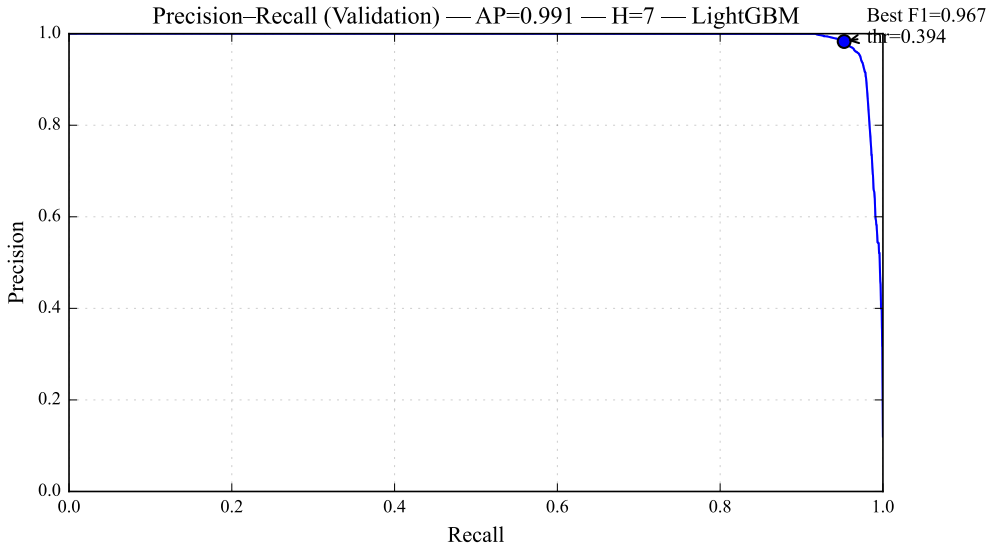


Figure 11: Precision–Recall curve with best-F_1 operating point (LightGBM, H=7). The curve's convexity and tall left shoulder indicate strong early precision on the most critical alerts.

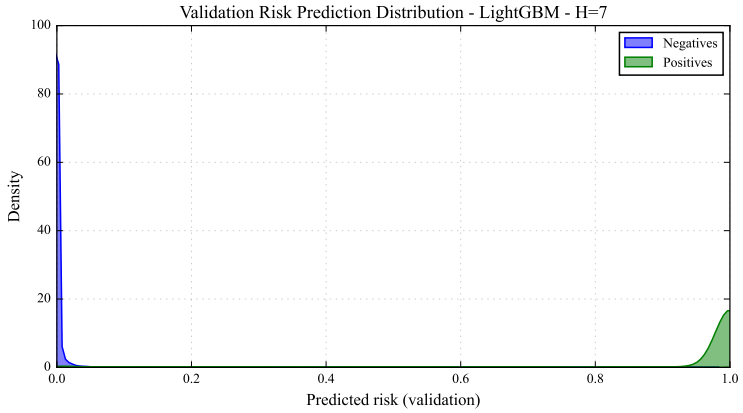


Figure 12: Risk score distributions for positives vs. negatives (LightGBM, H=7). The bimodal separation implies a wide decision margin, simplifying threshold selection for budgeted alerting.

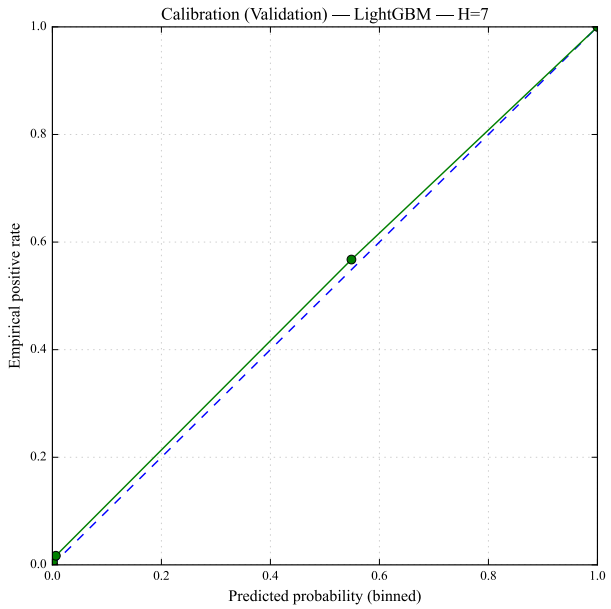
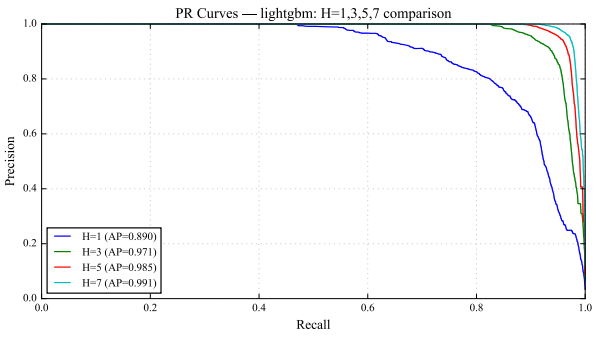
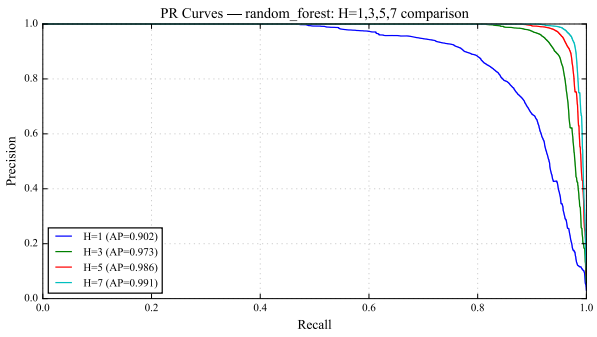


Figure 13: Calibration curve (LightGBM, H=7). The near-diagonal fit shows well-calibrated probabilities, so a nominal $x\%$ risk corresponds to an empirical $x\%$ event rate.

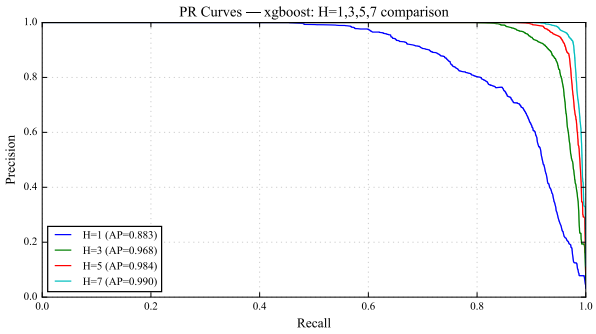
Multi-horizon analysis under frozen hyperparameters. With hyperparameters frozen, we evaluate $H \in \{1, 3, 5, 7\}$ across all models to isolate the window-length effect. PR curves in Figure 14 show the expected monotone pattern: as H shortens, labels become rarer and more volatile, compressing recall and lowering AP; as H lengthens, labels densify and precision can be maintained at higher recall. Even at H=1, early precision remains high, which is decisive for low-volume daily triage. Within-model comparisons (Figures 15–17) make this trade-off explicit: the H=1 curves sit inside their H=7 counterparts yet retain a tall left shoulder.



(a) LightGBM

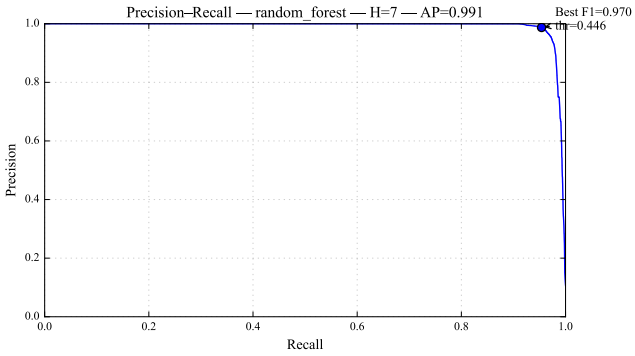


(b) Random Forest

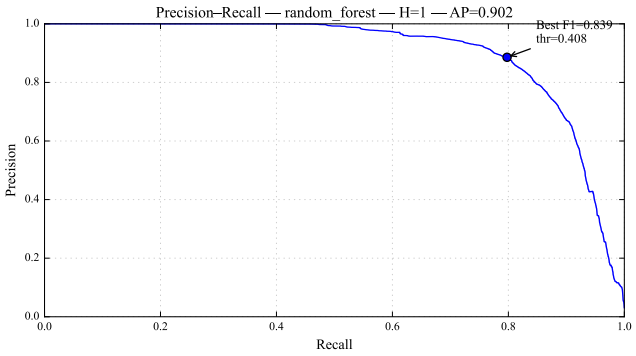


(c) XGBoost

Figure 14: PR curves across horizons $H \in \{1, 3, 5, 7\}$ with frozen hyperparameters. Longer windows translate into denser labels and improved recall at similar precision.

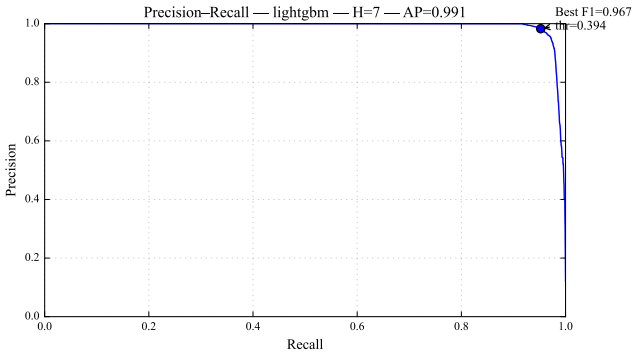


(a) Random Forest — H=7

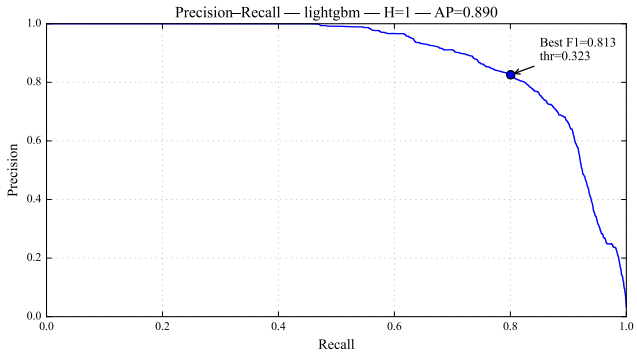


(b) Random Forest — H=1

Figure 15: Within-model PR comparison for Random Forest. The left shoulder remains high at H=1, preserving top-rank precision for budgeted alerts.

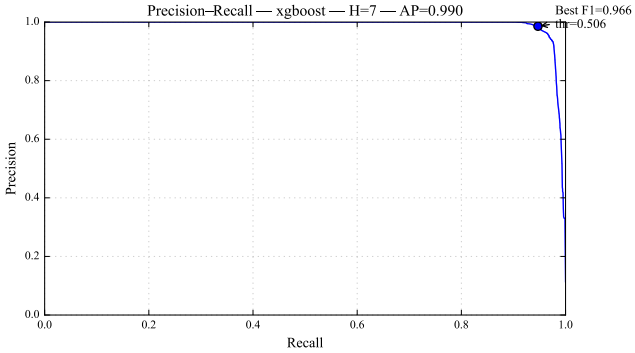


(a) LightGBM — H=7

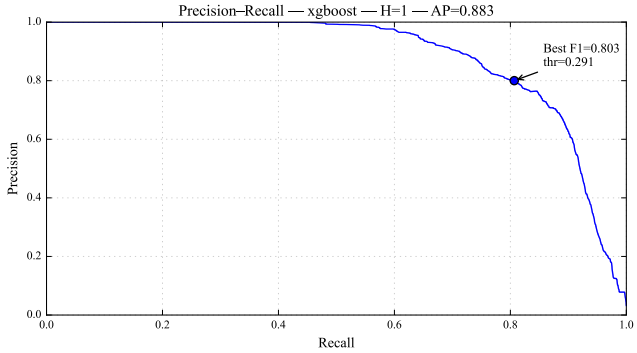


(b) LightGBM — H=1

Figure 16: Within-model PR comparison for LightGBM. The gap between H=1 and H=7 reflects sparsity rather than model capacity, given frozen hyperparameters.



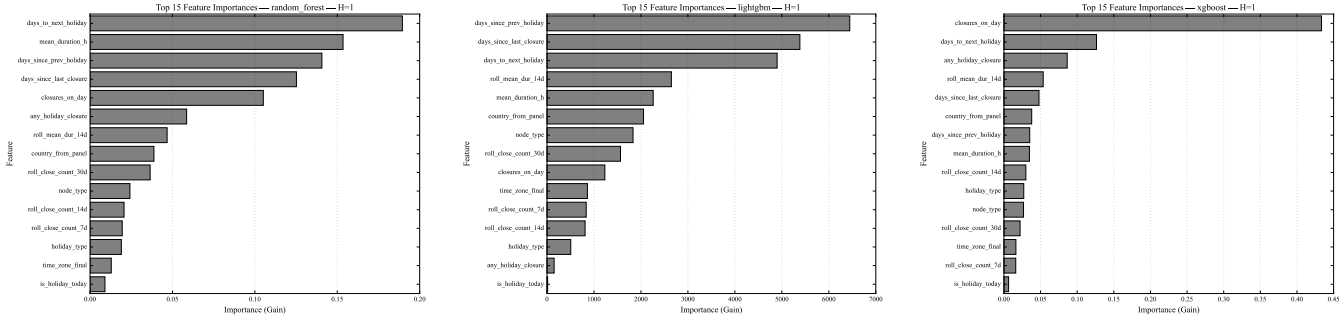
(a) XGBoost — H=7



(b) XGBoost — H=1

Figure 17: Within-model PR comparison for XGBoost.

Feature Importance: holiday proximity and self-excitation. Holiday proximity (`days_to_next_holiday`, `days_since_prev_holiday`) dominates at H=1, followed by self-excitation and local intensity (`days_since_last_closure`, `closures_on_day`) and recent duration (`mean_duration_h`). Figure 18 shows consistent patterns across algorithms; Table 1 reports Random Forest importances. For longer horizons, LightGBM elevates short-term rolling counts and weekly/seasonal encoders, aligning with calendar-aware forecasting (1; 2) and the survival/hazard framing from related work.



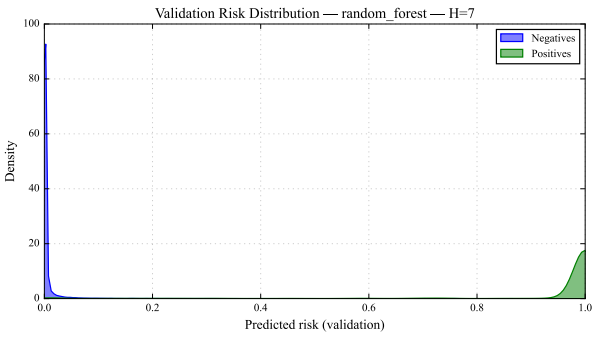
(a) Random Forest (b) LightGBM (c) XGBoost

Figure 18: Feature importances at H=1. Holiday proximity dominates across models, followed by self-excitation signals.

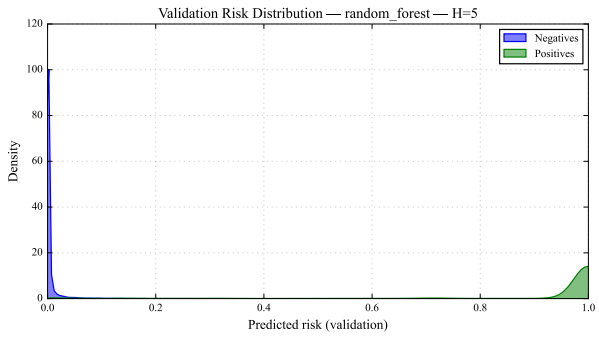
Table 1: Top predictive features by importance (Random Forest, H=1).

Feature	**Description**	**Importance (%)**
days_to_next_holiday	Days until the next public holiday	18.95
mean_duration_h	Mean duration of recent closures (hours)	15.35
days_since_prev_holiday	Days elapsed since the previous holiday	14.07
days_since_last_closure	Days elapsed since previous closure	12.52
closures_on_day	Closure occurrence on the current day	10.51
any_holiday_closure	Whether any closure coincides with a holiday	5.86
roll_mean_dur_14d	Rolling mean closure duration over 14 days	4.68
country_from_panel	Country indicator derived from panel data	3.88
roll_close_count_30d	Number of closures in the last 30 days	3.65
node_type	Node type (FC, SC, 3PL, ...)	2.42
roll_close_count_14d	Number of closures in the last 14 days	2.06
roll_close_count_7d	Number of closures in the last 7 days	1.95
holiday_type	Type of holiday (national, regional, ...)	1.90
time_zone_final	Time zone of the node location	1.29
is_holiday_today	Indicator if the current day is a holiday	0.91

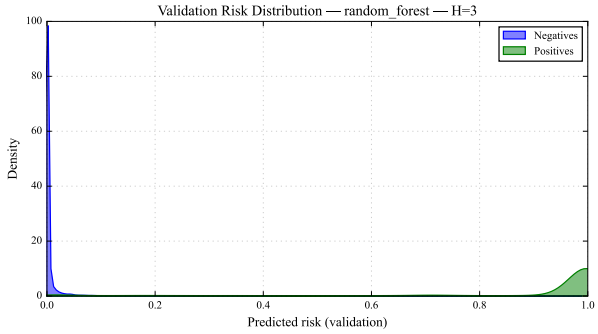
Risk separability and node-level timelines. Random Forest score densities remain well separated across horizons (Figure 19), despite the increased rarity at H=1. Node-level timelines for two representative sites (LEJ1, MRS1) in Figure 23 show sharp pre-closure spikes and rapid post-reopening decay, demonstrating that hazards track the operational cycle rather than drifting slowly, and providing face validity for on-call triage.



(a) H=7



(b) H=5



(c) H=3

Figure 19: Validation risk distributions for Random Forest across horizons. Even at H=1 (not shown), the top-rank region remains cleanly separated.

Cross-country robustness and cross-model comparison. Per-country metrics at H=1 for the tuned Random Forest show consistently high precision@top-1% across EU5, with variation in PR-AUC largely explained by prevalence and local calendar structure. This stability is key for deployment because alert volumes are budgeted per country. Comparative PR curves at H=1 (Figure 20) and $F_1(\tau)$ profiles (Figure 21) show smooth trade-offs for all models. The summary in Table 2 confirms that LightGBM and Random Forest dominate across horizons; differences are small relative to the gains delivered by the hazard framing and the holiday-aware features.

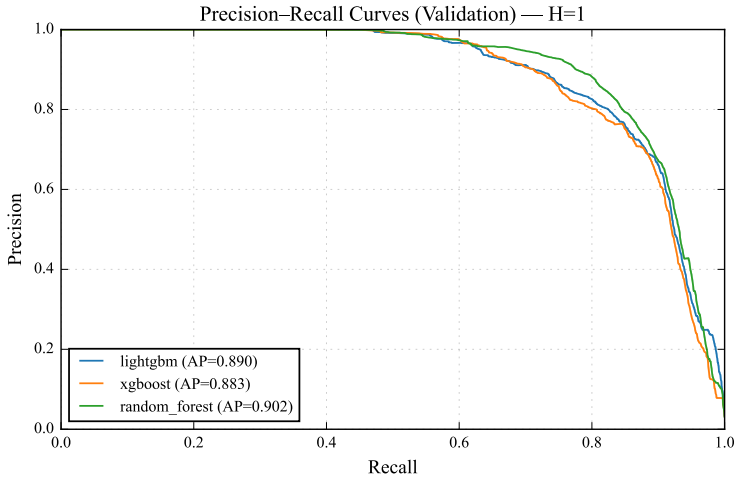
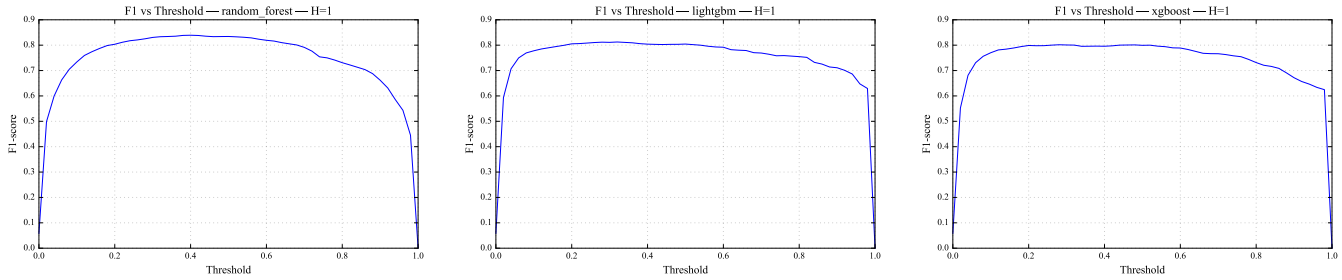


Figure 20: PR curves for tuned models at H=1. Random Forest and LightGBM nearly overlap in the high-precision, low-recall region that matters for budgeted triage.



(a) Random Forest (b) LightGBM (c) XGBoost

Figure 21: F_1 vs. decision threshold at H=1. Curves are smooth and unimodal, simplifying threshold selection to meet operational goals.

Horizon	Model	PR-AUC	Best F_1	Best Threshold	Precision@1%	Prevalence
H=1	Random Forest	**0.902077**	**0.839430**	0.408134	0.999689	0.030143
	LightGBM	0.889878	0.812989	0.322532	1.000000	0.030143
	XGBoost	0.883020	0.803406	0.291129	1.000000	0.030143
H=3	Random Forest	**0.973133**	**0.937995**	0.357524	1.000000	0.057944
	LightGBM	0.971440	0.927891	0.359866	1.000000	0.057944
	XGBoost	0.968428	0.924256	0.508075	1.000000	0.057944
H=5	Random Forest	**0.986083**	**0.959518**	0.396604	1.000000	0.083428
	LightGBM	0.985106	0.953624	0.312926	1.000000	0.083428
	XGBoost	0.983823	0.952922	0.582039	1.000000	0.083428
H=7	LightGBM	**0.991054**	0.967357	0.393673	1.000000	0.106154
	Random Forest	0.990743	**0.970400**	0.445538	1.000000	0.106154
	XGBoost	0.989969	0.965993	0.505760	1.000000	0.106154

Table 2: Cross-model performance across horizons with frozen hyperparameters. The hazard framing and holiday-aware features account for most of the lift; model-to-model gaps are modest.

Operational highlight: best H=1 model

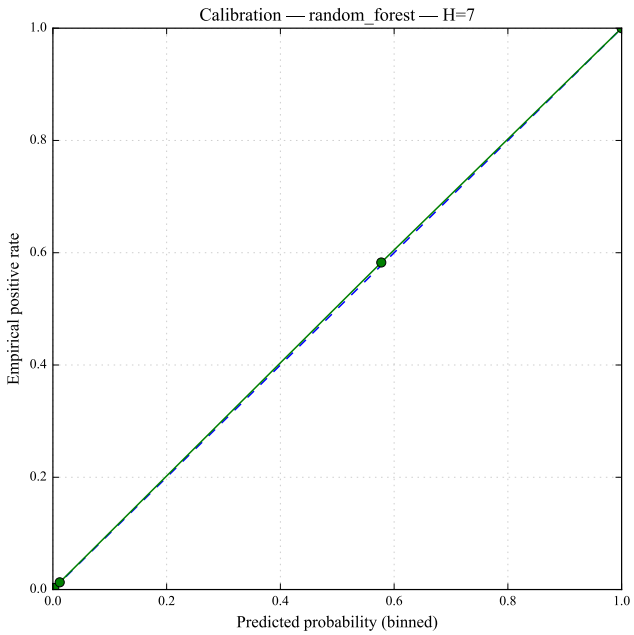
Random Forest (H=1) achieves **PR-AUC** = 0.9021 and $\mathbf{F_1}$ = 0.8394 at the best threshold $\tau^* = 0.408$. Under a budgeted policy, *precision@top-1%* is ≈1.00, enabling near-perfect triage of the highest-risk next-day node–days.

Per-country detail for the selected model. Random Forest shows strong and balanced performance across countries and horizons. Table 3 details per-country metrics; countries are comparable in overall quality, with IT and GB slightly ahead. This likely reflects that the model captures their calendar-driven patterns more sharply (holiday proximity and excitation features receive higher gain in those geographies), rather than any structural bias.

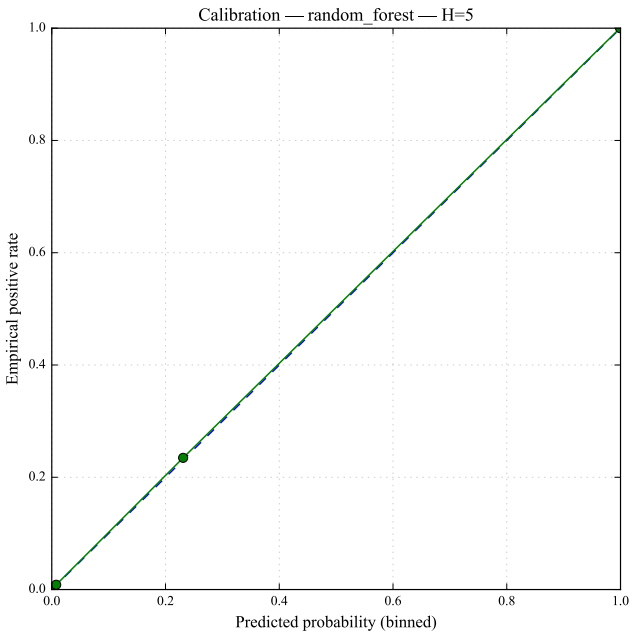
Table 3: Model performance by country and horizon — tuned Random Forest.

country	prevalence	pr_auc	precision_at_1pct	horizon
IT	0.036	**0.947**	1.000	1
GB	0.022	0.939	0.999	1
DE	0.043	0.887	1.000	1
FR	0.024	0.879	0.997	1
ES	0.021	0.819	0.916	1
IT	0.070	**0.988**	1.000	2
GB	0.042	0.981	1.000	2
FR	0.047	0.972	1.000	2
DE	0.082	0.968	1.000	2
ES	0.039	0.957	1.000	2
IT	0.098	**0.994**	1.000	5
GB	0.060	0.989	1.000	5
FR	0.069	0.987	1.000	5
DE	0.118	0.984	1.000	5
ES	0.056	0.979	1.000	5
IT	0.119	**0.996**	1.000	7
GB	0.077	0.993	1.000	7
FR	0.089	0.992	1.000	7
DE	0.152	0.988	1.000	7
ES	0.070	0.986	1.000	7

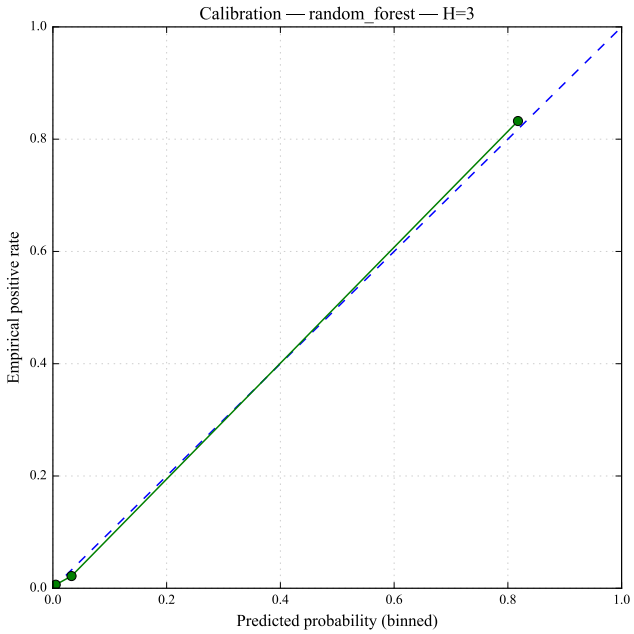
Calibration stability across horizons. Random Forest reliability curves in Figure 22 show that shorter horizons saturate earlier and occupy a narrower probability range due to extreme imbalance. Nevertheless, the curves remain close to the diagonal throughout the range used by top-rank alerts, so thresholds learned at one horizon transfer well to neighbouring horizons without retraining.



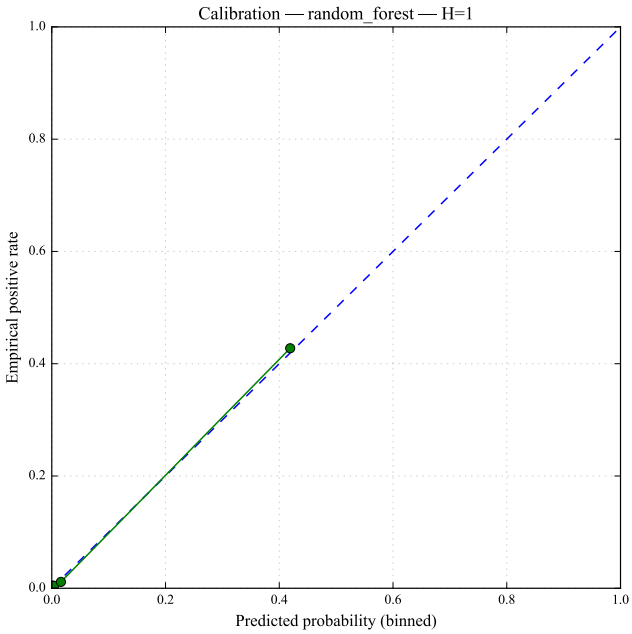
(a) H=7



(b) H=5



(c) H=3



(d) H=1

Figure 22: Calibration curves for Random Forest across horizons. Probability mass narrows as H decreases, but the range used by top-rank alerts remains well calibrated.

Node-level temporal validation. To assess temporal coherence and interpretability, we visualize daily risk trajectories for two highly active warehouses, `LEJ1` and `MRS1` (Figure 23). Predicted risk exhibits spikes immediately preceding historical closures, followed by rapid decay once operations resume, demonstrating that the model captures short-term hazard dynamics as well as seasonal recurrences. Additional per-node timelines for the highest-average-risk sites are provided in Appendix 30.

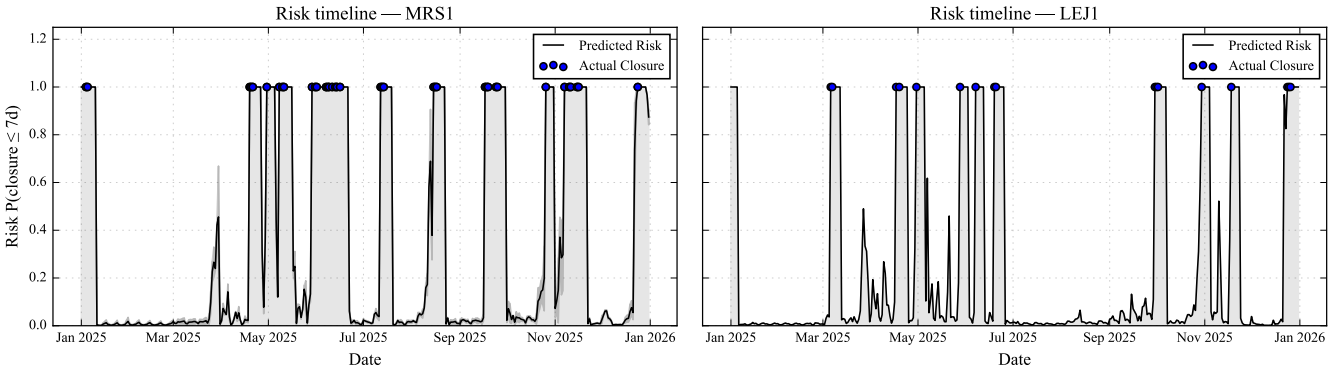


Figure 23: Daily risk timelines for `LEJ1` and `MRS1` during validation. Peaks align just before historical closures, then decay promptly as operations resume.

Discussion and connections to related work. These findings align with and extend prior literature. Calendar-aware frameworks such as Prophet and TBATS emphasize explicit seasonal/holiday effects (1; 2); our feature attributions show that a hazard formulation coupled with holiday *proximity* (time-to/ since-holiday) yields stronger operational signals than purely additive regressors, especially at short horizons where event sparsity dominates. Survival/hazard models classically target time-to-event risk (7); our discrete-time hazards adapt this view to a daily panel with rich exogenous structure while retaining calibrated probabilities. Point-process models capture burstiness and self-excitation (6); in our setting, self-excitation enters via recency and rolling-intensity features, which tree ensembles exploit effectively without full parametric Hawkes estimation. Deep probabilistic forecasters (e.g., DeepAR, TFT) often excel in dense multi-series regimes; here, sparse binary targets and the need for controllable thresholds favour tabular gradient boosting and ensembles with explicit holiday distances. Finally, hierarchical coherence (8; 9) motivates reporting and thresholding by country: per-country performance remains strong and stable, and budgeted alerting aligns with how Page 0 triage is staffed.

Limitations and extensions. Discrete hazards summarize *occurrence* within $(t,\, t+H]$ but do not predict duration; augmenting with a duration model would enable richer what-if validation in *TORCH*. Next-day prediction remains sensitive to rare, non-holiday disruptions; enriching the panel with maintenance calendars or strike indicators could harden recall. A natural extension is joint *multi-horizon* estimation, $(\hat{p}^{(1)}, \ldots, \hat{p}^{(H)})$, to produce a daily risk curve per node; survival-style auditors (Cox) or point-process diagnostics (Hawkes) can then test calibration around holiday clusters and burst periods before refreshing thresholds.

Part II
Project 2: AWS-Based Automation for Metric Computation, Reporting, and Monitoring

1 Introduction

The Transportation Network Configuration (TNC) team at Amazon plays a pivotal role in ensuring that configuration changes to the transportation network are introduced in ways that maximize speed and minimize defects. To sustain continuous development and monitor performance at scale, leadership introduced a weekly global report known as *Page 0*, which consolidates more than thirty key operational metrics. These metrics were initially collected through a fragmented and manual workflow: some were computed "offline" by metric owners using SQL queries or Excel spreadsheets, while others required manual filtering and extraction from *Amazon QuickSight*(21) dashboards. This process was inefficient, error-prone, and offered limited visibility into the historical evolution of the business. In addition, it lacked standardization, transparency, and centralization, making it difficult to maintain consistency across reporting cycles.

To address these shortcomings, we designed and implemented a fully automated pipeline for *Page 0* that leverages serverless and managed services on *AWS*. This project aimed not only to automate the computation of metrics, but also to establish a Single Source of Truth (SSOT) accessible to stakeholders in a consistent and timely manner.

During implementation, however, we encountered a significant challenge concerning one critical metric: the rate of Release Candidate (RC) rejections. RCs are bundles of configuration changes that must pass automated validation layers before deployment. When a candidate is rejected, it means that the configuration failed these validations and requires manual intervention before redeployment. Unlike other metrics, which had direct data availability through *Amazon Redshift*(13) or *Amazon S3*(12), RC rejection data originated from manually maintained files and *Quip* documents. This limitation required the design of a complementary automation to extract, transform, and report RC rejections reliably. The final outcome of the project was thus a unified automation framework composed of two layers: the general *Page 0* automation pipeline and the specialized RC rejections automation. Together, these efforts transformed an error-prone manual workflow into a scalable and auditable system.

2 Proposed Architecture

The overall architecture of the *Page 0* automation pipeline is illustrated in Figure 24. The solution was developed using the *AWS Cloud Development Kit (CDK)*(22), which allowed us to define infrastructure as code in TypeScript and deploy it in a version-controlled and modular manner. The central component of the system is an *AWS Lambda*(23)-based ETL framework. *AWS Lambda* is a serverless compute service that enables functions to run without provisioning servers; it was chosen because of its ability to scale automatically and execute ETL logic at low operational cost. For each metric, a parameterized ETL script was developed and stored in *Amazon S3*, from which *Lambda* functions dynamically retrieved and executed the appropriate transformations. Metrics derived from *Amazon Redshift* tables were handled through SQL ETLs, while those requiring external APIs or file-based data were processed through Python ETLs. The processed outputs were stored in a partitioned schema in *Amazon S3*, ensuring durable storage and simplifying downstream analytics.

To make the data consumable by stakeholders, we integrated the pipeline with the AWS analytics ecosystem. *AWS Glue*(24) crawlers were configured to automatically infer schema from the S3 datasets and register them in the *Glue Data Catalog*. Using *Amazon Redshift Spectrum*(25), these S3-backed tables were exposed to an *Amazon Redshift* cluster, providing seamless SQL access to computed metrics. On top of this storage and query layer, *Amazon QuickSight* dashboards were built to provide interactive

visualizations and dynamic comparisons of metrics over time. This integration is depicted in Figure 24, where *Amazon S3* serves as the central repository feeding both analytical queries and visualization layers. To ensure transparency in operations, an additional *Lambda* function was dedicated to monitoring quality checks on computed metrics and posting alerts or anomalies to a *Slack* channel used by the team. Design choices followed guidance from the *AWS Well-Architected Framework*(26).

A technical challenge arose from the fact that the data cluster and the application were deployed in different AWS regions. This required careful handling of *Virtual Private Clouds (VPCs)*(27), which are logically isolated sections of the AWS cloud where resources can be provisioned securely. Since cross-region communication is not enabled by default, we implemented *VPC peering*(28), which established a secure network path between the two VPCs. This ensured that *Lambda* functions could access *Redshift* clusters across regions while maintaining compliance with network security policies.

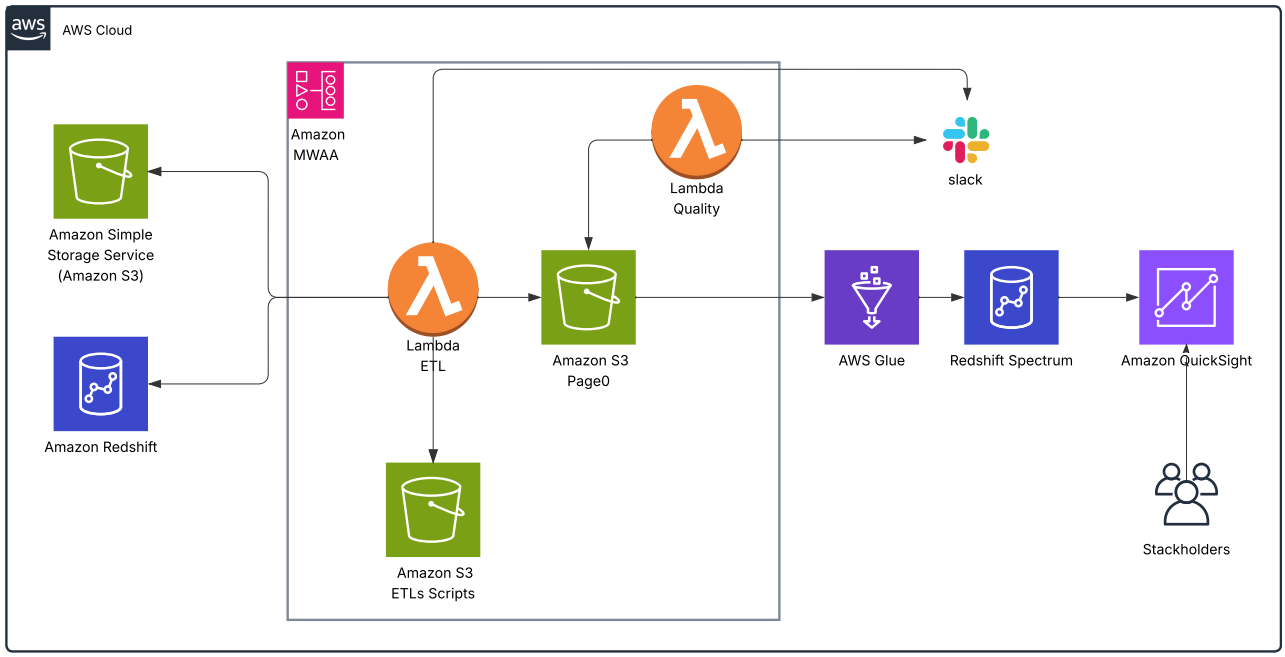


Figure 24: AWS infrastructure design for *Page 0* automation

The orchestration of these workflows was achieved through *Apache Airflow*, deployed on *Amazon Managed Workflows for Apache Airflow (MWAA)*(29). Two Directed Acyclic Graphs (DAGs) were created, one to trigger weekly metric computations and the other to trigger monthly aggregations, as represented in Figure 25. The DAG-based scheduling ensured that metrics were computed consistently at predetermined times, and the modularity of *Airflow* allowed for easy extension when new metrics were added to *Page 0*. This design emphasized scalability and maintainability, allowing future metric ETLs to be integrated into the pipeline with minimal additional effort.

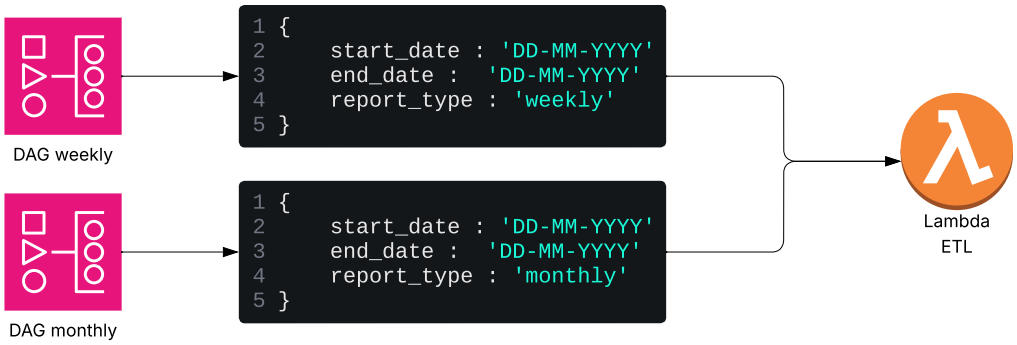


Figure 25: *Airflow* DAGs for *Page 0* automation orchestration

To ensure consistency across more than thirty metrics, we designed a unified schema for storing metric results in *Amazon S3*. Each computed metric is stored in partitioned form (by date and metric name), making the data queryable both individually and collectively. Table 4 presents an illustrative view of how this schema is structured.

Table 4: Unified schema for computed metrics (illustrative)

Metric	Group	Report	start_date	end_date	value	Num.	Den.	active	timestamp	Team	subteam
#metric_i	GROUPE_J	WEEKLY	5/5/2025	10/5/2025	70	70	1	1	24/6/2025 02:00AM	-	-
%metric_i	GROUPE_J	MONTHLY	1/5/2025	31/5/2025	50%	1200	2400	1	1/6/2025 02:00AM	-	-

3 RC Rejections Automation

3.1 Motivation

The RC rejections pipeline required a tailored design because the data was not directly available in AWS sources and because stakeholders requested that each rejection be transformed into an actionable task. When a candidate is rejected, no automation can automatically fix it, as it has already failed previous validation layers. Instead, it requires human investigation before redeployment. For this reason, the customer requested a feature that automatically creates an *Asana* task for each rejected candidate, assigning it to the responsible owner extracted from metadata. This addition provided accountability, visibility, and direct tracking of rejection handling.

The legacy process involved local scripts generating raw rejection reports, followed by manual post-processing and data-copying steps across *Quip* documents, which delayed visibility and introduced inconsistencies. Figure 26 illustrates the previous workflow.

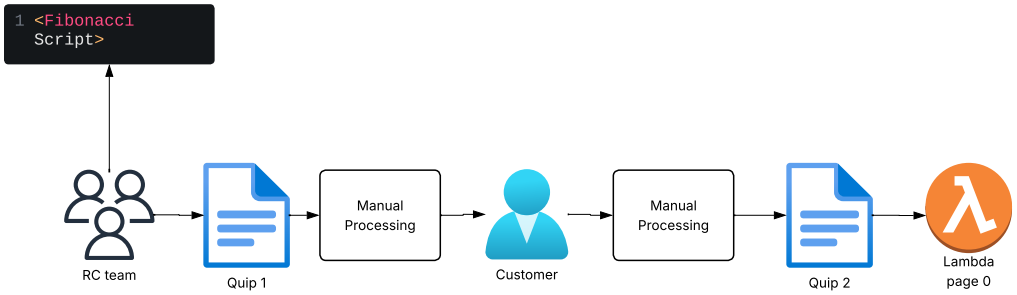


Figure 26: Legacy RC rejections processing workflow

3.2 Architecture Design

To overcome these issues, we built a serverless, event-driven pipeline on *AWS* (Figure 27). At its core, the system uses *AWS Step Functions*(30) to orchestrate a central *Lambda* function, triggered daily to transform

available data from multiple systems into structured rejection records. These records automated previous manual processing steps, producing consistent and auditable outputs. The processed datasets were then stored in partitioned *Amazon S3* buckets, providing durable and versioned storage for historical analysis.

A major improvement was the integration with task management tools. Each rejection was automatically converted into an *Asana* task via API integration, enriched with custom fields such as rejection reason and ownership. This transformed rejection tracking into an actionable workflow, ensuring accountability and faster resolution. *Slack* integration was also maintained to provide real-time operational visibility, including pipeline execution logs and failure alerts.

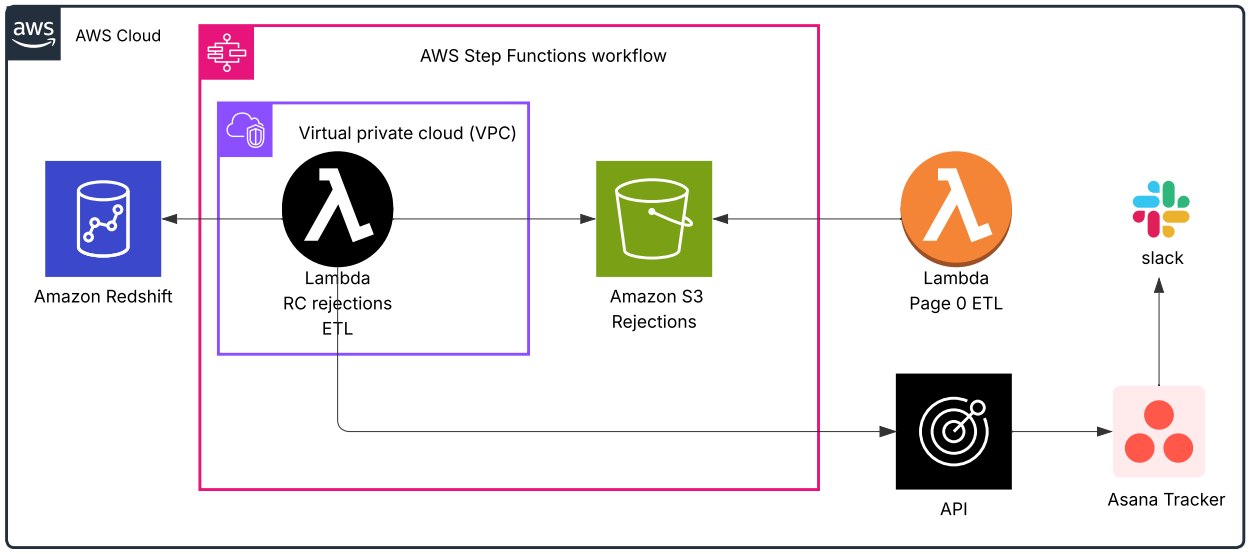


Figure 27: AWS design for RC rejections automation

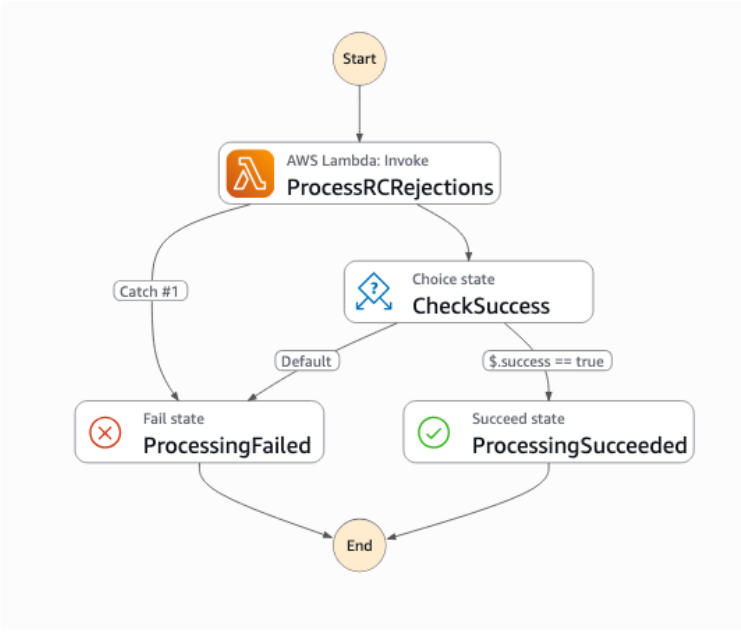
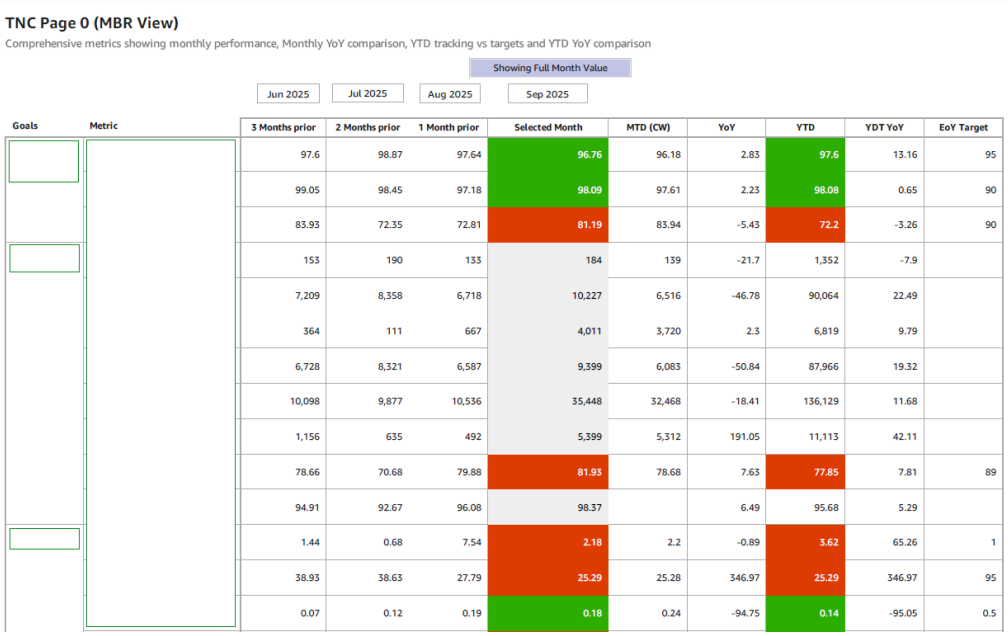


Figure 28: *AWS Step Functions* orchestration for RC rejections automation

4 Results and Impact

The implementation of these pipelines substantially improved the reliability, transparency, and timeliness of metric reporting. The latency of *Page 0* reporting was reduced from several days of manual compilation to fully automated runs within hours of data availability. The storage of results in *Amazon S3*, combined with *AWS Glue* and *Redshift Spectrum*, established a single authoritative repository for metrics, which improved data consistency across stakeholders and analytical models. *Amazon QuickSight* dashboards (Figure 29) provided stakeholders with dynamic and interactive access to metrics, significantly enhancing decision-making capabilities.



TNC Page 0 (MBR View)

Comprehensive metrics showing monthly performance, Monthly YoY comparison, YTD tracking vs targets and YTD YoY comparison

Showing Full Month Value

Jun 2025 | Jul 2025 | Aug 2025 | Sep 2025

Goals	Metric	3 Months prior	2 Months prior	1 Month prior	Selected Month	MTD (CW)	YoY	YTD	YDT YoY	EoY Target
		97.6	98.87	97.64	96.76	96.18	2.83	97.6	13.16	95
		99.05	98.45	97.18	98.09	97.61	2.23	98.08	0.65	90
		83.93	72.35	72.81	81.19	83.94	-5.43	72.2	-3.26	90
		153	190	133	184	139	-21.7	1,352	-7.9	
		7,209	8,358	6,718	10,227	6,516	-46.78	90,064	22.49	
		364	111	667	4,011	3,720	2.3	6,819	9.79	
		6,728	8,321	6,587	9,399	6,083	-50.84	87,966	19.32	
		10,098	9,877	10,536	35,448	32,468	-18.41	136,129	11.68	
		1,156	635	492	5,399	5,312	191.05	11,113	42.11	
		78.66	70.68	79.88	81.93	78.68	7.63	77.85	7.81	89
		94.91	92.67	96.08	98.37		6.49	95.68	5.29	
		1.44	0.68	7.54	2.18	2.2	-0.89	3.62	65.26	1
		38.93	38.63	27.79	25.29	25.28	346.97	25.29	346.97	95
		0.07	0.12	0.19	0.18	0.24	-94.75	0.14	-95.05	0.5

Figure 29: *Amazon QuickSight* dashboard view of automated *Page 0* metrics (metric names hidden for confidentiality)

The RC rejections automation had a transformative effect on operational workflows. By replacing *Quip*-based processes with structured data pipelines and integrating with *Asana*, the pipeline introduced accountability, enabling rejection issues to be tracked, assigned, and resolved in a timely fashion. This eliminated the risks of data loss, transcription errors, and delays inherent to the previous manual system, while also establishing a historical record for trend analysis and systemic improvements.

The impact of automation is summarized in Table 5, contrasting the legacy manual process with the new automated system.

Table 5: Comparison of metric reporting before and after automation

Dimension	Before (Manual)	After (Automated)
Metric collection	Manual SQL, Excel, *Quip*, dashboards	Automated ETLs on *AWS* infrastructure
Reporting latency	Several days	Hours after data availability
Consistency	Prone to human error, multiple sources of truth	Single Source of Truth in *Amazon S3*
Visualization	Static documents (*Quip*/Excel)	Dynamic *Amazon QuickSight* dashboards
RC rejections	Weekly manual reports, untracked ownership	Automated, tasks created in *Asana* with accountability
Extensibility	High manual overhead for new metrics	Modular ETL framework (new script = new metric)
Monitoring	No automated monitoring	*Slack* alerts and automated quality checks

5 Discussion and Outcomes

This project demonstrated the value of serverless and managed cloud services for large-scale process automation. *AWS Lambda* proved to be an ideal choice for lightweight ETL tasks due to its scalability and pay-per-use model. *Amazon S3* and *AWS Glue* provided a durable and flexible foundation for data storage and schema management, while *Redshift Spectrum* and *Amazon QuickSight* enabled seamless analytics and visualization. The integration of *Asana* in the RC rejections automation highlighted the flexibility of serverless design in addressing non-standard data sources and extending pipelines into operational workflows.

A key lesson learned was the importance of modularity. By designing metric ETLs as independent units stored in *Amazon S3*, the framework allowed new metrics to be integrated by simply uploading an additional script, without modifying the underlying infrastructure. The use of the *AWS CDK* for infrastructure as code also proved critical, ensuring that deployments were reproducible, permissions were explicitly managed, and future extensions could be made safely. We aligned architectural decisions with the *AWS Well-Architected Framework*.

Conclusion

This internship delivered two complementary contributions to Amazon's European Transportation Network. First, the ANCHOR forecasting component demonstrated that discrete-time, holiday-aware hazard modelling can reliably anticipate warehouse closures across EU5 countries and surface actionable early warnings for configuration validation in TORCH. In validation, the proof of concept achieved strong discrimination for next-day risk and very high precision for early-alert horizons, indicating readiness for operational integration. Second, the Page 0 automation replaced fragmented manual reporting with a serverless, auditable pipeline that standardizes metric computation and distribution, establishes a single source of truth, and adds accountability by automatically opening and tracking RC-rejection tasks in Asana with Slack visibility.

Together, these results show how calibrated ML risk signals and cloud-native automation reinforce each other: forecasts prevent avoidable configuration defects, while automated reporting compresses latency and improves governance. In the near term, we plan to (i) extend ANCHOR with joint multi-horizon risk curves and optional duration modelling, (ii) harden next-day recall using maintenance/strike signals, and (iii) broaden Page 0 data coverage and lineage dashboards. Longer term, integrating drift monitoring and periodic recalibration will support sustained performance as business patterns evolve.

References

[1] S. J. Taylor and B. Letham, "Forecasting at scale," *The American Statistician*, vol. 72, no. 1, pp. 37–45, 2018.

[2] A. De Livera, R. Hyndman, and R. Snyder, "Forecasting time series with complex seasonal patterns using exponential smoothing," *Journal of the American Statistical Association*, vol. 106, pp. 1513–1527, 01 2010.

[3] S. Scott and H. Varian, "Predicting the present with bayesian structural time series," *Int. J. of Mathematical Modelling and Numerical Optimisation*, vol. 5, pp. 4 – 23, 01 2014.

[4] D. Salinas, V. Flunkert, and J. Gasthaus, "Deepar: Probabilistic forecasting with autoregressive recurrent networks," 2019. [Online]. Available: https://arxiv.org/abs/1704.04110

[5] B. Lim, S. O. Arik, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," 2020. [Online]. Available: https://arxiv.org/abs/1912.09363

[6] A. G. Hawkes, "Spectra of some self-exciting and mutually exciting point processes," *Biometrika*, vol. 58, no. 1, pp. 83–90, 1971.

[7] D. R. Cox, "Regression models and life-tables (with discussion)," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 34, no. 2, pp. 187–220, 1972.

[8] R. J. Hyndman, R. A. Ahmed, G. Athanasopoulos, and H. L. Shang, "Optimal combination forecasts for hierarchical time series," *Computational Statistics Data Analysis*, vol. 55, no. 9, pp. 2579–2589, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167947311000971

[9] A. Panagiotelis, G. Athanasopoulos, P. Gamakumara, and R. J. Hyndman, "Forecast reconciliation: A geometric view with new insights on bias correction," *International Journal of Forecasting*, vol. 37, no. 1, pp. 343–359, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207020300911

[10] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and H. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys (CSUR)*, vol. 46, 04 2014.

[11] R. Richard and N. Belacel, "An online, adaptive and unsupervised regression framework with drift detection for label scarcity contexts," 12 2023.

[12] Amazon Web Services, "Amazon simple storage service (s3): Developer guide," https://docs.aws.amazon.com/AmazonS3/latest/dev/Welcome.html, 2023, accessed 2025-10-31.

[13] ——, "Amazon redshift: Management guide," https://docs.aws.amazon.com/redshift/latest/mgmt/welcome.html, 2024, accessed 2025-10-31.

[14] ——, "Amazon sagemaker: Developer guide," https://docs.aws.amazon.com/sagemaker/latest/dg/whatis.html, 2023, accessed 2025-10-31.

[15] G. Louppe, "Understanding random forests: From theory to practice," 2015. [Online]. Available: https://arxiv.org/abs/1407.7502

[16] 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. ACM, Aug. 2016, p. 785–794. [Online]. Available: http://dx.doi.org/10.1145/2939672.2939785

[17] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

[18] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd International Conference on Machine Learning*. ACM, 2006, pp. 233–240.

[19] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets," *PLOS ONE*, vol. 10, no. 3, p. e0118432, 2015.

[20] F. Sanders, "The verification of probability forecasts," *Journal of Applied Meteorology (1962-1982)*, vol. 6, no. 5, pp. 756–761, 1967. [Online]. Available: http://www.jstor.org/stable/26173495

[21] Amazon Web Services, "Amazon quicksight: User guide," https://docs.aws.amazon.com/quicksight/latest/user/welcome.html, 2024, accessed 2025-10-31.

[22] ——, "Aws cloud development kit (cdk): Developer guide," https://docs.aws.amazon.com/cdk/latest/guide/home.html, 2024, accessed 2025-10-31.

[23] ——, "Aws lambda: Developer guide," https://docs.aws.amazon.com/lambda/latest/dg/welcome.html, 2024, accessed 2025-10-31.

[24] ——, "Aws glue: Developer guide," https://docs.aws.amazon.com/glue/latest/dg/what-is-glue.html, 2024, accessed 2025-10-31.

[25] ——, "Amazon redshift spectrum: Querying external data," https://docs.aws.amazon.com/redshift/latest/dg/c-using-spectrum.html, 2024, accessed 2025-10-31.

[26] ——, "Aws well-architected framework: Machine learning lens," https://docs.aws.amazon.com/wellarchitected/latest/machine-learning-lens/welcome.html, 2022, accessed 2025-10-31.

[27] ——, "Amazon vpc: User guide," https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html, 2024, accessed 2025-10-31.

[28] ——, "Vpc peering guide," https://docs.aws.amazon.com/vpc/latest/peering/what-is-vpc-peering.html, 2024, accessed 2025-10-31.

[29] ——, "Amazon managed workflows for apache airflow (mwaa): User guide," https://docs.aws.amazon.com/mwaa/latest/userguide/what-is-mwaa.html, 2024, accessed 2025-10-31.

[30] ——, "Aws step functions: Developer guide," https://docs.aws.amazon.com/step-functions/latest/dg/welcome.html, 2024, accessed 2025-10-31.

Appendix

Appendix I

SQL Extraction Query

Listing 1: SQL query for closure data with holidays

```
WITH closure_data AS (
SELECT
wi.org as country,
wi.node_type,
whc.warehouse_id,
whc.closure_description,
whc.start_date,
whc.start_time::VARCHAR,
whc.end_date,
whc.end_time::VARCHAR,
DATEDIFF(minute,
(whc.start_date::date || ' ' || whc.start_time)::timestamp,
(whc.end_date::date || ' ' || whc.end_time)::timestamp
)::float / 60.0 as closure_duration_hours
FROM andes.cims_datawarehouse_prod.cims_warehouse_closures whc
LEFT JOIN andes.cims_datawarehouse_prod.cims_domain_warehouse_info wi
ON whc.warehouse_id = wi.warehouse
WHERE wi.node_type IN ('fc', 'sort_center', 'lma_ops', '3PL')
AND wi.is_deprecated = 'N'
AND wi.org IN ('DE', 'ES', 'FR', 'IT', 'GB')
AND whc.start_date::DATE >= '2021-01-01'
AND whc.end_date::DATE <= '2025-12-31'
)
SELECT
cd.*,
h.english_name as holiday_name,
h.holiday_date,
CASE WHEN h.holiday_date BETWEEN
DATEADD(day, -1, cd.start_date) AND DATEADD(day, 1, cd.end_date)
THEN 1 ELSE 0 END as is_holiday_closure
FROM closure_data cd
LEFT JOIN eu_config.holidays h
ON cd.country = h.country
AND h.holiday_date::date BETWEEN
DATEADD(day, -1, cd.start_date) AND DATEADD(day, 1, cd.end_date);
```

Hazard classification : Alert threshold analysis - LightGBM

Table 6 summarizes operating points used for alerting calibration. The `best_F1` row corresponds to the F1-optimal threshold on the validation set; the quantile rows ($q90, \ldots, q99.5$) correspond to fixed alert budgets.

Table 6: LightGBM : Threshold summary (essential columns).

label	threshold	precision	recall	F_1	alerts_rate
best_F1	0.975063	0.995842	0.890826	0.940411	0.085292
q90.0	0.214269	0.894722	0.938413	0.916047	0.100003
q95.0	0.999981	1.000000	0.526124	0.689491	0.050164
q97.5	0.999990	1.000000	0.262232	0.415505	0.025003
q99.0	0.999993	1.000000	0.105151	0.190293	0.010026
q99.5	0.999996	1.000000	0.052520	0.099798	0.005008

Table 7: Threshold summary (with counts).

label	threshold	positives_alerted	alerts	positives_total
best_F1	0.975063	55804	56037	62643
q90.0	0.214269	58785	65702	62643
q95.0	0.999981	32958	32958	62643
q97.5	0.999990	16427	16427	62643
q99.0	0.999993	6587	6587	62643
q99.5	0.999996	3290	3290	62643

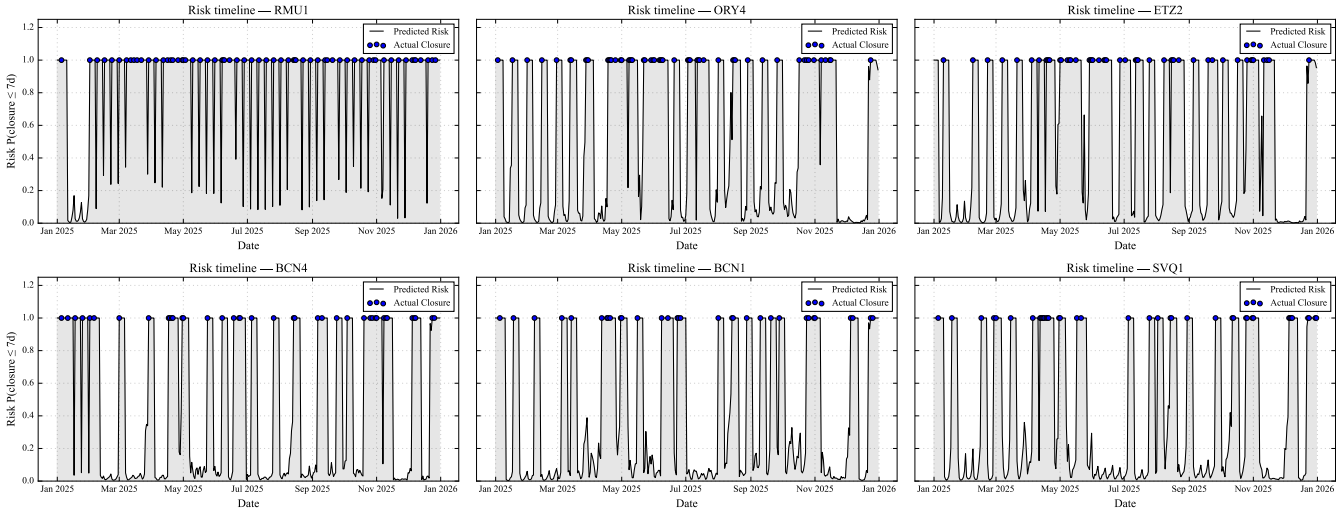


Figure 30: Risk timelines for the highest-average-risk nodes (validation).