Detecting Route Changes from RTT Measurements using XGBoost and Feature Engineering

Thiago Prado de Azevedo Andrade[1], **Bruno Llacer Trotti**[2], **Daniel Sadoc Menasché**[2]

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)
Niterói – RJ – Brazil

[2]Instituto de Computação – Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro, RJ – Brazil

01tpaa.br@gmail.com, brunollacer@hotmail.com, sadoc@dcc.ufrj.br

Abstract. *We address the route-change detection task of the Second Data Challenge CT-Mon/RNP 2025. Given consecutive traceroute measurements for a fixed origin–destination pair, the goal is to predict whether a route change occurred, using only last-hop RTT samples and probing metadata under extreme class imbalance. We propose a lightweight, pair-centric feature engineering pipeline that summarizes short-horizon RTT dynamics (e.g., step changes and deviations from rolling baselines), sampling irregularity, and reply/probe ratios, and trains a global XGBoost classifier with imbalance-aware weighting and validation-based threshold selection. On a held-out validation split, the resulting model achieves high discriminative performance (AUC-ROC 0.9956) and operates at a recall-oriented point (recall 0.93, precision 0.60 for route changes), making it suitable for monitoring and alerting applications. We further analyze gain-based feature importance, showing that consecutive RTT step differences and short-window deviations dominate the predictive signal.*

1. Introduction

Internet paths evolve continuously due to routing re-convergence, traffic engineering, failures, and load balancing [Measurement Lab (M-Lab) 2025, Chen and Guestrin 2016, Pedregosa et al. 2011]. These dynamics often translate into changes in end-to-end latency, commonly measured as round-trip time (RTT). Detecting when a path has *meaningfully* changed is important for operational monitoring and for performance-driven route selection, because reacting to every fluctuation can cause instability, while reacting too late can prolong degraded service. Prior work has therefore advocated detecting significant changepoints in RTT time series and studying how such RTT changes align with underlying path events, emphasizing that RTT is noisy, heterogeneous across source–destination pairs, and frequently affected by both routing changes and load-balancing-induced path variability.

A key practical challenge is that RTT and path changes do not match in a trivial one-to-one manner. Measurements are typically taken at different cadences (e.g., frequent RTT probing vs. sparser traceroute/path snapshots), and multiple detections may fall within a tolerance window around a single ground-truth event. The evaluation framework of Shao *et al.* [Shao et al. 2017] highlights the need for (i) change detection methods tailored to RTT time series, (ii) scoring schemes that tolerate small time shifts and

weight changes by operational importance (e.g., level/volatility shifts), and (iii) careful treatment of load balancing, which can induce apparent IP-path changes without a true routing event.

This paper tackles the *route-change detection* problem in the Second Data Challenge CT-Mon/RNP 2025:[1] given consecutive traceroute-based measurements for a fixed origin–destination (OD) pair, predict whether a route change occurred. Our goal is not to infer the underlying routing process, but to provide an accurate and lightweight predictive signal for operational monitoring. Unlike classical changepoint detection pipelines that aim to infer changepoints from full RTT time series, our setting is explicitly supervised and strongly imbalanced, with route changes being rare. We therefore design a lightweight learning pipeline that (i) summarizes last-hop RTT samples and probe metadata into robust, inexpensive features, (ii) computes all temporal/lags *within each OD pair* to preserve pair-specific baselines and sampling irregularity, and (iii) trains a tree-based gradient-boosting model (XGBoost) that can capture non-linear interactions between short-horizon dynamics (e.g., RTT steps) and stability indicators (e.g., rolling deviations, loss/reply ratios).[2]

Our approach is guided by two lessons distilled from prior measurement studies: first, RTT behavior is highly heterogeneous and often departs from simple parametric families, so robust feature summaries and non-linear decision boundaries are attractive; second, path-change interpretation must be cautious in the presence of load balancing and measurement artifacts, which motivates focusing on *predictive* signals available at the time of classification rather than attempting to perfectly reconstruct causality.

Contributions. We make the following contributions:

- **Exploratory measurement analysis.** We conduct an exploratory statistical study of last-hop RTT and loss behavior in the CT-Mon/RNP traces (Section 4), including distributional diagnostics (Section 4.1), hypothesis tests contrasting RTT behavior with vs. without route changes (Section 4.2), and a directional predictive analysis between loss and route changes (Section 4.3). These findings motivate our emphasis on robust, short-horizon RTT features.
- **Supervised route-change detector.** We propose a supervised, feature-based route-change detector for CT-Mon/RNP 2025 (Section 5) that relies only on last-hop RTT samples and traceroute probe metadata, and that explicitly addresses extreme class imbalance via training-time weighting and validation-time threshold selection. We introduce a *pair-centric* feature construction strategy (Section 5.1), where lagged and rolling statistics are computed independently per origin–destination pair, preserving local baselines and sampling cadence while training a single global model.
- **Empirical evaluation and interpretability.** We report end-to-end results on a held-out validation split (Section 6), including aggregate metrics, a confusion-matrix analysis, and gain-based feature importance to identify which feature

[1] `https://www.rnp.br/noticias/ct-mon-lanca-novo-data-challenge-e-convida-equipes-a-explorar-mudancas-de-rotas-na-internet/` e `https://www.kaggle.com/t/bba8b5ece5ef4681beae8e8f32f25481`

[2] The described method ranked as the second best solution on the official leaderboard. `https://rnp.br/noticias/ct-mon-divulga-vencedores-do-segundo-data-challenge/`

families drive performance. We show that short-horizon RTT dynamics (e.g., consecutive-step changes and deviations from short rolling means) dominate the predictive signal, while timing and probe/reply ratios provide complementary cues.

Organization. The remainder of this paper is organized as follows. Section 2 briefly describes the dataset, Section 3 presents our evaluation protocol and Section 4 reports exploratory statistical findings on RTT and loss that motivate our feature design. Section 5 describes the proposed learning pipeline, including pair-centric feature construction, the XGBoost model, and the decision-threshold selection procedure. Section 6 presents the main evaluation results, including aggregate metrics and an analysis of feature importance. Section 7 discusses broader implications of our work, Section 8 presents related work and Section 9 concludes the paper.

2. Dataset Description

The dataset comprised traceroute measurements between hundreds of IP pairs collected between March and July 2025. Each instance contained:

- RTT samples (`all_rtts`) for the last hop;
- Probe statistics (`tr_attempts`, `total_probes_sent`, `total_replies_last_hop`);
- Temporal indicators (`date_index`, `seconds_since_start`);
- A binary label (`route_changed`) for training data.

The evaluation metric defined by the organizers was the F1-score, chosen to handle the class imbalance present in the dataset.

Additional Dataset Characteristics. The training set contains approximately 2.0×10^7 samples, of which only 3.7×10^5 correspond to route changes. Thus, the positive class represents about 1.8% of the data, resulting in a highly imbalanced classification problem.

Pair-based Learning Perspective. A key aspect of our methodology is that all temporal processing and feature construction are performed *per origin–destination pair* $(\mathrm{tr\_src}, \mathrm{tr\_dst})$. In the CT-Mon/RNP dataset, each pair exhibits its own routing dynamics, sampling frequency, diurnal patterns, and baseline RTT levels. To capture these heterogeneous behaviors, we group the measurements by pair and compute all lagged features, rolling statistics, and inter-sample intervals within each group independently.

This pair-centric organization ensures that the model learns temporal signatures that are meaningful for each specific source–destination path, rather than relying on global averages that would obscure pair-level structure. Features encode localized patterns, allowing XGBoost to implicitly learn which pairs are more volatile and how their RTT patterns evolve over time. In practice, this mirrors real operational monitoring systems, where route-change detection is typically performed independently for each path.

Model Selection and Training. We employ XGBoost with tree-based gradient boosting and logistic loss. Training is performed using an 83/17 stratified train–validation split. Early stopping with a patience of 50 rounds is applied on a held-out validation set. The final decision threshold is selected by maximizing F1-score on the validation data.

No sequence models or deep architectures are used; all performance gains stem from feature engineering and tree-based modeling. The model was trained on the provided

training set and evaluated using a hold-out validation split. Performance was measured with the F1-score to ensure fair treatment of both classes.

3. Evaluation Protocol and Dataset Separation

The challenge organizers provided three distinct datasets with clearly defined roles, ensuring a strict separation between training and evaluation data. The performance of our solutions in the three datasets is reported in Table 1.

Dataset 1 (Training data with validation split). This is the only dataset for which both features and labels are available. To tune hyperparameters and select the decision threshold, we split this dataset into training and validation subsets using a stratified split (83% for training and 17% for validation). All model development decisions, including feature engineering and threshold selection, are based exclusively on this dataset. In particular, the metrics reported in Section 6 were obtained with this dataset.

Dataset 2 (Public test set/public leaderboard set). This dataset is used for intermediate evaluation through the Kaggle platform. Its labels are not accessible to participants; instead, performance metrics are returned upon submission. This allows iterative model improvement while preserving the integrity of the evaluation process. Despite multiple submissions being allowed, no direct information from this dataset is incorporated into training.

Dataset 3 (Private test set/final evaluation set). The final evaluation is performed on a separate, fully hidden dataset that is consulted only once to produce the official ranking. Participants have no access to this dataset, nor to its labels or intermediate feedback. The final leaderboard position is therefore based on a strictly unseen dataset. Our solution was ranked **second overall** in this final evaluation.

Strict separation and absence of data leakage. It is important to emphasize that we have no access to the labels or raw data of Datasets 2 and 3. All model training, feature construction, and parameter tuning are performed exclusively on Dataset 1. The evaluation protocol enforced by the challenge platform guarantees a strict separation between training and test data, eliminating any possibility of data leakage from test sets into the training process.

Feature computation protocol. Global statistics, such as `rtts_mean` and `route_change_mean`, are computed on the full Dataset 1 and used as fixed parameters of the model. In contrast, temporal features (e.g., rolling means and related statistics) are computed online for each origin–destination pair, based solely on past observations available up to the current time. This design ensures that feature construction respects the temporal structure of the data and avoids incorporating future information, thereby preserving a causal and deployment-consistent evaluation setting.

Role of the public leaderboard in model development. Since there is no prior literature or established benchmarks for this specific task, the public leaderboard (Dataset 2) played an important role in contextualizing performance.In particular, it allowed us to compare our results against other participants and against the competition baseline (F1-score = 0.77037), providing a reference to assess whether a given F1-score (e.g., around 0.60)

Table 1. Summary of our model performance across datasets.

Dataset	Role	F1-score	Notes
Dataset 1	Train/Validation	0.7327	Validation split ($\tau^\star = 0.12$)
Dataset 2	Public leaderboard	0.8604	Second place (multiple submissions)
Dataset 3	Private leaderboard	0.8595	Final ranking (second place)

should be considered strong or weak given the intrinsic difficulty of the problem. Importantly, Dataset 2 was used only for high-level model selection and performance validation, and not for direct training.

In the early stages of development, we experimented with simpler models, such as decision trees, primarily to gain insight into feature importance and model behavior. However, these approaches failed to surpass the competition baseline on the public leaderboard, leading us to discard them. We also explored alternative gradient-boosting methods (e.g., LightGBM), but ultimately found that XGBoost consistently provided more robust performance. Based on these observations, we focused our efforts on refining the XGBoost-based pipeline, performing all feature engineering, hyperparameter tuning, and threshold selection exclusively using Dataset 1. Our final solution ranked second overall. On the public leaderboard (Dataset 2), the top three teams achieved F1-scores of 0.86911, 0.86042, and 0.86041. On the private leaderboard (Dataset 3), the top three F1-scores were 0.86815, 0.85951, and 0.85908.

Temporal evaluation considerations. Although the internal validation uses a stratified random split within Dataset 1, the final performance assessment relies on Dataset 2 (public leaderboard) and Dataset 3 (private leaderboard), both of which correspond to strictly unseen data. In particular, Dataset 3 provides a forward-looking evaluation, ensuring that the reported ranking reflects generalization to unseen temporal regimes.

4. Exploratory Statistical Findings on RTT and Loss

Before presenting the learning pipeline, we summarize three empirical findings that motivated our feature design and our modeling choices: (i) RTT samples deviate from simple parametric families; (ii) route changes are associated with statistically significant shifts in RTT behavior; and (iii) losses and route changes exhibit a directional predictive relationship in the CT-Mon/RNP traces.

4.1. Distributional diagnostics for RTT

We first assess whether last-hop RTT samples adhere to common parametric distributions. We use quantile–quantile (QQ) plots to compare the empirical RTT quantiles against candidate families. Figure 1 reports representative QQ plots for Normal, Exponential, Lognormal, Gamma, and Weibull references. Overall, the plots show systematic departures from straight-line behavior, indicating that a single parametric family is insufficient to describe RTTs uniformly across origin–destination (OD) pairs and time windows. This motivates the use of distribution-free tests and robust summary features (e.g., quantiles and rolling deviations), rather than relying on a specific assumed RTT model.

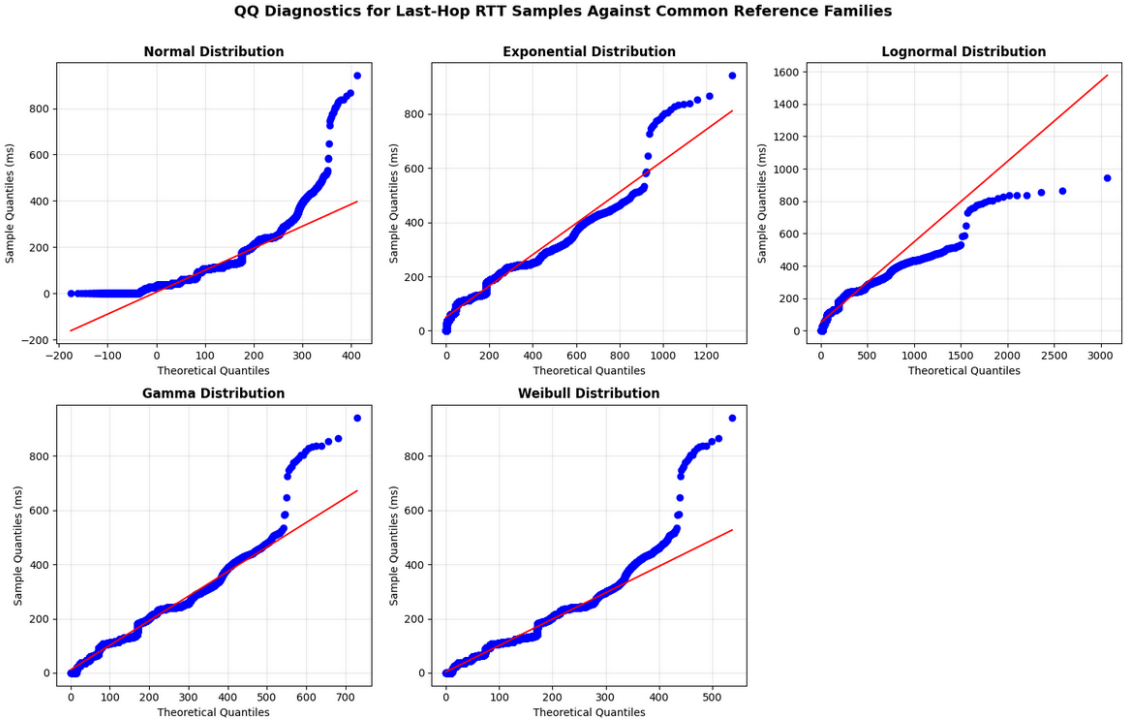

Figure 1. QQ plot comparing RTT fits

Table 2. Hypothesis testing summary comparing RTT behavior with vs. without a route change (aggregate across OD pairs, with subsampling/windowing to mitigate dependence). All tests reject H_0 at $\alpha = 10^{-3}$.

Test	H_0	H_1	Statistic	Decision
KS	RTT distributions are equal	RTT distributions differ	0.068	Reject H_0
CvM	RTT distributions are equal	RTT distributions differ	118.76	Reject H_0
MWU	$P(\mathrm{RTT}_{with} > \mathrm{RTT}_{without}) = 0.5$	$\neq 0.5$	1.55×10^{11}	Reject H_0
BF(Levene)	RTT variances are equal	RTT variances differ	2006.92	Reject H_0
Quantile(80)	$Q_{80}^{with} \leq Q_{80}^{without}$	$Q_{80}^{with} > Q_{80}^{without}$	16.19	Reject H_0

4.2. Hypothesis testing: RTT behavior with vs. without route change

We split RTT samples into two groups: measurements labeled with a route change and measurements without a route change. The null hypothesis is that both groups are drawn from the same RTT distribution (H_0), against the alternative that a route change is associated with a distributional shift (H_1). To avoid parametric assumptions, we apply distribution-free tests (Kolmogorov–Smirnov, Cramér–von Mises, Mann–Whitney U). We additionally test variance shifts (Levene/Brown–Forsythe) and compare high-percentile behavior via a one-sided 80th-quantile comparison. Because RTT series can exhibit autocorrelation and non-stationarity, we mitigate dependence effects via windowing and subsampling. We reject H_0 at significance level $\alpha = 10^{-3}$.

Table 2 summarizes the aggregate outcomes (across OD pairs). All tests strongly reject H_0, supporting the conclusion that route changes are associated with measurable shifts in RTT behavior, including at upper quantiles.

Table 3. Predictive-causality test results for Loss → Route change. Past losses significantly improve route-change prediction across all tested lags (10–50s).

Direction	Lag	Lag (s)	F-stat	p-value	df_1	df_2	Reject H_0
Loss → Route change	1	10	13.33	2.61×10^{-4}	1	406830	Yes
Loss → Route change	2	20	14.07	1.00×10^{-6}	2	406827	Yes
Loss → Route change	3	30	6.96	1.11×10^{-4}	3	406824	Yes
Loss → Route change	4	40	5.40	2.42×10^{-4}	4	406821	Yes
Loss → Route change	5	50	9.68	$< 10^{-6}$	5	406818	Yes

Table 4. Predictive-causality test results for Route change → Loss. The effect is not significant at short lags (10–30s) but becomes significant at longer lags (40–50s).

Direction	Lag	Lag (s)	F-stat	p-value	df_1	df_2	Reject H_0
Route change → Loss	1	10	1.74	0.187	1	406830	No
Route change → Loss	2	20	5.78	3.08×10^{-3}	2	406827	No
Route change → Loss	3	30	2.32	0.0736	3	406824	No
Route change → Loss	4	40	8.23	1.00×10^{-6}	4	406821	Yes
Route change → Loss	5	50	9.13	$< 10^{-6}$	5	406818	Yes

4.3. Directional predictive relationship: loss vs. route change

We next investigate whether losses help predict route changes, whether route changes help predict losses, or whether both directions are present. We extract time-binned (and lagged) event series per OD pair and apply Granger-style predictive causality tests in both directions: (i) Loss → Route change, and (ii) Route change → Loss. We evaluate multiple lags (10–50 seconds in our binning).

Tables 3 and 4 summarize the results. Loss events significantly improve the prediction of future route changes across all tested lags (10–50s). In contrast, the reverse direction is not significant at short lags (10–30s) but becomes significant at longer lags (40–50s). This pattern suggests that, in these traces, (a) losses often precede and help anticipate route changes, while (b) route changes may also induce subsequent loss bursts, but with a longer delay.

5. Data and Methods

We work on the RNP Data Challenge 2025 traces, using the provided `train.csv` as our labeled corpus. Each row aggregates active probing metadata for a pair $(\mathrm{tr\_src}, \mathrm{tr\_dst})$ at a time t and contains the binary label `route_changed`.

5.1. Feature Engineering

All features are computed independently for each origin–destination pair $(\mathrm{tr\_src}, \mathrm{tr\_dst})$ after sorting measurements by time. Let r_t denote the mean RTT at time t, and let t^- be the previous measurement time for the same pair.

1. **RTT difference** (`rtts_diff`), $x_t^{(1)} = |r_t - r_{t^-}|$. *Comment:* Captures abrupt RTT steps, which commonly occur when a path switch introduces different propagation or queueing conditions.

2. **Normalized RTT difference** (`rtts_diff_norm`)

$$x_t^{(2)} = \frac{|r_t - r_{t^-}|}{r_t + \varepsilon}.$$

Comment: Scale-invariant version of the RTT step, allowing comparison across pairs with very different baseline RTTs.

3. **Seconds since last sample** (`seconds_since_last_sample`)

$$x_t^{(3)} = \Delta_t = \texttt{seconds\_since\_start}(t) - \texttt{seconds\_since\_start}(t^-).$$

Comment: Encodes sampling irregularity; large gaps often coincide with measurement anomalies or operational events.

4. **Rolling mean RTT (window 5)**

$$\overline{r}_t^{(5)} = \frac{1}{5}\sum_{k=0}^{4} r_{t-k}.$$

Comment: Local baseline for recent RTT behavior.

5. **Distance from rolling mean** (`rtts_dist_from_mean_5`), $x_t^{(4)} = |r_t - \overline{r}_t^{(5)}|$. *Comment:* Measures how much the current RTT deviates from short-term history.

6. **Global mean RTT**

$$\mu = \frac{1}{T}\sum_{\tau=1}^{T} r_\tau.$$

Comment: Long-term baseline RTT for the pair.

7. **Distance from global mean** (`rtts_dist_from_mean`), $x_t^{(5)} = |r_t - \mu|$. *Comment:* Highlights sustained shifts away from typical operating regime.

8. **RTT z-score** (`rtts_zscore`)

$$x_t^{(6)} = \frac{r_t - \mu}{\sigma + 10^{-8}}, \qquad \sigma = \sqrt{\frac{1}{T}\sum_{\tau}(r_\tau - \mu)^2}.$$

Comment: Normalized anomaly score relative to historical variability.

9. **Sudden change indicator** (`rtts_sudden_change`)

$$x_t^{(7)} = \mathbf{1}\left[|r_t - r_{t^-}| > 2 \cdot \sigma_t^{(10)}\right],$$

where $\sigma_t^{(10)}$ is the rolling standard deviation over the last 10 samples. *Comment:* Binary flag for extreme local jumps.

10. **Reply-to-probe monotonicity score** (`ratio_mono_dist1`)

$$x_t^{(8)} = \frac{1}{1 + \left|\frac{\texttt{total\_replies\_last\_hop}_t}{\texttt{total\_probes\_sent}_t} - 1\right|}.$$

Comment: Penalizes missing replies; low values indicate loss or instability.

11. **Route change rate** (`route_change_rate`)

$$x^{(9)} = \frac{1}{T}\sum_{\tau=1}^{T} \mathbf{1}[\texttt{route\_changed}_\tau = 1].$$

Comment: Encodes historical volatility of the pair; some paths are intrinsically more unstable than others.

XGBoost input. All feature vectors are concatenated into $\mathbf{x}_t = (x_t^{(1)}, \ldots, x_t^{(9)})$ and provided as input to the XGBoost classifier.

Training/validation split. The training/validation split is stratified: 83% train and 17% validation. We account for class imbalance with

$$\texttt{scale\_pos\_weight} = \frac{\#\{\text{negatives}\}}{\#\{\text{positives}\}}.$$

Main hyperparameters (as in the notebook): `learning_rate`$=0.03433$, `max_depth`$=15$, `subsample`$=0.56212$, `colsample_bytree`$=0.86667$, `min_child_weight`$=4$, with early stopping on the validation set.

5.2. Decision threshold selection

XGBoost produces, for each traceroute measurement, a predicted probability $p_t = \Pr(\texttt{route\_changed}_t = 1 \mid \mathbf{x}_t)$. To obtain a binary prediction, this probability must be converted into a decision via a threshold τ:

$$\widehat{y}_t = \begin{cases} 1, & p_t \geq \tau, \\ 0, & p_t < \tau. \end{cases}$$

In highly imbalanced datasets such as CT-Mon/RNP, using the default threshold $\tau = 0.5$ leads to poor recall for the minority class (route changes). Instead, we treat τ as a tunable hyperparameter. After training the model, we generate predicted probabilities on the validation set and sweep τ across a grid from 0.10 to 0.89 in increments of 0.01. For each value, we compute the resulting F1-score. The threshold that maximizes the validation F1-score is selected for deployment.

Using this procedure, the optimal threshold is $\tau^\star = 0.12$, which significantly improves the balance between precision and recall when compared to the default cutoff. All results reported in Section 6 use $\tau^\star$ as the decision boundary.

6. Results

This section evaluates the effectiveness of the proposed feature-based XGBoost pipeline for detecting route changes from traceroute RTT measurements. We report performance on a held-out validation split using both threshold-free metrics (AUC-ROC) and operating-point metrics (precision, recall, F1-score) to account for the severe class imbalance in the dataset. In addition to aggregate classification results, we analyze the confusion matrix, examine feature importance to understand which signals drive predictions, and compare against a simple threshold-based baseline. Together, these results provide both quantitative evidence of accuracy and qualitative insight into the mechanisms by which route changes manifest in RTT time series.

Aggregate metrics (validation, $\tau^\star = 0.12$). On the validation split, the proposed model achieves an AUC-ROC of **0.9956**, an overall accuracy of **0.9875**, and an F1-score of **0.7327** for the positive (route-change) class. The near-perfect AUC indicates excellent ranking ability, i.e., the model assigns substantially higher scores to route-change samples than to non-change samples across a wide range of thresholds. This is particularly important in highly imbalanced settings, as it demonstrates that good performance is not tied to a single operating point.

The achieved accuracy should be interpreted with care, since the dataset is dominated by negative samples. More informative are the class-specific metrics reported below, which reveal a deliberate operating point favoring recall over precision. **Confusion matrix (validation, $\tau^\star = 0.12$).**

$$\begin{bmatrix} \text{TN} & \text{FP} \\ \text{FN} & \text{TP} \end{bmatrix} = \begin{bmatrix} 3{,}293{,}142 & 38{,}109 \\ 4{,}472 & 58{,}355 \end{bmatrix}.$$

From this matrix, we obtain a recall of **0.93** and a precision of **0.60** for the route-change class. The high recall indicates that the system successfully identifies the vast majority of true route changes, missing only a small fraction of events. This operating regime is well aligned with monitoring and alerting scenarios, where failing to detect a routing event is typically more costly than generating a moderate number of false alarms. The observed precision implies that approximately three out of five raised alarms correspond to genuine route changes, which is acceptable for large-scale telemetry pipelines that can apply secondary filtering or correlation across signals.

The false-positive count, while non-negligible in absolute terms, represents a very small fraction of the total number of negative samples. This reflects the extreme class imbalance and further highlights the importance of threshold selection rather than reliance on the default 0.5 cutoff.

Feature importance (gain). To better understand which signals drive the model's decisions, we examine the gain-based feature importance scores produced by XGBoost. Gain measures the average reduction in training loss brought by splits on a given feature across all trees, and therefore reflects each feature's relative contribution to discrimination. While such importance scores do not imply causality, they offer useful insight into which aspects of RTT behavior and probe metadata are most informative for detecting route changes.

Feature	Gain importance
`rtts_diff`	148.68
`rtts_dist_from_mean_5`	18.58
`route_change_rate`	14.44
`rtts_dist_from_mean`	7.46
`seconds_since_last_sample`	2.77
`rtts_zscore`	1.99
`ratio_mono_dist1`	1.70
`rtts_diff_norm`	0.99
`rtts_sudden_change`	0.82

The dominance of `rtts_diff` shows that abrupt step changes between consecutive RTT measurements are the strongest indicator of route changes. This supports the

intuition that routing transitions often manifest as sudden shifts in propagation delay and queueing conditions.

The second most important feature, `rtts_dist_from_mean_5`, captures deviations from a short rolling baseline, emphasizing that local context is more informative than global averages. Together, these two features confirm that *short-horizon RTT dynamics* drive most of the predictive power.

The feature `route_change_rate` ranks third, indicating that certain source–destination pairs exhibit persistent volatility and that incorporating a pair-specific prior improves discrimination. This suggests that route-change propensity is not uniform across the network and can be learned implicitly by the model.

Mid-ranked features such as `rtts_dist_from_mean`, `seconds_since_last_sample`, and `rtts_zscore` provide complementary information about longer-term baselines and sampling irregularity. Lower-ranked features, including `rtts_diff_norm` and `rtts_sudden_change`, appear largely redundant with `rtts_diff`, but still contribute marginal gains by refining decision boundaries.

Takeaways. Overall, the results indicate that route changes can be detected reliably using simple, inexpensive features derived from consecutive RTT measurements. Most of the predictive signal resides in short-term RTT discontinuities and deviations from recent baselines, while pair-level priors further enhance performance. These findings explain why a tree-based model with carefully engineered features is sufficient to achieve strong performance, without requiring sequential deep models or complex latent-state inference.

7. Discussion

This section consolidates and interprets our empirical findings and modeling choices, and outlines concrete directions to extend route-change detection beyond the CT-Mon/RNP challenge setting.

Why short-horizon RTT dynamics dominate. A consistent outcome of our experiments is that short-horizon features—most notably the absolute RTT step difference (`rtts_diff`) and deviations from rolling means—carry the strongest predictive signal. Intuitively, route changes manifest as abrupt shifts in propagation and queueing conditions, which appear as discontinuities in consecutive RTT measurements. This observation is also compatible with a minimal two-route bottleneck model: if each route is associated with distinct propagation delay and queueing parameters, then the distribution of RTT increments under *no change* is symmetric and centered near zero, whereas increments under a *change* are shifted and typically more asymmetric, making large steps informative. This provides a simple mechanistic explanation for why a tree-based learner such as XGBoost prioritizes `rtts_diff`-type splits.

Distributional heterogeneity and robustness implications. QQ diagnostics and distribution-free hypothesis tests jointly indicate that RTT behavior is heterogeneous across OD pairs and time periods, and does not admit a single stable parametric description. This has two practical consequences. First, it supports the use of robust summaries (quantiles, rolling deviations, and normalized differences) rather than relying on an assumed family for likelihood-based detection. Second, it suggests that evaluation proto-

cols that stress temporal generalization (e.g., time-based splits) are essential: the learner should be tested on future periods where baseline RTT levels and noise regimes may drift.

Loss versus route change: interpreting directional effects. Our predictive-causality analysis indicates evidence for *both* directions, but with distinct time scales. Loss $\rightarrow$ route change is significant across all tested lags (10–50 s), consistent with loss bursts acting as early warnings of impending routing events (e.g., failures or instability that precede reconvergence). Route change $\rightarrow$ loss is not significant at short lags (10–30 s) but becomes significant at longer lags (40–50 s), consistent with delayed loss effects following transitions (e.g., transient micro-loops, traffic shifts to congested links, or stabilization dynamics). Operationally, this supports treating loss-derived metadata as an anticipatory signal while still allowing for post-change loss patterns that may refine detection in longer windows.

Leveraging TCP timing as an auxiliary signal. Although TCP does not explicitly expose route changes, its timing machinery can serve as a useful side channel. The retransmission timeout (RTO) algorithm [Paxson and Allman 2011], $\mathrm{RTO} = \mathrm{SRTT} + \max(G, 4 \cdot \mathrm{RTTVAR})$, reacts to abrupt delay inflation and loss via spikes in RTTVAR, near-RTO flight times, and RTO expirations. Modern mechanisms such as PRR and RACK/TLP [Cheng et al. 2013, Cheng et al. 2021] rely on fine-grained timing and can produce distinctive recovery signatures during disruptions. In a deployment, these signals can be collected via standard OS instrumentation (e.g., `/proc/net/tcp`, socket statistics, or BPF tracing) and aligned with the active-probing timeline. Rather than replacing traceroute-based labels, TCP timing can be incorporated as additional features to (i) provide earlier warnings than the probing cadence permits, and (ii) disambiguate loss bursts caused by congestion from those associated with control-plane transitions.

Modeling strategies: per-pair, global, and hybrid. The dataset structure naturally supports multiple learning strategies. Per-pair classifiers can capture highly localized baselines but scale poorly and fail on sparse or unseen pairs. A single global model (our approach) is scalable and benefits from a large training corpus, provided that temporal features are computed within each pair to preserve locality. Hybrid approaches may offer a better Pareto point: clustering OD pairs by volatility, mixture-of-experts with a lightweight router, or pair embeddings that let a global model specialize smoothly. These directions are particularly relevant under distribution shift, where sharing statistical strength across "similar" pairs can improve generalization while preserving path-specific structure.

8. Related Work

RTT change detection and alignment with path changes. A central difficulty in latency-based monitoring is that observable RTT shifts do not map cleanly to underlying routing events. Shao *et al.* study this mismatch explicitly and propose a *one-to-one* matching formulation between RTT changepoints and path changes, together with evaluation principles that account for tolerance windows and measurement irregularities [Shao et al. 2017]. The accompanying implementation and extensions released by the authors provide a practical reference pipeline for RTT change detection and correlation analysis [Shao 2025]. Our work is complementary: rather than detecting changepoints and then correlating them with path events, we solve the supervised decision problem

posed by the CT-Mon/RNP challenge, predicting whether a route change occurred between consecutive measurements using features extracted from traceroute RTT samples.

RTT anomalies in large-scale time series. Several studies address detection and characterization of anomalous behavior in RTT time series at scale. Hou *et al.* propose methods to detect and characterize network anomalies from large RTT datasets, emphasizing robust detection under non-stationarity and heterogeneity across paths [Hou et al. 2021]. While these approaches typically frame the task as anomaly detection or unsupervised/semi-supervised change detection, our setting is supervised and strongly imbalanced, which motivates explicit threshold tuning and imbalance-aware training.

Statistical modeling of RTT distributions and path-delay regimes. Beyond point detection, prior work has modeled RTT behavior via probabilistic latent-state representations. Mouchet *et al.* propose nonparametric Hidden Markov Model (HMM) formulations for statistical characterization of RTTs [Mouchet et al. 2019], and extend these ideas to large-scale segmentation of Internet path delays using infinite HMMs [Mouchet et al. 2020]. These contributions reinforce that RTTs exhibit heterogeneous, non-Gaussian, and regime-switching behavior, motivating robust summaries (e.g., rolling deviations, quantiles) and flexible decision boundaries. In contrast to HMM-based modeling, we adopt a feature-based discriminative approach that is inexpensive to compute and suited to challenge-style supervised labels.

Inferring changepoints in unlabeled network time series. de Almeida *et al.* study changepoint inference in unlabeled time series collected from network diagnosis tooling, focusing on methods that can operate without ground-truth route-change labels [de Almeida et al. 2025]. This line of work targets operational settings where labels are scarce, whereas CT-Mon/RNP provides explicit binary supervision; our pipeline leverages this supervision to learn a direct decision rule, while still adopting distribution-free, robust feature construction inspired by the heterogeneity highlighted in prior studies.

Learning and data sources. Our classifier is based on gradient-boosted decision trees (XGBoost), a widely used and scalable learner for tabular features [Chen and Guestrin 2016], implemented and evaluated with standard machine-learning tooling [Pedregosa et al. 2011]. The measurements follow the traceroute probing paradigm and are consistent with public traceroute data documentation from M-Lab, which underpins the CT-Mon/RNP measurement context [Measurement Lab (M-Lab) 2025].

Positioning of this work. In contrast to prior approaches that first detect RTT changepoints and then attempt to align them with routing events, we directly learn a supervised per-interval route-change classifier under extreme class imbalance. Our method combines pair-centric temporal feature construction with a global tree-based learner, yielding a scalable solution that preserves path-specific baselines while avoiding the complexity of per-pair models or latent-state inference.

9. Conclusion

We presented a supervised, feature-based approach for detecting route changes from traceroute RTT measurements in the CT-Mon/RNP 2025 Data Challenge. By combining

pair-centric temporal feature engineering with a global XGBoost classifier, our method captures short-horizon RTT dynamics and stability patterns that are strongly associated with routing events, while remaining computationally lightweight and scalable. Experimental results show that the proposed pipeline achieves high discriminative performance under extreme class imbalance, with particularly strong recall for the route-change class, making it well suited for monitoring and alerting applications.

Beyond the challenge setting, our findings highlight the central role of abrupt RTT steps and deviations from short rolling baselines as robust indicators of path changes. Future work will investigate hybrid modeling strategies that combine global learning with pair-level specialization. We also plan to enrich the feature set with additional robust RTT summaries and auxiliary timing signals (e.g., TCP dynamics).

Reproducibility. All feature extraction, training, and evaluation scripts are publicly available in our repository.[3] The full pipeline can be executed from CSV inputs.

Acknowledgments. We gratefully acknowledge support from RNP scholarships awarded to all three authors for their second-place finish in the CT-Mon/RNP Kaggle competition. This work was partially supported by research grants from CNPq, CAPES, and FAPERJ E-26/204.268/2024.

References

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD 2016)*, pages 785–794.

Cheng, Y., Cardwell, N., Dukkipati, N., Jha, P., and Yeganeh, A. (2021). The rack-tlp loss detection algorithm for tcp. RFC 8985.

Cheng, Y., Chu, J., Radhakrishnan, S., and Jain, A. (2013). Proportional rate reduction for tcp. RFC 6937.

de Almeida, C. M., Leão, R. M., and de Souza e Silva, E. (2025). Inferring change points in unlabelled time series data collected from the network diagnosis tool. *Annals of Telecommunications*, pages 1–18.

Hou, B., Hou, C., Zhou, T., Cai, Z., and Liu, F. (2021). Detection and characterization of network anomalies in large-scale RTT time series. *IEEE Transactions on Network and Service Management*, 18(1):793–806.

Measurement Lab (M-Lab) (2025). Traceroute data documentation. `https://www.measurementlab.net/`. Accessed: 2025-10-13.

Mouchet, M. et al. (2019). Statistical characterization of round-trip times with nonparametric Hidden Markov Models. In *2019 IFIP/IEEE Symp. Integrated Network and Service Management*, pages 43–48.

Mouchet, M., Vaton, S., Chonavel, T., Aben, E., and Den Hertog, J. (2020). Large-scale characterization and segmentation of Internet path delays with infinite HMMs. *IEEE Access*, 8:16771–16784.

Paxson, V. and Allman, M. (2011). Computing tcp's retransmission timer. RFC 6298.

Pedregosa, F. et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Shao, W. (2025). RTT change detection and correlation with path changes. `https://github.com/WenqinSHAO/rtt`. GitHub repository, MIT License.

Shao, W., Rougier, J.-L., Paris, A., Devienne, F., and Viste, M. (2017). One-to-one matching of RTT and path changes. In *2017 29th International Teletraffic Congress (ITC 29)*, volume 1, pages 196–204. IEEE.

[3] `https://github.com/01tpaabr/ct_mon_route_change`