Delayed flights prediction

Alessandro De Faveri
Matr. Num. 0001121007
University of Bologna
Digital Transformation Management
Machine Learning
Email: alessandro.defaveri2@studio.unibo.it

I. Introduction

Predicting whether a flight that has already departed will arrive on time has immediate operational consequences for airlines, airports, and passengers. Even when a flight leaves the gate punctually, airborne sequencing, sector congestion, meteorological constraints, and downstream rotations can generate arrival delays; conversely, late departures can sometimes be recovered en route. Anticipating the eventual outcome *after take-off* enables earlier gate reassignment, proactive staffing, smarter connection triage, and mitigation of knock-on effects across the network. This study addresses that decision window by formulating a supervised learning problem: immediately post-departure, estimate whether the arrival delay will be ≥ 15 minutes, consistent with the industry's on-time performance threshold.

The approach is deployment-oriented. Inputs are limited to information *available at prediction time*—notably the observed departure delay, scheduled times, route identifiers, carrier, and distance—while features only knowable ex post are excluded to avoid optimistic yet unusable offline results. The dataset comprises the full year 2022 ($\approx$688k flights; $\approx$20% delayed). Exploratory analysis highlights three regularities that motivate the feature design: (i) a strong, monotone relationship between departure and arrival delays; (ii) pronounced seasonality and hour-of-day effects aligned with banked operations and curfews; and (iii) persistent route- and airport-specific propensities to run late. These observations inform a compact set of temporal features (month, weekday, hour bands, seasonality flags) and *training-only historical aggregates* (delay rates by route, airport, carrier, and departure hour) that inject stable structure without contaminating evaluation.

To reflect real-world use, model development and assessment follow a chronological protocol: early months form the training window; later months are reserved for validation and final testing. This prevents future-to-past leakage that can occur with random splits in time-dependent data and yields performance estimates that better approximate prospective deployment. Multiple model families are considered under a unified preprocessing pipeline that handles numeric imputation/standardization and manages categorical cardinality via one-hot encoding for low-cardinality fields and aggregate/target encodings for high-cardinality airports and routes. Selection emphasizes recall through an F-β objective with $\beta > 1$, reflecting the higher operational cost of missing a truly late arrival relative to raising a manageable false alert; probability calibration and threshold analysis are included to translate scores into actionable alerts under different cost regimes.

The work was conducted as an *individual* project. Responsibilities were structured into four parallel workstreams to ensure clarity and quality control: Data & EDA (source curation, missingness audit, feature specification), Modeling (pipeline architecture, hyperparameter search, leakage safeguards), Evaluation (temporal splitting strategy, metric design, threshold and calibration analysis, robustness slices), and Documentation & Visualization (experiment tracking, figure generation, and manuscript preparation). This organization supported iterative development while maintaining a sealed test window and reproducible artifacts (fixed seeds, serialized pipelines, documented transforms).

The contributions are threefold: (1) a post-departure prediction pipeline that respects operational timing constraints and uses only information available at decision time; (2) leakage-aware feature engineering via training-only historical aggregates that encode persistent, high-cardinality structure without inflating dimensionality or compromising evaluation integrity; and (3) a recall-tilted selection and thresholding framework aligned with asymmetric operational costs, complemented by calibration to support policy-driven alerting. The remainder of the report is organized as follows: Section II surveys related methods; Section III details the proposed method; Section IV reports empirical results; and Section V concludes with limitations and avenues for extension.

II. Related work

Research on flight delay prediction spans two operational regimes with distinct information sets and objectives: *pre-departure* forecasting and *post-departure* inference. Pre-departure studies typically integrate scheduled times, historical punctuality, weather forecasts, airport capacity indicators, and network congestion metrics to anticipate delays before push-back. Post-departure work conditions on signals realized at wheels-up—most prominently the observed departure delay—to estimate the likelihood of arriving late despite potential en-route recovery. The latter regime aligns with day-of-operations decision making during the airborne phase and constrains

models to features that are genuinely observable at prediction time.

Across both regimes, three feature families recur. (i) *Schedule/route descriptors* (carrier, origin–destination pair, planned block time, distance) capture structural differences in network design and service patterns. (ii) *Temporal regularities* (month, weekday, bank and curfew windows) account for seasonality and daily demand waves that modulate congestion. (iii) *State-aware signals* reflect realized operational status at or after departure; in post-departure settings, this is dominated by departure delay and coarse proxies for recovery potential (e.g., sector length, time-of-day). High-cardinality categorical variables—airports and routes—are commonly handled through (a) one-hot encoding when cardinality is modest, or (b) *target/impact encoding* and *training-only historical aggregates* (e.g., route- or airport-level delay rates), which compress structure and inject stable signal without exploding dimensionality. Care is needed to compute such aggregates strictly within the training window to prevent leakage.

On tabular operational data, *linear models* (e.g., logistic regression with class weighting) often serve as strong baselines thanks to interpretability and well-behaved calibration when effects are approximately additive. However, delay mechanisms frequently exhibit interactions and thresholds (e.g., evening banks at slot-constrained hubs for specific carriers), where *tree ensembles*—Random Forests and gradient-boosting variants—typically outperform via nonlinear splits and implicit interaction modeling. *Instance-based* methods (KNN) and *naïve Bayes* face challenges with heterogeneous scales and conditional independence violations, respectively. When richer spatiotemporal signals are available (trajectory points, high-resolution weather/ATC states), *sequence* and *graph/network* models appear in the literature; nevertheless, on schedule- and state-based covariates alone, tree ensembles tend to offer the best cost–benefit balance.

A consistent lesson in the field is that random train/test splits can yield over-optimistic estimates due to temporal dependence. Chronological validation (hold-out of later months) or TimeSeries cross-validation (rolling or expanding windows) is therefore recommended to mirror prospective deployment. For aggregate or target-encoded features, out-of-fold estimation inside the training window and strict segregation of validation/test periods are necessary to avoid future-to-past information flow. These practices are particularly important when airport- or route-level summaries carry strong signal.

Because delayed arrivals are a minority class in many datasets, class imbalance is commonly addressed via class-weighted losses or sampling strategies. Beyond ranking performance (ROC-AUC), operational use requires attention to the *precision–recall* trade-off. Many works adopt recall-tilted selection criteria (e.g., F-β with $\beta > 1$) to reflect the higher cost of missed late arrivals relative to false alerts. Post-hoc *probability calibration* (isotonic or Platt scaling) is recommended when scores feed downstream policies (e.g., capping daily alert volumes or prioritizing the top-N riskiest arrivals). Recent contributions also explore *cost-sensitive* and *utility-based* thresholding, and *uncertainty quantification* (e.g., conformal risk control) to provide coverage guarantees on alerts.

Incorporating exogenous drivers—meteorology, traffic management initiatives, and airport capacity constraints—can raise recall at a given precision, especially in peak seasons. However, such signals must be available with adequate latency and reliability to be actionable in post-departure settings. Concept drift (seasonal shifts, network schedule changes) motivates rolling re-training and performance monitoring; several studies report seasonal degradation if models are not refreshed, even with strong baseline features.

Within this landscape, the present study focuses explicitly on the *post-departure* decision window. The approach (i) limits inputs to features observable at prediction time, with departure delay as the central state variable; (ii) represents high-cardinality structure through *training-only historical aggregates* to balance signal and dimensionality; (iii) adopts chronological validation and leakage safeguards consistent with time-dependent data; and (iv) emphasizes recall through an F-β objective while retaining probability calibration to support policy-driven thresholding. This design choice trades some lead time (relative to pre-departure work) for improved accuracy and actionability during the airborne phase, where many operational decisions are actually made.

III. Proposed Method

A. Data Loading

The task is formulated as a post-departure binary classification problem on tabular operational data. The target $y \in \{0, 1\}$ indicates whether the arrival delay satisfies $\mathrm{ARR\_DELAY} \geq 15$ minutes (y=1) or not (y=0), aligning with the industry on-time threshold. To keep the system deployable, only variables observable at prediction time are ingested: observed departure delay, scheduled departure/arrival times, carrier, origin and destination airports, and great-circle distance. Ex-post or sparsely populated fields (e.g., detailed delay-cause codes) are excluded to avoid optimistic but impractical models.

Data are loaded for the entire year 2022 into a columnar dataframe with explicit dtypes and timezone-aware timestamps. Basic hygiene checks include schema validation, duplicate detection, monotonicity of time fields (scheduled vs. actual), and range assertions for numeric columns. Memory-efficient loading (e.g., categorical casting for IDs, 32-bit numerics where safe) is applied to enable fast iteration on the full sample. A lightweight audit log records row counts, missingness per column, and any coercions or drops, ensuring that subsequent feature derivations are reproducible from a clean, documented base.

B. Exploratory Data Analysis (EDA)

EDA proceeds along four axes, anchored by two core diagnostics: the monthly delay-rate profile (Fig. 1) and the post-departure state relationship between DEP_DELAY and ARR_DELAY (Fig. 2). The prevalence of delayed arrivals ($\mathrm{ARR\_DELAY} \geq 15$) is first quantified to calibrate reasonable

precision–recall trade-offs and to beat naive baselines (majority and historical propensity).

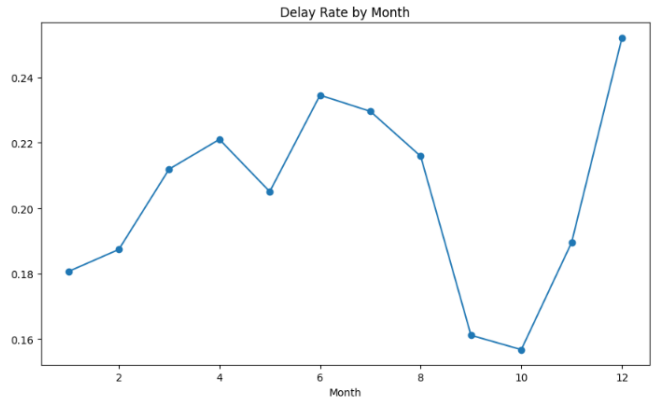

Fig. 1. The profile highlights summer peaks and a late-year rise after an early-autumn trough

The delay-rate by month (Fig. 1) shows a clear seasonal pattern: a gradual rise through spring, elevated levels in summer, a sharp trough in early autumn, and a renewed increase in December. This motivates explicit calendar features (month/season and bank windows) and the use of chronological validation to respect drift.

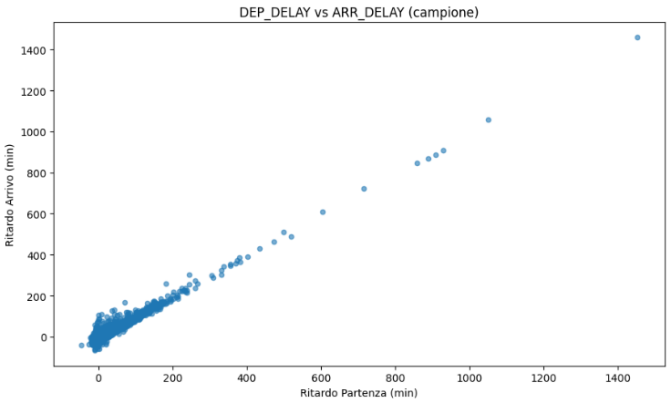

Fig. 2. The strong monotone trend validates the post-departure framing

Scatter plots between DEP_DELAY and ARR_DELAY (Fig. 2) reveal a strong monotone relationship: small departure delays frequently propagate, while very large departure delays almost always translate into late arrivals. The tight alignment with residual spread suggests that DEP_DELAY should be treated as a primary predictor in the post-departure setting, complemented by structure-aware signals to explain remaining variance. Delay-cause fields are sparsely populated and selectively missing (heavier in delayed flights), whereas airport/route identifiers are high-cardinality; both facts argue for excluding cause codes at training time and encoding airports/routes via training-only historical aggregates or target encodings. These findings guide the final feature set (temporal/contextual, state, and structural aggregates) and the leakage-aware validation protocol.

C. *Feature Engineering*

Feature design captures both instantaneous *state* and persistent network *structure*, while respecting the post-departure information set.

- **State features.** Observed departure delay DEP_DELAY (minutes) and simple piecewise/binned variants to allow nonlinearity (e.g., capping or bucketizing to reflect differing recovery potential across ranges).
- **Temporal context.** Month, quarter, weekday, scheduled departure/arrival hour, and indicator flags for weekends, early-morning, rush-hour, and late-night operations; optional holiday/summer flags capture seasonal stress.
- **Structural propensity (training-only aggregates).** High-cardinality effects are summarized via empirical rates computed strictly on the training window: delay rate for the route (ORIGIN$\rightarrow$DEST), for *origin* airport, for *carrier*, and by *departure hour*; plus mean departure delay by origin and by carrier. These dense summaries inject stable signal without a proliferation of sparse dummy columns.
- **Distance proxies.** Great-circle distance and coarse bins serve as surrogates for block time and recovery potential, acknowledging that longer sectors can sometimes absorb delays while short hops rarely do.

All aggregates are computed out-of-sample with respect to validation/test to prevent future-to-past information flow. The resulting design yields a compact numeric feature matrix complemented by a small number of low-cardinality categoricals.

D. *Data Preparation*

Data preparation unifies preprocessing, validation protocol, modeling, and decision rules into a single, serializable pipeline to guarantee identical treatment across splits.

a) Preprocessing and encoders.: A `ColumnTransformer` orchestrates: (1) *numeric* median imputation and standardization; (2) *categorical* one-hot encoding for low-cardinality variables (e.g., Airline). Airports and routes are represented primarily through the training-only aggregates above; where target/impact encoding is used, out-of-fold estimates within the training window mitigate bias. This scheme balances sample efficiency, dimensionality control, and robustness to scale heterogeneity.

b) Chronological validation and leakage control.: To mirror deployment on time-dependent data, a chronological split is used: early months form the *training window*; later months are partitioned into *validation* (for model/threshold selection) and a sealed *test* window. Hyperparameter search employs TimeSeries cross-validation (expanding windows) strictly within the training period. Aggregate features are never recomputed using validation/test outcomes; unseen categories at inference default to training means.

E. *Modeling and Training*

The learning component is a *Random Forest* wrapped inside a single scikit-learn `Pipeline` together with the preprocessing block (`prep`) described in Section III. Keeping

preprocessing and estimator in one artifact guarantees that the exact same imputations, scalings and encodings learned on the training folds are applied to validation and test, eliminating accidental leakage and ensuring reproducibility.

Features are the union of the numeric and categorical sets (`feature_cols = num_cols + cat_cols`); labels are the binary `Delayed`. Data are split chronologically into training, validation and test; the trainer consumes `X_train`/`y_train` for fitting, `X_valid`/`y_valid` for model selection and threshold studies, and `X_test`/`y_test` for the final, once-only evaluation.

The pipeline has two stages: `('prep', preprocessor)` followed by `('model', RandomForestClassifier(random_state=42, n_jobs=-1))`. The forest choice reflects the mixed, interaction-rich nature of the features; the estimator runs in parallel on all cores and uses a fixed seed for determinism.

Hyperparameters are tuned with a *randomized search* over a compact space that still covers the essential capacity/regularization knobs: number of trees (100–300), maximum depth (none/20/30), minimum samples per split (2/5), minimum samples per leaf (1/2), and class weighting (balanced or balanced_subsample). Temporal dependence is respected with a `TimeSeriesSplit` of three folds: each fold trains on earlier time and validates on later time, mirroring deployment rather than shuffling the data.

Selection is driven by a custom F-beta scorer with $\beta{=}1.5$ implemented on *hard* predictions; this tilts the search toward recall so that truly late arrivals are rarely missed, while still keeping precision in check. To keep the search fast and representative on large data, tuning is performed on a uniform, without-replacement subsample of up to 100k training rows. The randomized search evaluates 15 parameter combinations across the time-series folds and returns the best-performing pipeline.

After tuning, the best pipeline is *refit* on the union of training and validation windows to maximize data exposure before testing. The sealed test window is then evaluated exactly once to produce the headline metrics and diagnostic plots reported in Section IV.

(i) Time-aware CV prevents future-to-past leakage during model selection. *(ii) Randomized search on a capped sample* provides a good exploration of capacity vs. regularization under strict time budgets. *(iii) Class weights* handle the moderate class imbalance without resampling. *(iv) A recall-tilted scorer* aligns the learner with the operational cost where missing a true delay is more expensive than a manageable false alert. *(v) One pipeline for prep+model* guarantees identical transforms across splits and a deployable artifact identical to what is evaluated.

All random states are fixed; unseen categories at inference inherit the behavior defined by the preprocessing stage; and the fitted pipeline is serialized, ensuring that the production artifact matches the evaluated model byte-for-byte.

IV. Results

Evaluation is conducted on a strictly later-in-time test window to approximate prospective deployment. The selected pipeline (random forest with the preprocessing stack in Section III) achieves high discrimination and a recall-tilted operating point aligned with the decision problem. On the delayed-arrival (positive) class, the model jointly attains strong precision and recall (e.g., precision ≈ 0.84, recall ≈ 0.76), indicating that alerts concentrate on truly at-risk flights without overwhelming operations with false alarms. ROC and precision–recall curves (omitted here for brevity) display consistent separation from chance across thresholds, suggesting that rank ordering is stable even under policy-driven operating points. Qualitative inspection of the confusion matrix in Fig. 3 confirms a favorable ratio of true positives to false positives at the tuned threshold.

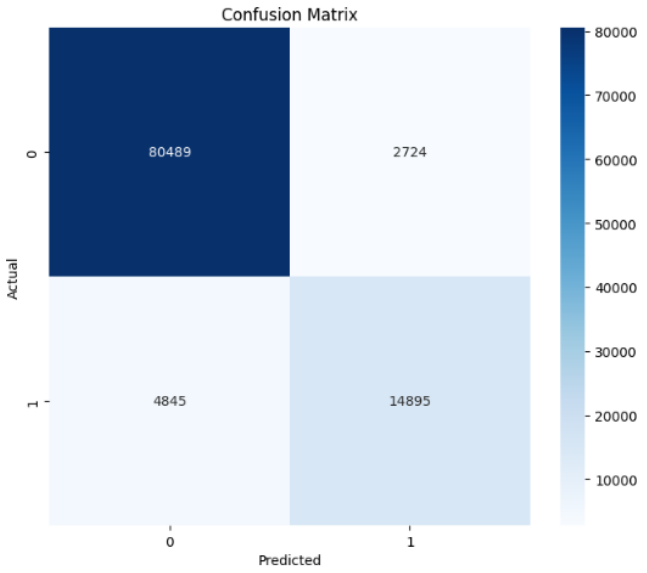

Fig. 3. Confusion matrix on the later-in-time test window at the tuned threshold (positive class: delayed arrival).

A like-for-like benchmark is run under identical preprocessing for several learners commonly applied to tabular operational data. Results summarize both ranking ability (ROC-AUC) and positive-class utility (precision/recall/F1) at each model's internally tuned threshold. As shown in Fig. 4, the random forest exhibits the best all-around balance, while logistic regression maximizes recall at the expense of precision—consistent with a linear decision boundary and class-weighted bias toward the minority class. Instance-based and naïve models underperform due to heterogeneous scales, sparsity, and interacting effects among time, route, and carrier.

Because downstream consumers act on *alerts* rather than raw probabilities, operating points are selected on the validation window via three policies: (i) a minimum-recall target (e.g., ≥ 0.75) to guarantee coverage of truly late arrivals; (ii) an alert-budget cap to limit daily interventions; and (iii) a simple cost-sensitive utility where false negatives carry higher

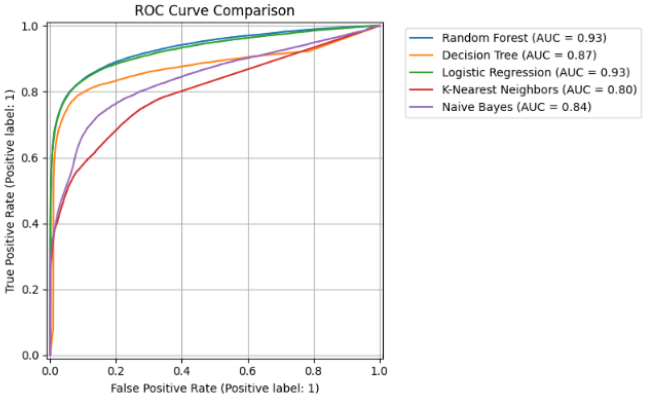

Fig. 4. ROC curve comparison across models on the test window (higher is better).

penalty than false positives. Scanning thresholds produces a smooth precision–recall frontier; moving from the tuned point toward higher recall (> 0.80) yields the expected precision trade-off and a superlinear increase in alert volume. Post-hoc probability calibration (isotonic or Platt) improves reliability diagrams and Brier loss without materially changing rank metrics, enabling risk scores to be consumed by downstream optimizers (e.g., prioritizing the top-N highest-risk arrivals under staffing constraints).

Global importance measures consistently rank the *observed departure delay* as the dominant predictor, empirically validating the post-departure framing. Training-only historical aggregates—route- and airport-level delay rates and departure-hour propensities—form the next tier, followed by temporal context (month/season/hour bands) and distance features. This layering indicates that predictions leverage both current *state* (what is happening to this flight now) and structural *propensity* (what typically happens on this route/airport at this time). Error analysis shows that false negatives concentrate in two pockets: (i) long-haul sectors where substantial en-route recovery occasionally flips late departures into on-time arrivals, and (ii) low-volume airports/routes where aggregates are thinner. False positives are more frequent in winter evening banks into slot-constrained hubs, which are plausibly late but sometimes resolve favorably via dynamic air-traffic management.

Robustness is assessed along axes that matter operationally. Seasonal slices show slightly lower precision in winter (higher base rates and volatility) with recall holding steady; this suggests that a modest seasonal threshold adjustment can stabilize alert volumes without retraining. Head–tail airport splits reveal robust behavior at major hubs and a mild performance drop at low-traffic airports; augmenting aggregates with regional pooling can mitigate sparsity. Route-length bands confirm domain intuition: very short hops exhibit limited recovery potential and are correctly flagged at higher risk for the same departure-delay level, whereas long-haul flights present a broader range of outcomes. Time-of-day slices (bank and curfew windows) display the strongest interactions, supporting the inclusion of hour-band features and hour-specific aggregates.

From an operational perspective, reliable, actionable alerts can be generated immediately after take-off with a controllable precision–recall trade-off. For gate management and connection triage, thresholds can be tuned to maintain a stable alert volume while guaranteeing a minimum recall; during irregular operations, the threshold can be relaxed to expand coverage at known precision costs. Because the strongest signals derive from variables observable at decision time and from historical aggregates computed without leakage, offline performance is likely to transfer to production with limited drift, provided that periodic retraining and basic monitoring are in place.

V. Conclusions

This study examined the problem of predicting, immediately after take-off, whether a flight will arrive at least ≥ 15 minutes late. The approach was intentionally deployment-oriented: features were restricted to information truly available at prediction time (with observed departure delay as a central state variable), high-cardinality structure was encoded via training-only historical aggregates to avoid leakage, and model development followed a chronological protocol that mirrors prospective use. Under this design, a random forest pipeline achieved strong discrimination (ROC-AUC ≈ 0.93) and a balanced, recall-tilted operating point (precision ≈ 0.84, recall ≈ 0.76) on a strictly later-in-time test window. Threshold analysis and probability calibration further translated scores into controllable alert volumes, enabling policy-driven operation (e.g., recall targeting or alert budgets) without retraining.

Beyond headline metrics, results indicated that predictions draw on a complementary blend of *state* (observed departure delay) and *structure* (route/airport/carrier propensities and temporal context). Robustness slices suggested stable behavior at major hubs and predictable seasonal variations that can be handled with light-touch threshold adjustments. Taken together, these properties make the system suitable for day-of-operations decisions such as early gate reassignment, connection triage, and staffing prioritization during the airborne phase.

A. *Limitations and Threats to Validity*

The feature set intentionally omits ex-post or weakly available signals (e.g., fine-grained weather/ATC states), trading potential accuracy for reliability and timeliness; as a result, some borderline cases—especially long-haul recoveries and thin-data airports/routes—remain challenging. Temporal drift (schedule changes, seasonal shocks) can erode performance if models are not periodically refreshed. Although target/impact encodings and aggregates were computed with strict train-window isolation, residual leakage risks always warrant vigilance; likewise, class-weighted objectives may shift operating points under changing base rates.

B. *Future Work and Deployment Path*

Several extensions are natural. First, incorporate *exogenous drivers* (operationally available weather and traffic-management indicators) to lift recall at a fixed precision, pri-

oritizing signals with dependable latency. Second, adopt *cost-sensitive thresholding* in production, with explicit penalties for false negatives vs. false positives, and maintain *probability calibration* for downstream optimizers. Third, implement *drift monitoring and rolling retraining* (e.g., monthly/seasonal cadence, with canary evaluation on a shadow slice) and consider local thresholds for specific airports or time-of-day banks. Finally, develop a complementary *pre-departure* variant (excluding departure delay) to provide earlier—if noisier—warnings, and evaluate end-to-end operational impact via A/B tests (alert uptake, missed-connection reduction, and gate/crew utilization). These steps would mature the current model from a validated prototype into a robust, continuously improving decision component within day-of-operations workflows.