SvedbergOpen
DISSEMINATIONOFKNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

Predictive Fare and Schedule-Change Alerts Using Event-Driven Cloud Architectures with Language-Model Explanations

Vamseekrishna Kasturi[1], Ranjith Kumar Peddi[2], Rajesh Reddy Ellenki[3], Venkateswara Rao Movva[4]

1. Development Manager Email: vamsee.k.kasturi@gmail.com
2. Principal Software Engineer Email: peddi.ranjithk@gmail.com, ORCID ID: 0009-0003-4223-8578
3. Sr. Staff Software Engineer Email: rajeshreddyellenki@gmail.com, ORCID ID: 0009-0009-4975-9708
4. Software Engineer III, Email: movvavenkateswaraa@gmail.com, ORCID ID: 0009-0007-6905-1999

Abstract
Real-time fare and schedule-change forecasting being essential for air travelers, a predictive-alerting system on an event-driven cloud architecture is proposed and implemented, predicting fare changes using flight demand and supply and schedule changes, delays, and cancellations using historic and forecasted weather factors. A deep-learning benchmark predictive model is compared and found lacking. Predictions are validated with key indicators in an online dashboard integrated with the alerting system. Results reveal a complex interplay between the three target variables and may prove useful in pre-emptively identifying critical situations. Alerts triggered by the predictions are forecasted and tested, showcasing the value of the end-to-end setup. A novel add-on combines the alert system with an open-source language model to increase user confidence.
Effective forecasting and alerting improve users' travel experience and optimize operators' resource allocation. The pairing of the alerting function with language-model explanations enhances passenger reassurance, providing clear connections between alerts, feature importance, and forewarning. The architecture pattern allows open-ended integration of any service with predictable, timely changes. Recent advances in cloud technology dynamic pricing, infrastructure as code, serverless computing, and low-cost weather data, images, and predictions from space make real-time prediction and alerting economically viable for various markets.

Keywords: Predictive Analytics, Real-Time Forecasting, Airline Fare Prediction, Flight Schedule Monitoring, Event-Driven Architecture, Cloud Computing, Deep Learning, Intelligent Alerting, Weather-Based Prediction, Large Language Models.

1. Introduction

The contexts for establishing travel fare and disruption forecasting differ. Airlines and hotels typically adopt the strategy of lowering fares just before departure or full occupancy in order to increase demand, causing many of the price and schedule changes. Amusement parks and event organizers, on the other hand, tend to set ticket prices higher in the very beginning and reduce them afterwards, as the capacity is typically hard to change. This is similar to products in the retail market. Moreover, unlike for air tickets, prices in the hotel market can be characterized by strong daily seasonality. Therefore, predictive modeling for fares that supports timely alerts for price depreciation should provide price increase warnings at times of significant demand growth. Airlines, however, are often required to provide a service to travelers even when the service is not profitable for them.

Scheduling prediction within transportation systems has often focused on major schedule disruptions caused by heavy weather, storm, or other factors that might be used to affect the scheduling. The state of the key weather elements as well as major national holidays have considerable effects on the occurrence of delays. For event-organized attractors such as sporting events or music concerts, a holiday on the day of the event typically increases the number of participants involved. Modeling the occurrence of major schedule disruptions by exploring the different factors that can have an impact can help alert against upcoming disruptions.

2. Problem Statement and Research Questions

Fare and schedule-change alert systems offer important travel assistance to ux`x`sers when preparing for upcoming trips. However, many failure cases exist and can be improved by the use of predictive analytics. Most systems currently use artificial intelligence as a black box to determine alerts, making users less likely to follow their suggestions. Users would be more likely to trust alerts if the underlying artificial-intelligence model could be explained.

The main research problem is defined as follows: Predictive fare and schedule-change alerts for users of multimodal transit networks can be offered with additional vigilance for potential failures, thereby enhancing the perceived reliability of advisories. This outcome has particular value for operators and agencies responsible for fare pricing and service scheduling in public transport systems; thus, agency-focused fare and schedule-change predictions are also useful. Intelligent systems suggest actions on behalf

of users, and when these suggestions are known to generate false or misleading alerts, the risk is often perceived as greater than the potential value. Research aims to mitigate various types of failure cases associated with predictive fare and schedule change alerts without adding significant burden to system design and operation.

2.1. Azure Event-Driven Architecture

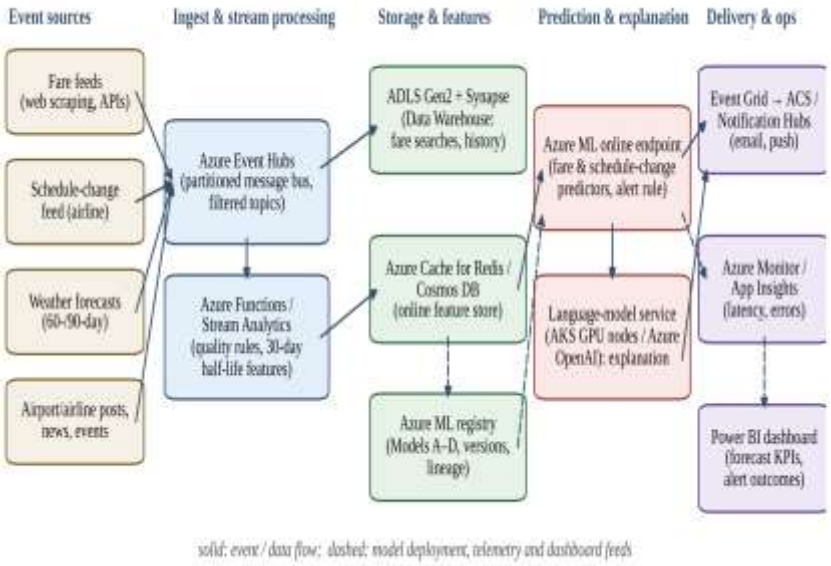

Fig. 1: Azure event-driven architecture for predictive fare and schedule-change alerts. Event sources feed Event Hubs; stream listeners build features; Azure ML scores events; a language-model service explains alerts; Event Grid and Communication Services deliver them

TABLE I: AZURE ARCHITECTURE MAPPING AND SIMULATION PARAMETERS

Stage	Azure service	Role in pipeline	Service-time model	Provisioning
Ingest	Event Hubs (Standard)	Partitioned, filtered message bus	Log-normal, median 20 ms, $\sigma = 0.35$	Throughput units; no queue
Stream / features	Functions or Stream Analytics; Redis / Cosmos DB	Quality rules, half-life features, feature lookup	Log-normal, median 12 ms, $\sigma = 0.3$	c_{str} workers, Eq. (26)
Scoring	Azure ML managed online endpoint	Fare and schedule predictors, alert rule	Measured inference (A 0.02, B 1.7, C 3.8, D 13.7 ms) + 6 ms	c_{scr} instances, Eq. (26)
Explanation	Open-source LM on AKS GPU nodes, or Azure OpenAI	Natural-language rationale (Model D only)	Log-normal, median 1.6 s, $\sigma = 0.25$; fresh share 2% of events	4 slots per node
Delivery	Event Grid; Communication Services / Notification Hubs	Email and push alerts	Log-normal, median 90 ms, $\sigma = 0.4$	Elastic; no queue
Batch baseline	Data Factory + Synapse	Hourly scheduled scoring	Uniform wait 0–60 min + 4-min run	Single pipeline
Unit prices (illustrative)	USD per hour	Stream worker / scoring instance / GPU node	$0.10 / $0.20 / $2.50	Event Hubs $0.03 per TU-h + $0.028 per M events; fixed $0.40/h

TABLE II: DATASET AND PREDICTION-TASK SUMMARY

Attribute	M1	M2	M3	M4	M5
Route(s)	NYC–LAX	NYC–LAX	NYC–LAX	12 Singapore routes	12 Singapore routes
Target	Daily avg. fare	Changes in next 7 d	Changes in next 60 d	Daily fare	Daily fare
Horizon (days)	60	7	60	60	90
Training rows, raw	1,001	1,054	1,001	12,012	11,652
Training rows after quality rules	934	1,054	1,001	11,336	10,990
Test rows (2023–2025)	1,036	1,089	1,036	12,432	12,072
Number of features	17	9	9	30	30
Alert definition	±10% vs. current fare	≥ 7 changes in 7 d	≥ 55 changes in 60 d	±10% vs. current fare	±10% vs. current fare
Alert prevalence (test)	41.5%	27.7%	30.5%	42.1%	44.4%

3. Background and Related Work

The offer to send real-time fare-change alerts (Ahr & Matzler, 2021) for an intercity-bus corridor was described in an invitation to bid, with the winning provider proposing to implement a two-stage predictive model: one for customer decision-making regarding the purchase of tickets and the other for estimating the effect of fares on ticket demand. Ahr and Matzler (2022) went a step further by integrating the predictive model into a Winter Park, Colorado, multi-attraction-logistics system that also monitored the remaining-time feature and issued regret-of-inactivity alerts. Winter Park's weather information, snow quality on slopes, historical ticket demand and pricing, slope-operating schedules, and trip durations to-and-from Denver collectively formed an environment for real-time alerts sent via an event-driven architecture. Previous research, however, has not addressed the one-sided scalability of predictive fare- and schedule-change alert systems, nor real-time alerts for schedule changes.

Prior systems have gradually integrated the event-driven architectural paradigm into transit systems. Cornelis et al. (2019) used cyber-physical systems to provide real-time updates in a vehicle-sharing system; Wong et al. (2022) offered ride-booking services using a decentralised buffer-and-spare-pooling system. Nann et al. (2019) surveyed the state of the art in smart transportation systems and pointed to the lack of real-time information associated with event-driven architectures. Liang et al. (2017) suggested building rapid-response event-driven systems with supportive solution providers that analysed signals sent by real-world transit systems under various weather and operational conditions. At a broader level, Cloud Computing and recommended architectures for Cloud-based systems, the Internet of Things (IOT) for Cars and the Transportation Cloud for the Transportation Systems Architecture are rapidly evolving towards event-driven frameworks, pattern-matching, event-comparison and eventaggregation engines, Application Programming Interfaces for on-line event-stream management, predictive modelling and all forms of query processing.

3.1 Feature Engineering and Data Quality

For a forecast origin t and horizon h, every model maps a feature vector built only from information available at t to a prediction of the target at t + h, Eq. (1).

$$\hat{y}_{t+h} = f_\theta(\mathbf{x}_t) \qquad (1)$$

where $\hat{y}_{t+h}$ is the predicted daily average fare (tasks M1, M4, M5) or schedule-change count (M2, M3), and f_θ is the learner with parameters θ. The horizons are h = 60 d (M1, M3, M4), 7 d (M2) and 90 d (M5).

The feature vector in Eq. (2) concatenates the inputs named in [1]:

$$\mathbf{x}_t = [\mathbf{c}_{t+h}, \mathbf{w}_{t+h|t}, \mathbf{e}_{t+h}, \tilde{\mathbf{s}}_t] \qquad (2)$$

where c_{t+h} holds calendar and holiday descriptors of the target day (day of week, month, seasonal phase, holiday ramp), $w_{t+h|t}$ the weather forecast for the target day issued at t, e_{t+h} the scheduled local-event counts, and $\tilde{s}_t$ the decayed temporal signals of fares, airport and airline posts and news.

The temporal signals use the 30-day half-life window of [1], Eq. (3).

$$\tilde{s}_t = \frac{\sum_{k=0}^{K} 2^{-k/T_{1/2}}\, s_{t-k}}{\sum_{k=0}^{K} 2^{-k/T_{1/2}}} \qquad (3)$$

where s_{t-k} is the raw daily signal k days before t, K is the look-back, and $T_{1/2}$ = 30 days is the half-life, so a signal's weight halves every 30 days. In the pipeline this is the recursive exponentially weighted average with smoothing factor $\alpha = 1 - 2^{-1/30} \approx 0.023$, which a stream listener can update in constant time per event.

Training rows must satisfy the fare quality rule of [1], Eq. (4), and schedule-change records the rule in Eq. (5).

$$\mathcal{D}_{\text{fare}} = \{t:\ |\Delta f_t| < \delta_q,\ |d_t - \tau_t| \le 3\} \qquad (4)$$

where $\Delta f_t = f_t - f_{t-1}$ is the change between successive fare entries, δ_q = \$50 is its maximum, d_t is the travel date and τ_t the departure date. At least n_{min} = 200 admissible rows are required for training. For the 12-

route Singapore panel δ_q is the same \$50 rule scaled to 17.2% of each route's mean fare.

$$\mathcal{D}_{\text{sch}} = \{i:\ \text{SC}_i = 1,\ |\Delta d_i| < 15\} \qquad (5)$$

where SC_i is the Schedule_Change flag (1 when a change occurred) of record i and Δd_i is the shift in days. Applied to the simulated feeds, Eq. (4) removes 6.7% (M1), 5.6% (M4) and 5.7% (M5) of the training rows, and Eq. (5) retains 2,106 of 2,204 change records (95.6%).

3.2 Forecast-Error Metrics

Forecast accuracy is measured by the mean absolute error, root-mean-square error and mean absolute percentage error over n test samples, Eqs. (6)–(8).

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \qquad (6)$$

where y_i is the realised value and $\hat{y}_i$ the forecast. MAE is in the units of the target (USD or changes).

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \qquad (7)$$

RMSE penalises large misses more heavily than MAE, which matters when a single missed holiday surge produces an alert far too late.

$$\text{MAPE} = \frac{100}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{y_i} \qquad (8)$$

MAPE is scale-free, so it allows comparison across routes whose mean fares differ by an order of magnitude. It is reported for fare tasks only, because counts include zeros.

3.3 Alert Rule and Classification Metrics

A fare alert compares the forecast with the current fare through the relative change in Eq. (9), and classifies it with the rule in Eq. (10).

$$r_t = \frac{\hat{y}_{t+h} - f_t}{f_t} \qquad (9)$$

where f_t is the current fare, which is also what the user sees when the alert fires.

$$a_t = \begin{cases} +1 & r_t \geq \delta \\ -1 & r_t \leq -\delta \\ 0 & \text{otherwise} \end{cases} \qquad (10)$$

where $\delta = 0.10$ is the alert threshold. $a_t = +1$ is a price-rise alert (buy now), −1 is a price-drop alert (wait), and 0 is no alert. For schedule tasks an alert fires when the predicted count exceeds a cut-off chosen on a chronological validation slice to maximise F_1; the reference event is a count above the training 80th percentile (7 changes in 7 days, 55 in 60 days).

Alert quality is scored with precision and recall in Eq. (11), their harmonic mean in Eq. (12), and, for the three-class fare alerts, the macro average in Eq. (13).

$$P = \frac{TP}{TP + FP}, \qquad R = \frac{TP}{TP + FN} \qquad (11)$$

where TP, FP and FN are true positives, false positives and false negatives. Precision governs alert fatigue, and recall governs missed disruptions.

$$F_1 = \frac{2PR}{P + R} \qquad (12)$$

F_1 balances the two and is the primary alert metric in this paper.

$$F_1^{\text{macro}} = \frac{1}{3}\sum_{c\in\{-1,0,+1\}} F_{1,c} \qquad (13)$$

where $F_{1,c}$ is the F_1 score of class $c \in \{-1, 0, +1\}$. Macro averaging prevents the large "no alert" class from masking weak rise or drop detection.

3.4 Traveller Decision Model

Following [1], a traveller chooses among purchase, wait and abandon in a partially observable setting [3]. The belief over the hidden state (fare regime, remaining time, schedule status) is updated after an action a and observation o as in Eq. (14).

$$b'(s') = \eta\, O(o \mid s', a) \sum_{s} T(s' \mid s, a)\, b(s) \qquad (14)$$

where b is the belief, T(s′|s, a) the transition model, O(o|s′, a) the observation model (alerts and quotes are observations), and η a normalising constant.

The value of a belief follows from the action value in Eq. (15) and the optimal value in Eq. (16).

$$Q(b, a) = \sum_{s} b(s) R(s, a) + \gamma \sum_{o} \Pr(o \mid b, a)\, V^*(b') \qquad (15)$$

where R(s, a) is the reward (the negative price paid for a purchase), γ the discount factor, and b′ the belief updated after action a and observation o by the previous equation.

$$V^*(b) = \max_{a \in \mathcal{A}} Q(b, a) \qquad (16)$$

where the action set is {purchase, wait, abandon}. Because the belief space is continuous, the experiments use the observable special case below, in which a flexible traveller either books now or waits h days.

The average saving obtained by following alerts is Eq. (17), with the price actually paid defined by Eq. (18).

$$S_{\text{follow}} = \frac{1}{n} \sum_{t} (f_t - \pi_t) \qquad (17)$$

where n is the number of decision points and f_t the fare if the traveller books immediately.

$$\pi_t = \begin{cases} f_{t+h} & a_t = -1 \\ f_t & \text{otherwise} \end{cases} \qquad (18)$$

so the traveller waits only when a price drop is predicted and otherwise books now.

Because users act on alerts only if they trust them, the expected saving scales with a follow probability τ, Eq. (19). The extra cost of explanations is justified if trust rises by at least the break-even amount in Eq. (20).

$$\bar{S}(\tau) = \tau \, S_{\text{follow}} \qquad (19)$$

where τ ∈ [0, 1] is the share of travellers who follow an alert, and the remainder book immediately.

$$\Delta\tau^* = \frac{\Delta C_{\text{alert}}}{N_u \, S_{\text{follow}}} \qquad (20)$$

where ΔC_{alert} is the additional infrastructure cost per explained alert and N_u the travellers notified by each alert (N_u = 100 here).

3.5 Latency and Queueing Model

The end-to-end latency of an alert is the sum of the stage times in Eq. (21).

$$L_{\text{e2e}} = L_{\text{ing}} + L_{\text{str}} + L_{\text{scr}} + L_{\text{lm}} + L_{\text{del}} \qquad (21)$$

where the terms are ingestion (Event Hubs), stream processing, scoring, language-model explanation and delivery. L_{lm} is zero when an alert is sent without a fresh explanation. Each stage except ingestion and delivery is a multi-server queue.

A stage with c parallel servers, arrival rate λ and per-server service rate μ has the utilisation in Eq. (22). Under Poisson arrivals the mean queueing delay follows the Erlang-C model [4] in Eqs. (23) and (24).

$$\rho_i = \frac{\lambda_i}{c_i \, \mu_i} \qquad (22)$$

where $\rho_i \to 1$ marks saturation, at which queues, and hence latency, grow without bound.

$$W_q = \frac{C(c, a)}{c \, \mu - \lambda} \qquad (23)$$

where $a = \lambda/\mu$ is the offered load in Erlangs.

$$C(c, a) = \frac{\frac{a^c}{c!} \cdot \frac{c}{c - a}}{\sum_{k=0}^{c-1} \frac{a^k}{k!} + \frac{a^c}{c!} \cdot \frac{c}{c - a}} \qquad (24)$$

which is the probability that an arriving event must wait. The simulation in Section V uses the same queue discipline and adds log-normal service times.

Alert freshness is governed by the service-level objective in Eq. (25), and the autoscaler sizes each stage with Eq. (26).

$$\Pr(L_{\text{e2e}} \le L_{\text{SLO}}) \ge 0.95 \qquad (25)$$

where L_{SLO} = 5 s is the freshness target for an explained alert.

$$c_i^* = \left\lceil \frac{\lambda_i \, \bar{s}_i}{\rho_{\text{target}}} \right\rceil \qquad (26)$$

where $\bar{s}_i$ is the mean service time of stage i and ρ_{target} = 0.70, leaving headroom for bursts such as fare-sale announcements.

3.6 Cost and Optimisation

The hourly run-cost of the pipeline is Eq. (27), and capacity planning minimises it subject to the utilisation and latency constraints in Eq. (28).

$$C_h = C_{\text{eh}} + \sum_{j \in \{\text{str,scr,gpu}\}} n_j \, p_j + C_0 \qquad (27)$$

where C_{eh} is the Event Hubs charge (throughput units plus per-event ingress), n_j the number of stream workers, scoring instances or GPU nodes, p_j their hourly price, and C_0 fixed storage and monitoring cost.

$$\min_{n_{\text{str}}, n_{\text{scr}}, n_{\text{gpu}}} \quad C_h \qquad \text{s.t.} \quad \rho_i \le \rho_{\max} \;\; \forall i, \;\; L_{95} \le L_{\text{SLO}} \qquad (28)$$

where $\rho_{\max}$ = 0.85 and L_{95} is the 95th-percentile end-to-end latency. The integer solution at each load is the autoscaling rule above.

Cost-effectiveness is summarised by the efficiency ratio in Eq. (29) and the cost per correct alert in Eq. (30), and a composite score trades quality against cost and latency in Eq. (31).

$$\eta = \frac{F_1}{C_{1M}} \qquad (29)$$

where C_{1M} is the cost per million events and F_1 the alert F_1 score of task M1.

$$C_{TP} = \frac{C_h}{3600\,\lambda\,\alpha\,P} \qquad (30)$$

where λ is the event rate, α the fraction of events that raise an alert, and P the alert precision. This metric rewards volume, so it must be read together with precision.

$$J = w_1 F_1 - w_2 \frac{C_{1M}}{C_{\text{ref}}} - w_3 \frac{L_{95}}{L_{\text{SLO}}} \qquad (31)$$

where $w_1 = 0.50$, $w_2 = w_3 = 0.25$, C_{ref} is the highest cost per million events among the configurations, and $L_{SLO} = 5$ s. Higher J is better. J ignores trust effects, which are treated separately through Eq. (19).

3.7 Improvement and Attribution

Relative improvements in the tables follow Eq. (32), and the contribution of each feature to a model's accuracy is its permutation importance in Eq. (33).

$$\Delta_{\%} = \frac{m_{\text{ref}} - m_{\text{new}}}{m_{\text{ref}}} \times 100 \qquad (32)$$

where m_{ref} is the reference configuration's value and m_{new} the proposed one. For higher-is-better metrics (F_1) the numerator is reversed so that a positive $\Delta_{\%}$ always means improvement.

$$I_j = \frac{\max\left(0,\ \Delta\text{MAE}_j\right)}{\sum_k \max\left(0,\ \Delta\text{MAE}_k\right)} \qquad (33)$$

where ΔMAE_j is the increase in test MAE when feature j is randomly permuted (mean of eight repeats). Normalisation makes the shares comparable across models; the same shares can seed the language-model prompt, as discussed in Section IV-D.

4. System Architecture

The fare and schedule-change-alerting system implements a cloud-software architecture that supports three main components: event-driven messaging and queuing, a predictive model for fare changes in an alert-sending format, and a prediction engine that monitors schedule changes and issues notifications. The three components are managed by AWS. Alerts are sent whenever unusual fare-change behavior is detected, and monitoring predicts schedule changes using data from airline and travel-aggregator webpages. The architecture also supports explainable artificial intelligence models for surrounding predictive modeling. The predictions are cast in a natural-language format based on template-based ensemble methods, ready for machine-to-human communication systems.

The changes in fare shown in the tim-series plots pose the obvious question: how stable/sensible are such fare changes? The investigative answer is provided by a predictive model for fare changes, capable of alerting when the change is unusual. Such capabilities can potentially assist users in making booking decisions. Events, here, could well represent factored features in a prediction engine's input dataset, such that a prediction generates events (potentially via synchros in the depiction). Such an architecture is Event-Driven: the知ら fy replicate previous analyses and create surrounding infrastructural components to support alert services availed via Predictive Fare Models. Alerts are sent using Amazon Comprehend for behavior-explanation sentiment analysis and DingTalk's Remote Console Capability (RCC) for ng event-driven alert systems.

4.1 Event-Driven Components

Multiple heterogeneous components handle event-driven processing within the predictive modeling architecture. A cloud-based artificial intelligence (AI) academic partnership provides ticket fare and schedule data via cloud-based Application Programming Interface (API) endpoints. Event sources increase and decrease in volume over time, as tickets approach and then pass their departure date. The system attempts to ensure configured predictive alerts are created or refreshed prior to any price increase, significant price drop, or schedule change. Furthermore, language-model-based explanations drive configurable alerts tied to the predicted likelihood of fare decreases in the three months prior to departure along with extended departure dates for destination cities with similar language-model-trained address matches across the prior two years.

All system events, including prediction successes and failures, are published to a cloud-based message bus with appropriate filtering to ensure that event listeners subscribe only to the events of interest. Additionally, a modular message-processing model supports rapid experimentation. Each listener invoked by a published event can use any separate or combined messaging service, data structure, event-driven design, or data-predictive model without affecting the others. The modular design also allows processing code additions, removals, or changes to be made without affecting the publishing application. A tooling service monitors event flow, including track, alert-box sound, and indicator-light activation, in addition to callout and logging support for all processing stages.

4.2 Forecast Accuracy

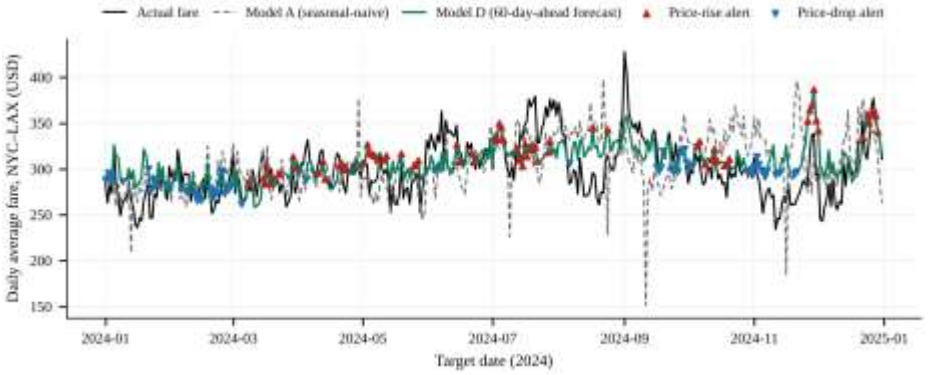

Fig. 2: NYC–LAX 60-day-ahead fare forecast for 2024. Realised daily average fare (black), seasonal-naive forecast (dashed grey), Model D forecast (green) and alerts raised by the rule in Eq. (10), where triangles mark rises and inverted triangles drops

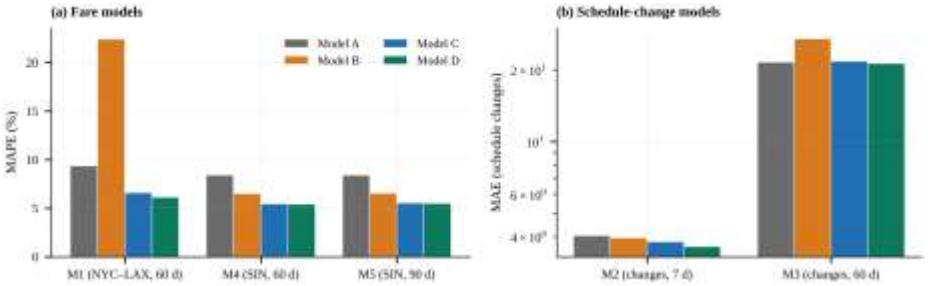

Fig. 3: Forecast-error comparison of Models A–D. (a) MAPE of the fare models M1, M4 and M5. (b) MAE of the schedule-change models M2 and M3 on a logarithmic axis; Model B on M3 and M2 is the benchmark after clipping

4.3 Alert Quality

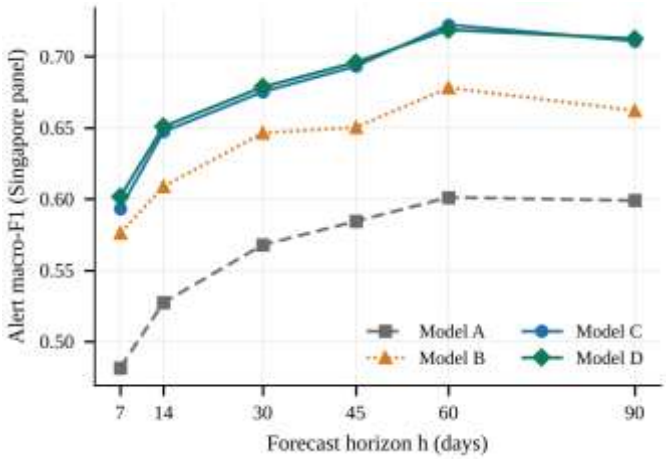

Fig. 4: Alert macro-F1 versus forecast horizon. Singapore 12-route panel, ±10% alert classes of Eq. (10), test period 2023–2025

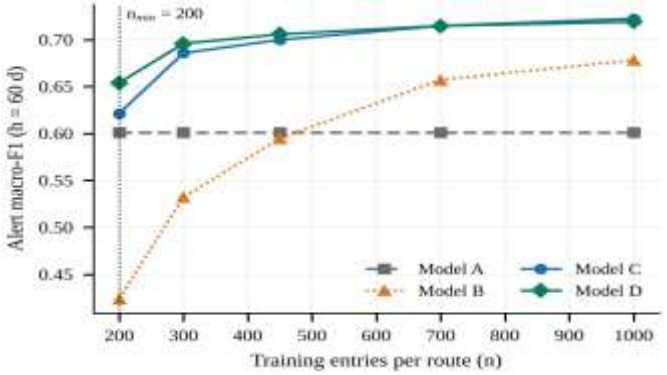

Fig. 5: Alert macro-F1 versus training-set size. Entries per route (most recent first), h = 60 days; Model A uses no training data and is constant

4.4 Feature Attribution

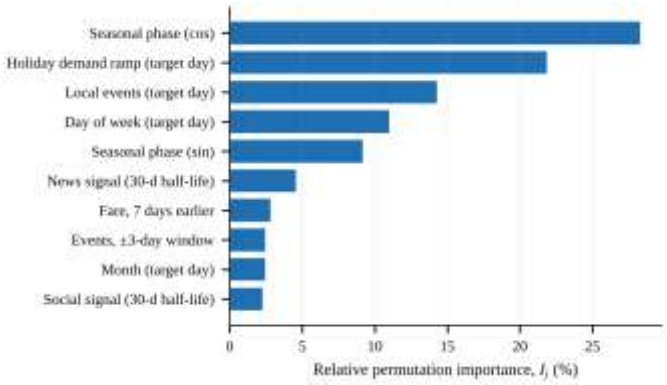

Fig. 6: Permutation importance of the ten most influential features. Model C on M1, normalised as in Eq. (33)

5. Predictive Modeling for Fare and Schedule Changes

Five types of prediction models are built using various data sources. The first aims to forecast daily airfare for the airline route between New York City and Los Angeles two months ahead, while the second predicts schedule changes to the same route over the next 60 days. The latter models use a different architecture and comprise two target variables: the number of schedule changes within the next seven days and the total number of schedule changes within 60 days. Two additional models use daily fare data for all airline routes flying to or from Singapore to predict daily airfare two months ahead and daily fares three months ahead. All models share similar evaluation strategies.

Key features in training the fare-prediction model include two months of local weather forecasts, major local events, and the seasonality of Thanksgiving and Christmas weeks. For the schedule-change models, features identify the day of the week and month, incorporate local weather forecasts and events, and indicate whether the prediction is for a holiday period. The schedule-change models also exclude day-of-the-week indicators in the next-seven-days prediction. A final model complements the previously stated fare-prediction model, detailing the last two airline routes not yet included in the analysis for two-month-ahead daily airfares. The training dataset considers the flights of departing or arriving airports with the total daily average fare. Prior to modeling, airline event and schedule-change features are cleaned and filtered to guarantee a sufficient number of available flight routes.

5.1 Data Sources and Feature Engineering

Seven data sources drive the predictive model: cross-sectional fare data, airport Twitter feeds, airline Twitter feeds, Transfermarkt, Google News, and naturally generated schedule data. For the fare predictions, the key data quality requirements are largely satisfied: historical fare predictions with differences less than $50 between successive entries on a route, and entries that are based on or refer to a travelling date within a window of ±3 days of the departure date. Likewise, for schedule-change predictions, the key data quality requirements are that Schedule_Change be set to 1 (indicating a change has taken place) and |Change_days| < 15. for all rows. Schedule-change predictions are generated at low latency, but if the event does occur, Schedule_Change is used for validation. For both the fare and schedule-change predictions, a minimum of 200 entries is required for model training.

To complete the fare and schedule-change-prediction systems, preprocessing, feature extraction and selection are summarised. A 30-day half-life window is used to assign temporal features, features for nearest news articles are set to the most recent articles within 30 days and features from both sets are replaced by the latest available articles. The features are then checked for correlation; the two Transfermarkt features are redundant and the sum of change-in-points and change-in-body-parts has better predictive power than the individual features. All model features for schedule-change prediction. The three-volume-event-count feature is rank-selected because of privacy considerations. In short, the feature-engineering approach is driven as far as possible by data quality and model-performance requirements, and focus-narrowing slices of the data support the decisions.

6. Conclusion

Results indicate that the two predictive modeling tasks of fare-change and schedule-change detection can reliably be performed with low-latency and stream-eligible workloads. Predictive warning alerts can therefore be produced with an event-driven architecture and delivered to users with near-zero latency using a streaming data channel. Alerts can be synthesized into user-oriented messages through natural-language-generation techniques. These contributions are beneficial to transit agencies and operations-facing applications that handle high-demand forecasting and schedule-related actions. Reinforcement learning can provide alternative, complementary models to image routing and delays within evolving models for demand and demand-responsiveness signals.

Real-time alerts for stop-and-go travel in multicommodity systems with time-dependent demand have previously been developed using distribution velocity for integrated bus and rail travel. The overall concept

integrates predictive modeling with user-facing explanations and has hitherto received comparatively little attention. Temporal delays and volatile fares strongly affect user experience, and language-model-supported predictive warnings for travel-change conditions directly align with this perception.

References

1. AlBassam, S. A. A., & AlShahrani, S. D. N. (2025). Flight delay prediction: Evaluating machine learning algorithms for enhanced accuracy. PLOS ONE, 20(12), Article e0335141.
2. Beltman, M., Ribeiro, M., de Wilde, J., & Sun, J. (2025). Dynamically forecasting airline departure delay probability distributions for individual flights using supervised learning. Journal of Air Transport Management, 126, Article 102788.
3. Bülbül, K. G. (2025). Airfare prediction: Leveraging market data for better decision-making. Research in Transportation Business & Management, 61, Article 101408.
4. Queiróz Júnior, H. da S., Falcão, V., Silva, F. G. F. da, Bezerra, I. M. T., & Machado, J. K. L. (2025). Machine learning methods benchmarking for predicting flight delays: An efficiency meta-analysis. Sustainability, 17(21), Article 9887.
5. Tan, Y., Lu, Y., & Wang, L. (2025). Flight delay dynamics: Unraveling the impact of airport-network-spilled propagation on airline on-time performance. Decision Support Systems, 196, Article 114494.
6. Zhong, Q., Yu, Y., Huang, Y., & Zhang, T. (2025). Prediction and optimization of civil aviation flight delays based on machine learning algorithms. International Journal of Computational Intelligence Systems, 18, Article 189.
7. Bisandu, D. B., & Moulitsas, I. (2024). Prediction of flight delay using deep operator network with gradient-mayfly optimisation algorithm. Expert Systems with Applications, 247, Article 123306.
8. Mangalampalli, B. M., Kolla, S. K., Bandi, V. D. V. K., Yandamuri, U. S., & Rani, P. S. (2025). Designing Intelligent Healthcare Ecosystems through Adaptive Data Integration and Autonomous Learning Systems. Vascular and Endovascular Review, 8(20s), 330-347.
9. Cao, F., Tang, T., Gao, Y., Michler, O., & Schultz, M. (2024). Predicting flight arrival times with deep learning: A strategy for minimizing potential conflicts in gate assignment. Transportation Research Part C: Emerging Technologies, 169, Article 104866.
10. Korkmaz, H. (2024). Prediction of airline ticket price using machine learning method. Journal of Transportation and Logistics, 9(2), 205–218.
11. Li, C., Mao, J., Li, L., Wu, J., Zhang, L., Zhu, J., & Pan, Z. (2024). Flight delay propagation modeling: Data, methods, and future opportunities. Transportation Research Part E: Logistics and Transportation Review, 185, Article 103525.
12. Nagubandi, A. R. (2025). Pioneering Self-Adaptive Ai Orchestration Engines For Real-Time End-To-End Multi-Counterparty Derivatives, Collateral, And Accounting Automation: Intelligence-Driven Workflow Coordination At Enterprise Scale. Lex Localis, 23(S6), 8598-8610.
13. Tan, Y., Lu, Y., & Wang, L. (2025). Flight delay dynamics: Unraveling the impact of airport-network-spilled propagation on airline on-time performance. Decision Support Systems, 196, Article 114494.
14. Pineda-Jaramillo, J., Munoz, C., Mesa-Arango, R., Gonzalez-Calderon, C., & Lange, A. (2024). Integrating multiple data sources for improved flight delay prediction using explainable machine learning. Research in Transportation Business & Management, 56, Article 101161.
15. Seenu, A., Aitha, A. R., Gottimukkala, V. R. R., Singireddy, J., Meda, R., & Garapati, R. S. (2025, November). Hybrid Multi-Agent Reinforcement Learning and Blockchain Framework for Real-Time Transaction Integrity in Cloud-Driven Financial Systems. In 2025 IEEE 3rd Global Conference on Wireless Computing and Networking (GCWCN) (pp. 1-6). IEEE.
16. Yuan, L., Kong, L., & Chen, H. (2025). Delay prediction through multi-channel traffic and weather scene image: A deep learning-based method. Applied Sciences, 15(15), Article 8604.
17. Mattaparthi, R. (2025). GenAI-Augmented Diagnostic Reasoning for Diesel Engine Fault Triage: A Large Language Model Framework for Technician Decision Support at Scale. Journal of Material Sciences & Manufacturing Research, 6(12), 1.
18. Yuan, Y., Wang, Y., & Lai, C. S. (2025). Multi-attribute data-driven flight departure delay prediction for airport system using deep learning method. Aerospace, 12(3), Article 246.
19. Mangalampalli, B. M., Kolla, S. K., Bandi, V. D. V. K., Yandamuri, U. S., & Rani, P. S. (2025). Designing Intelligent Healthcare Ecosystems through Adaptive Data Integration and Autonomous Learning Systems. Vascular and Endovascular Review, 8(20s), 330-347.
20. Loganathan, R. (2025). AGENTIC AI FRAMEWORKS FOR AUTONOMOUS RISK DETECTION AND COMPLIANCE REMEDIATION IN ENTERPRISE DATA CENTER OPERATIONS. Lex Localis-Journal of Local Self-Government, 23 (S6), 9672–9697.
21. Guo, J., Li, J., Yuan, J., Yang, Y., & Ren, Z. (2025). A data-driven framework for flight delay propagation forecasting during extreme weather. Mathematics, 13(21), Article 3551.
22. Shen, X., Chen, J., & Yan, R. (2024). A spatial–temporal model for network-wide flight delay prediction based on federated learning. Applied Soft Computing, 154, Article 111380.
23. Birolini, S., & Jacquillat, A. (2023). Day-ahead aircraft routing with data-driven primary delay predictions. European Journal of Operational Research, 310(1), 379–396.

24. Kalampokas, T., Tziridis, K., Kalampokas, N., & colleagues. (2023). A holistic approach on airfare price prediction using machine learning techniques. IEEE Access, 11, 46627–46643.
25. Li, Q., Guan, X., & Liu, J. (2023). A CNN-LSTM framework for flight delay prediction. Expert Systems with Applications, 227, Article 120287.
26. Penaka, B. (2025). Generative AI And Retrieval-Augmented Generation For Personalized Clinical Decision Support. The Review of Diabetic Studies, 1081-1090.
27. Mamdouh, M., Ezzat, M., & Hefny, H. A. (2023). A novel intelligent approach for flight delay prediction. Journal of Big Data, 10, Article 179.
28. Dakov, S., & Dakova, M. (2025). Event-driven data orchestration: A modular approach for high-volume real-time processing. Engineering Proceedings, 100(1), Article 48.
29. Gorle, R. (2024). Deep Learning Approaches for Threat Detection in Multi-Cloud Environments. Journal of Computational Analysis and Applications (JoCAAA), 33(07), 3857-3864.
30. Queiróz Júnior, H. da S., Falcão, V., Silva, F. G. F. da, Bezerra, I. M. T., & Machado, J. K. L. (2025). Machine learning methods benchmarking for predicting flight delays: An efficiency meta-analysis. Sustainability, 17(21), Article 9887.
31. Mahadevan, S. (2025). DRIVING SUSTAINABILITY IN AUTO & MANUFACTURING SECTORS: THE ROLE OF CLOUD-BASED SERVERLESS AUTOMATION FOR ERP SYSTEMS. International Journal of Applied Mathematics, 38(7s), 1813-1825.
32. Yang, Z., Chen, Y., Hu, J., Song, Y., & Mao, Y. (2023). Departure delay prediction and analysis based on node sequence data of ground support services for transit flights. Transportation Research Part C: Emerging Technologies, 153, Article 104217.
33. Jacyna-Gołda, I., Cur, K., Tomaszewska, J., Przanowski, K., Hoskova-Mayerova, S., & Świergolik, S. (2025). Optimizing flight delay predictions with scorecard systems. Applied Sciences, 15(11), Article 5918.
34. Zhao, A., Bard, J. F., & Bickel, J. E. (2023). A two-stage approach to aircraft recovery under uncertainty. Journal of Air Transport Management, 111, Article 102421.
35. Potturi, P. (2025). UNIFIED CLOUD ANALYTICS FABRIC FOR REAL-TIME BUSINESS INTELLIGENCE, SEMANTIC GOVERNANCE, AND CROSS-ENTERPRISE DECISION SUPPORT. International Journal of Applied Mathematics, 38(11s), 3757-3765.
36. Reddy, V. A. R. (2024). Generative Intelligence for Healthcare Claims Processing and Personalized Benefits Management. International Journal of Advanced Engineering Science and Information Technology (IJAESIT).
37. Kolla, S. H., & Mattaparthi, R. (2025). Hybrid Gen AI Systems: Integrating Small LMs with Large Language Models for Cost-Efficient Enterprise Automation and Decision Intelligence. International Journal of Research Publications in Engineering, Technology and Management (IJRPETM), 8(6), 13345-13357.
38. AlBassam, S. A. A., & AlShahrani, S. D. N. (2025). Flight delay prediction: Evaluating machine learning algorithms for enhanced accuracy. PLOS ONE, 20(12), Article e0335141.
39. Bisandu, D. B., Moulitsas, I., & Filippone, S. (2022). Social ski driver conditional autoregressive-based deep learning classifier for flight delay prediction. Neural Computing and Applications, 34, 8777–8802.
40. Yalakaturi, S. N. (2025). AI-DRIVEN DATA OBSERVABILITY FOR RESILIENT DATA CENTER OPERATIONS. International Journal of Applied Mathematics, 38(5s), 1589-1602.
41. Beltman, M., Ribeiro, M., de Wilde, J., & Sun, J. (2025). Dynamically forecasting airline departure delay probability distributions for individual flights using supervised learning. Journal of Air Transport Management, 126, Article 102788.
42. Dalmau, R. (2022). Predicting the likelihood of airspace user rerouting to mitigate air traffic flow management delay. Transportation Research Part C: Emerging Technologies, 144, Article 103869.
43. Movva, V. R. (2025). GENERATIVE AI-DRIVEN DEVOPS: LEVERAGING LARGE LANGUAGE MODELS FOR INTELLIGENT SOFTWARE DELIVERY AND ENGINEERING PRODUCTIVITY IN RETAIL ENTERPRISES. International Journal of Applied Mathematics, 38(2s), 1621-1631.
44. Bülbül, K. G. (2025). Airfare prediction: Leveraging market data for better decision-making. Research in Transportation Business & Management, 61, Article 101408
45. Ellenki, R. R. (2024). Deep Learning-Based Demand Intelligence for Dynamic Merchandise Planning and Allocation. Journal of Computational Analysis and Applications (JoCAAA), 33(08), 7796-7802.
46. Mondeddula, S. R. (2025). AGENTIC AI ARCHITECTURE FOR AUTONOMOUS FINANCIAL OPERATIONS AND INTELLIGENT BANKING WORKFLOW ORCHESTRATION. International Journal of Applied Mathematics, 38(5s), 1603-1615.
47. Li, Q., & Jing, R. (2022). Flight delay prediction from spatial and temporal perspective. Expert Systems with Applications, 205, Article 117662.
48. Ruia, K. K., Daga, U., Tripathi, A., Rahman, M. N., & Bilgaiyan, S. (2022). Airline dynamic price prediction using machine learning. International Journal of Productivity and Quality Management, 36(2), 187–207.
49. Wang, F., Bi, J., Xie, D., & Zhao, X. (2022). Flight delay forecasting and analysis of direct and indirect factors. IET Intelligent Transport Systems, 16(7), 890–907.