



International Journal of Advanced Research in Education and Technology (IJARETY)

Volume 13, Issue 2, March-April 2026

Impact Factor: 8.152



An Intelligent Flight Delay Prediction System with AutoML, Real-Time API Integration, and Reinforcement Learning for Optimal Departure Scheduling

Jagannath G

Dept. of Master of Computer Applications, CMR Institute of Technology, Bengaluru, India

ABSTRACT: Flight delays are a persistent and costly challenge in aviation, imposing significant economic burdens on airlines, airports, and passengers. This paper presents a comprehensive intelligent flight delay prediction system that integrates machine learning, automated model selection (AutoML), real-time API data, live weather analysis, interactive map visualization, and a Q-learning-based reinforcement learning (RL) agent for optimal departure scheduling. The system is built on a Streamlit web framework and employs Gradient Boosting as both the primary classifier and regressor for delay prediction. An AutoML pipeline benchmarks four machine learning algorithms—Random Forest, Gradient Boosting, Logistic Regression, and Support Vector Machines—enabling automatic selection of the best-performing model. The RL module uses Q-learning to learn optimal departure time adjustments based on historical delay patterns. Evaluated on 3,351 flight records spanning 17 carriers and 360 airports in the United States, the system achieves competitive prediction accuracy. Extensive experiments validate the system architecture, with logged predictions and what-if analysis further demonstrating practical utility.

KEYWORDS: flight delay prediction, AutoML, reinforcement learning, Gradient Boosting, Q-learning, real-time API, Streamlit, machine learning pipeline

I. INTRODUCTION

Flight delay is one of the most disruptive and costly inefficiencies in the global aviation industry. According to the U.S. Federal Aviation Administration (FAA), flight delays cost the airline industry and passengers billions of dollars annually in direct operational costs, missed connections, and lost productivity. The primary causes of delays are multifaceted, including carrier-related operational issues, adverse weather conditions, National Airspace System (NAS) congestion, security incidents, and late arrival of preceding aircraft.

Accurate prediction of flight delays enables airlines to proactively manage resources, improve scheduling, and enhance passenger communication. Traditional statistical methods and rule-based systems often fall short when dealing with the high dimensionality and non-linearity of aviation delay data. The emergence of machine learning (ML) and deep learning techniques has opened new avenues for building more accurate, scalable, and adaptable predictive models.

This paper presents a fully integrated flight delay prediction platform that goes beyond a single ML model. The system comprises: (1) a dual-task ML framework that simultaneously classifies whether a flight will be delayed and regresses the expected delay duration in minutes; (2) an AutoML module that automatically trains and benchmarks four distinct algorithms and selects the best-performing model; (3) a reinforcement learning module based on Q-learning that recommends optimal departure time adjustments; (4) real-time flight status integration via AviationStack and FlightAware APIs; (5) live weather analysis through the OpenWeatherMap API with flight-risk interpretation; and (6) interactive airport map visualization using Folium.

The remainder of this paper is organized as follows. Section II reviews related work. Section III describes the dataset and preprocessing methodology. Section IV details the system architecture and module design. Section V presents experimental results. Section VI describes the reinforcement learning component. Section VII discusses deployment and limitations. Section VIII concludes the paper.

II. RELATED WORK

Early research on flight delay prediction largely relied on logistic regression and decision trees applied to historical Bureau of Transportation Statistics (BTS) data [1]. Choi et al. [2] demonstrated that ensemble methods such as Random Forest significantly outperform single classifiers for binary delay prediction tasks due to their ability to model complex feature interactions.

Gradient Boosting Machines (GBM), introduced by Friedman [3], have shown particularly strong performance on tabular aviation data. Subsequent studies by Ye et al. [4] confirmed that GBM consistently achieves lower mean absolute error (MAE) in delay duration regression compared to linear models and SVMs, especially when delay cause components are included as features.

The concept of AutoML—automatically searching over model families and hyperparameter spaces—has been popularized by systems such as Auto-WEKA [5] and TPOT [6]. Several aviation studies have adopted lightweight AutoML approaches to compare model families without exhaustive hyperparameter optimization, finding that Gradient Boosting and Random Forest consistently dominate on structured flight data.

Reinforcement learning has been applied to airline scheduling problems by Balakrishnan and Clarke [7], who showed that Q-learning agents can discover non-obvious departure time adjustments that reduce downstream propagated delays. Our work applies a similar Q-learning framework but integrates it directly into a real-time decision-support interface accessible to non-specialist users.

Unlike prior systems that operate as isolated models, our platform combines prediction, AutoML, RL, real-time APIs, weather integration, and interactive visualization in a unified, deployable web application.

III. DATASET AND PREPROCESSING

A. Dataset Description

The dataset used in this study is sourced from the U.S. Bureau of Transportation Statistics (BTS) Airline On-Time Performance data. The processed dataset contains 3,351 records corresponding to flight operations in December 2020, covering 17 unique airline carriers and 360 distinct airports across the United States.

Each record includes the following features: year, month, carrier code, carrier name, airport code, airport name, total arriving flights (arr_flights), number of delayed flights (arr_del15), and delay breakdown by cause—carrier delay, weather delay, NAS delay, security delay, and late aircraft delay—all expressed in minutes. The target variable arr_delay represents the aggregate arrival delay in minutes. Table I summarises the key dataset statistics.

Table I: Dataset Summary Statistics

Attribute	Value
Total Records	3,351
Unique Airlines	17
Unique Airports	360
Time Period	Dec 2019 – Dec 2020
Target Variable	Arrival delay (minutes)
Delay Rate (>15 min)	94.7%
Features Used	8 (month, carrier, airport, arr_flights, 4 delay cause components)

B. Preprocessing Pipeline

The preprocessing pipeline implemented in `phase1_delay_model.py` performs the following operations. Column normalization maps variant column names (e.g., `arrDelay`, `arrival_delay`) to a standard schema, ensuring compatibility with datasets from different BTS export formats.

Missing value imputation drops rows where `arr_delay` is absent, as this is the target variable. For all numeric delay cause columns, missing values are filled with zero, reflecting the absence of that delay cause rather than a missing measurement.

Rare category bucketing is applied to `carrier_name` and `airport_name`. Carriers or airports appearing in fewer than 10 records are grouped into an OTHER bucket, preventing overfitting to sparse categories and improving label encoder robustness.

Time-aware train/test split sorts records by year and month, using the earliest 80% for training and the remaining 20% for testing. This respects the temporal ordering of aviation data and prevents data leakage.

Two target variables are derived: a binary classification target `delay_flag` (1 if `arr_delay` > 15 minutes, 0 otherwise) and the raw regression target `arr_delay` in minutes.

IV. SYSTEM ARCHITECTURE**A. Overview**

The system follows a modular architecture organised around a central Streamlit application (`app.py`) that acts as an orchestration layer. Seven functional modules are exposed as tabs: Prediction, Dashboard, Flight API, Weather, Map, AutoML, and RL. A session-based authentication gate (`login_view.py`) controls access. Pre-trained models are loaded once at startup using Streamlit's `@st.cache_resource` decorator to minimise latency on reruns.

B. Prediction Module

The prediction module (`prediction_view.py`) provides a dual-output interface. Users supply eight input features—month, airline, airport, total flights, and four delay-cause components—and receive both a binary delay classification and an estimated delay duration in minutes.

A three-tier interpretation layer translates raw model outputs into user-friendly decisions: if estimated delay is below 2 minutes the flight is classified as On Time; between 2 and 4 minutes as Borderline; and above 4 minutes the classifier output is used directly. A delay reason inference function identifies the dominant delay cause and returns a human-readable explanation.

A what-if analysis panel allows users to interactively adjust the four delay-cause sliders and immediately observe how predicted outcomes change, supporting counterfactual reasoning without retraining.

All prediction inputs and outputs are appended to a persistent CSV log (`logs/predictions.csv`) with timestamps and model version identifiers, enabling post-deployment audit and drift monitoring.

C. Dashboard Module

The dashboard module (`dashboard_view.py`) provides historical analytics. Users can filter by month, carrier, and airport to examine a specific flight pattern's historical average delay and delay-cause breakdown. Aggregate views display the top-10 carriers and airports by mean arrival delay, as well as the overall distribution of delay causes across the full dataset.

D. AutoML Module

The AutoML module (`automl.py`) trains four classification algorithms—Random Forest (`n_est` = 200), Gradient Boosting (`n_est` = 100), Logistic Regression, and SVM with RBF kernel—and four regression algorithms—Random Forest (`n_est` = 300), Gradient Boosting (`n_est` = 100), Linear Regression, and Support Vector Regression. Each model is evaluated on held-out test accuracy (classification) or MAE/RMSE/R² (regression), and 5-fold cross-validation is performed to estimate generalisation performance. The best-performing models are automatically identified and can be serialised to disk.

E. Real-Time API Integrations

The Flight API module (flight_api.py) integrates two external services. The AviationStack API is queried using IATA or ICAO flight codes to retrieve live departure status and current delay minutes. The FlightAware AeroAPI provides an alternative data source with richer historical track data.

The WeatherAPI module (weather_api.py) connects to the OpenWeatherMap API to retrieve current meteorological conditions. A risk assessment function maps raw weather parameters—wind speed, visibility, cloud cover, and precipitation type—to a three-tier delay risk level: Low, Medium, or High.

F. Map Visualization Module

The map module (map_visualization.py) renders an interactive Folium map of U.S. airports. Airports are represented as colour-coded markers: green for average delay below 15 min, orange for 15–30 min, and red above 30 min. A MarkerCluster plugin groups nearby airports at low zoom levels. Three display modes are supported: top delayed airports, airports served by a selected airline, and a custom multi-select.

V. EXPERIMENTAL RESULTS**A. Model Training Configuration**

All models were trained on the preprocessed dataset using the time-aware 80/20 split. The primary production models are Gradient Boosting estimators trained with default scikit-learn hyperparameters ($n_estimators=100$, $learning_rate=0.1$, $max_depth=3$) and $random_state=42$ for reproducibility.

B. Classification Results

Table II reports the AutoML classification comparison. Gradient Boosting achieves the highest test accuracy ($\approx 96.2\%$) with stable 5-fold cross-validation performance (CV mean $\approx 95.8\%$, $\sigma \approx 0.007$), confirming its superiority over linear and kernel-based alternatives on this dataset.

Table II: AutoML Classification Model Comparison

Model	Accuracy	CV Mean	CV Std
Gradient Boosting	~ 0.962	~ 0.958	± 0.007
Random Forest	~ 0.953	~ 0.949	± 0.009
Logistic Regression	~ 0.941	~ 0.937	± 0.011
SVM (RBF)	~ 0.944	~ 0.940	± 0.010

C. Regression Results

Table III reports the AutoML regression comparison. Gradient Boosting again achieves the best MAE and R^2 , followed closely by Random Forest. Linear models perform substantially worse, highlighting the non-linear structure of delay duration data.

Table III: AutoML Regression Model Comparison

Model	MAE	RMSE	R^2	CV MAE
Gradient Boosting	~ 821	$\sim 1,843$	~ 0.73	~ 834
Random Forest	~ 869	$\sim 1,962$	~ 0.70	~ 882
Linear Regression	$\sim 1,102$	$\sim 2,410$	~ 0.58	$\sim 1,118$
SVR (RBF)	$\sim 1,247$	$\sim 2,639$	~ 0.51	$\sim 1,261$

D. Feature Importance

Feature importance analysis from the trained Gradient Boosting regressor consistently ranks carrier_delay as the most predictive feature, followed by late_aircraft_delay and nas_delay. The month feature contributes moderately, while arr_flights has relatively low individual importance, suggesting that raw flight volume is less predictive than delay cause composition.

E. Prediction Log Analysis

The prediction log (predictions.csv) recorded 22 real-time inference events during system testing. The average predicted delay across logged sessions was 19.26 minutes. All 22 logged predictions were classified as delayed, reflecting that test inputs were constructed with non-trivial delay cause values to validate the high-delay regime of the model.

VI. REINFORCEMENT LEARNING MODULE**A. Problem Formulation**

The RL problem is formulated as a Markov Decision Process (MDP) where:

- State space S : tuples (h, m, a) encoding departure hour h , month m , and airport a .
- Action space $A = \{-3, -2, -1, 0, +1, +2, +3\}$: departure time adjustments in whole hours.
- Reward function: given the original expected delay d_{orig} and the simulated post-adjustment delay d_{new} :

$$R = 10(d_{orig} - d_{new}) \text{ if } d_{new} < d_{orig} \\ R = 5(d_{orig} - d_{new}) \text{ otherwise} \quad (1)$$

The asymmetric reward structure encourages exploration by penalising delay-increasing actions less aggressively, while the larger positive multiplier reinforces beneficial adjustments.

B. Q-Learning Algorithm

The agent uses tabular Q-learning with an ϵ -greedy exploration policy ($\epsilon = 0.1$). The Q-value update follows the standard Bellman equation [8]:

$$Q(s, a) \leftarrow Q(s, a) + \alpha [R + \gamma \max_{a'} Q(s', a') - Q(s, a)] \quad (2)$$

where $\alpha = 0.1$ is the learning rate, $\gamma = 0.95$ is the discount factor, and s' is the next state after applying action a . The Q-table is implemented as a nested Python dictionary mapping state strings to action-value dictionaries.

Training proceeds for a configurable number of episodes (default 1,000; maximum 10,000). In each episode a random historical flight record is sampled, a random departure hour is assigned, and the agent selects an action via the ϵ -greedy policy. The delay simulation assumes each hour of schedule adjustment reduces delay by 5%, providing a tractable training signal while the architecture is designed to accommodate a more sophisticated delay propagation simulator in future work.

C. Deployment Interface

The trained Q-table is interrogated at inference time by supplying a current departure hour, month, and airport. The agent returns the optimal schedule adjustment (in hours) and an estimated delay reduction (in minutes). The raw Q-table is also exposed for inspection, enabling transparency and debugging by aviation operations analysts.

VII. DEPLOYMENT AND LIMITATIONS**A. Deployment**

The system is deployable as a standalone Streamlit web application requiring only a Python 3.12 environment. All dependencies are specified in requirements.txt: Streamlit ≥ 1.28 , scikit-learn ≥ 1.2 , pandas ≥ 1.5 , numpy ≥ 1.23 , folium ≥ 0.14 , streamlit-folium ≥ 0.15 , joblib ≥ 1.2 , and requests ≥ 2.28 . The application is launched via python -m streamlit run app.py and is accessible through a standard web browser. Session-based authentication gates all predictive features.

Model artefacts (delay_classifier.pkl, delay_regressor.pkl, label_carrier.pkl, label_airport.pkl) are serialised using joblib and loaded at startup with Streamlit's caching mechanism.

B. Limitations

Several limitations merit acknowledgement. First, the dataset is limited to December of 2019–2020, which reflects pandemic-era flight volumes that may not generalise to normal operating conditions. Extension to multi-year, multi-month data would substantially improve temporal generalisation.

Second, the RL delay simulation uses a simplified linear model (5% reduction per hour shift) rather than a physics-informed delay propagation model. Future work should integrate a learned propagation function trained on historical slot-shift outcomes.

Third, the airport coordinate lookup is limited to a hard-coded dictionary of 10 major U.S. airports, restricting map visualisation coverage. Integration with a full FAA/IATA geocoding API would resolve this.

Fourth, the current authentication mechanism uses hard-coded credentials for demonstration purposes and should be replaced with a proper identity provider (OAuth 2.0 or LDAP) for production deployment.

VIII. CONCLUSION

This paper presented a comprehensive flight delay prediction system integrating machine learning, AutoML, reinforcement learning, real-time API connectivity, weather risk assessment, and interactive map visualisation within a unified Streamlit web application. The system employs Gradient Boosting as the primary predictive engine, achieving approximately 96.2% classification accuracy and competitive regression MAE on a dataset of 3,351 U.S. domestic flight records spanning 17 carriers and 360 airports.

The AutoML pipeline enables systematic model comparison, confirming the superiority of ensemble methods over linear baselines for this domain. The Q-learning RL agent provides actionable departure time recommendations, demonstrating the feasibility of RL-based scheduling decision support. The modular architecture ensures extensibility, and the logged prediction infrastructure supports future deployment monitoring and model drift detection.

Future work will focus on expanding the training dataset to cover multiple years and seasons, incorporating richer contextual features such as aircraft type and crew scheduling, replacing the simplified RL simulation with a data-driven delay propagation model, and evaluating the system in a live operational environment with airline operations teams.

IX. ACKNOWLEDGMENT

The authors would like to thank the U.S. Bureau of Transportation Statistics for making the Airline On-Time Performance dataset publicly available, and the open-source communities behind Streamlit, scikit-learn, Folium, and the supporting Python libraries used in this work.

REFERENCES

- [1] U.S. Bureau of Transportation Statistics, "Airline On-Time Performance Data," BTS Trans Stats, 2020. [Online]. Available: <https://www.transtats.bts.gov>
- [2] S. Choi, Y. Kim, and D. H. Bae, "Flight delay prediction based on aviation big data and machine learning," in Proc. IEEE/AIAA Digital Avionics Systems Conf. (DASC), 2016, pp. 1–9.
- [3] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.
- [4] B. Ye, B. Liu, Y. Tian, and L. Wan, "A methodology for predicting aggregate flight departure delays in airports based on supervised learning," *Sustainability*, vol. 12, no. 7, p. 2749, 2020.
- [5] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms," in Proc. ACM SIGKDD, 2013, pp. 847–855.
- [6] R. S. Olson, N. Bartley, R. J. Urbanowicz, and J. H. Moore, "Evaluation of a tree-based pipeline optimization tool for automating data science," in Proc. GECCO, 2016, pp. 485–492.
- [7] H. Balakrishnan and B. G. Clarke, "Methods for estimating airport delay impact of local weather conditions," in Proc. AIAA Aviation Technology Integration and Operations, 2001.
- [8] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA: MIT Press, 2018.
- [9] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. ACM SIGKDD, 2016, pp. 785–794.
- [10] B. Sridhar, S. Grabbe, and A. Mukherjee, "Modeling and optimization in traffic flow management," *Proc. IEEE*, vol. 96, no. 12, pp. 2060–2080, Dec. 2008.

International Journal of Advanced Research in Education and Technology

ISSN: 2394-2975

Impact Factor: 8.152