Flight Delay Prediction

Group 31: Nicolas Alvarez Ortiz, Neeti Tatiya, Hongyi Zhu, Aaron Li, Xinyue Jiang

Abstract

We analyze delay propagation in U.S. domestic flights by reconstructing aircraft rotation chains from January 2025 BTS data and integrating airport-level weather factors. Our results show that inbound delays translate non-linearly into outbound delays, with turnaround time acting as the strongest moderator. Significant variation exists across airlines and airports, reflecting differences in operational buffers and constraints. These insights support a predictive model that identifies conditions under which delays are most likely to amplify

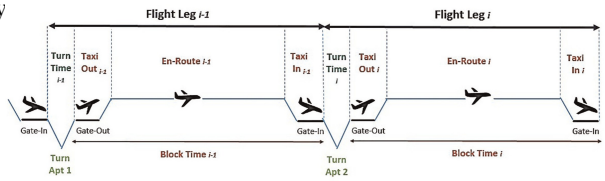

Figure 1: Overview of Flight Delay Propagation

1 Data Sources, Cleaning

We reconstruct aircraft rotation chains using the January 2025 U.S. Bureau of Transportation Statistics (BTS) On-Time Performance dataset, which offers comprehensive flight-level data such as Tail Number, timestamps, and delay components. We added weather information for 349 U.S. airports to include precipitation, wind, and cloud-cover metrics pertinent to winter operations through OpenMeteo´s API, as well as the OpenFlights airport database to obtain standardized airport identifiers (Using IATA codes) and coordinates of 349 airports.

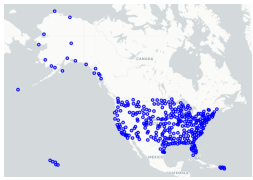

Figure 2: Coverage of U.S. Airports

2 Feature Engineering

The raw data needed to be thoroughly cleaned: all timestamps were standardized, flights without Tail Number were eliminated, corrupted time fields were fixed, and only legitimate aircraft sequences were kept by arranging flights chronologically and removing breaks brought on by maintenance gaps or incorrect origin-destination transitions. Each flight was connected to the aircraft's prior arrival using an as-of merge, creating core propagation fields.

In addition to schedule, airport, and weather features like DayOfWeek, distance, departure/arrival time blocks, and origin-destination weather indicators, we then created categorical delay and turn-time buckets. The final dataframe, which serves as the structured foundation for both the exploratory analysis and the predictive modeling stage, is made up of paired flight segments prior to arrival matched to subsequent departure.

As part of the Data processing in terms of Feature Engineering, aircraft rotation chains were created "Tail Number Created" rhe previous leg arrival and transformed for the next leg departure linkage. Required operational fields were computed for propagation analyzes: Such as "previous arrival delay", "departure delay" and "turnaround time".

3 Exploratory Data Analysis

The exploratory analysis looks at how delays from an aircraft's prior leg affect its next planned departure. The probability of an outgoing delay increases sharply once the previous arrival is more than 15 minutes late; inbound delays exceeding 60 minutes are linked to nearly a 60 percent chance of a 60 minute outbound delay, indicating a strong non-linear propagation effect. The majority of inbound arrivals occur within ±20 minutes, where propagation is limited. This relationship is significantly moderated by turnaround time: short turns lasting less than 45 minutes show the greatest amplification, while propagation gradually decreases with longer buffers and is almost eliminated for turns lasting more than 180 minutes.

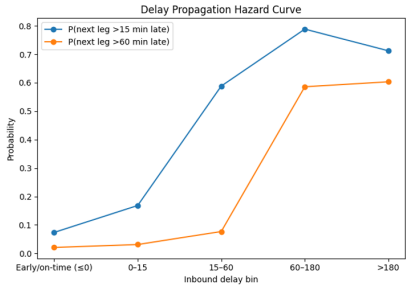

Figure 3: Delay Hazard Curve

Additionally, different carriers exhibit different operational behaviors. While American, Endeavor, and CommutAir exhibit greater resilience and delay absorption, Hawaiian, Southwest, and Allegiant exhibit the highest propagation slopes, indicating constrained recovery capacity or tight schedules. Airport patterns reflect these dynamics: major hubs are uncommon among high-slope airports, while mountain, island, and leisure destinations like Colorado, Florida and North Carolina (EGE, EYW, PNS, and AVL airports) exhibit elevated propagation due to infrastructure or weather challenges. Weather has an uneven impact on performance; wind and cloud cover have weaker and less consistent relationships with the risk of outbound delay, while heavy precipitation has a significant impact. All of these results demonstrate

that delay propagation is pervasive, non-linear, and strongly influenced by operational buffers, carrier policies, airport limitations, and winter weather.

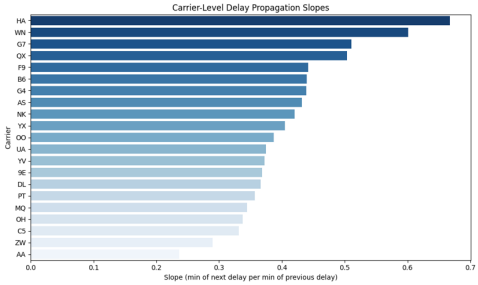

Figure 4: Delay Propagation by Airline

The distribution of departure delays is heavily right-skewed (Fig.16). Most flights depart close to schedule, with a noticeable spike around zero and a smaller share leaving slightly early. At the same time, there is a long tail of large positive delays, with some flights departing hours late. These extreme delays are relatively rare, but they account for much of the overall variability in the data. This pattern suggests that while delays are common, severe disruptions represent a distinct operational regime that should be treated carefully in downstream modeling.

Turnaround time plays a major role in shaping departure performance(Fig.17). Flights scheduled with very short turnarounds, i.e., less than 45 minutes, show the highest median delays and the widest spread, indicating that even small disruptions are difficult to absorb. Performance improves noticeably as turnaround time increases, with the most stable outcomes occurring within the 45- to 90-minute range. Beyond that, additional buffer continues to reduce delay risk, but the improvement is more gradual. This pattern highlights the trade-off airlines face between tight scheduling and operational reliability.

Delay behavior also differs by flight distance (Fig.18). Short- and medium-haul flights tend to show much higher variability, including the most extreme delays. In contrast, long-haul flights are more tightly clustered around zero delay, with fewer severe outliers. While median departure delays are similar across distances, the reduced dispersion for longer flights suggests that they benefit from larger schedule buffers and more flexible operations. Shorter flights, by comparison, appear more exposed to congestion and cascading delays earlier in the day.

The correlation heatmap (Fig.19) shows strong multicollinearity among several feature groups, particularly between distance-related variables (Distance, AirTime, and CRSElapsedTime) and between origin and destination weather variables (temperature, wind speed, and cloud cover). While this level of correlation would pose challenges for coefficient stability and interpretability in linear regression models, it is less problematic for predictive approaches. Since our analysis relies on Random Forests and Neural Networks, which are inherently robust to correlated inputs through feature subsampling and distributed learning, these relationships do not materially hinder predictive performance.

Overall, the exploratory analysis shows that departure delays are driven largely by delay propagation across aircraft rotations, with short turnaround times sharply amplifying inbound disruptions. Carrier practices, airport constraints, and winter precipitation further shape these patterns, reinforcing the importance of incorporating operational buffers, temporal dependence, and weather effects in predictive modeling.

4 Modeling

4.1 Feature Selections and Engineering

We construct features in a hierarchical manner, guided by the operational mechanisms governing flight delay propagation. Rather than treating delays as isolated events, our design explicitly models how upstream operational, environmental, and behavioral factors influence downstream departure performance.

(1) Aircraft rotation and schedule slack features

The first group captures delay propagation along aircraft rotations. We characterize both the magnitude of upstream arrival delay and the amount of scheduled and realized buffer between consecutive flights. These features quantify the aircraft's ability to absorb or transmit delays through turnaround operations and serve as the core propagation channel in our model.

Feature Category	Feature Name	Construction / Definition	Modeling Purpose
Aircraft rotation	prev_arr_delay	Arrival delay of previous flight by same aircraft	Capture upstream delay carry-over
	scheduled_turn_ time	Scheduled gap between previous arrival and next departure	Measure planned buffer capacity
	realized_slack	Next scheduled departure − actual previous arrival	Measure realized recovery slack
Schedule & route	CRSElapsedTime, Distance	Scheduled flight duration and route distance	Control structural route characteristics
Temporal context	DayOfWeek, is_day, time bins	Calendar and time-of-day indicators	Capture systematic temporal patterns
Weather (current)	temperature_2m, wind_speed_10m, precipitation, cloud_cover	Departure-airport weather conditions	Model immediate environmental constraints
Weather (previous)	*_pa weather variables	Weather at previous arrival airport	Capture upstream weather effects

(2) Temporal, route, and structural controls

To account for systematic heterogeneity unrelated to propagation, we include calendar indicators, time-of-day bins, and route-level structural variables such as scheduled duration and distance. These controls isolate recurrent scheduling patterns and baseline operational complexity.

Feature Category	Feature Name	Construction / Definition	Modeling Purpose
Slack-based propagation	slack_time	Scheduled turn − historical mean turn time	Measure turnaround tightness
	prev_arr_delay_clip	prev_arr_delay clipped to [−30, 180]	Reduce impact of extreme outliers
	prop_prev_delay_over_turn		
Short-turn regime	short_turn	Indicator: scheduled turn < 45 minutes	Identify high-risk turnaround regimes
Interaction effect	short_turn_prev_delay	short_turn × prev_arr_delay_clip	Capture nonlinear delay propagation

(3) Weather conditions and regime transitions

Weather is modeled not only in absolute terms at the departure airport, but also in relative terms across flight legs. By incorporating both current and upstream weather conditions, as well as their differences, the model captures regime shifts that may amplify or dampen delay propagation.

Feature Category	Feature Name	Construction / Definition	Modeling Purpose
Weather change	wind_speed_diff, temp_diff, precip_diff, cloud_diff	Current-leg − previous-leg weather	Capture weather regime transitions
Threshold effects	has_rain, has_rain_pa	Binary precipitation indicators	Model nonlinear weather risk

(4) Nonlinear propagation and interaction effects

We explicitly allow for nonlinear dynamics by introducing interaction terms that activate under tight turnaround regimes. These features reflect the empirical observation that similar arrival delays can have disproportionately larger downstream impacts when buffer capacity is constrained.

Feature Category	Feature Name	Construction / Definition	Modeling Purpose
Airport history	origin_delay_mean_7d	Shifted 7-day rolling mean delay by origin	Capture persistent airport congestion
Carrier history	carrier_delay_mean_7d	Shifted 7-day rolling mean delay by airline	Model carrier operational efficiency
Route history	route_delay_mean_7d	Shifted rolling mean by Origin–Destination	Capture route-specific structural risk

(5) Historical behavioral features with leakage control
Finally, we incorporate rolling historical delay averages at the airport, carrier, and route levels. These features proxy persistent operational efficiency and congestion effects. All historical aggregates are constructed using time-shifted windows to ensure that only information available prior to each flight is used, preventing any form of information leakage.

Feature Category	Feature Name	Construction / Definition	Modeling Purpose
Speed proxy	avg_speed	Distance ÷ scheduled elapsed time	Approximate operational cruising regime
Delay risk flag	prev_big_delay	Indicator: previous arrival delay ≥ 30 min	Capture threshold propagation effects

4.2 Modeling and Parameter Tuning

We evaluate a range of tree-based models and deep neural networks, using LASSO regression as a linear baseline. The dataset is temporally partitioned into training (January 1–21, 2025), validation (January 22–28, 2025), and test (January 29–31, 2025) sets in order to respect the chronological structure of flight operations.

For each model, training is performed on the training set, while hyperparameters are optimized using the Tree-structured Parzen Estimator (TPE) algorithm implemented in Optuna, with 30 trials conducted per model and performance evaluated on the validation set. The model is then retrained on the combined training and validation data using the selected hyperparameters, and final performance is assessed on the held-out test set.

4.2.1 Lasso

We selected the Lasso model as the baseline due to its high interpretability and its ability to shrink irrelevant feature coefficients toward zero, which facilitates feature selection and interpretation. Prior to model training, the following preprocessing steps were applied in accordance with the model's characteristics.

(1) Numerical features. Missing values were imputed using the median, followed by standardization to zero mean and unit variance.

(2) Categorical features. Categorical variables were encoded using one-hot encoding.

To accelerate training, the model was implemented using `SGDRegressor`. The final optimal hyperparameters were set as `alpha = 7.5e-05`, `epsilon = 0.54`, `eta0 = 0.055`, and `power_t = 0.087`.

4.2.2 XGBoost

XGBoost is a widely used tree-based learning algorithm that extends the traditional Gradient Boosted Decision Tree (GBDT) framework by leveraging a second-order Taylor expansion of the loss function, enabling more efficient and stable optimization. Given the characteristics of the model, the following preprocessing strategy was applied.

(1) Numerical features. Continuous variables were retained without normalization and cast to floating-point precision. Missing values were handled internally by XGBoost.

(2) Categorical features. Categorical variables were handled using XGBoost's native categorical encoding with globally consistent category sets. One-hot encoding was not applied.

Figure 12 shows the optimal hyperparameters for the XGBoost model.

4.2.3 LightGBM

LightGBM is an efficient tree-based gradient boosting framework that improves upon XGBoost by employing histogram-based splitting and a leaf-wise tree growth strategy, which enables faster training and improved scalability on large datasets. Given the characteristics of the model, the following preprocessing strategy was applied.

(1) Numerical features. No normalization or standardization was performed. Missing values were left unimputed and handled internally by the model.

(2) Categorical features. Categorical variables were explicitly cast to categorical data types and passed to the model via the `categorical_feature` argument. One-hot encoding was not applied.

Figure 13 shows the optimal hyperparameters for the LightGBM model.

4.2.4 CatBoost

CatBoost is a gradient boosting framework specifically designed to handle categorical features efficiently and robustly. It extends the traditional Gradient Boosted Decision Tree (GBDT) framework by employing ordered boosting and target-based encoding techniques, which reduce target leakage and prediction shift during training. Given the characteristics of the model, the following preprocessing strategy was applied.

(1) Numerical features. No normalization or standardization was performed. Missing values were handled internally by the model.

(2) Categorical features. Categorical variables were passed directly to the model via the `cat_features` indices. One-hot encoding was not applied. Instead, CatBoost internally uses ordered target encoding, with category statistics computed in an order-aware manner using permutations of the training data to prevent target leakage.

Figure 14 shows the optimal hyperparameters for the CatBoost model.

4.2.5 Random Forest

Random Forest is an ensemble learning method that constructs a large number of decision trees using bootstrap sampling and random feature selection, and aggregates their predictions to improve generalization performance and reduce variance. Given the characteristics of the model, the following preprocessing strategy was applied.

(1) Numerical features. Missing values were imputed using the median. No normalization or standardization was applied.

(2) Categorical features. Categorical variables were encoded using one-hot encoding. Unseen categories in the test data were safely ignored.

Figure 15 shows the optimal hyperparameters for the Random Forest model.

4.2.6 Deep Neural Network

To explore the limits of the training set, we implemented a neural network consisting of three hidden layers with widths of 256, 128, and 64, followed by a single linear output layer for regression. Several techniques were employed to improve training stability and generalization performance.

Categorical variables were represented using embedding layers, which map categories to dense, learnable vectors rather

than one-hot encoded representations. Batch normalization was applied to both numerical inputs and hidden layers to stabilize gradients and accelerate convergence. ReLU activation functions were used between layers to introduce nonlinearity. Dropout regularization was applied after each hidden layer to mitigate overfitting. Numerical and categorical features were processed separately and fused prior to entering the multilayer perceptron.

5 Model Performance

5.1 Quantitative Comparison

Table below presents the comprehensive performance evaluation of all five models on the test set.

```
============================================
MODEL PERFORMANCE SUMMARY TABLE
============================================
Model           MAE        RMSE       R²
--------------------------------------------
Lasso           9.899      27.147     0.386
Random Forest   8.093      20.178*    0.699*
CatBoost        6.925      22.244     0.607
LightGBM        6.891*     22.503     0.626
XGBoost         8.015      20.979     0.675
NeuralNet       7.245      24.122     0.570
============================================
* indicates best performance for each metric
```

Figure 5: Model Performance

5.2 Performance Insights

5.2.1 Accuracy Metrics Analysis

Best MAE: LightGBM achieves the lowest Mean Absolute Error (6.891 minutes), indicating superior precision in predicting flight delay times with minimal average deviation.

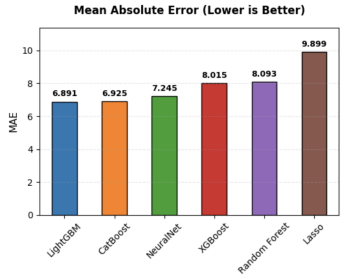

Figure 6: MAE Comparison

Best RMSE: Random Forest shows the lowest Root Mean Squared Error (20.18 minutes), suggesting better handling of extreme delay predictions and reduced prediction volatility.

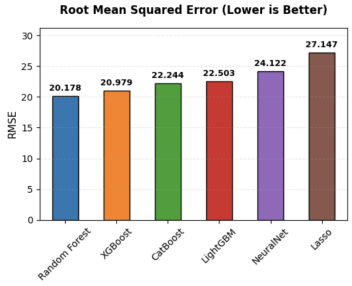

Figure 7: RMSE Comparison

Best R^2: Random Forest obtains the highest R^2 score (0.699), explaining approximately 70% of the variance in flight delays, demonstrating strong predictive power.

5.2.2 Model Category Comparison

The gradient boosting models: LightGBM, CatBoost, and XGBoost, achieve lower MAE than Random Forest, with LightGBM performing the best within this family. While they fit the data well, their generalization varies modestly across splits. Random Forest, however, shows the strongest overall stability and achieves the highest R^2, offering both robust performance and interpretable feature importance. The deep learning model, despite its use of embeddings, performs weaker ($R^2 = 0.570$), suggesting that tree-based methods remain better suited for structured tabular flight-delay data unless larger datasets or more specialized architectures are incorporated.

5.3 Feature Importance Consensus

When comparing the feature importance rankings across LightGBM, XGBoost, Random Forest, and CatBoost(Appendix: Feature Importance Figures), a fairly consistent story starts to appear. Operational variables such as *scheduled_turn_time* and *prev_arr_delay*, frequently appears on top across feature rankings of all models, which aligns with the intuition that turnaround efficiency and delay propagation from the previous leg are two of the most immediate drivers of outbound delay.

Geographic and carrier-level attributes show up as secondary but still meaningful contributors. Variables such as *Origin*, *Dest*, and *op_carrier* don't dominate the rankings, but they appear consistently enough to suggest that certain airports and carriers have structural differences in congestion patterns or operational reliability. These effects aren't as sharp as the turn-time and propagation features, but they still help the models refine their predictions.

5.4 Key Findings Summary

Overall, the performance across models was fairly consistent. No single method vastly outperformed the others, though Random Forest stood out with the highest R^2 and the most stable generalization. XGBoost and LightGBM also delivered competitive accuracy, especially in capturing non-linear relationships that naturally arise in flight operations data. CatBoost showed excellent validation performance and remained competitive on the test set, though with slightly more variance. The neural network trailed behind the tree-based models, which is expected given that deep learning methods usually require significantly larger datasets or architectures tailored specifically to tabular data.

6 Conclusion

In this project, we analyzed flight delay propagation by integrating operational and weather data, reconstructing aircraft rotation chains, and engineering features that capture upstream delays and buffer time. Tree-based and deep learning models were evaluated to reflect the non-linear nature of delay transmission, with tree-based methods showing strong predictive performance. Turnaround time and prior arrival delay consistently emerged as the most influential factors, highlighting that delays stem from operational constraints rather than isolated events, and demonstrating how data-driven modeling can support more proactive and resilient airline scheduling.

References

[1] BTS Page for data download: https://www.bts.gov/airline-data-downloads

[2] OpenMeteo API Link: https://open-meteo.com/en/docs/historical-weather-api?location_mode=csv_coordinates

[3] OpenFlights Database https://openflights.org/data

A Appendix

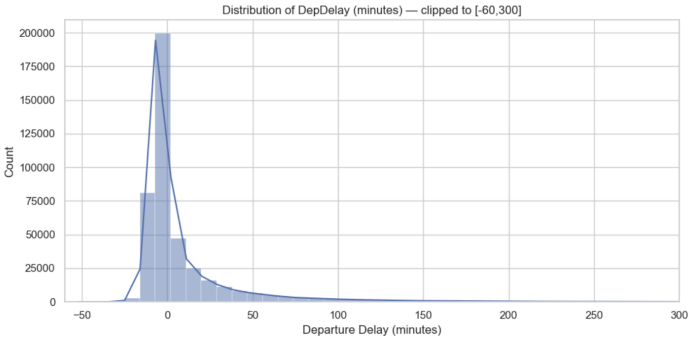

Figure 8: Distribution of Departure Delays

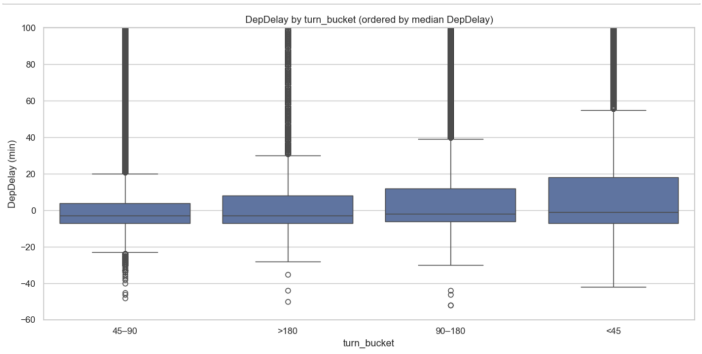

Figure 9: Effect of Turnaround Time on Departure Delay

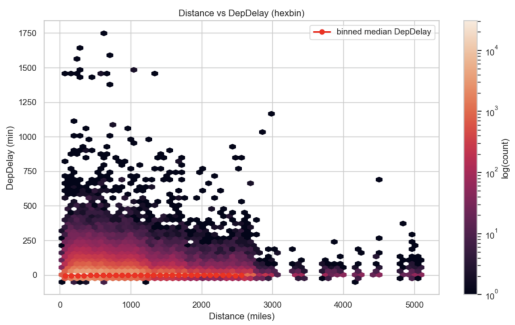

Figure 10: Relationship Between Flight Distance and Delay Variability

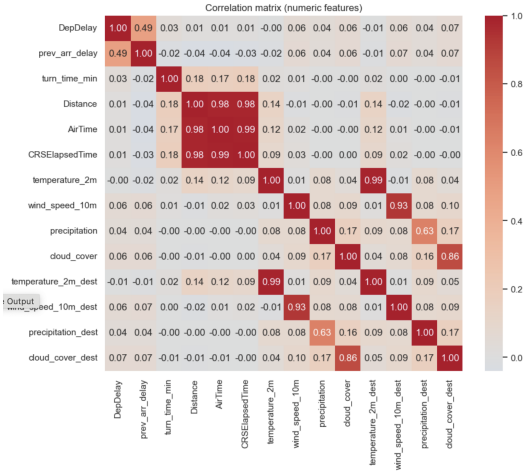

Figure 11: Correlation Heatmap of Numeric Variables

Hyperparameter	Value	Description
n_estimators	1965	Number of boosting trees; large value combined with small learning rate enables fine-grained learning
learning_rate	0.0136	Shrinks the contribution of each tree to improve generalization
max_depth	12	Maximum depth of individual trees, controlling model complexity
subsample	0.963	Fraction of training samples used per tree to reduce overfitting
colsample_bytree	0.661	Fraction of features sampled for each tree, improving robustness
min_child_weight	6.92	Minimum sum of instance weight in a child node; prevents overly specific splits
reg_lambda	7.09	L2 regularization strength on leaf weights
reg_alpha	4.69	L1 regularization strength promoting sparsity in tree splits
eval_metric	mae	Mean Absolute Error used for validation and early stopping
early_stopping_rounds	80	Stops training if validation performance does not improve

Figure 12: XGBoost Optimal Hyperparameters

Hyperparameter	Value	Description
n_estimators	2572	Number of boosting iterations; large value with small learning rate enables stable learning
learning_rate	0.0254	Controls the contribution of each tree; smaller values improve generalization
num_leaves	224	Maximum number of leaves per tree, controlling model expressiveness
min_child_samples	23	Minimum number of samples required in a leaf to prevent overfitting
min_child_weight	3.2348	Minimum sum of instance weight needed in a leaf, improving robustness
subsample	0.9881	Fraction of training data used per boosting iteration to reduce variance
colsample_bytree	0.6857	Fraction of features sampled for each tree, enhancing feature diversity
reg_lambda	7.2596	L2 regularization strength on leaf weights
reg_alpha	3.7213	L1 regularization strength encouraging sparsity
objective	mae	Objective function minimizing Mean Absolute Error

Figure 13: LightGBM Optimal Hyperparameters

Hyperparameter	Value	Description
iterations	805	Maximum number of boosting iterations
best_iteration	804	Optimal number of trees selected via early stopping
learning_rate	0.0661	Step size shrinkage controlling the contribution of each tree
depth	8	Depth of individual symmetric trees, controlling model complexity
l2_leaf_reg	9.4482	L2 regularization applied to leaf values to prevent overfitting
bagging_temperature	0.836	Controls the strength of Bayesian bootstrap sampling
subsample	0.9629	Fraction of samples used for each tree
border_count	255	Number of discretization bins for continuous features
loss_function	MAE	Objective function minimizing mean absolute error
eval_metric	MAE	Evaluation metric used for validation and early stopping
early_stopping_rounds	80	Stops training if validation performance does not improve

Figure 14: CatBoost Optimal Hyperparameters

Hyperparameter	Value	Description
n_estimators	267	Number of trees in the forest; controls ensemble size and stability
max_depth	20	Maximum depth of each tree, limiting model complexity and overfitting
min_samples_split	14	Minimum number of samples required to split an internal node
min_samples_leaf	6	Minimum number of samples required at a leaf node, improving generalization
max_features	0.5	Fraction of features randomly sampled at each split
bootstrap	TRUE	Enables bootstrap sampling to increase tree diversity

Figure 15: Random Forest Optimal Hyperparameters

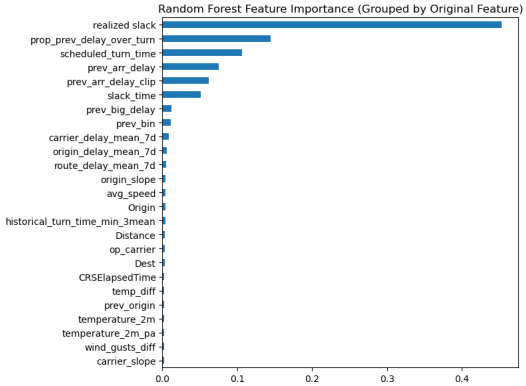

Figure 16: Random Forest Feature Importance

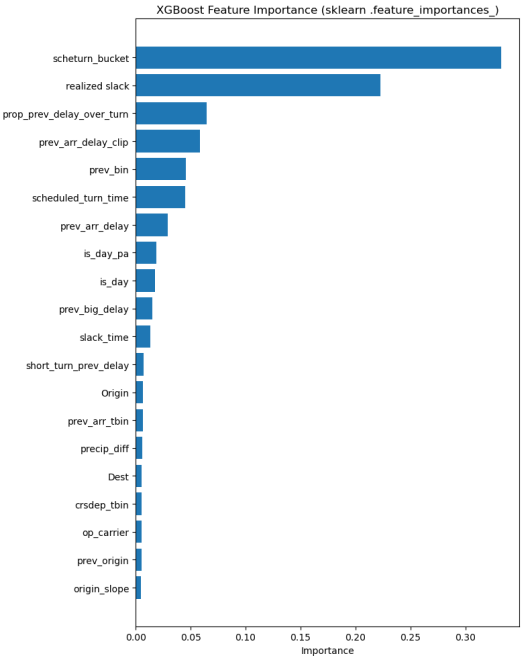

Figure 17: XGBoost Feature Importance

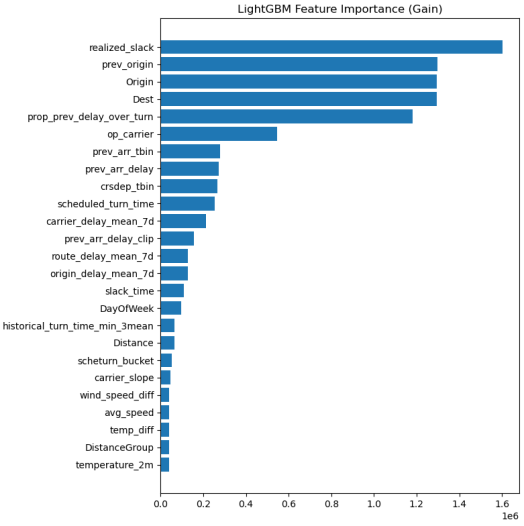

Figure 18: LightGBM Feature Importance

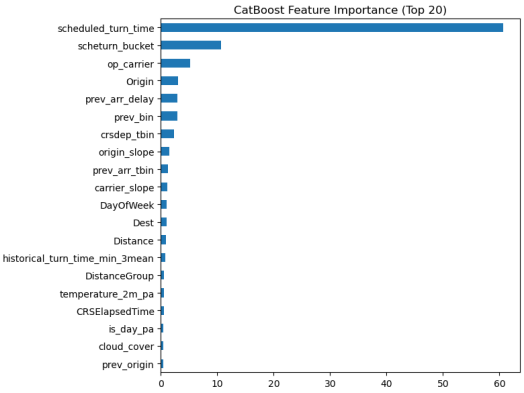

Figure 19: CatBoost Feature Importance