Hurdle Models for Flight Delay Prediction*

Angeline Fung, Garrett Lee, Roy Wang
manching@seas.upenn.edu, glee25@seas.upenn.edu, wangroy@sas.upenn.edu

December 8th, 2025

1 Abstract

As air travel demand is projected to reach historic highs by 2025, the ability to predict delays has become increasingly critical for airlines and airport operators. Prior research in flight delay prediction has predominately addressed the problem as a binary classification task without offering further granularity on the length of predicted delays. As a result, we developed machine-learning models to jointly predict both the occurrence and magnitude of U.S. domestic flight delays using a large-scale dataset comprising over 6.3 million flights from 2019, combined with airline, airport, route, and weather information. Given the severe class imbalance between delayed vs. non-delayed instances, we leveraged two-staged hurdle models that include both classification and regression sub-components. Across all experiments, gradient boosting exhibits the strongest performance. The LightGBM classifier achieves 0.96 accuracy and 0.98 AUROC, while the full hurdle model achieves an MAE of 1.66 minutes, an RMSE of 4.92 and a R^2 of 0.91.

2 Motivation

Delays in air travel impose significant economic, operational, and customer experience costs that impact passengers, airports, and airlines. In 2019, there was a total loss of approximately $33 billion in direct costs to airlines, passengers, lost demand, and indirect costs (FAA/Nextor, as cited in Airlines for America, 2025). The problem is even more relevant today: industry forecasts project that 2025 will be the most busy year in aviation history, with record passenger demand and increased congestion in airports (Gill, 2025). The ability to anticipate delay and severity becomes increasingly important. Doing so can allow airport managers, aviation companies, and staff to change flight schedules well before flight time and prevent costly cancellations. Machine learning offers a principled way to address this challenge by leveraging historical operational and environmental data. Framing the problem through both classification and regression tasks allow airlines to benefit from early warnings of potential disruptions, while duration estimates enable enhanced resource allocation.

3 Related Work

1. "Airline Flight Delay Prediction Using Machine Learning Models" (2021, Tang): Supervised machine learning models were trained on one year of JFK flight data to predict delays as a binary classification task by Tang et al. to address customer dissatisfaction for airlines. The seven algorithms used were Logistic Regression, KNN, Gaussian Naive Bayes, Decision Tree, SVM, Random Forest, and Gradient Boosted Trees. Performance was evaluated using accuracy, precision, recall, and F1-score, with weighting to address class imbalance. Decision Trees performed best (accuracy = 0.9777), while

*Code Repository: https://github.com/angeline-fung/cis_5200_final_project.git

KNN performed worst (F1 = 0.8039). An advantage of this research was the strong performance of multiple evaluation metrics across a diverse set of machine learning models used. Compared to other papers, this one achieved strong predictive performance with a wider variety of supervised learning algorithms. The paper also reinforced an established finding, showing how tree-based ensemble methods outperformed other model types. However, the paper's exploration of only binary classification limits applicability for operational decision-making. Additionally, the discussion of domain-informed feature engineering is limited.

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Logistic Regression	0.8675	0.8850	0.8675	0.8073
KNN	0.8661	<u>0.7501</u>	0.8661	<u>0.8039</u>
Gaussian Naïve Bayes	<u>0.8487</u>	0.8037	<u>0.8487</u>	0.8184
Decision Tree	**0.9778**	**0.9777**	**0.9778**	**0.9778**
SVM	0.8983	0.9067	0.8983	0.8707
Random Forest	0.9240	0.9250	0.9240	0.9126
Gradient Boosted Tree	0.9334	0.9377	0.9334	0.9237

Table 1: Performance results for estimation of flight delays

2. "Airline delay prediction by machine learning algorithm" (2017, Khaksar): Khaksar et al. used Bayesian models, decision trees, clustering, random forests, and hybrid approaches to predict both the occurrence and magnitude of delays. Using U.S. flight data and later adapting the models to a large Iranian airline network, key delay factors were identified: visibility, wind, and departure time in the U.S., and fleet age and aircraft type in Iran. The models achieved over 70% accuracy in both contexts, demonstrating their potential to improve delay prediction, optimize flight planning, and reduce delay propagation. The hybrid approach combining clustering and decision tree performed best in both regions, reaching 71–76% accuracy, while cluster and Bayesian methods showed lower accuracy. Results indicate that combining models (hybrid methods) provides the most reliable delay predictions across different airline systems. One strength of the model is that they address the class imbalance by applying random under-sampling. However, one weakness of the paper is that it mostly explores "classic" ML models. The work focuses on tree-based and Bayesian methods, but there is no exploration of boosting. Moreover, the researchers measure the magnitude of delays by separating delays into discrete classes.

Prediction method	**The US network (%)**		**Iran network (%)**	
	Delay occurrence	**Delay magnitude**	**Delay occurrence**	**Delay magnitude**
J48 pruned tree approach	64.28	63.89	70.87	70.79
Cluster classification approach	62.27	61.37	69.63	68.84
Hybrid approach	71.39	70.16	**76.44**	**75.93**
Bayesian approach	61.35	60.78	70.17	69.81
Random forest approach	67.43	66.27	72.11	71.23

Table 2: Accuracy of flight delay occurrence and magnitude prediction

3. "A hybrid machine learning-based model for predicting flight delay through aviation big data" (Dai, 2024): This technique classifies flights from JFK airport into "on-time", "low-delay", and "high-delay" buckets with a hybrid machine learning approach combining Forward Sequential Feature Selection, density-based clustering, and an optimized weighted random forest model. DBSCAN is used to remedy the challenge of immense volume of delay data by clustering samples into similar groups. Overall, the model achieves 97.2% accuracy. One strength of the paper is that it distinguishes between short-term and long-term delays. Another advantage is its hybrid approach to combine feature selection, clustering, and an optimized random forest. However, the method could be challenging

to deploy due to computationally complexity from multi-stage clustering steps. Also, the dataset is drawn from JFK airport in a limited time window, so generalization to other airports may be limited.

4 Dataset

The dataset for this project contains American domestic flights in the year 2019 [1]. The raw data files given include on-time monthly reporting for domestic US flights, airline and employment information, airport information, and weather data, which we merged together to form the final dataset. The Kaggle dataset provided a binary label DepDel15 for flights being delayed by 15 minutes or more, as well as a discrete feature DepDelayNew measuring the length of delays in minutes. A delayed flight, as defined by the dataset and by industry standard, is a late departure by 15 minutes or more, with flights otherwise considered as 'on-time'. As part of our exploratory data analysis, we examined the distributions of features based on different delay thresholds (e.g. 5, 10, and 15 minutes) to justify choosing a different delay threshold. Ultimately, we could not find substantial evidence that supported using an alternative definition for delays. Our data preprocessing included:

1. Dropping irrelevant features like Cancelled, CancellationCode, CityName, ScheduledArrivalTime, and others
2. Dropping features that cause data leakage, like SecurityDelay and NASDelay that indicate the cause of a delay and would/should not be available in a real scenario
3. Removing rows with negative delays or mismatching DepDel15 and DepDelayNew (e.g. flight was labeled as delayed but the flight was only delayed by 10 minutes)

After preprocessing, our final dataset had $n = 6,357,979$ rows and $p = 49$ features. This includes engineered features, which we will discuss in the next section. We reserved 80% ($5,086,383$ examples) of the dataset for training and 20% (1,271,596) for testing, where each subset retained the proportion of delayed to non-delayed flights as specified by DepDel15.

The dataset exhibits a severe class imbalance; about 17% of flights are classified as delayed (DepDel15 = 1), while about 83% are on-time (DepDel15 = 0). Among the delayed flights, the magnitude of delay follows a heavy-tailed distribution, with the vast majority of delays lasting less than 60 minutes, but rare outliers extending several hours. We found substantial distributional differences between delayed and on-time flights across several features, whereas we did not find large distributional differences between on-time and strictly 0-delayed flights. For example, delays occur more frequently later in the day (Figure 1). These findings informed our problem formulation and feature engineering.

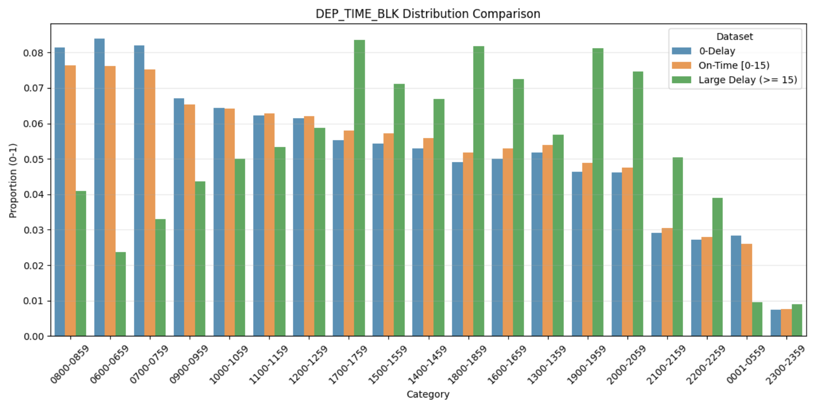

Figure 1: Histogram of flight delay in relation to departure time block.

5 Problem Formulation

5.1 EDA & Feature Engineering

Our exploratory analysis highlighted a strong temporal component to delay likelihood. As shown in Figure 1, the probability of a delay increases steadily throughout the day. This trend is consistent with the concept of delay propagation, where disruptions in early flights ripple through aircraft rotational schedules, causing accumulated delays for later legs. To capture this network dependency, we engineered specific features to represent the unobserved state of the network:

- IncomingRoute: We explicitly linked each flight to its aircraft's previous leg. This allows the model to capture upstream dependencies, distinguishing between 'fresh' aircraft (morning flights) and those arriving from delay-prone corridors.
- RouteName: We encoded the specific origin-destination pair to capture edge-specific risks, such as high-traffic corridors that are frequently bottlenecked.
- CarrierAirport: By combining carrier and departure hub (e.g., 'Delta-Atlanta'), we capture the interaction between airline-specific logistics and airport-specific congestion.

These features help implicitly capture the graph-like structure of the aviation network.

Since our dataset included both binary and discrete labels, we were interested in predicting both the occurrence and magnitude of delays. The large class imbalance in our dataset pointed us towards models that can capture zero-inflated structure. To enforce consistency between the binary and continuous targets, we engineered a new regression target $\text{DepAddedDelay} := \max(0,\ \text{DepDelayNew} - 14)$, which describes the added delay of a flight beyond 14 minutes. Crucially, this ensures that $\text{DepAddedDelay} > 0$ corresponds to $\text{DepDel15} = 1$. As mentioned in Section 4, the insubstantial distributional shifts between flights with 0-delay and on-time flights (0-14min delay) justified this choice. Figure 2 illustrates the long-tailed distribution of the positive values for this new target, which is a structure that our model will need to account for. Engineering this feature DepAddedDelay allowed us to model on-time and delayed flights separately with an architecture designed for zero-inflated data.

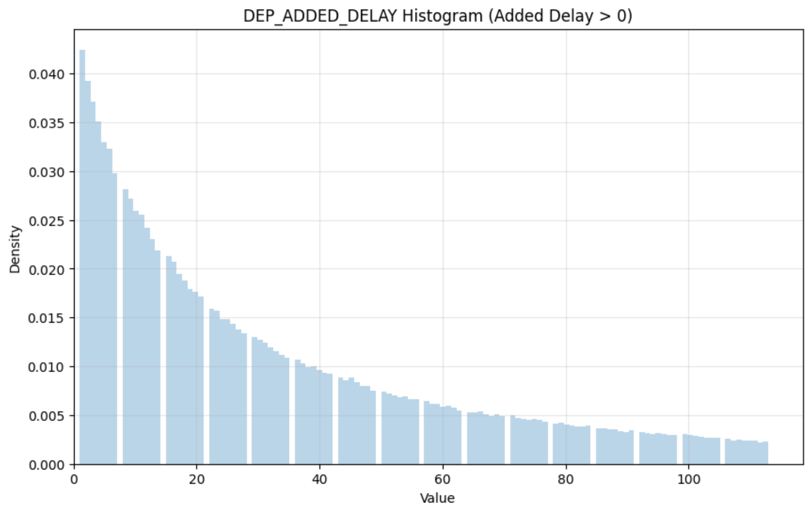

Figure 2: Distribution of positive DepAddedDelay

5.2 Hurdle Model

We formulate flight delay prediction as a supervised regression task, where the primary objective is to estimate the expected added delay beyond 14 minutes for a given flight. Formally, we want to model $\mathbb{E}[Y|X]$, where Y is the added delay of our input flight data X. Given the zero-inflated, semi-continuous nature of the target variable, we posit that the occurrence of a delay and the magnitude of a delay are governed by distinct generative processes. As such, we apply the Law of Total Expectation to decompose our target:

$$\begin{aligned}\mathbb{E}[Y \mid X] &= \mathbb{P}(Y > 0 \mid X) \cdot \mathbb{E}[Y \mid Y > 0, X] + \mathbb{P}(Y = 0 \mid X) \cdot \mathbb{E}[Y \mid Y = 0, X] \\ &= \mathbb{P}(Y > 0 \mid X) \cdot \mathbb{E}[Y \mid Y > 0, X]\end{aligned}$$

since $\mathbb{E}[Y \mid Y = 0, X] = 0$. With this formulation in mind, we employ a hurdle model (or two-staged model) to capture the zero-inflated structure of our target. This architecture manifests the above factorization into two sub-components: a binary classifier that models the probability $\mathbb{P}(Y > 0 \mid X)$ of crossing the 'hurdle' (e.g. on-time), and a regressor that models the conditional expectation $\mathbb{E}[Y \mid Y > 0, X]$ of positive delays given that the hurdle is crossed (e.g. a delay occurs).

We can also attempt to model the zero-inflated structure with only a single component, which we will discuss more in Section 6.

6 Methods

We implement and compare four distinct regression models to solve this problem. In our hurdle framework, the classification model is trained on the entire distribution of flights to estimate $\mathbb{P}(Y > 0 \mid X)$. The regression component is trained only on the true positives to estimate $\mathbb{E}[Y \mid Y > 0, X]$, since flights with added delay 0 are excluded from this conditional expectation by definition. During evaluation, both models report their predictions on the full test dataset, and those outputs are multiplied to obtain the final prediction. For our exploratory data analysis and visualization, we used NumPy, Pandas, Matplotlib, and Seaborn. We implemented our training and preprocessing pipelines with sckit-learn, leveraging classes like Pipeline, ColumnTransformer, and building our own custom transformers.

6.1 Baseline Methods (Single-Stage Regressors)

These models attempt to predict the final delay minutes in a single step.

- No-Skill Baseline: Given large class-imbalance of our data, an "always on-time" baseline is a simple benchmark for how our models perform, with an accuracy equal to the proportion of the majority-class (0.83).
- Ridge Regression: We'll implement ridge regression by minimizing L_2-penalized mean squared error (MSE). This will be one of our benchmarks, as the model is expected to fail due to its inability to handle the zero-mass and non-normal error distribution.

$$\hat{y}_i = \mathbf{w}^T \mathbf{x}_i + b, \quad \min_{\mathbf{w},b} \left(\sum_{i=1}^{N} (y_i - \hat{y}_i)^2 + \alpha ||\mathbf{w}||_2^2 \right)$$

- Random Forest: We'll also implement a nonlinear baseline, ensembling decision trees to form a Random Forest. We hope that the Random Forest can learn the zero-inflation structure through averaging many overfit decision trees. We'll also tune the hyper-parameters of the Random Forest model by running a randomized search to obtain the best hyper-parameters. We used `poisson` as our criterion for splits.

- LightGBM Regressor with Tweedie objective: Given the point mass at zero and the long-tailed nature of DepAddedDelay, it's reasonable to assume the $Y|X$ is Tweedie-distributed. The Tweedie distributions are a family of exponential dispersion models (EDM) defined by a variance power parameter, p, which dictates the relationship between the variance and the mean, $\mathrm{Var}(Y|X = x) = \phi\mu(x)^p$, where μ is the mean and ϕ is a dispersion parameter. When $p = 1$ ($p = 2$), this becomes a Poisson (Gamma) distribution. When $1 < p < 2$, this becomes a compound Poisson-Gamma process, which should fit our data well given our analysis.

  To estimate the conditional mean $\mu(\mathbf{x})$, we employ the LightGBM framework. The objective function when $1 < p < 2$ is the defined as the negative log-likelihood of the data summed over all N samples:

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^{N} \left(-y_i \frac{\hat{y}_i^{1-p}}{1-p} + \frac{\hat{y}_i^{2-p}}{2-p} \right)$$ [1]

  Note that this loss function gives less weight to large deviations between the true and predicted values: for heteroscedastic data, this implies that we are accounting for the greater uncertainty associated with large outliers. By setting the objective to `tweedie` regression (with a log link), LightGBM constructs an ensemble of decision trees to minimize this deviance. For our experiments, we set $p = 1.5$. We used Optuna to perform a hyperparameter search, and set aside 15% of the training set as a validation set during training to track performance.

6.2 Baseline Hurdle (Logistic and Linear Regression)

To operationalize the hurdle model framework that we previously described, we first implemented a two-stage predictive pipeline using the most classical statistical learners. In the first stage, we modeled the classification with logistic regression. This was a good starting point for interpretability and ease of implementation for binary classification. Next, we estimated the expected added delay magnitude in minutes using regularized linear regression, evaluating Lasso and Ridge variants to balance accuracy with coefficient stability. A hyperparameter search was conducted using the RandomizedSearchCV class from scikit-learn. The model was trained after removing features with extremely high VIF, cyclical encoding time-sensitive columns, and removing significant input outliers. Although the results were not ideal, this approach preserved the conceptual interpretation of a hurdle model using computationally lightweight methods. Based on our hyperparameter search, we found that using ridge penalization and class balancing were the most effective hyperparameter settings.

6.3 Random Forest Hurdle

We also implemented a hurdle model using Random Forests for both the classification and regression components. This approach is well-suited for the heterogeneous structure of flight operations data. Tree ensembles naturally capture complex interactions and are more robust to multicollinearity. The hyperparameters were tuned using the RandomizedSearchCV class in the sklearn suite. For the classifier, we used entropy as our criterion and the results were better than our baseline model across all several metrics: accuracy, precision, recall, area under receiver operating characteristic curve (AUROC), and area under precision recall curve (AUPRC). On the other hand, we used `poisson` (analogous to Tweedie regression in Section 6.1 with $p = 1$ and a slightly different loss function) as our criterion for the regressor to capture the long-tailed and zero-inflated nature of flight delays.

[1] The first term in the Tweedie deviance involving $\max(y_i, 0)^{2-p}$ is irrelevant to the loss since it does not depend on the model. `https://scikit-learn.org/stable/modules/model_evaluation.html#mean-poisson-gamma-and-tweedie-deviances`

6.4 LightGBM Hurdle

As we have previously discussed, flight delays are the result of complex interactions and caused by a variety of factors that result in a nonlinear relationship between our features and the expected added delay. We chose to implement the LightGBM gradient tree boosting framework to learn the non-linear structure in our data. As a scale-invariant architecture, decision tree-based frameworks can naturally handle a mixture of categorical, continuous, and discrete variables without the need for extensive data pre-processing. LightGBM can also handle categorical features internally without additional encodings, which was well-suited for our data due to some of our categorical features including thousands of unique categories. Training time and compute requirements were additional reasons for choosing LightGBM over other gradient tree boosting options like XGBoost; for large datasets, the leaf-wise tree growth that LightGBM implements can converge faster than the depth-wise tree growth that XGBoost implements.

Since the hurdle regressor will be trained only on the positive examples in the training set, we selected the `gamma` objective with LightGBM (analogous to Tweedie regression in Section 6.1 with $p = 2$ and a slightly different loss function) to fit the skewed distribution of positive examples (Figure 2). We performed hyperparameter searches using Optuna for the LightGBM Classifiers and Regressors, training the classifier on the entire training set while training the regressor on the examples with positive delays in the training set. We also set aside 15% of these positive examples as a validation set during training to track performance.

7 Experiments and Results

The LightGBM models were initialized primarily with the hyperparameters given by the best performing trial from an Optuna study, except for a few key choices:

- We implemented early stopping to prevent overfitting, where training was stopped if the validation loss (as defined by the given objective) did not improve for 100 iterations, at which the model state at the best iteration was returned. To give the models a chance to stop early, we generally overrode the learning rate to be low (< 0.1) and the number of iterations to be high (> 1000) unless chosen otherwise by Optuna.

For the classification subtask, we choose performance metrics that account for (or are more meaningful in the presence) of large class imbalance, such as balanced accuracy (bACC), recall, and area under precision recall curve (AUPRC). We will still report accuracy (ACC), precision, and area under receiver operating characteristic curve (AUROC), but note that these are both less meaningful for our problem.

For the overall regression task, we chose performance metrics that are interpretable, naturally leading to mean absolute error (MAE) and root mean squared error (RMSE), which are measured in minutes in this context. Since our target is highly skewed, we will focus on MAE for its resistance to outliers, but will still report RMSE and the coefficient of determination (R^2). Below are the summarized results for the classification and regression tasks. The decision threshold for all of our classifiers is 0.5, and the metrics and confusion matrices built upon binary output are computed with the model outputs corresponding to this threshold.

Classifier	**bACC**	**ACC**	**Precision**	**Recall**	**AUROC**	**AUPRC**
Baseline (No-Skill)	0.5	0.83	NaN	0.0	N/A	N/A
Baseline Logistic	0.63	0.62	0.26	0.65	0.68	0.31
Random Forest	0.76	0.76	0.40	0.77	0.84	0.59
LightGBM	0.89	**0.96**	**0.94**	**0.80**	**0.98**	**0.78**

Table 3: Classifier Subcomponent Results

Regression Model	**MAE**	**RMSE**	$\mathbf{R^2}$
Baseline (No-Skill)	5.48	17.72	−0.11
Baseline Ridge Regression	8.79	16.48	0.04
Random Forest Single-Stage Regressor	5.76	11.28	0.55
LightGBM Single-Stage Regressor	2.43	6.79	0.84
Baseline Hurdle (Logistic + Linear)	6.31	17.24	0.02
Random Forest Hurdle	10.89	14.23	0.29
LightGBM Hurdle	**1.66**	**4.92**	**0.91**

Table 4: Regression Component Performance Comparisons

Model	**Logistic Regression**		**Random Forest**		**LightGBM**	
True (Col) / Pred (Row)	On Time	Delayed	On Time	Delayed	On Time	Delayed
On Time	649,947	402,392	795,268	257,071	1,041,152	11,187
Delayed	77,578	141,679	49,912	169,345	44,507	174,750

Table 5: Model Confusion Matrices: the first row consists of TN and FP, and the bottom row consists of FN and TP

The performance of our ridge regression demonstrates the limitations of linear models for this zero-inflated dataset, performing worse than the No-Skill baseline in terms of MAE. This suggests that the assumption of Gaussian errors leads to predictions that are less accurate than even a constant, majority-class prediction. On the other hand, both Random Forest variants (single-stage and hurdle) outperform the baseline models. The single-stage regressor offers a substantial improvement over the no-skill and Ridge baselines, reducing RMSE from 17.72 to 11.28 and improving the R^2 score from negative or near-zero to 0.55, indicating it captures meaningful nonlinear signal absent in the simple baselines.

However, our experimental results demonstrate a consistent advantage for the gradient-boosted LightGBM compared to all other models. In the classification stage, LightGBM achieved the highest overall performance, exceeding logistic regression and random forests across all reported metrics (Table 4). Crucially, the AUPRC highlights the disparity in how these models handle the class imbalance. While the baseline logistic regression achieved a recall comparable to tree-based methods (0.65 vs 0.77 and 0.80), its AUPRC of 0.31 reveals that this sensitivity was achieved by aggressively over-predicting delays. Table 5 confirms this tradeoff: the logistic model generates over 400,000 false positives, predicting roughly three false positive for every true delay identified. The Random Forest classifier improves the discrimination substantially, raising the AUPRC to 0.59. Ultimately, the LightGBM classifier performs the best in terms of AUPRC (0.78), demonstrating that it can maintain high recall (0.80) without sacrificing precision (0.94). This indicates that LightGBM is far superior at isolating the specific signal of delay-prone flights without being overwhelmed by the noise of the majority class.

8 Discussion and Conclusion

The baseline hurdle model offers an interpretable framework that decomposes the delay prediction into occurrence and magnitude. However, the logistic and linear models struggle with the severe class imbalance and zero-inflated dataset structure, reflected by near-zero R^2 values and an MAE (6.31) worse than the No-Skill baseline. Thus, the baseline model serves as a primary point of comparison to highlight significant performance improvement by other models. On the other hand, tree-based methods perform significantly better across all metrics, showing their ability to capture the heterogeneity in our set of features.

The LightGBM hurdle model was the best performer across MAE, RMSE, and R^2, notably outperforming the single stage LightGBM regressor. By allowing the regression component to specialize exclusively on the distribution of positive delays, the hurdle model captured the variance of the "magnitude" process more effectively than the single-stage LightGBM model, which is forced to balance the zero-mass and the tail within a single objective function. Collectively, our results affirm that the hurdle architecture can be effective in providing accurate and robust predictions for delay occurrence and magnitude, particularly with gradient boosted trees as the underlying framework.

To interpret the drivers of our model's predictions, we analyzed global feature importance using Shapely values (Figures 3 and 4). We first clustered features using hierarchical clustering based on Spearman's rank of correlation coefficient to account for collinearity and nonlinear relationships between features. Across both the classification and regression components, Time of Day (DepTime, DepHour, and DepTimeBlk) emerged as the dominant predictor. This aligns with the concept of delay propagation, where minor disruptions accumulate throughout the day and increase the probability and severity of delays in the evening. Other informative features included Origin & Route (DepartingAirport and RouteName), Previous Airport Info (PreviousAirport and IncomingRoute), and Carrier Info (CarrierAirport and CarrierName), validating the use of our engineered features and confirming that aviation network effects are valuable for delay prediction.

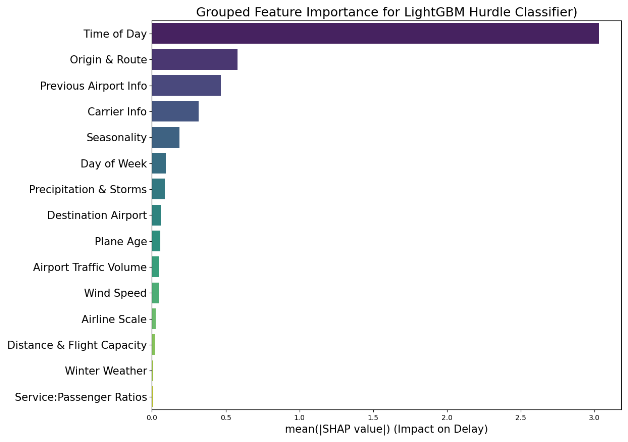

Figure 3: LightGBM Feature Importance (Hurdle Classifier)

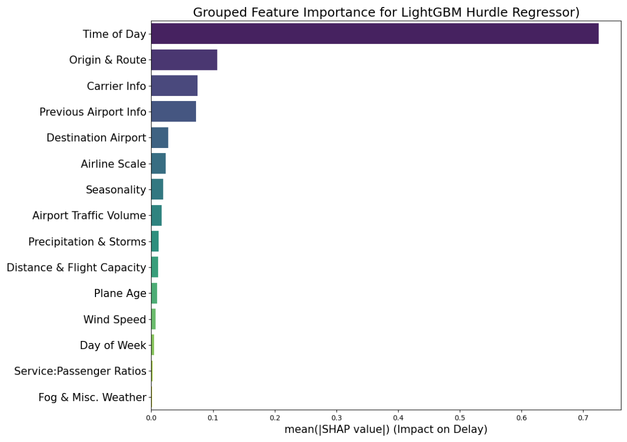

Figure 4: LightGBM Feature Importance (Hurdle Regressor)

Bin	**MAE**	**RMSE**	**Bias**	**Count**
$<$ 15m (On-time)	0.30	1.14	0.30	1,052,339
15-30m	4.43	6.13	-0.81	89,299
30-60m	9.51	12.16	-5.75	73,160
60-120m	12.05	15.88	-6.93	53,502
120m+	17.86	22.71	-14.54	3,296

Table 6: LightGBM Hurdle Model Detailed Errors. Bias = Mean(True − Pred).

Table 6 offers more insights into the prediction errors of the LightGBM Hurdle model across different magnitudes of delay. We observe a clear trend of increasing negative bias with delay severity, with the model systematically underestimating large delays. This reflects a common challenge with regression on heavy-tailed distributions, where the model tends to predict conservative mean values to minimize penalized errors. However, for the vast majority of operations, specifically on-time flights and flights delayed by no more than 30 minutes, the hurdle model remains highly calibrated with minimal bias.

In addition, when analyzing our missed delays in relation to the departure time, we noticed that the LightGBM classification model performs significantly worse during earlier parts of the day compared to later hours (Figure 5). With fewer historical instances of morning delays to learn from, the model struggles to distinguish risk factors in this window. To capture this remaining structure, future work could focus on engineering features that capture patterns in morning delays, such as mechanical or crew-related issues. Moreover, it would be interesting to explore if different decision thresholds for the LightGBM classifier would improve its ability in capturing these delays, or if tuning decision thresholds dynamically based on the time of day would improve our morning miss rate.

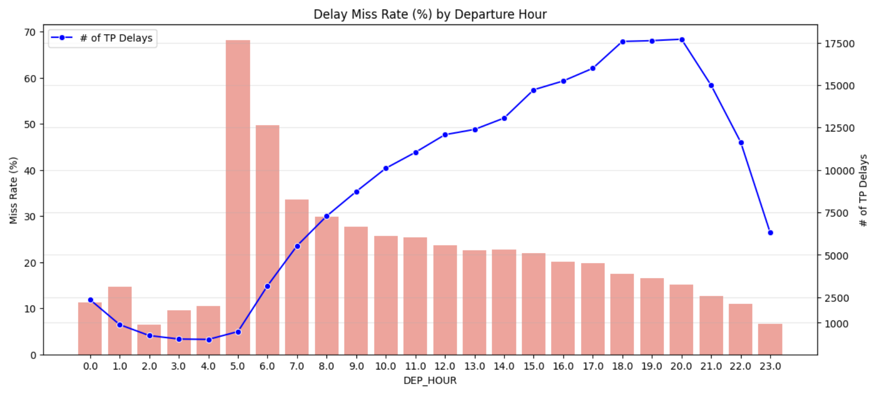

Figure 5: Distribution of false negative predicted delays by departure hour (bars) and the volume of TP predicted delays (blue line and points)

Future work would also expand our model implementations beyond tree-based approaches. Given the data-rich nature of this task, we see the potential of neural networks in capturing complex relationships in this dataset at the expense of explainability. A contrasting approach would be to implement generalized linear models (GLM), which offer superior interpretability and are less computationally expensive. This includes evaluating their performance in a single-stage framework, such as a Tweedie GLM for jointly modeling the discrete-continuous structure of delay outcomes, as well as leveraging a Gamma GLM as the continuous component within a two-stage hurdle model. These implementations would allow us to compare the more interpretable GLMs against our tree-based ensemble methods, which may provide additional insights into the underlying distributional characteristics of flight delays.

9 References

1. Tang, Yuemin. (2021). Airline Flight Delay Prediction Using Machine Learning Models. In 2021 5th International Conference on E-Business and Internet (ICEBI 2021), October 15-17, 2021, Singapore, Singapore. ACM, New York, NY, USA, 7 Pages. https://doi.org/10.1145/3497701.3497725

2. Dai, M. (2024). A hybrid machine learning-based model for predicting flight delay through aviation big data. Scientific Reports, 14, Article 4603. https://www.nature.com/articles/s41598-024-55217-z

3. Airlines for America. (2025). U.S. passenger carrier delay costs.
Retrieved from https://www.airlines.org/dataset/u-s-passenger-carrier-delay-costs/

4. Wadkins, Jen. (2019). 2019 Airline Delays w/ Weather and Airport Detail. Kaggle.com. Retrieved from https://www.kaggle.com/datasets/threnjen/2019-airline-delays-and-cancellations/data.

5. Gill, Rob. (2025). Global air traffic achieves new record high in summer 2025. Kaggle.com. Retrieved from https://www.businesstravelnewseurope.com/Air-Travel/Global-air-traffic-achieves-new-record-high-in-summer-2025.