(https://databricks.com)

Phase 3 Submission

Team and Project Meta Information

Project Title

AirTime

Group Number

2-4

Team Information

Sean Wei	Marlon Fu	Eric Jung	Hao Xie	Parker Brailow
seanwei2001 @berkeley.edu	marlonfu @berkeley.edu	erijung @berkeley.edu	haoxie912 @berkeley.edu	brailowparker @berkeley.edu
				

Updated Credit Assignment Plan

Task	Contributor(s)	Man Hours	Start Date	End Date
Exploratory Data Analysis	Marlon, Sean, Hao, Parker	20	10/27	11/12
Data cleaning and key transformations	Sean, Parker	20	11/06	11/17

Task	Contributor(s)	Man Hours	Start Date	End Date
Temporal, numerical, and categorical feature exploration and engineering	Parker, Sean	20	11/06	12/10
Graph-based features exploration and engineering	Marlon	15	11/06	12/10
Azure Blob Storage Setup, Data processing pipeline	Eric	15	10/27	11/17
Building Cross Validation and Hyperparameter Tuning Pipeline	Hao, Eric	20	11/24	12/10
Building Model pipelines and running experiments	Hao, Eric	30	11/24	12/10
Loss function selection and results analysis	Hao, Sean	5	12/01	12/10
Final Presentation	Marlon, Sean, Hao, Parker, Eric	20	12/07	12/10

Abstract

Every year, travelers waste precious time waiting for delayed flights. These delays cause inconvenience and frustration, impacting entire travel plans and forcing passengers to miss connecting flights. We hypothesize that machine learning pipelines integrating transportation and weather data with custom-engineered features can accurately predict flight delays and delay durations up to 2 hours before departure. Using data from the US Department of Transportation (2015–2021) and weather data from the National Oceanic and Atmospheric Administration (NOAA), we extracted operational and environmental features. These included graph-based features to capture delay propagation patterns, temporal features to account for trends such as day-of-week effects, and categorical features such as trends within airlines. A key innovation in our approach was incorporating graph-based features, such as PageRank centrality and delay-weighted edges, to model delay propagation. Despite challenges with skewness, these features provided predictive value. Our data

preprocessing pipeline handled missing values through rolling averages, and we explored feature selection with 818 features in the final dataset after exploding one-hot encodings.

Due to the skew of our data and the abundance of high outliers, we prioritized mean absolute error (MAE) as our primary regression metric over root mean squared error (RMSE), as RMSE is more sensitive to outliers caused by extreme delays. Our group was also conscious of the proportion of overpredictions, as overprojecting a delay may lead to customers arriving late to the airport, although we did not specifically tune models to take these factors into account.

For modeling, we compared classical linear regression and advanced ensemble models, including random forests and gradient-boosted trees (GBT). A baseline linear regression produced an MAE of 16.59 minutes, with 76.9% of predictions overestimating delays. Gradient-boosted trees outperformed other models with an MAE of 13.87 minutes, and 80.7% overpredictions. Hyperparameters were tuned with MAE as the primary metric and was performed using Tree of Parzen Estimators (TPE) with a blocking time series cross-validation split. Key hyperparameters for GBT included a maximum depth of 7, minimum instances per node of 3, and a learning rate of 0.15. We also trained a multi-layer perceptron (MLP) classifier to predict whether flights would be delayed by more than 15 minutes, achieving a precision of 0.624.

Ultimately, our best model—the gradient-boosted tree—offers actionable predictions for travelers by forecasting delay times in minutes. By prioritizing underprediction over overprediction and focusing on MAE to mitigate the effects of skewness, we provide consumers with more reliable estimates, helping them make informed decisions such as delaying their airport arrival to reduce wait times and boost satisfaction.

Data Lineage and Key Transformations

The dataset utilized in this project was compiled from two primary sources:

1. US Department of Transportation: Provided detailed operational flight data, including airline information, origin and destination airports, taxi times, and delay

metrics such as `DEP_DELAY_NEW` and `DEP_DEL15`. This data captured essential operational characteristics of flights.

2. National Oceanic and Atmospheric Administration (NOAA): Contributed weather data, including variables like rainfall, temperature, and wind speed, which allowed us to incorporate environmental conditions affecting delays. Key transformations performed on the data included:
    - Joining Datasets: Flight and weather data were merged using common attributes such as date and airport location to ensure temporal and spatial alignment.
    - Handling Missing Values: Missing data was imputed using rolling averages to reduce bias and maintain consistency in the dataset.
    - Exploding One-Hot Encodings: Categorical variables, such as airline and airport types, were one-hot encoded. This expanded the feature space, increasing the number of columns to 818 in the final dataset.

Once the merged dataset was created, the following key steps were implemented as part of data cleaning and transformation:

1. All rows with missing `DEP_DELAY_NEW` values (1.5% of 5 year data) were dropped from the dataset.
2. Columns that were entirely or almost entirely missing were dropped from the dataset.
3. All `DEP_DELAY_NEW` values greater than 720 (12 hours) were bounded to 720.
    - Our group decided that this was a high enough threshold where we can reasonable assume customers would want to alter their travel plans for any time longer than 12 hours.
    - Flights longer than 12 hours accounted for less than 0.05% of data, so the original distribution was not drastically changed.
    - This transformation helps mitigate the impact that abnormally high delay times have on our target variable and predictions.
4. Numerical columns had null values imputed with a rolling mean (the mean value

Data and Feature Engineering

In this section, we explore four key categories of features used to predict flight delays: Categorical Features, Time-based Features, Numeric Features, and Graph-Based Features. Each category provides unique insights into different aspects of airline operations, contributing to the overall predictive capability of our models.

1. **Categorical Features**: Captured the contextual factors influencing flight delays, such as airline and airport types. By encoding these features, we analyzed patterns and identified relationships between categories and delay times.
2. **Time-based Features**: Captured periodic patterns in flight schedules and delays such as time of expected departure, day of week, month of year, quarter of year, etc.
3. **Numeric Features**: Included prior departure delays, arrival delays, taxi times, air times, and others to capture sequential dependencies in flight operations.
4. **Graph-Based Features**: Network analysis introduces a structural perspective on flight delays by leveraging relationships between airports and operators.

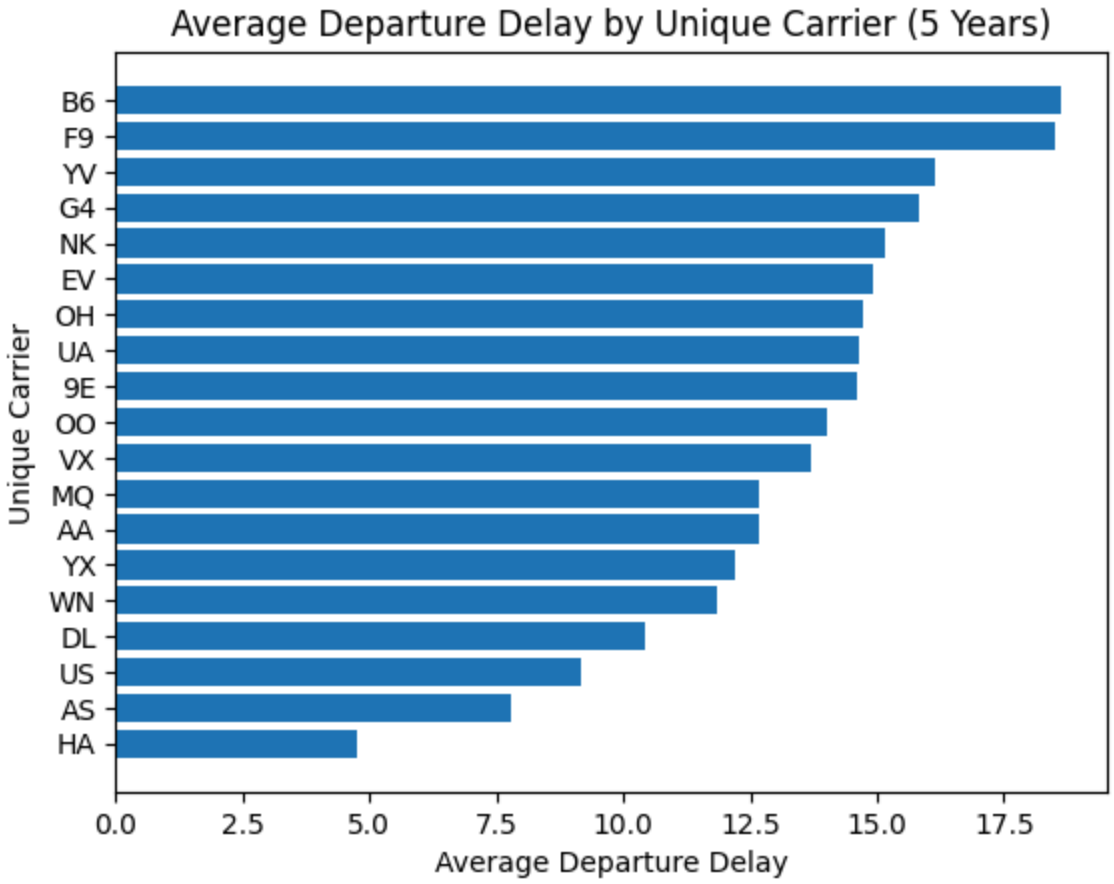
By combining these feature sets, we provide a holistic view of delay predictors, capturing both contextual and structural influences to improve model accuracy.

Categorical Features

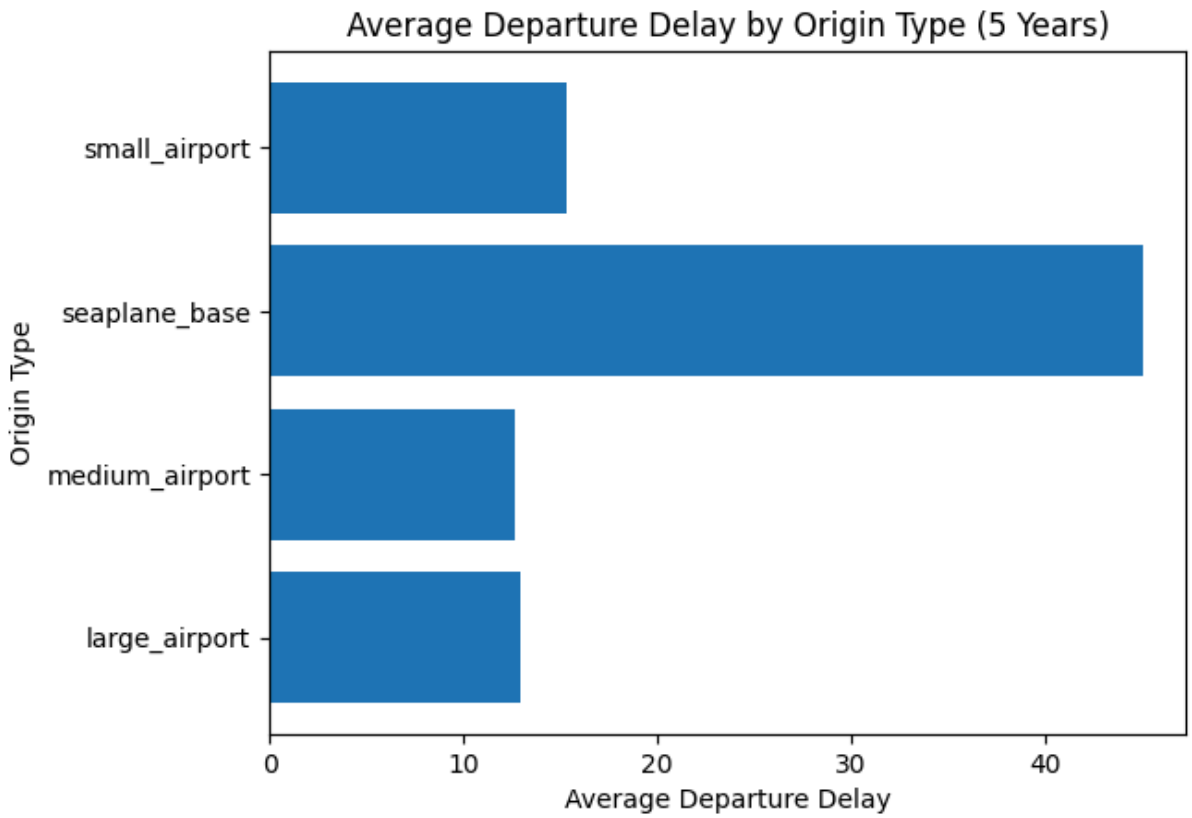
Categorical features are a key component of airline data. When customers often use their intuition to estimate their travel time, factors such as the airport they are flying from and what airline they are traveling with come into play.

The categorical features that were one-hot encoded and used in our models were:

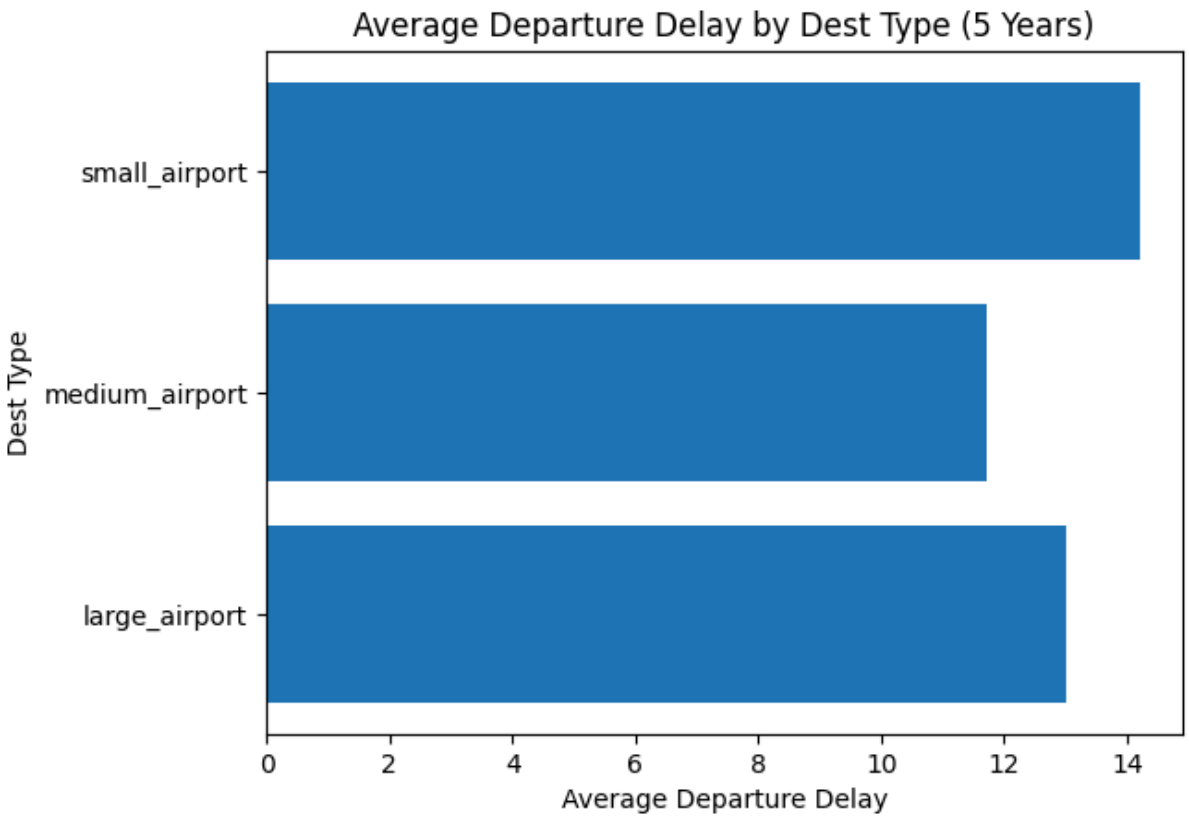
- `OP_UNIQUE_CARRIER`: **the airline for a given flight.** The bar chart below shows how some carriers are prone to longer delays on average than others, with the largest difference ranging over 12 minutes.



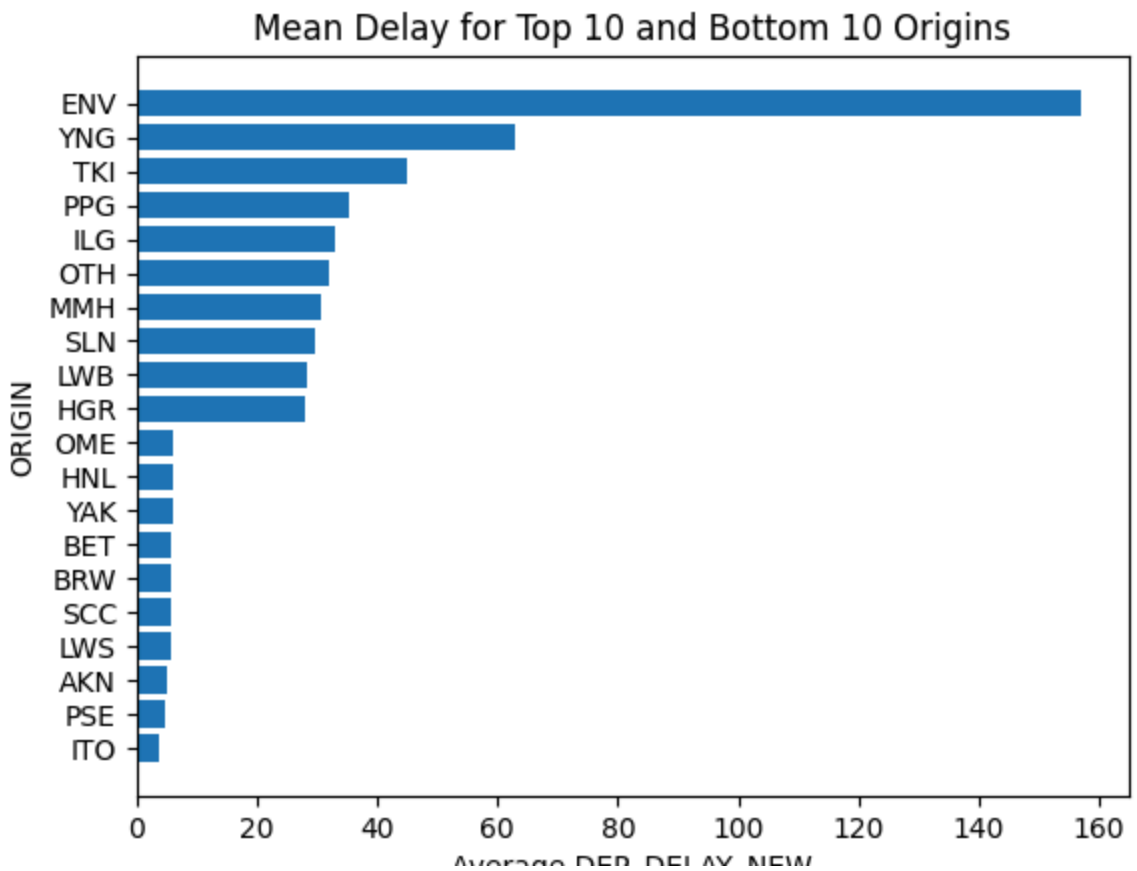
- `origin_type`: **the type of airport the flight departed from.** The most noticeable distinction here is the largely increased average delay for seaplane bases as compared to traditional airports. Additionally, there is a slightly higher delay for small airports as compared to other airport types, likely due to small airports having fewer resources dedicated towards security, luggage loading, and boarding.



- `dest_type`: **the type of airport the flight arrived at.** Once again, small airports show a slightly higher average delay as compared to other airports. While the difference in airport types is not as distinct, the group felt it was still worth exploring as a feature for the model.



- `ORIGIN`: **the airport the flight departed from.** The figure below depicts the 10 highest and lowest average delays per origin airport. There is a very distinct and noticeable difference between the high-delay airports and the low-delay ones.

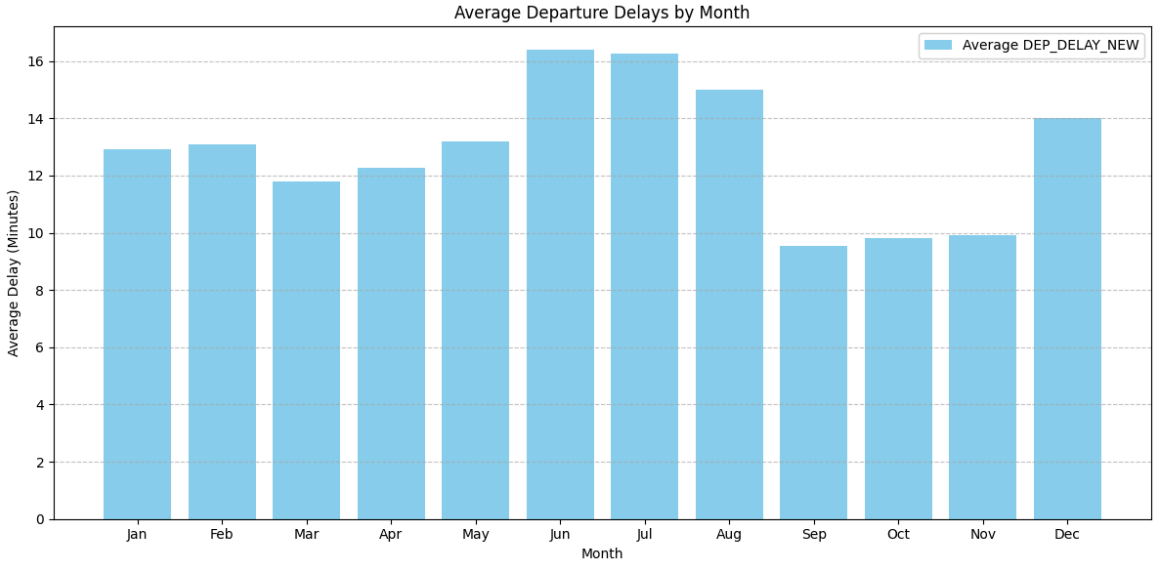


Time-based Features

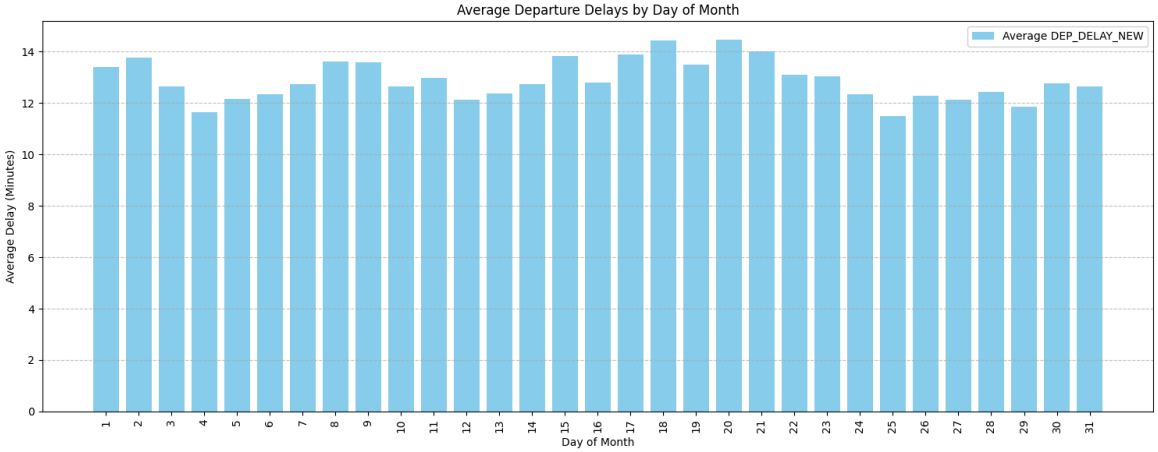
Time based features included in our analysis and as features for our models included:

- `MONTH`: **the month in which the flight departed.** The bar chart below displays the average flight delay by month. We notice that some months appear to have significantly greater delays than others. Surprisingly, the seemingly busiest

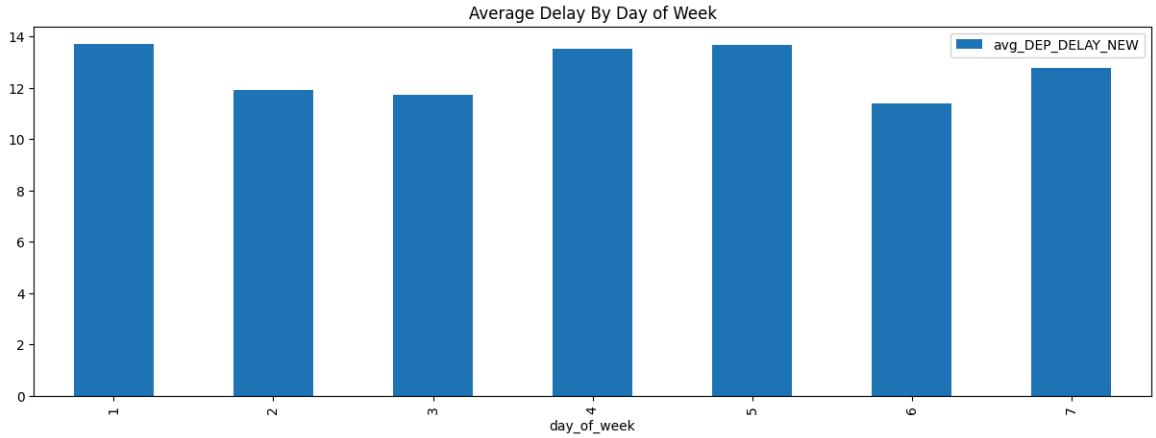
holiday travel months dont have as great of a delay as some of the summer months.



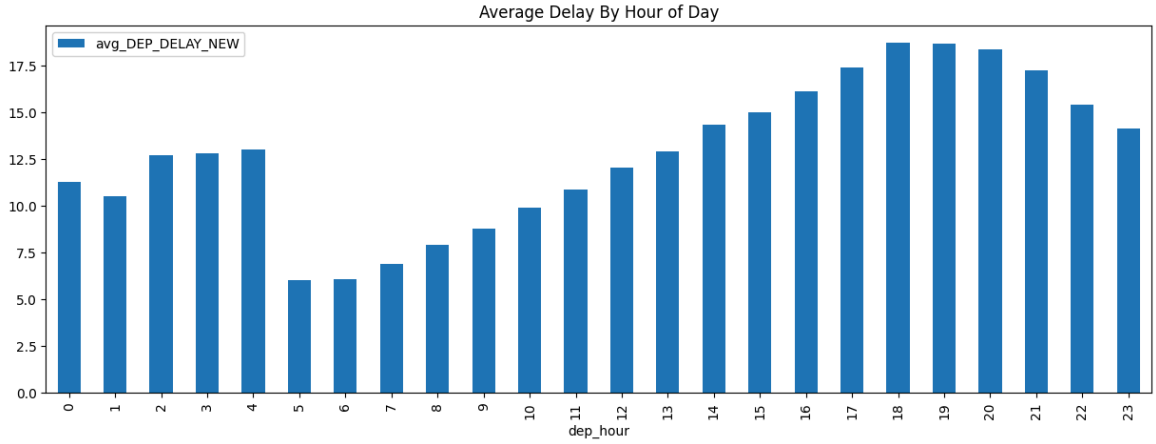
- `DAY_OF_MONTH`: **the day of month in which the flight departed.** The bar chart below displays the average flight delay by day of month. It appears that there may be a cyclical pattern, hence we look at the day of week next.



- `DAY_OF_WEEK`: **the day of week in which the flight departed.** The bar chart below displays the average flight delay by day of week. Some days appear to have significantly shorter flight delays than others.



- `Hour_Of_Day`: **the hour of day in which the flight departed.** The bar chart below displays the average flight delay by hour of day. There appears to be quite an interesting relationship with time of day here so we used this in our analysis as a quanititative variable but look at the absolute time of day instead of just the hour.

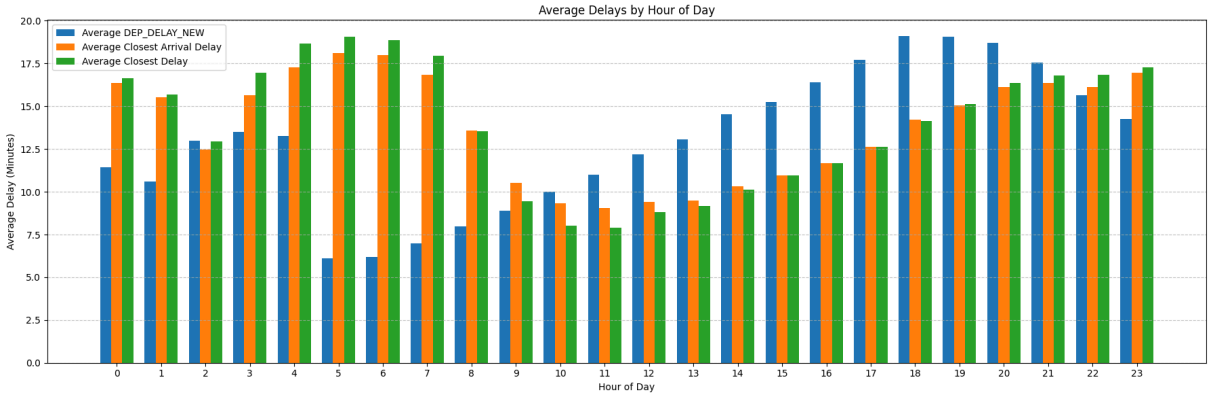


Numeric Features

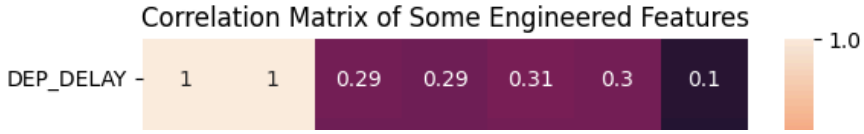
- `closest_delay_before_2_hours_new`: **the closest departure delay (at least 2 hours before) of the same plane as the flight we are inspecting. (Zeroes imputed for negative delays.)**

- `closest_arrival_delay_before_2_hours_new`: **the closest arrival delay (at least 2 hours before) of the same plane as the flight we are inspecting. (Zeroes imputed for negative delays.)**

The bar chart below displays the average closest departure and arrival delay as well as the departure delay of the inspected flights by hour of day. We can see that largely these are mirrors of one another but shifted by 2 hours with the exception of flights that arrive/depart within the 2 hour time gap; hence, why it is not simply the same plot shifted.



Furthermore, to explore the relationships some of these engineered features may have, we plotted a correlation heatmap. We see that the strongest correlations for `DEP_DELAY_NEW` are with the previous arrival and departure times.



Graph-based Features

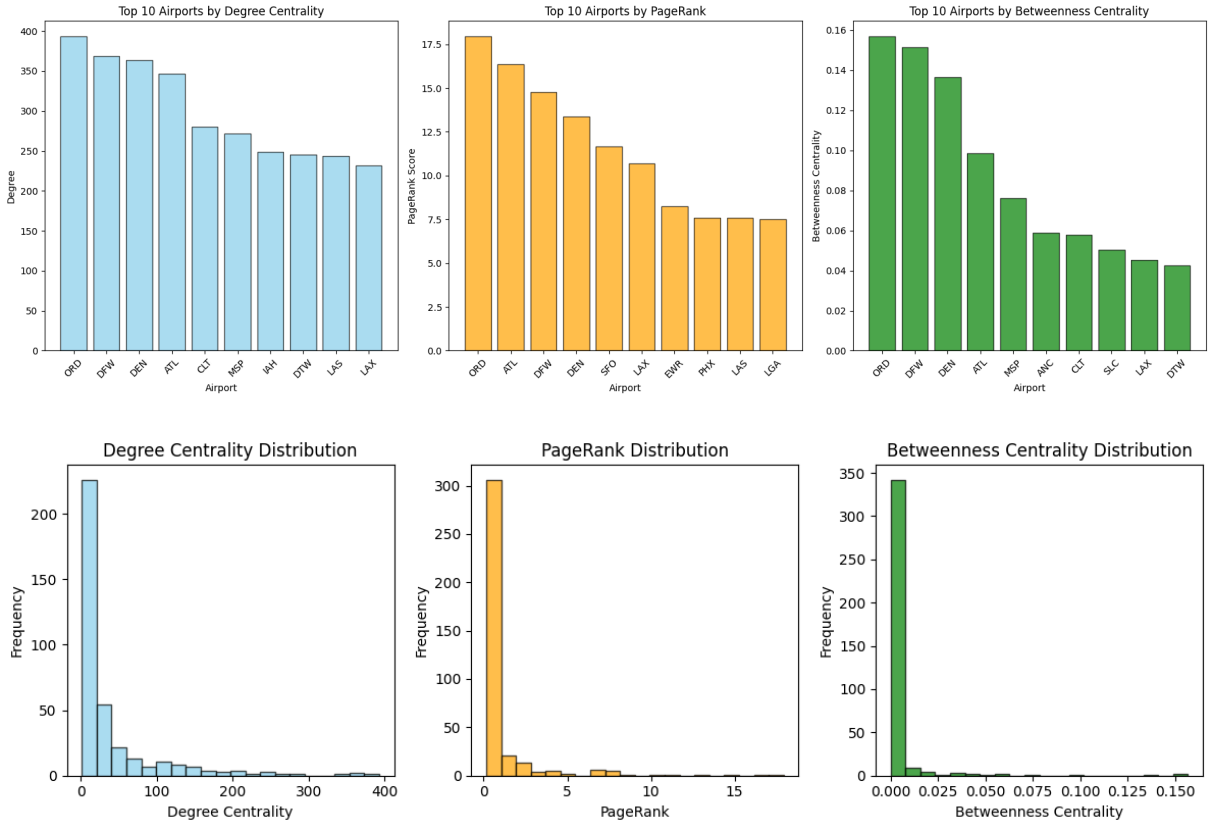
Graph features can provide valuable insights into the structural importance of airports within the flight network. For instance, airports with high degree centrality may experience congestion due to the sheer volume of connections, increasing the likelihood of delays. In addition to degree centrality, this section explores several other graph-based features. By incorporating these graph-based features, we hope to better understand the role of network dynamics in contributing to flight delays and improve the accuracy of predictive models.

To operationalize graph features for predicting flight delays, we use **GraphFrames** and **NetworkX**, which are Spark-based libraries that enable scalable computation of graph metrics. In this project, we explore two different graph configurations:

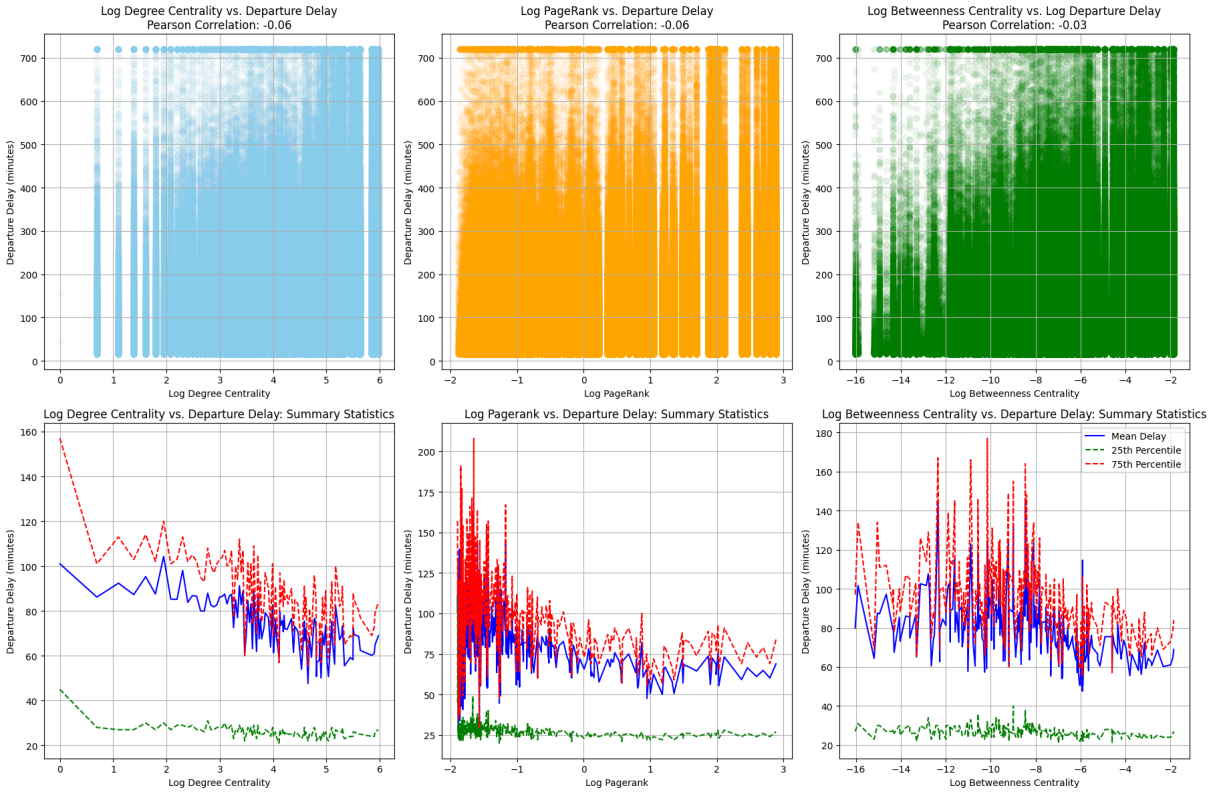
1. Airports and Route Based

The motive for this configuration is to use airport traffic to determine airport importance, and to explore any associations between flight delays and the importance of an airport in a network. The first graph is configured as follows:

- Nodes (vertices): Represent airports, uniquely identified by their IATA codes as seen in `ORIGIN` and `DEST`.
- Edges: Represent individual flights between airports, with a connection between each `ORIGIN` and `DEST`. These edges are unweighted.



Using this graph representation, we compute three graph metrics to measure each airport's importance: **degree centrality**, **PageRank**, and **betweenness centrality**. We observe that there is a large overlap of top ranked airports by each of these metrics in the first row of visualizations. In addition, we note that the distribution of graph metrics is highly skewed, with a large tail to the right. This suggests that very few airports are ranked as being highly important (likely the same ones that we see in the visualizations in the top row), while the vast majority of airports are ranked as being equally insignificant to the overall network. Thus, when plotting these metrics against individual flight delay times, we take the natural logarithm using `np.log1p`.



When plotted against the departure delays times (`DEP_DELAY_NEW`) of individual flights, it is difficult to visually interpret the association of each graph metric with the the target variable as seen in the top visualization. Firstly, we observe that there are weak negative Pearson correlations for all graph metrics (-0.03 for betweenness centrality, -0.06 for PageRank, and -0.06 for degree centrality). In addition, the cardinality of each graph metric is significantly smaller than the cardinality of departure delay times, which results in vertical strips of plotted points.

To better summarize this relationship, we produce the three visualizations in the second row, which showcase the mean, 25th percentile, and 75th percentile of departure times for each value of graph metrics. There are two main observations we make from these summary statistics:

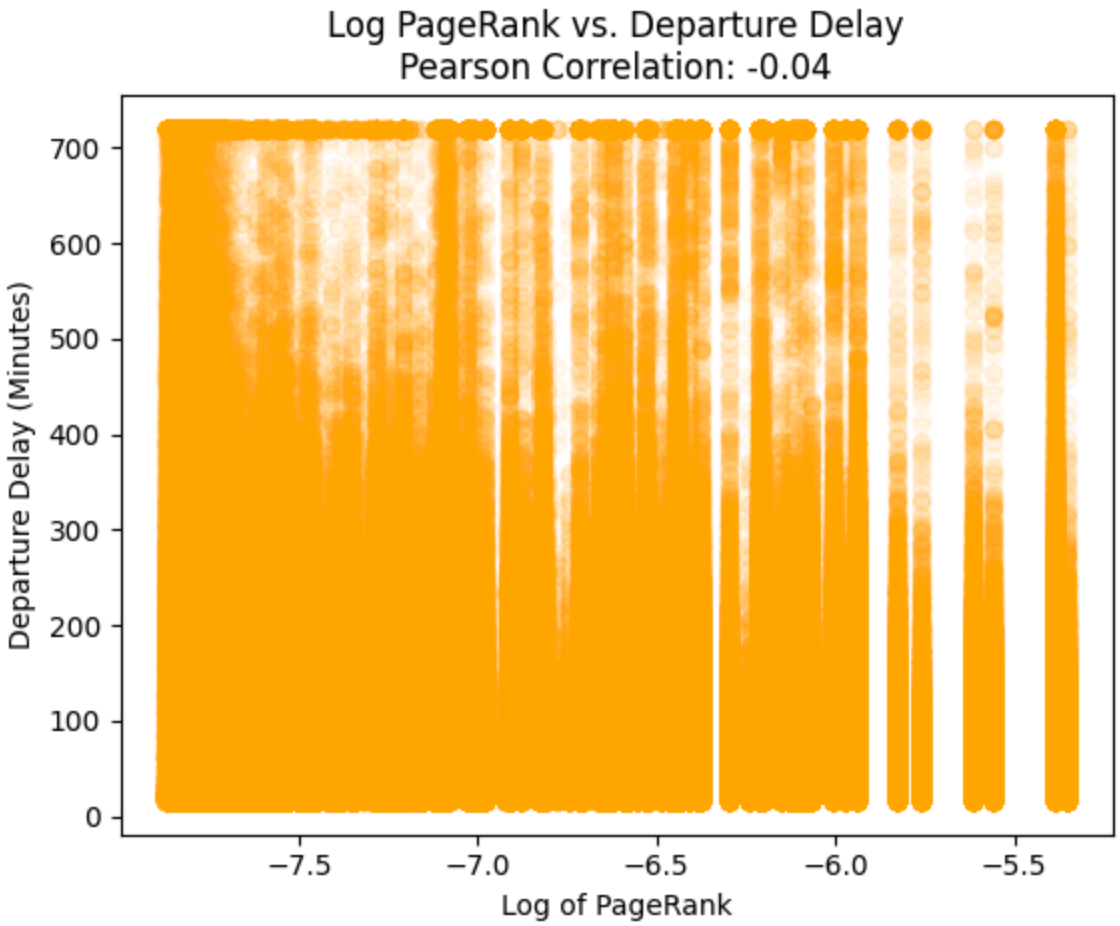
1. There appears to be an overall negative trend across all graph metrics, although there are inconsistencies particularly in the lower values of PageRank and the central values of betweenness centrality.
2. The scale of the y-axis in the summary statistics is significantly smaller than that of the original scatterplots. This further illustrates the skewness of the graph metrics that we produce.

The three features explored in this section are written as `log_degree`, `log_pagerank`, and `log_betweenness` during our modeling process.

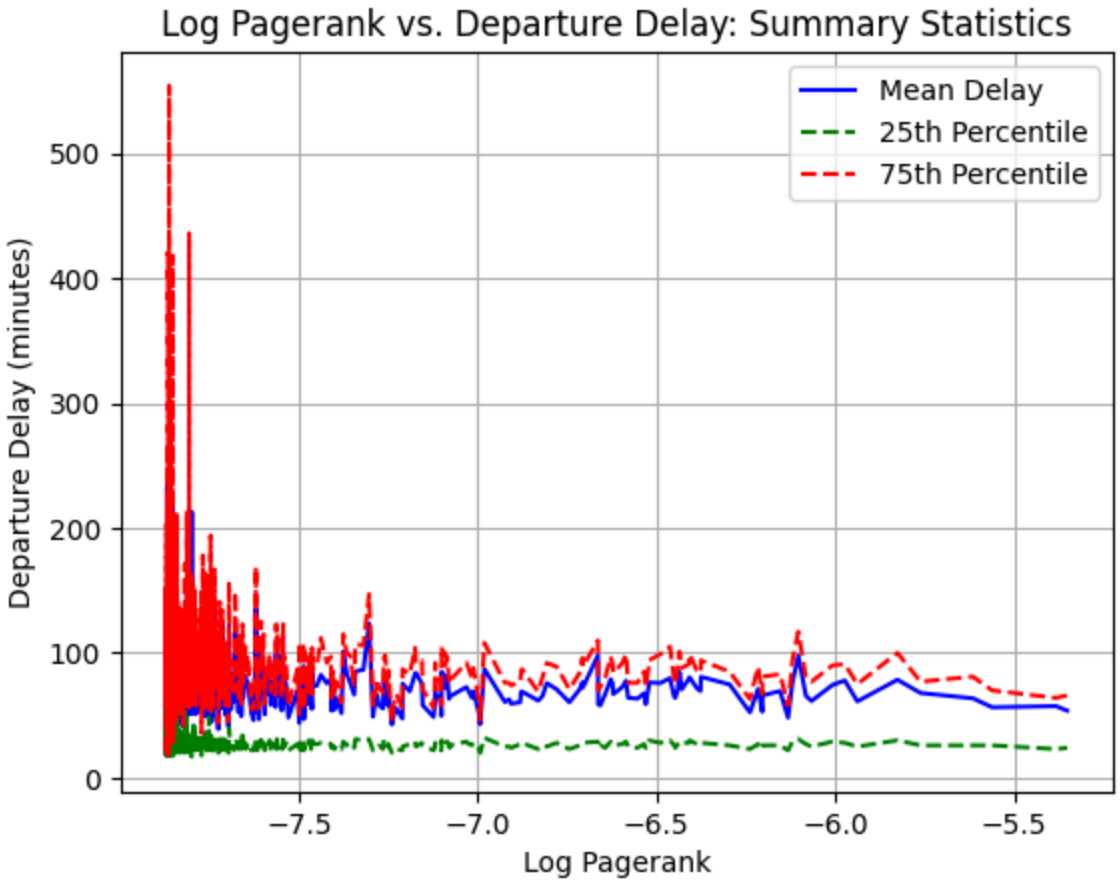
2. Airport-Operator Delay Based

This second configuration is aimed at exploring how delays from one flight might propagate to another. We restrict the edges in this configuration to only connect nodes that share the same operator. We believe this setup would best help to capture connecting flights, whereas connections across operators might introduce noise. The second graph is configured as follows:

- Nodes (vertices): Represent unique airport-operator combinations, identified by the IATA code followed by `OP_CARRIER_UNIQUE`.
- Edges: Represent individual flights between airport-operator combinations and are weighted by `DEP_DELAY_NEW`.



For this graph configuration, we work exclusively with PageRank as we are able to incorporate edge weights in producing the graph metrics. The motivation is that airport-operation combinations exhibiting high PageRank scores are most influential in propagating delays in the network. Following the same methods as the first approach, we visualize the natural logarithm of PageRank against individual delay times and note that there is a weak negative Pearson correlation coefficient of -0.04.



To acquire a better understanding on this phenomenon, we produce summary statistics and make the same observations as the first graph configuration. Namely

Data Leakage

This project involes time series data, so we must be wary of data leakage. In simple terms, this entails making sure that any flight delay prediction is only based off data that is available at the current time. Since we aim to make predictions 2 hours before

a flight's scheduled departure, the scope of data that can be used for feature engineering and inference is limitied to data with timestamps that are older than a flight's scheduled departure, as expressed in `CRS_DEP_TIME`, minus 2 hours.

One challenge in terms of data leakage was handling null values in the data. Since techniques such as mean imputation rely on taking averages across rows, there is the risk of including information in the aggregate that is not meant to be accessible at the time of prediction. We remedy this by incorporating rolling averages, where only entries of a given column that had a timestamp prior to the null entry were considered when it came to calculating average.

Graphs are also inherently prone to data leakage as they may be based off relations across rows in a dataset. To prevent leakage, we construct each graph using historical data from the training blocks of the 5-year dataset. Since these training blocks all occur chronologically ahead of their respective validation and test blocks, we can ensure that the timestamps used in creating features are from activity that has already occurred, and thus are available at the time of prediction. Features for the validation and test sets are extracted by joining on the pre-computed features based training set, using the node values as the primary key. For instance, with our first graph-based approach we compute PageRank for each airport on the training set. These values are saved and used to join on the airports in the validation and test

Feature Selection and Dimensionality Reduction

After performing feature engineering and Exploratory Data Analysis (EDA) detailed in the above sections, the initial step in dimensionality reduction was to select features based on domain knowledge, or their relevance and insights gained during the EDA process.

Selected Features

Categorical Features: (6 total)

- `OP_UNIQUE_CARRIER`
- `origin_type`
- `dest_type`

- `ORIGIN`
- `DEST`
- `DIVERTED_PREV`

Time-Based: (4 total)

- `DAY_OF_WEEK`
- `DAY_OF_MONTH`
- `MONTH`
- `crs_dep_time_minutes`

Numeric Features: (18 total)

- `HourlyPressureChange`
- `HourlyPressureTendency`
- `HourlyRelativeHumidity`
- `HourlySeaLevelPressure`
- `HourlyStationPressure`
- `HourlyVisibility`
- `HourlyWetBulbTemperature`
- `HourlyWindDirection`
- `HourlyWindSpeed`
- `DEP_DELAY_NEW_PREV`
- `ARR_DELAY_NEW_PREV`
- `TAXI_OUT_PREV`
- `TAXI_IN_PREV`
- `AIR_TIME_PREV`
- `DISTANCE_PREV`
- `closest_delay_before_2_hours_new`
- `closest_arrival_delay_before_2_hours_new`
- `DISTANCE`

Graph-Based: (4 total)

- `log_delay_prop_pagerank`,
- `log_degree`,
- `log_betweenness`,
- `log_pagerank`

Feature Transformation

From our 32 original features, categorical features were one-hot encoded. The features were aggregated into a feature vector and the one-hot-encoded vectors were flattened. This results in a feature vector with a dimensionality of 818.

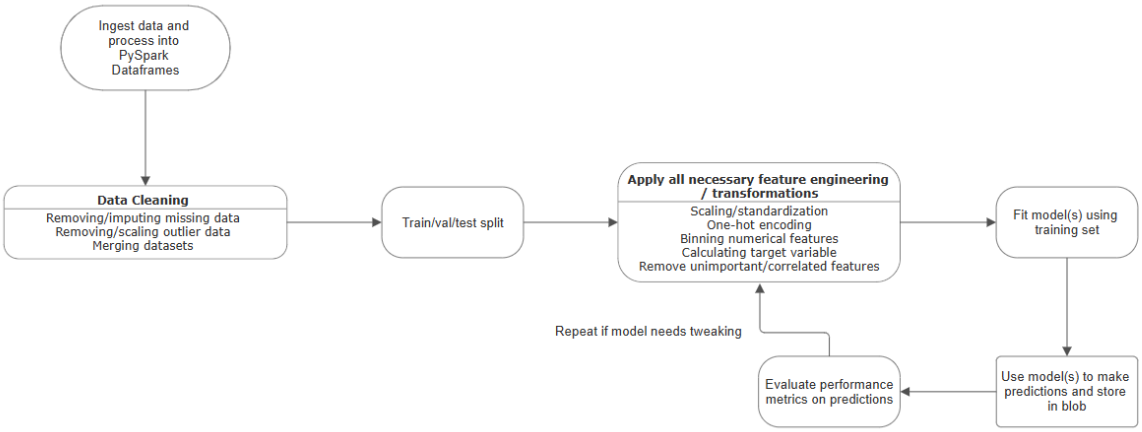
Dimensionality Reduction: Principal Component Analysis

With our naive approach of reducing the feature space by eliminating the features where all corresponding coefficients were 0 during our L1 Regression, we were unable to substantively reduce our feature vector's dimensionality. Given the limited reduction achieved with L1 regression, we experiment with **Principal Component Analysis (PCA)** to further explore dimensionality reduction while preserving variance

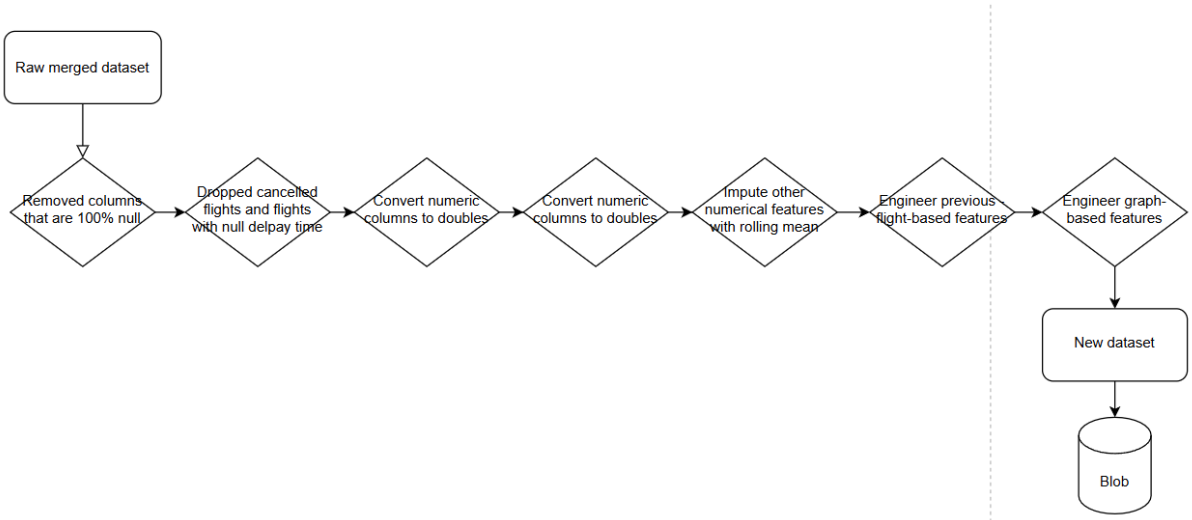
Data Processing and Model Pipelines

We take the combined data, apply our data cleaning and apply our feature engineering processes to the data. We then split into train/test sets of flights from 2015-2018 and flights from 2019, respectively, and compute the overall graph features to be used from the train set. We checkpoint along this process as necessary. We then split the training set into 5 folds for cross-validation and recompute the graph features for these folds. We then run the cross-validation with hyperparameter tuning using these folds, and then proceed to training on the full train set and testing on the test set. These processes are depicted below.

Overall ML Pipeline



Data Transformation and Feature Engineering Pipeline



Cross Validation and Hyperparameter Tuning

For our cross validation scheme, we chose a blocking time series split. We divided our training data of flights between 2015 and 2018 (inclusive) into 5 splits of 292 days where 200 days are training and 92 days are validation. A graphical depiction of how this splitting scheme conceptually works is provided below:

We leave flights during 2019 as our hold-out test-set. This gives us the ability to perform 5-fold cross validation without data leakage as the models will not see a data point as both the predictor and the response. Due to the limitations of our graph features, we recompute the graphs for the training sets of each of these 5 folds. We use this 5-fold cross validation scheme to conduct hyperparameter tuning for each class of models we explored. We used a Tree of Parzen Estimators (TPE) algorithm with 10 iterations through the hyperopt package for our parameter search. All experiments were run on an autoscaling cluster with between 6 and 10 workers, where the driver node has 56GB of memory and 16 CPU cores, and each worker has 14GB of memory and 4 CPU cores. Our main experiments are noted below:

Linear Regression

For linear regression, we explored the following search space:

Parameter Name	Minimum	Maximum	Increment
Number of Principal Components	1	7	1
Elastic Net Parameter	0	1	0.1
Regularization Parameter	0	0.2	0.05

Linear Regression best results:

Number of principal components	Elastic Net Parameter	Regularization Parameter	Best Average MAE
5	0.7	0.15	15.24

Random Forest

For random forest, we explored the following search space:

Parameter Name	Minimum	Maximum	Increment
Maximum Depth of Trees	5	8	1

Parameter Name	Minimum	Maximum	Increment
Number of Trees	10	100	10
Minimum Instances Per Node	1	5	1

Random Forest best results:

Maximum Depth of Trees	Number of Trees	Minimum Instances Per Node	Best Average MAE
8	70	2	13.15

Gradient Boosted Trees

For gradient boosted trees, we explored the following search space:

Parameter Name	Minimum	Maximum	Increment
Maximum Depth of Trees	5	8	1
Learning Rate	0.05	0.2	0.05
Minimum Instances Per Node	1	5	1
Maximum Number of Iterations	20	50	10

Gradient Boosted Trees best results:

Maximum Depth of Trees	Learning Rate	Minimum Instances Per Node	Maximum Number of Iterations	Best Average MAE
7	0.15	3	40	12.74

Multi-Layer Perceptron

For multi-layer perceptron classifiers, we explored the following search space:

Parameter Name	Minimum	Maximum	Increment
Number of Principal Components	5	7	1

Parameter Name	Minimum	Maximum	Increment
Learning Rate	0.01	0.1	0.01
Block Size	128	1024	128

Multi-Layer Perceptron 2 Hidden Layers best results:

Number of principal components	Learning Rate	Block Size	Best Average Precision
7	0.1	896	0.66

Multi-Layer Perceptron 3 Hidden Layers best results:

Number of principal components	Learning Rate	Block Size	Best Average Precision
7	0.04	384	0.65

All regression models underwent hyperparameter tuning for optimizing Mean Absolute Error (MAE) as defined below:

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

where $\hat{y}_i$ denotes the predicted delay for the ith flight

The classification models underwent hyperparameter tuning for optimizing precision as defined below:

- **True positives:** when a flight is predicted to be delayed and is actually delayed.
- **False positives:** when a flight is predicted to be delayed but is not actually delayed.
- Formula:

$$\frac{TruePositives}{TruePositives + FalsePositives}$$

Neural Network (MLP)

As requested, our team also trained two neural networks — a two hidden layer (layer sizes: 32, 16) multilayer perceptron (MLP), and a three hidden layer (layer sizes: 32, 16, 16) multilayer perceptron. Each layer has a sigmoid activation function, and the output layer has softmax activation function. We note that neural networks have been shown to be worse than gradient boosting and other tree ensemble methods for tabular and time series tasks like the current problem. For exploration, we tasked these networks with the slightly easier problem of predicting whether or not a delay will be greater than 15 minutes. In other words, we simplified the problem into an easier classification problem to see if the neural network shows any promise. In short, we will see that our MLP performance only achieves marginal gains on the easier task, which suggests our best regression models would outperform it if trained on the same classification task.

These are the figures depicting the training loss as our models train using logistic loss as the loss:

$$L = \frac{1}{n}\sum_{i=1}^{n} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$$

- 2 layer MLP
- 3 layer MLP

Results and Discussion

Experiments

To refresh, the metrics chosen for this model evaluation were:

- Mean Absolute Error (MAE)
  - Standard metric for regression models.
  - Main metric for model training due to its robustness against outliers.
  - Formula:

$$\frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

- Root Mean Squared Error (RMSE)
  - Another standard metric for regression models.
  - Easily influenced by outliers.
  - Formula:

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

- Proportion of Overpredictions
  - Overpredicting delay time can lead to customers arriving late for their flights, which we wish to avoid.
  - Hard to tune models for this, but it was a factor we kept in mind for analysis.
  - Formula:

$$\frac{1}{n}\sum_{i=1}^{n}(1 \text{ if } y_i < \hat{y}_i \text{ , else } 0)$$

- Precision
  - Main metric for classification model training.
  - **True positives:** when a flight is predicted to be delayed and is actually delayed.
  - **False positives:** when a flight is predicted to be delayed but is not actually delayed.
  - A higher precision is caused by fewer false positives. We wish to limit false positives to prevent customers from arriving late to the airport.
  - For regression models, any prediction > 15 was classified as positive, and those <= 15 were negatives.
  - Formula:

$$\frac{TruePositives}{TruePositives + FalsePositives}$$

The results for each model can be seen in the table below:

Model	Train Time	Eval Time (Train)	Eval Time (Test)	MAE (Train)	MAE (Test)	RMSE (Train)	RMSE (Test)
Linear Regression *(Baseline)*	7m13s	14m37s	8m40s	14.984	16.585	35.127	40.836
Random Forest	36m2s	20m9s	9m44s	13.148	14.530	33.424	39.157
Gradient Boosted Trees (GBT)	81m16s	17m48s	9m37s	12.509	13.873	32.721	38.609
Multi-Layer Perceptron 2-Layer *(Class. Only)*	15m35s	4m20s	8m54s	N/A	N/A	N/A	N/A
Multi-Layer Perceptron 3-Layer *(Class. Only)*	20m54s	4m9s	9m46s	N/A	N/A	N/A	N/A

Discussion

The more complex regression models (random forest and GBT) significantly outperformed the linear regression baseline. The random forest models achieved a 12.4% lower test MAE and 4.1% test RMSE compared to the baseline. Meanwhile, GBT demonstrated a 16.4% reduction in test MAE and 5.5% reduction in test RMSE, marking GBT as our best performing regression model. We hypothesize that these models were able to demonstrate lower losses due to their complex ensemble nature and their ability to make strong predictions for any data distribution, which both differentiate the models from least squares linear regression.

Regression metrics across all models in Phase 3 underperform the Phase 2 baseline (9.698 test MAE, 25.629 test RMSE). However, we hypothesize that this is due to the fact that the Phase 2 baseline was trained and evaluated on 1 year of data, as opposed to 5 years in this phase. Yearly changes to trends across airlines, airports, and other flight factors likely impacted the ability for delay time to be modeled. As evidence of this hypothesis, when the baseline model was trained and evaluated on the Phase 3 train/test sets with the Phase 2 hyperparameters and feature set, it yielded a 16.99 test MAE and 44.80 test RMSE.

Each regression model yielded a significantly higher RMSE than MAE. We hypothesize that this is largely due to the presence of abnormally high outliers within the target variable (caused by a few instances of flights being delayed by 12+ hours). RMSE will amplify the loss caused by these outlier errors due to its quadratic penalty, which is why MAE was chosen as the primary metric for this analysis.

The proportion of overpredictions was notably high for all models, exceeding 0.75 across both training and testing sets. We hypothesize that this is due to the fact that roughly 65% of flights in the dataset had a delay time of 0, and it is likely that even the best performing models will predict values greater than 0 for a vast majority of these data points. Moving forward, some postprocessing steps can be taken to have any low predictions (such as >15 minutes) bounded to 0 in order to provide a closer estimate to how delays are estimated. However, for the current situation, precision is a stronger indicator of false positive predictions. For the regression models, positive predictions were classfied as any prediction greater than 15 minutes. Based on this criteria, the baseline linear regression model achieved a train and test precision of 0.478 and 0.474, respectively. This is no better than a coin flip, essentially suggesting that around half of positive predictions made by this model were true positives and half were false positives. In contrast, the other models outperformed this baseline precision metric. Random forest, GBT, the 2-layer MLP classifier, and the 3-layer MLP classifier scored test precisions of 0.624, 0.597, 0.624, and 0.621, respectively. While these metrics are far from perfect, they demonstrate promise in the ability of our models to accurately estimate whether or not a flight is delayed without compromising customer airport punctuality. Of these models, it appears that the random forest and both MLP classifiers all roughly tied for the top of the leaderboard.

Across all metrics and all models, there is a lack of overfitting, as all metrics are

Conclusion

This project focused on predicting flight delays with machine learning models to minimize traveler dissatisfaction caused by long and unexpected waiting times. Accurate predictions of flight delay durations enable passengers to plan their schedules better, reducing stress and inefficiencies in air travel.

The hypothesis driving our project was that machine learning pipelines integrating transportation and weather data with custom-engineered features could accurately predict flight delays and delay durations.

Throughout our project, several key insights emerged:

- **Features and Engineering:** A comprehensive set of 818 features was developed, including categorical, time-based, numeric, and graph-based features. Key contributors included prior flight delay information and graph features like `log_delay_prop_pagerank`.
- **Models:** Gradient Boosted Trees (GBT) emerged as the best-performing model, achieving the lowest RMSE (38.61), MAE (13.87), and essentially tied for our highest precision (.597) in classifying delays over 15 minutes. Other models, such as Random Forest and Linear Regression, provided benchmarks but performed less effectively.
- **Hyperparameters:** Optimized parameters like a `maxDepth` of 7, `minInstancesPerNode` of 3, and a `learningRate` of 0.15 significantly contributed to the GBT model's performance.

Our results validate the hypothesis that machine learning pipelines can effectively predict flight delay durations. Our Gradient Boosted Trees model demonstrated a major improvement in MAE and RMSE compared to both Random Forests and Linear Regression (16.4% reduction in test MAE and 5.5% reduction in test RMSE compared to baseline, where Random Forests achieved a 12.4% lower test MAE and 4.1% test RMSE compared to the baseline). The incorporation of both transportation and weather data enhanced the models' predictive accuracy, providing a practical tool for airlines and passengers. However, the skewed data distribution and overprediction tendencies highlight areas needing further refinement.

In terms of classification, we were able to acheive a precision of .624 with both Random Forests and our MLP. We believe giving customers an estimated delay within 14 minutes on average (MAE) is more useful than a binary "Flight Delayed" indicator with a precision of .624. Furthermore, we note that our Random Forest was not optimized for this task and tends to outperform MLPs when directly trained on this nature of problem (https://arxiv.org/pdf/2207.08815 (https://arxiv.org/pdf/2207.08815)). Thus, we prefer our regression models.

Future Steps

- **Refine Hyperparameters:** Focus on minimizing MAE during further hyperparameter tuning to enhance precision and interpretability.
- **Feature Engineering:** Explore additional features, especially those that capture real-time conditions, to improve predictive power.
- **Neural Network Exploration:** Develop more complex neural network models for delay classification and prediction.
- **Recent Data Collection:** Incorporate updated data for training to ensure the models remain relevant with evolving trends.
- **Broader Model Testing:** Experiment with alternative machine learning models to benchmark performance further.

By pursuing these enhancements, our project can contribute to the creation of robust delay prediction systems, ensuring smoother travel experiences for passengers and

Code References

Data Cleaning/EDA: https://adb-4248444930383559.19.azuredatabricks.net/editor/notebooks/2276882173340804?o=4248444930383559 (https://adb-4248444930383559.19.azuredatabricks.net/editor/notebooks/2276882173340804?o=4248444930383559)

Feature Engineering/EDA: https://adb-4248444930383559.19.azuredatabricks.net/editor/notebooks/2276882173341278?o=4248444930383559 (https://adb-4248444930383559.19.azuredatabricks.net/editor/notebooks/2276882173341278?o=4248444930383559)

Event-Based EDA: https://adb-4248444930383559.19.azuredatabricks.net/editor/notebooks/1685902928458577?o=4248444930383559 (https://adb-4248444930383559.19.azuredatabricks.net/editor/notebooks/1685902928458577?o=4248444930383559)

Graph Features: https://adb-4248444930383559.19.azuredatabricks.net/editor/notebooks/1685902928456921?o=4248444930383559 (https://adb-4248444930383559.19.azuredatabricks.net/editor/notebooks/1685902928456921?o=4248444930383559)

Feature Engineering Utils (Condensed Cleaning/Feature Engineering Functions): https://adb-4248444930383559.19.azuredatabricks.net/editor/notebooks/2276882173344753?o=4248444930383559 (https://adb-4248444930383559.19.azuredatabricks.net/editor/notebooks/2276882173344753?o=4248444930383559)

Data Processing: https://adb-4248444930383559.19.azuredatabricks.net/editor/notebooks/2276882173344698?o=4248444930383559#command/2276882173344767 (https://adb-4248444930383559.19.azuredatabricks.net/editor/notebooks/2276882173344698?o=4248444930383559#command/2276882173344767)

Hyperparameter Tuning: https://adb-4248444930383559.19.azuredatabricks.net/editor/notebooks/57921511661541?o=4248444930383559 (https://adb-4248444930383559.19.azuredatabricks.net/editor/notebooks/57921511661541?o=4248444930383559)

Experiments: https://adb-4248444930383559.19.azuredatabricks.net/editor/notebooks/57921511674581?o=4248444930383559 (https://adb-4248444930383559.19.azuredatabricks.net/editor/notebooks/57921511674581?o=4248444930383559)