04.01.25

Phase 2: Flight Delay Prediction

SECTION 3, TEAM 1

Members: Sebastian Rosales, Kenneth Hahn, Benjamin He, Edgar Munoz, Adam Perez, Kent Bourgoing

Group Members

Sebastian Rosales
sbsrosales11@berkeley.edu

Kenneth Hahn
hahnkenneth@berkeley.edu

Benjamin He
ben_he@berkeley.edu

Edgar Munoz
edgarmunoz@berkeley.edu

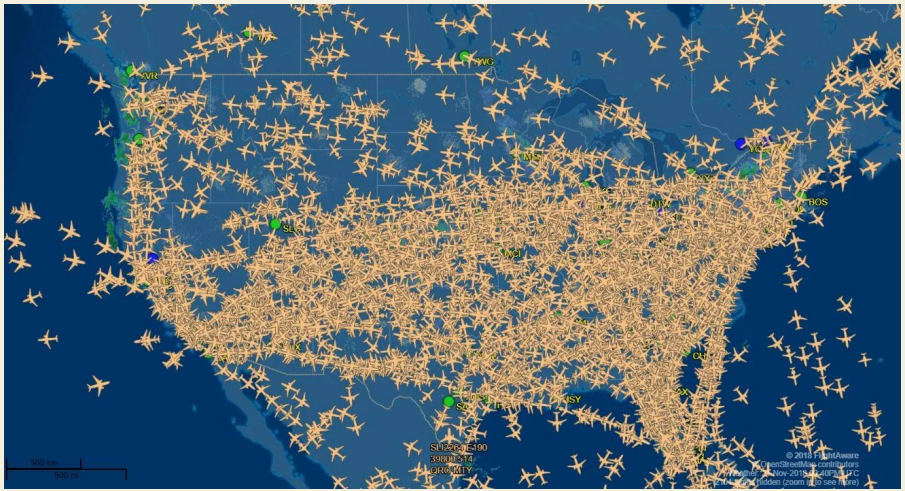
Adam Perez
adperez@berkeley.edu

Kent Bourgoing
kent1bp@berkeley.edu

1. **Project Description**
2. Exploratory Data Analysis
3. Preprocessing/Pipeline
4. Cross-Validation and Fine-Tuning
5. Model Results
6. Conclusion

Background

- Air travel is a complex web of logistics that needs to be handled precisely in order to coordinate departures and arrivals for 45,000 flights and 2.9M passengers per day.
- It is beneficial for both airlines and for passengers to be able to predict whether a given flight will be delayed or not for a variety of reasons:
    - *Operational Efficiency*: optimizing crew scheduling, ground handling, and gate assignments.
    - *Customer Satisfaction*: Allows for airlines to notify customers ahead of time to reduce stress and allow them to plan ahead.



Goal

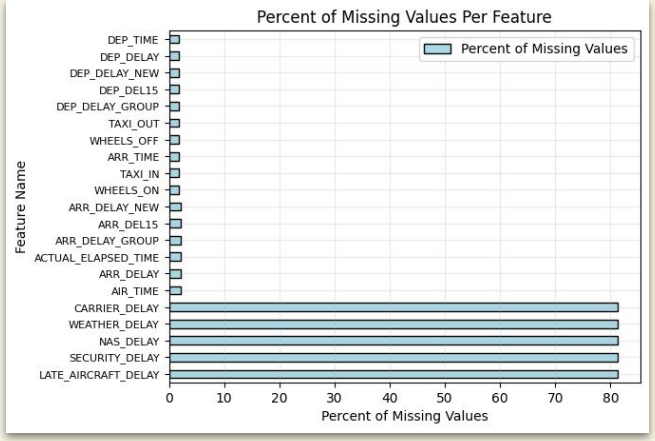
- Utilizing flight data provided by the U.S. Department of Transportation and meteorological data provided by the National Oceanic and Atmospheric Administration repository (NOAA), our goal for this project is as follows:

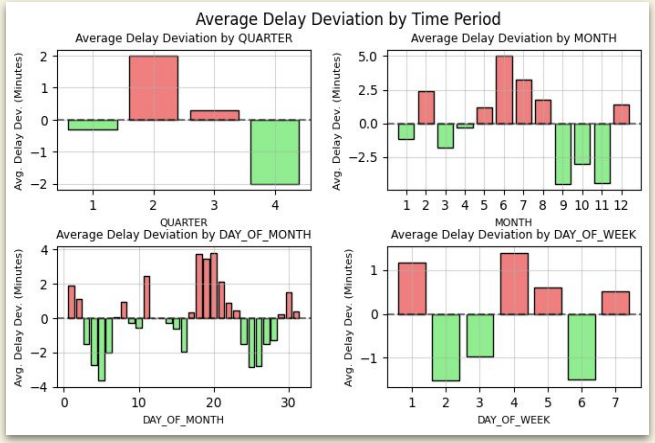
Classify and predict flight departure times ahead of the scheduled flight to determine whether a given flight will depart early, on-time, or late.

1. Project Description
2. **Exploratory Data Analysis**
3. Preprocessing/Pipeline
4. Cross-Validation and Fine-Tuning
5. Model Results
6. Conclusion

Flights Dataset

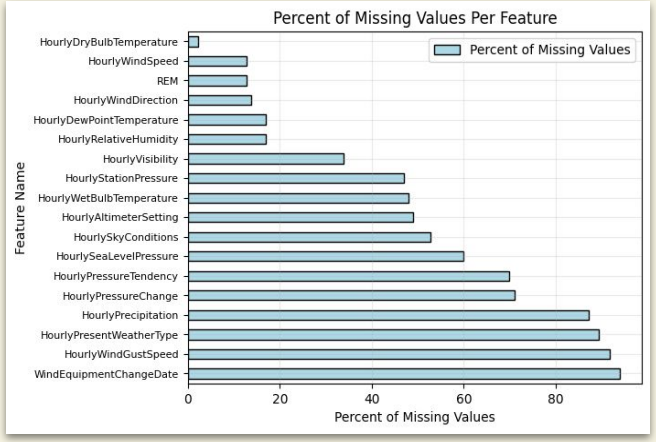
- Has 109 columns with 14,844,074 rows
- 48 columns had a missing value percentage above 95%
- 40 columns had a missing value percentage below 1%
- Quarter 2 seems to be the worst time to travel
- However, most of the flight delays are from the month of June
- Day of the month and week can help us pick the best days to travel

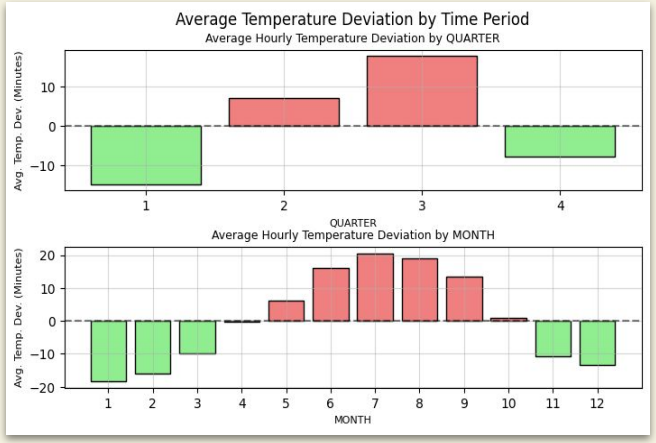




Weather Dataset

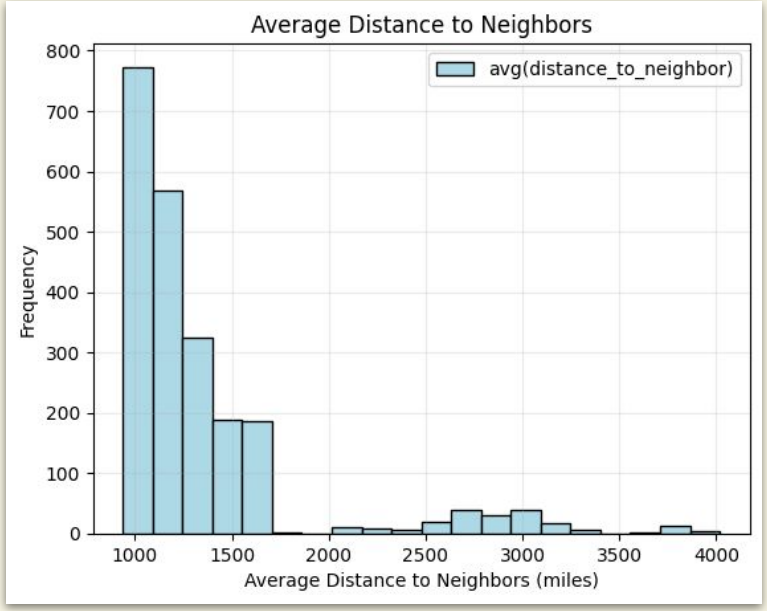
- Has 124 columns with 131,937,550 rows
- 97 columns had missing value percentages above 95%
- 9 columns had missing value percentages below 1%
- Conducted a quick sanity check to ensure our data makes sense





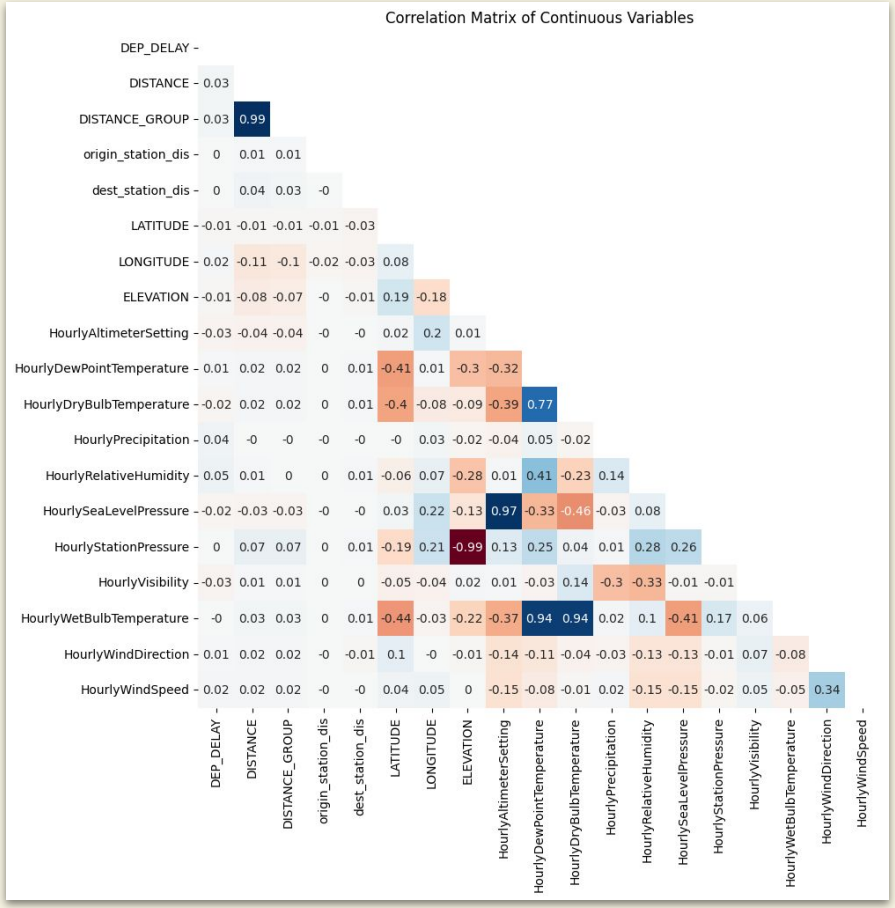
Airport Dataset

- Has 12 columns with 5,004,169
- There were no missing values on all 12 columns
- Most airports are on average 1000 to 1500 miles away from another airport
- There are some airports that are on average nearly 4000 miles away from another airport
- Could be help us predict delays since there might be greater delays when traveling further distances



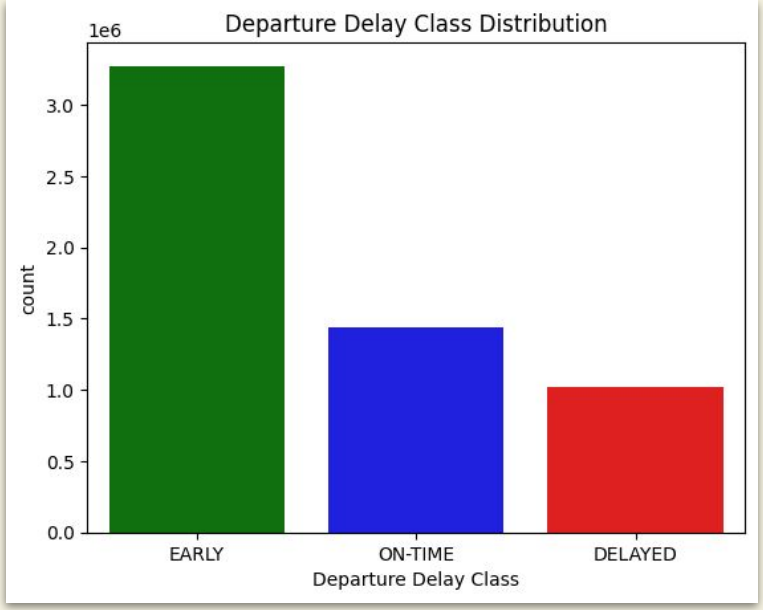
Numerical Analysis

- Strong negative correlations:
  - ELEVATION and HourlyStationPressure
- Strong positive correlations:
  - DISTANCE and DISTANCE_GROUP
  - HourlyAltimeterSetting and HourlySeaLevelPressure
  - HourlyDewPointTemperature, HourlyDryBulbTemperature, and HourlyWetBulbTemperature
- From 3-Month to 12-Month LONGITUDE lost all of its correlation to temperature features



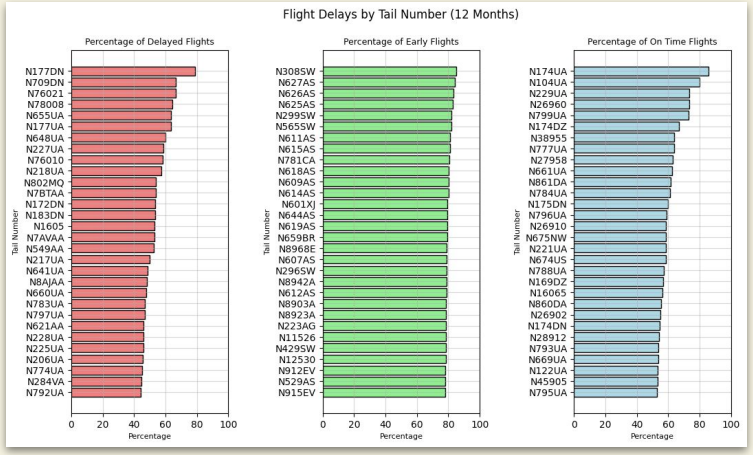
Target Variable Analysis

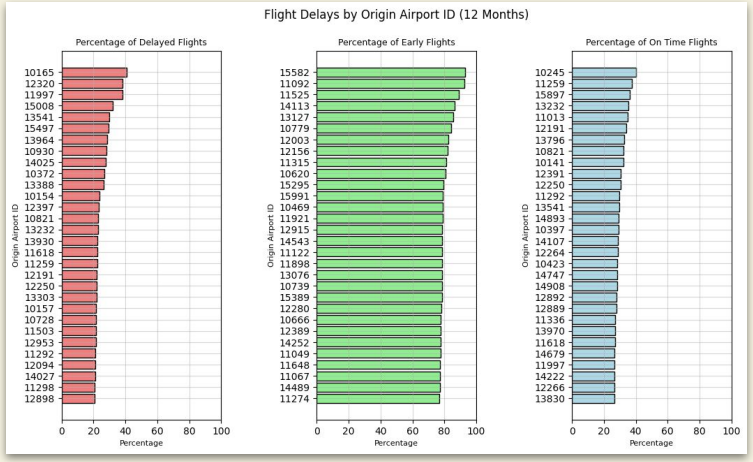
- We are interested in classifying 3 classes
  - Early (Delay < 0 minutes)
  - On-Time (0 <= Delay <= 15 minutes)
  - Delayed (15 minutes < Delay)
- Chose 15 minutes as Delayed based on the definition from Department of Transportation
- Early is by far our majority class representing 57.2% of flights
- On-Time is the second most common occurrence representing 25.1% of flights
- Delayed is the rarest occurrence representing 17.8% of flights



Categorical Analysis

- Tail numbers might be indicative of how reliable an aircraft is
  - Some are highly reliable with early flight percentages around 80%
  - Some are highly unreliable with delayed flight percentages around 50%
- Efficient airport designs and less travelers might make smaller delays more consistent
  - The 30 airports shown had an early flight percentage above 75%

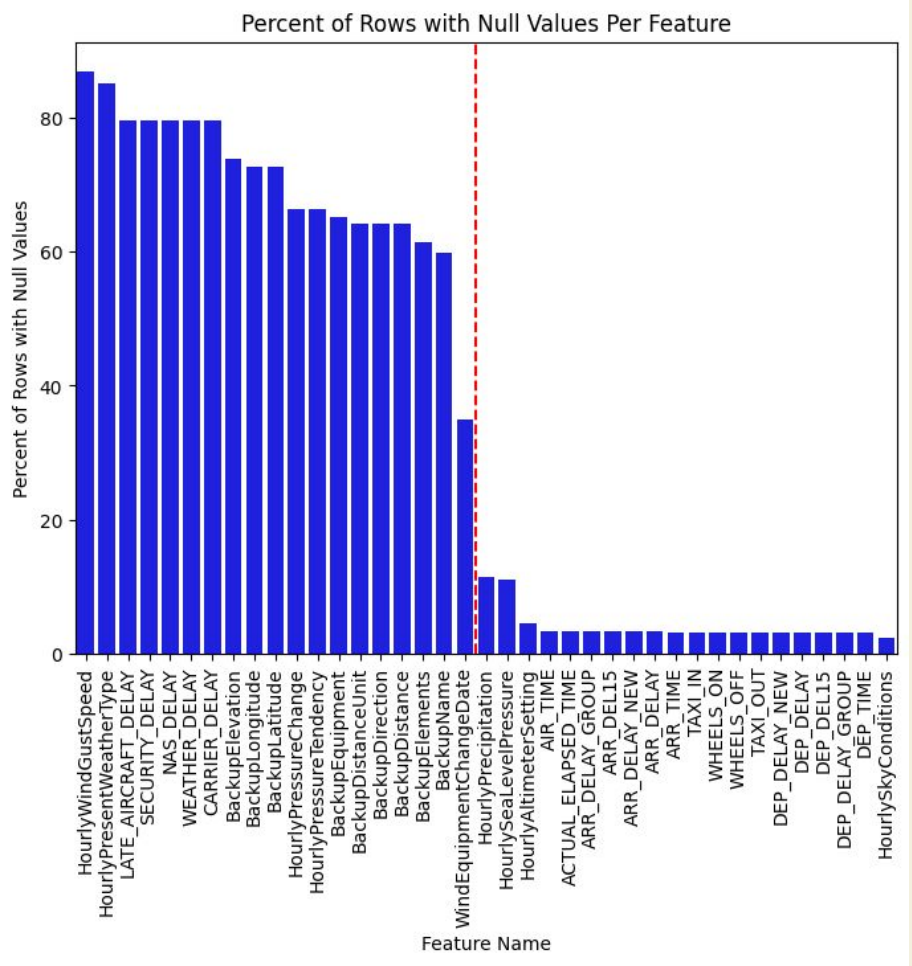




1. Project Description
2. Exploratory Data Analysis
3. **Preprocessing/Pipeline**
4. Cross-Validation and Fine-Tuning
5. Model Results
6. Conclusion

Missing Values

- Initial Inspection of Missing Values showed that 108 features had missing values above 30%.
- We removed features that had information about future events to prevent leakage (Flight Arrival Status, Taxi time, Any information about the delay, etc.)
- To avoid assumptions about the underlying distribution of features, we decided to impute a “missing” category or a “-1” for numeric data.



Time Series and Numeric Features

- Time Series Features that we already have: **Quarter, Month, Day of Week, Day of Month of Departing Flight**
- Time Series Features needing further processing: **Scheduled Departure Datetime, Four Hours Before Departure Datetime, Two Hours Before Departure Datetime, Weather Station Datetime** (4 hours before departure).
  - Extract only the hour and minute from these datetimes.
- Cyclically encode temporal features.
  - Preserves periodicity of the features so the model can learn continuous temporal patterns (weekly, monthly, quarterly, etc)
  - Added benefit of keeping the range of values from [-1, 1]

- For Numeric Features, utilize a RobustScalar to scale all columns based on the median and interquartile range.

$$\bar{x} = \frac{X - median(X)}{75\text{th quantile(x)} - 25\text{th quantile(x)}}$$

$$t_{sin} = sin(\frac{2\pi t}{max(t)})$$

$$t_{cos} = cos(\frac{2\pi t}{max(t)})$$

Categorical Features

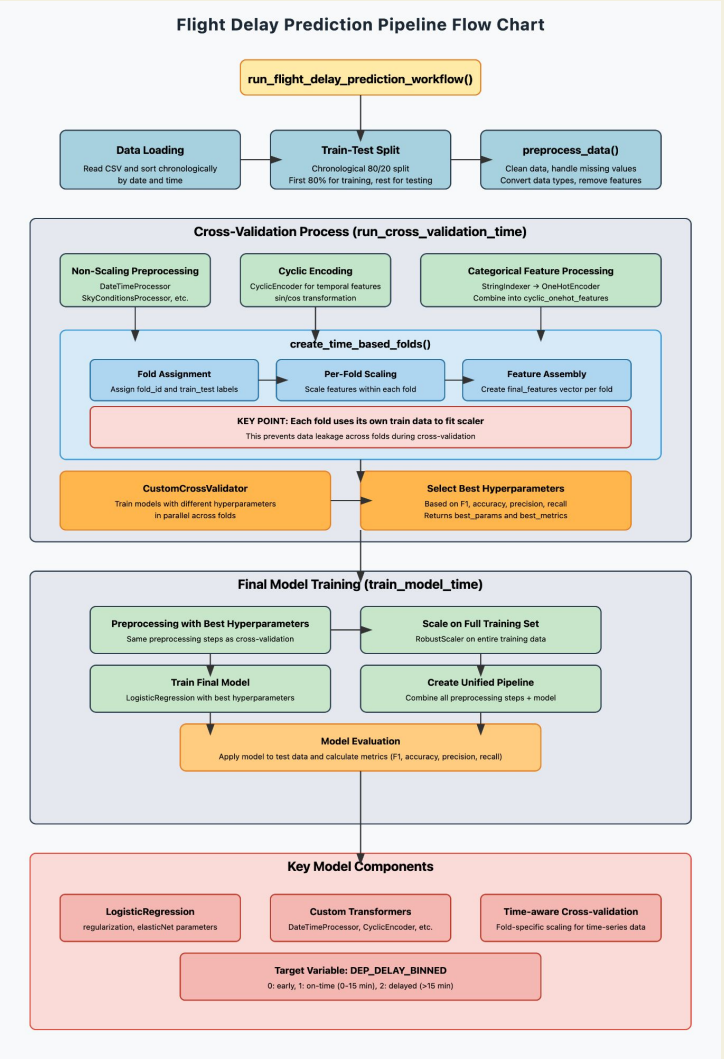
- HourlySkyConditions transformed to 3 features: HourlyCloudCoverage (str), HourlyCloudLayerAmount (str), and HourlyCloudBaseHeight (float)
    - HourlyCloudCoverage is the description of the clouds (CLR = clear day, OVC = overcast, etc.)
    - HourlyCloudLayerAmount: number of eighths that the sky is covered by (treated as a categorical variable)
    - HourlyCloudBaseHeight: The height of the lowest layer of clouds in ft.
- We then one hot encoded all the categorical features, leading to a vector of 13591 different features.
    - Will utilize LASSO in order to select the most relevant features.

LASSO Feature Selection

- Used Logistic Regression with Lasso (L1) Regularization to reduce many of the feature weights to 0 and utilize only the ones remaining.
- Remaining Numeric Columns:
  - Travel Distance, Longitude, Hourly Relative Humidity, Hourly Visibility, and Hourly Wind Speed
- Remaining Time Columns:
  - Day of month, day of week, month, scheduled departure time.
- Remaining Categorical One-Hot Columns:
  - Carriers: DL, OO, EV, UA, AS, NK, HA
  - **Origin** Airports: 10397, 11298, 10821, 11259
  - **Origin** States: IL, GA, HI, MD, NY
  - **Origin** Stations: 72219013874, 72259003927, 72406093721, 72258013960
  - **Origin** Airport Type: Large Airport
  - Hourly Cloud Coverage: **OVC** (overcast)
  - Hourly Cloud Layer Amount: **8** (8/8 of the visible sky is covered)

ML Pipeline: Overview

1. Data loading & Cleaning
2. Feature engineering
3. Time-based Cross-validation
4. Feature Scaling and Transformation
5. Model Training with Hyperparameter Tuning (Grid Search)
6. Final model Integration
7. Evaluation



ML Pipeline: Feature Families

Temporal Features (12 features)

- Year, Month, Quarter, Day of Month, Day of Week
- Scheduled departure hour/minute
- Four hours prior departure hour/minute
- Two hours prior departure hour/minute
- Station hour/minute

Weather Features (10 features)

- HourlyAltimeterSetting, HourlyDewPointTemperature
- HourlyPrecipitation, HourlyRelativeHumidity
- HourlyVisibility, HourlyWindDirection
- HourlyWindSpeed, HourlyCloudCoverage
- HourlyCloudLayerAmount, HourlyCloudBaseHeight

Geographical Features (8 features)

- Origin/Destination Airport IDs
- Origin/Destination State
- Origin/Destination station distances
- Latitude, Longitude, Elevation

Flight-specific Features (5 features)

- Carrier, Carrier ID, Flight Number
- Distance, Distance Group

Key Implementation Notes

Custom Transformers

Our pipeline uses several custom transformers for effective feature engineering:

- **DateTimeProcessor**: Extracts meaningful temporal features
- **SkyConditionsProcessor**: Processes complex weather data
- **CyclicEncoder**: Applies sine/cosine transformations to cyclical features

Time-Based Cross-Validation

We also use chronological splits of data instead of random shuffling

- Maintains temporal integrity by fitting scalers on training portions only
- Prevents data leakage across time boundaries

Feature Transformations

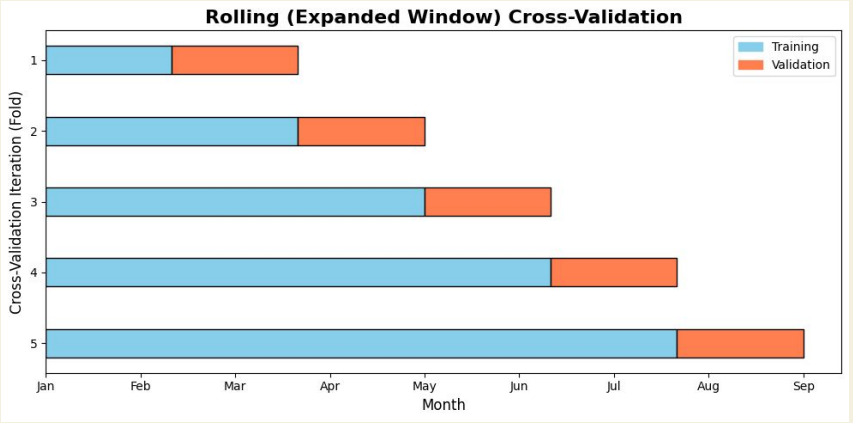
We utilize various feature transformations depending on the type of feature

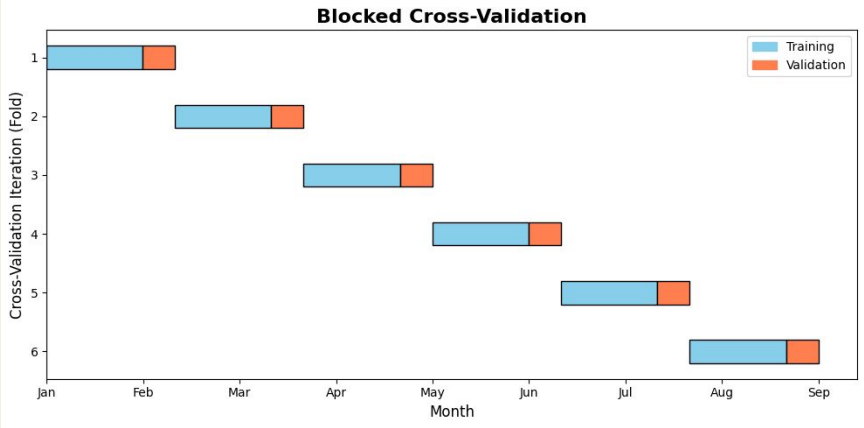
- **Categorical features:** String indexing and one-hot encoding
- **Temporal features:** Cyclic encoding using sine/cosine transformations
- **Numerical features:** Robust scaling for outlier handling

1. Project Description
2. Exploratory Data Analysis
3. Preprocessing/Pipeline
4. **Cross-Validation and Fine-Tuning**
5. Model Results
6. Conclusion

Blocked Cross-Validation (Time-Series)

- Why Cross-Validation?
  - Tunes hyperparameters systematically and evaluates model performance robustly.
  - Reduces overfitting risk by avoiding a single train–test split.
- Time-Series Cross-Validation Approaches:
  - Rolling (Expanding Window) CV:
    - Training set expands sequentially; validation always follows.
    - Overlapping sets may lead to data leakage.
  - Blocked (Non-Overlapping) CV:
    - Uses non-overlapping folds with margins to separate training and validation.
    - Strictly respects chronological order, preventing leakage and ensuring reliable evaluation.





Blocked Cross-Validation (Time-Series)

Our Project's Approach:

- Step 1: Data Partitioning
  - Use the first three quarters of our 12-month dataset.
  - Partition the data into non-overlapping folds (Blocked CV) that preserve the chronological order.
- Step 2: Custom Cross-Validation Setup
  - Implement a custom cross-validation class that lets you specify:
    - The number of folds
    - The training ratio for each fold
    - The model and hyperparameter grid
    - The evaluation metrics
  - Apply robust scaling on numerical features from the training subset only to prevent data leakage.
- Step 3: Training & Evaluation
  - For each fold:
    - Split data into training and test sets based on the defined ratio.
    - For each hyperparameter combination, train the model on the training set and evaluate on the test set.
    - Compute metrics (e.g., F1-score, accuracy) for each fold.
  - Average the metrics across folds and select the hyperparameter combination with the best average score.

Metrics Used

Accuracy: Measures the overall correctness of our model's predictions. It is defined as the ratio of the number of correct predictions to the total number of predictions made.

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

Weighted Precision: Computes precision for each class (early, on-time, delayed) and then averages them using weights based on the number of instances in each class. This helps account for class imbalances.

$$\text{Weighted Precision} = \sum_{c=1}^{C} \frac{N_c}{N} \cdot \text{Precision}_c$$

Weighted Recall: Calculates the recall (sensitivity) for each class, which is the fraction of actual instances correctly predicted, and averages these scores with weights proportional to the class frequencies.

$$\text{Weighted Recall} = \sum_{c=1}^{C} \frac{N_c}{N} \cdot \text{Recall}_c$$

F1 Score: Represents the harmonic mean of precision and recall for each class. The weighted F1 score averages these values according to each class's prevalence, providing a balanced measure of performance that considers both false positives and false negatives.

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

1. Project Description
2. Exploratory Data Analysis
3. Preprocessing/Pipeline
4. Cross-Validation and Fine-Tuning
5. **Model Results**
6. Conclusion

Model Comparisons Across Evaluation Metrics

- Decision Tree Classifier used as a baseline due to simplicity and interpretability
- Logistic Regression hyperparameters selected via grid search across: Lasso/Ridge/ElasticNet
- Random Forest tuned with a small grid to manage runtime while balancing performance

Model	Train F1	Train Precision	Train Recall	Train Accuracy	Test F1	Test Precision	Test Recall	Test Accuracy
Decision Tree	0.470	0.445	0.570	0.570	0.505	0.460	0.586	0.586
Logistic Regression	**0.520**	**0.525**	**0.582**	**0.582**	**0.537**	**0.523**	0.561	0.561
Random Forest	0.4240	0.4492	0.5762	0.5762	0.4461	0.3562	**0.5968**	**0.5968**
Random Forest (with Lasso Feature Selection)	0.4490	0.4935	0.5715	0.5715	0.4597	0.4297	0.5958	0.5958

Best Hyperparameters & Takeaways

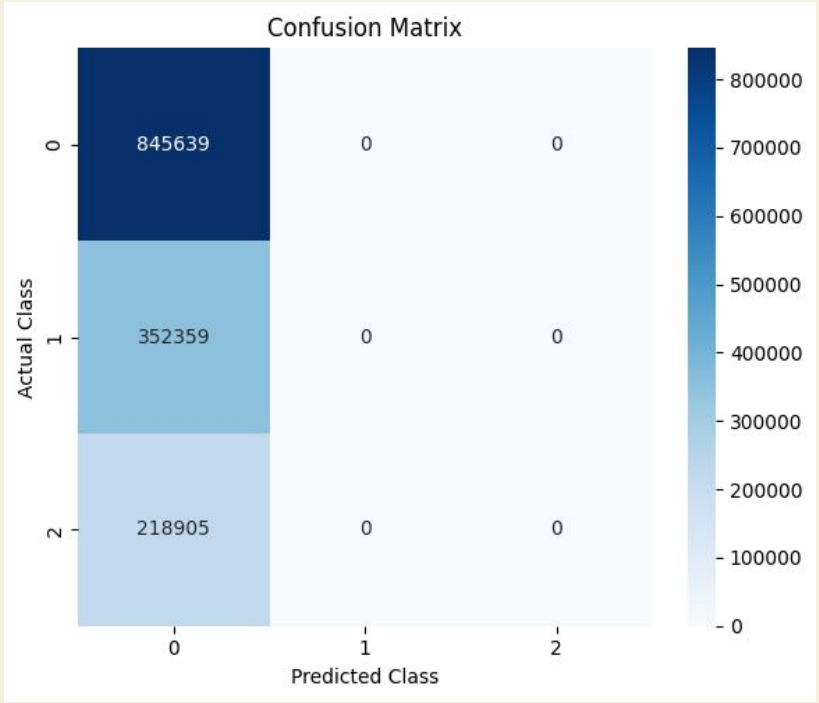
Model	Hyperparameters
Decision Tree (Baseline)	maxDepth = 3
Logistic Regression	regParam=0.001, elasticNetParam=0.0
Random Forest	numTrees=20, maxDepth=15
Random Forest (w/LASSO)	numTrees=5, maxDepth=5

Key Takeaways

- Decision Tree, while a baseline model, performed competitively and provided interpretability
- Logistic Regression achieved the highest overall performance, especially in F1 and precision on the blind test
- Random Forest has strong recall and accuracy but low precision, which indicates overprediction on the majority class
- Random Forest with Lasso-selected features slightly improved precision and F1 on the test set, suggesting feature reduction helped generalization.

Confusion Matrix & Class Imbalance

Confusion Matrix - Random Forest on Blind Test Set



- 0=Early, 1=On-Time, 2=Delayed
- All predictions fall into the majority class, Class 0 (Early), with no predictions made for Class 1 (On-Time) or Class 2 (Delayed).
- Poor generalization to delayed flights
- Precision likely affected due to class imbalance
- Also important is improving model recall across all classes
- In future iterations, SMOTE (Synthetic Minority Over-sampling Technique) will be considered as a resampling strategy to help balance classes & improve generalization for delayed flights.

1. Project Description
2. Exploratory Data Analysis
3. Preprocessing/Pipeline
4. Cross-Validation and Fine-Tuning
5. Model Results
6. **Conclusion**

Conclusion

- Our best model was Logistic Regression with light L2 regularization (regParam = 0.001, elasticNetParam=0.0). The overall F1 score performance was 0.537, with many opportunities for improvement.
- With the pipeline created and implemented along with cross-validation for time series, our team expects to improve the overall performance of our model with the following next steps:

Next Steps for Phase 3

- Improve Feature Engineering
  - Determine better way to transform geographic data (Latitude, Longitude, Elevation)
  - Review how different imputation techniques affect performance (median/mean imputation, KNN imputation, etc).
  - Implement Principal Component Analysis to reduce our extremely large vector space.
- Test out SMOTE to better deal with imbalanced classes.
- Implement Neural Networks to determine if there is any performance improvement.