04.01.25

Phase 3: 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. **Abstract**
2. Project Description
3. Exploratory Data Analysis
4. Feature Engineering
5. Model Pipeline
6. Model Experiments
7. Model Results
8. Conclusion

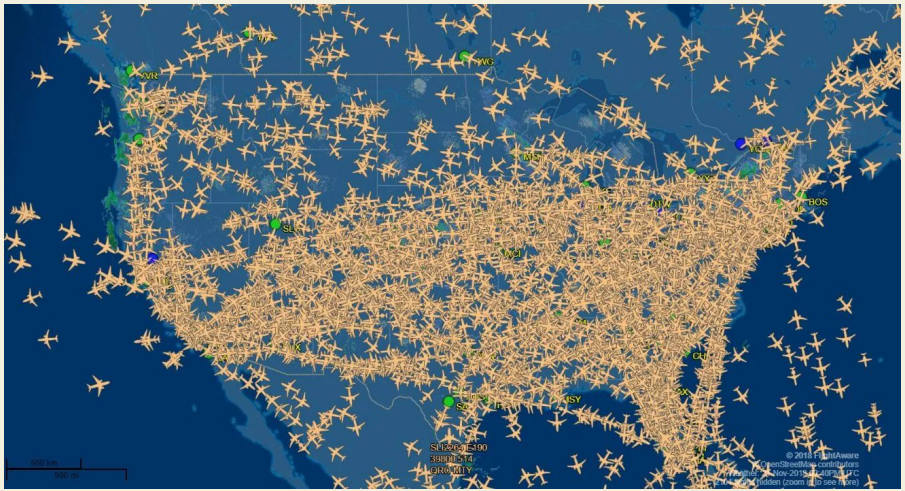
Abstract

- **Problem:**
  - Flight delays are a widespread issue; our team developed a predictive model to classify departures as Early, On-Time, or Delayed using flight and weather data.
- **Data Sources**
  - Combined U.S. Department of Transportation flight data with NOAA weather data; over 28 million records analyzed.
- **Modeling Approach**
  - Extensive feature engineering (including PageRank for airport centrality), time-series encoding, and resampling methods to address class imbalance.
  - Models: Logistic Regression, Decision Tree, Random Forest, and MLP
- **Best Model**
  - A Random Forest classifier with oversampling achieved the best performance (Test F1 = 54.4%) with high accuracy for Early departures.

1. Abstract
2. **Project Description**
3. Exploratory Data Analysis
4. Feature Engineering
5. Model Experiments
6. Model Pipeline
7. Model Results
8. 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

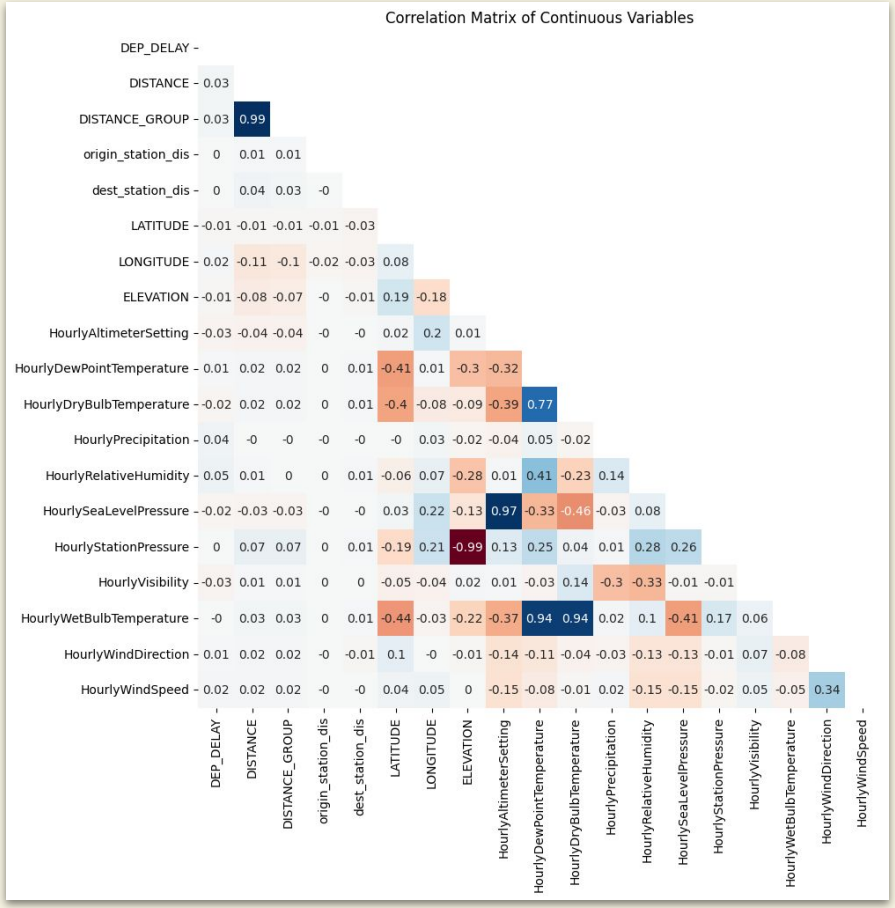
- Utilizing flight data provided by the U.S. Department of Transportation and meteorological data provided by the National Oceanic and Atmospheric Administration repository (NOAA), our goal for this project is as follows:

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. Abstract
2. Project Description
3. **Exploratory Data Analysis**
4. Feature Engineering
5. Model Pipeline
6. Model Experiments
7. Model Results
8. Conclusion

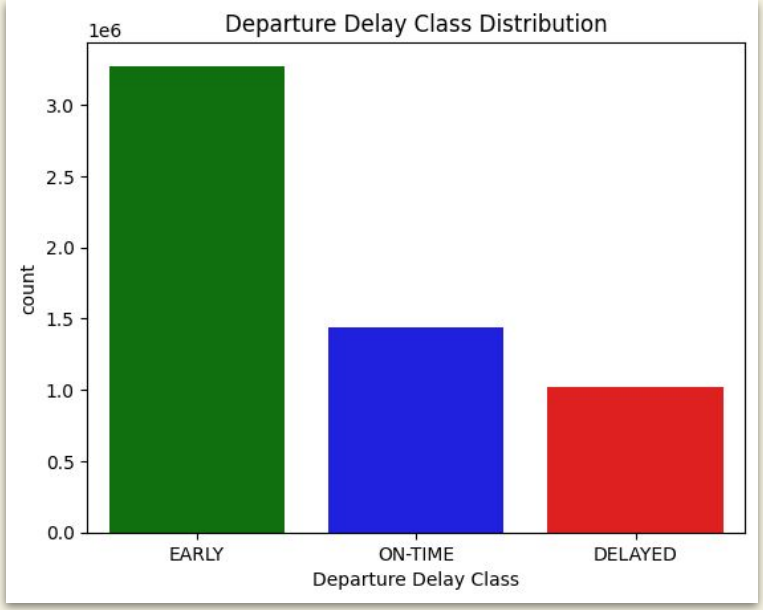
Numerical Analysis

- 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

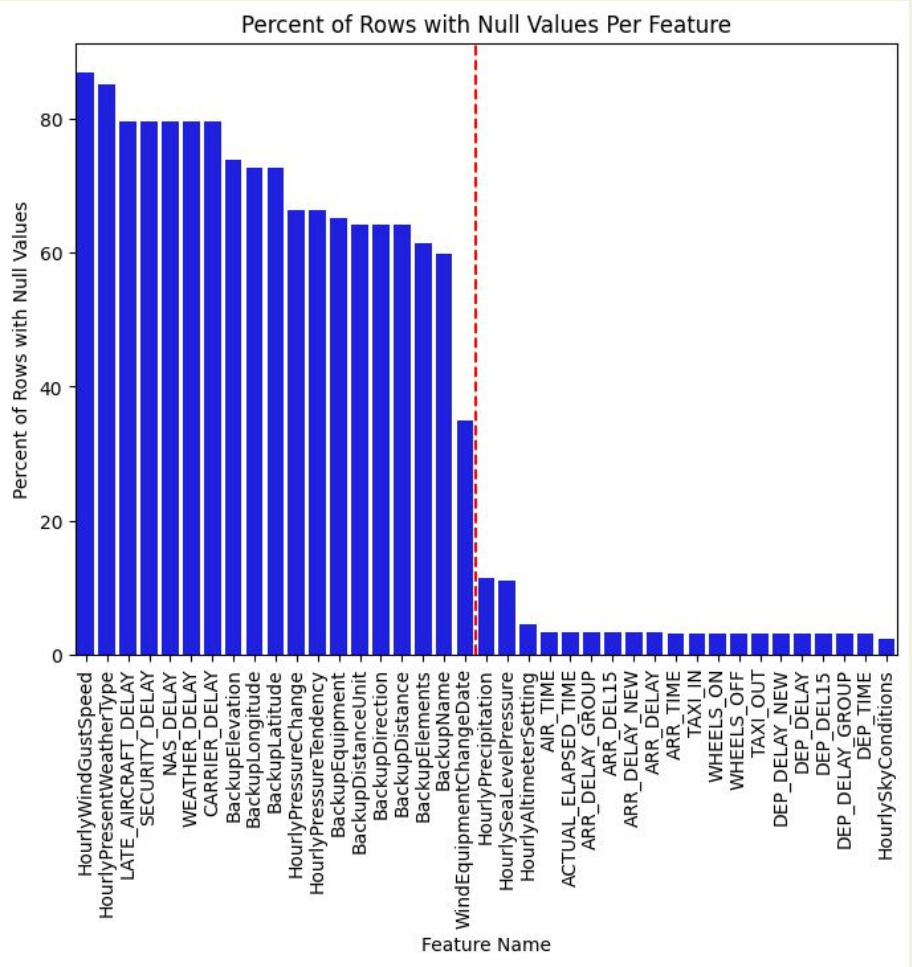
- We are interested in classifying 3 classes
  - Early (Delay < 0 minutes)
  - On-Time (0 <= Delay <= 15 minutes)
  - Delayed (15 minutes < Delay)
- Chose 15 minutes as Delayed based on the definition from Department of Transportation
- Early is by far our majority class representing 57.2% of flights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



1. Abstract
2. Project Description
3. Exploratory Data Analysis
4. **Feature Engineering**
5. Model Pipeline
6. Model Experiments
7. Model Results
8. Conclusion

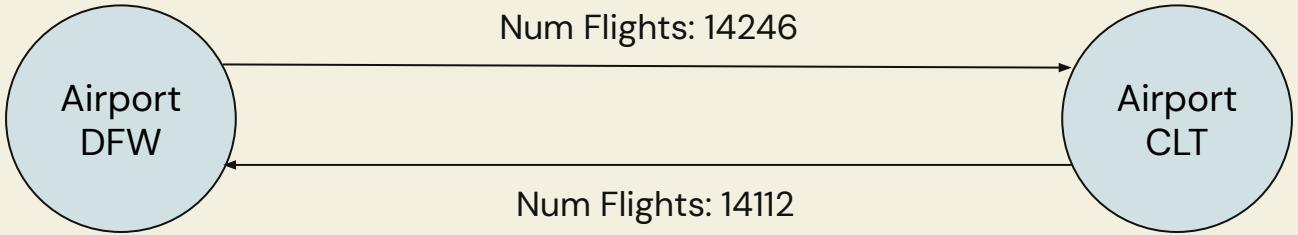
Missing Values

- We filled in missing values with a new “missing” category for categorical values and filled in “0” for Hourly Precipitation (as described in the data dictionary)
- For all other categories, we dropped missing values (leading to 9.65% of rows dropped).
    - Deemed acceptable as we still had 28M rows of data after dropping.



Graph Based Features

- Create a graph utilizing the source and destination airports for each flight.
- Calculated the frequency of flights between different airports (edge weights)
- Compute PageRank and rejoin PageRank with the datasets to replace ORIGIN_AIPORT_ID, and DEST_AIRPORT_ID
- PageRank of airports weighted by number of flights will rank relative importance of airports based on how many outgoing/incoming flights there are.
- Model can thus utilize this information to better predict flight delays.



Time Series and Numeric Features

- 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]
  - Utilized for MLP as it can help predict nonlinearity of time patterns.

- 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

High Cardinality Features (Target Encoding)

- We defined Highly Cardinal Features as any features with > 100 unique categories.
- We then grouped by each feature and calculated the mean delay time.
- Rejoined it back in order to have a numeric feature instead of a one-hot encoded feature and reduced cardinality significantly.
- Risk of Data Leakage with this method (Refer to later Section)

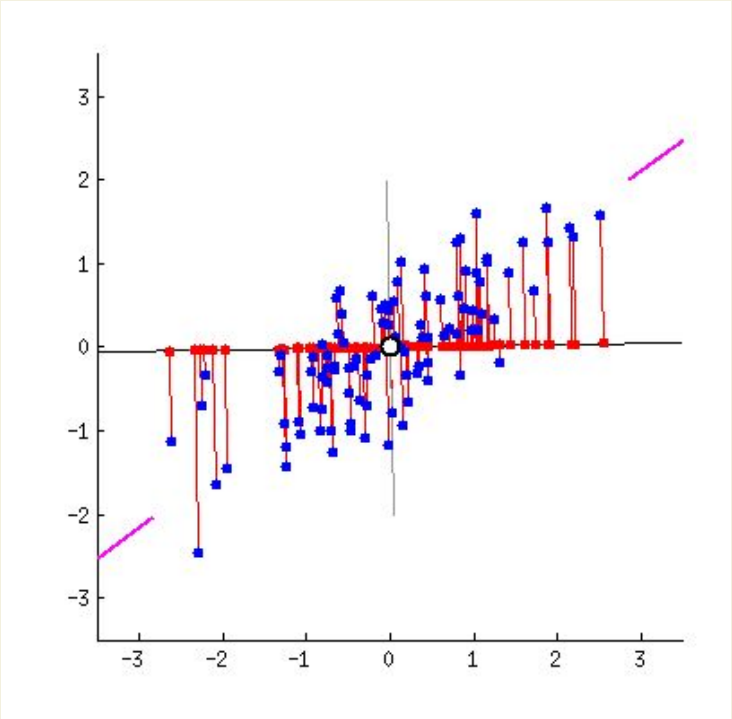
Low Cardinality Features

- One-Hot Encoded the remaining categorical features.
- Our entire Feature Space is now 221 different features (numeric + categorical features)

	num_unique	most_freq_cat	most_freq_count
OP_UNIQUE_CARRIER	14	WN	1245811
ORIGIN_AIRPORT_ID	320	10397	376945
TAIL_NUM	4896	N480HA	3767
OP_CARRIER_FL_NUM	6948	469	3942
ORIGIN_STATE_ABR	53	CA	699590
DEST_AIRPORT_ID	322	10397	376740
DEST_STATE_ABR	53	CA	699782
YEAR	1	2015	5725795
origin_station_id	319	72219013874	376945
origin_type	3	large_airport	5268302
dest_station_id	321	72219013874	376740
dest_type	3	large_airport	5265133
STATION	319	72219013874	376945
HourlyCloudCoverage	10	BKN	1569633
HourlyCloudLayerAmount	9	7	1569443

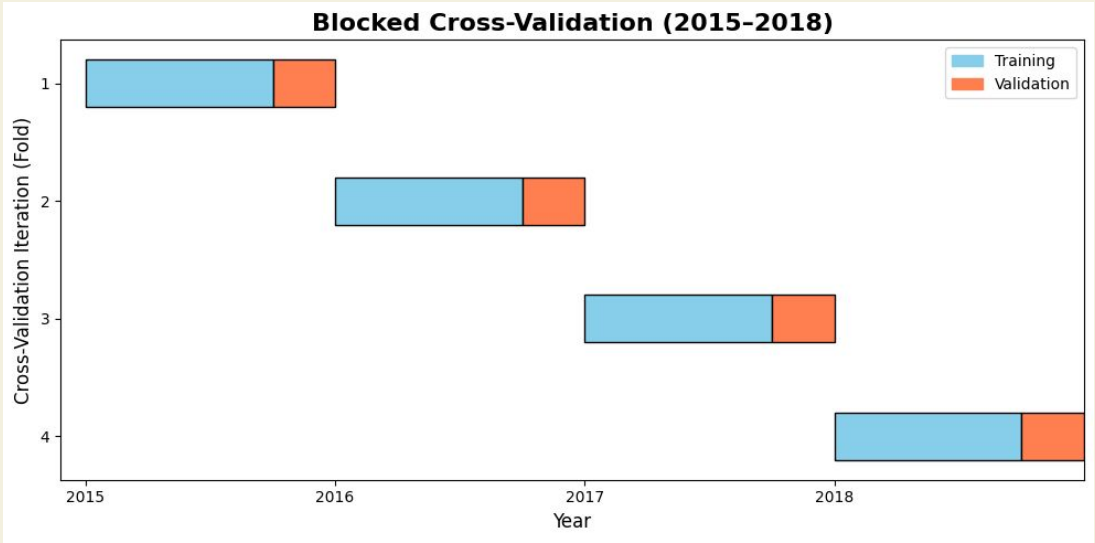
Principal Component Analysis

- For one experiment, run PCA, a dimensionality reduction technique to find a lower dimensional vector(s) that maximizes the variance.
    - To find 2nd, 3rd, 4th, nth principal component, repeat the process but with a new vector that is also orthogonal to the other ones.
    - This ensures that we reduce the dimensionality while minimizing the variance loss in the data.
- We used 110 principal components (50% reduction of feature space) for our testing.



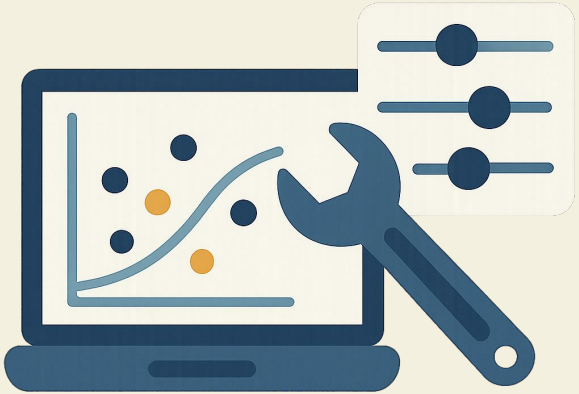
Hyperparameter Tuning

- Created subsamples from each year (2015–2018) instead of only using 2015.
- Each yearly fold contains ~500,000 records to keep processing manageable.
- Combined folds form a CV training set that reflects multi-year temporal variability.
- This method supports efficient hyperparameter tuning on a diverse, time-aware dataset.



Hyperparameter Tuning

- We experimented with three hyperparameter tuning techniques:
  - **Grid Search**: Exhaustively evaluates a predefined grid of hyperparameter values.
  - **Random Search**: Samples hyperparameter combinations randomly, offering a broader exploration.
  - **Bayesian Optimization**: Uses the Optuna library to “intelligently” sample hyperparameters based on past evaluations.



Grid vs Random vs Bayesian Search

Grid Search

- In our experiments, Grid Search and Random Search both took similar runtimes (for example, exploring 9 hyperparameter combinations over a similar duration).

Random Search

- It performs comparably to Grid Search in terms of runtime for the same number of evaluations.

Bayesian Search (Optuna)

- Uses a probabilistic model to guide the hyperparameter search based on previous outcomes.
- Each trial informs the next, allowing for more targeted exploration that often yields better metric scores.
- However, because trials run sequentially (each taking roughly 10 minutes), the overall runtime is significantly longer.

Conclusion: We chose Grid Search for our project because it is straightforward to implement, leverages parallel processing, and is more runtime efficient, even though Bayesian Search may provide slightly better metric scores per trial.

Data Leakage

Data leakage occurs when information from outside the training set is used during model training, leading to overly optimistic performance estimates. For example, if future flight delay data is included in training, the model might learn from data that won't be available at prediction time.

How To Avoid Data Leakage?

- **Robust Scaling**: Fit RobustScaler on numeric features (e.g., DISTANCE, ELEVATION, HourlyWindSpeed) using only training data in each fold, then transform the test data.
- **PageRank**: Compute the PageRank feature on training data within each fold and then transform the test data.
- **Categorical Target Encoding**: Encode high-cardinality features (e.g., TAIL_NUM, STATION) using training data only, then apply the transformation and robust scaling to the test data.
- **Resampling**: Optionally apply upsampling, downsampling, or SMOTE on the training data within each fold.
- **Avoid Training Target Variable**: Exclude the DEP_DELAY variable from the final feature vector.

Use these methods uniformly when training on the complete 2015-2018 dataset.

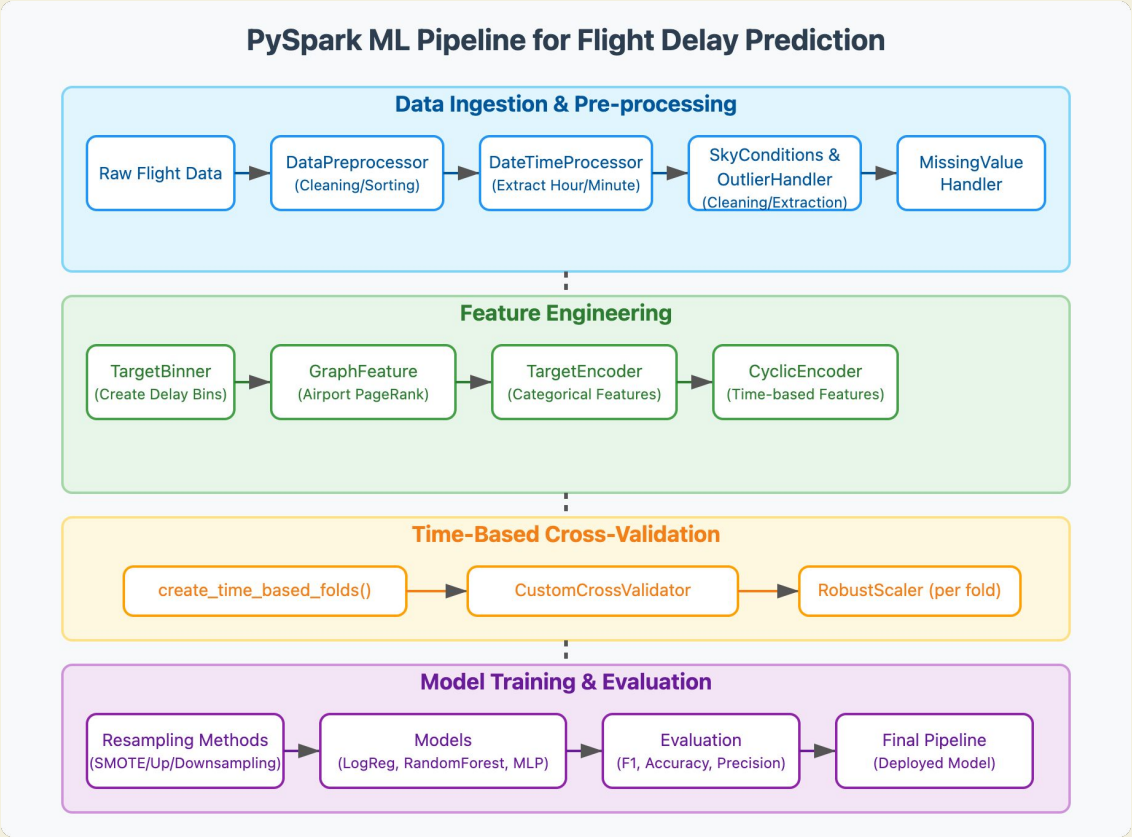
1. Abstract
2. Project Description
3. Exploratory Data Analysis
4. Feature Engineering
5. **Model Pipeline**
6. Model Experiments
7. Model Results
8. Conclusion

Model Pipeline - Inputs

- Inputs:
  - ~28 raw input features expanded to ~221 after feature engineering
- Checkpointing:
  - After major transformations on our dataset, we checkpointed the dataset for experimentation with models without having to re-process with whole pipeline.
- Cluster Details:
  - 5-10 Workers
    - 160-320 GB Memory
    - 40-80 Cores
  - 1 Driver
    - 128 GB Memory, 32 Cores

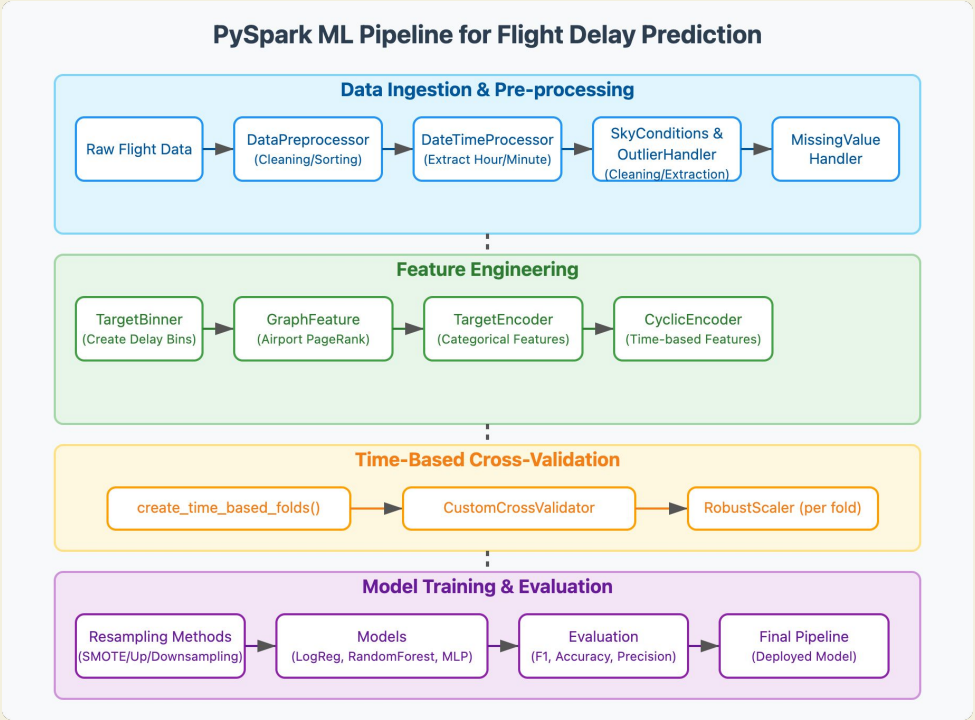
Feature Family	Description	Count	Examples (Raw Features)
Temporal	Date/time related features	~12	YEAR, MONTH, DAY_OF_MONTH, QUARTER, sched_depart_hour_UTC, etc.
Carrier & Flight	Airline and flight identifiers	~5	OP_UNIQUE_CARRIER, OP_CARRIER_AIRLINE_ID, TAIL_NUM, OP_CARRIER_FL_NUM
Airport & Geography	Origin/destination airports	~8	ORIGIN_AIRPORT_ID, DEST_AIRPORT_ID, origin_station_id, dest_station_id, origin_type, dest_type
Weather	Weather conditions	~10	HourlyDewPointTemperature, HourlyRelativeHumidity, HourlyVisibility, HourlyPrecipitation, HourlyCloudCoverage, etc.

Model Pipeline - Block Diagram



Model Pipeline - Improvements

- Refactored to be more modular for testing different models (previously hard-coded)
- Additional/modified Transformers:
  - OutlierHandler
  - GraphFeature
  - TargetEncoder
  - MissingValueHandler*
- Resampling capabilities for class imbalance - SMOTE, Upsampling, Downsampling



1. Abstract
2. Project Description
3. Exploratory Data Analysis
4. Feature Engineering
5. Model Pipeline
6. **Model Experiments**
7. Model Results
8. Conclusion

Data for Small Scale Experimentation

- 12 Month Dataset with Over-Sampling
  - For this experimentation we used a duplication over-sampling to match the majority class (Early)
- Class Distribution:
  - Early Class: 1842967
  - On-Time: 1844086
  - Delayed: 1844551
- Sequential Train/Test Split
  - First 80% of 2015 used for training
  - Last 20% of 2015 used for testing

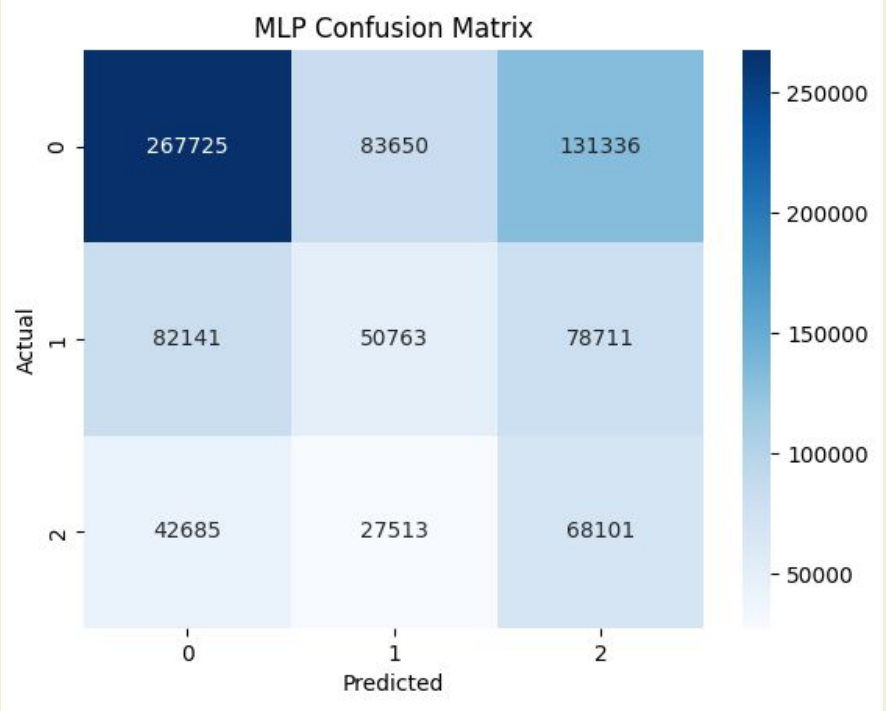
MLP Classifier (Experiment 1)

Architecture

- 6 layers:
  - Input layer: 189 features
  - 1st hidden layer: 100 neurons
  - 2nd hidden layer: 50 neurons
  - 3rd hidden layer: 25 neurons
  - 4th hidden layer: 10 neurons
  - Output layer: 3 classes

Results:

- Training:
  - Accuracy: 44.65%
  - Weighted Precision: 43.94%
  - F1 Score: 43.61%
- Test:
  - Accuracy: 46.43%
  - Weighted Precision: 51.57%
  - F1 Score: 47.81%



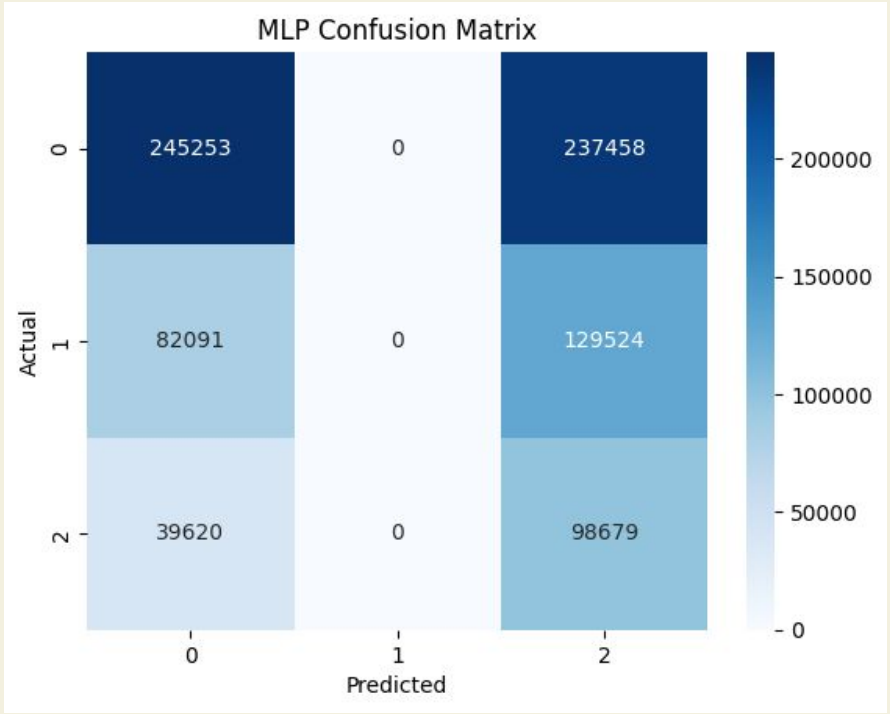
MLP Classifier (Experiment 2)

Architecture

- 7 layers:
  - Input layer: 189 features
  - 1st hidden layer: 150 neurons
  - 2nd hidden layer: 75 neurons
  - 3rd hidden layer: 50 neurons
  - 4th hidden layer: 25 neurons
  - 5th hidden layer: 10 neurons
  - Output layer: 3 classes

Results:

- Training:
  - Accuracy: 41.75%
  - Weighted Precision: 28.19%
  - F1 Score: 33.22%
- Test:
  - Accuracy: 41.31%
  - Weighted Precision: 42.26%
  - F1 Score: 38.90%



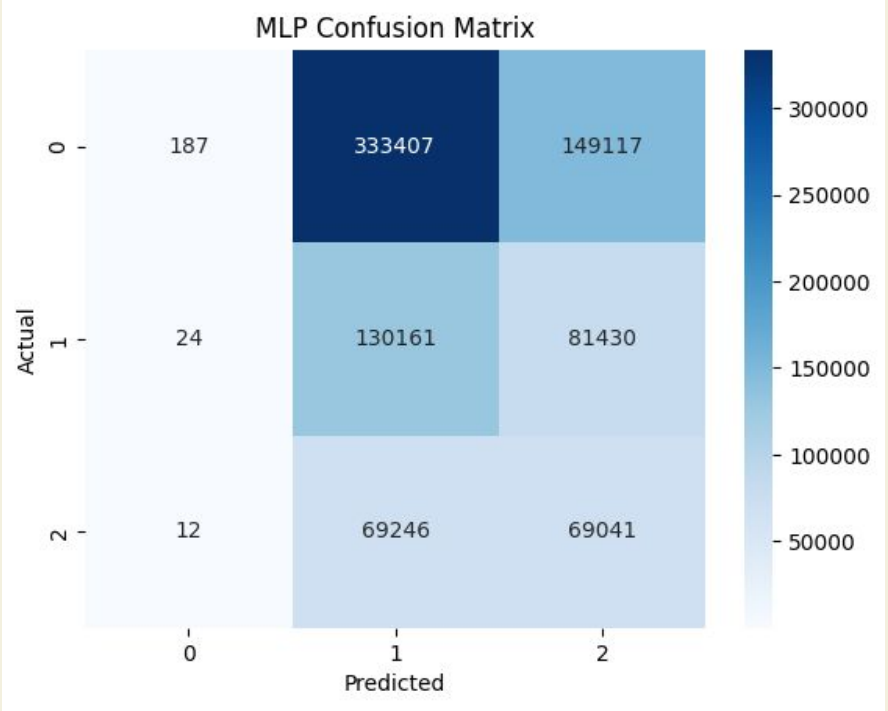
MLP Classifier (Experiment 3)

Architecture

- 5 layers:
  - Input layer: 189 features
  - 1st hidden layer: 150 neurons
  - 2nd hidden layer: 75 neurons
  - 3rd hidden layer: 20 neurons
  - Output layer: 3 classes

Results:

- Training:
  - Accuracy: 37.66%
  - Weighted Precision: 46.54%
  - F1 Score: 30.28%
- Test:
  - Accuracy: 23.95%
  - Weighted Precision: 58.65%
  - F1 Score: 14.17%



1. Abstract
2. Project Description
3. Exploratory Data Analysis
4. Feature Engineering
5. Model Pipeline
6. Model Experiments
7. **Model Results**
8. Conclusion

Phase 2 Model Comparisons Across Evaluation Metrics

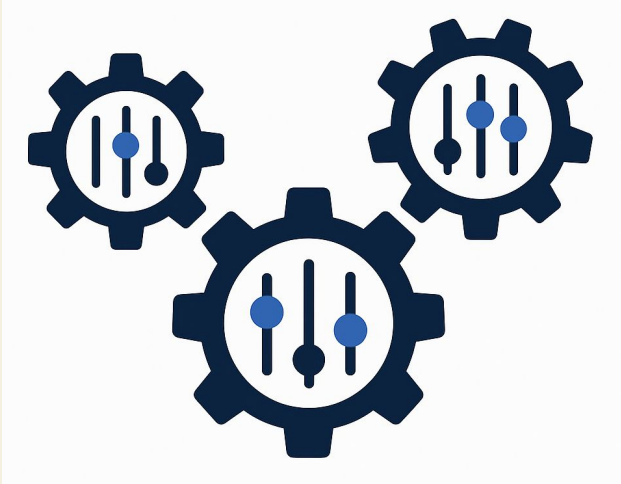
Model	Train F1	Train Precision	Train Recall	Train Accuracy	Test F1	Test Precision	Test Recall	Test Accuracy
Decision Tree (Baseline)	0.470	0.445	0.570	0.570	0.505	0.460	0.586	0.586
Logistic Regression	**0.520**	**0.525**	**0.582**	**0.582**	**0.537**	**0.523**	0.561	0.561
Random Forest	0.4240	0.4492	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

Phase 3 Model Comparisons Across Evaluation Metrics

Model	Train F1	Train Precision	Train Recall	Train Accuracy	Test F1	Test Precision	Test Recall	Test Accuracy
Logistic Regression w/under-sampling	0.450	0.451	0.455	0.455	0.493	0.537	0.479	0.479
Logistic Regression w/over-sampling	0.461	0.470	0.481	0.481	0.542	0.545	0.550	0.553
Random Forest w/under-samping	**0.545**	**0.552**	**0.550**	**0.550**	0.500	0.545	0.480	0.480
Random Forest w/ over-sampling	0.446	0.476	0.483	0.483	**0.544**	0.543	**0.575**	**0.575**
MLP 1 w/ under-sampling	0.454	0.460	0.462	0.462	0.503	**0.548**	0.488	0.488
MLP 2 w/ under-sampling	0.442	0.452	0.457	0.457	0.502	0.542	0.491	0.491
MLP 1 w/ over-sampling	0.456	0.469	0.481	0.481	0.539	0.543	0.549	0.549
MLP 2 w/ over-sampling	0.384	0.465	0.471	0.471	0.491	0.538	0.541	0.541
MLP 1 w/ over-sampling and PCA	0.455	0.467	0.479	0.479	0.534	0.542	0.547	0.547

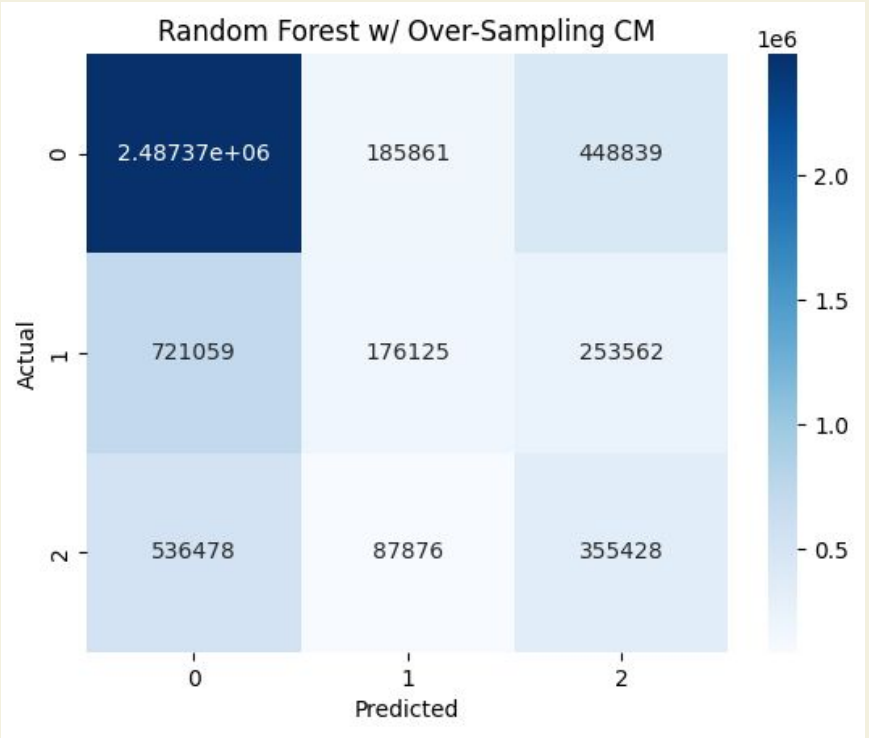
Phase 3 Hyperparameters

Model	Hyperparameters
Logistic Regression	maxIter=300, regParam=0.10, elasticNetParam=0.0
Random Forest w/under-sampling	numTrees=20, maxDepth=15
Random Forest w/over-sampling (Best Model)	numTrees=15, maxDepth=10
MLP 1 and 2	maxIter=50, StepSize=0.05



Best Model (RF w/ Over-Sampling)

- Individual Class Metrics:
  - Early:
    - Precision: 66.42%
    - Recall: 79.67%
  - On-Time:
    - Precision: 39.15%
    - Recall: 15.31%
  - Delayed:
    - Precision: 33.59%
    - Recall: 36.28%
- Even with class balancing techniques we still struggled with predicting the On-Time and Delayed classes



1. Abstract
2. Project Description
3. Exploratory Data Analysis
4. Feature Engineering
5. Model Pipeline
6. Model Experiments
7. Model Results
8. **Conclusion**

Conclusion

- **Designed for Scalability:**
  - Built a distributed, modular pipeline with strong data leakage prevention (e.g., fold-specific transformations, target encoding isolation).
  - Compared multiple hyperparameter tuning strategies (Grid Search, Random Search, and Bayesian Optimization) to balance performance with runtime efficiency.
- **Best Model:**
  - Our Random Forest classifier with oversampling achieved the best performance (Test F1 = 54.4%) with high accuracy for Early departures (the majority class).
- **Performance Gaps:**
  - Class imbalance remains a key challenge—strong performance for Early, but On-Time and Delayed classes need improvement.
- **Operational Value:**
  - Despite prediction challenges for on-time and delayed classes, early classifications from this model still have the potential to improve crew scheduling, gate planning, and customer communication.