databricks W261_Team5-1_Phase-III_Project_Report

(https://databricks.com)

Flight Ahead - Advanced Predictive Models for Flight Delays

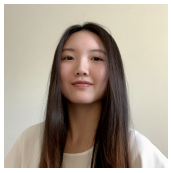
1. Team and Project Information

1.1Team Information

Group Number: Team 5-1

Phase III Leader: Carolyn Dunlap

Team Members, Emails, and Photos:

Members	Emails	Photos
Hamsini Sankaran	hamsini.sankaran@berkeley.edu (mailto:hamsini.sankaran@berkeley.edu)	
Sivakumar Thiyagarajan	siva.thiyagarajan@berkeley.edu (mailto:siva.thiyagarajan@berkeley.edu)	
Carolyn Dunlap (Phase III Leader)	**cadunlap@berkeley.edu (mailto:cadunlap@berkeley.edu)**	
Yin Yin Teo	teoyinyin@ischool.berkeley.edu (mailto:teoyinyin@ischool.berkeley.edu)	

1.2 Project Information

Project Title: Flight Ahead - Advanced Predictive Models for Flight Delays

Phase III presentation:
https://docs.google.com/presentation/d/1Cta1IY4YyFpzJNXAXTT6pNRCsVv5zrPQv2E
usp=sharing
(https://docs.google.com/presentation/d/1Cta1IY4YyFpzJNXAXTT6pNRCsVv5zrPQv2
usp=sharing)

Corresponding coding notebooks:

- EDA (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1346418319520942/command/13464183195

1.3 Phase Leader Plan

Phase Name	Description	Leader
Phase 1	Project Plan, describe datasets, joins, tasks, and metrics, set up blob storage	Hamsini Sankaran
Phase 2	EDA, baseline pipeline on all available data, Scalability, Efficiency, Distributed/parallel Training, Scoring Pipeline, Feature engineering and hyperparameter tuning	Sivakumar Thiyagarajan
Phase 3	**Select the optimal algorithm, fine-tune and submit a final report**	**Carolyn Dunlap**

1.4 Credit Assignment Plan (Updated 12/16/2023)

Phase III

Phase III Tasks.	Team Member	Notes/Details	Start Date	End Date	Hours of Effort
Presentation	Carolyn Dunlap, Sivakumar Thiyagarajan, Hamsini Sankaran, Jasmine Teo	creating presentation, speaking to slides	12/13	12/15	20
Abstract	Jasmine Teo	writing the section	12/14	12/16	1
Data and Feature Engineering	Jasmine Teo	writing the section	12/14	12/16	10
EDA of Final Features	Jasmine Teo	EDA of final features used in model	12/10	12/16	40
Data Leakage and Model Pipeline	Carolyn Dunlap	writing the section, all members contirubted to pipeline discussion	12/15	12/16	10
Feature Importance	Carolyn Dunlap, Sivakumar Thiyagarajan	random forest based feature importance, all members discussed results	12/07	12/10	30
Neural Network (MLP)	Hamsini Sankaran	writing the section, MLP model development and testing for 2 MLP models	12/08	12/16	50
Neural Network (GRU RNN)	Hamsini Sankaran	GRU RNN model development	12/13	12/16	20

Phase III Tasks.	Team Member	Notes/Details	Start Date	End Date	Hours of Effort
Naive Bayes model	Hamsini Sankaran	Naive Bayes model development and testing	12/13	12/16	20
Logisitic Regression, Random Forest, and GBT models	Sivakumar Thiyagarajan	model development and testing on 5 years	12/11	12/14	50
Hyperparameter Tuning	Sivakumar Thiyagarajan	hyperparameter tuning on logistic regression, random forest, MLP (2 hidden layers) and	12/13	12/16	20
Ensemble model	Sivakumar Thiyagarajan	ensemble model development and testing	12/13	12/16	10
MLP subsetted training models	Carolyn Dunlap	model testing and assessment for individual folds	12/14	12/15	10
Results and discussion	Carolyn Dunlap, Sivakumar Thiyagarajan, Hamisini Sankaran	report writing	12/15	12/16	10

2. Project Abstract

Flight delays cost airlines an estimated billions of dollars every year, as a result of both direct costs such as additional personnel wages or vouchers for passengers and indirect costs such as the value of passengers' time lost and impact in future sales from decreased customer trust. To address this need, we developed a

predictive model using historical flight and weather data to forecast delays exceeding 15 minutes at least 2 hours prior to departure. We developed baseline and advanced models using logistic regression, random forest, gradient-boost decision trees, and neural network classifiers. Because identifying delayed flights is paramount, but both recall and precision are important metrics to consider, we have chosen to evaluate our models using an F2-score (the weighted mean between precision and recall with more weighted placed on recall). Overcoming challenges with time-series data, our iterative model development includes data preprocessing, feature selection, imputation, one-hot encoding, downsampling, sliding window cross-fold validation, and hyperparameter tuning. We employed in depth feature engineering, including leveraging a flight's previous departure delay. Our top-performing model was our multi-layer perceptron model with 2 hidden layers, which achieved an F2-score on our blind test flight data of 52.26%, 81.86% recall, and 21.36% precision. Our model will enable airlines to better predict flight delays, mitigating costs and improving customer retention.

3. Data and Feature Engineering

3.1 The Dataset

1. Flights Data

Source: Obtained from the U.S. Department of Transportation (DOT) (https://www.transtats.bts.gov/Fields.asp?gnoyr_VQ=FGJ).

Description: This dataset represents a subset of passenger flight on-time performance data spanning from 2015 to 2021. With 31,746,841 rows and 109 columns, it provides essential flight-related information crucial for predicting delays.

2. Weather Data

Source: Acquired from the National Oceanic and Atmospheric Administration (NOAA) (https://www.ncei.noaa.gov/access/metadata/landing-page/bin/iso?id=gov.noaa.ncdc:C00679).

Description: Covering the years 2015 to 2021, this dataset includes weather conditions at the time of departure and arrival for origin and destination airports. With 630,904,436 rows and 177 columns, it facilitates the creation of features based on weather conditions.

3. Airport Dataset

Description: This dataset offers metadata about 18,097 airports, providing valuable insights into airport-specific factors that may influence flight delays. It encompasses 10 attributes for each airport.

4. Airport Codes Table

Source: Utilized an external airport code conversion set available here (https://datahub.io/core/airport-codes).

Description: Crucial for identification and categorization, this dataset includes IATA (International Air Transport Association) and ICAO (International Civil Aviation Organization) codes.

5. OTPW (Ontime Performance of Flights and Weather Dataset Joined):

This dataset combines the above 4 datasets and was subsetted into 3 months, 1 year, and 3 years, and 5 years data for development purposes. An overview of each of these subsetted data is described in the table below. This integration of flight performance and weather data enables more accurate predictions and analysis. Due to the convenience of this pre-appended dataset and given our team's time constraints, we focused on the OTPW dataset for all of our analysis and modeling results.

3.2 Features Explored Data Dictionary

The table below contains all features explored in this phase and previous phases.

Feature (Column Name)	Data Type	Description
QUARTER	numeric	Quarter in which the flight occurred (1-4).

Feature (Column Name)	Data Type	Description
DAY_OF_MONTH	numeric	Day of the month on which the flight occurred.
DAY_OF_WEEK	numeric	Day of the week on which the flight occurred.
FL_DATE	datetime	Date of the flight in yyyymmdd format.
OP_UNIQUE_CARRIER	string	Unique carrier code for the operating airline.
ORIGIN	string	Origin airport code.
DEST	string	Destination airport code.
CRS_DEP_TIME	numeric	CRS Departure Time (local time: hhmm).
DEP_DEL15	binary	**Target feature** Departure Delay Indicator, 15 Minutes or More (1=Yes).
TAXI_IN	numeric	Taxi In Time, in Minutes.
CRS_ARR_TIME	numeric	CRS Arrival Time (local time: hhmm).
ARR_TIME	numeric	Actual Arrival Time (local time: hhmm).
CANCELLED	binary	Cancelled Flight Indicator (1=Yes).
DIVERTED	binary	Diverted Flight Indicator (1=Yes).
CRS_ELAPSED_TIME	numeric	CRS Elapsed Time of Flight, in Minutes.
FLIGHTS	numeric	Number of Flights.
DISTANCE	numeric	Distance between airports (miles).
YEAR	numeric	Year of the observation
MONTH	numeric	Month of the observation
origin_type	string	Type of origin (e.g., airport)
origin_airport_lat	numeric	Latitude of the origin airport

Feature (Column Name)	Data Type	Description
origin_airport_lon	numeric	Longitude of the origin airport
origin_station_dis	numeric	Distance of the observation station from the origin
dest_type	string	Type of destination (e.g., airport)
dest_airport_lat	numeric	Latitude of the destination airport
dest_airport_lon	numeric	Longitude of the destination airport
dest_station_dis	numeric	Distance of the observation station from the destination
sched_depart_date_time	datetime	Scheduled departure date and time
ELEVATION	numeric	Elevation of the observation station
REPORT_TYPE	string	Type of weather report
SOURCE	string	Source of the weather report
HourlyAltimeterSetting	numeric	Altimeter setting reported hourly
HourlyDewPointTemperature	numeric	Dew point temperature reported hourly
HourlyPrecipitation	numeric	Precipitation reported hourly
HourlyRelativeHumidity	numeric	Relative humidity reported hourly
HourlySeaLevelPressure	numeric	Sea-level pressure reported hourly
HourlyStationPressure	numeric	Station pressure reported hourly
HourlyVisibility	numeric	Visibility reported hourly
HourlyWindDirection	numeric	Wind direction reported hourly
HourlyWindSpeed	numeric	Wind speed reported hourly
ARR_TIME_OF_DAY	string	Categorized part of the day for arrival time

Feature (Column Name)	Data Type	Description
DEP_TIME_OF_DAY	string	Categorized part of the day for departure time
prev_DEP_DELAY_NEW	numeric	Previous flight's departure delay (in minutes)
prev_DEP_DEL15	binary	Previous flight's departure delay indicator (1=Yes, 0=No)
FEW	binary	Indicator for Few Clouds
SCT	binary	Indicator for Scattered Clouds
BKN	binary	Indicator for Broken Clouds
OVC	binary	Indicator for Overcast Clouds
VV	binary	Indicator for Vertical Visibility
is_holiday	binary	Indicator for Range of Dates Considered a Holiday

3.3 Feature Engineering

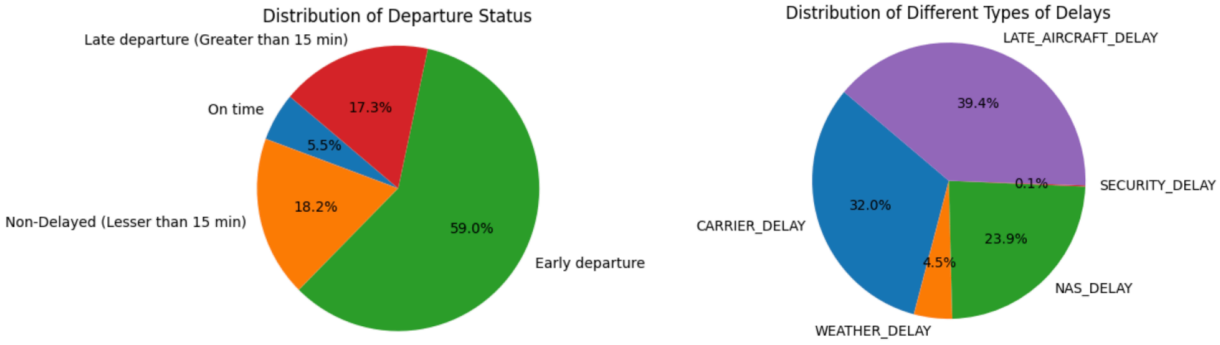
In addition to the features that were included in our dataset, we also derived several features for consideration into our final model, based on prior knowledge or EDA conducted on the 3 months, 1 year, or 3 years dataset, described in the earlier Phases of this project. We introduced indicators for holidays and weekends to capture unique travel patterns and delays during these periods. The segmentation of the day into departure blocks helped the model to disern time-related patterns crucial for delay prediction. Another notable addition was the previous flight's departure delay, considered only when the time difference between the previous departure and current scheduled departure exceeded 2 hours. Finally, we incorporated hourly indicators for sky conditions, such as few clouds, scattered clouds, broken clouds, overcast, and vertical visibility. All derived features can be found in the table below.

Derived Feature	Description of Derived Feature	Raw Feature
Flight Frequency	Number of flights based on specific tail number	Tail Number
Flights per day at Destination airport	Number of flights per day at the destination airport	Day of week, Carrier, Destination
Flights per day at Origin airport	Number of flights per day at the origin airport	Day of week, Carrier, Origin
Average Carrier Delay	Average delay attributed to a specific carrier	Departure Delay, Carrier
Departure time of the day	Time of day when a flight departs (0 <= hour < 6: 'Early Morning', 6 <= hour < 12: 'Morning', 12 <= hour < 18: 'Afternoon', 18 <= hour < 20: 'Evening', others: 'Night')	Scheduled Departure Time block
Arrival time of the day	Time of day when a flight arrives (0 <= hour < 6: 'Early Morning', 6 <= hour < 12: 'Morning', 12 <= hour < 18: 'Afternoon', 18 <= hour < 20: 'Evening', others: 'Night')	Scheduled Arrival Time Block
Prior Departure delay	Previous aircraft's departure delay information, if the departure delay is known at least 2 hours prior to the scheduled departure of interest	Departure Delay
Holiday (-5/+2 days)	Whether the flight falls within -5 to +2 days of a holiday	Flight Date
Few Clouds, Scattered Clouds, Broken Clouds, Over Cast, Vertical Visibility	Sky or cloud conditions at different hours of the day (coded as indicator variables)	Hourly Sky Conditions

3.4 Exploratory Data Analysis (EDA)

In the previous phases, our team worked mainly with 3 month, 1 year, and 3 year flight data. In order gain a deeper understanding of the 5 year dataset and identify any potential patterns we may have missed in previous EDA, our team delved into a few visualizations that depicted various aspects, such as the distribution of missing values. Additionally, we examined departure statuses and the types of flights that experienced delays. We compared these visuals to that of 3 year data and concluded that the data is relatively similar to the 5 year dataset.

Figure 1a. Departure Status (left) and Type of Delayed Flights (5-Year)



In Figure 1a, the left side features a pie chart showing the distribution of departure statuses, while the one on the right displays the distribution of various types of delays. Notably, approximately 17.3% of all flights experienced late departures (more than 15 minutes behind schedule). Furthermore, our observation reveals that the predominant causes of delays are late aircraft and carrier-related issues.

Figure 1b. Missing Values (5-Year)

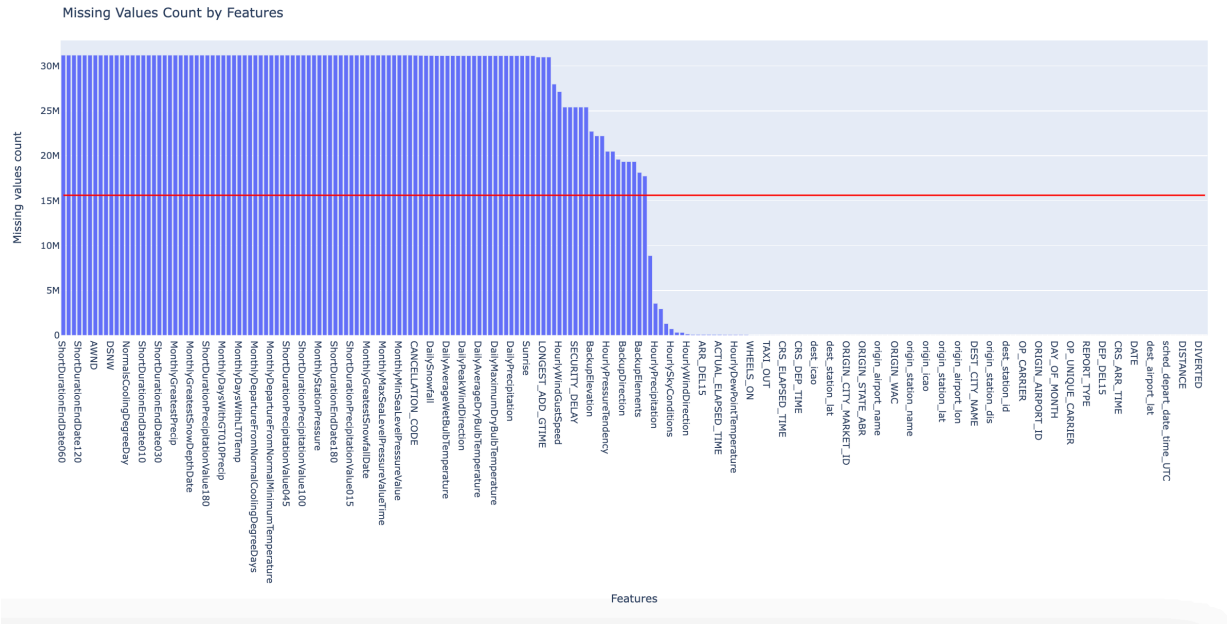
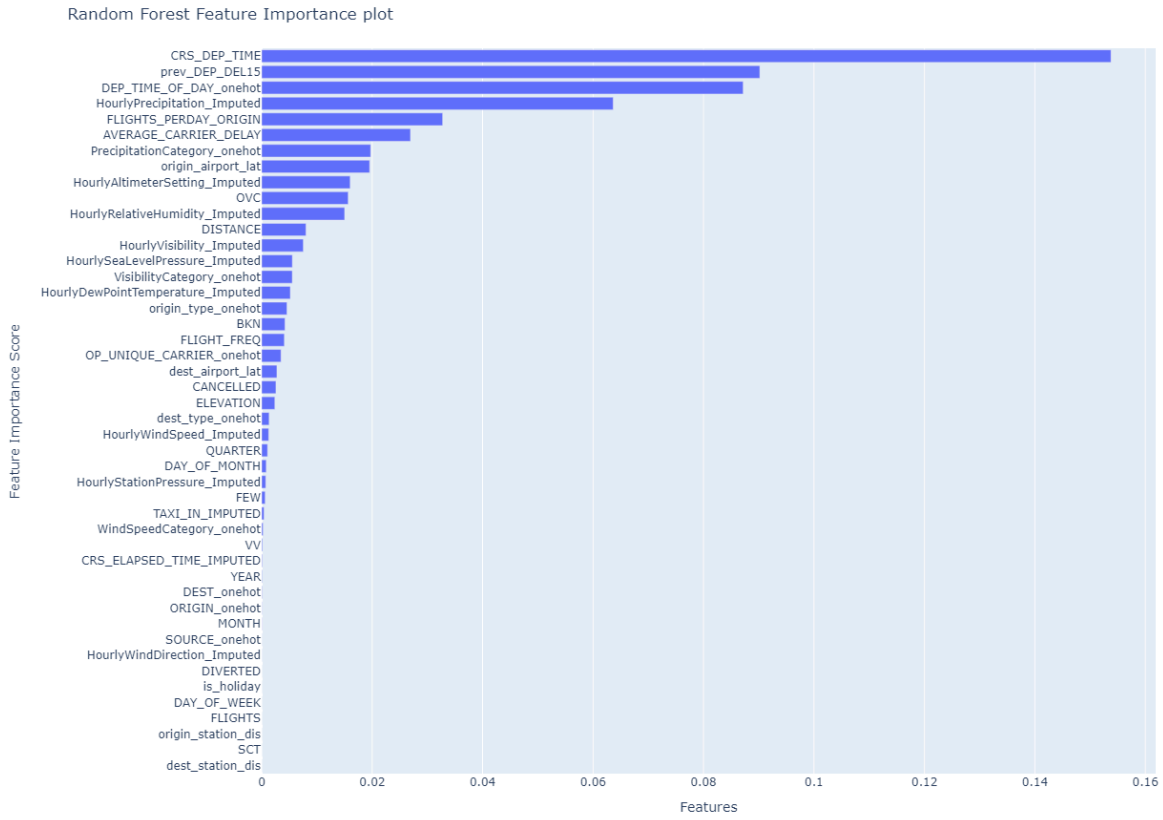


Figure 1b highlights a missing values analysis of the raw OPTW dataset spanning 3 months, 1 year, 3 years, and 5 years. Among these periods, 110 features consistently exhibited over 50% missing or null values, primarily pertaining to weather-related variables with daily or monthly observations. Notably, hourly weather data is largely intact. The dataset combines weather events averaged to individual flight information, suggesting that an aggregation or join step may have converted many daily or monthly weather readings into null values. Given time constraints, we've opted to remove columns with over 50% null values to ensure downstream processing reliability. Variables with less than 50% missing values will undergo imputation, detailed in the Modeling Pipelines section.

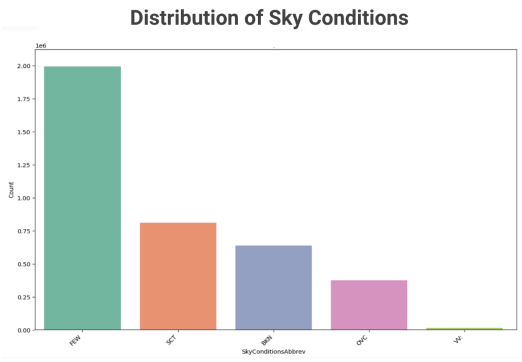
Based on this EDA, along with the extensive analyses conducted by our team previously in Phase II, we considered potential points of data leakage and devised a methodical modeling pipeline to remove features that were uninformative or unlikely to add value to our models in an unbiased manner. As part of this more unbiased approach to final feature selection, we were able to identify the top features that affected our model based on feature importance scores generated from a random forest classifier (Figure 2a). This classifier was trained on the 3 year dataset, and to better understand why these features were select, we decided to perform additional EDA to explore these features more on the same 3 year data.

Figure 2a. Top Features Determined by Random Forest (3-Year)

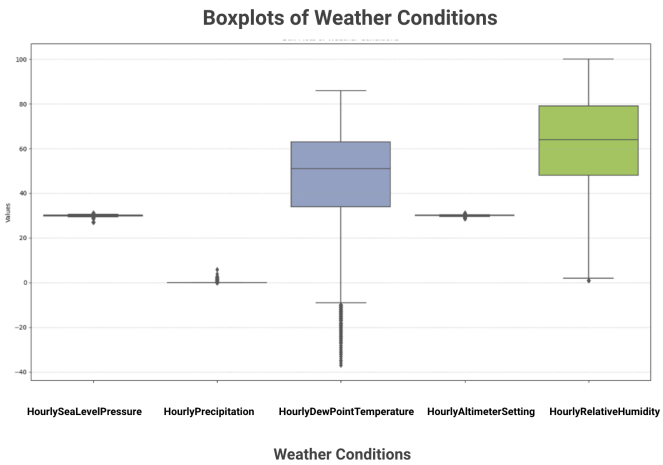


As mentioned above, Figure 2a displays the key features impacting our model as per their feature importance scores. Subsequently, we will examine each variable in the upcoming figures.

Figure 2b. Sky (left) and Weather (right) Conditions (3-Year)



FEW: Few clouds (1/8 to 2/8 of the sky covered)
SCT: Scattered clouds (3/8 to 4/8 of the sky covered)
BKN: Broken clouds (5/8 to 7/8 of the sky covered)
OVC: Overcast clouds (8/8 of the sky covered)
VV: Vertical visibility (sky is completely obscured, often due to fog or heavy precipitation)



In Figure 2b, the graph on the left reveals predominant cloud conditions. We identified "OVC" (overcast clouds, where virtually 100% of the sky is covered) as conditions with the potential to contribute significantly to flight delays. Notably, our random forest model, which will be discussed later, underscored the importance of "OVC" by highlighting it as one of the more influential features. In the boxplot of weather conditions on the right, we see that the hourly dew point temperature and relative humidity exhibit more dispersed distributions in comparison to the other weather condition types. This variability suggests that these parameters could contribute more significantly to the performance of our model when predicting flight delays.

Figure 2c. Histogram of Distance Travelled (3-Year)

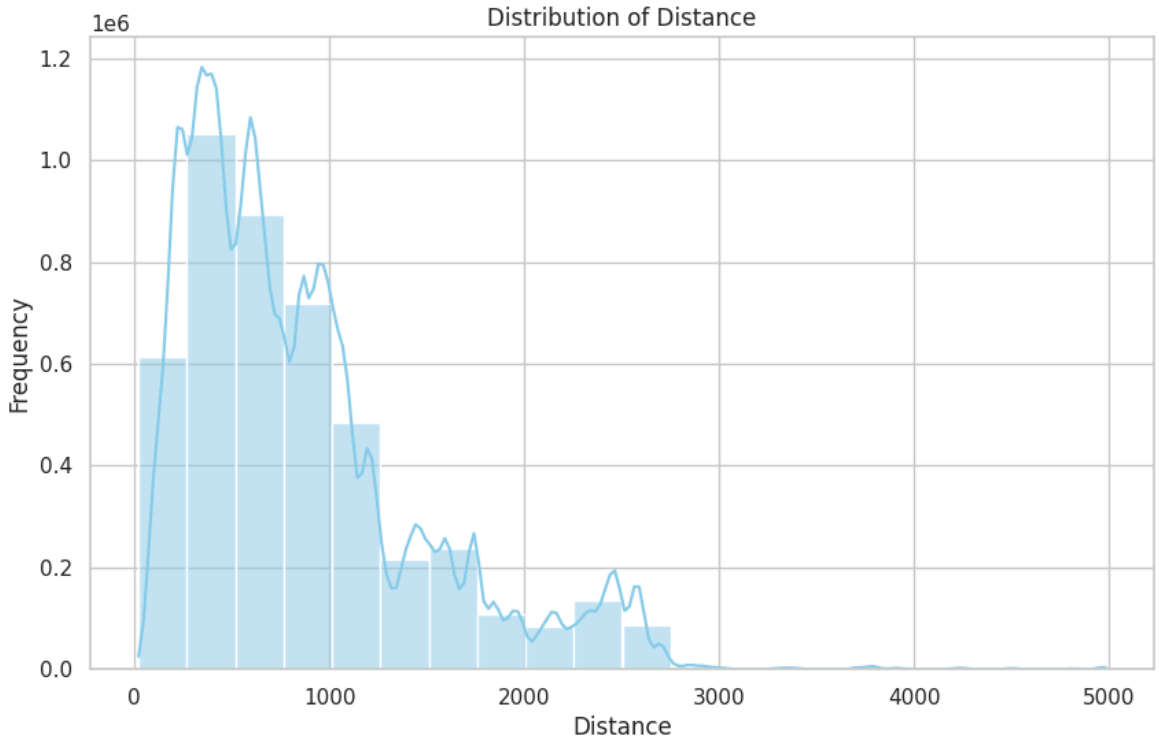
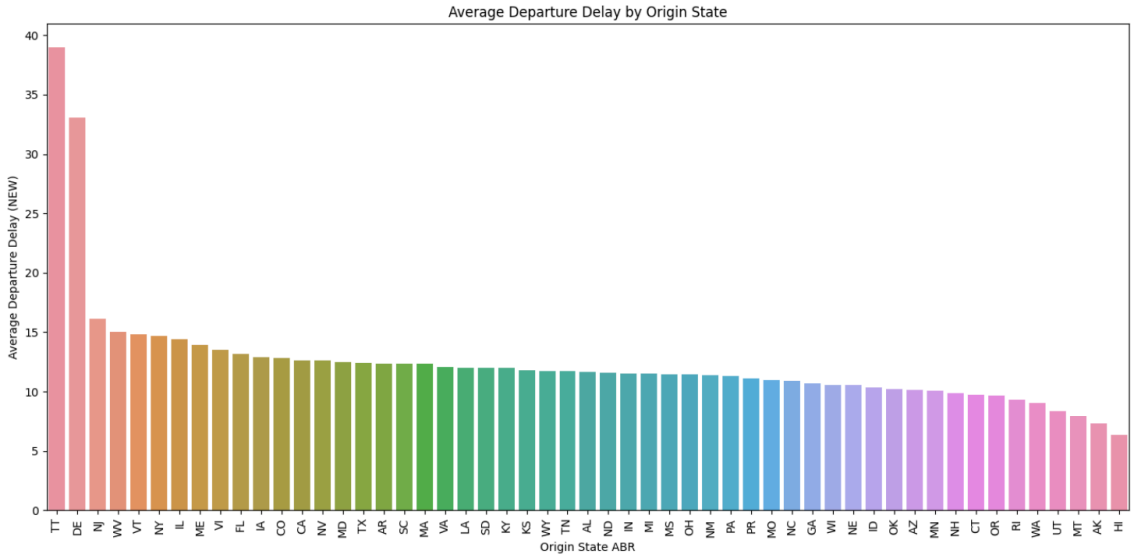


Figure 2c. shows the distribution of total distance travelled for each departured flight. We notice see that it exhibits a more dispersed distribution and somewhat skewed to the right, suggesting that these parameters could contribute more significantly to the performance of our model when predicting flight delays.

Figure 2d. Departure Delay by Origin State (3-Year)



In Figure 2d, we examined the average departure delay by origin state, with notable observations in Trust Territories (TT) and Delaware (DE). This is valuable for commercial airlines as it provides insights into regional patterns of departure delays, enabling targeted strategies for resource allocation and planning.

Figure 2e. Flight Delay Metrics Per Month (3-Year)

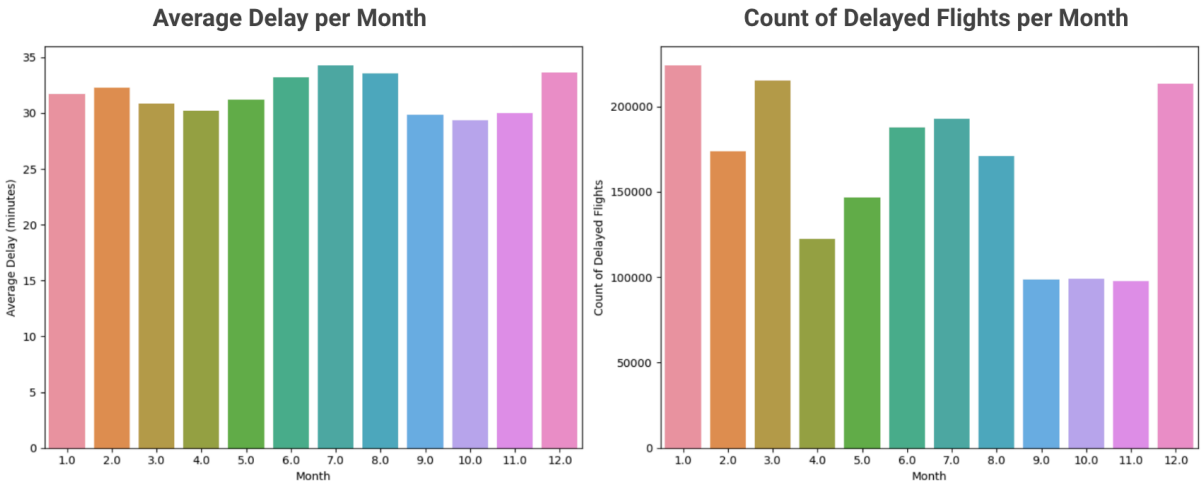
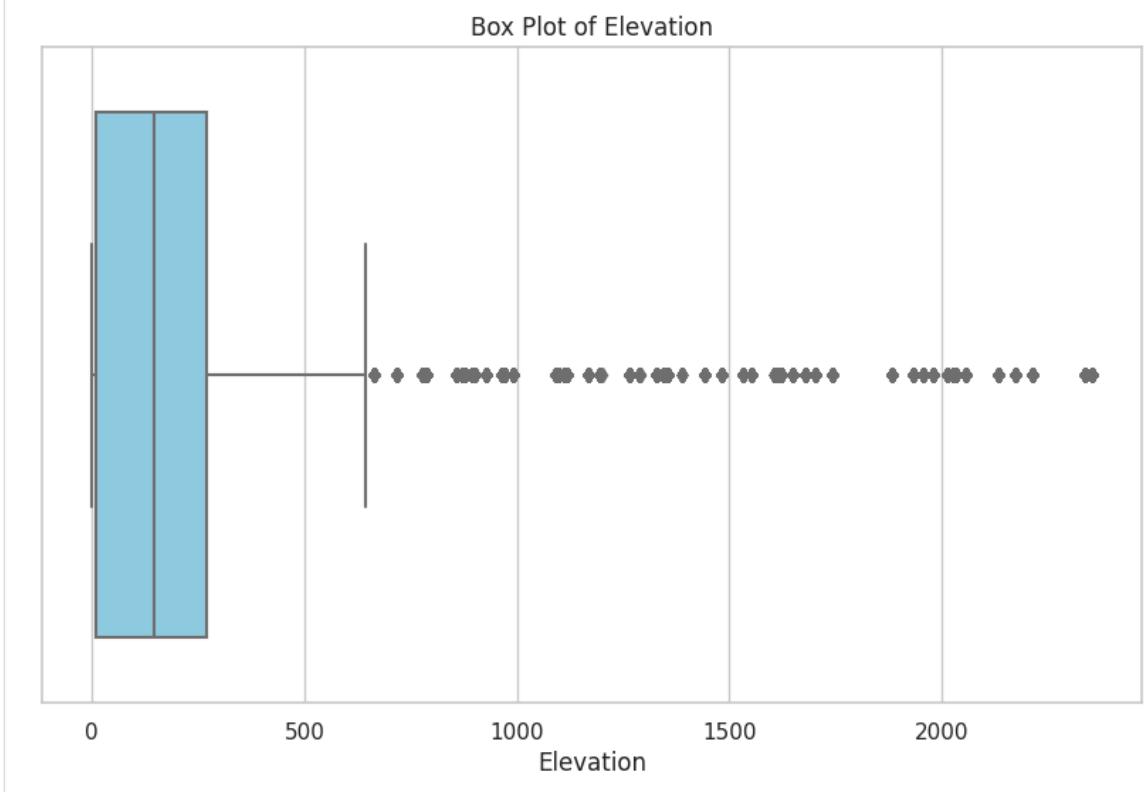


Figure 2e shows the patterns of flight delays across different months. The graph on the left illustrates the average delay per month, showcasing notable peaks in June through August and December, where the average delays exceeded 30 minutes. On the right, the graph portrays the count of delayed flights per month. Here,

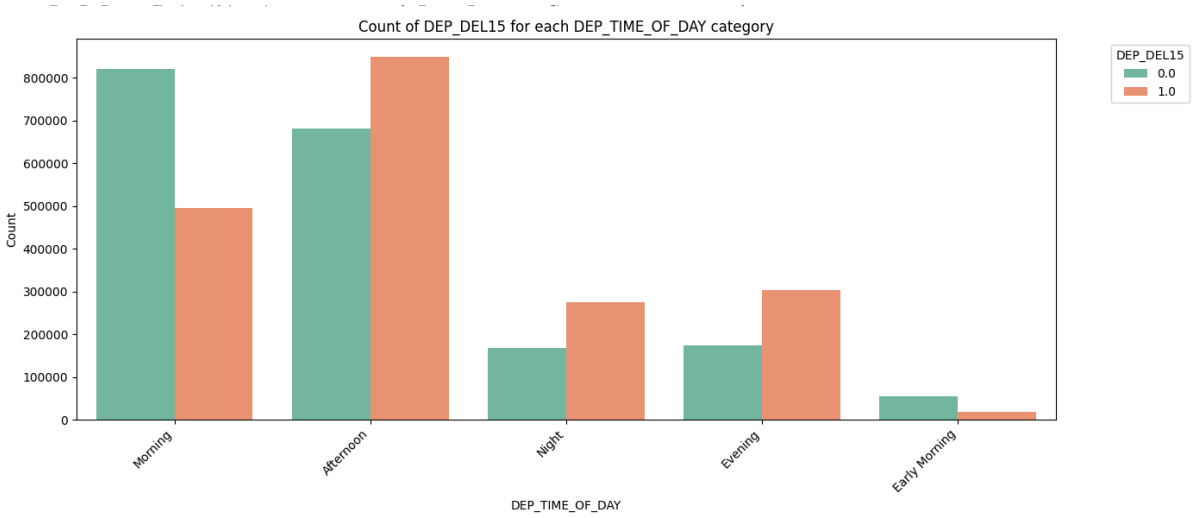
December, January, and March emerge as months with the highest frequencies of delays. Analyzing monthly trends allows us to capture seasonality and external factors that may influence flight delays.

Figure 2f. BoxPlot of Flight Elevation (3-Year)



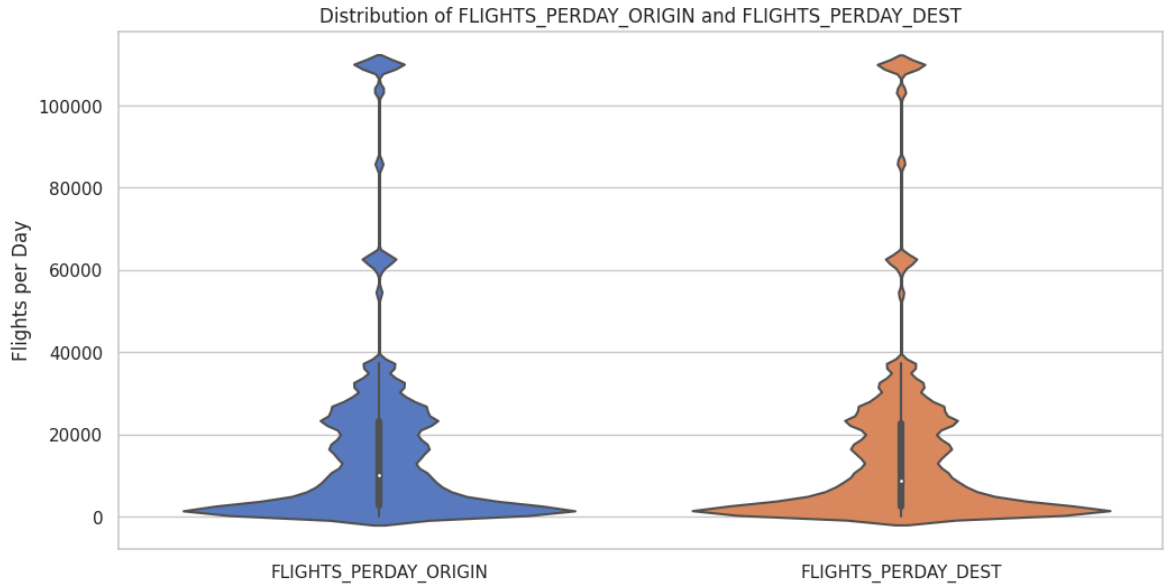
In Figure 2f, the distribution of flight elevation is observed. The data exhibits a dispersed distribution with noticeable variability. However,there are also numerous outliers. This prompts consideration for the utilization of robust machine learning models, such as Random Forests, which may generalize better in the presence of outliers.

Figure 2g. Count of Departure Delays vs Departure Time of Day (3-Year)



in Figure 2g, we explored the correlation between flight delays and the time of day. The notable reduction in delays for morning flights further underscores the impact of time of day on flight punctuality. Our model can be trained to recognize and weigh the impact of specific hours, enabling more precise predictions and proactive measures during high-risk periods.

Figure 2h. Distribution of FLIGHTS_PERDAY_ORIGIN and FLIGHTS_PERDAY_DEST (3-Year)



4. Data Leakage

Given the times series nature of this project, one of the most crucial considerations when developing our modeling pipeline is to ensure that we do not have data leakage. Data leakage is when future information or information that your model should not have access to in an actual real-time scenario, is accidentally used to train the model. A key hypothetical example of data leakage in our flight dataset would be including the both the scheduled departure time and actual departure time of a flight as features in a model to predict flight delays. Although we are not directly including flight delay information, we are indirectly infering if the flight was delayed based off of the discrepancy between the scheduled and actual departure time of that flight, and likely the model will learn to use those features to identify a delayed departure, even though it would not have access to that information in a real-time scenario.

Our modeling pipeline is specifically established to help prevent data leakage at multiple points. In data preprocessing, we immediately exclude actual departure and arrival times or arrival delay features from our analysis, because these features could contribute to data leakage. By removing any arrival-based time variables, we decreased our risk of data leakage if we wanted to also include departure-based time variables, like scheduled departure time. In creating new features, we made sure that the features would not inadvertently contain future information. For computing average carrier delay, we used the average values from the training dataset only. For deriving the previous aircraft's departure delay, we made sure that we could only impute the previous aircraft delay if we would have the actual departure time and delay information at least 2 hours prior to the current departure time of interest. When we assessed our final 20 features, we confirmed that no feature, inherent or derived, could be contributing to data leakage.

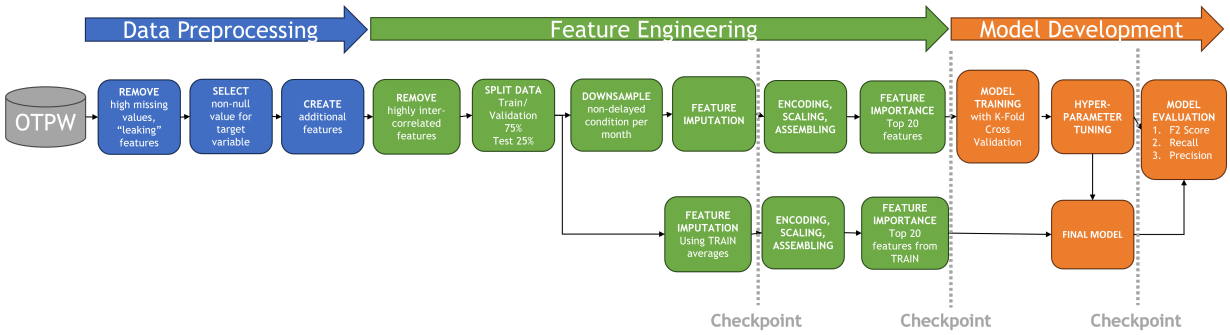
Another important aspect of data leakage is that our model and pipeline should never be able to operate on knowledge from the test dataset. This is a key cardinal sin of machine learning that we strived to avoid. When we performed downsampling, we only did so on the training dataset and left the blind validation and test datasets

alone with respect to class imbalance. For feature imputation, we made sure to use training data-derived averages to fill in missing values in the blind validation test dataset.

We also avoid data leakage and any cardinal sins of machine learning by splitting our dataset linearly in time for training, validation, and test, rather than a random 75/25 split that is commonly employed. This prevents future information from getting mixed in with training data. Similarly, we employ a cross-fold validation technique

5. Modeling Pipeline

Figure 3. Final Modeling Pipeline



From our exploratory data analysis, data engineering, and data leakage analysis, we developed a final modeling pipeline through which the 5 years OTPW dataset was fed through for model experimentation. This pipeline (Figure 3 above) is comprised of 3 main components: data preprocessing, feature engineering, and model development.

5.1 Data Preprocessing

In initial data preprocessing, we began by eliminating features with more than 50% missing values, based on our missing values analysis (Figure 1b). We excluded 110 features with more than 50% missing values. We also excluded 30 other variables that were duplicate information found in other columns and thus not additionally helpful in predicting our target variable, "DEP_DEL15" (binary variable of whether a flight delay was recorded). Because our target variable, DEP_DEL15, also had null values, potentially due to cancelled flights, we decided to not consider those flights as delays and remove any flights or rows where our target variable was null

(removing 0.4% of rows in our dataset). Based on our data leakage assessment, we also excluded from further consideration any features that could inadvertently introduce future information into our dataset (see Data Leakage section). We then created newly derived features, as described above in Data Engineering, based on our exploratory data analysis to be considered as final features for our models.

5.2 Feature Engineering

Once we created our additional features, we followed by addressing multicollinearity among continuous variables, removing one feature per each feature pair with a Pearson's correlation coefficient exceeding 0.8 (removing 16 features total). While some of our derived features were categorical in nature (hourly sky conditions, whether a date fell within a holiday window or not), other features were numerical (the number of flights per day from an origin or destination airport as well as the number of flights based on a given tail number). By re-calculating a Pearson's coefficient correlation analysis across all features including our derived features, we could better identify features that were highly correlated with each other and therefore contained potentially more redundant information for the model.

After determining highly correlated numercial features and excluding them from the downstream pipeline, we then split our dataset into training, validation, and testing. For our final 5 year models, our training and validation data consisted of all flights from 2015-2018 and our test dataset was on all flights from 2019. Within our training/validation dataset, we identified our training set for cross-fold validation to be 2015 through the third quarter of 2018. The last quarter of 2018 (October, November, and December) was held out as the blind validation set. We removed all the flights data for the first 5 days of the blind validation and test data sets to ensure no accidental crossover between training, validation, and test data. For all model development, we exclusively worked on the training and blind validation datasets. Once we identified our best model, only then did we assess it on the holdout test 2019 flight data.

For the training dataset only, we downsampled non-delayed flights to address the class imbalance (more flights are on time than delayed overall) on a per month basis. Distinct months were extracted from the "FL_DATE" column, iterating over each month and calculating the count of instances where departures were delayed for

each given month. For each month, the downsampling fraction is then determined by dividing the count of delayed instances by the total count of non-delayed instances. This approach effectively addressed class imbalance with monthly granularity and ensured a more balanced distribution of delayed and non-delayed instances in our training dataset, which we hoped would build a model that could better predict delayed flights. A summary of how the downsampling affected the size of our datasets are described below:

Dataset	Class	Original Number of Rows	Number of Rows Post-downsampling
5 years - training	Non-delayed	18,424,494	4,133,567
5 years - training	Delayed	4,131,041	4,131,041
5 years - blind validation	Non-delayed	1,414,350	NA
5 years - blind validation	Delayed	299,419	NA
5 years - test	Non-delayed	5,907,676	NA
5 years - test	Delayed	1,356,290	NA

After downsampling, we imputed any remaining features with null values. We used the back-filling technique with respect to time for the time-based continuous variables (mentioned in the Input Features table below). Any time-based variable with a missing value was back-filled using the most recent value available within 72 hours. If there were not values available within that time window, we filled the missing values with the average value of the variable from the training dataset. For the blind validation and test sets, we back-filled using the same 72 hour window but filled values with the average of the training set. We then calculated mean values for other continuous variables after grouping by relevant categories like carriers and airports (both origin and destination airports), because we noticed that origin and carrier affect departure delays from previous analyses. Mean values were again calculated using the training/validation set and the test set was imputed using the training averages.

Similar to our Phase II baseline report, we one-hot encoded our categorical variables, scaled our final numerical features using Standard Scaler, and assembled all features using VectorAssembler. Because one-hot encoding was a rate-limiting step in our pipeline and often caused our Spark clusters to crash, we checkpointed the data prior to one-hot encoding for development purposes.

A key change to our final modeling pipeline from our baseline model development in Phase II was to only select the top 20 most relevant features. We performed this additional engineering step because we noticed that our previous baseline models had long runtimes with hundreds of features due to many categorical variables with a lot of categories (49 variables were included in our original 3 year baseline models, which was more than 700 features after one-hot encoding). We also noted potential overfitting in our earlier models, and aimed to address this overfitting by reducing the number of features we included.

To identify the most important features, we used random forest algorithm on 3 years data to get feature importance scores (Gini coefficients) for each feature. Averaging scores from categorical variables, we could then rank which features were the most important in determining how the data was clustered (Figure 2a). From this analysis, we identified the top 20 features as:

1. Schedule departure time
2. Previous departure delay
3. Departure time of day (categorical)
4. Hourly precipitation
5. Flight per day at the origin
6. Average carrier delay
7. Precipitation Category (none, mild, heavy) *
8. Origin airport latitude
9. Hourly Altimeter Setting
10. Overcast clouds (indicator)
11. Hourly relative humidity
12. Flight distance
13. Hourly visibility
14. Hourly sea level pressure
15. Visibility category *
16. Hourly dew point temperature

17. Origin airport type (small, large)
18. Broken clouds (indicator)
19. Flight frequency
20. Airline carrier
21. Destination airport latitude
22. Cancelled flight (indicator)

*Note: We derived categorical weather features from the continuous weather variables. For precipitation and visibility, both the categorical and continuous feature appeared in the top 20 features of our random forest feature importance. Because these feature encode overlapping information, we decided to only retain the more important of the two features in our model, which was in both cases the continuous variables.

Based on this feature engineering, we selected the below final features to feed into our models. Among the 216 initial features, our team engineered 5 of the top 20 features selected, showcasing the effectiveness of our exploratory data analysis (EDA).

Feature (Column Name)	Data Type	Description
CRS_DEP_TIME	numeric	CRS Departure Time (local time: hhmm).
prev_DEP_DEL15	numeric	Previous flight's departure delay indicator (1=Yes, 0=No).
DEP_TIME_OF_DAY_onehot	binary	Categorized part of the day for departure time of day (morning, afternoon, night, evening, early morning).
HourlyPrecipitation_Imputed	numeric	Imputed hourly precipitation.
FLIGHTS_PERDAY_ORIGIN	numeric	Number of flights from specific origin each day.
AVERAGE_CARRIER_DELAY	numeric	Average delay caused by the carrier.

Feature (Column Name)	Data Type	Description
origin_airport_lat	numeric	Latitude of the origin airport.
HourlyAltimeterSetting_Imputed	numeric	Imputed hourly altimeter setting.
OVC	binary	Indicator for Overcast Clouds.
HourlyRelativeHumidity_Imputed	numeric	Imputed hourly relative humidity.
DISTANCE	numeric	Distance between origin and destination airports (miles).
HourlyVisibility_Imputed	numeric	Imputed hourly visibility.
HourlySeaLevelPressure_Imputed	numeric	Imputed hourly sea-level pressure.
HourlyDewPointTemperature_Imputed	numeric	Imputed hourly dew point temperature.
origin_type_onehot	string	Origin airport type (large/medium/small airport)
BKN	binary	Indicator for Broken Clouds
FLIGHTS_FREQ	numeric	Number of flights per a given tail number
OP_UNIQUE_CARRIER_onehot	string	One-hot encoded unique carrier code for the operating airline.
dest_airport_lat	numeric	Latitude of the destination airport.
CANCELLED	binary	Cancelled flight (indicator)

5.2.1 Input Feature Families

Family	Features	Count
One-Hot Encoded Categorical Features	DEP_TIME_OF_DAY_onehot, origin_type_onehot, OP_UNIQUE_CARRIER_onehot	3
Binary (Indicator) Features	prev_DEP_DEL15, OVC, BKN, CANCELLED	4
Numerical Features	CRS_DEP_TIME, HourlyPrecipitation_Imputed, FLIGHTS_PERDAY_ORIGIN, AVERAGE_CARRIER_DELAY, origin_airport_lat, HourlyAltimeterSetting_Imputed, HourlyRelativeHumidity_Imputed, DISTANCE, HourlyVisibility_Imputed, HourlySeaLevelPressure_Imputed, HourlyDewPointTemperature_Imputed, FLIGHTS_FREQ, dest_airport_lat	13

Total Features: 20 (43 model features accounting for one-hot encoded categories)

After this final feature selection, we checkpointed our datasets to more forward with model development.

Link to notebook (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/173910692692037/command/173910692692038 for Data pre-processing and Feature Engineering

5.3 Model Development

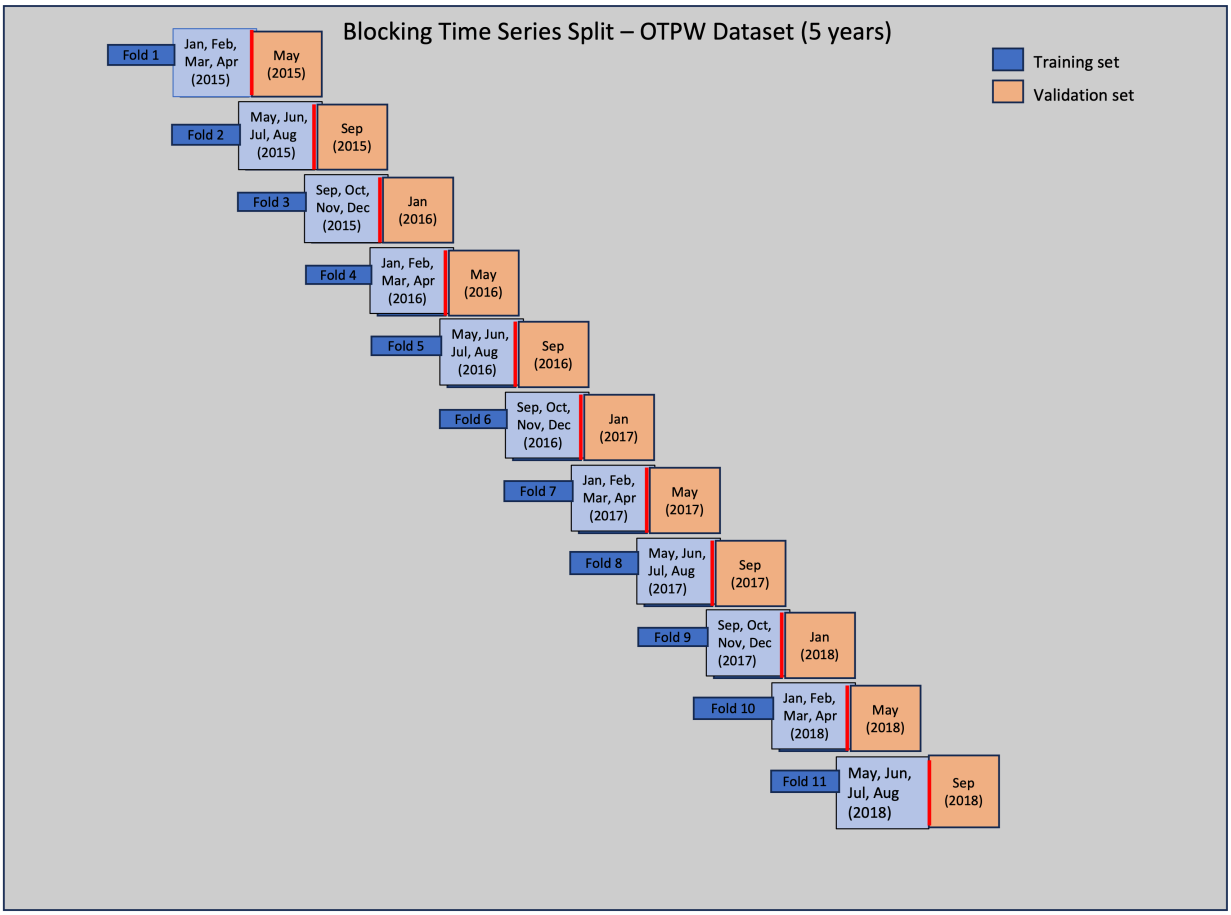
With our final 20 features, we moved into model development. For model development, we considered the following general models:

1. Logisitic Regression (LG)
2. Random Forest (RF)
3. Gradient-boosted decision tree (GBT)
4. Multi-layer perceptron (MLP) (1 or 2 hidden layers)
5. Naive Bayes (NB)
6. GRU recurrent neural network (RNN)

7. Ensemble model GBT + MLP

We emphasized training on models 1-4 and employed times-series block cross validation for training on our January 2015 - September 2018 dataset. We manually created folds (Figure 4), whereby each fold consisted of 4 months of training with the fifth month as validation. The second fold had overlapping training and validation, where the first month of the second fold was the validation month. In this way we created 11 folds for the 5 year training dataset.

Figure 4. Cross Validation Method:



We evaluated results primarily on the F2 score, which serves as a balance between precision and recall, with a heavier weightage on recall. We determined that both precision and recall were important, but that it is more important to prioritize false negatives, since flights that were a delay and not predicted as delay by our model were determined to potentially cause larger damage to flight carriers and their customers. We also recognize false positives could also pose some issues if a flight carrier or customer is notified of a flight delaye but actually there is no delay, and

thus decided on F2-score to evaluate our models. Because F2-Score is based on precision and recall, we also recorded those metrics when evaluating our models, but selected the best model to be the one with the highest F2-score.

5.3.1 Equations for Evaluation Metrics

F2-Score: $F_\beta = \frac{(1+\beta^2) \cdot \text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}}$ where beta = 2

Precision: $Precision = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$

Recall: $Recall = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$

We evaluated models with default hyperparameters or hyperparameters established from GridSearch in our 3 year baseline models. Additionally, we performed Random Search for hyperparameter tuning on the logistic regression, random forest, GBT, and MLP with 2 hidden layer models. We explored various combinations of hyperparameters, including max number of iterations, regularization parameter (closer to lasso), and the elastic net mixing parameter. For each set of hyperparameters, we trained the model on the train data from different blocked-time series folds, and then evalauting performance on the corresponding validation sets. Optimal hyperparameters and the associated performance metrics are determined.

5.4 Experimental Overview

For our final experiments on 5 years data, we conducted 11 main experiments, evaluating models on F2 score on the blind validation set. The best models were run on the test set for a final evaulation. In our Gap Analysis, we performed some additional experiments to try and identify how our models could be improved in the future. The list of experiments is noted below. All models were run using 1 driver with 28GB memory, 8 cores, with 1-10 workers, each 14GB memory and 4 cores on databricks runtime version 13.3 LTS ML (includes Apache Spark 3.4.1, Scala 2.12).

Experiment #	Model	Hyperparater Tuning	Loss Function	Evaluation Metrics	Runtime (minutes)
1	Logistic Regression	best parameters from baseline	cross-entropy	F2-Score, Precision, Recall	22.44
2	Random Forest	best parameters from baseline	gini	F2-Score, Precision, Recall	29.06
3	Gradient Boosted Decision Tree	no	gini	F2-Score, Precision, Recall	23.86
4	Naive Bayes	no	negative joint log probability	F2-Score, Precision, Recall	14.74
5	Multilayer Perceptron (1 hidden layer)	no	binary cross-entropy	F2-Score, Precision, Recall	9.40
6	Multilayer perceptron (2 hidden layers)	no	binary cross-entropy	F2-Score, Precision, Recall	15.83
7	GRU RNN	no	binary cross-entropy	F2-Score, Precision, Recall	NA (timed out/incomplete)

6. Neural Network (MLP)

To bolster our model performance compared to baseline with logistic regression and random forest, we specifically looked into more advanced models. In our recent analytical endeavor, we have developed and applied a Multilayer Perceptron (MLP) Neural Network model, specifically designed for predicting flight departure delays

(DEP_DEL15). The MLP model, a type of feedforward artificial neural network, is structured with multiple layers of nodes, encompassing an input layer, hidden layers, and an output layer.

6.1 Architecture of the MLP Model with One Hidden Layer without Grid Search (MLP-1)

Architecture: MLP-10-ReLU-2 Softmax

This MLP model comprises an input layer consisting of 10 nodes. This is then followed by a single hidden layer utilizing the ReLU (Rectified Linear Unit) activation function for non-linear processing. The model culminates with an output layer that contains 2 nodes employing a Softmax activation function, which is particularly effective for classifying the delayed and the non-delayed flights.

Features

Specific parameters were adjusted to optimize the model's performance. These include:

- Input Features Features: 43
- Target: DEP_DEL15
- Iterations: 20
- Block Size: 128
- Seed: 1234 (for reproducibility)
- Solver: 'l-bfgs' (Limited-memory Broyden–Fletcher–Goldfarb–Shanno" Algorithm. It is responsible for adjusting the weights of the network during training to minimize the loss function)
- Step Size: 0.03
- Tolerance: 1e-6

The model’s effectiveness was assessed based on F2 score. Further, other metrics like precision, recall and Area Under the Curve (AUC) were also computed.

6.2 Architecture of the MLP Model with Two Hidden Layers (MLP-2)

MLP-num_features-10-5-2 Softmax

This MLP model begins with an input layer that corresponds to the number of features in the training dataset. The architecture features two hidden layers; the first hidden layer contains 10 nodes, and the second hidden layer comprises 5 nodes, both employing the ReLU (Rectified Linear Unit) activation function for non-linear transformations. The network concludes with an output layer consisting of 2 nodes utilizing the Softmax activation function, adept at binary classification tasks such as distinguishing between delayed (DEP_DEL15 = 1) and non-delayed (DEP_DEL15 = 0) flights.

Features

The model's performance was optimized with the following specific parameters:

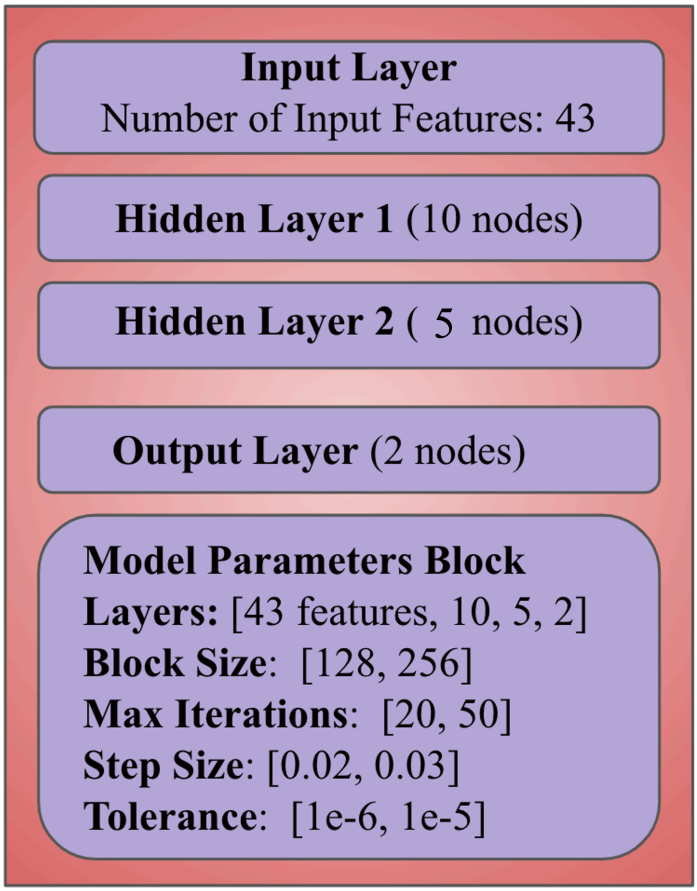
- Input Features: 43
- Target Variable: DEP_DEL15
- Iterations (Max Iterations): 20
- Block Size: 128
- Seed: 1234 (ensures reproducibility) The model’s effectiveness was gauged primarily on the F2 score, emphasizing recall. Complementary metrics, including precision, recall, and the Area Under the Receiver Operating Characteristic Curve (AUC), were also calculated to provide a thorough evaluation of the model's predictive accuracy.

6.3 Architecture of the MLP Model with Two Hidden Layers with Grid Search (MLP-2)

Architecture: MLP-43-10-5-2 Softmax

This MLP model initiates with an input layer comprising 43 nodes, corresponding to the number of features in the training dataset. It features two hidden layers; the first hidden layer has 10 nodes, while the second hidden layer contains 5 nodes. Both hidden layers employ the ReLU (Rectified Linear Unit) activation function for non-linear transformations. The architecture culminates with an output layer of 2 nodes using the Softmax activation function, ideal for binary classification tasks like distinguishing between delayed (DEP_DEL15 = 1) and non-delayed (DEP_DEL15 = 0) flights.

Figure 5. Optimized MLP Architecture with Grid Search Hyperparameter Tuning



Features and Hyper-parameter Tuning

The model's performance was evaluated with default parameters in Experiment 6, and further optimized through hyper-parameter tuning (Experiment 11).

The parameters explored included:

- Block Size: 128, 256 - This refers to the batch size. It determines the number of samples that will be passed through the network before the model updates its internal parameters (weights and biases)
- Maximum Iterations: 20, 50 - This is the maximum number of passes over the entire training dataset. Each pass over the entire dataset is called an epoch.
- Step Size: 0.02, 0.03 - Also known as the learning rate, it determines the size of the steps the model takes during optimization.
- Tolerance: 1e-6, 1e-5 - Tolerance is used as a stopping criterion to halt the training process. It sets the threshold for the improvement in the model's performance (like reducing the loss) from one iteration to the next.

The best hyper-parameters identified through grid search were:

- Block Size: 256
- Maximum Iterations: 20
- Step Size: 0.03
- Tolerance: 1e-05

These parameters were used to achieve optimal performance. The model's effectiveness was primarily gauged based on the F2 score, placing a higher emphasis on recall. Additional metrics computed for a comprehensive assessment included precision, recall, and the Area Under the Receiver Operating Characteristic

7. Additional Neural Network Training

In addition to a multilayer perceptron, we also attempted to use Horovod to implement **Gated Recurrent Neural Network**. Horovod is a distributed training framework that facilitates efficient and easy scaling of deep learning models across multiple nodes in a cluster. It integrates with popular deep learning frameworks like TensorFlow, Keras, and PyTorch, and it is designed to maintain the native feel of these frameworks while enabling distributed training.

7.1 Advantages of Using Horovod:

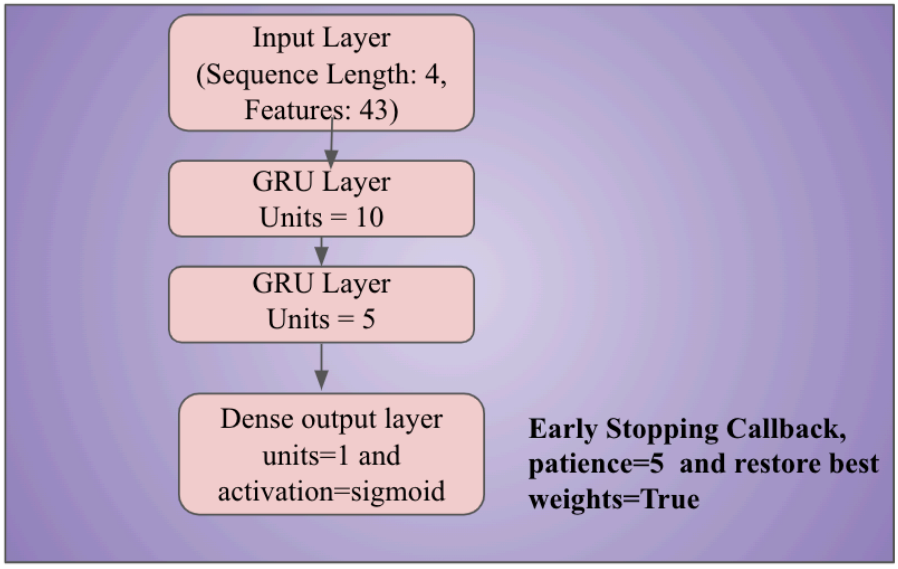
- Scalability: Horovod's design allows for efficient scaling from a single GPU to multiple nodes, harnessing the full potential of a cluster rather than being limited to a single node's resources.
- Resource Utilization: By distributing the computation across multiple nodes, Horovod ensures that each node's computational power, including all GPUs, is effectively utilized, leading to faster training times.

7.2 Model Architecture - GRU-RNN-10-5-1 Sigmoid:

The model is a Gated Recurrent Unit (GRU) based Recurrent Neural Network (RNN), well-suited for sequence data. Each input sequence to the model comprises 4 time steps, representing the temporal dimension of the data. At each time step, the input includes 43 distinct features, that we obtained through feature engineering from random forest. Key features like weather conditions, carrier delays, flight frequency, previous departure delays, airport characteristics were leveraged.

This architecture uses a Sequential model from Keras. The first GRU layer consist of 10 neurons. There is also a second GRU layer with 5 units. The output from the first GRU layer (which is a sequence) is fed into this layer. This layer further processes the data and condenses the information into a 5-dimensional output vector representing the entire input sequence. A dense layer which is a fully connected layer that outputs the final prediction is also added.The activation function used here is a sigmoid. For binary classification tasks, sigmoid is appropriate as it squashes the output between 0 and 1, which can be interpreted as a probability. An Earlystopping is also added. This callback monitors the validation loss (val_loss) during training and will stop the training process if the validation loss doesn't improve for 5 epochs (patience=5). It also restores the best weights achieved during training once it stops, ensuring the model retains the best learned weights even if the performance deteriorated in the last few epochs.

Figure 6. GRU RNN Architecture with cross fold validation



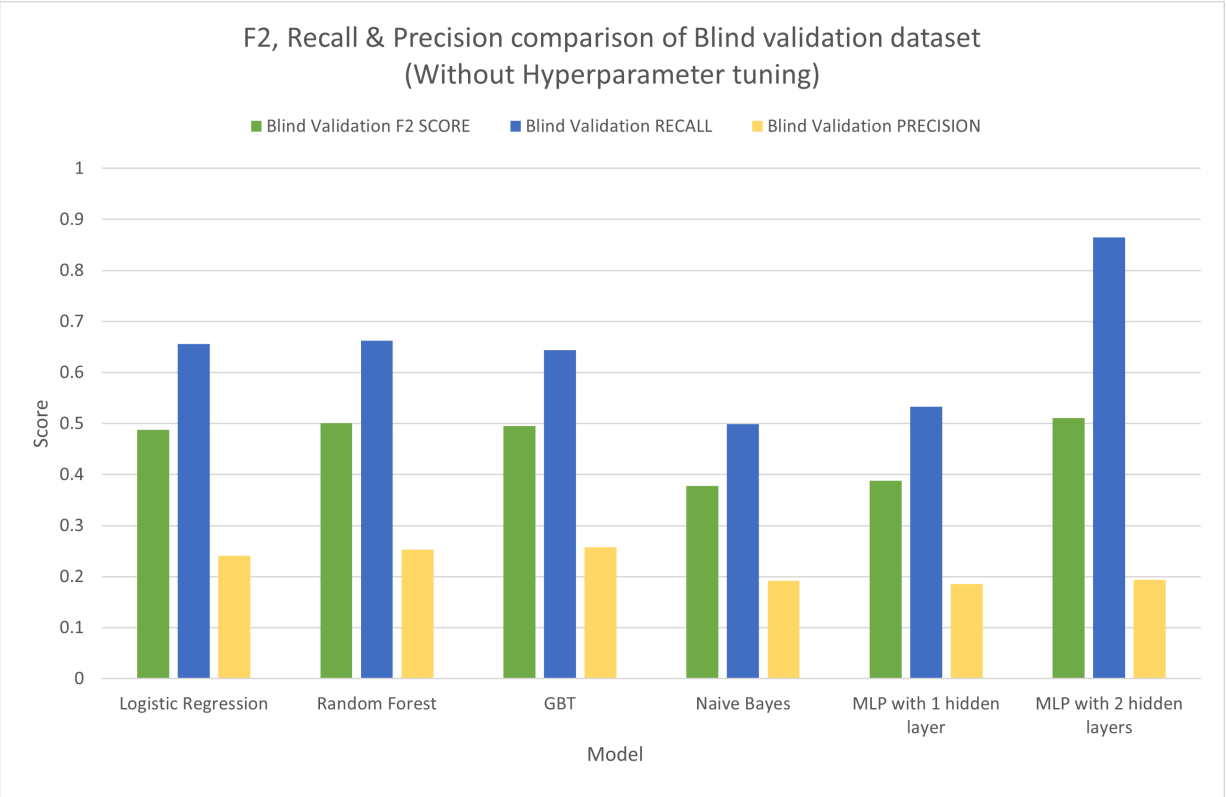
8. Results

Using all of these models, we performed cross-fold validation and full train and blind validation evaluation of each of the models (Experiment 1-11). The results of experiments 1-7 are described below.

8.1 Results on 5 year data without Hyperparameter tuning

Model	CV Avg Train F2 SCORE	CV Avg Train RECALL	CV Avg Train PRECISION	CV Best Hyper Param Training Time (min)	CV Avg Validation F2 SCORE	CV Avg Validation RECALL	CV Avg Validat PRECI
1. Logistic Regression	**0.6360**	0.6411	0.6198	22.44	**0.6178**	0.62	0.615
2. Random Forest	**0.6308**	0.6264	0.6575	29.06	**0.6171**	0.6134	0.641
3. GBT	**0.6111**	0.6008	0.6615	23.86	**0.5969**	0.5869	0.648
4. Naive Bayes	**0.5199**	0.5245	0.5277	14.74	**0.5110**	0.5155	0.518
5. MLP with 1 hidden layer	**0.5089**	0.5301	0.5186	9.40	**0.5068**	0.5289	0.515
6. MLP with 2 hidden layers	**0.6225**	0.6549	0.5309	15.83	**0.6204**	0.6540	0.528
7. GRU RNN*	0.7138	0.7960	0.5319	300	0.6905	0.7845	0.487

Figure 7. Comparative Model Performance (F2, Recall, and Precision) for Experiments 1-7



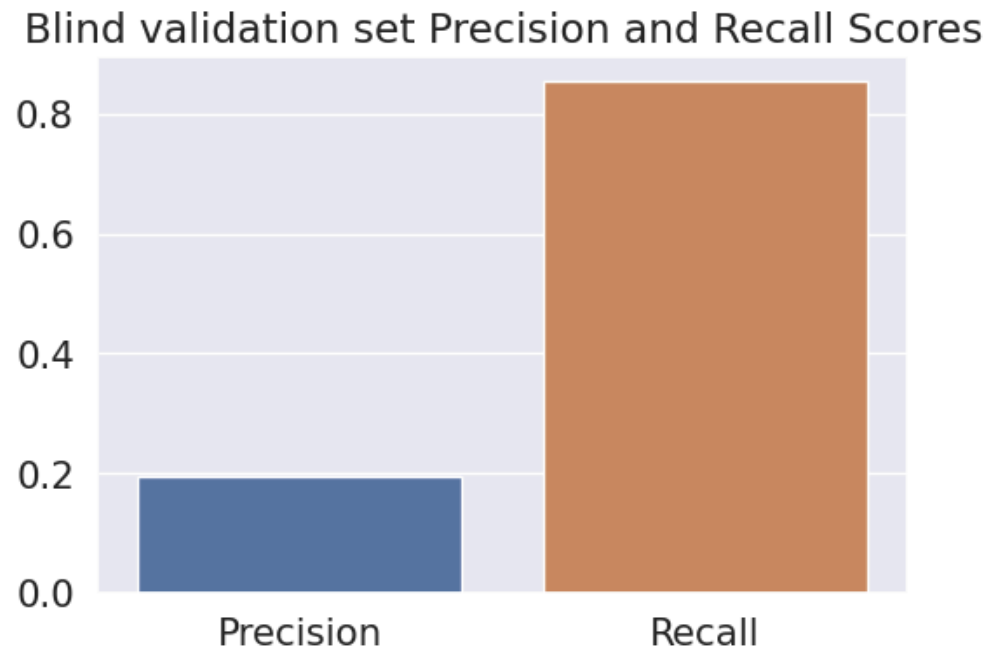
The below table shows parameters for the models in Experiments 1-7.

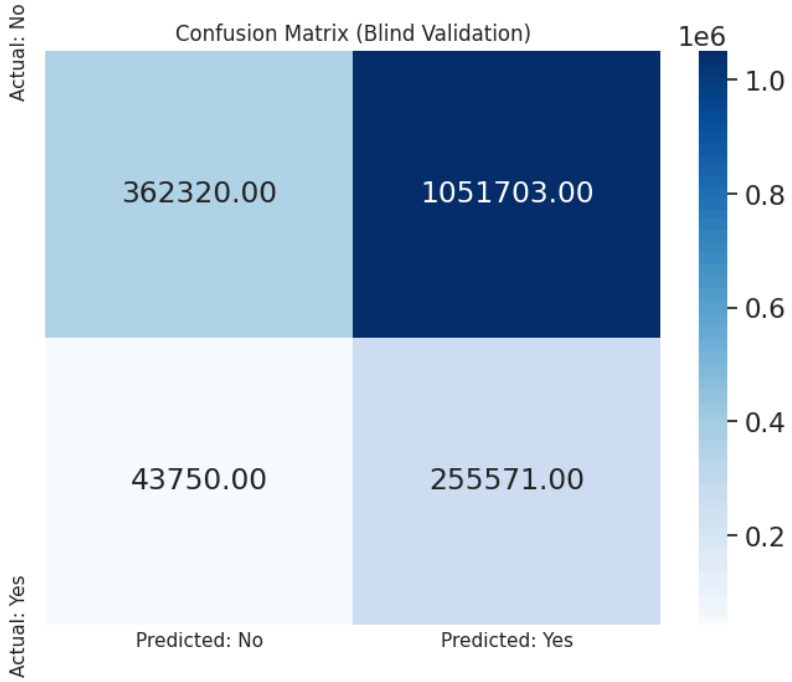
Model	Parameters
Logistic Regression	'elasticNetParam': 0.7, 'threshold': 0.5, 'regParam': 0.01, 'maxIter': 10
Naive Bayes	'smoothing': 1.0, 'modelType': "multinomial", 'thresholds': [0.5, 0.5]
Random Forest	'maxDepth': 10, 'numTrees': 20
GBT	'maxDepth': 5, 'maxIter': 10, 'maxBins': 32
MLP with 2 hidden layers	'blockSize': 128, layers=[2 hidden layers], 'maxIter': 20
MLP with 1 hidden layer	'blockSize': 256, layers=[1 hidden layer], 'maxIter': 20

Model	Parameters
GRU RNN	'units_GRU1': 10, 'return_sequences': True, 'units_GRU2': 5, 'units_Dense': output_units (1), activation_Dense: 'sigmoid',early_stopping_monitor: 'val_loss', early_stopping_patience: 1, restore_best_weights: True

For the first set of experiments, we trained 7 different models using cross fold validation on our 5 years OTPW training dataset (results table and Figure 7). Average cross-fold validation revealed that models performed similarly from training folds to validation folds, with at most a 0.02 decrease in F2 score from the training to the validation folds. However, the F2 score for blind validation drops considerably compared to the scores see on the full training dataset or the cross-fold training/validation (Figure 7-8).

Figure 8. MLP Precision/Recall and Confusion Matrix on Blind Validation

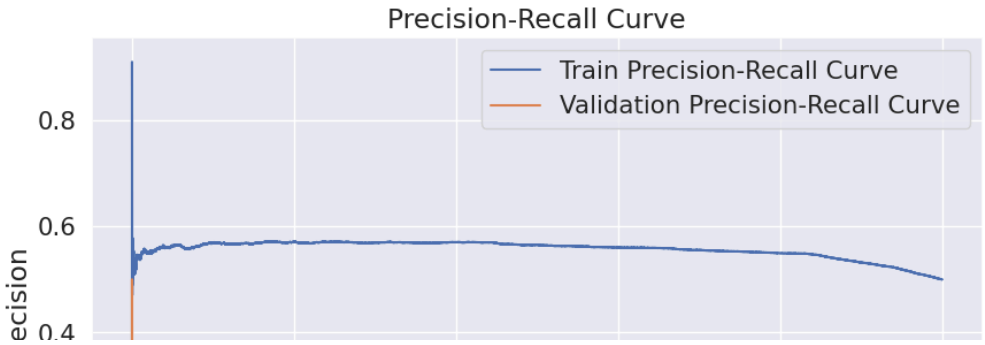




This drop in F2 score comes from the notably decreased precision for each model on the blind validation, compared to recall. Based on the blind validation assessment, the Naive Bayes (Experiment 4) and MLP with 1 hidden layer (Experiment 5) performed the worst, with 0.3780 and 0.3877 F2 scores respectively. Both of these models also performed more poorly in cross-fold training compared to the other experiments. The two best performing models was the MLP with 2 hidden layers (Experiment 6) and random forest classifier (Experiment 7), with a blind

validation F2 score of 0.5102 and 0.5010 respecitvely. For the MLP model with 2 hidden layers, blind validation recall is the highest at 0.8538 and interestingly the random forest classifier not only has the second highest recall (0.6628), but also has one of the highest precision (0.2534) in the blind validation set, along with gradient-boosted decision tree. The MLP model was also almost twice as fast as the random forest model, training in 15.83 minutes compared to 29.06 minutes. For the blind validation of the MLP model, the confusion matrix shows the model majority labelling flights as delayed, and the precision-recall curves show consistent precision on both the training and blind validation data (Figure 9).

Figure 9. MLP PR and ROC Curves on Blind Validation



8.2 Results on 5 year data with Hyperparameter Tuning (Random Grid Search)

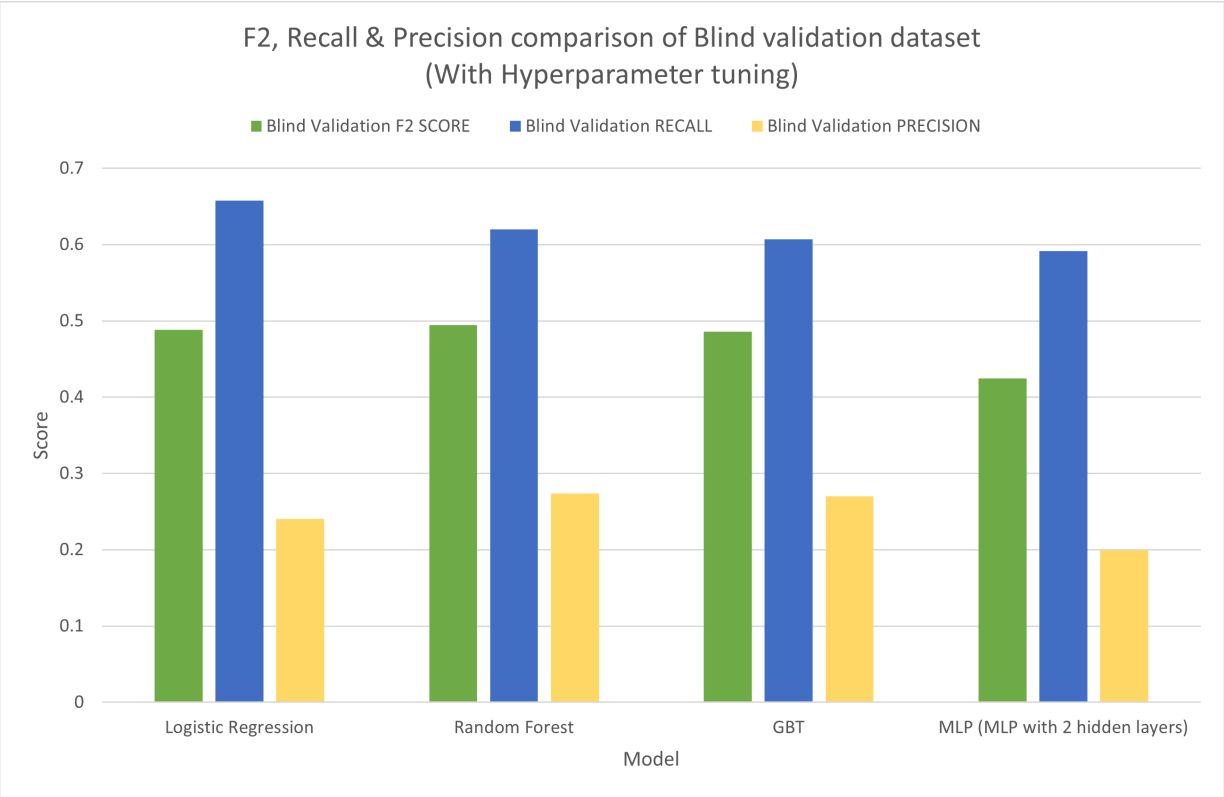
Based on our initial key experiments (1-7), we decided to attempt hyperparameter tuning for the top 4 models, logisitic regression, random forest, GBT, and MLP with 2 hidden layers, considering their similar performances. We conducted experiments 8-11 using a random grid search to conduct hyperparamter tuning. For each model, we specified the space of parameters indicated in the table below and randomly selected 5 combinations of those hyperparameters to test during cross-fold validation. Using the best parameters (also indicated in the table below), we continued with full training and blind validation comparisons.

Below table shows the best hyper parameters that came out of grid search CV evaluation

Model	Hyperparameter Space Assessed	Best Hyper Parameters
Logistic Regression	"regParam": [0.1, 0.01], "maxIter": [5, 10], "elasticNetParam": [0.5, 0.7], "threshold": [0.4, 0.5]	'elasticNetParam': 0.7, 'threshold': 0.5, 'regParam': 0.01, 'maxIter': 5
Random Forest	"maxDepth": [10, 15, 20], "numTrees": [10, 20]	'maxDepth': 15, 'numTrees': 10
GBT	"maxDepth": [5, 10], "maxIter": [5, 10], "maxBins": [32, 64]	'maxDepth': 5, 'maxIter': 10, 'maxBins': 32
MLP	"blockSize": [128, 256], "maxIter": [20, 50], "stepSize": [0.02, 0.03], "tol": [1e-6, 1e-5]	'blockSize': 256, 'maxIter': 20, 'stepSize': 0.03, 'tol': 1e-05

Model	CV Avg Train F2 SCORE	CV Avg Train RECALL	CV Avg Train PRECISION	CV Best Hyper Param Training Time (min)	CV Avg Validation F2 SCORE	CV Avg Validation RECALL	CV Avg Validat PRECIS
8. Logistic Regression	**0.6313**	0.6356	0.6195	15.84	**0.6129**	0.6142	0.614
9. Random Forest	**0.6557**	0.6470	0.6968	29.06	**0.6066**	0.5975	0.651
10. GBT	**0.6100**	0.5993	0.6621	36.44	**0.5971**	0.5871	0.648
11. MLP with 2 hidden layers	**0.5066**	0.5291	0.5301	8.13	**0.5037**	0.5265	0.525

Figure 10. Comparative Model Performance (F2, Recall, and Precision) for Experiments 8-11



Based on the best hyperparameters selected from our random grid search method, we found that, similar to without hyperparamter tuning, the logisitic regression, random forest, and GBT models continued to perform very similarly (Figure 10), with blind validation F2 scores of 0.4883, 0.4947, and 0.4856 respectively. We also see the same trend across all models where average cross-fold training and validation scores perform similarly, but we see a decrease in performance in F2 with the blind validation. Interestingly, full training evaluation scores improved for the MLP model by about 0.1 from cross-fold validation (0.5037) to full training data (0.6044). However, in blind validation, MLP with 2 hidden layers and hyperparameter tuning ended up performing the worst from experiments 8-11, with an F2 score of 0.4248.

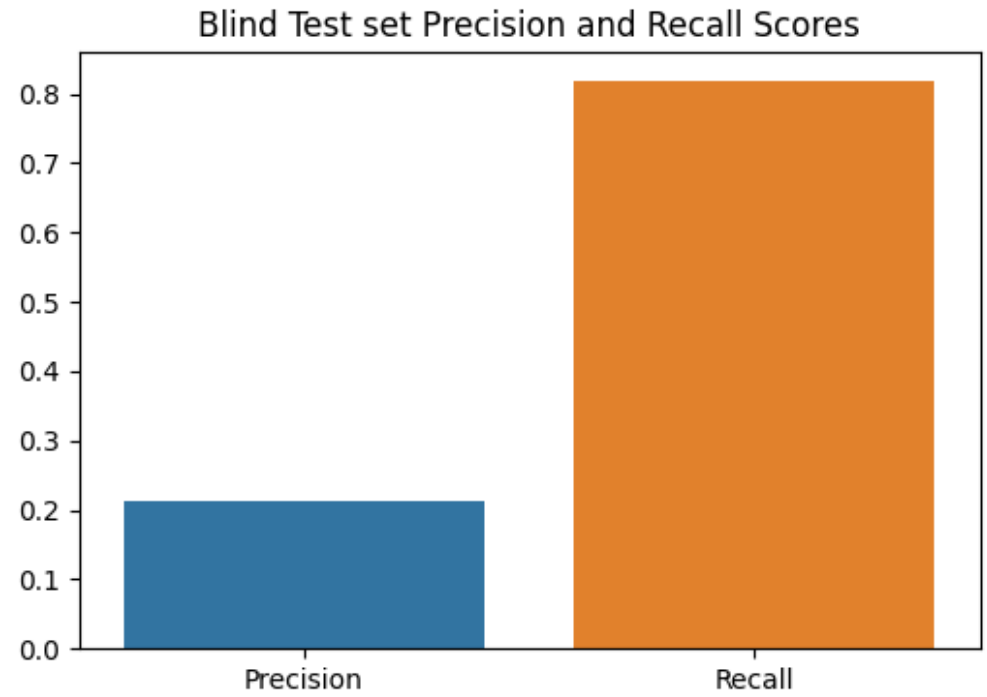
One important feature to note is that while these models underwent hyperparameter tuning, because we performed a random search, we did not necessarily get a direct comparison of the hyperparameters in Experiments 8-11 compared with the respectively non-tuned models in Experiments 1,2,3 and 6. In particular, our top performing model by blind validation score is still the MLP with 2 hidden layers model that was not hyperparamter tuned. When diving further, we noted that the

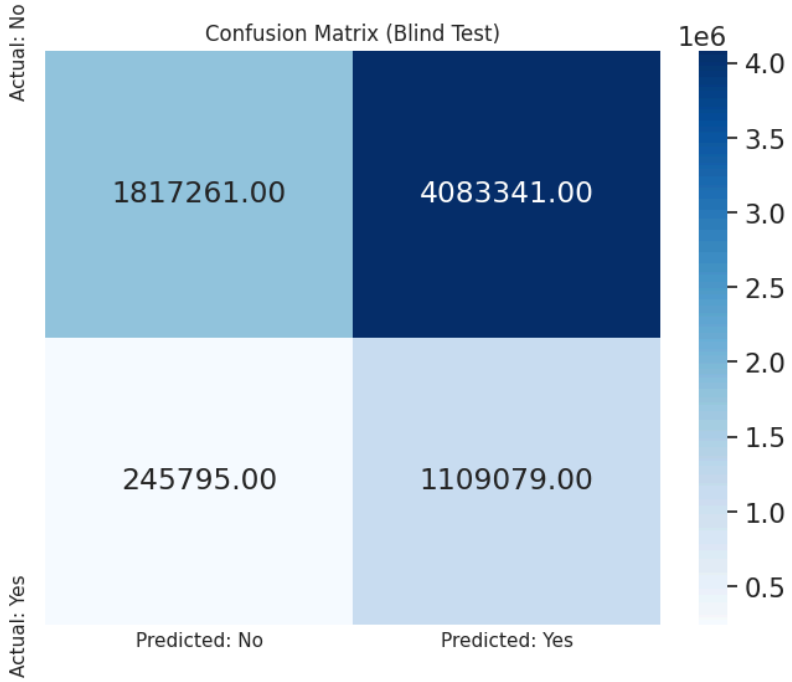
8.3 Blind Test Evaluation Results

For testing our blind test dataset, we took the MLP model with 2 hidden layers since it had good F2 score (our evaluation metric) when compared to other models.

Model	Parameters	F2 Score	Recall	Precision	
MLP with 2 hidden layers	'blockSize': 128, layers=[2 hidden layers], 'maxIter': 20	**0.5226**	0.8186	0.2136	Link to notebook (https://adb-4 o=4248444930383559#noteb

Figure 11. MLP Blind Test Precision, Recall Scores and Confusion Matrix

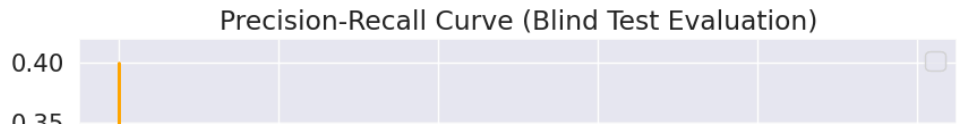




Based on our final test evaluation, we found that our MLP model with 2 hidden layers achieved an F2 score of 0.5226, with recall of 0.8186, and precision of 0.2136. This gave a similar performance of the model to the blind validation scores, where the model had an F2 score of 0.5102, recall of 0.8538, and precision of 0.1955, and still showed decreased performance compared to the initial cross-fold and full training metrics. The confusion matrix (Figure 11b) reveals that our model misclassifies a

majority of flights as delays when they are not delayed, which helps to explain the low precision score. This is also reflected by the precision-recall curve below (Figure 12a), where the low precision curve is consistent across recall values.

Figure 12. MLP Blind Test Precision-Recall and ROC Curves



9. Discussion

From our extensive experimentation, we found that our overall best model was the multi-layer perceptron with 2 hidden layers (43-10-5-2-Softmax), with a block size of 128 and 20 maximum iterations. This model had a final F2 score on the blind test set of 0.5226, recall of 0.8186, and precision of 0.1939. This model performed similar to the blind validation, but consistently worse than either the cross-fold training/validation scores (F2 scores of 0.6225 and 0.6204 respectively) or the full training score (F2 score of 0.7568). This lower score in validation and testing from training still indicates overfitting. This was an issue that our team struggled with and actively worked to avoid by reducing the number of features we included in our models and conducting Pearson's correlation analysis to remove redundant features. Additionally, with our best MLP model, we saw a consitently low precision across recall values (Figure 10a), which is indicative that our model consistently favors labeling delayed flights over non-delayed ones. While unideal, this can ensure that recall is kept high, and that the more costly mistakes of thinking a flight is not delayed when it is, is avoided.

Another interesting finding in our experimentation was the initial promise of the GRU-RNN. Ultimately, do to cluster stability issues we were not able to complete the model evaluation, however initial cross-fold metrics showed promissing results, outperforming our other models across all metrics. While we chose to move forward without this model, our team is committed to delving deeper into these stability issues. As a part of our future work, we aim to optimize the training efficiency of the GRU RNN model and enhance cluster stability to facilitate uninterrupted model

evaluation. Further investigations will focus on refining our training strategies and

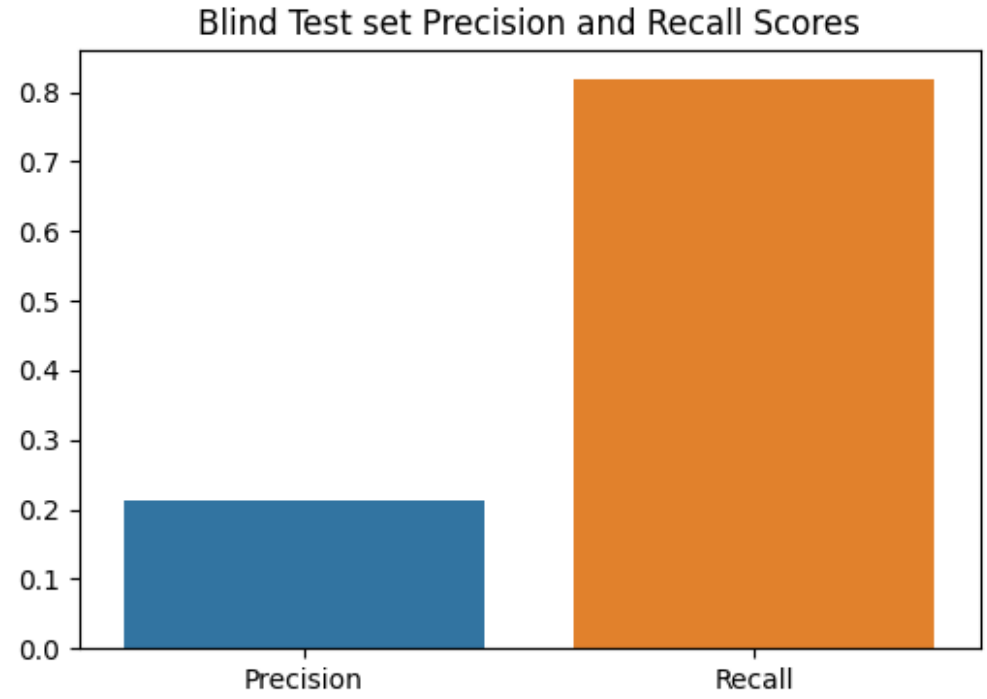
10. Gap Analysis

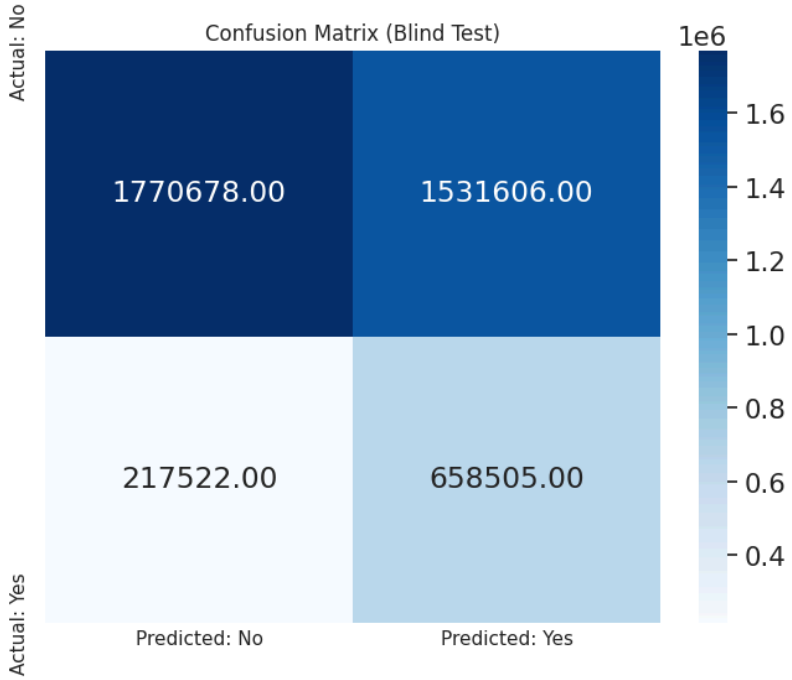
After assessing our best model on the blind test set, we decided to further conduct a deeper gap analysis to identify why the model might be underperforming. Our most obvious struggle was with our model's apparent overfitting. One observation to overfitting was that our individual folds in the cross-fold training process actually had similar training F2 scores and validation F2 scores. Because for the final model, we retrain on the full training data set, we had reason to believe that perhaps training our models only on the last fold of our cross-fold validation might produce more robust results in the blind validation testing. We evaluated just fold 11 (2018, May – September) and folds 10-11 (2018 January – September alone) on the blind validation dataset. Regardless of how we subsetted our training data, we continued to observe to obvious drop in precision, with almost identical precision-recall curves to that in Figures 9a or 12a.

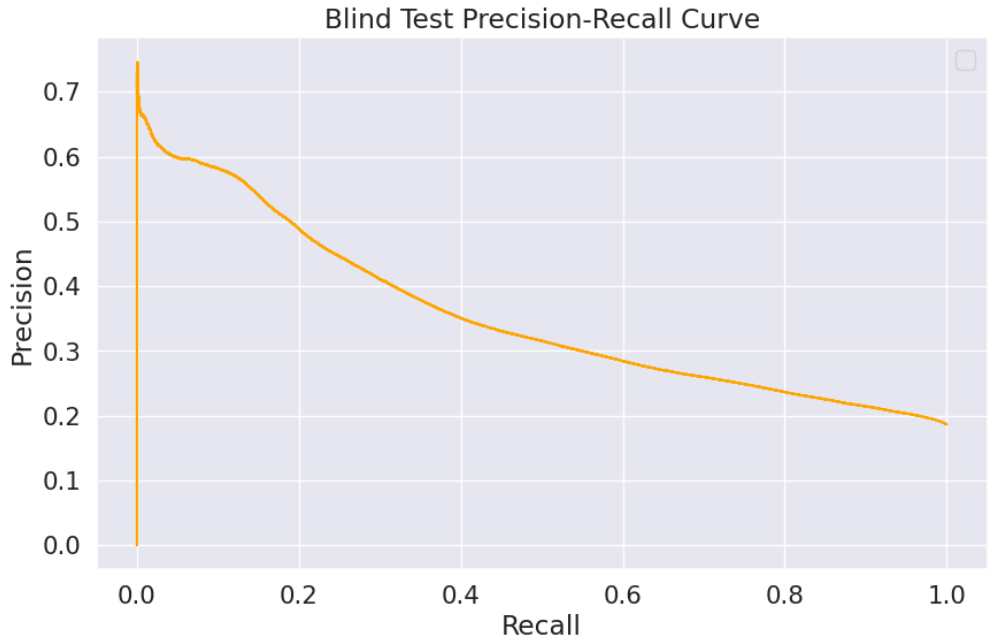
Link (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1012234209194311/command/10122342092010
to the individual fold assessment notebook

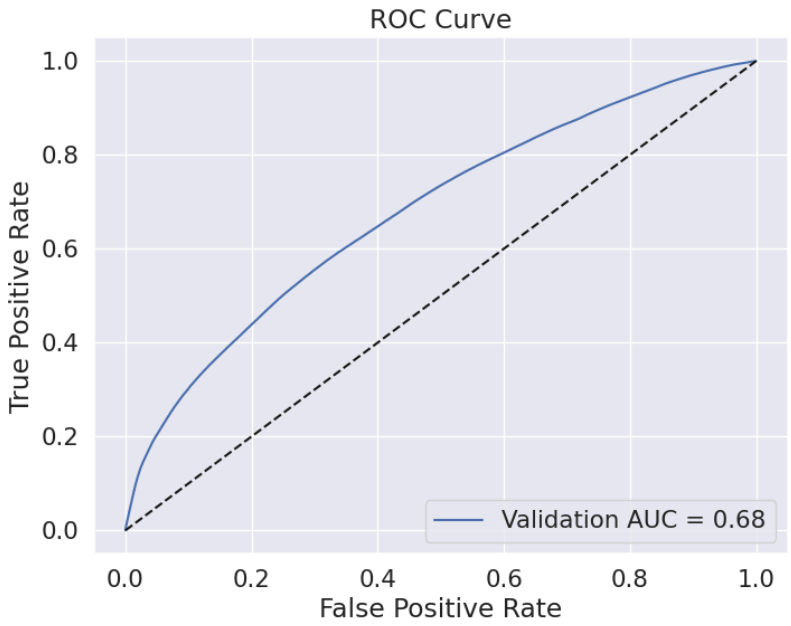
One other key observation we found was that while the MLP model had the top performing F2 score, the tree-based classifiers (random forest or gradient-boosted decision tree) had the highest precision values. Recognizing the low precision as a continued gap in our model, we therefore trained an ensemble model of MLP + GBT and evaluated it on the blind test data to see how it performed compared to our MLP model. As shown in Figure 13c, the precision-recall curve is notably displaying a more obvious precision/recall tradeoff, where we achieve lower precision as recall increases. This coupled with the confusion matrix (Figure 13b) demonstrates that this ensemble approach successfully identifies most flights as non-delayed, which is more reflective of our expected class imbalance. While this model was constructed for our gap analysis and would need signficant refining and further evaluation, these striking different in PR curve and confusion matrix hints that an ensemble approach might boost model performance.

Figure 13. GBT-MLP Ensemble Model PR plot, Confusion Matrix, PR curve and ROC curve









Link (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/173910692689969/command/173910692690859 to the GBT & MLP Ensemble notebook

Outside of additional model evaluation, we also looked more upstream at our modeling pipeline to try and identify what gaps in our pipeline may have contributed to our model's performance. Because we noted a drop in precision with all unseen data, we focused on how our pipeline treats training data differently from blind validation or test data. One of these junctures is downsampling. While

downsampling is imperative given our datasets class imbalance, our approach was to randomly downsample by month. Perhaps a more stratified downsampling approach would better retain the distribution of flights across various key features that would enable the training data to better represent the validation and test data sets.

Furthermore, while we worked diligently to address overfitting when designing our features, we also considered how our features may be another potential gap in our model. To help address the overfitting that we still see in our model, iteratively assessing model impact on additionally dropping individual features, or conducting dimensionality reduction to further reduce our features may have helped to decrease overfitting of the model. Additionally, creating and relying more heavily on derived

10.1 Enhanced Airline Dataset with 2020 Additions

To aid in our gap analysis and the development of future version of this model, our team began to successfully integrate a supplementary dataset into our main airline data repository, which encompasses records spanning over five years. The additional data, meticulously collected, enhances our database with up-to-date flight information, expanding our analytical capacity. Key attributes such as flight date (FL_DATE), airline identifier (AIRLINE_ID), aircraft tail number (TAIL_NUM), flight number (FLIGHT_NUM), sequence IDs for origin and destination airports (ORIGIN_SEQ_ID and DEST_SEQ_ID), airport codes (ORIGIN_AIRPORT and DEST_AIRPORT), departure and arrival times (DEP_TIME and ARR_TIME), and delays (DEP_DELAY, ARR_DELAY, WEATHER_DELAY) have been harmonized across both datasets, as showcased by the average departure and arrival delay trends in Figure 14.

In preparation for the union operation, we curated a subset of the main five-year dataset, selecting only the columns that align with the additional dataset's schema. This streamlined integration facilitated a seamless amalgamation of the two data sources, maintaining consistency and integrity in our consolidated dataset. Post-union, the merged data now presents a comprehensive view of airline operations from 2018 through 2020, allowing us to generate insightful trend analyses. This expanded timeframe is instrumental in our ongoing efforts to monitor and understand the dynamics of flight delays and operational patterns. Following the

successful union, we have written the amalgamated DataFrame into Azure Blob Storage, ensuring that our consolidated data is stored securely and efficiently in the cloud. This allows for scalable storage and accessibility for further analysis and model training.

Our collaborative team efforts are directed towards aggregating further data for the subsequent years of 2020, 2021, and 2022. The objective is to enrich our dataset continuously, thereby empowering our predictive models and decision-making processes with the most current and expansive data available. Through this rigorous data augmentation, we are poised to elevate the analytical capabilities to new heights, ensuring that our strategies are data-driven and reflective of the latest industry trends.

Figure 14. Average Departure and Arrival Delay Trends: (2018 - 2020)

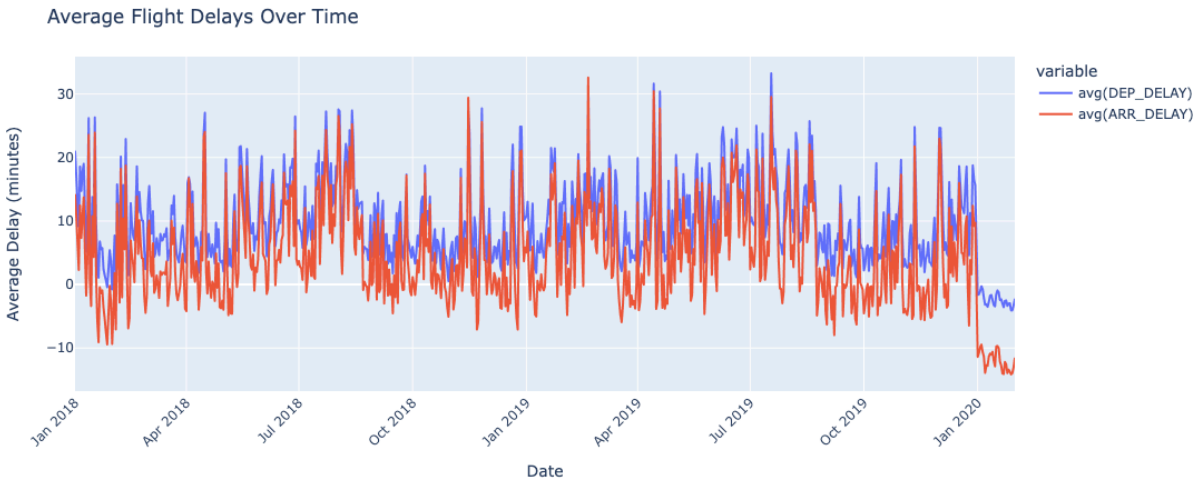
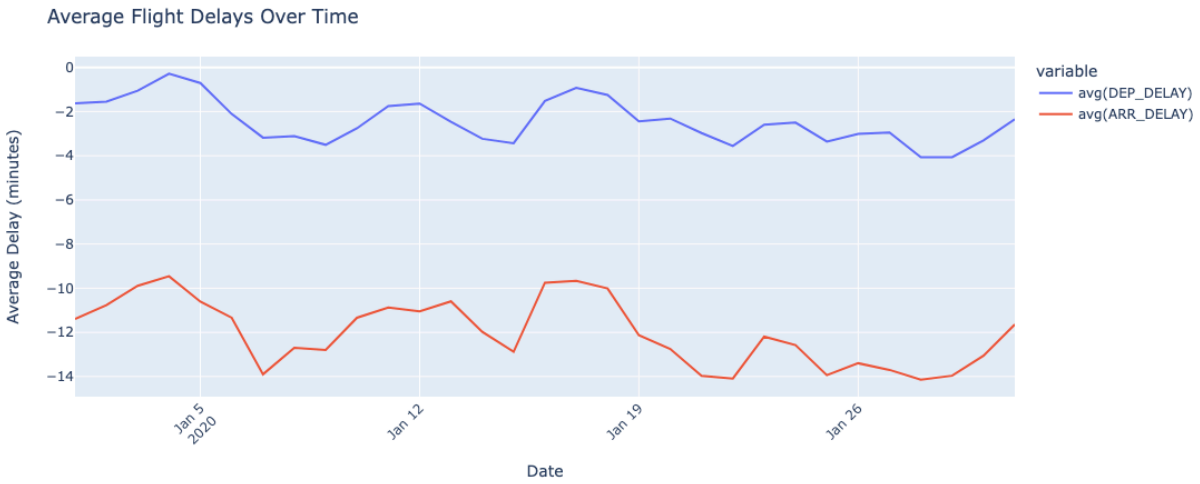


Figure 15. 2020 Average Departure and Arrival Delay Trends (2020)



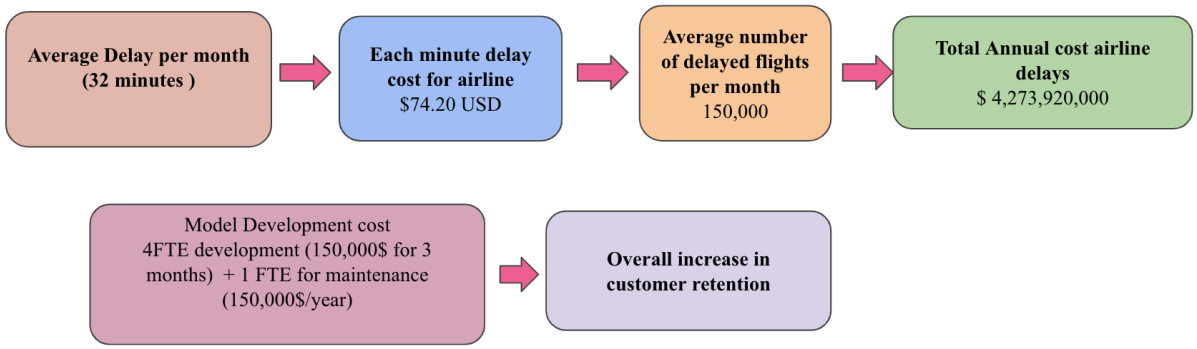
Link to notebook:

https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/173910692691634/command/173910692691652 (https://adb-4248444930383559.19.azuredatabricks.net/?

11. Cost Analysis

The financial impact of flight delays on airlines and passengers are profound. With an average hourly value of $47 assigned to a passenger's time, the cumulative cost resulting from delays escalates rapidly. Our analysis reveals that for every minute a flight is delayed, the airline bears an expense of $74.20. This cost, when extrapolated over the average duration of delays and the total number of flights delayed per month, culminates in a staggering annual financial burden of $4.2 billion.

Figure 20. Overall Cost Analysis



The implementation of a predictive model for forecasting flight departure delays 15 minutes or more prior to departure presents a strategic investment opportunity. This model is not merely a cost to be incurred; it is a vehicle for cost avoidance and enhanced revenue. The development of such a system entails a one-time setup expenditure and subsequent annual maintenance fees. However, the potential returns far outweigh the initial and ongoing costs.

The predictive model promises to enhance operational efficiencies by enabling better scheduling and resource allocation. It also plays a pivotal role in customer satisfaction and loyalty, as it equips the airline to proactively manage delays and communicate effectively with passengers. By minimizing the frequency and impact of delays, the airline not only saves on direct costs but also fortifies its market position by improving service excellence.

In summary, the investment in predictive modeling technology is a forward-thinking solution that mitigates the financial toll of flight delays, fosters customer trust, and secures long-term profitability for the airline.

12. Conclusion

Because delays are so costly for both airlines and passengers, it is imperative that we develop better models for predicting flight delays. The core hypothesis driving our efforts is that machine learning pipelines equipped with custom features can significantly enhance the accuracy of flight delay forecasts. Our focus on F2-score as the primary evaluation metric, striking a balance between precision and recall with a weighted emphasis on recall, ensures a comprehensive assessment of our models' performance. Through extensive exploratory data analysis, feature engineering, and feature selection, we have identified 20 key flight and weather features that are the most important for predicting delays. Throughout this project, we have rigorous perfected our modelling pipeline to prevent potential data leakage through feature removal, time-dependent cross-fold validation, and careful separation of training and test data. Our rigorous pipeline design, iterative approach and evaluation of many models revealed that our multilayer perceptron model with 2 hidden layers best predicts flight delays with an F2 score on unseen 2019 flight data (52.26%). This success underscores the efficacy of our iterative approach and the significance of addressing specific challenges associated with the temporal nature

of flight data. Looking ahead, this model paves the way for even more advanced approaches, such as deep neural networks or ensemble methods, to continually improve upon flight prediction, fostering a more reliable and efficient air travel experience for both carriers and passengers alike.