Team 4-1 Final Project Phase 3 Report

Team and project meta information

Phase Leader: Aditya

Project Title: Flight Delay Prediction Model

Group number: 4-1

Members:

Sophia Chen: sophiaxmchen@berkeley.edu (mailto:sophiaxmchen@berkeley.edu)

Samuel Ha: samqvha@berkeley.edu (mailto:samqvha@berkeley.edu)

Hans Hernandez: ucb_311@berkeley.edu (mailto:ucb_311@berkeley.edu)

Aditya Mengani: aditya.mengani@berkeley.edu (mailto:aditya.mengani@berkeley.edu)

1. Project Abstract

The critical challenge of flight delays, costing approximately $33 billion in 2019 according to the FAA, underscores the need for effective predictive models. Our project is motivated by the goal of minimizing the economic and logistical impacts of delayed flights on airlines, passengers, and air traffic control. By accurately predicting flight delays at least 2 hours before departure, our model aims to enable airlines and government agencies to better manage resources and improve passenger experiences.
The primary metric guiding our model evaluation and optimization is the F1 score, the harmonic mean of precision and recall. Favoring high precision over recall may lead to airlines to be unprepared to deal with delays while favoring recall over precision may lead to more customers being incorrectly alerted of a delay--as such we seek to optimize for both precision and recall. We worked on addressing class imbalances and optimizing model performance, guided by a detailed understanding of our dataset's characteristics. Our cross-validation results identified XGBoost as the top-performing model based on F1 scores. However, logistic regression with elastic net regularization exhibited the highest F1 score (0.646) on the 2018 validation set, especially excelling in recall (0.693). Consequently, we selected this model for the 2019 blind test. Retraining the logistic regression model on 2018 data, we achieved an F1 score of 0.585 on the 2019 test set. The model demonstrated a precision of 0.628 and recall of 0.548, reflecting a balanced trade-off between false positives and false negatives. This final phase marks a significant advancement in our ability to predict flight delays accurately. Through methodical data preprocessing, strategic model selection, and rigorous blind test evaluation, we have laid a robust foundation for future enhancements and practical applications of our predictive models in the airline industry.

2. Project Description (data and tasks)

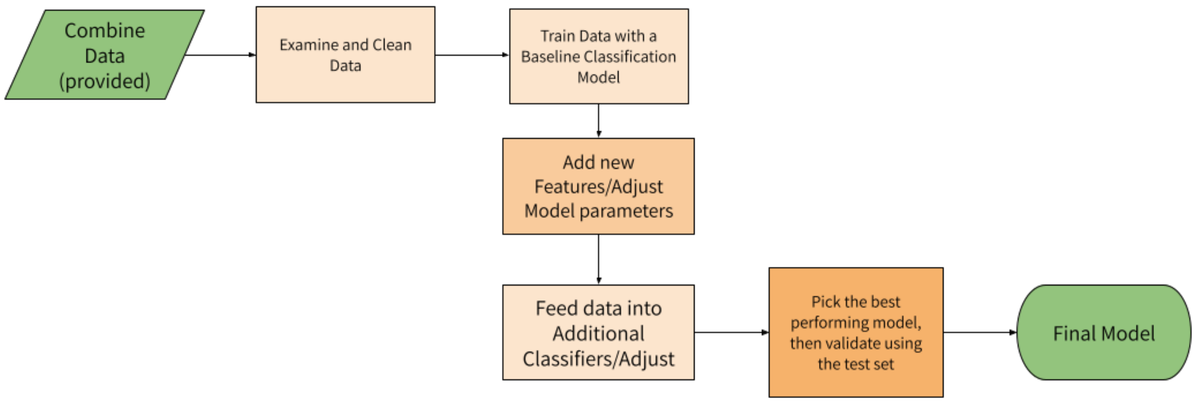
Motivation

Delayed flights are costly to airlines, consumers, and other parties of interest. Per the Federal Aviation Administration (FAA, which manages air traveling in the U.S.), the estimated costs of delayed flights in 2019 was about 33 billion U.S. Dollars, of which 8.3 billion were from the airlines, and 18.1 billion were from the passengers, taking into consideration of their time wasted at the airport. If we are able to provide any kind of prediction about these flight delays, we will be able to plan ahead and proactively accommodate for these situations and better allocate the costs as a result.

Task

The task of this project is to develop a predictive model to predict whether flights will delay no later than 2 hours before departure. Our primary target users for this task are the airline companies and the government agents for airway control. Being able to predict delays in flights can help the airlines provide proper accommodations to passengers stuck at the airport, helping them planning for additional services and managing costs. It can also help the Federal Aviation Administration (FAA, which manages air traveling in the U.S.) have better control over delayed flights and possibly reduce the effect of one delay flight have over other flights. The specific task for the project is to predict whether a flight is going to be delayed by 15 minutes or more at the time of its departure. We will classify each flight based on the selected flight information and weather information. Below is a simple flow chart for the process of generating our model.

Figure 1. Project Flow


Combine Data (provided)
Examine and Clean Data
Train Data with a Baseline Classification Model
Add new Features/Adjust Model parameters
Feed data into Additional Classifiers/Adjust
Pick the best performing model, then validate using the test set
Final Model

Data

The datasets we will use, which are described in detail in the following sections, are from authenticated goverment data sources (see Table 1 below). We will use a joined dataset that combines these datasets together The primary metric we will use to evaluate the models is the F1 score, given that our data has significant class imbalance. To train our algorithms we employ a blocked time series approach which respects the temporal order of flight data and prevents data leakage. Our training-test split was roughly 80-20 overall and for cross-validations within the training set.

Table 1. Datasets and sources

Dataset Name	**Data Source**
Flights	U.S. Department of Transportation
Weather	National Oceanic and Atmospheric Administration Repository
Stations	U.S. Department of Transportation
On-Time Performance of Flights and Weather (OTPW)	Joined from the three above

3. Exploratory Data Analysis

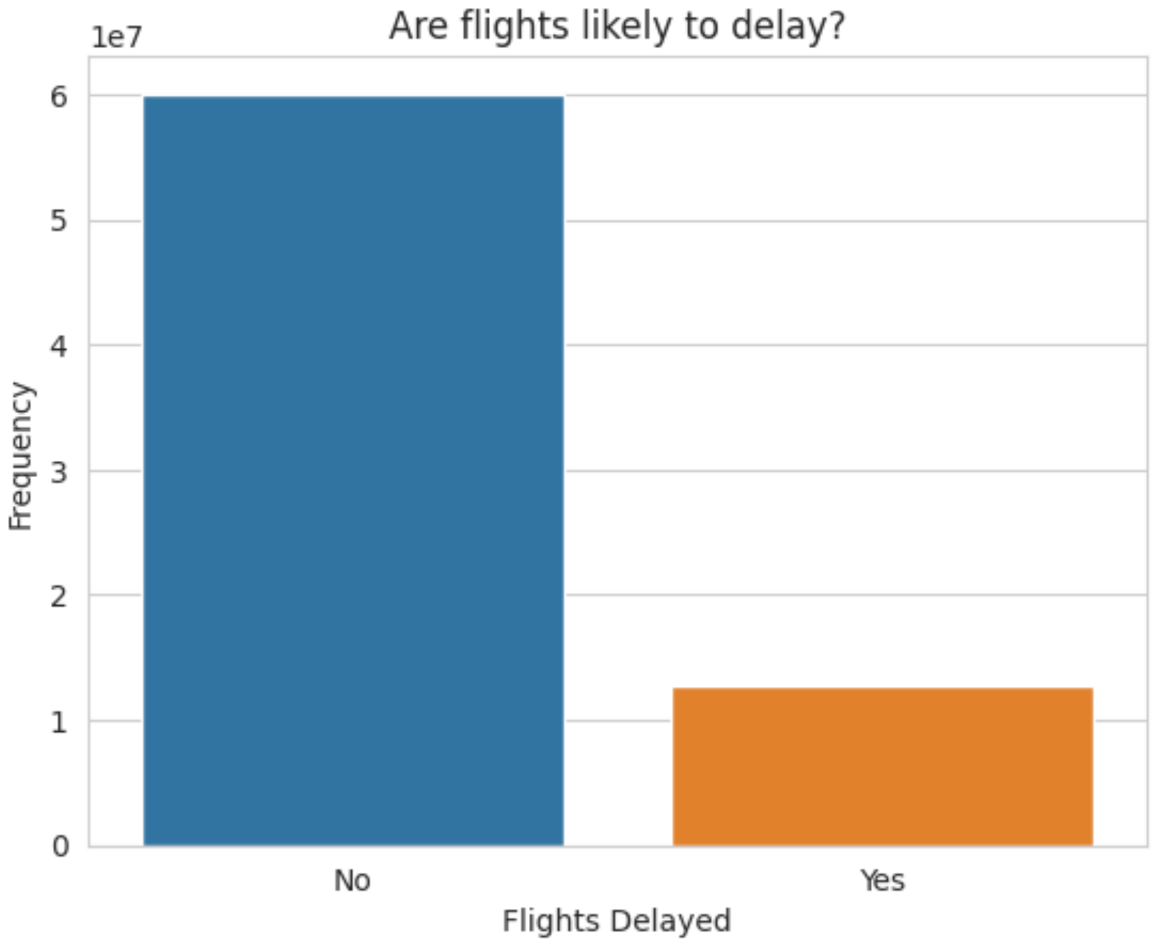
3.1 Raw Data

Flights

Flights data is sourced from the U.S. Department of Transportation (https://www.transtats.bts.gov/Fields.asp?gnoyr_VQ=FGJ). We will explore data from 2015 to 2021 in this analysis. The dataset consists of 74 million records. Of 109 features, 77 features had missing values in one or more observation. The variable 'DEP_DELAY' indicates the length of the delay in minutes and as such is a crucial feature for understanding flight punctuality, potential delays in arrivals, and for predicting future delays.

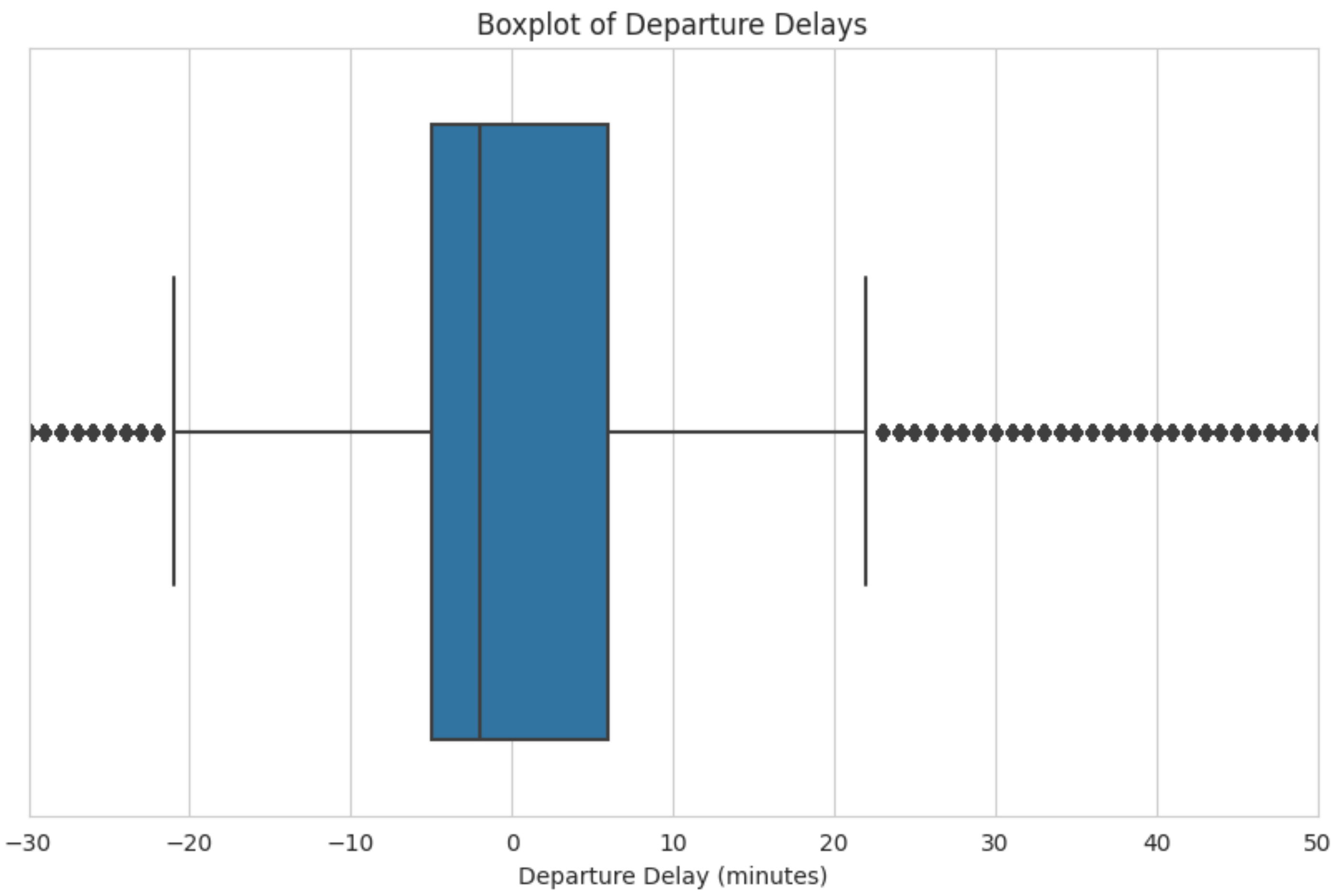
For the purpose of our prediction, our target variable is "DEP_DEL15" which is a flag that indicates whether the flight has been delayed for 15 minutes or more upon departure. Below shows a simple bar plot comparing the number of flights in the groups that are delayed and not delayed.

Figure 2: Are flights likely to delay?



Below is another representation of the distribution of how many minutes of delay does each flight has upon departure. The median is less than zero.

Figure 3: Boxplot on Flight Delays



As we can see in the plots, there are way more flights that departed on time, so we need to take this imbalance into consideration when we process the data and train the models.

To peek into the data and get a sense of how the delayed flights are different from the on-time flights, below are a couple of histograms showing the 2 groups by days of week and months. Both groups indicate that the distributions of delayed flights are different from the on-time flights.

Figure 4: Histograms of Flight Groups by Day of Week

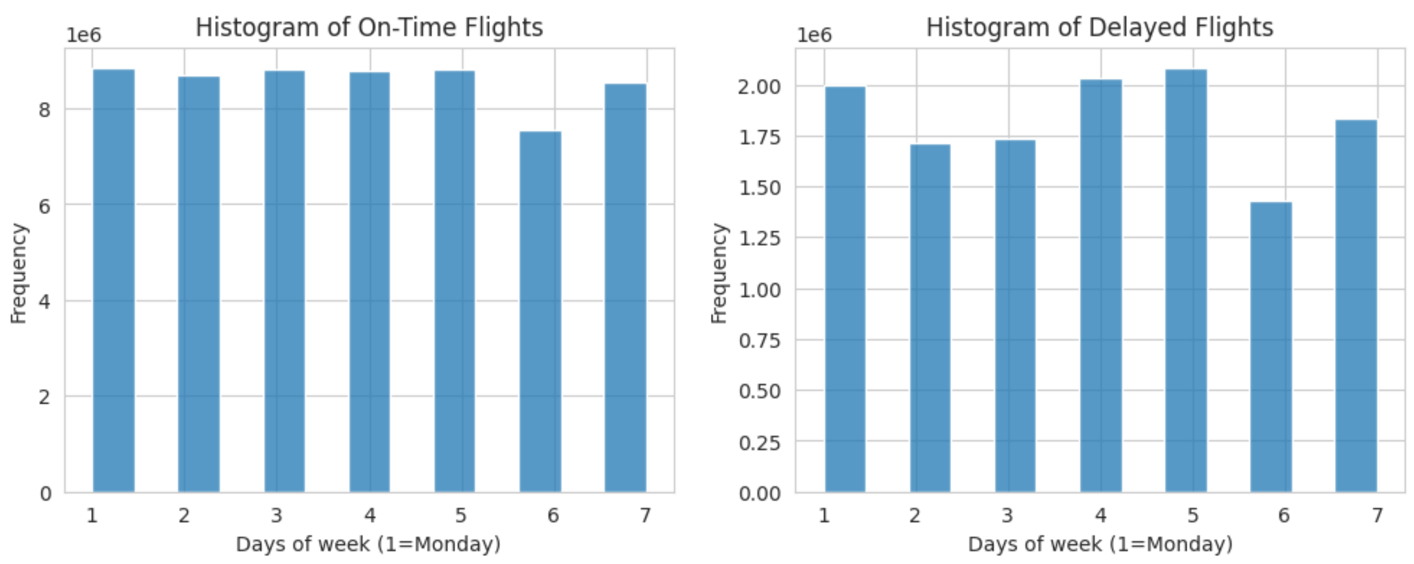
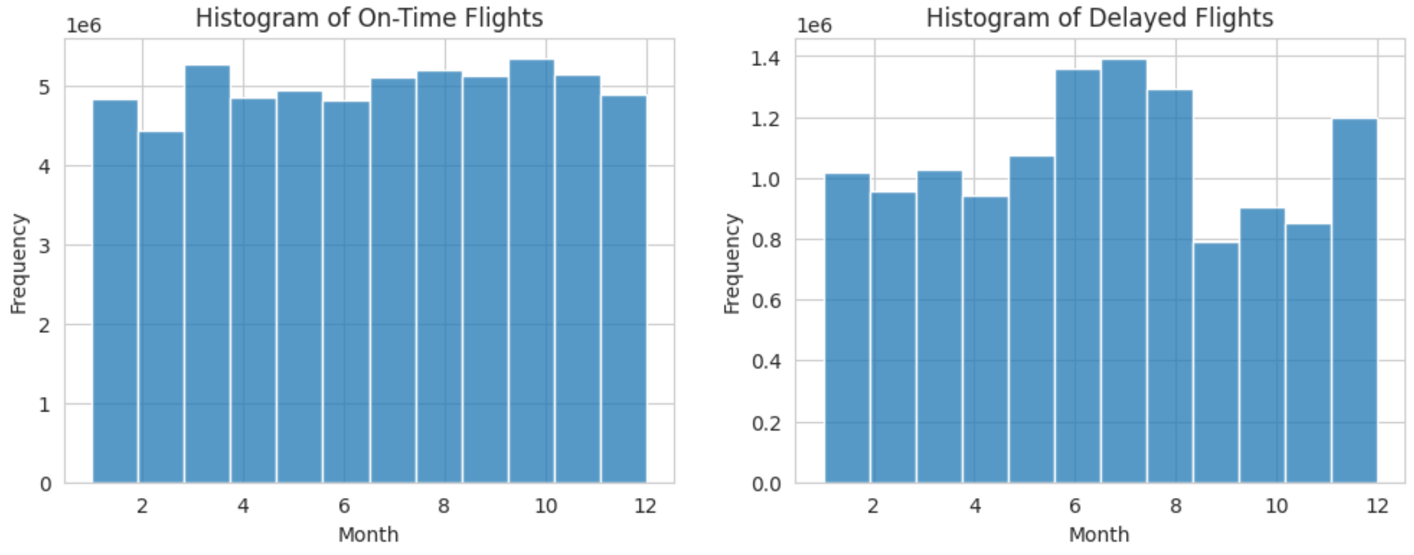


Figure 5: Histograms of Flight Groups by Month



For other categorical features, we constructed some simple stats tables to understand the ranges of delays in departure. The tables below summarizes the best and the worst performing airlines and airports. The numbers after the dash are the delay ratios calculated by dividing the number of delayed departures by the total number of flights. Note that a higher delay ratio indicates worse performance, hence will be at the bottom rankings.

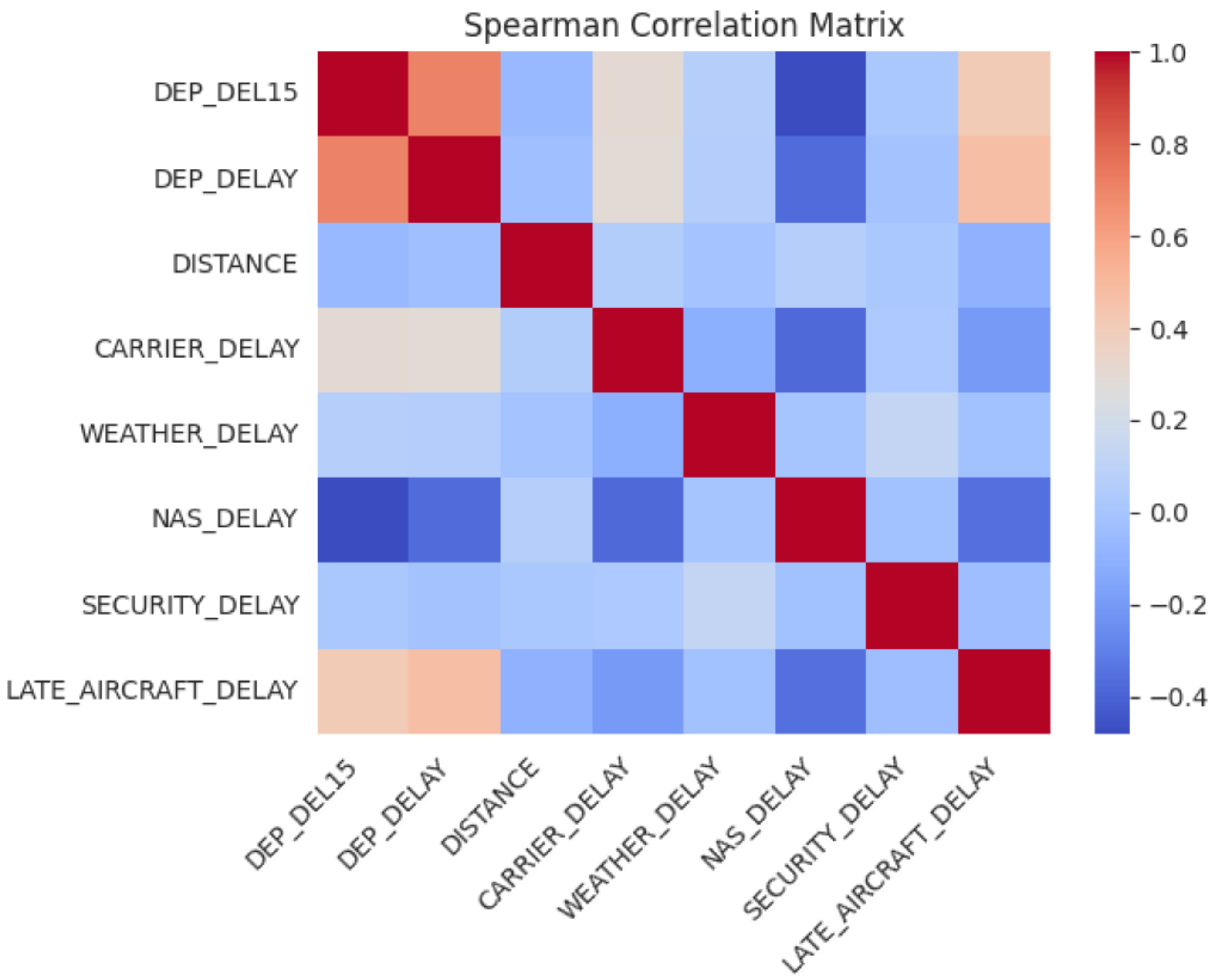
Table 2: Top and Bottom Airlines and Airports by Delay Ratio

Top 3 Airlines	**Bottom 3 Airlines**	**Top 3 Origin Airports**	**Bottom 3 Origin Airports**	**Top 3 Arrival Airports**	**Bott**
Hawaiian Airlines - 0.076	JetBlue - 0.246	EFD(Houston, TX) - 0	YNG(Youngstown/Warren, OH) - 1	FNL(Fort Collins/Loveland, CO) - 0	BIH(

Horizon Air - 0.117	Frontier - 0.233	TBN(Fort Leonard Wood, MD) - 0.045	ENV(Wendover, UT) - 1	LWS(Lewiston, ID) - 0.055	YNG OH)
Alaska Airlines - 0.128	Virgin America - 0.213	LWS(Lewiston, ID) - 0.051	TKI(Tokeen, AK) - 1	EKO(Elko, NV) - 0.056	PSE

For numerical features, this dataset contains several types of delay times for each flight; each type corresponds to a possible reason for the delay. Below is a Spearman Correlation matrix depicting how these delay variables are related. Note that we used the Spearman correlation because our target variable is not continuous.

Figure 6: Spearman Correlation on Delay Variables



While we expect the delay variables to positively correlate with each other, the NAS Delay variable actually negatively correlates with the target variable, which is interesting. However, we won't be able to use these delay variables in our predictive model because they are determined after the fact that the flight has been delayed. These are variables which if included in the training dataset would cause data leaks. We will need to remove these variables and others like it from the training set.

The fields with the most missing values are the different types of delay variables, but those won't be our concern since we have determined that those are not suitable for our model. Other variables have less than 3% of missing values, and we will be handling them in the joined dataset.

Table 3. Partial list of features with missing value counts and percentages for flight data

Num. missing	Perc. missing	Variable
0	0.00%	QUARTER
0	0.00%	MONTH
0	0.00%	DAY_OF_MONTH
0	0.00%	DAY_OF_WEEK
0	0.00%	FL_DATE
0	0.00%	OP_UNIQUE_CARRIER
0	0.00%	OP_CARRIER_AIRLINE_ID
0	0.00%	OP_CARRIER
16380	0.58%	TAIL_NUM
0	0.00%	OP_CARRIER_FL_NUM
0	0.00%	ORIGIN_AIRPORT_ID
0	0.00%	ORIGIN_AIRPORT_SEQ_ID
0	0.00%	ORIGIN_CITY_MARKET_ID
0	0.00%	ORIGIN
0	0.00%	ORIGIN_CITY_NAME
0	0.00%	ORIGIN_STATE_ABR
0	0.00%	ORIGIN_STATE_FIPS
0	0.00%	ORIGIN_STATE_NM

0	0.00%	ORIGIN_WAC
0	0.00%	DEST_AIRPORT_ID
84710	3.02%	DEP_TIME
84710	3.02%	DEP_DELAY
84710	3.02%	DEP_DELAY_NEW
84710	3.02%	DEP_DEL15
84710	3.02%	DEP_DELAY_GROUP
0	0.00%	DEP_TIME_BLK
86342	3.08%	TAXI_OUT
86342	3.08%	WHEELS_OFF
88736	3.16%	WHEELS_ON
88736	3.16%	TAXI_IN
0	0.00%	CRS_ARR_TIME

Another analysis we ran was to look at the airlines that might have "disappeared" over time. Through out the years, airline companies will have mergers or become bankrupt, etc. Below is a list of the airlines that disappeared after some historical records. Note that if an airline has only one year of data we would also note it here.

Table 4: Airlines to be removed from analysis

Airline Company Name	Years disappeared
Atlantic Southeast Airlines	2020-2021
American Eagle Airlines	2016-2017
Horizon Air	2015-2020
US Airways	2015(July)-2021
Virgin America	2018-2021

Weather

The weather data set is sourced from the National Oceanic and Atmospheric Administration Repository. (https://www.ncei.noaa.gov/access/metadata/landing-page/bin/iso?id=gov.noaa.ncdc:C00679) The raw data ranges from 2015 to 2021 as well. It contains 124 features and about 899 million records, each record containing weather information within an hour.

The features are a mix of numeric (double and integer) and string variables. The table below shows their data types and provides a brief description of each variable.

Table 5. Weather Data Type and Description

Variable	Type	Description
STATION	string	A unique station identifier
DATE	string	Date and time data was collected
LATITUDE	double	Latitude of weather station
LONGITUDE	double	Longitude of weather station
ELEVATION	double	Elevation of weather station
NAME	string	Name of city and country of weather station
REPORT_TYPE	string	Type of weather observation
SOURCE	string	Source or combination of sources used in creating the observation
HourlyDryBulbTemperature	double	Ambient air temperature
HourlyWindDirection	double	Wind direction
HourlyWindSpeed	double	Wind speed
REM	string	Remarks
YEAR	int	Year data was collected

With the numerical features, we found some significant amount of missing values - the highest was Hourly Wind Gust Speed, which had 92% missing values. Hourly Precipitation, ranked second in the list, had 87% missing values.

As such we have decided to only use the following features which have null values for less than 15 percent of observations from our 3-month sample dataset. Reducing the dataset, we have 13 features which have an adequate number of observations which are not null. The table below shows the amount of nulls for these variables in the full dataset.

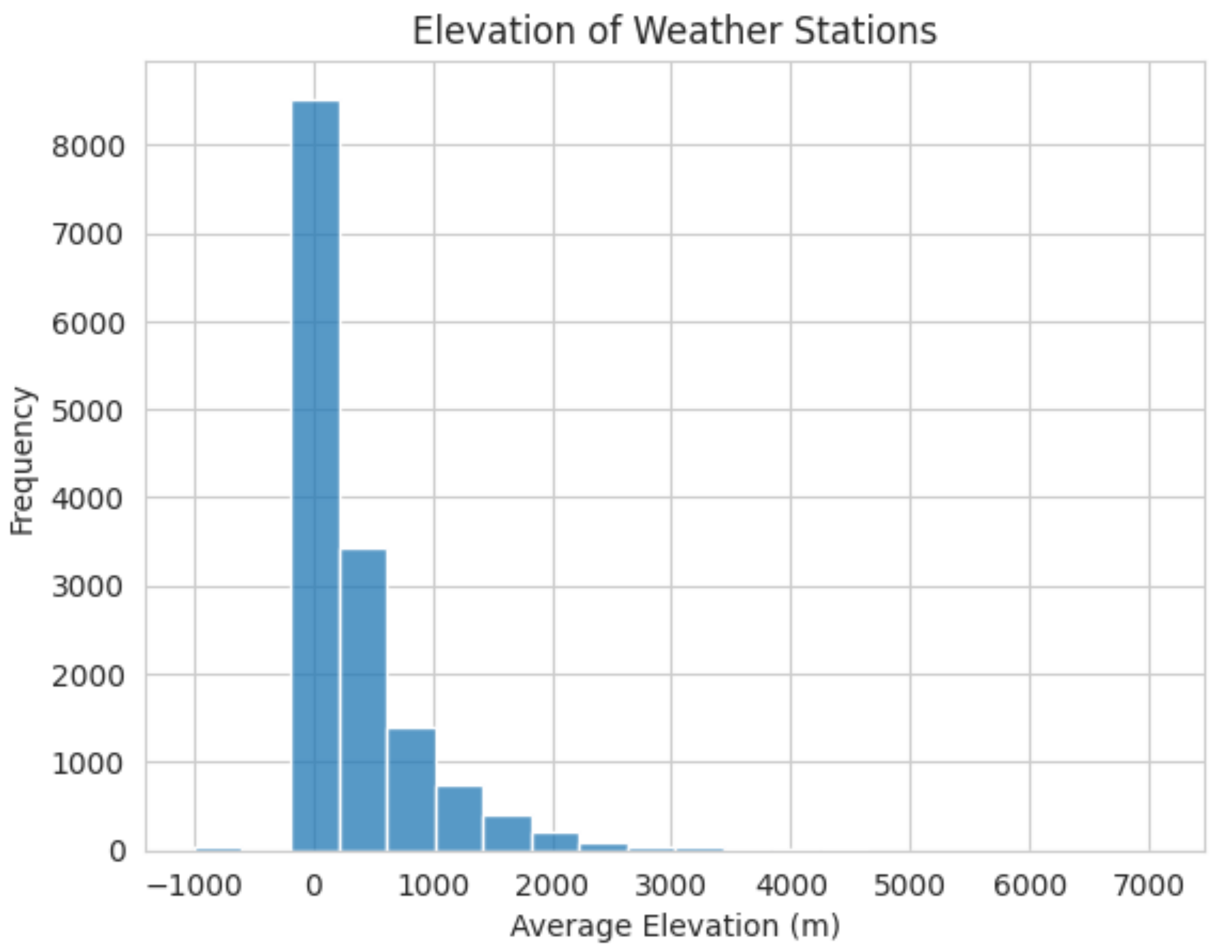
Table 6. Variables selected with least null values

Variable	**Num. missing**	**Perc. missing**
STATION	0	0%
DATE	0	0%
LATITUDE	6388837	1%
LONGITUDE	6388837	1%
ELEVATION	6392201	1%
NAME	6388837	1%
REPORT_TYPE	0	0%
SOURCE	0	0%
HourlyDryBulbTemperature	19395099	2%
HourlyWindDirection	125475322	14%
HourlyWindSpeed	116109419	13%
REM	228945826	25%
YEAR	0	0%

Below are a few histograms for selected variables. Note that since the dataset is too big for us to plot the raw data; all the visualizations provided here are aggregated at a certain level.

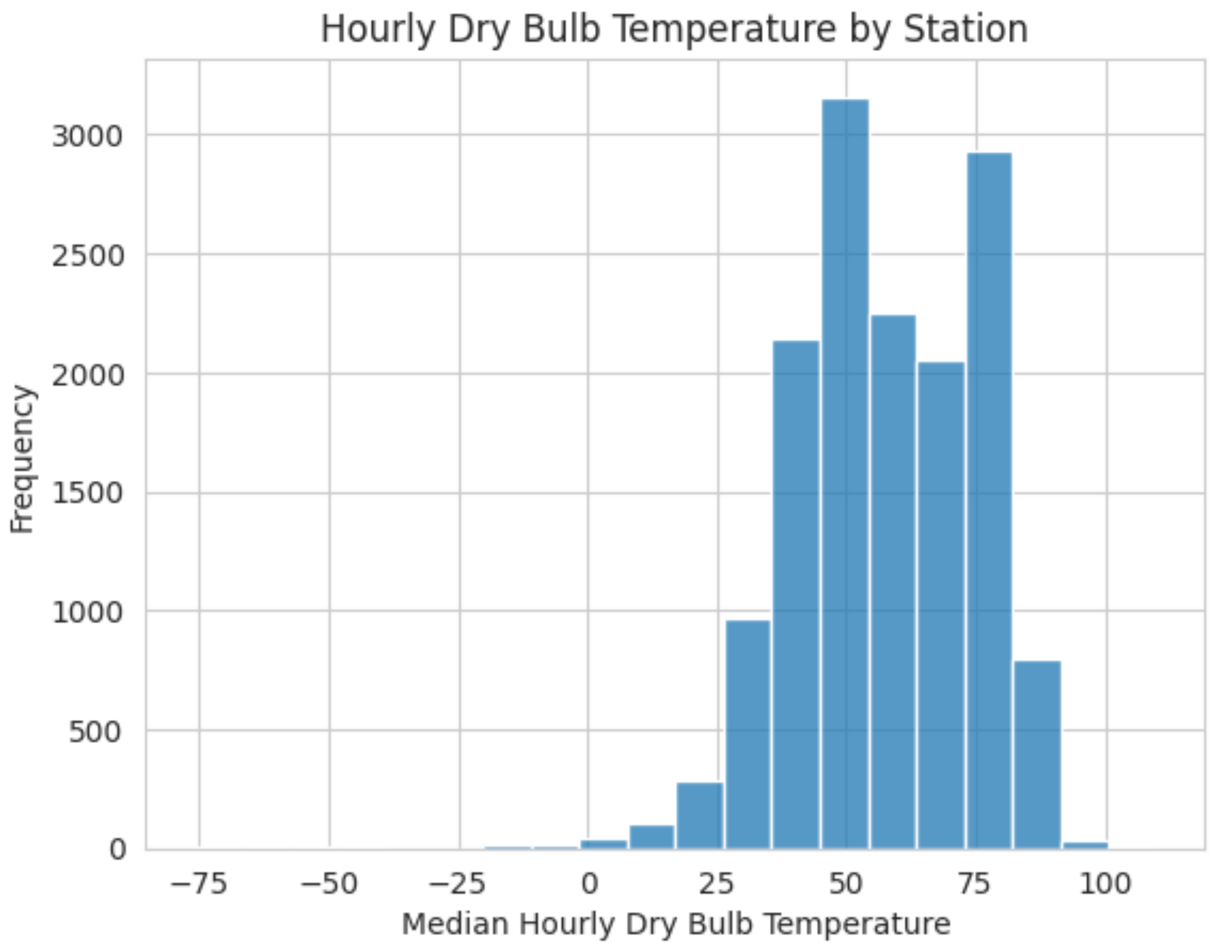
For example, we have aggregated the elevation data by taking the average by station since it is intuitive to assume that one station is only possible to report at one elevation. Here we see that most of the stations are around sea level.

Figure 7: Histogram of Average Elevation by Station



For the Hourly Dry Bulb Temperature, we have also aggregated the data by taking the median of each station to see the range. Majority of the records are between 0 and 100 degrees Fahrenheit.

Figure 8: Histogram of Average Hourly Dry Bulb Temperature by Station



Station

Lastly, we also have a supplemental dataset for airport stations that act as a bridge between airports (as well as weather stations). This dataset is also sourced from the U.S. Department of Transportation. The dataset contains 5 million records and mostly geographical information for the stations and their neighbors.

Below is a table showing the data types and brief description for each variable.

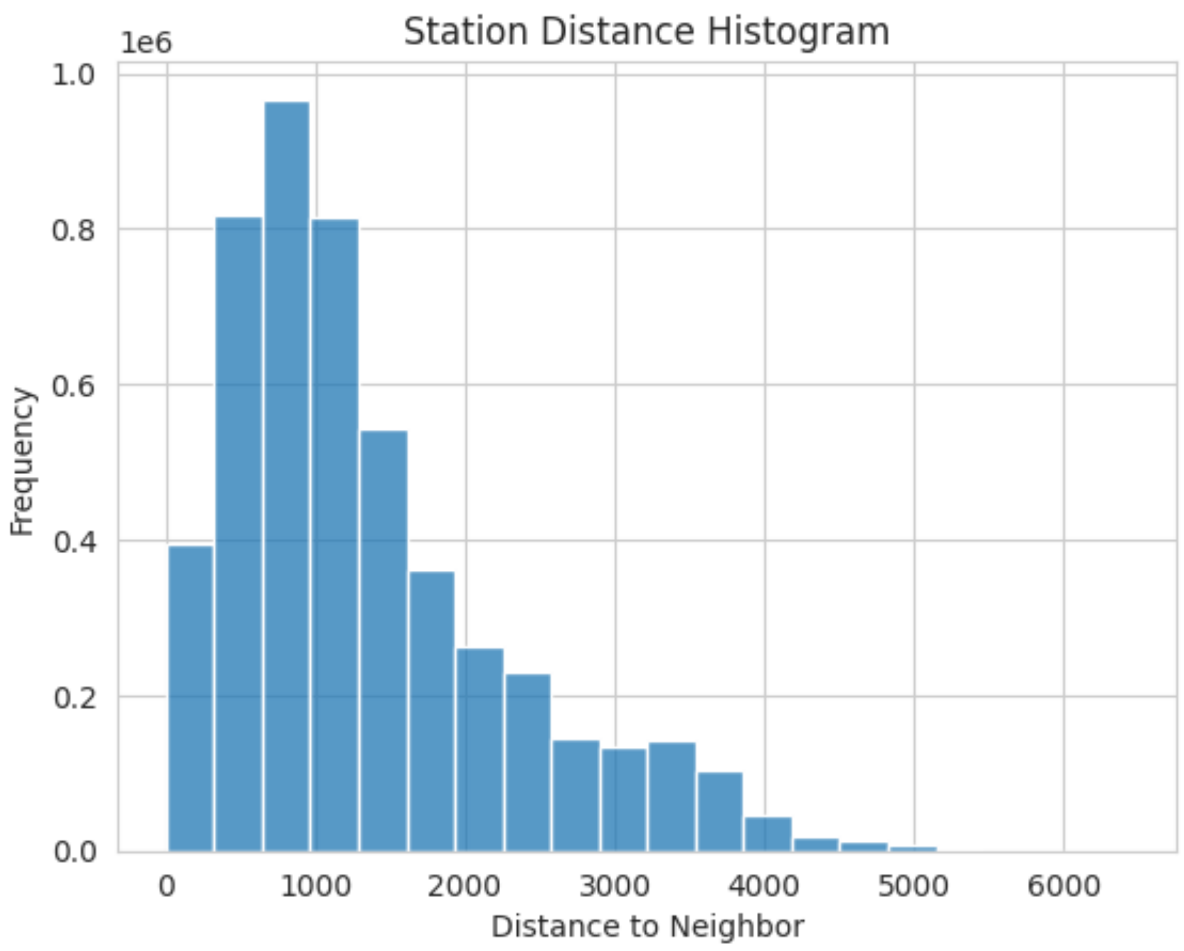
Table 7: Stations Data Description

Variable Name	Data Type	Description
usaf	string	The US Air Force identifier
wban	string	Wireless Body Area Network
station_id	string	Airport station identifier

lat	double	Latitude of the station
lon	double	Longitude of the station
neighbor_id	string	The neighboring airport station ID
neighbor_name	string	The neighboring airport station name
neighbor_state	string	The neighboring airport station State
neighbor_call	string	The neighboring airport station caller ID
neighbor_lat	double	The neighboring airport station's latitude
neighbor_lon	double	The neighboring airport station's longitude
distance_to_neighbor	double	Distance to the neiboring airport station in miles

Below is a histogram showing the distances between the stations and their neighbors. Most airports are around 1000 miles from each other.

Figure 9: Histogram of Distance between Neighboring Stations

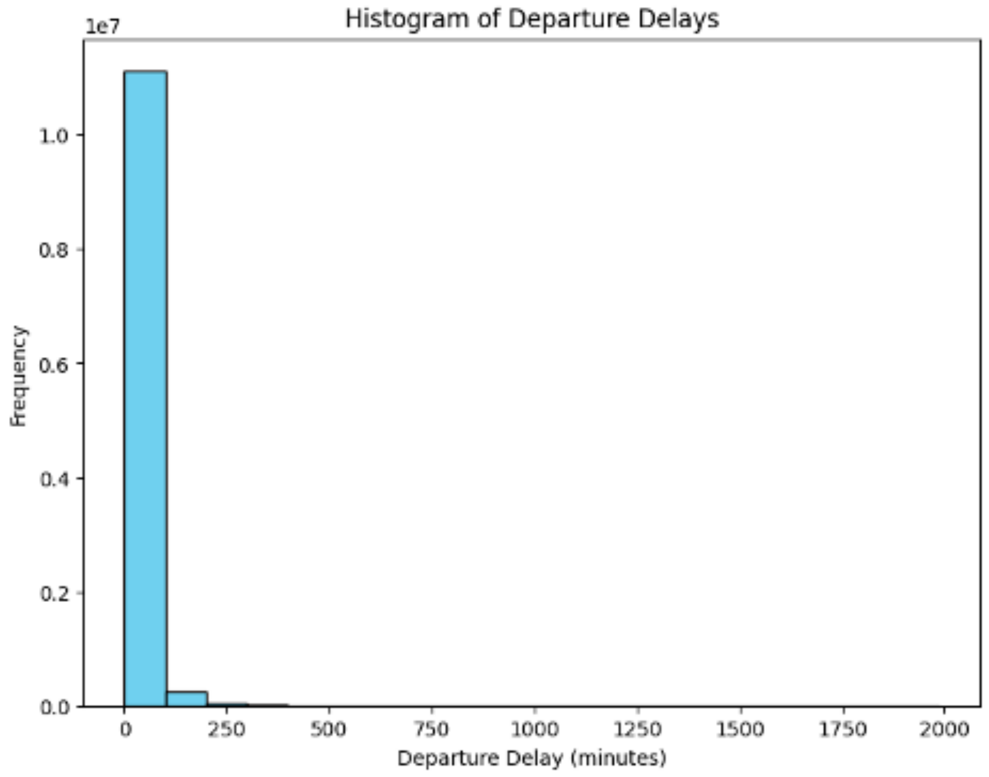


3.2 Joined OTPW Data

Combining the On-Time Performance of flights dataset and the weather data allow us to incorporate analysis of the effect of weather conditions on flight delays. The weather data from **two** hours before the flight and from the nearest weather station was joined to the on-time performance of the flight. The full 12 months of flights in 2015 contain a total of 11,451,590 flights. Of these flights, 18.4 percent of them were delayed, totaling 2,111,470 flights. Given the relatively low frequency of delays, we know that the target variable is imbalanced.

In addition to the class imbalance feature of the data, we observe that the vast majority of flights were delayed for less than 125 minutes, and the distribution of departure delays has an extreme right skew, similar to what we have seen in the raw data.

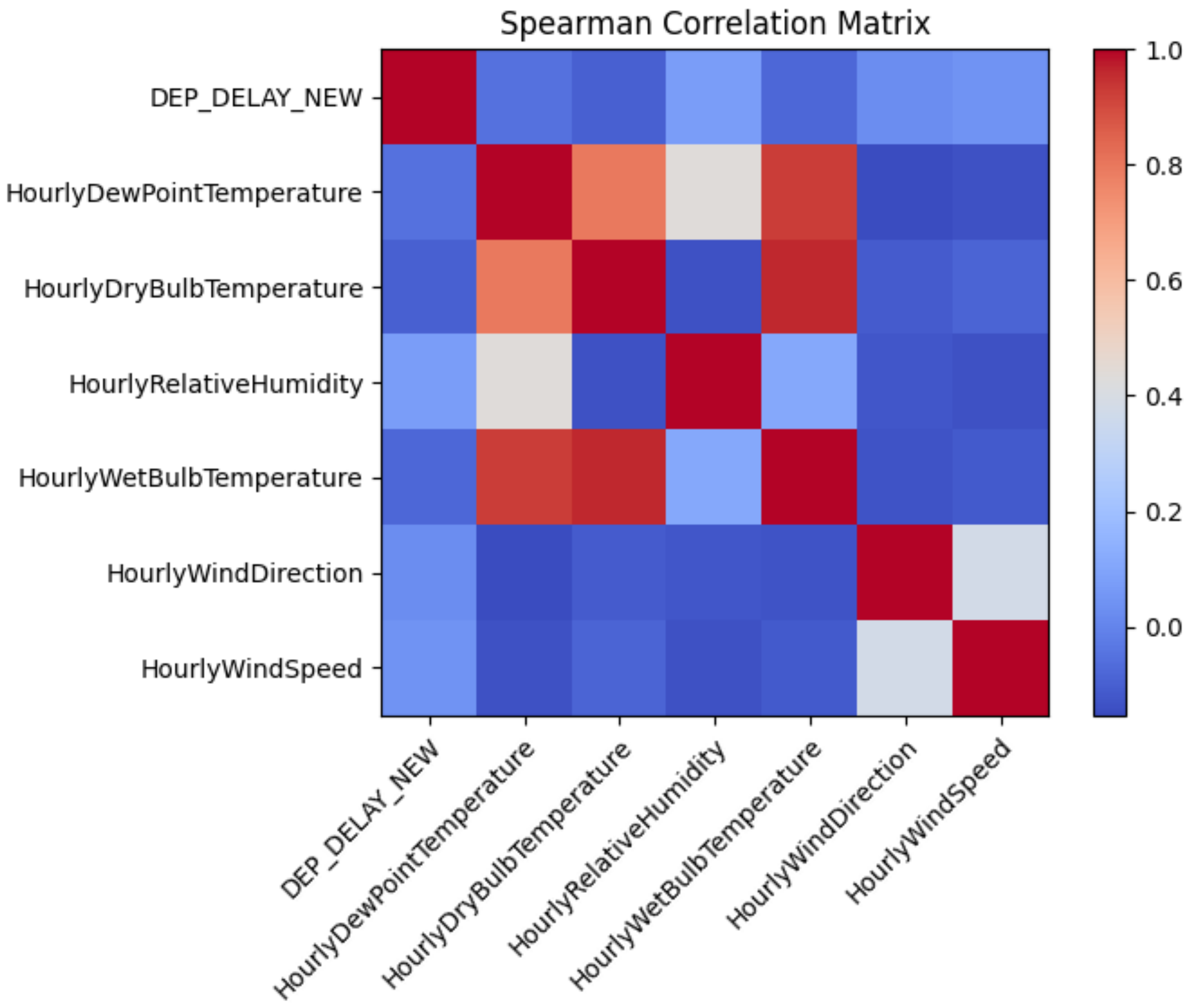
Figure 10: Histogram of Departure Delays, All Quarter 2015



Some features were removed to prevent data leakage, such as the delay times due to various reasons because those factors are determined after the fact that the flight has been delayed. Some were removed because they simply had too many distinct values. For example, the airport ID variable would likely have been relevant, but including it as a categorical variable generates hundreds more values to be included in the ML algorithm and proved a challenge to run within a reasonable amount of time.

Some features were removed due to redundancy. Below is the Spearman correlation matrix for numeric variables that were deemed to be relevant for flight delays in the joined dataset. We observed that there is a high correlation between HourlyDryBulbTemperture, HourlyWetBulbTemperature, and HourlyDewPointTemperature. We chose to keep HourlyDryBulbTemperature over HourlyWetBulbTemperature because HourlyWetBulbTemperature includes information provided by HourlyDryBulbTemperature and HourlyDewPointTemperature. To prevent redundancy and avoid multicollinearity, we removed the HourlyWetBulbTemperature variable.

Figure 11: Spearman Correlation on Delay Variable, Quarter 1 2015



For the joined OTPW data, we have selected the following variables based on the missing values analysis as well as how relevant the variables are to predict delays. Below is a table of the selected variables we will analyze and feed into our models as well as the number of missing values they have. In subsequent sections, we discuss how we addressed the missing values.

Table 8: OTPW Variables chosen for the baseline model

Variable Name	Type	Description	# Missing Values
DEP_DEL15	Integer	Departure Delay Indicator, 15 Minutes or More (1=Yes)	156
DEP_DELAY_NEW	Double	Difference in minutes between scheduled and actual departure time. Early departures set to 0	156
ORIGIN_AIRPORT_ID	String	Identification number of the departure airport	0
FL_DATE	String	Flight Date (yyyymmdd)	0
CRS_DEP_TIME	String	CRS Departure Time (local time: hhmm)	0
DAY_OF_MONTH	Integer	Day of the month	0
DAY_OF_WEEK	Integer	Day of week, 1=Monday	0
OP_UNIQUE_CARRIER	String	2-Character unique carrier code	0
DEP_TIME_BLK	Integer	CRS Departure Time Block, Hourly Intervals	0
TAIL_NUM	String	Tail Number on the aircraft	26
MONTH	Integer	Month of the flight date	0
YEAR	Integer	Year of the flight date	0
HourlyDewPointTemperature	Double	Hourly Dew Point Temp	21
HourlyDryBulbTemperature	Double	Hourly Dry Bulb Temperature	23
HourlyWetBulbTemperature	Double	Hourly Wet Bulb Temperature	44

HourlyRelativeHumidity	Double	Hourly Relative Humidity	24
HourlyWindDirection	Double	Hourly Wind Direction, 0-360 degrees	25
HourlyWindSpeed	Double	Hourly Wind Speed in km/h	23

3.3 Initial Exploration and Findings

During the exploratory data analysis (EDA) phase, we observed that some features in our dataset exhibited significant skewness, indicating an imbalance in their distribution. To address this effectively, we conducted a detailed skewness and kurtosis analysis. This analysis was crucial in determining the appropriate scaling and transformation techniques for different features, ensuring that our model could capture the underlying patterns more effectively.

Skewness and Kurtosis Analysis Our skewness and kurtosis analysis revealed the following:

Highly Skewed Features:
Prev_Flight_Delayed (Skewness: 4.36, Kurtosis: 17.05)
pagerank (Skewness: 0.81, Kurtosis: -0.25)

Moderately Skewed or Symmetric Features:
HourlyDewPointTemperature (Skewness: -0.06, Kurtosis: -0.68)
HourlyDryBulbTemperature (Skewness: -0.10, Kurtosis: -0.69)
HourlyRelativeHumidity (Skewness: -0.31, Kurtosis: -0.75)
HourlyWindSpeed (Skewness: 0.64, Kurtosis: 0.50)

Tailored Feature-wise Preprocessing
Based on these insights, we decided to implement feature-wise preprocessing by applying different scaling or transformation methods to different features:

Log Transformation: For Prev_Flight_Delayed and pagerank, which were highly skewed, we applied a log transformation. This approach was aimed at reducing the skewness and normalizing their distributions.

Min-Max Scaling Plan:
Initially, we planned to use Min-Max scaling for HourlyDewPointTemperature, HourlyDryBulbTemperature, HourlyRelativeHumidity, and HourlyWindSpeed, which were

less skewed. Min-Max scaling was chosen to normalize these features within a specific range, typically [0, 1]. Implementation Challenges and Adaptation However, during the implementation of our data preprocessing pipeline, we encountered issues with the Min-Max scaling, which impacted the model's performance. After thorough investigation and considering the constraints, we decided to proceed with only the log transformation for all features. This decision was made to ensure consistency in preprocessing and to avoid potential issues that could arise from using different scaling methods.

Our approach to addressing feature imbalance was grounded in a detailed statistical analysis of the dataset's characteristics. While our initial plan included a combination of log transformation and Min-Max scaling, practical challenges led us to adapt our strategy. The log transformation alone proved to be a significant step in mitigating the impact of skewed features, thereby enhancing the overall robustness of our predictive models.

Untitled

4. Modeling Pipelines

4.1 General data cleaning, feature selection, pipeline (Aditya)

Data Cleaning

From the initial EDA the following features were selected for further inclusion in a machine learning algorithm : DEP_DEL15, DEP_DELAY_NEW, DAY_OF_MONTH, DAY_OF_WEEK, OP_UNIQUE_CARRIER, DEP_TIME_BLK, MONTH, YEAR, HourlyDewPointTemperature, HourlyDryBulbTemperature,HourlyWetBulbTemperature, HourlyRelativeHumidity, HourlyWindDirection, HourlyWindSpeed.

The features selected above contained extra characters in some records that would make these numbers invalid. This required a data cleanup step where the unwanted characters were filtered out and only clean data was processed further. A set of native spark regular expressions were used to complete this task and were applied to below features: DEP_DELAY_NEW, Year, MONTH, DAY_OF_MONTH, DAY_OF_WEEK, HourlyDewPointTemperature, HourlyDryBulbTemperature, HourlyWetBulbTemperature, HourlyRelativeHumidity,HourlyWindDirection, HourlyWindSpeed

In addition, numeric variables were cast from a string data type to the numeric data type (integer, float, double, etc.) that would optimize for memory efficiency.

Finally, features that missed more than 60 of missing values were dropped alongside other features that were deemed to be irrelevant. Flights that had null values for delay data were dropped.

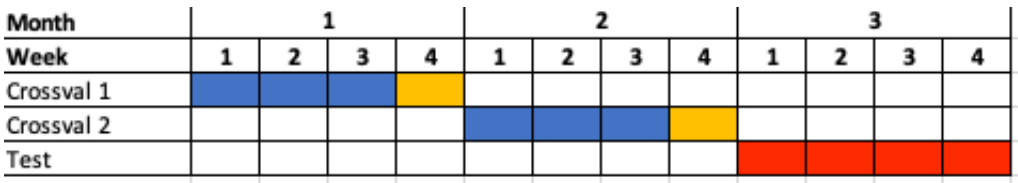
Feature Selection

The following attributes were selected for final features set for creating the baseline model Categoricals :
DAY_OF_MONTH,DAY_OF_WEEK,OP_UNIQUE_CARRIER,DEP_TIME_BLK,MONTH Numericals :
YEAR,HourlyDewPointTemperature,HourlyDryBulbTemperature,HourlyRelativeHumidity,Hour
Target labels: DEP_DEL15, MONTH

Splitting Time series data

We used the blocked time series approch for splitting the time series data into a set of folds. In this approach there is no overlap between the train and test data and the past data is not dependent on the future data events. The data is initially split into a training and test set based on an approximate 66/33 training and test split. Further, the training data split into train and test using 75/25 training and validation split in each fold. The diagram below summarizes our blocked time series approach in which the first three weeks of each month in the training data are used for training, and the last week is allocated as the validation set. Once the performance metrics are obtained, we select the best performing model and run it with the test data and again evaluate performance.

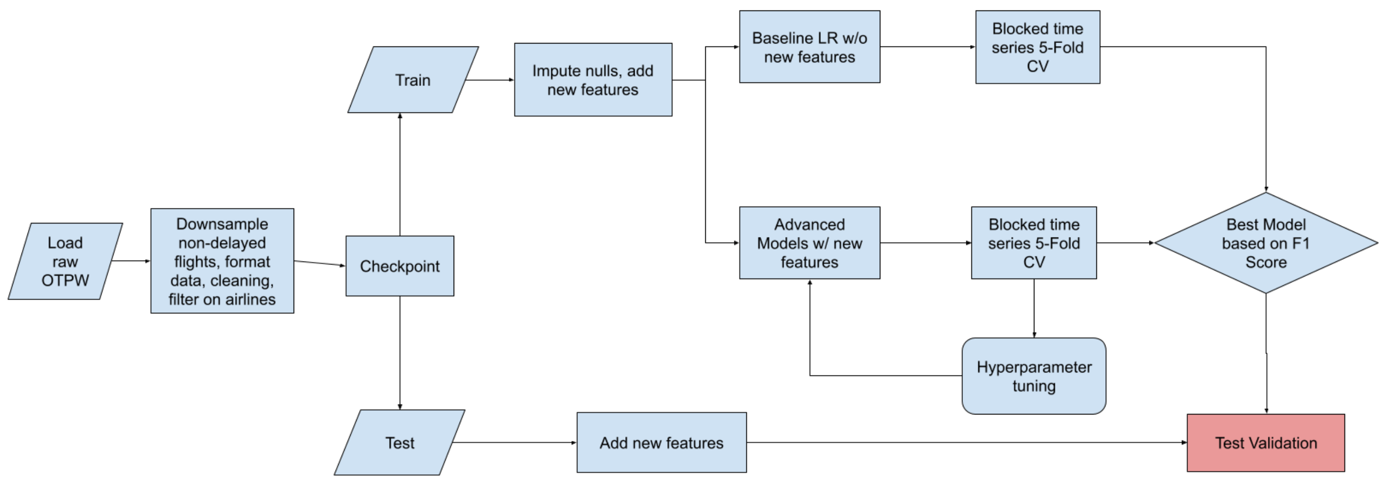
Table 9: Spearman Correlation Matrix, All Quarters 2015



Month	**1**				**2**				**3**			
Week	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
Crossval 1	🟦	🟦	🟦	🟨								
Crossval 2					🟦	🟦	🟦	🟨				
Test									🟥	🟥	🟥	🟥

Pipeline

Figure 12. Pipeline Block Diagram


Load raw OTPW
Downsample non-delayed flights, format data, cleaning, filter on airlines
Checkpoint
Train
Impute nulls, add new features
Baseline LR w/o new features
Blocked time series 5-Fold CV
Advanced Models w/ new features
Blocked time series 5-Fold CV
Best Model based on F1 Score
Hyperparameter tuning
Test
Add new features
Test Validation

The pipeline had the following steps:

Model Matrix stage

- In this stage, the model predictor variables underwent additional transformations depending on their datatypes
- The numerical features went through a native imputer process to impute any null values with the medians
- The categorical features underwent a two step process of conversion to strings using string indexers and then applying one-hot encoder for encoding the categoricals
- After the above steps the categorical data was passed through a vector assembler which will create a set of vectors for all the categorical data

Standard Scaler stage

- In this step StandardScalar functionality is applied to generate the scaled features between 0 and 1 that had a standard deviation of 1 and mean of 0

Logistic regression stage

- In this stage the logistic regression is applied with 10 iterations on scaled features

The loss function for logistic regression is binary cross-entropy loss, and the logistic regression models will use gradient descent to find model weights that minimize binary cross-entropy loss. Binary Cross-Entropy loss is defined as follows:

$$\text{Binary Cross-Entropy} = -1/m \sum_{i=1}^{m} [y^i] log(\hat{p}^i) + (1 - y^i) log(1 - \hat{p})]$$

Once the above steps are implemented the model was fit to the training data and ready for evaluation with the validation or test data. The key metric each model was evaluated against is the F1 score since it is compatiable with data that have imbalanced classes and both precision and recall are important to the airline business. High precision reduces false positives which may cause passenger inconvenience, but the airline will also want to have a high recall in order to correctly identify most of the actual delays there to improve operational efficiency and communicate to the passenger when there is a delay. Other metrics are also used secondarily to supplement our understanding of a model's performance.

F1 score is the harmonic mean of precision and recall:

$(2 * True\ Positives)/(2 * True\ Positives + False\ Positives + False\ Negatives$

Precision is the proportion of true positives out of the total predicted positives:

$$True\ Positives/(True\ Positives + False\ Positives)$$

Recall is the proprtion of true positives out of the total actual positives:

$$True\ Positives/(True\ Positives + False\ Negatives)$$

PR AUC is the area under the precision-recall curve:

$$[Average\ Precision = \int_0^1 p(r)dr]$$

Source: https://sinyi-chou.github.io/python-sklearn-precision-recall/ (https://sinyi-chou.github.io/python-sklearn-precision-recall/)

4.1 Downsampling Non-Delays

Initial Class Imbalance

The initial distribution of our target variable was heavily skewed towards non-delayed flights (80% no delays vs. 20% delayed flights). Such an imbalance can lead to biased model performance, where the model might overly favor predicting the majority class. The table below shows the initial distribution of flights across different years.

Table 10: Raw Distribution of Delayed Flights

Delayed?	Count	Percent
Yes	5676029	18.2%
No	25521301	81.8%
Total	31197330	100.0%

Downsampling Strategy

To mitigate this imbalance, we employed a downsampling strategy targeting the non-delayed flights. This involved randomly sampling from the non-delayed flights to reduce their count to match the number of delayed flights. The key steps in our downsampling process were as follows:

1. We first counted the occurrences of delayed (DEP_DEL15 == 1) and non-delayed (DEP_DEL15 == 0) flights.
2. Using these counts, we calculated a sampling ratio to downsample the non-delayed flights, ensuring the resulting dataset had a balanced distribution of the two classes.
3. We then created a new DataFrame with an equal number of delayed and non-delayed flights by randomly sampling from the non-delayed flights and combining them with the delayed flights.

Maintaining Temporal Integrity

An important aspect of our downsampling strategy was to maintain the temporal distribution of the data. We ensured that the proportion of data from each year remained consistent before and after downsampling. This approach preserved the integrity of the time series nature of our dataset, allowing for a more accurate representation of each year's data. The tables below demonstrate how the yearly distribution was preserved post-downsampling.

Table 11: Raw Data Distribution of Years

YEAR	count	percent
2015	5726181	18.4%
2016	5546861	17.8%
2017	5572592	17.9%

2018	7087730	22.7%
2019	7263966	23.3%
Total	31197330	100.0%

Table 12: Downsampled Data Distribution of Years

YEAR	count	percent
2015	2095294	18.5%
2016	1973748	17.4%
2017	2024605	17.8%
2018	2591451	22.8%
2019	2671488	23.5%
Total	11356586	100.0%

Post-Downsampling Distribution

The effectiveness of our downsampling strategy is evident in the distribution of the data post-processing:

By implementing a careful downsampling strategy, we addressed the class imbalance issue in our dataset without compromising the temporal distribution of the data. This approach substantially improved the performance our predictive models. Precision and recall improved tenfold or more for some models with the adoption of downsampling into our pipeline. Downsampling ensure that the models could effectively learn from both delayed and non-delayed flights and thus enhanced their predictive performance.

- The table below highlights the achieved balance between delayed and non-delayed flights, with each class now constituting 50% of the data.

Table 13: Downsampled Distribution of Delayed Flights

Delayed?	count	percent
Yes	5676029	50.0%
No	5680557	50.0%

Total	11356586	100.0%

4.2 Imputation of Null Values using Moving Averages

We employed a specialized method for imputing null values in our time series weather dataset, pivotal for predicting flight delays. This approach centers on the use of moving averages, calculated using the calculate_moving_averages function, to replace missing values in weather-related columns.

Advantages of Using Moving Averages:

- Preserving Temporal Integrity: By exclusively using past data points for each calculation, the integrity of the time series is maintained.
- Avoiding Data Leakage: The deliberate exclusion of future data in the moving average calculation prevents the model from accessing information not available at the prediction time, thus avoiding data leakage.
- Smooths Short-Term Fluctuations: Moving averages help in smoothing out short-term fluctuations and highlight longer-term trends or cycles in the data.
- Fills Gaps with Context-Relevant Data: Unlike simple imputation methods (like filling with mean), moving averages provide a more contextual and dynamic way of handling weather missing data, reflecting temporal variations.
- Improves Model Accuracy: Accurate imputation using moving averages leads to a more reliable dataset, which is crucial for predictive accuracy in our flight delay models.

Time: Imputation of null values for OTPW 12-months takes about 11 minutes for the train dataset and about 8 minutes for the test dataset.
Code Notebook Link (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/173910692692651/command/173910692692710)

4.3 Feature Engineering

To supplement the features in the joined OTPW dataset, we created three time-based features, one historical feature, and a graph feature:

Recency:
The Recency feature is a significant time-based attribute in our flight delay prediction model. It quantifies how recently a similar event (like a flight delay) occurred at the same airport. Recency provides insight into patterns of delay occurrences, under the premise that recent delays might indicate ongoing issues affecting subsequent flights, such as adverse weather conditions or operational challenges.

Part of Day:
Air travel efficiency varies throughout the day due to factors like air traffic, staffing levels, and passenger behavior. The Part of Day feature categorizes flight times into segments like Morning, Afternoon, Evening, and Night. Understanding how flight delays are distributed across different times of the day helps in identifying peak periods prone to delays and optimizing scheduling and resource allocation.

Seasonality:
Seasonal variations significantly influence flight patterns due to changes in weather conditions, passenger travel preferences during holidays, and vacation periods. The Seasonality feature reflects these variations. Seasonality is crucial in predicting delays as certain seasons have higher travel demand and weather-related disruptions. For example, winter might have more delays due to snow and ice, while summer might experience issues due to thunderstorms or high passenger volume.

Previous Flight Delay: In addition to time-based features, our model includes a crucial aircraft-specific historical feature, termed the 'Previous Flight Delay' feature. This attribute indicates whether the preceding flight of the same aircraft was delayed. Understanding an aircraft's recent operational history offers valuable insights into its current state, enhancing our ability to predict delays more accurately. While time-based features capture external factors affecting flight delays, the 'Previous Flight Delay' feature focuses on the internal, aircraft-specific aspect, offering a comprehensive view of potential delay causes.

PageRank: Airports and flights between airports form a natural node and edge data structure, and so we used PageRank to capture the relational nature of flight delays in which delays at one airport can lead to flight delays in other airports. In our graph data, airports were source and destination nodes and flights connecting them were edges. The weights of the edges were the sum total of delays that flights from that airport had from 2015 to 2017, the length of the training set excluding the 2018 validation set. The damping factor was set at the standard 0.15 and the tolerance was set at 0.01. To prevent leakage, the PageRank score was only calcuated from 2015 to 2017 and scores were joined to the 2018 validation and the 2019 test sets.

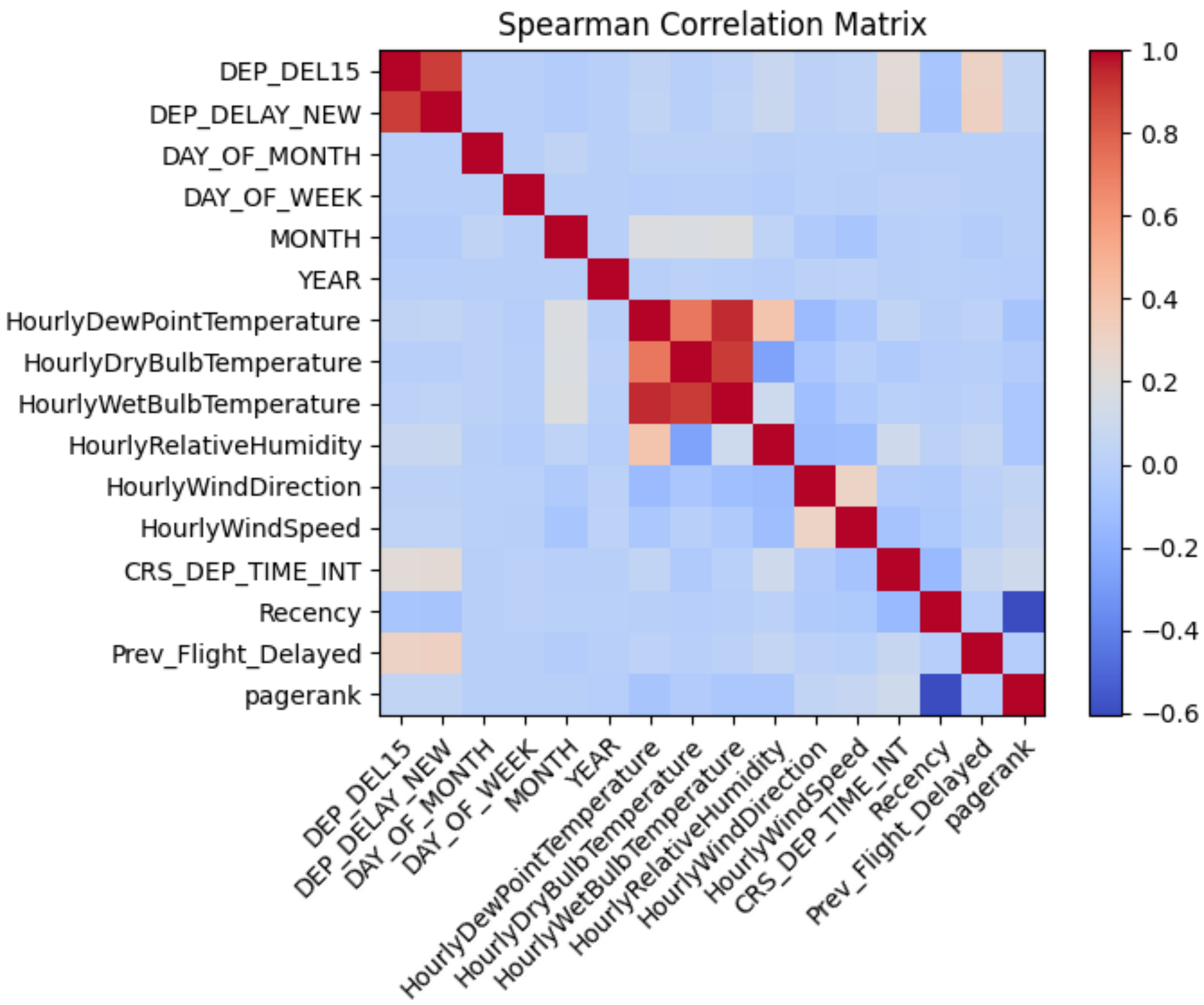
Time:

Adding the time-based and historical features for OTPW 12-months took about 18 minutes for the train dataset and about 4 minutes for the test dataset. Training the PageRank score on the training dataset using GraphFrames took 2 minutes.
Code Notebook Link (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/173910692692651/command/173910692692710)

4.4 EDA on New Features

For the numerical features we generated, below is a Spearman Correlation matrix to show how they might be relevant to the target and other variables. As we see in the heat map, recency is slightly negatively correlated with our target variable, while Previous Flight Delay and the PageRank are both slightly positively correlated with our target variable.

Figure 13: Spearman Correlation on Created Features



4.5 Blocked Time Series

Due to the time-series nature of the flight data there is always a scope for leakage of data when splitting the data between train, validation and test sets. In addition, the leakage can affect the individual folds of data during cross validation affecting the true observability and metrics generated from the machine learning predictions.

For example leakage can happen when the future data is used to predict or forecast events in the past. Past events can help to predict future but vice versa is not possible. In order to avoid these issues, we have approached the problem of leakage with Blocked time series splits of data during cross-validation.

We implemented a standardized algorithm based on the blocked timeseries data that can be applied to any data size. Initially this algorithm works by measuring the total records in the train set and dividing this by the number of folds that are pre-determined as part of the experimentation. This determines the fold size for each iteration or the total number of records that would be run in each iteration.

Once the fold size is determined, the train data is split into train and validation sets based on 75:25 splits for each fold. The concept of windowing is used to sort the data in the train data based on the date variable "DATE_VARIABLE" by ordering all the records based on the "DATE_VARIABLE" in ascending order. A rank is assigned to each record after applying the windows function. The first record in the entire set of train data is assigned rank 0 and the last record is given the rank of (record_size - 1).

During each iteration for every fold, the records are split on their index numbers starting from index 0 to index n-1 where n is the last record in the train set. Then the indexes are collected and filtered on original dataframe and further split between train and test sets over the ratios discussed above.

Thus, the entire data is split into blocks of data using Blocked time series data and split between train and validation sets. These are further used downstream as part of the cross validation experiments.

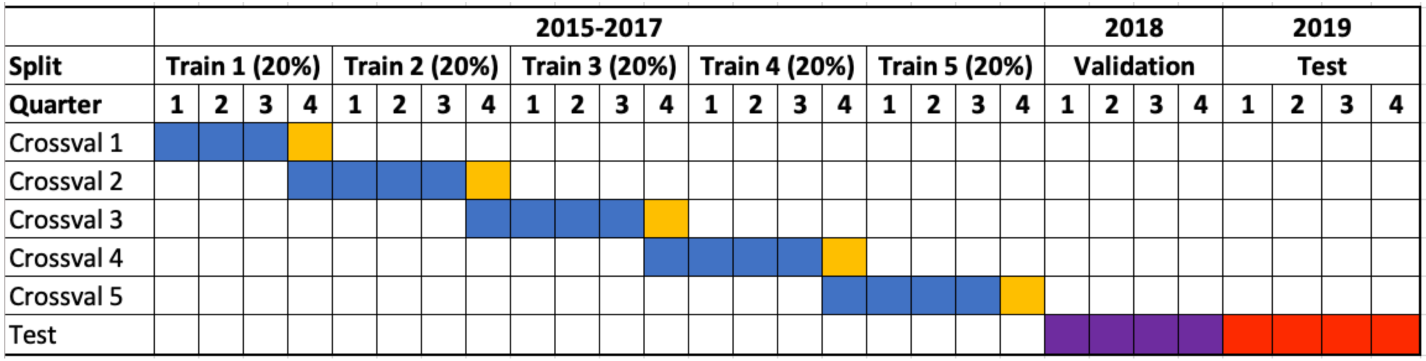
Here is the algorithm used for the blocked timeseries split:

```
class BlockingTimeSeriesSplit():
def __init__(self, n_splits):
    self.n_splits = n_splits

def get_n_splits(self, X, y, groups):
    return self.n_splits

def split(self, X, y=None, groups=None):
    n_samples = len(X)
    k_fold_size = n_samples // self.n_splits
    indices = np.arange(n_samples)

    margin = 0
    for i in range(self.n_splits):
        start = i * k_fold_size
        stop = start + k_fold_size
        mid = int(0.8 * (stop - start)) + start
        yield indices[start: mid], indices[mid + margin: stop]
```

references: Blocked timeseries split (https://hub.packtpub.com/cross-validation-strategies-for-time-series-forecasting-tutorial/)

Below is a diagrammatic representation of the time series data based on the flight date split across train/valid and test splits.

Table 14: Flight date split across train/valid and test splits



	2015-2017																				2018				2019			
Split	**Train 1 (20%)**				**Train 2 (20%)**				**Train 3 (20%)**				**Train 4 (20%)**				**Train 5 (20%)**				**Validation**				**Test**			
Quarter	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
Crossval 1	🟦	🟦	🟦	🟨																								
Crossval 2				🟦	🟦	🟦	🟦	🟨																				
Crossval 3								🟦	🟦	🟦	🟦	🟨																
Crossval 4												🟦	🟦	🟦	🟦	🟨												
Crossval 5																🟦	🟦	🟦	🟦	🟨								
Test																					🟪	🟪	🟪	🟪	🟥	🟥	🟥	🟥

The initial 3 years of data 2015-2017 is considered as train data and is applied with blocked time series split of 5 folds each. 2018 data is treated as validation data which is used to determine the best model out of all advance models. Further, 2019 data is exclusively used to evaluate the performance of the best model among all those proposed.

4.6 Hyperparameter Tuning

Our initial approach to hyperparameter tuning was to implement Bayesian search. Bayesian search is known for its efficiency in finding the best parameters as it builds a model of the objective function and uses that model to select the most promising hyperparameters to evaluate in the true objective function. Unfortunatelly we faced significant challenges with due to the custom cross-validation class. The time series nature of the data required a specialized approach to cross-validation to maintain the temporal order of observations. For this purpose, our teammate, Aditya, developed a custom class, BlockingTimeSeriesSplit, tailored to our specific needs. This class ensured that the data splits respected the temporal sequence, crucial for time series analysis. The integration of the BlockingTimeSeriesSplit with the hyperopt library, which we intended to use for Bayesian search, presented significant challenges. hyperopt is a powerful library for hyperparameter optimization but requires compatibility with the cross-validation methodology used. The custom nature of BlockingTimeSeriesSplit created complexities in integrating with hyperopt, which is typically designed to work seamlessly with standard cross-validation approaches provided by libraries like Scikit-Learn.

After spending several hours attempting to reconcile these differences and integrate the custom class with hyperopt, we realized that the integration was very complex due to the unique structure and requirements of the BlockingTimeSeriesSplit and it would take a significant amount of time.

Given the challenges faced with Bayesian search and the time constraints of the project, we pivoted to a more feasible yet effective approach: Random Search for hyperparameter tuning. Random Search, while not as sophisticated as Bayesian search, still offered a robust method to explore a wide range of parameter values. It randomly selects combinations of hyperparameters to evaluate, providing a good balance between exploration of the parameter space and computational efficiency.

This approach, coupled with our custom BlockingTimeSeriesSplit, allowed us to effectively tune our models while respecting the temporal nature of our time series data.

Code Notebook Link (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/173910692692651/command/173910692692710)

4.7 Cross-validation

4.7 Cross-validation

For the cross-validation process in our project, we faced the unique challenge of handling time series data. To address this, we utilized a custom cross-validation technique known as "Blocking Time Series Split." This approach was crucial in ensuring that our model's validation scheme respected the temporal sequence of the data, a critical aspect of time series analysis. This class divides the dataset into distinct training and validation sets while maintaining the chronological order, thereby preventing data leakage and ensuring that future data is not inadvertently used to predict past events.

Key Features of Blocking Time Series Split

- **Temporal Integrity:** By maintaining the sequence of observations, the class ensures that each split respects the temporal order, which is paramount in time series forecasting.
- **Avoiding Data Leakage:** It introduces margins between training and validation folds to prevent the model from seeing lagged values that could be used both as a regressor and as a response. This separation is crucial to avoid overfitting and to simulate real-world scenarios where future data is not available.
- **Iteration Isolation:** The second margin between the folds used in each iteration ensures that the model does not memorize patterns from one iteration to the next, thereby enhancing the robustness of our model evaluations.

Application in Hyperparameter Tuning In the context of hyperparameter tuning, we integrated the BlockingTimeSeriesSplit class to methodically evaluate various parameter configurations. The process was as follows:

Define Time Series Split:
We instantiated the BlockingTimeSeriesSplit with a specified number of splits, ensuring comprehensive and temporally consistent cross-validation.

Loop Through Splits:
We iterated through each split, using the custom time series split to generate training and validation indices. This ensured that each iteration of our model training and validation was based on a temporally coherent split of the data.

Hyperparameter Tuning:
Within this loop, we experimented with different hyperparameters for various models (Logistic Regression, XGBoost, Decision Tree, etc.), evaluating their performance on the

temporally split data. This approach allowed us to fine-tune our models while rigorously adhering to the temporal dynamics inherent in our dataset.

Code Notebook Link (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/173910692692651/command/173910692692710)

4.8 Blind Test

In line with the time series nature of our data, we split our dataset into training and testing sets based on the number of records and their chronological order. This approach ensured that the temporal sequence of the data was preserved, which is crucial for the integrity and relevance of our time series analysis.

Implementation of the Blind Test
For the blind test, we exclusively used data from the year 2019. This dataset was kept entirely separate from our training process to uphold the principle of model validation and to avoid any form of data leakage. The blind test set was not used in any way during the model training phase, ensuring that our evaluation was as unbiased and as realistic as possible.

Key Considerations in the Blind Test:

- **Temporal Integrity:** By using 2019 data for the blind test, we maintained the chronological sequence, thereby testing the model's ability to generalize and predict future trends based on past data.
- **Avoiding Data Leakage:** We strictly isolated the test data throughout the model development process. This isolation was critical to ensure that our model's performance on the blind test was a true reflection of its capability to handle unseen data.
- **Real-world Applicability:** The blind test essentially served as a real-world scenario where the model's predictive power was tested on completely new data, simulating how it would perform post-deployment.

Code Notebook Link (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/173910692692651/command/173910692692710)

Untitled

5. Results and Discussion

5.1 Hyperparameter Tuning Results

With the Random Search Cross-Validation approach, we set the values for a series of hyperparameters that would be randomly chosen for logistic regression, XGBoost, and decision tree algorithms. For each of the five folds of the training set, a model was trained on a fold and evaluated against a validation set. Table 15 shows the average metrics of the models trained and evaluated on each fold of the hyperparameter tuning.

Table 15: Metrics of hyperparameter tuning fold of models

Model	AUC	Precision	Recall	F1
XG Boost	0.661	0.645	0.520	0.568
Decision Tree	0.511	0.627	0.446	0.507
Logistic Regression	0.521	0.163	0.016	0.029

Since the hyperparameters for each fold's model were randomly chosen, the average performance of the model during hyperparameter tuning is only suggestive of the performance that could be expected from a tuned model during cross-validation.

After the Random Search Cross-Validation was completed, we found that the below parameters resulted in the highest F1 scores for each model:

- Logistic regression with elastic net regularization:
  - Regularization parameter: 0.1
  - Max iterations: 50
- XGBoost:
  - Learning rate: 0.1
  - Max depth: 7
  - Min child weight: 1
- Decision Tree:
  - Max depth: 10
  - Min instances per node: 2

5.2 Cross-validation Results

Among all models that were trained on the 2015 to 2017 cross-validation data, XGBoost performed the best with respect to our metric of F1 scores. The XGBoost model also outperformed the logistic regression with regularization and decision tree models but by fairly narrow margins. The average metric scores across the five folds of cross-validation for each model are shown below. The cross-validation process XGBoost, logistic regression, and decision trees in total took 2.2 hours in a loop. The MLP classifier was run by itself in a loop due to memory constraints of Databricks.

Table 16: Average Model Performance on 5-fold Cross-validation Sets

Model	**AUC**	**Precision**	**Recall**	**F1**	**Time**
XGBoost	0.666	0.649	0.529	0.573	2.2 hours
LR-EN	0.657	0.646	0.508	0.547	2.2 hours
Decision Tree	0.511	0.641	0.471	0.534	2.2 hours
MLP	0.500	0.075	0.000	0.000	1.4 hours
LR-BL	0.659	0.643	0.516	0.557	1.6 hours

From the cross-validation above, we observe that the XGBoost models on average received the highest scores on the metrics, but it was trailed closely by the logistic regression with elastic net regularization, logistic regression baseline, and decision trees. The only models that did not perform well at all belonged to the multi-layer perceptron model which did not benefit from hyperparameter tuning previously. Of note is the lack of difference between the logistic regression with regularization and the logistic regression baseline model which suggests that much of the performance of the former model is due to the simple logistic regression and not the additional features or regularization.

We selected the best-performing model for each type of model and evaluated it against the 2018 validation set. The purpose of doing so was to determine which model would perform the best on as of yet unseen validation data and select one which would be tested against the 2019 test data. Below are the results of evaluating the best model from cross-validation on the 2018 validation set. We observed that the logistic regression model has the highest F1 score of all models. The logistic regression model particularly excelled in its recall with a recall of 0.693. The logistic regression model performed marginally worse in precision and AUC than the XGBoosted model. Given its high score on our key F1 metric and strong overall performance across other metrics, we selected the logistic regression model with elastic net regularization as our best machine learning model with which to evaluate the 2019 test data.

Table 17: Best Trained Model Performance on 2018 Validation Set

Model	AUC	Precision	Recall	F1	Time
LR-EN	0.655	0.605	0.693	0.646	6.6 minutes
XGB	0.666	0.624	0.637	0.631	3.6 minutes
DT	0.451	0.608	0.511	0.555	3.2 minutes
LR-BL	0.659	0.617	0.659	0.637	6.6 minutes
MLP	0.499	0.500	0.000	0.000	7 minutes

5.3 Blind Test Results

Before evaluating the 2019 test data with the best model from the cross-validation step, we retrained the logistic regression model on the 2018 data. Using 2018 data for training allowed us to get model weights for our model that more closely resemble 2019 data than we otherwise would have had if we retrained the logistic regression model trained on 2015 to 2017 data. Once our logisitic model was trained on 2018, we used it to get predictions for 2019 flights and evaluated the model's performance. The metrics from the evaluation are shown in the table below.

Table 18: Best Model Performance on 2019 Test Set

Data	AUC	Precision	Recall	F1	Time
2019	0.659	0.628	0.548	0.585	8.6 minutes

The test results show that the model achieved a F1 score of 0.585. This score is slightly lower than the cross-validation training results, which suggests that we are not overfitting. Compared to cross-validation scores, we also observed that the recall dropped while the AUC and the precision were slightly higher. The model's precision (0.628) being slightly higher than recall (0.548) means that our model is somewhat less likely to be correct when it predicts that a flight is delayed than when it predicts a flight is on-time. More specifically, the recall means that of all the flight delays in the 2019 test set, our model correctly identified 54.8 percent of them while the precision means that of all predictions the model made, 62.8 percent of them correctly identified a delayed flight.

While improvements to the model have led to substantial improvements in its ability to predict flight delays, our model needs further refinement in order to be put to practical use in the real world. One obstacle to better performance may be that the set of features we selected are not the most representative for predicting flight delays. As we have mentioned previously, there are many more reasons for flight delays beyond what the datasets covered here (e.g., weather factors two hours before departure and flight performance at the same origin/destination).

To understand how we could improve our model in the future, we conducted the gap analysis in the following section where we drill down into the missed flight delays.

5.4 Gap Analysis

Using the logistic regression with elastic regularization which we chose for its superior performance, we used 2018 data to predict the 2019 test set. Below is the confusion matrix we observed on the test set. As we can see here, we still have more false negatives than we would like to.

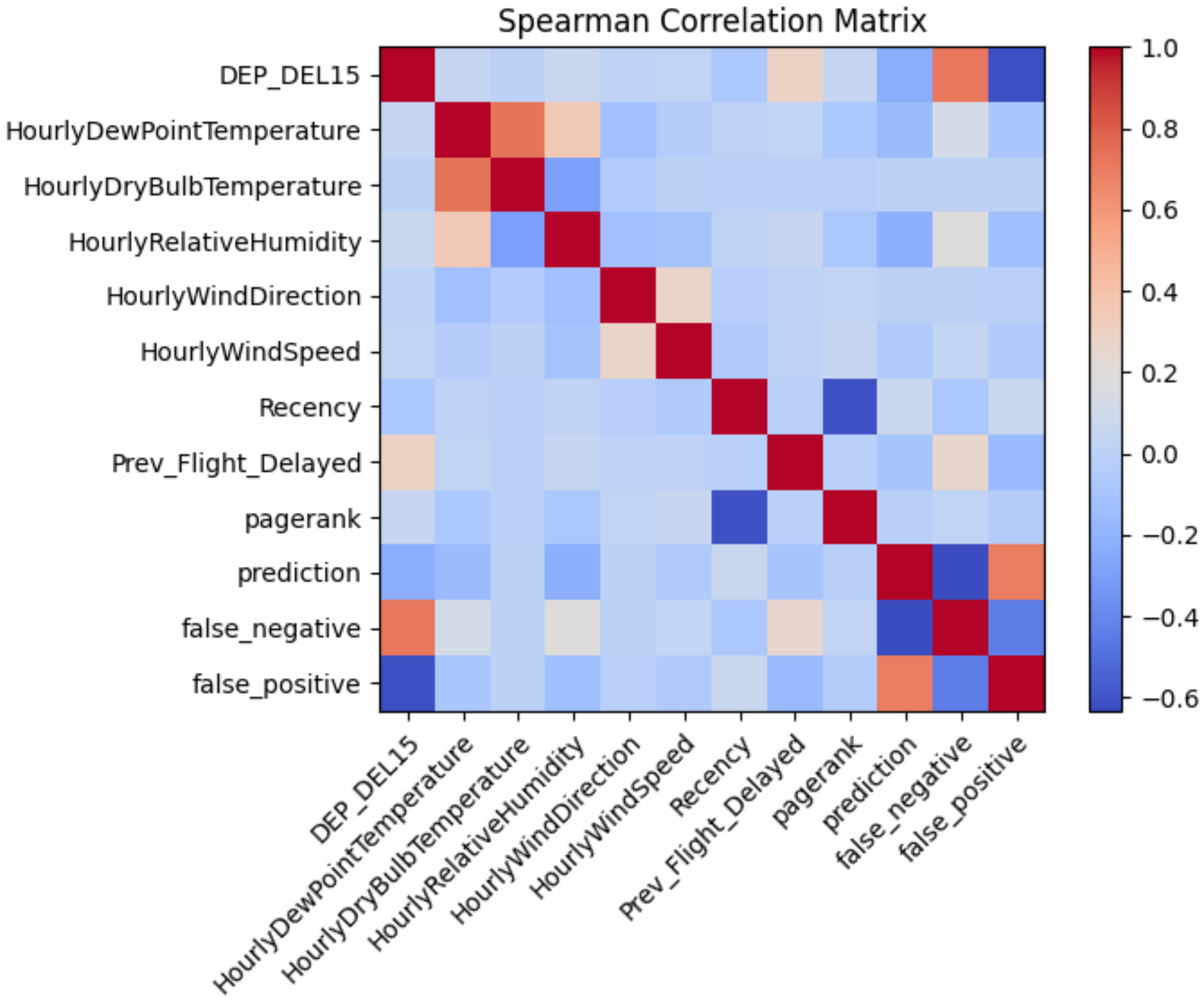
Table 19: Test data confusion matrix

	Predicted Delay	**Predicted Not Delay**
Actual Delay	364470	786852
Actual Not Delay	615793	508021

In order to look at which variable(s) could be the cause of the false predictions, we created another Spearman correlation matrix to see which variables correlate the most with the false positives and the false negatives. We realized that the Previous Flight Delay variable is the most positively correlated with the false negatives and most negatively correlated with the false positives. That means if more previous flights are delayed, the model relies on this fact so much that it will overlook the fact that most flights are not delayed. Somehow the model puts too much weight on this variable.

One way we can adjust this is to perhaps use an interaction term between this variable and another one, for example, some weather indicators. We also should have better performance if we were able to get an ensemble model to work, where there are different models that could help with the prediction, and that can alleviate one model's dictation on the results.

Figure 14. Spearman Correlations of Test Results


Spearman Correlation Matrix
DEP_DEL15
HourlyDewPointTemperature
HourlyDryBulbTemperature
HourlyRelativeHumidity
HourlyWindDirection
HourlyWindSpeed
Recency
Prev_Flight_Delayed
pagerank
prediction
false_negative
false_positive
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

6. Conclusion

The focus of the project is to predict flight delays of more than 15 minutes. Our aim is to help airlines better understand when a flight will be delayed and allow sufficient warnining for the airline to be able to arrange approriate accommodations for passengers if needed and permit the airline to manage costs better. In our EDA, we find that the majority of flights are on-time but around 20 percent of flights are delayed. We also discovered and cleaned a number of variables that were either irrelevant to our task or missed a high number of values in variables. Through hyperparameter tuning and cross-validation we arrived at our final model, a logistic regression with elastic net regularization, demonstrated a promising balance of precision and recall during cross-validation, achieving an F1 score of 0.646 on the 2018 validation set. Notably, the enhanced model's performance was not significantly superior to our baseline logistic regression model during cross-validation, indicating the efficacy of simpler models. The blind test on the 2019 test data further validated our model's efficacy, with an F1 score of 0.585, confirming its improved ability to predict real-world flight delays. Nevertheless, while iterating through hyperparameter tuning and cross-validation have helped to improve our model, the model still needs further refinement in order to be useful for customers and airlines, as slightly under half of all flight delays would be missed and under half of all predictions of delays would be incorrect.

Looking ahead, we have identified several promising avenues for further enhancing our model's performance including implementing a Min-Max scaler for variables that are not normally distributed, developing an ensemble method of algorithms to benefit from their different strengths, and integeration of more granular data as well as more years of data.

7. Credit Assignment Plan (updated)

Task	**Phase**	**Est. Effort (per person-hr)**	**Task Owner(s)**	**Due Date**	**Status**
Set up team cloud storage	1	5	Samuel	11/4	Complete
Abstract	1	1	Hans	11/5	Complete
Data Description	1	4	Samuel, Hans	11/5	Complete

ML Algorithm and metrics	1	3	Samuel	11/5	Complete
ML Pipelines	1	3	Aditya	11/5	Complete
Phase Leader Plan	1	1	Sophia	11/5	Complete
Credit Assignment Plan	1	2	Sophia	11/5	Complete
Gantt Chart	1	1	Aditya	11/5	Complete
Report Submission	1	1	Samuel	11/5	Complete
EDA on raw Data	2	5	Sophia,Hans	11/21	Complete
EDA on OTPW Data	2	3	Samuel, Aditya	11/21	Complete
Address missing data and non-numerical features	2	2	Samuel,Hans	11/21	Complete
Baseline pipelines creation + Cross Validation	2	6	Samuel,Aditya	11/26	Complete
Feature Engineering	2	2	Hans	12/3	Complete
Run experiments on ALL available data	2	5	Aditya	12/3	Complete
Fine-tuning the baseline and create new predictive features	2	4	Hans	12/3	Complete
Report Submission	2	1	Samuel	12/3	Complete
Explore advanced models	3	3	Aditya	12/12	Complete
EDA on the created features	3	2	Sophia	12/14	Complete
Pipeline Block Diagram	3	1	Sophia	12/14	Complete
Add on to the baseline and experiment	3	3	Hans	12/12	Complete
Hyperparameter tuning using cross-validation	3	4	Hans	12/12	Complete
Feature engineering and refinement	3	3	Samuel	12/12	Complete

Gap Analysis on the best pipeline	3	3	Sophia	12/14	Complete
Slides template for presentation	3	2	Sophia	12/14	Complete
Report Submission	3	1	Aditya	12/16	Complete

Note: Extra credit tasks are not included in this plan for now.

Appendix

Links to the code scripts

1. Raw Data Exploratory Analysis (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3865911540703826/command/546615846962764)
2. Joined Data (OTPW) Exploratory Analysis and Clean-up (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1537154891794290/command/1962809137470714)
3. General Data Cleaning, Feature Selection, and Pipeline (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1537154891794290/command/1962809137470714)
4. Data Processing - Imputation using Moving Averages (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/546615846962768/command/1962809137470420)
5. Baseline Experiments (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1537154891794290/command/1962809137470714)
6. Feature Engineering (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/546615846962768/command/1962809137470420)
7. EDA on New Features (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1012234209200834/command/1012234209200846
8. 3-Month Sample Data Code (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1346418319533003/command/1346418319533013)
9. 5-Year Full Data Experiment Code (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1012234209191218/command/1012234209191219)

References

1. INVESTIGATING THE COSTS AND ECONOMIC IMPACT OF FLIGHT DELAYS IN THE AVIATION INDUSTRY AND THE POTENTIAL STRATEGIES FOR REDUCTION (https://scholarworks.lib.csusb.edu/cgi/viewcontent.cgi?article=2885&context=etd)

2. Flights Data (https://www.transtats.bts.gov/Fields.asp?gnoyr_VQ=FGJ)

3. Weather Data (https://www.ncei.noaa.gov/access/metadata/landing-page/bin/iso?id=gov.noaa.ncdc:C00679)

4. Blocked Timeseries split https://hub.packtpub.com/cross-validation-strategies-for-time-series-forecasting-tutorial/ (https://hub.packtpub.com/cross-validation-strategies-for-time-series-forecasting-tutorial/)