databricks w261_final_project_Phase3_Submission_Team6-1_MASTER

(https://databricks.com)

Taming Turbulence with Machine Learning

Team: 6-1

Phase 3 leaders:

Phase	Week	Dates	Leader	Github
Phase 3	1	4/8 - 4/15	Dhyuti Ramadas	@dramadas
Phase 3	2	4/16 - 4/21	Rachel Gao	@rachelgaoMIDS

Team members

Name	Email	Photo
Juliana Gómez Consuegra	julianagc@berkeley.edu (mailto:julianagc@berkeley.edu)	
Ray Cao	caopuzheng@berkeley.edu (mailto:caopuzheng@berkeley.edu)	
Jenna Sparks	jenna_sparks@berkeley.edu (mailto:jenna_sparks@berkeley.edu)	
Dhyuti Ramadas	dramadas@berkeley.edu (mailto:dramadas@berkeley.edu)	

Name	Email	Photo

Credit assignment plan

Teammate	Start date	End date	Task	Hours
Juliana Gómez Consuegra	04/08/24	04/10/24	Prepared natural disaster and events dataframes (2016-2019)	9
	04/11/24	04/11/24	Sourced new data for 2020-2023 weather and flight datasets	9
	04/11/24	04/12/24	Created new parquet file for 2020-2023 weather data	2
	04/13/24	04/13/24	Created new parquet file for 2020-2023 flights data	2
	04/15/24	04/18/24	Performed gap analysis for best performing model	20
	04/15/24	04/20/24	Wrote additional Gap Analisis, Data Sourcing, Variable Dictionary and Notebook links parts of this report	7
Ray Cao	04/08/24	04/09/24	Conducted error analysis and results audit to identify untrustworth high metrics	5
	04/09/24	04/13/24	Incorporated all new features and ideas into the model pipeline	20
	04/13/24	04/14/24	Finished the hyper parameter tuning protocol type on Random Forest	20
	04/13/24	04/14/24	Experimented on the ensemble model, data preparation and aggregation for it	20
	04/15/24	04/15/24	Ensemble model running and validation	20

Teammate	Start date	End date	Task	Hours
Jenna Sparks	04/08/24	04/11/24	Previous flight features	5
	04/08/24	04/20/24	Phase 3 Report writing	7
	04/10/24	04/10/24	Prepared natural disaster and events dataframes (2015)	1
	04/10/24	04/16/24	XG Boost Model	8
	04/16/24	04/17/24	Cost Structure and Analysis	4
	04/15/24	04/17/24	Graphs and tables to summarize models	6
	04/10/24	04/19/24	Presentation design, writing	10
Dhyuti Ramadas	04/08/24	04/09/24	Prepared time series feature using Prophet	6
	04/12/24	04/17/24	Multilayer Perceptron Classifier implementation and tuning	20
	04/12/24	04/20/24	Wrote about modeling descriptions, Results, conclusion, Prophet	10
Rachel Gao	04/08/24	04/11/24	Prepared and compiled all features	20
	04/10/24	04/12/24	EDA on 5 year raw data and preprocessed data	10
	04/12/24	04/15/24	Joined raw datasets to create self-joined datasets using left join	20
	04/15/24	04/16/24	EDA on self-joined datasets	10

Abstract

This project predicts flight departure delays of 15 minutes or more at least 2 hours before the scheduled departure time to improve airport customer satisfaction. We leveraged the 5-year (2015-2019) On-Time Performance and Weather (OTPW) dataset that included flight, airport, and weather data for performing Exploratory Data Analyzes (EDA) and feature selection/engineering, data pipeline construction, and modeling. We experimented with various machine learning models to determine the best predictive results. We engineered features such as previous flight features, weather features, and time-dependent features, and built a modeling pipeline that could be used to predict on unseen data. We experimented with numerous models: linear regression (LR), random forest (RF), gradient-boosted random forest (GBRF), and Multilayer Perceptron Classifier (MLP). We evaluated the models with regression metrics and classification metrics. Our main metric was Fβ (β=0.5). Fβ combines precision and recall performance, considering both true delays (precision) and delays in all departures (recall). Our best (most comprehensive) model was Ensemble via MLP with an Fβ of 0.45 coupled with our best pipeline built in ML flow, taking the preprocessed dataset as input and model results as output. Included in the pipeline were feature normalization, training, and cross-validation. We also considered future works to increase value for our stakeholders.

Hypothesis: Our project hypothesis is that given our data, our best model will achieve Fβ better than 0.24, which is the Fβ of our baseline model.

Stakeholder: Airports interested in increasing customer satisfaction.

Data & Feature Engineering

Descriptions of all variables mentioned in this document can be found in Apprendix 3: Feature Dictionary.

Phase 1 Recap

During phase 1 of the project, we performed a series of exploratory data analysis on the 3-Months OTPW dataset to obtain an understanding of the dataset and the task at hand. Below is a summary of the data description and EDA findings from phase 1.

3-Months OTPW Dataset: Data Description

The 2015 3-months OTPW data contains 1.4M records of various flight information within the United States (US) and the US territories for the first 3 months of 2015. It contains 216 unique columns, consisting of 64 flight data columns, 28 airport and weather stations data columns, and 124 weather data columns, sourced from the respective raw datasets.

- Of the 64 flight variables included in the 2015 3-months OTPW dataset, only 27 columns contained null values. The variables contained information about departure and arrival times, destinations, delays, flight times, and causes of delays.
- Of the 28 airport and weather station variables included in the 2015 3-months OTPW dataset, none contained null values. The variables contained information on the origin and destination airports, information on the origin and destination weather stations, and the distance between the airport and the nearest weather station, from where the weather information is sourced.
- Of the 124 weather variables included in the 2015 3-months OTPW dataset, 66 were fully composed of null values. Out of the remaining variables, 24 variables contained nulls in 80% or more of the row. The variables contained various weather measurements for the weather stations based on the weather observed 4-hours prior to the scheduled departure time in Coordinated Universal Time (UTC).

3-Months OTPW Dataset: EDA Findings

- We visualized the relationship between the outcome variable `DEP_DELAY` (departure delay in minutes) and a few categorical variables in box plots and visualized the relationship between the outcome variable `DEP_DELAY` and a number of numerical weather variables in scatter plots and noted that feature

engineering is needed to derive meaningful relationships between these variables.
- We visualized the relationship between the outcome variable `DEP_DELAY` and two ordinal variables, `DAY_OF_MONTH` and `DAY_OF_WEEK` in line plots, and noted a cyclical trend in flight delays each week.
- We noted the majority of airports have weather stations on-site, with a handful of airports having weather stations within 10 miles. Few airports do not have nearby weather stations to provide reliable weather information, and we will drop these airports from our analysis.
- We observed numerous weather variables with high colineality where further analysis is needed to either remove duplication or to create compound variables using Principle Components Analysis (PCA). We also observed numerous

Phase 2 Recap

Once we obtained an understanding of the dataset and the task at hand using the 3-Months dataset from phase 1, we moved on to the 2015 1-Year raw and OTPW datasets to prepare for modeling. Below is a summary of the data description, feature selection, EDA findings, and feature engineering from phase 2.

1-Year OTPW Dataset: Data Description

The 1-Year OTPW data is a joined dataset of flights, weather, weather stations, and airports raw data. It contains 216 unique columns, consisting of 64 flight data columns, 28 airport and weather stations data columns, and 124 weather data columns, sourced from the respective raw datasets. The joined data consists of 5.8M records of various flight information for the whole year of 2015, of which the data from the first three quarters (4.4M records) are used for training while the data from the last quarter (1.4M records) is used for testing.

1-Year OTPW Dataset: Feature Selection

- We identified 35 columns from the OTPW datasets which we believe are important features we should use for detailed exploratory data analysis, feature engineering, and modeling. Of these columns, 3 columns are used to filter flights irrelevant for our task (refer to the phase 2 report (https://adb-

4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2513299033247568/command/2513299033 for a detailed analysis on what constitutes irrelevant flights), the remaining columns were used for EDA, feature engineering, and modeling. There are 2 outcome variable (departure delay in minutes and in binary), 4 datetime, 9 nominal, 6 ordinal, and 11 numerical feature variables.

- We noted that around 38% were delayed while the rest 62% were not delayed, posing a class imbalance in our dataset. Of the delayed flights, the majority (52%) of the delays were delayed for less than 15 minutes. Flights delayed for more than 15 minutes only account for 18% (773K records) of the training set. Due to the high skewness of the outcome variable, we applied a natural log transformation `log1p` to the outcome variable `DEP_DELAY` where flights not delayed were treated as 0 and flights delayed kept as is. log1p adds a 1 to each value in the log transformation, eliminating the issue of undefined log(0).

1-Year OTPW Dataset: EDA Findings & Feature Engineering

- We visualized the airport and weather station geographical distributions in our dataset and noted that almost all airports have a nearby weather station.
- We visualized the Pearson correlation between the numerical features and our log transformed outcome variable and identified features with highly correlation which we then used Principle Components Analysis (PCA) to create one composite feature per pair or trio of related variables, explaining more than 90% of the variance between the highly correlated features.
- We visualized the trend between the datetime features and our log transformed outcome variable in line plots and identified September having the least amount of average delays with June having the most, we believe there is some seasonality trend in our dataset, so we then used Prophet to create time-based features to better represent the trend and seasonality of our dataset.
- We visualized the relationship between the categorical features and our log transformed outcome variable in box plots and identified `HA` carrier (Hawaiian Airlines) as the carrier with the least amount of delays, smaller airports having more delays than larger ones, and days with disasters having more delays than those without disasters. We also observed spikes in weather variables which correlates with certain natural disasters.

- We also identified a number of weather features (such as the cloud related variables) requiring cleaning and potential for some weather variables to create composite variables which may be better suited for our task. Additionally, we used the `ORIGIN` and `DEST` airports as nodes and outlinks to create graph-based features (PageRank).

Raw Datasets: Data Description & EDA Findings

In addition to performing EDA on the 1-Year OTPW dataset, we also performed EDA on the raw flights, airports, weather stations, and weather datasets to gain a deeper understanding of the raw datasets and better informed of the data joining strategy (see Data Joining section).

- Flights: The 2015 1-year raw flights dataset, obtained from the Bureau of Transportation Statistics (https://www.transtats.bts.gov/Tables.asp?QO_VQ=EFD&QO_anzr=Nv4yv0r%FDb0-gvzr%FDcr4s14zn0pr%FDQn6n&QO_fu146_anzr=b0-gvzr), contains 14.8M records and a total of 64 variables detailing departure and arrival information for the flights across the United States (US) and US territories. This information includes origin and destination airport names, airline tail numbers, and indicators for flight cancellations or diversions, in addition to delay and on-time flight deatils.
- Airports: The raw airports dataset, obtained from Our Airports (https://ourairports.com/help/data-dictionary.html#countries), contains 78K records and 18 variables detailing information on airports from around the globe. This information includes the airport type, country, coordinates, and the airport International Air Transport Association (IATA) code in addition to other identification codes.
- Weather Stations: The raw weather stations dataset, obtained from National Oceanic and Atmospheric Administration (NOAA) (https://www.ncei.noaa.gov/access/metadata/landing-page/bin/iso?id=gov.noaa.ncdc:C00516), contains 5M records and 12 variables detailing weather stations information in the US and US territoritories. This information includes the weather station unique identification code, coordinates, and neighborhood information.

- Weather: The 2015 1-year raw weather dataset, also obtained from NOAA (https://www.ncei.noaa.gov/access/metadata/landing-page/bin/iso?id=gov.noaa.ncdc:C00516), contains 131.9M records and 124 variables, detailing weather information from weather stations around the world. This information

Phase 3

Data Description

From this point of the report onwards, we will be focused on phase 3 of our project, where we worked on the 5-Year OTPW dataset. The 5-Year OTPW dataset is a joined data of flights, weather, weather stations, and airports raw data, similar in format as the 3-months and 1-year OTPW datasets and from the same data sources as referenced above. It contains 214 unique columns, consisting of 63 flight data columns, 28 airport and weather stations data columns, and 123 weather data columns, sourced from the respective raw datasets. It consists of 31.6M records of various flight information for the whole year of 2015, 2016, 2017, 2018, and 2019, of which the data from the first 4-years (2015-2018 24.3M records) are used for training while the data from the last year (2019 7.4M records) is used for testing.

Data Preprocessing

We performed similar data preprocessing steps as phase 2 to clean and prepare our datasets for feature engineering and modeling. Below is a detailed summary of data preprocessing steps performed.

1. We kept the same initial list of 35 columns (as in phase 2) from the 5-Year OTPW dataset as a starting point for our feature engineering. We removed columns containing more than 50% null values as they cannot be reasonably imputed and we removed duplicate columns (such as `STATION` and `origin_station_id` which were used as join keys, in which case `origin_station_id` was kept), duplicate columns with similar information (such as detailed coordinates and state abbreviations, in which case state abbreviations were kept), and information not available 2-hours prior to departure (such as taxi time).

2. We then cast each column to the appropriate data type (integer, float, timestamp, or string) based on the documentation of the dataset.
3. We changed variable wind direction (VRB) to null for imputation.
4. We imputed for null values for all numerical variables based on daily, weekly, or monthly average by `origin_station_id`, as only weather related columns contained null values which required imputation. If a station did not have a daily average for the particular variable, then the weekly average was used. If a weekly average was also missing, then a monthly average was used. If a monthly average was still missing for that station, then the record was dropped, as we believe imputing average by quarter could create too much variability in the imputed data that may lead to noisy inputs in our models.
5. We dropped all remaining null values, including any instances with null values in categorical variables, as they cannot be reasonably imputed without creating noisy inputs in our models.
6. We then applied natural log transformation to the outcome variable `DEP_DELAY` to reduce the skewness of the variable, similar to what we did in phase 2.
7. Average null value for the 35 columns retained is less than 1%, with only weather variables containing null values. The null values in the `DEP_DELAY` and `DEP_DEL15` columns represent cancelled or diverted flights, or flights where the delay information is missing due to data quality issue. We removed these instances as predicting cancelled or diverted flights is not our target task. We also removed the instances where the delay information is missing (4.7K instances) due to data quality issue as we cannot reasonably impute the delay information without making assumptions about the outcome variable, potentially leading to noisy data.
8. We considered including cancelled flights in our prediction task by looking into the number of cancelled flights and found that cancelled flights represent 1.5% (or 354K records) of the 24.3M total training records in the span of 4 years. This poses a huge class-imbalance between cancelled and non-cancelled flights in our dataset. We further analyzed the reason for cancellation for these cancelled flights and found that more than half (55%) of the cancellations were related to weather, with another 26% relating to carrier, and 19% relating to National Air System with less than 500 relating to security (Source: Transtats (https://www.transtats.bts.gov/homepage.asp)). We believe it is likely that these cancelled flights were delayed at first and we understand the frustration of flight

cancellation for travellers, however, our dataset does not contain the neccessary information for us to gain a comprehensive understanding of the relationship between delay and cancellation. We believe this would be a valuable endeavor for future exploration once we are able to predict delays with precision and high confidence.

EDA & Feature Engineering

We performed EDA on the 5-Year OTPW training data to gain a better understanding of the data quality and identify feature engineering opportunities.

1. We identified a handful of instances where the `HourlyWindSpeed` is above 500 miles/hour, with some reaching as high as more than 2,000 miles/hour. We believe this is erroneous as it unlikely for the wind speed to reach that level on Earth surface. Therefore, we removed these records from our dataset.
2. We also identified some instances where the `CRS_ELAPSE_TIME` is below 0. We believe believe this is also erroneous as the scheduled elapse time (the amount of time between flight take-off and landing) is unlikely to be less than 0. Therefore, we removed these records from our dataset.
3. We further identified one airport without a nearby weather station, with the closest weather station being more than 2,000 miles away. We believe the weather information for this airport is unlikely to be reliable. Therefore, we removed these records from our dataset.

In the end, after dropping records with data quality issues and records with null values that we cannot reasonably impute, we retained 23.2M out of the 24.3M total training records for feature engineering and modeling. This is the `s1` dataset.

Listed below are the feature engineering steps we performed on the preprocessed datasets `s1` (both train and test) to create preprocessed datasets `s1_1`:

1. We created the `sched_depart_date_time` feature by concatenating `FL_DATE` and `CRS_DEP_TIME`. This feature was used to identify event dates and disaster dates for event and disaster flag features.
2. We created `sched_arrive_date_time_UTC` feature by adding the `CRS_ELAPSE_TIME` to the `sched_depart_date_time_UTC`. This feature was created for feature engineering use.

3. We identified event dates which consist of Federal holidays based on the US government website (https://www.usa.gov/holidays) and superbowl days and flagged the 5 days surrounding the event dates as `event_flag` features to account for travelling around event days.
4. We identified natural disaster types and dates based on three sources from NOAA: U.S. Billion-Dollar Weather & Climate Disasters 1980-2024 (https://www.ncei.noaa.gov/access/billions/events.pdf), Billion-Dollar Weather and Climate Disasters (https://www.ncei.noaa.gov/access/billions/events/US/1980-2017?disasters%5B%5D=all-disasters) and U.S. saw 10 billion-dollar disasters in 2015 (https://www.noaa.gov/news/us-saw-10-billion-dollar-disasters-in-2015) and flagged the disaster days based on the disaster type as `drought_flag`, `flooding_flag`, `freeze_flag`, `severe_storm_flag`, `tropical_cyclone_flag`, `wildfire_flag`, and `winter_storm_flag` to account for potential flight delays due to natural disasters in the states affected by those natural disasters.
5. We generated previous aircraft delay features to investigate the potential impact a delayed aircraft has on the next flight, flown by the same aircraft. We created features `last_delay` (the previous aircraft's delay in minutes, using plane `TAIL_NUM` as a unique identifier, in the 2-4 hour time window prior to `sched_depart_date_time_UTC`) and `incoming_flight_delay_ratio` (the ratio of the `last_delay` divided by the minutes between departures sharing the same aircraft). We also generated a feature, `log_average_delay` by finding the average delays for each airport and carrier combination for each day within the same 2-4 hour window, the average delays is log transformed to match the transformation performed on the main outcome variable.

Listed below are the feature engineering steps we performed on the raw training dataset:

1. We created the graph-based feature `pagerank` by using the `ORIGIN` airports as nodes and `DEST` airports as outlinks and the number of flights between each airport as weights. When creating this feature, we used GraphFrame to calculate the PageRank by year to match the cross-validation pattern for 2015-2018. To prevent data leakage, we calculated the average PageRank for all 4 years and

used the average PageRank as the `pagerank` feature for the test dataset. This process produced the graph-based feature dataset `s2_1a`.

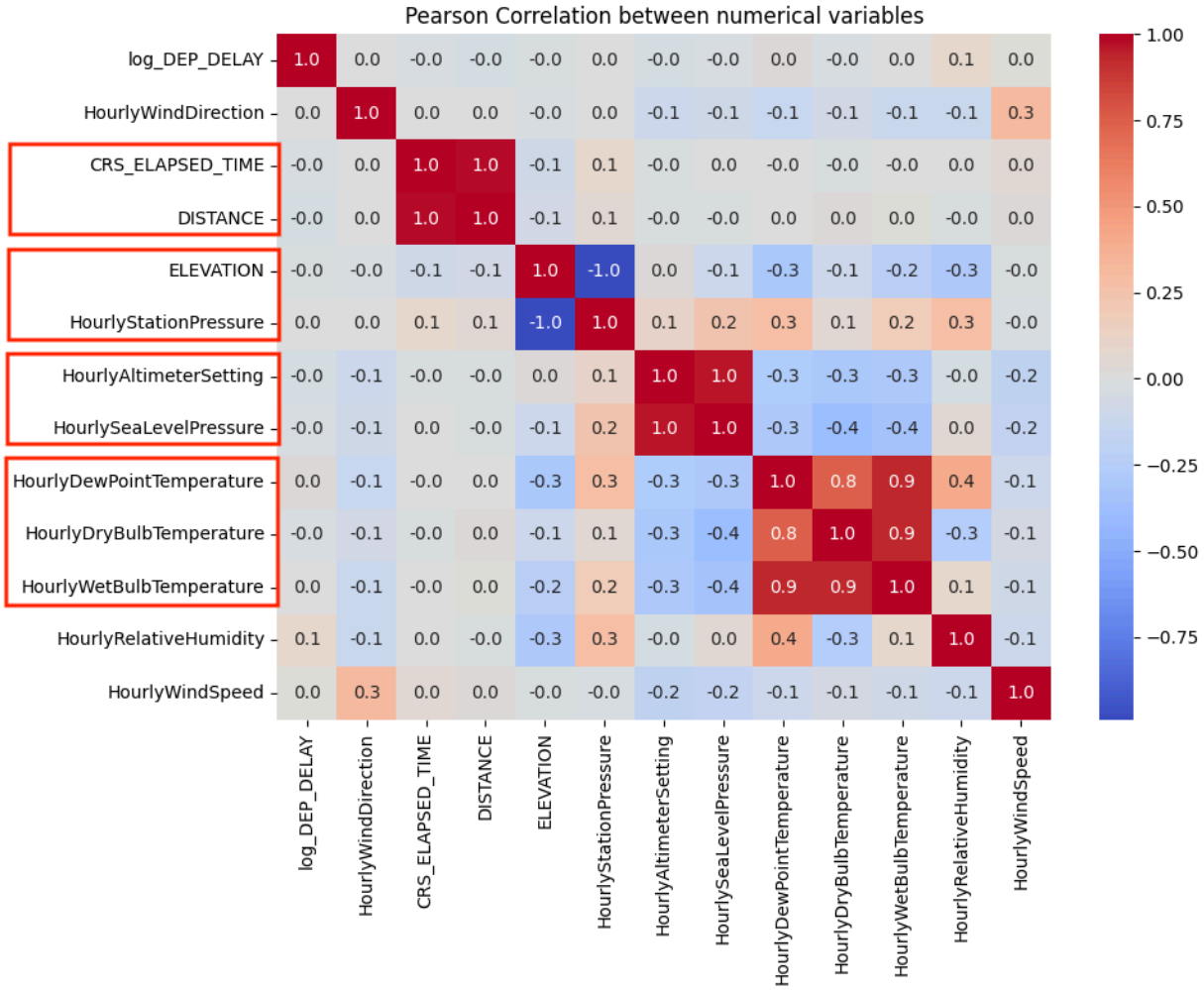
2. We created time-based features `trend` and `yhat` (`trend` + seasonality) by using the `ORIGIN` airports and `OP_UNIQUE_CARRIERS`. We first concatenated the `ORIGIN` airports and `OP_UNIQUE_CARRIERS` to create a composite `ORIG_CARRIER` feature for grouping the flights based on origin airports and carrier combinations. We used Prophet to forecast the daily and weekly `trend` and `yhat` and included US holiday seasonality. For combinations of carriers and origins that had at least 1 flight everyday, we forecasted daily predictions. For combinations that did not have 1 flight everyday but had at least 1 flight weekly, we forecasted weekly predictions. For all remaining combinations, we determined the frequency of flights to be too low to accurately forecast in the context of our experiments, therefore we imputed them as 0. To prevent data leakage, we used Prophet to forecast `trend` and `yhat` for 2019 based on the data from 2015-2018. This process produced the time-based feature dataset `s2_1b`.

Once the preprocessed dataset `s_1`, graph-based features `s2_1a` and time-based features `s2_1b` are ready, we performed the additional data processing and feature engineering steps described below to create the final dataset `s2_1` for both train and test sets.

1. We cleaned the `HourlySkyCondition` weather variable by decomposing it to create `SkyDarkness`, `CloudHeight`, and `CloudHeightandDarkness` features. `HourlySkyConditions` is a composite variable consisting of information for cloud layers 1, 2 and 3. For each layer, the overall condition of the sky is mentioned both as a category (ranging from CLR = clear skies to OVC = overcast) and as a number (ranging from 1 (CLR) to 9, with 9 signifying that the sky can't be seen because of smoke or fog, and 8 representing OVC). An additional number may or may not be included in each layer, and that is the cloud base height at the lowest point of the layer. Given that there is a redundancy in mentioning the overall conditions as letter (i.e. OVC) and numbers (i.e. 00), we removed the letters, and split the numbers into new variables called `SkyDarkness` and `CloudHeight`. This was done by selecting the numeric condition (1-9) corresponding to the highest cloud layer available, as that is the layer that best represents the overall conditions of the sky. Given that these two

variables are not independent, we concatenated them, and one-hot-encoded them into a variable called `CloudHeightandDarkness_encoded`.

2. We created a composite weather variable called `WindChill`, based on the formula $WindChill$ = 35.74 + 0.62115T – 35.75($V^{0.16}$) + 0.4275T($V^{0.16}$) mentioned in the data dictionary (https://www.ncei.noaa.gov/pub/data/cdo/documentation/LCD_documentation.pd from NOAA for the Local Climatological Data (LCD) weather dataset. In this formula, T represents `HourlyDryBulbTemperature` and V represents `HourlyWindSpeed`.
3. We identified a number of highly linearly correlated feature variables (see Pearson Correlation Matrix below) so we performed Principal Component Analysis (PCA) for each pairs or trios of variables in order to reduce the number of weather variables. Since `WindChill` is created using `HourlyDryBulbTemperature` as one of the inputs, `WindChill` was included in the PCA with `HourlyDryBulbTemperature`, `HourlyWetBulbTemperature`, and `HourlyDewPointTemperature`. In the end, four PCA features were created: `pca_time_distance` (from `CRS_ELAPSED_TIME` and `DISTANCE`), `pca_elevation_station_pressure` (from `ELEVATION` and `HourlyStationPressure`), `pca_altimeter_sea_level_pressure` (from `HourlyAltimeterSetting` and `HourlySeaLevelPressure`) and `pca_dew_windchill_wet_temp` (from `HourlyWetBulbTemperature`, `WindChill`, `HourlyDewPointTemperature` and `HourlySeaLevelPressure`) .



4. We joined the graph-based feature `pagerank` to the rest of the dataset on `ORIGIN` and `YEAR`, and joined the time-based feature `trend` and `yhat` (`trend` + seasonality) to the rest of the dataset on `ORIGIN`, `OP_UNIQUE_CARRIER`, and `FL_DATE_UTC` (by casting `sched_depart_date_time_UTC` to date type). Any records without these features are imputed with 0.

In the end, after all data preprocessing, data cleaning, and feature engineering, the final training set `s2_1` contains 56 columns and 23.2M rows and the final test set `s2_1` contains 56 columns and 7M rows. The 56 columns include the original 35

columns retained from the OTPW dataset and 21 engineered features.

Variable Name	Data Type	Source	Purpose
DEP_DELAY	float	OTPW	To create log_DEP_DELAY
log_DEP_DELAY	float	Engineered	Target variable for linear regression
DEP_DEL15	boolean	OTPW	Target Variable for final classification
sched_depart_date_time	timestamp	Engineered	To create event_flag & disaster flags
sched_depart_date_time_UTC	timestamp	OTPW	To create sched_arrive_date_time_UTC
two_hours_prior_depart_UTC	timestamp	OTPW	For visualization
four_hours_prior_depart_UTC	timestamp	OTPW	For visualization
sched_arrive_date_time_UTC	timestamp	Engineered	For feature engineering
YEAR	integer	OTPW	For visualization
QUARTER	integer	OTPW	For visualization
MONTH	integer	OTPW	For visualization
DAY_OF_MONTH	integer	OTPW	Feature variable
DAY_OF_WEEK	integer	OTPW	Feature variable

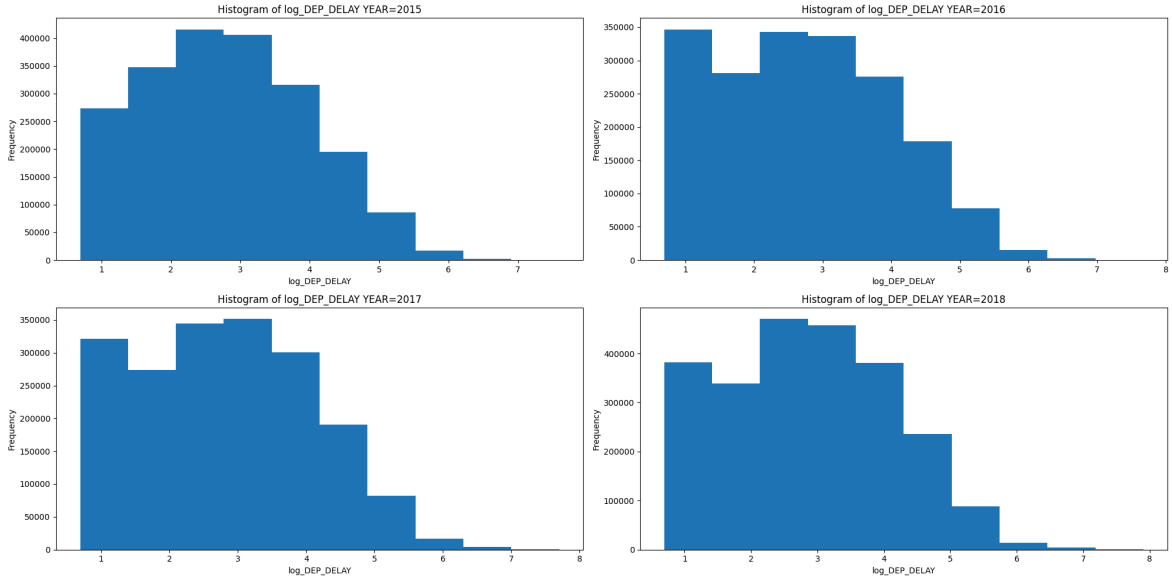
Legend:
Target Variable
Datetime
Categorical
Numerical

Variable Name	Data Type	Raw/Engineered	Purpose
OP_UNIQUE_CARRIER	string	OTPW	Feature variable
TAIL_NUM	string	OTPW	Feature variable
OP_CARRIER_FL_NUM	string	OTPW	Feature variable
ORIGIN	string	OTPW	Feature variable
DEST	string	OTPW	Feature variable
origin_type	string	OTPW	Feature variable
origin_region	string	OTPW	Feature variable
origin_station_id	string	OTPW	Feature variable
HourlySkyConditions	string	OTPW	Feature variable
SkyDarkness	string	Engineered	Feature variable
CloudHeight	string	Engineered	Feature variable
CloudHeightandDarkness	string	Engineered	Feature variable
event_flag	boolean	Engineered	Feature variable
drought_flag	boolean	Engineered	Feature variable
flooding_flag	boolean	Engineered	Feature variable
freeze_flag	boolean	Engineered	Feature variable
severe_storm_flag	boolean	Engineered	Feature variable
tropical_cyclone_flag	boolean	Engineered	Feature variable
wildfire_flag	boolean	Engineered	Feature variable
winter_storm_flag	boolean	Engineered	Feature variable

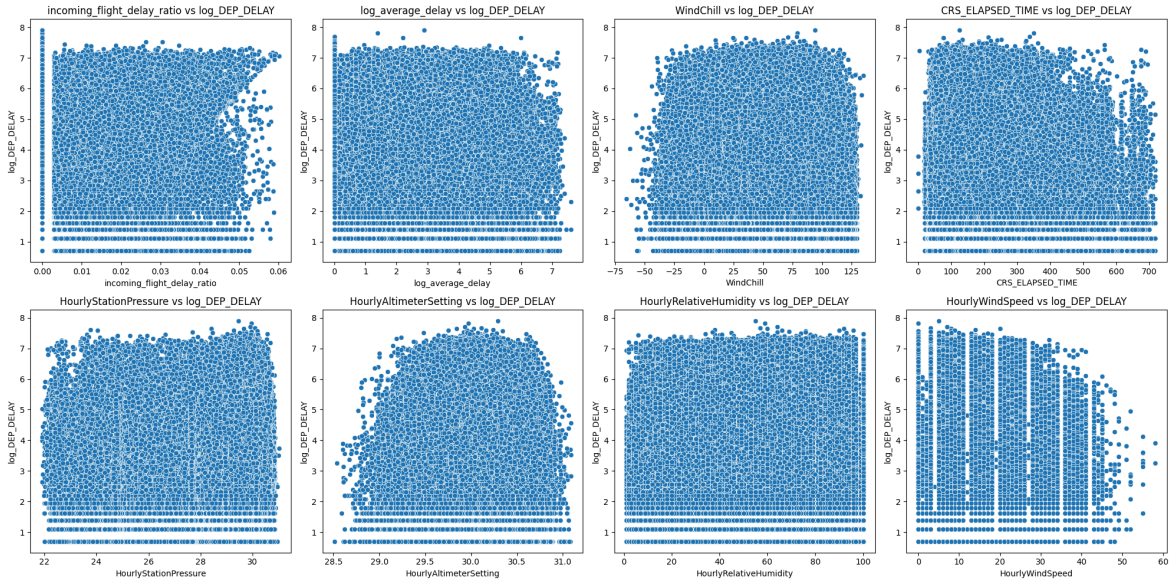
Variable Name	Data Type	Raw/Engineered	Purpose
HourlyWindDirection	integer	OTPW	Feature variable
CRS_ELAPSED_TIME	integer	OTPW	Feature variable
DISTANCE	float	OTPW	Feature variable
ELEVATION	float	OTPW	Feature variable
HourlyStationPressure	float	OTPW	Feature variable
HourlyAltimeterSetting	float	OTPW	Feature variable
HourlySeaLevelPressure	float	OTPW	Feature variable
HourlyDewPointTemperature	float	OTPW	Feature variable
HourlyDryBulbTemperature	float	OTPW	Feature variable
HourlyWetBulbTemperature	float	OTPW	Feature variable
HourlyRelativeHumidity	float	OTPW	Feature variable
HourlyWindSpeed	float	OTPW	Feature variable
last_delay	float	Engineered	Feature variable
incoming_flight_delay_ratio	float	Engineered	Feature variable
log_average_delay	float	Engineered	Feature variable
WindChill	float	Engineered	Feature variable
pca_time_distance	float	Engineered	Feature variable
pca_elevation_station_pressure	float	Engineered	Feature variable
pca_altimeter_sea_level_pressure	float	Engineered	Feature variable
pca_dew_windchill_wet_temp	float	Engineered	Feature variable
pagerank	float	Engineered	Feature variable
yhat	float	Engineered	Feature variable
trend	float	Engineered	Feature variable

We then performed a series of EDA to gain a batter understanding of the relationship between each feature variable and our outcome variable `log_DEP_DELAY`. As PCA features are in vector format and cannot be extracted out for visualization purpose, we omitted the PCA features in the plots.

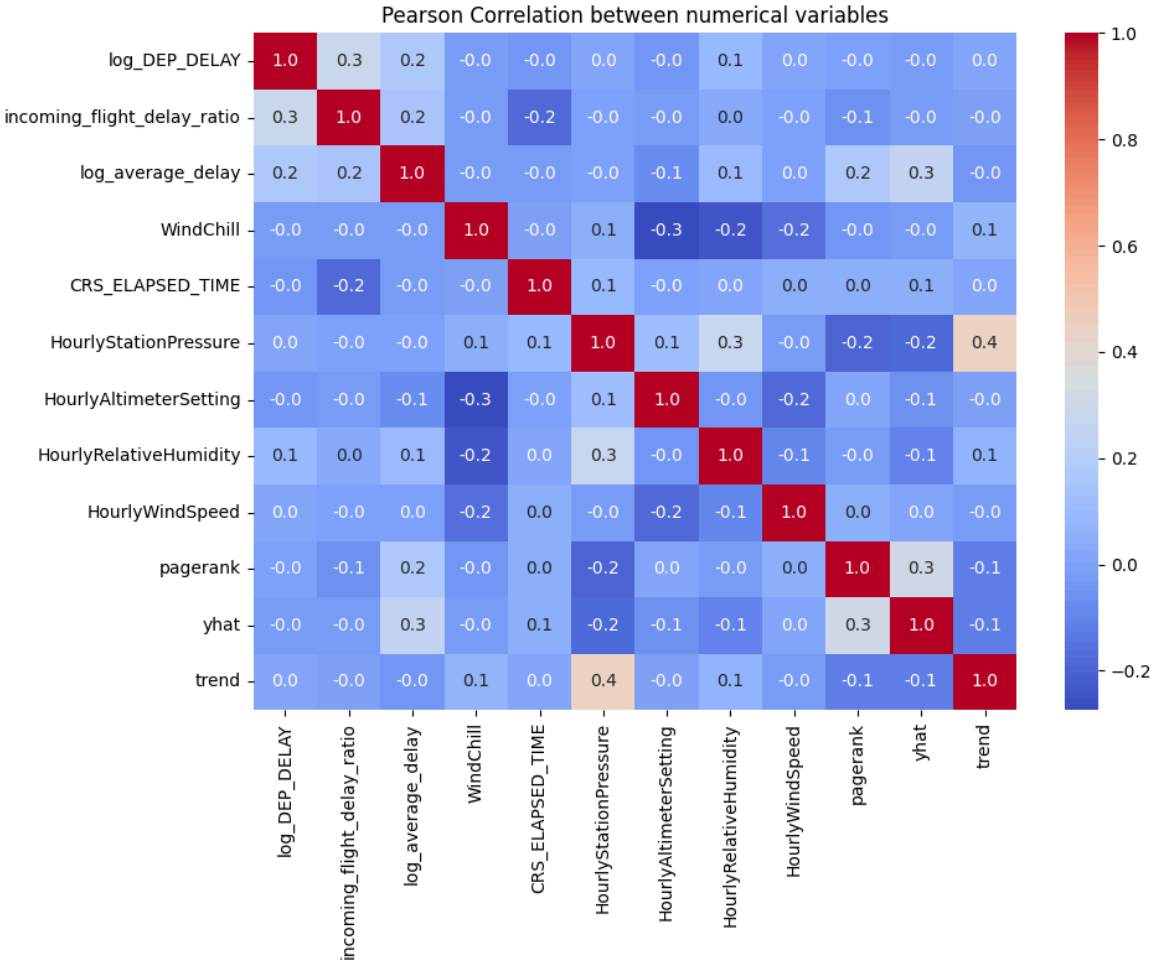
- We visualized the histogram of our outcome variable `log_DEP_DELAY` for each of the training years 2015-2018 and noted the distribution is mostly normal and identical between the years.



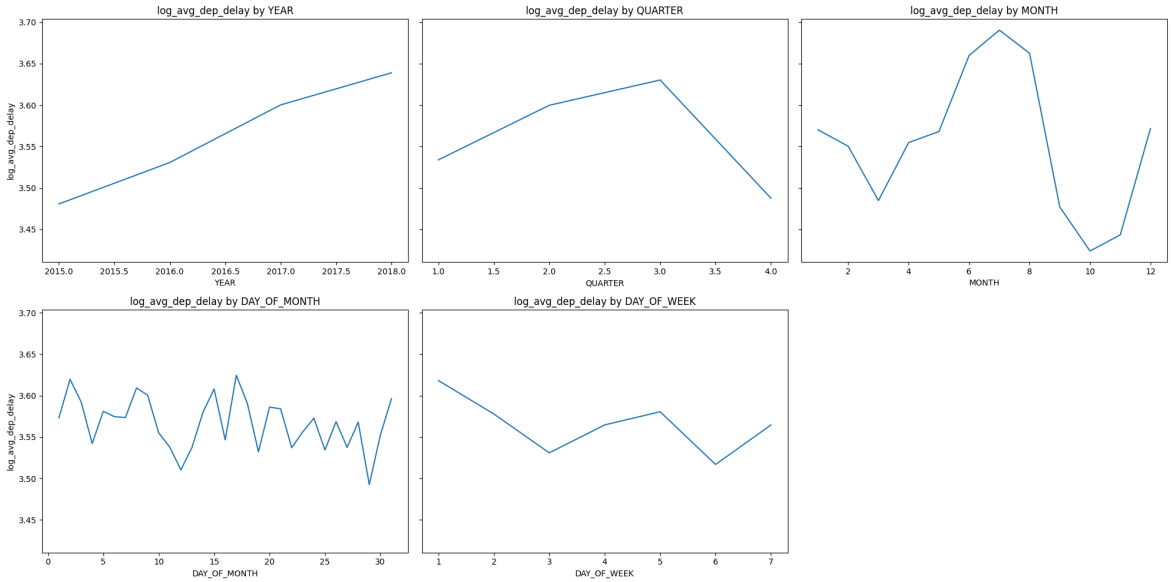
- We visualized the scatter plot of numerical features with our outcome variable `log_DEP_DELAY` and noted no clear linear relationship among the variables and noted no extreme outliers.



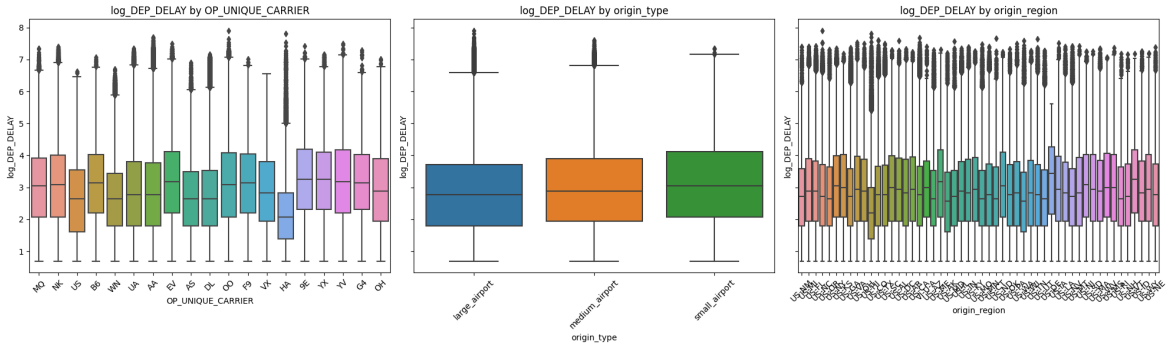
- We also visualized the Pearson correlation matrix of numerical features with our outcome variable `log_DEP_DELAY` and noted no strong correlation among the variables.



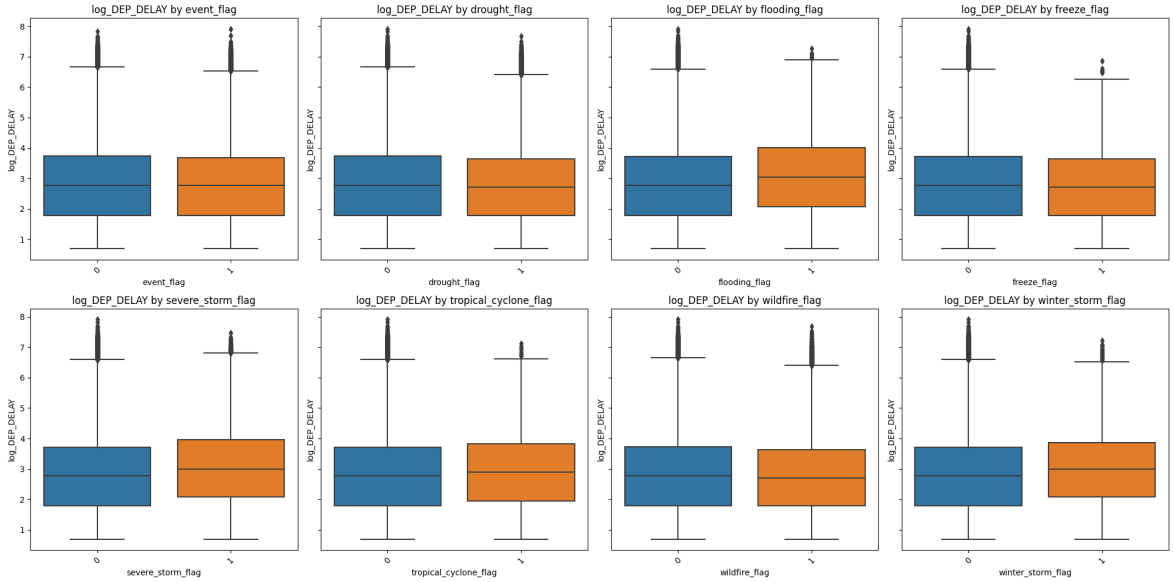
- We visualized the line plots of a few datetime features with our outcome variable `log_DEP_DELAY` by taking the average of the `log_DEP_DELAY` and noted some seasonality trends. Specifically, we noted that the delay seems to be on an increasing trend year after year, the 3rd quarters in a year have the longest delays whereas the 4th quarters have the shortest delays, delays peak during summer months and are the lowest in October, and Wednesdays and Saturdays have the shortest delays while Monday has the longest delays. These trends are somewhat consistent with the trend we observed in the 2015 1-year OTPW dataset as well.



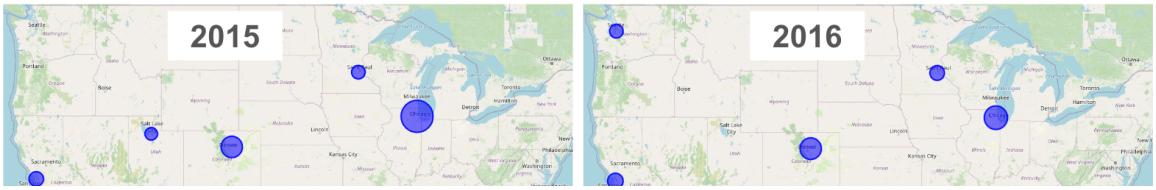
- We visualized the box plots of a few categorical features with our outcome variable `log_DEP_DELAY` and noted similar trends as observed in the 2015 1-year OTPW dataset in phase 2, where smaller airports appear to have longer delays than larger ones, and the Hawaiian Airlines (HA) carrier has the shortest delays among all carriers.



- We visualized the box plots of the flag categorical features with our outcome variable `log_DEP_DELAY` and noted there appear to be slightly more delays on event days than non-event days, although the difference appears minimal, and delays are more prominent during flood, storm, and cyclone related natural disasters, which is consistent with our expectations.



- We visualized the PageRank results by year and plotted the 10 airports with the highest PageRank for each year, with the circle size indicating the PageRank value. The larger the PageRank value is, the more connected an airport is. We noted that compared with 2015, in 2016, Seattle airport became more connected than Salt Lake City airport. In 2017, the Detroit airport became more connected than the Houston airport. In 2018, the Houston airport and the Charlotte airport became more connected than the Seattle airport and the Phoenix airport. In all years, Atlanta, Chicago, Dallas, and Denver airports are the most connected along with San Francisco and Los Angeles airports. The most connected airports are located in major US cities, which is consistent with our expectation.



Feature Refinement

After we performed all data preprocessing and feature engineering, we are left with 56 columns in total. In order to better understand the feature importance and their contribution to the modeling results, we utilized Linear Regression Absolute L1 Coefficient and Decision Tree Feature Importance to refine our feature selection. Based on the feature importance, we selected 32 columns (including 3 columns for outcome variable and evaluation purpose) for modeling. This includes 13 raw features and 16 engineered features.

When evaluating the feature importance, we extracted the Linear Regression Absolute L1 Coefficient and Decision Tree Feature Importance after passing all 56 features into the Linear Regression and the Decision Tree models, respectively. For categorical variables, we took the sum of all categories and then took the average of the two measurements to arrive at the average feature importance. For all other variables, we directly took the average of the two measurements to arrive at the average feature importance.

We analyzed the feature importance results closely and dropped some features due to redundency with other more important features, while prioritizing retaining PCA features, unless the raw features appear to be drastically more important than the PCA features.

- Among `HourlySkyConditions`, `SkyDarkness`, `CloudHeight`, and `CloudHeightandDarkness`, we retained only `CloudHeightandDarkness` as `SkyDarkness` and `CloudHeight` are decomposed from `HourlySkyConditions` and `CloudHeightandDarkness` is a composite feature with both `SkyDarkness` and `CloudHeight`.

- Among `CRS_ELAPSE_TIME`, `DISTANCE`, and `pca_time_distance`, we retained `CRS_ELAPSE_TIME` and `DISTANCE`, discarding `pca_time_distance` as the raw feature importance are much higher than the PCA feature. This is perhaps due to the PCA features not capturing the relationship the raw features had with the outcome variable.

- Among `HourlyAltimeterSetting`, `HourlySeaLevelPressure`, and `pca_altimeter_sea_level_pressure`, we retained only `HourlyAltimeterSetting`, as the feature importance of this raw feature is higher than the other two.

- Among `pca_dew_windchill_wet_temp`, `HourlyDewPointTemperature`, `HourlyDryBulbTemperature`, `HourlyWetBuldTemperature`, and `WindChill`, we retained only `pca_elevation_station_pressure`, to prioritize the PCA feature.

- Among `pca_elevation_station_pressure`, `ELEVATION`, and `HourlyStationPressure`, we retained only `pca_elevation_station_pressure` as none of the three features scored high on feature importance, so we prioritized the PCA feature.

Keep

Variable Name	Data Type	Variable	Source	Decision Tree Importance	L1 Coefficient	Average
DEP_DELAY	float	Target	Flight	-	-	-
log_DEP_DELAY	float	Target	Flight	-	-	-
DEP_DEL15	boolean	Target	Flight	-	-	-
last_delay	float	Engineered Feature	Flight	67.19%	2.05%	34.62%
ORIGIN	string	Raw Feature	Flight	<0.1%	32.32%	16.16%
DEST	string	Raw Feature	Flight	1.36%	30.43%	15.90%
log_average_delay	float	Engineered Feature	Flight	26.09%	1.70%	13.90%
OP_UNIQUE_CARRIER	string	Raw Feature	Flight	0.85%	7.59%	4.22%
origin_region	string	Raw Feature	Flight	<0.1%	6.02%	3.02%
MONTH	integer	Raw Feature	Flight	<0.1%	4.42%	2.21%
CloudHeightandDarkness	string	Engineered Feature	Weather	<0.1%	3.38%	1.69%
DAY_OF_MONTH	integer	Raw Feature	Flight	<0.1%	3.35%	1.67%
time_series_forecast	float	Engineered Feature	Flight	1.29%	<0.1%	0.65%
HourlyRelativeHumidity	float	Raw Feature	Weather	0.87%	0.41%	0.64%
CRS_ELAPSE_TIME	integer	Raw Feature	Flight	0.60%	0.47%	0.54%
DAY_OF_WEEK	integer	Raw Feature	Flight	<0.1%	0.90%	0.46%
incoming_flight_delay_ratio	float	Engineered Feature	Flight	0.12%	0.78%	0.45%
winter_storm_flag	boolean	Engineered Feature	Weather	<0.1%	0.67%	0.34%
DISTANCE	float	Raw Feature	Flight	0.56%	<0.1%	0.28%
severe_storm_flag	boolean	Engineered Feature	Weather	<0.1%	0.48%	0.24%
origin_type	string	Raw Feature	Flight	<0.1%	0.40%	0.20%
flooding_flag	boolean	Engineered Feature	Weather	<0.1%	0.34%	0.17%
pagerank	float	Engineered Feature	Flight	0.34%	<0.1%	0.17%
wildfire_flag	boolean	Engineered Feature	Weather	<0.1%	0.28%	0.14%
HourlyAltimeterSetting	float	Raw Feature	Weather	<0.1%	0.26%	0.13%
trend	float	Engineered Feature	Flight	0.26%	<0.1%	0.13%
tropical_cyclone_flag	boolean	Engineered Feature	Weather	<0.1%	0.22%	0.11%
drought_flag	boolean	Engineered Feature	Weather	<0.1%	0.17%	<0.1%
HourlyWindSpeed	float	Raw Feature	Weather	<0.1%	0.14%	<0.1%
event_flag	boolean	Engineered Feature	Weather	<0.1%	0.13%	<0.1%
pca_dew_windchill_wet_temp	float	Engineered Feature	Weather	<0.1%	<0.1%	<0.1%
pca_elevation_station_pressure	float	Engineered Feature	Weather	<0.1%	<0.1%	<0.1%

Discarded

Variable Name	Data Type	Source	Decision Tree Importance	L1 Coefficient	Average
sched_depart_date_time	timestamp	Flight	-	-	-
sched_depart_date_time_UTC	timestamp	Flight	-	-	-
two_hours_prior_depart_UTC	timestamp	Flight	-	-	-
four_hours_prior_depart_UTC	timestamp	Flight	-	-	-
sched_arrive_date_time_UTC	timestamp	Flight	-	-	-
YEAR	integer	Flight	-	-	-
QUARTER	integer	Flight	-	-	-
TAIL_NUM	string	Flight	-	-	-
OP_CARRIER_FL_NUM	string	Flight	-	-	-
origin_station_id	string	Weather	-	-	-
HourlySkyConditions	string	Weather	-	-	-
SkyDarkness	string	Weather	<0.1%	1.51%	0.76%
CloudHeight	string	Weather	<0.1%	1.39%	0.70%
HourlyDewPointTemperature	float	Weather	0.16%	0.10%	0.13%
HourlyDryBulbTemperature	float	Weather	<0.1%	<0.1%	<0.1%
HourlyWetBulbTemperature	float	Weather	<0.1%	<0.1%	<0.1%
WindChill	float	Weather	<0.1%	<0.1%	<0.1%
pca_time_distance	float	Flight	0.11%	<0.1%	<0.1%
HourlySeaLevelPressure	float	Weather	<0.1%	<0.1%	<0.1%
pca_altimeter_sea_level_pressure	float	Weather	<0.1%	<0.1%	<0.1%
ELEVATION	float	Weather	<0.1%	<0.1%	<0.1%
HourlyStationPressure	float	Weather	<0.1%	<0.1%	<0.1%
HourlyWindDirection	integer	Weather	<0.1%	<0.1%	<0.1%
freeze_flag	boolean	Weather	<0.1%	<0.1%	<0.1%

It's interesting to note that pagerank and time series based forecastings got

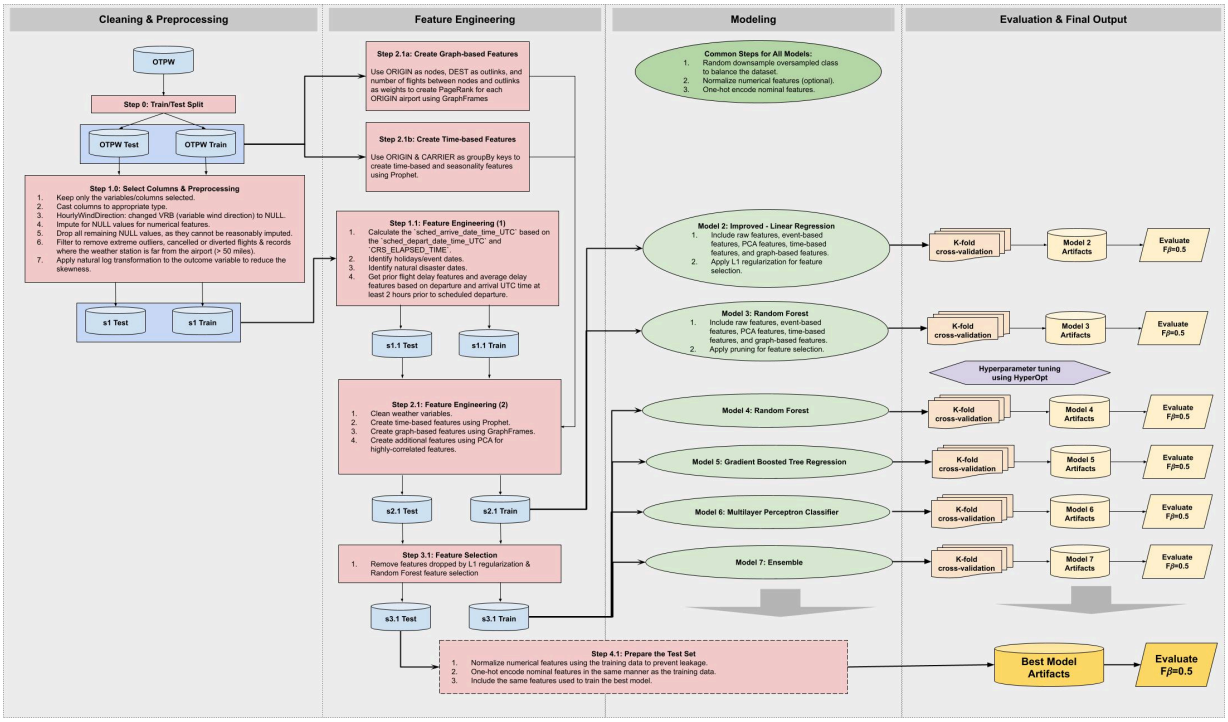
Modeling Pipelines

Data workflow block diagram

Below is an overview of the modeling pipeline. The cylinder shapes indicate checkpoints of datasets or model artifacts. Model artifacts include both traditional model artifacts such as weights and model structure, and train/validation/test prediction results which were used to create the ensemble models.

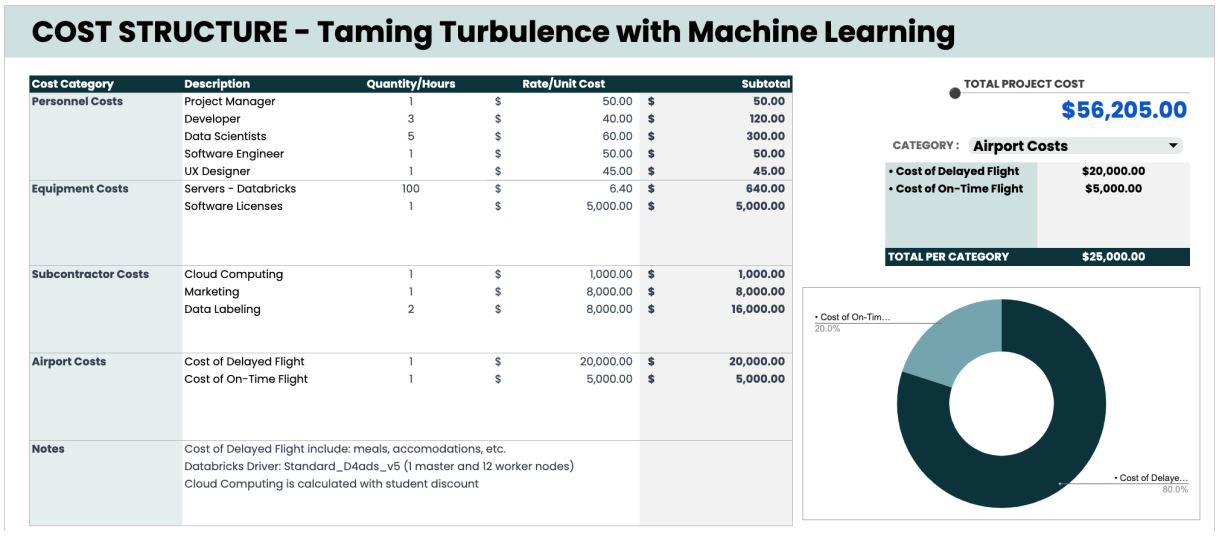
From the top left corner, we took the dataset and split it into train/test sets where the data from the first 4 years are used for train and the ones from the last year used for test. We performed a series of data preprocessing and cleaning (see phase 2 recap above) to generate the `s1` datasets. During phase 2, we trained our baseline model using only the raw features output from `s1` datasets. In phase 3, we then used the preprocessed and cleaned `s1` dataset to perform series of feature engineering which produced the `s1.1` datasets. We also used the train/test sets to generate the Graph-based features (PageRank) and Time-based features (Prophet forecast), which are the `s2.1a` and `s2.1b` datasets respectively. We then joined the `s1.1` dataset with the `s2.1a` and `s2.1b` datasets and performed additional feature engineering to generate the `s2.1` datasets which we used to train our improved linear regression model and our random forest model.

After the improved linear regression model and random forest model has been trained, we utilized regularization techniques to select features contributing the most to the models based on feature importance. This process helped us select our final sets of features in the `s3.1` datasets, which was used to train all our models before evaluating our final model result on the held-out test set (2019).



Cost Structure

Below is our Cost Structure for our project. Cost categories are highlighted in the leftmost column. Of particular importance for modeling is the Equipment Costs since they highlight our servers through Databricks and Microsoft Azure software license.



COST STRUCTURE - Taming Turbulence with Machine Learning

Cost Category	Description	Quantity/Hours	Rate/Unit Cost	Subtotal
Personnel Costs	Project Manager	1	$ 50.00	$ 50.00
	Developer	3	$ 40.00	$ 120.00
	Data Scientists	5	$ 60.00	$ 300.00
	Software Engineer	1	$ 50.00	$ 50.00
	UX Designer	1	$ 45.00	$ 45.00
Equipment Costs	Servers - Databricks	100	$ 6.40	$ 640.00
	Software Licenses	1	$ 5,000.00	$ 5,000.00
Subcontractor Costs	Cloud Computing	1	$ 1,000.00	$ 1,000.00
	Marketing	1	$ 8,000.00	$ 8,000.00
	Data Labeling	2	$ 8,000.00	$ 16,000.00
Airport Costs	Cost of Delayed Flight	1	$ 20,000.00	$ 20,000.00
	Cost of On-Time Flight	1	$ 5,000.00	$ 5,000.00
Notes	Cost of Delayed Flight include: meals, accomodations, etc. Databricks Driver: Standard_D4ads_v5 (1 master and 12 worker nodes) Cloud Computing is calculated with student discount			

Below table summarizes the training time for each of our models. All models were run in Databricks clusters, however, as we were able to increase the cluster size throughout the project due to increase in budget, the training time is not directly

comparable among the different models. Furthermore, the training time fluctuates as cluster traffic changes to accommodate for other processes (such as data preprocessing and parallel training of multiple models). To provide an accurate estimation of training time using the latest large cluster, a complete rerun of all models would be required, which we plan to complete as part of future works to better inform our customers' budgeting decisions.

Model Name	Average Training Time
Baseline - Linear Regression	9 minutes per year
Improved Baseline - Linear Regression	6 minutes per year
Random Forest	4 minutes per year

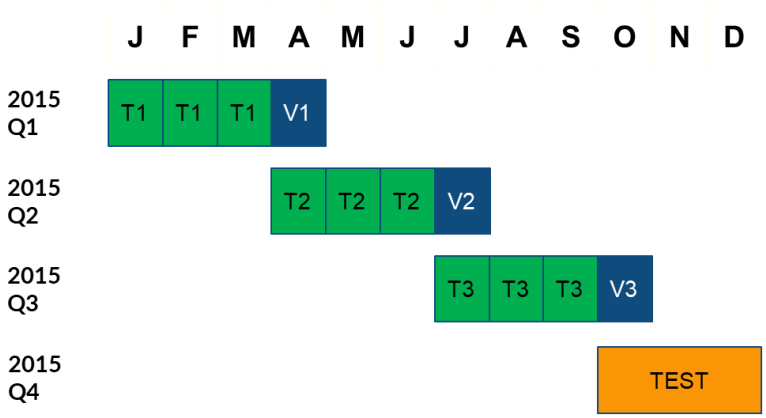
Experiments

Phase 2 Recap

Cross validation

We decided to use the rolling window to conduct cross validation, to land on the best set of parameters for each model. We used this strategy to maximize the training data, so that all of the data in the training set are used in the training once.

Note: this is the general strategy used for all the models, but for baseline model given there is no hyper-parameter tuning, the validation set is not really used except for performance validation and observation.



Regression vs Classification

We decided to use the regression approach leveraging the quantifiable information in `DEP_DELAY` (actual delay minutes) and training different models to predict the `log_DEP_DELAY` (which is a log tranformation from actual delaly minutes as stated above), and then derive the final (binary classification) output based on if the predicted delay is bigger than 15 minutes' criteria.

The prediction results of training, validation are checkpointed into the Azure Data Storage after this step, by each cross-validation fold.

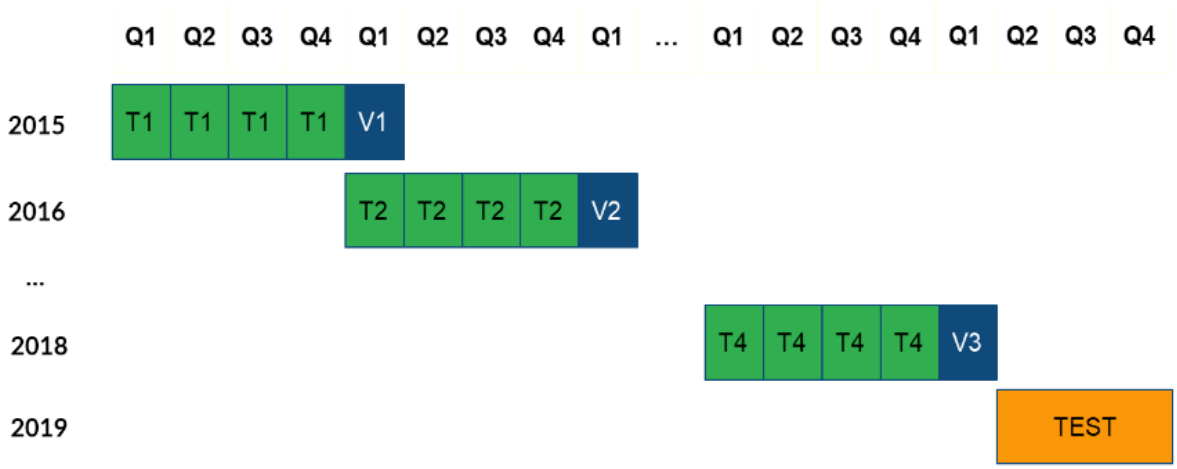
The test prediction results are saved per fold, and later aggregated using exponential average of predictions from different folds, to make sure the model that is closer to the test data time will account more heavily to the final prediction. We experimented and used the decay factor of 0.5, so that each fold is half as important as the next fold, with the fold closest to the test data being the most important.

Phase 3 Experiments

Cross Validation

We continued to use the rolling window approach to conduct cross-validation in phase 3, to land on the best set of parameters for each model.

It differed from phase 2, where the cross-validation is performed on quarterly basis, the phase 3 cross validation is performed on annual basis.



Also the validation set is a lot more important in Phase 3 than Phase 2, given that's the set we used to decide what is the best hyper-parameter for each model, during the hyper-parameter tuning phase.

Given time and resources constraints, we conducted hyper parameter tuning only on the very last year (2018) of the data as training and first quarter of 2019 as validation, as that year has the biggest impact on the final predictions on the test

Modeling Description & Decisions

Model Evaluation Framework

For the regression models, we used MSE as the training loss, but fine-tuned the models using MAE to limit the impact from extreme values.

For the classification model (which is only MLP in our case), we used the cross-entrophy loss and fined-tuned the models using Fβ, in consitent with our final evaluation metric across models.

Rationale of choosing MAE as the validation metric for regression models vs Fβ: as we are working with a regression model in most of the cases, although it's still possible to tune the hyper-parameters based on the binary classification results (via Fβ), we worry that's less intuitive for the models to learn via the hyper-parameter tuning process, as that's more separate with how the model is trained (via loss

function), thus MAE is chosen as the validation metric. Also in this way, we can later easily use these regression models for predicting the actual delay of the flights for the passengers, which is additional information that might expand the use case of our final product.

We chose Fβ to be 0.5 as we prioritized improving precision over recall. We reason that it is more important to avoid predicting a delayed flight when there is no delay (False Positive) rather than predicting no delay when there is a delayed flight (False Negative), hopefully minimizing the impact of flight delays to passengers.

Early Stopping

Early stopping is typically used on the validation set during the hyper-parameter tuning stage to avoid excessive tuning that leads to overfitting.

In our case, since our models only took one single pass through all data in the training phase, this won't make sense during the training process. Instead, we stopped hyper-parameter tuning when the target metrics began to plateau between different iterations of hyper-parameter combinations, to be efficient with our time and resources.

Improved Baseline with L2 Regularization

This method built on the baseline linear regression model by including L2 regularization. We also added all the additional features we engineered in Phase 2 and 3 to the linear model.

Refer to Appendix 1.2 for graph representations for the evaluation metrics for this model.

Random Forest

Each tree in the random forest was built by random subsampling from the training set, random feature selection and splitting on the best split criteria for the chosen subset. The hyperparameters that were optimized for this model include the number of trees (40) and maximum tree depth (8) considered at each split point.

Refer to Appendix 1.3 for graph representations for the evaluation metrics for this model.

Random Forest with XGBoost

This method leverages gradients and regularization to help prevent overfitting. Random forests build multiple independent decision trees in parallel, but each tree in gradient boosting random forest is dependent on the errors of the predecessor tree. Hyperparameters that we optimized for in this model include maximum step size/learning rate (0.1) and maximum tree depth (8). We executed early stopping during hyperparameter tuning when the variation remained constant.

Multilayer Perceptron Classifier

This method uses a fully connected neural network with a sigmoid activation function for intermediate layers and softmax activation for the output layer. The first layer is equal to the number of features in the model, 29, and the last layer is equal to the number of classes - for our binary classification model on whether a flight was delayed or not, this is 2. We used 3 hidden layers with 16, 8 and 4 perceptrons, respectively.

Hyperparameters that were optimized for this model include layers (number of perceptrons in the model, chose [29, 16, 8, 4, 2]), block size (number of instances processed per iteration, chose 512) and step size (learning rate, chose 0.03). Two examples of the architectures are "16 - sigmoid, 8 - sigmoid, 4 - sigmoid, 2 - softmax" and "29 - sigmoid, 8 - sigmoid, 4 - sigmoid, 2 - softmax".

When combining different predictions from each fold of model, we applied weights to each prediction as described above and applied a higher threshold (0.66) than 0.5 to the sum of these predictions to determine the final prediction. This allowed us to prioritize precision, though it had a slight negative impact on recall. As a result, precision was consistently higher than recall across all 4 years of train and validation datasets. This resulted in Fβ also consistently performing relatively well.

Refer to Appendix 1.5 for graph representations for the evaluation metrics for this model.

Ensemble Model - MLP

This method used all 29 features that the above models used, and also 3 new numerical features generated from the predictions of the Improved Baseline, Random Forest and Random Forest with XGBoost models. These features were inputs to the Multilayer Perceptron Model that used hyperparameters tuned as described above and generated a binary prediction of flight delay.

Refer to Appendix 1.6 for graph representations for the evaluation metrics for this model.

Ensemble Model - Logistic Regression

This method takes only 3 numerical features generated from the predictions of the Improved Baseline, Random Forest and Random Forest with XGBoost models. These features were inputs to the Logistic Regression Model and generated a binary prediction of flight delay

Results & Discussion

Untitled

Evaluation Metrics

We evaluated our models using the definitions below:

MAE: Mean Absolute Error

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

MSE: Mean Squared Error

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

RMSE: Root Mean Squared Error

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

Categorical Evaluations:

Precision:

$$Precision = \frac{True\ Delays\ Predicted}{True\ Delays\ Predicted + True\ Delays\ Unpredicted}$$

Recall:

$$Recall = \frac{True\ Delays\ Predicted}{True\ Delays\ Predicted + False\ Delays\ Predicted}$$

F_beta Score:

$$F_\beta = (1 + \beta^2) \cdot \frac{Precision \cdot Recall}{\beta^2 \cdot Precision + Recall}$$

where

$$\beta = 0.5$$

Train and Validation Results

Train and validation results averaged across 2015-2018 are listed below. We are only reporting the results for the delayed class (label = 1) which is the minority, and ignoring the non-delayed class even though that will present much better metrics.

The results are highly consistent between training and validation, suggesting little or no overfitting.

As expected, all the models implemented performed better than the Baseline Linear Regression model. This is mostly attributable to the addition of our engineered features.

Precision consistently performed better than recall, though recall also improved in models after the baseline. This means that the model was better able to avoid False Positives (predicting a delay when there is none), than False Negatives (predicting no delay when there is one).

From the results below, we determined that the ensemble model via MLP performed

Training Results Table

Model		Regression			Classification		
	Year	MAE	RSME	MSE	Fβ	Precision	Recall
Baseline: Linear Regression	2015	28.7	56.2	3160.3	51.2%	76.5%	22.1%
	2016	30.4	62.6	3921.7	52.4%	76.1%	23.3%
	2017	31.5	67.7	4581.0	53.2%	76.2%	24.0%
	2018	32.3	69.0	4765.9	51.4%	76.4%	22.3%
Improved Baseline	2015	26.6	54.3	2947.8	67.8%	86.2%	36.6%
	2016	28.3	60.7	3683.8	67.9%	86.3%	36.6%
	2017	29.3	65.9	4337.2	68.7%	86.9%	37.4%
	2018	30.0	67.1	4496.2	68.4%	86.6%	37.1%
Random Forest	2015	25.3	53.2	2830.5	69.4%	88.8%	37.0%
	2016	27.1	59.8	3570.6	68.6%	89.5%	35.5%
	2017	27.9	64.9	4206.1	69.9%	89.3%	37.4%
	2018	28.6	66.0	4357.6	69.3%	89.6%	36.4%
Gradient Boosted Tree Regression	2015	25.0	52.4	2749.6	70.5%	87.0%	40.1%
	2016	26.7	58.9	3470.7	69.9%	87.6%	38.6%
	2017	27.5	64.0	4097.2	71.1%	87.6%	40.5%
	2018	28.2	65.3	4258.8	70.4%	87.8%	39.3%
Multilayer Perceptron Classifier	2015				70.6%	76.2%	54.5%
	2016				70.3%	75.6%	55.1%
	2017				70.8%	76.6%	54.5%
	2018				70.1%	74.5%	56.6%
Ensemble via MLP	2015				70.6%	73.5%	60.7%
	2016				71.5%	75.9%	58.2%
	2017				72.3%	76.7%	59.0%
	2018				70.5%	73.5%	60.6%
Ensemble with Logistic Regression	2015				70.9%	73.1%	63.4%
	2016				70.8%	72.9%	63.6%
	2017				71.5%	73.5%	64.3%
	2018				71.0%	73.7%	61.9%

Validation Results Table

Model		Regression			Classification		
	Year	MAE	RSME	MSE	Fβ	Precision	Recall
Baseline: Linear Regression	2015	32.3	66.0	4350.2	39.1%	63.0%	15.6%
	2016	33.6	69.8	4867.5	29.3%	62.3%	9.4%
	2017	33.6	73.0	5326.1	34.7%	62.9%	12.4%
	2018	36.3	80.1	6415.3	35.2%	62.8%	12.8%
Improved Baseline	2015	27.9	59.1	3489.0	66.7%	86.5%	34.8%
	2016	29.2	64.7	4183.1	66.6%	86.9%	34.5%
	2017	29.3	66.8	4465.0	66.7%	86.9%	34.5%
	2018	31.9	74.6	5571.8	66.0%	86.7%	33.8%
Random Forest	2015	26.8	58.5	3418.4	65.9%	90.4%	31.6%
	2016	27.8	63.5	4029.3	68.3%	89.4%	35.2%
	2017	27.7	65.5	4290.0	68.4%	89.1%	35.5%
	2018	30.6	73.7	5434.8	66.1%	89.6%	32.3%
Gradient Boosted Tree Regression	2015	26.6	57.9	3355.0	67.1%	88.4%	34.1%
	2016	27.6	63.0	3963.8	68.7%	87.7%	36.9%
	2017	27.6	65.1	4232.6	69.1%	87.3%	37.6%
	2018	30.3	73.0	5334.9	67.2%	88.3%	34.3%
Multilayer Perceptron Classifier	2015				69.3%	75.7%	51.8%
	2016				69.8%	75.4%	53.8%
	2017				69.7%	77.2%	50.3%
	2018				68.8%	73.8%	54.2%
Ensemble via MLP	2015				68.6%	71.9%	58.0%

Out of Sample Test Results

Below are the results from our test.

Test Results Overview

The drop in precision between test and validation is expected: given the increased size of undelayed flights without down sampling, now there is more room to make false positive predictions.

The recall is persistent: we can reliably predict the given percentage of delayed flights based on existing features.

- There seems to be a ceiling in recall for the linear and tree based models, and we believe it's subject to missing variable constraints. We probably need to seek more data and manufacture more features to improve this.
- The neural network models seem to be capturing the recalls much better, and we suspect these models are "manufacturing" some of the features in the hidden layers as a proxy for the variables that are missing. This helped them to maintain high recall but at the cost of much lower precision.

Even though we emphasized the importance of precision rather than recall, it appears that our model is better at detecting delays (recall), rather than precisely differentiating delay from non-delays (precision).

Across Model Overview

The biggest improvements from Baseline can be observed on the improved linear model (Fβ increased by ~25%): that is mostly attributable to the inclusion of our engineered features, and consistent with what we observed in the train/validation results.

The linear regression vs tree based models have relatively similar performance on a high level, with precision around 60% and recall at 35%. When we dive into the specific examples the models seem to be capturing different signals from the data.

Neural networks seems to have better recall, above 55%, but fail to maintain precision as well as the other models.

Between the two ensembled models, the model ensembled via MLP has the best Fβ score at 45.5%, while the simple ensemble model with only 3 features (ensemble with logistic regression) as input achieved an impressive result of Fβ at 41.6%, and the recall is the highest among all the existing models at 59%.

The Random Forest model is the best performing one out of sample, with the highest Fβ score at 55.5%, thanks to the highest precision among all models.

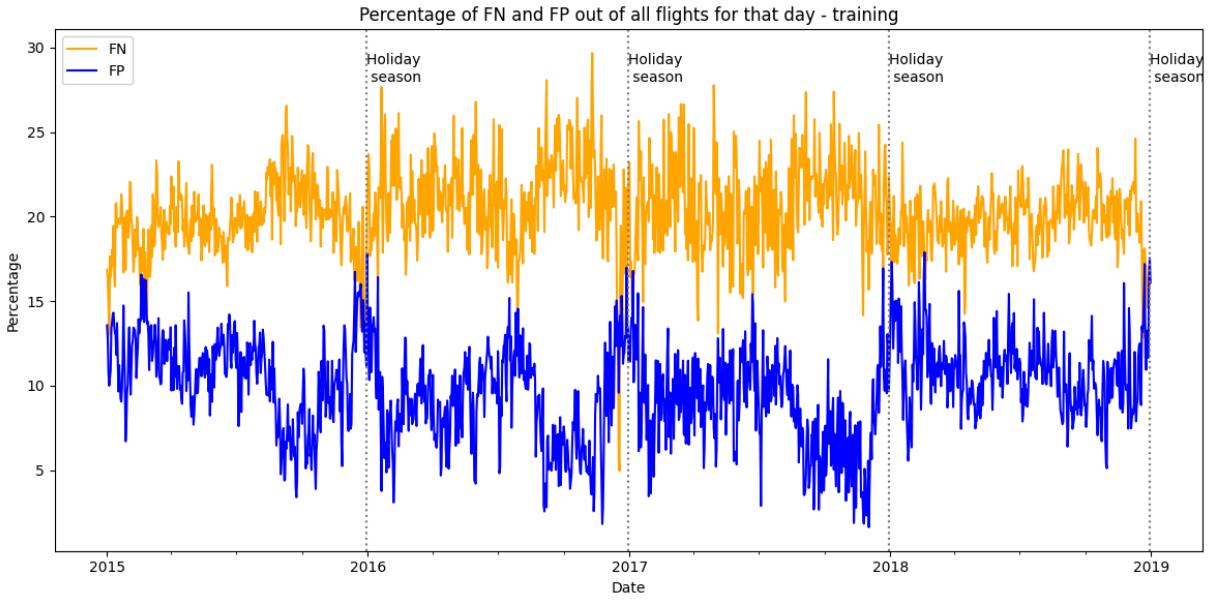
Overall, we still think the ensemble model is the most promising model given its

Test Results Table Per Year

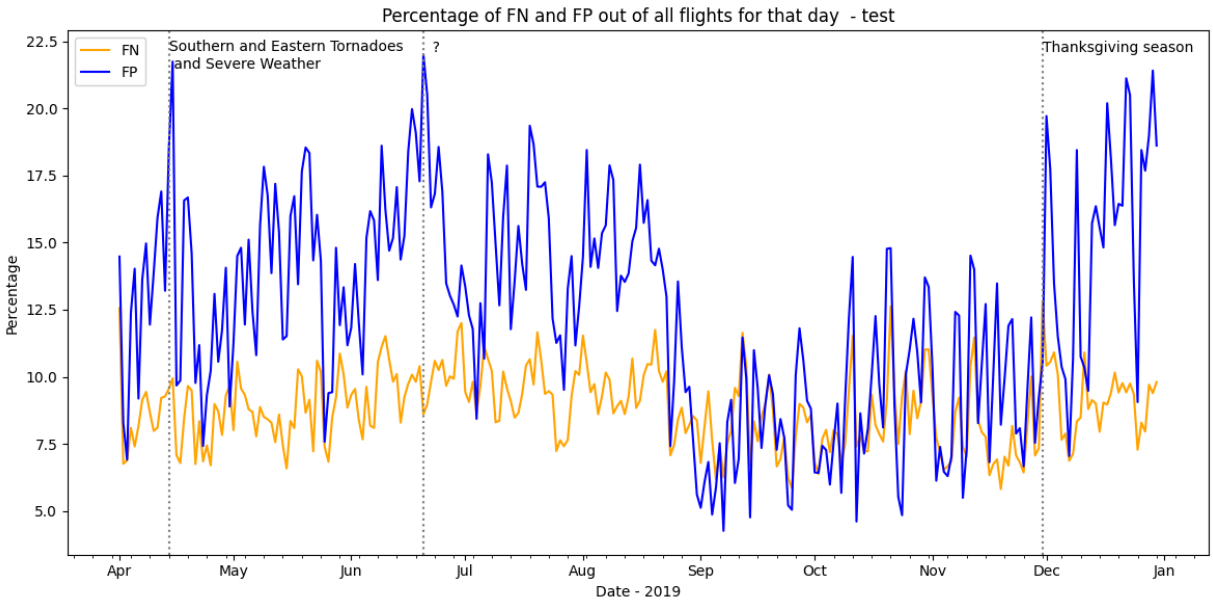
Model		Regression			Classification		
	Year	MAE	RSME	MSE	Fβ	Precision	Recall
Baseline: Linear Regression	2015	20.8	50.9	2590.7	22.0%	28.1%	11.8%
	2016	21.1	52.4	2741.6	24.3%	28.5%	15.2%
	2017	21.1	52.3	2732.4	23.2%	28.2%	13.6%
	2018	21	51.1	2615.5	24.0%	28.8%	14.4%
Improved Baseline	2015	17.8	46.2	2129.8	51.4%	58.2%	35.1%
	2016	18	46.2	2135.8	50.5%	55.4%	37.2%
	2017	17.6	46.1	2126.3	52.0%	59.8%	34.2%
	2018	17.7	46	2117.1	52.1%	59.3%	35.1%
Random Forest	2015	17.0	45.5	2069.6	55.1%	65.3%	34.0%
	2016	17.0	45.4	2062.3	55.1%	65.2%	34.1%
	2017	16.9	45.3	2055.8	55.1%	64.6%	34.6%
	2018	17.0	45.3	2055.0	55.6%	66.6%	33.5%
Gradient Boosted Tree Regression	2015	16.8	44.9	2016.1	54.6%	62.5%	36.1%
	2016	17	45.1	2034.4	53.1%	59.6%	36.9%
	2017	16.9	45.1	2032.2	53.9%	61.1%	36.5%
	2018	16.9	45.1	2032.8	54.4%	62.6%	35.7%
Multilayer Perceptron Classifier	2015				43.5%	41.7%	52.7%
	2016				42.9%	40.9%	53.7%
	2017				43.8%	42.2%	51.9%
	2018				42.0%	39.6%	55.3%
Ensemble via MLP	2015				40.9%	38.0%	58.4%
	2016				44.4%	42.6%	53.7%
	2017				44.9%	43.2%	53.1%
	2018				41.3%	38.5%	58.4%
Ensemble with Logistic Regression	2015				41.6%	38.8%	58.8%
	2016				41.5%	38.7%	59.0%

Gap Analysis

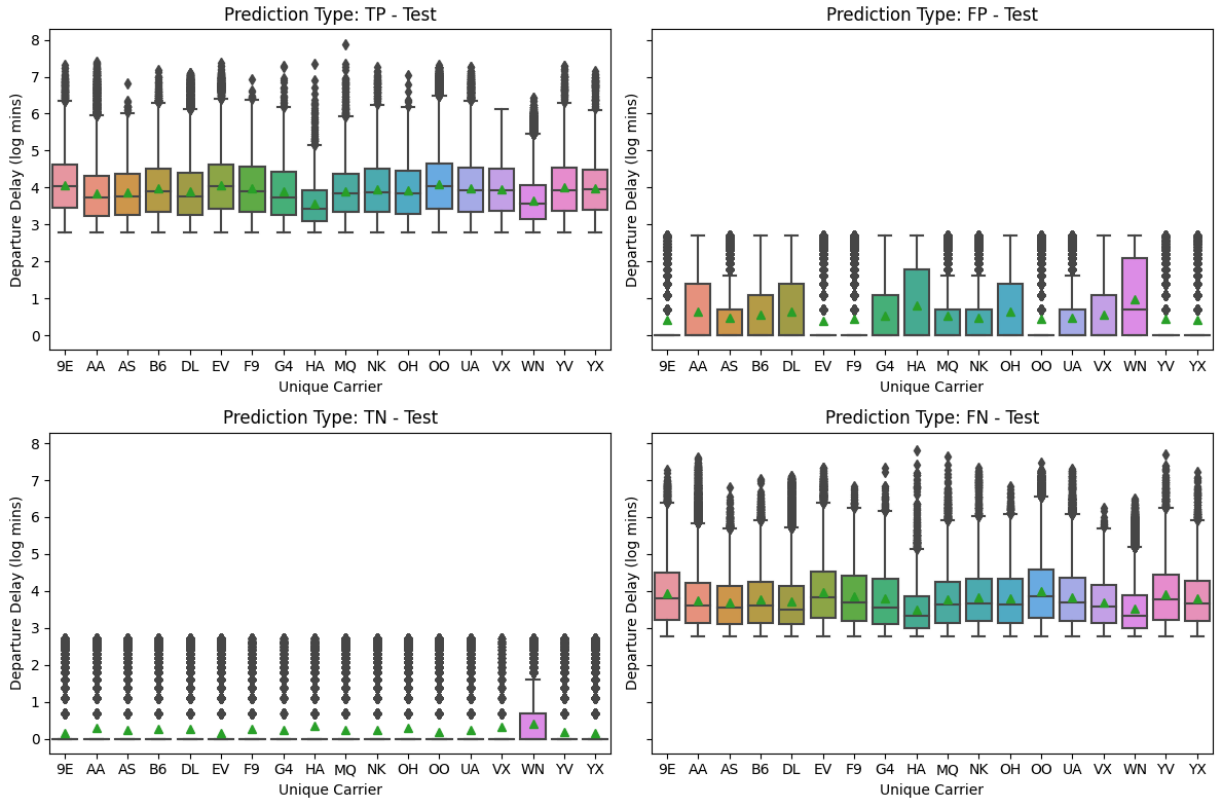
We explored our training, validation and test data, to find trends where our chosen best (ensemble via MLP) model was struggling. We noticed that even though the percentage of false negatives decreased from training to validation to test, 8.83% of the flights for 2019 were still predicted as not delayed when they were delayed (false negatives), and 12.59% of flights were predicted as delayed when they weren't delayed (false positives). We found seasonal trends in the percentage of false positives and false negatives, both in the training and test data. In the training data, we saw that the model tended to predict a higher percentage of false positives (FP), and a lower percentage of false negatives (FN) over the holiday season.



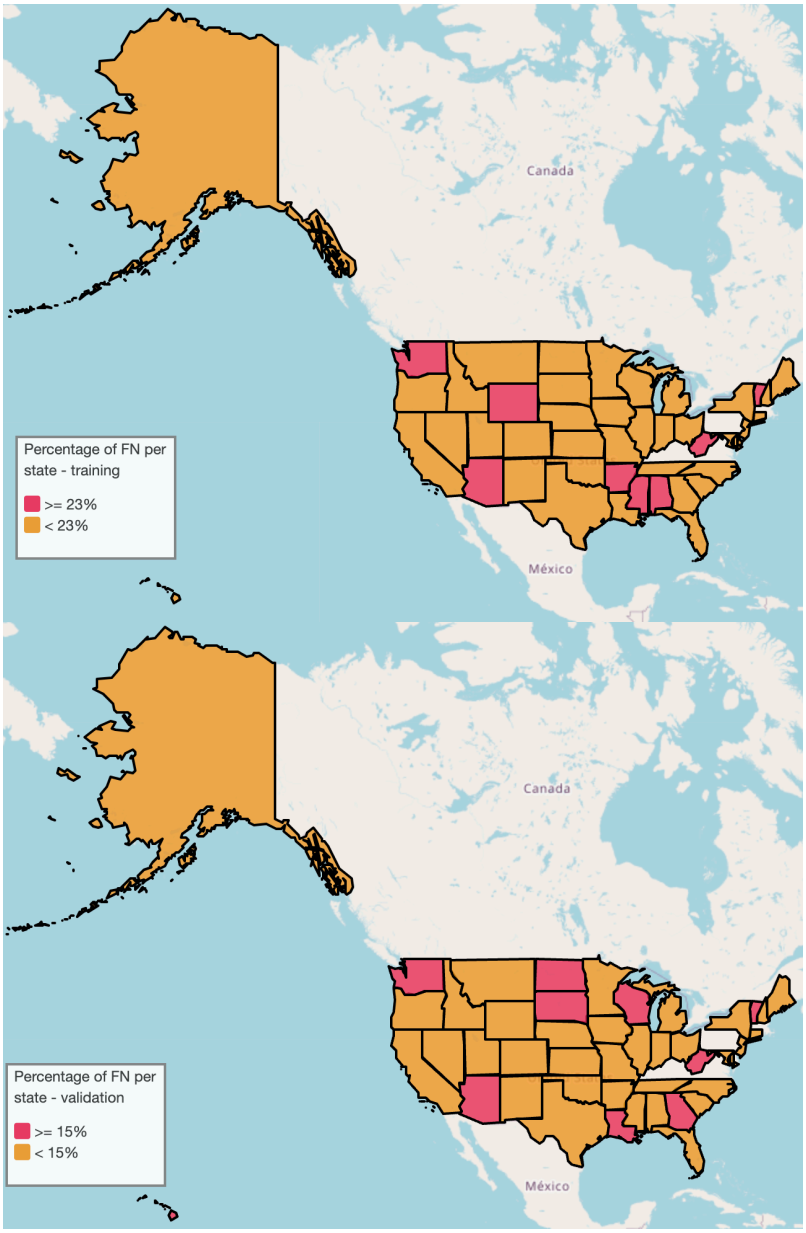
In the test data, we also saw this pattern around Thanksgiving, with an additional spike in false positives around the time when there was a natural disaster, the Southern and Eastern Tornadoes and Eastern Weather, which impacted multiple states across the US (CO, WY, NE, KS, OK, MO, IA, IL, IN, OH, PA and NJ). This prediction was probably informed by similar disasters which took place in 2018, 2017 and 2016 around the same time. We also saw a peak in false positives around late June, which we were not able to cross-reference with any natural distaster flags or national holidays, and which we suspect could be a reflection of the periodic trend in the percentage of false positives, which has a cycle of around 30 days from minimum to mimimum false positives, as oposed to the training cycle of false positives, which appear to be around 6 months long. It could also be based on information from natural disasters from previous years which took place around that same time, like the Colorado Hail Storms of 2018, Midwest Severe Weather of 2017, West Virginia Flooding and Ohio Valley Tornadoes of 2016 and Central and Northeast Severe Weather of 2015.

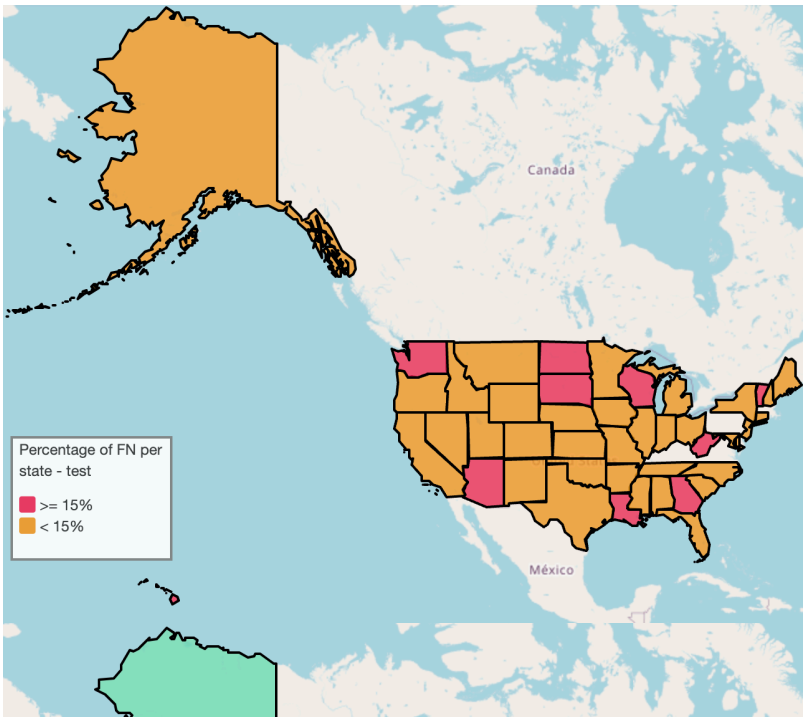


We analyzed the distribution of FP, FN, true positives (TP) and true negatives (TN) in the train, validation, and test data, and noticed no marked differences between the three sets on average, nor per year (between training and validation sets). We analyzed the correlations between variables among subgroups (TP/TN/FP/FN) and saw only slight changes in the correlations between training, validation, and tets sets. We compared the distribution of log delay by carrier, between the four groups (TP/TN/FP/FN) for training, validation and test set and saw that Hawaiian Airlines (HA) and Southwest Airlines (WN) had more false positives than other carriers during training, a pattern which was consistent in the validation and test sets. Likewise, Endeavor Air (9E), ExpressJet (EV) Skywest (OO) and Mesa Airlines (YV) had, on average, more false negatives than other carriers during training, which was consistent in the remaining sets. We analyzed the distribution of log delay based on airport size for the four groups (TP/TN/FP/FN) for training, validation and test set, and saw that while true positives showed difference in log delay based on airport size (with longer predicted delays at smaller airports), there was no difference in log delays by airport size in the false positives group, in all three sets.



We noticed differences in log delays by origin region (state), so we downloaded shape files for all US states and territories from Eric's Celeste's website (/%22https://eric.clst.org%22), and joined our train, test, and validation on them. We then analyzed the distribution of false positives and false negatives on all three sets, and plotted them on a map. The categories for higher/lower cut-off points were determined by the 75% percentile, so that darker areas on the maps are states found in the upper quartile of the distribution of percentage of false positives or false negatives over all flights for that state.


Canada
Percentage of FN per state - training
>= 23%
< 23%
México
Canada
Percentage of FN per state - validation
>= 15%
< 15%
México



Data Leakage

Data leakage is when information from outside the training dataset is used to create the model. In time-series prediction, one example of data leakage is when information from the future is included in the training dataset, resulting in the model to use future information to predict the past, creating overly optimistic results that lacks generalizability. This will also make it impossible to productionalize the model because in runtime we don't have access to that leaked feature.

We considered potential data leakage and concluded no data leakage is present in our pipeline. We also carefully considered the possibility of cardinal sins in machine learning and concluded there is no obvious cardinal sins in our pipeline.

Imputation:

When imputing for missing values for the test set, we utilized a similar approach as was done for the training set, where we imputed the missing values by taking the daily, weekly, and monthly average, in this order. We do not believe there is a concern for data leakage by using this approach as we only impute for weather features, which are unpredictable with high fluctuation throughout the day. Even though we calculated the missing value using the daily, weekly, and monthly

average, we view the imputed value as an 'estimate' of the weather value for the given time, which in real-life prediction, would be the weather forecast for the given time. We believe this estimated (or forecasted) weather value is likely to be far off from the actual weather at the time of prediction that it does not consititute data leakage and the potential impact to our model is remote.

As part of our future works, we also considered a different strategy to strengthen our imputation process.

PageRank:

In preparing the training set, we calculated the PageRank by year to match the cross-validation period to find the most accurate airport connectivity for each training period. For the test set, we used the average PageRank of prior 4 years to ensure no data leakage arise in creating the PageRank feature for the test set.

Cross-Validation:

When we were evaluating the model performance during the cross validation process, we used the first quarter of 2019 as the validation set for the model trained by data in 2018. As we excluded the first quarter of 2019 from the test results reporting, we believe there is no data leakage in our pipeline.

Future Works

Although we tried to be as robust as possible in completing the project, there are still areas for improvements which we would like to tackle in the next stage to bring the best product to production.

Weather Data Imputation

In imputing for missing values, we took an average approach, by day, by week, and by month. Even though we believe this approach is a reasonable estimate of the missing value, another potential approach to consider is to directly use the last

available record as imputation. By directly using the last available record, the imputation could be less noisy and more accurate, potentially providing higher quality data and improve our modeling result.

Data Sampling

Given we are interested in the delayed flights, which is a minority class, we had to downsample our data to balance the delayed and non-delayed flights. Also because the flight delay is heavily right skewed, we decided to take log of the delayed time as our outcome variable. Both of the decisions helped the regression model to better capture the delay pattern, but we noticed the model still have much higher MAE than RMSE. To fix this, we propose a better data sampling strategy:

1. The delayed flights should be further filtered to drop the fat tail - we are thinking the 5% most delayed flights should be dropped from the training set, as most of the delays to that extent are subject to unobserved variable bias and are not generalizable in our scenarios.
2. For the sampled un-delayed flights (including flights delayed but less than 15 mins), which accounted for ~80% of the data, we propose to have a better sampling strategy rather than purely random. We should first have a more scattered sample set from each date and origins in the training/validation set, and also consider to concentrate the sample on the flights that are delayed but less than 15 mins than the early flights.

The rationale is to have a more representative comparison set for the model to train, and hopefully by doing this it will better inform the model to pick up the features that will cause the small delays (<15 mins) and the big delays (>15 mins), which was the majority of the false positives that the models made.

Cancelled Flights

During our EDA, we identified cancelled flights existing in our datasets which lacked the neccessary information for us to gain a comprehensive understanding of the relationship between delay and cancellation. We understand flight cancellation is frustrating for our customers and often delayed flights could become canceled flights, leading to decreased customer satisfaction. We believe this would be a valuable endeavor for future exploration which we intend to tackle in the near future.

Time Series Handling

Some of the time series features seem highly contemporary, for example holidays, natural disasters, and other events. Even though they contribute heavily to the flight delays on the particular day, we expect a lot of the features to be bounded to the specific time and location. In production time, we believe it will be worth experimenting with a more localized model approach, based on more defined data for each airport/state, with maybe a shorter history to better fitted to those trends. This will also reduce the time and cost of training, and allow our models to respond to regime changes (COVID-19) more quickly.

Modeling From Gap Analysis

To improve our model, we would like to perform class balancing of delayed and non-delayed flights by state, and include data about natural disasters for a wider range of years. We have also been designing a neural network which might detect other non-linear features that our current model wasn't able to capture. To improve model predictions based on carrier, we would like to incorporate user rankings, given that some of the airlines which scored high in percentage of false positives actually ranked low in consumer complaints to the Transportation Department, according to a Wall Street Journal on The Best and Worst US Airlines of 2019 (https://www.wsj.com/articles/the-best-and-worst-u-s-airlines-of-2019-11579097301).

Neural Network Decision

Within our project, which serves as a proof of concept, we only experimented with Neural Networks via MLLib. This was decided mostly because of the time limit, but also because of the significant efforts that are involved when experimenting with other pipelines outside of MLLib in a distributed way via PySpark. There's promising progress (https://docs.databricks.com/en/machine-learning/train-model/dl-best-practices.html) in this area that makes it definitely possible, but still seems like a cutting edge area that would involve significant efforts for exploration. Given more time in the next stage, we believe this exercise is worthwhile, as it will open up other the opportunities of experimenting outside of what modelling is supported by MLLib, which is fairly limited as of the writing.

Better Ensembling

Given time constraints, we only experiemented the ensembling method in limited ways, but even with the results gathered so far, there is strong evidence of consolidated learning from all of the models in this way. We are optimistic about this approach and suggest more experiments should be done, including but not limited to:

1. Including the information from the MLP model into the ensembling. Given the output is binary classification result from MLP, we don't believe it's granular enough as an input to the ensembling model. Time permitted, we would like to extract the hidden layers from the MLP model, serving as the embedding that is generated by the neural network, and use them as the inputs into the final ensembling model. We believe that's a more organical way of merging the learnings from different models together.

2. Experiment ensembling via all the modeling framework. Currently we only experimented the ensembling via MLP and logistic classification model, but we would like to experiment tree based models, and other neural network approaches as the final ensembling layer.

3. Experiment with different features as input. We current use the log version of the predicted departure delay, but after the data sampling strategy change above, we could experiment with using the actual predicted time directly, as it is more relevant for the final decision making.

Productionalize Plan

Below areas are what we will consider to bring our model into production:

- We would like to train a final model using the entire 5-Year (2015-2019) data, with cross-validation on each year, using the best performing hyperparameters.
- We would like to also continue monitoring the performance of the model and proactively retrain the models every few months to reflect the most current trend. This will also allow our model to adjust for regime changes, such as the impact from COVID-19.

- We would also like to balance the cost and benefits of retaining the older years' data, given a decay factor of 0.5. As we add newer data into our model, we could consider removing the data from too long ago as the older data no longer reflects the most recent trend and regimes in the industry. we could also consider

Conclusion

Predicting flight delays gives travelers extra time to prepare when facing differences in the scheduled departure time and actual departure time, playing an essential role in customer satisfaction. Our project hypothesis is that given our data, our best model will achieve Fβ (β=0.5) better than 0.24, which is our baseline result. Our most influential features were delays caused by previous flights, city and state of origin, city of destination, day of month and month and how overcast the sky was. Our best model was the Ensemble via MLP Model which used a Multilayer Perceptron architecture with 3 hidden layers, and resulted in an Fβ of 0.45. These project findings are aligned with our original hypothesis. This project can be further improved by analyzing methods to decrease the number of False Positives, however we believe we can mitigate the impact to passengers by recommending on a marketing strategy that encourages them to enjoy the additional time and amentities at the airports, especially around the holidays.

Sourcing Additional Data

We decided to source additional weather and flights data, for the years 2020-2023, for future analysis with more recent data. Weather data was taken from the same source as previous datasets, NCEI (https://www.ncei.noaa.gov/data/local-climatological-data/archive/). This link contained a tar file, which was processed into multiple csv files (one per station), and then all csv files were joined as a single parquet file for each year. Those parquet files were opened as Spark dataframes, and then joined into a single parquet file. We then performed some sanity checks on our 2020-2023 dataset, to make sure it was comparable to the 2015-2019 dataset. We compared the dimensions, typecast the Schema to that of the 2015-2019 dataset, renamed the columns to match the old dataset and changed the datetype to match the format of the 2015-2019 dataset. We also checked whether the unique

weather stations of both datasets was the same. We found that the 2020-2023 dataset had 993 fewer stations than the 2015-2019 datset. After exploring several hypothesis as to why this could be the case, we concluded that the most likely option is that the missing stations no longer exist/report their data. A more thorough explanation of this analysis is available in the additional weather data sourcing notebook (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3544923356435327/command/3544923356435 but the following table confirms that the number of downloaded files corresponds to the number of unique stations in the dataset, which is why we believe that it is not a data engineering issue on our end.

Year	Number of source files in data frame	Number of files when manually decompressed	Number of unique stations per year
2020	13562	13562	13562
2021	13539	13539	13539
2022	13468	13468	13468
2023	13422	13422	13422

We performed an EDA by plotting all stations on a map, and comparing them to those for the 2015-2019 dataset and found that they were similar in nature, and visualy analyzed the top 5 rows for both the old and the new dataset to make sure they were comparable in format. The final 2020-2023 weather parquet file can be found on our team blob, under the following path: "wasbs://261storagecontainer@261storage.blob.core.windows.net (mailto:261storagecontainer@261storage.blob.core.windows.net)/full_weather_data_

The flights data was taken from the same source as previous datasets, Transtats (https://www.transtats.bts.gov/Fields.asp?gnoyr_VQ=FGJLinks). Csv files for each month and year were downloaded from the website, uploaded to the team blob, and then saved as a Spark dataframe per year by means of a SQL query. We then joined data for all years, checking that no columns or rows had been lost in the join. As part of our sanity checks, we analyzed columns which had different names in the 2015-

2019 dataset and the 2020-2023 dataset. By visually inspected the data in these columns, we confirmed that they were indeed the same. Below is a list of said columns.

2015-2019	2020-2023	Same?
OP_UNIQUE_CARRIER	Reporting_Airline	Yes
OP_CARRIER_AIRLINE_ID	DOT_ID_Reporting_Airline	Yes
OP_CARRIER	IATA_CODE_Reporting_Airline	Yes
OP_CARRIER_FL_NUM	Flight_Number_Reporting_Airline	Yes
DEP_DELAY_NEW	DepDelayMinutes	Yes
ARR_DELAY_NEW	ArrDelayMinutes	Yes

We then reorganized the columns in the new flights dataset, to match the order of the old flights dataset, and changed the columns to match the old flights dataset. We did some EDA to compare the old and the new flights dataset, and found a difference in the number of rows (flights). On average there were 14,835,487 flights per year in the 2015-2019 time range, whereas in 2020-2023, there were on average 4,852,155 flights. The decreased number of flights in the newer dataset can probably be due to the pandemic, and not to missing data on our end. We typecast the new flights data schema to match the old flights df_flights schema, and visually examined the first five rows of the new flights data and the old flights data, and confirmed that the variables looked the same, so we saved the new 2020-2023 flights dataset as a parquet file, which can also be found on our team blob, under

Joining the Data

Below are the steps we performed to join the raw datasets (flights, airports, weather stations, and weather). The process was first completed on the 3-months dataset to test the feasibility of the algorithm before being applied to the entire 2015-2021 raw datasets. When running the join on the entire raw datasets, all visual outputs were removed from the algorithm to speed up the process, except for necessary sanity

checks. The entire join is performed on a spark cluster with a Standard_DS5_v2 56GB 16 Cores Driver with up to 12 Standard_D4ads_v5 32-192GB 8-48 Cores workers.

1. We first left joined the airports and the weather stations data based on the shortest distance between airport and stations. The raw airport dataset contains 18 columns and 78.4K rows while the raw weather stations dataset contains 12 columns and 5M rows. Notably the raw airport dataset contains the airports from the globe while the weather stations dataset only contains the weather stations in the US. When the datasets were joined, only the weather stations closest to each airport is retained. The airports and the weather stations data are date-agnostic, and the entire process took around a half-hour. The output dataframe contains 23 columns and 78.5K rows and was saved in the blob storage for use in Step 2. Below are the steps performed to complete the left join.
    - Clean-up the raw airport dataset to get the airport unique identifier (IATA code in most cases)
    - Clean-up the raw weather stations dataset to get the weather station unique identifier (`station_id`)
    - Cross-join the raw airport dataset and raw weather stations dataset to find the station closest to each airport
        - Get the subset of raw airport dataset with only unique identifier (`ORIGIN`) & coordinates
        - Get the subset of raw weather stations dataset with only unique identifier (`station_id`)& coordinates
        - Cross join the weather stations subset to the airport subset
        - Calculate the distance between each airport and each weather stations
        - Filter for the weather station closest to each airport
    - Left-join the raw airport dataset with the cross-joined dataset with the closest weather station on `ORIGIN`
    - Left-join the result from above step with the raw weather stations dataset on station_id
    - Clean-up the resulting dataframe for use in step 2
        - Rename all columns
2. We then left joined the flights data with the joined airports and weather stations data based on `ORIGIN`. The raw flights dataset contains 109 columns and 74M rows while the raw joined airports and weather stations dataset contains 23

columns and 78.5K rows. To speed-up the processing time, we cleaned the datasets by removing any duplicate rows in either dataframes and dropped any irrelevant columns, such as columns containing diverted flights information. Notably the flights data only contains flights within the US and US territories for the years between 2015-2021, however, the airports data contained all airports from around the globe. As result of the left join on flights data, the airports from outside of US are dropped. The entire process took less than 20-minutes when the cluster was not occupied by other tasks. The output dataframe contains 84 columns and 42.4M rows and was saved in the blob storage for use in Step 3. Below are the steps performed to complete the left join.

- Confirm the `ORIGIN` in step 1 joined dataset all exist in the flights dataset
- Clean-up the step 1 joined dataset as needed to match the `ORIGIN` with flights dataset
- Left-join the raw flights dataset with the cleaned-up step 1 joined dataset on `ORIGIN`
- Clean-up the resulting dataframe for use in step 3
  - Get the local timezone `origin_airport_timezone` for each record
  - Get the `sched_depart_date_time` for each record based on `FL_DATE` and `CRS_DEP_TIME`
  - Get the `sched_depart_date_time_UTC` for each record based on `sched_depart_date_time` and `origin_airport_timezone`
  - Get the `two_hours_prior_depart_UTC`, `three_hours_prior_depart_UTC`, and `four_hours_prior_depart_UTC` for each record based on `sched_depart_date_time_UTC`

3. Finally, we left joined the joined flights, airports, and weather stations data with the weather data based on `STATION` (same as `station_id` from step 1) and time of scheduled departure in UTC. We produced three different join results, based on `two_hours_prior_depart_UTC`, `three_hours_prior_depart_UTC`, and `four_hours_prior_depart_UTC` time as we were curious whether the model performance would differ if we used the weather information closer in time to the scheduled departure time. The joined flights, airports, and weather stations data contains 84 columns and 42.4M rows while the raw weather data contains 124 columns and 899M rows. Similar to Step 2, to speed-up the processing time, we cleaned the datasets by removing any duplicate rows in either dataframes and dropped any irrelevant columns, such as columns containing daily, monthly, and

ShortDuration weather information as they are made-up of almost exclusively null values. Notably the weather data contains the weather information based on set time increments for each station, but we only want the weather information closest to and up until the cut-off time (`two_hours_prior_depart_UTC`, `three_hours_prior_depart_UTC`, or `four_hours_prior_depart_UTC`). The entire process took around 5-hours when the cluster was also occupied by other tasks. The output dataframe contains 84 columns and 42.4M rows and was saved in the blob storage for use in Step 3. Below are the steps performed to complete the left join.

- Confirm the `STATION` in step 2 joined dataset all exist in the weathers dataset
- Clean-up the `date` column in weathers dataset to prepare for join
- Full-left-join the step 2 joined dataset and raw weathers dataset to find the weather info closest to `two_hours_prior_depart_UTC` for each `STATION`
  - Get the subset of step 2 joined dataset with only unique identifier (`STATION`) & `two_hours_prior_depart_UTC`
  - Get the subset of raw weathers dataset with only unique identifier (`STATION`) & `date`
  - Full-left-join the step 2 joined subset and raw weathers subset on `STATION`
  - Calculate the time difference between each flight (`two_hours_prior_depart_UTC`) and each weather measurement time (`date`) for each `STATION`
  - Filter for the weather info closest to `two_hours_prior_depart_UTC` for each `STATION`
- Left-join the step 2 joined dataset with the full-left-join dataset with the closest time on `STATION`
- Left-join the result from above step with the raw weathers dataset on `STATION`
- Clean-up the resulting dataframe for EDA & modeling
  - Rename all columns

After the datasets were joined, we preprocessed the dataset using the same data preprocessing pipeline we built for the OTPW dataset, tweaking only slightly to adapt to the format of the self-joined datasets (such as filter to remove 2020 & 2021

data since our task is focused on 2015-2019). This preprocessing includes 1) split the data to train vs test, 2) clean the dataframes as needed, and 3) feature engineer select features. The output dataframe `s2_1` is identical to the processed OTPW dataset we used for training the models. We also performed a series of EDA on the `s2_1` dataset similar to the OTPW dataset and inspected the joined data visually and concluded that 1) the number of rows are largely consistent with the OTPW dataset, giving us confidence that the data was joined correctly, and 2) the visual EDA results are also identifical to that of the OTPW dataset, further reinforcing our confidence that the data was joined correctly. The processed training set contains 21M rows while the processed test set contains 6.4M rows, both the train and test sets have 56 columns after processing. Notably we populated the graph-based feature `pagerank` and the time-based features `yhat` and `trend` used the Prophet and GraphFrame results from OTPW, as the flight data from the datasets are identical.

We then passed the resulting preprocessed dataset through our modeling pipeline to evaluate the performance of our model on this new dataset. (We only used the joined data joined on `two_hours_prior_depart_UTC` as we believe the one joined on `four_hours_prior_depart_UTC` is identical to OTPW and repeating the model with this data would not add value to our analysis).

We ran Improved Baseline Linear Regression with L2, Random Forest, XGBoost, and Multilayer Perceptron on the new dataset and noted the results are similar to that of the OTPW dataset. Thanks to the work we have done to build robust data processing, EDA, and modeling pipelines, we were able to clean and process the joined datasets, perform EDA, and run all models within the span of 24-hours. This speaks to the importance of building robust pipelines to increase efficiency and scalability.

We have omitted the detailed EDA & modeling results here and would like to direct interested readers to the notebooks for further information. The notebooks used to perform the join and the subsequent preprocessing and EDA are stored here (https://adb-4248444930383559.19.azuredatabricks.net/browse/folders/2769031167784459?o=4248444930383559) and the notebooks used to model with the joined dataset

References

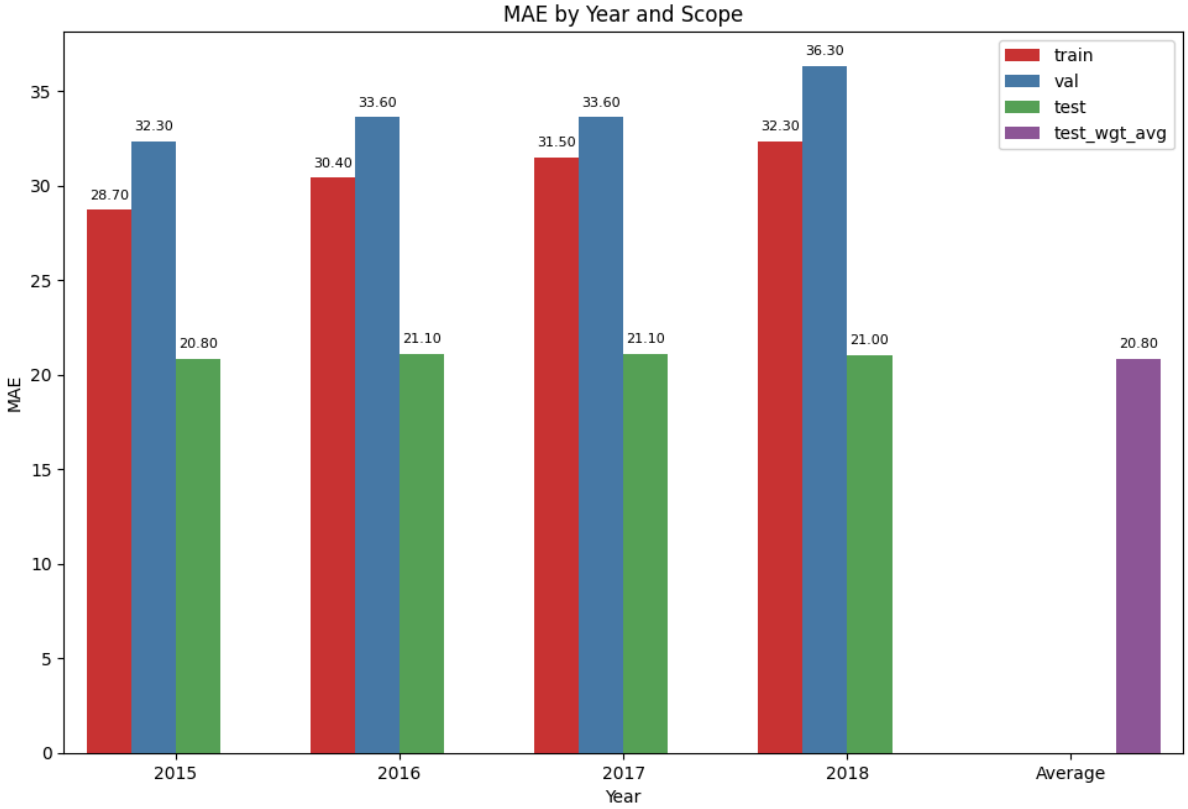
- Airport Codes - Dataset - Datahub - Frictionless Data. Airport Codes (/datahub.io/core/airport-codes). Accessed 7 Apr. 2024.
- American Mathematical Society, 2006. PageRank (https://www.ams.org/publicoutreach/feature-column/fcarc-pagerank) Accessed 7 Apr. 2024.
- Anupkumar, Ashmith. "Investigating the Costs and Economic Impact Of Flight Delays In The Aviation Industry and The Potential Strategies for Reduction." (2023).
- Bureau of Transportation Statistics: Transtats. Bureau of Transportation Statistics (/www.transtats.bts.gov/Tables.asp?QO_VQ=EFD&QO_anzr=Nv4yv0r%FDb0-gvzr%FDcr4s14zn0pr%FDQn6n&QO_fu146_anzr=b0-gvzr). Accessed 7 Apr. 2024.
- National Centers for Environmental Information (NCEI) - noaa.gov. “January 2015 National Climate Report.” January 2015 National Climate Report (/www.ncei.noaa.gov/access/monitoring/monthly-report/national/201501). Accessed 7 Apr. 2024.
- National Centers for Environmental Information (NCEI) Local Climatological Data (/www.ncei.noaa.gov/data/local-climatological-data/archive/). Accessed 7 Apr. 2024.
- National Oceanic and Atmospheric Administration. Global Surface Summary of the Day (https://www.ncei.noaa.gov/access/metadata/landing-page/bin/iso?id=gov.noaa.ncdc:C00516)
- National Centers for Environmental Information. Local Climatological Data (LCD) Dataset Documentation (https://www.ncei.noaa.gov/pub/data/cdo/documentation/LCD_documentation.pd
- USA.gov. American Holidays (https://www.usa.gov/holidays)
- National Centers for Environmental Information. U.S. Billion-Dollar Weather & Climate Disasters 1980-2024 (https://www.ncei.noaa.gov/access/billions/events.pdf)
- National Centers for Environmental Information.Billion-Dollar Weather and Climate Disasters (https://www.ncei.noaa.gov/access/billions/events/US/1980-2017?disasters%5B%5D=all-disasters)

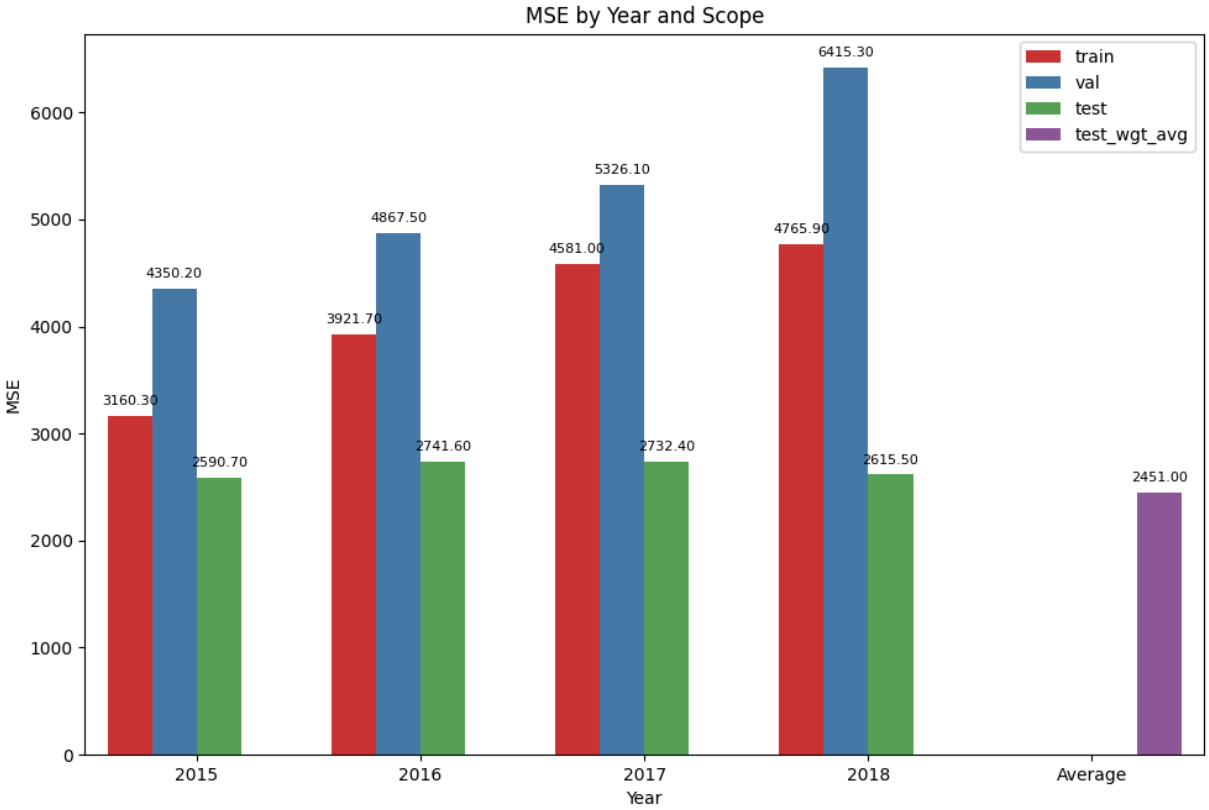
- National Oceanic and Atmospheric Administration.U.S. saw 10 billion-dollar disasters in 2015 (https://www.noaa.gov/news/us-saw-10-billion-dollar-disasters-in-2015)
- Prophet from Meta. Prophet Quick Start Guide (https://facebook.github.io/prophet/docs/quick_start.html). Accessed 7 Apr. 2024.
- Our Airports (https://ourairports.com/help/data-dictionary.html#countries). Accessed 7 Apr. 2024.
- The Wall Street Journal. The Best and Worst U.S. Airlines of 2019 (https://www.wsj.com/articles/the-best-and-worst-u-s-airlines-of-2019-11579097301). Accesed 20 Apr. 2024
- Eric Celeste. Shapefiles of all US States (/%22https://eric.clst.org%22)

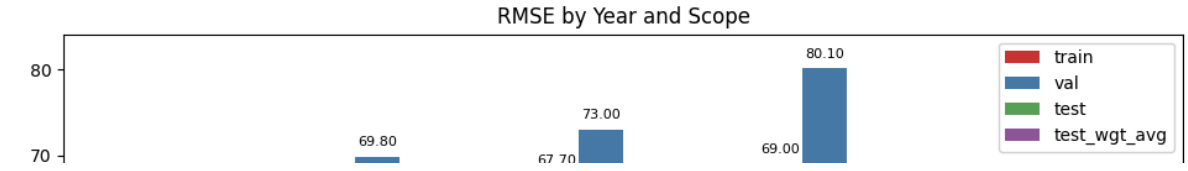
Appendix 1: Graphs for Models

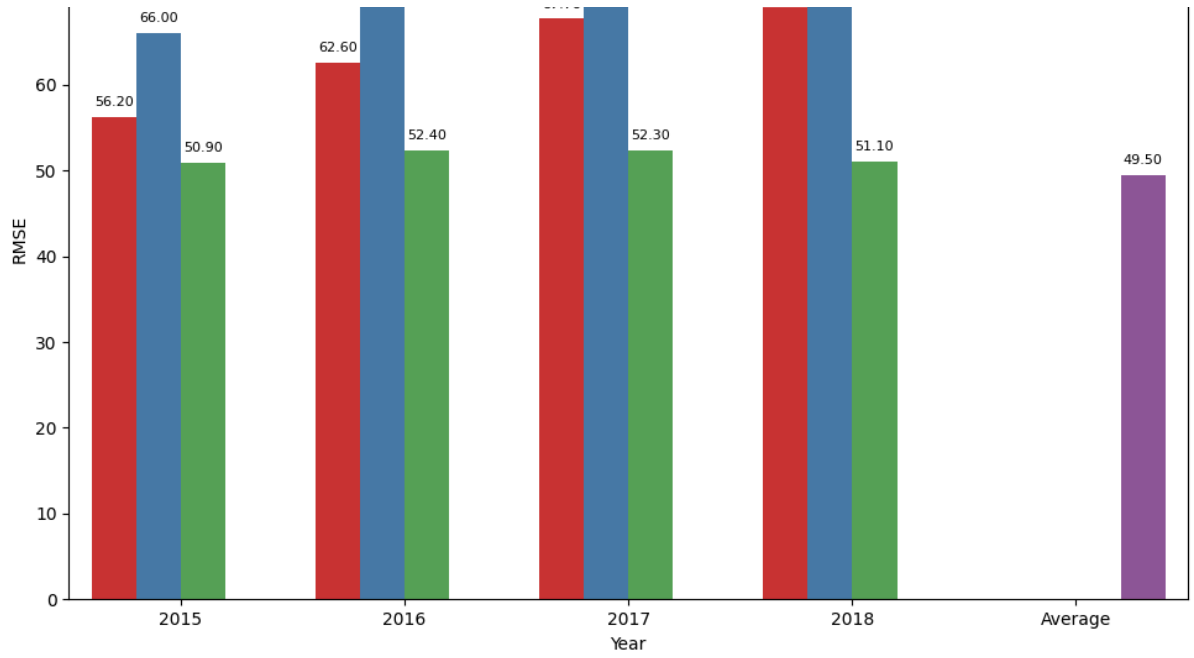
All of the graphs here only focuses on the delayed flights. We ignored the other label (non-delayed flights) as that is not interesting for our customers, even though we see very good results for the non-delayed flights predictions.

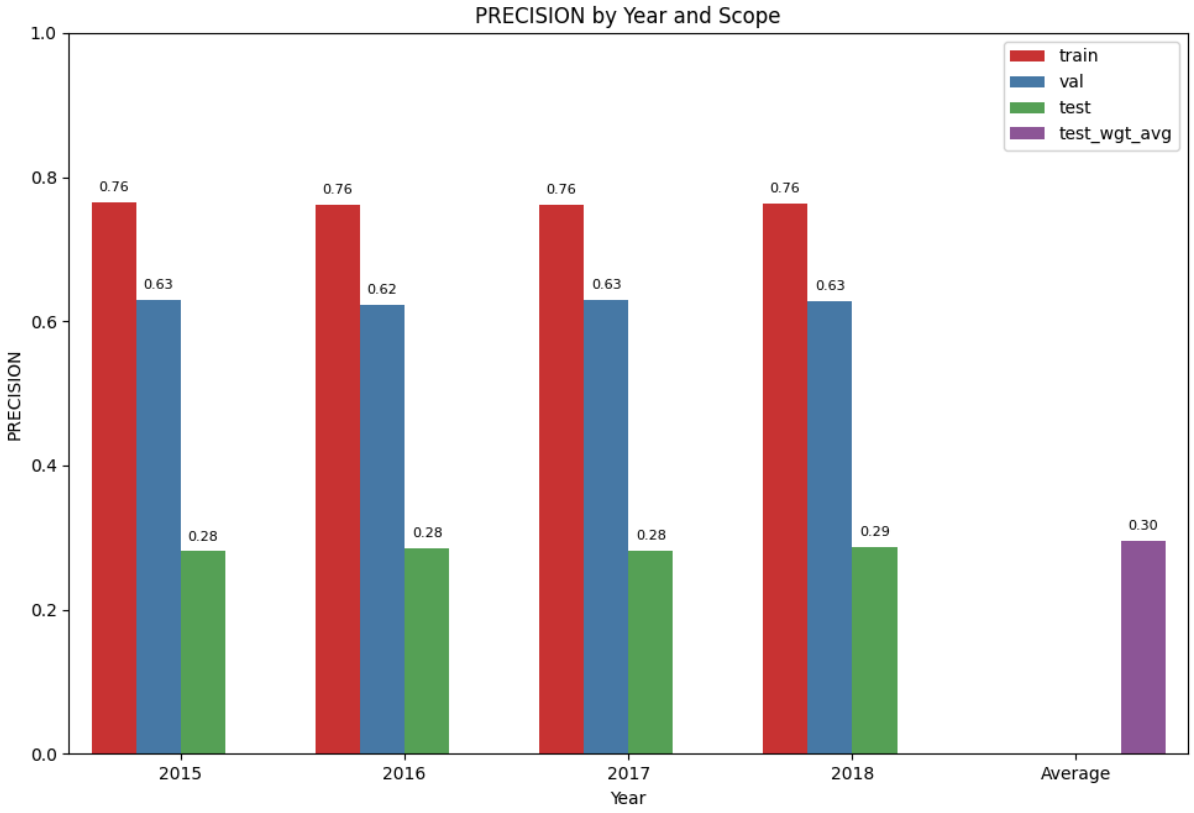
1. Baseline: Linear Regression

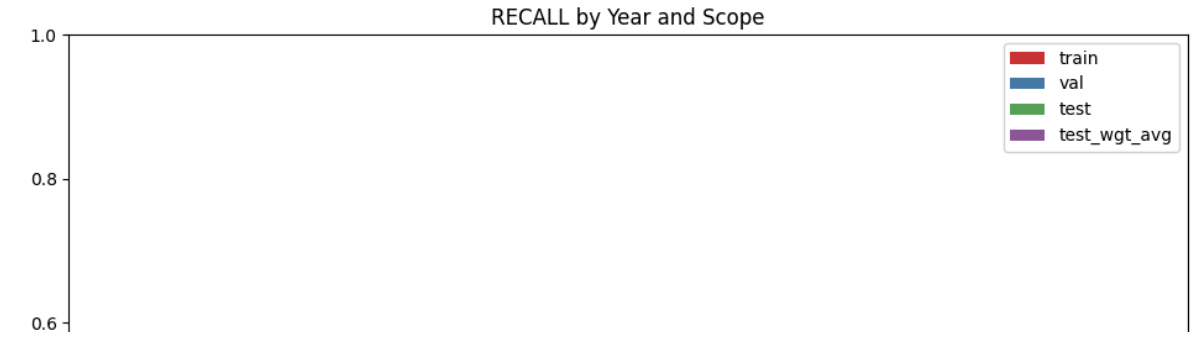

MAE by Year and Scope
train
val
test
test_wgt_avg
MAE
Year
2015
2016
2017
2018
Average
28.70
32.30
20.80
30.40
33.60
21.10
31.50
33.60
21.10
32.30
36.30
21.00
20.80

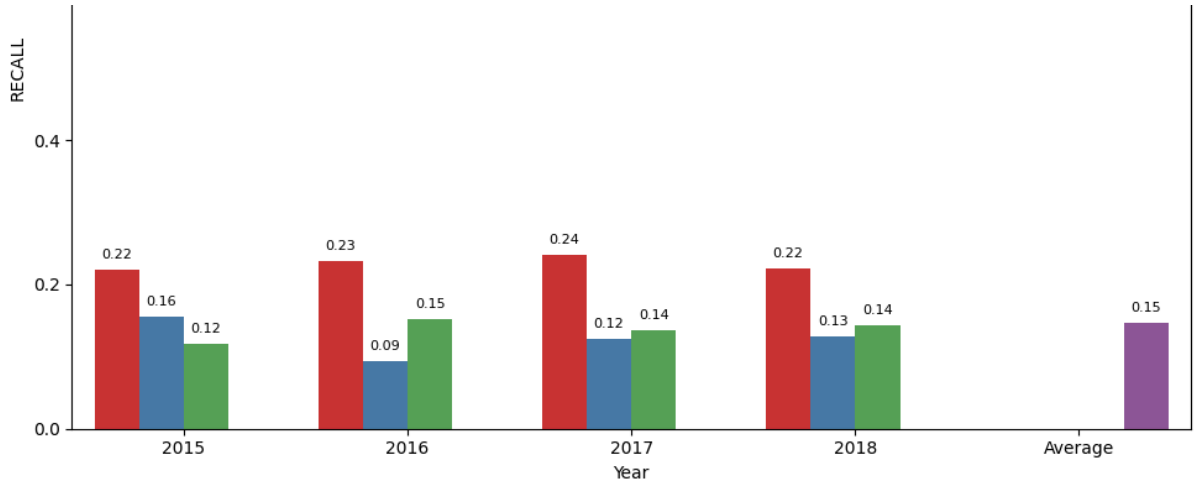

MSE by Year and Scope
train
val
test
test_wgt_avg
MSE
Year
2015
2016
2017
2018
Average
3160.30
4350.20
2590.70
3921.70
4867.50
2741.60
4581.00
5326.10
2732.40
4765.90
6415.30
2615.50
2451.00

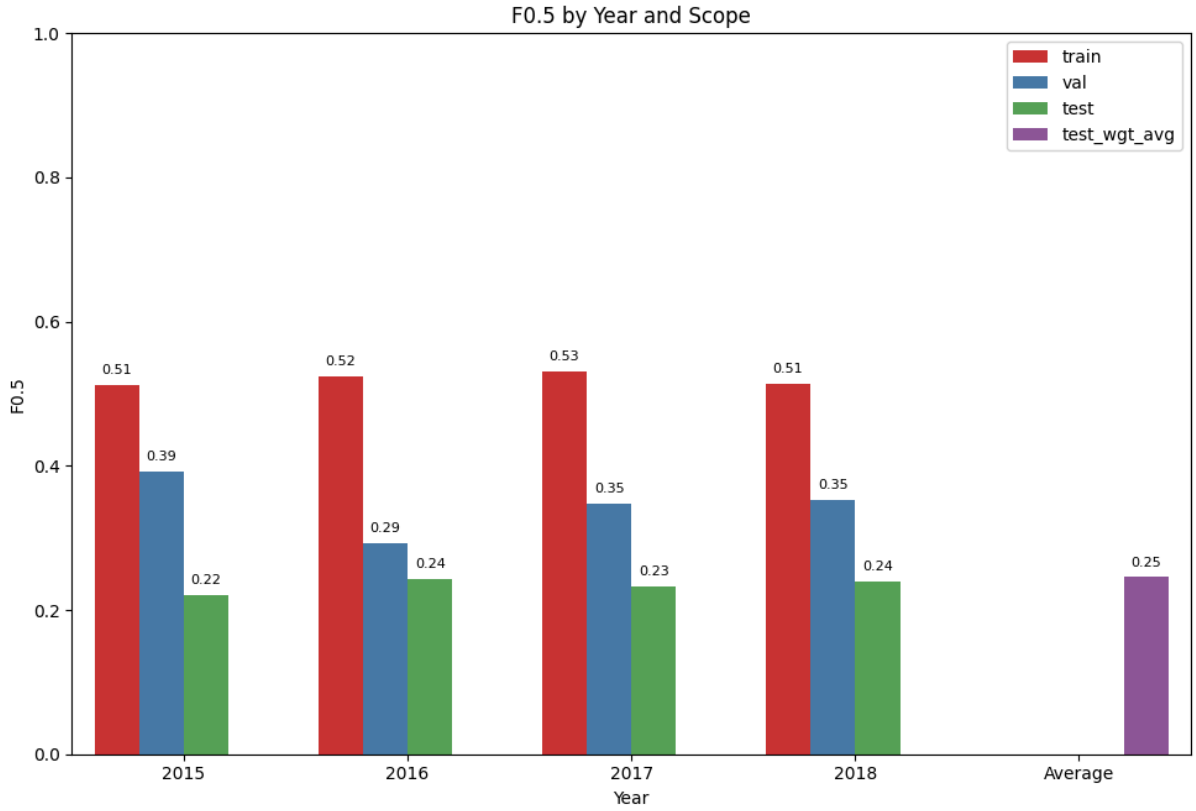

RMSE by Year and Scope
train
val
test
test_wgt_avg
69.80
73.00
69.00
80.10


RMSE
Year
2015
2016
2017
2018
Average
56.20
66.00
50.90
62.60
52.40
52.30
51.10
49.50

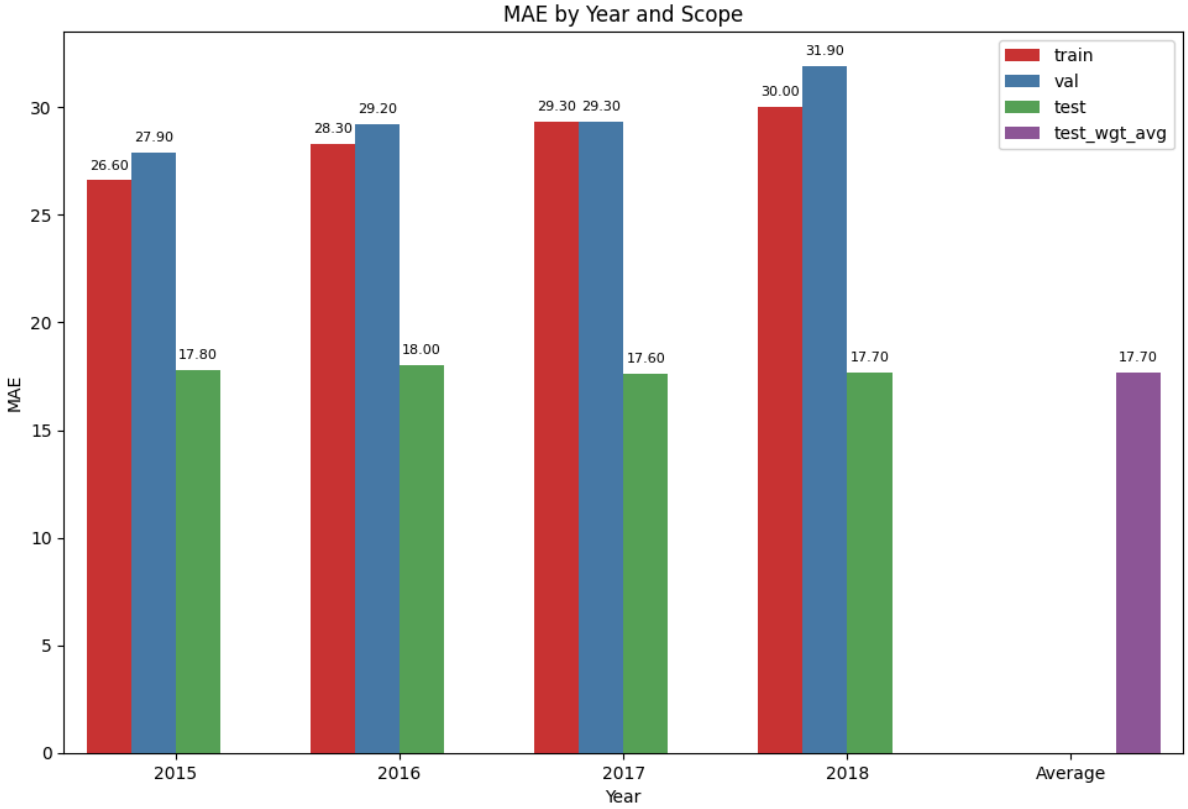

PRECISION by Year and Scope
PRECISION
Year
train
val
test
test_wgt_avg
2015
2016
2017
2018
Average
0.76
0.63
0.28
0.76
0.62
0.28
0.76
0.63
0.28
0.76
0.63
0.29
0.30

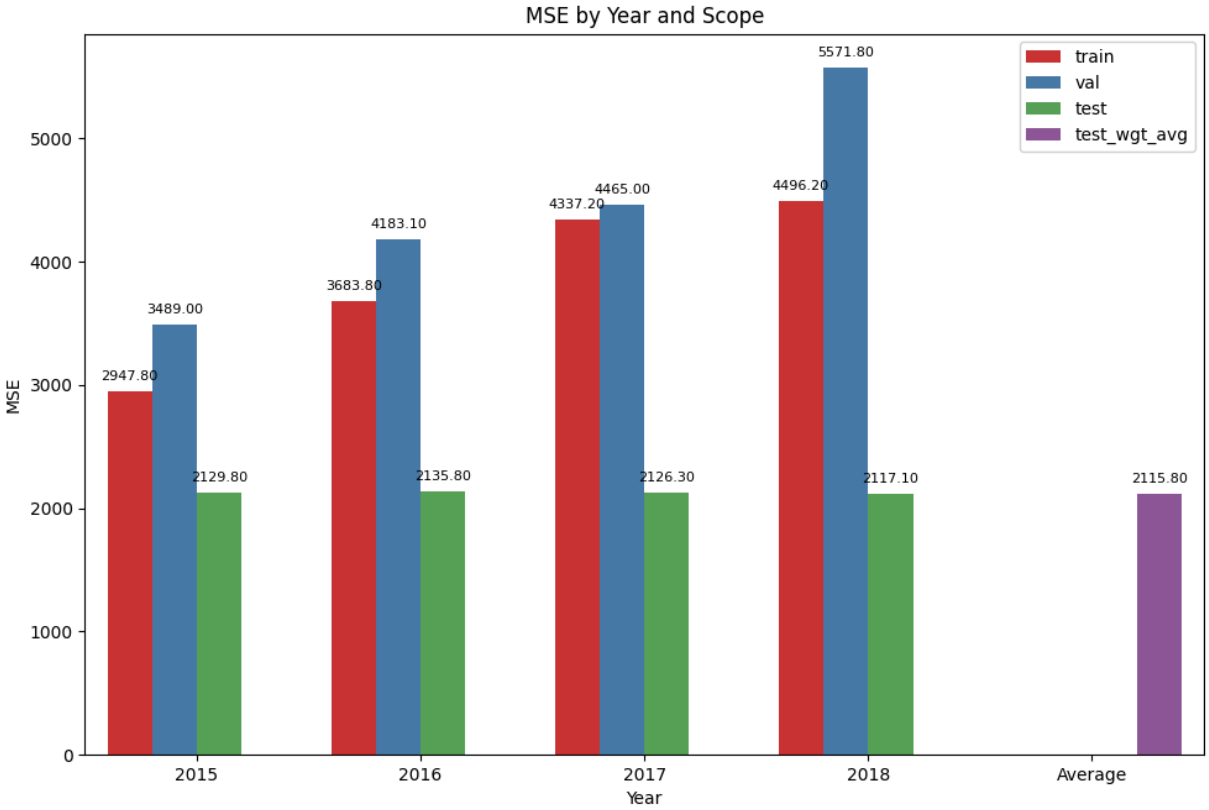

RECALL by Year and Scope
train
val
test
test_wgt_avg

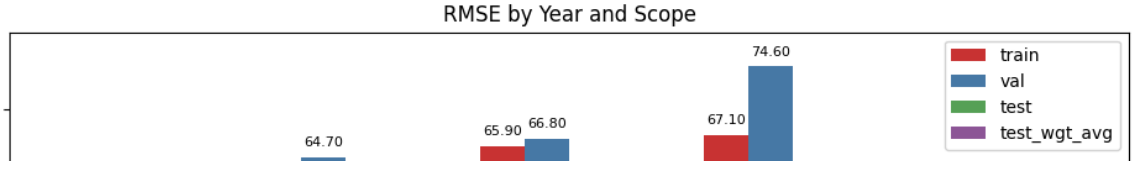


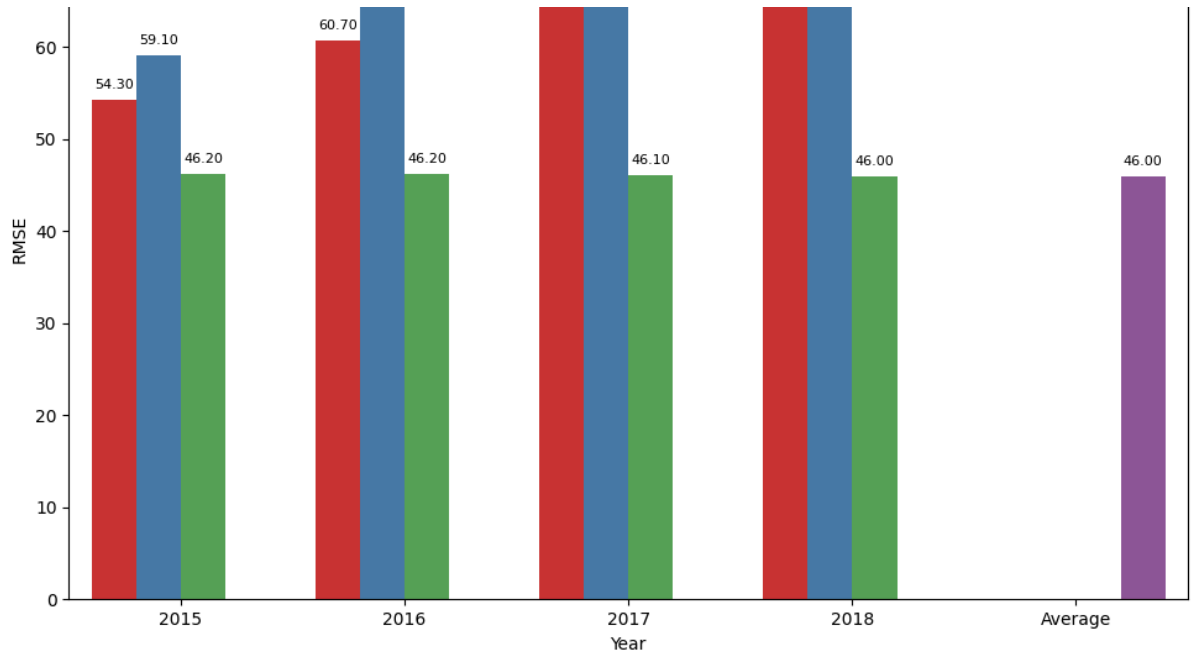


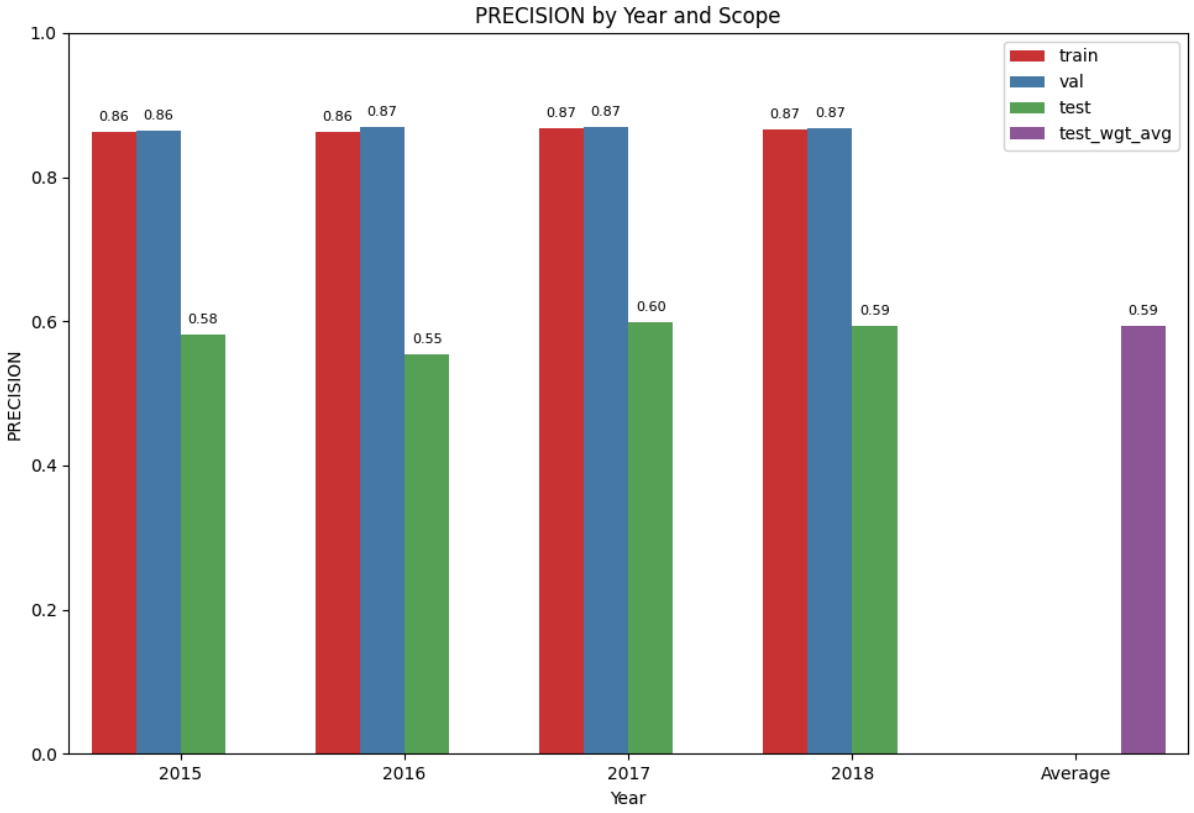
2. Improved Baseline

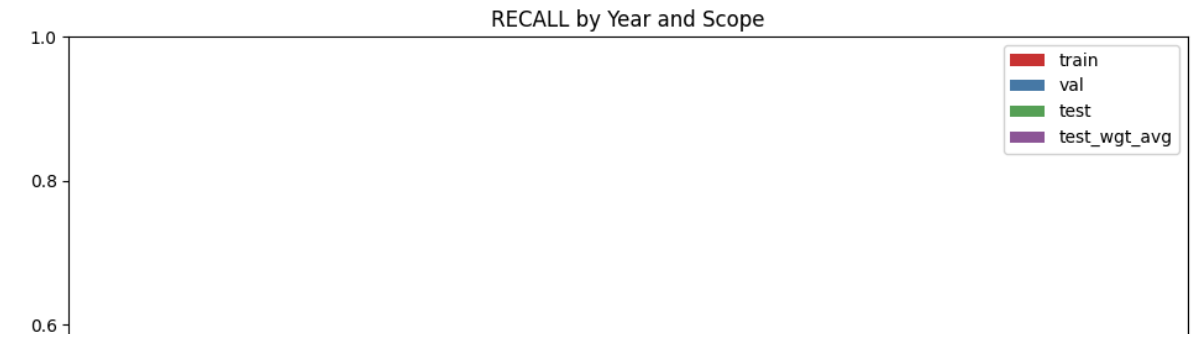

MAE by Year and Scope
train
val
test
test_wgt_avg
MAE
Year
2015
2016
2017
2018
Average
26.60
27.90
17.80
28.30
29.20
18.00
29.30
29.30
17.60
30.00
31.90
17.70
17.70

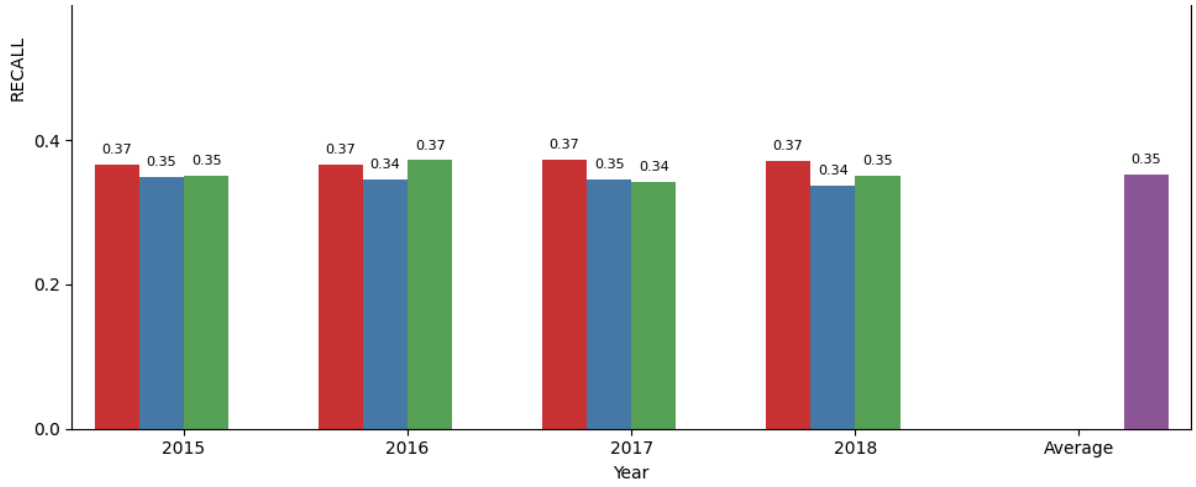

MSE by Year and Scope
train
val
test
test_wgt_avg
MSE
Year
2015
2016
2017
2018
Average
2947.80
3489.00
2129.80
3683.80
4183.10
2135.80
4337.20
4465.00
2126.30
4496.20
5571.80
2117.10
2115.80

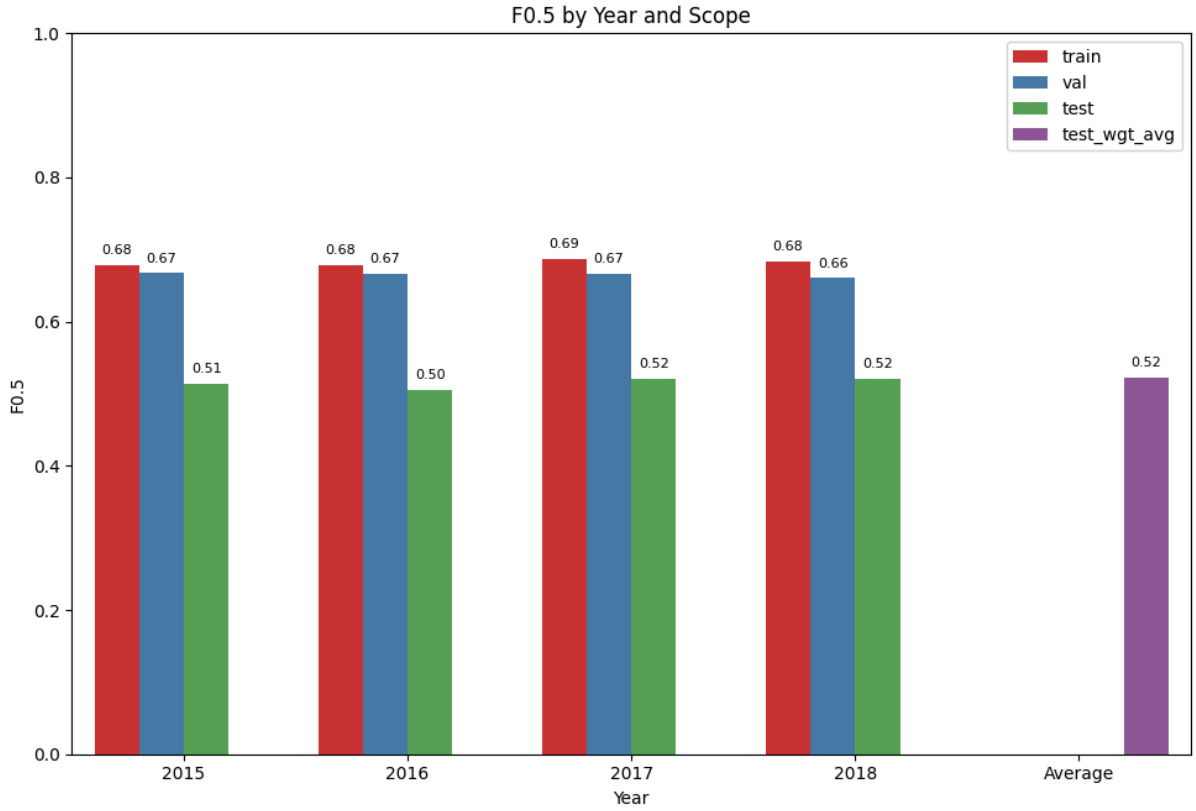

RMSE by Year and Scope
train
val
test
test_wgt_avg
64.70
65.90
66.80
67.10
74.60


RMSE
Year
2015
2016
2017
2018
Average
54.30
59.10
46.20
60.70
46.20
46.10
46.00
46.00

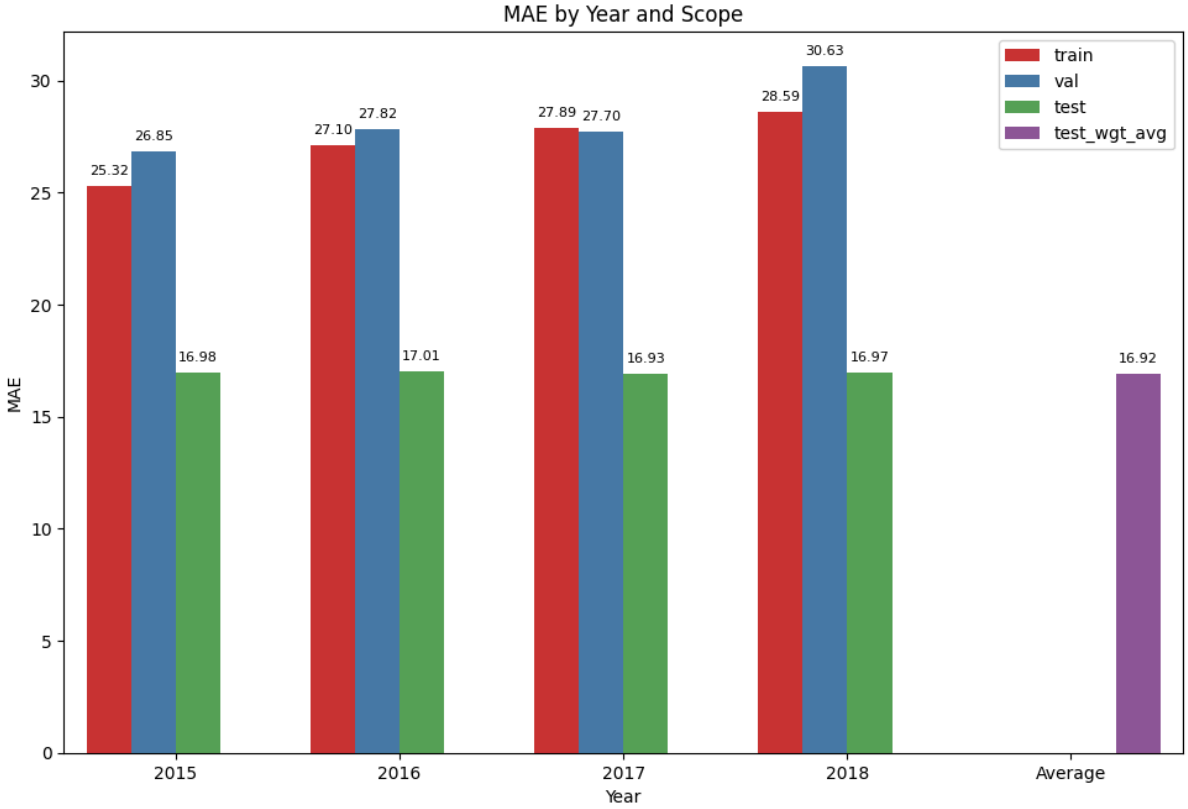

PRECISION by Year and Scope
PRECISION
Year
train
val
test
test_wgt_avg
2015
2016
2017
2018
Average
0.86 0.86 0.58
0.86 0.87 0.55
0.87 0.87 0.60
0.87 0.87 0.59
0.59

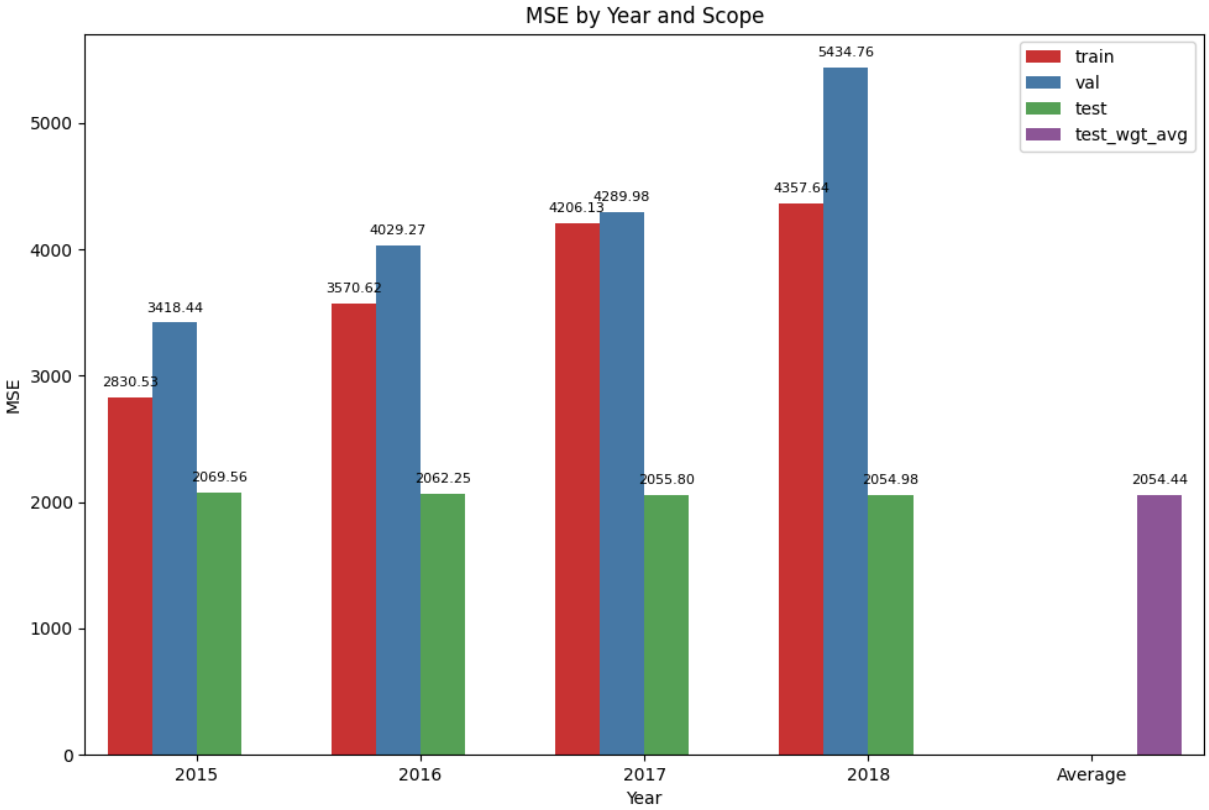

RECALL by Year and Scope
train
val
test
test_wgt_avg

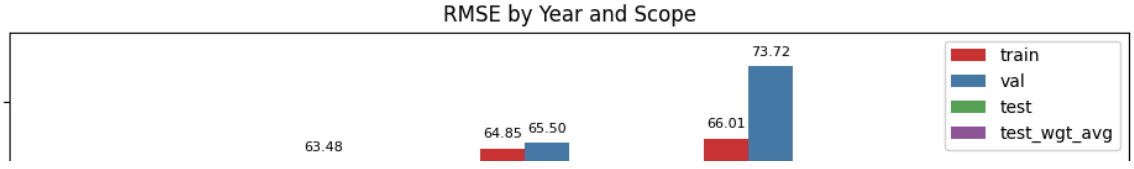


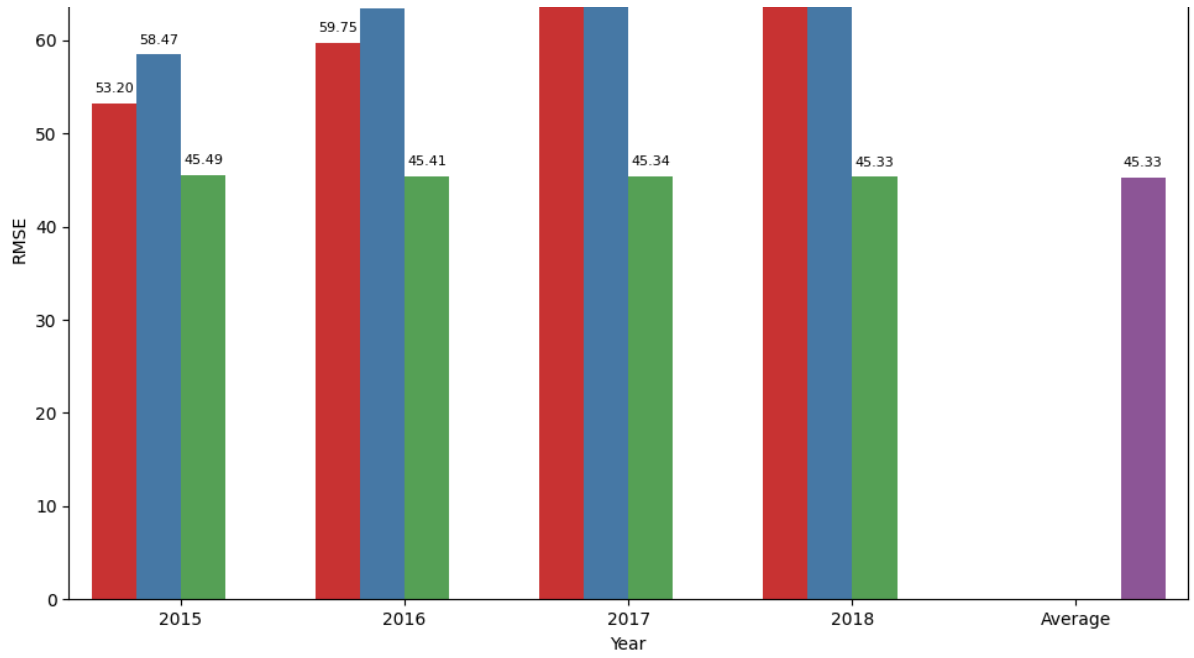


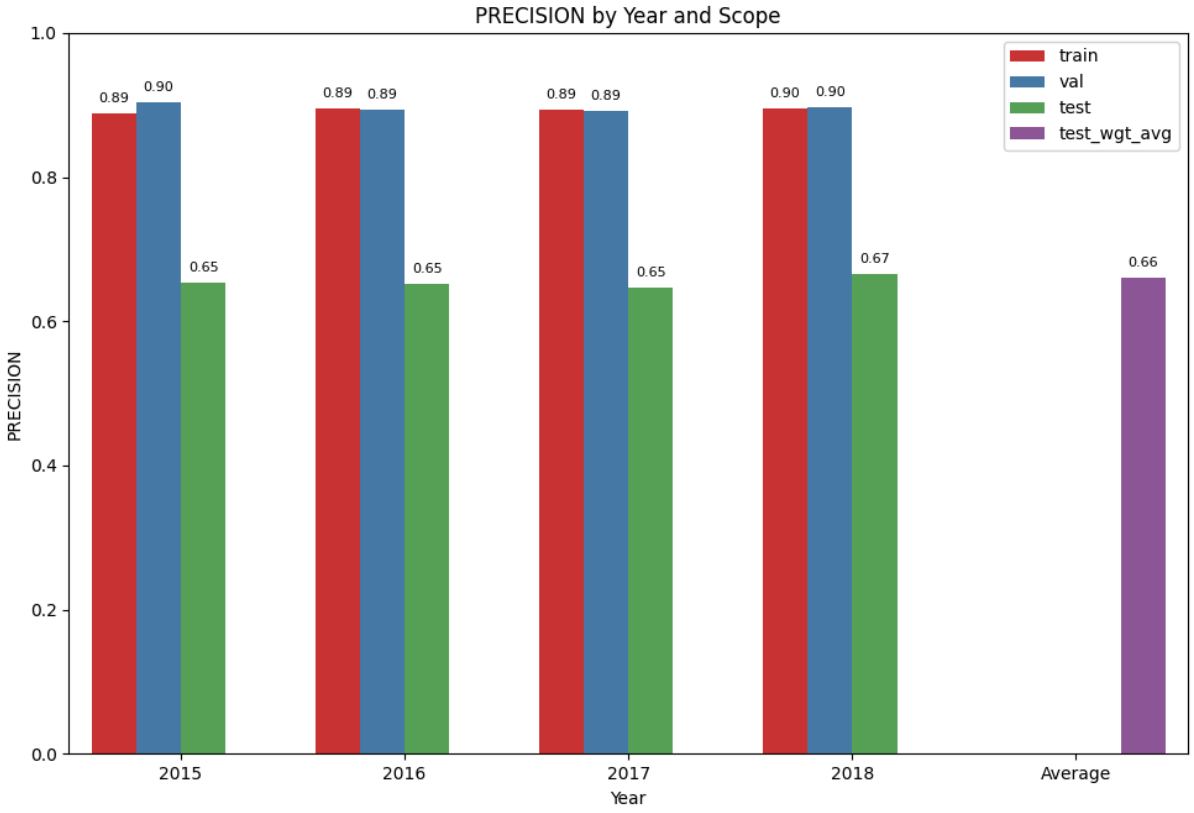
3. Random Forest

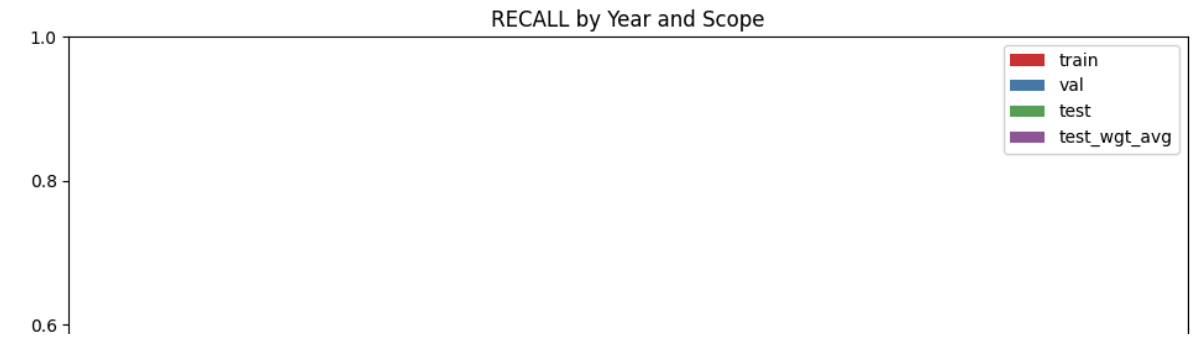

MAE by Year and Scope
train
val
test
test_wgt_avg
MAE
Year
2015
2016
2017
2018
Average
25.32
26.85
16.98
27.10
27.82
17.01
27.89
27.70
16.93
28.59
30.63
16.97
16.92

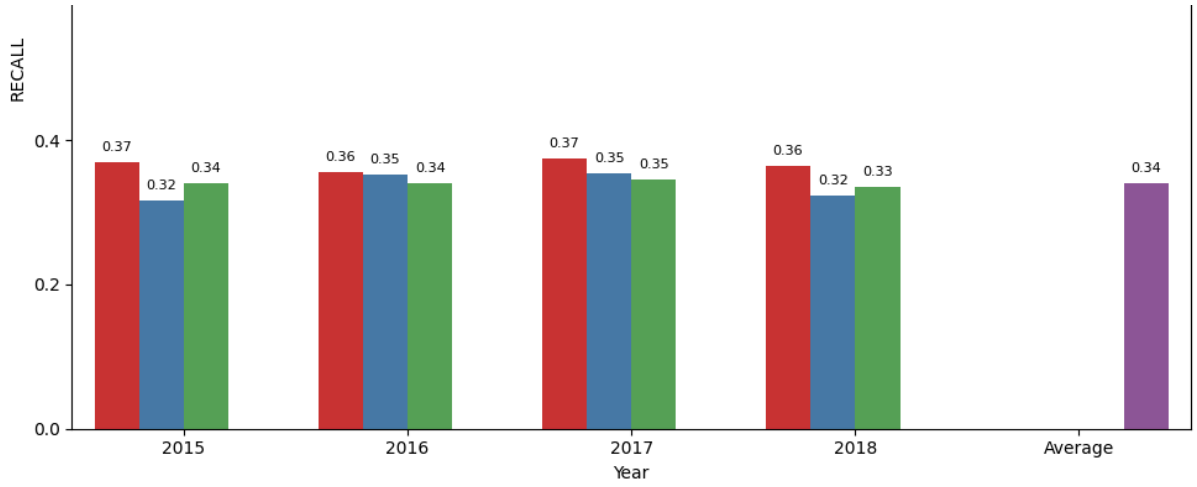

MSE by Year and Scope
train
val
test
test_wgt_avg
MSE
Year
2015
2016
2017
2018
Average
2830.53
3418.44
2069.56
3570.62
4029.27
2062.25
4206.13
4289.98
2055.80
4357.64
5434.76
2054.98
2054.44

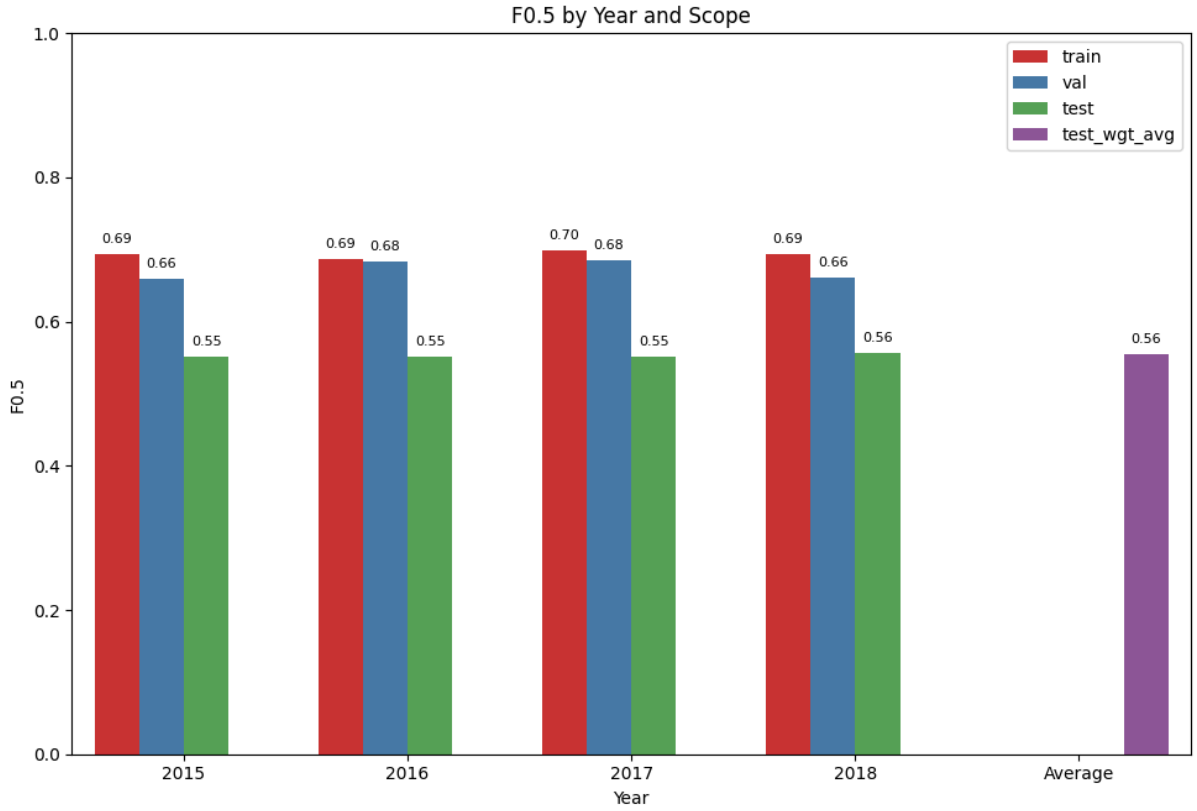

RMSE by Year and Scope
train
val
test
test_wgt_avg
63.48
64.85
65.50
66.01
73.72


RMSE
Year
2015
2016
2017
2018
Average
53.20
58.47
45.49
59.75
45.41
45.34
45.33
45.33

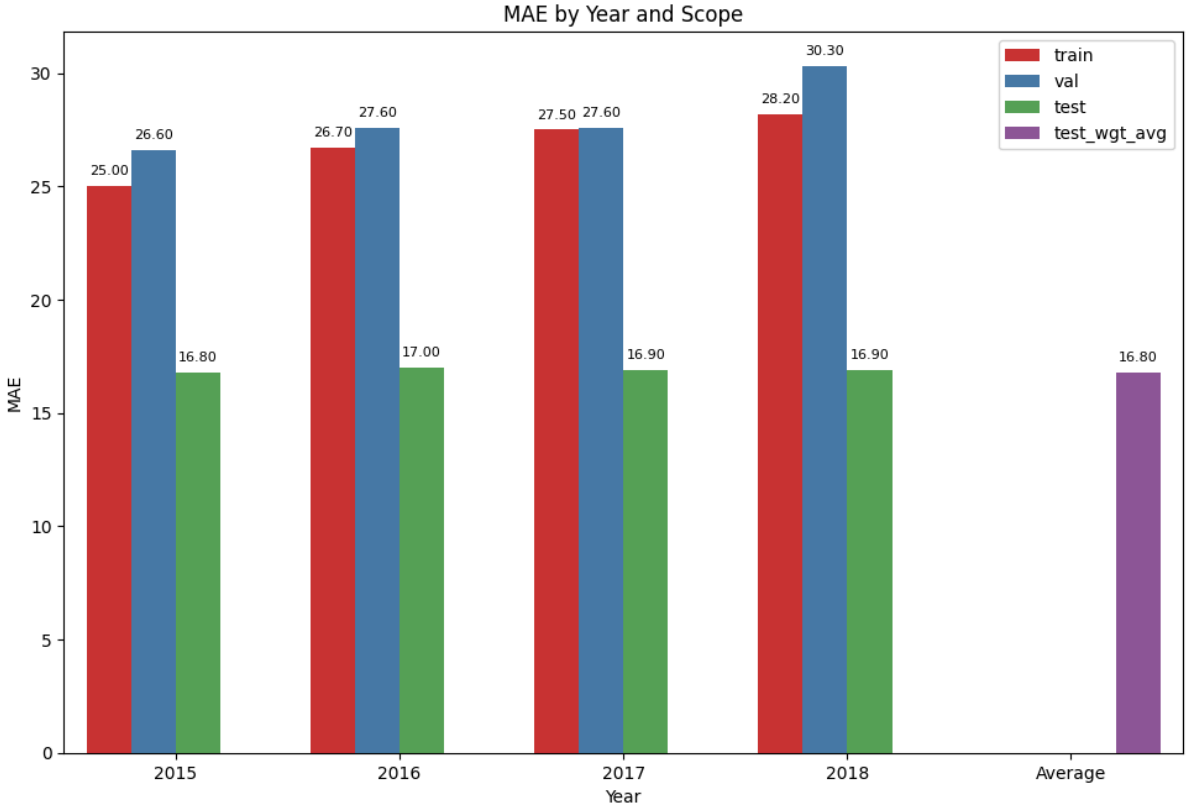

PRECISION by Year and Scope
PRECISION
Year
train
val
test
test_wgt_avg
0.89
0.90
0.65
0.89
0.89
0.65
0.89
0.89
0.65
0.90
0.90
0.67
0.66

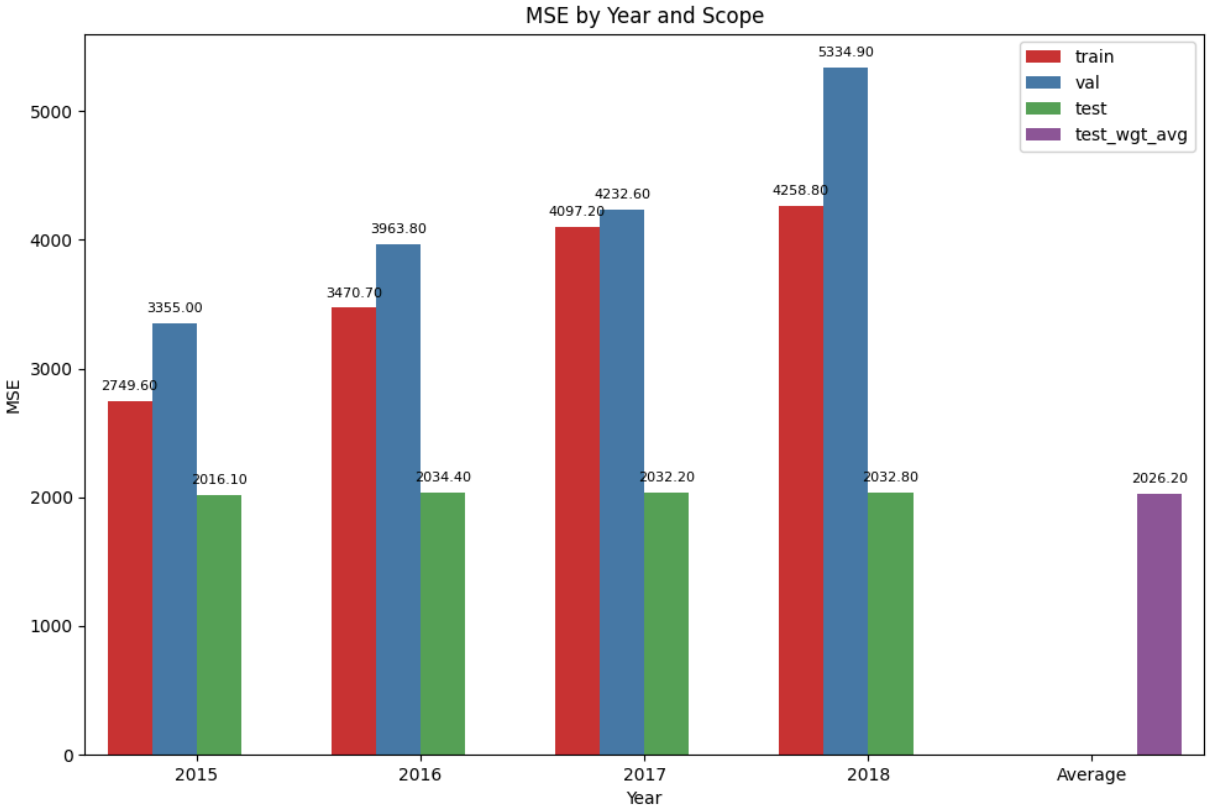

RECALL by Year and Scope
train
val
test
test_wgt_avg

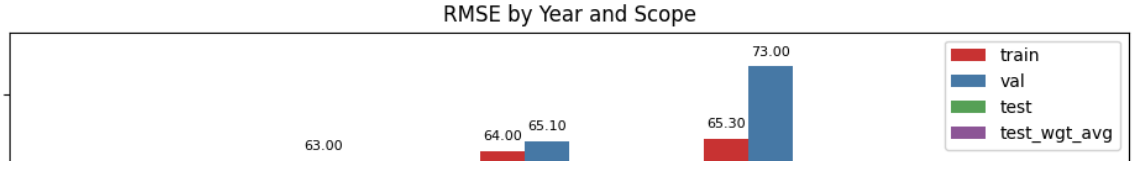


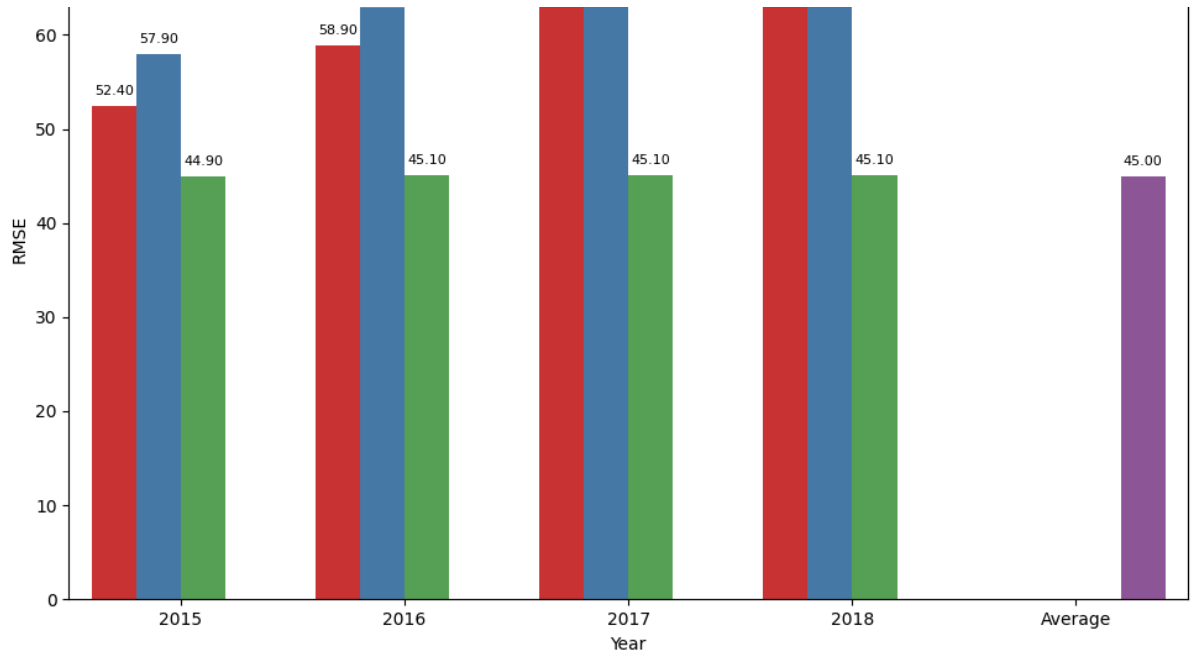


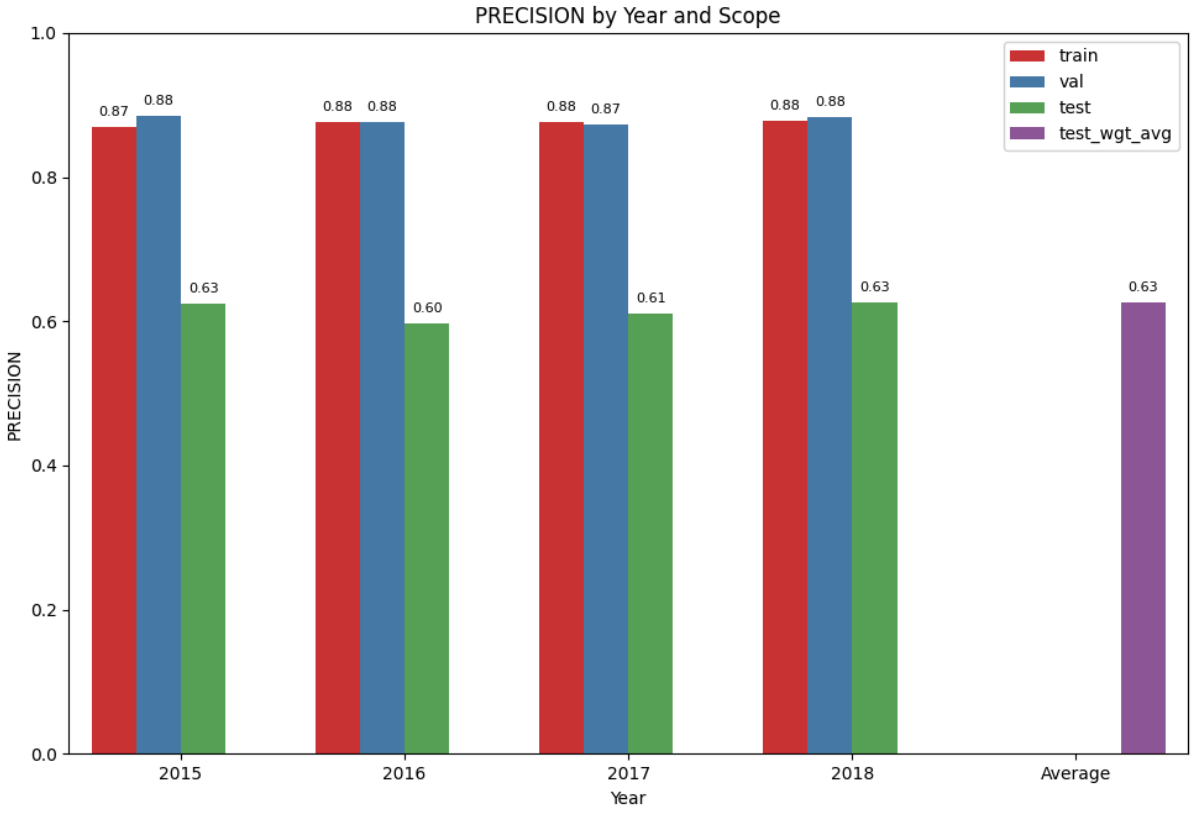
4. Gradient Boosted Tree Regression

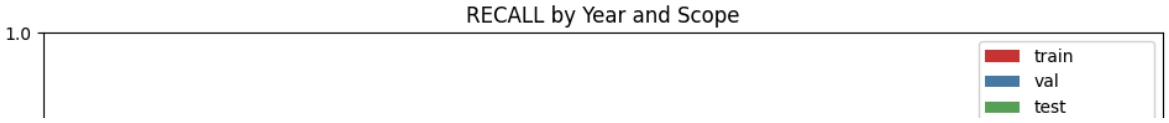

MAE by Year and Scope
train
val
test
test_wgt_avg
MAE
Year
2015
2016
2017
2018
Average
25.00
26.60
16.80
26.70
27.60
17.00
27.50
27.60
16.90
28.20
30.30
16.90
16.80


MSE by Year and Scope
train
val
test
test_wgt_avg
MSE
Year
2015
2016
2017
2018
Average
2749.60
3355.00
2016.10
3470.70
3963.80
2034.40
4097.20
4232.60
2032.20
4258.80
5334.90
2032.80
2026.20


RMSE by Year and Scope
train
val
test
test_wgt_avg
63.00
64.00
65.10
65.30
73.00







Appendix 2: Notebook Links

Phase 1

Team notebook

- Phase 1 report (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3295227484017544)

By Rachel Gao:

- w261_final_project_Phase1_Overview_RG (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2989157687081820/command/1800218785C
- w261_final_project_Phase1_OTPW_EDA_RG (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3295227484017931/command/3295227484C
- w261_final_project_Phase1_Airport_EDA_RG (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3295227484016866/command/3295227484
- w261_final_project_Phase1_EDA_RG (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3907228401243106/command/1800218785C

By Jenna Sparks:

- Phase1_weather_stations_EDA_JS (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1800218785099902)

By Dhyuti Ramadas:

- Flights EDA (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3295227484017584/command/3295227484

By Juliana Gómez Consuegra:

- 3m_OTPW_weather_EDA (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3295227484017656)

Phase 2

Team notebook

- Phase 2 report (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2513299033247568)

By Rachel Gao:

- 2015_1YR_OTPW_s1_GraphFrame_Visualize (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3944613696783393/command/3944613696
- 2015_1YR_OTPW_Train_nulls_EDA (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1835145152852872/command/18351451528
- 2015_1YR_OTPW_s1.1_Overall_EDA (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2513299033249705/command/2513299033
- 2015_1YR_OTPW_Train_Initial_Variable_Selection_support (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2513299033248162/command/2513299033
- Data Preprocessing Pipeline:
  - Step 0: Prepare Dataset (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2513299033247511)
  - Step 1.0: Select Columns & Preprocessing (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2513299033252113)
  - Step 1.0: Select Columns & Preprocessing_wo_imputation (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3944613696794151)
  - Step 1.1: Add arrival time in UTC, Holidays, & natural disasters (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1093733074362713)
  - Step 1.1: Add arrival time in UTC, Holidays, & natural disasters_wo_impute (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3944613696795112)
  - Step 2.1 Feature Engineering (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3944613696785431)

    - Step 2.1.HT Subset 2.1 for Hyperparameter Tuning_5YR (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3544923356432315)
    - Step 2.1a Run PageRank using GraphFrame (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1093733074357390)

By Ray Cao:

- Baseline Model - 1Y (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3944613696796844)

By Jenna Sparks:

- Phase2_raw_airports_EDA_JS (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2513299033252020)
- Phase2_raw_weather_stations_EDA (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/196965630959090/command/19696563095
- Phase2_1Y_OTPW_train_airports_and_stations_EDA (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1835145152851350)

By Dhyuti Ramadas:

- Time_Series_Forecasting_Prophet (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2513299033251756/command/3944613696

By Juliana Gómez Consuegra:

- 2015_1YR_OTPW_parquet_train_s1_weather_feature_engineering (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2513299033251720/command/2513299033
- 1Y_OTPW_train_weather_EDA (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2513299033247948)
- Raw weather EDA (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2513299033248551)

Phase 3

By Rachel Gao:

- 5YR_OTPW_s2.1_Prophet_Visualize (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717717960)
- 5YR_OTPW_s2.1_PageRank_Visualize (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3544923356437249/command/354492335
- 5YR_OTPW_s2.1_Overall_EDA (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3544923356433851/command/354492335
- 5YR_OTPW_null_cancelled_EDA (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3544923356431723/command/354492335
- Data Preprocessing Pipeline:
  - Step 0: Prepare Dataset_5YR (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3944613696800129)
  - Step 1.0: Select Columns & Preprocessing_5YR (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3944613696800301)
  - Step 1.1: Add events, natural disasters, & prior flight delays_5YR (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3544923356432145)
  - Step 2.1 Feature Engineering_5YR (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3544923356432120)
  - Step 2.1a Run PageRank using GraphFrame_5YR (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3544923356432000)
  - Step 2.1b Run Time-based features using Prophet_5YR (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3944613696799352)
- Self Join Raw Datasets

    - Step 1. Join airports & stations (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717720700)
    - Step 2. Join flights & step 1 (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717720700)
    - Step 2. Join flights & step 1_5YR (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717721381)
    - Step 2. Join flights & step 1_5YR_3hrs_prior (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717727169)
    - Step 3. Join weather & step 2 (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717722089)
    - Step 3. Join weather & step 2_5YR_2hrs_prior (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717726522)
    - Step 3. Join weather & step 2_5YR_3hrs_prior (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717727524)
    - Step 3. Join weather & step 2_5YR_4hrs_prior (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717725067)
- Process Self Joined Datasets & EDA (2hrs prior)
    - EDA 0: Train nulls EDA_self join 2hrs prior (/)
    - EDA 1: s2_1 overall EDA_self join 2hrs prior (/)
    - Step 0: Prepare Dataset_self join 2hrs prior (/)
    - Step 1.0: Select Columns & Preprocessing_self join 2hrs prior (/)
    - Step 1.1: Add events, natural disasters, & prior flight delays_self join 2hrs prior (/)
    - Step 2.1 Feature Engineering_self join 2hrs prior (/)
- Process Self Joined Datasets & EDA (3hrs prior)
    - EDA 0: Train nulls EDA_self join 3hrs prior (https://adb-4248444930383559.19.azuredatabricks.net/?

o=4248444930383559#notebook/2926108717727297)
    - EDA 1: s2_1 overall EDA_self join 3hrs prior (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717728161)
    - Step 0: Prepare Dataset_self join 3hrs prior (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717727347)
    - Step 1.0: Select Columns & Preprocessing_self join 3hrs prior (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717727384)
    - Step 1.1: Add events, natural disasters, & prior flight delays_self join 3hrs prior (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717727401)
    - Step 2.1 Feature Engineering_self join 3hrs prior (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717727326)
- Process Self Joined Datasets & EDA (4hrs prior)
    - EDA 0: Train nulls EDA_self join 4hrs prior (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717726629)
    - EDA 1: s2_1 overall EDA_self join 4hrs prior (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717727434)
    - Step 0: Prepare Dataset_self join 4hrs prior (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717726658)
    - Step 1.0: Select Columns & Preprocessing_self join 4hrs prior (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717726766)
    - Step 1.1: Add events, natural disasters, & prior flight delays_self join 4hrs prior (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717726926)
    - Step 2.1 Feature Engineering_self join 4hrs prior (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717726966)

By Ray Cao:

- Feature Selection with L1 - 5Y (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3544923356433789)
- Feature Selection with Decision Tree - 5Y (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3544923356436566/command/3544923356
- Baseline Model - 5Y (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3544923356432704)
- Improved Baseline with L2 - 5Y (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3544923356436607)
- Random Forest - 5Y (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717714103)
- Ensemble Data Pipeline (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717720551)
- Ensemble Pure Results - 5Y (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2340268507071807)
- Ensemble via MultilayerPerceptronClassifier- 5Y (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717726391)
- Self Join - Improved Baseline with L2 - 5Y (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717728631)
- Self Join - MultilayerPerceptronClassifier- 5Y (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717728528/command/29261087177
- Self Join - XGBoost Random Forest - 5Y (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717728495/command/29261087177
- Self Join - Random Forest - 5Y (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717728430/command/29261087177

By Jenna Sparks:

- Previous Flight Based Features (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3944613696799634/command/3544923356
- XGBoost Random Forest - 5Y (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717716591/command/2926108717711

By Dhyuti Ramadas:

- MultilayerPerceptronClassifier- 5Y (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2926108717718385/command/2926108717711

By Juliana Gómez Consuegra:

- additional_data_sourcing_weather_2020 (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/2769031167782415)
- additional_data_sourcing_weather_2021 (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3544923356435197)
- additional_data_sourcing_weather_2022 (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3544923356435283/command/3544923356
- additional_data_sourcing_weather_2023 (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3544923356435304/command/3544923356
- additional_data_sourcing_weather_full_2020_2023 (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3544923356435327/command/3544923356
- additional_data_sourcing_flights_2020 (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3544923356435367/command/3544923356
- additional_data_sourcing_flights_2021 (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3544923356435925/command/3544923356

- additional_data_sourcing_flights_2022 (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3544923356435950/command/3544923356
- additional_data_sourcing_flights_2023 (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3544923356435974/command/3544923356
- additional_data_sourcing_all_flights_2020_2023 (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3544923356435998/command/3544923356
- Events 2015-2019 (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/3544923356432940/command/3544923356
- Natural disasters, 2015-2019 (https://adb-4248444930383559.19.azuredatabricks.net/?

Appendix 3: Feature Dictionary

Description of variables, in order of appearance in this document.

Variable name	Description
`DEP_DELAY`	Departure delay in minutes
`DAY_OF_MONTH`	Day of month of departure
`DAY_OF_WEEK`	Day of week of departure
`ORIGIN`	Origin airport
`DEST`	Destination airport
`STATION`	Weather station id number
`origin_station_id`	Weather station id number
`DEL_DELAY15`	Binary feature, where 0 = departure not delayed by 15 minutes or more, 1 = departure delayed by 15 minutes or more

Variable name	Description
`HourlyWindSpeed`	Speed of the wind at the time of observation
`CRS_ELAPSE_TIME`	Scheduled amount of time (in minutes) between flight take-off and landing
`sched_depart_date_time`	Scheduled date and time of departure
`FL_DATE`	Scheduled departure date in YYYY-MM-DD format
`CRS_DEP_TIME`	Scheduled departure time in HHMM format
`sched_depart_time_UTC`	Scheduled departure time, in UTC
`event_flag`	Indicates 1 on a date when an American holiday took place, and the two days prior and after the event, and 0 on other dates
`drought_flag`	Indicates 1 on a date when a drought took place, on states affected by that drought, and 0 on other dates and states
`flooding_flag`	Indicates 1 on a date when flooding took place, on states affected by that flooding, and 0 on other dates and states
`freeze_flag`	Indicates 1 on a date when freezing took place, on states affected by that freezing, and 0 on other dates and states
`severe_storm_flag`	Indicates 1 on a date when a severe storm took place, on states affected by that severe storm, and 0 on other dates and states
`tropical_cyclone_flag`	Indicates 1 on a date when a tropical cyclone took place, on states affected by that tropical cyclone, and 0 on other dates and states

Variable name	Description
`wildfire_flag`	Indicates 1 on a date when a wildfire took place, on states affected by that wildfire, and 0 on other dates and states
`winter_storm_flag`	Indicates 1 on a date when a winter storm took place, on states affected by that winter storm, and 0 on other dates and states
`last_delay`	Previous aircraft's delay in minutes
`TAIL_NUM`	Aircraft's id number
`incoming_flight_delay_ratio`	The ratio of the `last_delay` divided by the minutes between departures sharing the same aircraft
`log_average_delay`	Average daily delay per airport-carrier combination within same 2-4 hour window, log transformed
`pagerank`	Airport connectivity based on number of incoming and outgoing flights
`trend`	The Prophet forecasted log departure delay trend, based on carrier and origin
`yhat`	The Prophet forecasted log departure delay, encompass both trend and seasonality, based on carrier and origin
`OP_UNIQUE_CARRIERS`	Airline id
`ORIG_CARRIER`	Groups flights based on origin airport-carrier combination
`HourlySkyCondition`	A composite variable consisting of cloud height and darkness information for up to 3 cloud layers
`SkyDarkness`	Darkness of sky at the highest layer, from 0 (clear skies) to 9 (no visibility)

Variable name	Description
`CloudHeight`	Height at which clouds begin to appear, at the highest layer
`CloudHeightandDarkness`	Composite variable of `CloudHeight` and `SkyDarkness`
`WindChill`	Feature engineered variable from `HourlyDryBulbTemperature` and `HourlyWindSpeed`
`HourlyDryBulbTemperature`	Hourly dry-bulb temperature and is commonly used as the standard air temperature reported
`HourlyWetBulbTemperature`	Hourly wet-bulb temperature
`HourlyDewPointTemperature`	The temperature to which a given parcel of air must be cooled at constant pressure and water vapor content in order for saturation to occur
`DISTANCE`	Distance between airports
`ELEVATION`	Elevation in meters over sea level
`HourlyStationPressure`	Hourly average station pressure (in inches of mercury, to hundredths)
`HourlyAltimeterSetting`	The pressure value to which an aircraft altimeter is set so that it will indicate the altitude relative to mean sea level of an aircraft on the ground at the location for which the value was determined.
`HourlySeaLevelPressure`	Hourly average sea level pressure (in inches of mercury, to hundredths)
`pca_time_distance`	Feature engineered variable from `CRS_ELAPSE_TIME` and `DISTANCE`

Variable name	Description
`pca_elevation_station_pressure`	Feature engineered variable from `ELEVATION` and `HourlyStationPressure`
`pca_altimeter_sea_level_pressure`	Feature engineered variable from `HourlyAltimeterSetting` and `HourlySeaLevelPressure`
`pca_dew_windchill_wet_temp`	Feature engineered variable from `WindChill`, `HourlyDryBulbTemperature`, `HourlyWetBulbTemperature`, and `HourlyDewPointTemperature`
`log_DEP_DELAY`	Outcome variable - log transformation of actual delay in minutes
`origin_airport_timezone`	Local timezone of departure