databricks Team 2-1 Final Notebook

(https://databricks.com)

Take Off with Data: Spark Airlines' Journey to Outsmart Flight Delays 🛫

Spark Airlines Flight Delay Prediction: Phase 3

W261 Final Project Section 2 - Team 1

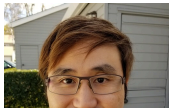
Team Members

Group number: Team 2-1

Phase Leader: Heesuk Jang

Team Members:

Name	Email	Photo
Heesuk Jang **(Phase 3 Leader)**	hankukin@berkeley.edu (mailto:hankukin@berkeley.edu)	
Stephanie Cabanela	scabanela@berkeley.edu (mailto:scabanela@berkeley.edu)	

Name	Email	Photo
Raymond Tang	raymond.tang@berkeley.edu	

Executive Summary

Flight delays create challenges with airport scheduling, passenger dissatisfaction, financial losses, and increased carbon emissions. At Spark Airlines, to better serve our passengers, our goal for this project is to build a binary classification model that predicts a 15-minute (or longer) departure delay two hours before the scheduled flight. Using the On-Time Performance Weather (OTPW) data from 2015-2019, we conducted EDA and statistical analysis to inspect dataset quality, conduct feature selection, and inform feature engineering. Then, we cleaned the data by dropping canceled flights, re-labelling disappearing and new carriers, imputing numeric features, and label/one-hot encoding categorical features. We

developed new features, notably features related to holidays and a lag feature tracking previous delays. We leveraged our baseline model pipeline developed in Phase 2 and extended it to conduct many experiments to compare the impact of various ML algorithms, our new features, and blocked time series cross validation:

1. Baseline model (Binomial Logistic Regression) + selected raw features only
2. Binomial Logistic Regression + new engineered features
3. Basic Decision Tree + new engineered features
4. Gradient Boosting Tree Ensemble + new engineered features
5. Random Forest Tree Ensemble + new engineered features
6. Multilayer Perceptron Classifier + new engineered features
7. XGBoost + new engineered features **(final model)**

Finally, we reported progress in terms of F2 Score (primary metric), precision, and recall. Our best pipeline model, XGBoost, yielded cross validated train F2 score = 0.6818, validation F2 score = 0.6327, and test F2

Note:

Please note that the Phase 3 Credit Assignment Plan is available at the very end of this report.

Project Description

Project Motivation and Task

At Spark Airlines, we believe in the transformative power of travel in connecting people to the world and we are dedicated to doing the right thing for our customers, communities, and planet. To achieve this, we

leverage data science and machine learning to understand how we can better serve our passengers and planet through data-driven decisions. In particular, we are interested in addressing flight delays, a well-known issue that leads to disruptions in airline and airport scheduling, passenger dissatisfaction, financial and time losses, and increased carbon emissions. As a result, to better serve our passengers, our goal is to build a binary classification model that will predict whether or not a departure delay will happen two hours before the scheduled flight. We define a delay as a 15-minute delay or longer from the scheduled departure time. A prediction should be retrieved from the model two hours before the scheduled departure time to give airlines and airports time to reassess and notify passengers in a timely manner.

Evaluation Metrics

For our industry, a false negative (where the flight is delayed, but the model fails to predict it) means high recall is crucial to identifying as many actual delays as possible. Balancing this with decent precision is also important, although it may not require as much emphasis as recall. Since maintaining customer satisfaction by providing accurate information about flight delays is our primary business objective, we wanted to have a weighting that prioritizes recall over precision.

Primary Metric: F2 Score

This is why we went with F2 score - a weighted harmonic mean between recall and precision where the beta value is 2.0. This means that recall is weighted twice as much as precision and this takes data imbalance into account. Another reason why we went with F_β score with beta=2.0 is because of the significant class imbalance between the positive class (1 for

15+ min delay) and the negative class (0, <15min delay) since airport departure delays are naturally the minority class in reality. Since our focus is the positive class for delays, we report F2 score in terms of the positive label `fMeasureByLabel(1.0, 2.0)`. The equation for the FBeta and F2 score is as follows:

- $F_\beta\ Score = (1 + \beta^2) * \frac{precision*recall}{(\beta^2*precision)+recall}$
- $F_2\ Score = 5 * \frac{precision*recall}{(4*precision)+recall}$

Secondary Metrics: Recall and Precision

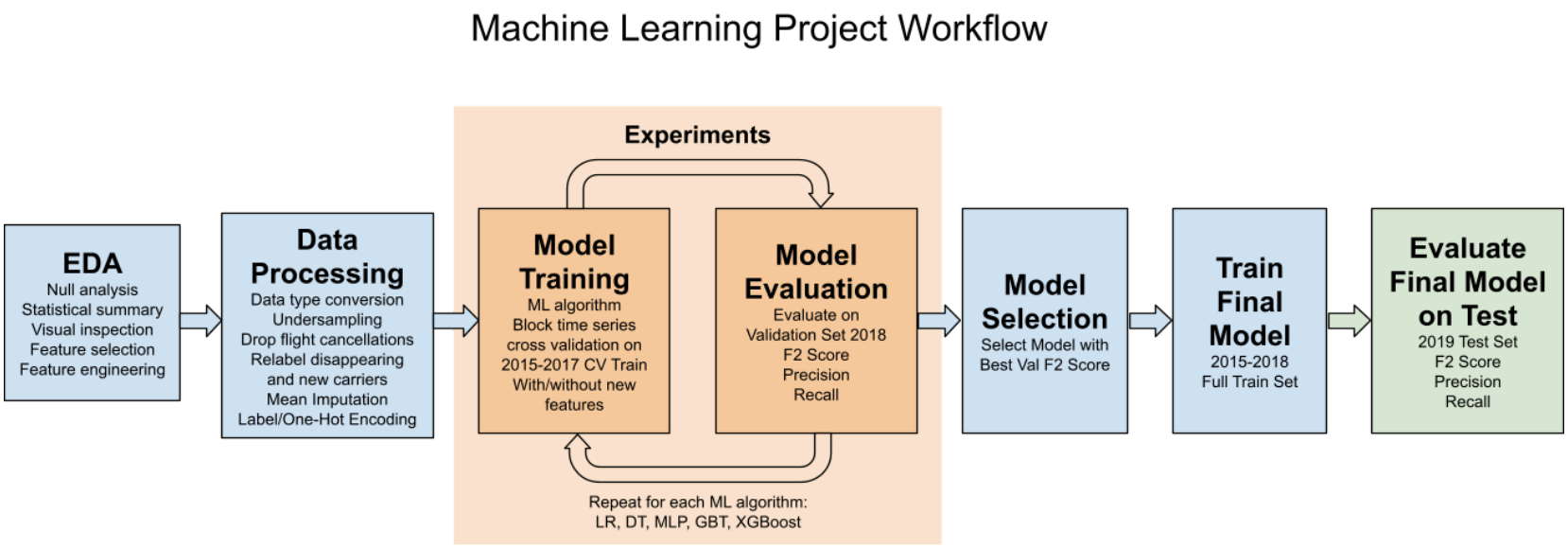
We also report recall and precision to get a deeper sense of where the model is performing. Since our airline would want to minimize disruptions caused by delays, we would want a high recall to ensure we can plan around most delays. At the same time, we would not want to expend resources preparing for delays that do not occur, so precision is also important. Since

our focus is the positive class for delays, we report precision and recall in terms of the positive label `precisionByLabel(1.0)` and `recallByLabel(1.0)`. Their equations are provided below:

- $Precision = \frac{TP}{TP+FP}$
- $Recall = \frac{TP}{TP+FN}$
- where TP is the number of true positives, FP is the number of false positives, and FN is the number of false negatives.

Machine Learning Project Workflow: Overview

The diagram below shows a high-level overview of the workflow steps we conducted for this machine learning project, which we will explain more in detail for throughout this report.



Data Description

We are primarily working with a joined time series dataset called OTPW consisting of on-time flight performance and weather data from 2015 to 2019. This 5 year raw dataset has 216 columns representing categorical and numerical features, and 31.7 million rows for the time period 2015-2019.

Exploratory Data Analysis (EDA)

Overall, we conducted exploratory data analysis (EDA) on the 4-year training set to get a sense of the data we're working with. We inspected data quality in terms of missing values and bad data. We visualized distributions of features and searched for any interesting relationships with the target variable DEP_DEL15. We provide a dataset description below. Then, we include visualizations and findings from our EDA that we used to inform our feature selection, feature engineering, and data processing.

Dataset Description

Train/Test Split

We used several splits of the 5-year OTPW dataset at various steps of this machine learning project. For our experiments, we created a train split for cross validation from 2015-2017 and used 2018 as our validation set for

model selection. Then, after selecting our best model out of all experiments, we use the full 2015-2018 train set for training our final model and evaluate on 2019 data as the test set. These splits and their sizes are as follows:

Set Name	Time Period	# Rows	Description
CV Train Set	Jan 2015 - Dec 2017	6,019,064 (after under-sampling)	Train set used for cross validation during experiments
Validation Set	Jan 2018 - Dec 2018	7,083,638	Validation set used for model selection from experiments
Full Train Set	Jan 2015 - Dec 2018	8,622,680 (after under-sampling)	Train set used for final model

Set Name	Time Period	# Rows	Description
Test Set	Jan 2019 - Dece 2019	7,259,155	Test set used for evaluating final model at the very end

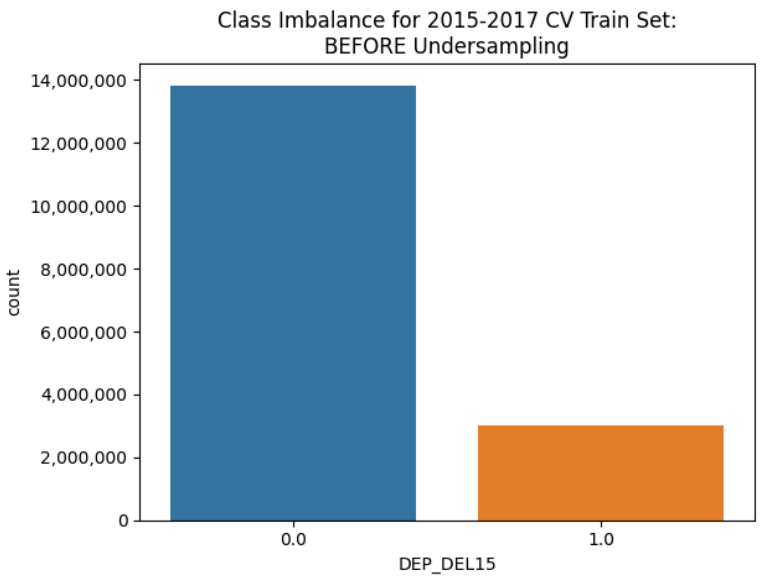
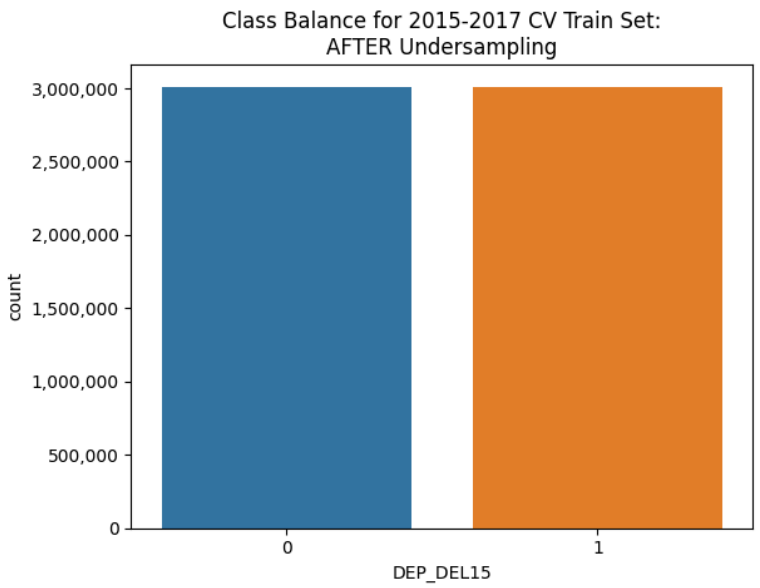
Class Imbalance

The class imbalance analysis for our 4-Year Train dataset revealed a substantial skew in the data: instances of on-time flights (label '0') vastly outnumbered the delayed flights (label '1'). Specifically, there were five times as many on-time flight observations as there were delays. To address this disparity and improve the model's ability to learn from an equitable distribution of data, we employed an under-sampling strategy.

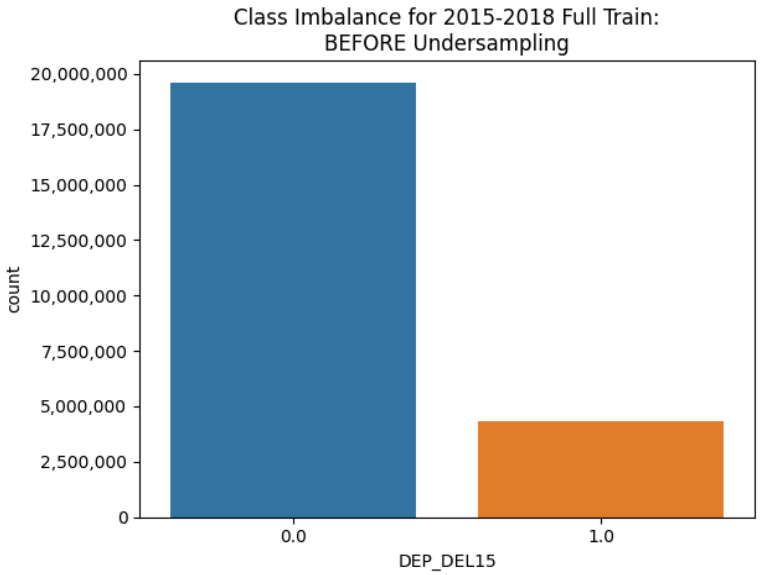
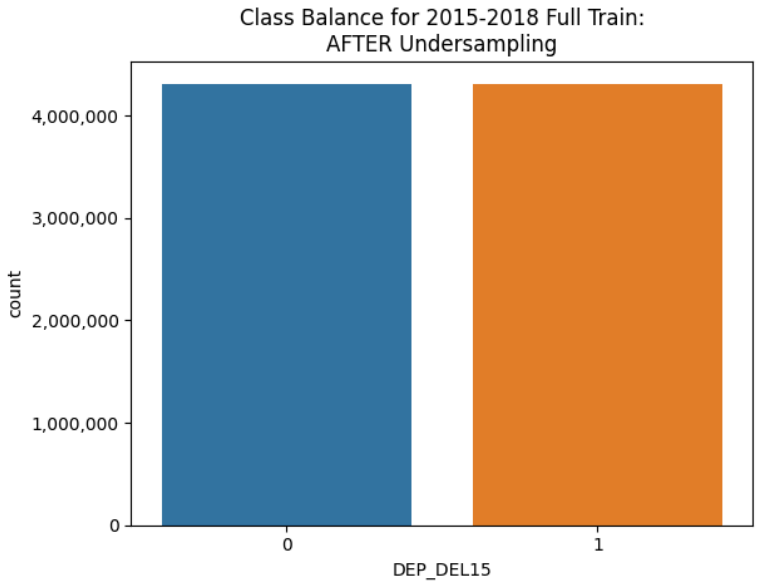
After under-sampling was performed, the class distribution was adjusted to balance the previously disproportionate dataset. The revised dataset presents a more uniform comparison, reducing the number of on-time flight

instances to align more closely with the number of delays. This rebalancing is critical for ensuring our model does not develop a bias toward predicting flights as on-time, thereby increasing its predictive accuracy for flight delays. We find that despite discarding a lot of data, the under-sampling technique is beneficial from a big data perspective: we still have substantial data remaining and this would decrease training runtime.

We performed under-sampling on the 2015-2017 train set used for cross validation:

Before Under-sampling: Train CV	After Under-sampling: Train CV
	

We also performed undersampling on the 2015-2018 full train set used for training the final model:

Before Under-sampling: Full Train	After Under-sampling: Full Train
	

Data and Feature Engineering

We primarily worked with the joined time series dataset OTPW, which combines on-time flight performance and weather data from 2015 to 2019. We provide a summary of its data lineage and key data transformations below:

1. Data Source:

- ***Source:*** The dataset originates from two primary sources: on-time flight performance data and weather data.
- ***Time Period:*** The data covers the span from 2015 to 2019.
- ***Volume and Structure:*** The raw dataset comprises a substantial volume, with 216 columns featuring a mix of categorical and numerical data, totaling 31.7 million rows.

2. Handling Data Scale:

- Due to the large size of the dataset, smaller subsets (3 months and 1 year) were utilized for initial exploratory data analysis (EDA) and experiments. This strategy helps in managing computational load and obtaining preliminary insights before addressing the complete dataset.

2. Data Cleaning and Normalization:

- ***Handling Missing Values:*** Identifying and either imputing or removing missing data to ensure data quality.

    - *Additional Data Cleanup:* Bad data that could cause discrepancies, such as any non-numeric characters like "V" within the HourlyPrecipitation values, were removed.
    - *Normalize Operating Carriers:* All canceled flights were removed, and carriers that emerged later in the given timeframe were relabeled accordingly.
- ***Normalization:*** Numerical features, such as HourlyPrecipitation and HourlyWindSpeed, were scaled to a standard range.

3. Data Categorization and Encoding:

- ***Categorical Encoding:*** Categorical data, such as unique carrier codes or flight diversion statuses, were converted into numerical formats using StringIndexer or OneHotEncoder.
- ***Binning:*** Continuous data were categorized, for instance by grouping different time intervals of a day or categorizing days of the week by delay frequency.

4. Feature Selection and EDA:

- ***Exploratory Data Analysis:*** Conducted on smaller datasets to identify patterns, anomalies, correlations, and a relevant subset of features for the classification task.
- ***Feature Relevance:*** Recognizing that not all 216 features may be relevant or directly correlated, possibly introducing noise into the model. A subset of the most impactful features was selected based on their perceived relevance to the task, informed by insights from EDA, correlation analysis, and LASSO regression.

5. Feature Engineering:

- ***Creating New Features:*** New variables that could be more predictive of outcomes were generated. For example, calculating the distance from the most impactful holiday and determining the optimal distance as indicated by the model runs that produce the best results, or creating a binary feature to indicate whether the previous flight was delayed by 15 minutes or more before a 2-hour departure.

- ***Aggregating Data:*** Data were compiled into meaningful categories, such as grouping carriers by the number of flights operated, or hourly visibility of less than or greater than or equal to 10 miles, as determined by the EDA boxplot insights.

Description of Feature Families Explored

TIME:

- This family captures temporal and seasonal aspects of the flight, including the scheduled departure time, year, month, day of the month, and day of the week. These features provide chronological time information for the flights.

NUMERIC (StandardScaled):

- Encompassing various numerical attributes, this family includes scaled measurements such as distance, elevation, and scheduled elapsed time. It also incorporates meteorological factors like hourly wind speed, precipitation, relative humidity, and visibility.

CATEGORICAL (StringIdx/OneHot):

- This includes nominal and ordinal categorical attributes, representing discrete information related to flights. It includes indicators for whether a flight was diverted, the destination airport, unique carrier/operator, origin airport, tail number of the aircraft, carrier size, day type (e.g., weekday, weekend), degree of visibility, and a categorical representation of flight distance groups.

Data Dictionary

Below is a data dictionary of our all features inputted to the model, including: selected features from the original OTPW columns and new features produced during feature engineering. This data dictionary includes a description, justification for including in the model, and summary statistics for each column. This list of features was produced as a result of EDA and statistical analysis findings detailed after this table.

Column Name	Description	Justification/Mc
YEAR	Year of weather observation	Necessary for series data; capturing cycl nature of dela
MONTH	Month of weather observation	Necessary for series data.
DAY_OF_MONTH	Flight date part - day of the month	Necessary for series data.
DAY_OF_WEEK	Flight date part - day of week	Necessary for series data; capturing cycl nature of dela

Column Name	Description	Justification/Mc
CRS_DEP_TIME	CRS (Computer Reservation System) Departure Time (local time: hhmm)	Necessary for series data.
DISTANCE	Distance between airports (miles)	Flight prepara implications; increased dist might lead to r delays.

Column Name	Description	Justification/Mo
ELEVATION	The height at which the aircraft is flying above a fixed reference point, usually sea level	Assess the imp elevation on predicting flig departure dela
CRS_ELAPSED_TIME	CRS Elapsed Time of Flight, measured in minutes, refers to the planned flight duration as determined by the pilot	Key element influencing flig punctuality.

Column Name	Description	Justification/Mo
HourlyWindSpeed	Speed of the wind at the time of observation given in miles per hour (mph)	Wind speed ca affect aircraft might lead to c
HourlyPrecipitation	Amount of precipitation in inches to hundredths over the past hour	Precipitation n impact passer aircrafts, and c
HourlyRelativeHumidity	The relative humidity given to the nearest whole percentage	Humidity migh impact passer aircrafts, and c

Column Name	Description	Justification/Mo
HourlyVisibility	The horizontal distance an object can be seen and identified, given in whole miles	Poor visibility r impact pilots b departure, lea delays.
CRS_DEP_HOUR_cos	CRS Departure Hour (cos transformation)	Transform line data into a for reflecting cycl events.
CRS_DEP_HOUR_sin	CRS Departure Hour (sin transformation)	Transform line data into a for reflecting cycl events.

Column Name	Description	Justification/Mo
DAY_HOUR_interaction	Combination of day of the week and hour of the day	Capture comb effect of both and hour on fli delays.
DAY_OF_WEEK_cos	Flight date part - day of the week (cos transformation)	Necessary for series data; capturing the cyclical nature delays.
DAY_OF_WEEK_sin	Flight date part - day of the week (sin transformation)	Necessary for series data; capturing the cyclical nature delays.

Column Name	Description	Justification/Mo
TIME_INTERVAL_OF_DAY	New Feature: Time intervals of the day, derived from KMeans clustering	Segment the d into intervals t identify patter specific to diff parts.
OP_UNIQUE_CARRIER	Unique airline company name	Useful for ana and presentati
ORIGIN	Origin Airport	Analyzing hist data in relatior origin and destination air
DEST	Destination Airport	Analyzing hist data in relatior origin and destination air

Column Name	Description	Justification/Mo
TAIL_NUM	Tail Number, a unique alphanumeric identifier assigned to an aircraft	Uniquely ident each aircraft f analyzing dela patterns.
CARRIER_SIZE	New Feature: The size of carriers (airlines) categorized based on the number of flights they operate	Categorizatior based on fligh volume.

Column Name	Description	Justification/Mo
DAY_TYPE	New Feature: Derived from the day of the week. It categorizes days into 'weekend' and 'weekday'	Captures diffe patterns in flig departure dela between week and weekends
DEGREE_VISIBILITY	New Feature: Visibility from a flight measured in miles (low or high)	Analyze likelih delays based visibility.

Column Name	Description	Justification/Mo
FL_DISTANCE_GROUP	New Feature: Flight distance in miles, categorized	Different flight lengths affect fueling time, boarding proc and air traffic patterns.
5_DAYS_DIST_FROM_Independence	New Feature: A seasonality-based feature that considers a range of 5 days before and after July 4th	To capture the nuances of ho related travel patterns arour 4th, crucial for accurately pre flight delays d these peak tim

Column Name	Description	Justification/Mo
7_DAYS_DIST_FROM_Christmas	New Feature: A seasonality-based feature that considers a range of 7 days before and after Christmas	To capture the nuances of ho related travel patterns arour Christmas, cru for accurately predicting fligl delays during peak times.
7_DAYS_DIST_FROM_NewYear	New Feature: A seasonality-based feature that considers a range of 7 days before and after New Year's day	To capture the nuances of ho related travel patterns arour Year's day, cru for accurately predicting fligl delays during peak times.

Column Name	Description	Justification/Mc
dep_del15_2hr_before	New Feature: A time-based feature indicating whether a departure delay of 15 minutes occurred (1) or not (0) within the last 2 hours in the same origin airport.	Previous recer delays within t same origin ai may affect de delays of subsequent fli

LASSO Feature Selection

Next, we performed LASSO feature selection. The coefficients produced allow us insights into features that have a large effect on our target variable and indicate the variables that are strong candidates to be selected in feature selection. The features with 0 as coefficients were removed from our list of selected features.

Here is our table that outputs the coefficients developed by LASSO feature selection:

Features	Coefficients
dep_del15_2hr_before	0.5359
HourlyRelativeHumidity_imputed	0.0249
CRS_DEP_TIME	0.0089

Features	Coefficients
7_DAYS_DIST_FROM_NewYear_idx	0.0044
HourlyWindSpeed_imputed	0.0029
ORIGIN_idx	0.0009
HourlyPrecipitation_imputed	0.0007
DAY_OF_WEEK_cos	0
ELEVATION	0
7_DAYS_DIST_FROM_Christmas_idx	0
5_DAYS_DIST_FROM_Independence_idx	0
FL_DISTANCE_GROUP_idx	0
DEGREE_VISIBILITY_idx	0
DAY_TYPE_idx	0

Features	Coefficients
CARRIER_SIZE_idx	0
OP_UNIQUE_CARRIER_idx	0
DIVERTED	0
MONTH	0
DISTANCE	0
CRS_ELAPSED_TIME_imputed	0
DAY_OF_WEEK_sin	0
DAY_HOUR_interaction	0
CRS_DEP_HOUR_cos	0
DAY_OF_MONTH	0
TAIL_NUM_idx	-0.0000033

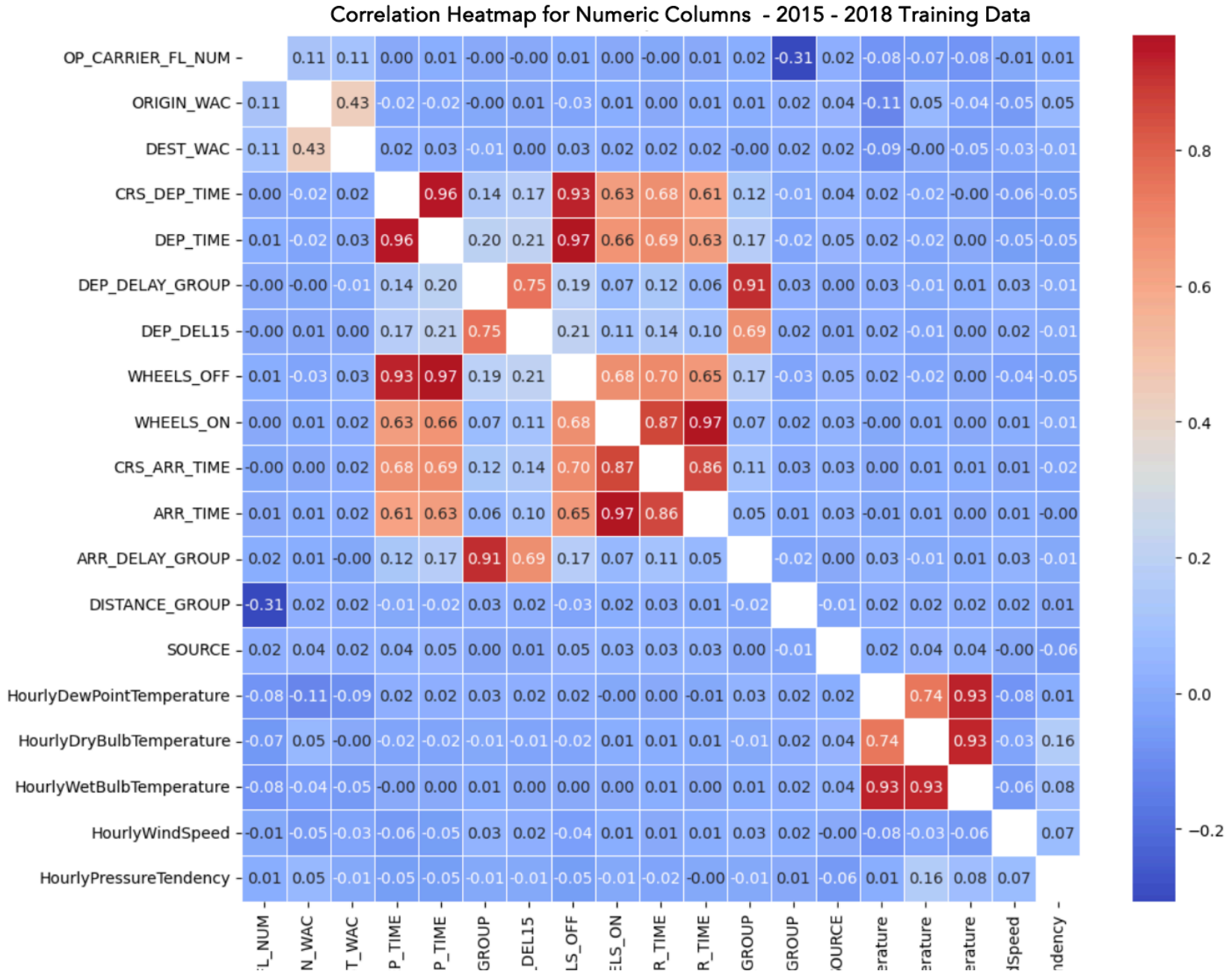
Features	Coefficients
DEST_idx	-0.0002647
TIME_INTERVAL_OF_DAY	-0.0007544
HourlyVisibility_imputed	-0.0046
CRS_DEP_HOUR_sin	-0.0536

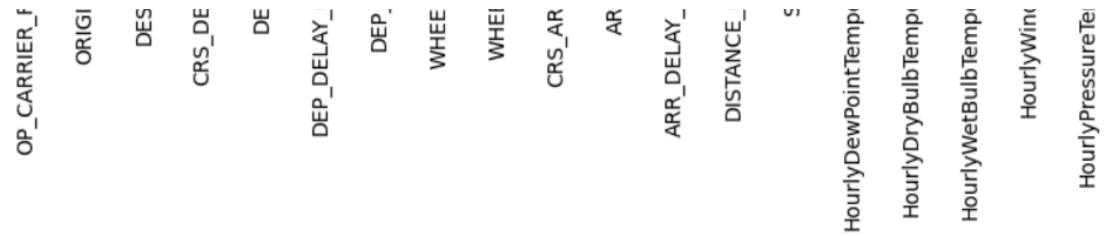
Pearson Correlation Analysis

Next, we ran a Pearson correlation analysis for the numeric variables of importance with Dep_TIME, which acts as proxy for our target variable, DEP_DEL15. This is because our target variable is a binary variable, which would not add as much value to our exploration. We found that 66% of numeric features had a positive correlation with the target variable and 52% had a correlation between 0 and 20%. We found the top 5 features in regards to highest correlation to the target variable and found that we

cannot use them due to data leakage. This is because if you include variables that are influenced by the target variable and occur after the target event in time or are derived directly from the target variable it can introduce leakage.

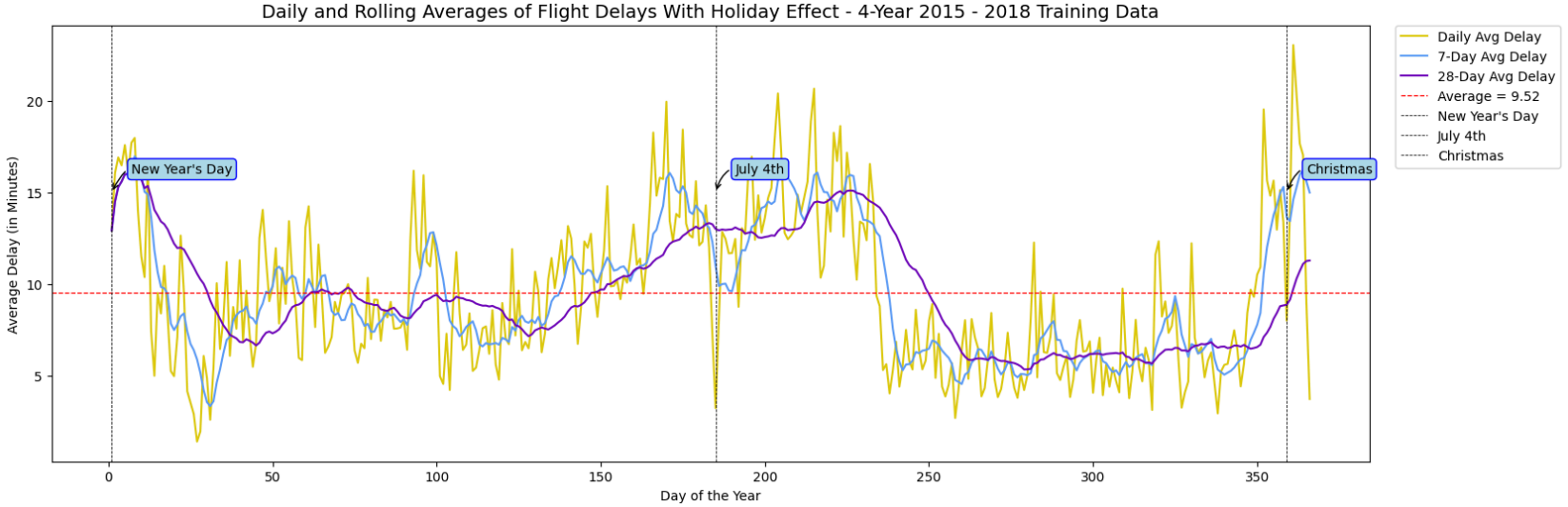
Below is a heatmap resulting from this Pearson Correlation Analysis with 2015-2018 Training Data Using Numeric Variables:


Correlation Heatmap for Numeric Columns - 2015 - 2018 Training Data
OP_CARRIER_FL_NUM
ORIGIN_WAC
DEST_WAC
CRS_DEP_TIME
DEP_TIME
DEP_DELAY_GROUP
DEP_DEL15
WHEELS_OFF
WHEELS_ON
CRS_ARR_TIME
ARR_TIME
ARR_DELAY_GROUP
DISTANCE_GROUP
SOURCE
HourlyDewPointTemperature
HourlyDryBulbTemperature
HourlyWetBulbTemperature
HourlyWindSpeed
HourlyPressureTendency
0.8
0.6
0.4
0.2
0.0
−0.2



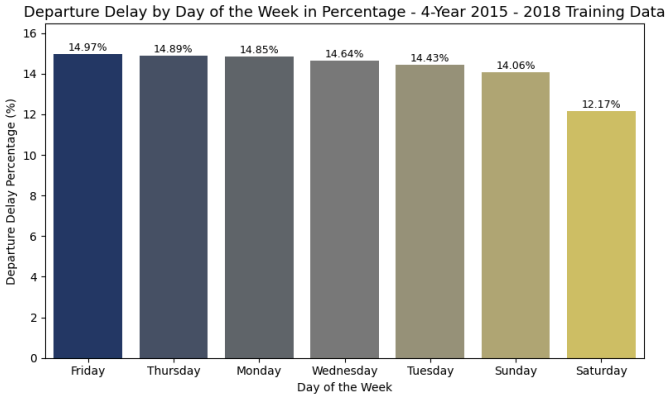
(Numeric) Daily and Rolling Averages of Flight Departure Delays with Holiday Effect

This report presents average daily delays and their 7-day and 28-day rolling averages to identify trends, smoothing daily fluctuations from the 4-year training dataset. It spotlights reduced delays on New Year's Day, July 4th, and Christmas, likely due to lower holiday air traffic. Seasonality has been factored into the predictive model with new features targeting the period around these holidays to predict potential delays.

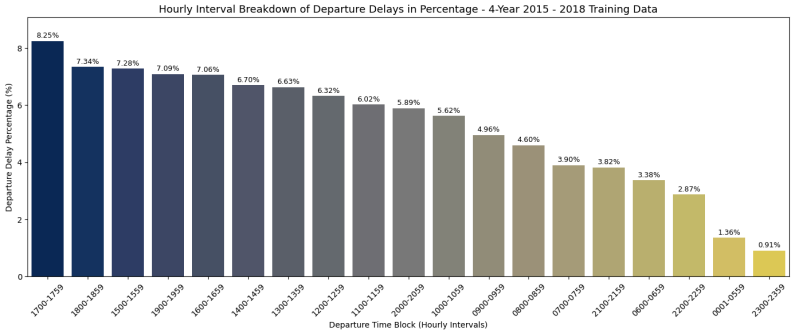

Daily and Rolling Averages of Flight Delays With Holiday Effect - 4-Year 2015 - 2018 Training Data
Average Delay (in Minutes)
Day of the Year
New Year's Day
July 4th
Christmas
Daily Avg Delay
7-Day Avg Delay
28-Day Avg Delay
Average = 9.52
New Year's Day
July 4th
Christmas

(Categorical) Flight Departure Delays by Day of the Week and Hour of the Day

Departure Delay by Day of Week



Departure Delay by Hour of Day



Departure Delay by Day of Week	Departure Delay by Hour of Day
The bar chart analyzing 4 years of data reveals that Friday experiences the highest flight departure delays at 14.97%, with Thursday and Monday closely following. A slight dip in delays is seen midweek, and the weekend,	The bar chart shows that flight delays peak in the afternoon and early evening (1 PM - 7 PM), and are less frequent in the early morning and late night. Four features were introduced to improve delay predictions: **Two for weekly patterns using trigonometric**

Addressing Leakage and ML Best Practices

Data Leakage Definition

In the realm of data science and machine learning at scale, data leakage is the unintentional addition of information into the training data that isn't actually relevant to the target variable, but artificially inflates the model's performance. This leakage can manifest most commonly through 3 ways:

1. Target Variable Leakage:
    - This occurs when training data includes information about the target variable that would not be available in real-world predictions or would have occurred after the target event.
    - Example: Training a model to predict customer churn based on whether they already contacted customer service about leaving.

2. Train-test Contamination:
    - This happens when information from the test dataset (used for evaluating the model) leaks into the training data.

- This gives an overly optimistic result and makes the model appear more accurate than it actually is.

3. Feature Engineering Leakage:
   - This involves creating features based on knowledge of the target variable.
   - Essentially, it gives the model the answer in advance.

A hypothetical example of leakage would be if we included the `WHEELS_OFF` time variable in the training data when predicting departure delays of a flight. This would be data leaking since `WHEELS_OFF` represents the actual time the aircraft's wheels leave the ground, which is a post-departure event. In a real world scenario, this information would not be available at the time of predicting departure delays. To avoid data leakage, it is crucial to ensure that the features used for training the model only include information that would realistically be available at the time of making predictions. In this case, `WHEELS_OFF` should be excluded from the training data when predicting departure delays.

Data Leakage Pipeline Analysis

Let's walk through our pipeline to check for any leakage:

1. EDA - We prevent data leakage by conducting EDA only on the 2015-2018 train set. We purposefully exclude the 2019 test set so that our findings such as feature distributions and statistical analysis are not influenced by the test data.
    - Feature selection - In our pearson correlation analysis, we found the top 5 features that were highest correlated features with the target variable and excluded them to prevent data leakage. This is because including variables influenced by the target variable, occurring after the target event in time, or derived directly from the target variable can introduce leakage.
2. Data Processing - We prevent data leakage at this step by fitting processing transformations on train, then applying to validation and test. For each data processing tasks, this means:

- Relabel disappearing and new carriers: We relabel dissapearing carriers based on whether they were acquired by another airline for the train set. For the test set, we relabel carriers appearing later in time as "Other".
- Mean Imputation - For a column that has missing data to be imputed, we calculate the mean of the train set for that column, then fill the missing values in train, validation, and test with the mean of train.
- Label/One-Hot Encoding - For a column requiring label/one-hot encoding for categorical features, we fit the encoder on the train set such that the labels/one-hot columns correspond to the unique categories in the train set, then use those train categories when encoding categrical features in validation and test.

3. Experiments
   - Model Training: Cross Validation - In our implementation of block time series cross validation, we create splits at each fold where the training set is before the validation set in time. This ensures the

model is not trained on future data, which mirrors the real-world scenario where future data is not available at the time of predictions.
    - Model Evaluation - We prevent data leakage by comparing F2 scores from the 2018 validation set for model selection, leaving test 2019 untouched.
4. Train Final Model - We train on the 2015-2018 train set, leaving 2019 test untouched.
5. Evaluate Final Model on Test - To align with our business objectives of delivering a nearly finalized product by Wednesday, December 13, 2023, we found it necessary to run our best available model on the test set, as our current final XGBoost model remained incomplete due to time constraints. This decision allowed us to present our optimal results at the time, specifically for the Gradient Boosted Tree Ensemble. While we recognize that this process involves data leaking, we want to emphasize that, since the presentation, we have diligently avoided being influenced by test set performance to preserve predictive modeling integrity. We resumed our experimentation using the CV train set, identified our best

model using the validation set, and evaluated our final model on the test set only once more.

Implementing ML Best Practices

In summary, we are preventing data leakage and are not violating any cardinal sins of ML by incorporating the following key measures into our pipeline:

1. Training/Validation/Test Data Split: In our implementation of block time series cross validation, we create splits at each fold where the training set is before the validation set in time. This ensures the model is not trained on future data, which mirrors the real-world scenario where future data is not available at the time of predictions.
2. Cross-Validation: We apply cross-validation with numFolds=5 to enhance model reliability and mitigate overfitting.
3. Hyperparameter Optimization: We employ grid search and Bayesian optimization to systematically identify the optimal model settings, thus

avoiding overfitting and selection bias that can result from manual parameter tuning.

4. Regularization: Our final XGBoost model uses L1 regularization (alpha, LASSO regularization) to deter overfitting by penalizing large coefficients.
5. Pipeline Feature Processing: The pipeline manages feature processing in one sweep, preventing data leakage by fitting transformations only to training data.
6. Hold-out Set Evaluation: We evaluate the model's performance on a separate test set, ensuring unbiased performance metrics.
7. No Peeking at the Test Data with Exception: There is no evidence in the code of using the test set for anything other than final evaluation. However, with the instructor's approval (Vinicio De Sola), we ran the selected final model, XGBoost, on the 2019 test data for the final evaluation after the mid-term presentation.
8. Diverse Metrics: We use multiple metrics like F2, precision, and recall to

Data Pre-Processing

Before feeding our selected features into the baseline pipeline, we performed 6 crucial data pre-processing steps.

1. Column Removal:
    - Dropped columns with 100% null values, duplicate columns, and features deemed not applicable to our scope.
    - There were 100% null values in approximately half of the raw columns, indicating an imperfect join when OTPW was created.

2. Feature Selection:
    - Used Pearson Correlation analysis with `DEP_TIME` as a proxy for the target variable.
    - Applied LASSO feature selection to drop features with a coefficient of 0.

3. Feature Type Casting:
    - Converted all string features into numeric, categorical, and date/time formats.

4. Additional Clean-Up:
    - Reformatted variables for improved data interpretation.
    - Changed labels of carriers that disappeared due to acquisition and relabeled new carriers as "Other."
        - During EDA we noticed that there are dissapearing carriers where they have rows in the beginning years, but not in later years, and new carriers that only appear in later years. The two dissapearing carriers we found were Virgin Airlines (VA), which became a part of Alaska Airlines (AS) and US Airways (US) which became part of American Airlines (AA). We replaced the labels of these carriers with 'AS-VA' to reflect that Alaska Airlines and Virgin Airlines represent one company but still capture both and 'AA-

US' for Alaska Airlines with US Airways. We relabeled the new carriers with the label "Other".

- Removed all cancelled flights.

5. Train/Test Split:
   - Completed a train/test split (explained further in the Train/Test Split Section).

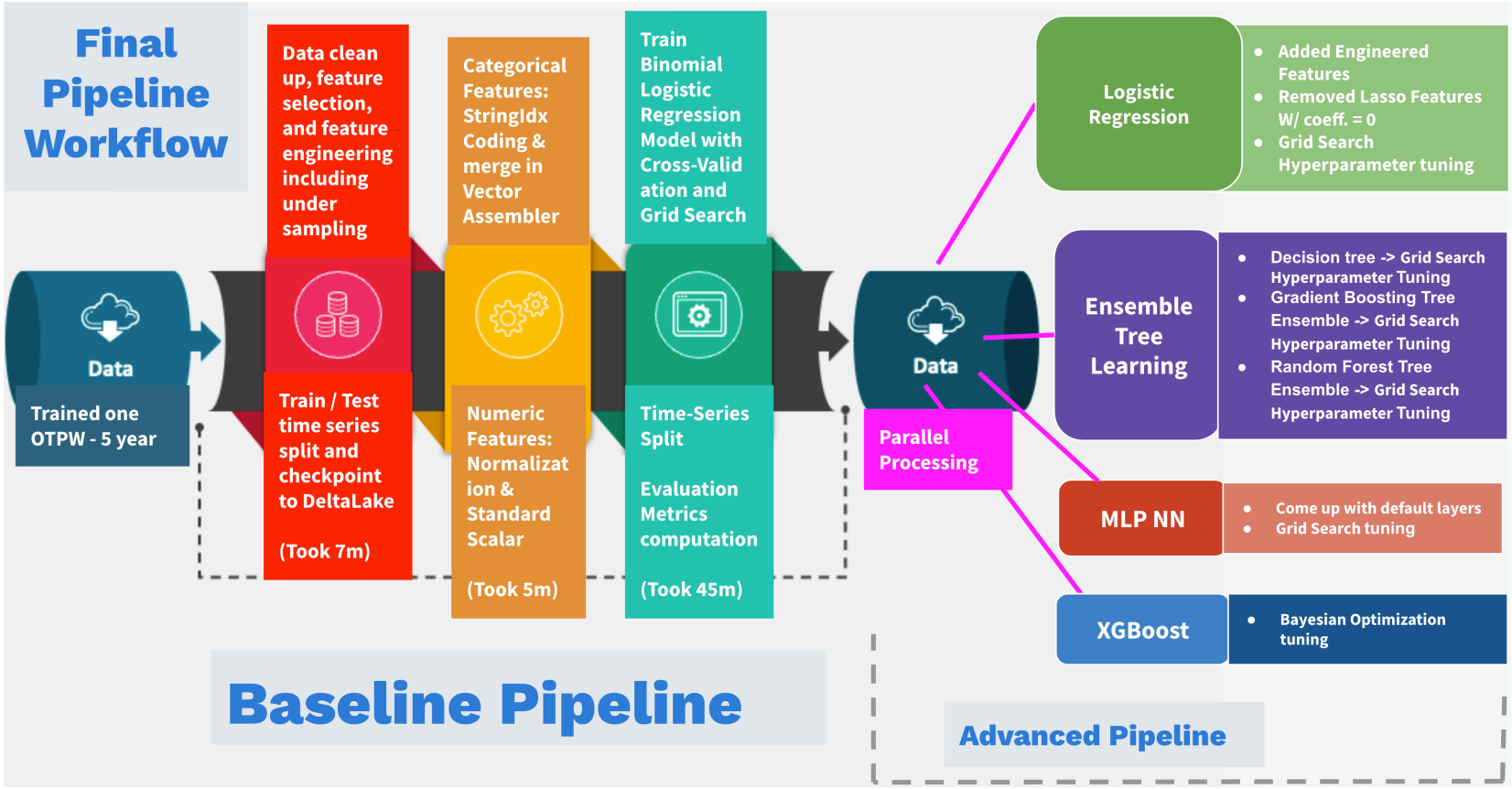
6. Data Imputation:
   - Used string indexing and one-hot encoding for categorical features.
   - Used standardization through feature vectors and standard scalar using mean for numeric features

Below is a diagram that represents the 6 data pre-processing steps:

Modeling Pipelines

ML Pipeline Overview

Below is a diagram that shows baseline pipeline as well as the advanced pipeline:



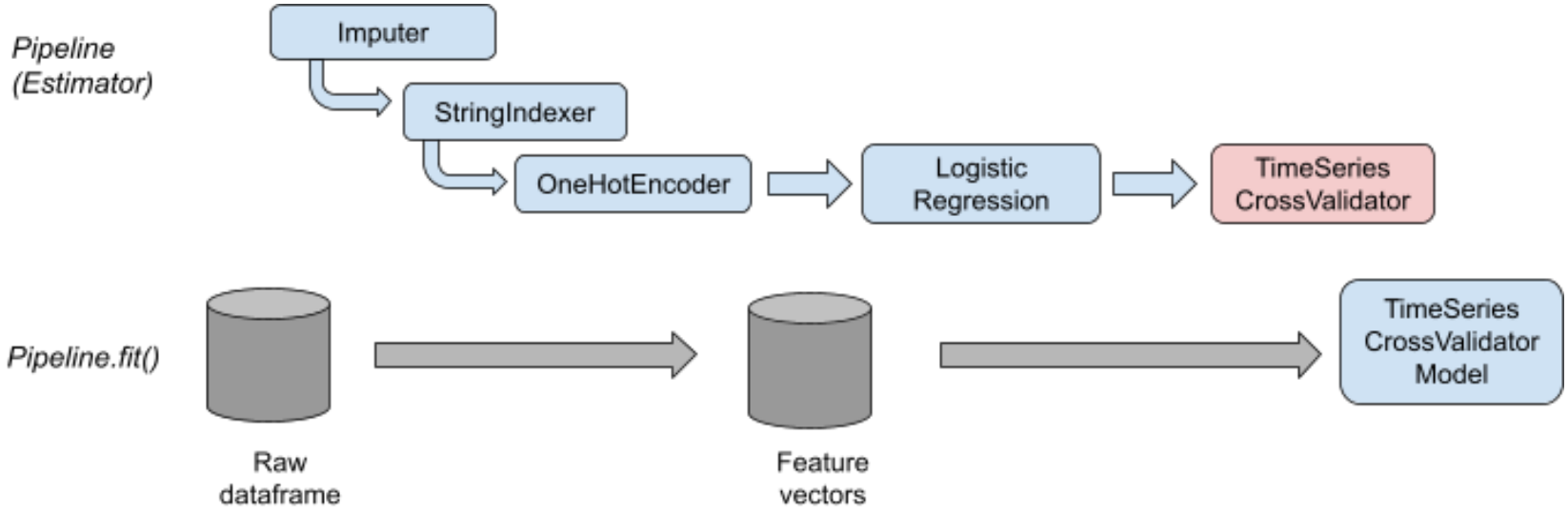
Pipeline Components

We built a machine learning pipline suitable for big data using the PySpark MLLib Pipeline library. Rather than running disjointed pieces of code, MLLib Pipeline makes it easier for us to combine multiple stages of data

processing and model building into a single pipeline or workflow. All of the pipelines for our experiments include the following pieces (except they use a different class for their respective ML algorithm/architecture):

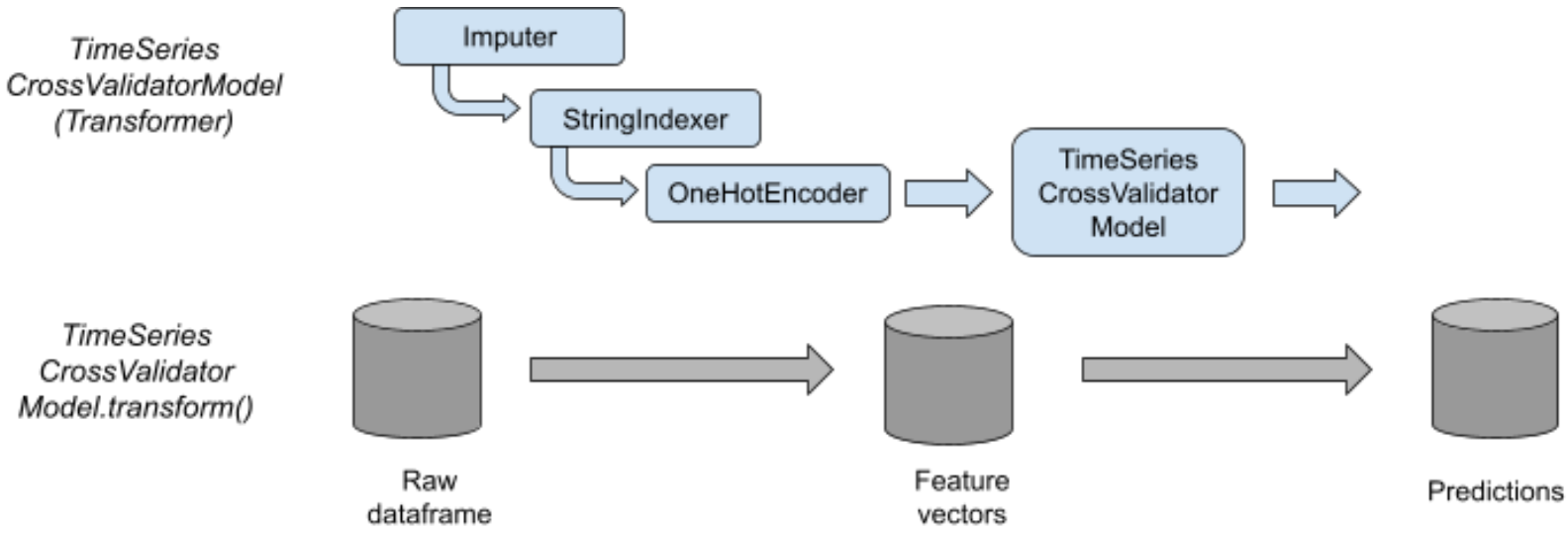
- Data processing - imputing missing values for numeric features using mean of each respective numeric column using `Imputer`, label encode ordinal categorical data using `StringIndexer`, one-hot encode categorical data using `OneHotEncoder`
- Logistic regression - our baseline algorithm
- `TimeSeriesCrossValidator` - a custom class similar to CrossValidator that creates folds using blocked time series cross validation technique described below

The diagram below shows the pipeline steps when fitting on input train data. As you can see, the pipeline accepts the raw dataframe and returns a TimeSeriesCrossValidatorModel.



Next, the diagram below shows the pipeline steps when transforming/evaluating on input test data. As you can see, the fitted TimeSeriesCrossValidatorModel accepts the raw test dataframe as well, takes care of data processing/encoding evaluates on the Logistic Regression algorithm using blocked time series cross validation with grid

search, and finally returns the predictions:



Description of Input Features

There were 31 final features that were input into our baseline model and therefore into our advanced pipeline as well. In future work, we would like to slim the number of features down, as we feel this could be an area which could help improve our results.

TIME:

- Total Count: 5
- YEAR: integer (nullable = true)
- MONTH: integer (nullable = true)
- DAY_OF_MONTH: integer (nullable = true)
- DAY_OF_WEEK: integer (nullable = true)
- CRS_DEP_TIME: integer (nullable = true)

NUMERIC (StandardScaled):

- Total Count: 13
- DISTANCE: float (nullable = true)
- ELEVATION: float (nullable = true)
- CRS_ELAPSED_TIME: float (nullable = true)
- HourlyWindSpeed: float (nullable = true)
- HourlyPrecipitation: float (nullable = true)
- HourlyRelativeHumidity: float (nullable = true)
- HourlyVisibility: float (nullable = true)
- CRS_DEP_HOUR_cos: double (nullable = true)

- CRS_DEP_HOUR_sin: double (nullable = true)
- DAY_HOUR_interaction: integer (nullable = true)
- DAY_OF_WEEK_cos: double (nullable = true)
- DAY_OF_WEEK_sin: double (nullable = true)
- TIME_INTERVAL_OF_DAY: integer (nullable = true)

CATEGORICAL (StringIdx/OneHot):

- Total Count: 9
- DIVERTED: integer (nullable = true)
- DEST: string (nullable = true)
- OP_UNIQUE_CARRIER: string (nullable = true)
- ORIGIN: string (nullable = true)
- TAIL_NUM: string (nullable = true)
- CARRIER_SIZE: string (nullable = true)
- DAY_TYPE: string (nullable = true)
- DEGREE_VISIBILITY: string (nullable = true)
- FL_DISTANCE_GROUP: string (nullable = true)

- 5_DAYS_DIST_FROM_Independence: string (nullable = true)
- 7_DAYS_DIST_FROM_Christmas string: (nullable = true)
- 7_DAYS_DIST_FROM_NewYear: string (nullable = true)
- dep_del15_2hr_before: string (nullable = true)

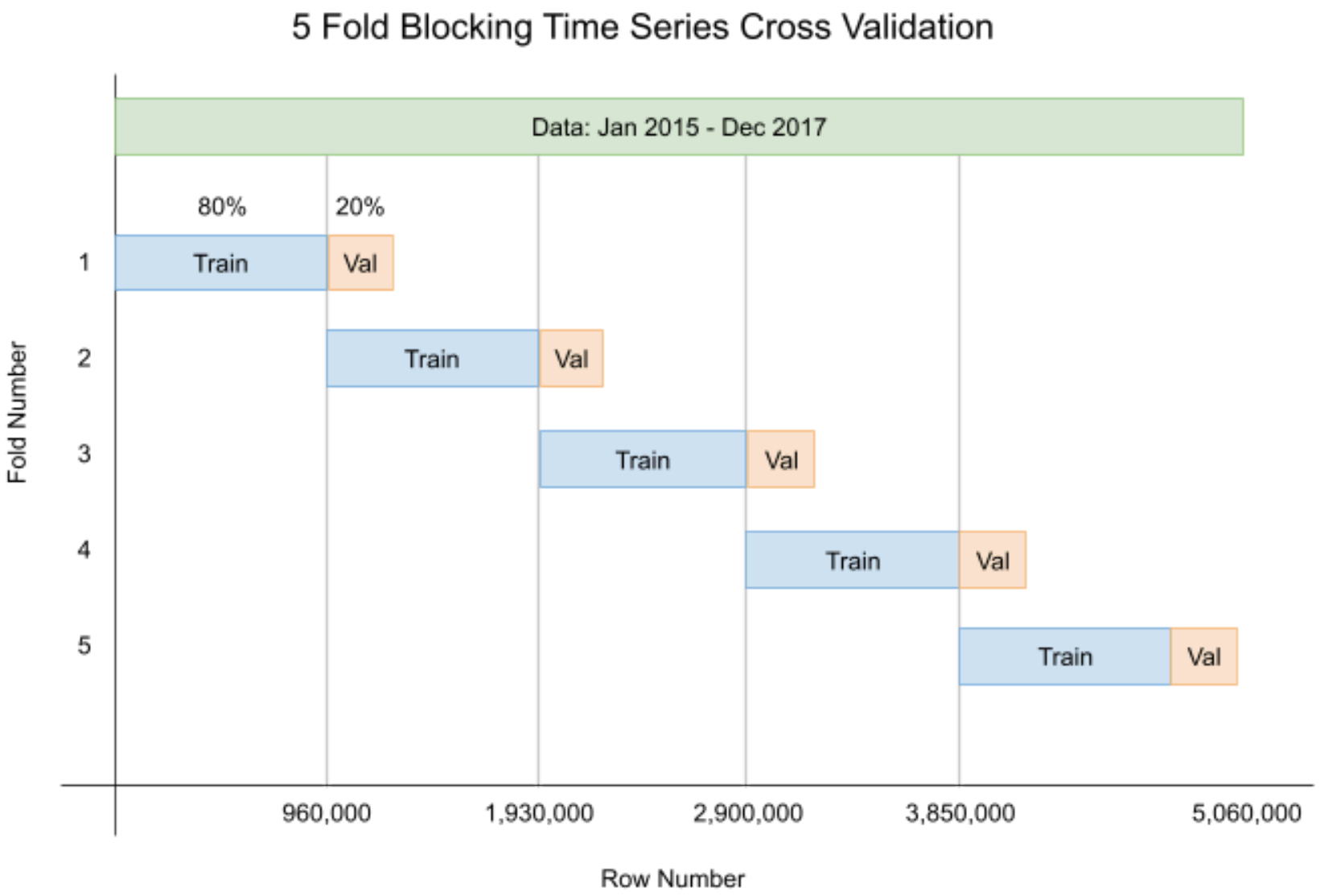
Here are our 10 best features based on LASSO feature selection:

Feature Name	**LASSO Coefficient**
dep_del15_2hr_before	0.5358587443
HourlyRelativeHumidity_imputed	0.02492615975
CRS_DEP_TIME	0.008931453813
7_DAYS_DIST_FROM_NewYear_idx	0.004401876106
HourlyWindSpeed_imputed	0.002943109879
ORIGIN_idx	0.0008584629491

Feature Name	LASSO Coefficient
HourlyPrecipitation_imputed	0.0007121531309
TIME_INTERVAL_OF_DAY	-0.0007543557787
HourlyVisibility_imputed	-0.004603613372
CRS_DEP_HOUR_sin	-0.0536135181

Time Series Cross Validation

We integrated cross-validation into our modeling pipeline for robust performance assessment across folds, reducing evaluation variability compared to a single train-validation split, and optimizing hyperparameters. In traditional cross-validation, data is randomly split, while time series cross-validation maintains the temporal order, ensuring realistic training on past data and evaluating predictions on future data. For our project, we implemented the blocking time series cross-validation technique.


5 Fold Blocking Time Series Cross Validation
Data: Jan 2015 - Dec 2017
80%
20%
Train
Val
Fold Number
1
2
3
4
5
960,000
1,930,000
2,900,000
3,850,000
5,060,000
Row Number

We performed blocking time series cross validation on the OTPW-3yr training set for Phase 2. The diagram above shows this technique, where we divided time into 5 contiguous, time-ordered pieces for each fold where the validation split is after the train split. Within each fold, the first 80% and last 20% of time-ordered rows were designated for training and validation, respectively. The model was trained on the train set and evaluated on the validation set at each fold. The average F2 score across the 5 folds was used as the cross-validated F2 score metric. To maximize training data utilization, each fold began at the end of the previous fold's training set.

Experiments

We conducted 9 primary experiments, which are elaborated on below:

1. **Baseline with Default Features:**
    - This experiment represents the baseline model using default features without any additional modifications.

- We use the LogisticRegression class for our baseline model, which uses Elastic Net regularization by default. The loss function is as follows:

$$\alpha\left(\lambda\|\mathbf{w}\|_1\right)+(1-\alpha)\left(\frac{\lambda}{2}\|\mathbf{w}\|_2^2\right), \alpha \in[0,1], \lambda \geq 0$$

  where alpha corresponds to `elasticNetParam` and lambda corresponds to `regParam` when fitting LogisticRegression.

2. **Logistic Regression with Added Engineered Features:**
   - Logistic regression model with additional features engineered to enhance predictive performance.

3. **Logistic Regression - Removed LASSO Features:**
   - Logistic regression model after removing features identified as unimportant by LASSO feature selection.

4. **Logistic Regression with Added Engineered Features - After Hyperparameter Tuning:**

- Fine-tuned logistic regression model with added engineered features to optimize performance.

5. **Decision Tree:**
   - Basic decision tree model using features without additional modifications.

6. **Gradient Boosting Tree Ensemble:**
   - Ensemble model using gradient boosting trees, a machine learning technique for combining multiple decision trees.

7. **Random Forest Tree Ensemble:**
   - Ensemble model using a random forest, which consists of multiple decision trees to improve predictive accuracy.

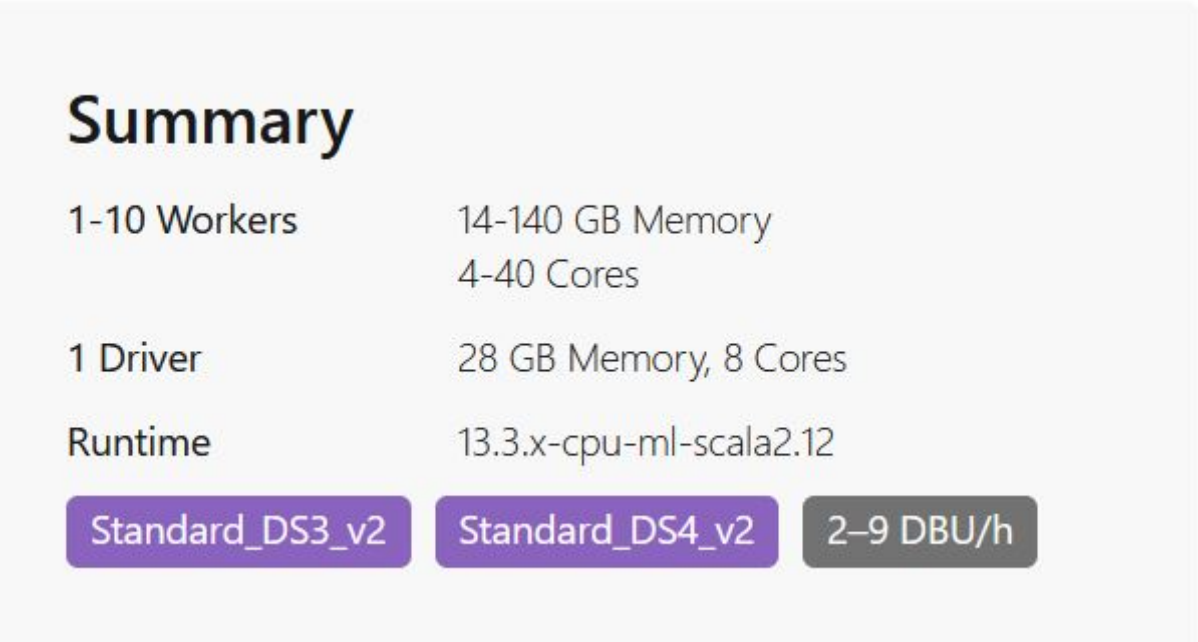
8. **Multilayer Perceptron Neural Network (MLP NN):**
   - Neural network model with multiple layers, a type of deep learning architecture.

9. **XGBoost with Bayesian Optimization Tuning:**
    - XGBoost model with hyperparameter tuning performed using Bayesian optimization for improved performance.
    - The objective is set to `binary:logistic`, which means it uses the logistic loss function for binary classification.

      $$L(y, p) = -y \log(p) - (1 - y) \log(1 - p)$$, where **y** is the true label (0 or 1 for binary classification) and **p** is the predicted probability of the instance being in the positive class (1).

Computational Configuration

Below is an image of our environment setup where we were able to autoscale between 1 and 6 worker nodes within our cluster:



Experiments Table

Below is a table of our cross validated train and validation scores from our experiments along with the top 3 models with the best results. XGBoost had the highest validation F2 score, so we deemed it best fit to select this for our final model.

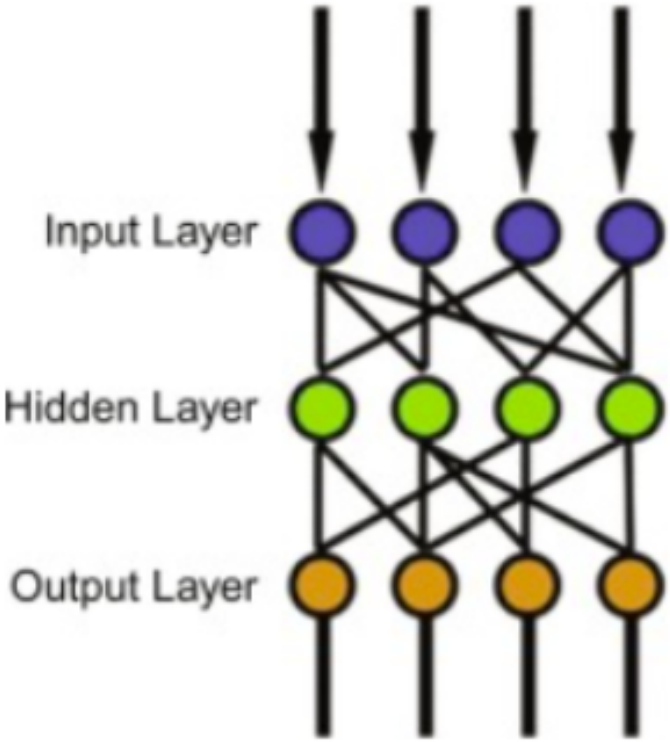
Top Three	Model	Best Hyperparameter	Train CV F2	Validation F2

Neural Network MLP

For more background please see here: https://spark.apache.org/docs/latest/ml-classification-regression.html#multilayer-perceptron-classifier

(https://spark.apache.org/docs/latest/ml-classification-regression.html#multilayer-perceptron-classifier)

We built a Multiple-layer Perceptron classification(MLPC) model using Spark's MLlib. The MLPC is based on the feedforward artificial neural network architecture which consists of multiple layers of node. Each Layer is fully connected to next layer in the network as this diagram:



The input layer size is equivalent to our feature size=31. Their can be various numbers of hidden layer and each uses a Sigmoid function.

$$f(z_i) = \frac{1}{1 + e^{-z_i}}$$ The output layer uses a Softmax function

$$f(z_i) = \frac{e^{z_i}}{\sum_{k=1}^{N} e^{z_k}}$$ in which we output 2 classes {0, 1}

We have tried a few different layer configurations and input them as part of the Grid Search parameters:

- Layer Configuration 1:

Input	Hidden	Output
31	16	2

- Layer Configuration 2:

Input	Hidden	Hidden	Hidden	Hidden	Output
31	64	32	16	8	2

- Layer Configuration 3:

Input	Hidden	Hidden	Hidden	Hidden	Hidden	Output
31	128	64	32	16	8	2

With 3 different configuratoins as part of Grid Serach, Layer configuration 2 gave us the highest validation F2 score = 0.6075 and this layer has the right balance of complexity to extract relationship between each input features. On top of layer configuratoin, our Grid Search also include blockSize [128,

Additional Neural Network (Extra Credit)

Our team found that compiling and training a deep learning neural network model in Spark is a very difficult task simply because Spark does not natively support neural network model other than MLP. We had previously tried:

- Sparkdl https://spark-packages.org/package/JeremyNixon/sparkdl (https://spark-packages.org/package/JeremyNixon/sparkdl)
- Sparkflow https://github.com/lifeomic/sparkflow (https://github.com/lifeomic/sparkflow)

Both of these frameworks were stale and we could not find the correct software package combination that would not cause a pip dependency resolver error.

We attempted Sparktorch, which is a framework to allow building Pytorch neural network models utilizing the TorchDistributor API to allow distributed training with Spark clusters. We finally were able to compile and train a NN model through this methodology.

- Sparktorch https://github.com/dmmiller612/sparktorch (https://github.com/dmmiller612/sparktorch)

We built a Convolution Neural Network(CNN) Model using Sparktorch. Our CNN model contains 3 convolution 1d layers followed by a Drop Out followed by a linear layer. The output layer is a sigmoid function.

Layer#	Layer Type	Input Size	Output Size	Kernel Size	Stride Size	Padding	Activation Function
1	Conv1d	31	24	8	1	none	ReLu
2	Conv1d	24	17	8	1	none	ReLu
3	Conv1d	17	10	8	1	none	ReLu -> DropOut(0.2
4	Linear	10	1	NA	NA	NA	Sigmoid

For the first three convolution layers, we applied ReLu activation function to prevent the exponential growth in the computation required to operate the neural network. It also prevents the vanish gradient effect. Between Layer3 and before going to Layer4, we applied DropOut function to nullify 25% of neurons to prevent overfitting on the training data. The output layer is a sigmoid function that predicts 0 or 1.

To train the model, we use the Binary Cross Entropy Loss as loss function

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^\top, \quad l_n = -w_n \left[y_n \cdot \log x_n + (1 - y_n) \cdot \log(1 - x_n) \right],$$

We used the Adam optimizer with learning rate of 0.001 for training. The goodness about CNN is that it can identify special meaning among features like when it recognizes animals and objects within a photo. Here is the training results. Since we are only experimenting with deep learning tools at scale, CNN is not included in the model selection within our main experiments. Since we're not comparing CNN to other models, we are not running valdation set and the test set.

METRICS ON TRAIN

- train_f125_score: 0.686129893211284
- train_f15_score: 0.722030224927298
- train_f2_score: 0.774924769102926 5
- train_precision: 0.501828353503501 1
- train recall: 0.896956312441904 9

Prophet (Extra Credit)

Data Preparation

Prophet exploration was conducted on the `delta_otpw_5yr` dataset, consistent with the rest of our project. The dataset underwent cleaning and data pre-processing through various steps, identical to the procedures performed before our initial baseline model.

Column Selection

Selected Columns:

- DEP_DELAY
- CRS_DEP_TIME
- ORIGIN
- TAIL_NUM

Data Filtering

Created a subset by filtering to rows where the origin is "CSG" and the tail number is "N801AY."

Military Time Conversion

Calculated the length of the 'CRS_DEP_TIME' column to dynamically extract hour and minute components. Derived 'time_in_minutes' by converting military time to minutes.

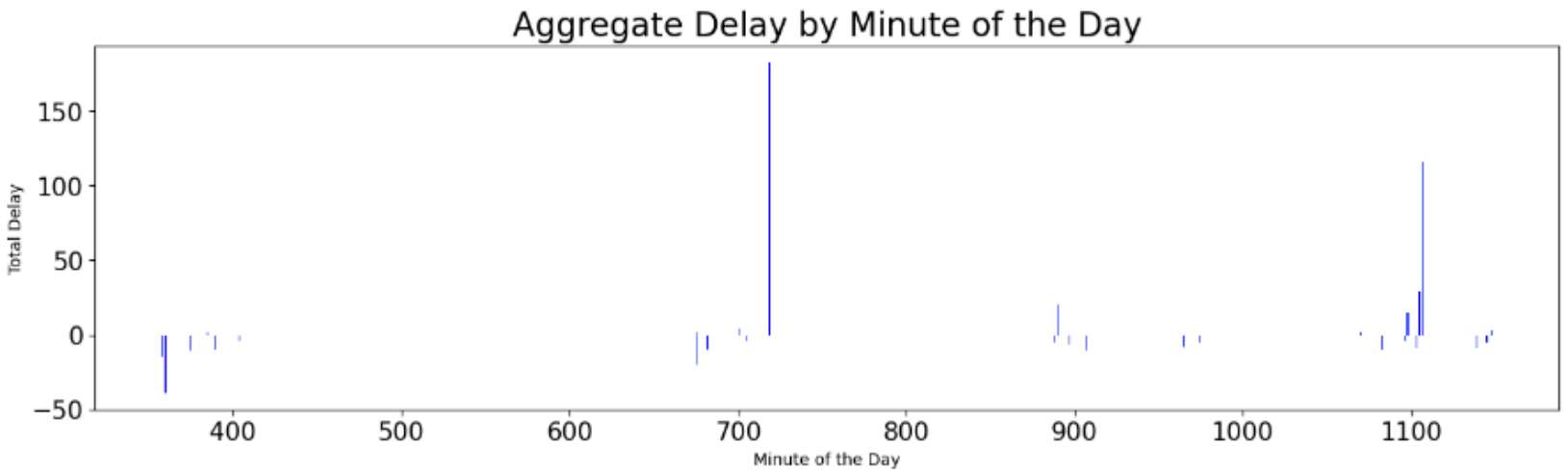
The first 5 rows of the dataframe after this conversion look like the following:

DEP_DELAY	CRS_DEP_TIME	ORIGIN	TAIL_NUM	time_in_minutes
-10	1122	CSG	N801AY	682
-4	600	CSG	N801AY	360
4	1141	CSG	N801AY	701
5	1507	CSG	N801AY	907
-9	1823	CSG	N801AY	1103

Aggregate Delay Visualization

Plotted the aggregate delay for each minute of the day:

- Utilized a bar chart to display the distribution of delays.
- Provided insights into the variation of delays throughout the day.



Prophet Application

We prepared the PySpark DataFrame for Prophet time series modeling. We selected relevant columns ('time_in_minutes' and 'DEP_DELAY') for modeling. We aggregated data where necessary, using the average delay for each minute. We transformed the PySpark DataFrame into a Pandas DataFrame for compatibility with Prophet. We applied the Prophet model to forecast future delays based on the time of the day. Finally, we plotted the forecasted results using the Prophet library.

In-Sample Prediction

The Prophet model is used for in-sample prediction. The model is trained on the dataset, and predictions are made using the `make_future_dataframe` method. The predictions are visualized using the `plot` and `plot_components` methods.

Here were the results we produced for In-Sample prediction:

Metric	Value
MAPE	inf %
SMAPE	143.68%

Out-Sample Prediction

We conducted out-of-sample prediction using the Prophet model. We converted the 'ds' column to datetime format for consistency. We merged actual and predicted values for evaluation. We calculated the Mean Absolute Percentage Error (MAPE) and Symmetric Mean Absolute Percentage Error (SMAPE) for evaluation. We plotted the out-of-sample forecast results, showing discrepancies in accuracy measures, with an infinite MAPE and 143.68% SMAPE.

Here were the results we produced for Out-Of-Sample prediction:

Metric	Value
MAPE	inf %
SMAPE	143.68%

Issues Identified

- Out-of-Sample and In-Sample Prediction: Mean Absolute Percentage Error (MAPE) and Symmetric Mean Absolute Percentage Error (SMAPE):

The evaluation metrics for in-sample prediction display "inf" for MAPE and a high percentage for SMAPE. This suggests an issue with the evaluation process or the data input in the model.

Next Steps and Thoughts for Debugging and Future Application

- Evaluation Metric Issue: Investigate and address the issue with the MAPE and SMAPE calculations. Check for potential division by zero or other errors in the evaluation process.
- Data Patterns: If the data exhibits complex seasonality or trends that are not well-captured by the model, it can result in poor forecasting accuracy.
- Model Review: Verify that the features used for time series forecasting are appropriate and relevant to the forecasting task. Adjust the feature selection if necessary.

Final Model Results: XGBoost

Baseline vs. Final Model: XGBoost

After training our models on data spanning from 2015 to 2017 and performing time series cross-validation with 2018 data, XGBoost emerged as the top performer. It boasts a validation F2 score of 0.6327, signifying a significant improvement — over a 15% increase — compared to the baseline Logistic Regression model. Our baseline achieved an F2 score of 0.5476 on the validation set prior to the introduction of engineered features. The advanced capabilities of XGBoost in handling complex data patterns, its robust regularization techniques, and the use of Bayesian optimization for hyperparameter tuning are crucial to its superior performance. The improvement from baseline to our final model is summarized in the table below:

Model	Validation F2 Score	Test F2 Score
Baseline: Logistic Regression	0.5476	n/a

Model	Validation F2 Score	Test F2 Score
Final Model: XGBoost	0.6327	0.6889

XGBoost with Bayesian Optimization Hyperparameter Tuning

We employed Bayesian Optimization for Hyperparameter Tuning based on the following hyperparameter choices:

Hyperparameter	Choices
eta	[0.01, 0.05, 0.1]
alpha	[0.1, 0.3, 0.7, 1.0]
maxDepth	[6, 7, 8, 9,10]

Hyperparameter	Choices
numRound	[100, 150, 200, 250, 300]

The best hyperparameters from this search are:

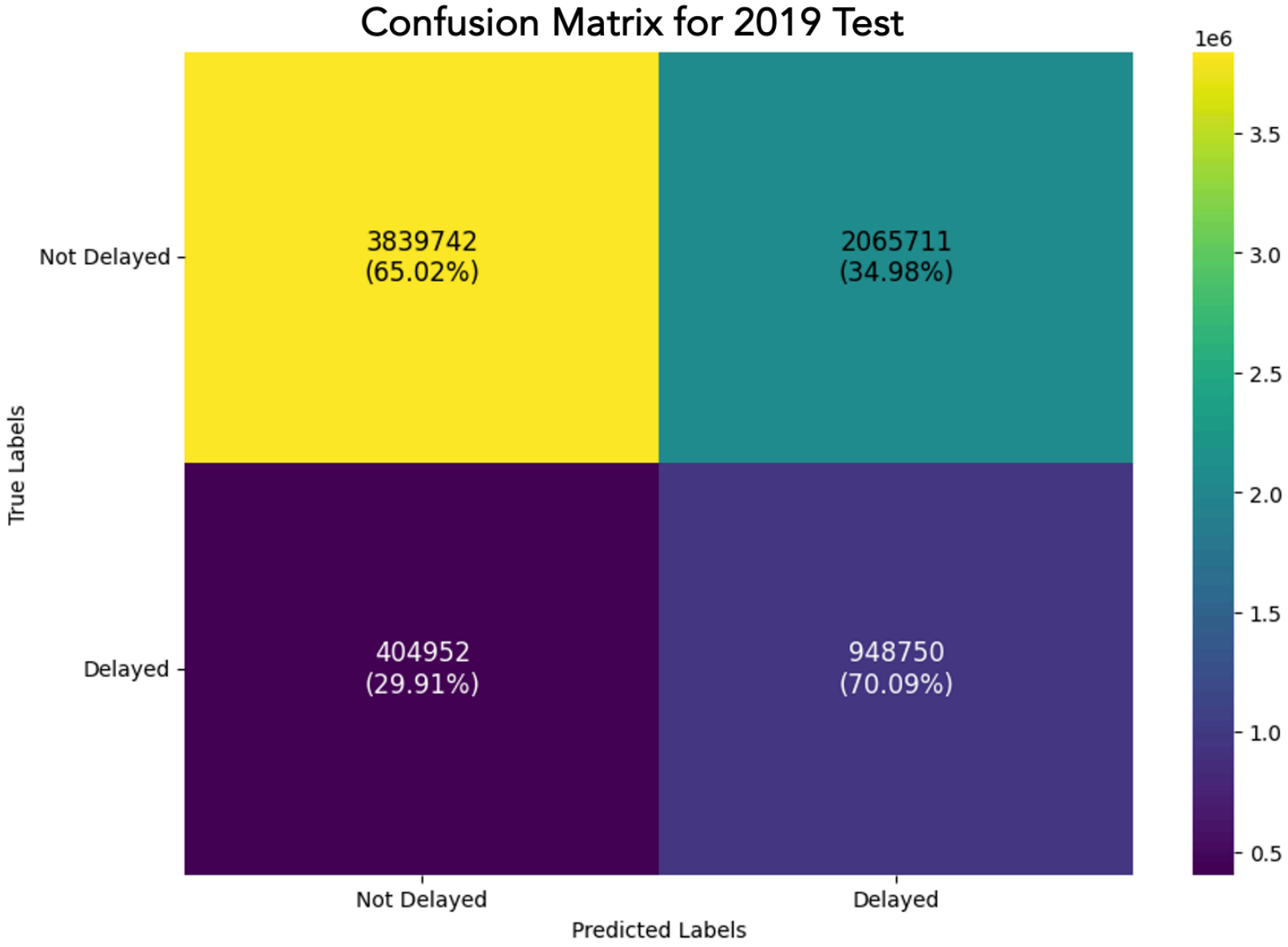
```
{'alpha': 0.1, 'eta': 0.05, 'maxDepth': 10, 'numRound': 300}
```

XG Boost Results and Discussion

XGBoost Performance Results on the 2019 Test Dataset:

Data Set	F2 Score	Precision	Recall
Train CV 2015-2018	0.674821640349716 4	0.731075731916343 7	0.662085269 89
Test 2019	0.688952895637508 4	0.904598069966881	0.650202787152

XGBoost Confusion Matrix:


Confusion Matrix for 2019 Test
1e6
3.5
3.0
2.5
2.0
1.5
1.0
0.5
Not Delayed
Delayed
True Labels
3839742
(65.02%)
2065711
(34.98%)
404952
(29.91%)
948750
(70.09%)
Not Delayed
Delayed
Predicted Labels

Key Points:

- XGBoost's model complexity and iterative boosting allow it to capture nuanced data patterns, unlike simpler models such as Decision Trees.
- Regularization within XGBoost, including L1 (LASSO) and early stopping, combats overfitting, as evidenced by closely aligned train (0.6748) and test (0.6890) F2 scores.
- Bayesian optimization efficiently refines hyperparameters based on past performance, outperforming grid search.

Analytical Insights on the Final XGBoost Model's Performance on the 2019 Test Data:

1. F-Scores:
    - The testing set's scores are notably higher, with the F2 score reaching 0.6889. This points to effective generalization and might imply that the model is conservative in training or that the test set slightly favors the model's predictions.

2. Precision and Recall:
    - The testing set's precision is extraordinarily high at 0.9046, affirming that the model is dependable in its predictions of flight delays.
        - Nonetheless, the recall of 0.6502 indicates the model does not identify about 35% of delayed flights, as evidenced by a considerable number of false negatives in the confusion matrix.

Gap Analysis of the Final XGBoost Model's Performance on the 2019 Test Data:

1. Recall vs. Precision:
    - The high precision in contrast to the moderate recall highlights a need to enhance the model's ability to detect all delayed flights, which is a common balancing act in classification tasks.

2. False Positives:
    - The false positive rate of nearly 35% on the test set suggests that among all flights that were not delayed, 34.98% were incorrectly

predicted as delayed by the model, leading to potential operational inefficiencies or unnecessary costs due to false alarms.

3. False Negatives:
    - A significant volume of false negatives, accounting for 29.91%, indicating that among all flights that were actually delayed, 29.91% were incorrectly predicted as not delayed by the model. This oversight could lead to missed opportunities for taking proactive measures.

4. Generalization:
    - Although the F2 scores for the training and test sets are very close, the superior performance on the test set compared to the training set warrants further investigation. It's important to examine the distributions of both datasets to ensure representativeness and to confirm that the model isn't overly conservative during training.

5. Optimization Focus:

- While Bayesian optimization has likely fine-tuned the model for precision, a shift towards improving recall is recommended, particularly if operational imperatives demand a higher detection rate

Conclusion

Our project, predicting flight departure delays, with a goal of acheiving high predictive accuracy, led us through a rigorous journey of data cleansing, feature selection and engineering, model experimentation, and analyzing results at scale. The XGBoost model emerged as a frontrunner, creating a notable test F2 score of 0.6889. This achievement is impressive, emphasizing the model's advancing capabilities in predicting flight delays. However, our focus now shifts to addressing the identified gap—improving recall without compromising precision. The interplay of false positives, false negatives, and generalization prompts us to fine-tune our approach. Looking at next steps, the path forward involves refinement and improvement of the XGBoost model, exploration of more advanced

ensemble models, continuous feature evaluation, and strategic analytical and statistical optimization. Our commitment remains unwavering; achieving a harmonious balance between precision and recall to enhance the operational efficiency of predicting flight departure delays as well as doing the right thing for our customers, communities, and planet.

Phase 3 Project Code

Here's a link to a notebook with our code for EDA:

- https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/546615846962872/command/5466
  (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/546615846962872/command/5466
- Correlation Study: https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/4034308520565479/command/403

(https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/403430852056547 9/command/403

Here's a link to a notebook with our code for feature engineering/new features:

- https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1346418319513685/command/1346 (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1346418319513685/command/1346

Here's a link to notebooks with our code for our Modeling Pipeline:

- Baseline run : https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1962809137470638 (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1962809137470638)
- Lasso run: https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1346418319513523 (https://adb-

424844930383559.19.azuredatabricks.net/?o=424844930383559#notebook/1346418319513523)

- Baseline + new features run: https://adb-424844930383559.19.azuredatabricks.net/?o=424844930383559#notebook/1962809137470638 (https://adb-424844930383559.19.azuredatabricks.net/?o=424844930383559#notebook/1962809137470638)
- MLP: https://adb-424844930383559.19.azuredatabricks.net/?o=424844930383559#notebook/1346418319529529 (https://adb-424844930383559.19.azuredatabricks.net/?o=424844930383559#notebook/1346418319529529)
- Decision Tree: https://adb-424844930383559.19.azuredatabricks.net/?o=424844930383559#notebook/1346418319522726 (https://adb-424844930383559.19.azuredatabricks.net/?o=424844930383559#notebook/1346418319522726)

- Gradient Boosting Tree: https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1346418319520653 (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1346418319520653)
- Random Forest Tree: https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1346418319520717 (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1346418319520717)
- Sparktorch CNN: https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1346418319520870 (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1346418319520870)
- Prophet Exploration: https://adb-4248444930383559.19.azuredatabricks.net/?

o=4248444930383559#notebook/1346418319523389/command/1346
(https://adb-4248444930383559.19.azuredatabricks.net/?
o=4248444930383559#notebook/1346418319523389/command/1346

- XGBoost: https://adb-4248444930383559.19.azuredatabricks.net/?
o=4248444930383559#notebook/1346418319522969/command/1346
(https://adb-4248444930383559.19.azuredatabricks.net/?
o=4248444930383559#notebook/1346418319522969/command/1346

Here's a link to notebook with our code for the best model, evaluating 2019 test dataset:

- XGBoost: https://adb-4248444930383559.19.azuredatabricks.net/?

Phase 3 Credit Assignment Plan

Each JIRA story was assigned an owner who determined how long the task would take to achieve the story. Individual tasks were derived from

estimated workload and assigned among team members. Here is a snapshot of all Stories and Tasks for Phase 3:

Type	Key	Summary	Status	Assignee	Due date
>	FP-1	Project Phase 1 - Project Plan, describe datasets, joins, tasks, ...	DONE	RT Raymond Tang	Nov 5, 2023
>	FP-2	Project Phase 2-1 - EDA and baseline pipeline on all available ...	DONE	RT Raymond Tang	Dec 3, 2023
>	FP-27	Project Phase 2-2 - Continue EDA, Feature engineering, and ...	DONE	SC Stephanie Cabanela	Dec 3, 2023
>	FP-28	Project Phase 2-3 - Model Building for Scalability, Efficiency, D...	DONE	SC Stephanie Cabanela	Dec 3, 2023
v	FP-3	Project Phase 3 - Select the optimal algorithm, fine-tune and ...	DONE	HJ Heesuk Jang	Dec 13, 2023
v	FP-119	Data Processing	DONE	RT Raymond Tang	Dec 7, 2023
	FP-120	Save cleaned 5year train DF and test DF to cloud storage	DONE	RT Raymond Tang	Dec 5, 2023
	FP-121	Checkpoint raw delta lake dataframe with Phase 2 new featur...	DONE	HJ Heesuk Jang	Dec 5, 2023
	FP-125	Feature engineering - PREV_FLIGHT_DELAY Lag Feature	DONE	SC Stephanie Cabanela	Dec 7, 2023
	FP-126	Feature engineering - N_DAYS_FROM_{HOLIDAY}	DONE	HJ Heesuk Jang	Dec 7, 2023
	FP-129	Exploration of EC: Provide a clean dataset for Flights and/or ...	DONE	K Karsyn	Dec 7, 2023
	FP-130	Create function that does LASSO feature selection for any dat...	DONE	K Karsyn	Dec 7, 2023
v	FP-122	EDA Processing and Refinements	DONE	HJ Heesuk Jang	Dec 10, 2023
	FP-123	Run EDA on 5-year training dataset and push the images of t...	DONE	HJ Heesuk Jang	Dec 9, 2023
	FP-141	Run EDA on features within each family and provide descripti...	DONE	HJ Heesuk Jang	Dec 9, 2023
	FP-142	Create a summary statistics on the features selected	DONE	RT Raymond Tang	Dec 9, 2023
	FP-150	Research change the CV code slightly to accommodate expon...	DONE	SC Stephanie Cabanela	Dec 10, 2023
	FP-151	Research how to calculate metrics by Label == 1, NOT weight...	DONE	RT Raymond Tang	Dec 10, 2023
v	FP-124	Modeling AND Fine-Tuning with Random or Grid Search	DONE	SC Stephanie Cabanela	Dec 15, 2023
	FP-127	(REQUIRED) Train a MLP Neural Network on the 5 year traini...	DONE	SC Stephanie Cabanela	Dec 14, 2023

	FP-128	(EXTRA CREDIT 5 pts) Train other NN or Deep Learning techni...	DONE	RT Raymond Tang	Dec 15, 2023
	FP-131	(OPTIONAL) Gradient Boosted Decision Trees or Prophet - Tr...	DONE	RT Raymond Tang	Dec 14, 2023
	FP-143	Create a visualization of the modeling pipeline(s) and sub-pip...	DONE	K Karsyn	Dec 13, 2023
	FP-144	Create an experiment table with the following details per expe...	DONE	K Karsyn	Dec 15, 2023
	FP-145	Do experiments on ALL the data for new features and experi...	DONE	RT Raymond Tang	Dec 15, 2023
	FP-148	(OPTIONAL) XGBOOST or LightBoost - Train other models in ...	DONE	HJ Heesuk Jang	Dec 15, 2023
	FP-149	Train the baseline (Logistic Regression) on the 5-year training ...	DONE	RT Raymond Tang	Dec 7, 2023
	FP-132	Presentation slides for the demo on Dec. 13th, Wednesday	DONE	K Karsyn	Dec 13, 2023
	FP-152	Heesuk: EDA slides	DONE	HJ Heesuk Jang	Dec 12, 2023
	FP-154	Raymond: Pipeline and Results slides	DONE	RT Raymond Tang	Dec 12, 2023
	FP-153	Stephanie: Business use case, Dataset, CV slides	DONE	SC Stephanie Cabanela	Dec 12, 2023
	FP-155	Karsyn: Heatmap correlation slides	DONE	K Karsyn	Dec 12, 2023
	FP-133	Phase 3 report	TO DO	K Karsyn	Dec 16, 2023
	FP-134	(10 pts) Team and project meta information	DONE	SC Stephanie Cabanela	Dec 16, 2023
	FP-135	(10 pts) Project Abstract (150 words approximately)	DONE	SC Stephanie Cabanela	Dec 16, 2023
	FP-136	(10 pts) Data and feature engineering	DONE	HJ Heesuk Jang	Dec 16, 2023
	FP-137	(10 pts) Define what is leakage and provide a hypothetical ex...	DONE	K Karsyn	Dec 16, 2023
	FP-138	(10 pts) Add a visualization of the modeling pipeline(s) and su...	DONE	K Karsyn	Dec 16, 2023
	FP-139	(10 pts) Results and discussion of results - Gap Analysis	DONE	HJ Heesuk Jang	Dec 16, 2023
	FP-140	(10 pts) Conclusion	DONE	K Karsyn	Dec 16, 2023
	FP-147	Clean up code - Make sure you have an end-to-end pipeline ...	DONE	RT Raymond Tang	Dec 16, 2023

Once a task was assigned to a team member, that person estimated and logged the number of man-hours needed to complete that task as shown below:

Key	Summary	Assignee	Time Spent	Σ Time Spent	Due date
FP-155	Karsyn: Heatmap correlation slides	Karsyn	3h	3h	12/Dec/23
FP-154	Raymond: Pipeline and Results slides	Raymond Tang	3h	3h	12/Dec/23
FP-153	Stephanie: Business use case, Dataset, CV slides	Stephanie Cabanela	3h	3h	12/Dec/23
FP-152	Heesuk: EDA slides	Heesuk Jang	3h	3h	12/Dec/23
FP-151	Research how to calculate metrics by Label == 1, NOT weightedPrecision	Raymond Tang	2h	2h	10/Dec/23
FP-150	Research change the CV code	Stephanie Cabanela	3h	3h	10/Dec/23

Phase 3 Project Execution Gantt Chart

Below is our Gantt Chart showing Phase 3 JIRA tickets. JIRA has the capability to show depandencies of tickets since some stories or tasks can be achieved only if some other tasks are achieved. If a critical task is delayed, it might effect the overall schedule. JIRA allows the project manager to see task dependencies visually.

Legend:

- RT = Raymond Tang
- HJ = Heesuk Jang
- K = Karsyn
- SC = Stephanie Cabanela


Items
NOV
DEC
FP-94 Feature Engineering
FP-99 Report Writeup
FP-28 Project Phase 2-3 - Model Building for ...
FP-3 Project Phase 3 - Select the optimal algo...
FP-119 Data Processing
FP-120 Save cleaned 5year train DF and t...
FP-121 Checkpoint raw delta lake datafra...
FP-125 Feature engineering - PREV_FLIG...
FP-126 Feature engineering - N_DAYS_FR...
FP-129 Exploration of EC: Provide a clean ...
FP-130 Create function that does LASSO ...
FP-122 EDA Processing and Refinements
FP-123 Run EDA on 5-year training datas...
FP-141 Run EDA on features within each ...
FP-142 Create a summary statistics on th...
FP-150 Research change the CV code slig...

Project Management Tool

Our team adopted JIRA Software as our agile project management tool for efficiently planning, tracking, and executing our machine learning project at scale. This framework enables us to establish phases and Gantt charts, facilitating the creation of S.M.A.R.T goals and weekly project manager assignments. In alignment with the Agile Software Development methodology, we define our goals and tasks through the following key components available in JIRA:

- **Initiatives:** Collections of epics aimed at achieving common goals.
- **Epics:** Large bodies of work that can be further broken down into smaller tasks (stories).
- **Stories:** Also known as "user stories," these are short requirements or requests presented from an end user's perspective.
- **Tasks:** Individual tasks assigned to team members, measured in person-hour granularity.

For our project, given it is relatively small, we are using only Epics, Stories, and Tasks for simplicity. Each project phase has one Epic and one project leader assigned. Phase 2 is subdivided into two sub-phases due to its extended timeline including holiday breaks. Stories represent high-level goals, each of which is dissected into measurable, time-bound tasks, allocated among team members for execution

Thank you for reading! 🛫

This has been the data science team at Spark Airlines. Special thanks to Vinicio De Sola and the rest of the W261 course staff for their invaluable expertise and guidance on this project. We had a rich experience exploring this complex dataset and learning advanced tools for this project; we learned so much! And now, we're ready to take off!