databricks 261 Final Project - Phase III

(https://databricks.com)

AeroProphet

Group 2.2

Show code

Team

Show code

Hailee Schuele

hschuele@berkeley.edu

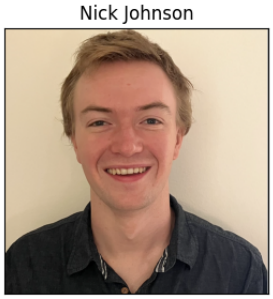
Landon Yurica

lyurica@berkeley.edu

Sreeram Ravinoothala

sreeram@berkeley.edu

Nick Johnson



nickjohnson@berkeley.edu

Phase Leaders

- Phase I: Hailee
- Phase II: Nick
- Phase III: Sreeram
- Phase IV: Landon

Final Phase Assignment Table

Date	Sreeram	Nick	Hailee	Landon
12/5	DL experiments	update cv fold data, clean up blob	cv staleness metric scaling, RF & GBDT experiments	F.E. max_weather
12/6	DL experiments	Restructure Cross Fold Data	RF & GBDT experiments	F.E. airport size

Date	Sreeram	Nick	Hailee	Landon
12/7	DL experiments	train MLP on 2015-2018 data	RF & GBDT experiments	F.E.
12/8	DL experiments	grid_search best model on 2015-2018 data	RF & GBDT experiments	F.E. airport centrality
12/9	DL experiments	re-run correlation analysis	RF & GBDT experiments	Re-checkpoint data
12/10	slides	grid_search best model	slides	E.C. data cleaning
12/11	slides	slides	slides	slides
12/12	slides	slides	slides	slides
12/13	presentation	presentation	presentation	presentation

Abstract

The overall objective of this project was to predict departure delays greater than 15 minutes, 2 hours before takeoff. The business case is described in more detail in the next section, but the results of this analysis have significant benefits for both airlines and passengers.

In Phase I of this project, we sorted out project management and our computing environment. We also made plans for our data pipeline and did our initial EDA to identify variables of interest.

In Phase II of this project, we first performed a join to combine our datasets. This was followed by the iterative data cleaning, EDA, and feature engineering process. We ran a logistic regression model as our baseline, after which we conducted LASSO regularizaiton. The resulting 24 features were used in another logistic regression, which showed substantial improvements over our baseline.

In the final Phase III, we started by performing additional feature engineering and EDA to ensure we were getting the most utility out of the data. We then experimented with a variety of models (logistic regression, random forest, gradient boosted decision

trees, multilayer perceptron, and long short-term memory), hyperparameters, and LASSO regularization to see how they performed on the validation data. The best model was selected, trained on the full data, and tested on an unseen data cut to give us our final performance.

Our final model showed a 35% increase in F1 score from our baseline model. Further investigation showed that precision and recall were very similar, indicating we have a balanced model

Business Case

The ability to anticipate delays will usher in a new era of empowered decision-making for passengers and seamless operations for airlines.

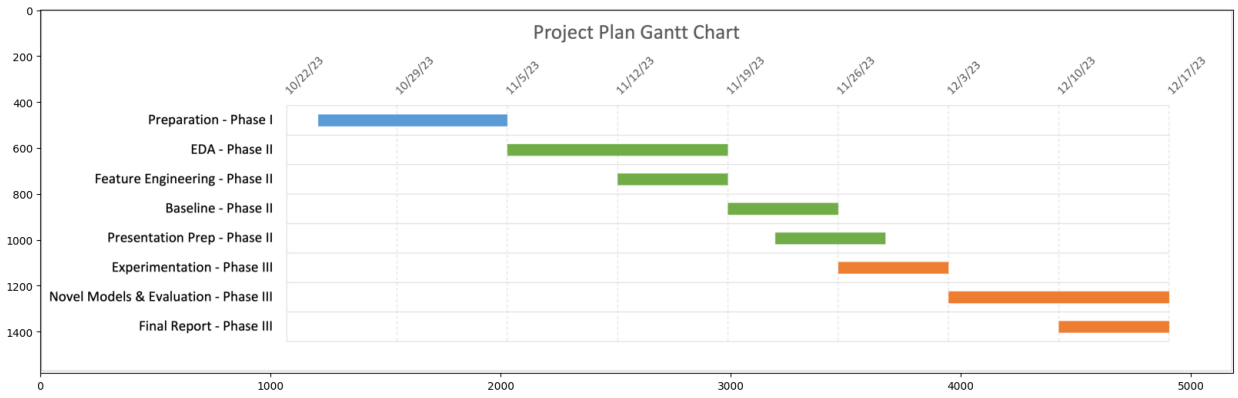
Armed with this foresight, passengers will be able to adapt their plans in real-time and make more informed choices from the outset.

Meanwhile, airlines will be able to more proactively address issues and come up with timely resolutions if they're able to foresee delays. Beyond these immediate benefits, this predictive analysis will also shed light on the major causes of delays, which airlines can use to improve their operational efficiency and elevate the customer experience.

In a world that puts an increasing emphasis on efficiency, the power of real-time insights will transform the travel landscape.

Timeline

Show code



Outcome Definition

For this project, we are dealing with a binary classification problem since our goal is to categorize each flight as "delayed" or "not delayed". With classification tasks, accuracy is a poor metric since always predicting the majority class can lead to high accuracy (especially in unbalanced datasets). As a result, we plan to focus on other evaluation metrics: *F1 Score* and *AUC*.

F1 Score: To get an idea of performance at a specific threshold value, we can leverage the F1 score. This is the harmonic mean between recall & precision.

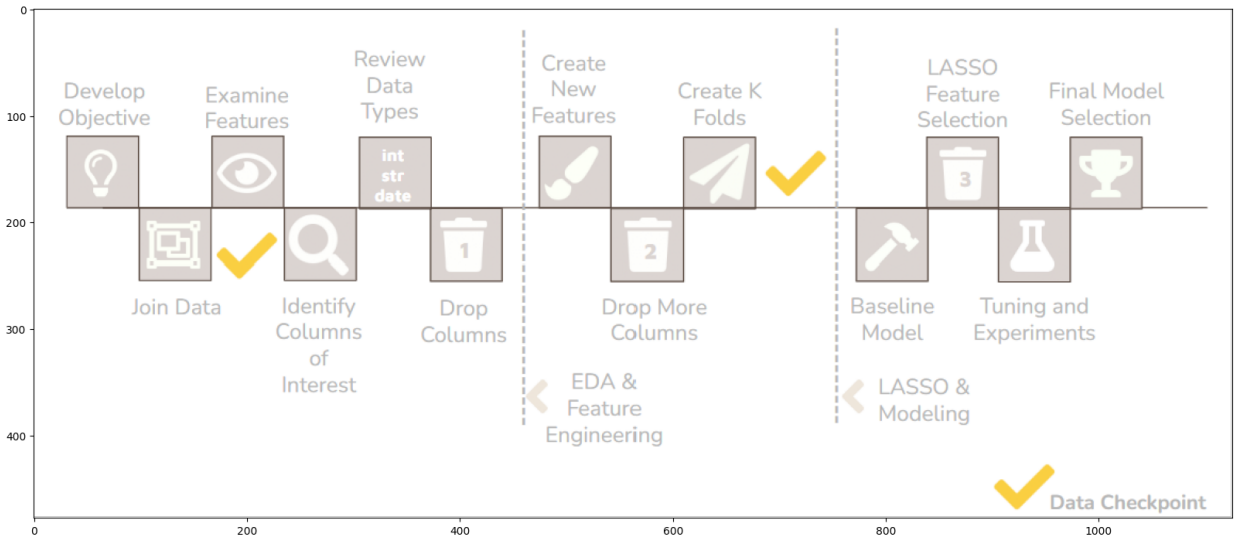
$$2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

AUC: There is a tradeoff between recall and precision as controlled by a threshold value. Typically, if the probabliy of the positive class is greater than 0.5 we predict a positive and negative otherwise. As the threshold is increased, we become more conservative and precision will increase at the expense of recall. As the threshold is lowered, the inverse occurs. We can visualize model performance at all threshold values by plotting an ROC curve (receiver operating characteristic curve). Finding the area under this curve, called the AUC, summarizes the total performance of the model for all threshold values.

F1 score was our primary evaluation metric because it uses a balanced approach between disrupting flight decisions and providing a reliable metric. If there were any tiebreakers or close calls, we decided we'd use AUC. Both metrics are common for evaluating classification models, like logistic regression. AUC also summarizes performance across different classification thresholds.

Modeling Pipeline

Show code



Data

Our model is built on publicly available data from the Bureau of Labor Statistics on US flights from 2015 to 2019, and weather station data from the National Center for Environmental Information during the same time period.

The 4 datasets are summarized in both tables below:

Dataset	Source	Description
Flights	Bureau of Labor Statistics	US flight data from 2015 to 2019, containing information on airlines, date, origin and destination airports, flight delay, time of departure, arrival. This is a historical data that can be used to figure out relevant features required to model to predict or classify if and when a flight is delayed.
Weather	National Center for Environmental Information	This dataset contains weather information including wind speed, dewpoint, visibility, elevation, humidity, precipitation, in hourly, weekly, and monthly intervals from weather stations located around the world.
Airport	DataHub	This dataset contains location, size, and identification information about US airports.
Weather Stations	DataHub	This dataset contains location and identification information about weather stations and their distance to nearby airports.

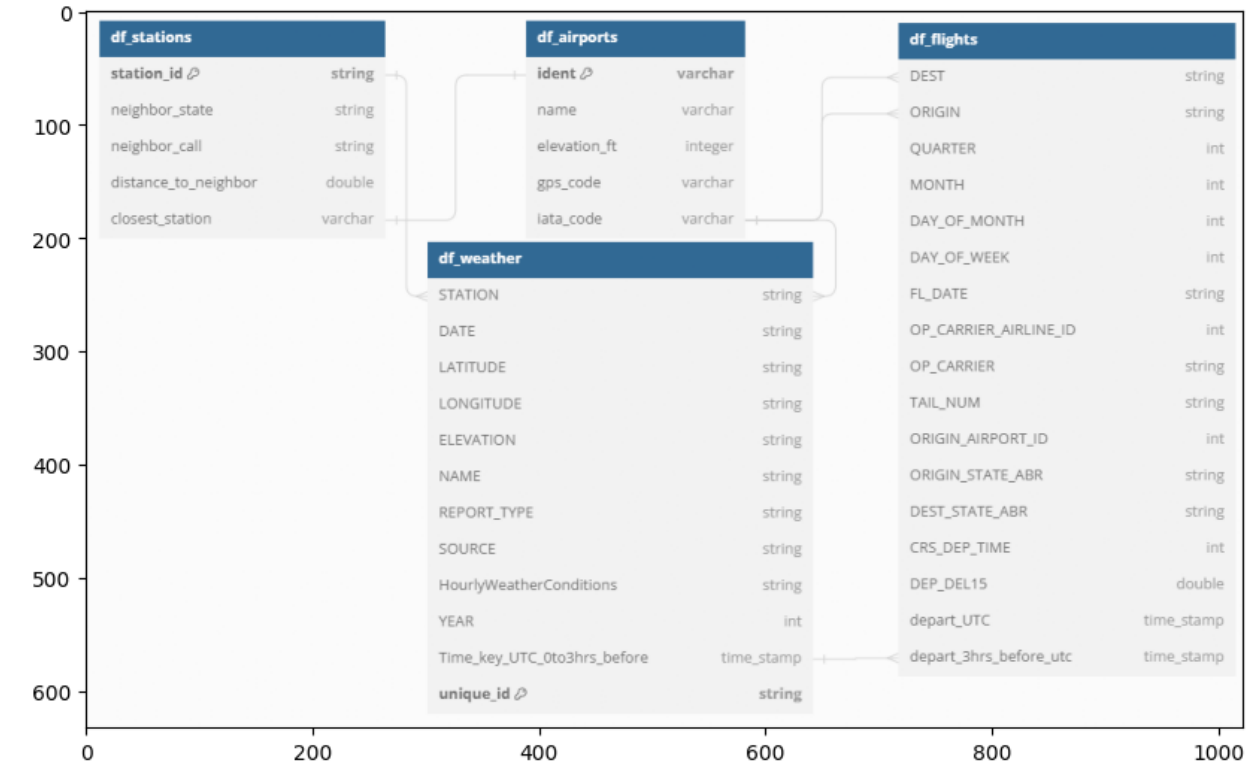
Dataset dimensions and memory requirements

Table	Rows	Columns	Memory (GB)
df_flights	74,177,433	109	2.93
df_weather	898,983,399	124	35.05
df_stations	5,004,169	12	1.3
df_airports	57,421	12	0.01

Data Join

We were able to combine these four datasets through a series of data transformations and joins, leveraging the relationships illustrated in the simplified schematic below.

Show code



The table below briefly summarizes the steps involved in our data join:

Step	Transformation	Notes
1	Calculate min distance of weather stations to airports	
2	Join Airport and Weather Station	Airports['ident'] ==Station[`neighbor_call`]
3	Join Flights	Flights[`ORIGIN`] == Airports_Stations[`iata_code`], Flights[`DEST`] == Airports_Stations[`iata_code`]
4	Convert Flight and Weather times to UTC	
5	Create time key in Flights and Weather	Flight and weather time rounded to nearest hour
6	Calculate Min, Max, Avg. and Most Recent Weather reports within 2 hour interval for every Station	
6	Weather from 2 hours prior joined with Flights	Flights_Airports_Weather['origin_station_id'] == Weather['STATION `] & FAW[` Time_key]== Weather[`time_key_2hr_utc`]

Joined data details and join time

Table	Rows	Columns	Memory (GB)	Run Time (HH:MM:SS)
df_flights	74,177,433	109	2.93	None
df_weather	898,983,399	124	35.05	None
df_stations	5,004,169	12	1.3	None
df_airports	57,421	12	0.01	None
df_FSW	72,515,921	292	43.5	02:13:00
df_FSW_Clean	64,457,088	87	64.78	06:18:00

Cluster Details: DBR 13.3 LTS ML, Spark 3.4.1, Scala 2.12, Standaard_DS3_v2, 14GB, 4 Cores, Standard_DS3_v2, 84GB, 24 Cores, 1-6 workers

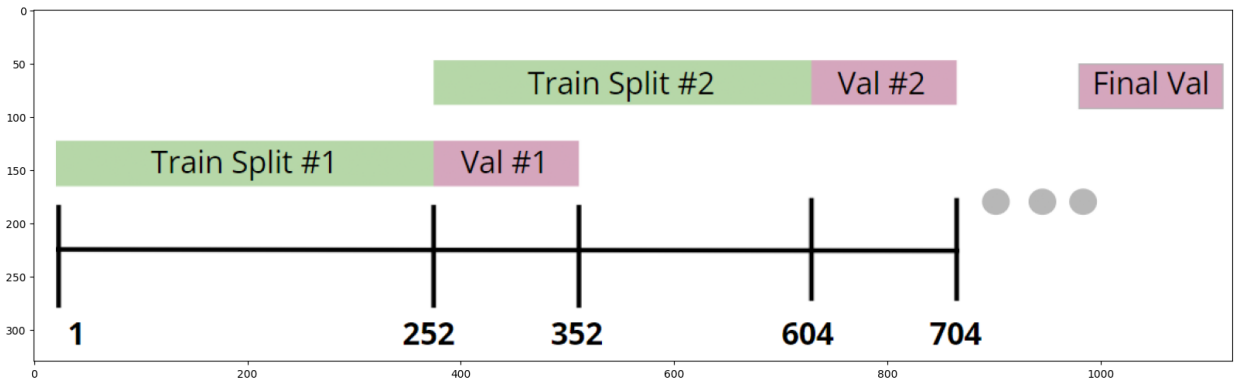
Please use the following link to see our full Phase III - Data Join code (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/661347987654537/command/15371548917966927

Train/Test Split and Cross Folds

Since our data is temporal in nature, we used standard time series methods for the train/validate/test splits and cross validation. This prevents data leakage by ensuring data in the future is not used to make predictions about the past.

To set up for modeling, we first split our train data (2015-2018) into train/validate cross folds. Each sequentual fold incorporates the prior validation dataset as illustrated in the figure below. We set aside 100 days of 2018 data for our larger validation set to be used during intermediary modeling and hyperparameter tuning. The test set was comprised of 2019 data and remained unused until our final model evaluation.

Show code



EDA & Data Cleaning

Please use the following link to see our full Phase III - EDA code (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1346418319517318 (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1346418319517318))

Our preliminary data cleaning included dropping columns with substantial missing data, redundant information, or that risked data leakage.

After dropping duplicate rows, we used mean imputation where applicable to fill missing data. We opted for using the mean over median because median resulted in increased shuffle operations resulting significantly longer compute requirements.

Pearson Correlation Analysis

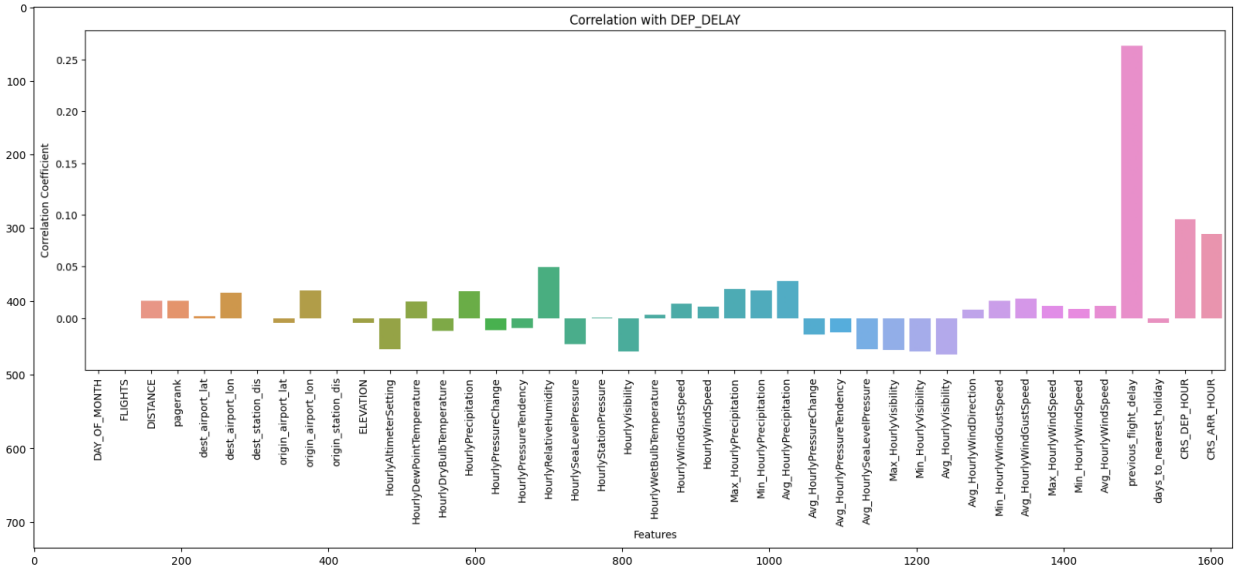
To evaluate our features, we ran a pearson correlation analysis. First, we checked the correlation between our features and found that the maximum and minimum features often correlated to the average of the same measurement. In cases where max or min correlated over 95% to the average, we decided to mantain only the average column.

Next we reviewed the correlation of each feature with a continous version of our outcome variable (DEP_DELAY). This was helpful in reviewing feature importance. Any non-dervived feature with 0.01 correlation or lower was dropped from our dataset.

Features Dropped Due to Low Correlation:

- FLIGHTS
- dest_station_dis
- origin_station_dis
- ELEVATION
- HourlyPressureTendency
- HourlyStationPressure
- HourlyWetBulbTemperature
- Avg_HourlyWindDirection
- Min_HourlyWindSpeed

Show code



Feature Engineering

With 292 features, dimensionality reduction was essential for a viable, scalable model. To do this, we evaluated redundancy, Pearson correlation with target variable, and LASSO regularization. We ultimately selected 40 variables with information we believed to be most useful in predicting if a flight will be delayed 2 hours prior to scheduled departure.

Feature transformations included standardization to bring each feature value to similar value scales, data type conversions from string to the appropriate integer/float/datetime, and one hot encoding. We imputed the mean for missing values. We also applied weights to account for class imbalance as the number of "not delayed" flights heavily outweighed "delayed" flights.

The table below details our final raw and derived features.

Feature	**Dataset Origination**	**Defintion**
OP_UNIQUE_CARRIER_dummy_DL	flights	dummy variable identifying a flight as being flown by Delta Airlines

Feature	Dataset Origination	Defintion
OP_UNIQUE_CARRIER_dummy_WN	flights	dummy variable identifying a flight as being flown by SouthWest Airlines
OP_UNIQUE_CARRIER_dummy_OO	flights	dummy variable identifying a flight as being flown by SkyWest Airlines
OP_UNIQUE_CARRIER_dummy_UA	flights	dummy variable identifying a flight as being flown by United Airlines
OP_UNIQUE_CARRIER_dummy_B6	flights	dummy variable identifying a flight as being flown by JetBlue Airlines
OP_UNIQUE_CARRIER_dummy_AS	flights	dummy variable identifying a flight as being flown by Alaska Airlines
OP_UNIQUE_CARRIER_dummy_HA	flights	dummy variable identifying a flight as being flown by Hawaiian Airlines
OP_UNIQUE_CARRIER_dummy_F9	flights	dummy variable identifying a flight as being flown by Frontier Airlines
OP_UNIQUE_CARRIER_dummy_VX	flights	dummy variable identifying a flight as being flown by Virgin America
OP_UNIQUE_CARRIER_dummy_NK	flights	dummy variable identifying a flight as being flown by Spirit Airlines
OP_UNIQUE_CARRIER_dummy_OH	flights	dummy variable identifying a flight as being flown by PSA America
OP_UNIQUE_CARRIER_dummy_US	flights	dummy variable identifying a flight as being flown by Noble Air Charter

Feature	Dataset Origination	Defintion
DAY_OF_WEEK_dummy_1	flights	dummy variable identifying if the flight departed on a Sunday.
DAY_OF_WEEK_dummy_2	flights	dummy variable identifying if the flight departed on a Monday.
DAY_OF_WEEK_dummy_3	flights	dummy variable identifying if the flight departed on a Tuesday.
DAY_OF_WEEK_dummy_4	flights	dummy variable identifying if the flight departed on a Wednesday.
DAY_OF_WEEK_dummy_5	flights	dummy variable identifying if the flight departed on a Thursday.
region_dummy_Midwest	derived from flights	dummy variable identifying if the flight departed from an airport in the "Midwest" region of the United States.
region_dummy_West	derived from flights	dummy variable identifying if the flight departed from an airport in the "West" region of the United States.
origin_type_dummy_large_airport	Airports	dummy variable identifying a origin airport as being large
MONTH	flights	Integer value representing the Month of a departing flight
DAY_OF_MONTH	flights	Integer representing the numerical day of the month of a departing flight
DISTANCE	flights	Integer distance between a flight's origin airport and destination airport

Feature	Dataset Origination	Defintion
pagerank	derived from flights	float32 Pagerank score of a origin airport measuring its centrality
dest_airport_lat	airports	float32 latitude of destination airport
dest_airport_lon	airports	float32 longitude of destination airport
origin_airport_lat	airports	float32 latitude of origin airport
origin_airport_lon	airports	float32 longitude of origin airport
CRS_DEP_HOUR	flights	local Datetime value of a flights scheduled departure hour
CRS_ARR_HOUR	flights	local Datetime value of a flights scheduled arrival hour
HourlyAltimeterSetting	Weather	float32 value measuring most recent hourly altimeter reading from closest weather station to departing airport 2 hours prior
HourlyDryBulbTempature	Weather	float32 value measuring most recent hourly Dry Bulb reading from closest weather station to departing airport 2 hours prior
HourlyDewPointTemperature	Weather	float32 value measuring most recent hourly Dew Point Temperature reading from closest weather station to departing airport 2 hours prior
HourlyRelativeHumidity	Weather	float32 value measuring most recent relative humdity reading from closest weather station to departing airport 2 hours prior

Feature	Dataset Origination	Defintion
HourlyPrecipitation	Weather	float32 value measuring average hourly precipitation reading from closest weather station to departing airport
HourlyRelativeHumidity	Weather	float32 value measuring average hourly relative humidity reading from closest weather station to departing airport
HourlyWindSpeed	Weather	float32 value measuring most recent hourly wind speed from closest weather station to departing airport
Avg_HourlyPrecipitation	Weather	float32 value measuring averag hourly precipitation from closest weather station to departing airport
Avg_HourlySeaLeavelPressure	Weather	float32 value measuring averag hourly Sea Level Pressure from closest weather station to departing airport
Avg_HourlyVisibility	Weather	float32 value measuring averag hourly visibility from closest weather station to departing airport
Avg_HourlyWindSpeed	Weather	float32 value measuring averag hourly Wind speed from closest weather station to departing airport
previous_flight_delay	derived from flights	Boolean value that is '1' if a departing flight's scheduled aircraft tailnumber was delayed on its previous flight

Feature	Dataset Origination	Defintion
Plane_Delays_last_24H	derived from flights	Integer value of the total number of times a departing flight's scheduled aircraft tailnumber was delayed in the previous 24 hours
days_to_nearest_holiday	derived from flights	Integer value of the total number of days to the closest US holiday before or after the scheduled flight

Derived Features

Below are the final feature we engineered for use in our models

Feature	Description
previous flight delay	length of previous flight delay for same aircraft (if takeoff was > 2 hrs before next flight)
depature region	west, south, midwest, northeast, southeast
plane_delays_last	sum of delays for the same plane over the last 24h

Feature Selection

LASSO Regularization

After experimenting with different feature selection methods in the previous phases, in this final phase we decided to perform LASSO regularization and retain any features that were not zeroed out by at least one fold. We started with 70 features and ended up with 43 (including derived features) after LASSO.

Modeling

Please use the following link to see our full Phase III - Modeling code (https://adb-4248444930383559.19.azuredatabricks.net/?o=4248444930383559#notebook/1962809137470270 (https://adb-

4248444930383559.19.azuredatabricks.net/?
o=4248444930383559#notebook/1962809127470270))

Logistic Regression

We used logistic regression for our initial models as it's a simple way to model classification problems and is easy to interpret.

Our very first logistic regression was run in Phase II and contained our original set of 50 cleaned features without any derived features. That was followed in Phase III by another logistic containing a newer set of 70 cleaned and derived features. Our final logistic contained the 38 features retained by LASSO and 5 derived features.

Random Forest

The Random Forest was only run on the LASSO + derived features outlined above. Before running on all folds, we tested a series of hyperparameters on a subset containing only the first train/val fold. These hyperparameters included the feature subset strategy, maximum depth of the tree, and the number of trees. The default hyperparameters typically had better or about the same performance on the validation fold as others that were tested. When performance was similar, we chose the option that would be less computationally expensive. Runtimes were pretty consistent across experiments. In some instances, we averaged results across multiple seeds. The hyperparameters chosen by experimentation and used in the final RF model are listed below.

- numTree = 10
- featureSubsetStrategy = auto
- maxDepth = 5

Gradient Boosted Decision Trees

A very similar approach to the Random Forest hyperparameter selection was used for GBDT. Again, only the LASSO + derived features were used on a subset of the folds. Tested hyperparameters included the feature subset strategy, step size (a.k.a learning

rate), and maximum number of iterations. Again, the default settings appeared to prevail and had similar runtimes to other options. The hyperparameters chosen by experimentation and used in the final GBDT model are listed below.

- maxIter = 10
- stepSize = 0.1
- featureSubsetStrategy = all (which in experiments appeared to be the same as auto)
- maxDepth = 5

Multilayer Perceptron (MLP-43 - Sigmoid - 2 Softmax)

MLP is a deep learning model which was applied on the lasso and derived features on all the folds. Multiple experiments were conducted on the vector representation of the features that were selected by changing the number of hidden layers, maxIter, blockSize. F1, Auc, were captured for each of the runs while the number of nodes in the last layer were chosen as 2 for binary classification. All the runs came with similar outpurts for these parameters.

Long Short-Term Memory (LSTM-43-Sigmoid-1tanh)

LSTM is another deep learning model that was applied on the lasso folds as well as the final validation data. Similar to MLP, multiple experiments were conducted by changing the number of layers, epochs, loss functions, optimizer, learning rate. The output layer chosen here is 1 as this is a binary classification. Finally the epochs were restricted to 10 as the loss converged to a minimum at 10th epoch. F1 and AUC scores were collected where the AUC score was close but less than the ones from MLP while F1 was half of the MLP one. As the model was taking too long to run on all the 5 folds, only 10% of that data was chosen to be run with LSTM. Given good resources, we would have been able to run the LSTM on entirety of dataset. If given more time, we would go in this path of experimentation using LSTM or other DL models.

Time taking to run 10 epochs with BCELoss and Adam optimizer with just 10% of data, learning rate of 0.001 was around 2.25hrs. With changing minimal parameters, with 10% of data we were getting the scores less than other models. Hence we stopped pursuing this route.

Runtime Table

Job	**Runtime**
Post-Cleaning Checkpoint	216 minutes
Create Folds	18 minutes
Initial Logistic Model w/o Derived Features	53 minutes
Updated Logistic Model w Derived Features	27 minutes
LASSO Regularization	31 minutes
Logistic Model w LASSO and Derived Features	24 minutes
Grid Search	69 minutes
Random Forest	29 minutes
GBDT	30 minutes

Job	Runtime
MLP 1 Hidden Layer	45 minutes
MLP 2 Hidden Layers	40 minutes
LSTM	136 minutes

Evaluation

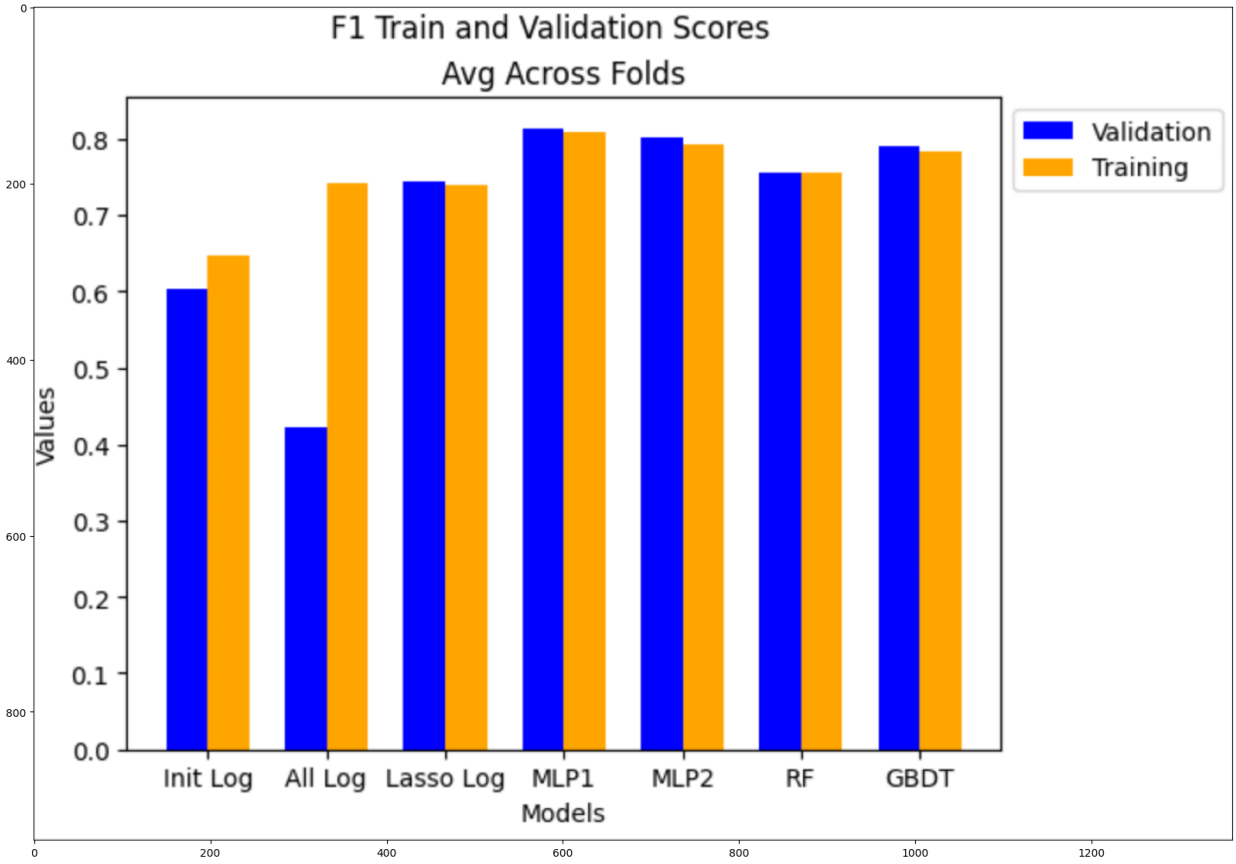
The results of the model experimentation can be seen in the table and graph below. We evaluated their performance on the large validation set (blue bars). We averaged results across folds to get the training scores (orange bars). There is a **35%** increase in validation F1 score from our best MLP1 model compared to our Initial Logistic.

It's worth nothing that the validate score is noticeably less than the train score for the second logistic regression model. However, after LASSO was performed, the validate scores jumped right back up to meet the train scores. After LASSO, the train/validate scores weren't too far off from one another so we don't think overfitting occurred.

The MLP and GBDT models performed similarly on the validation cut. That said, MLP with 1 hidden layer performed best (last row in the table), so we used it as our final model.

Model	Train F1	Val F1
Initial Logistic	0.648	0.604
Secondary Logistic	0.741	0.422
LASSO Logistic	0.740	0.743
MLP 2 Hidden Layers	0.792	0.801
Random Forest	0.755	0.756
Gradient Boosted Decision Trees	0.783	0.791
MLP 1 Hidden Layer	***0.808***	***0.814***

Show code



The final MLP model was fit on a larger training cut without folds (2015-2018) and evaluated using the test cut (2019), which had remained unseen up to this point.

The F1 score for the training data was 0.805 and **0.803** for the test data.

Gap Analysis

Our data join captured flight and weather data from 2015-2021. Out of personal curiosity we took our final model and used 2019-2021 data as our holdout test set (instead of 2019 only). Our model had an F1 Score of 0.656, a notable reduction in performance from our 2019 only F1-score. This is not entirely surprising given the severe impact of Covid-19 on airline traffic patterns, which occured primarily during that time frame. It gave the team a healthy appreciation for data shift that occurs naturally over time. It clearly illustrated the importance of regularly evaluating and retraining models to maintain desired performance.

A few other gaps our team indentified relate to pre-processing and feature engineering. We used standard scaler to rescale our data which may have skewed our distributions for non-normal features. In future work, we'd like to leverage min max scaling instead for non-normal features. Next we identified a small area of data leakage relating to page rank. We calculated page rank on the entire dataset instead of using past data. We don't believe this had a significant affect on performance, but when put into production, our model will need to leverage existing page ranks and could lead to slight performance changes.

One area our model performed quite well is striking a balance between recall and precision. As discussed, we achieved a high F1 score. However, this could be due to high precision and low recall, or vice versa. In reality, our model achieved a 0.806 precision and 0.831 recall. Our goal at the outset was to achieve balance so we can

Conclusion

Our original hypothesis was that thoughtful data engineering and feature selection would enable us to accurately forecast flight delays greater than 15 minutes, 2 hours ahead of time. After many rounds of exploration and experimentation, our final results indicate that we have a useful model for prediction. This product will help both airlines and customers make informed decisions and enhance operational efficiency.

Throughout the phases of this project, we performed iterative EDA, data cleaning, feature engineering, feature selection, and model experimentation. We used common practice time series analysis to identify 43 variables that were useful for prediction. Our efforts resulted in a final model that gave us a large improvement over our baseline model and is balanced in its predictions.

If we were to continue on with this project, we'd conduct additional hyperparameter tuning, look into creating more derived features, experiment with ensemble modeling methods, and try additional deep learning techniques.