Final Report

MIDS w261: Machine Learning at Scale
Steve Carr, Florencia Froebel, Mike Varner, and John Scott (Summer 2023)

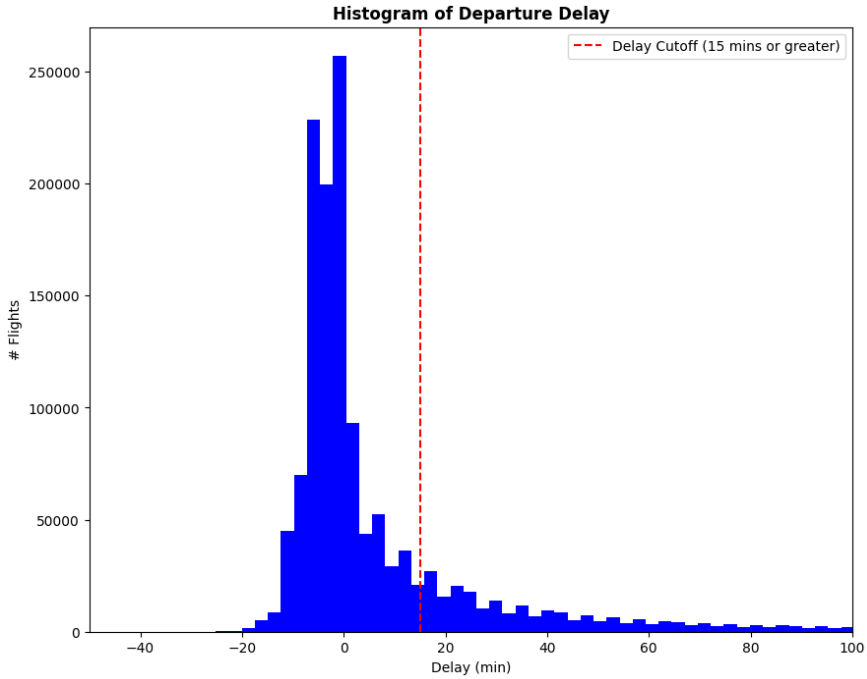
Abstract & Business Case

The airline industry finds itself in a particularly difficult position: no one cares about how smoothly it usually operates when problems and irritations arise. The complexities of the airline industry are global in scale, with millions of travelers each day supported in the air and on the ground by huge numbers of workers in extremely expensive airports. Despite the numerous impediments, the airline industry can be surprisingly efficient and resilient.

As annoying as flight delays are for passengers, they are also the bane of the airline industry, which runs extraordinarily capital-intensive operations at low margins. Delays mean unhappy customers, lost staff time and wages, a buildup of flights which causes ripple effects for other airports, and poor overall financial results. Some sources estimate net losses as greater than $30B per year (Ball et al., 2010). In view of this situation and the potential reward, realistic parameters on a predictive machine learning model would be to estimate the probability of a given flight being delayed by 15 or more minutes based on data two hours prior to scheduled departure. Based on flight and weather data from 2015-2019, we present data preparation methods as well as the results from several different delay-predicting machine learning models trained on this data.

Introduction to the Dataset

The originating sources of data being used are based on publicly-available weather and flight data collected from 2015 to 2019. Local Climatological Data (LCD), available from the National Centers for Environmental Information, allows us to integrate geographical weather data with our other primary data source: Airline On-Time Performance Data. The flight data contains a flag for whether a flight was delayed more than 15 minutes, which was considered our outcome feature for later machine learning modeling.



This figure shows the distribution of departure delays from the first year of data.

The flight data for this project has been provided already joined with the relevant geographical weather data, allowing for processing on a single monolithic dataframe representing a set amount of time (e.g. 6, 12, or 60 months). Some key groups of features from the dataset include our outcome feature, DEP_DEL15, which is 1 if a flight is delayed for more than 15 minutes, features related to identification of a single flight (LIST OF FEATURES), features used to locate the flight in space (LONGITUDE, LATITUDE) and time (YEAR, QUARTER, FL_DATE). Features that may feed into our models as input include HourlyDewPointTemperature, HourlyVisibility, and HourlyWindSpeed.

EDA and Feature Engineering

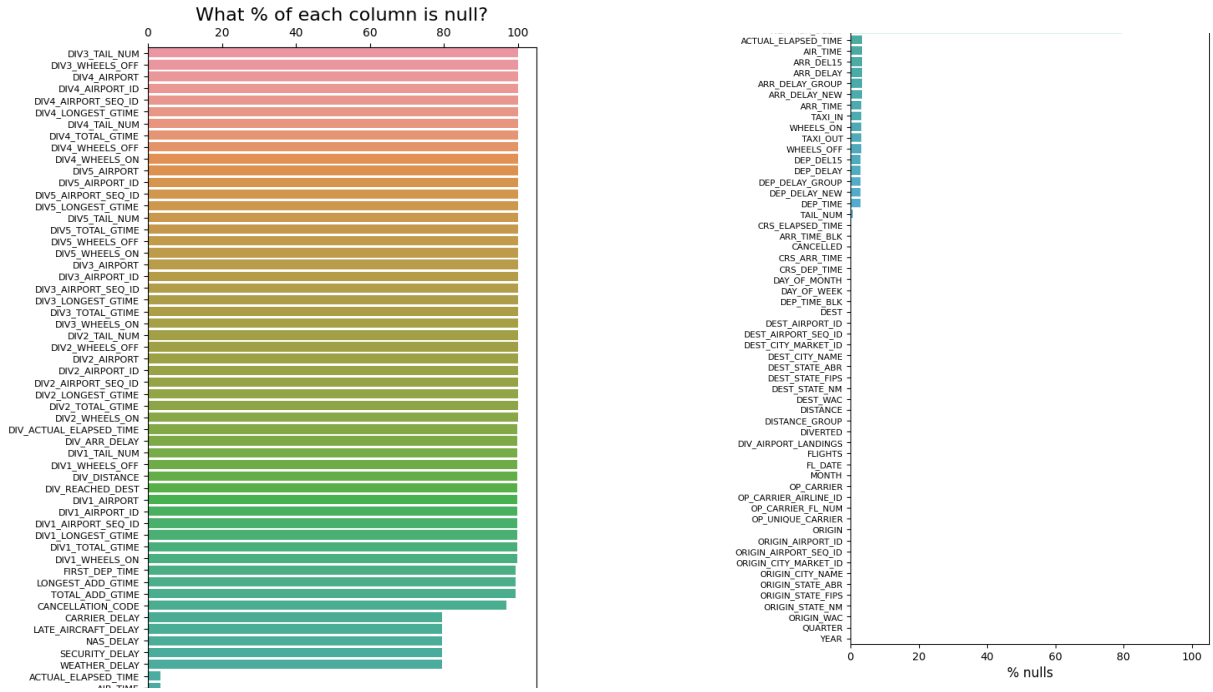
The OTPW_60M dataset, which contains 60 months (5 years) of flight and weather data starting on the first day of 2015, contains over 30 million records and over 200 columns. The list of subsetted columns is shown below:

Outcome Feature: 'DEP_DEL15'

Identification Features: ‘QUARTER’, 'DAY_OF_MONTH', 'DAY_OF_WEEK', 'YEAR', 'MONTH', 'FL_DATE', 'OP_UNIQUE_CARRIER', 'ORIGIN', 'DEST', 'DEP_TIME', 'DISTANCE', 'origin_type', 'sched_depart_date_time_UTC'

These Identification Features are essential for uniquely referencing a single plane.

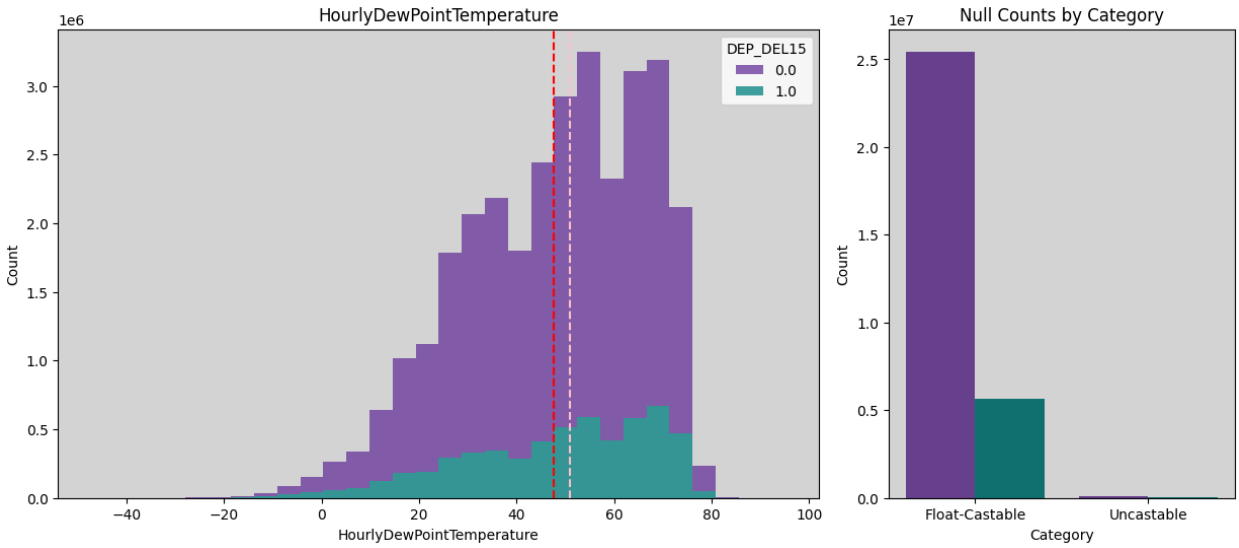
Below is a holistic view of the percentage of all null values in the airport side of our data set. As you can see, there are a lot of columns with null values. This was an additional feature that helped us narrow down our final feature selection. The features of interest we chose have few null values.



Weather Features: 'HourlyDewPointTemperature', 'HourlyDryBulbTemperature', 'HourlyPrecipitation', 'HourlyRelativeHumidity', 'HourlySeaLevelPressure', 'HourlyVisibility', 'HourlyWetBulbTemperature', 'HourlyWindDirection', 'HourlyWindGustSpeed', 'HourlyWindSpeed'

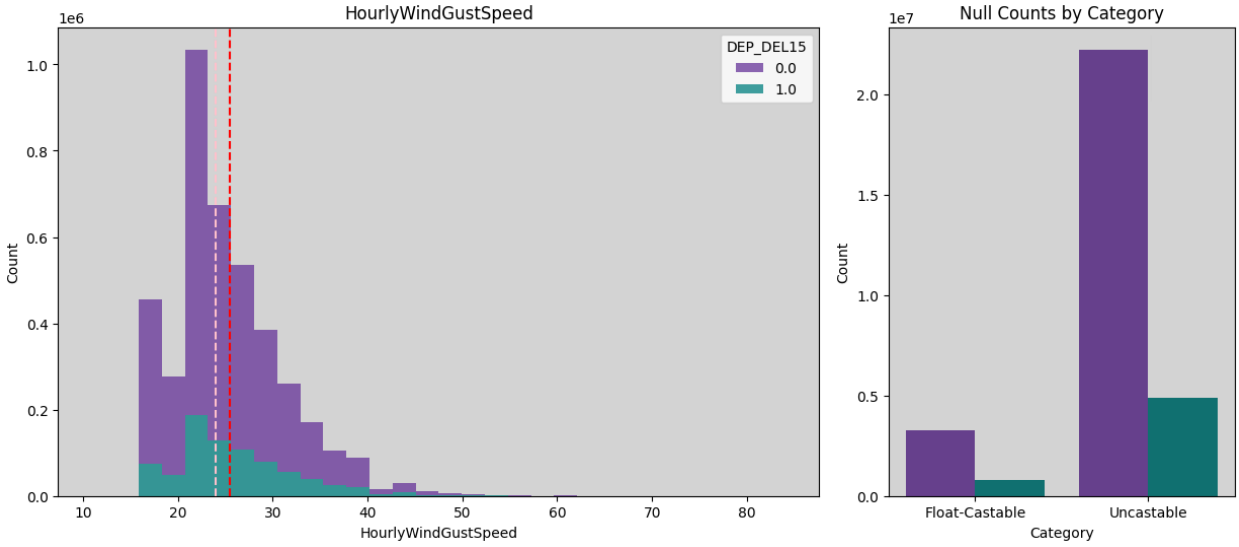
Due to the widespread prevalence of null values, imputation of null values was a concern, particularly for these Weather Features. Each numeric column was cast as float first, and uncastable values were replaced with the median value of that column. Each variable was inspected in this way (an included variable shown below):

As an example of an included variable, we show HourlyDewPointTemperature (this represents the entire dataset):



Two graphs include a stacked histogram on the left, and a count of the float-castable (numeric) vs uncastable variables on the right. The ratio of ~4:1 between our categories is apparent in the figure above, with purple representing the “on time” condition and teal the “delayed” group. The red and pink dashed lines show the mean and median values. On the right side, the amount of uncastable numbers was very small in comparison to the number of float-castable values in this column.

As an example of an excluded variable, we show HourlyWindGustSpeed.



While HourlyWindGustSpeed would seem like a valuable feature, the null-imbalance was too high on this column, and was therefore excluded. This was the only variable from above that was shown to have this pattern, leading to this feature’s exclusion from our modeling.

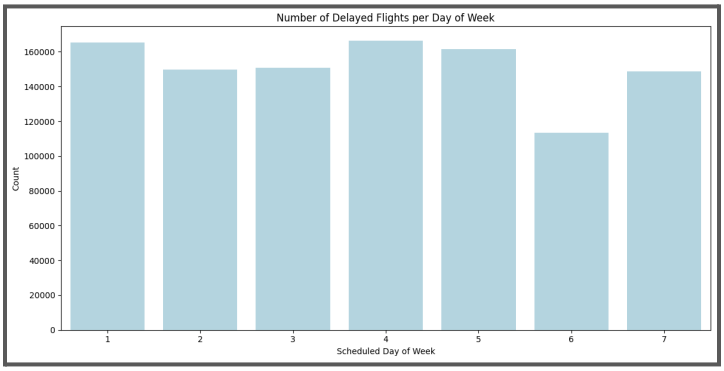
Engineered Features

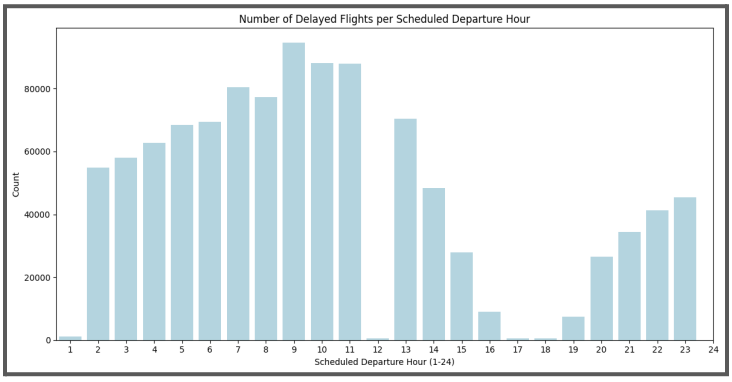
Our engineered features were made off of three distinct observations:

1. Simplify datetime into simpler features
    a. **FL_DATE:** convert datetime to only date (no time)
    b. **hour_of_day**: aggregate all flights within an hour without any minutes
    c. **part_of_day**: groups flights into 3, 8-hour buckets (evening to morning; mid-morning to mid-afternoon; afternoon to evening)
2. External features, bringing in outside information
    a. **is_holiday**: binary indicator for all US holidays +- 1 day
3. Derived features based on the data
    a. **DELAYED_PERCENTAGE_2_HOURS_PRIOR**: % of delayed flights 2 hours prior by date, hour, and origin airport
    b. Three features derived from the graph of airports + unique routes; for each airport
        i. **pagerank**: the pagerank of the airport based on the graph
        ii. **Out-degree**: number of unique routes that depart from this airport
        iii. **In-degree**: number of unique routes that arrive at this airport

These features should allow the various models to consider some information outside the scope of the initial table. Three of the features relate to time (simplifying or giving access to different timescales), one to external events that may impact air travel (holidays), and one is a measure of the percentage of delayed flights from the airport of origin.

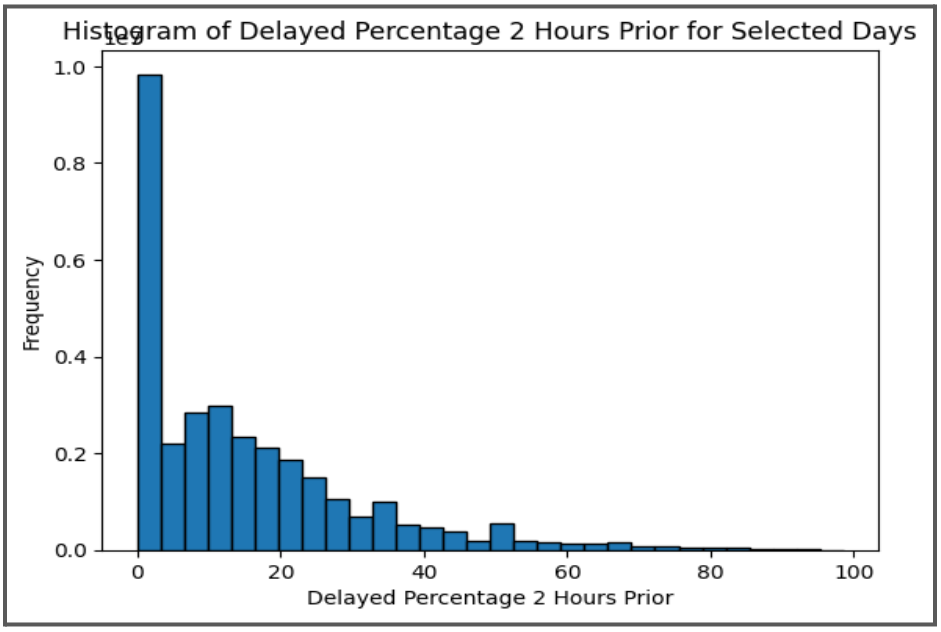
We were able to discover which of the identification time features were most relevant with respect to our label. The below EDA shows that the hour of the scheduled departure time seems to be likely to have more of an impact on our models than other time features such as day of the week.





Additionally, below is a histogram of our newly engineered feature DELAYED_PERCENTAGE_2_HOURS_PRIOR along with some key statistics. The purpose behind this variable is to represent the current state of the airport from which a given flight is preparing to takeoff. As can be seen in the above chart, many airports experience backlogs of flights, leading to a high percentage of delays two hours prior to departure. This is a reasonable, valuable, and easily knowable feature for a given airport to monitor.
As we can see the values for this feature are heavily skewed.

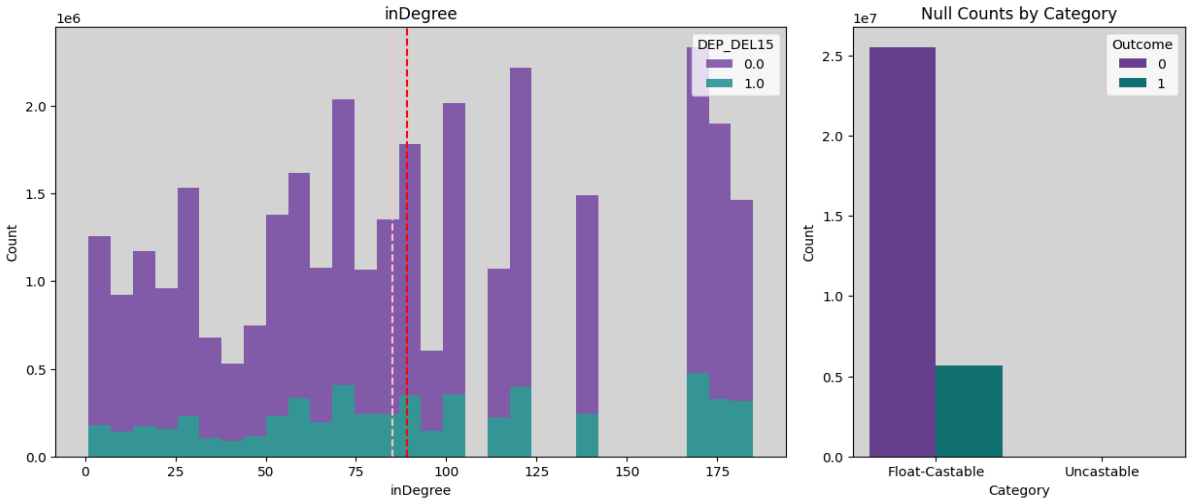
- Mean: 14.8%
- Standard Deviation: 16.3%
- Min: 0%
- Max: 98%
- Median: 10.95%

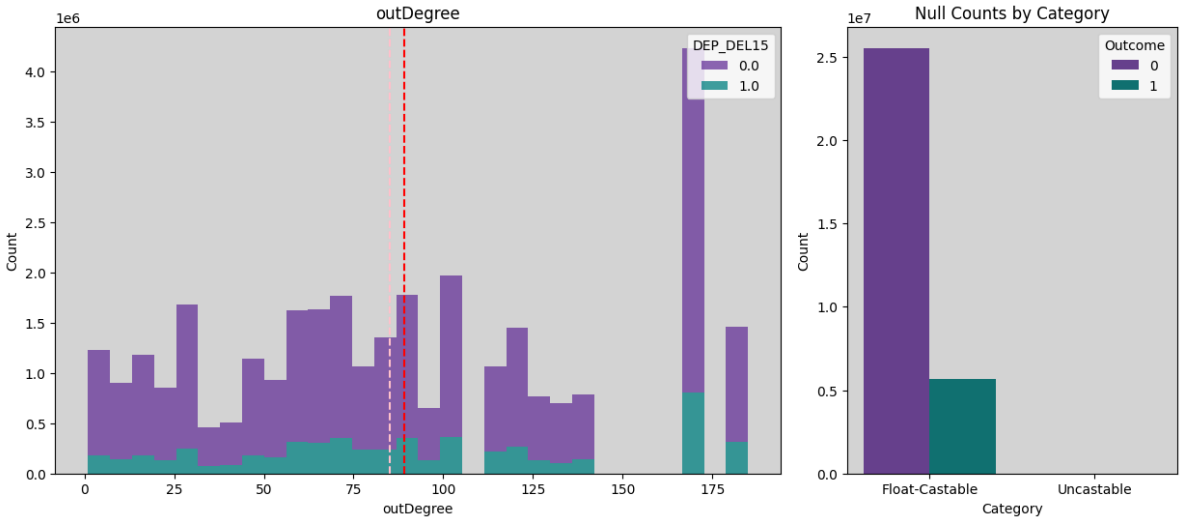


The most common state for an airport to be in is no-delays, but a long tail is visible, with decreasing amounts of percent delays.

Graph Features

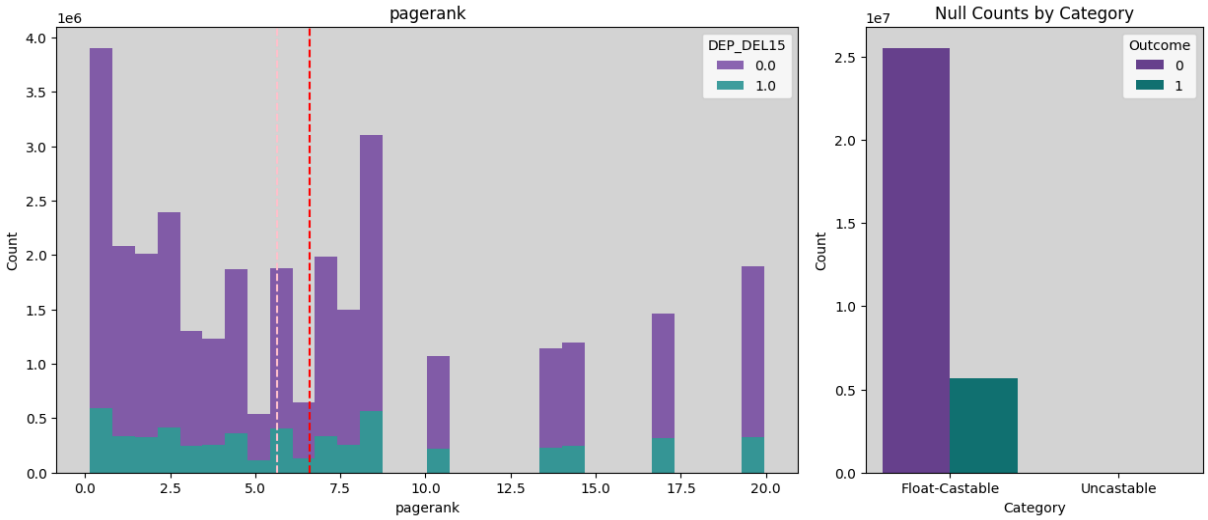
We can also explore the distributions for the graph features that were created below. Specifically the out and in degrees. The outdegree distribution typically stays between 0 and 135 but has a significant spike around 175 out degree nodes. Similarly for the in degree nodes, the largest spike is around 175.





The two distributions are very similar, which is to be expected, since flight routes most often would go both to and from a given airport to another.

Lastly, we take a look at the distribution of our pagerank feature, for which airline routes were used to construct a graph, and the pagerank of each airport included as information to each model along with each flight.



Given that pagerank is based on in/out edges, it is not surprising that a similar distribution is observed.

Pipeline

We structured our project into two separate pipelines. The first one ingested, transformed, and saved our data (data cleaning pipeline). Our second and final pipeline was used for model training, tuning, and lastly testing. We chose to break apart these pipelines to streamline our development process and minimize downtime. By having cleaned data, saved as parquet files, we could eliminate a dependency and tuned models independently of further feature engineering.

We implemented our data cleaning pipeline as a list of functions to step through sequentially. These functions consisted of the following:

- **Cleanser** - drop complete duplicates, cast features by appropriate type, created/cleaned date variables.
- **Feature_engineer** - Created holiday features, part of the day, and % of delayed flights 2 hours prior to flight hour.
- **OHE** - One hot encoded the non numeric variables into vectors for easy grouping after indexing.
- **Handle_nulls** - For the hourly weather variables, we chose to replace nulls with the median value. After this we dropped all rows that had any remaining nulls.
- **Cat_dataT** - Simply just converted categorical variables to strings to comply with data typing requirements
- **Custom_column_normalizer** - Normalize numeric features using MinMax scaling
- **Down_sample** - to account for the class imbalance between on time and late flights we downsampled our data.

- **Train_test_split** - Partitioned our data to isolate 2019 as the test data with everything prior being the training data. We did this to maintain model training integrity and generalizability.

We performed pagerank outside of our main data cleaning pipeline due to time constraints, but would ideally have been embedded within the feature engineering portion. We computed the pagerank features on the training data (through 2018) and included these values for 2019 via a left join, thus only including in the pagerank calculations our training data.

We would have ideally incorporated our time series cross-validation methodology into the data cleaning pipeline, but could not due to time constraints. That being said, this part of the project was not time intensive. To avoid overfitting, we used a 5 fold time series cross-validation methodology. Our training and validation data range from 2015 to 2018 with approximately yearly blocks each of which consisted of training and validation subblocks of 250 and 40 days respectively. Within each fold we trained on the first 250 subblock to preserve the temporal nature of the data (can't test on data that came before the training data). As the final test set, we used the last year of available data 2019.

The result of this process is an even distribution for each class so as to not bias the models. For modeling purposes, we combined all of our features into a single vector using Vectorassembler. We were able to combine categorical and normalized numeric variables into a single features vector but due to the mix of sparse and dense sub-vectors across the different folds it was not possible to perform validation on folds. Due to time constraints, we only used normalized numeric features into our models.

Modeling

Several different modeling algorithms were used in this project, in rough order of increasing complexity:

1. Logistic regression
2. Random Forest
3. Random Forest with hyperparameter tuning
4. Gradient Boosted Trees
5. Extreme Gradient Boosted Trees
6. MLP (multilayer perceptron) - only using one hidden layer
    a. Dimensions here
7. MLP (multilayer perceptron) - only using X hidden layers
    a. Dimensions here

Our first model is logistic regression as this is a foundation model for binary classification. This model takes all linear combinations of our vectorized features and assesses those inputs through a logistic function to determine if a flight will be delayed. Due to its simplicity, logistic regression without regularization is our baseline of comparison for all subsequent models. This model is easy to understand, implement and is expected to have the lowest performance out of

all tested models. Additionally, we ran logistic regression with lasso regularization but it does not yield any significant change of results.

Random forest is a common classification model that is constructed of multiple decision trees.This is a robust model and is able to process large numbers of variables which is critical for our dataset considering the number of 1-hot-encoded variables we have in our final dataset. In addition to the base random forest model we also performed hyperparameter tuning of the max depth and number of trees.

Extreme Gradient Boosting (XGBoost) builds upon the traditional gradient boosting algorithm that is scalable, efficient and potentially yielding better results than the previously discussed models. This model performs more precise regularization than the random forest models.

Multilayer perceptron (MLP) is our feed forward neural network model used in our research. Distinctly different in architecture from any previous model in that each variable is perceptron and we construct fully connected layers with a non-linear activation function to ultimately predict if a flight will be delayed or not. This construction may allow the model to see complex relationships between variables that other models could not.

Novel Approaches

We combined several different approaches to prepare and create our models for predicting flight delay, including some graph features based on airline routes (pagerank, incoming routes, and outgoing routes). These graph features are a way to give a measure of airport connectivity: airport hubs who service many incoming and outgoing flight routes would have high numbers of both in/out edges and a high pagerank.

Our choice for models were the among the most likely candidates based on the tabular and large dataset we had to train on. Our one-hot encoding and normalization steps are also likely common approaches to this type of problem.

We took a novel approach to analyzing our model results, which is fully described in the results section.

Evaluation Metrics

We've formulated our business case as a binary classification problem (delayed or not) and chosen to optimize for precision (Precision = True Positives / (True Positives + False Positives)). From a business perspective, we want to be able to trust that our models will correctly predict future flight delays because false positives are less costly than false negatives. Said differently, it's better to falsely predict a delay than falsely predict no delay. In our view, removing a delay (false positive) is more desirable than having an unanticipated delay (false negative). However, we will also carefully consider AUC in our model evaluation process.

Performance and Scalability Concerns

We had several performance and scalability issues that with more time and guidance could be mitigated. As described above, we had to break apart our data cleaning pipeline in order to prevent certain pieces, minmax scaling and vectorassembler, from crashing our instance. We would have ideally been able to do this on the entire dataset in a single go, but had to effectively break it into one year chunks by using the individual folds after the cross-validation splits. Additionally, we had significant issues with implementing SMOTE due to the large number of categorical variables that we one-hot-encoded.

Furthermore, this algorithm required a reduction of our data as well as a join of the new data with the original dataframe. This results in a lot of shuffling of our data as well as using the very expensive operation of table joins. This coupled with the large number of categorical variables that we one-hot-encoded resulted in a computationally expensive SMOTE algorithm that we did not have time to address. As a result, we decided to downsample our training data instead.

Results

In our experiments, several machine learning models were tested to evaluate their performance on the test datasets. The performance metrics used for evaluation include Area Under the Curve (AUC), Accuracy (Acc), Recall, Precision, and Training Time (in minutes). The results of the experiments are presented below:

Experimental Results on Test Datasets by Model %					
Model	**AUC**	**Acc**	**Recall**	**Precision**	**Train Time (min)**
Logistic Regression	74.52%	82.94%	20.67%	63.18%	18
Logistic Regression - regularization	74.52%	82.94%	20.67%	63.18%	16.2
Random Forest	68.78%	81.99%	5.28%	75.23%	17.5
Random Forest w/ Hyperparameter tuning	68.77%	81.99%	5.28%	75.23%	112.25
Gradient Boosted Trees	75.52%	82.96%	20.14%	63.89%	17
Extreme Gradient Boosted Trees	75.55%	82.96%	20.14%	63.89%	18.4
MLP (14, 30, 2)	75.20%	82.93%	20.97%	62.83%	32.2

Among the tested models, the XGBoost exhibited the highest AUC, albeit marginally better than the Gradient Boosted Trees and MLP. All models demonstrated an accuracy rate close to 82-83%, but there was a noticeable variation in recall. Notably, the Random Forest models displayed significantly lower recall rates than the other models but much higher precision. The training times varied considerably, with the random forest with hyperparameter tuning taking the longest time, which might be a factor to consider during model deployment. The consistent performance of the logistic regression model with and without regularization indicates that our data might not suffer from overfitting but remember that our dataset did not consider any categorical variables. Overall, we determine that the random forest models perform the best given their high precision scores.

To assess the results we look at the results of each model through 2 datasets: predictions by false positives and true negatives, and false negatives and true positives. Since we are only looking at numerical variables in our final test results we examine the differences in averages between the 2 subsets of datasets. This gives us insights to where the models are over or under predicting by variable. Below are the results from the logistic regression base model:

Logistic Regression Variable Analysis		
Category	**FN**	**FP**
HourlySeaLevelPressure	-0.025186	0.019044
HourlyVisibility	-0.024984	-0.279813
HourlyPrecipitation	0.000617	0.002117
DISTANCE	33.166226	5.280659
HourlyWindSpeed	0.04969	3.708013
pagerank	0.378874	-0.593636
HourlyRelativeHumidity	0.572311	1.665535
HourlyWetBulbTemperature	1.569604	-6.49105
HourlyDryBulbTemperature	1.789644	-8.079636
HourlyDewPointTemperature	1.852918	-6.702136
HourlyWindDirection	3.641685	18.774804
outDegree	3.891144	-4.256577
inDegree	3.946942	-4.263242
DELAYED_PERCENTAGE_2_HOURS_PRIOR	7.473148	10.631574

We can see that the logistic regression model does well in maintaining a similar average to either true positives or negatives for the variables highlighted in green. But for flights predicting false positives and negatives that it over indexes on delayed percentage of flights 2 hours prior.

Similarly for XGBoost, the best performing model by AUC, we show some clear differences when compared to logistic regression's variable analysis:

XGBoost Variable Analysis		
Category	**FN**	**FP**
HourlySeaLevelPressure	-0.021664	-0.020305
HourlyVisibility	-0.062085	-0.414436
HourlyPrecipitation	0.000933	0.003606
DISTANCE	34.798681	-0.997749
HourlyWindSpeed	0.358179	0.450747
pagerank	0.277154	-0.44349
HourlyRelativeHumidity	0.599328	7.107112
HourlyWetBulbTemperature	0.853617	-1.361894
HourlyDryBulbTemperature	0.968737	-3.430868
HourlyDewPointTemperature	1.084382	0.145437
HourlyWindDirection	5.752587	3.811973
outDegree	3.043153	-3.261304
inDegree	3.094946	-3.293972
DELAYED_PERCENTAGE_2_HOURS_PRIOR	8.231332	27.410501

In this case most of the variables in false negative and positive were similar to either true positive or negative flights respectively. Interestingly, the delayed percentage was overestimated even more so than in logistic regression.

Our best performing model by precision was random forest with hyperparameter tuning. The variable analysis showed quite different results from the previously shown table:

Random Forest with Hyperparameter Tuning Variable Analysis		
Category	**FN**	**FP**
HourlySeaLevelPressure	-0.025999	-0.068346
HourlyVisibility	-0.10551	-1.666156
HourlyPrecipitation	0.001274	0.020988
DISTANCE	27.187048	102.084401
HourlyWindSpeed	0.456404	0.17341
pagerank	0.249119	1.286712

HourlyRelativeHumidity	1.741551	23.976346
HourlyWetBulbTemperature	0.77721	1.403253
HourlyDryBulbTemperature	0.564374	-4.496227
HourlyDewPointTemperature	1.281119	6.719212
HourlyWindDirection	7.108431	-10.839283
outDegree	3.004349	12.738837
inDegree	3.052094	12.94189
DELAYED_PERCENTAGE_2_HOURS_PRIOR	14.589649	34.546294

This table shows quite a few differences in the false positive averages when compared to the true positives. Last in the previous model analysis, there is over indexing on the delayed percentage 2 hours prior and significant differences across other variables. This is interesting as precision is tied to false positives and considering the difference between false positives and true negatives this likely indicates that other factors, like categorical variables, could help improve model performance.

Across all variable analysis there are some key trends to help shape our future work. First, there are few significant differences in averages between variables highlighted in green. Which leads us to the conclusion that our models perform well with those variables. The delayed percentage 2 hours prior is consistently over estimated and may need to be further refined in later work. Lastly, including categorical variables will help model performance.

Limitations, Challenges, and Future Work

Within the raw data we could refine our imputation methodology to more strongly weight recent observations rather than the median. We imputed many values for weather related variables and could have included time/location into the methodology on the theory that stations near one another in time and space are likely to have similar values. We also thought of supplementing the joined data with traffic and event data to capture times/locations that would have unusually high traffic (ex. Superbowl, World Cup, Olympics) could be taken into account. Having additional data on airports would be helpful to understand their capacity relative to the amount of traffic (ex. capacity for takeoff/landing and number of gates). WIth a longer timeframe we would have further explored aspects of the unjoined data sets to see what additional information could be incorporated.

From a feature engineering perspective, we wanted to explore more weather/geographical features, but were not able to due to time constraints. When we ran pagerank we only accounted for unique flights and not the total traffic along each flight path. A feature such as normalized flight volume in/out degree and pagerank could also be established. Narrowing down the number of categorical features would allow us to implement SMOTE more easily and potentially improve performance. A further refinement would be to closely examine the most important features to better understand what similarities and differences exist amongst our

models. We did not get the chance, but looking at the impact of specific holidays on delays would be very interesting and highly actionable from an airline perspective.

Most importantly in regards to feature engineering and model pipelining, we would like to incorporate our categorical variables into the final model. As we were able to vectorize all features together we were constrained by our cluster's ability to handle all one hot encoding across the entire dataset. As a result our final feature vectors were not always the same size and this caused our model validation to error.

On the modeling side, we could have further tuned our model's hyperparameters. We performed tuning on our Random Forest model, but were unable to due to time constraints on our Neural Networks. Given the time-based element of this problem, a simple MLP the size of which we are able to train may not be complex enough of a neural net structure to handle the complexity of our data. Time-series models, such as recurrent neural network based models (RNNs, LSTMs, etc), or CNNs/Transformers, could be explored. Further modeling of the system as both a time series problem and as a graph problem (with airports as nodes, and routes as edges). Additional graph statistics like centrality or betweenness could be added to our featureset. Lastly, we could have changed our problem formulation to a regression approach by directly predicting the number of minutes a flight was delayed. It would be interesting to see if directly predicting the delay time and then classifying would yield better results than the binary delayed or not indicator.

Citations

Ball, Michael & Barnhart, Cynthia & Dresner, Martin & Hansen, Mark & Neels, Kevin & Odoni, Amedeo & Peterson, Everett & Sherry, Lance & Trani, Antonio & Zou, Bo & Britto, Rodrigo & Fearing, Doug & Swaroop, Prem & Uman, Nitish & Vaze, Vikrant & Voltes, Augusto. (2010). *Total Delay Impact Study: A Comprehensive Assessment of the Costs and Impacts of Flight Delay in the United States*. https://rosap.ntl.bts.gov/view/dot/6234