Predicting flight delays and cancellations using weather as a feature

CS 229 Final Project: Neil Movva (nmovva), Samir Menon (samir2)

1 Introduction

Commercial airlines are a backbone of the worldwide transportation system, bringing significant socioeconomic utility by enabling cheaper and easier long distance travel. After more than half a century of mainstream adoption (especially in the US), airline operations have seen major optimizations, and today function with excellent reliability even in the face of onerous engineering challenges. Still, the modern passenger is occasionally inconvenienced by aircraft delays, disrupting an otherwise exacting system and causing significant inefficiencies at scale - in 2007, 23% of US flights were more than 15 minutes late to depart (federal definition of flight delay), levying an aggregate cost of $32.9bn on the US economy [8]. Sub-optimal weather conditions were the direct cause of ~17% of those delays, suggesting that better understanding of aircraft-unfriendly weather could improve airline scheduling and significantly reduce delays.

2 Related Work

There are many studies that attempt to model or predict flight delays using data on origin, destination, time, and other non-weather features. These studies are generally trying to showcase some new theoretical work, and use flight delay prediction as a model problem because the dataset is widely available, and the problem is widely understood [1, 2].

Another, larger set of studies by economists and statisticians attempts to understand weather's impact on the flight system as a whole - how congestion moves through the network, and how some schedules and routing topologies (like 'hub-and-spoke') are more vulnerable to cascading delays [3, 4, 5]. We seek to complement this existing research by allowing air carriers and airports to predict delay probabilities for specific flights, and then adjust scheduling or staffing as their economic models dictate.

3 Dataset and Features

The US Bureau of Transport Statistics provides data on all domestic flights, including their scheduled and actual departure and takeoff times, date, origin, destination and carrier. We combined this with Local Climatological Data from NOAA to form a joined dataset that includes the closest available weather data at the departure airport along with the standard flight data. The departure airport weather features include temperature, humidity, air pressure, and precipitation type and amount, if any.

Joining the flight and weather datasets presents a significant challenge. The weather observation times are not the same as the flight departures, and the weather is not observed on a strict cycle - stations report more or less often depending on how quickly the weather is changing. Ideally, we would then join based on the closest possible weather observation time. However, this kind of minimum-distance join is very expensive to execute across millions of rows, and was ultimately infeasible given our computational power and timeframe. Instead, we arbitrarily chose one weather observation from each hour at each station (data is guaranteed to

be at least this frequent), and then perform an equi-join on the hour of the flight departure. We lose some of the exactness of the data (for example, it is possible to have weather data from 2:54 and a flight that leaves at 2:03), but this simplification is critical in making the problem tractable.

We had several options for what feature we should try to predict. Firstly, the BTS divides delays into 5 major categories: carrier, weather, NAS, security, and late aircraft. Our first instinct was to predict the weather delay, as we have access to high resolution weather data and would like to learn what we assume to be a deterministic, objective relationship. Unfortunately, as the BTS explains, most delays caused by weather are actually categorized as NAS delays, but not all NAS delays are caused by weather. The “Weather Delay” feature actually only includes delays caused by extreme weather (~20% of all weather-based delays). So, we instead made the general delay (the difference between actual and scheduled departure time) our goal.

We chose to make this a classification problem, by binarizing the data at the 15 minute threshold (the FAA’s formal definition of a ‘delayed’ flight). This choice is partly based on the utility of prediction: for planning purposes for airlines and passengers, it’s quite useful to simply know the probability a flight will be delayed, rather than the exact number of minutes it will be delayed. The binarization choice is also partly just a simplifying assumption; our data is much easier to work with, and we can apply a wide variety of well-known classification strategies to it.

4 Methods

We begin by generating simple graphs of various features versus flight delay, and immediately notice seasonality and location-sensitivity in the data.

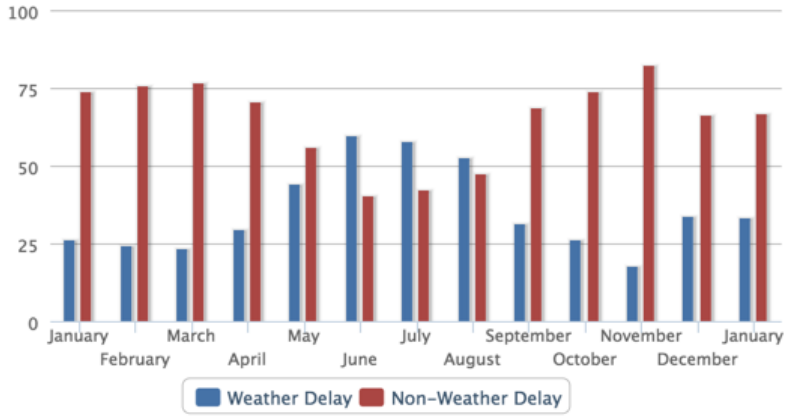

Figure 1: Seasonal variation in weather vs. non-weather delays at the Baltimore Washington International Airport. Delays are more commonly attributed to weather during Baltimore summers.

We parse continuous features, like wind speed and relative humidity, as floating point values and perform basic standardization (mean removal). For categorical features such as sky condition, weather type, and day of week, we use a one-hot encoding scheme; that is, every categorical feature that takes on n distinct values is encoded as n mutually-exclusive binary features. This is distinct from making the categorical feature numerical by assigning each feature an integer value (called ‘labelling’), because it implies no ordering of the possible values.

However, we find more nuance when considering the DayOfWeek and Month features, which we would normally assume to have linear ordering and meaningful “distance” between categories. Suppose that November was a very bad month for flight delays, because of air traffic during Thanksgiving. This doesn’t imply that flying during October or December should be worse, because that information (namely, Thanksgiving) is specific to November. On the other hand, if flying earlier in the week was less likely to produce delays, then we should rightly account for that general trend, rather than learning Monday and Tuesday individually. Therefore, we decided to include both representations for the month and day of week features.

We tried a variety of models, but found that gradient boosting (XGBoost) performed best. Looking at the decision trees produced, we find a nice semantic mapping to the actual process by which flights are delayed - airlines make a series of choices, some of which are regulated by the FAA for safety reasons, and then ultimately make a delay decision. When tuning our gradient boosting algorithm (through XGBoost), we found that increasing the max depth of our decision trees improves performance significantly without incurring too much computational cost. Intuitively, this makes sense given the high dimensionality of our data. This does allow for more overfitting, so we were careful to test it on validation splits to ensure that we weren’t just learning the idiosyncrasies of our training set.

5 Results

We created training, validation, and testing splits of our data in a 60%-30%-10% ratio. These partitions were invariant across various runs of the model; we used the validation split to learn training hyperparameters and the testing split to report our final evaluation metrics. We focused on only a few of the most important hyperparameters for each model to compare their performances. For XGBoost and random forest, we considered the number of trees and maximum depth of each tree, both of which control model complexity and ability to model nonlinearities in the data. We also tuned the L2-regularization strength for XGBoost, and, similarly, L1-regularization strength for Lasso. Linear regression was trained with a standard gradient descent implementation, with no nontrivial hyperparameters. We used default scikit-learn hyperparameters for our SVM classifier (notably with a linear kernel), as we saw no significant fluctuations in performance upon qualitative evaluation of other parameter choices.

To evaluate our models, we used our 60% training split and reported metrics on our 30% validation split with a classification threshold at 0.5 (where 0 = on-time, 1 = delayed). We initially considered using the Area Under the Receiver Operating Characteristic curve (AUROC). However, since our dataset was significantly imbalanced (20% delayed flights, 80% on-time flights), AUROC is sometimes misleading: we often saw high AUROCs (> 0.8) with very low recall (< 10%). Thus, to learn hyperparameters, we considered plots of recall and precision and searched for the optimum between the two metrics. Figure 2 illustrates this approach for learning max tree depth of XGBoost: we found that a max tree depth of 8 optimized both precision and recall, while lower values traded off with predictive power and higher values overfit the model. Similar evaluations for other parameters led to our choice of 100 trees for XGBoost and Random Forest, 0.1 for XGBoost L2-reg strength, and 1.0 for Lasso L1-reg strength.

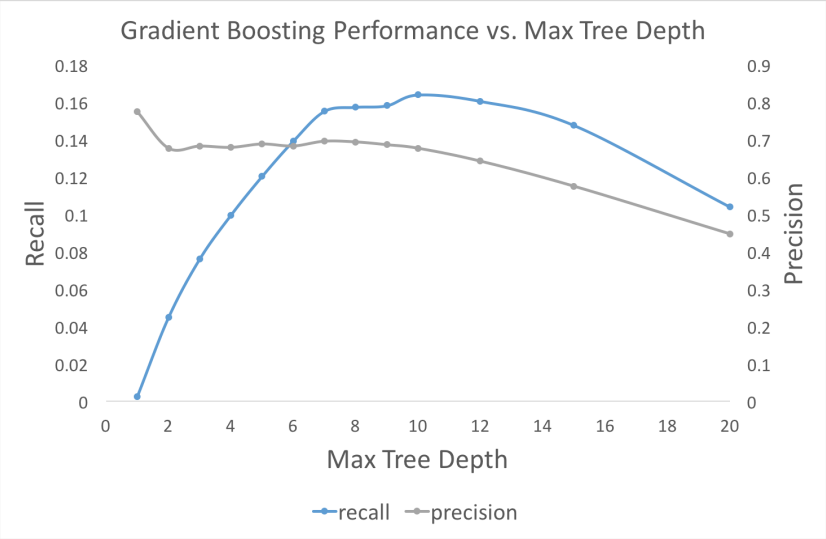

Figure 2: XGBoost performance vs. max tree depth. We tuned hyperparameters by training on our 60% split and testing on the 30% validation split, and evaluating resultant precision vs. parameter and recall vs. parameter plots. Here, we see that max tree depth = 8 optimizes XGBoost performance.

As final test metrics, we report AUROC of the five models (Fig. 3a). To overcome dataset imbalance, we randomly removed enough on-time flights so that the amount of on-time flight data matched the amount of delayed flight data. In total, we used n = 98,474 datapoints for testing. We found that XGBoost clearly worked better than the other models; Fig. 4b presents a confusion matrix for XGBoost's predictions on the full 10% split (without imbalance correction).

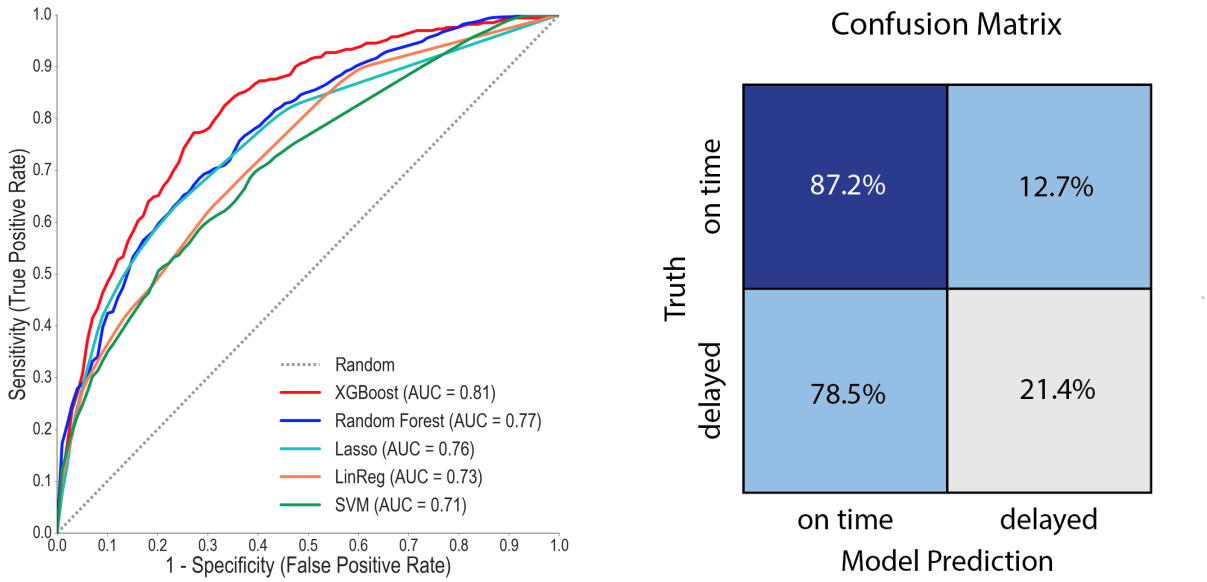

Figure 3: Model evaluation reveals that XGBoost performs best. (a) ROC curves for the five models generated by testing on an unseen, balanced split of the dataset. (b) Confusion matrix for XGBoost.

The XGBoost model's recall is 15.5%, and precision is 70%. The recall may seem rather poor, but note that the majority of flight delays are caused by non-weather events - e.g maintenance, crew, etc. Our model cannot be expected to account for most of these, as it lacks the related feature data. Based on data from BTS, we know that ~17% of flight delays are caused by weather, so our 15.5% recall is relatively performant in context. Some of the 83% of delays that we are currently not predicting could be weather-related, making our recall claims less sound, but without more detailed BTS data, there is no way to verify this.

To gain insight from our model's predictions (Fig. 4), we generated feature importances defined as the gain in the accuracy of the model when a certain feature was included vs. not included in training (according to scikit-learn's built-in feature importance implementation).

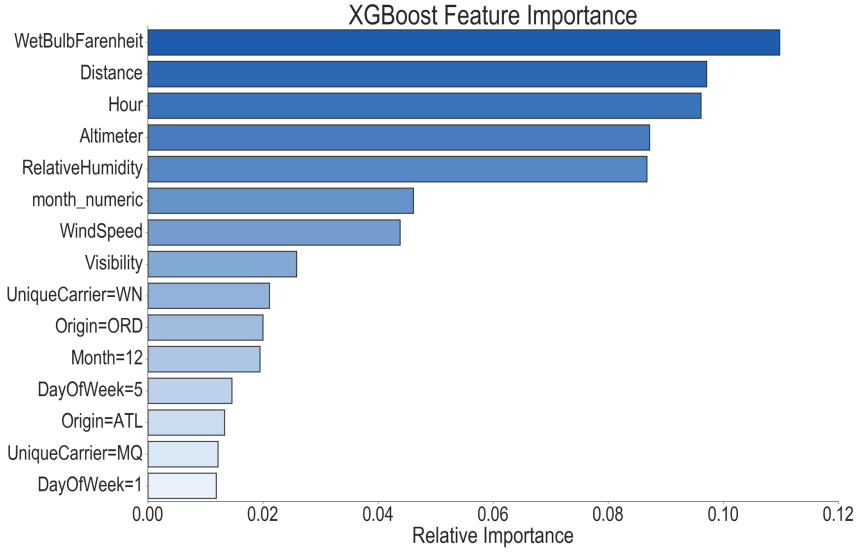

Figure 4: XGBoost feature importance. Generated by the default scikit-learn implementation.

The feature importance graph confirms our earlier choices on encoding categorical data. We see that the numeric month feature is important, but so is the binary "Month=12" (December) feature. We can see that the one-hot encoding of carrier data was successful as well; the "UniqueCarrier=WN" (Southwest) feature is actually quite important. This fits with existing research that shows that Southwest Airlines appears to operate flights in a substantially different way than other carriers [9].

6 Conclusion/Future Work

We used entirely our own computers to preprocess, train, and test. With more time, we'd set up a cluster and run larger models on data from 1978-2015. Our hardware limited our options when joining the weather and flight data, and made higher resolution analysis infeasible. With more computing power, we'd especially try to experiment with some exponential decay models on weather prior to the flight's departure, as we had originally intended.

More generally, we've shown that flight delay prediction is tractable, that local weather data at the origin airport is indeed predictive of delays, and that decision trees and random forests are probably the best way to approach this problem.

References

[1] Khanmohammadi, Sina, Salih Tutun, and Yunus Kucuk. "A New Multilevel Input Layer Artificial Neural Network for Predicting Flight Delays at JFK Airport." *Procedia Computer Science* 95 (2016): 237-244.
[2] Hensman, James, Nicolo Fusi, and Neil D. Lawrence. "Gaussian processes for big data." *CoRR, arXiv:1309.6835* (2013).
[3] Schaefer, Lisa, and David Millner. "Flight delay propagation analysis with the detailed policy assessment tool." *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*. Vol. 2. IEEE, 2001.
[4] Hansen, Mark, and Chieh Hsiao. "Going south?: Econometric analysis of US airline flight delays from 2000 to 2004." *Transportation Research Record: Journal of the Transportation Research Board* 1915 (2005): 85-94.
[5] Gilbo, Eugene P. "Airport capacity: Representation, estimation, optimization." *IEEE Transactions on Control Systems Technology* 1.3 (1993): 144-154.
[6] Belcastro, Loris, et al. "Using Scalable Data Mining for Predicting Flight Delays." *ACM Transactions on Intelligent Systems and Technology (TIST)* 8.1 (2016): 5.
[7] Bandyopadhyay, Raj, and Guerrero, Rafael. “Predicting airline delays.” *CS229 Final Projects* (2012).
[8] Guy, Ann Brody. “Flight delays cost $32.9 billion.” http://news.berkeley.edu/2010/10/18/flight_delays/
[9] Tierney, Sean, and Michael Kuby. "Airline and airport choice by passengers in multi-airport regions: The effect of Southwest airlines∗." *The Professional Geographer* 60.1 (2008): 15-32.