Predicting Airplane Delays

Bryce Hepner, Caelan Osman, Carter Landon

Fall 2021

1 Introduction

While airplane travel is certainly more convenient when compared to alternative modes of transportation, there is still much to be desired in terms of convenience, and the prevalence of delays is among those shortcomings. If an accurate means of predicting flight delays were found, then additional preemptive measure could be taken by airline companies to prevent delays, or travelers could be notified of the probability of delay, so as to potentially reduce frustration with waiting and be prepared to make necessary adjustments for connecting flights.

In this project, we examine data for American Airlines flights during the month of July, 2016, analyze classification methods to see which will be the most useful for predicting flight arrival lateness, and use test it with flights from the month of May, 2017. We will focus on answering three questions in this research: which classification method is most effective for predicting whether or not a flight will arrive late, which classification method is most effective for predicting how late a flight will arrive, and which features are most useful for predicting arrival lateness?

All the code we have written and developed for this project is on a public Bitbucket repository found here.

2 Techniques Used by Others

Javier Herbas describes his attempts to answer this question of flight delays in his article "Using Machine Learning to Predict Flight Delays". He first describes how he cleaned his data, by dropping unneeded columns and engineering a binary column based on whether the plane arrived late to its destination or not. He used the following machine learning methods with a 75/25 train-test split: building a regular tree as baseline, created bagged

trees, ran a random forest with no class weighting (ran a feature importance plot as a QC tool), random forest with bootstrat class weighting, ran a AdaBoost with and without the departure delay, ran Gradient Boosted Trees with and without the DEP_DELAY, ran XGBoost with and without the DEP_DELAY. The best score that he obtained with these methods was 86% accuracy using XGBOOST with DEP_DELAY, which is surprisingly low considering knowing the flight left late should correlate very highly to the flight arriving late. Without DEP_DELAY, the highest accuracy was 69%, also using XGBOOST. He also used deep neural net methods, which yielded slightly better results.

Another study (Zoutendijk and Mitici) took this problem a step further by not only analyzing flight delays, but also analyzing a method to reduce delays and make airports more efficient by assigning planes to gates more effectively. They used Mixture Density Networks and Random Forests, and were able to come up with a probabilistic gate-assignment model that "reduces the number of conflicted aircraft by up to 74% when compared to a deterministic flight-to-gate assignment model" (Zoutendijk, page 1).

3 Description of Data

Our data (downloaded from https://www.kaggle.com/mrferozi/flight-delays) contains plenty of information (including departure and arrival delays) for a large number of flights (4821). Unfortunately, many of the 35 columns are either linearly dependent or engineered data. For example, the data includes the columns 'Year', 'Month', and 'Day', as well as 'Flight_Date', which is essentially a reformatted combination of the first three.

This data set is legitimate because it is simply flight information directly from American Airlines, with some engineered data columns that we can easily drop. However, it does have a potential weakness in that it does not include the destination airport, which could have an impact on lateness. Another potential weakness is that the data is only concerning American Airlines, as different airlines could have greatly varying approaches to reducing delays, so a set of data that included flights for many airlines could be more generally useful.

Based on how good Javier Herbas' accuracy was (66-86%), and considering we are using a slightly more homogeneous set of data, we expect to get around 75% accuracy for our binary classification (whether the flight is late or not) and 65% for how late the flight is.

4 Data Cleaning and Feature Engineering

In terms of data cleaning, the only cleaning that was needed was to remove useless columns of data. We added code to remove any canceled flights, which made no difference for this dataset because there were no canceled flights, but this helps the code be generalized to other datasets. Furthermore, canceled flights is so qualitatively different from a delayed arrival that it would not classify well with the rest of our data. The rest of the data was complete, and nothing was obviously erroneous.

There is a good amount of feature engineering that is needed. To explore the best possible classifiers we added 3 options to our feature engineering function:

- Binary
- Binned (with an option to SMOTE the data)
- Non binned

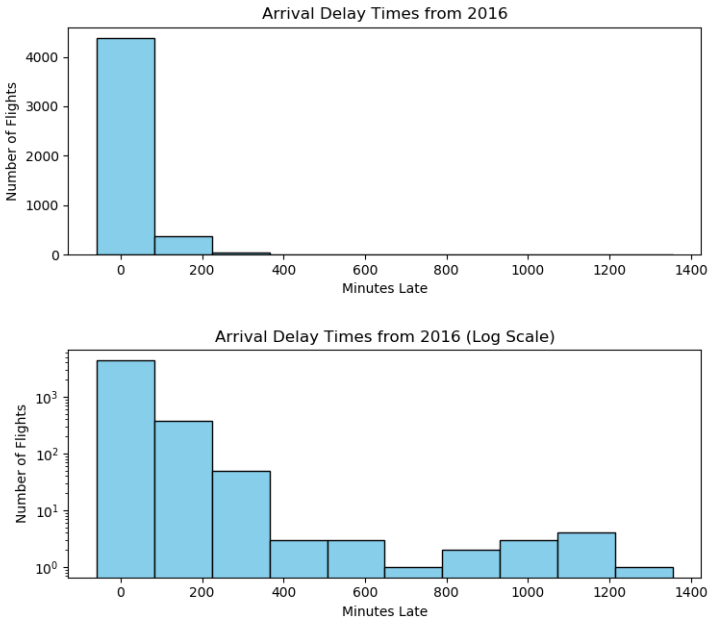

Figure 1: Arrival Delay Times From the Month of July, 2016

Originally we only had the first two options available. This is because most classifiers work best with either binary or discrete labels instead of

continuous, so we developed the first two types of labels: the binary one that just says whether the flight arrived on time or not, and a multi-label one that groups arrival delay into bins; on time, less than 15 minutes late, 15-30 minutes late, we continued to partition by 15 minutes until 1 hour, then we partitioned by 60 minutes until 5 hours, and from there partitioned by 100 minutes.

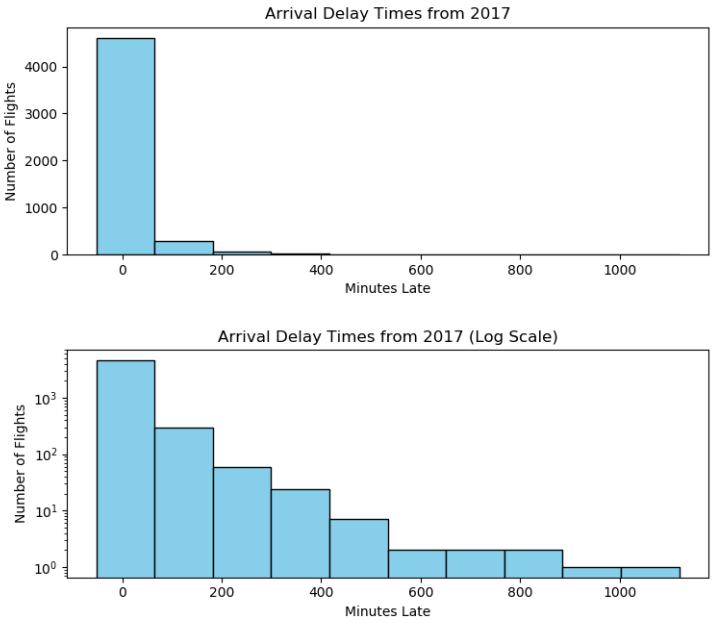

Figure 2: Arrival Delay Times From the Month of July, 2017

These bins may seem arbitrary, but given that the great majority of flights are delayed by only a small amount instead of a large amount, we felt like we got more information by making smaller bins for the smaller time delays, and then larger bins for the less common large time delays so as to not have a large number of bins with no entries or only one entry. Though this data is only representative of a single airline during a one month span that took place five years ago, we don't expect much different arrival delay distributions from other samples, so our reasoning for binning the way we did would still hold for more general data (if other airlines were significantly worse they would likely go out of business, and if they were significantly better they might greatly edge out the competition).

After running some initial tests we also decided to add the option of

using the non-binned data. This is because we decided to test classifiers such as *OLS* and *Logistic Regression* on the continuous data to see if we could get a better continuous classifier. These lead to some good discoveries on the p and R^2 values.

As we will discuss later, we also left the arrival delay time unaltered for the Ordinary Least Squares regression classifier.

Originally, we tried one-hot-encoding 'Origin_Airport', 'Tai_lNum', 'Origin_City_Name', and 'Origin_State', but mostly due to 'Tai_lNum' this led to having about 616 columns of data. The 'Tai_lNum' feature uniquely identifies each airplane (and unfortunately has the underscore in the wrong place), and while that could have a great effect we opted to simplify our model by removing it. We could have gotten down to around 50 total columns by just getting rid of 'Tai_lNum', but we instead went further by keeping only 'Origin_Airport', because the origin city and state are inherently connected to the airport.

Finally we found that 'Scheduled_Departure' was a cyclical value, which has the disconnect across midnight of going from 2359 to 0. Meaning these data points appear to far apart even though in reality they are very close. We fixed this by making time decrease by 1200 after noon.

Figure 1 and fig. 2 show histograms for the arrival delay for 2016 and 2017 (with linear scaling on the left and logarithmic on the right). Because so few flights arrived in the very late range, we decided to implement a SMOTE function to augment the data for the very late arrivals. Fortunately, we only need to use our SMOTE function for non-binary separation, when the flights are binned by minutes late.

When attempting to add synthetic data with d features, the SMOTE algorithm requires a compact set in d dimensional space to take random samples from which means we need at least two data points to accurately get a bound in each dimension. With the way we binned our data, getting a bound was not always possible. This is because in several of the bins there was only one flight and a few more there were zero. Unfortunately, this meant data augmentation in those bins was futile.

When training a binary classifier it was not necessary to augment the data. This is because the data was split roughly 50-50 between being on time and being late. The results we see in section 5 are due to this.

Additionally, we elected to not augment the data when we were using a continuous classifier. Because, as we will discuss later, augmenting our data via SMOTE led to harmful overfitting when binning the arrival delay, we reasoned it would be similarly unhelpful with nonbinned arrival delay.

5 Data Visualization and Basic Analysis:

Because we had several different ways of formatting our data (binary, binned, and non-binned; augmented or not), and several different tools available to us, we decided to run whatever we could on our datasets and see which would offer the highest accuracy and recall. We ran these classifiers, with the following results:

Classifier	Binary	SMOTE	Accuracy	Recall
KNN	Y	N	0.58742225293711	0.37837837837837
Logistic	Y	N	0.64132688320663	0.618776671408250
RFC	Y	N	0.64478230822391	0.664295874822190
Gaussian	Y	N	0.58465791292328	0.86628733997155
KNN	N	Y	0.447823082239115	0.077174425041288
RFC	N	Y	0.498272287491361	0.078394022888901
Gaussian	N	Y	0.04146510020732	0.0500601389945267
KNN	N	N	0.480304077401520	0.09155394141942
RFC	N	N	0.51416724257083	0.083333333333333
Gaussian	N	N	0.062888735314443	0.0592589847585993

The data shows the Random Forest Classifier having the highest accuracy and recall in both the binary and non binary data. The SMOTE-augmented data caused issues with both the accuracy and the recall. Though we were not able to achieve the hypothesized 75% accuracy, we were able to approximately match Herbas' 65% without including the departure delay among our features.

An additional thing to note is that we also ran a model with the knowledge of whether a plane had a delayed departure or not. Flight paths are carefully mapped out to make the flight optimal. There is a balance between, time to destination, the optimal altitude to fly at so that drag will be reduced (increasing MPG) and the engines will still perform optimally, as well as operating costs of the crew. With this it is intuitive to think that a plane that takes off late is more likely to land late unless some of the aforementioned factors are neglected or there is an effective of enough tailwind to make up the time. This feels a bit like cheating, but if we included the departure delay in our training data then our accuracy went up to 82%. This analysis was performed with the included 'Dep_Delay' feature and no modifications. Perhaps another thing to explore is the accuracy and recall of the model if we performed a binary or binning sorting on the departure delay feature before we trained and tested the model.

6 Learning Algorithms and In-depth Analysis:

1. K Nearest Neighbors

   The K Nearest Neighbors classifier was our second best classifier on both accuracy and recall when trying to train on the binned data. The most interesting thing we discovered here is that the optimal algorithm the KNN classifier was built with was a ball tree. A cursory reading of ball trees explain that they partition the known data points into a nested set of hyperspheres (i.e. balls). Then the KNN classifier uses these hyperspheres to judge "distance". Geometrically this is interesting to think about that our data is distributed in such a way that certain bins of data "form" hyperspheres. The quotations here were used because obviously the data isn't close enough to this structure to allow us to have any reasonable sort of accuracy.

   The lack of accuracy and poor recall scores from KNN tell us that using some sort of distance metric in $\mathbb{R}^d$ to classify our data is a poor choice.

2. OLS

   We used Ordinary Least Squares regression both to see if we could fit our data well, and to see if this would shed light on which features were most useful in predicting arrival delays. We ran OLS on six versions of our dataset: binned delay, no SMOTE, with airports; binned delay, no SMOTE, without airports; binned delay, with SMOTE, with airports; binned delay, with SMOTE, without airports; non-binned delay, no SMOTE, with airports; and non-binned delay, no SMOTE, without airports.

   The R-squared values were low for all of these, with the highest being 0.208 for binned delay, with SMOTE, with airports. The AIC and BIC were very high for all of them as well, never lower than 38,000. The condition number also exceeded 10,000 for each variation of the data, implying that our data is not well suited for OLS regression.

   Part of the reason for this is because the data is rather tabular. Even though we have numeric values for every column, the airport columns are one-hot-encoded; the day of the week column is just the integers from one through seven, even though assigning each day of the week a number depends on arbitrary choice–if day 1 is Monday, why should we represent Sunday as "greater than" Monday? Again this turns out

to be a cyclical value so perhaps performing a similar day reduction like we did with the 24 hour time would help.

We found that augmenting our data via SMOTE was very unhelpful. Because there were so few very late flights, there were only a select few airports that had an increase in very late flights. For example, without augmented data the coefficient for the Denver Airport was 179.5, with a p-value of 0.067. It seemed from this that flights from Denver could be delayed by quite a bit. However, when we augmented the data, the coefficient for Denver increased to 545.5, with a p-value of 0.0. If there were true justification for being so confident that flights out of Denver would be so late, that airport would be infamously bad to fly out of. However, that is not the case, and so we conclude that augmenting our data provides no benefit for our finding which feature best predicts flight lateness.

We also found that the p-value for almost every airport column was very high, so we ran half of our OLS regressions on datasets omitting the airports. Here is a table highlighting the differences between running with and without airports for our non-binned data that was not augmented:

Feature	Coef	p	Coef (w/o airpt)	p (w/o airpt)
const	-37.1034	0.022	-18.1712	0.001
DayOfWeek	1.0680	0.083	1.0028	0.104
Sched Dep	0.0195	0.000	0.0190	0.000
Sched Arr	0.0076	0.001	0.0078	0.001
Distance	-0.0023	0.160	-0.0023	0.090

Because our delay is measured in minutes, all we can really tell both with the airports and without the airports is that we are fairly confident that these features have very little to do with how late a flight arrives. Thus we conclude that Ordinary Least Squares does not help us answer the question of which feature best predicts how late a flight will be.

3. Logistic Regression

   Because we gained so little insight from OLS, we looked to Logistic Regression to see if that would be a better tool for fitting our data. Again, we began with including the airport columns, and were a bit surprised to see that the p-values were not nearly as bad as they were

with the OLS. They were, however, large enough that we wanted to see what difference removing them would make. The results are as follows:

Feat	Coef	p	Coef (w/o airpt)	p (w/o airpt)
const	-3.035	0.000	-1.3216	0.000
DayOfWeek	-0.0039	0.823	-0.0054	0.759
Sched Dep	0.0004	0.000	0.0005	0.000
Sched Arr	0.0004	0.000	0.0004	0.000
Distance	-0.00001	0.810	-0.00002	0.681

Again, we see that even for very low p-values, we aren't seeing any strong relationship between any of these features and whether or not a flight arrives late. Thus, we are unable to give any meaningful answer to the question of which feature best predicts whether a flight will be late, and how late it will be. Perhaps a much bigger data set, with more airports as well as more flights per airport would help us avoid these problems of only a few airports having greatly delayed flights, and this would help us see what is truly most important for predicting flight lateness.

4. Random Forest Classifier

   Because the Random Forest Classifier was our best performing for the binary classifiers (with 64.4% accuracy and 66.4% recall), we wanted to see how well it would perform being trained on the entirety of 2016's data and tested on 2017's data. The immediate problem came in one-hot-encoding the airports. More airports appeared in 2017's data than in 2016's, so we needed to remove the rows containing airports not used in 2016.

   We ran the Random Forest Classifier using the settings that yielded the best RFC in the binary case ('bootstrap': False, 'criterion': 'entropy', 'max_depth': 5, 'n_estimators': 100, 'n_jobs': -1). We got a resulting accuracy of 58.2%, which seems better than guessing. However, we found that that exact proportion of all flights were on time, and that the prediction that the forest gave was that all flights were on time. This means that the recall is 0, because no positives were predicted and thus no true positives were predicted. Unfortunately, with this we must say that the Random Forest Classifier is not helpful in taking past flight data and predicting whether or not a future flight will be delayed.

5. Gaussian Naive Bayes

   The Gaussian Naive Bayes approach was one of the worst tested on the binary classification, and was the worst with using the binned data to test how late the plane would be. The largest potential problem from this was the assumption that they are all independent. If the flight before was late, then the next one has to be late. There was 58.5% accuracy on the binary and 6.2% on the binned data. This approach worked especially poorly on the binned data due to its inability to combine the features well enough. It's possible that an airport would do well in the mornings and poor in the evenings and this approach can not pick up on that, only looking at the combination of airport and time in general, not specific to each airport.

7 Ethical Implications:

Fortunately, our data was very impersonal, so there are no personal private concerns. However, if similar analysis were run on a dataset that included flights from many airlines, data might predict that some airlines had higher rates of late arrival. This prediction–whether true or false–could lead to people avoiding that airline in favor of others. The ensuing decline in income for that airline may lead to poorer management and thus a subsequent increase in probability of late arrival.

To prevent this self-fulfilling feedback loop, models can be run for each individual airline, for the use of each individual airline. This would do a lot to reduce the possibility of any airline being classified as frequently late in comparison to others, and thus the feedback loop is cut off.

8 References

Javier Herbas, "Using Machine Learning to Predict Flight Delays", https://medium.com/analytics-vidhya/using-machine-learning-to-predict-flight-delays-e8a50b0bb64c

Micha Zoutendijk and Mihaela Mitici, "Probabilistic Flight Delay Predictions Using Machine Learning and Applications to the Flight-to-Gate Assignment Problem" https://www.mdpi.com/2226-4310/8/6/152

Dataset from Kaggle: https://www.kaggle.com/mrferozi/flight-delays